



Håndbok i SAS

Del 2: Oppslag



Forord

Dette heftet er en norsk dokumentasjon av sentrale deler av SAS. Målet er å gi SAS-brukerne i Statistisk sentralbyrå en hendig oppslagsbok på norsk med eksempler fra mange av våre arbeidsområder. Målgruppen er de som allerede har en viss kjennskap til SAS. Nybegynnere bør lese Håndbok i SAS Del 1: Innføring (Interne Dokumenter 96/16) før de begynner å bruke dette heftet.

Jeg har lagt vekt på å bruke eksempler fra program laget i SSB. Eksempelene spenner fra de helt enkle til de mer avanserte. Eksempelene er i stor grad komplette SAS program. Dette krever noe mer plass, men gjør det til gjengjeld lettere å bruke eksempelene som utgangspunkt for egne program.

SAS er så omfattende at det umulig å ta for seg alt i et hefte som dette. Jeg har prøvd å velge ut deler som kan være aktuelle for oss. Vi har nylig gått over til versjon 8.1 av SAS i Statistisk sentralbyrå. I versjon 8.1 er det litt mer forskjell på brukergrensesnittet mellom unix og Windows NT. Dette heftet er oppdatert på de viktigste endringene i de mest brukte delene av SAS-systemet.

Heftet er delt i to: Den første delen tar for seg noen viktige anvendelsesområder for SAS: kobling, omkoding av data, konvertering av data m.m. Den andre delen er en beskrivelse av programmeringsspråket og omgivelsene som møter en SAS-bruker.

Dette er en utvidet og oppdatert utgave av håndboken som tidligere er gitt ut som Interne Dokumenter 99/7.

Oslo, 2. november 2000

Kristian Lønø

Innhold

1.	Innledning.....	5
2.	Anvendelser.....	6
2.1	Konvertering av data	6
2.1.1	SAS datasett fra Sun Unix til Digital Unix og omvendt	6
2.1.2	SAS datasett fra PC til Unix	6
2.1.3	SAS datasett fra Unix til PC	7
2.1.4	Skilletegnseparerte filer	8
2.1.5	Excel regneark	10
2.1.6	SPSS-filer	11
2.2	Konvertering av variable	12
2.3	Kobling av SAS datasett	13
2.3.1	Merge	13
2.3.2	Update	15
2.3.3	Join	19
2.3.4	Koble et datasett med 1 observasjon til hver observasjon på et annet datasett	20
2.4	Omkoding ved hjelp av kataloger.....	21
3.	Referansedel	23
3.1	SAS-navn	23
3.2	SAS kommandoer	23
3.2.1	Viktige kommandoer under programmering.....	24
3.2.2	Kommandoer for kopiering, klipping og liming	24
3.2.3	Kommandoer før søk/erstatt	25
3.2.4	Kommandoer som kaller opp SAS vinduer	25
3.2.5	Kommandoer for navigering.....	26
3.2.6	Andre aktuelle kommandoer.....	26
3.3	SAS editor.....	27
3.3.1	Enhanced editor	27
3.3.2	Program editor	27
3.4	SAS-uttrykk.....	28
3.5	SAS dataformater	30
3.5.1	Innlesingsformater	30
3.5.2	Utskriftsformater	31
3.6	Automatiske hjelpevariable.....	33
3.7	SAS system options.....	34
3.8	SAS datasett options	39
3.9	DATA-steget.....	41
3.9.1	ABORT	41
3.9.2	ARRAY	42
3.9.3	assignment	44
3.9.4	ATTRIB	45
3.9.5	BY	45
3.9.6	CALL.....	48
3.9.7	CARDS.....	49
3.9.8	CARDS4.....	49
3.9.9	CONTINUE	49
3.9.10	DATA	50
3.9.11	DATALINES	50
3.9.12	DATALINES4	50
3.9.13	DELETE	50
3.9.14	DISPLAY	50
3.9.15	DO.....	51
3.9.16	DROP	52
3.9.17	END	52
3.9.18	ENDSAS	52
3.9.19	ERROR	52
3.9.20	FILE	53
3.9.21	FORMAT	56

3.9.22	GO TO	56
3.9.23	IF, subsetting	56
3.9.24	IF, THEN, ELSE	57
3.9.25	%INCLUDE	58
3.9.26	INFILE	58
3.9.27	INFORMAT	61
3.9.28	INPUT	62
3.9.29	KEEP	65
3.9.30	kommentar	65
3.9.31	LABEL	67
3.9.32	label	67
3.9.33	LEAVE	68
3.9.34	LENGTH	68
3.9.35	LINK	68
3.9.36	LIST	69
3.9.37	LOSTCARD	69
3.9.38	MERGE	69
3.9.39	MISSING	70
3.9.40	null	70
3.9.41	OUTPUT	71
3.9.42	PUT	71
3.9.43	RENAME	73
3.9.44	RETAIN	73
3.9.45	RETURN	74
3.9.46	RUN	74
3.9.47	SELECT	75
3.9.48	SET	76
3.9.49	SKIP	77
3.9.50	STOP	78
3.9.51	sum	78
3.9.52	UPDATE	78
3.9.53	WHERE	79
3.9.54	WINDOW	80
3.9.55	Andre SAS-setninger	83
3.10	Globale SAS-setninger	84
3.10.1	DM	84
3.10.2	ENDRSUBMIT	84
3.10.3	FILENAME	84
3.10.4	FOOTNOTE	85
3.10.5	%LET	86
3.10.6	LIBNAME	86
3.10.7	%LIST	87
3.10.8	ODS	87
3.10.9	OPTIONS	92
3.10.10	PAGE	93
3.10.11	%PUT	93
3.10.12	RGET	93
3.10.13	RSUBMIT	93
3.10.14	SYSTASK	94
3.10.15	TITLE	95
3.10.16	WAITFOR	95
3.10.17	X	95
3.11	PROC-steget	96
3.11.1	PROC APPEND	96
3.11.2	PROC COMPARE	97
3.11.3	PROC CONTENTS	103
3.11.4	PROC COPY	106
3.11.5	PROC DATASETS	107
3.11.6	PROC DOWNLOAD	111
3.11.7	PROC FORMAT	113

3.11.8	PROC FREQ	121
3.11.9	PROC FSLIST	124
3.11.10	PROC FSVIEW	124
3.11.11	PROC MEANS	126
3.11.12	PROC OPTIONS	134
3.11.13	PROC PRINT	135
3.11.14	PROC PRINTTO	137
3.11.15	PROC RANK	138
3.11.16	PROC REPORT	140
3.11.17	PROC SORT	156
3.11.18	PROC SQL	158
3.11.19	PROC TABULATE	165
3.11.20	PROC TRANSPOSE	180
3.11.21	PROC UNIVARIATE	185
3.11.22	PROC UPLOAD	191
3.12	SAS funksjoner	193
3.12.1	Karakterfunksjoner	193
3.12.2	Statistiske funksjoner	199
3.12.3	Dato- og tidsfunksjoner	200
3.12.4	Aritmetiske funksjoner	202
3.12.5	Trunkeringsfunksjoner	203
3.12.6	Spesialfunksjoner	203
3.12.7	Tilfeldige tallfunksjoner	207
3.12.8	Sannsynlighetsfunksjoner	208
3.12.9	Kvantile funksjoner	209
3.12.10	Matematiske funksjoner	210
3.12.11	Trigonometriske og hyperboliske funksjoner	210
3.12.12	Finansielle funksjoner	211
3.12.13	Logiske binærtall-funksjoner	211
4.	Vedlegg	212
4.1	Bruke SAS som klient/tjener-verktøy	212
4.2	Organisering av program og data	213
4.3	Testing og kvalitetssikring	213
4.4	Kontroll og konsistenssjekk av variabelnavn og tekster	216
4.5	Feilsituasjoner	218
4.5.1	Manglende semikolon	218
4.5.2	Fnutter i utakt	218
4.5.3	Stoppe et program	219
4.5.4	Opprinnelig SAS datasett har fått én observasjon og feil innhold	219
4.6	Definere stiler for html-filer	219
4.7	Nøyaktighet ved lagring av tall	222
4.8	Definisjoner av statistiske nøkkelord	223
4.9	Felles SAS-formater tilgjengelig for alle	224
4.10	Sorteringsrekkefølge for forskjellige tegnsett	226
5.	Stikkordregister	228
De sist utgitte publikasjonene i serien Statistisk sentralbyrås håndbøker		243

1. Innledning

SAS-program består av et eller flere DATA-steg og/eller PROC-steg. DATA-steget brukes til programmering i tradisjonell forstand som f.eks. å lage SAS datasett, omkodning, kobling og kontroller. I PROC-steget velger vi å bruke ferdig definerte prosedyrer for å utføre spesielle oppgaver som f.eks. å lage frekvenstabeller, sortere SAS datasett eller liste ut SAS datasett.

DATA-steget bygger vi opp med SAS-setninger (engelsk: statements). De fleste setninger begynner med et SAS nøkkelord (SAS keyword) som forteller hva slags setning det er, f.eks. INPUT, IF, DO, eller SELECT. Etter SAS nøkkelordet følger som oftest en eller flere parametre (arguments) hvor vi spesifiserer nærmere hva vi ønsker at SAS-setningen skal gjøre. Vi avslutter SAS-setninger med et semikolon. SAS-setningene i DATA-steget kalles opp etter SAS nøkkelordet i setningen, f.eks. INPUT-setningen.

PROC-steget består også av SAS-setninger. Hver prosedyre har egne setninger med sin egen syntaks. Disse er beskrevet i omtalen av den enkelte prosedyre. Syntaksen i prosedyrene ligner på den i DATA-steget: Hver SAS-setning starter med et SAS nøkkelord, vi kan angi parametre og vi må avslutte setningen med et semikolon. SAS-setningene i PROC-steget er kalt opp etter prosedyren den hører til, f.eks. kalles SAS-setninger i PROC PRINT for PROC PRINT setninger.

I mange SAS-setninger kan vi bruke SAS-uttrykk (SAS expressions). Disse er kort beskrevet på side 28.

Når vi beskriver forskjellige ord og uttrykk, bruker vi disse konvensjonene:

SAS nøkkelord er skrevet med store, fete bokstaver
 Parametre som *må* være med er skrevet med vanlig skrift
 Frivillige parametre er skrevet med vanlig skrift i <>
 Når én av flere parametre kan brukes, skilles disse med |
 Ved gjentatte parametre brukes ...
 SAS operatører, funksjoner etc. er skrevet med store bokstaver

INPUT
 variabel-1
 <variabel-n>
PRINT | FILE
 variabel-1 ... variabel-n
AND

Programeksempelene er vist slik:

```
DATA tilbfall ;
INFILE '$KRIMINAL/tilbfall/data/tilbf87';
INPUT @01 fnr $CHAR11.
      @13 dd_1 YYMMDD6.
      @19 dd_2 YYMMDD6.
      @25 dd_3 YYMMDD6.
      @31 dd_4 YYMMDD6.
      ;
RUN;
```

Utskrifter fra programeksempelene vises slik:

```
Helseundersøkelsen 1995. Kronisk syke = 1
```

KROSYK	Frequency	Percent	Cumulative Frequency	Cumulative Percent
.	1952.437	40.8	1952.437	40.8
1	2836.042	59.2	4788.479	100.0

2. Anvendelser

Vi tar her for oss noen anvendelser av SAS. Vi ser på hvordan vi:

- kopierer data fra forskjellige plattformer
- henter data fra forskjellige filformater
- kobler SAS datasett vha. og MERGE, UPDATE og SET-setningene i DATA-steget og vha. PROC SQL
- koder om data vha. omkodingskataloger lagret som SAS-formater

2.1 Konvertering av data

2.1.1 SAS datasett fra Sun Unix til Digital Unix og omvendt

Lagringsformatet for SAS-filer på Digital Unix er forskjellig fra Sun Unix sitt. Når vi vanligvis kjører på en Unix-type, men trenger et SAS datasett fra den andre, bør SAS datasettet konverteres før vi begynner å bruke det. Vi kan bruke PROC DOWNLOAD for å overføre et SAS datasett fra et format til et annet.

Vi logger oss først på den maskinen vi ønsker å ha dataene på (digital-maskinen *ursus* i eksempelet under) og starter SAS på vanlig måte. Så må vi fortelle SAS hvilken aksessmetode som skal brukes. Det gjør vi med SAS sytem option COMAMID . Deretter bruker vi FILENAME-setningen til å angi hvor unix-scriptet som definerer forbindelsen befinner seg. Så kan vi knytte forbindelsen (signon) til den maskinen dataene er på (f.eks sun-maskinen *lupus*). Når forbindelsen er opprettet overfører vi dataene våre med prosedyren PROC DOWNLOAD.

Her er et eksempel på et program som overfører en fil fra Sun-maskinen *lupus* til Digital:

```
FILENAME RLINK '/local/sas81/misc/connect/tcpunix.scr';
OPTIONS COMAMID=tcp ;

DM 'SIGNON lupus';
RUN;

RSUBMIT;
LIBNAME forbruk '$FORBRUK/data/g1997';
PROC DOWNLOAD DATA=forbruk.kriterie OUT=kriterie ;
RUN;
ENDRSUBMIT;

PROC CONTENTS DATA=kriterie;
TITLE 'Innhold nedlastet datasett';
RUN;
```

I eksempelet over legges den overførte filen på et temporært SAS datasett på Digital-maskinen *ursus*. Vi kan selvfølgelig legge den direkte ut på et permanent SAS datasett isteden.

For å få data den andre veien, logger vi oss på Sun-maskinen og starter SAS på vanlig måte. Deretter oppretter vi forbindelsen til Digital-maskinen og henter dataene på samme måte som i eksempelet over.

2.1.2 SAS datasett fra PC til Unix

Vi kan bruke PROC UPLOAD fra Windows SAS til å sende SAS datasett til Unix arbeidsstasjon. Først logger vi oss på ønsket arbeidsstasjon (her: *ursus*) med DM-setningen. Deretter kommer de setninger som skal sendes til Unix, som vi legger mellom RSUBMIT- og ENDRSUBMIT-setningene:

```
DM 'SIGNON ursus';
RUN;
LIBNAME levefunk 'x:\levefunk\data';
RSUBMIT;
LIBNAME levefunk '$SURVEY/levefunk/data';
PROC UPLOAD DATA=levefunk.levekaar OUT=levefunk.levekaar;
PROC CONTENTS DATA= levefunk.levekaar;
TITLE 'Innhold opplastet datasett';
RUN;
ENDRSUBMIT;
RUN;
```

Når variablene er tilknyttet formater, må disse overføres først. Formatene kan overføres slik:

```
LIBNAME library 'x:\hu95\kat';
RSUBMIT;
LIBNAME library '$HELSEUND/g95/kat';
PROC UPLOAD INLIB=library OUT=library ;
RUN;
ENDRSUBMIT;
RUN;
```

Ønsker vi ikke å overføre data med PROC UPLOAD, kan vi bruke en tre-trinns-metode med PROC CPORT, filoverføring og PROC CIMPORT isteden. Først kjører vi PROC CPORT i Windows:

```
PROC CPORT DATA=tilbfall.utvalg FILE='../work/transfil';
```

Programmet over vil overføre et SAS datasett til en transportfil. Libref (her: tilbfall) må være definert på forhånd. FILE option brukes til å gi transportfilen et navn (her ../work/transfil). Ønsker vi å overføre alle SAS datasett med samme libref, blir programmet slik:

```
PROC CPORT LIB=tilbfall FILE='../work/transfil' ;
```

Vi kan ikke velge å overføre noen få datasett fra et område (samme libref). Ønsker vi det, må disse datasettene kopieres til et eget område før vi kan lage transportfil av dem (SELECT- eller EXCLUDE-option kan brukes til å velge ut formater i et formatbibliotek, men ikke til å velge ut SAS datasett fra et område). Når PROC CPORT er utført, vil det være laget en transportfil som i dette tilfellet vil hete transfil. Denne må overføres binært til Unix (velg binær overføring i Windows FTP-filoverføring). Etter at transportfilen er overført til Unix, må den konverteres til SAS datasett igjen. Dette gjør vi med PROC CIMPORT i Unix SAS:

```
PROC CIMPORT LIB=tilbfall INFILE='../work/transfil' ;
```

Her gjelder også at libref må være definert på forhånd. Dette programmet overfører alle datasett som befinner seg i transportfilen. Ønsker vi bare å hente et av datasettene kan vi gjøre slik:

```
PROC CIMPORT DATA=tilbfall.utvalg INFILE='../work/transfil';
```

Vi kan også her hente enten alle eller kun ett datasett fra transportfilen.

2.1.3 SAS datasett fra Unix til PC

For å overføre data fra Unix til Windows kan vi bruke PROC DOWNLOAD. Vi lager et program der noen setninger skal eksekveres lokalt (under Windows) og noen på Unix. Vi bruker DM-setningen til å logge oss ønsket arbeidsstasjon (her: ursus). De setninger som skal sendes til Unix, legger vi mellom RSUBMIT- og ENDRSUBMIT-setningene:

```
DM 'SIGNON ursus';
RUN;
RSUBMIT;
LIBNAME tilbfall '$KRIMINAL/tilbfall/work';
PROC DOWNLOAD DATA=tilbfall.dod OUT=dod ;
RUN;
ENDRSUBMIT;
PROC CONTENTS DATA=dod;
TITLE 'Innhold nedlastet datasett';
RUN;
```

Har vi et formatbibliotek som skal overføres, bør dette ligge på et eget område. Hvis det ikke gjør det, vil alle andre datasett også overføres med dette programmet:

```
LIBNAME library '..\kat';
RSUBMIT;
LIBNAME library '$KRIMINAL/tilbfall/kat';
PROC DOWNLOAD INLIB=library OUT=library ;
RUN;
ENDRSUBMIT;
RUN;
```

Dette programmet definer formatbibliotek på begge plattformer. Deretter overføres formatene. Når vi skal overføre datasett der variable er tilknyttet formater, må formatene overføres først.

Hvis vi ikke ønsker å overføre data med PROC DOWNLOAD, kan vi bruke en tre-trinns-metode med PROC CPORT, filoverføring og PROC CIMPORT isteden. Til denne overføringen bruker vi først PROC CPORT på Unix:

```
PROC CPORT DATA=tilbfall.utvalg FILE='../work/transfil';  
RUN;
```

Dette programmet vil overføre et SAS datasett til en transportfil. Libref (her tilbfall) må være definert på forhånd. FILE option brukes til å gi transportfilen et navn (her transfil). Ønsker vi å overføre alle SAS datasett med samme libref, blir programmet slik:

```
PROC CPORT LIB=tilbfall FILE='../work/transfil' ;
```

Vi kan ikke velge å overføre noen få datasett fra et område (samme libref). Ønsker vi det, må disse datasettene kopieres til et eget område før vi kan lage transportfil av dem.

Når PROC CPORT er utført, vil det være laget en transportfil som i dette tilfellet vil hete transfil. Denne må overføres binært til PC'en (velg binær overføring i Windows FTP-filoverføring). Etter at transportfilen er overført til PC, må den konverteres til SAS datasett igjen. Dette gjør vi med PROC CIMPORT i Windows SAS:

```
PROC CIMPORT LIB=tilbfall INFILE='x:\tilbfall\transfil' ;
```

Her gjelder også at libref må være definert på forhånd. Dette programmet overfører alle datasett som befinner seg i transportfilen. Ønsker vi bare å hente et av datasettene kan vi gjøre slik:

```
PROC CIMPORT DATA=tilbfall.utvalg INFILE='x:\tilbfall\transfil';
```

Her kan vi også velge å hente enten alle eller kun ett datasett fra transportfilen.

2.1.4 Skilletegnseparerte filer

2.1.4.1 Lage SAS datasett

For at SAS skal lese filer der feltene er adskilt med et skilletegn, må vi bruke såkalt liste-input med angivelse av hva som skal være skilletegn. Skilletegnene er vanligvis semikolon, komma eller tabulator. Når feltene adskilles med skilletegn betyr det feltene posisjoner i datafilen vil variere fra record til record, avhengig av hvor lange feltverdiene er.

Eksempel på en semikolonseparert adressefil (vi legger merke til at Nøtterøy sosialkontor ikke har noen gateadresse):

```
RÅDE KOMMUNE;SOSIALKONTORET;1640;RÅDE  
KONGSVINGER SOSIALKONTOR;SKOLEGT. 1;2200;KONGSVINGER  
HAMAR SOSIALKONTOR;POSTBOKS 300;2301;HAMAR  
RINGSAKER KOMMUNE;SOSIALKONTORET;2380;BRUMUNDDAL  
LØTEN KOMMUNE;SOSIALKONTORET;2340;LØTEN  
STANGE KOMMUNE;SOSIALKONTORET;2310;STANGE  
NORD-ODAL KOMMUNE;SOSIALKONTORET;2120;SAGSTUA  
NØTTERØY SOSIALKONTOR;;3140;BORGHEIM  
TJØME SOSIALKONTOR;POSTBOKS 55;3145;TJØME
```

Program som leser inn en semikolonseparert fil:

```
DATA adr;  
INFILE 'x:\soshjelp\data\sosktr.adr' DELIMITER=';' DSD MISSEVER PAD LRECL=52;  
INPUT   navn      : $CHAR30.  
        adresse   : $CHAR40.  
        postnr    : $CHAR4.  
        poststed  : $CHAR25.  
;  
PROC PRINT DATA=adr;  
  TITLE 'Semikolonseparert fil innlest';  
RUN;
```

INFILE-setningen er den sentrale her. Skilletegnet defineres med DELIMITER=';' (er skilletegnet tabulator velges DELIMITER='09'X). Med DSD-parameteren sørger vi for at tomme felt ikke blir oversett (uten denne vil Nøtterøy sosialkontor få 3140 som gateadresse og MISSING både på postnr og poststed). LRECL=52 sier at

den lengste recorden som leses inn er 52 posisjoner. Denne er spesielt viktig å ha med når recorden er mer 132 posisjoner, men kan også brukes ved kortere recordlengder.

Med kolon etter variabelnavnet i INPUT-setningen kan vi bruke innlesingsformater når vi bruker liste-input. Dette må vi gjøre her, ellers vil hvert felt bli 8 posisjoner langt (Windows), noe som er for kort for alle unntatt postnr. DSD-parameteren vil også fjerne eventuelle fnutter rundt feltene (f.eks. "RÅDE KOMMUNE ") slik at vi ikke behøver å bekymre oss med å fjerne disse før innlesing.

Har vi numeriske variable med desimaler, må vi først finne ut hvordan desimaltegnene er representert:

4578876.323	Alt OK. SAS leser dette uten problem (bortsett fra COMMAX-formatet som gir feil tall)
4,578,876.323	SAS greier ikke å tolke kommaene uten videre. Blir riktig med innlesingsformatet COMMAw.
4578876,323	SAS greier ikke å tolke kommaene uten videre. Blir riktig med innlesingsformatet COMMAXw. eller NUMXw. (COMMAw gir feil tall)
4.578.876,323	SAS greier ikke å tolke kommaene uten videre. Blir riktig med innlesingsformatet COMMAXw.
4 578 876,323	SAS greier ikke å tolke blank som tusenskilte uten videre. Må bruke innlesingsformatet COMMAXw. (COMMAw gir feil tall)

Vi kan med en SAS-setning fortelle SAS at vi ønsker innlesingsformatet COMMAX32. brukt på alle numeriske variable. Denne SAS-setningen må plasseres før INPUT-setningen:

```
INFORMAT  DEFAULT=COMMAX32.;
```

Nå skal alle tall (også de med desimaler) bli riktig tolket ved innlesingen for tall med punktum eller blank som tusenskilte og komma som desimalskilte.

2.1.4.2 Lage skilletegnseparerte filer

Å lage skilletegnseparerte filer fra SAS er ganske enkelt. Har vi få variable som skal skrives ut kan vi bruke et program som dette (semikolon som skilletegn):

```
DATA _NULL_ ;
SET adr;
FILE 'sosktr3.adr' ;
PUT navn      +(-1) ' ';
   adresse  +(-1) ' ';
   postnr   +(-1) ' ';
   poststed
;
```

Variablene skrives ut med PUT-setningen i den rekkefølgen vi ønsker med et semikolon mellom. SAS lager automatisk en blank mellom hver variabel som skrives ut, så for å oppheve dette flytter vi lesehodet en posisjon tilbake (+(-1)) etter utskrift av en variabel.

Har vi mange variable vi ønsker å skrive ut og de er alle enten numeriske eller karaktvariable, kan vi skrive ut på denne måten (her bruker vi tabulator ('09'X) som skilletegn):

```
DATA _NULL_ ;
SET adr;
FILE 'sosktr4.adr' ;
ARRAY alle (*) $ navn -- poststed ;
DO i = 1 to DIM(alle) ;
   PUT alle(i)  +(-1) '09'X @ ;
END ;
PUT ;
```

Vi definerer alle variablene i en array kalt alle. Det kan vi gjøre fordi alle variablene er definert som karaktvariable (har vi både numeriske og karaktvariable, må de defineres i hver sin array). Videre lager vi en DO-løkke som går gjennom alle variablene i arrayen (DIM(alle) er antall elementer i arrayen alle). For hver variabel, skriver vi ut variabelen og semikolonet. Dessuten holder vi lesehodet på den samme recorden på fila (@), slik at vi for alle variablene på den samme recorden. Etter DO-løkken, bruker vi PUT for å flytte lesehodet til neste record på utfila.

Det vil ofte være en fordel å lage overskrifter til kolonnene som lages. Til dette kan vi bruke variabelenes labler. Programmet under lager overskrifter til kolonnene:

```
DATA _NULL_ ;
SET adr;
FILE 'sosktr.adr' ;
ARRAY alle (*) $ navn -- poststed ;
LENGTH overskr $ 40 ;
IF _N_ = 1 THEN
DO;
DO i = 1 TO DIM(alle) ;
CALL LABEL(alle(i),overskr);
PUT overskr +(-1) '09'X @ ;
END;
PUT ;
END;
DO i = 1 TO DIM(alle) ;
PUT alle(i) +(-1) '09'X @ ;
END ;
PUT ;
RUN;
```

Vi bruker CALL LABEL til å hente variabelenes labler og legger dem etter tur i variabelen overskr. Har ikke variablene labler, brukes variabelnavnet. Deretter skrives de til utfila. Dette gjøres kun en gang, aller først (IF _N_ = 1). Blir recordlengden lang, må vi ha med LRECL-option i FILE-setningen.

Programmet over kan gjøres mer generelt ved å definere arrayen på en litt annen måte. Vi kan definere en array som inneholder alle karaktervariable hittil brukt i programmet uten å navngi en eneste av dem. ARRAY-setningen blir da slik:

```
ARRAY alle (*) $ _CHARACTER_ ;
```

Vi må sørge for å definere ARRAY-setningen før LENGTH-setningen for å forhindre at variabelen *overskr* blir en del av arrayen.

2.1.5 Excel regneark

Foreløpig er den sikreste måten å overføre data fra Excel til SAS datasett å gå omveien om en semikolonseparert fil. Lagre regnearket som en SDV-fil, velg SDV(Semikolonskilt) som filtype i Lagre sommenyen i Excel. Les deretter inn den semikolonseparerte filen slik som beskrevet over.

Når vi skal overføre data fra SAS til Excel, er det altså sikrest å gå omveien om en skilletegnseparert fil. Datafilen lager vi som i eksempelet foran, bruk helst semikolon eller tabulator som skilletegn. For å få æ, ø og å riktig må vi velge enten Windows (ANSI) eller MS-DOS eller OS/2 som filopprinnelse i tekstimportveiviseren i Excel. Hva vi skal velge finner vi ved å se på teksten i forhåndsvisningen av regnearket. Når æ, ø eller å vises riktig har vi den riktige varianten.

Vi kan også bruke såkalt DDE-kobling for å hente data fra Excel til SAS. Dette krever at regnearket er hentet inn til Excel før vi foretar DDE-koblingen. For at koblingen skal gå bra, må vi ha med flere parametre i INFILE-setningen. Det lønner seg å bruke dette med forsiktighet, da det kan skje uforutsette ting.

Eksempel på DDE-kobling:

LUFT.XLS						
	A	B	C	D	E	F
1	Utslipp til luft 1. 1973-1992. 1 000 tonn dersom ikke annet er oppgitt					
2	År	CO2(Mill.tonn)	SO2	NOx		
3	1992	34,4	37,2	220,3		
4	1991	33,9	45,4	220,8		
5	1990	35,5	53,5	229,7		
6	1989	35,3	58,9	232,3		
7	1988	35	67	228,6		
8	1987	34,7	74,4	237		


```

FILENAME xl DDE 'EXCEL|X:\UTSLIPP\DATA\[LUFT.XLS]ark1!r3c1:r8c4' ;
DATA utslipp;
INFILE xl MISSOVER DLM='09'X NOTAB DSD ;
INPUT aar : $CHAR4.
      co2 : NUMX8.
      so2 : NUMX8.
      nox : NUMX8. ;

```

Vi bruker FILENAME-setningen til å beskrive DDE-koblingen, og rekke 3-8 og kolonne 1-4 fra regnearket.

2.1.6 SPSS-filer

Den sikreste måten å få lest SPSS-filer på, krever at de er laget med SPSS transportformat. Dette må gjøres i SPSS. Hvis filene er i SPSS transportformat vil vi kunne finne teksten **ASCII SPSS PORT FILE** i begynnelsen av filene. Dette kan vi f. eks. sjekke ved å se på den i Windows Notisblokk. Vi bør bruke SAS på den samme plattformen (Windows el. Unix) som SPSS-filen ble laget på når vi skal lese SPSS-filer. Foruten transportformatet, som kan brukes fra alle SPSS-versjoner, støtter SAS SPSS-filer laget med en gammel PC-dos-versjon.

Vi bruker LIBNAME-setningen med lagringsformat SPSS for å henvise til SPSS-filen, som her altså refererer til en fysisk fil, ikke bare et område slik den vanligvis gjør. Når referansen til filen er opprettet kan vi kopiere den over til SAS-format. SPSS opererer ikke med egne filnavn på sine filer, det betyr at vi kan velge helt fritt når vi leser filen, slik det er gjort i SET-setningen i programmet under. SAS kan få problemer med å tolke dato- og tid variable. Har vi slike bør disse sjekkes nøye etter at filen er gjort om til SAS datasett. Vi bør også sjekke innholdet med PROC CONTENTS og liste noen observasjoner med PROC PRINT slik som i eksempelet under.

```

LIBNAME konv SPSS 'sk22.por';
DATA skjema22;
SET konv.sk22;
PROC CONTENTS DATA=skjema22;
TITLE 'Innhold i SAS datasett laget fra SPSS' ;
PROC PRINT DATA=skjema22 (OBS=54) N UNIFORM;
TITLE 'Noen observasjoner fra skjema 22' ;

```

Når vi skal levere data til eksterne oppdragsgivere som bruker SPSS, kan vi levere SAS-datasett i et spesielt transportformat som SPSS kan lese (et av SAS sine gamle transportformater, versjon 6.06). Vi kan klargjøre et SAS-datasett for SPSS med et program som dette:

```

LIBNAME barnvern '$BARNVERN/wk12';          * Området med SAS-dataene ;
LIBNAME konv V606 '$BARNVERN/wk9/v606';      * Området med SAS-data i V606-format ;
LIBNAME xp XPORT '$BARNVERN/wk9/trans';      * Navn på transportfilen ;
PROC COPY IN=barnvern OUT=konv ;
SELECT bvud98 ;
PROC COPY IN=konv OUT=xp;
SELECT bvud98;

```

Først må vi gi en libref til SAS-dataene våre (barnvern) med en LIBNAME-setning, disse er lagret som versjon 8.0 SAS datasett. Deretter må vi definere en libref for versjon 6.06 SAS datasett (konv). Så må vi opprette en libref for data lagret i transportformat (xp). Til slutt bruker vi PROC COPY for å kopiere dataene først til V606 lagringsformat, deretter ut til transportformat.

Er det knyttet formater til variablene i datasettet, kan disse tas med over til SPSS. Da må vi definere libref for formatene (library), hente formatene fra formatbiblioteket ut til et SAS datasett i versjon 6.06 lagringsformat ved hjelp av PROC FORMAT og så til slutt ta med dette SAS datasettet i kopieringen.

```

LIBNAME barnvern '$BARNVERN/wk12';          * Området med SAS-dataene ;
LIBNAME library '$BARNVERN/kat';            * Området med SAS-formater ;
LIBNAME konv V606 '$BARNVERN/wk9/v606';      * Området med SAS-data i V606-format ;
LIBNAME xp XPORT '$BARNVERN/wk9/trans';      * Navn på transportfilen ;
PROC FORMAT CNTLOUT=konv.format LIBRARY=library;
PROC COPY IN=barnvern OUT=konv ;
SELECT bvud98 ;
PROC COPY IN=konv OUT=xp;
SELECT bvud98 format;

```

Transportfilen (her \$BARNVERN/wk9/trans) må så overføres binært til windows NT vha. et FTP-program. SPSS-programmet som leser inn dataene kan se slik ut (har vi ingen formater fjerner vi FSET(format):

```

GET SAS DATA='h:\krl\barnvern\trans' DSET(bv98) FSET(format).
SAVE OUTFILE='h:\krl\barnvern\wk9\bv98.sav'.

```

2.2 Konvertering av variable

Når en variabel har fått en lengde og type, kan denne ikke endres. Ønsker vi å endre lengden og/eller typen, må vi lage en ny variabel med de attributter vi ønsker. Det vil f.eks. være nødvendig å gjøre slike konverteringer når vi skal koble datasett, og koblingsvariabelen er av type numerisk i det ene datasettet og karakter i det andre.

Fra karakter til numerisk:

```
DATA persny (DROP=fnrmidl);
  SET pers (RENAME=(fnr=fnrmidl));
  fnr = fnrmidl + 0;
  LABEL fnr = 'Fødselsnummer' ;
RUN;
```

Fra numerisk til karakter:

Programmet under gjør fødselsnummer om fra numerisk variabel til karaktervariabel. Vi må huske å lage LABEL for den nye variabelen, den overføres ikke fra den gamle variabelen. Alle fødselsnummer som inneholder annet enn tall settes til MISSING.

```
DATA persny (DROP=fnrmidl);
  SET pers (RENAME=(fnr=fnrmidl));
  fnr = COMPRESS(PUT(fnrmidl,Z11.),'.');
  LABEL fnr = 'Fødselsnummer' ;
RUN;
```

Vi bruker PUT-funksjonen for å putte den numeriske variabelen *fnrmidl* ut i et annet format, *Z11*. Den nye karaktervariabelen får en lengde på 11. Utskriftsformatet *Z11*. gjør at det legges på ledende null når nummeret er færre enn 11 siffer. MISSING representeres vanligvis med punktum (.) for numeriske variable og blank for karaktervariable. Med COMPRESS-funksjonen fjernes eventuelle punktum (MISSING) og vi står igjen med blank (som er MISSING-verdi for karaktervariable).

Fra dato til karakter:

```
DATA persny (DROP=fdtmidl);
  SET pers (RENAME=(fdato=fdtmidl));
  fdato = COMPRESS(PUT(fdtmidl,YYMMDD6.),'.');
  LABEL fdato = 'Fødselsdato' ;
RUN;
```

Dette gir oss dato med 2-sifret årstall. Vi bør ha 4-sifret og bruker denne setningen isteden:

```
fdato = COMPRESS(PUT(fdtmidl,YYMMDDN8.),'.');
```

Fra karakter til dato:

```
DATA persny (DROP=fdtmidl);
  SET pers (RENAME=(fdato=fdtmidl));
  fdato = INPUT(fdtmidl,DDMMYY6.);
  LABEL fdato = 'Fødselsdato' ;
RUN;
```

Her bruker vi funksjonen INPUT. som returnerer en verdi ut fra et innlesningsformat. Vi må være klar over at ugyldige datoer (31. juni 1998, 29. februar 1997, 000000 etc.) settes til MISSING.

Fra numerisk til dato:

```
DATA persny (DROP=fdtmidl);
  SET pers (RENAME=(fdato=fdtmidl));
  fdato = INPUT(PUT(fdtmidl,Z6.),DDMMYY6.);
  LABEL fdato = 'Fødselsdato' ;
RUN;
```

Når datoen er lagret numerisk, men ikke som en SAS dato, må vi konvertere til karakterverdi med PUT-funksjonen først, for deretter å bruke INPUT-funksjonen som i forrige eksempel. Begge deler kan gjøres i samme SAS-setning.

2.3 Kobling av SAS datasett

2.3.1 Merge

Før vi kobler datasett med MERGE-setningen må vi vite hvordan SAS utfører en slik kobling. Kobling kan gjøres med eller uten nøkkel, vi anbefaler sterkt alltid å koble med nøkkel. Å koble med nøkkel vil si at vi på hvert av datasettene har en (eller flere) variable som skal brukes som koblingsbegrep. Nøkkelen, som ofte kalles BY-variable, angir vi ved å bruke en BY-setning. La oss først se på to SAS datasett, *pas94* og *pas95*. De ser slik ut:

<i>pas94</i> :	<i>pas95</i> :
1001 100	1001 700
2002 100	1001 800
2002 200	1001 900
2002 300	2002 700
3003 100	3003 700
3003 200	3003 800
3003 300	3003 900
4004 100	4004 700
4004 200	4004 800
5005 100	4004 900
5005 200	5005 700
5005 300	5005 800
6006 100	7007 700

Vi har to variable på datasettet *pas94*: *inr* og *innl94*. Datasettet *pas95* inneholder variablene *inr* og *innl95*.

De to datasettene skal kobles med *inr* som nøkkel. Ved å se nærmere på de to datasettene før koblingen, ser vi at begge har dubletter (flere observasjoner med samme nøkkel). Spørsmålet blir da: Hvordan kobler SAS datasett når vi bruker MERGE med BY?

La oss se på alle mulighetene ved en kobling:

1. Det først nevnte datasettet (*pas94*) har ingen dubletter, men det andre har (tilfelle for *inr* 1001). Her vil den ene observasjonen fra datasett *pas94* kobles med alle med tilsvarende nøkkel i *pas95*.
2. Det først nevnte datasettet (*pas94*) har dubletter, men det andre har ikke (tilfelle for nøkkel 2002). Her vil den ene observasjonen fra datasett *pas95* kobles med alle med tilsvarende nøkkel i *pas94*. Dette tilsvarende den første situasjonen.
3. Like mange dubletter i begge datasettet (tilfelle for nøkkel 3003). Den første fra *pas94* vil kobles med den første fra *pas95*. Den andre fra *pas94* vil kobles med den andre fra *pas95*. Slik fortsetter det så lenge det er dubletter med den samme nøkkelen.
4. Begge datasettene har dubletter, men datasett *pas94* har færre enn datasett *pas95* (tilfelle for nøkkel 4004). Den første dubletten i datasett *pas94* vil kobles med den første i datasett *pas95*. Videre vil den neste fra datasett *pas94* kobles med den neste fra datasett *pas95*. Slik vil koblingen fortsette så lenge antallet dubletter er likt i begge filene. Siden *pas94* har færre dubletter enn *pas95*, vil den siste dubletten kobles med de resterende dublettene i *pas95*.
5. Begge datasettene har dubletter, men datasett *pas95* har færre enn datasett *pas94* (tilfelle for nøkkel 5005). Situasjonen er lik den forrige og SAS vil behandle den på samme måte, bortsett fra at den siste dubletten fra *pas95* vil kobles med de resterende fra *pas94*.
6. Datasett *pas94* har ingen makker i *pas95* (tilfelle for nøkkel 6006). Her vil observasjonen havne på det nye datasettet med variabelen *innl95* satt til MISSING.
7. Datasett *pas95* har ingen makker i *pas94* (tilfelle for nøkkel 7007). Her vil observasjonen havne på det nye datasettet med variabelen *innl94* satt til MISSING.

Her er et enkelt program som kobler de to filene:

```
DATA pasient;
  MERGE pas94
        pas95;
  BY inr;
```

Resultatet:

OBS	INR	INN194	INN195
1	1001	100	700
2	1001	100	800
3	1001	100	900
4	2002	100	700
5	2002	200	700
6	2002	300	700
7	3003	100	700
8	3003	200	800
9	3003	300	900
10	4004	100	700
11	4004	200	800
12	4004	200	900
13	5005	100	700
14	5005	200	800
15	5005	300	800
16	6006	100	
17	7007	.	700

Resultatet vil bli det samme uansett hvilket datasett som nevnes først i MERGE-setningen. Dette gjelder så lenge alle variablene (unntatt BY-variabelen, nøkkelen) på de to datasettene har forskjellig navn. Har andre variable enn BY-variablene samme navn på begge datasettene, kan rekkefølgen de nevnes i ha stor betydning for resultatet. La oss se på et eksempel. Vi endrer navnet på variabelen *inn194* i datasett pas94 til *innlagt* og tilsvarende for pas95, slik at begge får variabelen innlagt.

Vi kjører koblingen om igjen uten endringer i programmet. Resultatet blir da slik:

OBS	INR	INNLAGT
1	1001	700
2	1001	800
3	1001	900
4	2002	700
5	2002	200
6	2002	300
7	3003	700
8	3003	800
9	3003	900
10	4004	700
11	4004	800
12	4004	900
13	5005	700
14	5005	800
15	5005	300
16	6006	100
17	7007	700

Ved å bytte om rekkefølgen datasettene nevnes i, blir programmet slik:

```
DATA pasient;  
  MERGE pas95  
        pas94;  
  BY inr;
```

Resultatet blir nå et annet:

OBS	INR	INNLAGT
1	1001	100
2	1001	800
3	1001	900
4	2002	100
5	2002	200
6	2002	300
7	3003	100
8	3003	200
9	3003	300
10	4004	100
11	4004	200
12	4004	900
13	5005	100
14	5005	200
15	5005	300
16	6006	100
17	7007	700

Hvorfor blir så resultatet forskjellig avhengig av hvilket datasett vi nevner først i MERGE-setningen? Grunnen er at SAS beholder verdien på den variabelen som sist har bidratt med data (også om denne verdien er MISSING). Når datasettene kobles, vil variabelen *innlagt* fra det sist leste datasettet bidra med data sist (den er lest sist) og derfor også bli tatt med på utdatasettet. Så lenge datasettene har like mange dubletter med samme nøkkel, vil data fra det sist nevnte datasettet bli skrevet til utdatasettet. Det samme vil skje når det er flere dubletter på det sist nevnte datasettet enn på det først nevnte. Hvis det derimot er flere dubletter på det først

nevnte datasettet, vil det første datasettet begynne å bidra med data når det andre datasettet ikke har flere dubletter å bidra med og data fra det første datasettet vil bli skrevet til utdatasettet.

Før vi skal koble 2 datasett, må vi, i tillegg til å ha bestemt oss for hva vi ønsker å gjøre med dubletter på datasettene, avklare hva vi ønsker å ha med på utdatasettet (eller utdatasettene). Hva skal vi gjøre i følgende situasjoner:

- Datasettene kobler (samme nøkkel i begge datasettene)
- Det første datasettet har ikke makker i det andre (ingen kobling)
- Det andre datasettet har ikke makker i det første (ingen kobling)

For å behandle disse 3 variantene, innfører vi en hjelpevariabel for hvert datasett som brukes til å sjekke om datasettet bidrar med data eller ikke. Disse (boolske) variablene defineres i MERGE-setningen med IN= option (se eksempelet under) og de tas ikke med på utdatasettet. I programmet under ønsker vi bare å ta med data som kobler på utdatasettet. Det vil ofte lønne seg å lage meldinger når de andre situasjonene oppstår, og det er gjort i programmet:

```
DATA pasient;
MERGE pas94 (IN=p94)
      pas95 (IN=p95);
BY inr;
IF p94 AND p95 THEN OUTPUT;
ELSE
  IF p94 THEN
    PUT 'Instnr ' inr ' med innlagte ' innl94 ' finnes ikke i pas95!' ;
  ELSE
    PUT 'Instnr ' inr ' med innlagte ' innl95 ' finnes ikke i pas94!' ;
```

Eksempel der vi ønsker å ta med alle observasjoner fra pas94:

```
DATA pasient;
MERGE pas94 (IN=p94)
      pas95 (IN=p95);
BY inr;
IF p94 THEN OUTPUT;
IF p94 AND NOT p95 THEN
  PUT 'Instnr ' inr ' med innlagte ' innl94 ' finnes ikke i pas95!' ;
ELSE
  IF p95 AND NOT p94 THEN
    PUT 'Instnr ' inr ' med innlagte ' innl95 ' finnes ikke i pas94!' ;
```

Et eksempel som lager en oppsummering av koblingen:

```
DATA pasient (DROP=koble bare94 bare95);
MERGE pas94 (IN=p94)
      pas95 (IN=p95) END=slutt;
BY inr;

IF p94 AND p95 THEN koble+1;
ELSE
  IF p94 AND NOT p95 THEN bare94 + 1;
  ELSE
    IF p95 AND NOT p94 THEN bare95 + 1;
IF slutt THEN
DO;
  PUT 'Antall som kobler : ' koble;
  PUT 'Antall bare i 94 : ' bare94;
  PUT 'Antall bare i 95 : ' bare95;
END;
```

2.3.2 Update

UPDATE bruker vi når vi ønsker å oppdatere et register (masterdatasett) med en korreksjonsfil (transaksjonsdatasett). UPDATE har en litt annen logikk enn MERGE. En hovedforskjell er behandling av manglende verdier. Har en variabel på transaksjonsdatasettet MISSING verdi, vil ikke masterdatasettets variabel oppdateres (dette kan overstyres med parameteren UPDATEMODE=MISSINGCHECK i UPDATE-setningen, se side 83). Ved bruk av MERGE skilles det ikke mellom MISSING og andre verdier på denne måten. Forskjeller mellom MERGE og UPDATE er det også når det er dubletter på et eller begge datasettene.

Dette er masterdatasettet før oppdatering (uten dubletter):

OBS	FNR	K	B I D R A G																
				L	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
				A	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
				A	D	D	D	D	D	D	D	D	D	D	D	D	1	1	1
S	R	M	G	N	1	2	3	4	5	6	7	8	9	0	1	2			
1	02016634501	0106	150234	0	00	02	00	00	00	00	00	00	00	00	00	00	00	00	00
2	04026542492	0101	.	324	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00	00	00
4	21114344242	0106	3453	12756	00	00	00	00	00	00	00	07	08	09	00	00	00	00	00
5	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00	00	00
6	30045623358	0101	234810	0	01	02	03	04	05	06	07	08	09	10	11	12			
7	30069200000	0211	11984	.	00	00	00	00	00	00	00	00	00	00	10	11	00		

Her er transaksjonsdatasettet (uten dubletter):

OBS	FNR	BIDRAG	LAAN
1	02016634501	50234	.
2	04026542492	0	.
3	30045623358	23481	.
4	30069200000	11984	1465

Programmet:

```
DATA sorskorr;  
  UPDATE soshjelp retting ;  
  BY fnr ;
```

Resultatet blir slik:

ETTER oppdatering. Ingen dubletter

OBS	FNR	K	B I D R A G																
				L	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
				A	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
				A	D	D	D	D	D	D	D	D	D	D	D	D	1	1	1
S	R	M	G	N	1	2	3	4	5	6	7	8	9	0	1	2			
1	02016634501	0106	50234	0	00	02	00	00	00	00	00	00	00	00	00	00	00	00	00
2	04026542492	0101	0	324	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00	00	00
4	21114344242	0106	3453	12756	00	00	00	00	00	00	00	07	08	09	00	00	00	00	00
5	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00	00	00
6	30045623358	0101	23481	0	01	02	03	04	05	06	07	08	09	10	11	12			
7	30069200000	0211	11984	1465	00	00	00	00	00	00	00	00	00	00	10	11	00		

Vi ser at verdier fra transaksjonsdatasettet (retting) blir lagt ut på det nye masterdatasettet (sorskorr) når fnr er likt i datasettene og variablene på transaksjonsdatasettet har verdier. MISSING-verdier i transaksjonsdatasettet blir altså ikke overført.

Er det dubletter på masterdatasettet, vil bare den første av disse bli oppdatert. Det vil også komme en advarsel i SASlog om dubletter på masterdatasettet. Vi kjører det samme programmet som over, men masterdatasettet har nå fått en dublett:

OBS	FNR	K	B I D R A G																
				L	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
				A	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
				A	D	D	D	D	D	D	D	D	D	D	D	D	1	1	1
S	R	M	G	N	1	2	3	4	5	6	7	8	9	0	1	2			
1	02016634501	0106	150234	0	00	02	00	00	00	00	00	00	00	00	00	00	00	00	00
2	04026542492	0101	.	324	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00	00	00
4	21114344242	0106	3453	12756	00	00	00	00	00	00	00	07	08	09	00	00	00	00	00
5	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00	00	00
6	30045623358	0101	234810	0	01	02	03	04	05	06	07	08	09	10	11	12			
7	30069200000	0211	11984	.	00	00	00	00	00	00	00	00	00	00	10	11	00		
8	30069200000	0235	4608	.	01	00	00	00	05	00	00	00	00	00	00	00	00	00	00

Bare den første av de to observasjonene med samme fnr blir oppdatert, den andre beholder sin opprinnelige verdi. Forskjellen fra å bruke MERGE er at MISSING-verdier ikke overføres ved UPDATE.

ETTER oppdatering. Dublett på masterdatasettet

O B S	F N R	K O M M	B I D R A G													M N D 0	M N D 1	M N D 2
				L	M	M	M	M	M	M	M	M	M	M	M			
				A	N	N	N	N	N	N	N	N	N	N	N			
				A	D	D	D	D	D	D	D	D	D	D	D			
				N	1	2	3	4	5	6	7	8	9					
1	02016634501	0106	50234	0	00	02	00	00	00	00	00	00	00	00	00	00	00	
2	04026542492	0101	0	324	00	00	00	00	00	00	00	00	00	00	00	00	12	
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00	
4	21114344242	0106	3453	12756	00	00	00	00	00	00	07	08	09	00	00	00	00	
5	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00	
6	30045623358	0101	23481	0	01	02	03	04	05	06	07	08	09	10	11	12		
7	30069200000	0211	11984	1465	00	00	00	00	00	00	00	00	00	10	11	00		
8	30069200000	0235	4608	.	01	00	00	00	05	00	00	00	00	00	00	00	00	

Nå prøver vi oss med dublett på transaksjonsdatasettet, men ikke på masterdatasettet.

Master (uten dubletter):

O B S	F N R	K O M	B I D R A G													M N D	M N D	M N D
				L	M	M	M	M	M	M	M	M	M	M	M			
				A	D	D	D	D	D	D	D	D	D	D	D	D	1	
N	1	2	3	4	5	6	7	8	9	0	1	2						
1	02016634501	0106	150234	0	00	02	00	00	00	00	00	00	00	00	00	00	00	
2	04026542492	0101	.	324	00	00	00	00	00	00	00	00	00	00	00	00	12	
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00	
4	21114344242	0106	3453	12756	00	00	00	00	00	00	07	08	09	00	00	00	00	
5	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00	
6	30045623358	0101	234810	0	01	02	03	04	05	06	07	08	09	10	11	12		
7	30069200000	0211	11984	.	00	00	00	00	00	00	00	00	00	10	11	00		

Transaksjon (med dublett):

OBS	FNR	BIDRAG	LAAN
1	02016634501	50234	.
2	04026542492	0	.
3	30045623358	23481	.
4	30069200000	11984	1465
5	30069200000	2365	.

Resultatet (programmet er det samme som sist):

ETTER oppdatering. Dublett på transaksjonsdatasettet

O B S	F N R	K O M	B I D R A G													M N D	M N D	M N D
				L A N	M A D	M N D	M N D	M N D	M N D	M N D	M N D	M N D	M N D	M N D				
1	02016634501	0106	50234	0	00	02	00	00	00	00	00	00	00	00	00	00	00	
2	04026542492	0101	0	324	00	00	00	00	00	00	00	00	00	00	00	00	12	
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00	
4	21114344242	0106	3453	12756	00	00	00	00	00	00	07	08	09	00	00	00	00	
5	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00	
6	30045623358	0101	23481	0	01	02	03	04	05	06	07	08	09	10	11	12		
7	30069200000	0211	2365	1465	00	00	00	00	00	00	00	00	00	10	11	00		

Vi ser at den observasjonen som har dublett på transaksjonsdatasettet er blitt oppdatert to ganger, en for hver gang den er på transaksjonsdatasettet. Her gir UPDATE et helt annet resultat enn MERGE!

Nå sjekker vi hvordan UPDATE virker når det er dubletter på begge datasettene:

Master (med dublett):

O B S	F N R	K O M M	B I D R A G													M N D	M N D	M N D
				L	M	M	M	M	M	M	M	M	M	M	M			
				A	N	N	N	N	N	N	N	N	N	N	N			
				A	D	D	D	D	D	D	D	D	D	D	D			
				N	1	2	3	4	5	6	7	8	9	0	1			
1	02016634501	0106	150234	0	00	02	00	00	00	00	00	00	00	00	00	00	00	
2	04026542492	0101	.	324	00	00	00	00	00	00	00	00	00	00	00	00	12	
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00	
4	21114344242	0106	3453	12756	00	00	00	00	00	00	07	08	09	00	00	00	00	
5	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00	
6	30045623358	0101	234810	0	01	02	03	04	05	06	07	08	09	10	11	12		
7	30069200000	0211	11984	.	00	00	00	00	00	00	00	00	00	10	11	00		
8	30069200000	0235	4608	.	01	00	00	00	05	00	00	00	00	00	00	00	00	

Transaksjon (med dublett):

OBS	FNR	BIDRAG	LAAN
1	02016634501	50234	.
2	04026542492	0	.
3	30045623358	23481	.
4	30069200000	11984	1465
5	30069200000	2365	.

Resultatet:

ETTER oppdatering. Dublett på begge datasett

O	F	K	B I D	L	M	M	M	M	M	M	M	M	M	M	M	M	M
B	N	M	R	A	A	D	D	D	D	D	D	D	D	D	1	1	1
S	R	M	G	N	1	2	3	4	5	6	7	8	9	0	1	2	
1	02016634501	0106	50234	0	00	02	00	00	00	00	00	00	00	00	00	00	00
2	04026542492	0101	0	324	00	00	00	00	00	00	00	00	00	00	00	00	12
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00
4	21114344242	0106	3453	12756	00	00	00	00	00	00	07	08	09	00	00	00	00
5	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00
6	30045623358	0101	23481	0	01	02	03	04	05	06	07	08	09	10	11	12	
7	30069200000	0211	2365	1465	00	00	00	00	00	00	00	00	00	10	11	00	
8	30069200000	0235	4608	.	01	00	00	00	05	00	00	00	00	00	00	00	00

Vi ser at den første observasjonen som er dublett på masterdatasettet er blitt oppdatert to ganger, en for hver gang den er på transaksjonsdatasettet. Den andre dubletten på masterdatasettet har beholdt sine opprinnelige verdier. Også her gir UPDATE et helt annet resultat enn MERGE.

Til slutt prøver vi med variable og observasjoner som bare er på et av datasettene.

Master (uten dubletter):

O	F	K	B I D	L	M	M	M	M	M	M	M	M	M	M	M	M	M
B	N	M	R	A	A	D	D	D	D	D	D	D	D	D	1	1	1
S	R	M	G	N	1	2	3	4	5	6	7	8	9	0	1	2	
1	02016634501	0106	150234	0	00	02	00	00	00	00	00	00	00	00	00	00	00
2	04026542492	0101	.	324	00	00	00	00	00	00	00	00	00	00	00	00	12
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00
4	21114344242	0106	3453	12756	00	00	00	00	00	00	07	08	09	00	00	00	00
5	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00
6	28114343908	0301	5932	0	01	00	00	00	00	00	00	00	00	00	00	00	12
7	30045623358	0101	234810	0	01	02	03	04	05	06	07	08	09	10	11	12	
8	30069200000	0211	11984	.	00	00	00	00	00	00	00	00	00	10	11	00	

Transaksjon (uten dubletter, men med observasjoner som ikke er på master):

OBS	FNR	BIDRAG	LAAN	STATSB
1	02016634501	50234	.	000
2	04026542492	0	.	106
3	15127347232	103403	0	000
4	30045623358	23481	.	
5	30069200000	11984	1465	000

Resultatet:

ETTER korreksjon. Ingen dubletter, men obs. som ikke kobler

O	F	K	B I D	L	M	M	M	M	M	M	M	M	M	M	M	M	M	S T A T S B
B	N	M	R	A	A	D	D	D	D	D	D	D	D	D	1	1	1	S
S	R	M	G	N	1	2	3	4	5	6	7	8	9	0	1	2		B
1	02016634501	0106	50234	0	00	02	00	00	00	00	00	00	00	00	00	00	00	000
2	04026542492	0101	0	324	00	00	00	00	00	00	00	00	00	00	00	00	12	106
3	04079242449	0211	345	.	00	00	00	04	00	00	00	00	00	00	00	00	00	
4	15127347232		103403	0														000
5	21114344242	0106	3453	12756	00	00	00	00	00	00	07	08	09	00	00	00	00	
6	22095300000	0106	23123	0	00	02	00	04	00	00	00	00	00	00	00	00	00	
7	28114343908	0301	5932	0	01	00	00	00	00	00	00	00	00	00	00	00	12	
8	30045623358	0101	23481	0	01	02	03	04	05	06	07	08	09	10	11	12		
9	30069200000	0211	11984	1465	00	00	00	00	00	00	00	00	00	10	11	00	000	

Forskjellene mellom MERGE og UPDATE inntreffer altså ved behandling av MISSING-verdier og dubletter.

2.3.3 Join

Når vi kobler datasett ved å bruke MERGE, vil vi ikke få koblet alle observasjoner med nøkkelbegrepene i det ene datasettet med alle med samme nøkkel i det andre. For å få til dette (såkalt join, alle kobles med alle), må vi bruke PROC SQL som har denne muligheten.

Vi skal joine de to filene som er brukt i eksemplene med MERGE. Programmet kan bli slik:

```
PROC SQL;
  CREATE TABLE pas9495a AS
    SELECT pas94.*, pas95.*
    FROM pas94, pas95
    WHERE pas94.inr = pas95.inr
  ;
```

Dette programmet lager en sql-tabell (pas9495a) ved å joine datasettene pas94 og pas95. Alle variablene selekteres (pas94.*, pas95.*) fra de to datasettene, og tabellen skal inneholde linjer der inr er like i de to datasettene. Vi må passe på at vi ikke har andre variable med samme navn i de to datasettene enn koblingsvariablene. Resultatet blir slik:

OBS	INR	INN194	INN195
1	1001	100	700
2	1001	100	800
3	1001	100	900
4	2002	100	700
5	2002	200	700
6	2002	300	700
7	3003	100	700
8	3003	100	800
9	3003	100	900
10	3003	200	700
11	3003	200	800
12	3003	200	900
13	3003	300	700
14	3003	300	800
15	3003	300	900
16	4004	100	700
17	4004	100	800
18	4004	100	900
19	4004	200	700
20	4004	200	800
21	4004	200	900
22	5005	100	700
23	5005	100	800
24	5005	200	700
25	5005	200	800
26	5005	300	700
27	5005	300	800

Vi ser at alle like inr er joinet i de to datasettene. Det eneste som mangler er de som bare finnes i et av datasettene. For å få disse med på utdatasettet må vi ta med en vanlig MERGE der vi henter de observasjonene som ikke kobler. Til slutt fletter vi de to datasettene sammen etter *inr* i et DATA-steg.

```
DATA pas9495b;
  MERGE pas94 (IN=p94)
        pas95 (IN=p95)
  ;
  BY inr ;
  IF NOT(p94 AND p95) ;

DATA pas9495;
  SET pas9495a pas9495b;
  BY inr;
```

Utskrift av datasettet:

OBS	INR	INNL94	INNL95
1	1001	100	700
2	1001	100	800
3	1001	100	900
4	2002	100	700
5	2002	200	700
6	2002	300	700
7	3003	100	700
8	3003	100	800
9	3003	100	900
10	3003	200	700
11	3003	200	800
12	3003	200	900
13	3003	300	700
14	3003	300	800
15	3003	300	900
16	4004	100	700
17	4004	100	800
18	4004	100	900
19	4004	200	700
20	4004	200	800
21	4004	200	900
22	5005	100	700
23	5005	100	800
24	5005	200	700
25	5005	200	800
26	5005	300	700
27	5005	300	800
28	6006	100	.
29	7007	.	700

Ønsker vi alle observasjoner fra det ene datasettet, men ikke de som bare er i det andre datasettet kan vi bruke en LEFT JOIN:

```
PROC SQL;
CREATE TABLE pas9495 AS
  SELECT pas94.*, pas95.*
  FROM pas94 LEFT JOIN pas95
  ON pas94.inr = pas95.inr
;
```

Denne gir oss samme resultat som over, bortsett fra at den siste observasjonen (29) ikke kommer med i utdatasettet. Ved å sette det andre datasettet først vil vi få den siste observasjonen av de over, men den nest siste vil ikke komme med.

```
PROC SQL;
CREATE TABLE pas9594 AS
  SELECT pas95.*, pas94.*
  FROM pas95 LEFT JOIN pas94
  ON pas95.inr = pas94.inr
;
```

2.3.4 Koble et datasett med 1 observasjon til hver observasjon på et annet datasett.

Det hender av og til at vi har et SAS datasett som inneholder én observasjon med en variabel. Denne variabelen ønsker vi å hekte på *alle* observasjonene i et annet SAS datasett.

Vi har et SAS datasett med forbruksdata på husholdningsnivå. Ved å bruke PROC MEANS kan vi finne gjennomsnittlig utgift til f.eks. matvarer. Dette tallet legges ut som én observasjon på et eget SAS datasett. Vi ønsker å sammenligne hver husholdnings matvareutgift med gjennomsnittlig utgift. For å få til dette, må vi koble gjennomsnittstallet på hver enkelt husholdningsobservasjon. Når koblingen er gjort kan vi sammenligne husholdningens forbruk med gjennomsnittet.

```
DATA hushold.forbruk;
  IF _N_ = 1 THEN SET gjsnitt;
  SET forbruk;
```

SAS datasettet *gjsnitt* har bare én observasjon og variablene hektes på og legges ut på alle observasjoner i SAS datasettet *hushold.forbruk*. Vi benytter oss her av at alle variable lest med SET-setningen beholdes (automatisk RETAIN). Datasettet *gjsnitt* leses bare én gang (IF _N_ = 1), og alle variablene på dette vil følgelig beholde sine verdier resten av DATA-steget.

2.4 Omkodning ved hjelp av kataloger

Det hender ofte at vi skal kode på grunnlag av kataloger. Når vi bruker SAS til slik omkodning, bør vi bruke utskriftsformater og PUT-funksjonen. Først lager vi et SAS utskriftsformat fra katalogen ved hjelp av et DATA-steg og en PROC FORMAT. Deretter kan vi bruke PUT-funksjonen til omkodningen i et DATA-steg.

Vi har en personfil med nåværende bokommune og alle kommuner hver person har bodd i. Vi ønsker nå å finne hver kommunes sentralitet. Sentraliteten ligger på en katalog. Vi må starte med å lage et SAS datasett fra katalogen. Dette SAS datasett bruker vi så til å lage et SAS utskriftsformat. Vi kaller kommunenummer for *start* (startverdi for utskriftsformatet) og sentralitet for *label* (omkodningen, teksten). Dette er navnene SAS bruker for å kunne lage utskriftsformatet fra et SAS datasett. Videre må vi ha med en variabel kalt *fmtname* som skal inneholde navnet på utskriftsformatet. Foruten de data som ligger i katalogen, bør vi legge inn kode og tekst for variable med MISSING-verdier og for ugyldige verdier. Dette gjør omkodningen sikrere.

```
DATA kommkat ;
  INFILE '$FELLES/ssbkat/kommune/g1994' MISSOVER PAD END=slutt;
  INPUT @01 start $CHAR4.
        @83 label $CHAR2.
        ;
  fmtname = '$sentr';
  OUTPUT;
  IF slutt THEN
    DO;
      start = ' ';
      label = ' ';
      OUTPUT;
      label = '9 ';
      hlo = 'O' ;
      OUTPUT;
    END;
  PROC FORMAT CNTLIN=kommkat;
```

Vi har nå laget utskriftsformatet *\$sentr*. Dette inneholder kommunenummer som koder og sentralitet som tekst. Videre har vi bestemt at MISSING-verdier får blank tekst og ugyldige kommunenummer (dvs. alle andre verdier, gitt ved at *hlo*-variabelen er satt til bokstaven O) får teksten 9. Nærmere beskrivelse av hvordan vi lager utskriftsformater fra SAS datasett finner vi på side 133. Nå kan vi bruke utskriftsformatet til omkodning:

```
DATA persreg ;
  INPUT @01 komm $CHAR4.
        @06 bokom_1 $CHAR4.
        @11 bokom_2 $CHAR4.
        @16 bokom_3 $CHAR4.
        @21 bokom_4 $CHAR4.
        ;
  sentr = PUT(komm,$sentr.);
  sentr_1 = PUT(bokom_1,$sentr.);
  sentr_2 = PUT(bokom_2,$sentr.);
  sentr_3 = PUT(bokom_3,$sentr.);
  sentr_4 = PUT(bokom_4,$sentr.);
  CARDS;
  0101 0138 1138 1252 1151
  0101 0121 1121 1151 1201
  0101 0122 1122
  0101 0211 1211
  0101 0101 1101
  0101 0104 1104 1103
  0101 0105 1105
  0101 0106 1106 2001 1934
  0101 0117 1117
  0101 2030 1030 0931 0920
  0101 0301 1201
  0101 0301 1201
  0102 0402 0403 1812 0301
  0102 0402 0403
  0102 0301 1443
  0211 0213 0236 0101 0706
  0211 0214 0517 0301 0803
  0211 0215 0412 1252 1201
  0213 0212 0225 2018 1601
  0213 0211 0226
  0214
  1601 1612 1633
  ;
```

```

PROC PRINT DATA=persreg ;
  VAR komm sentr bokom_1 -bokom_4 sentr_1 - sentr_4;
  TITLE 'Omkodet til sentralitet';
RUN;

```

Vi lager en sentralitetsvariabel for hver kommune personene har bodd. Ved å bruke PUT-funksjonen henter vi sentralitet fra utskriftsformatet \$*sentr* med kommunenummer som basis. Med dette omkodingsprogrammet slipper vi med ett gjennomløp av det store personregisteret. Vi legger merke til at manglende kommunenummer har fått blank som sentralitet, mens ugyldige kommunenummer har fått sentralitet 9.

```

Omkodet til sentralitet

OBS   KOMM  SENTR  BOKOM_1  BOKOM_2  BOKOM_3  BOKOM_4  SENTR_1  SENTR_2  SENTR_3  SENTR_4
1     0101  2A     0138     1138     1252     1151     3A       9        0B       0B
2     0101  2A     0121     1121     1151     1201     0A       3A       0B       3A
3     0101  2A     0122     1122             3A       3A
4     0101  2A     0211     1211             3A       0B
5     0101  2A     0101     1101             2A       1A
6     0101  2A     0104     1104     1103             3A       9        3A
7     0101  2A     0105     1105             2A       9
8     0101  2A     0106     1106     2001     1934     2A       2A       9        9
9     0101  2A     0117     1117             9
10    0101  2A     2030     1030     0931     0920     0B       9        9        9
11    0101  2A     0301     1201             3A       3A
12    0101  2A     0301     1201             3A       3A
13    0102  9      0402     0403     1812     0301     1A       2A       1B       3A
14    0102  9      0402     0403             1A       2A
15    0102  9      0301     1443             3A       0B
16    0211  3A     0213     0236     0101     0706     3A       3A       2A       2A

```

Har vi mange variable som skal omkodes på samme måte, kan vi bruke ARRAY-setningen. Resultatet blir det samme som over, men programmet er litt mer elegant og lettere å utvide.

```

DATA persreg (DROP=i);
INPUT @01 komm $CHAR4.
      @06 bokom_1 $CHAR4.
      @11 bokom_2 $CHAR4.
      @16 bokom_3 $CHAR4.
      @21 bokom_4 $CHAR4.
      ;

ARRAY kom (*) $ komm bokom_1 - bokom_4;
ARRAY sent (*) $ sentr sentr_1 - sentr_4;
DO i = 1 to DIM(kom);
  sent(i) = PUT(kom(i), $sentr.);
END;

CARDS;
0101 0138 1138 1252 1151
0101 0121 1121 1151 1201
0101 0122 1122
0101 0211 1211
0101 0101 1101
0101 0104 1104 1103
0101 0105 1105
0101 0106 1106 2001 1934
0101 0117 1117
0101 2030 1030 0931 0920
0101 0301 1201
0101 0301 1201
0102 0402 0403 1812 0301
0102 0402 0403
0102 0301 1443
0211 0213 0236 0101 0706
;

PROC PRINT DATA=persreg ;
  VAR komm sentr bokom_1 - bokom_4 sentr_1 - sentr_4;
  TITLE 'Omkodet til sentralitet';

```

3. Referansedel

I denne delen går vi igjennom mange av de viktige delene av SAS systemet. Vi ser nærmere på:

- interaktiv SAS og hvilke kommandoer vi bruker der
- SAS editor med kommandoene vi bruker når vi skriver programmene våre
- SAS-uttrykk, hva det er, hva de inneholder og hvordan de brukes
- SAS dataformater for innlesing og utskrift
- Automatiske hjelpevariable, hva det er og når og hvordan de brukes
- Mange SAS system options, når og hvordan de brukes
- Hva SAS datasett options er og brukes til
- De fleste SAS setningene som brukes i DATA-steget
- Mange nyttige SAS prosedyrer
- Mange nyttige SAS funksjoner

Selv om denne delen er omfattende og bygd opp som en manual, er den ikke uttømmende. Vi håper allikevel at den vil dekke vanlige og mer avanserte programmereres behov.

3.1 SAS-navn

Når vi programmerer i SAS må vi følge reglene som er satt for navnsetting av SAS datasett, variable, formater, array'er, makrovariable, librefs og filrefs. I tillegg har SAS egne navn på setninger, prosedyrer, parametre, automatiske hjelpevariable osv. Disse navnene som SAS selv bruker, bør vi ikke bruke i vår navnsetting, både fordi det kan medføre problemer for eksekveringen av SAS-programmet og fordi det gjør programmene vanskelige å lese.

Navnene vi bruker på variable, SAS datasett osv. har maksimal lengde på 32 tegn. De skal vanligvis begynne med en bokstav (A-Z) eller en understrek (), de skal ikke inneholde blanke eller andre spesialtegn enn understrek. Vi kan bruke små og store bokstaver (vanligvis ikke æøå) om hverandre uten at SAS skiller mellom store og små bokstaver. Det betyr at en variabel kalt Fylke senere kan benevnes fylke. For SAS er dette samme variabel.

Vi kan overstyre reglene for hvilke tegn det er lov å bruke i variabelnavn. Da må system option VALIDVARNAME=ANY settes. Videre må slike navn skrives i fnutter med en N bak siste fnutt, f.eks. 'Kjønn'N.

SAS formater har en maksimal lengde på 8 tegn og de kan ikke slutte med et tall.

3.2 SAS kommandoer

SAS Display Manager er en interaktiv, vindusbasert måte å jobbe med SAS på. Den består av mange forskjellige vinduer og tilhørende SAS kommandoer. Når vi starter SAS, får vi automatisk opp programeditor-vinduet, log-vinduet og output-vinduet. Andre vinduer kaller vi opp ved behov. Vi må passe på så vi ikke blander sammen SAS kommandoer med SAS-setninger i DATA- og PROC-stegene.

For å gjøre det enklere og raskere å arbeide i en SAS-sesjon, bør vi lære oss å bruke funksjonstaster. Funksjonstastene brukes for å utføre SAS Display Manager kommandoer. Vi har et standardoppsett for de viktigste kommandoene (oppsettet kan dessverre variere noe mellom de forskjellige plattformene):

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
CLEAR	KEYS	LOG	OUTPUT	END	PGM	ZOOM	RFIND	RECALL	SUB	PMENU	RSUB

Foruten disse er det greit å ha RIGHT, LEFT, TOP, BOT, UNDO, MARK, STORE og PASTE på funksjonstaster. Hvilke funksjonstaster som kan brukes, varierer noe fra plattform til plattform.

Vi kan utføre flere kommandoer vha. én funksjonstast. Ønsker vi det, må vi skille kommandoene med semikolon. F.eks. kan vi med kommandoene under markere all tekst i vinduet og kopiere den til utklippstavlen:

```
unmark all;curpos 1 1;mark;curpos max max;mark;store
```

For å sende det som ligger i utklippstavlen til eksekvering, kan vi bruke disse kommandoene:

```
pgm;submit buf=default
```

Hvis vi ønsker å hente tilbake et program som vi eksekverte tidligere i SAS-sesjonen bruker vi RECALL-kommandoen. Ønsker vi ikke å hente tilbake det siste programmet vi kjørte, men et tidligere, bruker vi kombinasjonen RECALL og CLEAR helt til vi får fram igjen det programmet vi ønsker. Dette kan vi forenkle ved å gi kommandoen:

```
clear; recall; ?
```

Da trenger vi bare å trykke *enter* for hver gang vi leter etter et tidligere kjørt program.

3.2.1 Viktige kommandoer under programmering

BYE	Avslutter SAS
CLEAR	Tømmer innholdet i et vindu
END	Lukker vinduet. Virker som SUBMIT når den utføres i programeditoren.
FILE	Lagrer innholdet i et vindu på en ekstern fil. Brukes helst til å lagre program, SASlog og resultater i utskriftsvinduet
HELP	Kaller opp interaktivt hjelpesystem. HELP med navn på prosedyre, SAS-setning e.l. som parameter vil ofte bringe oss direkte til det emne vi ønsker hjelp om. Bruk heller hjelpesystemet som ligger på Byrånettet, under IT-info ==> SAS ==> SAS OnlineDoc, versjon 8.
INCLUDE	Henter en ekstern fil inn til et vindu. Brukes vanligvis for å hente program inn til programeditoren
NUM	Nummerer linjene i programeditoren (NUM ON) og enkelte andre vinduer
RECALL	Henter tilbake eksekverte program. For hver RECALL kommando hentes et tidligere program (så lenge SUBMIT kommandoen ikke utføres, etter SUBMIT vil RECALL hente tilbake det sist eksekverte programmet). For hver RECALL kommando hentes et tidligere eksekvert program tilbake, så lenge det finnes noen.
RSUBMIT	Sender SAS-setninger til eksekvering på tjenermaskin.
SIGNOFF	Brukes i Windows SAS til å avslutte forbindelse til Unix.
SIGNON	Brukes i Windows SAS til å knytte forbindelse til Unix (f.eks. signon ursus).
SUBMIT	Sender SAS-setninger til eksekvering.
UNDO	Ange siste redigering i programeditoren. Gjentatt angring utføres med ny UNDO
X	Går midlertidig ut av SAS (Windows). For å komme tilbake til SAS, bruker vi kommandoen EXIT. Kan også brukes til å utføre en operativsystem-kommando, denne gis da som parameter til X-kommandoen
ZOOM	Zoomer et vindu opp eller ned
?	Henter tilbake forrige kommando

3.2.2 Kommandoer for kopiering, klipping og liming

For å kunne bruke disse effektivt, må de legges på funksjonstaster. I grafiske grensesnitt kan vi markere et område ved å bruke mus. Der finnes det også egne knapper for kopiering, klipping og liming.

MARK	Indikerer at et området skal markeres. MARK-kommandoen gis ved starten og slutten av området som skal markeres. Med parameteren BLOCK kan vi markere en blokk med tekst.
STORE	Kopierer markert tekst til utklippstavlen
CUT	Klipper ut markert tekst og legge den i utklippstavlen
PASTE	Limer inn tekst fra utklippstavlen
PCLEAR	Tømmer utklippstavlen (Unix)
UNMARK	Fjerner markering for et markert område

3.2.3 Kommandoer for søk/erstatt

Med søk og erstatt kan vi endre tekst etter spesielle kriterier. Syntaksen er slik:

CHANGE 'fra' 'til' <FIRST | LAST | NEXT | PREV | ALL > <PREFIX | SUFFIX | WORD>

FIRST	Første forekomst i teksten (uansett hvor markøren befinner seg)
LAST	Siste forekomst i teksten (uansett hvor markøren befinner seg)
NEXT	Neste forekomst i teksten
PREV	Forrige forekomst i teksten
ALL	Alle forekomster. Ved FIND gis det melding om antall forekomster, mens med CHANGE endres alle forekomster
PREFIX	Teksten må være en forstavelse til et ord for at søket skal slå til
SUFFIX	Teksten må være et suffiks til et ord for at søket skal slå til
WORD	Teksten må være et helt ord for at søket skal slå til. Æ, ø og å regnes som del av ord

BFIND	Søk tilbake i teksten
CHANGE	Søk og erstatt
DLGFIN	Åpner dialogboks for søk
DLGREPL	Åpner dialogboks for søk og erstatt
FIND	Søk
RCHANGE	Gjenta siste søk og erstatt
RFIND	Gjenta siste søk

3.2.4 Kommandoer som kaller opp SAS vinduer

ACCESS	Aksessvinduet. Lister alle tilgjengelige SAS-datasett for aksess.
DIR	Viser innholdet i et SAS library. Kalles vanligvis fra LIBNAME-vinduet
FILENAME	Viser hvilke eksterne filer som er definert (med FILENAME-setninger)
FOOTNOTE	Fotnotevinduet. Viser hvilke gjeldende fotnoter. Disse kan endres her
FSLIST	Viser opp en ekstern fil, filens navn må oppgis i fnutter.
FSVIEW	Se på eller endre et SAS datasett i FSVIEW-vinduet. Datasettets navn må oppgis, ellers vises det sist brukte datasettet.
KEYS	Oversikt over funksjonstastene. Disse kan endres her
LIB	Viser hvilke SAS librefs som er tilknyttet
LOG	Viser SASlog med informasjon om eksekveringen av programmene
MANAGER	Output Manager. Oversikt over resultater fra programmene.
OPTIONS	Oversikt over SAS system options. Disse kan endres her
OUT	Utskriftsvinduet. Viser resultatene fra programmene.
PGM	Går til programeditoren
TITLE	Tittelvinduet. Viser gjeldende overskrifter. Disse kan endres her
VAR	Viser hvilke variable som finnes i et SAS datasett. Kalles vanligvis fra DIR-vinduet
VIEWTABLE	Se på eller endre et SAS datasett i VIEWTABLE-vinduet. Datasettets navn må oppgis, ellers åpnes et tomt datasett.
WEDIT	Åpner et nytt editorvindu med den nye editor (windows). Kan hente en programfil til vinduet ved å føye til navnet i fnutter.

3.2.5 Kommandoer for navigering

BOT	Går til slutten av teksten
DOWN	Blar nedover i teksten. Parametre: n, MAX, HALF, PAGE, CURSOR
HOME	Flytter markøren til kommandolinja eller til første input-felt i et vindu
HSCROLL	Angir hva som skal være standard horisontal blaing; n, MAX, HALF, PAGE, CURSOR. Virker for kommandoene LEFT og RIGHT
LEFT	Blar til venstre i teksten. Parametre: n, MAX, HALF, PAGE, CURSOR
n	Går til linje nummer n i teksten
NEXT	Går til det neste vinduet
PREVWIND	Går til det forrige vinduet
RIGHT	Blar til høyre i teksten. Parametre: n, MAX, HALF, PAGE, CURSOR
TOP	Går til begynnelsen av teksten
UP	Blar oppover i teksten. Parametre: n, MAX, HALF, PAGE, CURSOR
VSCROLL	Angir hva som skal være standard vertikal blaing; n, MAX, HALF, PAGE, CURSOR. Virker for kommandoene DOWN og UP (PageUp- og PageDown-tastene påvirkes også av VSCROLL-kommandoen, dog ikke under Windows SAS). PAGE er standard, men CURSOR anbefales.

3.2.6 Andre aktuelle kommandoer

CANCEL	Lukker et vindu uten å utføre noe der
CASCADE	Setter alle åpnete vinduer etter hverandre (som arkivkort) i skjermbildet (tilbake igjen med UNCASCADE)
COMMAND	Bytter mellom rullgardinmenyer og kommandolinje for vinduet
CURPOS	Flytter markøren til linje x, kolonne y i vinduet (CURPOS x y). Første posisjon i teksten er CURPOS 1 1, mens siste er CURPOS MAX MAX.
DLGOPEN	Åpner dialogboks for å åpne filer.
DLGPRT	Åpner dialogboks for utskrift.
DLGSAVE	Åpner dialogboks for å lagre filer.
ICON	Ikoniserer et vindu
LINESIZE	Setter eller viser linjestørrelsen til SASlog eller utskriftsvinduet
PAGESIZE	Setter eller viser sidestørrelsen til utskriftsvinduet
PMENU	Bytter mellom kommandolinje og rullgardinmenyer for alle vinduer
RESHOW	Viser et vindu på nytt
RESIZE	Endrer tilbake til standard vindusstørrelse
SUBTOP	Sender de n første linjer til eksekvering. Utelates n, sendes den første linja
TILE	Viser åpnete vinduer i et firkantet mosaikkmønster (tilbake igjen med UNTILE)
WPOPUP	Kaller opp en tilpasset valgmeny (samme som å trykke høyre musetast). Virker i Windows SAS

3.3 SAS editor

Det finnes nå 2 editorer som vi kan bruke internt i SAS, enhanced editor og program editor.

3.3.1 Enhanced editor

Dette er SAS nye forbedrede editor, som foreløpig bare er tilgjengelig under windows. Med den fjernes siste rest fra stormaskineditoren som program-editoren baserer seg på. Enhanced editor ligner mer på windowsbaserte editorer, selv om den ikke er så veldig avansert. Editoren støtter syntaksfarging, som gjør det enklere å lese programmet og også å se syntaksfeil før programmet kjøres. Ellers fungerer editoren relativt likt andre windows-baserte editorer.

3.3.2 Program editor

Program editoren er SAS sin opprinnelige editor. Den er en ganske primitiv editor. For å gjøre en del vanlige operasjoner i denne editoren må vi bruke linjekommandoer. Disse linjekommandoene skriver vi hvor som helst innenfor nummerfeltet til venstre i editoren. Hvis linjene ikke er nummerert, gjøres dette med kommandoen NUM ON i kommandofeltet. Mange av linjekommandoene er slik at vi kan utføre dem for en linje, et antall linjer eller en blokk av linjer.

Noen eksempler:

```
R      Repeterer en linje
R4     Repeterer fire linjer

RR
RR     Repeterer alle linjer fra den første RR til den siste RR

RR2
RR     Repeterer alle linjer fra den første RR til den siste RR to ganger
```

3.3.2.1 Linjekommandoene

```
A      After. Angir at tekst skal plasseres etter linje merket med A
B      Before. Angir at tekst skal plasseres før linje merket med B
C      Kopier linje. Må angi plassering med kommando A, B eller O på en annen linje
CL     Gjør alle bokstaver på linja små. Virker også for ÆØÅ. Blokkkommando CCL.
COLS   Viser en linjal med posisjoner
CU     Gjør alle bokstaver på linja STORE. Virker også for æøå. Blokkkommando CCU.
D      Slett linje
I      Sett inn linje etter denne linja
IB     Sett inn linje før denne linja
JC     Sentrer tekst på linje. Blokkkommando JJC
JL     Venstrejuster tekst på linje. Blokkkommando JJJ
JR     Høyrejuster tekst på linje. Blokkkommando JJJ
M      Flytt linje
R      Repeter linje
TC     Slå sammen denne linje med neste
TS     Del linje der markøren står når kommandoen utføres
(n)    Flytt tekst n posisjoner til venstre. Tegn slettes hvis de forsøkes flyttet lenger til venstre enn første posisjon på linja
)n     Flytt tekst n posisjoner til høyre. Tegn slettes hvis de forsøkes flyttet lenger til høyre enn siste posisjon på linja
<n     Flytt tekst n posisjoner til venstre. Tegn flyttes ikke lenger enn til første posisjon på linja
>n     Flytt tekst n posisjoner til høyre. Tegn flyttes ikke lenger enn til siste posisjon på linja
```

3.4 SAS-uttrykk

Et SAS-uttrykk kan bestå av variable, konstanter, tekst (i fnutter), SAS funksjoner, SAS-datoverdier, SAS tidsverdier, parenteser og operatører. Operatørene er disse:

Aritmetiske operatører	+	addisjon
	-	subtraksjon
	*	multiplikasjon
	/	divisjon
	**	potensering
Logiske operatører	AND	og, kan også skrives &
	OR	eller, kan også skrives eller !
	NOT	ikke, kan også skrives ^ eller ~
Sammenlignende operatører	=	lik
	^=	ulik, kan også skrives NE
	<	mindre enn, kan også skrives LT
	>	større enn, kan også skrives GT
	<=	mindre enn eller lik, kan også skrives LE
	>=	større enn eller lik, kan også skrives GE
	IN	lik en i en liste (i parentes og verdier skilt med komma)
		sammenheking (konkatenering) av karakterverdier
Andre operatører	><	returnerer den minste av de sammenlignede verdier
	<>	returnerer den største av de sammenlignede verdier

Rekkefølgen de forskjellige operatørene utføres i er hierarkisk delt i 7 nivåer.

1	** + - NOT >< <>
2	* /
3	+ -
4	
5	< > <= >= ^= IN
6	AND
7	OR

+ og - er som prefikser i nivå 1 og som addisjons/subtraksjonsoperator i nivå 3. Beregninger utføres fra høyre mot venstre for operatører innen nivå 1 og fra venstre mot høyre for de andre nivåene. Dette betyr at beregninger utføres først for nivå 1, deretter nivå 2 osv. Uttrykk i parenteser utføres alltid før parentesen løses opp. Det vil ofte være lurt å bruke parenteser for å lage riktig SAS-uttrykk.

Et spesielt uttrykk er f.eks. '01' <= fylke <= '09'. Dette følger ikke de vanlige reglene, men utføres som ('01' <=fylke) AND (fylke <= '09') og slår til for fylkene 01-09.

Når vi skal liste flere verdier som ikke kommer etter hverandre kan vi bruk IN-operatoren. Her ramser vi opp de verdiene vi skal sammenligne med:

```
IF fylke IN('01','02','03','06','07');
```

SAS-uttrykk brukes i tester som IF, DO, SELECT og WHERE og til tilordning med assignment- og sumsetningene. De logiske og de sammenlignende operatørene kan bare brukes i tester, ikke ved tilordning.

De logiske operatorene binder flere betingelser sammen. Når vi bruker OR og AND i samme SAS-setning, er det viktig å **bruke parenteser** for å sikre at betingelsene utføres slik vi ønsker. Dette eksempelet viser hvor forskjellig resultatet kan bli avhengig av parenteser og rekkefølge:

```
DATA preg ;
  INPUT @01 inntekt 7.
        @08 skatt 7.
        @15 gjeld 7.
  ;
  IF inntekt > 100000 AND gjeld = 0 OR 20000 < skatt < 200000 THEN a=1;
  IF (inntekt > 100000 AND gjeld = 0) OR 20000 < skatt < 200000 THEN b=1;
  IF inntekt > 100000 AND (gjeld = 0 OR 20000 < skatt < 200000) THEN c=1;
  IF inntekt > 100000 OR 20000 < skatt < 200000 AND gjeld = 0 THEN d=1;
  IF inntekt > 100000 OR (20000 < skatt < 200000 AND gjeld = 0) THEN e=1;
  IF (inntekt > 100000 OR 20000 < skatt < 200000) AND gjeld = 0 THEN f=1;
CARDS;
0193812008934200000000
0197677001902000000000
0045354020923400000000
0071020002902000000000
015672000690410004525
036729800011010023340
004426100345990123491
006744401978760002130
;

PROC PRINT DATA=preg;
  TITLE 'Resultat av IF-tester med OR og AND';
RUN;
```

Slik blir utskriften:

Resultat av IF-tester med OR og AND

OBS	INNTEKT	SKATT	GJELD	A	B	C	D	E	F
1	193812	89342	0	1	1	1	1	1	1
2	197677	19020	0	1	1	1	1	1	1
3	45354	209234	0	.	.	.	.	.	.
4	71020	29020	0	1	1	.	1	1	1
5	156720	69041	4525	1	1	1	1	1	.
6	367298	1101	23340	.	.	.	1	1	.
7	44261	34599	123491	1	1	.	.	.	.
8	67444	197876	2130	1	1	.	.	.	.

3.5 SAS dataformater

Å formatere data vil si å vise eller konvertere data etter bestemte mønstre. Ønsker vi å vise et tall med f. eks 8 posisjoner hvorav 1 desimal, kan vi bruke utskriftsformatet 8.1 til å fortelle SAS dette. Når eksterne datafiler er lagret formatert, betyr det at de er lagret etter et bestemt mønster. Skal SAS tolke disse dataene riktig, må vi fortelle SAS hvilket mønster som er brukt. Det gjør vi ved å bruke et innlesingsformat.

Vi har altså to hovedtyper av dataformater: innlesingsformater og utskriftsformater. Hvert av disse deles igjen i:

- Karakter
- Numerisk
- Dato og tid

Mange dataformater finnes både som innlesingsformater og utskriftsformater. Innlesingsformater bruker vi kun når vi skal lese data inn til SAS når disse dataene er formatert etter spesielle mønstre og vi ønsker å konvertere og lagre disse dataene etter et annet mønster. Utskriftsformater bruker vi for å vise dataene etter spesielle mønstre. Disse lagres ikke på datasettet, men det knyttes en fast eller midlertidig forbindelse til utskriftsformatet.

3.5.1 Innlesingsformater

Her er noen utvalgte innlesingsformater, w er valgt bredde og d er antall desimaler. Hvis d utelates, leses tallet uten desimaler. De vanligste innlesingsformatene er uthevet:

For karaktervariable:

\$w.	Leser karakterer (ikke ledende blanke)
\$ASCIIw.	Konverterer ASCII tegnsett til lokalt tegnsett
\$BINARYw.	Konverterer binære verdier til karakterer
\$CHARw.	Leser karakterer (også ledende blanke)
\$EBCDICw.	Konverterer EDCDIC tegnsett til lokalt tegnsett
\$HEXw.	Konverterer heksadesimale verdier til karakterer
\$QUOTEw.	Fjerner fnutter (enkle og doble) rundt innlest tekst
\$UPCASEw.	Gjør alle bokstaver om til store (også æ, ø og å)
\$VARYINGw.	Leser karakterfelt med variabel lengde

For numeriske variable:

w.d	Leser vanlige tall
BINARYw.	Konverterer tall fra binær kode.
BZw.d	Konverterer blanke til nuller
COMMAw.d	Bruker komma som tusenskilte, punktum som desimaltall og parentes for negative tall
COMMAXw.d	Bruker punktum som tusenskilte, komma som desimaltall og parentes for negative tall
Ew.d	Viser tall oppgitt i 10-potens ($10E3 = 1000$)
HEXw.	Konverterer tall til heksadesimalt format
NUMXw.d	Leser tall med komma som desimaltall
OCTALw.	Konverterer tall fra åttetallsystemet
PDw.d	Konverterer tall fra pakket desimal (vanlig lagringsformat på stormaskin)
ZDw.d	Konverterer tall fra zoned desimal (vanlig lagringsformat på stormaskin)
ZDBw.d	Konverterer tall fra zoned desimal tall med ledende blanke

For dato og tid:

DATEw.	Leser SAS dato variabel etter mønster DDMMMÅÅ
DATETIMEw.d	Leser SAS dato og tid variabel etter mønster DDMMMÅÅ:TT:MM:SS.SS
DDMMYYw.	Leser SAS dato variabel etter mønster DDMMÅÅ
JULIANw.	Leser SAS dato variabel etter mønster ÅÅDDD el. ÅÅÅÅDDD
MMDDYYw.	Leser SAS dato variabel etter mønster MMDDÅÅ
MONYYw.	Leser SAS dato variabel etter mønster MMMÅÅ (dag i måned settes til 1)
TIMEw.	Leser SAS tid variabel etter mønster TT:MM:SS
YYMMDDw.	Leser datoer etter mønster ÅÅMMDD

For dato og tid med norsk stavemåte (minste bredde, største bredde og standardbredde varierer):

NORDFDEw. Tilsvare DATEw.
 NORDFDTw.d Tilsvare DATETIMEw.d
 NORDFDMYw. Tilsvare MONYYw.

Tilsvarende kan vi gjøre med andre språk, forstavelen NOR byttes med det andre språkets: dansk (DAN), finsk (FIN), fransk (FRA), italiensk (ITA), portugisisk (PTG), spansk (ESP), svensk (SVE), sveitsisk fransk (FRS), sveitsisk tysk (DES) og tysk (DEU).

3.5.2 Utskriftsformater

Her er noen utvalgte utskriftsformater, w er valgt bredde og d er antall desimaler. Hvis d utelates, vises tallet uten desimaler. De vanligste utskriftsformatene er uthevet:

For karaktervariable:

\$w.	Skriver karakterer uten å fjerne ledende blanke
\$ASCIIw.	Konverterer lokalt tegnssett til ASCII
\$BINARYw.	Konverterer karakterer til binære verdier
\$CHARw.	Skriver karakterer uten å fjerne ledende blanke. Er identisk med \$w.
\$EBCDICw.	Konverterer lokalt tegnssett til EBCDIC
\$HEXw.	Konverterer karakterer til heksadesimale verdier
\$REVERJw.	Skriver teksten ut baklengs uten justering
\$REVERSw.	Skriver teksten ut baklengs og venstrejustert
\$UPCASEw.	Viser alle bokstaver som store (også æ, ø og å)
\$VARYINGw.	Skriver karakterfelt med variabel lengde

For numeriske variable:

w.d	Skriver vanlige tall
BESTw.	Skriver ut tall etter beste mønster ut fra tallets størrelse og avsatt plass (desimaler skrives ikke ut). BESTw. er standardformatet SAS bruker.
BINARYw.	Konverterer tall til binær kode.
BZw.d	Skriver tall uten ledende nuller
COMMAw.d	Skriver tall med komma som tusenskilte, punktum som desimaltall og parenteser for negative tall. Vi kan velge enten 0 eller 2 desimaler.
COMMAXw.d	Skriver tall med punktum som tusenskilte, komma som desimaltall og parenteser for negative tall. Vi kan velge enten 0 eller 2 desimaler.
Ew.d	Skriver tall med 10-potens ($1000 = 10E3$)
FRACTw.	Konverterer desimaltall til brøker.
HEXw.	Skriver tall som heksadesimale verdier
NEGPARENw.d	Viser negative tall med parenteser rundt istedenfor med minustegn
NUMXw.d	Lager tall med komma som desimaltall
OCTALw.	Konverterer tall til åttetallsystemet
PDw.d	Skriver tall som pakket desimal tall
PERCENTw.	Skriver tall som prosenter. Negative tall vises med parentes rundt istedenfor minustegn. Prosentene skal oppgis som andelstall. 0.1 blir representert som 10%.
ROMANw.	Skriver tall som romertall
Zw.d	Skriver tall med ledende nuller
ZDw.d	Skriver tall som zoned desimal tall

For dato og tid:

DATEw.	Skriver SAS dato variabel etter mønster DDMMMAÅ
DATETIMEw.d	Skriver SAS dato og tid variabel etter mønster DDMMMAÅ:TT:MM:SS.SS
DAYw.	Skriver dagen i måneden i en SAS dato variabel
DDMMYYw.	Skriver SAS dato variabel etter mønster DDMMÅÅ med skilletegn /
DDMMYYNw.	Skriver SAS dato variabel etter mønster DDMMÅÅ uten skilletegn
DOWNAMEw.	Skriver SAS dato variabel som ukedag på engelsk
HHMMw.d	Skriver timer og minutter fra en SAS tid variabel
JULDAYw.	Skriver dagnr i et år fra en SAS dato variabel
JULIANw.	Skriver SAS dato variabel etter mønster ÅÅDDD el. ÅÅÅÅDDD
MMDDYYw.	Skriver SAS dato variabel etter mønster MMDDÅÅ med skilletegn /
MMDDYYNw.	Skriver SAS dato variabel etter mønster MMDDÅÅ uten skilletegn
MMYYw.	Skriver måned og år fra en SAS dato variabel skilt med M
MMYYDw.	Skriver måned og år fra en SAS dato variabel skilt med -
MMYYNw.	Skriver måned og år fra en SAS dato variabel uten skilletegn
MMYYSw.	Skriver måned og år fra en SAS dato variabel skilt med /
MONNAMEw.	Skriver SAS dato variabel som måned på engelsk
MONYYw.	Skriver SAS dato variabel etter mønster MMMÅÅ
QTRw.	Skriver SAS dato variabel som kvartalsnr
QTRRw.	Skriver SAS dato variabel som kvartalsnr med romertall
TIMEw.	Skriver SAS tid variabel etter mønster TT:MM:SS
WEEKDATEw.	Skriver SAS dato etter mønster <i>ukedag, mnd dagnr, år</i> på engelsk
WEEKDATXw.	Skriver SAS dato etter mønster <i>ukedag, dagnr måned år</i> på engelsk
WEEKDAYw.	Skriver SAS dato som et tall (1 er søndag, 2 er mandag osv.)
WORDDATXw.	Skriver SAS dato etter mønster <i>dagnr måned år</i> på engelsk
YEARw.	Skriver årstallet fra en SAS dato variabel
YYMMw.	Skriver år og måned fra en SAS dato variabel skilt med M
YYMMDw.	Skriver år og måned fra en SAS dato variabel skilt med -
YYMMNw.	Skriver år og måned fra en SAS dato variabel uten skilletegn
YYMMSw.	Skriver år og måned fra en SAS dato variabel skilt med /
YYMMDDw.	Skriver SAS dato variabel etter mønster ÅÅMMDD med skilletegn /
YYMMDDNw.	Skriver SAS dato variabel etter mønster ÅÅMMDD uten skilletegn
YYMONw.	Skriver SAS dato variabel etter mønster ÅÅMMM
YYQw.	Skriver år og kvartal fra en SAS dato variabel skilt med Q
YYQDw.	Skriver år og kvartal fra en SAS dato variabel skilt med -
YYQNw.	Skriver år og kvartal fra en SAS dato variabel uten skilletegn
YYQSw.	Skriver år og kvartal fra en SAS dato variabel skilt med /

For dato og tid med norsk stavemåte (minste bredde, største bredde og standardbredde varierer):

NORDFDEw.	Tilsvare DATEw.
NORDFDTw.d	Tilsvare DATETIMEw.d
NORDFDDw.	Tilsvare DDMMYYw.
NORDFDWNw.	Tilsvare DOWNAMEw.
NORDFMNw.	Tilsvare MONNAMEw.
NORDFMYw.	Tilsvare MONYYw.
NORDFDNw.	Tilsvare WEEKDAYw., men dag 1 er mandag, ikke søndag
NORDFWKXw.	Tilsvare WEEKDATXw.
NORDFWDXw.	Tilsvare WORDDATXw.

Tilsvarende kan vi gjøre med andre språk, forstavelsen NOR byttes med det andre språkets: dansk (DAN), finsk (FIN), fransk (FRA), italiensk (ITA), portugisisk (PTG), spansk (ESP), svensk (SVE), sveitsisk fransk (FRS), sveitsisk tysk (DES) og tysk (DEU). Forstavelsen EUR kan brukes istedenfor landforkortelsen når SAS System option DFLANG er satt til en av nasjonalitetene.

3.6 Automatiske hjelpevariable

Når et DATA-steg gjennomføres, lager SAS noen automatiske hjelpevariable. SAS bruker dem når programmet eksekveres, de lagres ikke på noe SAS datasett. Vi kan også benytte oss av dem i programmeringen.

<code>_N_</code>	Inneholder en teller som økes for hver iterasjon i DATA-steget. <code>_N_</code> har startverdi 1.
<code>_ERROR_</code>	Denne settes til en når 1 feil oppdages. Feilen kan f.eks. være en innlesingsfeil, en ikke-numerisk verdi som vi prøver å lagre i et numerisk felt, forsøk på å dele på null eller at tall er for store til å bli lagret nøyaktig (flytende desimal-feil). Vi kan bruke denne under feilbehandling. <code>_ERROR_</code> nullstilles for hver iterasjon av DATA-steget og settes til 1 når en feil oppstår. SAS skriver vanligvis feilmelding til SASlog når feil oppstår.
FIRST.by-variabel	Defineres av SAS når vi bruker BY-setningen i DATA-steget. Det lages en slik FIRST.BY-variabel for hver variabel nevnt i BY-setningen. FIRST.BY-variabel er beskrevet nærmere på side 54.
LAST.by-variabel	Defineres av SAS når vi bruker BY-setningen i DATA-steget. Det lages en slik LAST.BY-variabel for hver variabel nevnt i BY-setningen. LAST.BY-variabel er beskrevet nærmere på side 54.
#BYVAL(by-variabel)	Inneholder verdien til BY-variabelen som angis i parentes. Kan brukes til å få BY-variabelens verdier i overskriften, se eksempel under.

Eksempel 1:

Vi skal legge observasjonsnummeret på som et referansenummer på det nye SAS datasettet.

```
DATA rev.fylke01r;
  SET rev.fylke01;
  refnr = _N_ ;
  LABEL refnr = 'Referansenr';
RUN;
```

Eksempel 2:

Vi lager en tabell med PROC TABULATE der vi ønsker årstallet i overskriften. Datasettet inneholder kun én årgang og vi kan derfor bruke BY-setningen uten å sortere datasettet på forhånd.

```
PROC FORMAT ;
  VALUE hjoms
    1-23 = 'Omsorgs- tiltak'
    99   = 'Hjelpe- tiltak'
  ;

  OPTIONS NOBYLINE;
  PROC TABULATE DATA=bvern.meld95 NOSEPS MISSING FC=' -----' F=ssb6_0v.;
    BY aargang;
    CLASS antp18 antp29 flk ;
    VAR utgang nye;
    TABLE ALL='I alt' flk
      ,
      ALL='I løpet av året'*(ALL='I alt' antp18)*N=''
      utgang='Ved utgangen av året'*(all='I alt' antp29)*N=''
      /RTS=18;
    FORMAT flk $fylklur.
      antp18 antp29 hjoms.
    ;
  TITLE "Tabell 2.7 Barn med barneverntiltak. #BYVAL(aargang)";
RUN;
```

3.7 SAS system options

System options brukes for å gi standardverdier for forskjellige deler av SAS-sesjonen. Disse options har standardverdier. Noen er definert av SAS, og andre er definert i konfigurasjonsfilen *sasv8.cfg* (tidligere *config.sas*). Noen options kan endres underveis i en SAS-sesjon, andre kan bare endres i konfigurasjonsfilen eller ved oppstart. Når vi setter options ved oppstart gjøres det f.eks. slik:

```
sas -linesize 80 -pagesize 66 -nodate
```

For å endre options underveis i en SAS sesjon, bruker vi OPTIONS-setningen (se side 107) eller i noen tilfelle options-vinduet. Med OPTIONS-setningen som tilsvarer oppstartsparemetrene over gjøres det slik:

```
OPTIONS LINESIZE=80 PAGESIZE=66 NODATE ;
```

Endrede options beholder sin verdi til de blir overstyrt eller SAS sesjonen avsluttes. For de options som ikke har noen parametre gjelder at vi kan sette den motsatte option vha. prefikset NO (f.eks. DATE kontra NODATE). For å finne ut hvilke verdier som er satt som standard options, kan vi kjøre PROC OPTIONS. Denne skriver alle options ut til SASlog med en kort forklaring. Her følger de viktigste system options, de mest brukte er uthevet og de som kun kan settes ved oppstart av SAS står i kursiv.

<i>ALTLOG=</i>	Angir hvor en kopi av SASlog skal styres.
<i>ALTPRINT=</i>	Angir hvor en kopi av SAS utskrift skal styres.
<i>AUTOEXEC=</i>	Angir navnet på oppstartsfil (autoexec.sas).
<i>AUTOSIGNON=</i>	Angir om en SAS sesjon skal startes automatisk når passord allerede er oppgitt tidligere. Brukes sammen med RSUBMIT for parallellprosessering, se side 108.
<i>BATCH</i>	Angir at system options for batchkjøringer skal brukes.
<i>BUFNO=</i>	Angir antall buffere som skal brukes ved lesing og skiving av SAS datasett. Hvis BUFNO settes til 0, brukes operativsystemets verdi som stort sett er brukbar. Standard er 0.
<i>BUFSIZE=</i>	Angis i bytes og sier hvor store bufferne skal være. Hvis BUFSIZE settes til 0, brukes operativsystemets verdi. Standard er 0.
<i>BYERR</i>	Gir feilmelding BY-setning mangler eller BY-variabel ikke finnes på datasettet som skal sorteres. NOBYERR undertrykker denne feilmeldingen. Se også DSNFERR- og VNFERR-options.
<i>BYLINE</i>	Angir at en linje med BY-verdier skal skrives ut over hver BY-gruppe når BY-setningen brukes i rapportprosedyrer. Standard er BYLINE. Ved å bruke NOBYLINE kan vi få BY-variabelens verdi i overskriften. Overskriften må da stå i doble fnutter. BY-variabelens verdier kan hentes direkte fra hjelpevariabelen #BYVAL og plasseres i overskriften med en TITLE-setning.
<i>CAPS</i>	Gjør alle små bokstaver om til store på datafiler definert ved bruk av CARDS, CARDS4, DATALINES og DATALINES4-setninger.
<i>CARDIMAGE</i>	Gjør at linjer som leses inn med CARDS, CARDS4, DATALINES eller DATALINES4-setninger ikke kan være mer enn 80 tegn. NOCARDIMAGE, som er standard, opphever denne grensen. Derimot settes grensen for hvor lang en enkelt variable kan være til 83 posisjoner.
<i>CATCACHE=</i>	Angir antall SAS kataloger som kan være åpne samtidig.
<i>CENTER</i>	Midtstiller utskrift. Standard er CENTER for Windows og NOCENTER for Unix.
<i>CLEANUP</i>	Angir hvordan SAS skal behandle situasjoner der ressurser er brukt opp. NOCLEANUP er standard og stort sett brukbar.
<i>COMAMID</i>	Angir aksessmetode for kommunikasjon ved klient/tjener-bruk. Gyldige verdier under Unix er tcp, telnet og rasync. Vi bruker tcp.
<i>COMDEF=</i>	Angir for Windows SAS hvor dialogboks for SAS-kommandoer eventuelt skal plasseres. Standard er i midten nederst (BOTTOM CENTER). Vertikalt kan vi velge mellom TOP, CENTER og BOTTOM. Horisontale muligheter er LEFT, CENTER og RIGHT.
<i>COMPRESS =</i>	YES eller CHAR angir at alle nye SAS datasett skal komprimeres med RLE-teknikk (Run Legth Encoding), BINARY angir at det skal komprimeres med RDC-teknikk (Ross Data Compression), og NO angir at de ikke skal komprimeres. Når vi ønsker å komprimere SAS datasett bør vi bruke SAS datasett option COMPRESS= isteden, se side 48. Standard er NO.

<i>CONFIG=</i>	Angir hvilken konfigurasjonsfil som skal brukes (kan ikke brukes i en konfigurasjonsfil). Standard er sasv8.cfg.
<i>CONNECTREMOTE=</i>	Angir en tjenermaskin som skal brukes. Denne brukes når vi ønsker å ha tilgang til Unix fra Windows-SAS. Het før <i>REMOTE=</i> .
<i>CPUID</i>	Gjør at informasjon om CPU skrives ut i begynnelsen av SASlog (standard).
<i>DATASTMTCHK=</i>	<i>COREKEYWORDS</i> er standard og angir at visse SAS nøkkelord ikke kan brukes i ett-leddede navn i DATA-setningen. Disse nøkkelordene er <i>MERGE</i> , <i>RETAIN</i> , <i>SET</i> og <i>UPDATE</i> . <i>ALLKEYWORDS</i> angir at ingen SAS nøkkelord kan brukes i DATA-setningen og <i>NONE</i> angir at alle SAS nøkkelord er lovlige.
DATE	Angir at dagens dato (på engelsk) blir skrevet ut øverst til høyre på hver side i utskrifter. Standard er <i>DATE</i> under Windows og <i>NODATE</i> under Unix.
DETAILS	Gir mer informasjon (antall observasjoner og antall variable) om SAS datasett når disse listes (i <i>DIR</i> -vinduet, se side 29). Standard er <i>NODETAILS</i> .
<i>DEVICE=</i>	Angir en terminaldriver for SAS/GRAPH. Settes automatisk ved bruk av SAS/GRAPH (til <i>WIN</i> under Windows og <i>Xcolor</i> under Unix Xvision).
<i>DFLANG=</i>	Angir hvilket lands utskriftsformater som skal benyttes for tid- og datovariabel utskriftsformater som starter med <i>EUR</i> , (se side 38). <i>NORWEGIAN</i> er mest aktuelt.
<i>DKRICOND=</i>	Angir hva slags reaksjon SAS skal gi når SAS ikke finner variable listet med <i>DROP</i> , <i>KEEP</i> eller <i>RENAME</i> datasett options ved lesing. <i>ERROR</i> angir at feilmelding skal gis og DATA-steget stoppes, <i>WARN</i> angir at det bare skal gis en advarsel og <i>NOWARN</i> angir ingen reaksjon. Standard er <i>ERROR</i> .
<i>DKROCOND=</i>	Angir hva slags reaksjon SAS skal gi når SAS ikke finner variable listet med <i>DROP</i> , <i>KEEP</i> eller <i>RENAME</i> datasett options ved skriving. <i>ERROR</i> angir at feilmelding skal gis og DATA-steget stoppes, <i>WARN</i> angir at det bare skal gis en advarsel og <i>NOWARN</i> angir ingen reaksjon. Standard er <i>WARN</i> .
<i>DMS</i>	Muliggjør bruk av SAS-vinduene (interaktiv SAS).
<i>DSNFERR</i>	Angir at SAS gir feilmelding og stopper programmet når vi prøver å lese et SAS datasett som ikke eksisterer. <i>NODSNFERR</i> undertrykker feilmeldingen og kjører videre i programmet. Programmet vil altså bli gjennomført selv om det har oppstått feil, og det uten å skrive ut feilmeldinger. Standard er <i>DSNFERR</i> . Se også <i>BYERR</i> - og <i>VNFERR</i> -options.
<i>ECHO=</i>	Angir en melding som skal sendes til skjermen ved oppstart.
<i>ECHOAUTO</i>	Innholdet av en eventuell autoexec-fil skrives til SASlog.
<i>EDITCMD=</i>	Angir i Unix SAS hvilken editor som skal startes ved bruk av kommandoen <i>HOSTEDIT</i> . For å bruke <i>nedit</i> startes SAS med kommandoen : sas -editcmd 'nedit' For å starte <i>nedit</i> i SAS brukes så kommandoen <i>HOSTEDIT</i> .
<i>EMAILSYS=</i>	Angir for windows e-post aksess-metode eller for unix script-fil som brukes for å sende e-post fra SAS. <i>EMAILSYS=mapi</i> er standard for windows og det virker fint. For unix er <i>EMAILSYS</i> hos oss endret fra <i>!SASROOT/utilities/bin/sasmailer</i> til <i>!SASROOT/utilities/bin/sasm.elm.mime</i> for å kunne sende med vedlegg. Denne scriptfilen er også tilpasset i forhold til den som fulgte med SAS. For å få den til å virke må også programmene <i>splitmail</i> , <i>mimencode</i> og <i>elmaliases</i> finnes. Disse er lastet ned fra Internett og lagt under /local/bin-katalogen. Se også <i>FILENAME</i> -setningen, side 98, for hvordan vi sender e-post.
<i>ENGINE=</i>	Angir lagringsformat for SAS datasett som skal leses og lagres. Mer om dette under <i>LIBNAME</i> -setningen på side 100.
<i>ERRORABEND</i>	Tvinger SAS til å avbryte programmet ved de fleste situasjoner der SAS gir en feilmelding. Standard er <i>NOERRORABEND</i> .
<i>ERRORCHECK=</i>	<i>STRICT</i> angir at SAS skal gå over i syntakssjekkmodus når en feil oppstår i en <i>LIBNAME</i> - eller en <i>FILENAME</i> -setning. Videre at programmet eller sesjonen avbrytes hvis vi prøver å hente inn en ikke-eksisterende programfil med <i>%INCLUDE</i> -setningen. <i>NORMAL</i> , som er standard, lar disse passere uten å avbryte.
<i>ERRORS=</i>	Angir antall observasjoner SAS skal skrive ut komplette feilmeldinger for. Slike feilmeldinger lages når det er feil i dataene, f.eks. at vi prøver å lagre bokstaver i numeriske felt. Standard er 20.
<i>EXPLORER</i>	Angir at SAS bare skal åpne Explorer-vinduet og verktøykassen (toolbox) ved oppstart.

FIRSTOBS=	Angir observasjonsnr for den første observasjons SAS skal behandle. Bruk heller SAS datasett option FIRSTOBS= isteden (side 48), da det er lett å glemme at FIRSTOBS-option er i bruk.
FMterr	Angir at SAS skal gi feilmelding hvis utskriftsformater som er knyttet til variablene ikke blir funnet. Dette kan skje når vi har fått et eksternt SAS datasett med utskriftsformater, uten at utskriftsformatene er sendt med. Dataene kan brukes uten utskriftsformater, ved option NOFMterr. Standard er FMterr.
FMTSEARCH=	Angir (i parentes og med blank mellom hver) hvilke librefs det skal letes etter SAS dataformater i og rekkefølgen områdene skal gjennomføres. Det letes først på arbeidsområdet (WORK.FORMATS), deretter i det permanente formatbiblioteket (LIBRARY.FORMATS) og til slutt librefs listet i FMTSEARCH. Dette overstyres dersom WORK- og/eller LIBRARY listes i FMTSEARCH.
FORMCHAR=	Angir standard formaterings tegn for utskrift fra prosedyrene FREQ og TABULATE. Disse kan også angis i de gjeldende prosedyrene, se side 141.
FORMDLIM=	Angir skilletegn for sideskift. Når denne ikke er blank vil det istedenfor sideskift skrives en blank linje, en linje med tegnet angitt i FORMDLIM og en blank linje til. Dette kan spare papir fordi flere sider kan komme på samme papir.
FORMS=	Angir navn på et skjermbilde som angir hvordan standard layout for skjermbilder skal være. Dette lages vha. FSFORMS-vinduet.
FONT=	Angir for Windows SAS teksttype (i fnutter) og størrelse. Standard er 'SASFONT' 8.
FULLSTIMER	Angir at alle tilgjengelige informasjonen om ressursbruk skal skrives til SASlog. Standard er NOFULLSTIMER.
INITCMD=	Kan brukes for å angi en kommando for å starte en SAS/AF-applikasjon direkte, men dette bør heller gjøres i autoexec.sas.
INITSTMT=	SAS-setningen (i fnutter) angitt her vil bli eksekvert etter autoexec.sas, men før eventuelle andre program (sysin) eksekveres.
INVALIDDATA=	Angir hvilket tegn (i fnutter) som skal brukes som MISSING-tegn for ugyldige data. Standard er punktum (.).
LABEL	Tillater bruk av variabel-labler i prosedyrer.
LAST	Angir navn på datasett som skal regnes som sist laget. Standard er det datasettet som sist ble laget.
LINESIZE=	Angir standard linjestørrelse. Kan ha verdier mellom 64 og 256 (=MAX) tegn.
LOG=	Angir hvor SASlog skal styres ved batchkjøringer.
MACRO	Angir at SAS macro-språk kan brukes. Standard er MACRO.
MAPS=	Angir området der kart til bruk i SAS/GRAPH ligger lagret
MAUTOSOURCE	Gjør at SAS vil prøve å finne makroer som kalles opp ved å lete i SAS autocall library (se SASAUTOS under) når den ikke tidligere er definert i programmet.
MEMSIZE=	Angir hvor mye minne SAS kan bruke. Oppgis i <i>n</i> bytes, <i>nK</i> kilobytes, <i>nM</i> megabytes eller <i>nG</i> gigabytes. Standard er 67108864 tegn på unix, 0 under windows. 0 betyr at SAS kan bruke så mye minne det er tilgjengelig.
MERGENOBY=	Angir om kobling av SAS datasett med MERGE-setningen (se side 80) uten BY-variable (definert i BY-setningen) skal være lov (NOWARN, standard), gi advarsel (WARN) eller feilmeldig (ERROR).
MERROR	Gir advarsel (warning) når det i programmet finnes et makronavn (%makronavn) uten at denne er definert. Standard er MERROR.
MFILE	Angir om utskrift av SAS setninger som er generert fra SAS makroprosessor skal skrives ut til en ekstern fil. Virker bare sammen med MPRINT-option. Filen koden skal legges i må være definert i en FILENAME-setning med filref MPRINT.
MISSING=	Angir (i fnutter) et tegn som skal brukes for å skrive ut MISSING-verdier for numeriske variable. Standard er punktum (.).
MLOGIC	Skriver SAS makroprosessorens logikk til SASlog. Standard er NOMLOGIC. Nyttig til debugging av SAS-makroer, selv om det kan bli mye tekst som skrives ut.
MPRINT	Skriver SAS-setninger generert av SAS makroprosessoren til SASlog. Denne er nyttig ved feilsøking av makroer. Standard er NOMPRINT.
MRECALL	Søker igjennom SAS autocall library (Se SASAUTOS under) en gang til for å finne makro som ikke er funnet tidligere (ikke vært tilgjengelig tidligere). Standard er NOMRECALL.
MSG=	Angir plassering av library med SAS meldinger. Het tidligere SASMSG=.
MSGCASE	Angir om meldinger (feil, advarsler og noter) skal skrives med store bokstaver eller ei. Standard er NOMSGCASE.

MSGLEVEL=	Angir mengden informasjon som skal skrives til SAS log. Med N får vi normal mengde. Med I for litt mer informasjon (f.eks. får vi melding om hvilket sorteringsprogram SAS bruker og om variable blir overskrevet når vi kobler datasett med MERGE-setningen). Standard er MSGLEVEL=N.
MSTORED	Muliggjør bruk av ferdigkompileerte makroer. Standard er NOMSTORED.
MVARSIZE=	Angir hvor mye plass i minne som settes av til makrovariable. Standard er 32768 tegn. Hvis plassen brukes opp, må SAS begynne å lagre makrovariablenes innhold på disk.
NEWS=	Angir en fil som inneholder meldinger som skal skrives til SASlog ved oppstart
NOTES	Angir at meldinger (notes) skal skrives til SASlog. Standard er NOTES.
NUMBER	Skriver sidenr øverst til høyre på hver side i SAS utskrift. Standard er NUMBER.
OBS=	Angir observasjonsnr for den siste observasjon SAS skal behandle. Bruk heller SAS datasett option OBS= isteden (side 49), da det er lett å glemme at OBS-option er i bruk. Standard er MAX.
PAGENO=	Angir nytt sidennummer for begynnelsen av neste utskrift
PAGESIZE=	Angir standard sidestørrelse. Kan variere fra 15 til 32767 (=MAX) linjer.
PARM=	Angir (i fnutter) en parameter til et eksternt program som skal eksekveres
PARMCARDS=	Angir en filreferanse til en fil som leses når en PARMCARDS eller PARMCARDS4-setning brukes i en prosedyre.
PATH=	Angir et eller flere områder der SAS skal lete etter filer som er eksekverbare for SAS. Brukes vanligvis bare i konfigurasjonsfilen til SAS.
PRINT=	Angir hvor utskrift fra SAS skal styres ved batchkjøringer.
PRINTCMD=	Angir for Unix SAS hvilken kommando som skal brukes ved utskrift med PRINT-kommandoen. Denne er vanligvis satt slik at utskrift styres til nærmeste skriver.
PROBSIG=	Angir antall signifikante siffer p-verdier skal ha i statistiske prosedyrer. 0 angir at det skal være 4 signifikante siffer, 1 angir minst 1 og 2 angir minst 2. Standard er 0.
PROC	Angir at eksterne program som skal eksekveres i SAS, kan kalles med PROC og navnet på programmet.
PROCLEAVE=	Angir en verdi som viser hvor mye minne som ikke skal allokeres, men spares til bruk for å få en prosedyre til å avslutte kontrollert hvis en feilsituasjon skulle oppstå.
REPLACE	Angir at permanente SAS datasett blir erstattet ved en ny kjøring som skriver til datasettet, dette er standard.
REUSE=	Angir om plass frigjort ved komprimering av SAS datasett skal benyttes til nye observasjoner. REUSE=NO er standard.
RSASUSER	Angir at vi kun skal ha lesetilgang på SAS profilfil. Profilfilen inneholder bl.a. definisjoner av funksjonstaster. Med RSASUSER kan vi ha flere SAS-sesjoner samtidig med tilgang på funksjonstastene, men vi får da ikke lov til å endre dem. Standard er NORSASUSER.
S=	Angir maksimal lengde på en SAS-setning på en linje.
SASAUTOS=	Angir et eller flere libraries der SAS autocall makroer er lagret. Er det flere enn ett, må de settes i parentes med komma eller blank imellom.
SASCMD=	Angir standard kommando for å starte SAS fra en annen SAS-sesjon.
SASHELP=	Angir plassering av SASHELP library.
SASINITIALFOLDER=	Angir for Windows SAS hvilket område SAS skal hente og lagre programmer fra. Bør settes til det samme som Start i-området til ikonet SAS startes fra.
SASUSER=	Angir plassering av SASUSER library. Dette inneholder brukerens profilkatalog der funksjonstaster etc. er definert.
SERROR	Gjør at det skal skrives ut feilmelding når vi refererer til en makrovariabel som ikke er definert eller gitt en verdi. Standard er SERROR.
SEQ= n	Angir at de n første tegn på linjer som hentes inn til SAS regnes som linjenr.
SET=	Brukes for å sette en SAS miljøvariabel. Denne inneholder en filreferanse for eksterne filer. Brukes vanligvis bare i konfigurasjonsfilen til SAS. Når den er gitt en verdi, beholder den denne under hele SAS-sesjonen. En ny tilordning vil ikke overstyre den opprinnelige verdien.
SETINIT	Tillater endring av lisensinformasjonen. For administrativ bruk, krever passord.
SKIP=	Angir antall linjeskift som skal utføres på begynnelsen av en side før første overskrift skrives ut. Disse linjeskiftene inngår ikke i linjene definert med PAGESIZE option, se denne. Standard er 0.
SOLUTIONS	Angir at en SOLUTIONS-rullegardinmeny kommer opp i vinduene som åpnes ved oppstart.

SORTCUT=	Angir for Unix SAS observasjonsgrensen for når SAS sorteringsprogram skal erstattes av et eksternt sorteringsprogram.
SORTCUTP=	Angir for Unix SAS grensen for antall bytes (evt. Kb el. Mb) for når SAS sorteringsprogram skal erstattes av et eksternt sorteringsprogram.
SORTDEV=	Angir for Unix SAS område der eksternt lagring under sortering skal legges.
SORTDUP=	PHYSICAL angir at sortering med sorteringsoption NODUPREC i PROC SORT fjerner dubletter der alle variable er identiske på sorterte observasjoner som kommer etter hverandre. Dette er standard. LOGICAL angir at bare variable angitt med KEEP- eller DROP-option må være identiske for at NODUPREC skal tre i kraft.
SORTLIB=	Angir for Unix SAS område der eksternt sorteringsprogram finnes.
SORTNAME=	Angir for Unix SAS navn på eksternt sorteringsprogram.
SORTPARM=	Parametre for eksternt sorteringsprogram hvis det brukes.
SORTPGM=	Angir hvilket sorteringsprogram som skal velges. Alternativene er BEST som ber SAS velge det beste, HOST som er operativsystemets sorteringsprogram evt. det som er oppgitt i SORTNAME= option og SAS som er SAS' egen sorteringsrutine. Standard er SAS.
SORTSEQ=	Angir at nasjonale spesialbokstaver skal sorteres riktig. NORWEGIAN burde virke, men gir XYZÅÆØ.
SORTSIZE=	Angir hvor mye minne SAS kan bruke til sortering. Oppgis i <i>n</i> bytes, <i>n</i> K kilobytes, <i>n</i> M megabytes eller <i>n</i> G gigabytes. Standard er 33554432 tegn på unix, 2097152 tegn for windows.
SOURCE	Angir at SAS-setninger som eksekveres skal skrives ut til SASlog i motsetning til NOSOURCE. Standard er SOURCE
SOURCE2	Angir at SAS-setninger som inkluderes i SAS programmet med %INCLUDE-setningen (side 69), skal skrives ut til SASlog. Standard er NOSOURCE2.
SPOOL	Angir at eksekverte SAS-setninger lagres på et arbeidsområde, slik at de kan kalles med en %INCLUDE (side 69) eller %LIST-setning (side 80). Standard er SPOOL.
STIMER	Gjør at viktig informasjon om ressursbruk skrives til SASlog. Standard er STIMER.
SYMBOLGEN	Angir at verdier makrovariable settes til skrives ut i SASlog. Nyttig ved feilsøking av makroer. Standard er NOSYMBOLGEN.
SYSIN=	Angir filnavn for SAS program som skal eksekveres. Når denne brukes vil SAS bli kjørt i batch. Ved batchkjøring av SAS program, kan vi som oftest sløyfe SYSIN= og bare skrive navnet på SAS programmet (f.eks. sas tab01.sas).
SYSLEAVE=	Angir en verdi som viser hvor mye minne som ikke skal allokeres, men spares slik at SAS kan avsluttes kontrollert hvis en feilsituasjon skulle oppstå. Standard er 8096.
SYSPARM=	Angir (i fnutter) en parameter som kan brukes i SAS sesjonen. Denne kan hentes fra den automatiske makrovariabelen &SYSPARM eller fra funksjonen SYSPARM (se side 247). Parameterteksten kan være maksimalt 32767 tegn.
SYSPRINT=	Angir for Unix en skriverdriver for utskrifter eller options til lp eller lpr kommando.
S2=	Angir maksimal lengde på en SAS-setning på en linje i filer som inkluderes med %INCLUDE-setningen (side 69).
TERMINAL	Angir om skjermdriver skal benyttes under SAS sesjonen.
TOOLDEF=	Angir for Windows SAS hvor Toolbox vinduet eventuelt skal plasseres. Standard er øverst til høyre (TOP RIGHT). Vertikalt kan vi velge mellom TOP, CENTER og BOTTOM. Horisontale muligheter er LEFT, CENTER og RIGHT.
USER=	Angir standard library for datasett med ett ledd. Settes USER= til noe annet en WORK eller blank, lagres SAS datasett med ett ledd permanent under dette library.
VALIDVARNAME=	V7, som er standard, angir at SAS variabelnavn kan ha en lengde på 32 tegn. Store og små bokstaver kan brukes, men det skilles ikke mellom disse. UPCASE er som V7, men alle bokstaver blir gjort til store. Følger ellers vanlige regler for navnsetting i SAS, se side 26. ANY angir at alle tegn er lov i variabelnavn, men hvis vi skal bruke alle mulige tegn må navnene stå i fnutter etterfulgt av bokstaven N for å angi at det er et SAS variabelnavn.
VERBOSE	Gjør at SAS system options skrives til skjermen ved oppstart. Standard er NOVERBOSE.
VNFERR	Gir feilmelding hvis SAS ikke finner en variabel, f.eks. en BY-variabel i en sortering. NOVNFERR undertrykker feilmeldingen. Se også BYERR- og DSNFERR-options.
WORK=	Angir et arbeidsområde for temporære filer.
WORKINIT	Angir at arbeidsområde for temporære filer skal tømmes ved oppstart av SAS hvis det finnes fra før. Standard er WORKINIT.

WORKTERM	Angir at arbeidsområde for temporære filer skal tømmes ved avslutning av SAS. Standard er WORKTERM.
XCMD	Angir om det skal være lov å utføre operativsystemkommandoer fra SAS.
XWAIT	Angir for Windows SAS at vi må taste en tast etter at en X-setning (se side 110) er utført. For å slippe dette kan vi bruke NOXWAIT isteden.
XSYNC	Gjør under Windows SAS at en X-setning (operativsystemkommando) utføres før SAS fortsetter med neste SAS-setning. NOXSYNC bevirker at X-setningen utføres i bakgrunnen mens SAS-programmet fortsetter. Standard er XSYNC.
YEARCUTOFF=	Angir hvilket årstall som skal være begynnelsen på den hundreårsperiode som skal være standard når bare de to siste sifrene i årstallet vises. Med 1900 som verdi vil årstall med to siffer regnes for å tilhøre 1900-tallet. Standardverdi er satt til 1900.

3.8 SAS datasett options

Alle steder vi nevner et SAS datasettnavn, kan vi angi options til dette. De fleste options kan brukes både når vi leser og når vi lager SAS datasett, men noen options kan bare brukes når SAS datasett leses og andre kan kun brukes når vi lager SAS datasett. SAS datasett options skal komme rett etter datasettnavnet, inne i en parentes. De mest brukte SAS datasett options er uthevet.

ALTER=	Angir passord for tilgang til å skrive, slette og endre datasettet
BUFNO=	Angir antall buffere som skal brukes når datasettet behandles
BUFSIZE=	Angir bufferstørrelse som skal brukes når datasettet behandles. Oppgis i bytes
CNTLLEV=	Angir om datasettet skal låses på en observasjon (REC) eller på hele datasettet (MEM), når flere prøver å behandle datasettet samtidig. Kan også låse hele området (librefen) for skriving (LIB).
COMPRESS=	YES eller CHAR når vi ønsker å lagre datasettet som inneholder mange karaktervariable i komprimert form. Bør gjøres for store datasett. Komprimeringen foretas på observasjonsnivå, det vil si at det komprimeres observasjon for observasjon. Det legges også ut informasjon om hvordan observasjonen skal blåses opp igjen. Denne informasjonen tar noe plass, hvilket kan føre til at et komprimert SAS datasett blir større enn om det ikke var komprimert. SAS datasett med recordlengde over 64 (f.eks. 8 variable á 8 posisjoner) vil egne seg for komprimering og også SAS datasett med mange like tegn etter hverandre (f.eks. blanke). For SAS datasett med mange numeriske og få karaktervariable bør vi velge COMPRESS=BINARY.
DLDMGACTION=	Angir hva som skal gjøres om det oppdages at SAS datasettet er ødelagt. FAIL er standard og det gjør at steget stoppes og det gis en feilmelding til SAS-log. ABORT gjør det samme som FAIL, i tillegg avsluttes SAS. REPAIR forsøker å reparere skaden og gir en melding til SAS-log. PROMPT gir et vindu med de valgene som er nevnt over.
DROP=	Ved lesing og skriving av SAS datasett utelates variablene som listes i DROP=. Se for øvrig beskrivelse av DROP-setningen på side 62.
ENCRYPT=	Angir om datasettet skal krypteres (YES) eller ikke (NO). Brukes sammen med passord (PW=, READ=).
FILEFMT=	Angir om datasettet skal lagres med filformat for versjon 606 el. 607.
GENMAX=	Angir maksimalt antall utgaver av SAS datasettet det tas vare på. GENMAX=5 gir oss 1 gjeldende og opptil fire historiske versjoner av SAS datasettet.
GENNUM=	Brukes for å angi hvilken utgave av SAS datasettet vi vil behandle.
FIRSTOBS=	Angir observasjonsnummeret på den første observasjon som skal behandles
IDXNAME=	Angir at SAS skal benytte den angitte indeks til å oppfylle en WHERE-betingelse. Dette gjør vi når vi mener at et søk ved hjelp av denne indeks vil gå raskere enn om SAS selv skulle avgjøre om en indeks skal brukes eller om dataene skal gjennomløpes sekvensielt.
IDXWHERE=	YES angir at indeks skal brukes til å oppfylle WHERE-betingelsen, NO at indeksen ikke skal brukes. Når vi har laget indekser til SAS datasett, vil SAS avgjøre om indeksen skal brukes eller ikke. Vi overstyrer dette ved å bruke IDXWHERE eller IDXNAME-options.
IN=	Brukes når vi kobler data for å holde rede på om datasettet bidrar med data eller ei, se kobling av datasett, side 17.

INDEX=	Brukes for å lage indeks(er) til et nytt SAS datasett. INDEX=(indeks-1<=(variabel-1 <... variabel-n>)</UNIQUE> </NOMISS>> ... <indeks-n<=(variabel-n1 <... variabel-nn>)</UNIQUE> </NOMISS>>); UNIQUE krever at verdiene for indeksen må være unike, mens NOMISS gjør at MISSING-verdier ikke får indekser.
KEEP=	Ved lesing og skrijving av SAS datasett inkluderes variablene som listes i KEEP=. Se for øvrig beskrivelse av KEEP-setningen på side 76.
LABEL=	Lager en beskrivende tekst til SAS datasettets innhold. Må ikke forveksles med LABEL-setningen (side 78), som brukes for å gi beskrivende tekster til variable. Teksten kan være opptil 256 tegn.
OBS=	Angir observasjonsnummeret på den siste observasjon som skal behandles.
OUTREP=	Angir operativsystem SAS datasettet skal lagres for. Aktuelle verdier er ALPHA_OSF (Digital unix), ALPHA_VMS, CMS, HP_UX, MAC, MVS, OS2, RS_6000_AIX, SOLARIS (Sun unix), VAX_VMS, VSE og WINDOWS (Windows NT).
POINTOBS=	YES, som er standard, angir at komprimerte SAS datasett kan aksesseres med direkte oppslag på observasjonsnr med POINT=option i SET-setningen (se side 88). Hvis vi ikke trenger denne muligheten, kan vi angi NO som utelukker det. Det vil spare ca. 10% av behandlingstiden når SAS datasettet lages. POINTOBS=YES overstyres av REUSE=YES og blir da satt til NO.
PW=	Angir passord for tilgang til å lese og oppdatere datasettet
PWREQ=	YES, som er standard, angir at vi skal få opp et vindu med mulighet til å taste inn passord, hvis passordet på filen som leses er feil eller ikke oppgitt. Passordet opprettes med PW-option.
READ=	Angir passord for tilgang til å lese datasettet.
RENAME=	Endre navn på variable. RENAME=(gml-var-1 = ny-var-1 gml-var-2 = ny-var-2). Hvis både KEEP= og RENAME= brukes, skal det gamle navnet brukes i KEEP=
REPEMPTY=	YES, som er standard, angir at nytt datasett skal overskrive gammelt, selv om det nye er tomt. Med NO overskriver vi ikke det gamle med et tomt et. Se også REPEMPTY-option til LIBNAME-setningen, side 101.
REPLACE=	Angir om datasettet skal overskrives (YES) eller ikke (NO) når det finnes fra før. Hvis REPEMPTY er satt til NO vil ikke datasettet bli overskrevet selv med REPLACE=YES når det nye datasettet er tomt.
REUSE=	Angir om plass frigjort ved komprimering skal benyttes til nye observasjoner (YES), NO er standard.
SORTEDBY=	Angir hvilke variable datasettet er sortert etter.
TOBSNO=	Angir antall observasjoner som skal overføres til/fra server hver gang .
TYPE=	Angir at datasettet er strukturert for spesiell bruk.
WHERE=	Brukes for å selektene data i datasettet. WHERE=(betingelse). WHERE= kan også brukes på utdatasett. Se for øvrig beskrivelse av WHERE-setningen på side 91.
WRITE=	Angir passord for tilgang til å lese og oppdatere datasettet.

Andre SAS datasett options: TRANTAB og WHEREUP.

3.9 DATA-steget

DATA-steget er den delen av SAS systemet som ligner mest på vanlig programmering. I DATA-steget bearbeider vi dataene våre. DATA-steget brukes blant annet til lage SAS datasett datasett fra rådatafiler og omvendt, omkodning av data, lage nye variable på eksisterende datasett, omstrukturere datasett og koble datasett.

DATA-steget har automatisk iterasjon. Det betyr i utgangspunktet at alle SAS-setningene i et DATA-steg utføres etter tur for hver observasjon så lenge det er observasjoner å behandle. Når det ikke er flere observasjoner igjen, avsluttes DATA-steget. Videre har DATA-steget en usynlig OUTPUT-setning på slutten av DATA-steget. Denne sørger for å skrive en observasjon ut til SAS datasett. Når vi bruker en egen OUTPUT-setning, fjernes den automatiske OUTPUT-setningen.

Alle filer og datasett åpnes og lukkes automatisk av SAS, så vi slipper å tenke på det underveis i programmeringen.

Et DATA-steg avsluttes med en RUN-setning, en ny DATA-setning eller en PROC-setning. Vi kan ha mange DATA-steg og PROC-steg i ett og samme program, de prosesseres i tur og orden.

Noen SAS-setninger og parametre er utelatt, fordi de er lite brukt eller fordi de må brukes med stor forsiktighet. For fullstendig oversikt over alle muligheter, henviser vi til manualene. Mye er også beskrevet i det interaktive hjelpesystemet til SAS.

Her følger en oversikt over de fleste setninger vi kan bruke når vi programmerer i SAS.

3.9.1 ABORT

Når ABORT-setningen utføres, stopper DATA-steget. Vi bruker ABORT-setningen når vi ønsker å stoppe et DATA-steg når en feilsituasjon oppstår. SAS vil ikke erstatte et eksisterende SAS datasett med et som skulle lages i DATA-steget når ABORT-setningen blir utført.

SAS program vil reagere litt forskjellig på ABORT-setningen avhengig av om det kjøres interaktivt eller i batch. Ved interaktiv kjøring vil programmet fortsette med neste PROC eller DATA-steg, mens ved batchkjøring vil resten av programmet bli kjørt med OBS=0, hvilket i praksis innebærer at programmet kun sjekkes for syntaksfeil, ingenting utføres.

ABORT <ABEND | RETURN> <n> ;

ABEND	I tillegg til å avslutte DATA-steget avsluttes SAS automatisk på unormal måte ved utførelsen av ABORT-setningen (anbefales ikke).
RETURN	I tillegg til å avslutte DATA-steget avsluttes SAS på automatisk på normal måte ved utførelsen av ABORT-setningen.
n	Et heltall som sendes som returkode til operativsystemet ved utførelse av ABORT. Hva slags returkode som bør sendes avhenger av hva vi ønsker å gjøre og hvilke koder operativsystemet bruker.

3.9.2 ARRAY

En array er en liste med variable. Vi kan ikke blande karaktervariable og numeriske variable i en array, alle må være av samme type. Ved å lage bruke arrayer er at det mulig å gjøre databehandlingen enklere og programmene kortere og sikrere. arrayer bør vi først begynne å bruke når vi har god programmeringskompetanse. Vi kan ha arrayer i en eller flere dimensjoner. Flere dimensjoner gir mer kompliserte arrayer og større muligheter for å gjøre feil.

Alle variablene som defineres i en array må være enten numeriske eller karaktervariable.

ARRAY array-navn (dimensjon) <\$> <lengde> <array-elementer> <(startverdier)> ;

array-navn navn på array. Følger navnestandard for SAS-navn. Ikke bruk SAS funksjoner som array-navn.

dimensjon angir dimensjoner og antall elementer i en array.

(<startverdi-1 :> sluttverdi-1 ... <, <startverdi-n :> sluttverdi-n)

Istedenfor å angi en dimensjon kan vi ha en stjerne (*) som dimensjonsangivelse. Da overlater vi til SAS å finne ut hvor mange elementer arrayen består av. Istedenfor ()-parenteser, kan vi bruke {} eller [] når dimensjonen defineres.

\$ brukes for å angi at array-elementene er karaktervariable

lengde angir lengde på array-elementene

array-elementer her skrives navnene på de variablene arrayen skal bestå av. Vi har 3 spesialtilfeller som vi kan bruke istedenfor variabelnavn. `_NUMERIC_` definerer alle numeriske variable som en array. `_CHARACTER_` definerer alle karaktervariable som en array. `_ALL_` definerer alle variable som en array (disse må da være av samme type, enten karakter eller numeriske). `_TEMPORARY_` lager en liste over midlertidige variable. Disse blir ikke skrevet ut på noe SAS datasett. Videre blir variable i en `_TEMPORARY_` array ikke satt til MISSING ved innlesing av en ny observasjon, slik variable i vanlige arrayer blir. Utelates array-elementer defineres variable på grunnlag av array-navn og antall elementer.

startverdier Her angir vi startverdier for variablene i arrayen. Disse initialverdiene beholdes ved innlesing av en ny observasjon hvis de ikke er overskrevet. Er variablene i arrayen på SAS datasettet som leses er det liten vits i å angi startverdier, fordi disse variablenes verdier blir brukt i arrayen.

Eksempler på array-definisjoner:

```
ARRAY p18t (*) p18t_1 - p18t_23 ;
ARRAY aar (85:95) aar_85 - aar_95 ;
ARRAY mnd (12) jan feb mar apr mai jun jul aug sep okt nov des ;
ARRAY mnd (12); /* definerer variablene mnd1-mnd12 */
ARRAY char (*) _CHARACTER_ ; /* alle karaktervariable */
ARRAY num (*) _NUMERIC_ ; /* alle numeriske variable */
ARRAY hjelp (5) _TEMPORARY_ ; /* 5 midlertidige hjelpevariable */
```

Eksempel der vi skal lage nye variable om sykdomstyper ut fra diagnoser fra forskjellige variable. For hver person skal vi søke gjennom vedkommendes diagnoser og kode dem når de treffer en eller flere sykdomstyper. Fordi en person kan ha flere sykdomstyper, må vi teste på alle personens diagnoser (`DIM(sykdom)` inneholder antall sykdomsvariable), selv om vi finner en sykdomstype for den første diagnosen. Personen kan jo ha flere sykdomstyper. Etter at vi har funnet sykdomstypene, bruker vi disse til å finne ut om personen har en sykdom eller ei. Når vi har funnet ut at en person har en sykdom, kan vi avslutte testingen for denne personen med en `LEAVE`-setning. Etter kodingen lager vi en frekvensliste og skriver ut litt av det nye SAS datasettet for å sjekke at alt er i orden.


```

DATA diag;
  INFILE CARDS MISSOVER;
  INPUT @01 ident $CHAR7.
        @08 sykdom1 $CHAR5.
        @13 sykdom2 $CHAR5.
        @18 sykdom3 $CHAR5.
        @23 sykdom4 $CHAR5.
        @28 sykdom5 $CHAR5.
        @33 sykdom6 $CHAR5.
        ;
CARDS;
0078213799,8401 320
0078214741
0078217477 784,6
0078218493 477
0078219412 780
0078220413
0078221
0078222389 454
0078223
0078224799,8346
0078226346 477 401
0078227477
0078228346 490
0078229780
0078230
0078232
0078233414 401 369 454 440 743
0078236369
0078237311
0078238493
0078239784,0
0078240465
0078241
0078243413 428 412 291 365
;

DATA helse (DROP=i);
  SET diag;
  ARRAY sykdom (*) $ sykdom1 - sykdom6 ;
  ARRAY sykd (*) sykd1 - sykd5;
  DO i=1 TO DIM(sykdom);
    IF '290'<=sykdom(i)<='311' OR
       '784'<=sykdom(i)<='784,6' OR
       '799'<=sykdom(i)<='799,2' OR
       sykdom(i) in ('315,0','780','799,8') THEN
      sykd1=1;

    IF '320'<=sykdom(i)<='359' OR '740'<=sykdom(i)<='742' THEN
      sykd2=1;

    IF '360'<=sykdom(i)<='389' OR '743'<=sykdom(i)<='744' THEN
      sykd3=1;

    IF '390'<=sykdom(i)<='459' OR '745'<=sykdom(i)<='747' THEN
      sykd4=1;

    IF '460'<=sykdom(i)<='519' OR sykdom(i)='748' THEN
      sykd5=1;
  END;

  DO i=1 TO DIM(sykd);
    IF sykd(i)=1 THEN
      DO;
        syk=1;
        LEAVE;
      END;
    END;
  END;

LABEL
  syk      ='Sykdom'
  sykd1    ='Nervøse lidelser'
  sykd2    ='Sykd.i nervesytemet'
  sykd3    ='Øye- og øre sykdommer'
  sykd4    ='Hjerte- og karsykdommer'
  sykd5    ='Sykdommer i åndedrettsorganene'
;
RUN;

PROC FREQ DATA=helse;
  TABLES syk sykd1-sykd5 /MISSING;
  TITLE 'Helseundersøkelsen 1995. Antall med sykdommer';

PROC PRINT DATA=helse (OBS=111);
  TITLE 'Helseundersøkelsen 1995. Sykdomskategorisering';

RUN;

```

Eksempel der vi leser inn domsdatoer (dd1 - dd7) for straffede personer. Vi ønsker å finne hvor mange dommer hver person har (antdom) og hvor mange dager korteste (mintid) og lengste (maxtid) tid mellom to dommer utgjør:

```
DATA tilbfall (DROP=i);  
    ARRAY dd  (*)    dd1 - dd7  ;  
    INPUT fnr dd(*) yymmdd6.;  
  
    antdom = 0;  
    DO i = 1 TO DIM(dd);  
        IF dd(i) NE . THEN antdom + 1 ;  
    END;  
  
    IF antdom > 1 THEN  
        DO i = 2 TO antdom;  
            maxtid = MAX(maxtid,dd(i)-dd(i-1));  
            mintid = MIN(mintid,dd(i)-dd(i-1));  
        END;  
  
    CARDS;  
01014531131 870201870402881112890131920129931213 .  
30105441335 871211871223880412880824 . . .  
23046230045 870228870402 . . . . .  
24115334100 870731 . . . . .  
02087245555 870803931215 . . . . .  
02087143138 871212871230880106 . . . . .  
13126542319 871209880413920201920208921130930202931111  
;
```

3.9.3 assignment

Assignment- eller tilordnings-setningene er en av de mest brukte SAS-setninger. Det er også en av de få SAS-setninger som ikke begynner med et nøkkelord. Assignment-setningen brukes til å gi en variabel en verdi. Verdien tilordnes med et SAS-uttrykk. Dette uttrykket kan bestå av aritmetikk, SAS funksjoner, konstanter, sammenslåinger (konkatenering) eller kombinasjoner av disse. Det er mange eksempler på bruk av assignment i dette heftet.

variabel = SAS-uttrykk ;

variabel	Her legges resultatet av uttrykket på høyre side av likhetstegnet. Variabel er navnet på en ny eller eksisterende variabel, en array-referanse eller SUBSTR-funksjonen.
SAS-uttrykk	Inneholder et gyldig SAS-uttrykk. Dette kan også inneholde variabelen på venstre side av likhetstegnet.

Eksempler:

Aritmetikk:

```
stonad = bidrag + laan ;  
mmoms = umoms * 1.23 ;
```

SAS funksjoner:

```
stonad = SUM(bidrag,laan) ;  
idag = DATE() ;
```

Sammenheking, konkatenering,:

```
navn = fnavn || ' ' || enavn ;
```

3.9.4 ATTRIB

Tilordner egenskaper ved en eller flere variable. Egenskapene er FORMAT, INFORMAT, LABEL og LENGTH. En eller flere egenskaper kan tilordnes. De forskjellige egenskapene kan tilordnes hver for seg med FORMAT-, INFORMAT-, LABEL- og LENGTH-setninger.

ATTRIB variabelliste-1 egenskap-1 ... <variabelliste-n egenskap-n> ;

Eksempel:

```
ATTRIB kjonn LABEL='Kjønn' LENGTH=$1 ;
```

3.9.5 BY

BY bruker vi for å gruppere observasjoner. BY-variable er variablene som listes i BY-setningen, BY-verdier er verdiene til BY-variablene og en BY-gruppe er observasjoner som har samme verdi for BY-variablene. BY brukes når vi ønsker å behandle observasjoner gruppevis. Når vi behandler observasjoner som en gruppe, kan vi tenke oss at en gruppe bestående av tre deler: den første, de midterste og den siste observasjonen. Denne tredelingen gjør vi fordi den første og den siste ofte behandles på spesielle måter. For å skille mellom disse tre, har SAS to hjelpevariable for hver BY-variabel, FIRST.BY-variabel og LAST.BY-variabel. Disse hjelpevariablene er meget nyttige og vi må forstå hvordan de fungerer. FIRST-variablene settes til 1 når den første observasjonen i en gruppe med samme verdi for BY-variabelen leses, ellers er den 0. Tilsvarende gjøres for LAST-variablene, bortsett fra at de gis verdien 1 når den siste i en gruppe leses. FIRST- og LAST-variablene lagres ikke på det nye SAS datasettet. Et eksempel som viser hvilke verdier FIRST- og LAST-variablene får under bearbeidingen:

OBS NR	AAR	KOMM	FNR	INNETKT	FIRST. AAR	FIRST. KOMM	FIRST. FNR	LAST. AAR	LAST. KOMM	LAST. FNR
1	1994	0301	28096146120	420345	1	1	1	1	1	1
2	1995	0402	02113532209	153452	1	1	1	0	0	0
3	1995	0402	02113532209	009002	0	0	0	0	0	1
4	1995	0402	04043534343	110301	0	0	1	0	0	0
5	1995	0402	04043534343	123099	0	0	0	0	0	0
6	1995	0402	04043534343	033123	0	0	0	1	1	1

IF FIRST.fnr vil slå til for observasjon 1,2 og 4.

IF LAST.komm slår til for observasjon 1 og 6.

IF FIRST.fnr AND LAST.fnr vil slå til for observasjon 1.

IF NOT FIRST.fnr AND NOT LAST.fnr vil slå til for observasjon 5.

IF NOT (FIRST.fnr AND LAST.fnr) vil slå til for observasjon 2,3,4,5 og 6.

BY <DESCENDING> <GROUPFORMAT> variabel-1 ...

<<DESCENDING> <GROUPFORMAT> variabel-n> <NOTSORTED> ;

DESCENDING Angir at variabelen er sortert etter synkende verdier

GROUPFORMAT Angir at variabelen skal grupperes etter et utskriftsformat. Eksempel under.

NOTSORTED Angir at BY-variablene ikke er sortert. Det kan være nyttig hvis observasjonene kommer i grupper uten å være sortert. Se eksempel under

Eksempel der vi slår sammen observasjoner med samme fødselsnummer. Bidrag og lån summeres, kommunenummer derimot kan ikke summeres. Derfor må vi velge hva vi ønsker å gjøre med det. I dette tilfelle henter vi kommunenummer fra den siste observasjon for BY-gruppene.

```
DATA soshjelp;
  LENGTH fnr $ 11 kommune $ 4 ;
  INPUT @01 fnr          $CHAR11.
        @13 kommune     $CHAR4.
        @18 bidrag      6.
        @25 laan        6.
        ;
CARDS;
01012939013 0301 003134 000324
03023934114 0211 012199 003031
03023934114 0211 012199 000000
09075543532 0301 052944 .
16014940085 0301 003543 001611
16014940085 0301 012343 000000
19044843209 0211 000000 000232
19044843209 0211 000000 000428
19044843209 1601 000000 000660
26116141107 0211 . 000870
31126532212 0101 022100 000000
;

DATA soshjelp (drop=hbidrag hlaan);
  SET soshjelp ;
  BY fnr;

  hbidrag + bidrag;
  hlaan   + laan;

  IF LAST.fnr THEN
    DO;
      bidrag = hbidrag ;
      laan   = hlaan ;
      OUTPUT;
      hbidrag = .;
      hlaan   = .;
    END;

PROC PRINT DATA=soshjelp;
  TITLE 'Etter sammenslåing' ;
RUN;
```

Resultatet:

```
Etter sammenslåing

OBS          FNR          KOMMUNE    BIDRAG    LAAN
1      01012939013      0301         3134      324
2      03023934114      0211        24398     3031
3      09075543532      0301         52944      .
4      16014940085      0301        15886     1611
5      19044843209      1601           0     1320
6      26116141107      0211           .      870
7      31126532212      0101        22100       0
```

Skal vi slå sammen like fødselsnummer innen en kommune, holder det at alle observasjoner for en kommune er sortert etter fødselsnummer og at observasjonene kommer kommunevis, datasettet behøver ikke å være sortert etter kommunenummer. Da kan vi bruke NOTSORTED option på fødselsnummeret:

```

DATA soshjelp;
  INPUT @01 fnr          $CHAR11.
        @13 kommune     $CHAR4.
        @18 bidrag       6.
        @25 laan         6.
        ;
CARDS;
31126532212 0101 022100 000000
19044843209 1601 000000 000660
01012939013 0301 003134 000324
09075543532 0301 052944 .
16014940085 0301 003543 001611
16014940085 0301 012343 000000
03023934114 0211 012199 003031
03023934114 0211 012199 000000
19044843209 0211 000000 000232
19044843209 0211 000000 000428
26116141107 0211 . 000870
;

DATA soshjelp (drop=hbidrag hlaan);
  SET soshjelp ;
  BY fnr NOTSORTED;

  hbidrag + bidrag;
  hlaan   + laan;

  IF LAST.fnr THEN
    DO;
      bidrag = hbidrag ;
      laan   = hlaan ;
      OUTPUT;
      hbidrag = .;
      hlaan   = .;
    END;

  PROC PRINT DATA=soshjelp;
    TITLE 'Etter sammenslåing innen kommunene' ;
  RUN;

```

Nytt resultat:

Etter sammenslåing innen kommunene

OBS	FNR	KOMMUNE	BIDRAG	LAAN
1	31126532212	0101	22100	0
2	19044843209	1601	0	660
3	01012939013	0301	3134	324
4	09075543532	0301	52944	.
5	16014940085	0301	15886	1611
6	03023934114	0211	24398	3031
7	19044843209	0211	0	660
8	26116141107	0211	.	870

Eksempel på bruk av GROUPFORMAT, observasjonene må være sortert slik at alle innen en gruppe ligger samlet. I dette eksempelet betyr det at alle dødsfall i 1991 må komme før 1992, men dødsfallene i 1991 behøver ikke å være sortert etter dødsdato.

```

DATA dodaar;
  SET dode.g91g93 (KEEP=dodsdato);
  FORMAT dodsdato YEAR4.;
  BY GROUPFORMAT dodsdato ;
  RETAIN antall 0 ;
  antall + 1;
  IF LAST.dodsdato THEN
    DO;
      OUTPUT;
      antall = .;
    END;

  PROC PRINT DATA=dodaar;
    TITLE 'Antall døde etter i 1991-1993 etter år for dødsfall' ;
  RUN;

```

Dette programmet lager 1 observasjon for hvert årstall:

Antall døde etter i 1991-1993 etter år for dødsfall

OBS	DODSDATO	ANTALL
1	1991	44822
2	1992	44735
3	1993	46623

3.9.6 CALL

Kaller en SAS CALL rutine. Rutinene kalles med parametre og returnerer verdier. De minner om SAS funksjoner, men det er enkelte forskjeller. Parametrene (argumentene) til CALL-rutinene kan skrives direkte inn (i fnutter), legges i en variabel eller genereres ved et SAS-uttrykk. For sammenligning med tilfeldige tallsfunksjonene (de som begynner med RAN), se tilsvarende funksjoner side 248.

CALL	rutine (parameter-1 ... <parameter-n>);
	rutine parameter navnet på den rutinen som skal kalles parametre til rutinene
EXECUTE	Argumentet som returneres skal være en eller flere SAS-setninger. Disse blir utført ved neste DATA-steg, PROC-steg eller RUN-setning.
LABEL	Henter en variabls label og legger den i en egen variabel. Hvis variabelen ikke har noen label, hentes variabelens navn isteden. Eksempel, se side 10.
RANBIN	Returnerer numerisk verdi etter binomial fordeling CALL RANBIN(startverdivariabel,p,n,resultatvariabel); variabelen p er numerisk sannsynlighet for suksess ($0 \leq p \leq 1$) variabelen n er uavhengig Bernoulli forsøksparameter
RANCAU	Returnerer numerisk verdi etter Cauchy fordeling CALL RANCAU(startverdivariabel,resultatvariabel);
RANEXP	Returnerer numerisk verdi etter eksponentiell fordeling CALL RANEXP(startverdivariabel,resultatvariabel);
RANGAM	Returnerer numerisk verdi etter gamma fordeling CALL RANGAM(startverdivariabel,a,resultatvariabel); variabelen a er fasong-parameter, verdien til a må være større enn 0.
RANNOR	Returnerer numerisk verdi etter normalfordelingsprinsipp med gjennomsnitt 0 og varians lik 1. CALL RANNOR(startverdivariabel,resultatvariabel);
RANPOI	Returnerer numerisk verdi etter Poisson fordeling. CALL RANPOI(startverdivariabel,m,resultatvariabel); variabelen m er parameter for gjennomsnittsverdi
RANTBL	Returnerer numerisk verdi etter tabulert sannsynlighet. CALL RANTBL(startverdivariabel,p1,...,pn,resultatvariabel); variabelen p er sannsynlighetsparametre, hvor $0 \leq p \leq 1$
RANTRI	Returnerer numerisk verdi etter triangulær fordeling. CALL RANTRI(startverdivariabel,h,resultatvariabel); variabelen h er numerisk og $0 \leq h \leq 1$
RANUNI	Returnerer numerisk verdi etter uniform fordeling. CALL RANUNI(startverdivariabel,resultatvariabel);
RXCHANGE	Skifter ut tekst ved hjelp av regulære uttrykk definert med RXPARSE-funksjonen (side 232). CALL RXPARSE(rx,antall, fra-variabel <til-variabel>); rx er resultatvariabelen fra RXPARSE-funksjonen. antall er maks. antall CALL-rutinen kan kalles (maks er 2147483647 i et datasteg).
RXFREE	Frigjør minne brukt av RX-funksjoner. CALL RXFREE(rx); rx er resultatvariabelen fra RXPARSE-funksjonen.
RXSUBSTR	Finner posisjon, lengde og antall treff for et mønster definert med RXPARSE-funksjonen (se side 232). CALL RXSUBSTR(rx,posisjon<,lengde<,treff>>); rx er resultatvariabelen fra RXPARSE-funksjonen. posisjon er hvor i teksten mønsteret startes, 0 hvis det ikke finnes. lengde er lengden til det mønsteret som finnes i teksten. teff skal basere seg på antall ganger mønsteret finnes i teksten, men ser stort set ut til å bli 0.
SOUND	Gir en lyd i en viss tid. Lyden angis mellom 20 og 20000 og tiden angis i 1/80 deler sekund. CALL SOUND(frekvens,tid);
SYMPUT	Gir en makrovariabel en verdi. To parametre gis: navnet på makrovariabelen og verdien den skal få.

SYSTEM Utfører en operativsystemkommando. Denne kan brukes i IF-tester og blir da bare utført når testen slår til. Det gjelder ikke for kommandoer i X-setningen (side 110), de utføres umiddelbart og uansett om IF-testen slår til eller ei.

VNAME Henter en variabls navn og legger det i en egen variabel

Andre CALL-rutiner: MODULE, MODULEI, POKE, SET

3.9.7 CARDS

Vi bruker CARDS for å fortelle at de neste linjene i programmet er inndata til bruk i programmet. CARDS-setningen skal stå etter alle andre SAS-setninger i et DATA-steg. Etter inndataene skal det være et semikolon på en egen linje. CARDS egner seg å bruke når vi skal teste program, vi lager testdata som skal ta for seg de forskjellige muligheter programmet skal behandle. Vi kan bare ha en CARDS-setning i et DATA-steg.

CARDS er navngitt etter hullkort, som hadde maksimalt 80 posisjoner pr. linje. Dette gjaldt for CARDS også, men dette er som standard overstyrret med SAS system option NOCARDIMAGE.

CARDS ;

Eksempel:

```
* Lager testfil;
DATA tilbfall ;
  INPUT @01 fnr $CHAR11.
        @13 dd_1 YYMMDD6.
        @19 dd_2 YYMMDD6.
        @25 dd_3 YYMMDD6.
        @31 dd_4 YYMMDD6.
        ;
  CARDS;
01014531131 870201870402881112890131
30105441335 871211871223880412880824
23046230045 870228870402 . .
24115334100 870731 . .
02087245555 870803931215 . .
02087143138 871212871230880106 .
13126542319 871209880413920201920208
;
RUN;
```

3.9.8 CARDS4

CARDS4 brukes som CARDS, med unntak av at vi kan bruke semikolon i inndataene. Etter inndataene må vi dessuten avslutte med 4 semikolon etter hverandre.

CARDS4 ;

3.9.9 CONTINUE

Brukes i DO løkker og SELECT til å starte neste iterasjon i løkken

CONTINUE ;

3.9.10 DATA

Starter et DATA-steg og angir navn på SAS datasett som skal lages i DATA-steget.

DATA <datasettnavn-1 <(datasett-options-1)>> ... <datasettnavn-n <(datasett-options-n)>>
</ VIEW=view-navn | DEBUG>;

datasettnavn navn på SAS datasett. Kan være temporære (ett ledd) eller permanente (to ledd) datasett. Hvis vi ikke ønsker å lage noe SAS datasett i DATA-steget bruker vi `_NULL_` istedenfor datasettnavn.

datasett-options options til SAS datasettet som lages. Se SAS datasett options, side 47.

VIEW= brukes for å lage et data-view istedenfor eller i tillegg til et SAS datasett

DEBUG kjører programmet i DEBUG-modus (for feilsøking). Vi kan da eksekvere en og en linje i programmet og sjekke underveis hvordan det går. Det åpnes et DEBUG-vindu med diverse menyer for feilsøkingen. Fra dette vinduet kan vi gjøre forskjellig feilsøking.

Eksempel der vi lager nye variable og legger resultatet ut på et nytt SAS datasett:

```
DATA bvern.klientny ;  
  SET bvern.klienter ;  
  alder = stat_aar - f_aar ;  
  fylke = SUBSTR(kommune,1,2) ;  
RUN;
```

3.9.11 DATALINES

DATALINES kan brukes istedenfor CARDS for å angi start på inndata. Se CARDS.

DATALINES ;

3.9.12 DATALINES4

DATALINES4 kan brukes istedenfor CARDS for å angi start på inndata. Se CARDS4.

DATALINES4 ;

3.9.13 DELETE

Stopper prosessering av gjeldende observasjon. Observasjonen skrives ikke ut til SAS datasett som lages i datasteget (med mindre den allerede er skrevet ut).

DELETE ;

3.9.14 DISPLAY

Viser et vindu på skjermen. Dette vinduet må være beskrevet tidligere i programmet med en WINDOW-setning, se side 92.

DISPLAY vindu-navn<.gruppenavn> <NOINPUT> <BLANK> <BELL> ;

vindu-navn navn på vinduet. Dette vil vises øverst i vinduet.

gruppenavn har vinduet mer enn en gruppe av felt, må gruppens navn angis.

NOINPUT angir at det ikke er lov å skrive noe i feltene definert i vinduet

BLANK angir at vinduet skal tømmes før det vises.

BELL angir at det skal komme en lyd når vinduet vises

3.9.15 DO

Avgrenser en gruppe SAS-setninger til en enhet. DO må brukes når flere SAS-setninger skal utføres i en IF-test (det finnes ikke noe ENDIF i SAS). Gruppen utgjør de SAS-setninger som står mellom DO og END. Vi bruker også DO for behandling i løkker, da gjerne sammen med arrayer (array, se side 51). Betingelser som brukes sammen med DO, må være lovlige SAS-uttrykk, se side 34.

```
DO;
    SAS-setninger ;
END;

DO indeksvariabel= startverdi <TO sluttverdi> <BY endringsverdi>
    <WHILE | UNTIL (betingelse)>;
    SAS-setninger ;
END;

DO indeksvariabel= verdi-1 ... <, verdi-n> ;
    SAS-setninger ;
END;

DO UNTIL (betingelse) ;
    SAS-setninger ;
END;

DO WHILE (betingelse) ;
    SAS-setninger ;
END;
```

Eksempler:

Når flere SAS-setninger skal utføres når en betingelse oppfylles, må disse avgrenses av DO og END, ellers blir den første SAS-setningen utført når betingelsen slår til, mens de andre alltid blir utført.

```
IF kjonn = '' THEN
DO;
    PUT 'Ugyldig kjonn for obs ' _N_ ' med fnr ' fnr ' . SLETTES!!' ;
    DELETE;
END;
```

Ved hjelp av arrayer og iterativ DO kan vi forkorte tester. Uten ARRAY og DO:

```
* Finner antall omsorgstiltak i løpet av året, ==> antp18;
antp18 = 0;
IF p18ot_1 > '0' THEN antp18 + 1 ;
IF p18ot_2 > '0' THEN antp18 + 1 ;
IF p18ot_3 > '0' THEN antp18 + 1 ;
IF p18ot_4 > '0' THEN antp18 + 1 ;
IF p18ot_5 > '0' THEN antp18 + 1 ;
IF p18ot_6 > '0' THEN antp18 + 1 ;
IF p18ot_7 > '0' THEN antp18 + 1 ;
IF p18ot_8 > '0' THEN antp18 + 1 ;
IF p18ot_9 > '0' THEN antp18 + 1 ;
IF p18ot_10 > '0' THEN antp18 + 1 ;
```

Samme test med ARRAY og DO (SAS funksjonen DIM(p18ot) finner ut hvor mange variable som er i arrayen, her 11. Da overlater vi til SAS å regne ut antallet, hvilket betyr at vi kan endre array-definisjonen uten å endre DO-setningen):

```
* Finner antall omsorgstiltak i løpet av året, ==> antp18;
ARRAY p18ot (*) $ p18ot_1 - p18ot_10 ;
antp18 = 0;
DO i = 1 TO DIM(p18ot);
    IF p18ot(i) > '0' THEN
        antp18 + 1;
END;
```

Eksempel på DO-setning med WHILE. Vi ønsker å finne ut hvor mange dommer en person har. Dette finner vi ut ved telle dommer så lenge datovariablene har gyldige domsdatoer. Når domsdatoen er ugyldig, har vi funnet alle dommene og kan avslutte DO-løkken:

```
DATA tilbfall (DROP=i);
INPUT (fnr dd1-dd9) ($CHAR11. 9*YMMDD6.);
ARRAY dd (*) dd1-dd9;
antdom = 0;
DO i = 1 TO DIM(dd) WHILE (dd(i) NE .);
    antdom + 1;
END;
CARDS;
01014531131870201870402881112890131920129931213940104940731 .
30105441335871211871223880412880824 . . . . .
23046230045870228870402 . . . . .
24115334100870731 . . . . .
0208724555870803931215 . . . . .
02087143138871212871230880106 . . . . .
13126542319871209880413920201920208921130930202931111940228940928
131167443155 . . . . .
;

PROC PRINT DATA=tilbfall ;
    TITLE 'Utlisting av dommer';
    FORMAT dd1-dd9 DMMYY6.;
RUN;
```

3.9.16 DROP

Brukes for å utelate variable på utdatasett. De variablene som skal utelates, listes med DROP-setningen. DROP utelater variablene på alle datasett som lages, i motsetning til DROP-option til SAS datasett, som bare gjelder for et datasett. DROP brukes når vi skal utelate noen få variable. Skal vi utelate mange og bare beholde noen få, bruker vi KEEP isteden, se side 76.

DROP variable ;

Eksempler:

```
DROP i aar_87-aar_90 ;

DROP i_ : ; /* dropper alle variable som begynner med i_ */
```

3.9.17 END

Denne er ikke som en vanlig SAS-setning. Brukes kun for å avslutte DO- og SELECT-setninger. Disse er beskrevet på side 61 (DO) og 86 (SELECT).

END ;

3.9.18 ENDSAS

ENDSAS avslutter en SAS-sesjon eller en SAS batch kjøring. Avslutningen skjer etter at DATA- eller PROC-steget før ENDSAS er ferdig eksekvert. ENDSAS kan ikke knyttes til betingelser som IF THEN etc.

ENDSAS ;

3.9.19 ERROR

Setter den automatiske hjelpevariabelen `_ERROR_` til 1 og skriver eventuelt en feilmelding til SASlog.

ERROR <melding> ;

```
IF kjonn NOT IN ('1','2') THEN
    ERROR 'Ugyldig kjønnskode for obs. ' _N_ ' med fnr ' fnr ;
```

3.9.20 FILE

Vi bruker FILE til å identifisere en utdatafil som skal lages fra SAS. Vi kan lage en eller flere rådatafiler og/eller en eller flere utskriftsfiler. En rådatafil inneholder data, mens en utskriftsfil inneholder en ferdig rapport eller tabell med overskrifter, tekster, fotnoter etc. Med FILE forteller vi hva filen heter og hva slags fil vi skal lage. Dersom det er nødvendig, kan vi også gi mere informasjon om filen i FILE-setningen. For å skrive til filen, må vi bruke PUT-setningen (se side 82), FILE beskriver bare hva filen heter etc.

FILE fil-navn | fil-referanse | LOG | PRINT < options > ;

fil-navn	navn på rådatafilen som skal lages
fil-referanse	kortnavn til rådatafilen angitt med FILENAME-setningen. FILENAME, se side 97.
LOG	angir at det skal skrives til SASlog istedenfor en rådatafil
PRINT	angir at det skal skrives til utskriftsfil (den samme som SAS output styres til) istedenfor en rådatafil

Options til FILE (de vanligste er uthevet):

ANSI	angir at filen som det skal skrives til bruker ansi-tegnsett. Dette er standard Windows, mens DOS bruker ascii (oem). ANSI er standard.
BLKSIZE=	angir blokkstørrelsen på utdatafilen.
COLUMN=variabel	definerer en hjelpevariabel som SAS legger pekerens posisjon på recorden som det skal skrives til på rådatafilen. Denne endres hver gang pekeren starter på et nytt felt i recorden. Kan forkortes til COL=.
DELIMITER=	Angir skilletegn som skrives ut mellom hver variabel. Vanlige skilletegn er semikolon (DLM=',';) og tabulator (DLM='09'X).
DEVICE=	DDE for DDE-koblinger mot regneark etc. DISK for vanlig lesing fra disk. Dette er standard. DUMMY for null device. Kan brukes til testing av programkode. NAMEPIPE for navngitt rørkobling. PIPE for ikke navngitt rørkobling. PLOTTER angis når utskrift fra SAS/GRAPH skal styres til en plotter PRINTER angir at utskrift skal styres direkte til standardskriver
DROPOVER	Hindrer at det skrives på en ny record hvis det som skrives går ut over recordlengden. Istedenfor å skrive det resterende på neste linje, droppes det.
DSD	Gjør at variable som inneholder skilletegnet, definert med DELIMITER=, pakkes inn med anførselstegn på hver side.
FILE=variabel	Angir en hjelpevariabel som vil inneholde utskriftsbufferet, dvs. det som skal skrives ut pr. PUT-setning. Tømmes før ny linje skal skrives ut.
FILENAME=variabel	Bevirker at navnet på rådatafilen legges i en hjelpevariabel.
FILEVAR=variabel	Definerer en hjelpevariabel som inneholder navnet på den rådatafilen som det skal skrives til. Denne kan endres underveis i programmet, hvilket betyr at den kan brukes til å styre data til forskjellige filer. Det er viktig å merke seg hvordan FILEVAR= option påvirker åpning og lukking av disse filene. Når variabelen definert i FILEVAR= option endres, lukkes den filen som var åpnet (hvis det var noen). Deretter åpnes filen med navnet angitt i variabelen. Hvis denne filen har vært brukt tidligere i DATA-steget, vil tidligere utskrevne data bli overskrevet. Dette kan overstyres med MOD-option, som gjør at nye records skrives ut i tillegg til eksisterende. Hvis vi bruker MOD-option, bør vi slette filene før DATA-steget utføres.

Eksempel der SAS datasettet er sortert slik at filene aldri må åpnes mer enn en gang (det skal lages en fil pr. fylke):

```
DATA persreg;
  INPUT @01 fylke $CHAR2.
        @04 fnr $CHAR11.
        @16 stonad 5
        @22 inntekt 6.
        ;
CARDS;
01 30125628194 00000 124534
11 05103428199 04587 023488
12 06103524194 03233 023488
01 25103625194 00000 223411
03 04103457893 00000 200097
01 16103418896 00000 167435
03 08103438797 07123 045225
03 04103458698 12888 011222
11 22103438592 00000 187944
11 06103448223 05645 120999
01 07103463154 00000 452673
11 08103442095 03674 055664
01 04103434095 00000 145348
03 12033341099 01344 067897
12 22043528820 23455 009348
03 23055338690 00000 245871
03 04011448491 04561 087656
03 23063448292 00000 145310
01 24103458395 00000 155630
03 06103468392 00000 158775
03 05103487591 00000 142322
;

PROC SORT DATA=persreg;
  BY fylke;

DATA _NULL_;
  SET persreg;
  filnavn = "FY" || fylke || ".DAT";
  FILE ut FILEVAR=filnavn;
  PUT @01 fylke $2.
      @03 fnr $11.
      @14 stonad Z5.
      @19 inntekt Z6.
      ;

RUN;
```

Hvis datasettet ikke er sortert og vi ikke ønsker å sortere det, kan vi bruke denne varianten med å slette filene først for så å bruke MOD-option:

```
X 'DEL FY*.DAT';

DATA _NULL_;
  SET persreg;
  filnavn = "FY" || fylke || ".DAT";
  FILE ut FILEVAR=filnavn MOD;
  PUT @01 fylke $2.
      @03 fnr $11.
      @14 stonad Z5.
      @19 inntekt Z6.
      ;

RUN;
```

Hvis mulig, bør filen sorteres først, slik at hver fil det skal skrives til bare må åpnes og lukkes én gang. Dette gir en mer leselig SASlog.

FLOWOVER

Angir at SAS skal fortsette å skrive på neste record hvis ikke alle data som skal skrives til en record får plass på denne. Det er sjelden vi ønsker at dette skal skje, det vanlige er å utvide recordlengden og kjøre programmet på nytt. FLOWOVER er standard, dette kan skape uventede resultater. Resultatet av kjøringen vil ofte være annerledes enn forventet og derfor gir SAS en melding. Sjekk derfor resultatet nøye når denne meldingen kommer i SASlog:

SAS went to a new line when PUT statement reached past the end of a line.

For å hindre at SAS går til en ny linje uten at vi vil det, kan vi bruke option DROPOVER, se denne. Se også PUT-setningen, side 82.

Angir at fotnoter, definert med FOOTNOTE-setningen, skal skrives ut.

FOOTNOTES

HEADER=label	Angir en label hvis SAS-setninger skal brukes til å lage en kolonneoverskrift til variablene. Skrives ut øverst på hver side av utskriften. Se eksempel på side 78.
LINE=variabel	Definerer en hjelpevariabel som inneholder det relative nummeret innen en gruppe records definert med N= option eller #n peker kontroll (se PUT- setningen, side 82) til den recorden SAS skriver til rådatafilen.
LINESIZE=linjebredde	Avgrenser skiving av records til rådatafilen til det antall posisjoner som oppgis. Hvis denne grensen overskrides under skiving, vil SAS skrive resten av recorden ut på neste record (med mindre DROPOVER eller STOPOVER options er valgt).
LINESLEFT=variabel	Angir en hjelpevariabel som inneholder antall linjer som gjenstår på siden. Aktuell å bruke når bruker PUT-setningen for å skrive ut rapporter og tabeller. Eksempel. Skriver ut et datasett i to spalter:
	<pre> TITLE 'Tabell 10. Innførsel. Verdi av viktige varer etter hs. 1000 kr'; TITLE2 '-----'; TITLE3 ' Varenr. Februar Jan-Feb Varenr. Februar Jan-Feb'; TITLE4 '-----'; TITLE5 ' '; DATA _NULL_; FILE PRINT N=PAGESIZE LINESLEFT=restlin; * kol angir posisjoner der spaltene skal starte; DO kol=1,35; linje = 0; DO until (restlin = 1) ; linje + 1; SET uriks.import; * Ønsker et punktum mellom 4. og 5. siffer i hs ; hsp = SUBSTR(hs,1,4) '.' SUBSTR(hs,5,4); PUT #linje @kol hsp \$9. v 9. vsum 9. ; END; END; PUT _PAGE_; RUN; </pre>
LRECL=record-lengde	Angir logisk recordlengde. Er recorden mer en 256 tegn (132 for Windows SAS), må LRECL (eventuelt LINESIZE) oppgis, ellers skrives resten av recorden ut på neste record (med mindre DROPOVER eller STOPOVER options er valgt). LRECL definerer fysisk recordlengde, mens LINESIZE forteller hvor mange posisjoner av recorden som maksimalt skal brukes.
MOD	Hvis filen finnes fra før vil recordene skrives ut etter siste record på filen, hvis den ikke finnes blir den laget. Se også OLD option under.
N=n PAGESIZE	Angir antall linjer som skal behandles om gangen. Brukes PAGESIZE for å angi antallet linjer, må filen vi skriver være en utskriftsfil (ikke en rådatafil). Kan brukes for å kunne behandle en utskriftsside som en helhet. Se eksempel over.
NOTAB	Sammen med option DEVICE=DDE gjør NOTAB at SAS ikke skiller kolonner som er adskilt med tabulator tegn (for Windows).
OLD	Overskriver innholdet i filen det skrives til hvis den finnes fra før. Dette er standard. Se også MOD option over.
PAD	Angir at records til rådatafilen som er kortere enn logisk recordlengde (LRECL) skal fylles ut med blanke ved skiving. Standard er at blanke ikke fylles på (NOPAD).
PAGESIZE=n	n angir sidestørrelse for utskriftsfiler, der n kan være 20 - 500 linjer.
PRINT	Angir at vi skal skrive til en utskriftsfil med skriverstyringstegn og overskrifter. NOPRINT er standard.
RECFM=recordformat	Angir recordformatet til rådatafilen. F er fast lengde, V er variabel, N er binært format og P er utskriftsformat.
STOPOVER	Indikerer at DATA-steget skal stoppes hvis SAS blir bedt om å skrive utenfor oppgitt recordlengde. Feilmelding skrives til SASlog. Se også DROPOVER og FLOWOVER options over.
TITLES	Angir at eventuelle overskrifter (definert med TITLE-setninger) skal skrives ut når vi skriver til utskriftsfil eller ei (NOTITLES).
UNBUF	Angir at SAS ikke skal skrive records i buffere (Unix).

Andre options: NBYTE= og ODS.

3.9.21 *FORMAT*

Binder utskriftsformater til variable i et datasett. Når denne bindingen er utført, vil utskriftsformatene automatisk bli brukt når variablene brukes senere, men de opprinnelige verdiene vil bli lagret på datasettet. Det betyr f.eks. at vi automatisk får ut tekster istedenfor koder når vi lager frekvenslister. Dette høres kanskje besnærende ut, men på grunn av bivirkningene bør vi være forsiktige med å koble utskriftsformater til variable. Problemer kan oppstå når andre skal bruke datasettene. Hvis utskriftsformatene ikke sendes med, vil mottageren få problemer med å lese datasettet (kan overstyres med SAS system option `nofmterr`). Videre vil det være vanskelig å utføre IF- og WHERE-setninger på datasettet når vi ikke vet hvilke koder som ligger bak utskriftsformattekstene.

Vi kan fjerne denne bindingen hvis vi ikke ønsker den lenger. Det gjøres ved en nytt FORMAT-setning der vi ramser opp variablene uten utskriftsformater (FORMAT-setningen må komme etter SET-setningen). Dette vil løse opp bindingen.

Vær klar over at FORMAT-setningen har en annen mening i DATA-steget enn i PROC-steget. I DATA-steget bindes et utskriftsformat til en variabel og dette opprettholdes helt til vi opphever bindingen i en ny FORMAT-setning i et annet DATA-steg. Når vi bruker FORMAT-setningen i PROC-steget vil bindingen bare gjelde utførelsen av prosedyren.

FORMAT variable <utskriftsformat> <variable <utskriftsformat>> <DEFAULT=standard-utskriftsformat> ;

3.9.22 *GO TO*

Vi kan bruke GO TO for å hoppe til et bestemt punkt (label) i programmet. Utstrakt bruk av GO TO vil gjøre programmet uoversiktlig, vi bør derfor prøve å unngå å bruke GO TO, som oftest kan vi bruke DO eller IF isteden. GO TO ligner på LINK, forskjellen ligger i hvordan en eventuelt RETURN-setning utføres. RETURN etter en LINK-setning returnerer til neste setning etter LINK, mens ved bruk av GO TO og RETURN fortsetter programmet med neste iterasjon av første SAS-setning i DATA-steget (med mindre en LINK-setning kommer før GO TO, da fortsetter programmet etter LINK).

GO TO label ;

3.9.23 *IF, subsetting*

Subsetting IF brukes hvis bare noen av de observasjonene som leses inn i DATA-steget skal være med i datasettet som lages. Subsetting IF minner om WHERE-setningen, men WHERE er begrenset til å omfatte variable som leses fra SAS datasett. Andre forskjeller mellom WHERE og subsetting IF er også beskrevet i forbindelse med WHERE-setningen på side 91.

IF betingelse ;

Betingelsen må være et SAS-uttrykk, se side 34. Kun de observasjoner der betingelsen er oppfylt blir behandlet videre. Når betingelsen ikke er oppfylt, går SAS til neste observasjon.

Eksempel:

```
DATA preg ;
  INPUT @01 fnr      $CHAR11.
        @12 kommune $CHAR4.
        @16 bydel   $CHAR2.
        ;
  * Vi ønsker bare å ta med de med gyldig kjønnskode i fødselsnummeret;
  IF '0' <= SUBSTR(fnr,9,1) <= '9';
CARDS;
01014531131030107
301054      030122
23046230045030105
24115334100030103
02087245555160104
02087143138030122
13126542319030122
040496134610211
;
```

3.9.24 IF, THEN, ELSE

En IF-THEN-konstruksjon bruker vi for bare å utføre en SAS-setning når en eller flere betingelser er oppfylt. Vi kan utføre flere SAS-setninger når én betingelse er oppfylt, da må SAS-setningene avgrenses av en DO-END konstruksjon, se side 61. Istedenfor å bruke IF-setninger kan det ofte lønne seg å bruke SELECT, se side 86. Dette gjelder spesielt for å unngå å få mange IF THEN ELSE inne i hverandre.

IF betingelse **THEN** SAS-setning ;
<ELSE SAS-setning ;>

Betingelsen som skal oppfylles for at SAS-setningen skal utføres må være et SAS-uttrykk, se side 34.

Eksempler:

```
IF skatt > inntekt THEN
  PUT 'Feil??. Skatt større enn inntekt ' skatt= inntekt= ;

IF kommune = '0301' THEN
  OUTPUT oslo;

IF SUBSTR(kommune,1,2) = '02' THEN
  OUTPUT akershus;

IF fdato > '31DEC1995'D THEN
  PUT 'Født etter statistikkår!' fdato= DDMYY6. ;

IF lovbr IN ('075','076','331','332') THEN
  forbr = 'Narko';

IF kjonn = '1' THEN OUTPUT menn;
ELSE
  IF kjonn = '2' THEN OUTPUT kvinner ;
  ELSE
    OUTPUT uoppg;

IF import < 1000 THEN
  DELETE;

IF 0 < inntekt < 100000 THEN
  OUTPUT lavlonn;

IF vklo = "1" AND arbsit NOT IN('1','2') THEN
  PUT 'Har arbeid, men jobber ikke? ' vklo= arbsit=;

IF SUBSTR(kommune,1,2) || bydel = '0322' THEN
  OUTPUT sogn;

IF inntekt > 100000 THEN
  IF gjeld > 0 THEN
    klasse=1;
  ELSE
    ;
ELSE
  klasse=2;
```

3.9.25 %INCLUDE

Sørger for å hente inn SAS-setninger fra en eller flere programfiler. Egner seg bra når vi har ett sett med SAS-setninger som skal brukes i flere program. Da legges disse ut på en egen fil, som inkluderes med %INCLUDE.

%INCLUDE filnavn-1 ... <filnavn-n> </SOURCE2> <S2=lengde> <ANSI | OEM>

filnavn	navn på program som skal inkluderes
SOURCE2	angir at inkluderte linjer skal skrives ut til kjørerapporten. Kan være nyttig å bruke.
S2	angir recordlengde på filen som skal inkluderes
ANSI OEM	angir for Windows SAS om filen skal lese som en ansi eller OEM (ascii-fil)

Eksempel:

```
DATA rev.odoslo01;
  INFILE '$BARNVERN/work/data/odoslo01.dat' PAD MISSOVER LRECL=1310;

  * Henter inn beskrivelse av sekvensiell fil;
  %INCLUDE 'bv94.sas' ;

  refnr = _N_ ;

  * Henter inn omkodingene ;
  %INCLUDE "omkod.sas" / SOURCE2;

PROC SORT DATA=rev.odoslo01;
  BY komm dist fnr;
RUN;
```

3.9.26 INFILE

Vi bruker INFILE til å identifisere en rådatafil som skal leses inn til SAS. Med INFILE forteller vi først og fremst hva filen heter. Dersom det er nødvendig, kan vi også gi mer informasjon om filen i INFILE-setningen. For å lese filen, må vi bruke INPUT-setningen, INFILE forteller hva filen heter etc. Når vi leser inn sekvensielle filer, bør vi alltid ha med MISSOVER, PAD og LRECL-options, uten disse kan innlesingen bli feil.

INFILE fil-navn | fil-referanse | CARDS < options > ;

fil-navn	navn på rådatafilen som skal leses inn
fil-referanse	kortnavn til rådatafilen angitt med FILENAME-setningen. FILENAME, se side 97.
CARDS	angir at rådatafilen er inkludert i programmet med en egen CARDS-setning

Aktuelle options til INFILE (de vanligste er uthevet):

BLKSIZE=	angir blokkstørrelsen på rådatafilen.
COLUMN=variabel	definerer en hjelpevariabel som SAS legger pekerens posisjonen i på recorden som leses fra rådatafilen. Denne endres hver gang pekeren starter på et nytt felt i recorden. Kan forkortes til COL=.
DELIMITER=	Angir skilletegn for feltene i rådatafilen. Det er vanlig å ha et skilletegn, men vi kan velge flere skilletegn. To skilletegn etter hverandre i rådatafilen blir vanligvis regnet som ett. Ønsker vi at etterfølgende skilletegn ikke skal regnes som ett, må vi bruke DSD option, se denne. Skilletegnet kan ligge i en variabel som benyttes i DELIMITER option. Er skilletegnet tabulator, skriver vi slik: DELIMITER='09'X. DELIMITER kan forkortes til DLM. Se for øvrig mer om skilletegnseparerte filer på side 8.
DEVICE=	DDE for DDE-koblinger mot regneark etc. DISK for vanlig lesing fra disk. Dette er standard.

DSD	Endrer behandlingsmåte for skilletegnseparerte filer. Gjør at tekster som står i fnutter (enkle eller doble) leses uten fnuttene. Gjør også at flere skilletegn etter hverandre ikke slås sammen til et skille, det betyr at tomme felt ikke blir oversett og vi unngår at felt fra rådatafilen havner i feil variable. Når vi bruker DSD option, settes DELIMITER til komma, med mindre vi overstyrer dette med DELIMITER option. Se for øvrig mer om skilletegnseparerte filer på side 8.
END=variabel	definerer en hjelpevariabel som settes til 1 når den siste recorden i rådataene leses, ellers er verdien 0. Denne er nyttig å bruke når vi ønsker å utføre spesielle ting etter at siste record er lest inn. END option virker ikke ved bruk av CARDS eller UNBUFFERED option. Eksempler, se side 17, 23 og 134.
EOF=label	Angir at SAS skal gå til en label når den siste record i rådatafilen er behandlet. Kan være nyttig ved lesing av flere datafiler samtidig for å hoppe over behandling en fil som ikke har flere records.
EOV=variabel	Definerer en hjelpevariabel som settes til 1 for den første record i en fil i en serie rådatafiler som skal konkateneres. Filene som skal konkateneres må være definert i en FILENAME-setning. EOV-variabelen settes til 1 ved begynnelsen av hver lest fil unntatt den første. Den settes ikke tilbake til 0, derfor må vi selv gjøre det.
EXPANDTABS	Angir at tabulatorsetting skal brukes. Det betyr tabulator for hver 8. posisjon med start i posisjon 9. NOEXPANDTABS er standard.
FILENAME=variabel	Bevirker at navnet på rådatafilen legges i en hjelpevariabel.
FILEVAR=variabel	Definerer en hjelpevariabel som inneholder navnet på den rådatafilen som skal leses.
FIRSTOBS=rec-nr	Angir recordnummer til den første record som skal leses
FLOWOVER	Angir at SAS skal fortsette å lese på neste record hvis ikke alle variable for en observasjon får verdier ved innlesing fra en record i rådatafilen. Dette inntreffer når de siste felt på en linje er blanke. Det betyr at felt fra 2 eller flere records i rådatafilen kan havne i samme observasjon på SAS datasettet som lages. FLOWOVER er standard, dette kan skape uventede resultater. Resultatet av kjøringen vil ofte være annerledes enn forventet og derfor gir SAS en melding. Sjekk derfor resultatet nøye når denne meldingen kommer i SASlog: SAS went to a new line when INPUT statement reached past the end of a line.
INFILE=variabel	For å hindre at SAS går til en ny linje uten at vi vil det, kan vi bruke option MISSOVER, se denne. Se også INPUT-setningen, side 73.
	Angir en hjelpevariabel som vil inneholde inndatabufferet, dvs. det som leses inn pr. INPUT-setning. Tømmes før ny linje skal leses inn.
LENGTH=variabel	Definerer en hjelpevariabel som inneholder lengden til den recorden SAS leser fra rådatafilen.
LINE=variabel	Definerer en hjelpevariabel som inneholder det relative nummeret innen en gruppe records definert med N= option eller #n peker kontroll (se INPUT-setningen, side 73) til den recorden SAS leser fra rådatafilen.
LINESIZE=linjebredde	Avgrenser lesing av records fra rådatafilen til det antall posisjoner som oppgis.
LRECL=record-lengde	Angir logisk recordlengde. Er recorden mer en 256 tegn (132 for Windows SAS), må LRECL (eventuelt LS) oppgis, ellers kuttet recorden.
MISSOVER	Hindrer at SAS går til neste linje i rådatafilen hvis ikke alle variablene får verdier ved lesing på den aktuelle linjen. Dette inntreffer når de siste posisjoner på en linje er blanke. Variable som ikke har fått verdi settes til MISSING. Bruk heller TRUNCOVER-option.
N=n	Angir antall linjer som skal behandles om gangen. Kan brukes ved listeinput når hvert felt ligger på en egen record og ingen records eller felt er blanke. Utgjør flere records i rådatafilen det som skal bli én observasjon i SAS-datasettet og feltene ligger i faste posisjoner bør vi heller bruke / option i INPUT-setningen, se side 73.

Eksempel der hver observasjon skal bestå av to innleste records:

```

OPTIONS NODATE NONUMBER NOCENTER;
DATA dode93;
INFILE CARDS N=2;
INPUT mnd $
      antdode
      ;
CARDS ;
JAN93
3985
FEB93
3520
MAR93
3917
APR93
3738
MAI93
3802
JUN93
3619
JUL93
3606
AUG93
3486
SEP93
3579
OKT93
3732
NOV93
3990
DES93
5649
;

PROC PRINT DATA=dode93;
  TITLE 'Antall døde fordelt etter måned';
run;

```

Dette programmet gir følgende resultat:

Antall døde fordelt etter måned		
OBS	MND	ANTDODE
1	JAN93	3985
2	FEB93	3520
3	MAR93	3917
4	APR93	3738
5	MAI93	3802
6	JUN93	3619
7	JUL93	3606
8	AUG93	3486
9	SEP93	3579
10	OKT93	3732
11	NOV93	3990
12	DES93	5649

NOTAB	Sammen med option DEVICE=DDE gjør NOTAB at SAS ikke skiller kolonner som er adskilt med tabulator tegn.
OBS =rec-nr	Angir recordnummer til den siste record som skal leses
PAD	Angir at records fra rådatafilen som er kortere enn logisk recordlengde (LRECL) skal fylles ut med blanke ved innlesing. Standard er at blanke ikke fylles på (NOPAD). Bruk heller TRUNCOVER-option
PRINT	Angir om rådatafilen inneholder skriverstyringstegn. Med PRINT option vil disse automatisk overses. NOPRINT er standard.
RECFM=record-format	Angir recordformatet til rådatafilen. F er fast lengde, V er variabel, N er binært format og P er format for utskrift.
SCANOVER	Gjør at INPUT-setningen leter gjennom dataene helt til teksten angitt med '@tekst' finnes i inndataene.
SHAREBUFFERS	Sørger for at SAS bruker mindre CPU-tid når det skal leses fra en rådatafil og deretter skrives ut til en rådatafil i samme datasteg uten å lage SAS datasett. SHAREBUFFERS vi forhindre at data fra INPUT-setningen kopieres til et output buffer og sparer derved CPU-tid.

START=variabel	Definerer en hjelpevariabel som skal inneholde startposisjonen for hvor på recorden som leses fra rådatafilen utskrift med SAS-setningen PUT _INFILE_ skal starte. En slik PUT-setning vil vanligvis skrive ut hele recorden, men ønsker vi å skrive ut recorden fra posisjon 41 og utover isteden, må angi et variabelnavn i START option (f.eks. START=startpos). Denne variabelen må gis verdi før PUT-setningen eksekveres. Eksempel der vi kopierer en dødsårsaksfil, men utelater personnavn på utfilen.
	<pre>DATA _NULL_ ; INFILE ' \$DODARSAK/aar/dode.g93' START=startpos MISSOVER PAD LRECL=143; INPUT; startpos = 41; FILE 'dode93' ; PUT _INFILE_ ; RUN;</pre>
STOPOVER	Indikerer at DATA-steget skal stoppes hvis SAS når slutten av en record fra rådatafilen uten å få gitt alle variable i INPUT-setningen en verdi. Feilmelding skrives til SASlog. SAS datasett blir ikke laget eller erstattet når STOPOVER trer i funksjon. Se også FLOWOVER og MISSOVER options.
TRUNCOVER	Gjør at vi kan ha records på innfilen som er kortere enn angitt i INPUT-setningen uten at SAS går til neste linje i datafilen for å finne data. Minner om MISSOVER-option, men mens MISSOVER setter en variabel til missing hvis ikke hele feltet den skal fylles fra er utfylt, vil TRUNCOVER ta det som er fylt ut. TRUNCOVER-option bør som regel være med ved innlesing av rådatafiler, fordi den omvendte option FLOWOVER er standard, men sjelden ønskes utført. FLOWOVER overstyrer TRUNCOVER dersom begge options benyttes. Variable som ikke har fått verdi settes til MISSING. TRUNCOVER tilsvarer kombinasjonen av MISSOVER- og PAD- options, men er mer maskineffektiv.
UNBUFFERED	Angir at SAS ikke skal lese records i buffere. Denne medfører at en eventuell END= parameter aldri blir 1. UNBUFFERED option brukes automatisk når CARDS er i bruk.

3.9.27 INFORMAT

Brukes når variable som leses fra sekvensielle filer er formaterte. Vi må fortelle SAS hvordan variablene er formatert. I motsetning til FORMAT-setningen, vil variablenes formaterte verdi lagres på datasettet. Dette gjør at vi ikke får tilbake igjen den koden som var på den sekvensielle filen som ble lest inn hvis vi fjerner innlesingsformatet. For øvrig ligner INFORMAT-setningen på FORMAT-setningen, se side 66.

INFORMAT variable <innlesingsformat> <variable <innlesingsformat>>
 <DEFAULT=standard-innlesingsformat> ;

3.9.28 INPUT

Når vi lager SAS datasett fra rådata bruker vi INPUT-setningen for å beskrive rådataene. Det er forskjellige måter å gjøre dette på, f.eks.:

Formatert:	Vi oppgir peker, startposisjon, variabelnavn og innlesingsformat (felt i faste posisjoner, uten skilletegn) <pre>INPUT @01 kommune \$CHAR4. @05 fnr \$CHAR11. @16 inntekt 11. ;</pre>
Kolonne:	Vi oppgir variabelnavn, variabeltype (karakter el. numerisk) og start- og sluttposisjon (felt i faste posisjoner, uten skilletegn) <pre>INPUT kommune \$ 1-4 fnr \$ 5-15 inntekt 16-26 ;</pre>
Liste:	Vi oppgir variabelnavn og variabeltype (felt med varierende plassering og skilletegn mellom hver variabel) <pre>INPUT kommune \$ fnr \$ inntekt ;</pre>
Formatert liste:	Vi oppgir variabelnavn, kolon, og innlesingsformat (felt med varierende plassering og skilletegn mellom hver variabel) <pre>INPUT kommune : \$CHAR4. fnr : \$CHAR11. inntekt : 11. ;</pre>

For å få best mulig kontroll over innlesingen bør vi bruke formatert innlesing, med startposisjon for filer med felt i faste posisjoner og formatert listeinnlesing for filer med feltskilletegn.

Datafilen som skal leses inn må være definert med en INFILE-setning, se side 69. Til testing kan vi bruke CARDS-setningen istedenfor INFILE, se side 58.

```
INPUT <peker-1> variabel-1 <? | ??> innlesingsformat. ...<<peker-n> variabel-n innlesingsformat.>  
      <@ | @@> ;  
INPUT variabel-1 <? | ??> <$> startposisjon-1 <- sluttposisjon-1> <.desimaler>  
... <variabel-n <? | ??> <$> startposisjon-n <- sluttposisjon-n> <.desimaler> > <@ | @@> ;  
INPUT <peker-1> (variabel-liste-1) (<? | ??> innlesingsformat-liste-1)  
... <<peker-n> (variabel-liste-n) (<? | ??> innlesingsformat-liste-n)> <@ | @@> ;  
INPUT <peker-1> (variabel-liste-1) (<? | ??> <n*> innlesingsformat.)  
...<<peker-n> (variabel-liste-n) (<? | ??> <n*> innlesingsformat.)> <@ | @@> ;  
INPUT <peker-1> variabel-1 <? | ??> <$> <&> ...<<peker-n> variabel-n <? | ??> <$> <&>> <@ | @@> ;  
INPUT <peker-1> variabel-1 <? | ??> <: | &> <innlesingsformat.>  
...<<peker-n> variabel-n <? | ??> <: | &> <innlesingsformat.>> <@ | @@> ;  
INPUT <peker-1> variabel-1= <? | ??> <$> ...<<peker-n> variabel-n= <? | ??> <$>> <@ | @@>;  
INPUT variabel-1= <? | ??> <$> startposisjon-1 <- sluttposisjon-1> <.desimaler>  
... <variabel-n= <? | ??> <$> startposisjon-n <- sluttposisjon-n> <.desimaler>> <@ | @@> ;  
INPUT <peker-1> variabel-1= <? | ??> innlesingsformat.  
...<<peker-n> variabel-n= <? | ??> innlesingsformat.> <@ | @@> ;  
INPUT <peker-1> array-navn(*) <@ | @@> ;
```

\$ angir at vi skal lage en karaktervariabel. Utelates \$ lages det en numerisk variabel.
peker angir pekertegn og posisjon for lesing av inndata. Har tallet som angir posisjon eller flytting desimaler, trunkeres disse. Er tallet 0 eller negativt, flyttes lesehodet til posisjon 1. Ender lesehodet utenfor recordlengden, flyttes det til posisjon 1 i neste record. Flytter vi bakover vil vi aldri kunne komme lenger tilbake enn til posisjon 1 på recorden. Når vi bruker tekster for å flytte lesehodet og teksten ikke finnes på recorden, søkes det på neste record. Slik fortsetter søket helt til teksten er funnet eller slutten på inndatafilen er nådd.

Pekere:

@n	n angir posisjonen der lesehodet skal plasseres
@variabel	når variabelen er numerisk skal den inneholde posisjonen der lesehodet skal plasseres og når den er en karaktersvariabel skal teksten den inneholder befinne seg på inndataene slik at lesehodet kan plasseres rett etter denne.
@(SAS-uttrykk)	Er SAS-uttrykket et tall flyttes lesehodet til denne posisjonen. Når resultatet av uttrykket er en tekst flyttes lesehodet til posisjonen etter der teksten befinner seg i inndataene.
@'tekst'	flytter lesehodet til posisjonen etter der den angitte tekst befinner seg i inndataene.
+n	flytter lesehodet n posisjoner.
+variabel	variabelen skal være numerisk og inneholde det antall posisjoner lesehodet skal flyttes
+(SAS-uttrykk)	SAS-uttrykket skal være numerisk og inneholde det antall posisjoner lesehodet skal flyttes.
#n	angir at lesehodet skal flyttes til record n i en gruppe records
#variabel	variabelen skal være numerisk og inneholder hvilken record i en gruppe records lesehodet skal flyttes til
#(SAS-uttrykk)	SAS-uttrykket skal være numerisk og inneholde hvilken record i en gruppe records lesehodet skal flyttes til
/	flytter lesehodet til posisjon 1 på neste linje i inndataene

- ? angir at det ikke skal skrives ut melding når det er feil format i inndataene. Inndataene og variabelverdier listes ut i SASlog. Den automatiske hjelpevariabelen `_ERROR_` settes til 1.
- ?? angir at det ikke rapporteres i det hele tatt når det er feil format i inndataene. Den automatiske hjelpevariabelen `_ERROR_` forblir 0.
- @ Forhindrer at INPUT-setningen flytter lesehodet til neste record når det avsluttes. Det leses ikke inn noen ny record og neste INPUT-setning vil behandle den samme recorden. Denne holdingen oppheves når en INPUT-setning som ikke holder recorden eksekveres (uten @ eller @@) eller når neste iterasjon av DATA-steget utføres. Vi kan altså ha en ny INPUT-setning som leser den samme recorden.

Eksempel der vi leser inn en variabel først for å se om vi ønsker å lese inn flere felt. Det kan spare tid når vi skal lese store sekvensielle filer der vi ikke er interessert i alle records:

```
DATA persreg;
INFILE 'persreg.dat' MISOVER PAD;
INPUT @10 kjonn $CHAR1. @;
IF kjonn = 'K' THEN
DO;
    INPUT @01 komm $CHAR4.
           @06 gkrets $CHAR3.
           @12 aldgrp $CHAR4.
           @18 skatt 6.
           @25 inntekt 7.
    ;
    OUTPUT;
END;
```

- @@ Holder recorden slik at den kan gjennomgå flere iterasjoner av DATA-steget. Holdingen frigis når lesehodet når slutten på recorden, eller ved en ny INPUT-setning som ikke holder recorden som eksekveres. Eksempel:

```
DATA dode93;
INFILE CARDS ;
INPUT mnd $ antdode @@ ;
CARDS ;
JAN93 3985 FEB93 3520 MAR93 3917 APR93 3738
MAI93 3802 JUN93 3619 JUL93 3606 AUG93 3486
SEP93 3579 OKT93 3732 NOV93 3990 DES93 5649
;

PROC PRINT DATA=dode93;
TITLE 'Antall døde fordelt etter måned';
RUN;
```

Resultatet:

Antall døde fordelt etter måned

OBS	MND	ANTDODE
1	JAN93	3985
2	FEB93	3520
3	MAR93	3917
4	APR93	3738
5	MAI93	3802
6	JUN93	3619
7	JUL93	3606
8	AUG93	3486
9	SEP93	3579
10	OKT93	3732
11	NOV93	3990
12	DES93	5649

n* angir at et innlesingsformat skal gjentas n ganger (for n variable). Eksempel:

```
DATA tilbfall ;
  INPUT (fnr ddl-dd7) ($CHAR11. 7*YYMMDD6.);
CARDS;
01014531131870201870402881112890131920129931213
30105441335871211871223880412880824 . . .
23046230045870228870402 . . .
24115334100870731 . . .
0208724555870803931215 . . .
02087143138871212871230880106 . . .
13126542319871209880413920201920208921130930202931111
;
PROC PRINT DATA=tilbfall ;
  TITLE 'Utlisting av dommer';
  FORMAT ddl-dd7 DDMMYY6.;
RUN;
```

gjør at vi kan bruke innlesingsformater på inndata som ikke har feltplasseringer i faste posisjoner. Med kolon foran innlesingsformatet, vil SAS lese et felt til lesehodet kommer til det neste blanke tegn i recorden eller lesehodet kommer til enden av recorden, avhengig av hva som skjer først. Selv om det leses helt til neste blanke tegn, trunkeres verdien som skal lagres i variabelen dersom den er lenger enn innlesingsformatets lengde.

Eksempel:

```
DATA persreg;
  INPUT komm : $CHAR4.
        gkrets : $CHAR3.
        kjonn : $CHAR1.
        aldgrp : $CHAR5.
        inntekt : 6.
        skatt : 6.
;
CARDS ;
0101 001 M ald1 317327 145102
0101 001 M ald2 122341 34213
0101 001 M ald3 213340 .
. . M ald4 141341 24029
0101 001 M ald5 2313 .
0101 001 M ald1 442341 187292
0101 001 K . 355345 67326
0101 001 K ald2 242351 78219
0101 001 K ald3 116341 13232
0101 001 K ald4 326340 134216
. 008 M ald1 248118 87631
0101 001 K ald5 442341 234145
;
```

& En ampersand etter en variabel, gir beskjed om at variabelen kan inneholde en blank. SAS vil lese data inn til variabelen helt til 2 blanke tegn etter hverandre finnes. Det betyr at felt som leses kan inneholde et blankt tegn en eller flere ganger (men ikke etter hverandre). SAS leser helt til 2 blanke tegn kommer etter hverandre eller til slutten av recorden nås. Er antall posisjoner flere enn bredden på variabelen eller et eventuelt innlesingsformat, trunkeres verdien.

Eksempel:

```
DATA adr ;
INPUT      navn      & $CHAR30.
           adresse   & $CHAR40.
           postnr    & $CHAR4.
           poststed  & $CHAR25.
           ;

CARDS;
RÅDE KOMMUNE  SOSIALKONTORET 1640 RÅDE
KONGSVINGER SOSIALKONTOR  SKOLEGT. 1 2200 KONGSVINGER
HAMAR SOSIALKONTOR  POSTBOKS 300 2301 HAMAR
RINGSAKER KOMMUNE  SOSIALKONTORET 2380 BRUMUNDDAL
LØTEN KOMMUNE  SOSIALKONTORET 2340 LØTEN
STANGE KOMMUNE  SOSIALKONTORET 2310 STANGE
NORD-ODAL KOMMUNE  SOSIALKONTORET 2120 SAGSTUA
NØTTERØY SOSIALKONTOR  . 3140 BORGHEIM
TJØME SOSIALKONTOR  POSTBOKS 55 3145 TJØME
;
PROC PRINT DATA=adr;
  TITLE 'To blanke mellom hvert felt';
RUN;
```

3.9.29 KEEP

Brukes for å ta med variable på utdatasett. De variablene som skal være med på utdatasett, listes med KEEP-setningen. KEEP tar med variablene på alle datasett som lages, i motsetning til KEEP-option til SAS datasett, som bare gjelder for et datasett. KEEP brukes når vi skal ta med noen få variable. Skal vi ta med mange variable og bare utelate noen få, bruker vi DROP isteden, se side 62.

KEEP variable ;

Eksempler:

```
KEEP kommune fnr aar_91-aar_95 ;

KEEP fnr dom: ; /* Beholder fnr og alle variable som begynner med dom */
```

3.9.30 kommentar

Når vi programmerer er det viktig å ha med kommentarer. SAS har to forskjellige SAS-setninger for kommentarer:

```
/* tekst */
* tekst ;
```

Den første av disse to er den eneste SAS-setningen som ikke avsluttes med semikolon. All tekst skrevet mellom tegnene /* og */ blir regnet som en kommentar. Denne måten å kommentere på er fin å bruke når vi ønsker å utelate SAS-setninger fra et program uten å slette dem. Plasserer vi SAS-setningene mellom disse tegnkombinasjonene, blir de regnet som en kommentar. Dette kaller vi å kommentere vekk SAS-setninger. Når vi kommenterer vekk SAS-setninger er det viktig at denne type kommentartegn ikke finnes i den delen av programmet vi skal kommentere vekk, det vil føre til feil. Hvis de finnes, kan vi enten fjerne dem eller bruke RUN-setningen med option CANCEL isteden, se side 108.

Til kortere kommentarer er den andre syntaksen grei. Vi må bare være forsiktige med å kommentere vekk IF-setninger, da disse ofte har sitt semikolon plassert etter andre SAS-setninger, noe som kan føre til feil i programmet. Vi bør heller ikke plassere slike kommentarer etter THEN i IF-setninger.

Eksempel 1:

Alle program som skal bevares bør ha en kommentarblokk øverst. Den bør se ut omtrent som denne:

```
/******  
Prosjekt .....: Sosialhjelp  
Programnavn .....: $SOSHJELP\prog\mott96.sas  
Skrevet av .....: Kristian Lønø, 303  
Dato .....: 29.4.1998  
Versjon .....: 1.0  
Programmets funksjon ..: Lager SAS-datasett fra mottakerfil 1996  
Anmerkning.....:  
Programmet kaller .....: skjem94.sas og nyvar95.sas  
Endret når .....:  
Endret av .....:  
Grunn til endring .....:  
*****/
```

Eksempel 2:

Her kommenterer vi hvorfor vi har en IF-setning:

```
* Feil dato, skal ha feilmelding;  
IF fdato > '31DEC1995'D THEN  
DO;  
  PUT 'Født etter statistikkår! ' fdato= DDMMYY6.;  
  DELETE;  
END;
```

Vi må passe på å ha kommentaren før IF-setningen, kommentaren under fører til **syntaksfeil!**

```
IF fdato > '31DEC1995'D THEN  
* Feil dato, skal ha feilmelding;  
DO;  
  PUT 'Født etter statistikkår! ' fdato= ddmmyy6.;  
  DELETE;  
END;
```

Skal vi kommentere vekk IF-setningen må vi gjøre det slik:

```
/*  
IF fdato > '31DEC1995'D THEN  
DO;  
  PUT 'Født etter statistikkår! ' fdato= ddmmyy6.;  
  DELETE;  
END;  
*/
```

Prøver vi varianten under, får vi **syntaksfeil**:

```
* IF fdato > '31DEC1995'D THEN  
DO;  
  PUT 'Født etter statistikkår! ' fdato= ddmmyy6.;  
  DELETE;  
END;
```

Vi må sette inn et semikolon før DO-setningen (programmet blir ikke logisk, men det er ikke viktig i denne forbindelse):

```
* IF fdato > '31DEC1995'D THEN  
;  
DO;  
  PUT 'Født etter statistikkår! ' fdato= ddmmyy6.;  
  DELETE;  
END;
```


Eksempel 3:

Når vi ikke ønsker at DATA-steget skal utføres, men bare tabellprogrammet, kommenterer vi vekk DATA-steget istedenfor å slette det fra programmet:

```

/*
DATA a3694;
  SET tilbfall.a3694 (KEEP= tilbf teller alder reak51 antrf kjonn);
  SELECT (reak51);
    WHEN ('01','07','18','19','43') reak = 'A';
    WHEN ('02','13') reak = 'B';
    WHEN ('03') reak = 'C';
    WHEN ('04') reak = 'D';
    WHEN ('14') reak = 'E';
    WHEN ('05','17','60') reak = 'F';
    WHEN ('52','61') reak = 'G';
    OTHERWISE reak = 'Z';
  END;
*/
PROC TABULATE DATA=a3694 NOSEPS FC='-----' F=6. MISSING;
  VAR tilbf teller;
  CLASS reak alder antrf kjonn;
  TABLE
    reak='*(all=' alder=')*'
    (tilbf='*pctsum<teller>=')
    ,
    (ALL='Begge kjønn' kjonn='')*
    (ALL='I alt' antrf='Tidligere reaksjoner')
  /BOX='Utgangsreaksjon og alder'
  RTS=42 ROW=float MISSTEXT='-' INDENT=0;
  FORMAT alder alder.
    antrf anttbf.
    reak $reak.
    kjonn $kjonn.
  ;
  TITLE 'Tabell 8. Tilbakefallsprosent for straffbare.
RUN;

```

3.9.31 LABEL

Brukes for å forklare innholdet i en variabel. Alle variable vi lager bør ha en LABEL. Dette gjør det lettere for oss å lese programmet senere, samtidig som teksten som brukes LABEL kan benyttes i programmene våre. En LABEL kan maksimalt være 40 posisjoner.

LABEL variabel-1 = 'Utfyllende tekst variabel 1'

```

...
<variabel-n = 'Utfyllende tekst variabel n'>
;

```

3.9.32 label

Identifiserer et punkt (label) i programmet som en annen SAS-setning skal hoppe til. Brukes vanligvis sammen med en GO TO, LINK, FILE (i HEADER option) eller INFILE (i EOF option) setning. Vi må huske på kolon på slutten av label.

label: SAS-setning ;

Eksempel der vi bruker label for å lage kolonneoverskrifter til en liste vi skriver ut vha DATA-steget:

```

DATA _NULL_;
  SET innbygg;
  FILE PRINT HEADER=hode ;
  PUT @2 kommune $4.
    @13 folkialt 7.
  ;
  RETURN;
hode:
  PUT 'Kommune Befolkning' / ;
  RETURN;
RUN;

```

3.9.33 LEAVE

Brukes i DO-løkker og SELECT for å gå ut av løkken.

LEAVE ;

Eksempel:

```
DATA bvern (DROP=i);
  ARRAY p18t (*) $ p18t_1 - p18t_5 ;
  ARRAY p29t (*) $ p29t_1 - p29t_5 ;
  INPUT fnr
         p18t(*) $CHAR2.
         p29t(*) $CHAR2.;
  * Hvis et av feltene p18t eller p29t er fylt ut er det en barneverns klient ;
  DO i = 1 TO DIM(p18t);
    IF p18t(i) > '00' or p29t(i) > '00' THEN
      DO;
        bvk = '1';
        LEAVE;
      END;
    END;
  CARDS;
01019231131 00020000050000000000
30108741335 00000000000000000000
23047930045 01020300050000000005
24119434100 00000000000100000000
02089545555 00000000000002000000
02088143138 00000000000000000000
13128242319 00000004000000000000
;
PROC PRINT DATA=bvern;
  TITLE 'Lister ut barnevernsdataene';
RUN;
```

I dette programmet trenger vi bare å utføre DO-løkken inntil første gang IF-testen i løkken slår til. Når denne har slått til én gang, forlater vi løkken med LEAVE-setningen.

3.9.34 LENGTH

Angir lengden til en eller flere variable og om de skal være av type numerisk eller karakter. Numeriske variable kan ha lengde 3-8 (standard er 8, kan endres med DEFAULT parameteren som angir ny standard lengde for numeriske variable) mens karaktervariable (angis med \$) kan være 1-200 (standard er 8 ved listeinput fra rådata, antall posisjoner ved kolonneinput fra rådata, formatbredde ved formatert innlesing og lengden til verdien ved første gangs tilordning ellers). Det er som oftest greit å ha lengde 8 på numeriske variable. Karaktervariable må vi ofte angi lengde på for at denne skal bli riktig. Når lengden og typen til en variabel er definert, kan de ikke omdefineres.

LENGTH <variabel-1 ... <variabel-1n> <\$> lengde>
... <variabel-2 ... <variabel-2n> <\$> lengde > > <DEFAULT=n>;

```
LENGTH kommune $ 4 statsb lovbr1-lovbr8 $ 3 ;
/* kommune får lengde 4 og de andre lengde 3, alle blir karaktervariable */
```

3.9.35 LINK

Vi kan bruke LINK for å hoppe til et bestemt punkt (label) i programmet. Utstrakt bruk av LINK vil gjøre programmet uoversiktlig, vi bør derfor prøve å unngå å bruke LINK, som oftest kan vi bruke DO eller IF isteden. LINK ligner på GO TO, forskjellen ligger i hvordan en eventuell RETURN-setning utføres. RETURN etter en LINK-setning returnerer til neste SAS-setning etter LINK, mens ved bruk av GO TO og RETURN fortsetter programmet med neste iterasjon av første SAS-setning i DATA-steget (med mindre en LINK-setning kommer før GO TO, da fortsetter programmet etter LINK).

LINK label ;

3.9.36 LIST

Skriver inndata til SASlog. Gjelder bare data lest inn med INPUT-setningen. Kan f.eks. brukes til å liste ut inndata som inneholder ugyldige verdier. LIST skriver ut ved slutten av en iterasjon av DATA-steget, i motsetning til PUT, som skriver ut med en gang.

LIST ;

3.9.37 LOSTCARD

Når en rådatafil består av et sett med records for hver ident, leser vi dem inn i grupper av records. Hver av disse gruppene må ha samme antall records. Dette antallet bruker vi i INPUT-setningen. Hvis det skjer at det for en gruppe ikke er det foreskrevne antall records på rådataene, vil innlesingen komme i utakt. For å komme i takt igjen når slike situasjoner oppstår, bruker vi LOSTCARD-setningen. Bruker vi LOSTCARD, gis det en melding om det i SASlog. De gruppene av records som ikke har riktig antall, blir ikke skrevet til SAS datasettet. Videre vil den automatiske variabelen `_N_` ikke økes med 1. Deretter vil SAS prøve å starte en ny gruppelesing med den siste recorden som ble lest.

LOSTCARD ;

3.9.38 MERGE

Kobler observasjoner fra to eller flere datasett. Vi bør vanligvis ikke koble mer enn to datasett om gangen, for å ha kontroll med koblingen. Kobling kan skje med og uten BY-setning. Uten BY-setning kobles en og en observasjon fra datasettene uten bruk av noen nøkkel (BY-variable). Med BY-setning kobles dataene med BY-variablene som nøkler. Det er viktig å vite hvordan kobling av datasett foregår, slik at vi sikrer oss et riktig resultat. Kobling av datasett er beskrevet på side 15.

MERGE datasett-1 <(datasettoptions-1)> datasett-2 <(datasettoptions-2)>
... <datasett-n <(datasettoptions-n)>> <END=hjelpevariabel> ;

datasett navn på datasett som skal kobles

datasettoptions options til SAS datasettet, se side 47.

END= navn på en hjelpevariabel som indikerer slutten på datasettene. END-variabelen settes til 1 når siste observasjon på alle datasett er lest, ellers er den 0. Det betyr at vi ikke kan bruke END= til å teste om vi har kommet til slutten av et av de datasettene vi kobler.

Eksempel:

Skal koble personers diagnoser til et datasett med andre personopplysninger. Vi ønsker å skrive ut meldinger når en person ikke har diagnose og når vi ikke finner personopplysninger om personer som har en diagnose. Videre ønsker vi bare å ha med de som finnes i persondatasettet (pers) på utdatasettet.

```
DATA persdiag;
  MERGE pers (IN=p)
        diag (IN=d);
  BY ident;
  IF p AND NOT d THEN
    PUT 'Mangler diagnose ' ident= kjonn= alder=;
  IF d AND NOT p THEN
    PUT 'Mangler personopplysninger!!! ' ident= diagnose=;
  IF p THEN OUTPUT;
```

3.9.39 MISSING

Kan brukes på numeriske variable for å skille mellom forskjellige grunner til at verdien er MISSING. Vil gjelde for alle numeriske variable som leses inn.

MISSING tegn-1 ... <tegn-n> ;

tegn bokstaver (a-z, A-Z) som indikerer hvilke tegn som skal være MISSING. Foruten bokstaver kan understrek brukes.

Eksempel:
I en spørreundersøkelse spør vi: Hvor mye brukte du på brennevin i forrige måned? Her kan vi velge bokstaven U for uvisst antall kroner og bokstaven N for nekter å svare. Både kodene N og U vil bli definert som MISSING. Videre kan de som ikke blir stilt spørsmålet få . som MISSING. Alle tre kodene behandles som MISSING, men de blir listet som forskjellige koder i f.eks. PROC FREQ.

```
DATA forbruk;
MISSING u U n N;
INPUT     @01 fnr        $CHAR11.
          @14 komm      $CHAR4.
          @19 brvin      6.
          ;
CARDS;
01010032424 0301 000501
09035431287 0301        N
04042421334 0211 000238
30116231832 0402 000757
             0503        U
28096132535            n
02113553435 0238 001046
31058341220 0301        .
          ;

PROC FORMAT ;
VALUE kr
      0 = '0'
     1- 499 = 'Under 500 kr'
     500- 999 = '500-999 kr'
     1000-high = '1000 kr og over'
      ;
PROC FREQ DATA=forbruk ;
TABLES brvin ;
TABLES brvin / missing;
FORMAT brvin kr. ;
TITLE 'Penger brukt på brennevin siste måned';
```

Slik blir utlistingen:

Penger brukt på brennevin siste måned					
	BRVIN	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Under 500 kr		1	25.0	1	25.0
500-999 kr		2	50.0	3	75.0
1000 kr og over		1	25.0	4	100.0
Frequency Missing = 4					
	BRVIN	Frequency	Percent	Cumulative Frequency	Cumulative Percent
	.	1	12.5	1	12.5
	N	2	25.0	3	37.5
	U	1	12.5	4	50.0
Under 500 kr		1	12.5	5	62.5
500-999 kr		2	25.0	7	87.5
1000 kr og over		1	12.5	8	100.0

3.9.40 null

Dette SAS-setningen består kun av et eller fire semikolon, og brukes når vi trenger en SAS-setning som ikke skal gjøre noen ting. Vanligste bruk er som avslutning av data innlest med CARDS-setningen, se side 58 .

;

3.9.41 OUTPUT

Med OUTPUT-setningen skriver vi en observasjon skrives til et eller flere SAS datasett. I et DATA-steg er det vanligvis automatisk OUTPUT til det SAS datasettet vi oppgir navnet på i DATA-setningen. Det betyr at vi ikke trenger noen egen OUTPUT-setning. I noen tilfeller må vi allikevel bruke OUTPUT-setningen og da sløyfes den automatiske OUTPUT-setning:

- når vi skal lage 2 eller flere SAS datasett
- når vi skal lage flere observasjoner av hver observasjon som leses inn fra et SAS datasett
- når vi skal lage 1 observasjon av flere observasjoner som leses inn fra et SAS datasett

OUTPUT <datasettnavn-1> ... <datasettnavn-n>;

Eksempel der vi deler et SAS datasett i tre nye: et for kvinner, et for menn og et for de med uoppgitt kjønn.

```
DATA persreg;
  INPUT @01 kommune $CHAR4.
        @06 gkrets $CHAR3.
        @10 kjonn $CHAR1.
        @12 alldgrp $CHAR4.
        @16 inntekt 7.
        @24 skatt 6.
        ;
CARDS;
0101 001 K alld1 230327 85023
0101 001 M alld1 317327 145102
0101 001 M alld2 122341 34213
0101 001 M alld3 213340
0101 001 M alld4 141341 24029
0101 001 M alld5 2313
0101 001 M alld1 442341 187292
0101 001 K alld1 355345 67326
0101 001 K alld2 242351 78219
0101 001 K alld3 116341 13232
0101 001 . alld4 326340 134216
0101 001 K alld5 442341 234145
;
DATA kvinner menn uopp;
SET persreg;
IF kjonn = 'K' THEN OUTPUT kvinner;
ELSE
  IF kjonn = 'M' THEN OUTPUT menn;
  ELSE
    OUTPUT uopp;
RUN;
```

3.9.42 PUT

Vi bruker først og fremst PUT til å skrive til en sekvensiell fil. Denne filen må først defineres med en FILE-setning, se side 63. PUT er også nyttig når vi tester programmene våre. Når vi bruker PUT uten å definere en utfil med FILE-setningen, skriver PUT til SASlog.

Det er viktig å skille mellom PUT, som skriver data til en sekvensiell fil, og OUTPUT (se side 82), som skriver til et SAS datasett.

```
PUT variabel-1 <$> startposisjon-1 <- sluttposisjon-1> <.desimaler>
... <variabel-n <$> startposisjon-n <- sluttposisjon-n> <.desimaler> > <@ | @@>;
PUT <peker-1> variabel-1 utskriftsformat. ... <<peker-n> variabel-n utskriftsformat.> <@ | @@>;
PUT <peker-1> (variabel-liste-1) (utskriftsformat-liste-1)
... <<peker-n> (variabel-liste-n) (utskriftsformat-liste-n)> <@ | @@>;
PUT <peker-1> (variabel-liste-1) (<n*> utskriftsformat.)
... <<peker-n> (variabel-liste-n) (<n*> utskriftsformat.)> <@ | @@>;
PUT <peker-1> variabel-1 <$> ... <<peker-n> variabel-n <$> > <@ | @@>;
PUT <peker-1> <n*>'Tekst' <@ | @@>;
PUT <peker-1> variabel-1 : utskriftsformat. ... <<peker-n> variabel-n : utskriftsformat.> <@ | @@>;
PUT <peker-1> variabel-1= ... <<peker-n> variabel-n= > <@ | @@>;
PUT variabel-1= <$> startposisjon-1 <- sluttposisjon-1> <.desimaler>
... <variabel-n= <$> startposisjon-n <- sluttposisjon-n> <.desimaler>> <@ | @@>;
PUT <peker-1> variabel-1= utskriftsformat. ... <<peker-n> variabel-n= utskriftsformat.> <@ | @@>;
PUT <peker-1> array-navn(*) <@ | @@>;
```

\$ peker	angir at variabelens verdi skal behandles som en karakterverdi ved skriving. angir pekertegn og posisjon for skriving til utfil. Har tallet som angir posisjon eller flytting desimaler, trunkeres disse. Er tallet 0 eller negativt, flyttes skrivehodet til posisjon 1. Ender skrivehodet utenfor recordlengden, flyttes det til posisjon 1 i neste record. Flytter vi bakover vil vi aldri kunne komme lenger tilbake enn til posisjon 1 på recorden. Pekere: @n n angir posisjonen der skrivehodet skal plasseres @variabel variabelen må være numerisk og skal inneholde startposisjonen det skal skrives ut til. @(SAS-uttrykk) SAS-uttrykket må gi et positivt heltall. Dette inneholder startposisjonen det skal skrives ut til. +n flytter lesehodet n posisjoner. +variabel variabelen skal være numerisk og inneholde det antall posisjoner skrivehodet skal flyttes +(SAS-uttrykk) SAS-uttrykket skal være numerisk og inneholde det antall posisjoner skrivehodet skal flyttes. #n angir at skrivehodet skal flyttes til record n i en gruppe records #variabel variabelen skal være numerisk og inneholder hvilken record i en gruppe records skrivehodet skal flyttes til #(SAS-uttrykk) SAS-uttrykket skal gi et positivt heltall som resultat. Dette tallet inneholder hvilken record i en gruppe records skrivehodet skal flyttes til / flytter skrivehodet til posisjon 1 på neste linje i utdatafilen OVERPRINT gjør at det som etterfølger OVERPRINT skrives ut oppå den linja som sist ble skrevet ut til utfilen. _PAGE_ Flytter skrivehodet til neste side på utskriftsfilen. Har ingen effekt ved utskrift til rådatafiler.
@	Forhindrer at PUT-setningen flytter skrivehodet til neste record når det avsluttes. Det skrives ikke til noen ny record og neste PUT-setning vil behandle den samme recorden. Denne holdingen oppheves når en PUT-setning som ikke holder recorden eksekveres (uten @ eller @@). I motsetning til INPUT-setningen holdes recorden også når neste iterasjon av DATA-steget utføres. Vi kan altså ha en ny PUT-setning som skriver til den samme recorden.
@@	Holder recorden slik at den kan gjennomgå flere iterasjoner av DATA-steget. Det betyr at vi kan skrive data fra flere observasjoner ut til en record. Holdingen frigis når skrivehodet når slutten på recorden, eller når en ny PUT-setning som ikke holder recorden som eksekveres.
n*'tekst'	Angir at en tekststreng skal skrives ut n ganger etter hverandre. Eksempel: PUT 80*'-' ;
n*format	angir at et format skal gjentas n ganger (for n variable). Eksempel: DATA _NULL_ ; SET tilbfall ; FILE 'tilbfall.dat' ; PUT fnr \$11. (dd1-dd7) (7*DDMMYY6.) ; gjør at vi kan bruke utskriftsformater på data som ikke skal skrives ut i faste posisjoner. Variabelens verdi skrives ut i følge utskriftsformatet, men den legges ikke ut bestemte posisjoner på utfilen.
utskriftsformat.	Angir et utskriftsformat som variabelverdiene skal skrives ut etter. Merk at karaktervariable vanligvis venstrejusteres og numeriske variable høyrejusteres. Dette kan overstyres/endres ved å føye til -R (høyrejustering), -L (venstrejustering) eller -C (sentring) etter formatet.
utskriftsformat-liste	Angir en liste over utskriftsformater som skal brukes. Denne må brukes sammen med variabelliste.
INFILE	Skriver ut en record som er lest inn. Hvis flere records er lest inn samtidig, skrives bare den siste av disse ut.
ALL	Skriver ut alle variablene som er definert i DATA-steget (inkludert hjelpevariable) med navn og verdi.

3.9.43 RENAME

Endre navn på variable som skal skrives til SAS datasett. Navnene vil endres på alle SAS datasett de inngår i og som lages i DATA-steget der RENAME brukes. Vi må bruke de gamle navnene når vi refererer til variablene i DATA-steget, navnet endres når de skrives til nye datasett (ved bruk av variablene i datasett options til nye datasett skal vi bruke de nye navnene). Ønsker vi kun å endre navn på variable i et av datasettene som inngår i DATA-steget eller endre navn i det dataene leses, må vi bruke SAS datasett option RENAME isteden, se side 49. Det er som oftest mer ryddig å bruke SAS datasett option RENAME istedenfor RENAME-setningen.

RENAME gammelt-navn-1 = nytt-navn-1 ... <gammelt-navn-n = nytt-navn-n> ;

RENAME g-navn-1 - gnavn-n = n-navn-1 - n-navn-n ... <g-navn-n1 - g-navn-nn = n-navn-n1 - n-navn-nn> ;

3.9.44 RETAIN

Brukes til å angi variable som skal beholde sin verdi fra forrige observasjon. Variablene beholder verdien helt til vi endrer dem. Variable som lages i DATA-steget settes vanligvis til MISSING før en observasjon leses, dette gjøres ikke for variable nevnt i RETAIN-setningen.

RETAIN <element-liste-1 <startverdi-1 | (startverdiliste-1)>> ... <element-liste-n <startverdi-n | (startverdiliste-n)>> ;

I elementlista skriver vi variabelnavn, variabellister eller array-navn. Hvis disse ikke gis startverdier, settes de til MISSING.

Eksempler:

```
RETAIN stonad bidrag laan;
RETAIN mnd1-mnd12 0 ; /* mnd1 til mnd12 gis startverdi 0 */
RETAIN kv1-kv4 (1 2 3 4) /* kv1 til kv4 gis startverdier henholdsvis 1,2,3 & 4 */
```

RETAIN uten parametre gir RETAIN på alle variable laget med INPUT og alle laget med assignment-setninger.

Vi kan definere RETAIN _ALL_ ; . Denne vil sørge for RETAIN for alle variable definert før RETAIN-setningen. Det samme gjelder for alle karaktervariable med RETAIN _CHARACTER_ ; og alle numeriske variable med RETAIN _NUMERIC_ ; .

Eksempel på aggregering av data ved hjelp av RETAIN. Vi skal lage et datasett som har en observasjon pr. kommune. Vi ønsker å finne ut hvor mange som har fått sosialhjelp (bidrag og/eller lån), hvor mye som er gitt og gjennomsnittsbeløp for de som har mottatt. Først sorteres datasettet etter kommune. Deretter aggregerer vi det i et DATA-steg. Vi leser datasettet pr. kommune (BY kommune). Da vet vi når vi leser siste observasjon innen en kommune (IF LAST.kommune). Det er bare for den siste observasjon innen kommunen vi ønsker å skrive ut til det nye datasettet. Bidragene som skal aggregeres legges i RETAIN-variable.

```
DATA soshjelp;
  INPUT fnr $ kommune $ bidrag laan ;
  CARDS;
  01012939013 0301 003134 000324
  03023934114 0211 012199 003031
  31126532212 0101 022100 000000
  09075543532 0301 052944 .
  19044843209 0211 000000 000232
  26116141107 0211 . 000870
  16014940085 0301 003543 001611
;

PROC SORT DATA=soshjelp;
  BY kommune ;
```

```
DATA soskomm (KEEP=kommune antbid sumbid snitbid
                                antlaan sumlaan snitlaan) ;
SET soshjelp;
RETAIN sumbid sumlaan 0;
BY kommune;
IF bidrag > 0 THEN
DO;
    antbid + 1;
    sumbid = SUM(sumbid,bidrag);
END;
IF laan > 0 THEN
DO;
    antlaan + 1;
    sumlaan = SUM(sumlaan,laan);
END;
IF LAST.kommune THEN
DO;
    snitbid = sumbid/antbid;
    snitlaan = sumlaan/antlaan;
    OUTPUT;
    antbid = 0;
    sumbid = 0;
    antlaan = 0;
    sumlaan = 0;
END;
```

Variablene brukt i sum-setningen (antbid og antlaan) trenger ikke å defineres vha. RETAIN, fordi de automatisk får RETAIN. Dette gjelder også variablene `_N_`, `_ERROR_`, `_I_`, `_CMD_` og `_MSG_`, variable lest med SET, MERGE eller UPDATE, hjelpevariable definert med END= eller IN= options og variable definert i en midlertidig (TEMPORARY) array.

3.9.45 RETURN

Med RETURN stopper vi eksekvering av SAS-setninger og starter på neste iterasjon av DATA-steget. Før stoppen utføres den automatiske OUTPUT-setningen (med mindre vi har egne OUTPUT-setninger). Ved bruk av GO TO, LINK eller HEADER option i FILE-setningen vil det ikke bli utført noen automatisk OUTPUT-setning. RETURN etter en LINK-setning returnerer til neste SAS-setning etter LINK, mens ved bruk av GO TO og RETURN fortsetter programmet med neste iterasjon av første SAS-setning i DATA-steget (med mindre en LINK-setning kommer før GO TO, da fortsetter programmet etter LINK).

RETURN ;

3.9.46 RUN

RUN bruker vi for å få eksekvert programmet vårt. Alle program skal ha en RUN-setning som siste linje. Vi kan også bruke RUN inne i et program for å skille DATA-steg fra hverandre.

SAS legger automatisk inn RUN etter CARDS- og DATALINES-setning, før et nytt DATA-steg, før et nytt PROC-steg og før en ENDSAS-setning. RUN CANCEL virker ikke ved bruk av CARDS, CARDS4, DATALINES og DATALINES4.

RUN <CANCEL> ;

CANCEL Hopper over steget foran RUN-setningen og skriver en melding om det i loggen. Kan brukes når vi ønsker å hoppe over et steg, men ikke ønsker å slette det fra programmet.

Eksempel:

```
DATA hatt;
DO x=-5 TO 5 BY .25;
DO y=-5 TO 5 BY .25;
    z= SIN(SQRT(x*x + y*y));
    OUTPUT;
END;
END;

PROC G3D DATA=hatt;
PLOT y*x=z /CTOP = yellow CTEXT = white;
TITLE 'Barsk hatt';
RUN CANCEL;
```



```

PROC MEANS DATA=hatt ;
  TITLE 'Hattens deskriptive mål' ;
RUN;

OPTIONS PS=80 NONUMBER NODATE NOCENTER;

PROC PRINT DATA=hatt (OBS=80);
  TITLE 'Del av hatten';
RUN;

```

Over ble prosedyren G3D hoppet over p.g.a. etterfølgende RUN CANCEL. OPTIONS-setningen gjelder kun for PROC PRINT, fordi vi har RUN før OPTIONS. Fjernes denne RUN, vil OPTIONS også gjelde PROC MEANS.

3.9.47 SELECT

SELECT kan ofte brukes istedenfor IF-setninger inne i hverandre. Med SELECT blir oppsettet mer strukturert og lettere å lese. SELECT virker slik at den utfører SAS-setningen til den betingelsen som først slår til. Deretter hoppes det til SAS-setningen etter testene. Det betyr at hvis den samme betingelsen går igjen to ganger i en SELECT-setning vil aldri den siste av disse slå til. Hvis vi ønsker å utføre flere SAS-setninger når en betingelse slår til, må disse avgrenses med en DO-setning og en END-setning.

```

SELECT <( seleksjonskriterium )>;
  WHEN ( betingelse-1 ... <,betingelse-n> ) SAS-setning ;
  <... WHEN ( betingelse-1 ... <,betingelse-n> ) SAS-setning ;>
  <OTHERWISE SAS-setning ;>
END;

```

Eksempel 1:

Koder om forbrytelser til bokstavkoder. Disse tekstes vha PROC FORMAT og vi lager en tabell med PROC TABULATE. På denne måten kan vi bestemme rekkefølgen forbrytelsene skal listes i.

```

PROC FORMAT ;
  VALUE $kjoenn
    '1' = 'Menn'
    '0' = 'Kvinner'
  ;
  VALUE $forbr
    'A' = 'Simpelt tyveri'
    'B' = 'Grovt tyveri'
    'C' = 'Brukstyveri av motorkjøretøy'
    'D' = 'Underslag, bedrageri og dokumentfalsk'
    'E' = 'Forbrytelse mot sedeligheten'
    'F' = 'Forbrytelse mot liv, legeme og helbred'
    'G' = 'Narkotikaforbrytelse'
    'Z' = 'Annen forbrytelse'
  ;
DATA a3694;
  SET tilbfall.a3694 (KEEP= lovb51 kjonn);
  SELECT (lovb51);
    WHEN ('173') forbr = 'A';
    WHEN ('174') forbr = 'B';
    WHEN ('176') forbr = 'C';
    WHEN ('111','112','113','171','172',
          '191','192','193','194') forbr = 'D';
    WHEN ('121','122','123','124','125',
          '126','127','128','129') forbr = 'E';
    WHEN ('151','152','153','154','155',
          '156','157','158','159') forbr = 'F';
    WHEN ('075','076','331','332') forbr = 'G';
    OTHERWISE forbr = 'Z';
  END;
PROC TABULATE DATA=a3694 NOSEPS FC=' -----' F=10. MISSING;
  CLASS forbr kjonn;
  TABLE ALL='Alle reaksjoner' forbr='
    '
    (ALL='I alt' kjonn='')*N='
  /BOX='Hovedforbrytelse ved utgangsreaksjon'
  RTS=45 MISSTEXT='-';
  FORMAT kjonn $kjoenn.
         forbr $forbr. ;
  TITLE 'Straffbare etter hovedforbrytelse ved utgangsreaksjon og kjønn';
RUN;

```

Resultatet:

Tabell 3. Straffbare etter hovedforbrytelse ved utgangsreaksjon og kjønn

Hovedforbrytelse ved utgangsreaksjon	I alt	Kvinner	Menn
Alle reaksjoner	10631	987	9644
Simplett tyveri	806	93	713
Grovt tyveri	2821	139	2682
Brukstyveri av motorkjøretøy	561	24	537
Underslag, bedrageri og dokumentfalsk	1116	231	885
Forbrytelse mot sedeligheten	315	2	313
Forbrytelse mot liv, legeme og helbred	1120	39	1081
Narkotikaforbrytelse	1694	280	1414
Annen forbrytelse	2198	179	2019

Eksempel 2:

Vi lager en verdi for forskjellige innbyggertall i kommuner. Her benytter vi oss av at SAS hopper til setningen etter testene når en betingelse er utført. En kommune med 4000 innbyggere vil ikke slå til på de to første testene, men den treffer på den tredje. Dermed settes *innbg* til 2, og flere tester i SELECT-setningen utføres ikke.

```
SELECT;
  WHEN (folktot < 1)      innbg = '9';
  WHEN (folktot < 2000)   innbg = '1';
  WHEN (folktot < 5000)   innbg = '2';
  WHEN (folktot < 10000)  innbg = '3';
  WHEN (folktot < 20000)  innbg = '4';
  WHEN (folktot < 30000)  innbg = '5';
  WHEN (folktot < 50000)  innbg = '6';
  WHEN (folktot < 100000) innbg = '7';
  OTHERWISE               innbg = '9';
END;
```

3.9.48 SET

SET brukes for å lese allerede eksisterende SAS datasett. SET-setningen kan brukes til å kopiere, sette etter hverandre, eller flette eksisterende SAS datasett.

SET datasett <(datasettoptions-1)> ... <datasett-n <(datasettoptions-n)>> <POINT=hjelpevariabel>
<NOBS=hjelpevariabel> <END=hjelpevariabel> ;

- datasett navn på datasett som skal leses
- datasettoptions options til datasettet, se side 47.
- POINT= navn på en hjelpevariabel som skal inneholde observasjonsnr som skal leses. Denne variabelen må endres ved hver SET-setning, slik at forskjellige observasjoner leses. Når vi bruker POINT= bør vi ha med et STOP-setning, ellers kan vi risikere at programmet går i en evig løkke. POINT= kan ikke brukes sammen med BY eller WHERE-setninger, WHERE= datasettoption, for datasett i transportformat eller på tape.
Eksempel på program som utfører trekking med tilbakelegging. Det skal trekkes like mange observasjoner som det er i det opprinnelige datasettet, men alle observasjoner skal ha like stor sjanse for å bli trukket ut hver gang:

```
DATA varer;
DO i = 1 TO 13;
  OUTPUT;
END;
DATA nyvare;
  recno = ROUND(UNIFORM(0)*antobs+0.5);
  SET varer NOBS=antobs POINT=recno;
  IF _N_ > antobs THEN
    STOP;
```

Eksempel der vi bruker POINT og NOBS i binær søk for å sjekke om en fødselsdato finnes i et datasett som er sortert etter fødselsdato:

```

DATA _NULL_;
  lav = 1;
  hoy = antall;
  DO WHILE (lav <= hoy);
    mid = INT((lav + hoy) / 2) ;
    SET persreg NOBS=antall POINT=mid;
    teller + 1;
    IF fdato > '19DEC1986'D THEN hoy = mid - 1;
    ELSE
      IF fdato < '19DEC1986'D THEN lav = mid + 1;
    ELSE
      DO;
        PUT 'Funnet! ' fdato= teller=;
        STOP;
      END;
    END;
  END;
  STOP;
  RUN;

```

NOBS= navn på en hjelpevariabel som inneholder totalt antall observasjoner på datasettene som listes i SET-setningen. Denne kan vi referere til før SET-setningen.

Eksempel på program som finner antall observasjoner i et SAS datasett og legger resultatet ut på et nytt datasett:

```

DATA antobs (KEEP=ant );
  SET dode.g91g93 (OBS=1) NOBS=antall ;
  ant= antall;

```

END= navn på en hjelpevariabel som indikerer slutten på datasettene. END-variabelen settes til 1 når siste observasjon på siste datasett er lest, ellers er den 0.

Eksempel 1:

Lager et SAS datasett som består av de første 200 observasjoner av et annet.

```

DATA utdrag;
  SET befolk.innb (OBS=200);
  RUN;

```

Eksempel 2:

Lager et SAS datasett ved å sette flere SAS datasett etter hverandre. Det nye datasettet vil inneholde alle observasjoner fra de leste datasettene.

```

DATA samle;
  SET bvern.flk01_07 bvern.fylk08_12 bvern.fylk14_20;
  RUN;

```

Eksempel 3:

Under revisjonsarbeid hender det ofte at vi må få inn data på nytt fra en dataleverandør. Dette kan også skje ganske sent i revisjonsprosessen. Er alle innkomne data lagt ut på et datasett før vi får en ny utgave av en del av materialet, må det gamle byttes ut med det nye. Her følger et eksempel der vi har fått inn nye barnevernsdata fra Trondheim (1601) og ønsker å bytte de gamle dataene med de nye i totaldatasettet:

```

DATA bvern.landetny;
  SET bvern.landet (WHERE=(komm ne '1601'))
    bvern.k1601ny ;

```

Med SAS datasett option WHERE= utelater vi kommune 1601 fra det gamle totaldatasettet. De nye dataene fra Trondheim blir så lagt til på slutten av det nye datasettet.

Hvis begge SAS datasettene er sortert, kan vi sørge for at også det nye SAS datasettet får den riktige sortering. Det betyr at dataene fra Trondheim flettes inn på riktig sortert plass i det nye SAS datasettet. Ved å bruke BY-setningen ordner vi det:

```

DATA bvern.landetny;
  SET bvern.landet (WHERE=(komm ne '1601'))
    bvern.k1601ny ;
  BY komm fnr;

```

3.9.49 SKIP

Hopper n linjer i SASlog. Brukes for å lage luft i SASlog

SKIP <n> ;

3.9.50 *STOP*

Sørger for å stoppe DATA-steget. SAS lager datasett fra DATA-steget inneholdende de observasjoner som er skrevet ut inntil STOP-setningen eksekveres (gjeldende observasjon skrives ikke ut såfremt den ikke allerede er skrevet ut). SAS fortsetter deretter med neste DATA eller PROC-steg. STOP må brukes hvis data leses med POINT= option i SET-setningen, se side 88, fordi SAS med denne aksessmetoden ikke finner end-of-file selv.

STOP ;

3.9.51 *sum*

Sum-setningen er en av de få setninger som ikke starter med et nøkkelord. Vi bruker det til å addere verdien til en variabel. Variabelen initieres automatisk til 0 og beholder sin verdi selv når ny observasjon leses (automatisk RETAIN). Dessuten behandles MISSING-verdier i uttrykket som 0, akkurat som ved bruk av SUM-funksjonen, se side 235. Brukes ofte for å lage tellere. En sum-setning tilsvarer en RETAIN-setning og en assignment-setning med SUM-funksjonen. Ønsker vi å ha en annen startverdi enn 0, må vi angi denne i en egen RETAIN-setning (se side 84).

variabel + SAS-uttrykk ;

```
IF alder = . THEN  
  ald_miss + 1 ;
```

3.9.52 *UPDATE*

UPDATE bruker vi når vi ønsker å oppdatere et SAS datasett med data fra et annet SAS datasett ved hjelp av BY-variable (koblingsnøkkel) angitt med en egen BY-setning. Datasettet som skal oppdateres kalles masterdatasett, mens datasettet med korreksjonene kalles transaksjonsdatasett. Ved hjelp av UPDATE kan eksisterende variable oppdateres, nye variable legges til observasjonene og nye observasjoner kan legges til datasettet. Variable med samme navn i masterdatasettet og transaksjonsdatasettet blir oppdatert når verdiene på transaksjonsdatasettet ikke er MISSING, er verdien MISSING beholdes verdien fra masterdatasettet (dette kan overstyres med option UPDATEMODE=NOMISSINGCHECK). Variable som bare er i transaksjonsdatasettet blir lagt til på masterdatasettet. Disse variablene blir satt til MISSING på observasjoner som bare er i masterdatasettet. Observasjoner som har koblingsnøkkel som bare er på transaksjonsdatasettet blir lagt til på masterdatasettet (variable som bare finnes på masterdatasettet settes da til MISSING). UPDATE minner om MERGE, men logikken er litt annerledes, se MERGE på side 80 og Kobling av datasett, side 15.

**UPDATE master <(datasettoptions)> trans <(datasettoptions)> <END=variabel>
<UPDATEMODE=MISSINGCHECK | NOMISSINGCHECK>;**

master navn på masterdatasettet, registeret som skal oppdateres

trans navn på transaksjonsdatasettet, datasettet med korreksjonene

datasettoptions options til datasettet, se side 47

END navn på en hjelpevariabel som indikerer slutten på datasettene. END-variabelen settes til 1 når siste observasjon på alle datasett er lest, ellers er den 0.

UPDATEMODE= Angir om MISSING-verdier i transaksjonsdatasettet skal overskrive verdier i registeret (NOMISSINGCHECK) eller om registerets verdier skal beholdes som de er (MISSINGCHECK). MISSINGCHECK er standard.

3.9.53 *WHERE*

WHERE brukes for å selektare data. Standard er at vi kan bare ha én WHERE-setning i et DATA- eller PROC-steg. Skriver vi to WHERE-setninger, vil den første bli ignorert. Dette kan vi unngå ved å bruke WHERE SAME AND på den andre WHERE-setningen, da vil begge WHERE-setningene bli utført.

WHERE tilsvarer nesten subsetting IF (beskrevet på side 66), men det er noen forskjeller:

- WHERE selekterer observasjoner ved innlesing av disse slik at seleksjonen kan bli utført her, mens subsetting IF utføres etter lesing. Det betyr at WHERE-setningen er mer effektiv.
- For data lest med BY-setningen, vil FIRST og LAST hjelpevariablene gis verdi etter at WHERE-setningen er utført, men før en subsetting IF-setning.
- WHERE kan ikke utføres som en betingelse til en IF-setning, det kan subsetting IF.
- WHERE selekterer fra eksisterende datasett, subsetting IF selekterer ved innlesing av rådata også.
- WHERE-setningen er litt strengere på konvensjonene (numeriske blir ikke konvertert og omvendt om argumentet har feil type) når vi bruker funksjoner.
- WHERE kan brukes til seleksjoner i prosedyrer, det kan ikke subsetting IF.
- WHERE-setningen har noen uttrykk som IF-setningen ikke har, se under.

WHERE where-uttrykk ;

Where-uttrykket kan inneholde alle typer SAS-uttrykk som er beskrevet på side 34. I tillegg finnes noen uttrykk som kun kan brukes i WHERE-setningen:

WHERE SAME AND where-uttrykk ;

WHERE variabel <NOT> **BETWEEN** verdi **AND** verdi ;

WHERE variabel **IS** <NOT> **MISSING**;

WHERE variabel <NOT> **LIKE** mønster ;

WHERE variabel <NOT> **CONTAINS** | ? verdi ;

WHERE variabel <NOT> **=*** verdi ;

WHERE SAME AND brukes for nye WHERE-setninger der vi allerede har en fra før. Betingelsene vil slå til når alle WHERE-setningene oppfylles.

WHERE ... BETWEEN ... AND brukes for intervaller. Intervallets grenseverdier tas med.

WHERE ... IS MISSING slår til når where-uttrykket er MISSING.

WHERE ... LIKE tester om where-uttrykket inneholder et mønster. For å beskrive mønsteret brukes to spesialtegn (som jokernotasjon), _ (understrek) for ett hvilket som helst tegn og % (prosent) for et uspesifisert antall tegn (også null).

Eksempel:

```
DATA pers;
INPUT @1 lidelse      $CHAR50. ;
CARDS;
ALLEGI
ALLERGI
ALLLERGI
ASTMAALERGI
ATOPIISK ALLERGI
DYR/POLLEN ALLERGI
DYREALLERGI
FIBROMYALLERGI
FRUKTALLERGI MM
GRESSPOLLENALLERGI
POLLENALLERGI/HØYSNUE
KROMMALLERGI
MATALLERGI
MATVAREALLERGI
NIKKELALLERGI
PENSELINALLERGI
POLLEN ALERGI
POLLEN ALLERGI
VEPSEALLERGI
;
DATA persny ;
SET pers ;
WHERE lidelse LIKE 'AL_E%GI' ;
```

Dette programmet velger ut de to første, ALLEGI og ALLERGI. Ønsker vi å godta flere feilstavinger kan vi prøve denne:

```
WHERE lidelse LIKE 'AL%E%GI' ;
```

Da får vi med ALLLERGI også. Hvis vi ønsker alle der ALLERGI forekommer og er stavet riktig kan vi bruke denne:

```
WHERE lidelse LIKE '%ALLERGI%' ;
```

Ønsker vi å godta feilstavinger i tillegg, kan vi bruke denne som slår til på alle i eksempelet:

```
WHERE lidelse LIKE '%AL%E%GI%' ;
```

WHERE ... CONTAINS tester om where-uttrykket inneholder en tekst.

Eksempel der vi plukker ut alle som har ALLERGI stavet riktig:

```
WHERE lidelse CONTAINS 'ALLERGI' ;
```

WHERE ... *=* tester om where-uttrykket er fonetisk likt oppgitt verdi. Denne utføres etter samme regler som SOUNDEX-funksjonen bruker (se side 233).

3.9.54 WINDOW

Definerer et vindu som skal vises i løpet av programmet. Dette kan brukes til å taste inn parametre som skal brukes senere i programmet. For å vise vinduet på skjermen, må vi bruke DISPLAY-setningen, se side 60.

WINDOW vindu-navn <vindu-options> feltdefinisjon-1 ... <feltdefinisjon-n> ;

WINDOW vindu-navn <vindu-options> gruppedefinisjon-1 ... <gruppedefinisjon-n> ;

vindu-navn	navn på vinduet
vindu-options	options for vinduet:
COLOR= farge	angir farge for vinduet. Disse farger kan vi velge mellom: WHITE, BLUE, YELLOW, GREEN, PINK, GRAY, RED, BLACK, BROWN, CYAN, MAGENTA og ORANGE.
COLUMNS=n	angir bredden på vinduet, n angir antall posisjoner unntatt rammen. Utelates denne vil vinduet dekke hele bredden
ROWS=n	angir høyden på vinduet, n angir antall posisjoner unntatt rammen. Utelates denne vil vinduet dekke hele høyden
ICOLUMN=n	angir kolonnennummer for venstre side av vinduet. Utelates denne plasseres vinduet til venstre i skjermbildet
IROW=n	angir linjenummer for øvre del av vinduet. Utelates denne plasseres vinduet øverst i skjermbildet
KEYS=	angir navn på funksjonstaster som skal brukes i vinduet. Utelates denne vil funksjonstastene være som ellers i SAS.
MENU=	angir navnet på en rullegardinmeny som er laget med PMENU-prosedyren som kan brukes i vinduet.

feltdefinisjon	definerer et felt og tekster i vinduet. For hvert felt må vi angi startposisjon (linje og kolonne) og feltnavn. Vi kan også angi innlesingsformat for felt som vi kan skrive i, og utskriftsformat for felt som er beskyttet mot skriving. Syntaks for feltdefinisjoner: <linje kolonne> variabel-navn <utskriftsformat innlesingsformat> options Syntaks for tekster i vinduet: <linje kolonne> 'tekst' options
linje	angir hvilken linje i vinduet feltet skal plasseres i: #n angir at feltet skal plasseres i linje n #variabel variabelen skal være numerisk og inneholder hvilken linje feltet skal plasseres i #(SAS-uttrykk) SAS-uttrykket skal gi et positivt heltall som resultat. Dette tallet inneholder hvilken linje feltet skal plasseres i flytter pekeren til posisjon 1 på neste linje i vinduet
kolonne	angir hvilken kolonne i vinduet feltet skal plasseres i: @n n angir kolonnen der feltet skal starte @variabel variabelen må være numerisk og skal inneholde kolonnen feltet skal starte i. @(SAS-uttrykk) SAS-uttrykket må gi et positivt heltall. Dette inneholder kolonnen skal starte i. +n flytter n posisjoner. +variabel variabelen skal være numerisk og inneholde det antall posisjoner som skal flyttes
dataformat	til felt som det er lov å skrive i kan vi bruke et innlesingsformat, og for beskyttede felt kan vi bruke et format. Mer om formater og innlesingsformater på side 36.
'tekst'	tekst som skal vises i vinduet
options	options til et felt i vinduet: ATTR= BLINK feltet skal blinke HIGHLIGHT feltet skal lyse kraftig REV_VIDEO feltet skal vises i negativ UNDERLINE feltet skal understrekes Ved bruk av flere attributter samtidig skal de skrives i parentes. AUTOSKIP= YES feltet skal hoppes over NO feltet skal ikke hoppes over COLOR= angir fargen til feltet, mulige farger er de samme som over DISPLAY= YES feltets innhold skal vises NO feltets innhold skal ikke vises PROTECT= YES ikke lov å skrive i feltet NO lov å skrive i feltet REQUIRED= YES feltet må fylles ut NO frivillig å fylle ut feltet
gruppedefinisjon	definerer flere felt som en gruppe. Et vindu kan bestå av flere grupper av felt. Vi kan variere hvilken gruppe av felt som skal vises avhengig av forskjellige betingelser. En gruppe defineres med denne syntaksen: GROUP=gruppenavn feltdefinisjon-1 ... <feltdefinisjon-n>

Eksempel på et program der vi bruker et vindu for å la brukeren taste inn datasettnavn og årstall. Disse lagrer vi i SAS makrovariable vha. CALL SYMPUT. Vi vil da kunne bruke det samme program fra år til år uten endringer.

```

DATA sosial;
  INPUT @01 alder      2.
        @04 fylke    $CHAR2.
        ;
CARDS;
16 01
23 01
25 01
26 01
43 02
43 02
23 02
45 03
25 03
16 05
23 05
;
DATA _NULL_ (KEEP= innds aar);
  WINDOW vt01 COLOR=MAGENTA COLUMNS=72 ROWS=17 ICOLUMN=2 IROW=4
    #03 @4 'Sosialhjelp. Tabell 1.1'
    #10 @4 'Oppgi navn på datasett tabellen skal lages fra: ' innds $17.
            ATTR=(REV_VIDEO) REQUIRED=YES
    #11 @4 'Oppgi årstall: ' aar $4.
            ATTR=(UNDERLINE,REV_VIDEO) REQUIRED=YES
        ;
  DISPLAY vt01;
  CALL SYMPUT('innds',innds);
  CALL SYMPUT('aar',aar);
  STOP;
;

PROC TABULATE DATA=&innds NOSEPS FC=' -----' MISSING F=7.;
  CLASS fylke alder;
  TABLE all='I alt' fylke='',
        (all='I alt' alder='') *N=''
        /RTS=16;
  TITLE "Tabell 1.1 Stønadstilfeller med økonomisk sosialhjelp"
        " etter alder og fylke. &aar";
RUN;

```

Vi kan utvide programmet til å ta vare på det brukeren sist tastet inn, slik at vi kan plassere disse opplysningene i vinduet før brukeren får det opp. Informasjonen lagres på et eget SAS datasett. Lagres dette permanent, vil vi kunne hente opp informasjon fra tidligere SAS sesjoner også. Vi må først gjøre en sjekk på om SAS datasettet med informasjonen finnes fra før, det gjør vi med å sjekke at datasettet har minst 1 observasjon. Har det ikke det oppstår en feil som vanligvis avbryter programmet. Ved å bruke system option NODSNFERR vil ikke programmet avbrytes når datasettet ikke eksisterer.

```

LIBNAME parm '.';
OPTIONS NODSNFERR;
DATA parm.tab01 (KEEP= innds aar);
  IF ant > 0 THEN
    SET parm.tab01 NOBS=ant;
  WINDOW vt01 COLOR=MAGENTA COLUMNS=72 ROWS=17 ICOLUMN=2 IROW=4
    #03 @4 'Sosialhjelp. Tabell 1.1'
    #10 @4 'Oppgi navn på datasett tabellen skal lages fra: ' innds $17.
            ATTR=(REV_VIDEO) REQUIRED=YES
    #11 @4 'Oppgi årstall: ' aar $4.
            ATTR=(UNDERLINE,REV_VIDEO) REQUIRED=YES
        ;
  DISPLAY vt01;
  CALL SYMPUT('innds',innds);
  CALL SYMPUT('aar',aar);
  OUTPUT;
  STOP;
;
OPTIONS DSNFERR;

```


En utvidelse til er å sjekke om SAS-datasettet, hvis navn brukeren taster inn, eksisterer. For å få til dette bruker vi en SAS makro. Vi lager en løkke som går helt til brukeren taster inn navnet på et datasett som finnes.

```

OPTIONS NODSNFERR ;
%GLOBAL innds aar;
%MACRO param;
  %LET merke = 0;
  %DO %UNTIL (&merke = 1);
    DATA ptab01 (KEEP= innds aar feil);
      IF ant > 0 THEN
        SET ptab01 NOBS=ant;
        WINDOW vt01 COLOR=PINK COLUMNS=72 ROWS=17 ICOLUMN=2 IROW=4
          #03 @4 feil $65. PROTECT=YES AUTOSKIP=YES
          #06 @4 'Sosialhjelp. Tabell 1.1'
          #10 @4 'Oppgi navn på datasett tabellen skal lages fra: ' innds $17.
            ATTR=(REV_VIDEO) REQUIRED=YES
          #11 @4 'Oppgi årstall: ' aar $4.
            ATTR=(UNDERLINE,REV_VIDEO) REQUIRED=YES ;
        DISPLAY vt01;
        CALL SYMPUT('innds',innds);
        CALL SYMPUT('aar',aar);
        OUTPUT;
        STOP;

    DATA ptab01 (KEEP=innds aar feil);
      IF ant > 0 THEN
        SET ptab01 NOBS=ant;
        IF ant2 > 0 THEN
          DO;
            SET &innds (OBS=1) NOBS=ant2;
            CALL SYMPUT('merke',1);
            feil = " " ;
          END;
        ELSE
          DO;
            feil = "DATASETTET EKSISTERER IKKE: &innds" ;
          END;
        OUTPUT;
        STOP;
      RUN;
  %END;
%MEND;
%param;
OPTIONS DSNFERR;

```

3.9.55 Andre SAS-setninger

MODIFY, REMOVE, REPLACE bør vi være meget forsiktig med å bruke, da data kan gå tapt hvis systemet går ned når disse utføres. %RUN er heller ikke beskrevet

3.10 Globale SAS-setninger

SAS har noen globale setninger som vanligvis plasseres utenfor DATA- og PROC-stegene. Disse setningene gjøres uavhengig av DATA- og PROC-stegene.

3.10.1 DM

Utfører en SAS kommando (Display Manager kommando). DM virker bare ved interaktiv bruk av SAS. Kan brukes til å automatisere rutiner.

DM <vindu> 'kommando-1 <; .. kommando-n>' <vindu>;

Vindu kan brukes først for å velge hvilket vindu kommandoen skal utføres i, og til slutt for å velge vindu som skal vises etter at kommandoen er utført. Kommandoer som kan utføres er alle SAS kommandoene, som f.eks. CLEAR, FILE og TOP, se side 26. Alle kommandoer som er gyldige i vinduet som er valgt, kan brukes.

3.10.2 ENDRSUBMIT

Vi bruker ENDRSUBMIT sammen med RSUBMIT til å avgrense SAS-setninger som skal eksekveres på en tjenermaskin. De SAS-setningene som står mellom RSUBMIT og ENDRSUBMIT blir eksekvert på tjenermaskinen når programmet kjøres med vanlig SUBMIT-kommando. Mer om klient/tjener bruk på side 257.

ENDRSUBMIT ;

3.10.3 FILENAME

FILENAME gir et SAS-navn til en eller flere rådatafiler. Dette SAS-navnet bruker vi videre i SAS-programmet som en referanse til rådataene. Vi bruker også FILENAME når vi ønsker å fjerne filreferansen.

FILENAME filreferanse <device-type> 'rådatafil' <options>;
FILENAME filreferanse <device-type> ('rådatafil-1' ...<,'rådatafil-n') <options>;
FILENAME filreferanse device-type <options>;
FILENAME filreferanse | _ALL_ <LIST>;
FILENAME filreferanse CLEAR;
FILENAME filreferanse EMAIL TO="e-post-adresse" CC="e-post-adresse" SUBJECT="emne"
 ATTACH=("vedlegg-1" ... "vedlegg-n");

filreferanse	SAS-navn for rådatafilen(e). For bruk i FILE, INFILE eller %INCLUDE senere i programmet.
device-type	spesifiserer hvilken form rådataene er i:
COOMPORT	åpner, under windows, en com-port på PC-en for skriving eller lesing.
DDE	for dynamisk datautveksling under Windows, fra f.eks. Excel. Se side 11.
DISK	stryker data til disk. Dette er standard.
DRIVEMAP	lister alle tilgjengelige disketter (Windows). Eksempel: <pre>FILENAME disketter DRIVEMAP; DATA _NULL_; INFILE disketter; INPUT; PUT _INFILE_; RUN;</pre>
DUMMY	angir null inndata eller utdata (for testing av program).
FTP	Til ftp-oppkobling av filer.
NAMEPIPE	åpner for kommunikasjon mellom forskjellige applikasjoner (windows).
PIPE	styrer inn- eller utdata gjennom et annet program, f.eks. egrep. (Unix SAS). Eksempel på program som lister unix-komprimerte filer under et område: <pre>FILENAME komprimert pipe "ls \$SOSHJELP/wk12/*.Z"; DATA komprimert; INFILE komprimert; LENGTH filnavn \$ 80; INPUT filnavn \$char80.; PROC PRINT DATA=komprimert; TITLE 'Unix-komprimerte filer'; RUN;</pre>
PLOTTER	styrer utdata til plotter (kan brukes sammen med SAS/GRAPH).

PRINTER styrer utdata til skriver.
 TEMP angir for Unix SAS en fil som lagres på temp-område.
 TERMINAL styrer utdata til skjermen.
 rådatafil Vanligvis foreligger rådataene som sekvensielle filer. Vi kan bruke FILENAME til å si at vi ønsker å lese flere sekvensielle filer som én logisk fil (konkatenering). Eksempel:

```
DATA dode.g91g93 (DROP=fnr);
  FILENAME inn ('$DODARSAK/aar/dode.g91'
               '$DODARSAK/aar/dode.g92'
               '$DODARSAK/aar/dode.g93');
  INFILE inn MISSOEVER PAD LRECL=143;
  INPUT      @41 fnr      $CHAR11.
             @41 fdato    DDMYY6.
             @63 fylke    $CHAR2.
             @63 komm     $CHAR4.
             @80 dfylke   $CHAR2.
             @80 dkomm    $CHAR4.
             @52 dodsdate YYMMDD6.
             ;
```

RUN;

ALL Fjerner alle tilordninger som er gjort med FILENAME-setninger.

ALL LIST Lister alle tilordninger som er gjort med FILENAME-setninger.

CLEAR Fjerner tilordning for filreferansen som nevnes.

EMAIL For å sende e-post. Eksempel:

```
FILENAME epost EMAIL
  TO="krl@ssb.no"
  SUBJECT="Befolkningstabeller"
  ATTACH=("beftab.html" "beftabp.html")
  ;
```

```
DATA _NULL_ ;
  FILE epost;
  PUT 'Hei,';
  PUT ' ';
  PUT 'her er befolkningstabellene du bestilte.';
  PUT ' ';
  PUT 'Kristian';
  RUN;
```

Utvalgte options til FILENAME-setningen:

BLKSIZE= Angir rådataenes blokkstørrelse.
 LINESIZE angir linjelengde på utdata (Windows SAS).
 LRECL= Angir logisk recordlengde. Er recorden mer en 256 tegn (132 for Windows SAS), må LRECL (eventuelt LS) oppgis, ellers kuttet recorden. Lengden kan være 1-65488.
 MOD angir at utdata skal legges til bakerst på eksisterende fil.
 NEW | OLD Unix, angir om filen eksisterer fra før eller ei.
 NOTAB Sammen med device-type DDE gjør NOTAB at SAS ikke skiller kolonner som er adskilt med tabulator tegn.
 RECFM= Angir recordformatet til rådatafilen. F er fast lengde, V er variabel, N er binært format og P er utskriftsformat.

3.10.4 FOOTNOTE

Lager fotnoter. Vi kan ha opptil 10 linjer med fotnoter. Fotnotene vil komme på alle sider utskrift helt til vi opphever fotnoten med en FOOTNOTE-setning uten tekst. En FOOTNOTE setning overskriver et tidligere med samme linjenummer og nullstiller alle etterfølgende fotnotelinjer. Maksimal lengde er 255 tegn pr. fotnote. Fotnoten kuttet dersom den er lengre en maksimal linjebredde, definert med system option LINESIZE= (se side 43). Når vi lager html-filer fra SAS kan vi knytte en forbindelse til et html-dokument ved å bruke LINK-option.

FOOTNOTE<n> <LINK="html-fil"> <'Fotnotetekst'> ;

Eksempel:

```
FOOTNOTE1 'Kilde: Statistisk sentralbyrå' ;
```

3.10.5 %LET

Gir en verdi til en makrovariabel. Denne kan brukes uavhengig av SAS makroer og er nyttig til å angi filnavn og annet som ofte skal endres hver gang et SAS-program skal kjøres. Da kan vi plassere %LET-setninger øverst i programmet slik at vi sørger for at alle rettelser kan gjøres på ett sted. Verdien skal ikke stå i fnutter.

%LET makrovariabel= verdi ;

Eksempel:

```
%LET aar = 1999 ;
%LET innds = pasient.brt1998 ;
```

Når vi skal bruke makrovariabelen, refererer vi til den med ampersand foran og punktum bak: &aar. Skal de brukes i en tekst, må vi bruke doble fnutter rundt teksten ("Data for &aar.").

3.10.6 LIBNAME

Med LIBNAME oppretter vi forbindelse til et område der vi ønsker å lagre permanente SAS datasett. Denne forbindelsen gis et kortnavn i LIBNAME-setningen, en såkalt libref. Vi forteller SAS hvor dataene våre skal plasseres og også hvor SAS kan finne SAS datasettene vi skal bruke. I LIBNAME-setningen oppgir vi område som en libref, dvs. et SAS-navn på området. Videre må vi angi det faktiske navnet på området. Dette området er en vanligvis en katalog (directory) på en disk.

En LIBNAME-setning gjelder for hele SAS-sesjonen. Det betyr at når LIBNAME-setningen først er utført, holdes forbindelsen så lenge vi er inne i SAS. Faste forbindelser kan med fordel legges i en *autoexec.sas* fil, se side 258. Libref for SAS formater som skal lagres permanent må hete library, ellers kan vi velge egne navn på librefs.

LIBNAME libref <lagringsformat> | <REMOTE> <SLIBREF=tjenermaskin-libref> <'data-område'>
 <SERVER=tjenermaskin> <ACCESS=readonly | temp>
 <OUTREP=format> <REPEMPTY=YES | NO>;

LIBNAME libref <lagringsformat> | <REMOTE> <SLIBREF=tjenermaskin-libref> <'data-område-1' ...
 <'data-område-n'>> <SERVER=tjenermaskin> <ACCESS=readonly | temp>
 <OUTREP=format> <REPEMPTY=YES | NO>;

LIBNAME libref | _ALL_ CLEAR ;

LIBNAME libref | _ALL_ LIST ;

libref	SAS-navnet på filreferansen. Dette er betegnelsen vi skal bruke i SAS-programmene.
lagringsformat	angir SAS lagringsmetode eller ekstern lagringsmetode.
V6	SAS versjon 6. Kan lese SAS datasett fra tidligere versjoner.
V611	SAS versjon 6.11. Kan lese SAS datasett fra tidligere versjoner.
V610	SAS versjon 6.10. Kan lese SAS datasett fra tidligere versjoner. Konverterer ikke versjon 6.10 datasett til versjon 6.11.
V608	SAS versjon 6.08. Kan lese SAS datasett fra tidligere versjoner. Konverterer ikke versjon 6.08 datasett til versjon 6.11.
V8	SAS versjon 8. Dette er standard så lenge det ikke finnes SAS datasett fra før under området.
XPORT	Brukes for filer i SAS transportformat
OSIRIS	Brukes for å lese filer i OSIRIS format
SPSS	Brukes for å lese filer i SPSS format
XML	Brukes for å lage eller lese xml-filer (eksperimentell). Greier å lage xml-filer og kan lese xml-filer laget vha. PROC COPY.
REMOTE	angir at området befinner seg på en tjenerstasjon (arbeidsstasjon). Krever at SERVER=angis.
SLIBREF=	gjør et område på tjenermaskinen tilgjengelig interaktivt fra klienten.
data-område	angir fysisk plassering for området det refereres til.
SERVER=	angir hvilken tjenermaskin området er tilknyttet.
ACCESS=	READONLY gjør at det bare gis leseaksess til området, TEMP at området behandles som et arbeidsområde.

OUTREP= Angir operativsystem SAS datasettet skal lagres for. Aktuelle verdier er ALPHA_OSF (Digital unix), ALPHA_VMS, CMS, HP_UX, MAC, MV5, OS2, RS_6000_AIX, SOLARIS (Sun unix), VAX_VMS, VSE og WINDOWS (Windows NT).

REPEMPTY= YES, som er standard, angir at nye tomme SAS datasett som lages skal overskrive gamle med samme navn. NO hindrer dette. Se ellers SAS datasett option REPEMPTY, side 49.

ALL med _ALL_ ønsker vi å behandle alle tilknyttede librefs (enten med LIST eller CLEAR)

CLEAR CLEAR fjerner tilknyttede librefs.

LIST lister alle tilknyttede librefs.

Eksempel:

```
LIBNAME bvern '$BARNVERN/wk12';
LIBNAME library '$BARNVERN/kat';
LIBNAME hu95 '$HELSEUND/g95/wk12';
```

Tilgang til datasett på arbeidsstasjon (her: ursus) fra PC (sendes til eksekvering lokalt med vanlig SUBMIT).

```
LIBNAME tilbfall REMOTE '$KRIMINAL/tilbfall/work' SERVER=ursus;
```

Tilgang til arbeidsområdet på unix arbeidsstasjon (her: ovibos) fra PC (sendes til eksekvering lokalt med vanlig SUBMIT).

```
LIBNAME workunix REMOTE SLIBREF=work SERVER=ovibos;
```

Når vi skal vise til SAS datasett som er lagret under et område vi har definert med LIBNAME-setningen, brukes librefen som første ledd i navnet på SAS datasettet. Det andre leddet er navnet på selve datasettet (uten filhale). Har vi definert en libref *bvern* som refererer til området \$BARNVERN/wk12, og vi har et SAS-datasett kalt *meld1998.sas7bdat* (sas7bdat er filhalen til SAS datasett), får SAS datasettet navnet *bvern.meld1998* i SAS.

3.10.7 %LIST

Lister noen eller alle SAS-setninger som er eksekvert i denne SAS-sesjonen. Resultatet legges i SASlog. Kan være nyttig når vi har slettet SASlog, men ønsker å se hvilke SAS-setninger vi har eksekvert. %LIST virker ikke når SAS system option NOSPOOL er satt.

%LIST <n< - m >> ;

n første linje som skal listes
m siste linje som skal listes

3.10.8 ODS

SAS har laget et eget system for å håndtere den utskriften vi lager. Det kalles Output Delivery System og forkortes ODS. All utskrift som vi gjør fra prosedyrer og data-steg tas hånd om av dette systemet. Vi kan bruke det til å velge format og layout for utskriften.

Tidligere ble all utskrift styrt til utskriftsvinduet som tekstfiler. Disse krevde det ofte mye arbeid å ferdigstille videre bruk. Med ODS blir ferdigstillingsprosessen kortere og enklere.

Vi kan velge disse utskriftsformatene i ODS:

Listing	utskrift sendes til utskriftsvinduet (standard)
Html	html-fil
Output	lagrer utskrift på SAS-datasett etter gitte regler
Printer	utskrift styres direkte til skriver, anbefales ikke
Gif	grafikk lagres som gif-fil
Java	grafikk lagres som Java applets
ActiveX	grafikk lagres som ActiveX-fil
Xml	xml-fil (eksperimentell versjon)

De vanligste valgene vil være listing og html. Etter hvert vil xml bli mer og mer brukt.

ODS fungerer som en bryter. Vi kan slå den på og vi kan slå den av. Vi kan også ha flere brytere på samtidig, da får vi utskrift styrt til flere steder samtidig. Hvis vi ikke får noen utskrift i det hele tatt, kan det skyldes at alle ODS-bryterne er skrudd av. I så fall kommer det en advarsel i SASlog.

Det kan være en fordel å lage Html-filer når utskriften vår skal behandles videre i Excel eller Word. Når vi skal publisere tabeller, kan det også være en fordel å lage en html-fil før videre bearbeiding.

Et lite eksempel:

```
DATA bef;
INPUT @01 Fylke    $CHAR2.
      @04 Menn      6.
      @11 Kvinner   6.
      ;
      Folk = SUM(menn,kvinner);
CARDS;
01 119617 123968
02 224381 229109
03 240276 259417
04  91945  94173
05  90379  91783
06 114826 118141
07 102106 106581
08  80641  83216
09  50293  50859
10  75489  77064
11 181692 182649
12 212786 216037
14  54532  53258
15 121184 120788
16 128086 131091
17  63485  63300
18 119549 119731
19  75551  74737
20  38288  36591
;
ODS html FILE="innbygg98.html";
PROC PRINT DATA=bef NOOBS;
      TITLE 'Innbyggere i Norge 1998.' ;
ODS html CLOSE;
```

Dette programmet lager først et SAS datasett med 4 variable. Deretter slås ODS på for html-fil. I det vi slår på html-bryteren må vi navngi html-filen. Så skriver vi filen ut med PROC PRINT. Til slutt skruer vi av html-bryteren. Utskrift fra de programsetningene mellom de to ODS-setningene, havner i html-filen innbygg98.html.

De html-filene SAS lager kan vi åpne i Excel eller Word med kommandoene Fil og Åpne. Det forutsetter at html-filene er overført fra unix til windows.

ODS består av følgende setninger:

ODS EXCLUDE	Angir utskriftsobjekter som ikke skal skrives ut.
ODS HTML	Åpner, administrerer eller lukker html-filer.
ODS LISTING	Åpner, administrerer eller lukker utskrift til utskriftsvinduet.
ODS OUTPUT	Åpner, administrerer eller lukker utskrift til SAS datasett.
ODS PATH	Angir søkesti og søkerekkefølge for layout-definisjoner (fra PROC TEMPLATE).
ODS PRINTER	Åpner, administrerer eller lukker utskrift til skriver.
ODS PROCTITLE	Angir om prosdyrens navn skal skrives ut på utskriftsfiler (=ON) eller ei (=OFF). Virker ikke over alt.
ODS RTF	Åpner, administrerer eller lukker rtf-filer.
ODS SELECT	Angir utskriftsobjekter som skal skrives ut.
ODS TRACE	Skrur på eller av en beskrivelse av de utskriftsobjekter som er brukt. Beskrivelsen sendes til SASlog.
ODS VERIFY	Skrur på eller av om advarsler skal skrives til SASlog når vi prøver å bruke stiler som ikke er støttet av SAS.

For å få en oversikt over hvilke stildefinisjoner som finnes kan vi bruke prosedyren PROC TEMPLATE. Standardstilen vår, som er en tilpasset utgave av SAS sin standard, kan listes ut med følgende SAS-setninger:

```
PROC TEMPLATE ;
SOURCE styles.default;
RUN;
```

Utlistingen viser hvilke stiler som finnes, men sier lite om når de brukes.

3.10.8.1 ODS EXCLUDE

ODS <ODS-destinasjon> EXCLUDE utskriftsobjekt <(PERSIST)> | ALL | NONE ;

ODS-destinasjon	hva slags sted eller fil utskriften havner, kan være html, listing eller printer.
utskriftsobjekt	angir hvilke utskriftsobjekter som skal ekskluderes. Utskriftsobjektene navn kan vi finne ved å bruke ODS TRACE, se denne.
PERSIST	utskriftsobjektet ekskluderes helt til den overstyres med ODS SELECT, ODS EXCLUDE ALL, ODS EXCLUDE NONE eller en ODS EXCLUDE der utskriftsobjektet ikke har PERSIST-option tilknyttet. Ellers gjelder eksklusjoner bare til neste ODS EXCLUDE.
ALL	Ekskluderer alt fra destinasjonen.
NONE	Tar med alt fra destinasjonen.

3.10.8.2 ODS HTML

ODS HTML CLOSE | SHOW | html-filtype="filnavn" <(fil-options)> <options> ;

CLOSE	Lukker html-filen.
SHOW	Viser i SASlog seleksjoner av utskriftsobjekter definert i ODS SELECT eller ODS EXCLUDE.
html-filtyper	Åpner html-filer som skal lages. Vi kan lage forskjellige typer html-filer og disse er:
BODY=	html-filen utskriften skal styres til (det samme som FILE=).
CONTENTS=	angir navn på en html-fil som skal inneholde en innholdsfortegnelse over de utskrifter det lages html av.
FRAME=	angir navn på en ramme-fil som vil inneholde innholdsfortegnelse, sidebeskrivelse og selve utskriften. Krever at CONTENTS= eller PAGE= også angis.
PAGE=	angir navn på en html-fil som vil inneholde en beskrivelse av de forskjellige sidene i utskriften.
STYLESHEET=	Angir navn på et mal-sett (CSS-fil) som skal brukes.
Til html-filene kan vi angi følgende fil-options:	
NO_BOTTOM_MATTER	Gjør at det ikke skrives ut avsluttende html-kode til utskriftsfilen. Dette åpner for å skrive mer til samme html-fil senere.
NO_TOP_MATTER	Gjør at den html-kode som vanligvis starter er html-dokument ikke skrives ut. Dette gir oss mulighet til å legge til nye utskrifter på en html-fil. Filen vi legger til på må ha brukt option NO_BOTTOM_MATTER.
TITLE=	Tekst som plasseres mellom <TITLE> og </TITLE> i html-filen.
Andre fil-options:	DYNAMIC, INCLUDE og URL.
Options til ODS HTML-setningen:	
ANCHOR=	Angir et basisnavn for filreferanser. Referansene nummereres så fortløpende etter hvert som de blir brukt.
BASE=	Angir en adresse som blir brukt som basis for alle referanser.
HEADTEXT=	Tekst som plasseres mellom <HEAD> og </HEAD> i html-filen.
METATEXT=	Tekst som plasseres i <META> i html-filen.
NEWFILE=	Lager nye html-filer. NONE (standard) angir at all utskrift havner i samme html-fil inntil html-filen lukkes, OUTPUT at det lages en ny html-fil for hver utskrift som lages, PAGE at det skal ny side for hver side som skrives ut og PROC at det skal lages en ny html-fil for hvert PROC-steg. Når det lages nye html-filer, får de fortløpende nummerering i forhold til basisfilen.
PATH=	Angir en adresse som blir basis for html-filene som lages
STYLE=	Angir hvilken layout-stil som skal brukes. Standard er styles.default. Nye stiler kan lages med PROC TEMPLATE.
Andre options:	ATTRIBUTES, GFOOTNOTE, GPATH, GTITLE, PARAMETERS, RECORD_SEPARATOR, TRANTAB, TYPE=.

3.10.8.3 ODS LISTING

ODS LISTING CLOSE | SHOW | <sidedeling=tall | DATA | PAGE> ;

CLOSE	Lukker utskrift til utskriftsvinduet.
SHOW	Viser i SASlog seleksjoner av utskriftsobjekter definert i ODS SELECT eller ODS EXCLUDE.
sidedeling=	Angir hvordan SAS skal dele sidene i rader og kolonner når utskriften går over mer enn en side. Sidebredde og sidelengde defineres med systemoption LINESIZE og PAGESIZE. Ved å angi et tall deles siden etter det angitte antall observasjoner. Med DATA deles siden etter hvordan dataene som skal skrives ut er lagret i minnet og med PAGE prøver SAS å lage sider som rommer mest mulig. Hvis sidebredden er mer enn 200 settes DATAPANEL=200.

3.10.8.4 ODS OUTPUT

ODS OUTPUT CLEAR | CLOSE | SHOW |
utskriftsobjekt <(MATCH_ALL<=makro-variabelnavn> PERSIST=PROC | RUN)><=SAS datasett>;

CLEAR	Setter ODS Output til EXCLUDE ALL.
CLOSE	Lukker utskrift til SAS datasett.
SHOW	Viser i SASlog seleksjoner av utskriftsobjekter definert i ODS SELECT/EXCLUDE.
utskriftsobjekt	angir hvilke utskriftsobjekter som skal skrives ut. Utskriftsobjektene navn kan vi finne ved å bruke ODS TRACE, se denne.
MATCH_ALL	Lager et nytt SAS datasett for hver utskrift. Nye html-filer får fortløpende nummerering i forhold til navnet det første gis. Angis et makrovariabelnavn lagres navnene på alle SAS datasettene i makrovariabelen. Denne kan så brukes senere.
PERSIST=	Angir når SAS datasett som lages vha. ODS skal lukkes og når utskriftsobjekter fjernes fra seleksjonslister. Brukes bare sammen med MATCH_ALL. Med PROC beholdes seleksjonslister helt til vi selv endrer disse med ODS SELECT eller ODS EXCLUDE. Med RUN holdes SAS datasettene åpne helt til vi selv lukker dem.
SAS datasett	Navn på SAS datasettet som lages.

3.10.8.5 ODS PATH

ODS PATH <libref>stildefinisjoner <(READ | UPDATE | WRITE)> ;

libref	SAS-navn på området der stilene er lagret.
stildefinisjoner	Samlenavn på stildefinisjonene.
READ	Angir at det bare gis lesetilgang.
UPDATE	Angir at det gis adgang til oppdatering av stiler.
WRITE	Angir at det gis skrives til ny stil hver gang og alle gamle stiler slettes. Bør ikke brukes.

Som standardområder for stildefinisjoner er disse definert: sasuser.templat(update) og sashelp.tmplmst(read).

3.10.8.6 ODS PRINTER

ODS PRINTER CLOSE | SHOW | options ;

CLOSE	Lukker utskrift til skriver.
SHOW	Viser i SASlog seleksjoner av utskriftsobjekter definert i ODS SELECT/ODS EXCLUDE.
options	Options for utskriften:
COLOR=	YES åpner for farger, NO gir svart/hvitt.
FILE=	Angir en fil det skal skrives til istedenfor en skriver.
FONTSCALE=	Angir en skaleringsprosent for skriftstørrelse.
POSTSCRIPT	Lager postscript-kode før utskrift.
PRINTER=	Angir en skriver det skal skrives til.
SAS	Angir at det skal brukes skriverdriverne som SAS har tilgjengelig.
STYLE=	Angir stildefinisjon som skal brukes. Standard er styles.printer.
UNIFORM	Sørger for like kolonnebredder for alle sider som skrives ut.
Andre options:	ANCHOR, AUTHOR, BASE, KEYWORDS, PDFMARK, PDFNOTE, SUBJECT, TITLE.

3.10.8.7 ODS PROCTITLE

ODS PROCTITLE <=ON | OFF> ;

ON Gjør at prosedyrens navn kommer på utskriften for de prosedyrene der det er standard.
 OFF Undertrykker prosedyrens navn ved utskrift.

3.10.8.8 ODS RTF

ODS RTF < CLOSE | SHOW | rtf-fil="filnavn" <options> ;

CLOSE Lukker rtf-filen.
 SHOW Viser i SASlog seleksjoner av utskriftsobjekter definert i ODS SELECT eller ODS EXCLUDE.

Options til ODS RTF-setningen:

ANCHOR= Angir et basisnavn for filreferanser. Referansene nummereres så fortløpende etter hvert som de blir brukt.
 AUTHOR= Angir forfatter av rtf-filen. Havner i overskriften i rtf-filen.
 BASE= Angir et prefiks som skal brukes til alle rtf-filene som blir laget.
 FILE= Navn på rtf-filen der utskriften havner.
 KEEPEN Minimaliserer oppsplitting av tabeller over flere sider
 NEWFILE= Lager nye rtf-filer. NONE (standard) angir at all utskrift havner i samme rtf-fil inntil rtf-filen lukkes, OUTPUT at det lages en ny rtf-fil for hver utskrift som lages, PAGE at det skal ny side for hver side som skrives ut og PROC at det skal lages en ny rtf-fil for hvert PROC-steg. Når det lages nye rtf-filer, får de fortløpende nummerering i forhold til basisfilen.
 STYLE= Angir hvilken layout-stil som skal brukes.
 TITLE= Angir overskrift til rtf-filen.
 Andre options: GFOOTNOTE, GTITLE, OPERATOR, RECORD_SEPARATOR, TRANTAB.

3.10.8.9 ODS SELECT

ODS <ODS-destinasjon> SELECT utskriftsobjekt <(PERSIST)> | ALL | NONE ;

ODS-destinasjon hva slags sted eller fil utskriften havner, kan være html, listing eller printer.
 utskriftsobjekt angir hvilke utskriftsobjekter som skal selekteres. Utskriftsobjektene navn kan vi finne ved å bruke ODS TRACE, se denne.
 PERSIST utskriftsobjektet selekteres helt til den overstyrer med ODS EXCLUDE, ODS SELECT ALL, ODS SELECT NONE eller en ODS SELECT der utskriftsobjektet ikke har PERSIST-option tilknyttet. Ellers gjelder seleksjoner til neste ODS SELECT.
 ALL Selekterer alt fra destinasjonen.
 NONE Fjerner alt fra destinasjonen.

3.10.8.10 ODS TRACE

ODS TRACE OFF | ON </ <LABEL> <LISTING> > ;

OFF Skru av utlisting av informasjon om de utskriftsobjekter som brukes.
 ON Skru på utlisting av informasjon om de utskriftsobjekter som brukes.
 LABEL Gir litt mer informasjon om utskriftsobjektet.
 LISTING Styrer informasjonen til utskriftsvinduet istedenfor til SASlog.

3.10.8.11 ODS VERIFY

ODS VERIFY ON | OFF | ERROR ;

ON Angir at det skrives ut advarsler når stildefinsjoner ikke finnes.
 OFF Angir at det ikke skrives ut advarsler når stildefinsjoner ikke finnes.
 ERROR Angir at det skrives ut advarsler når stildefinsjoner ikke finnes og det lages ingen utskrift.

3.10.8.12 ODS XML

ODS XML CLOSE | SHOW | xml-filtype="filnavn" <options> ;

CLOSE Lukker html-filen.
 SHOW Viser i SASlog seleksjoner av utskriftsobjekter definert i ODS SELECT eller ODS EXCLUDE.

xml-filtyper Åpner xml-filer som skal lages. Vi kan lage forskjellige typer xml-filer og disse er:
 BODY= xml-filen utskriften skal styres til (det samme som FILE=).
 CONTENTS= angir navn på en html-fil som skal inneholde en innholdsfortegnelse over de utskrifter det lages xml av.
 FRAME= angir navn på en ramme-fil som vil inneholde innholdsfortegnelse, sidebeskrivelse og selve utskriften. Krever at CONTENTS= eller PAGE= også angis.
 PAGE= angir navn på en xml-fil som vil inneholde en beskrivelse av de forskjellige sidene i utskriften.
 STYLESHEET= Angir navn på et mal-sett (CSS-fil) som skal brukes.

Options til ODS XML-setningen:

ANCHOR= Angir et basisnavn for filreferanser. Referansene nummereres så fortløpende etter hvert som de blir brukt.
 AUTHOR= Angir forfatter av xml-filen. Havner i overskriften i xml-filen.
 BASE= Angir en adresse som blir brukt som basis for alle referanser.
 HEADTEXT= Tekst som plasseres mellom <HEAD> og </HEAD> i xml-filen.
 METATEXT= Tekst som plasseres i < /META> i xml-filen.
 NEWFILE= Lager nye xml-filer. NONE (standard) angir at all utskrift havner i samme xml-fil inntil xml-filen lukkes, OUTPUT at det lages en ny xml-fil for hver utskrift som lages, PAGE at det skal ny side for hver side som skrives ut og PROC at det skal lages en ny xml-fil for hvert PROC-steg. Når det lages nye xml-filer, får de fortløpende nummerering i forhold til basisfilen.
 PATH= Angir en adresse som blir basis for xml-filene som lages
 STYLE= Angir hvilken layout-stil som skal brukes. Standard er styles.default. Nye stiler kan lages med PROC TEMPLATE.
 Andre options: ARCHIVE, ATTRIBUTES, CHARSET, CODE, ENCODING, GFOOTNOTE, GPATH, GTITLE, OPERATOR, PARAMETERS, RECORD_SEPARATOR, TRANTAB, TYPE.

3.10.9 OPTIONS

Endrer verdiene til en eller flere av SAS system options. Disse forblir slik vi setter dem helt til vi endrer dem igjen. Beskrivelse av SAS system options, se side 40. Vi plasserer OPTIONS-setningen før et DATA-steg eller før en prosedyre.

OPTIONS option-1 ... <option-n> ;

3.10.10 PAGE

Starter på ny side i SASlog.

PAGE ;

3.10.11 %PUT

Brukes for å skrive innholdet i en eller flere makrovariable til SASlog.

%PUT <tekst | &makrovariabel | _ALL_ | _AUTOMATIC_ | _GLOBAL_ | _LOCAL_ | _USER_> ;

tekst	Teksten skrives ut.
&makrovariabel	Makrovariabelens innhold skrives ut.
ALL	Alle makrovariable skrives ut.
AUTOMATIC	Alle automatisk definerte makrovariable skrives ut.
GLOBAL	Alle globale makrovariable skrives ut.
LOCAL	Alle lokale makrovariable skrives ut.
USER	Alle brukerdefinerte makrovariable skrives ut.

3.10.12 RGET

Brukes til å hente SASlog til nåværende sesjon fra sesjoner startet med RSUBMIT-blokker som kjøres parallelt. Se eksempel under RSUBMIT.

RGET oppgavenavn ;

3.10.13 RSUBMIT

Vi bruker RSUBMIT sammen med ENDRSUBMIT til å avgrense SAS-setninger som skal eksekveres på en tjenermaskin. De SAS-setningene som står mellom RSUBMIT og ENDRSUBMIT blir eksekvert på tjenermaskinen når programmet kjøres med vanlig SUBMIT-kommando. Se mer om klient/tjener bruk på side 257. RSUBMIT kan også brukes til parallellprosessering på en maskin, se eksempel under.

RSUBMIT options ;

Options til RSUBMIT-setningen:

oppgavenavn	Navn på den kjøringen som eksekveres med RSUBMIT-blokken.
WAIT=	Angir om SAS skal vente med videre kjøring til etter at RSUBMIT er gjennomført (YES) eller ei (NO). WAIT=NO gjør at vi kan kjøre flere SAS-program parallelt.
MACVAR=	Legger status for kjøringen i en makrovariabel. Denne får verdi 0 (RSUBMIT er ferdig), 1 (RSUBMIT gikk feil) eller 2 (RSUBMIT pågår fortsatt).
CONNECTSTATUS=	Angir om statusvindu skal vises (YES) når dataoverføring gjøres innenfor RSUBMIT-blokken.
SASCMD=	Angir kommando for å starte SAS-sesjonen, f.eks. /local/sas81/sas. Settes helst sentralt vha. system option SASCMD, se side 45.

Andre options: PASSWORD, PERSIST, SCRIPT, SYSRPUTSYNC, USER.

Eksempel på parallellprosessering:

```
OPTIONS autosignon= YES sascmd="/local/sas81/sas";
LIBNAME bef '~krl/testsas8';
RSUBMIT oppg1 WAIT=no ;
LIBNAME bef '~krl/testsas8';
PROC SORT DATA=bef.bef1998 ;
  BY kommune;
RUN;
ENDRSUBMIT;

RSUBMIT oppg2 WAIT=no ;
LIBNAME bef '~krl/testsas8';
PROC SORT DATA=bef.bef1999 ;
  BY kommune;
RUN;
ENDRSUBMIT;

RGET oppg1;
RGET oppg2;
WAITFOR _ALL_ oppg1 oppg2;

DATA bef;
  MERGE bef.bef1998
        bef.bef1999
  ;
  BY kommune;
RUN;
```

3.10.14 SYSTASK

Utfører operativsystemkommandoer uavhengig av SAS-sesjonen. Disse oppgavene (kommandoene) kjøres i bakgrunnen som egne program mens SAS-programmet går. SYSTASK kan ikke brukes til å starte nye interaktive sesjoner, da må vi bruke X-setningen isteden.

SYSTASK COMMAND "operativsystemkommando" <WAIT|NOWAIT> <TASKNAME=oppgavenavn>
<MNAME=makrovariabel><STATUS=makrovariabel><SHELL<="shell">><CLEANUP>;

SYSTASK LIST < ALL | oppgavenavn> <STATE> <STATVAR>;

SYSTASK KILL oppgavenavn-1 ... <oppgavenavn-n>;

SYSTASK COMMAND starter operativsystemkommandoer. Følgende options kan brukes:

WAIT	Angir at SAS-sesjonen skal vente med videre kjøring til kommandoen er utført.
NOWAIT	An gir at SAS-sesjonen skal kjøre videre uavhengig av eksekvering av kommandoen.
TASKNAME=	Navngir kommandoen.
MNAME=	Navn på en makrovariabel som vil inneholde oppgavenavnet gitt med TASKNAME= eller navnet som gis av SAS hvis TASKNAME= ikke er brukt.
STATUS=	Navn på en makrovariabel som vil inneholde oppgavens status.
SHELL=	Angir hvilket unix-shell som skal benyttes. Vanlige shell er sh, csh, ksh og bash. Utelater vi denne brukes standard shell.
CLEANUP	Gjør at oppgaven fjernes fra oppgavelisten og at oppgavenavnet kan brukes igjen.

SYSTASK LIST viser en oversikt over alle kommandoer som utføres eller er utført uten at den er fjernet fra oppgavelisten. Vi kan bruke disse options:

ALL	Lister alle aktive oppgaver
STATE	Gir en statusoversikt for oppgavene, enten Start Failed, Running eller Complete.
STATVAR	Viser navnet på makrovariabel definert i STATUS-option i SYSTASK COMMAND.

SYSTASK KILL stopper oppgaver som er startet.

3.10.15 *TITLE*

Lager overskrift. Vi kan ha opptil 10 linjer overskrift. Overskriften vil komme på alle sider utskrift helt til vi opphever den med en TITLE-setning uten tekst. En TITLE-setning overskriver en tidligere med samme linjenummer og nullstiller alle overskriftslinjer med større nummer. Maksimal lengde er 254 tegn pr. overskrift ved interaktiv kjøring og 200 ved batch-kjøring. Overskriften vil kuttes dersom den er lengre en maksimal linjebredde, definert med system option LINESIZE= (se side 43). Når vi lager html-filer fra SAS kan vi knytte en forbindelse til et html-dokument ved å bruke LINK-option.

TITLE<n> <LINK="html-fil"> <Overskrift>;

Eksempel:

```
TITLE 'Tabell 1. Innførselen av varer i juli og januar-juli 1994 og 1995.';
TITLE3 ; /* Nullstiller alle tittellinjer fra 3-10 */
```

3.10.16 *WAITFOR*

Sørger for at SAS-sesjonen settes på vent til operativsystemkommandoene som angis er utført. Disse oppgavene (kommandoene) må være startet med SYSTASK-setningen.

WAITFOR <_ANY_ | _ALL_> oppgavenavn-1 ... <oppgavenavn-n> <TIMEOUT=sekunder>;

<u>_ANY_</u>	Angir at SAS-sesjonen skal fortsette så fort en av oppgavene SAS venter på er ferdig.
<u>_ALL_</u>	Angir at SAS-sesjonen skal vente helt til alle oppgavene SAS venter på er ferdige.
TIMEOUT=	Antall sekunder SAS-sesjonen skal vente før den fortsetter, uavhengig av om oppgavene er ferdige eller ei.

3.10.17 *X*

Utfører en operativsystem-kommando fra SAS. Kommandoene varierer fra et operativsystem til et annet. Typiske anvendelser for X-setningen er kopiering, sletting og listing av filer. Med X kan vi også eksekvere andre program fra SAS, f.eks. komprimering av filer. Det er viktig å være klar over at X-kommandoen uføres uansett, uavhengig av om den står inne i en IF-test eller ikke. Ønsker vi å utføre en operativsystem-kommando hvis en betingelse slår til, må vi bruke CALL SYSTEM isteden, se side 57.

X <'kommando'>;

3.11 PROC-steget

I SAS finnes det mange prosedyrer. Prosedyrene brukes til forskjellige mer eller mindre avanserte oppgaver. Prosedyrene er laget slik at vi som programmere har muligheter til å påvirke hva prosedyren skal gjøre. Dette gjøres innenfor rammen av prosedyrens SAS-setninger og forskjellige parametre. Disse SAS-setningene og parametrene varierer fra prosedyre til prosedyre.

Hvilken oppgave vi skal utføre, avgjør hvilken prosedyre vi bør bruke, om det da finnes en prosedyre for oppgaven, og det gjør det ofte. For utskrift fra et SAS datasett bruker vi PROC PRINT, frekvenstabeller lager vi vha. PROC FREQ, sortering av datasett gjøres med PROC SORT osv. Det er ikke alltid like lett å finne ut om det finnes passende prosedyrer, og i såfall hvilke vi skal bruke til våre oppgaver. Her følger en beskrivelse av noen av de vanligste og viktigste prosedyrene. I tillegg til disse finnes det som sagt mange flere, blant annet for regresjonsanalyse (PROC REG og PROC LOGISTIC) og matrisebearbeiding (PROC IML).

3.11.1 PROC APPEND

PROC APPEND bruker vi for å hekte et datasett til på slutten av et annet. Ved å bruke PROC APPEND istedenfor et DATA-steg, slipper vi å lese basisdatasettet (det datasettet som skal få nye observasjoner). Hvis dette er stort vil vi spare både tid og plass på å bruke PROC APPEND.

```
PROC APPEND BASE=SAS datasett < DATA=SAS datasett <(datasett-options)>> <APPENDVER=V6>  
    <FORCE> ;  
WHERE where-uttrykk ;
```

Basisdatasettet nevnt i BASE= er det som skal få tilføyd observasjoner. Disse observasjonene hentes fra tilføyingsdatasettet nevnt med DATA=. Dersom datasettet som er nevnt i BASE= ikke finnes, vil det bli opprettet. Det nye SAS datasettet vil bli lagret i det SAS lagringsformatet BASE-datasettet har. SAS datasett med annet lagringsformat (lagret med en annen SAS-versjon) blir automatisk konvertert. Dette gjelder ikke på tvers av unix-plattformer.

Prøver vi å slå sammen to datasett med forskjellige variable, må vi bruke FORCE-option for å få det utført. Variable som ikke er i basisdatasettet vil ikke bli tatt med. Hvilke dette er kommer som en melding (MESSAGE) i SASlog. Har variablene forskjellige attributter (lengde, dataformat osv.), vil attributtene fra basisdatasettet bli brukt på de nye observasjonene også. Er variabelenes lengde kortere i basisdatasettet eller av en annen type (karakter i det ene , numerisk i det andre), må vi bruke FORCE-option for å utføre sammenslåingen. Dette medfører at alle observasjoner med forskjellig type fra tilføyingsdatasettet settes til MISSING, dessuten risikerer vi at verdier på variable med forskjellig lengde kan bli trunkert. I versjon 8 er det noe forskjell fra versjon 6 når det gjelder å oppdatere indekserte SAS datasett. I versjon 6 ble indeksen oppdatert for hver observasjon som ble lagt til, noe som brukte mye ressurser. I versjon 8 oppdateres indeksene først når alle observasjoner er lagt til. Ønsker vi allikevel å bruke versjon 6 sin måte å oppdatere SAS datasettet, kan vi bruke option APPENDVER=V6. Hvis SAS datasettet som skal legges til har knyttet gyldighetsregler til seg, vil ikke disse bli kopiert over til BASE-datasettet, med mindre BASE-datasettet ikke eksisterer. Da vil både observasjoner og gyldighetsregler legges til.

Eksempel:

```
PROC APPEND BASE=befolk.persreg DATA=fodt9509;
```

SAS-setning til PROC APPEND:

WHERE

Selekterer hvilke observasjoner som skal legges til. Utføres for de observasjoner som tilfredsstiller WHERE-betingelsen i datasettet som skal hektes på. WHERE-setningen får ingen betydning for dataene i basisdatasettet som sammenlignes. Beskrivelse av WHERE-setningen finnes på side 91.

3.11.2 PROC COMPARE

Vi bruker PROC COMPARE til å sammenligne to datasett eller variable i ett datasett, for å se om det er forskjeller eller de er like. De to datasettene som sammenlignes, betegnes som basisdatasett og sammenligningsdatasett. Når vi sammenligner to datasett, vil først datasett attributter (TYPE= og LABEL=) sammenlignes. Videre sjekkes det om datasettene inneholder de samme variablene og om disse har like attributter (type, label, utskriftsformat, innlesingsformat). Så sjekkes observasjon for observasjon for å finne forskjeller i variablenes verdier.

Vanligvis lages det en rapport som viser hvor like datasettene er. I tillegg til denne rapporten kan vi velge å få meldinger i SASlog, få rapporten ut som et nytt SAS datasett og/eller gi en numerisk returkode som legges i systemvariabelen &SYSINFO. Dette tallet er summen av følgende tall:

- 1 Datasettenes LABEL er forskjellige
- 2 Datasettenes TYPE er forskjellige
- 4 Variable har forskjellig innlesingsformat
- 8 Variable har forskjellig utskriftsformat
- 16 Variable har forskjellig lengde
- 32 Variable har forskjellig LABEL
- 64 Basisdatasettet har observasjoner som ikke er i sammenligningsdatasettet
- 128 Sammenligningsdatasettet har observasjoner som ikke er i basisdatasettet
- 256 Basisdatasettet har BY-gruppe som ikke er i sammenligningsdatasettet
- 512 Sammenligningsdatasettet har BY-gruppe som ikke er i basisdatasettet
- 1024 Basisdatasettet har variable som ikke er i sammenligningsdatasettet
- 2048 Sammenligningsdatasettet har variable som ikke er i basisdatasettet
- 4096 Forskjellige verdier for sammenlignede variable
- 8192 Variable som skal sammenlignes har forskjellig type
- 16384 BY-variable matcher ikke
- 32768 Fatal feil, sammenligning avbrutt!

```
PROC COMPARE <options>;
  WHERE where-uttrykk ;
  BY <NOTSORTED> variabel-1 <... <NOTSORTED> variabel-n <NOTSORTED>> ;
  ID <DESCENDING> variabel-1 ... <<DESCENDING> variabel-n> <NOTSORTED> ;
  VAR variable ;
  WITH variable ;
```

Options til PROC COMPARE-setningen:

ALLOBS

For numeriske variable vil alle forskjeller listes i rapporten, selv om de dømmes som like.

ALLSTATS

Lager en todelt tabell med statistiske beregninger for alle par med sammenlignede variable. Den første delen inneholder N, MEAN, STD, MAX, MIN, STDERR, T og PROB>|T|. Del to inneholder:

- NDIF antall observasjoner som regnes som forskjellige
- DIFMEANS forskjellen mellom gjennomsnittet av variablene som sammenlignes (verdi og prosenter)
- R korrelasjon for ikke MISSING-verdier av variablene som sammenlignes
- RSQ kvadratet av korrelasjonen for ikke MISSING-verdier av variablene som sammenlignes

ALLVARS

Tar med alle observasjoner i rapporten, selv om variablene har like verdier.

BASE= SAS datasett

SAS datasett som brukes som basis for sammenligningen.

BRIEFSUMMARY

Lager en mindre utgave av sammenligningsrapporten

COMPARE=SAS datasett

Datasettet som skal sammenlignes med basis datasettet. Hvis COMPARE utelates, vil sammenligningen foregå mellom variable i et datasett (listet i VAR- og WITH-setning).

CRITERION=n

Bestemmer hvor stort avvik numeriske verdier kan ha for å regnes som like. Standardverdi for avviket er 0.00001. Settes avviket negativt, brukes datamaskinens nøyaktighetsverdi.

DATA= SAS datasett

SAS datasett som brukes som basis for sammenligningen.

ERROR

Skriver en feilmelding til SASlog når forskjeller finnes. ERROR overstyrer en eventuell WARNING-option.

FUZZ=n

Kan brukes til å overse ubetydelige forskjeller i utskriften. Er forskjellen mindre enn n, vil differansen settes til 0 i utskriften, men de vil fortsatt regnes som forskjellige. Er prosentavviket mindre enn n, blankes dette i utskriften. n må være mellom 0 og 1, standardverdi er 0.

LIST

Lister alle observasjonsnr og eventuell ident til observasjoner som bare finnes i et av datasettene som sammenlignes. Lister også navnene til de variablene som bare finnes på et av datasettene.

LISTBASE

Lister alle observasjonsnr og eventuell ident til observasjoner som bare finnes i basisdatasettet. Lister også navnene til de variablene som bare finnes på dette datasettet.

LISTBASEOBS

Lister alle observasjonsnr og eventuell ident til observasjoner som bare finnes i basisdatasettet.

LISTBASEVAR

Lister navnene til de variablene som bare finnes på dette basisdatasettet.

LISTCOMP

Lister alle observasjonsnr og eventuell ident til observasjoner som bare finnes i sammenligningsdatasettet. Lister også navnene til de variablene som bare finnes på dette datasettet.

LISTCOMPOBS

Lister alle observasjonsnr og eventuell ident til observasjoner som bare finnes i sammenligningsdatasettet.

LISTCOMPVAR

Lister navnene til de variablene som bare finnes på dette sammenligningsdatasettet.

LISTEQUALVAR

Lister alle variable som er like for alle observasjoner. Standard er NOLISTEQUAL.

LISTOBS

Lister alle observasjonsnr og eventuell ident til observasjoner som bare finnes i et av datasettene som sammenlignes.

LISTVAR

Lister navnene til de variablene som bare finnes på et av datasettene som sammenlignes.

MAXPRINT=total | (pr-variabel,total)

Angir maksimalt antall forskjeller som skrives ut i alt (total) og pr. variabel (pr-variabel). Standardverdi for total er 500 (32000 hvis ALLOBS option eller ALLVAR og TRANSPOSE options brukes) og pr-variabel 50 (1000 hvis ALLOBS option eller ALLVAR og TRANSPOSE options brukes).

METHOD=ABSOLUTE | EXACT | PERCENT | RELATIVE

Forskjellige metoder for å bedømme om numeriske verdier regnes som like.

NODATE

Undertrykker utskrift av datoer for når datasettene som sammenlignes er laget og sist endret.

NOMISSBASE

Regner MISSING-verdier lik en hvilken som helst verdi i basisdatasettet. Det betyr at MISSING-verdier i basisdatasettet regnes for å være lik verdiene den tilsvarende variabel har i sammenligningsdatasettet. Kan f.eks. brukes til å finne ut hvilke oppdateringer som ville bli utført ved å bruke UPDATE-setningen i DATA-steget på de to datasettene.

NOMISSCOMP

Regner MISSING-verdier lik en hvilken som helst verdi i sammenligningsdatasettet. Det betyr at MISSING-verdier i sammenligningsdatasettet regnes for å være lik verdiene den tilsvarende variabel har i basisdatasettet.

NOMISSING

Regner MISSING-verdier lik en hvilken som helst verdi i både basis- og sammenligningsdatasettet. Det betyr at hvis en variabel er MISSING i et av datasettene vil den alltid regnes som lik verdien til den variabelen den sammenlignes med.

NOOBS

Undertrykker utskrift av observasjoner med forskjeller for sammenlignede variabelverdier.

NOPRINT

Undertrykker utskrift av sammenligningsrapporten

NOSUMMARY

Undertrykker den summariske informasjonen fra sammenligningsrapporten.

NOTE

Skriver en melding til SASlog om det er funnet forskjeller eller ei.

NOVALUES

Undertrykker utskrift av verdisammenligningene fra sammenligningsrapporten.

OUT=SAS datasett

Skriver forskjellene ut til et SAS datasett. Som standard skrives det opptil 4 observasjoner for hvert par med sammenlignede variable. Disse observasjonene skilles med verdien til variabelen `_TYPE_` som kan ha disse verdiene:

BASE	Verdiene til variablene fra basisdatasettet
COMPARE	Verdiene til variablene fra sammenligningsdatasettet
DIF	Forskjellene mellom variablene som sammenlignes. For numeriske verdier regnes differansen. For karaktervariable sammenlignes posisjon for posisjon og markeres med . når de er like og X når de er forskjellige.
PERCENT	Prosentvis differanse for numeriske verdier. For karakterer settes PERCENT lik DIF, se over.

Foruten `_TYPE_` variabelen skrives alle BY- og ID variable til utdatasettet. Videre skrives alle variable som sammenlignes ut. I tillegg lages variabelen `_OBS_` som inneholder observasjonsnr fra innlesingen.

OUTALL

Skriver ut observasjoner også når de sammenlignede variablene er like.

OUTBASE

Skriver en observasjon til utdatasettet for hvert par med sammenlignede variable. Verdien til basisdatasettets variabel skrives ut. Variabelen `_TYPE_` på utdatasettet får verdien BASE.

OUTCOMP

Skriver en observasjon til utdatasettet for hvert par med sammenlignede variable. Verdien til sammenligningsdatasettets variabel skrives ut. Variabelen `_TYPE_` på utdatasettet får verdien COMPARE.

OUTDIF

Skriver en observasjon til utdatasettet for hvert par med sammenlignede variable. Differansen mellom variablenes verdier skrives ut. Variabelen `_TYPE_` på utdatasettet får verdien DIF.

OUTNOEQUAL

Skriver ikke ut noen observasjon til utdatasettet når alle sammenlignede verdier regnes som like. Hvis noen variable regnes som like og noen som ulike, gis differansen og prosentavviket til de som regnes som like .E som en spesial MISSING-verdi.

OUTPERCENT

Skriver en observasjon til utdatasettet for hvert par med sammenlignede variable. er Prosentvis differanse mellom variablenes verdier skrives ut. Variabelen `_TYPE_` på utdatasettet får verdien PERCENT.

OUTSTATS=SAS datasett

Skriver ut forskjellige statistiske beregninger på utdatasettet som defineres med OUTSTATS. Det lages en observasjon for hver statistisk beregning. Disse skilles med variablene `_TYPE_` som har N, MEAN, STD, MIN, MAX, STDERR, T, PROBT, NDIF, DIFMEANS, R og RSQ som mulige verdier (se option ALLSTATS). Andre variable som lages er:

<code>_VAR_</code>	Navnet på variabelen på basisdatasettet som beregningene er basert på
<code>_WITH_</code>	Navnet på variabelen på sammenligningsdatasettet som beregningene er basert på
<code>_BASE_</code>	Resultatet av beregningen gjort på variabelen i basisdatasettet
<code>_COMP_</code>	Resultatet av beregningen gjort på variabelen i sammenligningsdatasettet
<code>_DIF_</code>	Resultatet av beregningen gjort på differansen mellom de sammenlignede variablene.
<code>_PCTDIF_</code>	Resultatet av beregningen gjort på prosentvis avvik mellom de sammenlignede variablene.

PRINTALL

ALLVARS, ALLOBS, ALLSTATS, LISTALL og WARNING som en option.

STATS

Lager samme statistiske tabeller som med ALLSTATS option, men ikke for variable uten forskjeller.

TRANSPPOSE

Skriver rapporten ut pr. observasjon istedenfor pr. variabel.

WARNING

Skriver en advarsel til SASlog når forskjeller finnes.

SAS-setninger til PROC COMPARE:

WHERE

Selekterer hvilke observasjoner som skal sammenlignes. Bare de som tilfredsstiller WHERE-betingelsen sammenlignes. Beskrivelse av WHERE-setningen finnes på side 91.

BY

Med BY-setningen vil sammenligningen utføres etter BY-variablene. Det lages en rapport for hver verdi av BY-variablene. BY-variablene må være i basisdatasettet. Hvis ingen av BY-variablene finnes i sammenligningsdatasettet, sammenlignes hver BY-gruppe i basisdatasettet med hele sammenligningsdatasettet. Er bare noen av BY-variablene i sammenligningsdatasettet, vil ikke sammenligningen bli utført. Det samme gjelder når korresponderende BY-variable har forskjellig type i de datasettene som skal sammenlignes.

ID

Variable listet i ID-setningen brukes for å sammenligne datasettene etter ident. ID minner om BY, men det lages bare en rapport. ID-variablene må være med i basisdatasettet. Hvis en ID-variabel ikke finnes i sammenligningsdatasettet, blir den ikke brukt i sammenligningen (men den tas med på et eventuelt utdatasett). ID-variablene skal til sammen danne en unik nøkkel, hvis ikke skrives det feilmelding om dubletter og bare den første vil bli brukt i sammenligningen.

VAR

Her lister vi de variablene vi ønsker å sammenligne. Utelater vi VAR-setningen, vil alle variable med like navn i datasettene sammenlignes (bortsett fra variable listet i BY og/eller ID-setning).

WITH

Hvis variablene som skal sammenlignes har forskjellige navn, må vi bruke WITH for å fortelle hvilke variable som skal sammenlignes. Variablene må listes i samme rekkefølge som i VAR-setningen.

Eksempel 1:

Vi har 2 forskjellige utgaver av data for barnevernet og ønsker å se hvilke forskjeller det er på datasettene. Alle forskjeller er interessante, så vi lager en generell sammenligning av alle variablene.

```
PROC COMPARE BASE=rev.saml95mr COMPARE=rev.saml95ms; ;  
  TITLE 'Sammenligner to utgaver av barnevernsfila' ;  
RUN;
```

Dette lille programmet gjør mange sammenligninger og gir oss mye informasjon i flere deler:

```

Sammenligner to utgaver av barnevernsfila

COMPARE Procedure
Comparison of REV.SAML95MR with REV.SAML95MS
(Method=EXACT)

Data Set Summary

Dataset                Created          Modified  NVar    NObs
-----
REV.SAML95MR  09OCT96:15:33:21  09OCT96:15:34:34  637    41686
REV.SAML95MS  15OCT96:14:46:29  15OCT96:14:47:37  637    41686

Variables Summary

Number of Variables in Common: 637.

Observation Summary

Observation      Base  Compare
-----
First Obs        1      1
First Unequal    409    409
Last  Unequal    41678  41678
Last  Obs        41686  41686

Number of Observations in Common: 41686.
Total Number of Observations Read from REV.SAML95MR: 41686.
Total Number of Observations Read from REV.SAML95MS: 41686.

Number of Observations with Some Compared Variables Unequal: 235.
Number of Observations with All Compared Variables Equal: 41451.

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 629.
Number of Variables Compared with Some Observations Unequal: 8.
Number of Variables with Missing Value Differences: 2.
Total Number of Values which Compare Unequal: 301.
Maximum Difference: 0.
    
```

Den første delen sier hvor mange observasjoner det er på de to datasettene, hvor mange felles variable de har, samt en oversikt over sammenligningen. Den neste delen oppsummerer variablene som har forskjellige verdier:

```

Sammenligner to utgaver av barnevernsfila

COMPARE Procedure
Comparison of REV.SAML95MR with REV.SAML95MS
(Method=EXACT)

Variables with Unequal Values

Variable  Type  Len  Label                                Max Miss
                                Ndif Dif Dif
-----
UNDAVSL   NUM    8   Dato undersøkelsen ble avsluttet      63  0  63
UNDAVSLC  CHAR    6   Dato undersøkelsen ble avsluttet      63   0
RES_1     CHAR    1   Resultat av undersøkelsen (1)        162  0
RES_2     CHAR    1   Resultat av undersøkelsen (2)         6   0
RESD_2    CHAR    4   2. Dato for res. av undersøkelsen      1   0
RES_3     CHAR    1   Resultat av undersøkelsen (3)         4   0
RESD_3    CHAR    4   3. Dato for res. av undersøkelsen      1   0
RES_4     NUM    8                                1  0  1
    
```

Den siste delen ser nærmere på forskjellene. For hver variabel med forskjeller, lages det en liste som viser hvilke observasjoner forskjellene gjelder og hvilke forskjeller det er (for de 50 første). For numeriske variable lages en slik liste:

Value Comparison Results for Variables				
Obs	Dato undersøkelsen ble avsluttet			
	Base UNDAVSL	Compare UNDAVSL	Diff.	% Diff
1559	13148	.	.	.
1612	12795	.	.	.
1726	13148	.	.	.
1738	13148	.	.	.
1743	13148	.	.	.
1763	13148	.	.	.
1990	13005	.	.	.
2636	13122	.	.	.
2659	13117	.	.	.
4239	13138	.	.	.
5385	13148	.	.	.
5389	13148	.	.	.

Karaktervariable må nøye seg med en liste av denne typen:

Obs	Resultat av undersøkelsen (2)	
	Base Value RES_2	Compare Value RES_2
	—	—
10426	0	2
31348	0	2
35252	0	2
35271	0	2
40074	0	2
41678	0	2

Eksempel 2.

Nå ønsker vi å sammenligne to variable i samme datasett, I dette tilfelle bostedskommune hentet fra to forskjellige kilder. Det kan være greit å vite hvor sammenfallende disse opplysningene er. Vi lager et nytt lite program:

```
PROC COMPARE BASE=reaksjon.reak87 ;
  VAR bostedp;
  WITH bostedgj;
  TITLE 'Sammenligner bosted fra politiet med bosted fra personregigisteret' ;
RUN;
```

Utlistingen gir oss først en sammenfattet informasjon:

```
Sammenligner bosted fra politiet med bosted fra personregigisteret

COMPARE Procedure
Comparisons of variables in REAKSJON.REAK87
(Method=EXACT)

Data Set Summary

Dataset              Created              Modified  NVar    NObs
WORK.REAK87  29OCT96:11:40:50  29OCT96:11:41:08    62    11899

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 0.
Number of Variables Compared with Some Observations Unequal: 1.
Number of Variables with Missing Value Differences: 1.
Total Number of Values which Compare Unequal: 3380.
```

Deretter får vi den detaljerte listen (de 50 første, her en noen av dem listet):

```
Value Comparison Results for Variables

Obs      Bosted (fra politiet)
         Bosted i (på gjerningstidspunktet)
         Base Value      Compare Value
         BOSTEDP         BOSTEDGJ
-----
3        6905            1928
5        8000
7        0123            0402
9        1501            1401
10       1501            1401
11       0301
14       0301            0102
NOTE: The MAXPRINT=50 printing limit has been reached for the comparison of variable BOSTEDP
with BOSTEDGJ. No more values will be printed for this comparison.
```

Har vi en ident på datasettet kan vi få denne ut på listen istedenfor observasjonsnummeret. Vi bruker da ID-setningen:

```
PROC COMPARE BASE=reak87 ;
  VAR bostedp;
  WITH bostedgj;
  ID fnr NOTSORTED;
  TITLE 'Sammenligner bosted fra politiet med bosted fra personregigisteret' ;
RUN;
```

3.11.3 PROC CONTENTS

Vi bruker PROC CONTENTS til å få oversikt over innholdet i ett eller flere datasett. Informasjonen vi får er data om datasettet (hva det heter, hvor mange observasjoner det har, hvor stor plass det tar, hvilket operativsystem det er laget under, hvem som eier datasett etc.) og filbeskrivelse (oversikt over variablene).

PROC CONTENTS <options>;

Options til PROC CONTENTS-setningen:

CENTILES

Skriver ut en liste over indeksenenes verdier. Indeksens verdi skrives pr. hver 5. prosent av observasjonene i SAS datasettet. Aktuell når SAS datasettet er indeksert

DATA= SAS datasett

Her oppgis navnet på SAS datasettet vi ønsker informasjon om. Ønskes informasjon om alle datasett i et library, velges `_ALL_` isteden.

DETAILS

Brukes til å velge detaljert liste eller ei (NODETAILS)

DIRECTORY

Lager en liste over alle SAS datasett i det valgte library. Det er den samme lista som lages ved bruk av `_ALL_` i DATA-option.

FMTLEN

Skriver ut standard lengde for eventuelle utskrifts- og innlesingsformater som er knyttet til variablene.

MEMTYPE=

Brukes til å avgrense informasjonen til å gjelde datasett av en spesiell type. De forskjellige typene er:

ACCESS	access filer laget med SAS/ACCESS
ALL	Alle datasett-typer
CATALOG	SAS-kataloger og dataformater
DATA	SAS datasett
PROGRAM	Kompilerte SAS program
VIEW	SAS views

MEMTYPE kan forkortes til MT.

NODS

Undertrykker utskrift av innholdet av de enkelte datasett når `_ALL_` er spesifisert i DATA-option.

NOPRINT

Undertrykker utskrift. Gir ingen mening uten bruk av OUT= også.

OUT= SAS datasett

Hvis vi ønsker å legge informasjonen ut på et datasett istedenfor som utskrift, velges navnet med OUT=.

OUT2= SAS datasett

Hvis vi ønsker å legge informasjon om indekser og restriksjoner for dataene i SAS datasettet ut på et datasett istedenfor som utskrift, velges navnet med OUT2=.

POSITION

Lager en variabelliste som er sortert etter variablenes rekkefølge i SAS datasettet i tillegg til den vanlige som sorteres alfabetisk etter variabelnavn.

SHORT

Lager en kortversjon av variabellista, kun variabelnavnene listes ut

VARNUM

Lager en variabelliste som er sortert etter variablenes rekkefølge i SAS datasettet.

Denne informasjonen gir PROC CONTENTS (en observasjon for hver variabel lages når vi legger informasjonen på et nytt datasett med OUT= option):

CHARSET	Karaktersettet som er brukt til å lagre datasettet
COLLATE	Angir om det er brukt en alternativ sorteringsrekkefølge
COMPRESS	YES hvis datasettet er komprimert av SAS, ellers NO.
CRDATE	Dato datasettet er laget
DELOBS	Antall observasjoner som er markert for sletting (f.eks. brukt av PROC FSEDIT)
ENGINE	SAS versjon datasettet er lagret med
ENCRYPT	Angir om datasettet er kryptert
FLAGS	Oppdateringsflagg
FORMAT	Variablenes utskriftsformat
FORMATD	Antall desimaler i utskriftsformatet
FORMATL	Utskriftsformatets bredde
GENMAX	Maksimalt antall generasjoner vi kan ha av SAS datasettet
GENNEXT	Neste generasjonsnummer
GENNUM	Generasjonsnummer
IDXCOUNT	Antall indekser for SAS datasettet
IDXUSAGE	Angir om en variabel er indeksert og i så fall med hvilken type
INFORMAT	Variablenes innlesingsformat
INFORMD	Antall desimaler i innlesingsformatet
INFORML	Innlesingsformatets lengde
JUST	Justering av dataene i variabelen (0 er venstre, 1 er høyre)
LABEL	Variabelens label (utfyllende tekst)
LENGTH	Variabelens lengde
LIBNAME	Områdenavn (libref)
MEMLABEL	Datasettets label (utfyllende tekst)
MEMNAME	Navnet på datasettet variabelen tilhører
MEMTYPE	Datasettets type (DATA eller VIEW)
MODATE	Dato for siste endring av datasettet
NAME	Variabelens navn
NOBS	Antall observasjoner i datasettet
NODUPKEY	YES hvis datasettet er sortert med option NODUPKEY, ellers NO
NODUPREC	YES hvis datasettet er sortert med option NODUPREC, ellers NO
NPOS	Variabelens relative posisjon
PROTECT	R for lesebeskyttet, W for skrivebeskyttet og/eller A for beskyttet mot å slette datasettet, endre variabelnavn og indekser SAS datasettet.
REUSE	YES hvis REUSE option er brukt da datasettet ble laget, ellers NO
SORTED	1 hvis datasettet er sortert, ellers MISSING
SORTEDBY	1 for sorteringsvariable, ellers MISSING.
TYPE	Variabelen er numerisk (1) eller karakter (2)
TYPOMEM	Datasettets type, blank hvis det er et vanlig datasett
VARNUM	Variabelens nummer i datasettet

Denne informasjonen gir PROC CONTENTS når vi legger informasjonen på et nytt datasett med OUT2= option:

ICBUILT	Indeks laget av gyldighetskontroll gir verdi YES, ellers blank. Dette skjer når en gyldighetskontroll krever indeksering.
ICOWN	Indeks eies av gyldighetskontroll gir verdi YES. Dette skjer når en gyldighetskontroll krever indeksering.
IDXNOMISS	Indeks uten MISSING-verdier (verdi blir YES).
IDXUNIQUE	Indeks er unik (verdi blir YES).
INACTIVE	Angir om en gyldighetskontroll er inaktiv. Kan skje for fremmednøkler.
MESSAGE	Melding som gis ved feil.
NAME	Indeksnavn eller navn på gyldighetskontroll.
NUMVALS	Antall unike verdier indeksen har.
NUMVARS	Antall variable indeksen er bygd opp av.
ONDELETE	Ved sletting i forbindelse med fremmednøkler i et annet SAS-datasett. Gyldige verdier er <i>Restrict</i> og <i>Set to NULL</i> .
ONUPDATE	Ved oppdatering i forbindelse med fremmednøkler i et annet SAS-datasett. Gyldige verdier er <i>Restrict</i> og <i>Set to NULL</i> .
RECREATE	SAS-setning som må kjøres for å gjenskape indeks eller gyldighetskontroll.
REF	Navn på SAS datasett som har fremmednøkkelen.
TYPE	Observasjonstype, gyldige verdier er <i>Check</i> , <i>Foreign Key</i> , <i>Index</i> , <i>Not Null</i> , <i>Primary Key</i> og <i>Unique</i> . Se PROC DATASETS, side 128.
UPERC	Prosent av indeksoppdateringer som trenger trigger-oppfrisking.
UPERCMX	Prosent av indeksoppdateringer siden centiler oppdatert.
VARn	Variabelnavn for variable som inngår i en indeks. n starter på 1 og økes for hver variabel.
WHERE	WHERE-betingelse for gyldighetskontroll.

Eksempler:

```
PROC CONTENTS DATA=bvern.f03 POSITION;
  TITLE 'Innhold barnevernsfil Oslo';

PROC CONTENTS DATA=bvern._ALL_;

PROC CONTENTS DATA=bvern.f03 OUT=varf03;
```

3.11.4 PROC COPY

Dette er en prosedyre for å kopiere SAS datasett.

PROC COPY <options> ;

EXCLUDE datasett-liste </MEMTYPE=datasett-type> ;

SELECT datasett-liste </ <MEMTYPE=datasett-type> <ALTER='passord'>>;

Options til PROC COPY-setningen:

ALTER='passord'

Angir et passord som skal brukes for datasettene

CLONE | **NOCLONE**

Avgjør om datasettenes buffer-størrelse (BUFSIZE=), om datasettet skal komprimeres (COMPRESS=) eller ikke og om frigjort plass skal brukes om igjen (REUSE=), skal kopieres (CLONE) eller ei (NOCLONE). CLONE er standardverdi.

CONSTRAINTS=

YES angir at eventuelle integritetsregler også skal kopieres, NO at de ikke skal kopieres. Standardverdi er YES.

DATECOPY

Angir at dato for når SAS datasettet var laget og sist endret ikke skal oppdateres ved kopieringen.

IN=område

SAS-navn på det området datasettene skal kopieres fra

INDEX=

YES angir at eventuelle indekser også skal kopieres, NO at de ikke skal kopieres. Standardverdi er YES.

OUT= område

SAS-navn på det området datasettene skal kopieres til

MEMTYPE=(datasett-type-liste)

Her lister vi opp en eller flere av datasett-typerne vi ønsker å kopiere. Følgende datasett-typer kan brukes: ACCESS (access filer laget med SAS/ACCESS), ALL (alle typer), CATALOG (kataloger/dataformater), DATA (Vanlige SAS datasett), PROGRAM (kompilerte SAS program), VIEW (SAS-views).

MOVE

Angir at datasettene skal flyttes, ikke kopieres.

SAS-setninger til PROC COPY:

EXCLUDE

Her listes datasett som ikke skal kopieres. Option MEMTYPE bruker vi for å ekskludere datasett av en bestemt type.

SELECT

Her lister vi opp de datasettene vi ønsker å kopiere. Ved å bruke MEMTYPE begrenser vi kopieringen til å omfatte datasett med oppgitt MEMTYPE. Vi kan angi et passord for de kopierte datasettene med ALTER=.

Eksempel 1:

```
PROC COPY IN=rev OUT=bvern;  
  SELECT meld95 bvud95 bvmd95 ;  
RUN;
```

Eksempel 2:

```
PROC COPY IN=rev OUT=bvern;  
  SELECT bv: ; /* Alle som starter med bv kopieres */  
RUN;
```


3.11.5 PROC DATASETS

PROC DATASETS kan vi blant annet bruke til å slette og kopiere SAS datasett. Vi kan også endre navn og attributter til SAS datasett. Videre kan vi reparere ødelagte SAS datasett og vi kan lage og slette indekser på SAS datasett. Dessuten kan vi bruke PROC DATASETS til å utføre det samme som PROC APPEND, PROC CONTENTS og PROC COPY kan gjøre.

Options MEMTYPE og ALTER kan gjelde alle datasett som nevnes. Da bruker vi / bakerst i setningene. Ønsker vi at de skal gjelde for et datasett, settes de i parentes etter dette.

```
PROC DATASETS <options> ;
  AGE gjeldende-navn navneliste </ <MEMTYPE=datasett-type> <ALTER='passord'>> ;
  AUDIT SAS datasett <passord> ;
  INITIATE ;
    <LOG <BEFORE_IMAGE=YES|NO> <DATA_IMAGE=YES|NO> <ERROR_IMAGE=YES|NO>;>
    <USER_VAR=variabel-1 <$> <lengde> <LABEL='tekst'>
      <... variabel-n <$> <lengde> <LABEL='tekst'>>;>
    <SUSPEND|RESUME|TERMINATE>;>
  APPEND BASE=SAS datasett <DATA=SAS datasett <(datasett-options)>><APPENDVER=V6> <FORCE> ;
  CHANGE gammelt-navn-1=nytt-navn-1 ... <gammelt-navn-n=nytt-navn-n>
    </ <MEMTYPE=datasett-type> <ALTER='passord'>> ;
  CONTENTS <DATA=<libref.>datasett | _ALL_ > <DIRECTORY>
    <MEMTYPE=(datasett-type-liste)> <NODS> <NOPRINT>
    <OUT=SAS datasett<(datasett-options)>> <POSITION> <SHORT>
    <DETAILS | NODETAILS> ;
  COPY OUT=libref <IN=libref> <MEMTYPE=(datasett-type-liste)>
    <MOVE> <CLONE> <INDEX=YES | NO> <ALTER='passord'>;
  EXCLUDE datasettliste </MEMTYPE=datasett-type> ;
  SELECT datasettliste </ <MEMTYPE=datasett-type> <ALTER='passord'>> ;
  DELETE datasettliste </ <MEMTYPE=datasett-type> <ALTER='passord'>> ;
  EXCHANGE navn-a-1=navn-b-2 ... <navn-a-n=navn-b-n> </ <MEMTYPE=datasett-type>
    <ALTER='passord'>> ;
  MODIFY datasettnavn <(<LABEL='datasett-label'> <TYPE=spesiell-datasett-type)>> ;
  FORMAT variabel-liste-1 utskriftsformat-1 ... <variabel-liste-n utskriftsformat-n> ;
  IC CREATE <gyldighetsregel-navn> = gyldighetsregel <MESSAGE='Melding'> ;
  IC DELETE gyldighetsregel-navn-1 ... <gyldighetsregel-navn-n> | _ALL_ ;
  IC REACTIVATE fremmednøkkel-navn REFERENCES libref ;
  INDEX CREATE indeks =(variabel-liste) </ <NOMISS> <UNIQUE>
    <UPDATECENTILES=ALWAYS | NEVER | n>> ;
  INDEX DELETE indeks-liste | _ALL_ ;
  INFORMAT variabel-liste-1 innlesingsformat-1 ... <variabel-liste-n innlesingsformat-n> ;
  LABEL variabel-1='label-1' ... <variabel-n='label-n'> ;
  RENAME gammelt-variabelnavn-1=nytt-navn-1 ... <gammelt-variabelnavn-n=nytt-navn-n>;
  REPAIR datasettnavn </ <MEMTYPE=(datasett-type)> <ALTER='passord'>> ;
  SAVE datasettliste </MEMTYPE=datasett-type> ;
```

Options til PROC DATASETS-setningen:

ALTER='passord'

Angir passord for tilgang til å skrive, slette og endre datasettet

DETAILS

Brukes for å velge mer detaljert liste ved utskrift. Standard er NODETAILS.

FORCE

Angir standard for APPEND-setningen. Godkjenner sammenhegting av to SAS datasett selv om de inneholder forskjellige variable. Se PROC APPEND, side 111.

KILL

Sletter alle datasett under området. Denne må vi bruke med stor forsiktighet.

LIBRARY=område

SAS-navn på det området hvis SAS datasett skal behandles

MEMTYPE=(datasett-type-liste)

Angir hvilke datasett-typer som skal behandles:

ACCESS	aksessfiler laget vha. SAS/ACCESS
ALL	alle typer
CATALOG	SAS kataloger og dataformater
DATA	Vanlige SAS datasett
PROGRAM	Lagrede kompilerte SAS-program
VIEW	SAS-views

NOLIST

Undertrykker utlisting av datasett-navn i SASlog

NOWARN

Sørger for at programmet utfører prosedyren så godt som mulig selv om vi prøver å behandle SAS datasett som ikke eksisterer.

PW='passord'

Angir password for tilgang til å lese og oppdatere datasettet

READ='passord'

Angir password for tilgang til å lese datasettet

SAS-setninger til PROC DATASETS:

AGE

Forflytter datasett i en rekke. Det siste datasettet i navnelista slettes og det nest siste overtar dette navnet. Slik rykker datasettene bakover i køen. Det gjeldende datasettet får navnet til det først i navnelista.

Eksempel:

```
PROC DATASETS LIBRARY=indeks;  
  AGE siste mnd1-mnd12;  
RUN;
```

Datasettet *mnd12* vil slettes, *mnd11* blir endret til *mnd12*, *mnd10* til *mnd11* osv. Til slutt endres det siste til *mnd1*.

APPEND

Brukes for å tilføye observasjoner fra et datasett til et annet. Tilsvarende PROC APPEND, se side 111.

AUDIT

Starter og kontrollerer logging til en overvåkningsfil som viser hva som er gjort med SAS datasettet. Denne overvåkningsfilen kan fort bli meget stor. Overvåkningsfilen får filhale *sas7baud*. Filen kan listes vha PROC PRINT, PROC FREQ eller PROC TABULATE. Vi må da angi SAS datasett option TYPE=audit.

Undersetninger til AUDIT-setningen:

LOG

Gir mulighet til å bestemme hvilke endringer og endringsforsøk som skal logges. Disse options kan vi bruke:

BEFORE_IMAGE=	YES skriver ut meldinger før en observasjon endres, NO gjør ikke det.
ERROR_IMAGE =	YES skriver ut feilmeldinger når endring ikke er utført, NO gjør ikke det.
DATA_IMAGE=	YES skriver ut melding når en observasjon er oppdatert, NO dropper det.

USER_VAR=

Gir muligheter for å definere egne variable som kommer med i overvåkningsfilen.

SUSPEND

Angir at loggingen skal stoppe, men sletter ikke overvåkningsfilen.

RESUME

Angir at loggingen skal starte igjen hvis den var stoppet.

TERMINATE

Angir at loggingen skal stoppe og at overvåkningsfilen skal slettes.

CHANGE

Brukes til å endre navn på datasett.

CONTENTS

Brukes til å få oversikt over innholdet i ett eller flere SAS datasett. Tilsvarende PROC CONTENTS, se side 119.

COPY

Brukes til å kopiere SAS datasett. Tilsvarende PROC COPY, se side 123.

DELETE

Angir navn på SAS datasett som skal slettes.

EXCHANGE

Brukes for å bytte om navnene på to datasett.

EXCLUDE

Brukes, sammen med COPY-setningen, til å velge bort de datasett vi ikke ønsker å behandle.

FORMAT

Brukes, etter en MODIFY-setning, til å knytte SAS utskriftsformater til variable eller til å fjerne utskriftsformattilknytninger. Ved å liste variablene uten å oppgi utskriftsformat, fjernes tilknytningen.

IC CREATE

Brukes, etter en MODIFY-setning, til å definere gyldighetsregler (integrity constraints) for SAS datasettet. Vi kan gi dem egne navn eller la SAS gi dem navn.

Gyldighetsregler:

CHECK (WHERE-uttrykk)

Gjør at bare data som tilfredstiller WHERE-uttrykket blir lagt til SAS datasettet.

FOREIGN KEY (variable) REFERENCES datasettnavn

<ON DELETE aksjon> <ON UPDATE aksjon>

Definerer en fremmednøkkel. Denne er en variabel som må finnes med samme navn i SAS datasettet nevnt i REFERENCES-option. Ved sletting eller oppdatering av SAS datasettet, kan vi enten velge RESTRICT, CASCADE eller SET NULL. Med RESTRICT slettes/oppdateres primærnøkkelens observasjon hvis ingen fremmednøkler med samme verdi finnes. Med CASCADE oppdateres fremmednøkler i samsvar med primærnøkklene. SET NULL gjør at primærnøkkelens observasjon slettes/endres og at korresponderende fremmednøklers verdier settes til MISSING.

NOT NULL (variabel)

Gjør at variabelen ikke kan få MISSING-verdi.

PRIMARY KEY (variabel)

Definerer variabelen som en primærnøkkel.

UNIQUE (variable)

Gjør at variablene må ha unike verdier.

IC DELETE

Brukes, etter en MODIFY-setning, til å slette gyldighetsregler fra et SAS datasett.

IC REACTIVATE

Brukes, etter en MODIFY-setning, til å reaktivere en gyldighetsregel som er satt inaktiv. De kan bli inaktive ved prosesseringsfeil. Vi må angi libref til det SAS datasettet som inneholder primærnøkkel.

INDEX CREATE

Brukes, etter en MODIFY-setning, til å lage indeks på SAS datasett. Indeks-filer får filhale *sas7bndx*.

INDEX DELETE

Brukes, etter en MODIFY-setning, til å slette indekser. Angi indekser som skal slettes eller `_ALL_` som sletter alle indekser.

INFORMAT

Brukes, etter en MODIFY-setning, til å knytte SAS innlesingsformater til variable eller til å fjerne innlesingsformattilknytninger. Ved å liste variablene uten å oppgi innlesingsformat, fjernes tilknytningen.

LABEL

Brukes, etter en MODIFY-setning, til å endre labler på variable.

MODIFY

Endrer attributter for et SAS datasett. Vi bruker FORMAT, IC CREATE, IC DELETE, IC REACTIVATE, INDEX CREATE, INDEX DELETE, INFORMAT, LABEL og RENAME til dette.

Options til MODIFY-setningen:

ALTER='passord'

Angir password for tilgang til å skrive, slette og endre datasettet

PW='passord'

Angir password for tilgang til å lese og oppdatere datasettet

READ='passord'

Angir password for tilgang til å lese datasettet

WRITE='passord'

Angir password for tilgang til å lese og oppdatere datasettet

LABEL='datasett-label'

Angir en beskrivende tekst til SAS datasettets innhold

SORTEDBY=by-variable | _NULL_

TYPE=spesiell-datasett-type

Lager eller endrer en spesiell datasett-type. Dette er datasett-typer som lages av spesielle SAS prosedyrer. Typene er CORR, COV, EST, FACTOR og SSCP.

RENAME

Brukes, etter en MODIFY-setning, til å endre variabelnavn.

REPAIR

Brukes for å reparere ødelagte SAS datasett. Noen situasjoner der SAS datasett kan bli ødelagt:

- Systemet går ned under oppdatering
- Disken går i stykker under databehandlingen
- Disken som brukes er full før hele datasettet er lagret
- En I/O-feil oppstår under skriving

Slike reparasjoner vil ikke alltid lykkes, heldigvis er det sjelden slike feil oppstår.

SAVE

Beholder de datasettene som nevnes i SAVE-setningen, resten slettes. Bruk heller DELETE-setningen, den gir bedre kontroll.

SELECT

Brukes, sammen med COPY-setningen, til å selektene de datasett vi skal behandle.

Eksempel 1: PROC DATASETS kan brukes for å slette filer. Her sletter vi alle filer som begynner med fylke samt filen distrikt.

```
PROC DATASETS LIB=soshjelp;  
  DELETE fylke: distrikt;  
RUN;
```

Eksempel 2: Vi bruker PROC DATASETS til å gi forklarende tekst ("labler") til variable som mangler dette.

```
PROC DATASETS LIB=soshjelp;  
  MODIFY mott1994;  
    LABEL fnr      = 'Fødselsnummer'  
          kommune = 'Kommune'  
          bidrag  = 'Bidrag'  
          laan    = 'Lån'  
          ;  
RUN;
```

3.11.6 PROC DOWNLOAD

PROC DOWNLOAD bruker vi for å overføre data fra en tjenermaskin til en klientmaskin. Vanligvis medfører det å overføre data fra unix til Windows NT, men det kan også være fra en unix-maskin til en annen. PROC DOWNLOAD utføres på tjenermaskinen, det betyr at programmet må eksekveres med RSUBMIT-kommandoen eller at PROC DOWNLOAD må stå inne i en RSUBMIT-blokk for eksekvering med SUBMIT-kommandoen. For praktisk bruk, se side 6.

```
PROC DOWNLOAD <options> ;
WHERE where-uttrykk ;
EXCLUDE datasettliste </MEMTYPE=datasett-type | ENTRYTYPE=katalog-type> ;
SELECT datasettliste </MEMTYPE=datasett-type | ENTRYTYPE=katalog-type>> ;
```

Options til PROC DOWNLOAD-setningen:

AFTER= dato

Laster ned alle datasett som er endret etter angitt dato. Datoen kan angis som et gyldig SAS datoformat (f.eks. '09MAR1999'D) eller som en SAS datoverdi. Virker bare på SAS datasett, SAS kataloger og SAS formater som er laget eller endret i løpet av eller etter angitt dato.

BINARY

Angir at filen skal overføres binært. Skal ikke brukes ved overføring av SAS datasett.

DATA= SAS datasett

Her angir vi navn på SAS datasett som skal lastes ned

ENTRYTYPE= katalog-type

Angir hvilken SAS katalog-type som skal lastes ned. Brukes sammen med INCAT-option. Kan overstyres av ENTRYTYPE-option i en SELECT- eller EXCLUDE-setning.

INCAT=libref.katalog

Navnet på SAS katalog som skal lastes ned (SAS kataloger må ikke forveksles med SAS formater). Brukes sammen med OUTCAT-option.

INDEX=

Y angir at indeksfiler for indekserte SAS datasett også skal lastes ned, N at de ikke skal.

INFILE= 'filnavn' | fil-referanse

Angir navnet på den sekvensielle filen som skal lastes ned. Enten skrives filnavnet direkte i INFILE-option, ellers kan vi angi det først i en FILENAME-setning for deretter å bruke fil-referansen i INFILE-option. Brukes sammen med OUTFILE-option.

INLIB= libref

SAS-navn på det området datasettene som skal lastes ned ligger under. Kan forkortes til IN=. Brukes sammen med OUTLIB-option.

OUT= SAS datasett

Angir navnet på det nedlastede SAS datasettet.

OUTCAT=libref.katalog

Navnet på SAS katalog som det skal lastes ned til (SAS kataloger må ikke forveksles med SAS formater). Brukes sammen med INCAT-option.

OUTFILE= 'filnavn' | fil-referanse

Angir navnet på den nedlastede sekvensielle filen. Enten skrives filnavnet direkte i OUTFILE-option, ellers kan vi angi det først i en FILENAME-setning for deretter å bruke fil-referansen i OUTFILE-option. Brukes sammen med INFILE-option.

OUTLIB=libref

SAS-navn på det området datasettene skal lastes ned til. Kan forkortes til OUT=. Brukes sammen med INLIB-option.

MEMTYPE=

Brukes til å avgrense nedlasting til å gjelde SAS datasett av en spesiell type. Gyldige typer er ALL, CATALOG og DATA. Kan overstyres av MEMTYPE-option i en SELECT- eller EXCLUDE-setning.

STATUS=

Y angir at et statusvindu vises under overføringen, N at statusvinduet ikke vises.

SAS-setninger til PROC DOWNLOAD:

WHERE

Selekterer hvilke observasjoner som selekteres fra SAS datasettet som lastes ned. Virker bare når DATA-option i PROC DOWNLOAD-setningen brukes. Beskrivelse av WHERE-setningen finnes på side 91.

EXCLUDE

Her listes datasett som ikke skal lastes ned. Option MEMTYPE bruker vi for å ekskludere SAS datasett av en bestemt type, mens ENTRYTYPE-option kan brukes for å ekskludere SAS kataloger av en bestemt type. Disse options overstyrer tilsvarende options brukt i PROC DOWNLOAD-setningen.

SELECT

Her listes de datasett som skal lastes ned. Option MEMTYPE bruker vi for å selektere SAS datasett av en bestemt type, mens ENTRYTYPE-option kan brukes for å selektere SAS kataloger av en bestemt type. Disse options overstyrer tilsvarende options brukt i PROC DOWNLOAD-setningen.

Eksempel 1:

Overfører alle SAS datasett fra området *pasient* på tjenermaskin til området *work* på lokal maskin:

```
PROC DOWNLOAD IN=pasient OUT=work MEMTYPE=data;  
RUN;
```

Eksempel 2:

Overfører SAS datasettet *pasient.sykh1998* på tjenermaskin til *work.sykh1998* på lokal maskin. Selekterer de som har utdato i 1998 eller senere:

```
PROC DOWNLOAD DATA=pasient.sykh1998 OUT=sykh1998 MEMTYPE=data;  
WHERE utdato >= '01JAN1998'D ;  
RUN;
```

Eksempel 3:

Henter dagens dato og legger den i en makrovariabel ved hjelp av CALL SYMPUT og overfører SAS datasett som er laget i dag fra området *pasient* på tjenermaskinen til *work* på lokal maskin:

```
DATA _NULL_ ;  
CALL SYMPUT('idag',today());  
RUN;  
  
PROC DOWNLOAD AFTER=&idag. IN=pasient OUT=work MEMTYPE=data;  
RUN;
```

Eksempel 4:

Overfører en html-fil fra tjenermaskin til lokal maskin:

```
PROC DOWNLOAD INFILE='indu3_45.html' OUTFILE='h:\krl\industri\nos\indu3_45.html';  
RUN;
```

3.11.7 PROC FORMAT

Proc format bruker vi til å definere tekster eller koder (formaterte verdier) til variabelverdier, enten ved at hver kode får en egen tekst eller ved at en gruppe koder får en egen tekst. Denne teksten kan være opptil 200 tegn lang. Videre kan vi bruke PROC FORMAT til å bestemme et mønster (PICTURE) for hvordan en eller flere numeriske variable skal skrives ut, f.eks. kan vi lage et mønster som vi kan bruke når vi skal skrive ut tall med tusenskilte.

```
PROC FORMAT <options>
  EXCLUDE format-liste ;
  INVALUE navn <(inn-format-options)>
    intervall = 'verdi 1'
    ...
    <intervall='verdi n'> ;
  PICTURE navn <(format-options)>
    intervall = 'mønster 1' <mønster-options>
    ...
    <intervall='mønster n' <mønster-options>> ;
  SELECT format-liste ;
  VALUE navn <(format-options)>
    intervall = 'tekst 1'
    ...
    <intervall='tekst n'> ;
```

Options til PROC FORMAT-setningen:

CNTLIN= SAS datasett
Navn på SAS datasett som det skal lages et format fra. Før vi bruker PROC FORMAT med CNTLIN= option, må vi ha laget et SAS datasett som inneholder variablene START, LABEL og FMTNAME. Variablene som kan benyttes for å lage formater fra et SAS datasett og deres innhold (de som er understreket må være med):

<u>FMTNAME</u>	Navn på formatet
<u>START</u>	Kode (startverdi)
<u>END</u>	Kode (sluttverdi)
<u>LABEL</u>	Tekst
MIN	Inneholder tekstens minimumslengde
MAX	Inneholder tekstens maksimumslengde
DEFAULT	Inneholder tekstens standardlengde
LENGTH	Inneholder lengden til den lengste brukte teksten
FUZZ	Inneholder fuzz-verdien, se side 137.
PREFIX	Inneholder eventuelt prefiks for mønsterformat.
MULT	Inneholder multiplikasjonsfaktor for mønsterformat.
FILL	Inneholder tegn som skal skrives ut hvis tallet ikke fyller hele den avsatte plassen.
NOEDIT	1 når mønsteret ikke skal brukes som et mønster, men som en tekst, ellers 0.
TYPE	Formattype: C karakter utskriftsformat N numerisk utskriftsformat P mønsterformat (picture) I innlesingsformat J venstrejustert innlesingsformat
SEXCL	Verdi Y når startkoden ikke skal være med i intervallet, verdi N når startkoden skal være med
EEXCL	Verdi Y når sluttkoden ikke skal være med i intervallet, verdi N når sluttkoden skal være med

HLO Verdi H sier at tekst nevnt i LABEL-variabelen brukes for høyeste verdi (HIGH)
Verdi L sier at tekst nevnt i LABEL-variabelen brukes for laveste verdi (LOW)
Verdi O sier at tekst nevnt i LABEL-variabelen brukes for restgruppen (OTHER).
Verdi S sier at NOTSORTED format-option er brukt. Når NOTSORTED format-option er brukt, vil HLO-variabelen alltid starte med S.
Verdi R angir for PICTURE-formater at verdiene skal avrundes før mønsteret brukes.
Verdi J angir for INVALUE-karakterformater at den innleste teksten venstrejusteres før den sammenlignes med kodene i innlesingsformatet.

Eksempel:

```
DATA kommkat ;
  INFILE '$FELLES/ssbkat/kommune/g1994' MISSOVER PAD;
  INPUT  start      $ 1-4
         label      $ 6-25
        ;
  fmtname = '$komm' ;

PROC FORMAT CNTLIN=kommkat;

PROC FREQ DATA=persreg ;
  TABLES komm/MISSING;
  FORMAT komm $komm. ;
  TITLE 'Antall innen hver kommune';
```

Utlistingen:

Antall innen hver kommune				
KOMM	Frequency	Percent	Cumulative Frequency	Cumulative Percent
	1	3.8	1	3.8
Halden	13	50.0	14	53.8
0102	4	15.4	18	69.2
0103	2	7.7	20	76.9
Vestby	3	11.5	23	88.5
Ski	2	7.7	25	96.2
Ås	1	3.8	26	100.0

Her lages utskriftsformatet \$komm fra SAS datasettet kommkat. Vi har ikke definert noen restgruppe, derfor brukes de ugyldige kommunenummer sin verdi som tekst.

Alternativt kan vi definere en restgruppe for de ugyldige. Da må en observasjon i SAS datasettet ha HLO-variabelen satt til O, som vist under.

```
DATA kommkat ;
  INFILE '$FELLES/ssbkat/kommune/g1994' MISSOVER PAD END=slutt;
  INPUT  start      $ 1-4
         label      $ 6-25
        ;
  fmtname = '$komm' ;
  OUTPUT;
  IF slutt THEN
  DO;
    hlo = 'O' ;
    start = '' ;
    label = 'Ugyldige nr' ;
    fmtname= '$komm' ;
    OUTPUT;
  END;

PROC FORMAT CNTLIN=kommkat;

PROC FREQ DATA=persreg ;
  TABLES komm/MISSING;
  FORMAT komm $komm. ;
  TITLE 'Antall innen hver kommune. Ugyldige kommuner samles';
```


Resultatet:

Antall innen hver kommune. Ugyldige kommuner samles

KOMM	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Ugyldige nr	7	26.9	7	26.9
Halden	13	50.0	20	76.9
Vestby	3	11.5	23	88.5
Ski	2	7.7	25	96.2
Ås	1	3.8	26	100.0

Ønsker vi å skille mellom de som mangler kommunenummer og de som har ugyldige kan vi endre programmet slik:

```
DATA kommkat ;
  INFILE '$FELLES/ssbkat/kommune/g1994' MISSOVER PAD END=slutt;
  INPUT start $ 1-4
         label $ 6-25
        ;
  fmtname = '$komm' ;
  OUTPUT;
  IF slutt THEN
  DO;
    start = '';
    label = 'Mangler nr';
    fmtname= '$komm';
    OUTPUT;
    hlo = 'O' ;
    start = '';
    label = 'Ugyldige nr';
    fmtname= '$komm';
    OUTPUT;
  END;
```

Nytt resultat:

Antall innen hver kommune. Ugyldige og manglende kommuner samles

KOMM	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Mangler nr	1	3.8	1	3.8
Halden	13	50.0	14	53.8
Ugyldige nr	6	23.1	20	76.9
Vestby	3	11.5	23	88.5
Ski	2	7.7	25	96.2
Ås	1	3.8	26	100.0

Ønskr vi å beholde rekkefølgen i SAS datasettet, kan vi legge inn S som første tegn i HLO-variabelen. Da vil vi kunne beholde denne rekkefølgen nå vi bruker PROC TABULATE eller PROC MEANS:

```
DATA kommkat ;
  INFILE '$FELLES/ssbkat/kommune/g1994' MISSOVER PAD END=slutt;
  INPUT start $ 1-4
         label $ 6-25
        ;
  fmtname = '$komm' ;
  hlo = 'S ' ;
  OUTPUT;
  IF slutt THEN
  DO;
    start = '';
    label = 'Mangler nr';
    fmtname= '$komm';
    OUTPUT;
    hlo = 'SO' ;
    start = '';
    label = 'Ugyldige nr';
    fmtname= '$komm';
    OUTPUT;
  END;

PROC FORMAT CNTLIN=kommkat;
```

```
PROC TABULATE DATA=persreg FC='-----' ORDER=DATA NOSEPS MISSING;
  CLASS komm/PRELOADFMT;
  TABLE ALL='I alt' komm=' '
    /
    N*F=7. PCTN*F=7.1
  /RTS=25 MISSTEXT='- ' ;
  FORMAT komm $komm.;
  TITLE 'Antall innen hver kommune. Beholder formatrekkefølgen' ;
RUN;
```

Legg merke til plasseringen av linjene med Mangler nr og Ugyldige nr:

Antall innen hver kommune. Beholder formatrekkefølgen

	N	PctN
I alt	26	100.0
Halden	13	50.0
Vestby	3	11.5
Ski	2	7.7
Ås	1	3.8
Mangler nr	1	3.8
Ugyldige nr	6	23.1

Se også eksempel på side 23.

CNTLOUT= SAS datasett

Brukes for å lage et SAS datasett fra et formatbibliotek. SAS datasettet vil inneholde de mest brukte variablene FMTNAME, START, END, LABEL, HLO og TYPE, samt MIN, MAX, DEFAULT, LENGTH, FUZZ, PREFIX, MULT, FILL, NOEDIT, SEXCL og EEXCL.

```
PROC FORMAT CNTLOUT=kommut;
  SELECT $komm ;
```

FMTLIB

Skriver ut innholdet i formatbiblioteket.

LIBRARY=library

For å lagre formatene permanent må denne option være med. Referansen til området (libref) med navn *library* må være definert tidligere med en LIBNAME-setning, se side 100.

MAXLABELEN=lengde

Setter maks. lengde på LABEL-variabelen når formatet skrives ut med CNTLOUT=.

MAXSELEN=lengde

Setter maks. lengde på START- og END-variablene når formatet skrives ut med CNTLOUT=.

NOREPLACE

Gjør at et format som skal legges inn ikke erstattes hvis det finnes fra før.

PAGE

Brukes sammen med FMTLIB for å få et format pr. side ved utskrift av formatbiblioteket.

SAS-setninger til PROC FORMAT:

EXCLUDE

Brukes til å velge bort de formater vi ikke ønsker å skrive ut på SAS datasettet nevnt med CNTLOUT=. Vi kan bare bruke én EXCLUDE-setning i et PROC FORMAT-steg. Se eksempler i SELECT-setningen under.

INVALUE

Definerer et format for bruk ved innlesing av data (innlesingsformat) fra sekvensiell fil. Innlesingsformatets verdi (tekst), vil bli lagret som variabelens verdi på det nye SAS datasettet.

```
PROC FORMAT ;
  INVALUE $kjonn
    '1' = '1'
    '2' = '2'
    'K' = '2'
    'M' = '1'
    ' ' = ' '
    other = '9'
  ;
```

```
DATA persreg;
INPUT @02 komm $CHAR4.
      @07 gkrets $CHAR3.
      @11 kjonn $kjonn.
      @13 aldgrp $CHAR4.
      @18 inntekt 7.
      @26 skatt 7.
;
CARDS;
0101 001 K ald1 230327 85023
0101 001 1 ald2 122341 34213
0101 001 K ald1 355345 67326
0101 001 2 ald2 242351 78219
0101 001 K ald3 116341 13232
0101 001 3 ald4 326340 134216
0102 002 M ald3 242463 100230
0211 004 1 ald5 245470 121002
0214 008 ald1 158118 34142
      . 008 M ald1 248118 87631
;

PROC FREQ DATA=persreg ;
TABLES kjonn/MISSING;
TITLE 'Antall innen hvert kjønn.';
RUN;
```

Utlistingen:

Antall innen hvert kjønn.

KJONN	Frequency	Percent	Cumulative Frequency	Cumulative Percent
	1	10.0	1	10.0
1	4	40.0	5	50.0
2	4	40.0	9	90.0
9	1	10.0	10	100.0

Innlesingsformat-options:

- DEFAULT=**
Setter standard bredde for innlesingsformatet.
- FUZZ=**
Kan brukes til å sørge for at verdier som avviker lite fra intervallet som angis allikevel kommer med i intervallet. FUZZ-verdien sier hvor stort dette avviket skal være. Standardverdi er 1E-12.
- JUST**
Venstrejusterer den innleste teksten før den sammenlignes med kodene i innlesingsformatet.
- MAX=**
Største bredde vi ønsker innlesingsformatet skal ha.
- MIN=**
Minste bredde vi ønsker innlesingsformatet skal ha.
- NOTSORTED**
Gjør at innlesingsformatene lagres i den rekkefølgen de har i INVALUE-setningen. Bevirker at HLO får S som første tegn på alle verdiene.
- UPCASE**
Gjør den innleste teksten om til store bokstaver før den sammenlignes med kodene i innlesingsformatet. æ, ø og å går også bra.

PICTURE

Definerer et mønster for utskrift av en eller flere **numeriske** variable.

Format-options:

- DEFAULT=**
Setter standard bredde for formatet.
- FUZZ=**
Kan brukes til å sørge for at verdier som avviker lite fra intervallet som angis allikevel kommer med i intervallet. FUZZ-verdien sier hvor stort dette avviket skal være. Standardverdi er 1E-12.
- MAX=**
Største bredde vi ønsker formatet skal ha.

MIN=

Minste bredde vi ønsker formatet skal ha.

MULTILABEL

Angir at formatet kan ha overlappende verdier. HLO-variabelen får en M som første eller andre tegn på alle verdiene.

NOTSORTED

Angir at formatet lagres i den rekkefølge det er skrevet. Bevirker at HLO-variabelen får S som første tegn på alle verdiene.

ROUND

Runder av verdiene før de sammenlignes med kodene (intervallet) i formatet.

Mønster-options:

FILL=

Angir et tegn som skal skrives ut hvis tallet ikke fyller hele den avsatte plassen.

MULT=

Faktor som tallet ganges med før utskrift. $MULT=0.001$ vil gjøre at tallet deles på 1000 ($=*0.001$) før utskrift.

NOEDIT

Sier at mønsteret ikke skal brukes som et mønster, men som en tekst.

PREFIX=

Brukes til å angi ett eller to prefiks til mønster. Prefikset vil bruke av den plassen som er satt av.

Eksempler:

Vi lager mønstre for utskrift av tall med tusenskilte, med tusenskilte og divisjon med 1000 samt for tall med tusenskilte med en desimal med komma som desimalskilletegn.

```
DATA kommreg;
INPUT      komm      $
           innt
           saldo
           skatt
           ;

CARDS;
0101      912131499  -99999999.4      99999999.94
0211      912131001           0      99999999.939999
0238      912131500          -0      -99999999.949999
0301 9999999999500  100000000      -99999999.95
1151           0           5           0.95001
1201      999999600  -99999999.5      24243.4566
1252           499          -313           0.94999
1601 999999999499  99999999.49      -9999999.951
1532           500          -1           0
2030           501           2           .
;

PROC FORMAT ;
PICTURE ssb9_0v (ROUND)
-99999999.5 <-<           0 = '000 000 009' (PREFIX='-')
                        0 = '- '
                        0 <-<999999999.4 = '000 000 009'
                        . = '- '
                        OTHER = '*****'
;

PICTURE ssb9_0t (ROUND)
-99999999500 <-<           0 = '000 000 009' (PREFIX='- ' MULT=0.001)
                        0 = '- '
                        0 <-< 999999999500 = '000 000 009' (MULT=0.001)
                        OTHER = '*****'
                        . = '- '
;

PICTURE ssb9_1v (ROUND)
-99999999.95 <-<           0 = '00 000 009,9' (PREFIX='- ' MULT=10)
                        0 = '- '
                        0 <-<99999999.94 = '00 000 009,9' (MULT=10)
                        . = '- '
                        OTHER = '*****'
;

PROC PRINT DATA=kommreg ;
FORMAT innt ssb9_0t. saldo ssb9_0v. skatt ssb9_1v.;
RUN;
```

Resultatet av kjøringen:

OBS	KOMM	INNT	SALDO	SKATT
1	0101	912 131	-99 999 999	*****
2	0211	912 131	-	99 999 999,9
3	0238	912 132	-	-9 999 999,9
4	0301	*****	100 000 000	*****
5	1151	-	5	1,0
6	1201	1 000 000	*****	24 243,5
7	1252	0	-313	0,9
8	1601	999 999 999	*****	*****
9	1532	1	-1	-
10	2030	1	2	.

Vi definerer mønster for postnummer og telefonnummer:

```
DATA adr;
INPUT      navn      $ 1-24
           adresse   $ 26-39
           postnr    41-44
           poststed  $ 46-56
           tlf       58-65
           ;

CARDS;
KONGSVINGER SOSIALKONTOR SKOLEGT. 1      2200 KONGSVINGER 62820200
HAMAR SOSIALKONTOR      POSTBOKS 300    2301 HAMAR      62510222
RINGSAKER KOMMUNE      SOSIALKONTORET 2380 BRUMUNDDAL 62340682
LØTEN KOMMUNE          SOSIALKONTORET 2340 LØTEN      62593000
STANGE KOMMUNE         SOSIALKONTORET 2310 STANGE     62571100
NORD-ODAL KOMMUNE      SOSIALKONTORET 2120 SAGSTUA    62971488
;

PROC FORMAT ;
  PICTURE postnr
    1-9999 = ' 9999' (PREFIX='N-')
  ;
  PICTURE tlf
    0-HIGH = '99-99 99 99'
  ;
PROC PRINT DATA=adr ;
VAR navn postnr tlf;
FORMAT postnr postnr. tlf tlf.;
TITLE 'Sosialkontors postnr og telefonnr';
```

Utskrift med mønster:

Sosialkontors postnr og telefonnr				
OBS	NAVN	POSTNR	TLF	
1	KONGSVINGER SOSIALKONTOR	N-2200	62-82	02 00
2	HAMAR SOSIALKONTOR	N-2301	62-51	02 22
3	RINGSAKER KOMMUNE	N-2380	62-34	06 82
4	LØTEN KOMMUNE	N-2340	62-59	30 00
5	STANGE KOMMUNE	N-2310	62-57	11 00
6	NORD-ODAL KOMMUNE	N-2120	62-97	14 88

SELECT

Brukes til å velge de formater som skal skrives ut på SAS datasettet nevnt med CNTLOUT=. Vi kan bare bruke én SELECT-setning i et PROC FORMAT-steg.

```
SELECT $kom92nr /* selekterer formatet $kom92nr */ ;
SELECT $kom:    /* selekterer alle formater som starter med $kom */ ;
SELECT $fyl-$kon /* selekterer alle formater i intervallet $fyl-$kon */ ;
```

VALUE

Definerer et utskriftsformat for teksting av en variabls verdier. Navnet på utskriftsformatet skal være max 8 tegn, utskriftsformater beregnet på karaktvariable skal begynne med \$. Når utskriftsformatene brukes skal de ha med seg et punktum til slutt.

```
PROC FORMAT;
VALUE $kjonn
  'K' = 'Kvinner'
  'M' = 'Menn'
  OTHER = 'Uoppgitt kjønn'
;
```

```

VALUE inntekt
  0- < 100000 = 'Under 100000'
  100000- < 200000 = '100000 - 200000'
  200000- < 300000 = '200000 - 300000'
  300000- HIGH = '300000 og over'
  OTHER = 'Uoppgitt inntekt'
;
PROC FREQ DATA=persreg;
TABLES kjonn inntekt/MISSING;
FORMAT kjonn $kjonn.
       inntekt inntekt.
;
TITLE 'Frekvenser for kjønn og inntekt';

```

Eksempelet viser hvordan utskriftsformatene \$kjonn og inntekt lages med PROC FORMAT, for siden å brukes i PROC FREQ. \$kjonn er et karakter utskriftsformat som brukes til å lage tekster til kodene fra SAS datasettet. Legg merke til at vi med OTHER lager en restgruppe som vil omfatte alle andre verdier enn K og M. Utskriftsformatet inntekt brukes til å gruppere inntekter. Verdien HIGH angir høyeste mulige verdi, mens LOW angir laveste verdi.

Vi må passe på at intervallene våre lukkes, dvs. at det ikke blir igjen noen verdier som ikke får noen tekst. Disse vil havne i restgruppen OTHER. Hadde vi definert utskriftsformatet med intervallene 0-99999 og 100000-199999 istedenfor slik det er gjort i eksempelet over, ville f.eks. ikke verdien 99999,50 blitt med i noen av intervallene og derfor havnet i restgruppen.

De verdiene som defineres for en tekst kan være enkeltverdier, intervaller eller kombinasjoner av disse.

```

PROC FORMAT;
VALUE $kjonn
  'K','2' = 'Kvinner'
  'M','1' = 'Menn'
  OTHER = 'Uoppgitt kjønn'
;

```

Vi kan også angi datoer i utskriftsformatene:

```

PROC FORMAT;
VALUE periode
  '01jan1900'd - '31dec1994'd = '1900-1994'
  '01jan1994'd - '31dec1999'd = '1994-1999'
  '01jan2000'd - '31dec2010'd = '2000-2010'
;

```

Utskriftsformat-options:

DEFAULT=

Setter standard bredde for utskriftsformatet.

FUZZ=

Kan brukes til å sørge for at verdier som avviker lite fra intervallet som angis allikevel kommer med i intervallet. FUZZ-verdien sier hvor stort dette avviket skal være. Standardverdi er 1E-12.

MAX=

Største bredde vi ønsker utskriftsformatet skal ha.

MIN=

Minste bredde vi ønsker utskriftsformatet skal ha.

MULTILABEL

Angir at formatet kan ha overlappende verdier. Den første gang en verdi blir nevnt i et MULTILABEL-format defineres denne som primær-label. Brukes verdien flere ganger, regnes disse som sekundær-labeler. Foreløpig er det bare i prosedyrene MEANS og TABULATE vi kan bruke slike formater, se sidene 151 og 198. I alle andre prosedyrer vil kun primær-label kunne benyttes. Primær-label blir den teksten som har den laveste startverdien når verdier overlapper. Andre tekstlinjer som har overlappende kodeverdier regnes som sekundær-labeler. HLO-variabelen får en M som første eller andre tegn på alle verdiene ved bruk av MULTILABEL-formater.

NOTSORTED

Gjør at utskriftsformatene lagres i den rekkefølgen de har i VALUE-setningen. Bevirker at HLO-variabelen får S som første tegn på alle verdiene. Ved bruk av formater med NOTSORTED-option satt, kan vi i prosedyrene MEANS og TABULATE få formatene skrevet ut i den rekkefølgen de har i PROC FORMAT.

3.11.8 PROC FREQ

Proc freq er en prosedyre som lager frekvenstabeller. Tabellene kan lages i en eller flere dimensjoner. Proc freq gir oss frekvenser, kumulerte frekvenser, prosent og kumulerte prosent som standard. Proc freq egner seg godt for kategoriske variable (kjønn, fylke etc.). Kontinuerlige variable (inntekt, omsetning etc.) vil vanligvis lage svære, ubrukelige lister. Om vi likevel ønsker å lage frekvenstabeller over kontinuerlige variable, kan disse først grupperes vha. PROC FORMAT (se side 133). Vi må da ha med FORMAT-setningen i PROC FREQ, se under. Største antall linjer utskrift fra PROC FREQ er 32767 (Unix) pr. BY-gruppe.

```
PROC FREQ <options>;
  WHERE where-uttrykk ;
  BY <DESCENDING> variabel-1 ...<<DESCENDING> variabel-n> <NOTSORTED>;
  TABLES tabellspesifikasjoner</options>;
  OUTPUT OUT=SAS datasett <statistikknøkkelord> ;
  FORMAT variabel utskriftsformat. <variable utskriftsformat.>;
  WEIGHT variabel;
  EXACT statistikknøkkelord </ options>;
  TEST statistikknøkkelord ;
```

Options til PROC FREQ-setningen:

COMPRESS

Fortsetter med den neste 1-veis frekvenstabell på samme side, selv om det ikke er plass til hele tabellen på siden.

DATA= SAS datasett

Her oppgis navnet på SAS datasettet vi ønsker å skrive ut

FORMCHAR(1,2,7)= '3 tegn'

Brukes for å fortelle hvordan innrammingen av tabellen skal bli. Tegnene angir vertikale streker (1), horisontale streker (2) og krysningspunktene mellom horisontale og vertikale streker (7). Formchar (1, 2, 7) = ' | - + ' er standard.

ORDER=rekkefølge

Angir hvilken rekkefølge verdiene til grupperingsvariable skal skrives ut i. Velger vi ORDER=FORMATTED skrives de ut sortert etter teksten i utskriftsformatet som eventuelt brukes istedenfor etter koden. ORDER=FREQ sørger for at den verdi med flest observasjoner skrives først. ORDER=DATA gjør at verdiene beholder den rekkefølgen de har i datasettet, den første forekomsten av verdiene bestemmer rekkefølgen. Standardverdi er ORDER=INTERNAL som skriver ut verdiene i sortert rekkefølge. MISSING-verdier vil alltid komme ut først, uansett hvilken rekkefølge vi velger.

NOPRINT

Undertrykker utskrift. Brukes når tabellen skal skrives til et SAS-datasett med TABLES-option OUT=.

PAGE

Sørger for at det skrives ut en tabell pr. side. Vanligvis skrives så mange tabeller som mulig ut på hver side.

SAS-setninger til PROC FREQ:

WHERE

Selekterer hvilke observasjoner det skal lages frekvenstabeller for. Bare de som tilfredsstiller WHERE-betingelsen tas med. Beskrivelse av WHERE-setningen finnes på side 91.

BY

Det vil lages en tabell for hver verdi av BY-variablene. Datasettet må være sortert etter BY-variablene på forhånd (med mindre vi bruker sorteringsoption NOTSORTED).

TABLES

TABLES-setningen brukes til å definere tabellen(e) vi skal lage. Vi kan lage så mange tabeller vi ønsker i ubegrenset antall dimensjoner med en PROC FREQ. En enveis frekvenstabell spesifiseres med den ønskede variabelens navn. For å lage flere dimensjoner krysses variable. Dette gjøres med å sette en asterisk (*) mellom variablene. Parenteser kan også brukes for å forenkle skrivingen. Vi kan ha en eller flere TABLES-setninger i en PROC FREQ.

```
PROC FREQ DATA=innb;  
  TABLES fylke kjonn; /* lager en tabell for fylke og en for kjønn */  
PROC FREQ DATA=innb;  
  TABLES fylke*kjonn; /* lager en krysstabell for fylke*kjønn */  
  TABLES (fylke sivstat)*kjonn; /* lager krysstabeller fylke*kjønn og  
                                sivilstatus*kjønn */  
  TABLES fylke*kjonn sivstat; /* lager krysstabell fylke*kjønn og  
                                tabell for sivilstatus */  
  TABLES _CHARACTER_ ; /* lager en tabell for hver karaktervariabel */
```

Options til TABLES-setningen:

ALPHA=verdi

Spesifiserer konfidensintervallet 100(1 - verdi). Gyldige verdier er 0.0001-0.9999.

ALPHA=0.05 gir konfidensintervall på 95% (standard).

CONTENTS='tekst'

Angir tekst til html innholdsfil (laget med CONTENTS= i ODS-setningen) for krysstabeller.

CUMCOL

Skriver ut kumulerte kolonneprosenter, hvilket er standard.

DEVIATION

Beregner for hver celle avviket fra cellefrekvensens forventede verdi.

FORMAT=

Angir format for frekvenstallene i krysstabeller, f.eks. FORMAT=9.1.

LIST

Sørger for at tabellen skrives ut som en liste istedenfor som en krysstabell.

MISSING

Regner med MISSING som en gyldig verdi som tas med på lik linje med andre verdier.

MISSPRINT

Tar MISSING med i tabellen, men (i motsetning til MISSING-option) utelates i statistiske beregninger.

NOCOL

Undertrykker utskrift av kolonneprosenter for krysstabeller.

NOCUM

Undertrykker utskrift av kumulerte frekvenser og kumulerte prosenter.

NOFREQ

Undertrykker utskrift av frekvenser i krysstabeller

NOPERCENT

Undertrykker utskrift av prosentberegninger: prosenter av totalen fjernes ved krysstabeller, mens alle prosenter utelates for enveistabeller og når LIST-option brukes.

NOPRINT

Sørger for at tabellen ikke skrives ut. Bør brukes når tabellen lagres på et SAS datasett.

NOROW

Undertrykker rekkeprosenter i krysstabeller.

OUT=SAS datasett <statistikk>

Lagrer frekvenstabellen(e) på et SAS datasett. Bare den siste tabellspesifikasjonen i hver TABLES-setning blir skrevet ut på SAS datasettet. Variablene på SAS datasettet er (i tillegg til variable brukt til å spesifisere tabellen) COUNT (frekvensen) og PERCENT (prosenten). Foruten disse kan vi få med flere beregninger ved option OUTEXPECT og OUTPCT. MISSING-verdier blir med i COUNT, men ikke i de andre beregningene.

```
DATA persreg;  
INPUT komm $ 2-5 gkrets $ 7-9 kjonn $ 11 aldgrp $ 13-16 innt 18-23;  
CARDS;  
0101 001 K ald1 130327  
0101 001 M ald1 30327  
0211 004 ald5 45470  
0211 004 M ald5 45470  
0213 006 K ald2 55077  
    . 008 M ald1 58118  
;  
PROC FREQ DATA=persreg ;  
  TABLES kjonn / NOPRINT OUT=antkj ;  
PROC PRINT DATA=antkj;
```


Utskriften:

OBS	KJONN	COUNT	PERCENT
1		1	.
2	K	2	40
3	M	3	60

OUTCUM

Angir at kumulerte frekvenser og prosenter skal skrives ut når enveis-tabeller skrives ut til et nytt SAS datasett, med variabelnavn CUM_FREQ og CUM_PCT.

OUTEXPECT

Brukes for å få skrevet ut variabelen EXPECTED til SAS datasettet nevnt i OUT= option. Denne inneholder forventet frekvens for hver celle i tabellen.

OUTPCT

Brukes for å skrive ut rekke-, kolonne- og tabellprosenter til SAS datasettet nevnt i OUT= option. Variablene kalles PCT_ROW, PCT_COL og PCT_TABL (når tabellen har mer enn 2 dimensjoner).

SPARSE

Vil, brukt sammen med LIST- eller OUT= option, skrive ut alle mulige kombinasjoner av variabelenes verdier selv om de ikke finnes på SAS datasettet tabellen lages fra. Eksempel:

```
DATA persreg;
INPUT komm $ kjonn $ innb;
CARDS;
0101 K 30327
0101 M 30452
0106 M 42341
;
PROC FREQ DATA=persreg ;
TABLES komm*kjonn / LIST ;
TITLE 'Uten SPARSE';
PROC FREQ DATA=persreg ;
TABLES komm*kjonn / LIST SPARSE ;
TITLE 'Med SPARSE';
```

Forskjellen ser vi av utskriften under:

Uten SPARSE					
KOMM	KJONN	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0101	K	1	33.3	1	33.3
0101	M	1	33.3	2	66.7
0106	M	1	33.3	3	100.0
Med SPARSE					
KOMM	KJONN	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0101	K	1	33.3	1	33.3
0101	M	1	33.3	2	66.7
0106	K	0	0.0	2	66.7
0106	M	1	33.3	3	100.0

Vi kan bruke SPARSE-option for å få en oversikt over kombinasjoner av variable som mangler på et SAS-datasett. Det gjør vi ved å skrive ut de som har variabelen count lik 0 til et nytt SAS datasett. Deretter kan vi skrive dem ut.

```
PROC FREQ DATA=persreg NOPRINT ;
TABLES komm* kjonn / SPARSE OUT=mangler (WHERE=(COUNT=0) DROP=percent);
PROC PRINT DATA=mangler;
TITLE 'Kombinasjoner uten data' ;
```

Options og statistikknøkkelord som ikke er beskrevet her, men beskrives i det interaktive hjelpesystemet i SAS: AGREE, ALL, CELLCHI2, CHISQ, CL, CMH, CMH1, CMH2, CONVERGE=, EXACT, EXPECTED, JT, MAXITER=, MEASURES, PLCORR, RELRISK, RISKDIFF, SCORES=, TESTF=, TESTP=, TOTPCT, TREND og V5FMT.

OUTPUT OUT=

Brukes for å skrive ut resultatet til et SAS datasett. Denne varianten krever at vi bruker et eller flere av statistikknøkkelordene i TABLES-setningen (beskrevet i det interaktive hjelpesystemet). Statistikknøkkelordene: AJCHI, BDCHI, CMHCOR, CMHGA, CMHRMS, CONTGY, CQ, CRAMV, EQKAPS, EQWTKAPS, EXACT, JT, KAPPA, LAMCR, LAMDAS, LAMRC, LGOR, LGRRC1, LGRRC2, LRCHI, MCNEM, MHCHI, MHOR, MHRRC1, MHRRC2, N, NMISS, PCHI, PCORR, PHI, PLCORR, RDIF1, RDIF2, RRC1, RRC2, RROR, RSK1, RSK2, RSK11, RSK12, RSK21, RSK22, RELRISK, RISKDIFF, RISKDIFF1, RISKDIFF2, SCORR, SMDCR, SMDRC, STUTC, TREND, TSYMM, U, UCR, URC, WTKAPPA

FORMAT

Ved hjelp av FORMAT-setningen kan vi erstatte variabelverdier med mer forståelig *tekst* (f.eks. at verdien 1 erstattes med teksten Menn og verdien 2 erstattes med teksten Kvinner). Se for øvrig PROC FORMAT, side 133.

WEIGHT

For å veie dataene i tabellen bruker vi WEIGHT-setningen. Vi må da ha en variabel på datasettet som inneholder vekten (oppblåsningsfaktoren). Vektvariabelen kan gjerne være et desimaltall. Istedenfor at hver observasjon teller 1, vil hver telle vektvariabelens verdi. Er vektens verdi 0 eller MISSING, ignoreres observasjonen. Hvis vektvariabelen er negativ beregnes frekvenser, men prosentuering utelates.

EXACT

For å spesifisere statistikk til beregning av p-verdier. Følgende kan brukes: AGREE, BINOMIAL, CHISQ, FISHER, JT, KAPPA, LRCHI, MCNEM, MEASURES, MHCHI, OR, PCHI, PCORR, SCORR, TREND eller WTKAP. Krever spesielle options til TABLES-setningen.

TEST

For å spesifisere statistikk til asymptotiske tester. Følgende kan brukes: AGRRE, GAMMA, KAPPA, KENTB, MEASURES, PCORR, SCORR, SMDCR, SMDRC, STUTC og WTKAP. Krever AGREE, ALL eller MEASURES som option til TABLES-setningen.

3.11.9 PROC FSLIST

PROC FSLIST bruker vi til å se på eksterne filer. De filene det er aktuelt å se på er sekvensielle filer, ofte kalt flate filer. Når vi eksekverer prosedyren, vises filen i et eget vindu, kalt FSLIST-vinduet. I dette vinduet kan vi bla oppover og nedover i filen som vises. Vi kan også søke etter tekststrenger. Ellers er mulighetene nokså begrensede. Vi kan bruke en del av SAS kommandoene i FSLIST-vinduet, se side 26. Vær oppmerksom på at funksjonstastene i FSLIST vanligvis er forskjellige fra de andre vinduene.

PROC FSLIST <options>;

Aktuelle options til PROC FSLIST-setningen:

FILE='fil-navn'

Vi angir navnet på den eksterne filen vi skal se på her.

CAPS

Angir at søkestrenger gjøres om til store bokstaver før søket utføres.

HSCROLL=

n angir at det blas n posisjoner til siden når LEFT- eller RIGHT-kommandoene brukes, HALF at det skal blas et halvt skjermbilde og PAGE et skjermbilde

NOBORDER

Angir for Windows at filen vises i et eget vindu uten rammer.

NUM

Angir at det skal vises linjenr til venstre på hver linje når de står i posisjon 73-80 og recordlengden er 80. Dette er en overlevning fra stormaskin og ser heller ikke ut til å virke hverken under unix eller windows.

3.11.10 PROC FSVIEW

PROC FSVIEW bruker vi til å se på eller endre SAS datasett. Når vi kjører prosedyren, vises filen i et eget vindu, kalt FSVIEW-vinduet. I dette vinduet kan vi bla oppover og nedover i SAS datasettet som vises. Vi kan også søke etter tekststrenger. Vi kan bruke en del av SAS kommandoene i FSVIEW-vinduet, se side 26. I tillegg kan vi bruke DUP-kommandoen for å legge til en ny observasjon lik den vi står på. Vær oppmerksom på at funksjonstastene i FSVIEW vanligvis er forskjellige fra de andre vinduene. FSVIEW-vinduet minner om en forenklet utgave av VIEWTABLE-vinduet. FSVIEW er mer funksjonelt enn VIEWTABLE på noen områder.

PROC FSVIEW <options>;

WHERE where-uttrykk ;

FORMAT variabel utskriftsformat. <variable utskriftsformat.>;

ID variabel;

INFORMAT variabel innlesingsformat. <variable innlesingsformat.>;

VAR variabel ;

Aktuelle options til PROC FSVIEW-setningen:

DATA=SAS datasett

Navn på SAS datasettet som skal vises

NEW= SAS datasett

Angir at det skal lages et nytt SAS datasett. Hvis vi ikke bruker LIKE-option også, vil vi få opp et vindu der vi må definere variablene våre.

LIKE= SAS datasett

Kan brukes for nye SAS-datasett for å angi et mal-datasett. Det nye SAS-datasettet vil få foreslått de samme variable som mal-datasettet har.

AUTOADD

Det legges til en tom observasjon når SAS datasettet åpnes og hver gang ENTER-tasten trykkes. Brukes sammen med MODIFY-option.

BROWSEONLY

Angir at det bare skal være lov å se på SAS datasettet. Overstyrer MODIFY-option.

MODIFY

Angir at vi kan endre på SAS datasettet.

NOADD

Angir at vi ikke kan legge til nye observasjoner på SAS datasettet.

NOBORDER

Angir for Windows at filen vises i et eget vindu uten rammer.

NODELETE

Angir at vi ikke kan slette observasjoner fra SAS datasettet.

Aktuelle SAS-setninger til PROC FSVIEW:

WHERE

Selekterer hvilke observasjoner som skal vises. Bare de som tilfredsstiller WHERE-betingelsen tas med. Beskrivelse av WHERE-setningen finnes på side 91.

FORMAT

Vi bruker FORMAT-setningen til å erstatte variabelverdier med mer forståelig tekst. Disse utskriftsformatene må være definert på forhånd vha. PROC FORMAT, se side 133.

ID

Brukes for å angi en variable til å identifiserer observasjonen. ID-variabelen bør ha unike verdier.

VAR

I VAR-setningen lister vi opp variablene vi ønsker skal vises.

Aktuelle kommandoer i FSVIEW-vinduet:

=variabel

Blar fram til variabelen.

n

Blar til observasjonsnr n.

ADD <ON | OFF>

Legger til ny observasjon hver gang ENTER-tasten trykkes.

AUTOSAVE <n>

Angir antall endringer før dataene automatisk lagres.

DUP <n obsnr>

Dupliserer observasjon n ganger.

MODIFY <RECORD | MEMBER>

Utfører endringer på observasjonsnivå (RECORD) eller på SAS datasett-nivå (MEMBER)

PROTECT ON | OFF <variabel-1 ... <variable-n>

Beskytter variable mot endringer.

RENAME gammelt-variabelnavn nytt-variabelnavn

Endrer navn på en variabel.

SAVE

Lagrer SAS datasettet.

SORT <ASCENDING | DESCENDING> variabel1 ...<<ASCENDING | DESCENDING> variabeln>

Sorterer SAS datasettet.

WHERE <ALSO> where-uttrykk | <CLEAR> | <UNDO>

Selekter observasjoner som skal vises. Med ALSO legges WHERE-uttrykket til tidligere

WHERE-uttrykk. CLEAR får tilbake alle observasjoner. UNDO angrer siste WHERE-setning.

3.11.11 PROC MEANS

PROC MEANS er en meget anvendelig prosedyre som brukes både til statistiske beregninger og til å aggregere datasett. Vi bør bruke den når vi skal bli kjent med dataene våre. Vi kan lage forskjellige statistikker på numeriske data med PROC MEANS. Disse statistikkene hjelper oss til å få bedre oversikt over datamaterialet vårt. De vanligste statistikkene PROC MEANS produserer er gjennomsnitt (MEAN), sommer (SUM), maksimums- og minimumsverdier (MIN og MAX) og standardavvik (STD).

Vi bruker også PROC MEANS når vi ønsker å aggregere data. Når vi f.eks. ønsker å lage et datasett med 1 observasjon pr. kommune fra et datasett med 1 observasjon pr. lønsmottaker, er det PROC MEANS vi bruker.

```
PROC MEANS <options>;
WHERE where-uttrykk ;
BY <DESCENDING> variabel-1 ... <<DESCENDING> variabel-n> <NOTSORTED>;
CLASS grupperingsvariable </ options> ;
VAR analysevariable </WEIGHT=vekt-variabel> ;
TYPES grupperingsvariable ;
WAYS n1 <TO x1 <BY endringsverdi-1>> <, ... , nn <TO xn <BY endringsverdi-n>>>;
ID variable;
OUTPUT OUT=SAS datasett <utdataspesifikasjoner> <id-gruppe-spesifikasjon></options>;
FORMAT variabel utskriftsformat. <variable utskriftsformat.>;
WEIGHT variabel;
FREQ variabel;
```

Options til PROC MEANS-setningen:

ALPHA=verdi

Spesifiserer konfidensintervallet 100(1 - verdi). Gyldige verdier er 0.0001-0.9999.
ALPHA=0.05 gir konfidensintervall på 95%.

CHARTYPE

Angir at _TYPE_ skal være en karakttervariabel. Innholdet vil da være nuller og enere i totalsystemet (f.eks. er 0101 = 5 og 1111 = 15) som tilsvarer de tallene _TYPE_ vil inneholde når vi ikke bruker CHARTYPE-option.

CLASSDATA=

Angir et annet SAS-datasett som inneholder de kombinasjonene av grupperingsvariablene vi ønsker i resultatet. Se tilsvarende option i PROC TABULATE, side 194.

COMPLETETYPES

Angir at vi skal lage alle mulige kombinasjoner av grupperingsvariablene, også variabelverdier som ikke finnes i SAS-datasettet. Med option PRELOADFMT i CLASS-setningen og ORDER=DATA i PROC MEANS-setningen vil alle definerte tekster (formater) også komme med.

DATA= SAS datasett

Her oppgis navnet på SAS datasettet vi ønsker å behandle.

DESCENDTYPES

Sorterer utdatasettet (definert med OUTPUT OUT=) synkende etter _TYPE_-variabelen. Utfyllende om _TYPE_ under. Kan også skrives DESCENDING eller DESCEND.

EXCLUSIVE

Ekkluderer variable med kombinasjoner av grupperingsvariable som ikke finnes i SAS-datasettet definert i CLASSDATA=-option.

EXCLNPWGTS

Holder observasjoner som ikke har positive vektorer utenfor beregningene.

FW=feltbredde

Angir feltenes bredde på utskriften. FW=8 gir en feltbredde på 8 tegn.

IDMIN

Angir at minimumsverdier for ID-variable skal skrives ut istedenfor maksimumsverdier, som er standard.

MAXDEC=desimaler

Angir antall desimaler feltene i utskriften skal få (gyldige verdier 0-8).

MISSING

Sørger for at MISSING behandles som gyldig verdi for grupperingsvariable (definert i CLASS). Påvirker ingen statistiske beregninger.

NONOBS

Fjerner kolonnen i utskriften som inneholder antall observasjoner for hver gruppering. Denne kolonnen tilsvarer _FREQ_-variabelen på aggregert SAS-datasett, men den fjernes ikke med NONOBS-option.

NOPRINT

Undertrykker utskrift, brukes sammen med OUTPUT OUT=.

NWAY

Sørger for at utskriften kun inneholder de aggregerte data med høyest verdi for _TYPE_-variabelen, dvs. de mest detaljerte av de aggregerte dataene. Se under.

ORDER=rekkefølge

Angir hvilken rekkefølge verdiene til grupperingsvariable skal skrives ut i. Velger vi ORDER=FORMATTED sorteres de alfabetisk etter teksten i formatet som eventuelt brukes istedenfor etter koden. ORDER=FREQ sørger for at den verdi med flest observasjoner skrives først. ORDER=DATA gjør at verdiene beholder den rekkefølgen de har i datasettet. Den første forekomsten av verdiene bestemmer rekkefølgen. Dette gjelder ikke når vi bruker CLASSDATA- eller PRELOADFMT-option. Da blir rekkefølgen som definert i CLASSDATA-datasettet eller formatet til variabelen som har PRELOADFMT-option. Standardverdi er ORDER=UNFORMATTED (= INTERNAL) som skriver ut verdiene i sortert rekkefølge.

PRINT

Vis utskrift.

PRINTALLTYPES

Angir at alle kombinasjoner av grupperingsvariablene skal skrives ut. Vanligvis skrives bare den mest detaljerte kombinasjon av grupperingsvariablene ut (NWAY). Ingoneres hvis vi bruker NWAY-option, TYPES-setningen eller WAYS-setningen.

PRINTIDVARS

Skriver verdiene til ID-variablene i utskriften.

SUMSIZE

Angir hvor mye minne som kan brukes i aggregeringsberegningene. Angis i n, nK, nM eller nG bytes. Kan også angis som 0 (får da standardverdi angitt med den globale SUMSIZE=-option) eller MAX som tar alt tilgjengelig minne (vær forsiktig med dette!).

VARDEF=divisor

Angir divisor som brukes til å regne ut varianser. Gyldige verdier er:

DF	N-1 skal brukes som divisor (standard)
N	N skal brukes som divisor
WDF	Summen av vektene - 1 skal brukes som divisor
WEIGHT	Summen av vektene skal brukes som divisor

Foruten disse options kan vi angi de statistiske beregningene vi ønsker i PROC MEANS-setningen. De deskriptive er: CSS, CV, KURTOSIS, MAX, MEAN, MIN, N, NMISS, RANGE, SKEWNESS, STD, STDERR, SUM: SUMWGT, USS, og VAR, for nærmere beskrivelse, se side 269. I tillegg kan vi velge CLM (øvre og nedre grenseverdier for konfidensintervallet, definert med ALPHA=), LCLM og UCLM som er henholdsvis nedre og øvre grenseverdier for konfidensintervallet. De kvantile er MEDIAN, P1, P5, P10, Q1, Q3, P90, P95, P99 og QRANGE. I tillegg har vi PROBT og T for hypotesetesting.

Andre options: NOTRAP, QMARKERS, QMETHOD og QNTLDEF.

SAS-setninger til PROC MEANS:

WHERE

Selekterer hvilke observasjoner det skal lages statistikk for. Bare de som tilfredsstiller WHERE-betingelsen tas med. Beskrivelse av WHERE-setningen finnes på side 91.

BY

Det vil beregnes deskriptiv statistikk for hver verdi av BY-variablene. Datasettet må være sortert etter BY-variablene på forhånd (med mindre vi bruker sorteringsoption NOTSORTED).

Vi kan som oftest bruke CLASS istedenfor BY, da slipper vi å sortere datasettet på forhånd. Unntaket er når tabellen blir så stor at den ikke får plass i minnet. Da kan vi bruke BY isteden, fordi SAS da slipper å lagre hele tabellen i minnet og derfor greier seg med mindre.

CLASS

I CLASS-setningen lister vi opp grupperingsvariablene. Det lages deskriptiv statistikk for hver verdi av grupperingsvariablene. Når vi skal lage store aggregeringer, vil det ofte lønne seg å sortere datasettet først, for så å bruke BY-setningen istedenfor CLASS, se over.

Vi kan bruke følgende options til CLASS-setningen:

ASCENDING

Gjør at variabelen sorteres stigende i tabellen.

DESCENDING

Gjør at variabelen sorteres synkende i tabellen.

EXCLUSIVE

Ekkluderer variable med kombinasjoner av grupperingsvariable som ikke finnes i SAS-datasettet definert i CLASSDATA=option.

GROUPINTERNAL

Formater til grupperingsvariablene hektes på etter at SAS har gruppert dem. Dette kan spare noe maskintid. Kan brukes når formatene ikke grupperer forskjellige variabelverdier. Når vi bruker formater til å slå sammen grupper, f.eks. fra alder til 5-årige aldersgrupper, bør vi ikke bruke GROUPINTERNAL. Vi må også være oppmerksomme på restgrupperinger definert med OTHER i PROC FORMAT. Har disse forskjellige verdier, vil hver verdi komme på hver sin linje (ev. kolonne) i utskriften.

MISSING

Sørger for at MISSING behandles som gyldig verdi for grupperingsvariablene. Påvirker ingen statistiske beregninger.

MLF

Gjør at vi kan bruke to formattekster for hver verdi. Disse må være definert i PROC FORMAT, se side 140.

ORDER=

Angir hvilken rekkefølge verdiene til grupperingsvariable skal skrives ut i. Velger vi ORDER=FORMATTED blir de sortert alfabetisk etter teksten i formatet som eventuelt brukes istedenfor etter koden. ORDER=FREQ sørger for at den verdi med flest observasjoner skrives først. ORDER=DATA gjør at verdiene beholder den rekkefølgen de har i datasettet. Den første forekomsten av verdiene bestemmer rekkefølgen. Dette gjelder ikke når vi bruker CLASSDATA- eller PRELOADFMT-option. Da blir rekkefølgen som definert i CLASSDATA-datasettet eller formatet til variabelen som har PRELOADFMT-option. Standardverdi er ORDER=UNFORMATTED som skriver ut verdiene i sortert rekkefølge.

PRELOADFMT

Vi bruker PRELOADFMT for å si at vi ønsker å beholde rekkefølgen formatet fikk da det ble lagret. Med denne option og NOTSORTED-option til VALUE-setningen i PROC FORMAT, kan vi sikre at vi får linjene og kolonnene i den riktige rekkefølgen i utskriften. For at denne skal virke må option ORDER=data være med i PROC MEANS-setningen.

VAR

I VAR-setningen lister vi opp analysevariablene våre. Analysevariablene er de det beregnes statistikk på (SUM, MEAN, MAX, MIN etc.). Analysevariablene *må* være numeriske. Manglende verdier for analysevariable teller ikke med i beregningene (bortsett fra NMISS som er antall observasjoner med MISSING). Om vi definerer eventuell vekt-variabel her eller i WEIGHT-setningen blir det samme.

TYPES

Brukes for å selektare aggregeringsnivåer som skal ut på utskrift og/eller aggregert SAS-datasett. Aggregeringsnivåene skiller SAS ved å gi hvert nivå en egen verdi for variabelen `_TYPE_`. Variablene kombineres med * og det skal være minst en blank mellom kombinasjonene. Ønsker vi bare totalsummen kan vi angi TYPES ();.

Eksempel på program som aggregerer en fil med 3 grupperingsvariable hvor vi bare ønsker noen av nivåene på utdatasettet:

```
PROC MEANS DATA = sykh1999 NOPRINT MISSING;
  CLASS fylke aldgr gr1 gr3;
  VAR liggedag;
  TYPES fylke*aldgr
        fylke*aldgr*gr1
        fylke*aldgr*gr1*gr3
  ;
  OUTPUT OUT=pasagg (RENAME=( _FREQ_=antutskr ))
        SUM=
  ;
RUN ;
```

Dette programmet vil skrive ut de kombinasjonene av grupperingsvariable som er angitt i TYPES-setningen. Dette tilsvarer å selekttere slik (WHERE=(_TYPE_ IN(12,14,15))). Fordelen med å bruke TYPES-setningen istedenfor WHERE=-option er at det er lettere å gjøre riktig seleksjon.

WAYS

Dette er en annen måte å angi hvilke aggregeringsnivåer som skal være med på utskrift og/eller aggregert SAS-datasett enn den som brukes i TYPES-setningen. I WAYS-setningen angir vi et eller flere tall som definerer hvilke aggregeringsnivåer som skal være med på resultatet. Gyldige verdier er fra 0 (kun totalnivået) til antall grupperingsvariable i CLASS-setningen. Eksempel der alle to-veis-kombinasjoner av grupperingsvariablene:

```
PROC MEANS DATA = sykh1999 NOPRINT MISSING;
  CLASS fylke aldgr gr1 gr3;
  VAR liggedag;
  WAYS 2;
  OUTPUT OUT=pasagg (RENAME=( _FREQ_=antutskr ))
        SUM=
  ;
RUN ;
```

Dette tilsvarer TYPES fylke*aldgr fylke*gr1 fylke*gr3 aldgr*gr1 aldgr*gr3 gr1*gr3 eller (WHERE=(_TYPE_ IN(3,5,6,9,10,12))). Se også eksempel som viser _WAY_-variabelen under.

ID

Brukes for å inkludere ekstra variable fra inndatasettet på utdatasettet. Disse variablene brukes ikke i beregninger eller til gruppering. Derimot velges den høyeste verdi for ID-variabelen innen hver gruppe av BY-variable (med mindre IDMIN-option er angitt i PROC MEANS-setningen, da velges den laveste). Bruker vi flere ID-variable velges verdiene til den observasjonen som har høyest verdi når alle ID-variable settes sammen til en.

OUTPUT OUT=

Brukes når vi ønsker å legge resultatet av PROC MEANS ut på et SAS datasett. Dette brukes spesielt til å aggregere SAS datasett. De statistiske beregningene vi ønsker å utføre velges sammen med variabelnavn de skal få på det nye SAS datasettet.

```
PROC MEANS DATA=persreg NOPRINT;
  CLASS komm;
  VAR inntekt skatt;
  OUTPUT OUT=kommdata MEAN(inntekt)=gj_innt MEAN(skatt)=gj_skatt
        SUM(inntekt)=suminnt SUM(skatt)=sumskatt;
```

Resultatet:

KOMM	_TYPE_	_FREQ_	GJ_INNT	GJ_SKATT	SUMINNT	SUMSKATT
	0	25	230297.25	90695.88	5527134	2176701
0101	1	13	269340.00	93949.85	3232080	1221348
0102	1	4	261261.50	135670.50	1045046	542682
0103	1	2	117940.00	50847.50	235880	101695
0211	1	3	183006.00	88141.50	549018	176283
0213	1	2	153496.00	50275.50	306992	100551
0214	1	1	158118.00	34142.00	158118	34142

Som vi ser får utdatasettet to variable i tillegg til de variablene vi bestemmer; _TYPE_ og _FREQ_. _FREQ_ inneholder antall observasjoner innen hver verdi av grupperingsvariablene, mens _TYPE_ angir aggregeringsnivået. Den første observasjon i eksempelet over har _TYPE_ = 0, det betyr at observasjonen inneholder totaltall (25 observasjoner i alt, gjennomsnittsinntekt 230297.25 osv.) De resterende observasjoner har _TYPE_ = 1, de inneholder data på kommunenivå.

For hver grupperingsvariabel og hver kombinasjon av disse, får `_TYPE_`-variabelen en egen verdi, som brukes til å skille dem fra hverandre.

```
PROC MEANS DATA=persreg NOPRINT;
  CLASS komm kjonn;
  VAR inntekt skatt;
  OUTPUT OUT=kommdata SUM(inntekt)=suminnt SUM(skatt)=sumskatt;
```

Slik blir utskriften:

KOMM	KJONN	_TYPE_	_FREQ_	SUMINNT	SUMSKATT
		0	25	5527134	2176701
	K	1	12	3010535	1179510
	M	1	13	2516599	997191
0101		2	13	3232080	1221348
0102		2	4	1045046	542682
0103		2	2	235880	101695
0211		2	3	549018	176283
0213		2	2	306992	100551
0214		2	1	158118	34142
0101	K	3	7	1995390	735398
0101	M	3	6	1236690	485950
0102	K	3	2	515723	244179
0102	M	3	2	529323	298503
0103	K	3	1	192430	99382
0103	M	3	1	43450	2313
0211	M	3	3	549018	176283
0213	K	3	2	306992	100551
0214	M	3	1	158118	34142

Dette SAS datasettet inneholder en aggregert utgave av SAS datasettet persreg. Aggregeringer på alle kombinasjoner av grupperingsvariablene er utført og skrevet ut. Vi har totalsum (`_TYPE_ = 0`), summer fordelt etter kjønn (`_TYPE_ = 1`), summer etter kommune (`_TYPE_ = 2`) og summer fordelt etter kommune*kjønn (`_TYPE_ = 3`). Ved viderebehandling av datasettet er det viktig å bruke `_TYPE_`-variabelen for å sørge for å ha kontroll over hvilket aggregeringsnivå vi til enhver tid behandler. For hver grupperingsvariabel vi benytter, vil antallet nivåer øke.

Når vi aggregerer data til nye SAS datasett, er det som oftest kombinasjonen av grupperingsvariablene vi er interessert i (`_TYPE_ = 3` i eksempelet over), ikke de øvrige aggregeringsnivåer (`_TYPE_ = 0, 1` og `2`). Ved å bruke `NWAY`-option er det bare disse som blir skrevet ut.

```
PROC MEANS DATA=persreg NOPRINT NWAY;
  CLASS komm kjonn;
  VAR inntekt skatt;
  OUTPUT OUT=kommdata
    SUM(inntekt)=suminnt
    SUM(skatt)=sumskatt
  ;
PROC PRINT DATA=kommdata;
```

Her er resultatet:

KOMM	KJONN	_TYPE_	_FREQ_	SUMINNT	SUMSKATT
0101	K	3	7	1995390	735398
0101	M	3	6	1236690	485950
0102	K	3	2	515723	244179
0102	M	3	2	529323	298503
0103	K	3	1	192430	99382
0103	M	3	1	43450	2313
0211	M	3	3	549018	176283
0213	K	3	2	306992	100551
0214	M	3	1	158118	34142

`_FREQ_` variabelen inneholder altså antall observasjoner innen hver verdi av grupperingsvariablene. Når vi teller opp antall observasjoner med `N`, vil `MISSING` holdes utenfor, se neste eksempel.

```
PROC MEANS DATA=persreg NOPRINT;
  CLASS komm;
  VAR inntekt skatt;
  OUTPUT OUT=kommdata
    N(inntekt)=antinnt
    N(skatt)=antskatt
    SUM(inntekt)=suminnt
    SUM(skatt)=sumskatt
  ;
PROC PRINT DATA=kommdata;
```


Nytt resultat:

KOMM	_TYPE_	_FREQ_	ANTINNT	ANTSKATT	SUMINNT	SUMSKATT
	0	25	24	24	5527134	2176701
0101	1	13	12	13	3232080	1221348
0102	1	4	4	4	1045046	542682
0103	1	2	2	2	235880	101695
0211	1	3	3	2	549018	176283
0213	1	2	2	2	306992	100551
0214	1	1	1	1	158118	34142

Hvis vi ikke angir hvilke statistiske beregninger vi ønsker på det nye SAS datasettet, får vi N, MIN, MAX, MEAN og STD, som standard. Disse legges på hver sin observasjon, og kan skilles med verdien av _STAT_-variabelen, som også havner på det nye SAS datasettet, se under.

```
PROC MEANS DATA=persreg NOPRINT;
  CLASS kjonn;
  VAR inntekt skatt;
  OUTPUT OUT=kommdata;
PROC PRINT DATA=kommdata;
```

Nå blir resultatet slik:

KJONN	_TYPE_	_FREQ_	_STAT_	INNTEKT	SKATT
	0	26	N	25.00	25.00
	0	26	MIN	43450.00	2313.00
	0	26	MAX	442341.00	234145.00
	0	26	MEAN	231010.08	90573.28
	0	26	STD	103818.13	64117.86
K	1	12	N	12.00	12.00
K	1	12	MIN	55077.00	2313.00
K	1	12	MAX	442341.00	234145.00
K	1	12	MEAN	250877.92	98292.50
K	1	12	STD	106509.33	64226.95
M	1	14	N	13.00	13.00
M	1	14	MIN	43450.00	2313.00
M	1	14	MAX	442341.00	198273.00
M	1	14	MEAN	212670.54	83447.85
M	1	14	STD	101972.67	65774.37

Vi kan få skrevet ut kjennetegn ved den observasjonen som høyest (MAXID) og den som har lavest (MINID) verdi for en eller flere analysevariable. Hvis flere observasjoner har høyeste eller laveste verdi, velges den som leses (behandles) først.

```
PROC MEANS DATA=persreg NOPRINT ;
  CLASS komm;
  VAR inntekt skatt;
  OUTPUT OUT=skt
    MAXID(skatt(kjonn))=maxkjonn
    MINID(skatt(kjonn))=minkjonn
    MAX(skatt)=maxskatt
    MIN(skatt)=minskatt
  ;
PROC PRINT DATA=skt;
```

Her er utskriften:

OBS	KOMM	_TYPE_	_FREQ_	MAXKJONN	MINKJONN	MAXSKATT	MINSKATT
1		0	25	K	M	234145	2313
2	0101	1	13	K	M	234145	2313
3	0102	1	4	M	K	198273	67829
4	0103	1	2	K	M	99382	2313
5	0211	1	3	M	M	121002	55281
6	0213	1	2	K	K	98238	2313
7	0214	1	1	M	M	34142	34142

I eksempelet over fant vi kjønnet til den som har høyest skatt og til den som har lavest skatt. Siden flere har like lav skatt, velges den første av disse.

```
PROC MEANS DATA=persreg NOPRINT ;
  CLASS komm;
  VAR inntekt skatt;
  OUTPUT OUT=sktinnt
    MAXID(skatt(kjonn))=maxskkj
    MAXID(inntekt(kjonn))=maxinkj
    MAX(skatt)=maxskatt
    MAX(inntekt)=maxinnt
  ;
PROC PRINT DATA=sktinnt;
```

Eksempelet over lager et SAS datasett som viser hvilket kjønn henholdsvis den som tjener mest og den som skatter mest har. Resultatet ser vi her:

OBS	KOMM	_TYPE_	_FREQ_	MAXSKKJ	MAXINKJ	MAXSKATT	MAXINNT
1		0	25	K	M	234145	442341
2	0101	1	13	K	M	234145	442341
3	0102	1	4	M	K	198273	328863
4	0103	1	2	K	K	99382	192430
5	0211	1	3	M	M	121002	245470
6	0213	1	2	K	K	98238	251915
7	0214	1	1	M	M	34142	158118

For å finne hvilket kjønn og hvilken aldersgruppe den med høyest skatt tilhører kan vi bruke dette programmet.

```
PROC MEANS DATA=persreg NOPRINT ;
  CLASS komm;
  VAR inntekt skatt;
  OUTPUT OUT=skt
    MAXID(skatt(kjonn))=maxskkj
    MAXID(skatt(aldgrp))=maxskald
    MAX(skatt)=maxskatt
  ;
PROC PRINT DATA=sktinnt;
```

OBS	KOMM	_TYPE_	_FREQ_	MAXSKKJ	MAXSKALD	MAXSKATT
1		0	25	K	ald5	234145
2	0101	1	13	K	ald5	234145
3	0102	1	4	M	ald2	198273
4	0103	1	2	K	ald4	99382
5	0211	1	3	M	ald5	121002
6	0213	1	2	K	ald1	98238
7	0214	1	1	M	ald1	34142

Vi kan ha flere OUTPUT OUT= i samme PROC MEANS. Dette gjelder også når vi ikke bruker MAXID og MINID:

```
PROC MEANS DATA=persreg NOPRINT ;
  CLASS komm;
  VAR inntekt skatt;
  OUTPUT OUT=skt
    MAXID(skatt(kjonn))=maxkjonn
    MINID(skatt(kjonn))=minkjonn
    MAX(skatt)=maxskatt
    MIN(skatt)=minskatt
  ;
  OUTPUT OUT=innt
    MAXID(inntekt(kjonn))=maxkjonn
    MINID(inntekt(kjonn))=minkjonn
    MAX(inntekt)=maxinnt
    MIN(inntekt)=mininnt
  ;
PROC PRINT DATA=skt;
PROC PRINT DATA=innt;
```

OBS	KOMM	_TYPE_	_FREQ_	MAXKJONN	MINKJONN	MAXSKATT	MINSKATT
1		0	25	K	M	234145	2313
2	0101	1	13	K	M	234145	2313
3	0102	1	4	M	K	198273	67829
4	0103	1	2	K	M	99382	2313
5	0211	1	3	M	M	121002	55281
6	0213	1	2	K	K	98238	2313
7	0214	1	1	M	M	34142	34142
OBS	KOMM	_TYPE_	_FREQ_	MAXKJONN	MINKJONN	MAXINNT	MININNT
1		0	25	M	M	442341	43450
2	0101	1	13	M	K	442341	116341
3	0102	1	4	K	K	328863	186860
4	0103	1	2	K	M	192430	43450
5	0211	1	3	M	M	245470	145470
6	0213	1	2	K	K	251915	55077
7	0214	1	1	M	M	158118	158118

Id-gruppe-spesifikasjoner gir flere muligheter enn ID-setningen og MAXID/MINID til å se nærmere på ekstremverdier. IDGROUP spesifiseres slik:

IDGROUP (<MIN|MAX (variabel-liste-1) <...MIN|MAX (variabel-liste-n)>> <<MISSING> <OBS>
<LAST>> OUT <[n]>(id-variabel-liste)=<navn>)

MIN	Angir variable som skal sjekkes for lave ekstremverdier.
MAX	Angir variable som skal sjekkes for høye ekstremverdier.
MISSING	Angir at MISSING regnes som en verdi (den laveste).
OBS	Angir at det skal lages en eller flere variable som inneholder observasjonsnr til ekstremverdiene i inndatasettet.
LAST	Angir at den siste observasjonen skal velges hvis flere har samme ekstremverdier.
[n]	Angir antall ekstremverdier som skal skrives ut. Hver ekstremverdi får sin egen variabel definert.
navn	Angir navn på variable som skal lages.

Options til OUTPUT OUT=--setningen:

AUTOLABEL

Angir at SAS automatisk skal opprette labler til nye variable som lages. De lages ved å bruke eksisterende label med tillegg av den statistikk som er beregnet. Har ikke den opprinnelige variabelen label, brukes variabelnavnet pluss statistikk som er beregnet.

AUTONAME

Angir at SAS automatisk skal lage nye variabelnavn som består av opprinnelig variabelnavn pluss statistikk som er beregnet.

KEEPLEN

Angir at nye variable skal få samme lengde som de de er beregnet fra.

LEVELS

Nummerer observasjonene i utdatasettet innenfor hvert aggregeringsnivå og legges i variabelen `_LEVEL_`.

NOINHERIT

Angir at nye variable ikke skal arve labler eller formater fra variablene de er beregnet fra.

WAYS

Angir hvor mange grupperingsvariable som er krysset for å lage observasjonene som skrives ut. Resultatet legges i variabelen `_WAY_`.

Eksempel som viser verdiene til variablene `_TYPE_`, `_FREQ_`, `_LEVEL_` og `_WAY_`:

```
DATA bef;
  INPUT @01 Region      $CHAR1.
        @03 Kjonn       $CHAR1.
        @05 Aar         $CHAR4.
        @10 Innbyggere   7.
        ;
CARDS;
1 1 1998 1064171
1 1 1999 1083508
1 2 1998 1106388
1 2 1999 1123656
2 1 1998 307474
2 1 1999 314292
2 2 1998 310572
2 2 1999 316787
3 1 1998 580073
3 1 1999 586075
3 2 1998 584474
3 2 1999 589851
4 1 1998 233388
4 1 1999 233265
4 2 1998 231059
4 2 1999 231063
;
```

```
PROC MEANS DATA=bef NOPRINT ;
  CLASS region aar;
  VAR innbyggere;
  OUTPUT OUT=befagg SUM= /LEVELS WAYS;

PROC PRINT DATA=befagg ;
  TITLE 'Variablene _TYPE_, _FREQ_, _LEVEL_ og _WAY_ ' ;
RUN;
```

Utskrift fra PROC PRINT:

Variablene _TYPE_, _FREQ_, _LEVEL_ og _WAY_							
Obs	Region	Aar	_WAY_	_TYPE_	_LEVEL_	_FREQ_	Innbyggere
1			0	0	1	16	8896096
2		1998	1	1	1	8	4417599
3		1999	1	1	2	8	4478497
4	1		1	2	1	4	4377723
5	2		1	2	2	4	1249125
6	3		1	2	3	4	2340473
7	4		1	2	4	4	928775
8	1	1998	2	3	1	2	2170559
9	1	1999	2	3	2	2	2207164
10	2	1998	2	3	3	2	618046
11	2	1999	2	3	4	2	631079
12	3	1998	2	3	5	2	1164547
13	3	1999	2	3	6	2	1175926
14	4	1998	2	3	7	2	464447
15	4	1999	2	3	8	2	464328

FORMAT

Vi bruker FORMAT-setningen til å erstatte variabelverdier med mer forståelig tekst. Disse utskriftsformatene må være definert på forhand vha. PROC FORMAT, se side 133. FORMAT-setningen kan også brukes til å gruppere variabelverdier, f.eks. etter aldersgrupper eller inntektsgrupper.

WEIGHT

For å veie dataene bruker vi WEIGHT-setningen. Vi må da ha en variabel på datasettet som inneholder vekten (oppblåsningsfaktoren). Vektvariabelen kan gjerne være et desimaltall.

FREQ

Hvis hver observasjon på datasettet skal telle for mer enn 1, kan vi bruke FREQ-setningen. I FREQ-setningen kan vi da angi variabelen hvis verdi inneholder det antall observasjoner denne observasjonen skal telle som. Dette betyr at N også økes med verdien til FREQ-variabelen. FREQ trunkerer alle desimaler i FREQ-variabelen før den brukes.

3.11.12 PROC OPTIONS

Ved å kjøre PROC OPTIONS får vi en oversikt over hva de forskjellige SAS system options er satt til. Denne lista skrives til SASlog.

```
PROC OPTIONS <SHORT | LONG> <HOST> <OPTION=option> <PORTABLE | NOHOST> ;
```

Proc options gir som standard en liste over alle SAS system options. Option SHORT gir en kortere liste, mens OPTION=option gir opplysninger om den option vi ber om (maksimalt en pr. PROC-steg) blant de som er definert som PORTABLE. HOST og PORTABLE avgrenser også utlistingen.

Oversikt over en del SAS system options finnes på side 40.

3.11.13 PROC PRINT

Proc print bruker vi til å skrive ut observasjoner fra et SAS datasett. Vi kan modifisere utseendet og innholdet i utskriften ved å bruke SAS-setninger og options som er beskrevet under.

```
PROC PRINT <options>;
WHERE where-uttrykk ;
BY <DESCENDING> variabel-1 ... <<DESCENDING> variabel-n> <NOTSORTED>;
VAR variable;
ID variable;
PAGEBY BY-variabel;
SUMBY BY-variabel;
SUM variable;
FORMAT variabel utskriftsformat. <variabel utskriftsformat.>;
```

Options til PROC PRINT-setningen:

CONTENTS='tekst'

Angir tekst til html innholdsfil (laget med CONTENTS= i ODS-setningen) for utskriften.

DATA= SAS datasett

Her oppgis navnet på SAS datasettet vi ønsker å skrive ut

DOUBLE

Skriver ut en blank linje mellom hver observasjon på utskriften. Kan forkortes til D

HEADING=

Vi kan her selv velge å skrive ut variablenes overskrift horisontalt (H) eller vertikalt (V)

LABEL

Skriver ut variablenes label (tekst) i hode istedenfor variablenes navn. Kan forkortes til L.

N

Skriver ut antall observasjoner i datasettet på slutten av utskriften. Når vi bruker BY-setningen, vil antall observasjoner skrives ut etter hver verdi for BY-variablene. For å få ut totalt antall observasjoner ved bruk av BY, må vi ha med en SUM-setning, se denne. Vi kan her angi alternative N-tekster, den første vi nevner vil gjelde for hver BY-grupper når vi har det, ellers vil den gjelde for N-teksten nederst i utlistingen. Bruker vi BY-grupper og SUM-setning kan vi, ved å ha med en tekst til, få den første av disse etter hver BY-gruppe og den andre nederst i utlistingen (f.eks. N='Antall pr. gruppe: ' 'Antall i alt: ').

NOOBS

Skriver ikke ut observasjonsnummer

OBS=

Angir tekst som skal stå over kolonnen med antall observasjoner.

ROUND

Runder av numeriske variable til 2 desimaler eller til det antall desimaler som er angitt i en FORMAT-setning for variablene. Kan forkortes til R. Har vi med SUM-setning vil det summerte tallet bli summen av de enkelte tall etter at de er avrundet.

ROWS=PAGE

Når det ikke er plass til alle variablene på en sidebredde, deler vanligvis SAS siden slik at alle variablene til en observasjon kommer på den samme siden (hvis det er plass). Med ROWS=PAGE vil SAS skrive ut variablene som får plass i bredden for så mange observasjoner det er plass til på siden, og deretter fortsette på neste side med de resterende variablene for disse observasjonene.

SPLIT='delingstegn'

Deler kolonneoverskrifter der delingstegnet er satt. Delingstegnet settes med LABEL-setningen som en del av teksten til variabelnavnet. Kan forkortes til S.

```
PROC PRINT DATA=pasient.g95 SPLIT='\' ;
LABEL komm 'Kommune-\nummer'
```

UNIFORM

Sørger for at layouten blir lik på alle sider i utskriften, dvs at alle variable plasseres i samme posisjoner på utskriften. Kan forkortes til U

WIDTH=

Brukes til å bestemme kolonnenes (variablenes) bredde på utskriften

FULL	Bruker variablenes formaterte bredde som kolonnebredde
MINIMUM	Bruker minimumsbredde for variablene. Kan variere fra side til side. Kan forkortes til MIN
UNIFORM	Sørger for at layouten blir lik på alle sider i utskriften, dvs at alle variable plasseres i samme posisjoner på utskriften. Kan forkortes til U
UNIFORMBY	Sørger for at layouten blir lik på alle sider i samme BY-gruppe på utskriften. Kan forkortes til UBY

SAS-setninger til PROC PRINT:

WHERE

Selekterer hvilke observasjoner som skal skrives ut. Bare de som tilfredsstiller WHERE-betingelsen skrives ut. Beskrivelse av WHERE-setningen finnes på side 91.

BY

Utskriften vil bli inndelt med en liste for hver verdi av BY-variablene. Datasettet må være sortert etter BY-variablene på forhånd (med mindre vi bruker sorteringsoption NOTSORTED)

VAR

Her lister vi de variablene vi ønsker å skrive ut. Variablene skrives ut i den rekkefølgen vi lister dem i.

ID

Variable listet her brukes som identifikasjon istedenfor observasjonsnummeret. Variablene som listes her bør til sammen være unike (bare en kombinasjon av disse i datasettet).

PAGEBY

Bevirker at det skiftes side for hver ny verdi av PAGEBY-variabelen. Krever at PAGEBY-variabelen også er en BY-variabel. Vi kan bare angi en variabel her selv om det kan være flere BY-variable i BY-setningen. Angir vi den siste av BY-variablene skiftes det side når en av BY-variablene endrer verdi.

SUMBY

Bevirker at det lages summer for numeriske variable for hver ny verdi for SUMBY-variabelen. Krever at SUMBY-variabelen også er BY-variabel. Vi kan bare angi en variabel her selv om det kan være flere BY-variable i BY-setningen. Angir vi den siste av BY-variablene lages det også delsummer for de andre BY-variablene.

SUM

Lager summer for de numeriske variablene som listes i SUM-setningen.

FORMAT

Vi bruker FORMAT-setningen til å erstatte variabelverdier med mer forståelig *tekst*. Disse formatene må være definert på forhånd vha. PROC FORMAT, se side 133. FORMAT-setningen kan også brukes for å forbedre layout på tall. Til det kan vi bruke egendefinerte mønster-formater laget med PROC FORMAT eller vi kan bruke innebygde SAS-formater.

Eksempler:

Vi skal skrive ut alle observasjoner med *inndato* større en *utdato*, vi ønsker å vite hvor mange det er (option N) og vi vil ha utfyllende tekster (option LABEL) over hver variabel:

```
PROC PRINT DATA=passtat.pas95 LABEL N;  
  WHERE inndato > utdato;  
  VAR inr pasnr inndato utdato ;  
  FORMAT inndato utdato YYMMDD6. ;  
  TITLE 'Pasienter skrevet inn etter at de skrevet ut';  
RUN;
```

Vi ønsker å skrive ut sosialhjelpsmottagere i Evje og Hornnes (0937) og Åseral (1026) kommuner, vi vil vite antallet (option N), summere stønadsbeløpene (SUM-setningen) og skrive ut slik at variablene plasseres likt på hver side (UNIFORM). SAS datasettet er sortert etter kommuner, da kan vi få en liste for hver (BY-setningen):

```
PROC PRINT DATA=soshjelp.sos95 N UNIFORM;  
  WHERE kommnr IN ('0937','1026');  
  BY kommnr;  
  VAR fnr bidrag laan stonad ;  
  SUM bidrag laan stonad ;  
  TITLE 'Sosialhjelpsmottakere';
```

3.11.14 PROC PRINTTO

Brukes for å endre destinasjon for utskrift fra andre prosedyrer eller for SASlog. Ønsker vi å styre utskrift fra forskjellige prosedyrer til forskjellige filer i samme program, kan vi bruke PROC PRINTTO. Først kjører vi PRINTTO for å fortelle hvor utskriften skal havne, deretter den aktuelle prosedyren, så en ny PRINTTO og en ny prosedyre osv. For å endre tilbake til standard utskriftssted, kjøres PROC PRINTTO uten parametre.

PROC PRINTTO <LOG=filreferanse | 'filnavn' | LOG> <NEW>;
PROC PRINTTO <PRINT | FILE | NAME=filreferanse | 'filnavn' | PRINT> <NEW>;

LOG= Angir hvor SASlog skal skrives.
 PRINT= Angir hvor utskrifter skal sendes.
 NEW Angir at eksisterende utskrifter skal slettes før neste bruk.

Eksempel:

```
PROC PRINTTO PRINT='tab01.lst' NEW;

PROC TABULATE DATA=tilbfall.a3694 NOSEPS FC=' -----' F=6. MISSING;
VAR  tilbf teller;
CLASS kjonn alder antrf;
TABLE
    ALL='Alle straffbare'*(ALL='I alt' alder='')*
    (tilbf=' '*PCTSUM<teller>='')
    kjonn=' '* (ALL='I alt' alder='')*
    (tilbf=' '*PCTSUM<teller>='') ,
    ALL='I alt' antrf='Tidligere reaksjoner'
    /BOX='Kjonn og alder' RTS=40 ROW=FLOAT MISSTEXT='-';
FORMAT alder alder.
       antrf anttb.
       kjonn $kjonn.
;
TITLE 'Tabell 1. Tilbakefall blant straffbare. Prosent';

PROC PRINTTO PRINT='tab02.lst' NEW;

PROC TABULATE DATA=tilbfall.a3694 NOSEPS FC=' -----' F=6. MISSING;
VAR  tilbf teller;
CLASS forbr alder antrf;
TABLE
    forbr=' '* (ALL='I alt' alder='')*
    (ALL='I alt' *n=' ' tilbf='M/tilbakefall' *SUM=' '
    tilbf='Prosent' *PCTSUM<teller>='')
    ,
    ALL='I alt' antrf='Tidligere reaksjoner'
    /BOX='Hovedforbrytelse og alder' RTS=70 ROW=FLOAT MISSTEXT='-';
FORMAT alder alder.
       antrf anttb.
       forbr $forbr.
;
TITLE 'Tabell 2. Tilbakefall blant straffbare. Prosent';

PROC PRINTTO;

RUN;
```

Den første tabellen styres til filen *tab01.lst* og den andre til *tab02.lst*. Til slutt settes utskriftssted tilbake til standard.

3.11.15 PROC RANK

Vi bruker PROC RANK til å rangere data etter verdiene i en eller flere numeriske variable. Rangeringene kan vi legge i egne variable eller de kan legges ut i de opprinnelige variablene istedenfor de opprinnelige verdiene. MISSING-verdier holdes utenfor rangeringen, dvs. rangeringsvariabelen blir også MISSING. Like verdier vil alltid få samme rangering. Vi kan rangere observasjonene i like store grupper, dvs. med like mange observasjoner i hver gruppe (hvis mulig).

```
PROC RANK <options> ;  
  WHERE where-uttrykk ;  
  BY <DESCENDING> variabel-1 ... <<DESCENDING> variabel-n> <NOTSORTED>;  
  VAR variabel-1 ... <variable-n> ;  
  RANKS variabel-1 ... <variable-n> ;
```

Options til PROC RANK-setningen:

DATA=SAS datasett

Her oppgir vi navnet på SAS datasettet vi ønsker å behandle.

OUT=SAS datasett

Navn på utdatasett

DESCENDING

Omvendt rangering. Den største verdi får lavest rangering.

FRACTION

Gjør rangeringen til en brøk som er rangeringen delt på antall observasjoner i alt som ikke er MISSING. Dette gir desimaltall mellom 0 og 1 som rangering. Bruker TIES=HIGH som standard.

GROUPS=n

Deler rangeringen i like store grupper (hvis mulig). F.eks. vil GROUPS=4 dele datasettet i 4 like store deler, noe vi kan benytte oss av for å beregne kvartiler.

NORMAL=BLOM | TUKEY | VW

Deler rangeringen etter normalfordelingsprinsipp, som rangering gis normalfordelte verdier etter litt forskjellige formler.

NPLUS1

Samme som FRACTION, men nevneren er n+1 istedenfor n. Bruker TIES=HIGH som standard.

PERCENT

Samme som FRACTION, men resultatet er multiplisert med 100, dermed vises rangeringen som en prosent. Bruker TIES=HIGH som standard.

SAVAGE

Deler rangeringen etter Savage (eksponentiell) fordeling

TIES=MEAN | HIGH | LOW

Brukes til å spesifisere hvilken verdi som skal velges som rangeringsverdi, når flere observasjoner har samme verdi for variabelen som rangeres.

MEAN velger gjennomsnittsverdien

HIGH velger den høyeste verdien

LOW velger den laveste verdien

SAS-setninger til PROC RANK:

WHERE

Selekterer hvilke observasjoner som skal rangeres. Bare de som tilfredsstiller WHERE-betingelsen tas med. Beskrivelse av WHERE-setningen finnes på side 91.

BY

Det lages egne rangeringer for hver verdi av BY-variablene. Datasettet må være sortert etter BY-variablene på forhånd (med mindre vi bruker sorteringsoption NOTSORTED).

VAR

I VAR-setningen lister vi opp de variablene som skal rangeres. Disse må være numeriske. Utelater vi VAR-setningen utføres beregningene på alle numeriske variable. VAR-setningen må være med når RANKS-setningen brukes.

RANKS

I RANKS-setningen navngir vi rangeringsvariablene. Den som nevnes først, vil inneholde rangeringen til den som er nevnt først i VAR-setningen, den andre vil inneholde rangeringen til den andre i VAR-setningen osv. Hvis vi utelater RANKS-setningen, vil rangeringen legges tilbake i den opprinnelige variabelen. Hvis vi ønsker å beholde de opprinnelige verdiene må vi altså bruke RANKS-setningen.

Eksempel der vi ønsker å lage en tabell som inneholder desiler for inntekt, dvs. vi ønsker å finne gjennomsnittsinntekt og prosent av total for lønnstakere, fordelt etter 10 grupper med like mange personer i hver gruppe. Vi bruker PROC RANK til å lage denne tidelingen og PROC TABULATE til å lage tabellen etterpå.

```
* Lager testdatasett;
DATA inntekt;
DO fylke = 1 TO 5;
  DO j = 1 TO 150 + fylke**2;
    innt = ROUND(250000*ABS(RANNOR(346836317)));
    OUTPUT;
  END;
END;

* Rangerer inntektene i 10 grupper;
PROC RANK DATA=inntekt OUT=rinntekt GROUPS=10;
  BY fylke;
  VAR innt;
  RANKS rinnt;

PROC FORMAT;
  VALUE desil
    0 = '1. desil'
    1 = '2. desil'
    2 = '3. desil'
    3 = '4. desil'
    4 = '5. desil'
    5 = '6. desil'
    6 = '7. desil'
    7 = '8. desil'
    8 = '9. desil'
    9 = '10. desil'
  ;

PROC TABULATE DATA=rinntekt NOSEPS FC=' -----' F=7.;
  CLASS fylke rinnt;
  VAR innt ;
  TABLE all='I alt' fylke='',
    (ALL='I alt' rinnt='')
    *innt='*(MEAN='Snitt' PCTSUM<rinnt ALL>='Prosent'*F=5.1)
    innt='Antall personer'*N='
    /RTS=26;
  FORMAT rinnt desil.;
  TITLE 'Desiler for inntekter etter fylke';
RUN;
```

3.11.16 PROC REPORT

PROC REPORT er en prosedyre for å lage rapporter. Disse rapportene er i utgangspunktet ikke så veldig brukbare for våre formål, men ved å utnytte mulighetene i prosedyren, kan vi få rapportene til å se ut som ganske pene tabeller. Å få til dette krever imidlertid at vi har en del programmeringserfaring.

Prosedyren lager rapporter. Disse rapportene deles opp i kolonner. Vi skal definere de forskjellige kolonnene rapporten skal bestå av. Forspalten regnes også som en eller flere kolonner. I REPORT kan vi definere flere forskjellige typer kolonner; grupperingskolonner nedover (forspalten), grupperingskolonner bortover (i hodet), analysekolonner (sum, gjennomsnitt osv.) og beregnede kolonner (sum av to andre kolonner etc.). Noen av kolonnene kan krysses med hverandre, f. eks. kan vi ha en gruppering i hodet hvor vi i tillegg bruker en analysevariabel.

Utskriften fra PROC REPORT kan vi enten få i det vanlige OUTPUT-vinduet (anbefales) eller i et eget resultatvindu for prosedyren. Vi kan lage rapportene på to forskjellige måter, enten ved å skrive REPORT-setninger i program-editoren, som erfarne programmerere vil foretrekke, eller med pek og klikk ved hjelp av egne vinduer. Her vil vi beskrive hvordan vi lager rapportene i program-editoren. Options som brukes til interaktiv rapportskriving er stort sett utelatt her.

PROC REPORT arbeider med kolonner. Disse refererer vi vanligvis til med de tilhørende variabelnavn. Det er mest oversiktlig og gir mindre sjanser for at tabellen blir feil. Av og til kan det allikevel være nødvendig å referere til kolonnen ved å henvise til kolonnennummeret. For å referere til en bestemt kolonne, bruker vi `_Cn_`, der n er kolonnennummeret (som vi selv må finne). Bruker vi slike referanser må vi huske på å oppdatere dem hvis vi legger til eller fjerner kolonner til venstre for de refererte fra rapporten.

```
PROC REPORT <options>;
  WHERE where-uttrykk ;
  BY <DESCENDING> variabel-1 ... <<DESCENDING> variabel-n> <NOTSORTED>;
  COLUMN kolonnespesifikasjoner ;
  DEFINE report-variabel </> <options> <'overskrift-1' <... 'overskrift-n'>> ;
  BREAK brudd-variabel </brudd-options> ;
  RBREAK bruddplassering </brudd-options> ;
  COMPUTE plassering <brudd-variabel> | <_PAGE_ <LEFT | CENTER | RIGHT>>
    </> STYLE=[stil-spesifikasjoner] ;
  LINE <peker> variabel <format> | <n*> 'tekst' ;
  ENDCOMP;
  COMPUTE report-variabel </> CHARACTER <LENGTH=lengde>;
  CALL DEFINE (_COL_, 'attributtnavn', attributtverdi);
  ENDCOMP;
  FORMAT variabel utskriftsformat. <variable utskriftsformat.>;
  FREQ variabel;
  WEIGHT variabel;
```

Options til PROC REPORT-setningen:

BOX

Angir at rapporten og de enkelte celler skal rammes inn med de tegnene som er definert med system option FORMCHAR . Kan ikke brukes sammen med WRAP-option eller med FLOW-option i DEFINE-setningen.

CENTER

Angir om rapporten skal sentreres (CENTER) eller venstrejusteres (NOCENTER).

COLWIDTH=kolonnebredde

Angir standard bredde for kolonnene i rapporten.

COMMAND

Angir at kommandolinje skal vises i REPORT-vinduene når vi bruker dem.

CONTENTS='tekst'

Angir tekst til html innholdsfil (laget med CONTENTS= i ODS-setningen) for rapporten.

DATA= SAS datasett

Her oppgis navnet på SAS datasettet vi ønsker å lage rapporten fra.

EXCLNPWGT

Ekskluderer observasjoner der vektvariabelens verdi (definert i WEIGHT-setningen) er 0 eller mindre. Uten denne option regnes negative verdier for vektvariablene som 0 og observasjonene inngår i opptelling av antall observasjoner (N).

FORMCHAR

Brukes for å angi tegn for innramming av tabellen. Tegnenes rekkefølge er de samme som i PROC TABULATE: 1. er vertikal strek (| er standard), 2. er horisontal strek (-), 3. er øvre venstre hjørne (-), 4. er midten øverst (-), 5. er øvre høyre hjørne (-), 6. er midten til venstre (|), 7. er midt i tabellen (+), 8. er midten til høyre (|), 9. er nedre venstre hjørne (-), 10. er midten nederst (-) og 11. er nedre høyre hjørne (-). I tillegg brukes 13 til å angi tegn for dobbel understreking av sumlinjer (=). Vi kan f.eks. endre tegn slik: FC(7,13)='-'

HEADLINE

Legger inn en strek under kolonneoverskriftene.

HEADSKIP

Legger inn en blank linje under kolonneoverskriftene.

HELP=libref.katalog

Angir en katalog for hjelpeskjermbilder.

LIST

Lister ut SAS-programmet som er kjørt til SASlog. Det programmet SAS kjører avviker noe fra det vi skriver uten at det skal ha noen betydning.

LS=linjestørrelse

Angir linjestørrelsen for rapporten.

MISSING

Sørger for at MISSING behandles som gyldig verdi for grupperingsvariable (definert med GROUP, ORDER eller ACROSS i DEFINE-setningen). Påvirker ingen statistiske beregninger.

NAMED

Skriver kolonneoverskriften ut foran hvert tall i tabellen.

NOEXEC

Gjør at rapporten ikke lages, men programmet syntakssjekkes.

NOHEADER

Undertrykker kolonneoverskrifter.

OUT=

Navn på SAS datasett rapporten skal skrives ut til. SAS datasettet vil inneholde en variabel for hver kolonne i rapporten og en observasjon for hver linje. I tillegg vil det lages en variabel kalt `_BREAK_` når vi bruker `BREAK-` eller `RBREAK-` setningene. Denne inneholder navn på bruddvariabelen for hver enkelt observasjon. Grupperingskolonner i hodet (definert som ACROSS i DEFINE-setningen) vil få kolonnennummeret som variabelnavn etter mønster `_Cn_`, der n er kolonnennummeret. Kolonneoverskrifter vil bli brukt som utfyllende tekst (LABEL). Vi kan ikke bruke `OUT=` sammen med `BY`-setningen i PROC REPORT.

OUTREPT=libref.katalog.programnavn

brukes for å lagre rapporten i en SAS-katalog. Dette gjør vi vanligvis ikke, vi lagrer programmet i programfiler isteden.

PANELS=spalter

Angir antall spalter rapporten skal skrives ut i på hver side.

PROFILE=libref.katalog

Navngir en SAS-katalog som inneholder en standardprofil for PROC REPORT.

PROMPT

Angir at vi får opp et dialogbildet når vi lager rapporten

PS=sidestørrelse

Angir sidestørrelsen for rapporten.

PSPACE=n

Antall posisjoner mellom hver spalte. Brukes sammen med PANELS. Standard er 4.

REPORT=libref.katalog.programnavn

Navngir et program som inneholder en standardrapport

SHOWALL

Sørger for å skrive alle kolonner og verdier, selv om de egentlig er undertrykket med NOPRINT- eller NOZERO-option i DEFINE-setningen.

SPACING=n

Antall posisjoner mellom kolonnene i rapporten. Standard er 2.

SPLIT='tegn'

Angir et tegn for linjebrudd i kolonneoverskriftene. Tegnet kommer ikke med i utskriften.
Standard er linjebrudd ved /.

STYLE<(rapportdel)> = <stilnavn> <(stilattributter)>

Brukes for å endre utseende på forskjellige deler av rapporten for html-filer. Rapportdelene er:

CALLDEF	Alle celler identifisert ved CALL DEFINE-setninger
COLUMN	Cellene i kolonnene
HEADER	Kolonneoverskrifter
LINES	Alle linjer definert med LINE-setninger
REPORT	Generelt for hele rapporten

Stilnavn og attributter er beskrevet i vedlegget om html-stiler, se side 266

VARDEF=divisor

Angir divisor som brukes til å regne ut varianser. Gyldige verdier er:

DF	N-1 skal brukes som divisor (standard)
N	N skal brukes som divisor
WDF	Summen av vektene - 1 skal brukes som divisor
WEIGHT	Summen av vektene skal brukes som divisor

WINDOWS

WINDOWS gjør at rapporten havner i et eget rapportvindu, mens NOWINDOWS sørger for at rapporten havner i OUTPUT-vinduet (SASlist).

WRAP

Lister ut alle verdiene for en logisk linje i rapporten på linjer etter hverandre istedenfor å liste ut kolonnene det er plass til på linjen i puljer.

SAS-setninger til PROC REPORT:

WHERE

Selekterer hvilke observasjoner det skal lages rapport for. Bare de som tilfredsstiller WHERE-betingelsen tas med. Beskrivelse av WHERE-setningen finnes på side 91.

BY

Det vil lages en rapport for hver verdi av BY-variablene. Datasettet må være sortert etter BY-variablene på forhånd (med mindre vi bruker sorteringsoption NOTSORTED).

COLUMN

Angir hvilke variable som skal brukes i rapporten. Alle variable som skal brukes på en eller annen måte må angis her. I kolonnedefinisjonene kan vi også angi disse statistikkene; CSS, CV, MAX, MEAN, MIN, N, NMISS, PCTN, PCTSUM, PRT, RANGE, STD, STDERR, SUM, SUMWGT, T, USS, og VAR, for nærmere beskrivelse, se side 269. N kan brukes på alle typer variable, når denne knyttes mot analysevariable telles antall observasjoner som ikke er MISSING for denne variabelen, for grupperingsvariable fordeles antall observasjoner i SAS datasettet etter grupperingsvariabelens verdier og MISSING regnes som en egen verdi. Denne kommer med som en egen kategori hvis MISSING-option er med i PROC REPORT-setningen. De andre statistikkene må knyttes til analysevariable. I tillegg har vi PCTN og PCTSUM som regner ut prosent av henholdsvis antall observasjoner og summen av en tilordnet analysevariabel. Prosentene vises som andelstall, ikke som prosent. Ved bruk av vektor vil PCTSUM veies, PCTN derimot veies ikke. Ønsker vi prosentene ut på vanlig måte, kan vi avlede dem med en beregnet variabel:

```
PROC REPORT DATA=b_97 HEADLINE HEADSKIP NOWINDOWS MISSING;
  COLUMN still3 N lonnialt, (MEAN SUM PCTSUM) prosent;
  DEFINE still3 / GROUP 'Stillingsgruppe' F=$still. SPACING=0;
  DEFINE lonnialt / ANALYSIS F=10. '';
  DEFINE MEAN / F=8. 'Gjennom-/snittlig/lønn';
  DEFINE SUM / F=8. 'Lønn/totalt';
  DEFINE PCTSUM / F=8.2 'Prosent av SUM' NOPRINT;
  DEFINE prosent / COMPUTED F=8.1 'Prosent av SUM';
  COMPUTE prosent;
    prosent = lonnialt.pctsum*100;
  ENDCOMP;
  RBREAK BEFORE / SUMMARIZE SKIP;
  TITLE 'Lønn etter stilling og kjønn. 1998';
RUN;
```

Resultat:

Lønn etter stilling og kjønn. 1998

Stillingsgruppe	N	Gjennom- snittlig lønn	Lønn totalt	Prosent av SUM
	17	23844	381500	100.0
Avdelingssjefer	3	27500	55000	14.4
Filialsjefer/bestyrere	5	23200	116000	30.4
IT-sjefer	3	26500	79500	20.8
Kontorsjefer	2	22000	44000	11.5
Soussjefer	4	21750	87000	22.8

Vi bruker komma for å knytte variable og statistikker sammen. Vi kan f. eks. ha en grupperingsvariabel i hodet (defineres med ACROSS i DEFINE-setningen) som knyttes sammen med en analysevariabel (defineres med ANALYSIS i DEFINE-setningen) som igjen kan knyttes sammen med en eller flere statistikker (f. eks. MEAN og SUM). Vi kan ikke knytte statistikker til mer enn et nivå i disse definisjonene.

Eksempel:

```
PROC REPORT DATA=b_97 HEADLINE HEADSKIP NOWINDOWS;  
  COLUMN still3 lønnialt, (N MEAN SUM) ;  
  DEFINE still3 / GROUP 'Stillingsgruppe' F=$still. SPACING=0;  
  DEFINE lønnialt/ ANALYSIS F=10. '' ;  
  DEFINE N/ F=8. 'Ansatte/med i/under-/søkelsen';  
  DEFINE MEAN/ F=8. 'Gjennom-/snittlig/lønn';  
  DEFINE SUM/ F=8. 'Lønn/totalt';  
  RBREAK BEFORE/SUMMARIZE SKIP;  
  TITLE 'Lønn etter stilling. 1998';  
RUN;
```

Utskrift:

Lønn etter stilling. 1998

Stillingsgruppe	Ansatte med i under- søkelsen	Gjennom- snittlig lønn	Lønn totalt
	16	23844	381500
Avdelingssjefer	2	27500	55000
Filialsjefer/bestyrere	5	23200	116000
IT-sjefer	3	26500	79500
Kontorsjefer	2	22000	44000
Soussjefer	4	21750	87000

Vi kan også skrive overskrifter i kolonnedefinisjonen. Det gir oss blant annet muligheten til å få en overskrift over flere kolonner (variable) og heltrukne linjer over flere kolonner (se eksempelet under).

Syntaks for overskrifter:

('overskrift-1' <... 'Overskrift-n'> kolonnespesifikasjoner)

Denne syntaksen brukes for å få felles overskrifter over kolonner som skal ha dette, men som ikke vil få det automatisk.

Hvis første og siste tegn i overskriften er -, =, _, ., *, or +, brukes det tegnet til å fylle ut kolonneoverskriften på hver side av teksten. Tilsvarende hvis vi setter teksten inne i <> vil det fylles ut med < foran og > etter teksten. Dette kan vi bruke til å få streket under tekst i kolonneoverskriftene og også til å få en streklinje etter overskriften.

Eksempel:

```
PROC REPORT DATA=b_97 HEADLINE HEADSKIP NOWINDOWS;
  COLUMN ('--' still3 n ('Lønn' '--' lonnialt fastlonn bon til ot));
  DEFINE still3 / GROUP 'Stillingsgruppe' F=$still. SPACING=0;
  DEFINE N/ F=8. 'Ansatte/med i/under-/søkelsen';
  DEFINE lonnialt/ ANALYSIS MEAN F=10. 'I alt';
  DEFINE fastlonn / ANALYSIS MEAN F=10. 'Avtalt lønn';
  DEFINE bon / ANALYSIS MEAN F=9. 'Bonus, prov. mv';
  DEFINE til / ANALYSIS MEAN F=9. 'Uregel-/messige tillegg';
  DEFINE ot / ANALYSIS MEAN F=9. 'Overtids-/godtgj.';

  RBREAK BEFORE / SUMMARIZE SKIP ;
  COMPUTE AFTER;
  LINE @01 91*'-';
  ENDCOMP;

  TITLE 'Lønn og tillegg etter stilling. 1998';

RUN;
```

Resultat:

Lønn fordelt etter stilling. 1998

Stillingsgruppe	Ansatte med i under- søkelsen	Lønn				
		I alt	Avtalt lønn	Bonus, prov. mv	Uregel- messige tillegg	Overtids- godtgj.
	16	23844	22625	678	278	263
Avdelingssjefer	2	27500	26000	750	500	250
Filialsjefer/bestyrere	5	23200	21800	1060	160	180
IT-sjefer	3	26500	25000	417	417	667
Kontorsjefer	2	22000	20750	850	300	100
Soussjefer	4	21750	21125	275	200	150

Ønsker vi å bruke en variabel eller statistikk på flere måter i en og samme rapport, kan vi gi den et alias:

```
variabel = alias
statistikk = alias
```

Samme variabel eller statistikk kan gis flere alias, men hver variabel og alias kan bare brukes i en DEFINE-setning hver.

eksempel.:

```
PROC FORMAT ;
  VALUE innt
    1-<10000 = 'Under 10000'
    10000-<20000 = '10000 - 20000'
    20000-<25000 = '20000 - 25000'
    25000-<30000 = '25000 - 30000'
    30000- high = '25000 - 30000'
  OTHER = 'Uoppgitt inntekt'
;

PROC REPORT DATA=b_97 HEADLINE HEADSKIP NOWINDOWS MISSING;
  COLUMN ('--' still3 N lonnialt lonnialt=lonna,lonnialt=lonnb);
  DEFINE still3 / GROUP 'Stillingsgruppe' F=$still. SPACING=0;
  DEFINE lonnialt / ANALYSIS MEAN F=10. 'Inntekt';
  DEFINE lonna / ANALYSIS MEAN F=10. 'Inntektsgrupper';
  DEFINE lonnb / ACROSS ' ' F=innt. WIDTH=8 '--';
  RBREAK BEFORE / SUMMARIZE SKIP;
  TITLE 'Gjennomsnittlig månedsinntekt etter stilling og kjønn. 1998';

RUN;
```

Gjennomsnittlig månedsinntekt etter stilling og kjønn. 1998

Stillingsgruppe	N	Inntekt	Inntektsgrupper			
			10000 - 20000	20000 - 25000	25000 - 30000	Uoppgitt inntekt
	17	23844	19000	22750	27000	-
Avdelingssjefer	3	27500	-	-	27500	-
Filialsjefer/bestyrere	5	23200	-	22750	25000	-
IT-sjefer	3	26500	-	24500	27500	-
Kontorsjefer	2	22000	-	22000	-	-
Soussjefer	4	21750	19000	22667	-	-

DEFINE

Brukes til å definere kolonnene rapporten skal bestå av. Vi trenger en DEFINE-setning pr. del av rapporten, vanligvis tilsvarende hver del én kolonne. Hver variabel må angis med en av disse options for bruksmåte:

- ACROSS
- Variabelen grupperes i kolonner. Det lages en kolonne for hver verdi variabelen har, eller en kolonne for hver formaterte verdi (kodetekst). Når vi bruker formater plasseres kolonnene stigende etter formatenes verdi. Ellers plasseres kolonnene stigende etter variabelens verdi. Dette kan overstyres med ORDER-option og/eller DESCENDING-option i DEFINE-setningen. ACROSS tilsvarende å bruke en CLASS-variabel i hodet i PROC TABULATE.
- ANALYSIS
- Variabelen må være numerisk og skal brukes til en statistikk, f. eks SUM el. MEAN. Hvis vi ikke angir noen statistikk i DEFINE-setningen, brukes SUM.
- COMPUTED
- Definerer en hjelpevariabel som ikke finnes på SAS datasettet rapporten lages fra, men som skal beregnes i rapportprosedyren. Variabelens plassering angis i COLUMN-setningen og den defineres i DEFINE-setningen, mens den gis verdi i en COMPUTE-blokk, se COMPUTE-setningen.
- DISPLAY
- Variable definert med DISPLAY, vil bli listet ut for hver observasjon i SAS-datasettet, disse får altså ikke gruppert like verdier.
- GROUP
- Variabelen grupperes slik at like variabelverdier grupperes til hver sin linje i rapporten. Når vi bruker formater plasseres kolonnene stigende etter formatenes verdi. Ellers plasseres kolonnene stigende etter variabelens verdi. Dette kan overstyres med ORDER-option og/eller DESCENDING-option i DEFINE-setningen. GROUP tilsvarende å bruke en grupperingsvariabel (definert i en CLASS-setning) i forspalten i PROC TABULATE..
- ORDER
- Variable definert med ORDER listes ut i sortert rekkefølge for hver observasjon i SAS datasettet. Sorteringsrekkefølgen er etter formaterte verdier når vi knytter et format til variabelen, ellers sorteres de etter variabelens verdi. Dette kan overstyres med ORDER-option og/eller DESCENDING-option i DEFINE-setningen.

Vi trenger ikke å angi bruksmåte når vi definerer en statistikkberegning i DEFINE-setningen, men kan gjerne gjøre det for oversiktens skyld.

Vi kan også angi disse options i DEFINE-setningen:

- CENTER
- Midtstiller kolonneoverskriften og kolonnens innhold. Tall justeres allikevel slik at de får rett høyremarg
- COLOR=
- angir kolonneoverskriftens farge. Virker bare når WINDOWS-option er satt i PROC REPORT-setningen. Disse fargene kan brukes: BLACK, BLUE, BROWN, CYAN, GRAY, GREEN, MAGENTA, ORANGE, PINK, RED, WHITE og YELLOW.
- DESCENDING
- Angir at variabel med bruksmåte ACROSS, DISPLAY eller GROUP skal sorteres synkende
- FLOW
- Gjør at tekst i kolonnen deles over flere linjer hvis den ikke får plass på avsatt område. Andre kolonnens verdier skrives ut på første linje med tekst. Teksten deles hvis mulig ved delingstegn eller blank.

FORMAT=	Angir hvordan kolonnen skal formateres. Vi kan bruke egendefinerte formater SAS egne formater. Standard er BESTw. for numeriske variable og \$w. for karaktvariable hvor w er standard kolonnebredde.
ID	Gjør at kolonnen og alle kolonner til venstre for ID-variabelen gjentas hvis rapporten må deles fordi den er for bred for siden. Mest aktuelt for variable i forspalten.
kolonneoverskrift	Teksten som skrives i fnutter vil bli kolonnens overskrift. Teksten brytes hvis den ikke får plass i kolonnebredden. For å sikre riktig tekstbrudd kan vi bruke / eller et annen definert tekstdelingstegn i teksten.
LEFT	Venstrejusterer kolonneoverskriften og kolonnens innhold. Tall justeres allikevel slik at de får rett høyremarg.
NOPRINT	Gjør at kolonnen ikke skrives ut. Overstyres av SHOWALL-option i PROC REPORT-setningen
NOZERO	Undertrykker kolonneutskrift når verdien er 0 eller MISSING. Overstyres av SHOWALL-option i PROC REPORT-setningen
ORDER=	Angir hvilken rekkefølge verdiene til variable med bruksmåte ACROSS, DISPLAY og GROUP skal skrives ut i. Velger vi ORDER=FORMATTED skrives de ut sortert etter teksten i utskriftsformatet som eventuelt brukes istedenfor koden. Dette er standard når vi bruker formater på variabelen. ORDER=FREQ sørger for at den verdi med færrest observasjoner skrives først (med mindre vi også bruker DESCENDING-option også brukes, da kommer den med flest observasjoner kommer først). ORDER=DATA gjør at verdiene beholder den rekkefølgen de har i datasettet, den første forekomsten av verdiene bestemmer rekkefølgen. Standardverdi når vi ikke bruker formater er ORDER=INTERNAL som skriver ut verdiene i sortert rekkefølge. MISSING-verdier vil alltid komme ut først, uansett hvilken rekkefølge vi velger.
PAGE	Gir sideskift før kolonnen skrives ut
RIGHT	Høyrejusterer kolonneoverskriften og kolonnens innhold.
SPACING=	Angir hvor mange blanke posisjoner som skal skrives ut før kolonnen. Standard er 2.
statistikk	Angir hvilken statistikkberegning vi skal utføre på variable med ANALYSIS som bruksmåte, se COLUMN-setningen for oversikt over statistikker.
STYLE=	Angir stilspesifiseringer til html-utskrift, se side 266.
WIDTH=	Angir kolonnebredde
BREAK	BREAK-setningen bruker vi for å definere brudd i forspalten i rapporten. Vi kan ha variable definert med GROUP eller ORDER i DEFINE-setninger som bruddvariable. Vanligvis vil vi lage summer for bruddene, selv om vi kan ha brudd uten å summere også. I BREAK-setningen angir vi først om bruddlinjen skal plasseres før (BEFORE) eller etter (AFTER) at underliggende nivåer skrives ut. Deretter angir vi hvilken variabel bruddet skal skje ved, en variabel pr. BREAK-setning. Til slutt kan vi angi følgende options til BREAK-variabelen: COLOR=farge Angir en farge for bruddlinjen, alle bokstaver og tall på linjen får den valgte fargen. Virker bare når option WINDOWS er satt i PROC REPORT-setningen. Følgende farger er gyldige: BLACK, BLUE, BROWN, CYAN, GRAY, GREEN, MAGENTA, ORANGE, PINK, RED, WHITE og YELLOW. DOL Legger en linje med doble streker før summeringen. DUL Legger en linje med doble streker etter summeringen. OL Legger en linje med enkle streker før summeringen.

PAGE

Gir sideskift ved brudd. Kan f. eks. brukes til å lage en rapport pr. grupperingsvariabel. Eksempel som gir en rapport pr. kjønn:

```
PROC REPORT DATA=b_97 HEADLINE HEADSKIP NOWINDOWS MISSING;
  COLUMN  kjønn still3 N lonnialt;
  DEFINE kjønn    / GROUP 'Kjønn' F=$kjønn. SPACING=0;
  DEFINE still3   / GROUP 'Stillingsgruppe' F=$still.;
  DEFINE N        / 'Antall';
  DEFINE lonnialt / ACROSS F=innt. '' WIDTH=8;
  BREAK BEFORE  kjønn / SUMMARIZE SKIP ;
  BREAK AFTER   kjønn / PAGE;
  TITLE 'Antall i forskjellige inntektsgrupper etter kjønn og stilling. 1998';
RUN;
```

SUMMARIZE

Sørger for at vi får laget summer ved brudd i verdiene i forspalten.

SUPPRESS

Undertrykker verdien til bruddvariabelen (forspaltevariabelen) i utskriften.

SKIP

Gir linjeskift etter brudd. Nyttig for å få inn blanke linjer i rapporten.

STYLE

Til stilspesifikasjoner for html-utskrift. Se html-stiler, side 266.

UL

Legger en linje med enkle streker etter summeringen.

RBREAK

RBREAK-setningen bruker vi for å angi at vi skal ha med totalsummer, enten øverst (BEFORE) eller nederst (AFTER) i rapporten. Vi kan bruke de samme OPTIONS til RBREAK som til BREAK, unntatt PAGE og SUPPRESS, se over.

COMPUTE

Brukes til å utføre beregninger på rapporten. Kan brukes til å avlede nye kolonner fra eksisterende, skifte ut variabelverdier med tekst og utføre forskjellige beregninger. I beregningene i COMPUTE-setningen kan vi ikke bruke variable som er definert lenger til høyre i tabellen enn COMPUTE-variabelen. Vi kan også bruke COMPUTE til å gjøre spesielle ting ved brudd i rapporten, se LINE-setningen, og til å tilpasse utseende på deler av rapporten, se CALL DEFINE. Inne i en COMPUTE-blokk kan vi utføre mange av de vanlige SAS-setninger som vi ellers bruker i DATA-steget:

assignment (tilordning), CALL, DM, DO, END, GO TO, IF-THEN/ELSE, %INCLUDE, kommentar, LENGTH, LINK, RETURN, SELECT og sum, for beskrivelse av disse se kapittelet om DATA-steget. I tillegg til disse setningene, kan vi bruke CALL DEFINE- og LINE- setningene i COMPUTE-blokker. Vi kan også bruke alle SAS funksjoner, makro-variable og kalle opp makroer i COMPUTE-blokker.

Hver COMPUTE-blokk skal avsluttes med en ENDCOMP-setning.

Vi har to varianter av COMPUTE-setningen, en for å gjøre beregninger ved brudd, og en for andre beregninger. Til beregninger ved brudd bruker vi LINE-setningen for å skrive ut egne linjer i forbindelse med bruddet, se LINE-setningen. I COMPUTE-setningen angir vi da om linjene skal plasseres før (BEFORE) eller etter (AFTER) bruddet. Vi kan ha med en brudd-variabel i COMPUTE-setningen, i så fall skrives linjene ut for hvert brudd denne variabelen får. Har vi ikke med noen bruddvariabel, får vi skrevet ut de ønskede linjer enten øverst eller nederst i tabellen. Vi kan også få skrevet ut en tekst først (dvs. etter overskriften) eller sist (dvs. før fotnoter) på hver side ved å bruke `_PAGE_`. Teksten vi skriver ut kan vi da plassere til venstre (LEFT), i midten (CENTER) eller til høyre (RIGHT), eller vi kan bestemme plassering i LINE-setningen der vi angir teksten som skal skrives ut.

For beregninger som gjøres uavhengig av brudd i forspalten, kan vi definere dette i COMPUTE-setningen:

report-variabel angir navnet på den hjelpevariabelen som skal gis verdi. Denne må være nevnt i COLUMN-setningen og definert som COMPUTED i DEFINE-setningen.
 CHARACTER definerer en karaktervariabel

LENGTH= karaktervariabelens lengde
STYLE= layout-stil for html-filer, se html-stiler side 266.

Når vi henviser til analysevariable for å gjøre beregninger på dem, bruker vi denne formen:

variabel.statistikk

Eksempel:

```
PROC REPORT DATA=uriks HEADLINE HEADSKIP NOWINDOWS;  
  COLUMN sitc2 msum vsum pris ;  
  DEFINE sitc2 / GROUP 'Sitc' SPACING=0 ;  
  DEFINE aar / ACROSS '' WIDTH=8;  
  DEFINE msum / ANALYSIS SUM 'Mengde' F=8. ;  
  DEFINE vsum / ANALYSIS SUM 'Verdi' F=8. ;  
  DEFINE pris / COMPUTED 'Pris' F=8.2;  
  
  RBREAK BEFORE / SUMMARIZE SKIP;  
  
  COMPUTE pris;  
    pris = vsum.sum/mlsum.sum;  
  ENDCOMP;  
  FORMAT sitc2 $sitc2f. ;  
  TITLE 'Innførsel av viktige varer';  
RUN;
```

Resultat:

Innførsel av viktige varer			
Sitc	Mengde	Verdi	Pris

	6003	36747	6.12
01 Kjøtt og kjøttvarer	22	524	23.80
02 Meierivarer og egg	14	429	30.61
03 Fisk, krepsdyr og bløtdyr	964	7614	7.90
04 Korn og kornvarer	1820	5297	2.91
05 Grønnsaker og frukt	1578	11410	7.23
06 Sukker, sukkervarer og honning	826	2866	3.47
07 Kaffe, te, kakao, krydderier	211	4898	23.21
08 Dyrefor (unntatt umalt korn)	513	2188	4.27
09 Forskjellige matvarer	55	1522	27.67

For kolonner laget fra variable definert som ACROSS i DEFINE-setningen, må vi bruke kolonnennummeret som referanse: `_Cn_`, der n er kolonnennummeret. Kolonne `_C1_` er lengst til venstre. Hvis antallet forskjellige verdier for ACROSS-variable endres fra en kjøring til en annen, må vi passe på å sjekke at kolonnennumrene det refereres til er de riktige.

Eksempel:

```
PROC REPORT DATA=uriks HEADLINE HEADSKIP NOWINDOWS;  
  COLUMN sitc2 msum, ('--' aar) vsum, ('--' aar) ('Pris' '--' '' pris1 pris2) ;  
  DEFINE sitc2 / GROUP 'Sitc' SPACING=0 ;  
  DEFINE SUM / '--';  
  DEFINE aar / ACROSS '' WIDTH=8;  
  DEFINE msum / ANALYSIS SUM 'Mengde' F=8. ;  
  DEFINE vsum / ANALYSIS SUM 'Verdi' F=8. ;  
  DEFINE pris1 / COMPUTED '1995' F=8.2;  
  DEFINE pris2 / COMPUTED '1996' F=8.2;  
  
  RBREAK BEFORE / SUMMARIZE SKIP;  
  
  COMPUTE pris1;  
    pris1 = _C4_/_C2_;  
  ENDCOMP;  
  COMPUTE pris2;  
    pris2 = _C5_/_C3_;  
  ENDCOMP;  
  FORMAT sitc2 $sitc2f. ;  
  TITLE 'Innførsel av viktige varer';  
RUN;
```

Resultat:

Sittc	Mengde		Verdi		Pris	
	1995	1996	1995	1996	1995	1996
	-----	-----	-----	-----	-----	-----
	2742	3261	17572	19175	6.41	5.88
01 Kjøtt og kjøttvarer	11	11	248	275	22.58	25.01
02 Meierivarer og egg	7	7	207	222	29.54	31.69
03 Fisk, krepsdyr og bløtdyr	435	529	3547	4066	8.15	7.69
04 Korn og kornvarer	753	1067	2229	3068	2.96	2.88
05 Grønnsaker og frukt	763	815	5485	5925	7.19	7.27
06 Sukker, sukkervarer og honning	402	424	1392	1474	3.46	3.48
07 Kaffe, te, kakao, krydderier	101	110	2555	2344	25.29	21.31
08 Dyrefor (unntatt umalt korn)	242	271	1097	1091	4.53	4.03
09 Forskjellige matvarer	28	27	811	710	28.98	26.30

Vi kan også bruke COMPUTE-setningen for å legge tekster inn i en kolonne. Det kan vi f. eks. bruke for å trekke informasjon fra forskjellige kolonner og ut fra hvilke verdier disse har plassere tekst i en ny kolonne. Dette er spesielt nyttig når vi har to eller flere nivåer i forspalten (flere variable definert med GROUP i DEFINE-setningen), men ønsker å skrive nivåene ut i en felles kolonne. PROC REPORT er som kjent kolonnestyrt, dvs at vi arbeider med kolonner. Når vi har flere nivåer i forspalten, vil hvert nivå havne i hver sin kolonne. Dette fører til en layout som ikke passer med tabellene vi publiserer. Ved å definere en ny kolonne som COMPUTED kan vi få til en bedre layout, i hvert fall for noen tabelltyper. Ut fra de kolonnene som inneholder forspalten, bygger vi opp den nye kolonnens innhold. Vi bruker i eksempelet under kolonnene til to forspaltenivåer (kjonn og still3) for å lage den forspalten (forsp) som skrives ut. De opprinnelige forspaltekolonnene skrives ikke ut (de bør skrives ut mens vi bygger opp den forspaltekolonnen slik at vi kan få oversikt over hvordan den nye forspaltekolonnen skal lages).

I COMPUTE-setningen tester vi hva som er i de forspaltekolonnene som ikke skrives ut, og ut fra det henter vi riktige tekster med PUT-funksjonen. I SELECT-setningen tester vi først om still3 er ulik blank. Da vet vi at vi behandler en linje på stillingstypenivå og kan hente teksten til den aktuelle stillingstype. Neste test er om kjonn er ulik blank, hvilket forteller oss at nå behandler vi en linje på kjønnsnivå. Kjønnsstekst hentes ut fra koden. Til slutt er det bare totallinjen igjen og den gis tekst. Hvorfor testene blir slik, kan vi se ved å skrive ut kolonnene kjonn og still3.

```
DATA b_97;
  INPUT @01 fnr          $CHAR11.
        @12 kjonn       $CHAR1.
        @14 still3      $CHAR3.
        @17 lonnialt    6.
        @24 fastlonn    6.
        @31 bon         6.
        @38 til         6.
        @45 ot          6.
        ;
CARDS;
111111111111 123 24000 23000 300 200 500
222222222222 124 21000 20500 500 0 0
333333333331 125 23000 22500 100 100 300
444444444442 125 21000 20500 200 200 100
444344444442 123 19000 18000 800 100 100
444344445442 124 23000 21000 1200 600 200
555555555552 125 24000 23000 600 200 200
556555555651 125 25000 23000 1400 300 300
556555555652 125 23000 20000 3000 0 0
556555556652 123 23000 23000 0 0 0
556555458152 123 21000 20500 0 500 0
556554555151 127 28000 26500 1000 500 0
556545555151 127 27000 25500 500 500 500
556455555151 128 27500 25500 500 500 1000
556555535252 128 27500 25500 500 500 1000
556555535252 128 24500 24000 250 250 0
;
```

```
PROC FORMAT ;
VALUE $kjonn
  '1' = 'Menn'
  '2' = 'Kvinner'
  OTHER = 'Uoppgitt'
;
VALUE $still
  '123' = 'Soussjefer'
  '124' = 'Kontorsjefer'
  '125' = 'Filialsjefer/bestyrere'
  '126' = 'Banksjefer ved filial'
  '127' = 'Avdelingssjefer'
  '128' = 'IT-sjefer'
  OTHER = 'Ikke oppgitt'
;
PROC REPORT DATA=b_97 HEADLINE HEADSKIP NOWINDOWS;
COLUMN kjonn still3 forsp lonnialt=ans lonnialt fastlonn bon til ot;
DEFINE kjonn / GROUP NOPRINT;
DEFINE still3 / GROUP NOPRINT;
DEFINE forsp / COMPUTED SPACING=0 'Kjønn og stillingstype';
DEFINE ans / ANALYSIS N F=ssb6_0v. 'Ansatte med i/under-/søkelsen';
DEFINE lonnialt / ANALYSIS MEAN F=ssb5_0v. 'I alt';
DEFINE fastlonn / ANALYSIS MEAN F=ssb6_0v. 'Avtalt lønn';
DEFINE bon / ANALYSIS MEAN F=ssb7_0v. 'Bonus, prov. mv';
DEFINE til / ANALYSIS MEAN F=ssb7_0v. 'Uregel-/messige tillegg';
DEFINE ot / ANALYSIS MEAN F=ssb7_0v. 'Overtids-/godtgj.';

RBREAK BEFORE / SUMMARIZE SKIP ;
BREAK BEFORE kjonn / SUMMARIZE ;
BREAK AFTER kjonn / SKIP ;

COMPUTE forsp /CHARACTER LENGTH=25;
SELECT ;
  WHEN (still3 NE '') forsp = ' ' || PUT(still3,$still.);
  WHEN (kjonn NE '') forsp = PUT(kjonn,$kjonn.);
  OTHERWISE forsp = 'Menn og kvinner';
END;
ENDCOMP;
COMPUTE AFTER;
  LINE @01 90*'-' ;
ENDCOMP;

TITLE 'Lønn';
TITLE2 '-----';

RUN;
```

Resultat:

Lønn							

Kjønn og stillingstype	Ansatte med i under-søkelsen	I alt	Avtalt lønn	Bonus, prov. mv	Uregel-messige tillegg	Overtids-godtgj.	

Menn og kvinner	16	23 844	22 625	678	278	263	
Menn	6	25 750	24 333	633	350	433	
Soussjefer	1	24 000	23 000	300	200	500	
Filialsjefer/bestyrere	2	24 000	22 750	750	200	300	
Avdelingssjefer	2	27 500	26 000	750	500	250	
IT-sjefer	1	27 500	25 500	500	500	1 000	
Kvinner	10	22 700	21 600	705	235	160	
Soussjefer	3	21 000	20 500	267	200	33	
Kontorsjefer	2	22 000	20 750	850	300	100	
Filialsjefer/bestyrere	3	22 667	21 167	1 267	133	100	
IT-sjefer	2	26 000	24 750	375	375	500	

Ønsker vi også å fordele sumtallene for menn og kvinner på stillingstype, kan vi få til det ved å aggregere dataene med PROC MEANS først, for deretter å bruke PROC REPORT til å skrive ut tabellen. I eksempelet under har vi også lagt på utfyllende punkter på teksten:

```
PROC MEANS DATA=b_97 NOPRINT;
  CLASS kjonn still3;
  VAR lonnialt fastlonn bon til ot;
  OUTPUT OUT=b_97s MEAN=;
RUN;

* Bruker REPORT til å skrive ut tallene, ingen aggregering gjøres her;
PROC REPORT DATA=b_97s HEADLINE HEADSKIP NOWINDOWS MISSING;
  COLUMN kjonn still3 forsp_FREQ lonnialt fastlonn bon til ot;
  DEFINE kjonn / GROUP NOPRINT;
  DEFINE still3 / GROUP NOPRINT;
  DEFINE forsp / COMPUTED SPACING=0 'Kjønn og stillingstype';
  DEFINE_FREQ / ANALYSIS SUM F=ssb6_0v. 'Ansatte med i/under-/søkelsen';
  DEFINE lonnialt / ANALYSIS MEAN F=ssb5_0v. 'I alt';
  DEFINE fastlonn / ANALYSIS MEAN F=ssb6_0v. 'Avtalt lønn';
  DEFINE bon / ANALYSIS MEAN F=ssb7_0v. 'Bonus, prov. mv';
  DEFINE til / ANALYSIS MEAN F=ssb7_0v. 'Uregel-/messige tillegg';
  DEFINE ot / ANALYSIS MEAN F=ssb7_0v. 'Overtids-/godtgj.';

  BREAK AFTER kjonn / SKIP ;

  COMPUTE forsp /CHARACTER LENGTH=25;
  prikker = ' .....';
  SELECT ;
    WHEN (still3 NE '') forsp = ' ' || TRIM(PUT(still3,$still.)) || prikker ;
    WHEN (kjonn NE '') forsp = TRIM(PUT(kjonn,$kjonn.)) || prikker ;
    OTHERWISE forsp = 'Menn og kvinner' || prikker ;
  END;
ENDCOMP;
COMPUTE AFTER;
  LINE @01 90*'-';
ENDCOMP;

  TITLE 'Lønn';
  TITLE2 '-----';

RUN;
```

Tabellen:

Lønn						

Kjønn og stillingstype	Ansatte med i under-søkelsen	I alt	Avtalt lønn	Bonus, prov. mv	Uregel-messige tillegg	Overtids-godtgj.

Menn og kvinner	16	23 844	22 625	678	278	263
Soussjefer	4	21 750	21 125	275	200	150
Kontorsjefer	2	22 000	20 750	850	300	100
Filialsjefer/bestyrere	5	23 200	21 800	1 060	160	180
Avdelingssjefer	2	27 500	26 000	750	500	250
IT-sjefer	3	26 500	25 000	417	417	667
Menn	6	25 750	24 333	633	350	433
Soussjefer	1	24 000	23 000	300	200	500
Filialsjefer/bestyrere	2	24 000	22 750	750	200	300
Avdelingssjefer	2	27 500	26 000	750	500	250
IT-sjefer	1	27 500	25 500	500	500	1 000
Kvinner	10	22 700	21 600	705	235	160
Soussjefer	3	21 000	20 500	267	200	33
Kontorsjefer	2	22 000	20 750	850	300	100
Filialsjefer/bestyrere	3	22 667	21 167	1 267	133	100
IT-sjefer	2	26 000	24 750	375	375	500

Vi kan også bruke STYLE-option til å angi stilspesifikasjoner til html-utskrift. Se html-stiler, side 266.

CALL DEFINE

Brukes for å angi hvordan en spesiell celle i rapporten skal se ut. CALL DEFINE må stå inne i en COMPUTE-blokk. Vi kan bare endre attributter på kolonner definert som COMPUTED i DEFINE-setningen. COLOR-attributten virker bare når WINDOWS-option er satt i PROC REPORT-setningen. Aktuelle attributter:

color angir en farge innholdet i cellen skal få. Gyldige verdier er BLACK, BLUE, BROWN, CYAN, GRAY, GREEN, MAGENTA, ORANGE, PINK, RED, WHITE og YELLOW.

format angir formatet innholdet i cellen skal benytte. Dette må stå i fnutter.

style= stiler for html-utskrift

url brukes for å lage linker til andre filer (URL=Uniform Resource Locator).

urlbp brukes for å lage linker til andre filer. Begynnelsen av linken skal være definert i ODS med BASE- eller PATH-option.

urlp brukes for å lage linker til andre filer. Begynnelsen av linken skal være definert i ODS med PATH-option.

Eksempel:

```
PROC REPORT DATA=b_97 HEADLINE HEADSKIP WINDOWS MISSING;
COLUMN ('--' kjonn still3 N lonnialt lonns);
DEFINE still3 / GROUP 'Stillingsgruppe' F=$still. SPACING=0;
DEFINE kjonn / GROUP 'Kjønn' F=$kjonn. ;
DEFINE lonnialt / ANALYSIS MEAN F=10. NOPRINT;
DEFINE lonns / COMPUTED 'Inntekt' WIDTH=8 F=8.;
RBREAK BEFORE / SUMMARIZE SKIP ;
BREAK BEFORE kjonn / SUMMARIZE SKIP ;

COMPUTE lonns;
lonns = lonnialt.mean;
IF kjonn = '' AND still3 = '' THEN
lonngs = lonnialt.mean;
SELECT;
WHEN (0 < lonnialt.mean/lonngs < 1.00)
CALL DEFINE (_COL_,'color','red');
WHEN (0.95 <= lonnialt.mean/lonngs < 1.05)
CALL DEFINE (_COL_,'color','yellow');
WHEN (1.05 <= lonnialt.mean/lonngs )
CALL DEFINE (_COL_,'color','green');
OTHERWISE;
END;
ENDCOMP;
```

LINE

Angir en linje som skal skrives ut i rapporten. LINE-setningen må plasseres inne i en COMPUTE-blokk og kan bare brukes på spesielle steder. Den må ha en forbindelse til et brudd i forspalten definert med BREAK- eller RBREAK-setninger. LINE-setningen kan ikke brukes i forbindelse med IF, SELECT eller DO-setninger, den utføres uansett sist i COMPUTE-blokken uavhengig av slike tester.

LINE-setningen fungerer omtrent som en PUT-setning i DATA-steget (se side82), men har ikke så mange muligheter som denne. Følgende kan angis i LINE-setningen:

peker Angir pekertegn og posisjon for utskrift på linjen. Pekertegn kan være @ eller + som etterfølges av et tall. Tallet etter @ angir posisjon på linja, mens det etter + angir at vi skal forflytte oss et antall posisjoner på linjen før utskrift.

variabel Hvordan variabelen (egentlig kolonnen) skal refereres avhenger av hvordan den er definert i DEFINE-setningen. Er den definert som GROUP, ORDER, COMPUTED eller DISPLAY, skal referansen være variabelnavnet. Variable definert med ANALYSIS skal refereres med variabelnavn.statistikk (f. eks inntekt.sum). Kolonner som inngår i ACROSS-definisjoner må refereres med kolonnennummeret (f. eks _C13_).

format Formatet kan være egendefinert eller definert av SAS.

'tekst' Teksten vil komme ut på linjen slik den står i fnuttene. Er teksten blank, kommer det ut en blank linje i rapporten.

n*'tekst' Teksten i fnutter vil gjentas n ganger på linjen (brukes f. eks. for å få en linje med streker)

ENDCOMP

Avslutter en COMPUTE-blokk.

FORMAT

Vi bruker FORMAT-setningen til å erstatte variabelverdier med mer forståelig *tekst*. Disse utskriftsformatene må være definert på forhand vha. PROC FORMAT, se side 133. FORMAT-setningen kan også brukes til å *gruppere* variabelverdier, f. eks. etter aldersgrupper eller inntektsgrupper.

WEIGHT

For å veie dataene bruker vi WEIGHT-setningen. Vi må da ha en variabel på datasettet som inneholder vekten (oppblåsningsfaktoren). Vektvariabelen kan gjerne være et desimaltall. Når vektens verdi er mindre enn 0, settes den til 0 og observasjonen telles med i antallet (N). Når verdien er 0 telles observasjonen med i antallet. Dette kan overstyres med PROC REPORT-option EXCLNPWGHTS som ekskluderer disse observasjonene fra tabellen. Er vektvariabelen MISSING, ekskluderes observasjonen fra tabellen.

FREQ

Hvis hver observasjon på datasettet skal telle for mer enn 1, kan vi bruke FREQ-setningen. I FREQ-setningen kan vi da angi variabelen hvis verdi inneholder det antall observasjoner denne observasjonen skal telle som. Dette betyr at N også økes med verdien til FREQ-variabelen.

FREQ trunkerer alle desimaler i FREQ-variabelen før den brukes. Er verdien på FREQ-variabelen 0 eller mindre, teller den ikke med i statistiske beregninger.

Eksempler:

Lister ut observasjoner der alle med samme familienr kommer samlet og hver familie skilles med en blank linje.

```
DATA fam;
  INPUT @01 famnr      $CHAR11.
        @13 fnr       $CHAR11.
        @24 kjonn     $CHAR1.
        @26 sivst     $CHAR1.
        @28 perskode  $CHAR1.
        ;
  LABEL
    famnr    = 'Familienr'
    fnr      = 'Fødselsnr'
    kjonn    = 'Kjønn'
    sivst    = 'Sivilstatus'
    perskode = 'Personkode'
    ;

CARDS;
01045455515 010454555151 7 1
10101555555 046555555652 5 3
10101555555 101015555552 5 1
11111111111 111111111111 3 1
11111111111 123333333331 5 3
12434444444 124344444442 3 1
10101555555 126555555651 5 2
10101555555 1365555556652 3 3
15434444454 1543444445442 4 1
11111111111 1744444444442 5 3
11111111111 221122121222 4 2
30655553525 226553535252 8 2
10101555555 276555458152 3 3
30655553525 306555535252 8 1
31645555515 316455555151 8 1
01045455515 556545555151 7 2
;

PROC REPORT DATA=fam HEADLINE HEADSKIP NOWINDOWS;
  COLUMN famnr fnr kjonn sivst perskode;
  DEFINE famnr / ORDER SPACING=0 ;
  DEFINE fnr / DISPLAY;
  DEFINE kjonn / DISPLAY WIDTH=7;
  DEFINE sivst / DISPLAY WIDTH=12;
  DEFINE perskode / DISPLAY WIDTH=12;

  BREAK AFTER famnr / SKIP ;

  TITLE 'Familier';

RUN;
```

Resultat:

Familier				
Familienr	Fødselsnr	Kjønn	Sivilstatus	Personkode
01045455515	01045455515	1	7	1
	55654555515	1	7	2
10101555555	04655555565	2	5	3
	10101555555	2	5	1
	12655555565	1	5	2
	13655555565	2	3	3
	27655545815	2	3	3
11111111111	11111111111	1	3	1
	12333333333	1	5	3
	17444444444	2	5	3
	22112212122	2	4	2
12434444444	12434444444	2	3	1
15434444544	15434444544	2	4	1
3065553525	22655353525	2	8	2
	3065553525	2	8	1
31645555515	31645555515	1	8	1

Ny rapport fra samme SAS datasett, denne gang grupperes sivilstatus i forspalten og kjønn i hodet i rapporten.

```
PROC REPORT DATA=fam HEADLINE HEADSKIP NOWINDOWS MISSING;
  COLUMN sivst kjønn ;
  DEFINE sivst / GROUP WIDTH=12;
  DEFINE kjønn / ACROSS WIDTH=7;

  TITLE 'Antall personer etter sivilstatus og kjønn';

RUN;
```

Utskrift:

Antall personer etter sivilstatus og kjønn

Sivilstatus	Kjønn	
	1	2
3	1	3
4	.	2
5	2	3
7	2	.
8	1	2

Rapport med to nivåer i forspalten med summer og totalsum samt totalkolonne og teksting av kjønn. Regner også ut prosentandel kvinner:

```
PROC FORMAT;
  VALUE $kjønn
    '1' = 'Menn'
    '2' = 'Kvinner'
    OTHER = 'Uoppgitt'
  ;

PROC REPORT DATA=fam HEADLINE HEADSKIP NOWINDOWS MISSING;
  COLUMN sivst perskode N kjønn proskv;
  DEFINE sivst / GROUP WIDTH=12 SPACING=0;
  DEFINE perskode / GROUP WIDTH=12;
  DEFINE kjønn / ACROSS WIDTH=7 ' ';
  DEFINE N / 'I alt' ;
  DEFINE proskv / COMPUTED 'Prosent kvinner' F=numx8.1 ;

  RBREAK BEFORE / SUMMARIZE SKIP;
  BREAK BEFORE sivst / SUMMARIZE ;
  BREAK AFTER sivst / SKIP;

  COMPUTE proskv;
    IF _C4_ > 0 THEN
      proskv = _C4_*100/_C3_;
    ELSE
      proskv = 0;
  ENDCOMP;

  FORMAT kjønn $kjønn.;
  TITLE 'Antall personer etter sivilstatus, personkode og kjønn';
RUN;
```


Rapporten:

Antall personer etter sivilstatus, personkode og kjønn					
Sivilstatus	Personkode	I alt	Kvinner	Menn	Prosent kvinner

		16	10	6	62,5
3		4	3	1	75,0
3		2	1	1	50,0
		2	2	.	100,0
4		2	2		100,0
4		1	1		100,0
		1	1		100,0
5		5	3	2	60,0
5	1	1	1	.	100,0
	2	1	.	1	0,0
	3	3	2	1	66,7
-		2		2	0,0
7	1	1		1	0,0
	2	1	.	1	0,0
8		3	2	1	66,7
8	1	2	1	1	50,0
	2	1	1	.	100,0

3.11.17 PROC SORT

Vi bruker Proc sort til å sortere datasett. Dataene sorteres stigende eller synkende, etter en eller flere sorteringsvariable. Sorteringen vil skje etter rekkefølgen tegnene har i det tegnsettet som brukes på maskinen. Operativsystemet UNIX på arbeidsstasjonene bruker ASCII tegnsystem, mens MVS på stormaskin bruker EBCDIC. Rekkefølgen kan derfor bli forskjellig fra en maskintype til en annen. Det betyr at et datasett som er sortert på stormaskin ikke nødvendigvis er riktig sortert når det blir overført til UNIX. Se vedlegg om sorteringsrekkefølger på side 272.

Når vi bruker SAS datasett options OBS=, KEEP= eller DROP= må vi passe på å lagre resultatet av sorteringen på et nytt datasett. Dette spesifiserer vi med option OUT= i PROC SORT-setningen.

PROC SORT <options> ;
WHERE where-uttrykk ;
BY <DESCENDING> variabel-1 ... <<DESCENDING> variabel-n> ;

Options til PROC SORT-setningen:

ASCII

Kan brukes på stormaskin for å sikre at et datasett som skal overføres til UNIX blir sortert riktig for UNIX

DATA= SAS datasett

Navnet på SAS datasettet som skal sorteres

DATECOPY

Angir at dato for når SAS datasettet var laget og sist endret ikke skal oppdateres ved sorteringen.

EDCDIC

Kan brukes på UNIX for å sikre at et datasett som skal overføres til stormaskin blir sortert riktig for stormaskinen.

EQUALS

EQUALS sørger for at rekkefølgen for observasjoner som har samme verdi for BY-variablene opprettholdes på utdatasettet. NOEQUALS gjør at denne rekkefølgen ikke behøver å bli beholdt. EQUALS er standard. Filer > 2 mrd. observasjoner kan ikke bruke EQUALS.

FINNISH

Sorterer dataene etter det finske (og svenske) alfabetet. Disse alfabetene ender med XYZÄÅÖ. Ser ikke ut til å virke etter hensikten (blir XYZÄÄÖ).

FORCE

Tvinger SAS til å erstatte et indeksert SAS datasett når OUT= option ikke brukes.

NATIONAL

Sorterer dataene etter nasjonale tegn når dette er satt i installasjonen. Det er ikke gjort hos oss.

NODUPKEY

Fjerner observasjoner med like verdier for BY-variablene (dubletter). Når flere observasjoner har samme verdier for BY-variablene, fjernes alle unntatt den første av disse. Denne option er grei for å bruke når vi skal sikre oss at vi får et datasett med unike verdier for BY-variablene, men den må brukes med varsomhet. Her slettes dubletter innen BY-variablene brutalt og ubønnhørlig! Når SAS har fjernet dubletter vil det komme en melding om dette i SASlog, men det er også all informasjon som gis om dublett fjerningen.

NODUPREC

Fjerner i gitte tilfeller observasjoner der alle variablene innbyrdes er like. Hva som regnes som alle variable er avhengig av system option SORTDUP. SORTDUP=PHYSICAL, som er standard, angir at sorteringen fjerner dubletter der alle variable er identiske på sorterte observasjoner som kommer etter hverandre. LOGICAL angir at bare variable angitt med KEEP- eller DROP-option må være identiske for at NODUPREC skal tre i kraft. Når flere observasjoner er identiske, fjernes alle unntatt den første av disse så lenge de observasjonene som er identiske kommer etter hverandre ved utskrift til det nye datasettet. Hvis det kommer en observasjon mellom to like observasjoner som er forskjellig fra disse, vil ikke SAS fjerne den som er identisk med den tidligere, fordi de identiske ikke kommer etter hverandre. Det betyr at vi ikke har noen garanti for at det ikke er identiske observasjoner på utdatasettet etter at vi har brukt NODUPREC-option. For å garantert fjerne alle identiske observasjoner, må vi sortere datasettet på alle variable og bruke option NODUPKEY eller vi kan bruke PROC SQL (se eksempel side 193).

NORWEGIAN

Sorterer dataene etter det norske (og danske) alfabetet. Ser ikke ut til å virke etter hensikten (blir XYZÅÆØ).

OUT=

Navnet på det sorterte SAS datasettet. Hvis OUT= utelates, erstatter den sorterte versjonen den usorterte.

REVERSE

Sorterer etter omvendt tegnrekkefølge. Kan kun brukes når karaktervariable sorteres.

SORTSEQ=

Sorterer etter en bestemt tegnrekkefølge. Mulige rekkefølger er ASCII, DANISH, EBCDIC, FINNISH, NATIONAL, NORWEGIAN og SWEDISH

SORTSIZE=

Angir maksimum minne som kan brukes i sorteringen. Standardverdien finnes i systemoption SORTSIZE.

MAX	Bruk maksimalt minne
n	Bruk n tegn (bytes) minne
nK	Bruk n kilobytes minne

TAGSORT

Kan brukes når vi har lite tilgjengelig diskplass. Når TAGSORT er spesifisert, lagres bare BY-variablene og observasjonsnumrene i midlertidige datasett. Deretter brukes disse variablene (tag-ene) til å hente inn observasjonene i sortert rekkefølge.

SAS-setninger til PROC SORT:

WHERE

Selekterer hvilke observasjoner som skal sorteres. Når vi bruker WHERE, må vi huske på å legge resultatet ut på et nytt SAS datasett. Bare de som tilfredsstiller WHERE-betingelsen tas med. Beskrivelse av WHERE-setningen finnes på side 91.

BY

Datasettet sorteres etter BY-variablene våre. Ønsker vi å sortere en variabel synkende istedenfor stigende skriver vi DESCENDING foran variabelen.

Eksempel:

```
PROC SORT DATA=bvmd87 OUT=bvern.bvmd87;
  BY fnr komm;
RUN;
```

3.11.18 PROC SQL

SQL (Structured Query Language) er opprinnelig et språk laget for spørring mot databaser. SAS-prosedyren SQL kan benyttes både på databasetabeller og på vanlige SAS datasett. Den bygger på vanlig SQL og syntaksen er svært lik vanlig SQL.

Notasjonen i SQL er annerledes enn i vanlig SAS, datasett kalles tabeller, variable blir kolonner og observasjoner kalles linjer eller rader. En tabell kan være en databasetabell, et SAS access-view, et PROC SQL-view eller et SAS datasett.

```
PROC SQL <options> ;
ALTER TABLE tabell-navn <gyldighet-1 <, gyldighet-n>>
    <ADD kolonne-definisjon <, kolonne-definisjon> ...>
    <MODIFY kolonne-definisjon <, kolonne-definisjon> ...>
    <DROP kolonne-definisjon <, kolonne-definisjon> ...> ;
CONNECT TO databasesystem <AS alias> <(<tilknytningskommandoer> <tilknytningsparametre>)> ;
CREATE <UNIQUE> INDEX indeks-navn ON tabell-navn (kolonne-navn <, kolonne-navn> ... ) ;
CREATE TABLE tabell-navn (kolonne-definisjon <, kolonne-definisjon> ...> )
    <(kolonne-definisjon , gyldighet)>;
CREATE TABLE tabell-navn AS spørre-uttrykk ;
CREATE TABLE tabell-navn LIKE tabell-navn ;
CREATE VIEW view-navn AS spørre-uttrykk <ORDER BY variabel <, variabel> ...>
    <USING sql-setning <libname-setning> ...>;
DELETE FROM tabell-navn | <libref.> SAS-access-view <AS alias> <WHERE sql-uttrykk> ;
DESCRIBE TABLE tabell-navn <, tabell-navn> ... ;
DESCRIBE TABLE CONSTRAINTS tabell-navn <, tabell-navn> ... ;
DESCRIBE VIEW view-navn <, view-navn> ... ;
DISCONNECT FROM database-system | alias ;
DROP INDEX indeks-navn <, indeks-navn> ... FROM tabell-navn ;
DROP TABLE tabell-navn <, tabell-navn> ... ;
DROP VIEW view-navn <, view-navn> ... ;
EXECUTE (database-systems SQL-setning) BY database-system | alias ;
INSERT INTO tabell-navn | <libref.> SAS-access-view <(kolonne-navn <, kolonne-navn> ... )>
    SET (kolonne-navn = sql-uttrykk <, kolonne-navn = sql-uttrykk> ... )
    <SET (kolonne-navn = sql-uttrykk <, kolonne-navn = sql-uttrykk> ... ) > ... ;
INSERT INTO tabell-navn | <libref.> view-navn <(kolonne-navn <, kolonne-navn> ... )>
    VALUES (verdi <, verdi> ... )
    <VALUES (verdi <, verdi> ... )> ... ;
INSERT INTO tabell-navn | <libref.> view-navn <(kolonne-navn <, kolonne-navn> ... )>
    spørre-uttrykk ;
RESET <options> ;
SELECT <DISTINCT> objekt <, objekt>
    <INTO :makronavn <, :makronavn> ... >
    FROM fra-liste
    <WHERE sql-uttrykk>
    <GROUP BY gruppering <, gruppering> ... >
    <HAVING sql-uttrykk>
    <ORDER BY sorteringsbegrep <,sorteringsbegrep>...> ;
UPDATE tabell-navn | <libref.> SAS-access-view <AS alias>
    SET kolonne-navn = sql-uttrykk <, kolonne-navn = sql-uttrykk> ...
    <SET kolonne-navn = sql-uttrykk <, kolonne-navn = sql-uttrykk> ... > ...>
    <WHERE sql-uttrykk> ;
VALIDATE spørre-uttrykk ;
```

Utfyllende syntaks for parametrene som kan brukes i SQL-setningene:

betingelse:

```
sql-uttrykk <NOT> IN(konstant ...< ,konstant>)
sql-uttrykk <NOT> IN(spørre-uttrykk)
sql-uttrykk <NOT> LIKE spørre-uttrykk
<NOT> EXISTS(spørre-uttrykk)
```

databasesystem:

```
ORACLE eller andre databasesystemer
```

fra-liste:

```
tabell-navn <AS> alias >
<libref.>view <AS> alias >
<libref.>SAS-access-view <AS> alias >
joined-tabell
(spørre-uttrykk) <AS> alias > <(kolonne-navn ... < , kolonne-navn)>>
CONNECTION TO databasesystem (sql-uttrykk)
CONNECTION TO alias (sql-uttrykk)
```

gruppering:

```
kolonne-nr
kolonne-navn
sql-uttrykk
```

gyldighet:

```
ADD <CONSTRAINT gyldighetsregel-navn> gyldighetsregel
DROP CONSTRAINT gyldighetsregel-navn
```

gyldighetsregel:

```
NOT NULL(kolonne)
CHECK (WHERE-betingelse)
PRIMARY KEY (kolonner)
UNIQUE (kolonner)
FOREIGN KEY (kolonner)
REFERENCES tabell-navn <ON DELETE restrict | set null cascade>
<ON UPDATE restrict | set null | cascade>
```

join-uttrykk:

```
konstant
kolonne-navn
SAS funksjon
USER
<ALL | ANY > (spørre-uttrykk)
```

joined tabell:

```
tabell <<AS> alias> , tabell <<AS> alias> ... < ,tabell <<AS> alias>>
tabell <INNER> JOIN tabell ON join-uttrykk
tabell LEFT|RIGHT|FULL JOIN tabell ON join-uttrykk
```

for joined tabell er tabell definert slik:

```
tabell-navn
<libref.>view
<libref.>SAS-access-view
spørre-uttrykk
```

kolonne-attributt:

```
<INFORMAT=innlesingsformat>
<FORMAT=utskriftsformat>
<LABEL='tekst'>
```

kolonne-definisjon:

```
kolonne CHARACTER | VARCHAR <(bredde)> <kolonne-attributt>
kolonne INTEGER | SMALLINT <kolonne-attributt> ... < ,kolonne-attributt>
kolonne DECIMAL | NUMERIC | FLOAT <(bredde< ,desimaler)>
<kolonne-attributt> <... < ,kolonne-attributt>>
kolonne REAL | DOUBLE PRECISION <kolonne-attributt> ... < ,kolonne-attributt>
kolonne DATE <kolonne-attributt>
```

kolonne-navn:

kolonne
tabell.kolonne
tabell-alias.kolonne
view.kolonne
view-alias.kolonne
SAS-access-view.kolonne
SAS-access-view-alias.kolonne
CALCULATED alias

objekt:

*
tabell.*
view.*
view-alias.*
SAS-access-view.*
SAS-access-view-alias.*
kolonne-navn <AS alias> <kolonne-attributt ... <kolonne-attributt>>
omkodingsuttrykk <AS alias>
sql-uttrykk <AS alias>

omkodingsuttrykk:

CASE
 WHEN <betingelse> THEN <resultat>
 WHEN <betingelse> THEN <resultat>
 ...
 ELSE <resultat>
END

sorteringsbegrep:

kolonne-nr <ASC | DESC>
kolonne-navn <ASC | DESC>
sql-uttrykk <ASC | DESC>

spørre-uttrykk:

tabell-uttrykk <<INTERSECT | OUTER UNION | UNION | EXCEPT>
 <CORRESPONDING> <ALL> tabell-uttrykk> ...

sql-funksjon:

AVG | MEAN | COALESCE | COUNT | FREQ | N | CSS | CV | MAX | MIN | NMISS | PRT | RANGE
| STD | STDERR | SUM | SUMWGT | T | USS | VAR (<DISTINCT> sql-uttrykk)>

sql-uttrykk:

konstant
kolonne-navn
sql-funksjon
SAS funksjon
USER
<ALL | ANY> (spørre-uttrykk)

tabell-uttrykk:

SELECT <**DISTINCT**> objekt < , objekt>
<**INTO** :makronavn < , :makronavn> ... >
 FROM fra-liste
<**WHERE** sql-uttrykk>
<**GROUP BY** gruppering < , gruppering > ... >
<**HAVING** sql-uttrykk>

Options til PROC SQL-setningen:

DOUBLE

Angir at resultatet av en SELECT-setning skal skrives ut med dobbel linjeavstand. Tilsvarende DOUBLE i PROC PRINT, se side 159. Standard er NODOUBLE.

DQUOTE=

ANIS angir at tekst som står i doble frutter regnes som variabelnavn, ikke tekststrenger. Dette medfører at vi kan bruke navn som vanligvis ikke tillates i SAS som variabelnavn. SAS , som er standard, angir at tekst i doble frutter skal behandles som tekststrenger.

ERRORSTOP

ERRORSTOP virker når SAS kjører i batch. Med ERRORSTOP i effekt vil SAS stoppe eksekveringen av SQL-setninger når en feil oppstår. Resten av SQL-setningen vil syntakssjekkes, men ikke eksekveres. Med NOERRORSTOP vil SAS prøve å utføre resten av SQL-setningene etter at en feil er oppstått (når EXEC-option er satt).

EXEC

Med NOEXEC vil vi bare sjekke syntaksen til SQL-setningene uten at programmet kjøres etterpå. EXEC er standard.

FEEDBACK

Angir om den utvidede (tilpassede) SQL-setningen skal vises i SASlog. Standard er NOFEEDBACK.

FLOW | FLOW=n <m>

Angir at lange karakterkolonner skal deles over flere linjer. n angir fast bredde for karakterkolonner, mens n m angir at bredden skal være et passende sted mellom n og m.

INOBS=

Angir maksimalt antall tabellinjer som skal benyttes av en SQL-setning.

PASSTHRU

Angir om PASSTHRU-metode skal benyttes ved eksekvering

LOOPS=antall

Angir antall iterasjoner som maksimalt skal utføres i en spørrings innerste løkke.

_METHOD

Viser i SASlog hvilke eksekveringsmetoder som velges:

sqxcrt	Create table as select
sqxslct	Select
sqxjst	Step loop join (Cartesian)
sqxjm	Merge join
sqxjndx	Index join
sqxjhsh	Hash join
sqxsort	Sortering
sqxsrc	Source rows
sqxfl	Filter rows
sqxsumg	Summary statistics (group by)
sqxsumn	Summary statistics (ikke gruppert)
sqxuniq	Distinct rows

NUMBER

Angir om resultatet av en SELECT-setning skal inneholde en kolonne med navn ROW som inneholder linjenr. Denne tilsvarer OBS-option i PROC PRINT, se side 159. Standard er NONUMBER.

OUTOBS=antall

Angir maksimalt antall tabellinjer som skal lages med en SQL-setning

PRINT

Angir at resultatet av en SELECT-setning skal skrives ut eller ei (NOPRINT). PRINT er standard

PROMPT

Endrer effekten til LOOPS, INOBS og OUTOBS-options. I stedet for å avbryte når et antall iterasjoner er nådd, vil vi bli spurt om vi ønsker å stoppe eller fortsette. Dette gjentar seg for hver gang vi har gjennomført det maksimale antall iterasjoner.

SORTSEQ=tegnrekkefølge

Følger en bestemt tegnrekkefølge ved sortering. Mulige rekkefølger er ASCII, DANISH, EBCDIC, FINNISH, NATIONAL, NORWEGIAN og SWEDISH

STIMER

Angir at SAS skriver ut informasjon om tidsbruk til SASlog for hver SQL-setning istedenfor som en akkumulert verdi når PROC SQL avsluttes. For at denne skal virke må også SAS system option STIMER (se side 46) være satt. Standard er NOSTIMER.

_TREE

Viser en trestruktur i SASlog for eksekveringen av SQL-setningene

UNDO POLICY=REQUIRED | NONE | OPTIONAL

Brukes for å bestemme hva som skal gjøres hvis det oppstår en feil mens en INSERT eller en UPDATE-setning utføres. Når en slik feil oppstår utføres en angrefunksjon på de dataene som var behandlet før feilen oppsto. REQUIRED er standard.

REQUIRED	PROC SQL vil utføre angrefunksjon hvis mulig. Hvis dette ikke kan gjøres pålitelig, vil ikke SQL-setningen bli utført i det hele tatt, og en feilmelding skrives til SASlog.
NONE	PROC SQL vil hoppe over tabellinjer som ikke kan bli eksekvert med INSERT eller UPDATE. Det skrives ut en melding til SASlog om dette.
OPTIONAL	Denne er en kombinasjon av REQUIRED og NONE. Hvis SQL-setningen kan utføres pålitelig, så blir den det, hvis ikke hoppes den over.

SAS-setninger til PROC SQL:

ALTER TABLE

Brukes for å legge til eller slette kolonner i en tabell. Vi kan også endre lengde, innlesingsformat, utskriftsformat og label på eksisterende kolonner. Alle verdier i nye kolonner settes til MISSING. ALTER TABLE kan ikke brukes på view-descriptors.

CONNECT TO

Kan brukes til å få forbindelse til et database-system.

CONNECTION TO

Kan brukes til å få forbindelse til et database-system ved bruk av FROM i SELECT-setninger.

CREATE

Med CREATE kan vi lage tabeller, views basert på tabeller eller andre views, samt indekser og gyldighetsregler til kolonner i tabeller.

DELETE

Fjerner alle linjer som oppfyller where-uttrykket som spesifiseres.

DESCRIBE

Skriver definisjonen av databasetabeller, PROC SQL-view eller gyldighetsregler til SASlog.

DISCONNECT FROM

Avslutter forbindelsen til et database-system.

DROP

Sletter en tabell, et PROC SQL-view eller en indeks.

EXECUTE

Utfører ikke-spørrende SQL-setninger, f.eks. CREATE, DELETE, INSERT og UPDATE, direkte på databasen. Disse setningen skal ha den syntaksen databasesystemet krever, ikke SAS-syntaks. Vi kan ikke bruke semikolon i denne syntaksen.

INSERT

Legger til nye linjer i en tabell, disse gir vi verdier i samme slengen.

RESET

Setter PROC SQL-options tilbake til opprinnelig verdi, eller gir dem nye verdier.

SELECT

SELECT-setningen brukes til å lage en spørring mot en eller flere databasetabeller og/eller SAS datasett. Her kan vi utføre seleksjoner, koblinger, beregninger, aggregeringer og annet nyttig.

UPDATE

Oppdaterer verdier i en tabell.

VALIDATE

Brukes for å syntakssjekke SELECT-setninger.

SQL er i seg selv et meget omfattende språk. Vi nøyer oss her med noen eksempler på bruk av PROC SQL.

Eksempel 1:

Lager en utlisting av observasjoner for unge straffedømte. Utlistingen tilsvare PROC PRINT:

```
PROC SQL ;
  TITLE 'Unge straffedømte';
  SELECT kjonn, alder, dd51, dd52, antre, antrf, tilbf
  FROM tilbfall.a3694
  WHERE alder < 18;
RUN;
```

Eksempel 2:

Vi lager en aggregert oversikt over straffedømte etter kjønn. Vi bruker SQL-funksjonene N og MEAN for å beregne antall observasjoner og gjennomsnitt. Disse tilsvare SAS funksjonene med samme navn.

```
PROC FORMAT ;
  VALUE $kjonn
    '1' = 'MENN'
    '0' = 'KVINNER'
  ;
PROC SQL ;
  TITLE 'Kjønnforskjeller blant straffede i 1987';
  SELECT kjonn          FORMAT=$kjonn. ,
         N(antr)        FORMAT=7. LABEL='Antall med dom',
         MEAN(antr)     FORMAT=7.1 LABEL='Gjennomsnittlig antall dommer',
         N(tilbf)       FORMAT=7.1 LABEL='Antall med tilbakefall'
  FROM tilbfall.a3694
  GROUP BY kjonn
  ORDER BY kjonn;
RUN;
```

Her er resultatet av programmet:

```
Kjønnforskjeller blant straffede i 1987
```

Kjønn	Antall med dom	Gjennomsnittlig antall dommer	Antall med tilbakefall
KVINNER	987	3.0	399.0
MENN	9644	4.9	5073.0

Eksempel 3:

Vi kan bruke PROC SQL til å gi en makrovariabel verdi. Denne verdien henter vi fra et SAS datasett eller et SAS-view. Fordelen med dette er at vi kan fylle makrovariabelen med data fra alle observasjonene i SAS-datasettet.

I eksempelet under skal vi bruke dette til å generere labler til variable med fortløpende nummerering. Vi skal lage et SAS datasett med variablene vare1, vare2, ... , vare9. Disse variablene skal tekstes med samme tekst (Varekode 1), men med nummerering som tilsvarer variabelnavnet.

Vi gjør dette i 3 steg. Først et datasteg der vi lager tekstene som skal brukes for å lage labler. Det lages et temporært datasett med en observasjon for hver variabel. Deretter bruker vi PROC SQL til å legge alle tekstene i makrovariabelen *vari*. Til slutt kjøres datasteget der tekstene hentes fra makrovariabelen slik at lablene lages. Med mange variable og lange tekster kan vi sprengre grensen for en makrovariables lengde, men opptil 1000 variable med label-lengde på ca. 13 tegn bør gå bra.

```
DATA tekster;
DO i = 1 TO 9;
  vari = "vare" || TRIM(LEFT(i)) || " = 'Varekode " || TRIM(LEFT(i)) || "' " ;
  OUTPUT;
END;

PROC SQL NOPRINT;
  SELECT vari
  INTO :vari SEPARATED BY " "
  FROM tekster
  ;

DATA forbruk (DROP=i j);
  ARRAY vare (*) vare1 - vare9;
  * Generer testdata;
  DO j = 1 TO 17;
    DO i = 1 TO DIM(vare);
      vare(i) = i + ROUND(RANUNI(0)*903);
    END;
  OUTPUT;
END;
LABEL &vari ;
RUN;
```

Eksempel 4:

Program som fjerner alle identiske observasjoner på et SAS datasett (meddubl) og legger resultatet sortert (variablene sorteres etter den rekkefølge de har i SAS datasettet) ut på et nytt (udubl):

```
PROC SQL;
  CREATE TABLE udubl AS
  SELECT DISTINCT * FROM meddubl ;
```

Eksempel 5:

Et program som lister alle dubletter i et SAS datasett og et program som skriver dublettene ut til et nytt SAS datasett:

```
PROC SQL;
  TITLE 'Oversikt over dubletter';
  SELECT * FROM meddubl
  GROUP BY fnr
  HAVING COUNT(fnr) > 1

  CREATE TABLE dubletter AS
  SELECT * FROM meddubl
  GROUP BY fnr
  HAVING COUNT(fnr) > 1
  ;
```

3.11.19 PROC TABULATE

PROC TABULATE er SAS' spesialverktøy for å lage tabeller. Med utgangspunkt i et SAS datasett kan vi lage tabeller i opptil 3 dimensjoner (side, forspalte og hode). Det vanligste er likevel 2 dimensjoner: forspalte og hode (eller om man vil: linje og kolonne). Resultatet fra TABULATE er en ferdig tabell som legges enten i utskriftsvinduet i SAS eller på en fil (altså ikke et SAS datasett).

SAS datasettets variable brukes i PROC TABULATE enten til gruppering eller klassifisering (variablene omtales som CLASS-variable), eller til analyse (variablene omtales som VAR-variable). Variable kan også brukes til veiing (WEIGHT). Grupperingsvariablene danner forspalten og/eller hodet, mens analysevariablene utgjør selve tabellinnmaten i form av summer, gjennomsnitt, standard avvik, etc. Ved hjelp av PROC TABULATE kan vi lage alt fra enkle til relativt komplekse tabeller med korte programmer.

PROC TABULATE er mer utfyllende beskrevet i heftet SAS som tabellverktøy, Interne Dokumenter 98/11.

PROC TABULATE <options>

WHERE where-uttrykk;

BY <DESCENDING> variabel-1 <... <DESCENDING> variabel-n <NOTSORTED>>;

CLASS grupperingsvariable </ options>;

CLASSLEV variable / **STYLE**=[stil-spesifikasjoner] | <PARENT> [stil-spesifikasjoner];

VAR analysevariable </ options>;

TABLE tabellspesifikasjon </ tabelloptions>;

FORMAT variabel format. <...variable format.>;

KEYLABEL statistikknøkkelord ='tekst' | ALL='tekst' <...statistikknøkkelord ='tekst'>;

KEYWORD statistikknøkkelord | ALL / **STYLE**=[stil-spesifikasjoner] | <PARENT> [stil-spesifikasjoner];

LABEL variabel='tekst' <...variabel ='tekst'>;

WEIGHT variabel;

FREQ variabel;

Ikke alle setningene er nødvendige, minstekravet er **PROC TABULATE**, **CLASS** eller **VAR** og **TABLE**.

Options til PROC TABULATE-setningen:

CONTENTS=

Endre innholdet i innholdsfortegnelse til angitt tekst. Brukes sammen med CONTENTS=option i ODS html, se side 51.

CLASSDATA=

Navn på et annet SAS datasett som inneholder kombinasjoner av grupperingsvariable. Alle kombinasjoner kommer med i tabellen, med mindre vi velger EXCLUSIVE-option. Da selekteres bare de som er med i CLASSDATA-datasettet. Fin å bruke når vi ønsker å ha med alle kategorier i tabellen, selv om ikke alle har data i SAS datasettet. Alle grupperingsvariable som brukes i tabellen må være med i CLASSDATA-datasettet, fortrinnsvis med alle kombinasjoner av alle verdier. Kan også brukes til å lage seleksjonslister.

Eksempel der vi genererer en kombinasjonsliste av fylke og kjønn på grunnlag av SAS datasett for hver av dem ved hjelp av DATA-steg og PROC FREQ. Deretter lager vi en tabell som bare inneholder disse kombinasjonene.

```
DATA a3694;
  INFILE CARDS;
  INPUT @01 kjonn      $CHAR1.
        @03 forbr     $CHAR3.
        @07 antall     4.
        @12 fylke     $CHAR2.
        ;
CARDS;
1 173      713 02
2 173      93 02
1 174 2682 03
2 174      139 04
1 176      537 05
2 176      24 06
1 111      85 07
1 112      400 08
1 194      400 09
2 194      100 10
2 171      100 11
2 192      31
1 121      13 03
1 129      100 14
1 123      200 15
2 121      1 16
2 124      1 17
1 151      500 18
1 159      81 19
1 153      500 11
2 153      39 02
1 075      700 03
1 331      500 11
1 332      214 02
2 332      250 03
2 076      30 11
1 002 1019 02
1 004 1000 03
2 777      79 14
2 188      100 23
;
RUN;

DATA fylkkod (DROP=i);
  LENGTH fylke $ 2;
  DO i = 1 TO 20;
    fylke = PUT(i,Z2.);
    IF i NE 13 THEN
      OUTPUT;
  END;
RUN;

DATA kjonnkod;
  INPUT @01 kjonn $CHAR1.
CARDS;
1
2
;
RUN;

DATA fylkkj;
  SET fylkkod kjonnkod ;
RUN;

PROC FREQ DATA=fylkkj NOPRINT ;
  TABLES fylke*kjonn /SPARSE
          OUT=fylkkj (WHERE=(count+percent = 0));
RUN;

PROC TABULATE DATA=a3694
  CLASSDATA=fylkkj EXCLUSIVE FC=' -----' NOSEPS MISSING F=8.;
  CLASS fylke kjonn;
  VAR antall;
  TABLE ALL= 'I alt' fylke= ' '
          '
          (ALL='I alt' kjonn=' ')*SUM=' '*antall=' '
          /RTS=8 MISSTEXT='- ' BOX=[LABEL='Fylker']
          ;
  TITLE 'Tabulate classdata, tar bare med de i CLASSDATA (EXCLUSIVE)' ;
RUN;
```

Utskriften:

Tabulate classdata, tar bare med de i CLASSDATA (EXCLUSIVE)

Fylker	I alt	1	2
I alt	10500	9644	856
01	-	-	-
02	2078	1946	132
03	4645	4395	250
04	139	-	139
05	537	537	-
06	24	-	24
07	85	85	-
08	400	400	-
09	400	400	-
10	100	-	100
11	1130	1000	130
12	-	-	-
14	179	100	79
15	200	200	-
16	1	-	1
17	1	-	1
18	500	500	-
19	81	81	-
20	-	-	-

DATA= SAS datasett

Her oppgis navnet på SAS-datasettet tabellen skal lages fra. (Utelates DATA= vil det datasettet som er 'ferskest' i samme program eller interaktive SAS sesjon bli brukt.)

EXCLNPWGHTS

Selekter kun observasjoner som har positive vekter. Se også WEIGHT-setningen.

EXCLUSIVE

Brukes sammen med CLASSDATA=option. Bare de verdier av grupperingsvariablene som er i SAS datasettet nevnt i CLASSDATA=option selekteres.

FORMAT=formatnavn

Brukes hvis vi ønsker å angi felles formatering av tallene i tabellen, altså at alle kolonnene i utgangspunktet skal være like brede. FORMAT= kan forkortes til F= . FORMAT=7.2 gir en kolonnebredde på syv posisjoner, herav to desimaler. Bemerk at standard-formatet kan overstyres for enkeltkolonner ved at vi angir et annet format i selve TABLE-setningen. Utelates FORMAT= blir formatet BEST12.2 benyttet.

FC= ' 11 tegn '

FC står for FormChar og brukes for å angi hvilke tegn tabellen skal rammes inn med. Det er 11 forskjellige møtepunkter i en tabellramme: Rekkefølgen er slik: 1. er vertikal strek, 2. er horisontal strek, 3. er øvre venstre hjørne, 4. er midten øverst, 5. er øvre høyre hjørne, 6. er midten til venstre, 7. er midt i tabellen, 8. er midten til høyre, 9. er nedre venstre hjørne, 10. er midten nederst og 11. er nedre høyre hjørne, se figur til venstre.

FC='-----' (1 blank og 10 streker) gir kun horisontale streker.

MISSING

Angis hvis vi vil at observasjoner som mangler verdi i *grupperingsvariable* likevel skal være med i tabellen. (Dersom MISSING ikke er med, blir observasjoner med missing i grupperingsvariablene utelatt.) Bemerk at MISSING ikke har noen innvirkning på behandlingen av missing i *analysevariable*.

NOSEPS

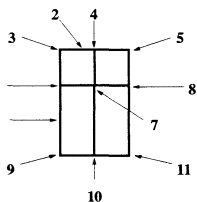
Fjerner alle vannrette delelinjer i tabellinnmaten. Linjene skrives da ut uten luft imellom, selv om vi har angitt blank som vannrett strek i FC=

NOTRAP

FPE (Floating point exception) recovery skrur av. Kan være gunstig for kjøretiden.

ORDER=rekkefølge

Angir hvilken rekkefølge verdiene til grupperingsvariable skal skrives ut i. Velger vi ORDER=FORMATTED sorteres de alfabetisk etter teksten i formatet som eventuelt brukes istedenfor etter koden. ORDER=FREQ sørger for at den verdi med flest observasjoner skrives først. ORDER=DATA gjør at verdiene beholder den rekkefølgen de har i datasettet. Den første forekomsten av verdiene bestemmer rekkefølgen. Dette gjelder ikke når vi bruker CLASSDATA- eller PRELOADFMT-option. Da blir rekkefølgen som definert i CLASSDATA-datasettet eller formatet til variabelen som har PRELOADFMT-option. Standardverdi er ORDER=UNFORMATTED (= INTERNAL) som skriver ut verdiene i sortert rekkefølge.



OUT=

Legger tabellen ut på SAS datasett. Følgende variable lages:

Variable definert med BY-setningen

Variable definert i CLASS-setningen

TYPE, viser sammenhengen mellom grupperingsvariablene

PAGE, den logiske siden som inneholder observasjonen

TABLE, tabellnummeret observasjonen hører til

Statistikkvariable, utførte beregninger

SAS datasettet som lages vil kunne være vanskelig å bruke videre. Det avhenger av hvor kompliserte TABLE-setninger vi har brukt. Vi får et SAS datasett som minner om de vi kan lage ved å aggregere i PROC MEANS. Hver BY-variabel og grupperingsvariabel kommer ut på det nye SAS datasettet, og det lages en variabel pr. analysevariabel og statistikk.

STYLE=

Brukes i forbindelse med html-generering for å angi standard utseende for tabellen. Dette er beskrevet nærmere i et eget vedlegg, se side 264.

VARDEF=divisor

Angir divisor som brukes til å regne ut varianser. Gyldige verdier er:

DF N-1 skal brukes som divisor (standard)

N N skal brukes som divisor

WDF Summen av vektene - 1 skal brukes som divisor

WEIGHT Summen av vektene skal brukes som divisor

Options som ikke er beskrevet her: QMARKERS, QMETHOD og QNTLDEF.

Setninger til PROC TABULATE:

WHERE

Selekterer hvilke observasjoner det skal lages tabeller for. Bare de som tilfredstiller WHERE-betingelsen tas med.

BY

BY-setningen brukes på samme måte i TABULATE som i de andre SAS-prosedyrene. Bruker vi BY-setningen, vil vi få én tabell for hver verdi av BY-variabelen. Hvis vi bruker BY-setningen må SAS-datasettet være sortert på forhånd (med mindre vi har med sorteringsoption NOTSORTED).

Istedenfor å bruke BY i TABULATE, vil det som oftest lønne seg å bruke sidedimensjonen i TABLE-setningen. Denne gjør samme nytte som BY, men krever **ikke** at SAS-datasettet er sortert på forhånd.

CLASS

I CLASS-setningen lister vi opp grupperingsvariablene, klassifiseringsvariablene. Grupperingsvariable er variable med kategoriske verdier. Disse er som oftest definert som karaktvariable (av type *character*), men de kan også være numeriske (av type *numeric*). Numeriske, (dvs. kontinuerlige) variable bør grupperes ved hjelp av et format for å kunne brukes som grupperingsvariable.

Vær oppmerksom på at observasjoner med missing i en eller flere av grupperingsvariablene utelates fra tabellen såfremt vi ikke har med MISSING-option i selve PROC TABULATE-setningen. Dette gjelder for alle variablene i CLASS-setningen, selv om de ikke brukes i tabellen. Vi kan ha flere CLASS-setninger i samme PROC TABULATE-program.

Options til CLASS-setningen:

ASCENDING

Grupperingsvariablene skrives ut i stigende rekkefølge, dette er standard.

DESCENDING

Grupperingsvariablene skrives ut i synkende rekkefølge.

EXCLUSIVE

Brukes sammen med PRELOADFMT=option. Bare de verdier av grupperingsvariablene som er i de nevnte SAS-formatene selekteres.

GROUPINTERNAL

Formater til grupperingsvariablene hektes på etter at SAS har gruppert dem. Dette kan spare noe maskintid. Kan brukes når formatene ikke grupperer forskjellige variabelverdier. Når vi bruker formater til å slå sammen grupper, f.eks. fra alder til 5-årige aldersgrupper, bør vi ikke bruke GROUPINTERNAL. Vi må også være oppmerksomme på restgrupperinger definert med OTHER i PROC FORMAT. Har

disse forskjellige verdier, vil hver verdi komme på hver sin linje (ev. kolonne) i tabellen.

MISSING

Sørger for at MISSING behandles som gyldig verdi for grupperingsvariablene. Påvirker ingen statistiske beregninger.

MLF

Angir at vi kan bruke SAS formater med overlappende verdier. Krever at formatene er definert som MULTILABEL i PROC FORMAT, se side 140. Gjør at samme variabel kan grupperes på forskjellige måter i samme format, og vi kan få dem ut i en tabell.

Eksempel:

```
DATA bef;
INPUT @01 fylke      2.
      @04 menn       6.
      @11 kvin       6.
      ;
      folk = SUM(menn,kvin);
CARDS;
01 119617 123968
02 224381 229109
03 240276 259417
04  91945  94173
05  90379  91783
06 114826 118141
07 102106 106581
08  80641  83216
09  50293  50859
10  75489  77064
11 181692 182649
12 212786 216037
14  54532  53258
15 121184 120788
16 128086 131091
17  63485  63300
18 119549 119731
19  75551  74737
20 382888 36591
;
RUN;

PROC FORMAT ;
VALUE landsde (MULTILABEL NOTSORTED)
  01 - 08 = 'Østlandet'
  01      = '01 Østfold'
  02      = '02 Akershus'
  03      = '03 Oslo'
  04      = '04 Hedmark'
  05      = '05 Oppland'
  06      = '06 Buskerud'
  07      = '07 Vestfold'
  08      = '08 Telemark'
  09 - 11 = 'Sørlandet'
  09      = '09 Aust-Agder'
  10      = '10 Vest-Agder'
  11      = '11 Rogaland'
  12 - 15 = 'Vestlandet'
  12      = '12 Hordaland'
  14      = '14 Sogn og Fjordane'
  15      = '15 Møre og Romsdal'
  16 - 17 = 'Trøndelag'
  16      = '16 Sør-Trøndelag'
  17      = '17 Nord-Trøndelag'
  18 - 20 = 'Nord-Norge'
  18      = '18 Nordland'
  19      = '19 Troms'
  20      = '20 Finnmark'
OTHER = 'Uoppgitt'
;

PROC TABULATE DATA=bef FC='-----' NOSEPS MISSING F=8. ORDER=data;
CLASS fylke / MLF PRELOADFMT;
VAR folk menn kvin;
TABLE ALL= 'I alt'  fylke= ' '
      ,
      (folk='I alt' menn='Menn' Kvin='Kvinner')*(SUM=' ')
      kvin='% kvinner'*PCTSUM<folk>=' '*F=7.1
      /RTS=22 MISSTEXT='- ' BOX=[LABEL='Fylke']
      ;
FORMAT fylke landsde. ;
TITLE 'Innbyggere. 1998';
```

Dette gir denne utskriften:

Innbyggere. 1998

Fylke	I alt	Menn	Kvinner	% kvinner
I alt	4417599	2185106	2232493	50.5
Østlandet	2170559	1064171	1106388	51.0
01 Østfold	243585	119617	123968	50.9
02 Akershus	453490	224381	229109	50.5
03 Oslo	499693	240276	259417	51.9
04 Hedmark	186118	91945	94173	50.6
05 Oppland	182162	90379	91783	50.4
06 Buskerud	232967	114826	118141	50.7
07 Vestfold	208687	102106	106581	51.1
08 Telemark	163857	80641	83216	50.8
Sørlandet	618046	307474	310572	50.3
09 Aust-Agder	101152	50293	50859	50.3
10 Vest-Agder	152553	75489	77064	50.5
11 Rogaland	364341	181692	182649	50.1
Vestlandet	778585	388502	390083	50.1
12 Hordaland	428823	212786	216037	50.4
14 Sogn og Fjordane	107790	54532	53258	49.4
15 Møre og Romsdal	241972	121184	120788	49.9
Trøndelag	385962	191571	194391	50.4
16 Sør-Trøndelag	259177	128086	131091	50.6
17 Nord-Trøndelag	126785	63485	63300	49.9
Nord-Norge	464447	233388	231059	49.7
18 Nordland	239280	119549	119731	50.0
19 Troms	150288	75551	74737	49.7
20 Finnmark	74879	38288	36591	48.9

Et problem er at det kun virker når fylke er definert som numerisk variabel.

ORDER=rekkefølge

Angir hvilken rekkefølge verdiene til grupperingsvariable skal skrives ut i. Velger vi ORDER=FORMATTED blir de sortert alfabetisk etter teksten i formatet som eventuelt brukes istedenfor etter koden. ORDER=FREQ sørger for at den verdi med flest observasjoner skrives først. ORDER=DATA gjør at verdiene beholder den rekkefølgen de har i datasettet. Den første forekomsten av verdiene bestemmer rekkefølgen. Dette gjelder ikke når vi bruker CLASSDATA- eller PRELOADFMT-option. Da blir rekkefølgen som definert i CLASSDATA-datasettet eller formatet til variabelen som har PRELOADFMT-option. Standardverdi er ORDER=UNFORMATTED som skriver ut verdiene i sortert rekkefølge.

PRELOADFMT

Vi bruker PRELOADFMT for å si at vi ønsker å beholde rekkefølgen formatet fikk da det ble lagret. Med denne option og NOTSORTED-option til VALUE-setningen i PROC FORMAT, kan vi sikre at vi får linjene og kolonnene i den riktige rekkefølgen i forspalte og hodet i tabellen. For at denne skal virke må option ORDER=data være med i PROC TABULATE-setningen.

Eksempel:

```
DATA a3694;
  INFILE CARDS;
  INPUT @01 kjonn $CHAR1.
        @03 forbr $CHAR3.
        @07 antall 4.
        ;
CARDS;
1 173 713
0 173 93
1 174 2682
0 174 139
1 176 537
0 176 24
1 111 885
0 111 231
1 121 313
0 121 2
1 151 1081
0 151 39
1 075 1414
0 075 280
1 002 1019
1 004 1000
0 777 79
0 188 100
;
```



```

PROC FORMAT ;
  VALUE $kjonn (NOTSORTED)
    '1' = 'Menn'
    '0' = 'Kvinner'
    OTHER = 'Uoppgitt'
  ;

  VALUE $forbr (NOTSORTED)
    '173' = 'Simpelt tyveri'
    '174' = 'Grovt tyveri'
    '176' = 'Brukstyveri av motorkjøretøy'
    '111' = 'Underslag, bedrageri og dokumentfalsk'
    '121' = 'Forbrytelse mot sedeligheten'
    '151' = 'Forbrytelse mot liv, legeme og helbred'
    '075' = 'Narkotikaforbrytelse'
    OTHER = 'Annen forbrytelse'
  ;

PROC TABULATE DATA=a3694 NOSEPS FC='-----' MISSING ORDER=DATA;
  CLASS forbr kjonn /PRELOADFMT ;
  VAR antall ;
  TABLE ALL='Alle reaksjoner' forbr=''
    (ALL='I alt' kjonn='')*antall=''*SUM=''*F=10.
  /BOX='Hovedforbrytelse ved utgangsreaksjon'
  RTS=45 MISSTEXT='- ' ;
  FORMAT kjonn $kjonn.
  forbr $forbr. ;
  TITLE 'Straffbare etter hovedforbrytelse ved utgangsreaksjon og
kjønn';
RUN;

```

Resultatet:

Straffbare etter hovedforbrytelse ved utgangsreaksjon og kjønn

Hovedforbrytelse ved utgangsreaksjon	I alt	Menn	Kvinner
Alle reaksjoner	10631	9644	987
Simpelt tyveri	806	713	93
Grovt tyveri	2821	2682	139
Brukstyveri av motorkjøretøy	561	537	24
Underslag, bedrageri og dokumentfalsk	1116	885	231
Forbrytelse mot sedeligheten	315	313	2
Forbrytelse mot liv, legeme og helbred	1120	1081	39
Narkotikaforbrytelse	1694	1414	280
Annen forbrytelse	2198	2019	179

STYLE=stilspesifikasjoner

Brukes i forbindelse med html-generering for å angi standard utseende for tabellen. Dette er beskrevet nærmere i et eget vedlegg om html-stiler, se side 266.

CLASSLEV

CLASSLEV-setningen kan vi bruke i forbindeles med utskrift til html-filer til å endre på utseende til grupperingsvariable i forspalten eller kolonnehoder. Vi kan ha flere CLASSLEV-setninger i samme program. Kan f.eks. brukes til å gi forskjellig utseende til forskjellige nivåer i forspalten. Se html-stiler, side 266.

VAR

I VAR-setningen lister vi opp analysevariablene våre. Analysevariablene er de det beregnes statistikk på (SUM, MEAN, MAX, MIN etc). Analysevariablene *må* være numeriske. Manglende verdier for analysevariable teller ikke med i beregningene (bortsett fra NMISS som er antall observasjoner med missing). Vi kan ha flere VAR-setninger med forskjellige options. Dette kan være nyttig når vi ønsker å benytte forskjellige vekter for analysevariablene.

Options til VAR-setningen:

STYLE=stilspesifikasjoner

Brukes i forbindelse med html-generering for å angi standard utseende for tabellen. Dette er beskrevet nærmere i et eget vedlegg om html-stiler, side 266.

WEIGHT=vektvariabel

Angir vektvariabel for analysevariablene. WEIGHT-option i VAR-setningen overstyrer WEIGHT-setningen.

TABLE

I TABLE-setningen beskrives tabellen i detalj. Derfor kan TABLE-setningen ofte bli ganske lang. Del den gjerne over flere linjer, det øker forståeligheten og sjansen for å oppdage feil i tide. Generelt kan TABLE-setningen skrives slik: TABLE sideuttrykk ,
 forspalteuttrykk ,
 hodeuttrykk
 /options ;

En tabell i TABULATE ha opptil 3 dimensjoner: side, forspalte og hode. Uttrykkene som beskriver side-, forspalte- og hode-dimensjonen skilles fra hverandre med komma. Det første uttrykket definerer siden, det neste forspalten og det siste hodet. Dersom man har med 2 uttrykk, blir det første forspalte og det siste hode. Med bare 1 uttrykk får tabellen bare hode, ingen forspalte. Side, forspalte og hode defineres etter samme prinsipper.

Uttrykkene som definerer side, forspalte og hode kan inneholde følgende elementer:

Analysevariable	disse <i>må</i> være definert i VAR-setningen
Grupperingsvariable	disse <i>må</i> være definert i CLASS-setningen
ALL	egen grupperingsvariabel for totaler
Statistikk-nøkkelord	CSS, CV, MAX, MEAN, MIN, N, NMISS, PCTN, PCTSUM, PRT, RANGE, STD, STDERR, SUM, SUMWGT, USS, T og VAR, for nærmere beskrivelse, se side 269. Videre kan disse beregnes: Q3 Øvre kvartil (75. prosentil = P75) MEDIAN Medianverdi (50. prosentil = P50) Q1 Nedre kvartil (25. prosentil = P25) QRANGE Avstanden mellom øvre og nedre kvartil, (Q3-Q1) P1 1. prosentil P5 5. prosentil P10 10. prosentil P90 90. prosentil P95 95. prosentil P99 99. prosentil For prosentuering kan vi spesifisere hvilken måte det skal prosentures på med disse variantene: COLPCTN, COLPCTSUM, PAGEPCTN, PAGEPCTSUM, REPPCTN, REPPCTSUM, ROWPCTN og ROWPCTSUM. Hvis ingen av disse gir ønsket prosentuering, må vi bruke PCTN eller PCTSUM med <> for å spesifisere prosentueringen, se under.
Formater	bestemmer tallenes utseende og kolonnebredde (antall desimaler etc.)
Tekst	erstatter variabelnavn eller statistikk-nøkkelord med tekst

Dessuten kan følgende operatorer brukes i TABLE-setningen:

,	komma	skiller side-, forspalte- og hode-dimensjonene i tabellen
*	stjerne	<i>krysser</i> elementer (setter <i>innenfor</i> hverandre)
	blank	i forspalten: setter elementer <i>etter</i> hverandre i hodet: setter elementer <i>ved siden av</i> hverandre Dette kalles også <i>konkatenering</i>
()	parenteser	grupperer deler av uttrykk
<>	klammer	spesifiserer nevner i prosentuering
=	likhetstegn	1. for å angi tekst til en variabel eller et statistikk-nøkkelord 2. for å angi et format (f.eks ...*F=8.2)

Elementene i tabellen kan krysses, settes etter hverandre og grupperes i de kombinasjoner vi måtte ønske. Når vi krysser to grupperingsvariable, vil de fordeles hierarkisk med den først nevnte som høyeste (forspalten: ytterste, hode: øverste) nivå. Krysses en grupperingsvariabel med en

analysevariabel, vil det utføres statistiske beregninger på analysevariabelen fordelt etter grupperingsvariabelen.

Kryssing av to grupperingsvariable i forspalten: fylke * kjonn

Østfold	Menn	16038
	Kvinner	18995
Akershus	Menn	23070
	Kvinner	29629

Konkatenering av grupperingsvariable: fylke kjonn

Østfold	35033
Akershus	52699
Menn	39108
Kvinner	48624

Kryssing og gruppering av grupperingsvariable: fylke * (ALL kjonn)

Østfold	ALL	35033
	Menn	16038
	Kvinner	18995
Akershus	ALL	52699
	Menn	23070
	Kvinner	29629

En TABLE-setning definerer én tabell, men vi kan ha flere TABLE-setninger i én TABULATE. Vi må da være oppmerksom på at alle tabellene får samme overskrift og at missing i grupperingsvariable ikke tas med, selv om variabelen ikke brukes i TABLE-setningen (med mindre MISSING-option er med i PROC TABULATE-setningen).

Options til TABLE-setningen:

BOX=verdi

Angir tekst for forspaltehode (tekst over linjene, 'boksen' i øverste venstre hjørne). Som verdi kan vi velge en tekst angitt i fnutter (f.eks BOX='Omnibus 1999') eller et variabelnavn (gir variabelens label som forspaltehode). Vi kan også skrive BOX=_PAGE_ men dette kan vi bare gjøre hvis vi har med sidedimensjonen i tabellen. Det som skrives i 'boksen' er da verdien av grupperingsvariabelen som utgjør sidedimensjonen. Vi kan også angi stilsesifikasjoner for html-utskrift her.

CONDENSE

Hvis vi har en tabell med 3 dimensjoner (mao en tabell som også har en sidedimensjon), kan vi bruke CONDENSE til å få skrevet ut så mange av 'sidene' som mulig på ett ark. Dette for å begrense papirmengden. Hvis vi definerer en tabell som er bredere enn linjebredden tillater, vil SAS dele tabellen og hver bit vil bli skrevet ut på hver sin side. Med CONDENSE vil bitene skrives ut (under hverandre) på samme side dersom det er plass til det. CONDENSE har ingen effekt på sideskift generert ved bruk av BY-setningen

CONTENTS=tabell-referanse

Kan brukes for å angi en referanse som skal bruke til å bygge opp en liste over tabeller når vi lager html-filer. Se html-stiler, side 266.

FUZZ=verdi

Kan brukes til å eliminere ubetydelige verdier. Verdier mindre enn FUZZ-verdien for analysevariable blir tolket som 0 i beregninger og utskrift.

INDENT=verdi

Angir at vi ønsker at underliggende nivåer i rekke-dimensjonen skal skrives ut under hverandre med den oppgitte verdi som innrykk. Dette istedenfor at de settes ved siden av hverandre. Gyldig fra versjon 6.10.

MISSTEXT='tekst'

Istedenfor at manglende verdier (missing) skrives ut i tabellinnmaten som punktum, kan vi selv velge en tekst for å angi missing. Med MISSTEXT='.' vil missing i tabellen bli skrevet ut som bindestrek. Missingteksten kan maksimalt være 20 tegn lang. Vi kan også angi stilsesifikasjoner for html-utskrift her.

NOCONTINUED

Brukes for å slippe å få fortsettelsesteksten (CONTINUED) på utskriften fra andre side og utover. Gyldig fra versjon 6.08.

PRINTMISS

PRINTMISS kan brukes når vi krysser grupperingsvariable. Med PRINTMISS blir *alle* verdiene til den innerste grupperingsvariabelen skrevet ut *for hver verdi* av den ytre grupperingsvariabelen, selv om tabellcellen for den aktuelle kombinasjonen er missing.

I eksempelet under finnes det ingen observasjoner med fylke=02 og kjønn=1, heller ingen med fylke=03 og kjønn=2. Uten PRINTMISS vil det ikke lages kolonner for disse kombinasjonene. Med PRINTMISS lages det kolonner også for de kombinasjoner av fylke og kjønn som ikke er på datasettet.

```
DATA test;
  INPUT fylke   $ 1-2
        kjonn   $ 4-4 ;
  CARDS;
01 1
01 2
02 2
03 1
;
PROC TABULATE DATA=test FC='-----' NOSEPS F=8.;
  CLASS fylke kjonn ;
  FORMAT fylke $fylke.
        kjonn $kjonn. ;
  TABLE fylke=' '*kjonn=' '*N='Antall'
        ;
  TITLE 'Uten PRINTMISS.';

PROC TABULATE DATA=test FC='-----' NOSEPS F=8.;
  CLASS fylke kjonn ;
  FORMAT fylke $fylke.
        kjonn $kjonn. ;
  TABLE fylke=' '*kjonn=' '*N='Antall'
        /PRINTMISS;
  TITLE 'Med PRINTMISS';
```

Uten PRINTMISS.					
Østfold		Akershus		Oslo	
Menn	Kvinner	Kvinner	Menn		
Antall	Antall	Antall	Antall		
1	1	1	1		

Med PRINTMISS.					
Østfold		Akershus		Oslo	
Menn	Kvinner	Menn	Kvinner	Menn	Kvinner
Antall	Antall	Antall	Antall	Antall	Antall
1	1	.	1	1	.

ROW=

Standard verdi er ROW=CONSTANT, alternativet er ROW=FLOAT. For å forstå hvordan vi kan bruke ROW= må vi først være klar over det faktum at tilgjengelig forspaltebredde *fordeles likt* mellom de elementene som utgjør forspalten. Hvis forspalten består av flere elementer krysses, vil hvert element få like stor plass i forspalten. Hvis forspalten f.eks er fylke*kjonn, og tilgjengelig forspaltebredde er 20 tegn, vil fylke og kjønn få 10 tegn hver til rådighet. (Dette er en stor svakhet: Hvis vi skal lage en bred tabell, må vi ofte knipe inn forspaltebredden med RTS=. Hvis forspalten vår da er fylke krysses med kjønn og RTS=28, hadde det vært ønskelig å kunne fordele forspaltebredden med 18 tegn til fylke og 10 tegn til kjønn, 'Sogn og Fjordane' er mye bredere enn 'Kvinner'. Slik det er, får fylke og kjønn 14 tegn hver, hvilket fører til at 'Sogn og Fjordane' skrives ut på to linjer.)

ROW= kan ikke løse problemet i eksempelet med fylke og kjønn. Men ROW= kan benyttes dersom det innerste elementet av forspalten er blank! Dette kan f.eks forekomme når vi krysser en variabel med en statistisk beregning og vi knytter blank tekst til statistikk-nøkkelordet (f.eks FYLKE*N=" ,). I dette tilfelle kan vi unngå at forspalten fordeles likt mellom fylke og N (antall). Med ROW=FLOAT vil nemlig forspaltebredden fordeles likt mellom de *ikke-blanke* elementer, så hvis forspaltebredden er 20 tegn vil fylke få alle 20, i motsetning til med ROW=CONSTANT der fylke vil få 10 og antall (selv om den er blank) vil få 10.

```
DATA test;
  INPUT fylke $ 1-2
        kjonn $ 4-4 ;
  CARDS;
14 1
14 2
15 2
16 1
;

PROC TABULATE DATA=test FC=' -----' NOSEPS F=8.;
  CLASS fylke kjonn ;
  FORMAT fylke $fylke.
        kjonn $kjonn. ;
  TABLE fylke=' '*N=' ',
        kjonn=' '
        /RTS=18 ;
  TITLE 'ROW=CONSTANT (standard) ' ;

PROC TABULATE DATA=test FC=' -----' NOSEPS F=8.;
  CLASS fylke kjonn ;
  FORMAT fylke $fylke.
        kjonn $kjonn. ;
  TABLE fylke=' '*N=' ',
        kjonn=' '
        /RTS=18 ROW=FLOAT;
  TITLE 'ROW=FLOAT ' ;
```

ROW=CONSTANT (standard)		
	Menn	Kvinner
Sogn og Fjordane	1	1
Møre og Romsdal	.	1
Sør-Trøndelag	1	.

ROW=FLOAT		
	Menn	Kvinner
Sogn og Fjordane	1	1
Møre og Romsdal	.	1
Sør-Trøndelag	1	.

RTS=antall tegn
 RTS (står for Row Title Space) er det antall tegn som settes av til forspalten. Hvis vi ikke angir noen RTS, blir forspaltebredden automatisk en fjerdedel av linjebredden.

STYLE=
 Spesifiserer stiler for bruk på html-tabeller. Se html-stiler, side 266.

FORMAT
 FORMAT-setningen brukes på samme måte i TABULATE som i de andre SAS-prosedyrene. Ved hjelp av FORMAT-setningen kan vi erstatte variabelverdier med mer forståelig *tekst* (f.eks at verdien 1 erstattes med teksten Menn og verdien 2 erstattes med teksten Kvinner). Vi kan også bruke FORMAT-setningen til å *gruppere* variabelverdier (lage aldersgrupper, inntektsgrupper etc). Det sistnevnte kan være spesielt nyttig i TABULATE, for på den måten kan vi bruke variable som tradisjonelt er analysevariable (alder, inntekt og andre numeriske, gjerne kontinuerlige variable) som grupperingsvariable.

KEYLABEL

Vil vi ha egne tekster på statistikknøkkelordene som brukes i tabellen kan vi bruke KEYLABEL. Nøkkelord er SUM, MEAN, N etc. KEYLABEL MEAN='Gjennomsnitt' ; Bemerk at dette også kan gjøres i TABLE-setningen. Vi kan også tekste totalvariabelen ALL med KEYLABEL.

KEYWORD

KEYWORD-setningen kan vi bruke i forbindeles med utskrift til html-filer til å endre på utseende til statistiske nøkkelord og ALL i forspalten eller kolonnehoder. Vi kan ha flere KEYWORD-setninger i samme program. Se html-stiler, side 266.

LABEL

Hvis variablene vi bruker har forklarende tekster ('labler'), er det disse som blir skrevet ut i tabellen. Dersom variablene *ikke* har labler, eller vi ønsker å *endre* lablene for anledningen, kan vi gjøre dette med LABEL-setningen. LABEL kjonn = 'Kjønn' ; Bemerk at dette også kan gjøres i TABLE-setningen.

WEIGHT

For å veie dataene i tabellen bruker vi WEIGHT-setningen. Vi må da ha en variabel på datasettet som inneholder vekten (oppblåsingsfaktoren). Vektvariabelen kan gjerne være et desimaltall. Vær oppmerksom på at N aldri veies, heller i ikke ved prosentregning (PCTN). Det samme gjelder for NMISS. Hvis vi skal ha veide prosenter, må vi bruke PCTSUM.

Bemerk at WEIGHT-setningen brukes annerledes i TABULATE enn i PROC FREQ! Bruker vi WEIGHT-setningen i PROC FREQ blir N og PCTN veid, i motsetning til i PROC TABULATE der disse *ikke* blir veid.

Når vektvariabelen er negativ, får den verdien 0 og observasjonen telles med i antallet. Når verdien er 0 telles observasjonen med i antallet (N). Dette kan overstyres med PROC TABULATE-option EXCLNPWGHTS som ekskluderer disse observasjonene fra tabellen. Er vektvariabelen MISSING, ekskluderes observasjonen fra tabellen. WEIGHT-setningen overstyres av WEIGHT-option i VAR-setningen.

FREQ

Hvis hver observasjon på datasettet skal telle for mer enn 1, kan vi bruke FREQ-setningen. I FREQ-setningen kan vi da angi variabelen hvis verdi inneholder det antall observasjoner denne observasjonen skal telle som. Dette betyr at N også økes med verdien til FREQ-variabelen. Er verdien på FREQ-variabelen 0 eller mindre, teller den ikke med i statistiske beregninger. FREQ trunkerer alle desimaler i FREQ-variabelen før den brukes. FREQ regner altså kun med heltall. Skal vi veie observasjonene i henhold til en variabel bør vi bruke WEIGHT istedenfor FREQ.

Eksempel på et enkelt tabellprogram med PROC FORMAT først for teksting av kodeverdier:

```
PROC FORMAT ;
  VALUE $yrkstat
    '1'   = 'Sysselsatt'
    '2'   = 'Arbeidssøker'
    '3'   = 'Utenfor arbeidsstyrken'
    OTHER = 'Uoppgitt'
;
PROC TABULATE DATA=aku.g1997k4 NOSEPS FC=' -----' MISSING;
  CLASS kjonn ystatus;
  VAR antpers;
  TABLE ALL='I alt' kjonn='
    ,
    antpers=' '*
    (ALL='I alt' ystatus='')*SUM=' '*F=ssb8_0v.
    N*F=ssb5_0v.
    /RTS=11 ;
  FORMAT ystatus $yrkstat.;
  TITLE 'AKU. Personer fordelt etter kjønn og yrkesstatus';
RUN;
```

AKU. Personer fordelt etter kjønn og yrkesstatus								
	I alt		Sysselsatt		Arbeidssøker		Utenfor arbeidsstyrken	N
I alt	3	151 503	2	206 005	74	144	871 353	21 712
1	1	585 397	1	187 074	41	454	356 868	10 982
2	1	566 106	1	018 931	32	690	514 485	10 730

Eksempel der vi prosentuerer en kolonne ut fra en annen ved hjelp av PCTSUM:

```
DATA bef;
  INPUT @01 Region      $CHAR1.
        @03 Aar         $CHAR4.
        @08 Menn        7.
        @16 Kvinner     7.
        ;
  BeggeKjonn = SUM(Menn,Kvinner);
CARDS;
1 1998 1064171 1106388
2 1998 307474 310572
3 1998 580073 584474
4 1998 233388 231059
;
RUN;

PROC FORMAT ;
  VALUE $region
    '1' = 'Øst'
    '2' = 'Sør'
    '3' = 'Vest'
    '4' = 'Nord'
    OTHER = 'Uoppgett'
  ;
RUN;

PROC TABULATE DATA=bef FC='-----' NOSEPS MISSING F=9.;
  CLASS Region ;
  VAR Menn Kvinner BeggeKjonn;
  TABLE ALL='I alt' Region=' '
    ,
    (BeggeKjonn='Begge kjønn' Menn Kvinner)*SUM=' '
    ,
    Kvinner=' '*PCTSUM<BeggeKjonn>='Prosent kvinner'*F=7.2
  /RTS=10 MISSTEXT='- ' BOX='Region';
  FORMAT region $region.;
  TITLE 'Tabell 2. Innbyggere etter region og kjønn. 1999' ;
RUN;
```

Resultatet:

Tabell 2. Innbyggere etter region og kjønn. 1999

Region	Begge kjønn	Menn	Kvinner	Prosent kvinner
I alt	4478497	2217140	2261357	50.49
Øst	2207164	1083508	1123656	50.91
Sør	631079	314292	316787	50.20
Vest	1175926	586075	589851	50.16
Nord	464328	233265	231063	49.76

Eksempler på prosentuering:

```
DATA bef ;
INPUT @01 Aar          $CHAR4.
      @06 Kjønn        $CHAR1.
      @08 Landsdel     $CHAR1.
      @10 Folk         7.
      ;
CARDS;
1998 1 1 983530
1998 1 2 388115
1998 1 3 388502
1998 1 4 424959
1998 2 1 1023172
1998 2 2 393788
1998 2 3 390083
1998 2 4 425450
1999 1 1 1002227
1999 1 2 395573
1999 1 3 392342
1999 1 4 426998
1999 2 1 1039899
1999 2 2 400544
1999 2 3 393624
1999 2 4 427290
;
```

```
PROC FORMAT ;
  VALUE $kjonnn
    '1' = 'Menn'
    '2' = 'Kvinner'
    OTHER = 'Uoppgitt'
  ;
  VALUE $region
    '1' = 'Øst'
    '2' = 'Sør'
    '3' = 'Vest'
    '4' = 'Nord'
    OTHER = 'Uoppgitt'
  ;
PROC TABULATE DATA=bef FC=' -----' NOSEPS MISSING F=numx7.1;
  CLASS aar kjonnn landsdel;
  VAR folk;
  TABLE ALL='I alt' aar=' '
    ,
    ALL='I alt' landsdel=' '
    ,
    (ALL='I alt' kjonnn=' ')*PCTSUM=' '*folk=' '
  /RTS=10 MISSTEXT='- ' ;
  FORMAT kjonnn $kjonnn.
    landsdel $region.
  ;
  TITLE 'Standardprosentuering' ;
PROC TABULATE DATA=bef FC=' -----' NOSEPS MISSING F=numx7.1;
  CLASS aar kjonnn landsdel;
  VAR folk;
  TABLE ALL='I alt' aar=' '
    ,
    ALL='I alt' landsdel=' '
    ,
    (ALL='I alt' kjonnn=' ')*REPPCTSUM=' '*folk=' '
  /RTS=10 MISSTEXT='- ' ;
  FORMAT kjonnn $kjonnn.
    landsdel $region.
  ;
  TITLE 'Totalprosent' ;
PROC TABULATE DATA=bef FC=' -----' NOSEPS MISSING F=numx7.1;
  CLASS aar kjonnn landsdel;
  VAR folk;
  TABLE ALL='I alt' aar=' '
    ,
    ALL='I alt' landsdel=' '
    ,
    (ALL='I alt' kjonnn=' ')*PAGEPCTSUM=' '*folk=' '
  /RTS=10 MISSTEXT='- ' ;
  FORMAT kjonnn $kjonnn.
    landsdel $region.
  ;
  TITLE 'Sideprosent' ;
RUN;

PROC TABULATE DATA=bef FC=' -----' NOSEPS MISSING F=numx7.1;
  CLASS aar kjonnn landsdel;
  VAR folk;
  TABLE ALL='I alt' aar=' '
    ,
    ALL='I alt' landsdel=' '
    ,
    (ALL='I alt' kjonnn=' ')*COLPCTSUM=' '*folk=' '
  /RTS=10 MISSTEXT='- ' ;
  FORMAT kjonnn $kjonnn.
    landsdel $region.
  ;
  TITLE 'Kolonnepresent' ;
PROC TABULATE DATA=bef FC=' -----' NOSEPS MISSING F=numx7.1;
  CLASS aar kjonnn landsdel;
  VAR folk;
  TABLE ALL='I alt' aar=' '
    ,
    ALL='I alt' landsdel=' '
    ,
    (ALL='I alt' kjonnn=' ')*ROWPCTSUM=' '*folk=' '
  /RTS=10 MISSTEXT='- ' ;
  FORMAT kjonnn $kjonnn.
    landsdel $region.
  ;
  TITLE 'Rekkeprosent' ;
RUN;
```


Resultatet fordelt over to spalter

Standardprosentuering			
I alt			

	I alt	Menn	Kvinner

I alt	100,0	49,5	50,5
Øst	45,5	22,3	23,2
Sør	17,7	8,8	8,9
Vest	17,6	8,8	8,8
Nord	19,2	9,6	9,6

Standardprosentuering			
1998			

	I alt	Menn	Kvinner

I alt	49,7	24,6	25,1
Øst	22,6	11,1	11,5
Sør	8,8	4,4	4,4
Vest	8,8	4,4	4,4
Nord	9,6	4,8	4,8

Standardprosentuering			
1999			

	I alt	Menn	Kvinner

I alt	50,3	24,9	25,4
Øst	23,0	11,3	11,7
Sør	8,9	4,4	4,5
Vest	8,8	4,4	4,4
Nord	9,6	4,8	4,8

Totalprosent			
I alt			

	I alt	Menn	Kvinner

I alt	100,0	49,5	50,5
Øst	45,5	22,3	23,2
Sør	17,7	8,8	8,9
Vest	17,6	8,8	8,8
Nord	19,2	9,6	9,6

Totalprosent			
1998			

	I alt	Menn	Kvinner

I alt	49,7	24,6	25,1
Øst	22,6	11,1	11,5
Sør	8,8	4,4	4,4
Vest	8,8	4,4	4,4
Nord	9,6	4,8	4,8

Totalprosent			
1999			

	I alt	Menn	Kvinner

I alt	50,3	24,9	25,4
Øst	23,0	11,3	11,7
Sør	8,9	4,4	4,5
Vest	8,8	4,4	4,4
Nord	9,6	4,8	4,8

Sideprosent			
I alt			

	I alt	Menn	Kvinner

I alt	100,0	49,5	50,5
Øst	45,5	22,3	23,2
Sør	17,7	8,8	8,9
Vest	17,6	8,8	8,8
Nord	19,2	9,6	9,6

Sideprosent			
1998			

	I alt	Menn	Kvinner

I alt	100,0	49,5	50,5
Øst	45,4	22,3	23,2
Sør	17,7	8,8	8,9
Vest	17,6	8,8	8,8
Nord	19,3	9,6	9,6

Sideprosent			
1999			

	I alt	Menn	Kvinner

I alt	100,0	49,5	50,5
Øst	45,6	22,4	23,2
Sør	17,8	8,8	8,9
Vest	17,5	8,8	8,8
Nord	19,1	9,5	9,5

Kolonneprosent			
I alt			

	I alt	Menn	Kvinner

I alt	100,0	100,0	100,0
Øst	45,5	45,1	45,9
Sør	17,7	17,8	17,7
Vest	17,6	17,7	17,4
Nord	19,2	19,4	19,0

Kolonneprosent			
1998			

	I alt	Menn	Kvinner

I alt	100,0	100,0	100,0
Øst	45,4	45,0	45,8
Sør	17,7	17,8	17,6
Vest	17,6	17,8	17,5
Nord	19,3	19,4	19,1

Kolonneprosent			
1999			

	I alt	Menn	Kvinner

I alt	100,0	100,0	100,0
Øst	45,6	45,2	46,0
Sør	17,8	17,8	17,7
Vest	17,5	17,7	17,4
Nord	19,1	19,3	18,9

Rekkeprosent			
I alt			

	I alt	Menn	Kvinner

I alt	100,0	49,5	50,5
Øst	100,0	49,0	51,0
Sør	100,0	49,7	50,3
Vest	100,0	49,9	50,1
Nord	100,0	50,0	50,0

Rekkeprosent			
1998			

	I alt	Menn	Kvinner

I alt	100,0	49,5	50,5
Øst	100,0	49,0	51,0
Sør	100,0	49,6	50,4
Vest	100,0	49,9	50,1
Nord	100,0	50,0	50,0

Rekkeprosent			
1999			

	I alt	Menn	Kvinner

I alt	100,0	49,5	50,5
Øst	100,0	49,1	50,9
Sør	100,0	49,7	50,3
Vest	100,0	49,9	50,1
Nord	100,0	50,0	50,0

3.11.20 PROC TRANSPOSE

PROC TRANSPOSE bruker vi til å transponere (vri, snu) SAS datasett. Vi gjør observasjonene om til variable og variablene om til observasjoner. Hvis vi ikke nevner noen variable i VAR-setningen, transponeres de **numeriske** variablene i SAS datasettet. For å få transponert karaktervariable, må vi bruke VAR-setningen.

Når vi transponerer SAS datasett, lager SAS en variabel med navn `_NAME_`. Denne inneholder navnet til variablene som er transponert. Har de opprinnelige variablene labler, lages også variabelen `_LABEL_` hvor disse legges. Når variabelen `_NAME_` finnes på et datasett som transponeres, benyttes denne for å lage variabelnavn på det transponerte datasettet (når ID-setningen ikke er i bruk).

```
PROC TRANSPOSE <options> ;  
  WHERE where-uttrykk ;  
  BY <DESCENDING> variabel-1 ... <<DESCENDING> variabel-n> <NOTSORTED>; ;  
  COPY variabel-1 ... <variabel-n> ;  
  ID variabel ;  
  IDLABEL variabel ;  
  VAR variabel-1 ... <variabel-n> ;
```

Options til PROC TRANSPOSE-setningen:

DATA=SAS datasett

Her oppgir vi navnet på SAS datasettet vi ønsker å transponere.

LABEL=variabel

Angir navnet på en variabel i det transponerte SAS datasettet som inneholder label (tekst) til de variablene som transponeres. Standardnavnet er `_LABEL_`.

LET

Tillater dubletter i en ID-variabel i SAS datasettet eller innen en BY-gruppe. Med LET transponeres den siste av disse dublettene, de andre utelates.

NAME=variabel

Angir navnet på en variabel i det transponerte SAS datasettet som inneholder navnene til de variablene som transponeres. Standardnavnet er `_NAME_`.

OUT=SAS datasett

Navnet på det transponerte SAS datasettet.

PREFIX=prefiks

Prefiks for de nye variablene som lages. Standardprefiks er COL, den første variabelen som lages, blir kalt COL1, den neste COL2 osv. Ved å angi et prefiks, brukes dette istedenfor COL.

SAS-setninger til PROC TRANSPOSE:

WHERE

Selekterer hvilke observasjoner som skal transponeres. Bare de som tilfredsstiller WHERE-betingelsen tas med. Beskrivelse av WHERE-setningen finnes på side 91.

BY

Transponeringen skal utføres for hver verdi av BY-variablene. Datasettet må være sortert etter BY-variablene på forhånd (med mindre vi bruker sorteringsoption NOTSORTED).

COPY

Kopier variablene i COPY-setningen uten å transponere dem. Det lages således like mange observasjoner i det nye SAS datasettet som det var i det gamle. Dette kan påvirke resultatet av transponeringen.

ID

ID-variabelen må inneholde data som skal brukes til å navngi de nye variablene. Denne ID-variabelen må ha unike verdier innen SAS datasettet eller hver BY-gruppe (med mindre LET-option brukes). Inneholder ID-variabelen tall gis variabelen prefikset `_` (understrek) hvis ikke annet prefiks er angitt. Plusstegn endres til P, minustegn til N og desimalpunktum til D. Ugyldige tegn endres til understrek. Overstiger variabelens verdi 32 tegn kuttet det fra høyre ned til 32 tegn. Observasjoner med MISSING ID-variabel utelates og det gis en melding om dette i SASlog.

IDLABEL

IDLABEL-variabelens innhold brukes som labler for til de nye variablene hvis navn hentes fra ID-variabelen.

VAR

Her angir vi hvilke variable vi ønsker å transponere. Må brukes når vi ønsker å transponere karaktervariable. Utelater vi VAR-setningen, transponeres alle numeriske variabel, unntatt de som er nevnt i BY, COPY, ID eller IDLABEL-setningene, disse transponeres aldri.

Eksempler:

Vi bruker konsumprisindeksen som eksempel. Vi har den lagret med en observasjon pr. måned. Først ønsker vi å utføre en standard transponering. Det er bare indeksvariabelen som er definert numerisk og derfor er det bare denne som transponeres.

```
DATA kpimnd;
  INFILE CARDS MISSOVER PAD;
  INPUT  @01 aar          $CHAR4.
         @05 mnd         $CHAR3.
         @01 aarmnd      $CHAR7.
         @09 indeks      5.1
        ;
  LABEL aar      ='År'
        mnd      ='Måned'
        aarmnd   ='År og måned'
        indeks   ='Konsumprisindeks'
        ;
CARDS;
1994JAN 250.4
1994FEB 251.4
1994MAR 252.7
1994APR 252.9
1994MAI 253.1
1994JUN 253.7
1994JUL 254.2
1994AUG 254.2
1994SEP 255.4
1994OKT 255.6
1994NOV 255.8
1994DES 255.7
1995JAN 257.0
1995FEB 257.9
1995MAR 259.4
1995APR 259.6
1995MAI 259.9
1995JUN 260.6
1995JUL 260.4
1995AUG 259.9
1995SEP 261.4
1995OKT 261.4
1995NOV 261.2
1995DES 261.2
1996JAN 260.0
1996FEB 260.1
1996MAR 261.3
1996APR 262.2
1996MAI 262.6
1996JUN 262.9
        ;
PROC TRANSPOSE DATA=kpimnd OUT=kpi;
PROC PRINT DATA=kpi;
  TITLE 'Konsumprisindeksen 1994-1996. Standard transponering';
RUN;
```

Resultatet gir oss et datasett med en observasjon med en variabel pr. måned:

Konsumprisindeksen 1994-1996. Standard transponering												
OBS	_NAME_	_LABEL_		COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	
1	INDEKS	Konsumprisindeks		250.4	251.4	252.7	252.9	253.1	253.7	254.2	254.2	
OBS	COL9	COL10	COL11	COL12	COL13	COL14	COL15	COL16	COL17	COL18	COL19	COL20
1	255.4	255.6	255.8	255.7	257	257.9	259.4	259.6	259.9	260.6	260.4	259.9
OBS	COL21	COL22	COL23	COL24	COL25	COL26	COL27	COL28	COL29	COL30		
1	261.4	261.4	261.2	261.2	260	260.1	261.3	262.2	262.6	262.9		

Disse variablene navngis som vi ser fortløpende med prefikset COL. For å få bedre navn på variablene kan vi bruke PREFIX option:

```
PROC TRANSPOSE DATA=kpimnd OUT=kpi PREFIX=IND_;
PROC PRINT DATA=kpi;
  TITLE 'Konsumprisindeksen 1994-1996. Standard transponering med PREFIX';
RUN;
```

Variablene får nå prefikset IND_:

```
Konsumprisindeksen 1994-1996. Standard transponering med PREFIX

OBS  _NAME_      _LABEL_      IND_1  IND_2  IND_3  IND_4  IND_5  IND_6  IND_7  IND_8  IND_9
1    INDEKS    Konsumprisindeks  250.4  251.4  252.7  252.9  253.1  253.7  254.2  254.2  255.4

OBS  IND_10  IND_11  IND_12  IND_13  IND_14  IND_15  IND_16  IND_17  IND_18  IND_19  IND_20
1    255.6   255.8   255.7   257     257.9   259.4   259.6   259.9   260.6   260.4   259.9

OBS  IND_21  IND_22  IND_23  IND_24  IND_25  IND_26  IND_27  IND_28  IND_29  IND_30
1    261.4   261.4   261.2   261.2   260     260.1   261.3   262.2   262.6   262.9
```

Både bedre og sikrere blir det ved å bruke informasjon fra datafilen til å lage variabelnavn. Da kan vi få år og måned i variabelnavnene. Vi bruker ID-setningen for å angi hvilken variabel som skal brukes for å navngi de nye transponerte variablene:

```
PROC TRANSPOSE DATA=kpimnd OUT=kpi PREFIX=I;
  ID aarmnd;
PROC PRINT DATA=kpi;
  TITLE 'Konsumprisindeksen 1994-1996. Standard transponering med prefix og ID';
RUN;
```

Resultatet gir oss en fil med meningsfylte variabelnavn med datafilen som grunnlag:

```
Konsumprisindeksen 1994-1996. Standard transponering med prefix og ID

OBS  _NAME_      _LABEL_      I1994JAN  I1994FEB  I1994MAR  I1994APR  I1994MAI  I1994JUN
1    INDEKS    Konsumprisindeks  250.4     251.4     252.7     252.9     253.1     253.7

OBS  I1994JUL  I1994AUG  I1994SEP  I1994OKT  I1994NOV  I1994DES  I1995JAN  I1995FEB  I1995MAR
1    254.2     254.2     255.4     255.6     255.8     255.7     257       257.9     259.4

OBS  I1995APR  I1995MAI  I1995JUN  I1995JUL  I1995AUG  I1995SEP  I1995OKT  I1995NOV  I1995DES
1    259.6     259.9     260.6     260.4     259.9     261.4     261.4     261.2     261.2

OBS  I1996JAN  I1996FEB  I1996MAR  I1996APR  I1996MAI  I1996JUN
1    260       260.1     261.3     262.2     262.6     262.9
```

Nå ønsker vi oss et SAS datasett med en observasjon pr. år. Vi bruker BY-setningen med årstallsvariabelen. Det gir oss en observasjon pr. år:

```
PROC TRANSPOSE DATA=kpimnd OUT=kpi PREFIX=IND;
  BY aar;
PROC PRINT DATA=kpi;
  TITLE 'Konsumprisindeksen 1994-1996. Transponering med BY';
RUN;
```

Nå får vi et fint organisert SAS datasett:

```
Konsumprisindeksen 1994-1996. Transponering med BY

OBS  AAR      _NAME_      _LABEL_      IND1  IND2  IND3  IND4
1    1994    INDEKS    Konsumprisindeks  250.4  251.4  252.7  252.9
2    1995    INDEKS    Konsumprisindeks  257.0  257.9  259.4  259.6
3    1996    INDEKS    Konsumprisindeks  260.0  260.1  261.3  262.2

OBS  IND5  IND6  IND7  IND8  IND9  IND10  IND11  IND12
1    253.1  253.7  254.2  254.2  255.4  255.6  255.8  255.7
2    259.9  260.6  260.4  259.9  261.4  261.4  261.2  261.2
3    262.6  262.9
```

Variablene IND1 til IND12 inneholder indekser for de tolv månedene. Siden vi ikke bruker data for å navngi variablene kan det skje at IND1 ikke vil inneholde tall for januar, IND2 for februar osv. Hadde vi manglet

observasjoner for f.eks. januar og februar 1994, ville data for mars 1994 havne i variabelen IND1. Det ser vi ved å kjøre samme program på nytt med FIRSTOBS=3 option:

```
PROC TRANSPOSE DATA=kpimnd (FIRSTOBS=3) OUT=kpi PREFIX=IND;
  BY aar;
PROC PRINT DATA=kpi;
  TITLE 'Konsumprisindeksen 1994-1996. Transponering med BY';
RUN;
```

Dette viser at vi bør bruke PROC TRANSPOSE med forsiktighet:

```
Konsumprisindeksen 1994-1996. Transponering med BY
```

OBS	AAR	_NAME_	_LABEL_	IND1	IND2	IND3	IND4
1	1994	INDEKS	Konsumprisindeks	252.7	252.9	253.1	253.7
2	1995	INDEKS	Konsumprisindeks	257.0	257.9	259.4	259.6
3	1996	INDEKS	Konsumprisindeks	260.0	260.1	261.3	262.2

OBS	IND5	IND6	IND7	IND8	IND9	IND10	IND11	IND12
1	254.2	254.2	255.4	255.6	255.8	255.7	.	.
2	259.9	260.6	260.4	259.9	261.4	261.4	261.2	261.2
3	262.6	262.9	.	.	.	.	.	.

Ved å bruke informasjon som ligger på datasettet når vi skal lage de nye variablene, gjør vi samtidig programmet sikrere. Vi unngår slike feil som vist over:

```
PROC TRANSPOSE DATA=kpimnd (FIRSTOBS=3) OUT=kpi ;
  BY aar;
  ID mnd;
PROC PRINT DATA=kpi;
  TITLE 'Konsumprisindeksen 1994-1996. Transponering med BY og ID';
RUN;
```

Nå blir resultatet riktig:

```
Konsumprisindeksen 1994-1996. Transponering med BY og ID
```

OBS	AAR	_NAME_	_LABEL_	JAN	FEB	MAR	APR
1	1994	INDEKS	Konsumprisindeks	.	.	252.7	252.9
2	1995	INDEKS	Konsumprisindeks	257	257.9	259.4	259.6
3	1996	INDEKS	Konsumprisindeks	260	260.1	261.3	262.2

OBS	MAI	JUN	JUL	AUG	SEP	OKT	NOV	DES
1	253.1	253.7	254.2	254.2	255.4	255.6	255.8	255.7
2	259.9	260.6	260.4	259.9	261.4	261.4	261.2	261.2
3	262.6	262.9	.	.	.	.	.	.

For å oppnå ønsket resultat, kan det hende vi må transponere to ganger. Vi ønsker et datasett med en observasjon pr. måned og en variabel pr. år:

```
PROC TRANSPOSE DATA=kpimnd OUT=kpi1 ;
  BY aar;
  ID mnd;
PROC TRANSPOSE DATA=kpi1 OUT=kpi2 PREFIX=i_ NAME=mnd;
  ID aar;
PROC PRINT DATA=kpi2;
  TITLE 'Konsumprisindeksen 1994-1996. Dobbelt transponering med BY og ID';
RUN;
```

Den første transponeringen er lik det forrige eksempelet. Den andre transponerer så dette resultatet og får en ny organisering av dataene:

```
Konsumprisindeksen 1994-1996. Dobbelt transponering med BY og ID
```

OBS	MND	I_1994	I_1995	I_1996
1	JAN	250.4	257.0	260.0
2	FEB	251.4	257.9	260.1
3	MAR	252.7	259.4	261.3
4	APR	252.9	259.6	262.2
5	MAI	253.1	259.9	262.6
6	JUN	253.7	260.6	262.9
7	JUL	254.2	260.4	.
8	AUG	254.2	259.9	.
9	SEP	255.4	261.4	
10	OKT	255.6	261.4	
11	NOV	255.8	261.2	
12	DES	255.7	261.2	

Neste eksempel viser at vi ved transponere to ganger kan komme tilbake til samme resultat. Det kan være nyttig om vi finner det lettere å gjøre beregninger med en annen organisering av dataene, men ønsker å ende opp med samme organisering som vi startet med.

```
PROC TRANSPOSE DATA=kpimnd OUT=kpi1 ;
  BY aar;
  ID mnd;
PROC TRANSPOSE DATA=kpi1 OUT=kpi2 (RENAME=(coll=indeks)) NAME=mnd;
  BY aar;
PROC PRINT DATA=kpi2;
  TITLE 'Konsumprisindeksen 1994-1996. Dobbel transponering';
RUN;
```

Resultatet er likt utgangspunktet:

Konsumprisindeksen 1994-1996. Dobbel transponering

OBS	AAR	MND	INDEKS
1	1994	JAN	250.4
2	1994	FEB	251.4
3	1994	MAR	252.7
4	1994	APR	252.9
5	1994	MAI	253.1
6	1994	JUN	253.7
7	1994	JUL	254.2
8	1994	AUG	254.2
9	1994	SEP	255.4
10	1994	OKT	255.6
11	1994	NOV	255.8
12	1994	DES	255.7
13	1995	JAN	257.0
14	1995	FEB	257.9
15	1995	MAR	259.4
16	1995	APR	259.6
17	1995	MAI	259.9
18	1995	JUN	260.6
19	1995	JUL	260.4
20	1995	AUG	259.9
21	1995	SEP	261.4
22	1995	OKT	261.4
23	1995	NOV	261.2
24	1995	DES	261.2
25	1996	JAN	260.0
26	1996	FEB	260.1
27	1996	MAR	261.3
28	1996	APR	262.2
29	1996	MAI	262.6
30	1996	JUN	262.9
31	1996	JUL	.
32	1996	AUG	.
33	1996	SEP	.
34	1996	OKT	.
35	1996	NOV	.
36	1996	DES	.

3.11.21 PROC UNIVARIATE

PROC UNIVARIATE utfører statistiske beregninger på numeriske variable. Beregningene som utføres er til dels overlappende med PROC MEANS, men PROC UNIVARIATE har flere beregninger, blant annet medianverdi, prosentiler og hyppigst forekommende verdi (MODE).

```
PROC UNIVARIATE <options> ;
  WHERE where-uttrykk ;
  BY <DESCENDING> variabel-1 ... <<DESCENDING> variabel-n> <NOTSORTED>;
  CLASS variabel-1 <(variabel-options)> < variabel-2 <(variabel-options)>> </options> ;
  FREQ variabel ;
  HISTOGRAM variable / options ;
  ID variable ;
  INSET statistikk-nøkkelord<='tekst'> <DATA= SAS dataset> </ options> ;
  OUTPUT <OUT=dataset> <utdataspesifikasjoner> ;
  PROBLOT variable </ options> ;
  QQPLOT variable </ options> ;
  VAR variable ;
  WEIGHT variabel ;
```

Options til PROC UNIVARIATE-setningen:

DATA=SAS datasett

Her oppgir vi navnet på SAS datasettet vi ønsker å behandle.

EXCLNPWGT

Selekter kun observasjoner som har positive vekter. Se også WEIGHT-setningen.

FREQ

Bevirker at det lages en frekvensliste. Denne kan bli veldig lang.

MODES

Det kan være flere forskjellige verdier som forekommer hyppigst. Vanligvis vil SAS vise den laveste av dem. Med MODES vil vi få en oversikt over alle verdier som forekommer hyppigst.

NEXTROBS

Angir hvor mange av de ekstreme observasjonene som skal listes ut. Standard er 5.

NEXTRVAL

Angir hvor mange av ekstremverdiene som skal listes ut. Standard er 5.

NOBYPLOT

Undertrykker boks-diagram ved siden av hverandre ved bruk av BY-grupper.

NOPRINT

Undertrykker utskrift, brukes sammen med OUTPUT.

NORMAL

Utfører en testberegning på hypotesen at dataene har en normalfordeling. Beregner også sannsynligheten for en mer ekstrem verdi for testberegningene.

PCTLDEF=n

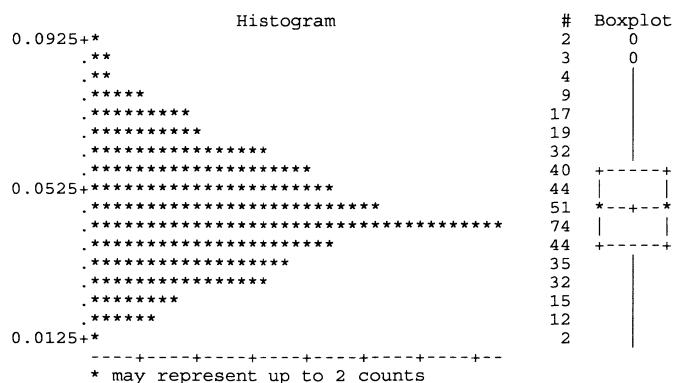
Velger metode for å beregne prosentiler. Det er 5 metoder (1-5), disse er beskrevet nærmere i SAS Procedures Guide. Standardverdi er 5.

PLOT

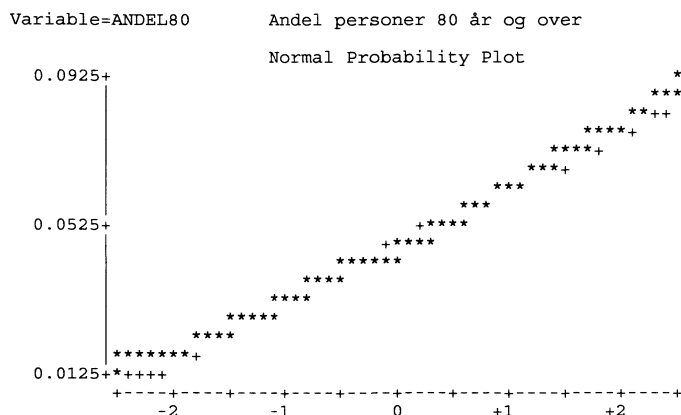
Lager tre grafer (diagram): stamme- og blad (stem-and-leaf) (hvis antall observasjoner pr. intervall er mindre enn 49, ellers lages et histogram), boks og normalfordeling.

```
Stem Leaf
 6 5
 5
 4 6
 3
 2 56
 1 23
 0 113334445679
-----+-----+
Multiply Stem.Leaf by 10**+4
```

Dette er et stamme- og blad-diagram som viser antall innbyggere i kommunene i Østfold. Langs den vertikaleaksen (stammen) er antall innbyggere (oppgitt i 10000). Horisontalt fordeles kommunene i sine intervaller (blader). Et tall i grafen representerer en kommune og tallet sier hvor mange innbyggere kommunen har. Vi ser at det er en kommune i gruppe 6 (60000-69999 innbyggere). Tallet 5 forteller at denne kommunen har 5000 innbyggere i tillegg til de 60000 som den vertikaleaksen indikerer, altså 65000 (avrundet). Videre er det en kommune med ca. 46000 innbyggere, 1 med ca. 25000 og 1 med ca. 26000. Det er 12 kommuner i Østfold med mindre enn 10000 innbyggere.



Dette er et histogram og et boks-diagram som viser hvilken andel personer 80 år og over utgjør av befolkningen i kommunene. Histogrammet gir oss en rask oversikt over fordelingene av andelene. Boks-diagrammet viser oss fordelingen på en litt annen måte. Den øverste horisontale streken, viser hvor 75. prosentil er, mens den nederste viser 25. prosentil. Den midterste horisontale streken viser medianverdien. Plusstegnet inne i boksen viser gjennomsnittsverdien. Den vertikale streken vises som 0, når verdiene overskrider 1.5 ganger avstanden mellom 25. og 75. prosentil, og med * når verdiene overskrider 3 ganger avstanden. Er de innenfor, markeres det med |.



Det siste diagrammet som lages viser hvor nær normalfordeling dataene er. Plusstegnene (+) utgjør en linje basert på gjennomsnitt og standardavvik. Asteriskene (*) viser dataverdiene. Når dataene er normalfordelt, vil plussene og asteriskene være sammenfallende.

Med disse grafene kan vi danne oss et raskt bilde av datamaterialet.

PLOTSIZE=

Angir hvor mange linjer et diagrammet skal bruke. Standard er verdien til PAGESIZE-option.

ROUND=avrundingsfaktor-1 ... <avrundingsfaktor-n>

Angir avrundingsfaktorer for variabelverdier. Velger vi en avrundingsfaktor, vil denne bli brukt på alle variable, velger vi flere refererer de til variable listet i VAR-setningen i samme rekkefølge.

VARDEF= divisor

Angir divisor som brukes til å regne ut varianser. Gyldige verdier er:

DF	N-1 skal brukes som divisor (standard)
N	N skal brukes som divisor
WDF	Summen av vektene - 1 skal brukes som divisor
WEIGHT	Summen av vektene skal brukes som divisor

Andre options: ALL, ANNOTATE=, CIBASIC, CIPCTLDF, CIPCTLNORMAL, GOUT=, LOCCOUNT, MU0=, ROBUSTSCALE, TRIMMED= og WINSORIZED.

SAS-setninger til PROC UNIVARIATE:

WHERE

Selekterer hvilke observasjoner det skal lages statistikk for. Bare de som tilfredsstiller WHERE-betingelsen tas med. Beskrivelse av WHERE-setningen finnes på side 91.

BY

Det beregnes deskriptiv statistikk for hver verdi av BY-variablene. Datasettet må være sortert etter BY-variablene på forhånd (med mindre vi bruker sorteringsoption NOTSORTED).

CLASS

Det beregnes statistikk for grupperingsvariablene. Options til CLASS-setningen:

KEYLEVEL='verdi-1' | ('verdi-1' 'verdi-2')

Brukes sammen med HISTOGRAM-, PROBLOT- eller QQPLOT-setningen for å angi hvilke verdier av grupperingsvariable som skal være basis for beregning av verdier på x-aksen.

MISSING

Angis hvis vi vil at observasjoner som mangler verdi i *grupperingsvariable* likevel skal være med i tabellen. (Dersom MISSING ikke er med, blir observasjoner med missing i grupperingsvariablene utelatt.) MISSING-option har ingen innvirkning på behandlingen av missing i *analysevariable*.

ORDER=rekkefølge

Angir hvilken rekkefølge verdiene til grupperingsvariable skal skrives ut i. Velger vi ORDER=FORMATTED blir de sortert alfabetisk etter teksten i formatet som eventuelt brukes istedenfor etter koden. ORDER=FREQ sørger for at den verdi med flest observasjoner skrives først. ORDER=DATA gjør at verdiene beholder den rekkefølgen de har i datasettet. Den første forekomsten av verdiene bestemmer rekkefølgen. Standardverdi er ORDER= INTERNAL som skriver ut verdiene i sortert rekkefølge.

FREQ

Hvis hver observasjon på datasettet skal telle for mer enn 1, kan vi bruke FREQ-setningen. I FREQ-setningen kan vi da angi variabelen hvis verdi inneholder det antall observasjoner denne observasjonen skal telle som. Dette betyr at N også økes med verdien til FREQ-variabelen.

FREQ trunkerer alle desimaler i FREQ-variabelen før den brukes. FREQ regner altså kun med heltall.

Skal vi veie observasjonene i henhold til en variabel bør vi bruke WEIGHT istedenfor FREQ.

HISTOGRAM

Lager histogram som sendes til grafikk-vinduet. Noen options:

BARWIDTH=	Angir bredden for hver kolonne i prosent av tilgjengelig plass.
CAXIS=	Fargen til aksene i histogrammet.
CBARLINE=	Fargen til kantlinjene til kolonnene.
CFILL=	Fargen kolonnene skal fylles med.
CFRAME=	Bakgrunnsfarge i selve histogrammet.
CFRAMESIDE=	Bakgrunnsfarge til tekst for grupperingsvariable definert i CLASS-setningen.
CGRID=	Fargen til hjelpelinjer i histogrammet.
CTEXT=	Fargen til tekst i histogrammet.
GRID	Angir at hjelpelinjer skal vises i histogrammet.

Andre options: ANNOKEY, ANNOTATE, BETA, CFRAME TOP, CHREF, CPROP, CURVELEGEND, CV, CVREF, DESCRIPTION, EXPONENTIAL, FONT, GAMMA, HAXIS, HEIGHT, HMINOR, HOFFSET, HREF, HREFLABELS, HREFLABPOS, INFONT, INHEIGHT, INTERBAR, INTERTILE, KERNEL, LEGEND, LGRID, LHREF,

LINELEGEND, LOGNORMAL, LVREF, MAXNBIN, MAXSIGMAS, MIDPERCENTS, MIDPOINTS, NAME, NCOL, NCOLS, NOBARS, NOCHART, NOFRAME, NOHLABEL, NOPLOT, NORMAL, NOVLABEL, NOVTICK, NROW, NROWS, OUTHISTOGRAM, PFILL, RTINCLUDE, SPECLEGEND, TILELEGLABEL, TURNVLABELS, VAXIS, VAXISLABEL, VMINOR, VOFFSET, VREF, VREFLABELS, VREFLABPOS, VSCALE, WAXIS, WBARLINE, WEIBULL, WEIBULL2 og WGRID.

ID

ID-variablenes verdier brukes for identifisere hvilke observasjoner som har de n høyeste og n laveste verdiene for analysevariablene. Hvor mange ekstremverdier som skal skrives ut angis med NEXTROBS-option i PROC UNIVARIATE-setningen. Disse ekstremverdiene skrives ut. Hvis det lages et nytt datasett, skrives verdien til ID-variabelen på første observasjon eller første observasjon innen en BY-gruppe til datasettet.

INSET

Legger statistikk inn i egne bokser grafikkvinduet. Det lages en boks pr. INSET-setning. Brukes sammen med HISTOGRAM-setningen. Disse statistikk-nøkkelordene kan vi bruke: BETA, CSS, CV, EXPONENTIAL, GAMMA, GINI, KERNEL, KERNEL1, KERNEL2, KERNEL3, KERNEL4, KERNEL5, KURTOSIS, LOGNORMAL, MAD, MAX, MEAN, MEDIAN, MIN, MODE, MSIGN, N, NMISS, NOBS, NORMAL, P1, P10, P5, P90, P95, P99, PNORMAL, PROBM, PROBN, PROBS, PROBT, Q1, Q3, QN, QRANGE, RANGE, SIGNRANK, SKEWNESS, SN, STD, STDMEAN, STD_GINI, STD_MAD, STD_QN, STD_QRANGE, STD_SN, SUM, SUMWGT, T, USS, VAR, WEIBULL og WEIBULL2.

Options til INSET-setningen:

CFILL=	Bakgrunnsfarge for statistikkboksen.
CFILLH=	Bakgrunnsfarge for overskrift i statistikkboksen.
CFRAME=	Farge til rammen til statistikkboksen.
CHEADER=	Farge til overskriftstekst.
CSHADOW=	Farge til skyggelegging av rammen.
CTEXT=	Farge til tekst i statistikkboksen.
FONT=	Font til tekst i statistikkboksen.
FORMAT=	Angir utskriftsformat for tallene i statistikkboksen.
HEADER=	Overskrift i statistikkboksen, maks. 40 tegn.
HEIGHT=	Angir tekstens høyde.
NOFRAME=	Fjerner ramme rundt statistikkboksen.
POSITION=	Angir plassering av statistikkboksen. Vi kan plassere boksen over histogrammet med TM, under med BM, til venstre for med LM og til høyre for med HM. Den kan plasseres inne i histogrammet etter kompassanvisning (N, NE, E, SE, S, SW, W eller NW) eller med koordinater mellom 0 og 100 horisontal og vertikalt, f.eks. vil POSITION=(1,1) plassere boksen nederst til venstre i selve histogrammet. Koordinatet som settes viser vanligvis til nederste venstre hjørnet av boksen.
REFPOINT=	Angir hvilket hjørne et koordinat definert med POSITION= skal gjelde for. Vi kan velge mellom de fire hjørnene, BL (venstre nederst, standard), BR (høyre nederst), TL (venstre øverst) og TR (høyre øverst).

OUTPUT

Brukes når vi ønsker å legge resultatet av PROC UNIVARIATE ut på et SAS datasett. UNIVARIATE kan utføre mange statistiske beregninger, her er de som også beregnes med PROC MEANS: CSS, CV, KURTOSIS, MAX, MEAN, MIN, N, NMISS, PROBT (tilsvarer PRT i PROC MEANS), RANGE, SKEWNESS, STD, STDERR, SUM, SUMWGT, T, USS, og VAR, for nærmere beskrivelse, se side 269. Videre beregnes:

Q3	Øvre kvartil (75. prosentil)
MEDIAN	Medianverdi (50. prosentil)
Q1	Nedre kvartil (25. prosentil)
QRANGE	Avstanden mellom øvre og nedre kvartil, (Q3-Q1)
P1	1. prosentil
P5	5. prosentil
P10	10. prosentil
P90	90. prosentil
P95	95. prosentil
P99	99. prosentil

MODE Den hyppigst forekommende verdi. Hvis flere verdier forekommer like ofte, velges den laveste.
Dessuten beregnes MSIGN, NORMAL, PROBM, PROBN, PROBS og SIGNRANK.

Vi kan få beregnet andre prosentiler enn de som er standard. Til dette bruker vi PCTLPTS. For å navngi variablene for prosentiler på utdatasettet, kan vi bruke PCTLPRE for å angi prefikser og/eller PCTLNAME for suffikser. Vi kan også få beregnet disse: GINI, MAD, QN, SN, STD_GINI, STD_MAD, STD_MEAN, STD_QN, STD_QRANGE og STD_SN.

Eksempel:

```
PROC UNIVARIATE DATA=finans.komm93 NOPRINT ;
BY fylke;
VAR sumint93 ant_ialt;
OUTPUT OUT=komstat N=n
      PCTLPTS=33 66 PCTLPRE=p_int_ p_inb_
      MEDIAN=md_int93 md_innb
      ;
PROC PRINT DATA=komstat;
  TITLE 'Diverse statistikker';
```

Diverse statistikker

OBS	FYLKE	N	MD_INT93	MD_INNB	P_INT_33	P_INT_66	P_INB_33	P_INB_66
1	01	18	13322.5	5392.5	12857	14262	3965	9194
2	02	22	13396.5	13990.5	13059	13895	11420	16796
3	03	1	17881.0	483401.0	17881	17881	483401	483401
4	04	22	15133.5	5492.0	14016	16684	4259	7299
5	05	26	15983.5	5432.5	15274	16876	3430	6341
6	06	21	14970.0	4637.0	14291	16854	3442	12196
7	07	15	13641.0	6981.0	13013	13984	4073	9234
8	08	18	15547.5	4317.0	14888	16550	2740	6661
9	09	15	15499.0	3355.0	15171	17405	1862	4322
10	10	15	14581.0	5768.0	14054	15092	2236	8091
11	11	26	15981.0	6679.5	14229	18216	3072	12605
12	12	34	16847.0	4317.5	15799	18244	3224	6985
13	14	26	18713.5	3137.0	17007	19992	2686	5087
14	15	38	16338.5	4579.0	15178	17907	3060	6467
15	16	25	16292.0	4315.0	14866	17635	4057	5397
16	17	24	17820.5	2633.5	17212	18859	1638	3756
17	18	45	20441.0	2465.0	17507	21831	1738	4790
18	19	25	20828.0	2677.0	19386	24155	1939	3473
19	20	19	24560.0	2788.0	21918	30286	1578	3272

PROBPLOT

Lager et sannsynlighets-diagram og viser det i grafikkvinduet. Vi kan bruke disse options (noen av dem er beskrevet under HISTOGRAM- eller INSET-setningene): ANNOKEY, ANNOTATE, BARWIDTH, BETA, CAXIS, CBARLINE, CFILL, CFRAME, CFRAMESSIDE, CFRAMESTOP, CGRID, CHREF, CPROP, CTEXT, CURVELEGEND, CV, CVREF, DESCRIPTION, EXPONENTIAL, FONT, GAMMA, GRID, HAXIS, HEIGHT, HMINOR, HOFFSET, HREF, HREFLABELS, HREFLABPOS, INFONT, INHEIGHT, INTERBAR, INTERTILE, KERNEL, LABEL, LEGEND, LGRID, LHREF, LINELEGEND, LOGNORMAL, LVREF, MAXNBIN, MAXSIGMAS, MIDPERCENTS, MIDPOINTS, NADJ, NAME, NCOL, NCOLS, NOCHART, NOFRAME, NOHLABEL, NOLINELEGEND, NOOBSLEGEND, NOPLOT, NORMAL, NOVLABEL, NOVTICK, NROW, NROWS, OUTHISTOGRAM, PCTLORDER, PFILL, PPSYMBOL, PROBSYMBOL, QQSYMBOL, RANKADJ, RTINCLUDE, SPECLEGEND, TILELEGLABEL, TURNVLABELS, VAXIS, VAXISLABEL, VMINOR, VOFFSET, VREF, VREFLABELS, VREFLABPOS, VSCALE, WAXIS, WBARLINE, WEIBULL, WEIBULL2 og WGRID.

QQPLOT

Lager et kvantil-kvantil-diagram og viser det i grafikkvinduet. Vi kan bruke de samme options som i PROBPLOT-setningen, unntatt LABEL.

VAR

I VAR-setningen lister vi opp analysevariablene våre. Disse må være numeriske. Utelater vi VAR-setningen utføres beregningene på alle numeriske variable.

WEIGHT

For å veie dataene bruker vi WEIGHT-setningen. Vi må da ha en variabel på datasettet som inneholder vekten (oppblåsingsfaktoren). Vektvariabelen kan gjerne være et desimaltall.

Når vektvariabelen er negativ, får den verdien 0 og observasjonen telles med i antallet. Når verdien er 0 telles observasjonen med i antallet (N). Dette kan overstyres med PROC

UNIVARIATE-option EXCLNPWGT som ekskluderer disse observasjonene fra beregningene. Er vektvariabelen MISSING, ekskluderes observasjonen fra beregningene.

Eksempel:

Vi har et SAS datasett med en observasjon pr. kommune og ønsker å vite mer om kommunenes inntekter pr. 1000 innbyggere (sumint93). En liten PROC UNIVARIATE gir oss mye informasjon.

```
PROC UNIVARIATE DATA=finans.komm93 ;
ID knr;
VAR sumint93 ;
```

Univariate Procedure			
Variable=SUMINT93		Kommunale inntekter, tusen kr. pr.innb.	
Moments			
N	435	Sum Wgts	435
Mean	17940.71	Sum	7804208
Std Dev	5100.004	Variance	26010040
Skewness	1.438883	Kurtosis	3.71116
USS	1.513E11	CSS	1.129E10
CV	28.42699	Std Mean	244.5265
T:Mean=0	73.36919	Pr> T	0.0001
Num ^= 0	435	Num > 0	434
M(Sign)	216.5	Pr>= M	0.0001
Sgn Rank	47414	Pr>= S	0.0001
Quantiles (Def=5)			
100% Max	45360	99%	33937
75% Q3	20073	95%	28091
50% Med	16591	90%	24701
25% Q1	14315	10%	13207
0% Min	-1806	5%	12857
		1%	12106
Range	47166		
Q3-Q1	5758		
Mode	13188		
Extremes			
Lowest	ID	Highest	ID
-1806 (1665	)	33937 (1029	)
7660 (0940	)	36198 (0941	)
11894 (0625	)	38333 (1232	)
12093 (0135	)	38396 (1151	)
12106 (0228	)	45360 (1252	)

3.11.22 PROC UPLOAD

PROC UPLOAD bruker vi for å overføre data fra en klientmaskin til en tjenermaskin. Vanligvis medfører det å overføre data fra Windows NT til unix, men det kan også være fra en unix-maskin til en annen. PROC UPLOAD utføres på tjenermaskinen, det betyr at programmet må eksekveres med RSUBMIT-kommandoen eller at PROC UPLOAD må stå inne i en RSUBMIT-blokk for eksekvering med SUBMIT-kommandoen.

```
PROC UPLOAD <options> ;
WHERE where-uttrykk ;
EXCLUDE datasettliste </MEMTYPE=datasett-type | ENTRYTYPE=katalog-type> ;
SELECT datasettliste </MEMTYPE=datasett-type | ENTRYTYPE=katalog-type>> ;
```

Options til PROC UPLOAD-setningen:

AFTER= dato

Laster opp alle datasett som er endret etter angitt dato. Datoen kan angis som et gyldig SAS datoformat (f.eks. '09MAR1999'D) eller som en SAS datoverdi. Virker bare på SAS datasett, SAS kataloger og SAS formater som er laget eller endret i løpet av eller etter angitt dato.

BINARY

Angir at filen skal overføres binært. Skal ikke brukes ved overføring av SAS datasett.

DATA= SAS datasett

Her angir vi navn på SAS datasett som skal lastes opp

ENTRYTYPE= katalog-type

Angir hvilken SAS katalog-type som skal lastes opp. Brukes sammen med INCAT-option. Kan overstyres av ENTRYTYPE-option i en SELECT- eller EXCLUDE-setning.

INCAT=libref.katalog

Navnet på SAS katalog som skal lastes opp (SAS kataloger må ikke forveksles med SAS formater). Brukes sammen med OUTCAT-option.

INDEX=

Y angir at indeksfiler for indekserte SAS datasett også skal lastes opp, N at de ikke skal.

INFILE= 'filnavn' | fil-referanse

Angir navnet på den sekvensielle filen som skal lastes opp. Enten skrives filnavnet direkte i INFILE-option, ellers kan vi angi det først i en FILENAME-setning for deretter å bruke fil-referansen i INFILE-option. Brukes sammen med OUTFILE-option.

INLIB=libref

SAS-navn på det området datasettene som skal lastes opp ligger under. Kan forkortes til IN=. Brukes sammen med OUTLIB-option.

OUT= SAS datasett

Angir navnet på det overførte SAS datasett.

OUTCAT=libref.katalog

Navnet på SAS katalog som det skal lastes opp til (SAS kataloger må ikke forveksles med SAS formater). Brukes sammen med INCAT-option.

OUTFILE= 'filnavn' | fil-referanse

Angir navnet på den overførte sekvensielle filen. Enten skrives filnavnet direkte i OUTFILE-option, ellers kan vi angi det først i en FILENAME-setning for deretter å bruke fil-referansen i OUTFILE-option. Brukes sammen med INFILE-option.

OUTLIB=libref

SAS-navn på det området datasettene skal lastes opp til. Kan forkortes til OUT=. Brukes sammen med INLIB-option.

MEMTYPE=

Brukes til å avgrense overføring til å gjelde SAS datasett av en spesiell type. Gyldige typer er ALL, CATALOG og DATA. Kan overstyres av MEMTYPE-option i en SELECT- eller EXCLUDE-setning.

STATUS=

Y angir at et statusvindu vises under overføringen, N at statusvinduet ikke vises.

SAS-setninger til PROC UPLOAD:

WHERE

Selekterer hvilke observasjoner som selekteres fra SAS datasettet som lastes opp. Virker bare når DATA-option i PROC UPLOAD-setningen brukes. Beskrivelse av WHERE-setningen finnes på side 91.

EXCLUDE

Her listes datasett som ikke skal lastes opp. Option MEMTYPE bruker vi for å ekskludere SAS datasett av en bestemt type, mens ENTRYTYPE-option kan brukes for å ekskludere SAS kataloger av en bestemt type. Disse options overstyrer tilsvarende options brukt i PROC UPLOAD-setningen.

SELECT

Her listes de datasett som skal lastes ned. Option MEMTYPE bruker vi for å selekttere SAS datasett av en bestemt type, mens ENTRYTYPE-option kan brukes for å selekttere SAS kataloger av en bestemt type. Disse options overstyrer tilsvarende options brukt i PROC UPLOAD-setningen.

Eksempel :

Overfører SAS datasettet *medie.m98aaret* på lokal maskin *medie.m98aaret* på tjenermaskin.

```
PROC UPLOAD DATA=medie.m98aaret OUT=medie.m98aaret MEMTYPE=data;  
RUN;
```

3.12 SAS funksjoner

En SAS funksjon er en rutine som returnerer en verdi utfra null, et eller flere oppgitte argumenter. Det finnes mange forskjellige SAS funksjoner. Det finnes SAS funksjoner for mange forskjellige oppgaver. SAS funksjoner kan brukes ved tilordning av nye variable (assignment- og sumsetningene) og ved testing (IF, DO, SELECT og WHERE). En SAS funksjon kan ha en annen SAS funksjon som argument.

Syntaksvarianter for SAS funksjoner:

```
FUNKSJON(argument-1,...,argument-n)
FUNKSJON(OF variabel-1 - variabel-n)
FUNKSJON(OF variabel-1 ... variabel-n)
FUNKSJON(OF array-navn(*))
FUNKSJON(OF _NUMERIC_)
FUNKSJON(OF variabel-a -- variabel-z)
```

For noen funksjoner kan vi bare bruke en eller noen av disse variantene.

Vi må være klar over bruken av OF. Den sier at alle variablene fra og med variabelen nevnt før streken til og med variabelen etter streken skal behandles. Glemmer vi OF vil variabelen nevnt først minus variabelen nevnt sist bli behandlet. Se eksempel under.

```
stontot = SUM(OF mnd_1-mnd_12) ; * gir summen av variablene fra mnd_1 til mnd_12;
stontot = SUM(mnd_1-mnd_12) ; * gir mnd_1 minus mnd_12 ;
```

3.12.1 Karakterfunksjoner

Disse funksjonene brukes på karaktersvariable eller tekststrenger. Mange av disse er nyttige når vi jobber med karaktersvariable. I forklaringen under brukes betegnelsene argument, n og x. Som argumenter kan vi bruke variable, tekststrenger eller andre funksjoner. Når n kreves, skal vi oppgi et heltall eller en variabel som inneholder et heltall. Kreves x, må vi oppgi et tegn eller en variabel som inneholder et tegn.

BYTE Returnerer tegnet som svarer til et gitt nummer i tegnsettet som brukes.

BYTE(n)

Eksempel der vi finner hvilken rekkefølge tegnene kommer i:

```
DATA rekke;
DO nr = 0 to 255;
  tegn = BYTE(nr);
  OUTPUT;
END;
LABEL tegn = 'Tegn'
      nr = 'Nummer' ;
PROC SORT DATA=rekke ;
BY tegn;
PROC PRINT DATA=rekke NOOBS LABEL;
TITLE 'Sorteringsrekkefølge';
```

COLLATE Returnerer tegn tilsvarende de gitte nummer i tegnsettet som brukes som argument.

COLLATE(startposisjon<,sluttposisjon>
COLLATE(startposisjon<,,lengde>)

COMPBL Bytter gjentatte blanke tegn fra en tekststreng med en blank

COMPBL(argument)

COMPRESS Fjerner spesifiserte tegn fra en tekst.

COMPRESS(argument<,tegn-som-skal-fjernes>)

Eksempel der vi fjerner punktum i diagnoser og prefiks i operasjonskoder:

```
DATA journal;
  INPUT pasnr $ diagnose $ opkode $;
  diagnose = COMPRESS(diagnose,'.');
  opkode   = COMPRESS(opkode,'POX');
  CARDS;
450309 279.3 P3341
133135 323.5 X5561
261130 567.8 O8652
;
PROC PRINT DATA=journal;
  TITLE 'Diagnoser og operasjoner renset';
```

Resultatet:

OBS	PASNR	DIAGNOSE	OPKODE
1	450309	2793	3341
2	133135	3235	5561
3	261130	5678	8652

DEQUOTE Fjerner fnutter (enkle og doble) rundt en tekststreng

DEQUOTE(argument)

INDEX Søker gjennom en tekst for å finne første forekomst av en tekststreng og returner posisjonen til begynnelsen av strengen. Hvis tekststrengen ikke finnes, returneres 0.

INDEX(argument,tekststreng)

Eksempel der vi finner navn som inneholder SOSIALKONTOR:

```
DATA adr;
  INPUT @01 navn $CHAR24.
        @26 adresse $CHAR14.
        @41 postnr 4.
        @46 poststed $CHAR11.
  ;
  IF INDEX(navn,'SOSIALKONTOR') THEN OUTPUT;
  CARDS;
KONGSVINGER SOSIALKONTOR SKOLEGT. 1 2200 KONGSVINGER
HAMAR SOSIALKONTOR POSTBOKS 300 2301 HAMAR
RINGSAKER KOMMUNE SOSIALKONTORET 2380 BRUMUNDDAL
LØTEN KOMMUNE SOSIALKONTORET 2340 LØTEN
STANGE KOMMUNE SOSIALKONTORET 2310 STANGE
NORD-ODAL KOMMUNE SOSIALKONTORET 2120 SAGSTUA
;
PROC PRINT DATA=adr;
  VAR navn postnr poststed;
  TITLE 'Sosialkontor i navnet';
```

Resultatet:

OBS	NAVN	POSTNR	POSTSTED
1	KONGSVINGER SOSIALKONTOR	2200	KONGSVINGER
2	HAMAR SOSIALKONTOR	2301	HAMAR

INDEXC

Søker gjennom en tekst for å finne første forekomst av et tegn i en av de spesifiserte tekststrengene og returnerer posisjonen til tegnet. Hvis ingen av tegnene finnes, returneres 0.

INDEXC(argument,tekststreng-1<,tekststreng-n>)

Eksempel der vi finner navn som inneholder Æ, Ø eller Å:

```
DATA adr;
INPUT  @01  navn      $CHAR24.
       @26  adresse   $CHAR14.
       @41  postnr    4.
       @46  poststed  $CHAR11.
;
IF INDEXC(navn, 'ÆØÅ') THEN OUTPUT;
CARDS;
RÅDE KOMMUNE                SOSIALKONTORET 1640 RÅDE
KONGSVINGER SOSIALKONTOR SKOLEGT. 1      2200 KONGSVINGER
HAMAR SOSIALKONTOR          POSTBOKS 300  2301 HAMAR
RINGSAKER KOMMUNE          SOSIALKONTORET 2380 BRUMUNDDAL
LØTEN KOMMUNE               SOSIALKONTORET 2340 LØTEN
STANGE KOMMUNE              SOSIALKONTORET 2310 STANGE
NORD-ODAL KOMMUNE           SOSIALKONTORET 2120 SAGSTUA
;
PROC PRINT DATA=adr;
VAR navn postnr poststed;
TITLE 'Æ, Ø, Å i navnet';
```

Resultatet:

OBS	NAVN	POSTNR	POSTSTED
1	RÅDE KOMMUNE	1640	RÅDE
2	LØTEN KOMMUNE	2340	LØTEN

INDEXW

Søker gjennom en tekst for å finne første forekomst av et ord i den spesifiserte tekststrengen og returnerer posisjonen til begynnelsen av ordet. Hvis ordet ikke finnes, returneres 0.

INDEXW(argument,tekststreng)

LEFT

Venstrejusterer tegnene i en tekst (fjerner ledende blanke)

LEFT(argument)

LENGTH

Returnerer antall tegn en tekst har

LENGTH(argument)

LOWCASE

Skifter alle store bokstaver til små i en tekst. Greier også Æ, Ø, Å og Ü for Windows (unntatt Ø når SAS system option wincharset er satt til oem) og Unix, men ikke DOS.

QUOTE

Legger doble fnutter rundt en tekststreng. Hvis det ikke er plass til fnuttene, vil teksten trunkeres fra høyre.

QUOTE(argument)

RANK

Returner et tegns tallverdi i tegnsettet. Gjør det motsatte av BYTE.

RANK(x)

REPEAT

Gjentar et uttrykk et spesifisert antall ganger

REPEAT(argument,n)

REVERSE

Snur rekkefølgen til tegnene i en tekst

REVERSE(argument)

RIGHT Høyrejusterer tegnene i en tekst (legger til ledende blanke)

RIGHT(argument)

RXMATCH Søker i en streng etter et mønster definert med RXPARESE-funksjonen og returnerer startverdien til begynnelsen av søkestrengen. Finnes ikke strengen, returneres 0. et mønster det skal søkes etter i en tekst. Resultatet fra funksjonen kan f.eks. brukes i CALL RXSUBSTR. Hvilke mønstre vi kan søke etter er beskrevet under RXPARESE-funksjonen.

RXMATCH(mønster-uttrykk,tekststreng)

Eksempel der vi finner ut hvilken posisjon det første tall står i.

```
x = 'IsIt2L84AD8?';  
pos = RXMATCH(RXPARESE("$'0-9'"),x) ;
```

gir pos verdien 5.

RXPARESE Definerer et mønster det skal søkes etter i en tekst. Resultatet fra funksjonen brukes i RXMATCH, CALL RXCHANGE eller CALL RXSUBSTR.

RXPARESE("mønster-uttrykk <TO mønster-uttrykk>")

Mønsteret defineres som et regulært uttrykk og minner nye om tilsvarende uttrykk brukt i program som c og perl. Bokstaver, tall, punktum, understrek behandles som de tegn de er. Spørsmålsteget brukes for å søke etter et hvilket som helst tegn i en tekst. Kolon søker etter en sekvens med null eller flere tegn i en deltekst. Dollartegn brukes for å søke etter tegn som angis i fnutter etter dollartegnet. Tilde (~) eller hatt (^) brukes for å søke etter tegn som ikke er blant de som angis i fnutter etter tegnet. Det finnes en mengde forskjellige muligheter å definere regulære uttrykk på, her er et par eksempler:

Fjerner alle tegn som ikke er norske bokstaver eller blanke i et navnefelt:

```
DATA navnedata;  
  INPUT @01 navn $CHAR35. ;  
  bokstaver = RXPARESE("^a-zæøåA-ZÆØÅ ' TO ''");  
  CALL RXCHANGE(bokstaver,2147483647,navn);  
CARDS;  
K2r3i"!4"s7t5i6a7n! L"ø^n'*ø 12  
Scarlet O'Hara  
Harald 5  
;  
PROC PRINT DATA=navnedata;  
  TITLE 'Kun norske bokstaver';  
RUN;
```

Alle norske tegn byttes ut (gjør teksten lenger, kan føre til trunkering):

```
DATA navnedata;  
  INPUT @01 navn $CHAR35. ;  
  LENGTH nynavn $ 35 ;  
  inorsk = RXPARESE("'æ' TO 'ae','æ' TO 'Ae',  
                    'ø' TO 'oe','ø' TO 'Oe',  
                    'å' TO 'aa','Å' TO 'Ae'  
                    ' ');  
  CALL RXCHANGE(inorsk,2147483647,navn,nynavn);  
CARDS;  
Sæmund Knåten  
Øyvør Bærhøfstad  
;  
PROC PRINT DATA= navnedata N UNIFORM;  
  TITLE 'Ingen norske bokstaver';  
RUN;
```

SCAN Deler opp en tekst i ord og returnerer ordet som har det spesifiserte nummer i rekken. Hvis vi ikke velger skilletegn brukes disse som skilletegn:

```
Windows: blank . < ( + | & ! $ ) ; ^ - / , % *  
Unix:      blank . < ( + | & ! $ ) ; ^ - / , % *
```

SCAN(argument,n<,skilletegn>)

Eksempel der vi legger deler av et navn i egne variable:

```
DATA adr;
INPUT   @01  navn      $CHAR24.
        @26  adresse   $CHAR14.
        @41  postnr    4.
        @46  poststed  $CHAR11.
        ;
delnavn1 = SCAN(navn,1,' - ');
delnavn2 = SCAN(navn,2,' - ');
CARDS;
RÅDE KOMMUNE                SOSIALKONTORET 1640 RÅDE
KONGSVINGER SOSIALKONTOR    SKOLEGT. 1    2200 KONGSVINGER
HAMAR SOSIALKONTOR          POSTBOKS 300  2301 HAMAR
RINGSAKER KOMMUNE          SOSIALKONTORET 2380 BRUMUNDDAL
LØTEN KOMMUNE              SOSIALKONTORET 2340 LØTEN
STANGE KOMMUNE             SOSIALKONTORET 2310 STANGE
NORD-ODAL KOMMUNE          SOSIALKONTORET 2120 SAGSTUA
        ;
PROC PRINT DATA=adr;
VAR delnavn1 delnavn2;
```

Utskriften:

OBS	DELNAV1	DELNAV2
1	RÅDE	KOMMUNE
2	KONGSVINGER	SOSIALKONTOR
3	HAMAR	SOSIALKONTOR
4	RINGSAKER	KOMMUNE
5	LØTEN	KOMMUNE
6	STANGE	KOMMUNE
7	NORD	ODAL

SOUNDEX SOUNDEX er en enkel algoritme for å sammenligne (engelske) navn fonetisk. Kort fortalt går den ut på at vokaler ikke teller mye fonetisk og derfor ikke brukes i beregningene. Det samme gjelder konsonanten H. De andre konsonantene kan grupperes i grupper og gis tallverdier etter betydning (f.eks. blir B 1, mens M og N blir 5). Den første bokstaven regnes som viktig og beholdes. Doble konsonanter slås sammen til en. Navn som ender opp med samme verdi, regnes som fonetisk like. Dette er en enkel metode som ofte virker. For eksempel beregner SOUNDEX KRISTIAN, KRISTIN og KRISTINE til verdien K6235 og disse regnes da som fonetisk like, mens CHRISTIAN får C6235 og blir dermed forskjellig fra de andre.

SOUNDEX(argument)

SUBSTR Selektierer valgte posisjoner fra en tekst. Når SUBSTR står på høyre side av et likhetstegn (dette er det vanlige) i en assignment-setning, vil resultatet av SUBSTR-funksjonen legges ut i uttrykket på venstre side av likhetstegnet. Står SUBSTR-funksjonen på venstre side av likhetstegnet vil uttrykket på høyre side legges inn i argumentet til SUBSTR.

SUBSTR(argument,startposisjon<,lengde>)

Lager fylke ut fra kommune og sjekker om en kommune er en bykommune:

```
fylke = SUBSTR(kommune,1,2);
IF SUBSTR(kommune,3,1) = '0' THEN bykom = '1';
```

TRANSLATE	Bytter et eller flere tegn i en tekst med et eller flere andre tegn. Tegn for tegn byttes ut, ikke tekststrenger. TRANSLATE(argument,til-1,fra-1<,til-n,fra-n>)
TRANWRD	Bytter en tekst fra en tekststreng ut med en annen tekst. TRANWRD(argument,fra,til)
TRIM	Fjerner blanke på slutten av en tekst (er det kun blanke, beholdes 1) TRIM(argument)
TRIMN	Fjerner blanke på slutten av en tekst (er det kun blanke, beholdes ingen) TRIMN(argument)
UPCASE	Skifter alle små bokstaver til store i en tekst. Greier også æ, ø, å og ü for Windows (unntatt ø når SAS system option wincharset er satt til oem) og Unix, men ikke DOS. UPCASE(argument)
VERIFY	Returnerer posisjonen innen en tekst til den første som ikke er spesifisert for sammenligning. Hvis bare sammenligningstegnene finnes returneres 0. VERIFY(argument,sammenligning-1 ... <,sammenligning-n>)

Eksempel der vi legger ut observasjoner med ugyldig kjønnskode på et eget datasett:

```
DATA feil;
INPUT @02 komm      $CHAR4.
      @07 gkrets    $CHAR3.
      @11 kjonn     $CHAR1.
      @13 aldgrp    $CHAR4.
      @19 inntekt    6.
      @27 skatt      6.
      ;
IF VERIFY(kjonn,'KM') > 0 THEN OUTPUT;
CARDS;
0101 001 K ald1 230327 85023
0101 001 1 ald2 122341 34213
0101 001 K ald1 355345 67326
0101 001 2 ald2 242351 78219
0101 001 K ald3 116341 13232
0101 001 3 ald4 326340 134216
0102 002 M ald3 242463 100230
0211 004 1 ald5 245470 121002
0214 008 ald1 158118 34142
      . 008 M ald1 248118 87631
      ;

PROC PRINT DATA=feil ;
TITLE 'Disse har feil kjønnskode';
```

Slik blir resultatet:

OBS	KOMM	GKRETS	KJONN	ALDGRP	INNTEKT	SKATT
1	0101	001	1	ald2	122341	34213
2	0101	001	2	ald2	242351	78219
3	0101	001	3	ald4	326340	134216
4	0211	004	1	ald5	245470	121002
5	0214	008		ald1	158118	34142

3.12.2 Statistiske funksjoner

De statistiske funksjonene utfører beregninger på de argumentene vi lister opp. De fleste må ha minst to argumenter, MEAN, N og NMISS trenger bare ett, mens SKEWNESS må ha minst tre og KURTOSIS minst fire. Felles for alle (unntatt ORDINAL, se under) er at MISSING-verdier ikke tas med i beregningene. For nærmere beskrivelse av beregningsmetoder, se side 269.

Funksjonene er: **CSS** (veid kvadratavvik), **CV** (variasjonskoeffisient), **KURTOSIS** (halens tyngde), **MAX** (maksimumsverdi), **MEAN** (gjennomsnittsverdi), **MIN** (minimumsverdi), **N** (antall som har verdi), **NMISS** (antall som ikke har verdi), **RANGE** (avstand mellom minste og største verdi), **SKEWNESS** (skjevhet), **STD** (standardavvik), **STDERR** (standardfeil), **SUM** (veid sum), **USS** (veid kvadratsum) og **VAR** (varians).

Eksempel der vi bruker MAX-funksjonen til å sørge for at bidragsbeløp og lånebeløp ikke blir negative og SUM-funksjonen for å legge sammen bidrag og lån:

```
DATA sosial;
INPUT @02 fnr          $CHAR11.
      @14 bidrag        6.
      @21 laan          6.
      ;
bidrag = MAX(bidrag,0) ;
laan   = MAX(laan,0)   ;
stonad = SUM(bidrag,laan);
CARDS;
01010032424 002543 -02654
03095431287          012390
04052421334 011230 008770
11015231832 -00234 -00543
28104132535
;

PROC PRINT DATA=sosial ;
  TITLE 'Oversikt over sosialhjelpsstønad';
```

Oversikt over sosialhjelpsstønad

OBS	FNR	BIDRAG	LAAN	STONAD
1	01010032424	2543	0	2543
2	03095431287	0	12390	12390
3	04052421334	11230	8770	20000
4	11015231832	0	0	0
5	28104132535	0	0	0

ORDINAL Sorterer argumentene i stigende rekkefølge og returnerer verdien til det n'te argumentet. Eventuelle MISSING-verdier kommer først i den sorterte rekken.

ORDINAL (n,argument<,argument> ... <,argument>)

ORDINAL (n,of variabel1-variabeln)

Eksempel. Ønsker å sortere variable som inneholder barns alder slik at de kommer i rekkefølge med det eldste barnet først.

```
DATA mor (DROP = barnu1-barnu7 i); ;

  ARRAY barnu (*)   barnu1 - barnu7 ;
  ARRAY barn  (*)   barn1 - barn7 ;

  INPUT @01 fnr      $CHAR11.
        @13 barnu{*} 2.
        ;

  * Sorterer variablene (eldste barn først);
  DO i = 1 TO DIM(barn) ;
    barn(DIM(barn)-i+1) = ORDINAL(i,of barnu(*));
  END;
  antibarn = N(of barn(*));
  CARDS;
01014531431 17232115152512
30105441635 120716 . . . .
23046230045 2325 . . . .
24115334200 12141618202224
02087245855 03030307 . . .
02087143238 06040201 . . .
13126542019 1216 . . . .
11106342619 . . . . .
;

PROC PRINT DATA=mor;
  TITLE 'Mors barn etter alder (eldste barn først)';
RUN;
```

Ønsker vi å sortere barna med det yngste først, må vi ta hensyn til at MISSING-verdier havner først i den sorterte lista og derfor må overskrives. Programmet blir som over, men DO-løkken må endres noe og dessuten beregnes antall barn i denne:

```
* Sorterer variablene (yngste barn først);
antibarn = 0;
DO i = 1 TO DIM(barn);
  barn(antibarn+1) = ORDINAL(i,of barnu(*));
  IF barn(antibarn+1) NE . THEN
    antibarn + 1;
END;
```

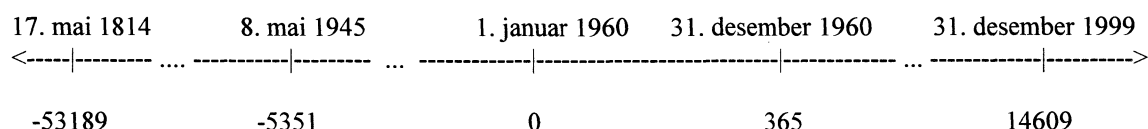
3.12.3 Dato- og tidsfunksjoner

Dato- og tidsfunksjonene kan deles i to:

de som lager en SAS dato og/eller tid verdi

de som henter ut informasjon fra en SAS dato og/eller tid verdi

SAS lagrer datoer som en tallrekke med heltall, se under. En SAS dato er et heltall beregnet som antall dager fra 1. januar 1960 (som er gitt verdi 0). Alle datoer etter denne er positive tall, mens tidligere datoer er negative tall. Datoene har et gyldig intervall for årene 1582-20000. For tidspunkter i døgnet gjøres tilsvarende, SAS tid regnes i antall sekunder fra midnatt. Videre kalkuleres SAS dato og tid som antall sekunder et tidspunkt er fra midnatt 1 januar 1960.



Disse funksjonene lagrer datoer som SAS dato og/eller tid verdier:

DATE	Returnerer dagens dato som en SAS dato verdi. DATE()
DATEJUL	Konverterer en dato på formen ÅÅDDD eller ÅÅÅÅDDD til en SAS dato verdi. DATEJUL(dato).
DATETIME	Returnerer dato og tid som en SAS dato og tid verdi. DATETIME()
DHMS	Konverterer et oppgitt tidspunkt til en SAS dato og tid verdi. DHMS(dato,time,minutt,sekund)
HMS	Konverterer et oppgitt tidspunkt til en SAS tid verdi. HMS(time,minutt,sekund)
INTNX	Konverterer til en SAS dato og/eller tid verdi ut fra et starttidspunkt og et angitt antall tidsintervaller.
MDY	Konverterer en dato til en SAS dato verdi. MDY(måned,dag,år)
TIME	Returnerer det nåværende tidspunkt som en SAS dato og tid verdi. TIME()
TODAY	Returnerer dagens dato som en SAS dato verdi. TODAY()
YYQ	Konverterer den første dagen i det oppgitte kvartal til en SAS dato verdi. YYQ(årstall,kvartal)

Disse funksjonene henter ut informasjon fra en SAS dato og/eller tid verdi

DATDIF	Finner antall dager mellom 2 datoer. Kan operere med forskjellig lengde på måneder og år, dette angis i beregningsbasis, f.eks. '30/360' eller 'Actual'. DATDIF(startdato,sluttdato,beregningsbasis)
DATEPART	Henter datoen fra en SAS dato og tid verdi. DATEPART(tidspunkt)
DAY	Henter ut dagen (1-31) fra en SAS dato verdi. DAY(dato)
HOURL	Henter ut timen (0-23) fra en SAS dato og tid verdi. HOUR(tidspunkt)
INTCK	Regner ut antall tidsintervaller for en gitt periode. INTCK(intervall, fra, til). Gyldige verdier for intervallet er DAY, WEEK, TENDAY, SEMIMONTH, MONTH, QTR, SEMIYEAR, YEAR, DTDAY, DTWEEK, DTTENDAY, DTSEMIMONTH, DTMONTH, DTQTR, DTSEMIYEAR, DTYEAR, HOUR, MINUTE, SECOND. Beregningene foretas ut fra faste intervallbegynnelser. Det betyr at WEEK intervall teller antall søndager i intervallet, ikke faktisk antall dager. Tilsvarende gjelder for år, det beregnes som antall ganger 1. januar finnes i intervallet.
INTNX	Øker verdien med antall tidsintervaller for en gitt periode. INTNX(intervall, fra, antall<, 'målepunkt'>). Målepunktet er BEGINNING, MIDDLE eller END og viser til hvor i intervallet det skal måles fra. Se for øvrig INTCK over.
JULDATE	Konverterer en SAS dato verdi til en dato på formen 2-sifret år og dagnr. JULDATE(dato)
JULDATE7	Konverterer en SAS dato verdi til en dato på formen 4-sifret år og dagnr. JULDATE7(dato)
MINUTE	Henter ut minuttet (0-59) i en time fra en SAS dato og tid verdi. MINUTE(tidspunkt)
MONTH	Henter ut måneden (1-12) fra en SAS dato verdi. MONTH(dato)
QTR	Henter ut kvartalet (1-4) fra en SAS dato verdi. QTR(dato)
SECOND	Henter ut sekundet (0-59) i et minutt fra en SAS dato og tid verdi. SECOND(tidspunkt)
TIMEPART	Henter ut tidspunktet fra en SAS dato og tid verdi. TIMEPART(tidspunkt)
WEEKDAY	Henter ukedag fra en SAS dato verdi. 1=Søndag, 2=mandag osv. WEEKDAY(dato)
YEAR	Henter ut årstall (ÅÅÅÅ) fra en SAS dato verdi. YEAR(dato)
YRDIF	Finner antall år mellom 2 datoer. Kan operere med forskjellig lengde på måneder og år, dette angis i beregningsbasis, f.eks. '30/360', 'ACT/365' eller 'Actual'. YRDIF(startdato,sluttdato,beregningsbasis)

3.12.4 Aritmetiske funksjoner

ABS	Returnerer den absolutte verdien (tallverdien) til et tall. ABS(argument)
DIM	Returnerer antall elementer i en endimensjonal array eller antall elementer i en dimensjon i en flerdimensjonal array. Brukes gjerne i DO-setningen, se side 61. DIM(array-navn<,dimensjon>)
HBOUND	Returnerer nummeret til det siste elementet i en endimensjonal array eller nummeret til det siste elementet i en dimensjon i en flerdimensjonal array. HBOUND vil ofte vær det samme som DIM, men kan bl.a. avvike når det første elementet i en array ikke er 1. HBOUND(array-navn<,dimensjon>)
LBOUND	Returnerer nummeret til det første elementet i en endimensjonal array eller nummeret til det siste elementet i en dimensjon i en flerdimensjonal array. LBOUND(array-navn<,dimensjon>)
MOD	Returnerer resten fra en heltallsdivisjon. Hvis teller eller nevner ikke er et tall, returneres MISSING. MOD(teller,nevner)

Eksempel der vi bruker MOD og SUBSTR til å kode kjønn ut fra fødselsnummer (siffer nr. 9 viser kjønn, menn hvis oddetall, kvinner hvis partall, menn skal få kode M og kvinner kode K):

```
DATA persreg;
INPUT @02 fnr          $CHAR11.
      @14 komm         $CHAR4.
      ;
IF    MOD(SUBSTR(fnr,9,1),2) = 1 THEN kjonn = 'M';
ELSE
  IF MOD(SUBSTR(fnr,9,1),2) = 0 THEN kjonn = 'K';
ELSE
  kjonn = '';
CARDS;
01010032424 0301
09035431287 0301
04042421334 0211
30116231832 0402
                0503
28096132535
02113553435 0238
;
```

SIGN	Returnerer fortegnet til et tall: -1 hvis negativt, 0 hvis 0 og 1 hvis positivt. SIGN(argument)
SQRT	Returnerer kvadratroten til et tall SQRT(argument)

3.12.5 *Trunkeringsfunksjoner*

Argumentene til disse funksjonene må være numeriske.

CEIL	Returnerer heltallet som er større enn eller lik argumentet (avrunder oppover). CEIL(argument)
FLOOR	Returnerer heltallet som er mindre enn eller lik argumentet (avrunder nedover). FLOOR(argument)
FUZZ	Gjør om argumentet til nærmeste heltall når det er mindre enn 1E-12 fra heltallet. FUZZ(argument)
INT	Returnerer heltallet fra et argument (kutter desimaler). INT er lik FLOOR for positive tall og lik CEIL for negative tall. INT(argument)
ROUND	Avrunder et tall til nærmeste angitte avrundingsfaktor. Hvis avrundingsfaktoren utelates, avrundes det til nærmeste heltall. ROUND(argument,avrundingsfaktor) <pre> inntekt = ROUND(8371.53); /* gir inntekt = 8372 */ inntekt = ROUND(8371.53,100) /* gir inntekt = 8400 */ inntekt = ROUND(8371.53,.1) /* gir inntekt = 8371.5 */ </pre>
TRUNC	Trunkerer numeriske variable til en spesifisert lengde. Påvirker ikke variablenes lengde, men kan gjøre tall unøyaktige. Se Nøyaktighet ved lagring av tall, side 268. TRUNC(argument,lengde) Lengden kan være 3-8.

3.12.6 *Spesialfunksjoner*

EXIST	Returnerer 1 hvis libref er opprettet (i en LIBNAME-setning) og 0 hvis den ikke er det eller filtypen er ugyldig. EXIST(libref,filtype) Filtype kan være DATA (SAS-datasett) som er standard, ACCESS (aksess deskriptor), CATALOG (SAS katalog) eller VIEW (SAS view).
FEXIST	Returnerer 1 hvis filreferanse er opprettet i en FILENAME-setning (se side 97), 0 ellers. FEXIST(filreferanse)
FILEEXIST	Returnerer 1 hvis filen finnes og 0 hvis den ikke gjør det. FILEEXIST(filnavn)
FILENAME	Gir et SAS-navn til en rådatafil. Se FILENAME-setningen, side 97. FILENAME(filreferanse,filnavn)

FILEREF Returnerer 0 hvis både referansen og filen finnes. Et negativt tall returneres hvis filreferansen definert i en FILENAME-setning (se side 97) finnes, men den fysiske adressen ikke eksisterer. Det returneres et positivt tall hvis filreferansen ikke er opprettet.

FILEREF(filreferanse)

GETOPTION Returnerer verdien til en SAS system option (se side 40) eller en SAS grafisk option. Denne kan vi f.eks. bruke til å endre verdien til en system option midlertidig for så å sette den tilbake igjen etterpå.

GETOPTION(option<,<IN><,<CM><,<KEYWORD>)

IN angir for grafiske options at målene skal vises i tommer
CM angir for grafiske options at målene skal vises i centimeter
KEYWORD angir at verdien som returnes skal være på formen KEYWORD=verdi. Det vil f.eks si at hvis vi henter sidestørrelse og den er 66, returneres PS=66.

Eksempel på et program som midlertidig endrer sidestørrelse og sidelengde til MAX, lager en tabell med PROC TABULATE, og til slutt setter sidestørrelsen tilbake til det den var før programmet ble kjørt.

```
DATA soshjelp;
INPUT fylke $ ant_mnd stonad;
CARDS;
01 3 32323
01 3
02 . 34567
02 1 50009
;

%LET ps=%SYSFUNC(GETOPTION(ps,KEYWORD));
%LET ls=%SYSFUNC(GETOPTION(ls,KEYWORD));

OPTIONS PS=MAX LS=MAX;

PROC TABULATE DATA=soshjelp FC='-----' NOSEPS F=7.;
CLASS fylke;
VAR stonad;
TABLE ALL='I alt' fylke='',
       stonad='Stønad'*SUM='';

RUN;
OPTIONS &ps &ls;

RUN;
```

INPUT Returnerer en verdi ved å bruke et spesifisert SAS innlesingsformat. Brukes på samme måte som PUT, men med innlesingsformater istedenfor utskriftsformater.

INPUT(kilde,<?|??> innlesingsformat)

kilde er en SAS variabel eller en konstant.
? undertrykker eventuelle feilmeldinger når ugyldige data leses. Den automatiske hjelpevariabelen _ERROR_ settes til 1.
?? gjør det samme som ?, men _ERROR_ settes ikke til 1
innlesingsformat er innlesingsformatet som skal benyttes

LIBNAME Oppretter en forbindelse (libref) til et område. Se LIBNAME-setningen, side 100.

LIBNAME(libref<,'data-område'<,<lagringsformat>>)

LIBREF Returnerer 0 hvis område (libref) finnes og et annet tall hvis den ikke finnes.

LIBREF(libref)

PATHNAME Returnerer den fysiske adressen til en filreferanse tilordnet med en FILENAME-setning (se side 97) eller en libref tilordnet med en LIBNAME-setning (se side 100).

PATHNAME(filreferanse|libref)

PUT Returnerer en verdi ved å bruke et spesifisert SAS utskriftsformat. Dette er en meget nyttig funksjon som blant annet kan brukes til omkodning av data ved hjelp av kataloger, se side 23. Se også eksempler på side 13.

PUT(kilde,<?|??> utskriftsformat)

kilde er en SAS variabel eller en konstant.
 ? undertrykker eventuelle feilmeldinger når ugyldige data leses.
 Den automatiske hjelpevariabelen `_ERROR_` settes til 1.
 ?? gjør det samme som ?, men `_ERROR_` forblir 0.
 utskriftsformat utskriftsformatet som skal benyttes.

Eksempel:

```
f_aar = PUT(fdato, YEAR4.) ; /* Henter fødselsår fra fødselsdato */
```

PUTC henter et karakter utskriftsformat hvis navn er lagret i en variabel og bruker dette. Det betyr at vi kan bruke forskjellig utskriftsformat avhengig av en annen variabels verdi.

PUTN henter et numerisk utskriftsformat hvis navn er lagret i en variabel og bruker dette. Det betyr at vi kan bruke forskjellig utskriftsformat avhengig av en annen variabels verdi.

RESOLVE Henter en makrovariabels verdi i DATA-steget etter at makro 'oversetteren' er ferdig.

RESOLVE minner om SYMGET-funksjonen, men det er noen forskjeller:

- Når en macro variabel inneholder en referanse til en annen makrovariabel, vil RESOLVE hente denne makrovariabelens verdi, det vil ikke SYMGET.
- RESOLVE godtar flere typer argumenter enn SYMGET.
- Hvis det refereres til en ikkeeksisterende makrovariabel, vil RESOLVE returnere variabelens navn, SYMGET vil returnere MISSING.
- RESOLVE krever litt mer ressurser enn SYMGET.

RESOLVE(argument)

Henter dagens dato til en variabel med RESOLVE og med SYMGET:

```
dato_r = RESOLVE('&sysdate');  
dato_s = SYMGET('&sysdate');
```

SPEDIS Returnerer den asymmetriske staveavstanden mellom to strenger, dvs. hvor like to ord. Jo mindre verdier som returneres, jo nærmere nøkkelordet skal sammenligningsteksten være. Små og store bokstaver regnes som forskjellige tegn.

SPEDIS(sammenligning,nøkkelord)

SYMGET Henter en makrovariabels verdi

SYMGET('makrovariabel' | variabel)

Makrovariabelnavnet må stå i fnutter, men uten & foran. Hvis navnet på makrovariabelen er lagret i en vanlig variabel, kan denne brukes som argument i SYMGET-funksjonen

SYSGET Henter verdien til en operativsystemvariable

`SYSGET('operativsystemvariabel')`

Denne kan for eksempel brukes til å hente brukerident (username under windows NT og USER på unix) og navn (usr under windows) på den som kjører SAS-sesjonen. Navnene på operativsystemvariablene får vi med kommandoen *env* på unix og *set* under windows NT.

```
bruker = SYSGET('username');  
navn = SYSGET('usr');
```

SYSPARM Henter verdien fra SYSPARM= -option (se side 46).

`SYSPARM();`

SYSTEM Utfører en eller flere operativsystemkommandoer, returnerer 0 hvis kommandoene var vellykket, ellers returneres et annet tall.

`SYSTEM('kommandoer')`

Andre spesialfunksjoner: CEXIST, DIF, LAG, SYSMSG, SYSRC

3.12.7 Tilfeldige tallsfunksjoner

Disse funksjonene er fine å bruke for å lage utvalg. Felles for disse funksjonene er at de returnerer en verdi ut fra en bestemt fordeling av data. Ut fra en startverdi (seed) genereres tall som ved å gjenta funksjonen mange ganger vil fordele resultatet etter bestemte metoder. En normalfordeling lager vi med NORMAL-funksjonen. Funksjonen UNIFORM kan vi bruke til å lage en uniform fordeling, noe som ofte brukes ved utvalgstrekking (uten tilbakelegging). Startverdien kan være tall som er mindre enn $2^{31}-1$. Denne startverdien brukes den første gangen funksjonen brukes i datasteget. Ut fra startverdien vil det så genereres nye tall for hver gang (en strøm av tall). Er startverdien mindre enn eller lik 0, vil datamaskinens klokke brukes som utgangspunkt for funksjonen. Siden tiden stadig går, vil vi med 0 som startverdi få forskjellig resultatet ved hver kjøring av programmet. Bruker vi et positivt tall som startverdi, vil programmet gi samme resultat, om vi kjører det på nytt (så lenge vi bruker funksjonen i kun en SAS-setning). Dette tallet bør være et oddetall med minst 7 siffer (og < 2147483647). For å få gode fordelinger, må funksjonen kalles mange ganger. Jo flere iterasjoner, jo nærmere ønsket fordeling.

Alle disse funksjonene kan byttes ut med SAS call rutiner, se side 57. Forskjellen mellom disse er:

- Call rutine har en strøm av genererte tall for hver CALL-setning, mens en funksjon bruker samme strøm uansett hvor mange SAS-setninger den brukes i.
- Vi kan sette nye startverdier underveis når vi bruker call rutiner, mens disse ikke vil bli tatt hensyn til ved bruk av funksjoner.

Eksempel der vi trekker ut ca. 10 prosent av observasjonene fra en populasjon. Hver gang vi kjører dette programmet vil vi trekke forskjellig utvalg fordi startverdien er 0:

```
DATA befolkn.utvalg ;
  SET befolkn.persreg ;
  IF UNIFORM (0) <=.1;
```

Eksempel der vi generer et SAS datasett med en normalfordelt variabel. Hver gang vi kjører dette programmet vil resultatet bli det samme fordi startverdien er større enn 0:

```
DATA tilf;
  DO i=1 TO 10000;
    normf = RANNOR(375499137);
    OUTPUT;
  END;
RUN;
```

NORMAL	Returnerer verdi etter normalfordelingsprinsipp (samme som RANNOR) med gjennomsnitt 0 og varians lik 1. NORMAL(startverdi)
RANBIN	Returnerer verdi etter binomial fordeling RANBIN(startverdi,p,n) p er numerisk sannsynlighet for suksess ($0 \leq p \leq 1$) n er uavhengig Bernoulli forsøksparameter, hvor $n > 0$
RANCAU	Returnerer verdi etter Cauchy fordeling RANCAU(startverdi)
RANEXP	Returnerer verdi etter eksponentiell fordeling RANEXP(startverdi)
RANGAM	Returnerer verdi etter gamma fordeling RANGAM(startverdi,a) a er fasongparameter, hvor $a > 0$.
RANNOR	Returnerer verdi etter normalfordelingsprinsipp (samme som NORMAL) med gjennomsnitt 0 og varians lik 1. RANNOR(startverdi)
RANPOI	Returnerer verdi etter Poisson fordeling. RANPOI(startverdi,m) m er parameter for gjennomsnittsverdi
RANTBL	Returnerer verdi etter tabulert sannsynlighet. RANTBL(startverdi,p ₁ ,...,p _n) p er sannsynlighetsparametre(?), hvor $0 \leq p \leq 1$

RANTRI	Returnerer verdi etter triangular fordeling. RANTRI(startverdi,h) h er numerisk og $0 \leq h \leq 1$
RANUNI	Returnerer verdi etter uniform fordeling (samme som UNIFORM). RANUNI(startverdi)
UNIFORM	Returnerer verdi etter uniform fordeling (samme som RANUNI). UNIFORM(startverdi)

3.12.8 Sannsynlighetsfunksjoner

Disse funksjonene returnerer sannsynligheter ut fra gitte fordelinger. Resultatet vil ligge i intervallet 0-1.

POISSON	Returnerer sannsynligheten for at en observasjon fra en Poisson fordeling med gjennomsnitt m , er lik eller mindre enn n . POISSON(m,n) m er numerisk gjennomsnitt, $m \geq 0$. n er et heltall, $n \geq 0$
PROBBETA	Returnerer sannsynligheten for at en observasjon fra en Beta fordeling, med fasongparametre a og b , er lik eller mindre enn x . PROBBETA(x,a,b) x er et tall i intervallet 0-1 a er en numerisk fasongparameter, $a > 0$ b er en numerisk fasongparameter, $b > 0$
PROBBNML	Returnerer sannsynligheten for at en observasjon fra en binomisk fordeling med suksesssannsynlighet p på n forsøk med m suksesser, er lik eller mindre enn m . PROBBNML(p,n,m) p er suksesssannsynlighet, $0 \leq p \leq 1$ n er parameter for antall Bernoulli forsøk, $n > 0$ m er et heltall som angir antall suksesser, $0 \leq m \leq n$
PROBCHI	Returnerer sannsynligheten for at en observasjon fra en Chi-kvadrat fordeling med df frihetsgrad og ikke-sentralitetsparameter nc , er lik eller mindre enn x . PROBCHI(x,df,nc) x er numerisk, $x \geq 0$ df er parameter for frihetsgrad, $df > 0$ nc er ikke-sentralitetsparameter, $nc \geq 0$
PROBF	Returnerer sannsynligheten for at en observasjon fra en F-fordeling med ndf og ddf frihetsgrader og ikke-sentralitetsparameter nc , er lik eller mindre enn x . PROBF(x,ndf,ddf,nc) x er et numerisk, $x \geq 0$ ndf er parameter for tellerens frihetsgrad, $ndf > 0$ ddf er parameter for nevnerens frihetsgrad, $ddf > 0$ nc er ikke-sentralitetsparameter, $nc \geq 0$
PROBGAM	Returnerer sannsynligheten for at en observasjon fra en Gamma fordeling, med fasongparameter a , er lik eller mindre enn x . PROBGAM(x,a) x er numerisk, $x \geq 0$ a er en numerisk fasongparameter, $a > 0$
PROBHYPR	Returnerer sannsynligheten for at en observasjon fra en hypergeometrisk fordeling fra populasjon N med K elementer, utvalgsstørrelse n og odds r , er lik eller mindre enn x . PROBBNML(N,K,n,x,r) N er populasjonen, $N \geq 1$ K er antall elementer, $0 \leq K \leq N$ n er utvalgsstørrelsen, $0 \leq n \leq N$ x er et heltall, $\text{MAX}(0, K+N-n) \leq x \leq \text{MIN}(K, n)$ r er odds-rate, $r \geq 0$

PROBNEGB	Returnerer sannsynligheten for at en observasjon fra en negativ binomisk fordeling med suksesssannsynlighet p med n suksesser, er lik eller mindre enn m . PROBNEGB(p,n,m) p er suksesssannsynlighet, $0 \leq p \leq 1$ n er et heltall som angir antall suksesser, $n \geq 1$ m er et heltall som angir antall fiaskoer, $m \geq 0$
PROBNORM	Returnerer sannsynligheten for at en observasjon fra en normalfordeling er lik eller $< x$. PROBNORM(x) x er numerisk
PROBT	Returnerer sannsynligheten for at en observasjon fra en Student's t-fordeling med df frihetsgrad og ikke-sentralitetsparameter nc , er lik eller mindre enn x . PROBT($x,df,<,nc>$) x er numerisk df er parameter for frihetsgrad, $df > 0$ nc er ikke-sentralitetsparameter, $nc \geq 0$

Eksempel der vi lager en oversikt over binomiske punktsannsynligheter:

```
DATA binomi (DROP=y p);
  ARRAY pr (50) pr_1 - pr_50;
  DO n = 1 TO 20;
    DO x = 1 TO n;
      DO p = 0.01 TO 0.5 BY 0.01;
        y + 1;
        pr (y) = PROBBNML(p,n,x) - PROBBNML(p,n,x-1);
      END;
      y = 0;
      OUTPUT;
    END;
  END;
PROC PRINT DATA=binomi;
  TITLE 'Tabell over binomiske punktsannsynligheter';
RUN;
```

Andre sannsynlighetsfunksjoner: CDF, LOGPDF, LOGSDF, PDF, PROBBNRM, PROBMCM og SDF.

3.12.9 Kvantile funksjoner

De kvantile funksjonene er inverse av sannsynlighetsfunksjoner.

BETAINV	Returnerer kvantilet til p fra en Beta fordeling med fasongparametre a og b . BETAINV er invers av PROBBETA. BETAINV(p,a,b) p er numerisk sannsynlighet, $0 \leq p \leq 1$ a er en numerisk fasongparameter, $a > 0$ b er en numerisk fasongparameter, $b > 0$
CINV	Returnerer kvantilet til p fra en Chi-kvadrat fordeling med df frihetsgrad og ikke-sentralitetsparameter nc . CINV er invers av PROBCHI. CINV($p,df,<,nc>$) p er numerisk sannsynlighet, $0 \leq p \leq 1$ df er parameter for frihetsgrad, $df > 0$ nc er ikke-sentralitetsparameter, $nc \geq 0$
FINV	Returnerer kvantilet til p fra en F-fordeling med ndf og ddf frihetsgrader og ikke-sentralitetsparameter nc . FINV er invers av PROBF. FINV($p,ndf,ddf,<,nc>$) p er numerisk sannsynlighet, $0 \leq p \leq 1$ ndf er parameter for tellerens frihetsgrad, $ndf > 0$ ddf er parameter for nevnerens frihetsgrad, $ddf > 0$ nc er ikke-sentralitetsparameter, $nc \geq 0$
GAMINV	Returnerer kvantilet til p fra en Gamma fordeling med fasongparameter a , er lik eller mindre enn x . GAMINV er invers av PROBGAM. GAMINV(p,a) p er numerisk sannsynlighet, $0 \leq p \leq 1$ a er en numerisk fasongparameter, $a > 0$

PROBIT	Returnerer kvantilet til p fra en normalfordeling. PROBIT er invers av PROBNORM. PROBIT(p) p er numerisk sannsynlighet, $0 \leq p \leq 1$
TINV	Returnerer kvantilet til p fra en Student's t -fordeling med df frihetsgrad og ikke-sentralitetsparameter nc . TINV er invers av PROBT. TINV(p, df, nc) p er numerisk sannsynlighet, $0 \leq p \leq 1$ df er parameter for frihetsgrad, $df > 0$ nc er ikke-sentralitetsparameter, $nc \geq 0$

3.12.10 Matematiske funksjoner

Felles for de fleste matematiske funksjonene er at de bare har ett argument. Den valgte matematiske beregning utføres på argumentet. Resultatet legger vi vanligvis i en ny variabel. Ønsker vi f.eks. å finne eksponential-verdien til en variabel velger vi funksjonen EXP slik: $e = \text{EXP}(1)$; gir e verdien 2,71828.

AIRY	Airy beregning
DAIRY	Derivatet av Airy beregning
DIGAMMA	Digamma beregning
ERF	Feil-funksjon
ERFC	Komplementær feil-funksjon
EXP	Eksponential beregning
FACT	Returner faktor av argumentet
GAMMA	Gamma beregning, faktorisering. $x = \text{GAMMA}(7)$ gir $x = 720$ som er $6!$.
LGAMMA	Logaritme av gamma
LOG	Logaritme
LOG10	Logaritme med base 10
LOG2	Logaritme med base 2
TRIGAMMA	Trigamma beregning

Andre matematiske funksjoner: CNONCT, COMB, CONSTANT, DEVIANCE, FNONCT, IBESSEL, JBESSEL, PERM og TNONCT.

3.12.11 Trigonometriske og hyperboliske funksjoner

Disse funksjonene beregner trigonometriske eller hyperboliske verdier fra et argument.

```
cosinus = cos(0) ; /* gir cosinus = 1 */
```

ARCOS	Returnerer invers cosinus. $\text{Pi}(3,14)$ med nøyaktighet på 14 desimaler er $\text{arcos}(-1)$
ARSIN	Returnerer invers sinus
ATAN	Returnerer invers tangens
COS	Returnerer cosinus
COSH	Returnerer hyperbolisk cosinus
SIN	Returnerer sinus
SINH	Returnerer hyperbolisk sinus
TAN	Returnerer tangens
TANH	Returnerer hyperbolisk tangens

3.12.12 Finansielle funksjoner

Disse funksjonene brukes til å regne ut den manglende komponenten i finansielle ligninger. Vi velger selv den vi vil regne ut ved å sette denne til MISSING i funksjonen.

COMPOUND vanlig renteberegning: $s = b(1 + r)^p$, der s er sluttbeløp, b er begynnelsesbeløp, r er rentefot og p er antall perioder.

COMPOUND (b, s, r, p)

Finner ut hvilken hvor mye 10000 kroner forrenter seg på 8 år med en rentefot på 10%:

avkast = COMPOUND(10000,.,0.1,8) ;

Andre finansielle funksjoner: CONVX, CONVXP, DACCDB, DACCDBSL, DACCSSL, DACCSD, DACCTAB, DEPD, DEPD, DEPSL, DEPSYD, DEPTAB, DUR, DURP, INTRR, IRR, MORT, NETPV, NPV, PVP, SAVING og YIELDP.

3.12.13 Logiske binærtall-funksjoner

Disse funksjonene opererer på tall i 2-tall-systemet. Der representeres f.eks. tallet 5 som 00000101 (lest bakfra 1+0+4, den bakerste er 2⁰, nest bakerste 2¹, tredje bakerste 2² osv.). Når disse tallene sammenlignes gjøres det ved å sjekke tall mot tall bakfra. Skal vi finne samsvar mellom 5 og 14, vil sammenligningen gjøres mellom 00000101 og 00001110. Det som er likt er 00000100 som med vanlige tall representerer 4 (blir resultatet fra BAND-funksjonen). Funksjonene under opererer på slike binære tall.

Tabellen viser sammenligning mellom 5 og 14 og resultatet fra noen av funksjonene:

	BAND		BOR		BXOR	
Argument-1	5	00000101	5	00000101	5	00000101
Argument-2	14	00001110	14	00001110	14	00001110
Resultat	4	00000100	15	00001111	11	00001011

BAND Returnerer tallet som er felles i to binære tall. Tallene som sammelignes må være 0 eller større.

BAND(argument-1,argument-2)

BOR Returnerer tallet som er representert i minst et av de i to binære tall som sammenlignes. Tallene som sammelignes må være 0 eller større.

BOR(argument-1,argument-2)

BXOR Returnerer tallet som er representert forskjellig i de to binære tall som sammenlignes. Tallene som sammelignes må være 0 eller større.

BXOR(argument-1,argument-2)

Andre binærtall-funksjoner: BLSHIFT, BRSHIFT og BNOT.

4. Vedlegg

4.1 Bruke SAS som klient/tjener-verktøy

Når vi bruker SAS kan vi velge om vi vil bruke Windows- eller Unix-versjonen som brukergrensesnitt. Bruker vi Unix vil vi arbeide i et miljø som minner mindre om Windows enn det vi er vant til. Ved å arbeide direkte på Unix blir vi kjent med Unix og Unix-kommandoer. Dette er viktig når vi skal organisere statistikkrutinene våre. Det er viktig å kunne de Unix-kommandoer som skal til for dette. Videre må vi kunne kommandoer for å kopiere filer, flytte filer, slette filer osv. Ved å arbeide direkte på Unix, vil vi bli kjent med de nødvendige kommandoer. Når vi bruker SAS under Unix vil det på den annen side være noe tungvint å få resultater inn i f.eks. et Word-dokument. Dette er mye enklere når vi bruker Windowsversjonen av SAS. Når vi bruker Windows-versjonen av SAS må vi allikevel kjøre programmene våre på en Unix-maskin. Dette fordi alle våre produksjonsdata skal lagres på Unix. Vi bruker da Windows-SAS som klient og Unix-maskinen som tjener. For å bruke dette i praksis må vi gi SAS de nødvendige instruksjoner. Disse kan med fordel lagres i et `autoexec.sas` program. Dette eksekveres når SAS startes med et ikon der `autoexec.sas` er lagret under det området vi har angitt som start-område under *snarvei* i egenskapene til SAS-ikonet (meny med egenskaper får vi opp ved å klikke på ikonet med høyre musetast).

Når vi skal eksekvere et program fra Windows SAS på Unix, bruker vi vanligvis SAS kommandoen `RSUBMIT`. Den sørger for at programmet vårt sendes til tjener-maskinen for eksekvering.

`Autoexec.sas` programmet skal inneholde pålogging til riktig Unix-maskin. Videre må vi opprette forbindelse til de områdene på Unix der dataene våre befinner seg. Dette gjør vi en gang for å kunne eksekvere dem og en gang for å kunne se direkte på dataene. `Autoexec.sas` kan se slik ut:

Eksempel på `autoexec.sas` for klient/tjener fra Windows SAS:

```
* Logger på Unix-maskinen;
DM 'SIGNON ursus';

* Setter options lokalt;
OPTIONS LINESIZE=80 PAGESIZE=66 NOCENTER NODATE NONUMBER;

RSUBMIT;

* Setter options på Unix-maskinen;
OPTIONS LINESIZE=80 PAGESIZE=66 NOCENTER NODATE NONUMBER;

* Tilordner filreferanser på Unix-maskinen for eksekvering der;
LIBNAME library '$SOSHJELP/kat';
LIBNAME soshjelp '$SOSHJELP/wk12';
LIBNAME rev '$SOSHJELP/work';

ENDRSUBMIT;

* Tilordner filreferanser på Unix-maskinen for å kunne se på
  SAS-datasett på Unix fra Windows SAS;
LIBNAME library REMOTE '$SOSHJELP/kat' SERVER=ursus;
LIBNAME soshjelp REMOTE '$SOSHJELP/wk12' SERVER=ursus;
LIBNAME rev REMOTE '$SOSHJELP/work' SERVER=ursus;

RUN;
```

De SAS-setningene som står mellom `RSUBMIT` og `ENDRSUBMIT` vil bli eksekvert på Unix. De andre eksekveres lokalt i Windows SAS.

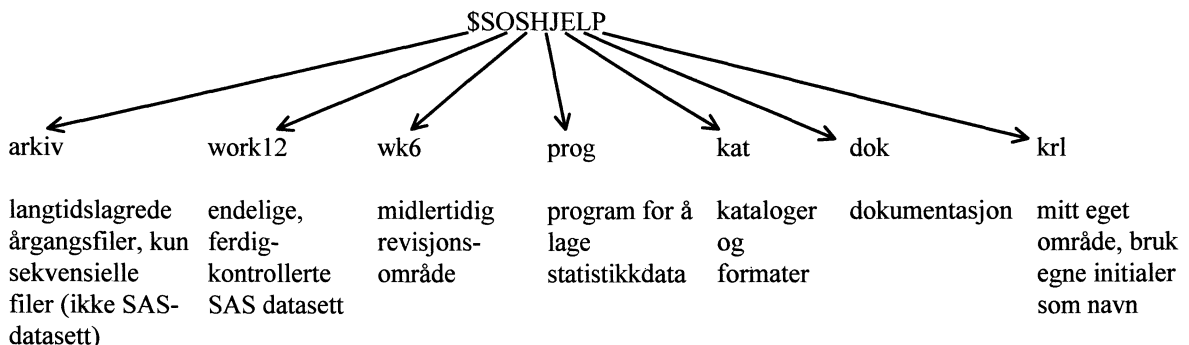
Vi bør sette de samme system options med de samme verdier både lokalt og på Unix-maskinen. Dette gjelder spesielt `LINESIZE`-option. Hvis linjebredden er større på Unix enn i Windows SAS, vil utskrifter i `OUTPUT`-vinduet kunne få et merkelig utseende.

Vi skal være litt forsiktige med å se på store datasett lokalt. Hele datasettet må nemlig overføres fra Unix til nettverket og det kan ta tid.

4.2 Organisering av program og data

Det er meget viktig å ha en strukturert og gjennomtenkt organisering av både data og program. For en typisk statistikk rutine kan vi ha en slik organisering:

- Et område for årgangsfiler (offisielle statistikkdata)
- Et område for endelige, ferdigkontrollerte SAS datasett
- Et område for data til midlertidig bruk/revisjon
- Et område for kataloger og SAS-formater (skal ha SAS-navn *library*)
- Et område for programmene vi bruker for å lage statistikkfilene
- Et område for hver bruker (til egne program)
- Et område for dokumentasjon



Dataene som ligger under området data og under work bør ha hvert sin libref. Videre kan det være aktuelt å lagre data midlertidig under «eget» testområde.

Alle faste libref'er bør defineres i et program kalt *autoexec.sas*. Dette er et oppstartsprogram som automatisk blir kjørt når SAS blir startet fra det området den er lagret under. For en filorganisering som i eksempelet over bør det være én *autoexec.sas* under prog og én under krl.

autoexec.sas under området prog:

```
LIBNAME soshjelp '$SOSHJELP/wk12' ; /* Endelige, ferdige SAS datasett */
LIBNAME rev '$SOSHJELP/wk6' ; /* Data under revisjon */
LIBNAME library '$SOSHJELP/kat' ; /* SAS-formater */
```

autoexec.sas under området krl:

```
LIBNAME soshjelp '$SOSHJELP/wk12' ; /* Endelige, ferdige SAS datasett */
LIBNAME rev '$SOSHJELP/wk6' ; /* Data under revisjon */
LIBNAME test '$SOSHJELP/krl/wk3' ; /* Data til «eget» bruk */
LIBNAME library '$SOSHJELP/kat' ; /* SAS-formater */
```

Under produksjonen av årgangsfilene startes SAS fra området *\$SOSHJELP/prog*, mens for privat uttesting av program etc. startes SAS fra *\$SOSHJELP/krl*.

Programnavnene bør si noe om hva programmet skal gjøre. Program skal ha .sas som hale og bør være maksimalt åtte tegn (halen ikke medregnet). Med åtte tegn til rådighet, vil det ofte bli vanskelig å gi gode navn, derfor trenger vi mer informasjon om programmet. Dette lønner det seg å ha som en kommentar i begynnelsen av programmet.

4.3 Testing og kvalitetssikring

Når vi har skrevet et program er det viktig å teste at programmet gjør det vi ønsker. Det kan vi gjøre ved å lage en liten testfil som vi kjører programmet på. Denne testfilen bør inneholde alle mulige varianter av data som vi kan tenke oss at filen programmet etter hvert skal kjøres på har. Det er spesielt viktig å lage observasjoner med

manglende verdier (MISSING), med gale verdier og med ekstreme verdier. Hva skjer f.eks. i programmet når en variabel vi bruker mangler verdi? Har vi tatt hensyn til dette? Jo flere slike hensyn vi tar når vi programmerer, desto mer robust blir programmet. Ideelt sett skal et program kunne behandle alle tenkelige og utenkelige situasjoner som kan oppstå. Dette er selvfølgelig vanskelig å få til, men med litt erfaring i programmering og evne til å stille de rette spørsmålene og finne svarene på dem, vil vi lære oss å behandle de fleste situasjoner.

La oss først ta et enkelt eksempel på en omkoding; fra en persons fødselsnummer kan vi utfra det niende siffer finne personens kjønn. Er dette sifferet et partall er det en kvinne, mens oddetall viser at fødselsnummeret tilhører en mann. For å kode kjønn kan vi tenke oss å programmere slik:

```
IF SUBSTR(fnr,9,1) IN ('1','3','5','7','9') THEN kjonn = '1';
ELSE                                kjonn = '2';
```

Dette ser jo for så vidt greit ut, men her gjør vi antagelser som kan vise seg å føre galt hen. Hva skjer hvis fødselsnummeret er blankt, eller inneholder bokstaver? Jo, da kodes kjønn til 2, hvilket er galt. Denne testen antar at fødselsnummeret alltid inneholder tall. Er det alltid tilfelle? Dette spørsmålet må vi stille. Svaret er som oftest nei eller vet ikke. Det betyr at vi må forbedre programmet.

```
IF SUBSTR(fnr,9,1) IN ('1','3','5','7','9') THEN kjonn = '1';
ELSE
  IF SUBSTR(fnr,9,1) IN ('0','2','4','6','8') THEN kjonn = '2';
  ELSE                                kjonn = '9';
```

Nå har vi sikret oss at fødselsnummerets niende siffer inneholder tall. Hvis det ikke gjør det, settes kjønn til 9. Hva så med fødselsnummeret? Kan innlesingsformatet påvirke resultatet? Svaret er ja! Leser vi inn fødselsnummeret som en karaktervariabel, har vi to muligheter som kan gi forskjellig resultat. Leses fødselsnummeret inn som en vanlig karaktervariabel, vil eventuelle innledende blanke bli fjernet og fødselsnummeret justert til venstre.

```
INPUT @1 fnr $11. ;
```

Dette innlesingsformatet kan føre til at den egentlige posisjon 9 i fødselsnummeret plutselig havner som nr. 8. Da vil omkodingen bli utført på feil siffer i fødselsnummeret. Dette kan f.eks. skje hvis innledende null i datoer før den 10. i en måned er skrevet som blank. For å sørge for at posisjonene beholder sin plass også om ledende null i fødselsnummeret er blank, bør vi lese inn fødselsnummeret slik:

```
INPUT @1 fnr $CHAR11. ;
```

Dette innlesingsformatet sørger for at eventuell blank istedenfor 0 i første posisjon i fødselsnummeret beholdes. Det forhindrer at fødselsnummeret justeres til venstre og posisjon nr. 9 forblir som nr. 9. Neste spørsmål som dukker opp er hva som skjer hvis fødselsnummeret er definert som et numerisk felt isteden for som et karakterfelt.

```
INPUT @1 fnr 11. ;
```

Dette innlesingsformatet vil føre til problemer fordi SUBSTR-funksjonen er en karakterfunksjon. Når denne brukes på en numerisk variabel vil posisjon nr 9. i det numeriske feltet ikke være 9. siffer i fødselsnummeret! Av dette ser vi at en omkoding kan gi forskjellig resultat alt etter hvilket innlesingsformat en variabel har.

Selv etter å ha behandlet alle disse forskjellige problemstillingene, er vi ikke i mål. Hva hvis fødselsnummeret ikke er gyldig? Kan vi hente kjønnskode fra et fødselsnummer som ikke er gyldig? Vi kan jo ikke vite hva det er i fødselsnummeret som er galt, det kan jo være akkurat det sifferet vi bruker til å finne kjønnskoden. Dessuten hender det ofte at personer vi bare har fødselsdatoen for (de 6 første posisjoner i fødselsnummeret) har fått personnummer (de fem siste siffer i fødselsnummeret) 00000 el. 99999, som er typiske koder for uoppgitt. Det vil bli feil å gi de som har personnummer 00000 kjønnskode 2 og de med 99999 kjønnskode 1. Det betyr at vi må teste om fødselsnummeret er gyldig før vi kan kode kjønn fra det. Gyldighetssjekk av fødselsnummer finnes som en SAS macro på Unix og den returnerer verdien 1 i variabelen retur når det er gyldig.

```
%INCLUDE '$FELLES/sasprog/fnr sjekk.sas' ;
%fnrsjekk(fnr, retur);
IF retur = '1' THEN
  DO;
    IF SUBSTR(fnr,9,1) IN ('1','3','5','7','9') THEN kjonn = '1';
    ELSE
      IF SUBSTR(fnr,9,1) IN ('0','2','4','6','8') THEN kjonn = '2';
  END;
ELSE
  kjonn = '9';
```

Selve omkodningen kan kortes ned ved bruk av SAS funksjoner:

```
%INCLUDE '$FELLES/sasprog/fnr sjekk.sas' ;
%fnrsjekk(fnr, retur);
IF retur = '1' THEN
  kjonn= PUT(MOD(SUBSTR(fnr,9,1)+1,2)+1,$1.);
ELSE
  kjonn = '9';
```

For å teste programmet vi skriver trenger vi en testfil. Den bør inneholde de fleste varianter av fødselsnummer vi kan tenke oss. Programmet som lager testfilen kan se slik ut:

```
DATA sjekk;
  INPUT @1 fnr $CHAR11. ;
  CARDS;
000000000000

4
78wrasfkl76
30116242131
09035427236
09035427 92
09 35427292
090 5427292
9035427292
9035427292
30025443469
290264fsdgs
29020043469
01224673473
1010000110
1010000463
;
```

Etter at programmet er kjørt, må vi sjekke om resultatet ser bra ut. Prosedyrene PRINT, FREQ og MEANS er gode å bruke til slike sjekker. Disse bruker vi aktivt i programmeringsfasen. Eksempelet over viser at det er viktig å tenke nøye gjennom hva vi ber programmet gjøre. Omkodinger som på papiret ser enkle ut, kan ofte bli noe mer kompliserte i praksis. Det som er viktig er å sørge for å få med alle eventualiteter. Dette gjelder spesielt ved manglende og gale verdier, noe svært mange registre har i noen grad. Det er viktig å vite hvordan MISSING-verdier (manglende verdier) skal behandles. Feil bruk av IF-tester kan gi katastrofale resultater. Å teste på om en inntekt er mindre enn 200000 kr. kan vi tenke oss utført slik:

```
IF inntekt < 200000 THEN
```

Denne testen slår til for alle inntekter mindre enn 200000. Det er greit, men testen slår også til på negative inntekter, så vi må avgjøre om dette er akseptabelt. Mer som gjør denne testen tvilsom er at MISSING-verdi også slår til på testen. Med testen over sier vi implisitt at alle uoppgitte inntekter er mindre enn 200000 og det har vi ikke belegg for. Testen bør derfor begrenses til å gjelde alle tall under 200000 hvis negative beløp skal inngå:

```
IF . < inntekt < 200000 THEN
```

Punktum representerer MISSING for numeriske variable. Med testen over sørger vi for at MISSING-verdier ikke slår til på testen. Ønsker vi ikke å ta med de som har negativ inntekt, må testen starte med null:

```
IF 0 <= inntekt < 200000 THEN
```

Neste spørsmål er så om de som har 0 i inntekt skal regnes med eller ikke. Dette må vi også ta stilling til. Blir konklusjonen at de med 0 i inntekt ikke skal være med, må testen endres igjen:

```
IF 0 < inntekt < 200000 THEN
```

Nok en gang ser vi at noe som i utgangspunktet ser ut som en enkel problemstilling, er noe mer komplisert og må tenkes nøye igjennom. Problemstillingen med manglende verdier er den samme for karakttervariable.

4.4 Kontroll og konsistenssjekk av variabelnavn og tekster

Når vi programmerer i SAS må vi følge noen retningslinjer når vi navngir variable og gir dem forklarende tekster ("labler"). Her følger retningslinjene:

- Alle variable **skal** ha en forklarende tekst (label) som er forståelig og entydig
- Variable med samme navn i forskjellige SAS datasett **skal** ha samme innhold og samme tekst
- Variable med forskjellig innhold i forskjellige SAS datasett **skal ikke** ha samme navn og samme tekst

Følger vi disse retningslinjene vil vi ha god mulighet til å få SAS datasett som er oversiktlige og lette å bruke for oss og andre. Vi vil også ha konsistens i navnsettingen fra et SAS datasett til et annet.

Når vi har SAS datasett som inneholder fødselsnummer, skal variabelen hete det samme i alle SAS datasett, f.eks. FNR, og variabelen skal ha samme forklarende tekst ('Fødselsnummer'). Har SAS datasettene også fødselsnummer til personens mor og far, skal disse ha hvert sitt navn, f.eks. MORSFNR og FARSFNR. Disse må så tekstes med 'Mors fødselsnummer' og 'Fars fødselsnummer'.

Det blir altså viktig å finne hvilke variable som har samme innhold i forskjellige SAS datasett, slik at vi kan få gitt dem samme variabelnavn og tekst. Det er ikke alltid like lett å si. Generelt kan vi si at innholdet er det samme så lenge verdienes innhold (koder) ikke endres, men kun utvides med nye verdier eller verdier utgår. Som eksempel på dette kan vi se på en variabel som sivilstand. I familiestatistikken bruker vi disse verdiene (kodene):

```
1 = ugift
2 = gift
3 = enke/enkemann
4 = skilt
5 = separert
```

I den senere tid har kodeverket blitt utvidet med disse:

```
6 = registrert partner
7 = separert partner
8 = skilt partner
9 = gjenlevende partner
```

Så lenge det kommer til nye koder og gamle eventuelt går ut, kan vi si at variabelen har det samme innholdet. Hvis innholdet endres, f.eks. ved å skille ugift i enslig og samboer, må vi lage en ny sivilstandsvariabel. Tilsvarende må vi gjøre hvis vi slår sammen skilte og separerte til en kategori. Da kan ikke variabelen sies å ha det samme innholdet som før.

Når vi vet hvor vanskelig det er å være konsekvent, bør vi bruke litt tid på å sjekke SAS datasettene med blick på variabelnavn og tekster. Retningslinjene sier altså at **alle** variable skal ha tilhørende tekst. Vi ønsker derfor å kunne sjekke dette på en enkel måte. For å gjøre dette kan vi utnytte en SAS tabell som SAS selv bruker til å holde styr på variable i SAS datasett. Den heter DICTIONARY.COLUMNS og inneholder navn på alle variable til alle datasett som er tilknyttet en SAS-sesjon (dvs. de er gitt en libref med en LIBNAME-setning). Ved å lage et lite PROC SQL-program lister vi raskt ut alle variable som mangler tekst. Dette og de etterfølgende programmene er helt generelle og kan brukes når som helst slik de er.

```
PROC SQL ;
  TITLE "SAS-datasett med variable som mangler forklaringstekst (LABEL)";
  SELECT libname, memname, name, label
  FROM DICTIONARY.COLUMNS
  WHERE label = '' AND libname NOT IN('SASHELP','SASUSER');
```

Programmet lager en liste over alle variable som mangler tekst. Listen inneholder libref for datasettet (libname), navnet på datasettet variabelen befinner seg i (memname), variabelens navn (name) og teksten (label, som ikke har noe innhold i denne lista). Forhåpentligvis lages det ikke noen liste, det betyr at alle variable har tekst, eller lista er så kort at det blir en grei jobb å tekste de variablene som mangler tekst. For å gjøre dette bør vi rette de programmene der variablene lages og kjøre dem på nytt. Hvis dette ikke er mulig, kan vi bruke PROC DATASETS, se eksempel side 129. Neste kontroll er å finne ut om det er variable med samme navn som har forskjellige tekster. Dette gjør vi også med PROC SQL på tabellen DICTIONARY.COLUMNS. Først lager vi en midlertidig tabell som inneholder en gruppering av alle variable med alle tilhørende tekster.

```
PROC SQL;
CREATE TABLE navnlab AS
SELECT DISTINCT name, label, COUNT(label) AS antall
FROM dictionary.columns
WHERE libname NOT IN ('SASHELP','SASUSER') AND label > ''
GROUP BY name, label
;
```

Det blir laget en linje pr. variabel med navn på variablene, tilhørende tekst og antall SAS datasett som har variablene.

Nå gjelder det å lage en liste herfra som inneholder hvilke datasette de variabelnavnene som har mer enn én tekst befinner seg på. Hvis det er noen, må tekstene samordnes. Først lager vi en midlertidig tabell med de variablene som går igjen mer enn én gang, dvs de som har mer enn én tekst.

```
PROC SQL;
CREATE TABLE navndub AS
SELECT name
FROM navnlab
GROUP BY name
HAVING COUNT(name) > 1
;
TITLE 'Samme variabelnavn, men FORSKJELLIG forklaringstekst (oppsummering)';
SELECT navnlab.*
FROM navndub , navnlab
WHERE navnlab.name = navndub.name
;
```

Nå som disse er lokalisert, er det greit å finne ut hvilke datasett de befinner seg på, slik at vi kan få rettet uoverenstemmelsene.

```
PROC SQL;
TITLE 'Samme variabelnavn, men FORSKJELLIG forklaringstekst på ulike datasett';
SELECT DISTINCT libname, memname, columns.name, label
FROM dictionary.columns, navndub
WHERE columns.name = navndub.name
ORDER BY columns.name
;
```

Siste kontroll er å sjekke om det er noen tekster som brukes på variable med forskjellig navn. Vi holder oss til PROC SQL denne gangen også.

```
PROC SQL;
CREATE TABLE labeldub AS
SELECT label
FROM navnlab
GROUP BY label
HAVING COUNT(label) > 1
;
TITLE 'Ulike variabelnavn, men med SAMME forklaringstekst (oppsummering)';
SELECT navnlab.*
FROM labeldub , navnlab
WHERE navnlab.label = labeldub.label
;
TITLE 'Ulike variabelnavn, men med SAMME forklaringstekst';
SELECT DISTINCT libname, memname, name, columns.label
FROM dictionary.columns, labeldub
WHERE columns.label = labeldub.label
ORDER BY columns.label
;
```

Når alt er i orden er det greit å lage en liste over innholdet i SAS datasettene, hvilke datasett de forskjellige variablene befinner seg i og det kan vi med nok en PROC SQL. Denne lista er oversiktlig og grei å bruke. Den er nyttig å ha som hjelpemiddel når vi bruker SAS datasettene i arbeidet vårt.

```
PROC SQL;
TITLE "Oversikt over alle variable i alle tilknyttede datasett";
SELECT DISTINCT libname, memname, name, label
FROM dictionary.columns
WHERE libname NE 'SASHELP'
ORDER BY name, label, libname, memname
;
```

4.5 Feilsituasjoner

4.5.1 Manglende semikolon

Den vanligste feilen vi gjør når vi skriver SAS-program er å glemme semikolon. Et uteglemt semikolon kan føre til et ras av feilmeldinger. Disse meldingene er ofte vanskelige å tyde. Derfor bør vi alltid lete etter manglende semikolon når vi får rare feilmeldinger. Feilen ligger da som regel på setningen over den feilmeldingen viser til.

4.5.2 Fnutter i utakt

Når vi skriver SAS-program er det viktig å følge syntaksen. Hvis vi glemmer et så uskyldig tegn som en fnutt (får fnutter i utakt), vil ikke SAS kunne kjøre programmet eller senere program i sesjonen inntil fnuttene er kommet i takt igjen. Dette fordi programmene vi kjører i en SAS-sesjon er ikke uavhengige av hverandre. Vi kan se på alle de programmene vi kjører i løpet av en sesjon som ett program som kjøres i forskjellige deler. Det er vanlig at en syntaksfeil i program, fører til flere syntaksfeil senere i programmet. Dette gjelder spesielt for manglende fnutter. Har vi glemt en fnutt i et program, sier vi at vi har ubalanserte apostrofer (Unbalanced apostrophes) eller fnutter i utakt. SAS har ofte problemer med å finne ut av slike manglende fnutter, og de har en tendens til å skape mange merkelige feilmeldinger.

Symptomer på manglende fnutt:

- Melding **data step running** eller **PROC xxx running** uten timeglass, klokke eller R nederst i skjermbildet
- Programmet kommer opp i loggen uten feilmeldinger og uten å bli eksekvert
- Melding om **unbalanced apostrophes**

Hvordan komme i takt med fnuttene igjen:

- Send en fnutt til eksekvering (se eksempel under). Rett så programmet og kjør det på nytt
- Rett programmet, lagre det, avslutt SAS, start SAS igjen og kjør programmet på nytt

Program som mangler en fnutt:

```
PROC FORMAT;
  VALUE $kjønn
    'M' = 'Menn'
    'K' = 'Kvinner'
    OTHER = 'Uoppgitt kjønn'
  ;
  VALUE inntekt
    0- < 100000 = 'Under 100000'
    100000- < 200000 = '100000 - 200000'
    200000- < 300000 = '200000 - 300000'
    300000- high = '300000 og over'
    OTHER = 'Uoppgitt inntekt'
  ;
run;
```

Vi ser her at det mangler en fnutt i teksten til restgruppen i det første utskriftsformatet. Dette fører til at SAS tror at teksten til restgruppen skal være

```
Uoppgitt kjønn ; VALUE inntekt 0- < 100000 =
```

SAS vil ikke avslutte teksten før en fnutt er funnet. Videre vil SAS tro at vår neste tekst er en SAS-setning.

```
Under 100000
```

Det er det jo ikke, og slik vil det fortsette så lenge fnuttene er i utakt. Det hjelper ikke å rette programmet og kjøre det på nytt, fordi fnuttene er i utakt fra forrige program. For å fnuttene i takt, må vi sende en fnutt. Det gjør vi med følgende lille program:

```

run;
```

Etter å ha eksekvert dette "programmet", skal fnuttene være i takt igjen, og vi kan rette det opprinnelige programmet og kjøre det på nytt. Hvis programmet fortsatt ikke virker, kan vi prøve å avslutte SAS og starte opp på nytt igjen. Dette vil hjelpe såfremt programmet er i orden.

4.5.3 Stoppe et program

Hvis vi under eksekveringen finner at vi har gjort noe feil og ønsker å stoppe programmet, kan vi gjøre det med tastekombinasjonen CTRL+BREAK under Windows. For Unix ved bruk av X-vision må vi gå til vinduet **Session Management** i SAS (f.eks. vha. ALT+TAB-tastene) for deretter å velge **Interrupt** for å stoppe eksekveringen av programmet eller **Terminate** for å avslutte SAS. For batchkjøringer på Unix som legges i bakgrunnen stoppes disse med kommandoen **kill -9 prosessnr** (prosessnr vises idet jobben startes og kan også finnes med Unix-kommandoen **ps**).

4.5.4 Opprinnelig SAS datasett har fått én observasjon og feil innhold

Tidligere var verste stedet vi kunne glemme semikolon i DATA-setningen i DATA-steget. Gjorde vi det samtidig med at vi ønsket å lese et eller flere SAS datasett med SET-setningen (kunne også skje med MERGE- eller UPDATE-setningene), ville SAS datasettene vi hadde tenkt å lese bli ødelagt. Dette skyldes at SAS ikke fant avslutningen på DATA-setningen der vi har ment at den skulle være, siden semikolonet manglet. SAS tolket da SET-setningen som en fortsettelse av DATA-setningen og ville dermed lage eller overskrive SAS datasettene vi hadde ment å lese (hvis det manglende semikolonet ikke førte til syntaksfeil). Fra versjon 8 finnes det en system-option DATASTMTCHK som forhindrer dette, så lenge den er satt til riktig verdi. Hvis denne option ikke er satt og vi glemmer det viktige semikolonet får vi håpe at de ødelagte SAS datasettene kan rekonstrueres eller det finnes backup av dem, ellers vil de være tapt.

Eksempel på hva som skjer når DATASTMTCHK=none:

```
DATA forbruk.landet
    SET forbruk.soer forbruk.nord ;
RUN;
```

Dette programmet vil SAS tolke slik:

```
DATA forbruk.landet set forbruk.soer forbruk.nord ;
RUN;
```

Dermed vil DATA-steget lage eller overskrive 4 SAS-datasett; forbruk.landet, set, forbruk.soer og forbruk.nord og intet mer. Alle SAS datasettene vil inneholde én observasjon og null variable. Hvis derimot DATASTMTCHK=corekeywords, vil vi få en feilmelding istedenfor ødelagte SAS datasett:

```
ERROR 56-185: SET is not allowed in the DATA statement when option
DATASTMTCHK=COREKEYWORDS. Check for a missing semicolon in the DATA statement, or
use DATASTMTCHK=NONE.
```

Dette er slik vi har satt denne system-option.

4.6 Definere stiler for html-filer

Når vi sender utskriften vår til html-filer, har vi mange muligheter til å tilpasse layout. Ved publisering av tabeller på Internett, lager vi html-filer. Skal vi viderebehandle tabellen vår vha. f.eks. Excel er det enkelt ved å lage html-filer først og deretter hente dem inn til Excel.

Det følger med et standardsett med stiler. Disse er forskjellige fra den layout vi ønsker. For å lage en ny stil, bruker vi prosedyren PROC TEMPLATE. Med denne prosedyren kan vi definere forskjellige stiler, men å gjøre det er en tidkrevende og komplisert jobb. Derfor bør vi helst ikke lage stiler på egen hånd.

Hvis vi allikevel må lage en egen stil henter vi den inn i ODS-setningen slik:

```
ODS html
FILE="nace8s.html"
STYLE=minegenstil;
```

I noen prosedyrer, PROC REPORT og PROC TABULATE, kan vi endre layout i selve prosedyren. Det gjør vi ved å bruke STYLE-options. Vi kan gjøre mange slags endringer på denne måten, men vi bør begrense det til et minimum. Hvis vi allikevel må gjøre endringer har vi mange muligheter. Dette kan vi endre:

ASIS=ON|OFF

ON angir at ledende blanke, blanke på slutten og linjeskift beholdes som de er. OFF angir at blanke på begynnelsen eller slutten av en tekst fjernes og også at linjeskift fjernes.

BACKGROUND=farge

Angir bakgrunnsfarge. Kan angis på forskjellige måter, mest nøyaktig kan vi være ved å angi dem heksadesimalt etter RGB-skala (RGB=Rød, Grønn, Blå), f.eks. er #FFFFFF hvit, #FFCC99 vår gulfarge og #000000 er sort. Ellers kan vi også bruke det engelske navnet på fargen.

BACKGROUNDIMAGE='tekst'

Angir en GIF- eller JPG-fil som skal brukes som bakgrunn.

BORDERCOLOR=farge

Angir farge på rammer.

BORDERCOLORDARK=farge

Angir den mørkeste fargen på en ramme når den har 2 farger. Dette brukes for å oppnå 3D-effekt.

BORDERCOLORLIGHT=farge

Angir den lyseste fargen på en ramme når den har 2 farger. Dette brukes for å oppnå 3D-effekt.

BORDERWIDTH=bredde måleenhet

Angir bredden på tabellens ramme. Oppgis helst piksler (px) men kan også oppgis i antall cm, tommer (in), millimeter (mm), punkter (pt).

CELLHEIGHT= bredde måleenhet | heltall%

Angir cellehøyde. Oppgis enten med måleenhet eller som prosent av tabellens høyde.

CELLPADDING= bredde måleenhet | heltall%

Angir hvor mye luft det skal være rundt tall i hver celle i html-tabeller.

CELLSPACING= bredde måleenhet

Angir mellomrom mellom hver celle i html-tabeller.

CELLWIDTH= bredde måleenhet | heltall%

Angir kolonnebredder. Hvis den oppgis i prosent, regnes den i prosent av tabellens bredde.

FLYOVER='string'

Angir en tekst som vises når markøren flyttes over cellen.

FONT=("skrifttype-1 ... skrifttype-n", skriftstørrelse, skriftattributter)

Angir en font som skal brukes. Definerer først skrifttyper, deretter skriftstørrelse og til slutt skriftattributter (som kan være BOLD, COMPRESSED, DEMI_BOLD, DEMI_LIGHT, EXPANDED, EXTRA_BOLD, EXTRA_COMPRESSED, EXTRA_LIGHT, ITALIC, LIGHT, MEDIUM, NARROW, NORMAL, ROMAN, SLANT og WIDE). Ikke alle nettlesere takler alle skriftattributtene.

FONT_FACE='string-1<... , string-n>'

Angir skrifttyper som skal brukes.

FONT_SIZE= størrelse måleenhet | relativ-størrelse

Angir skriftstørrelse. Relativ-størrelse kan være 1-7 og beregnes i forhold til de andre skriftstørrelsene i html-dokumentet.

FONT_STYLE=ITALIC | ROMAN | SLANT

Angir font-stil.

FONT_WEIGHT=tyngde

Angir font-tyngde. Kan være MEDIUM, BOLD, DEMI_BOLD, EXTRA_BOLD, LIGHT, DEMI_LIGHT eller EXTRA_LIGHT.

FONT_WIDTH=relativ-bredde

Angir fontens bredde. Kan være NORMAL, COMPRESSED, EXTRA_COMPRESSED, NARROW, WIDE eller EXPANDED.

FOREGROUND=farge

Angir forgrunnsfarge.

FRAME=ramme-type

Angir ramme rundt tabellen. Disse kan brukes: ABOVE (over tabellen), BELOW (under), BOX (rundt hele), HSIDES (øverst og nederst), LHS (venstre), RHS (høyre), VOID (ingen) eller VSIDES (venstre og høyre).

HREFTARGET='mål'

Angir hvordan en link, definert med URL=, skal vises i nettleseren.

_BLANK	Åpner linken i et eget vindu.
_PARENT	Åpner linken i det samme vinduet som det aktive vinduet ble åpnet i .
_SEARCH	Åpner linken i et søkevindu, kun mulig med Internet Explorer 5 eller nyere.
_SELF	Åpner linken i det aktive vinduet, dette er standard.
_TOP	Åpner linken i det øverste vinduet.
navn	Åpner linken i vinduet med det angitte navnet.

HTMLCLASS='tekst'

Angir navnet på klassen med maler (stylesheet class) som skal brukes.

HTMLID='tekst'

Angir en ident som kan brukes av et Java script.

HTMLSTYLE='tekst'

Angir attributter for layout.

JUST=plassering

Angir plassering av en html-tabell eller celle i en html-tabell. Mulige plasseringer:

CENTER (sentrert), LEFT (venstre) eller RIGHT (høyre).

NOBREAKSPACE=ON | OFF

Angir hvordan blanke skal behandles. ON hindrer linjeskift ved blanke, OFF tillater linjeskiftet.

OUTPUTWIDTH= bredde måleenhet | heltall%

Angir kolonnebredder. Hvis den oppgis i prosent, regnes den i prosent av vinduet

POSTHTML='tekst'

Html-kode som skal plasseres etter html-tabellen i html-dokumentet.

POSTIMAGE='string'

Angir en GIF- eller JPG-fil som skal vises etter html-tabellen.

POSTTEXT=' tekst'

Tekst som skal vises etter html-tabellen i html-dokumentet.

PREHTML='tekst'

Html-kode som skal plasseres før html-tabellen i dokumentet.

PREIMAGE='string'

Angir en GIF- eller JPG-fil som skal vises før html-tabellen.

PRETEXT=' tekst'

Tekst som skal vises etter html-tabellen i html-dokumentet.

PROTECTSPECIALCHARACTERS=ON | OFF | AUTO

Skal angi hvordan spesialtegnene <, > og & skal tolkes.

ON Spesialtegnene skal tolkes som vanlig tekst

OFF Spesialtegnene skal tolkes som html-kode

AUTO Spesialtegnene < og > skal tolkes som html-kode, mens & tolkes som vanlig tekst.

RULES=rammetype

Angir hva slags innramming cellene i html-tabellen skal få.

ALL Vertikale og horisontale rammer.

COLS Vertikale rammer.

GROUP Vertikale og horisontale rammer rundt kolonnehoder og fotnoter

NONE Ingen rammer

ROWS Horisontale rammer.

TAGATTR='string'

Angir en tekst som vil komme som en parameter til html-koden <TD>. Teksten må ha gyldig html-syntaks.

URL='uniform-resource-locator'

angir en link til en annen fil (URL=Uniform Resource Locator).

VJUST='plassering'

Angir hvor innholdet skal plasseres i cellene i html-tabellen. TOP, BOTTOM eller MIDDLE kan velges.

4.7 Nøyaktighet ved lagring av tall

SAS lagrer tall som flyttall (floating point). Dette gjøres for å få best mulig presisjon på desimaltallene. Men uansett hvor nøyaktig vi prøver å lagre tall, vil det være mange brøker som ikke kan representeres helt nøyaktig. For eksempel vil $1/3$ aldri bli helt nøyaktig når den representeres som et desimaltall (0,333333333.....). Slike unøyaktigheter vil kunne forsterke seg når vi utfører mange kalkulasjoner etter hverandre uten å runde av tallene. Eksempelet under viser oss at vi ikke trenger å legge sammen så veldig mange tall før unøyaktighetene merkes.

```
DATA pleie;
  * alds sum skal være summen av ald1-ald4 ;
  INPUT ald1 ald2 ald3 ald4 alds;
  alds=SUM(OF ald1-ald4);
  IF alds NE alds THEN
    OUTPUT;
  CARDS;
23.34 33.33 67.77 12.91 137.35
12.33 33.66 67.11 12.17 125.27
11.01 12.01 13.01 15.01 51.04
0.01 0.33 0.77 0.11 1.22
11.33 11.33 11.33 12.33 46.32
;
PROC PRINT DATA=pleie;
  TITLE 'Uoverensstemmelse mellom enkelttall og sum';
RUN;
```

Utskriften fra programmet viser at summeringen ikke er gjort helt nøyaktig (ingen linjer skulle vært skrevet ut):

```
Uoverensstemmelse mellom enkelttall og sum
OBS      ALD1      ALD2      ALD3      ALD4      ALDSUM      ALDS
1         0.01      0.33      0.77      0.11      1.22      1.22
```

Ved å skrive tallene ut med formatet HEX16. ser vi at det er blitt en liten forskjell:

```
Uoverensstemmelse mellom enkelttall og sum
OBS      ALD1      ALD2      ALD3      ALD4      ALDSUM      ALDS
1         0.01      0.33      0.77      0.11      3FF3851EB851EB85  3FF3851EB851EB86
```

For å unngå at unøyaktigheter skaper problem, bruker vi ROUND-funksjonen. Her avrunder vi til 2 desimaler:

```
DATA pleie;
  INPUT ald1 ald2 ald3 ald4 alds;
  alds=ROUND(SUM(OF ald1-ald4),.01);
  IF alds NE alds THEN
    OUTPUT;
```

Grenseverdier for når heltall vil bli nøyaktig lagret. For desimaltall er grensene lavere.

Lengde	Grenseverdi
3	8 191 (= $2^{13}-1$)
4	2 097 151 (= $2^{21}-1$)
5	536 870 911 (= $2^{29}-1$)
6	137 438 953 471 (= $2^{37}-1$)
7	35 184 372 088 831 (= $2^{45}-1$)
8	9 007 199 254 740 991 (= $2^{53}-1$)

4.8 Definisjoner av statistiske nøkkelord

Følgende notasjon brukes i formlene:

x_i	Observasjon nummer i av en variabel blant de som ikke er MISSING
w_i	Vekten til observasjon x_i , hvis WEIGHT-setningen er brukt, ellers 1.
n	Antall observasjoner som ikke har MISSING-verdier for en variabel
$\bar{x}$	$= \sum w_i x_i / \sum w_i$
d	$= n$ når VARDEF=N VARDEF, se side 150
	$= n - 1$ når VARDEF=DF (standard)
	$= \sum w_i$ når VARDEF=WEIGHT
	$= \sum w_i - 1$ når VARDEF=WDF
s^2	$= \sum w_i (x_i - \bar{x})^2 / d$
s	$= \sqrt{s^2}$
z_i	$= (x_i - \bar{x}) / s$, standardiserte variable

De statistiske beregningene er definert slik:

CSS	$\sum (w_i (x_i - \bar{x})^2)$, veid kvadratavvik
CV	$100s / \bar{x}$, variasjonskoeffisient, prosent
KURTOSIS	$\sum z_i^4 \times n(n+1) / ((n-1)(n-2)(n-3) - 3(n-1)^2 / ((n-2)(n-3)))$, måler halens tyngde, dvs. om noen verdier er veldig langt fra gjennomsnittet.
MAX	maksimumsverdi
MEAN	$\bar{x}$, aritmetisk gjennomsnitt
MIN	minimumsverdi
N	antall observasjoner som har verdi
NMISS	antall observasjoner som mangler verdi
PRT	tosidig p -verdi for Student's t , med $n-1$ frihetsgrader; sannsynligheten under nullhypotesen av å få en absoluttverdi for t større enn t -verdien observert fra utvalget. Kalles PROBT i PROC UNIVARIATE.
RANGE	MAX - MIN, avstanden mellom største og minste verdi
SKEWNESS	$\sum z_i^3 \times n(n-1)(n-2)$, måler skjevheten.
STD	s , standardavviket
STDERR	$s / \sqrt{\sum w_i}$, standardfeil (for gjennomsnittet). Før versjon 6.11: $s / \sqrt{(n)}$
SUM	$\sum w_i x_i$, veid sum
SUMWGT	$\sum w_i$, summen av vektene
T	$t = \bar{x} \sqrt{(n)} / s$, Student's t for H_0 : populasjonsgjennomsnitt = 0
USS	$\sum w_i x_i^2$, veid kvadratsum
VAR	s^2 , variansen

4.9 Felles SAS-formater tilgjengelig for alle

Nå er det ikke lenger nødvendig for hver enkelt av oss å lage sitt eget SAS-format for fylke, kommune, land osv. Ferdig definerte SAS-formater ligger klare til bruk på unix, tilgjengelig for alle. Det eneste vi trenger å vite er hva formatet vi vil bruke heter. Når vi vet navnet på formatet, kan vi bruke dette direkte i en FORMAT-setning. (Vi trenger altså *ikke* noen LIBNAME-setning, den felles SAS-formatkatalogen gjøres tilgjengelig automatisk når vi starter unix SAS.)

Foreløpig er det laget formater for **kommune, fylke, land, politidistrikt, handelsdistrikt** (basert på kommune-nr) og **sykdomsdiagnose**. Det er dessuten laget formater for å skrive tall med **tusenskilte** og **desimalkomma**.

KOMMUNE-formater:

	Anno	Små eller store bokstaver?	Både nummer og navn?	Restgruppering?
SKOM94NR	1994	Små	Ja	Ja
SKOM94NU	"	Små	Ja	Nei
SKOM94UR	"	Små	Bare navn	Ja
SKOM94UU	"	Små	Bare navn	Nei
SKOM92NR	1992	Små	Ja	Ja
SKOM92NU	"	Små	Ja	Nei
SKOM92UR	"	Små	Bare navn	Ja
SKOM92UU	"	Små	Bare navn	Nei
SKOM88NR	1988	Små	Ja	Ja
SKOM88NU	"	Små	Ja	Nei
SKOM88UR	"	Små	Bare navn	Ja
SKOM88UU	"	Små	Bare navn	Nei
SKOM80NR	1980	Små	Ja	Ja
SKOM80NU	"	Små	Ja	Nei
SKOM80UR	"	Små	Bare navn	Ja
SKOM80UU	"	Små	Bare navn	Nei

KOMMUNE-format med utpunktering ('prikking'):

	Anno	Små eller store bokstaver?	Både nummer og navn?	Restgruppering?
SKOP94NR	1994	Små	Ja	Ja

KOMMUNE-formater med ytre områder inkludert (Svalbard etc.)

	Anno	Små eller store bokstaver?	Både nummer og navn?	Restgruppering?
SKOX94NR	1994	Små	Ja	Ja
SKOX94NU	"	Små	Ja	Nei
SKOX94UR	"	Små	Bare navn	Ja
SKOX94UU	"	Små	Bare navn	Nei

FYLKE-formater:

	Små eller store bokstaver?	Både nummer og navn?	Restgruppering?
SFYLKLNR	Små	Ja	Ja
SFYLKLNU	Små	Ja	Nei
SFYLKLUR	Små	Bare navn	Ja
SFYLKLNU	Små	Bare navn	Nei
SFYLKSNR	Store	Ja	Ja
SFYLKSNU	Store	Ja	Nei
SFYLKSUR	Store	Bare navn	Ja
SFYLKSUU	Store	Bare navn	Nei

FYLKE-formater med ytre områder inkludert (Svalbard etc.):

	Små eller store bokstaver?	Både nummer og navn?	Restgruppering?
SFYLXLNR	Små	Ja	Ja

SFYLXLNU	Små	Ja	Nei
SFYLXLUR	Små	Bare navn	Ja
SFYLXLUU	Små	Bare navn	Nei
SFYLXSNR	Store	Ja	Ja
SFYLXSNU	Store	Ja	Nei
SFYLXSUR	Store	Bare navn	Ja
SFYLXSUU	Store	Bare navn	Nei

LAND-formater:

	Små eller store bokstaver?	Både nummer og navn?	Restgruppering?
SLBS95NR	Små	Ja	Ja
SLBS95NU	Små	Ja	Nei
SLBS95UR	Små	Bare navn	Ja
SLBS95UU	Små	Bare navn	Nei

POLITIDISTRIKT-formater:

	Små eller store bokstaver?	Både nummer og navn?	Restgruppering?
SPURKDUR	Små	Nei	Ja
SPURKDUU	Små	Nei	Nei

Format for HANDELSDISTRIKT utfra kommunenr:

	Små eller store bokstaver?	Både nummer og navn?	Restgruppering?
SKOM2HAN	Små	Nei	Ja

DIAGNOSE-formater (3-sifrede ICD-9 koder):

	Små eller store bokstaver?	Både nummer og navn?	Restgruppering?
SICD93NR	Små	Ja	Ja
SICD93NU	Små	Ja	Nei

Restgruppering:

Restgruppering betyr at de som mangler kode samles i en gruppe merket 'Uoppgitt' og at de med ugyldige koder samles i en gruppe merket 'Ugyldig'. Uten restgruppering vil de uoppgitte komme i en umerket gruppe og ugyldige koder skrives ut slik de er.

Picture-format for TUSENSKILLE og DESIMALKOMMA:

Formatene for tusenskilte og desimalkomma har navn som følger mønsteret **SSBn_dV** der **n** betyr antall siffer i tallet totalt (inkl. desimalene) og **d** betyr antall desimaler. $n \in [5,6,7,...,15]$ $d \in [0,1,2,3]$

De minste formatene brukes på tall med opptil 5 siffer hvorav 0, 1, 2 eller 3 desimaler (formatnavn: ssb5_0v ssb5_1v ssb5_2v ssb5_3v), mens de største formatene brukes på tall med opptil 15 siffer hvorav 0, 1, 2 eller 3 desimaler (formatnavn: ssb15_0v ssb15_1v ssb15_2v ssb15_3v). Det er antall siffer i tallet som utgjør **n**, et eventuelt desimalkomma og blanke tegn skal ikke telles med. Dersom det mot formodning skulle forekomme et tall som er større enn formatet tillater, vil det bli skrevet ut *********.

Det er også laget tilsvarende formater med tusenskilte og desimalkomma der tallene divideres ved bruk av formatet:

SSBn_dX	Tallene deles på 10 ved utskrift
SSBn_dC	Tallene deles på 100 ved utskrift
SSBn_dM	Tallene deles på 1000 ved utskrift
SSBn_dT	Tallene deles på 10000 ved utskrift
SSBn_dH	Tallene deles på 100000 ved utskrift
SSBn_dA	Tallene deles på 1000000 ved utskrift

Ved å bruke disse formatene vil vi kunne skille mellom 0 som skal skrives ut som - og mindre en 0,5 av enheten som skal skrives ut som 0 i våre tabeller.

4.10 Sorteringsrekkefølge for forskjellige tegnsett

Under følger en oversikt over tegnrekkefølgen for forskjellige tegnsett. Tegn som er markert med □ er spesialtegn (brukes ofte av operativsystemet). Disse tegnene kan være usynlige for oss, og vi bør være forsiktige med å bruke dem. Et unntak er tabulatortegnet som kan brukes når vi lager filer med tabulator som skilletegn. Når et SAS datasett som er sortert på en maskin med ett tegnsett overføres til en maskin med et annet tegnsett, kan det hende at SAS datasettet ikke lenger er sortert. Data fra Comparex-maskinen ble ofte sortert i en litt annen rekkefølge enn den som er listet under. De særnorske bokstavene ble sortert der de hører hjemme og ikke etter den heksadesimale koden. Blank har heksadesimaltegn 20 for ansi og ascii og 40 for ebcdic. Tabulator har heksadesimalverdi 09 for ansi og ascii. I Windows kan noen tegn endre utseende avhengig av skrifttype.

Nr	Unix ansi	Win ansi	DOS ascii	Comparex ebcdic	Hex	Nr	Unix ansi	Win ansi	DOS ascii	Comparex ebcdic	Hex
0	□	□	□	□	00	57	9	9	9	□	39
1	□	□	□	□	01	58	:	:	:	□	3A
2	□	□	□	□	02	59	;	;	;	□	3B
3	□	□	□	□	03	60	<	<	<	□	3C
4	□	□	□	□	04	61	=	=	=	□	3D
5	□	□	□	□	05	62	>	>	>	□	3E
6	□	□	□	□	06	63	?	?	?	□	3F
7	□	□	□	□	07	64	@	@	@		40
8	□	□	□	□	08	65	A	A	A	□	41
9	□	□	□	□	09	66	B	B	B	□	42
10	□	□	□	□	0A	67	C	C	C	□	43
11	□	□	□	□	0B	68	D	D	D	□	44
12	□	□	□	□	0C	69	E	E	E	□	45
13	□	□	□	□	0D	70	F	F	F	□	46
14	□	□	□	□	0E	71	G	G	G	□	47
15	□	□	□	□	0F	72	H	H	H	□	48
16	□	□	□	□	10	73	I	I	I	□	49
17	□	□	□	□	11	74	J	J	J	#	4A
18	□	□	□	□	12	75	K	K	K	.	4B
19	□	□	□	□	13	76	L	L	L	<	4C
20	□	□	□	□	14	77	M	M	M	(	4D
21	□	□	□	□	15	78	N	N	N	+	4E
22	□	□	□	□	16	79	O	O	O	!	4F
23	□	□	□	□	17	80	P	P	P	&	50
24	□	□	□	□	18	81	Q	Q	Q	□	51
25	□	□	□	□	19	82	R	R	R	□	52
26	□	□	□	□	1A	83	S	S	S	□	53
27	□	□	□	□	1B	84	T	T	T	□	54
28	□	□	□	□	1C	85	U	U	U	□	55
29	□	□	□	□	1D	86	V	V	V	□	56
30	□	□	□	□	1E	87	W	W	W	□	57
31	□	□	□	□	1F	88	X	X	X	□	58
32				□	20	89	Y	Y	Y	□	59
33	!	!	!	□	21	90	Z	Z	Z	□	5A
34	"	"	"	□	22	91	[	[	[	Å	5B
35	#	#	#	□	23	92	\	\	\	*	5C
36	\$	\$	\$	□	24	93	]	]	]	)	5D
37	%	%	%	□	25	94	^	^	^	:	5E
38	&	&	&	□	26	95	`	`	`	^	5F
39	'	'	'	□	27	96	-	-	-	-	60
40	(	(	(	□	28	97	a	a	a	/	61
41	)	)	)	□	29	98	b	b	b	□	62
42	*	*	*	□	2A	99	c	c	c	□	63
43	+	+	+	□	2B	100	d	d	d	□	64
44	,	,	,	□	2C	101	e	e	e	□	65
45	-	-	-	□	2D	102	f	f	f	□	66
46	.	.	.	□	2E	103	g	g	g	□	67
47	/	/	/	□	2F	104	h	h	h	□	68
48	0	0	0	□	30	105	i	i	i	□	69
49	1	1	1	□	31	106	j	j	j	ø	6A
50	2	2	2	□	32	107	k	k	k	,	6B
51	3	3	3	□	33	108	l	l	l	%	6C
52	4	4	4	□	34	109	m	m	m	-	6D
53	5	5	5	□	35	110	n	n	n	>	6E
54	6	6	6	□	36	111	o	o	o	?	6F
55	7	7	7	□	37	112	p	p	p	□	70
56	8	8	8	□	38	113	q	q	q	□	71
						114	r	r	r	□	72

Nr	Unix ansi	Win ansi	DOS ascii	Comparex ebcdic	Hex
115	s	s	s	□	73
116	t	t	t	□	74
117	u	u	u	□	75
118	v	v	v	□	76
119	w	w	w	□	77
120	x	x	x	□	78
121	y	y	y	`	79
122	z	z	z	:	7A
123	{	{	{	Æ	7B
124				Ø	7C
125	}	}	}	'	7D
126	~	~	~	=	7E
127	□	□	□	"	7F
128	□	€	Ç	□	80
129	□	□	ü	a	81
130	□	é	é	b	82
131	□	f	â	c	83
132	□	„	ä	d	84
133	□	„	à	e	85
134	□	†	â	f	86
135	□	‡	ç	g	87
136	□	^	ê	h	88
137	□	%	ë	i	89
138	□	Š	è	□	8A
139	□	‹	ì	□	8B
140	□	Œ	î	□	8C
141	□	□	ì	□	8D
142	□	Ž	Ā	□	8E
143	□	□	Ā	□	8F
144	□	□	É	□	90
145	□	‘	æ	j	91
146	□	’	Æ	k	92
147	□	“	ô	l	93
148	□	”	ö	m	94
149	□	•	ò	n	95
150	□	—	û	o	96
151	□	—	ù	p	97
152	□	~	ÿ	q	98
153	□	™	Ö	r	99
154	□	š	Ü	□	9A
155	□	›	ç	□	9B
156	□	œ	£	□	9C
157	□	□	¥	□	9D
158	□	ž	£	□	9E
159	□	Ÿ	f	□	9F
160	□	□	à	□	A0
161	ı	ı	ı	ü	A1
162	¢	¢	ó	s	A2
163	£	£	ú	t	A3
164	¤	¤	ñ	u	A4
165	¥	¥	Ñ	v	A5
166	¦	¦	°	w	A6
167	§	§	ˆ	x	A7
168	¨	¨	˜	y	A8
169	©	©	˘	z	A9
170	ª	ª	ı	□	AA
171	«	«	½	□	AB
172	¬	¬	¾	□	AC
173	-	-	ı	□	AD
174	®	®	«	□	AE
175	¯	¯	»	□	AF
176	°	°	■	□	B0
177	±	±	■	□	B1
178	²	²	■	□	B2
179	³	³	ı	□	B3
180	´	´	ı	□	B4
181	µ	µ	ı	□	B5
182	¶	¶	ı	□	B6
183	·	·	ı	□	B7
184	¸	¸	ı	□	B8
185	ı	ı	ı	□	B9

Nr	Unix ansi	Win ansi	DOS ascii	Comparex ebcdic	Hex
186	°	°		□	BA
187	»	»	»	□	BB
188	¼	¼	¼	□	BC
189	½	½	½	□	BD
190	¾	¾	¾	□	BE
191	¿	¿	¿	□	BF
192	À	À	À	æ	C0
193	Á	Á	Á	A	C1
194	Â	Â	Â	B	C2
195	Ã	Ã	Ã	C	C3
196	Ä	Ä	Ä	D	C4
197	Å	Å	Å	E	C5
198	Æ	Æ	Æ	F	C6
199	Ç	Ç	Ç	G	C7
200	È	È	È	H	C8
201	É	É	É	I	C9
202	Ê	Ê	Ê	□	CA
203	Ë	Ë	Ë	□	CB
204	Ì	Ì	Ì	ø	CC
205	Í	Í	Í	□	CD
206	Î	Î	Î	ó	CE
207	Ï	Ï	Ï	□	CF
208	Ð	Ð	Ð	à	D0
209	Ñ	Ñ	Ñ	J	D1
210	Ò	Ò	Ò	K	D2
211	Ó	Ó	Ó	L	D3
212	Ô	Ô	Ô	M	D4
213	Õ	Õ	Õ	N	D5
214	Ö	Ö	Ö	O	D6
215	×	×	×	P	D7
216	Ø	Ø	Ø	Q	D8
217	Ù	Ù	Ù	R	D9
218	Ú	Ú	Ú	□	DA
219	Û	Û	Û	□	DB
220	Ü	Ü	Ü	□	DC
221	Ý	Ý	Ý	□	DD
222	Þ	Þ	Þ	□	DE
223	ß	ß	ß	□	DF
224	à	à	α	\	E0
225	á	á	β	□	E1
226	â	â	Γ	S	E2
227	ã	ã	Π	T	E3
228	ä	ä	Σ	U	E4
229	å	å	σ	V	E5
230	æ	æ	μ	W	E6
231	ç	ç	τ	X	E7
232	è	è	φ	Y	E8
233	é	é	Θ	Z	E9
234	ê	ê	Ω	□	EA
235	ë	ë	δ	□	EB
236	ì	ì	ε	Ö	EC
237	í	í	φ	□	ED
238	î	î	ε	□	EE
239	ï	ï	η	□	EF
240	ð	ð	≡	0	F0
241	ñ	ñ	±	1	F1
242	ò	ò	≥	2	F2
243	ó	ó	≤	3	F3
244	ô	ô	┌	4	F4
245	õ	õ	┐	5	F5
246	ö	ö	÷	6	F6
247	÷	÷	≈	7	F7
248	ø	ø	°	8	F8
249	ù	ù	·	9	F9
250	ú	ú	³	□	FA
251	û	û	√	□	FB
252	ü	ü	n	□	FC
253	ý	ý	²	□	FD
254	þ	þ	■	□	FE
255	ÿ	ÿ	□	□	FF

5. Stikkordregister

Sidehenvisninger med fet skrift viser hvor begrepene er forklart, kursiverte sidenummer viser til utfyllende forklaringer, mens resten er andre henvisninger.

#

#BYVAL, hjelpevariabel 33,34

%

%INCLUDE, SAS-setning **58,147**
 %LET, SAS-setning **86**
 %LIST, SAS-setning **87**
 %PUT, SAS-setning **93**

?

?
 SAS-kommando 24

—

ERROR 33,52,63,74,204,205
 LAST =
 system option **36**
 N 10,33,69,74

A

ABORT, SAS-setning **41**
 ABS, SAS funksjon **202**
 ACCESS
 SAS kommando 25
 addere verdien til en variabel 78
 AGE
 proc datasets setning 107,108
 aggregere datasett 126
 AIRY, SAS funksjon **210**
 ALTER TABLE
 proc sql setning 158,162
 ALTER=, SAS datasett option **39**
 ALTLOG=
 system option **34**
 ALTPRINT=
 system option **34**
 analysevariabel
 126,128,131,140,142,143,148,165,167,172,173,187,
 188,189
 angi lengden til variable 68
 angre redigering 24
 ansi tegnsett 226
 APPEND
 proc datasets setning 107,108
 APPEND, prosedyre **96**
 ARCOS, SAS funksjon **210**

aritmetiske funksjoner **202**
 aritmetiske operatorer 28
 ARRAY, SAS-setning 10,22,42
 ARSIN, SAS funksjon **210**
 ascii tegnsett 226
 assignment, SAS-setning 28,44,73,78,147,193,197
 ATAN, SAS funksjon **210**
 ATTRIB, SAS-setning **45**
 AUDIT
 proc datasets setning 107,108
 autoexec.sas 34,36,86,212,213
 AUTOEXEC=
 system option **34**
 automatiske hjelpevariable **33**
 #BYVAL 33,34
 ERROR 33,52,63,74,204,205
 N 10,33,69,74
 FIRST 33,45,79
 LAST 33,45,73,79
 avgrense SAS-setninger til en enhet 51
 avrundingsfeil 222
 avslutte SAS 24,41,52

B

BAND, SAS funksjon **211**
 BATCH=
 system option **34**
 beholde variable 40,65
 beskrive rådata 62
 BETAINV
 SAS funksjon **209**
 BFIND
 SAS kommando 25
 binde utskriftsformater til variable 56
 binær søk 76
 bli kjent med dataene 126
 BOR, SAS funksjon **211**
 BOT
 SAS kommando 26
 BREAK
 proc report setning 140,146
 BUFNO=
 SAS datasett option **39**
 system option **34**
 BUFSIZE=
 system option **34**
 BUFSIZE=, SAS datasett option **39**
 BXOR, SAS funksjon **211**
 BY
 proc compare setning 97,100
 proc freq setning 121
 proc means setning 126,127
 proc print setning 135,136
 proc rank setning 138
 proc report setning 140,142
 proc sort setning 156,157
 proc tabulate setning 165,168,173
 proc transpose setning 180

proc univariate setning.....	185,187
SAS-setning.....	13,33,45,69,77,78,79
BYE	
SAS kommando.....	24
BYERR=	
system option.....	34
BYLINE=	
system option.....	34
BYTE, SAS funksjon.....	193
BY-variabel	
13,33,34,45,69,78,100,121,127,136,138,142,157,180	
,187	

C

CALL DEFINE	
proc report setning.....	140,152
CALL, SAS-setning.....	48,147,207
CANCEL	
SAS kommando.....	26
CAPS=	
system option.....	34
CARDIMAGE	
system option.....	34
CARDS, SAS-setning.....	49
CARDS4, SAS-setning.....	49
CASCADE	
SAS kommando.....	26
CATCACHE=	
system option.....	34
CEIL, SAS funksjon.....	203
CENTER=	
system option.....	34
CHANGE	
proc datasets setning.....	107,108
SAS kommando.....	25
CIMPORT, prosedyre.....	7,8
CINV	
SAS funksjon.....	209
CLASS.....	172
proc means setning.....	126,128
proc tabulate setning.....	165,168
proc univariate setning.....	185,187
CLASSLEV	
proc tabulate setning.....	171
CLEANUP=	
system option.....	34
CLEAR	
SAS kommando.....	24
CNTLLEV=, SAS datasett option.....	39
COLLATE, SAS funksjon.....	193
COLUMN	
proc report setning.....	140,142
COMAMID=	
system option.....	6,34
COMDEF=	
system option.....	34
COMMAND	
SAS kommando.....	26
COMPARE, prosedyre.....	97
COMPBL, SAS funksjon.....	193
COMPOUND, SAS funksjon.....	211
COMPRESS, SAS funksjon.....	12,194
COMPRESS=	
system option.....	34

COMPRESS=, SAS datasett option.....	34,39
COMPUTE	
proc report setning.....	140,147
CONFIG=	
system option.....	35
CONNECT TO	
proc sql setning.....	158
CONNECTREMOTE=	
system option.....	35
CONTENTS	
proc datasets setning.....	107,109
prosedyre.....	11,103
CONTINUE, SAS-setning.....	49
COPY	
proc datasets setning.....	107,109
proc transpose setning.....	180
prosedyre.....	11,86,106
COS, SAS funksjon.....	210
COSH, SAS funksjon.....	210
CPORT, prosedyre.....	7,8
CPUID=	
system option.....	35
CREATE	
proc sql setning.....	158,162
CSS	
SAS funksjon.....	199
statistisk nøkkelord.....	127,142,172,188,223
CURPOS	
SAS kommando.....	26
CUT	
SAS kommando.....	24
CV	
SAS funksjon.....	199
statistisk nøkkelord.....	127,142,172,188,223

D

DAIRY, SAS funksjon.....	210
DATA, SAS-setning.....	41,50
databehandling i løkker.....	51
dataformater.....	30
DATALINES, SAS-setning.....	50
DATALINES4, SAS-setning.....	50
DATASETS, prosedyre.....	107,216
datasett options.....	39
ALTER=.....	39
BUFNO=.....	39
BUFSIZE=.....	39
CNTLLEV=.....	39
COMPRESS=.....	39
DROP=.....	39,156
ENCRYPT=.....	39
FILEFMT=.....	39
FIRSTOBS=.....	36,39,183
IN=.....	15,39,74
INDEX=.....	40
KEEP=.....	40,156
LABEL=.....	40
OBS=.....	37,40,41,156
PW=.....	40
READ=.....	40
RENAME=.....	40
REPLACE=.....	40
REUSE=.....	40
SORTEDBY=.....	40

TOBSNO=	40
TYPE=	40
WHERE=	40,76,77
WRITE=	40
DATA-steget	5,33,41,50,56,73
DATA\$MTCHK =	
system option	35
DATDIF, SAS funksjon	201
DATE, SAS funksjon	201
DATE=	
system option	35
DATEJUL, SAS funksjon	201
DATEPART, SAS funksjon	201
DATETIME, SAS funksjon	201
dato og tidsfunksjoner	200
datovariabel	11,12,30,32,35,52,200
datoverdier	12,28,30,32,35,111,191,200
DAY, SAS funksjon	201
debugging	50
DEFINE	
proc report setning	140,145
definere et vindu	80
DELETE	
proc datasets setning	107,109
proc sql setning	158,162
SAS-setning	50
DEQUOTE, SAS funksjon	194
DESCRIBE	
proc sql setning	158,162
desiler	139
desimalkomma	30,31,225
DETAILS=	
system option	35
DEVICE=	
system option	35
DFLANG=	
system option	35
DHMS, SAS funksjon	201
DICTIONARY.COLUMNS	216
DIGAMMA, SAS funksjon	210
DIM, SAS funksjon	9,42,51,202
DIR	
SAS kommando	25
DISCONNECT FROM	
proc sql setning	158
DISPLAY, SAS-setning	50,80
DKRCOND=	
system option	35
DKROCOND=	
system option	35
DLDMGACTION=, SAS datasett option	39
DLGFIND	
SAS kommando	25
DLGREPL	
SAS kommando	25
DM, SAS-setning	6,7,84,147
DMS=	
system option	35
DO, SAS-setning	9,51,147
DOWN	
SAS kommando	26
DOWNLOAD, prosedyre	6,7,111
DROP	
proc sql setning	158,162
SAS-setning	52
DROP=, SAS datasett option	39,156

DSNFERR=	
system option	35
dubletter	13,14,15,16,18,156

E

ECHO=	
system option	35
ECHOAUTO=	
system option	35
edcdic tegnsett	226
EDITCMD=	
system option	35
eksekvere program	24
eksterne filer	124
EMAILSYS=	
system option	35
ENCRYPT=, SAS datasett option	39
END	
SAS kommando	24
SAS-setning	52,147
ENDCOMP	
proc report setning	140,152
endre attributter til SAS datasett	107,110
endre destinasjon for utskrift	137
endre navn på SAS datasett	107
endre SAS system options	92
endre variabelnavn	40,73,110
ENDRSUBMIT	
SAS-setning	6,7,84,93,212
ENDSAS, SAS-setning	52
ENGINE=	
system option	35
e-post	35,85
ERF, SAS funksjon	210
ERFC, SAS funksjon	210
ERROR, SAS-setning	52
ERRORABEND=	
system option	35
ERRORCHECK=	
system option	35
ERRORS=	
system option	35
EXACT	
proc freq setning	121,124
Excel regneark	10
EXCHANGE	
proc datasets setning	107,109
EXCLUDE	
proc copy setning	106
proc datasets setning	107,109
proc download setning	111,112
proc format setning	113,116
proc upload setning	191,192
EXECUTE	
proc sql setning	158
SAS call rutine	48
EXIST	
SAS funksjon	203
EXP, SAS funksjon	210
EXPLORER=	
system option	35

F

FACT, SAS funksjon	210
faktorisering	210
feilsituasjoner	218
feilsøking	50
FEXIST	
SAS funksjon	203
FILE	
SAS kommando	24
SAS-setning	53
FILEEXIST	
SAS funksjon	203
FILEFMT=, SAS datasett option	39
FILENAME	
SAS funksjon	203
SAS kommando	25
SAS-setning	111,191
FILENAME, SAS-setning	6,11,84,203,205
FILEREF	
SAS funksjon	204
filhale	
sas7baud	108
sas7bdat	87
sas7bndx	109
finansielle funksjoner	211
FIND	
SAS kommando	25
FINV	
SAS funksjon	209
FIRST, hjelpevariabel	33,45,79
FIRSTOBS=	
SAS datasett option	36,39,183
system option	36
flette SAS-datasett	77
FLOOR, SAS funksjon	203
flyttall	222
FMterr=	
system option	36,56
FMTSEARCH=	
system option	36
fnutter i utakt	218
FONT=	
system option	36
FOOTNOTE	
SAS kommando	25
SAS-setning	54,85
forklare innholdet i variable	67
forklaringstekst til variable	67,104,110,135,180,216
FORMAT	
proc datasets setning	107,109
proc freq setning	121,124
proc fsview setning	124,125
proc means setning	126,134
proc print setning	135,136
proc report setning	140,153
proc tabulate setning	165,175
prosedyre	21,75,113,176
SAS-setning	45,56,224
FORMAT, prosedyre	11
formater	30
formaterte rådata	61
FORMCHAR=	
system option	36,140
FORMDLIM=	
system option	36

FORMS=	
system option	36
forspalte	165
footnoter	85
frekvenstabeller	121
FREQ	
proc means setning	126,134
proc report setning	140,153
proc tabulate setning	165,176
proc univariate setning	185,187
prosedyre	121,165,176,215
FSLIST	
prosedyre	124
SAS kommando	25
FSVIEW	
prosedyre	124
SAS kommando	25
FULLSTIMER=	
system option	36
funksjoner	193
funksjonstaster	23
FUZZ, SAS funksjon	203

G

GAMINV	
SAS funksjon	209
gamma fordeling	209
GAMMA, SAS funksjon	210
generere labler til variable	164
GENMAX=, SAS datasett option	39
GENNUM=, SAS datasett option	39
GETOPTION	
SAS funksjon	204
gi en variabel en verdi	44
gi SAS-navn til rådata	84
GO TO, SAS-setning	56,147
gruppere observasjoner	45
grupperingsvariabel	
..... 141,142,143,145,147,165,167,168,172,173,187	
gyldighetsregler	96,109,159
gå midlertidig ut av SAS	24
gå ut av en løkke	68

H

HBOUND, SAS funksjon	202
hekte sammen SAS datasett	96
HELP	
SAS kommando	24
hente en variabls label	10,48
hente inn SAS-setninger	58
hente program inn til editoren	24
hente tilbake eksekvert program	24
hente tilbake forrige kommando	24
HISTOGRAM	
proc univariate setning	185,187
hjelp	24
hjelpevariable	33
#BYVAL	33,34
ERROR	33,52,63,74,204,205
N	10,69,74
FIRST	33,45,79

LAST	33,45,73,79
HLO, proc format variabel	114,116,118,120
HMS, SAS funksjon	201
HOME	
SAS kommando	26
hoppe i programmet	56,68
HOURL, SAS funksjon	201
HSCROLL	
SAS kommando	26
html	87
html-filer	85,95,112,142,219
html-stiler	142,147,171,219
hyppigst forekommende verdi	185,189
høyrejustering	27,72,104,196

I

IC CREATE	
proc datasets setning	107,109
IC DELETE	
proc datasets setning	107,109
IC REACTIVATE	
proc datasets setning	107,109
ICON	
SAS kommando	26
ID	
proc compare setning	97,100
proc fsview setning	124,125
proc means setning	126,128,129
proc print setning	135,136
proc transpose setning	180
proc univariate setning	185,188
identifisere et punkt i programmet	67
identifisere rådatafil	58
identifisere utdatafil	53
IDLABEL	
proc transpose setning	180,181
IDXNAME=, SAS datasett option	39
IDXWHERE=, SAS datasett option	39
IF, subsetting, SAS-setning	56
IF, THEN, ELSE, SAS-setning	57,65,147
IN, SAS operator	28
IN=, SAS datasett option	15,39,74
INCLUDE	
SAS kommando	24
indekserte SAS datasett	39,40,103,104,107,109,111,156,162,191
INDEX CREATE	
proc datasets setning	107,109
INDEX DELETE	
proc datasets setning	107,109
INDEX, SAS funksjon	194
INDEX=, SAS datasett option	40
INDEXC, SAS funksjon	195
INDEXW, SAS funksjon	195
INFILE, SAS-setning	58
INFORMAT	
proc datasets setning	107,109
proc fsview setning	124
SAS-setning	45,61
INITCMD=	
system option	36
INITSTMT=	
system option	36
innhold i SAS datasett	103,109,217

innlesingsformater	30
INPUT	
SAS funksjon	12,204
SAS-setning	9,62
INSERT	
proc sql setning	158,162
INSET	
proc univariate setning	185,188
INT, SAS funksjon	203
INTCK, SAS funksjon	201
integrity constraints	109
interaktivt hjelpesystem	24,41,123
INTX, SAS funksjon	201
INVALIDDATA=	
system option	36
INVALUUE	
proc format setning	113,116

J

join	19
JULDATE, SAS funksjon	201
JULDATE7, SAS funksjon	201

K

kalle opp SAS vinduer	25
karakterfunksjoner	193
KEEP, SAS-setning	65
KEEP=, SAS datasett option	40,156
KEYLABEL	
proc tabulate setning	165,176
KEYS	
SAS kommando	25
KEYWORD	
proc tabulate setning	176
kjøre program	24
klipping og liming	24
kobling av SAS datasett	13,69
kommentar	
SAS-setning	65,147
komprimering av SAS datasett	34,39,106
konkatenering	28,44,59,85
kontroll av variabelnavn og tekster	216
konvertering av data	6
Excel regneark	10
fra Digital Unix til Sun Unix	6
fra PC til Unix	6
fra Sun Unix til Digital Unix	6
fra Unix til PC	7
skilletegnseparerte filer	8
SPSS-filer	11
konvertering av variable	12
kopiere SAS datasett	106,107,109
KURTOSIS	
SAS funksjon	199
statistisk nøkkelord	127,188,223

L

LABEL	
proc datasets setning	107,109,110

proc tabulate setning.....	165,176
SAS call rutine.....	10,48
SAS-setning.....	45,67
label, SAS-setning.....	67
LABEL=	
SAS datasett option	40
system option.....	36
lage fotnoter	85
lage labler.....	164
lage luft i SASlog.....	77
lage overskrift	95
lage testdata.....	49
lagre program	24
lagring av tall	222
LAST, hjelpevariabel	33,45,73,79
laste ned data.....	111,191
LBOUND, SAS funksjon.....	202
LEAVE, SAS-setning	68
LEFT	
SAS funksjon.....	195
SAS kommando.....	26
LENGTH	
SAS funksjon.....	195
SAS-setning.....	10,45,68,147
lese SAS datasett.....	76
LGAMMA, SAS funksjon	210
LIB	
SAS kommando.....	25
LIBNAME	
SAS funksjon.....	204
SAS-setning.....	11,86,203,205
libref.....	86,104,116,203,204,205,213
LIBREF	
SAS funksjon.....	204
LINE	
proc report setning.....	140,152
LINESIZE	
SAS kommando	26
LINESIZE=	
system option.....	36,212
linjestørrelse.....	26,36
LINK, SAS-setning.....	68,147
LIST, SAS-setning.....	69
liste eksekverte SAS-setninger.....	87
LOG	
SAS funksjon.....	210
SAS kommando	25
LOG=	
system option.....	36
LOG10, SAS funksjon	210
LOG2, SAS funksjon	210
logiske operatorer.....	28
LOSTCARD, SAS-setning.....	69
LOWCASE, SAS funksjon	195
lukke et vindu.....	24

M

MACRO=	
system option.....	36
makrovariabel	37,38,48,81,86,112,164,205
MANAGER	
SAS kommando	25
manglende semikolon	218
MAPS=	

system option.....	36
MARK	
SAS kommando.....	24
matematiske funksjoner	210
MAUTOSOURCE=	
system option.....	36
MAX	
SAS funksjon.....	199
statistisk nøkkelord.....	127,142,171,172,188,223
MDY, SAS funksjon.....	201
MEAN	176
SAS funksjon.....	199
statistisk nøkkelord.....	127,142,145,171,172,188,223
MEANS, prosedyre.....	115,120,126,151,215
MEDIAN	
statistisk nøkkelord.....	127
medianverdi	172,185,186,188
MEMSIZE=	
system option.....	36
merge	13
MERGE, SAS-setning	13,69
MERGENOBY=	
system option.....	36
MERROR=	
system option.....	36
MFILE=	
system option.....	36
MIN	
SAS funksjon.....	199
statistisk nøkkelord.....	127,142,171,172,188,223
MINUTE, SAS funksjon.....	201
MISSING, SAS-setning	70
MISSING=	
system option.....	36
MISSING-verdier	
8,12,14,15,21,36,42,70,73,78,96,98,122,124,127,128	
,138,141,142,146,167,168,171,173,199,202,214,215	
MLOGIC=	
system option.....	36
MOD, SAS funksjon.....	202
MODE, hyppigst forekommende verdi.....	185,189
MODIFY	
proc datasets setning.....	107,110
MONTH, SAS funksjon.....	201
MPRINT=	
system option.....	36
MRECALL=	
system option.....	36
MSG=	
system option.....	36
MSGCASE=	
system option.....	36
MSGLEVEL=	
system option.....	37
MSTORED=	
system option.....	37
MVARSIZE=	
system option.....	37
mønster for utskrift av tall.....	113,117,225

N

N	134,153,187
SAS funksjon.....	199
statistisk nøkkelord.....	127,142,172,176,188,223

NEWS=	
system option.....	37
NEXT	
SAS kommando.....	26
NMISS	
SAS funksjon.....	199
statistisk nøkkelord.....	127,142,171,172,176,188,223
NORMAL	
SAS funksjon.....	207
NOTES=	
system option.....	37
null, SAS-setning.....	70
NUM	
SAS kommando.....	24
NUMBER=	
system option.....	37
nummerere linjer i programeditoren.....	24,27
nøyaktighet ved lagring av tall.....	222

O

OBS=	
SAS datasett option.....	37,40,41,156
system option.....	37
ODS, SAS-setning.....	87
operatorer.....	28
andre.....	28
aritmetiske.....	28
logiske.....	28
sammenlignende.....	28
oppblåsing.....	124,134,153,176,189
oppdatere SAS datasett.....	78
OPTIONS	
prosedyre.....	34,134
SAS kommando.....	25
SAS-setning.....	34,92
ORDINAL, SAS funksjon.....	200
OUT	
SAS kommando.....	25
OUTPUT	
proc means setning.....	126,129
proc univariate setning.....	185,188
SAS-setning.....	41,71
OUTREP=, SAS datasett option.....	40
overføre data mellom maskiner.....	6,111,191

P

P1	
statistisk nøkkelord.....	127
P10	
statistisk nøkkelord.....	127
P5	
statistisk nøkkelord.....	127
P90	
statistisk nøkkelord.....	127
P95	
statistisk nøkkelord.....	127
P99	
statistisk nøkkelord.....	127
PAGE, SAS-setning.....	93
PAGEBY	
proc print setning.....	135,136

PAGENO=	
system option.....	37
PAGESIZE	
SAS kommando.....	26
PAGESIZE=	
system option.....	37
PARM=	
system option.....	37
PARMCARDS=	
system option.....	37
PASTE	
SAS kommando.....	24
PATH=	
system option.....	37
PATHNAME	
SAS funksjon.....	205
PCLEAR	
SAS kommando.....	24
PCTN	
statistisk nøkkelord.....	142,172,176
PCTSUM	
statistisk nøkkelord.....	142,172,176
PGM	
SAS kommando.....	25
Pi.....	210
PICTURE	
proc format setning.....	113,117
PMENU	
SAS kommando.....	26
POINTOBS=, SAS datasett option.....	40
POISSON	
SAS funksjon.....	208
PREVWIND	
SAS kommando.....	26
PRINT, prosedyre.....	11,135,215
PRINT=	
system option.....	37
PRINTCMD=	
system option.....	37
PRINTTO, prosedyre.....	137
PROBBETA	
SAS funksjon.....	208,209
PROBBNML	
SAS funksjon.....	208
PROBCHI	
SAS funksjon.....	208,209
PROBF	
SAS funksjon.....	208,209
PROBGAM	
SAS funksjon.....	208,209
PROBHYP	
SAS funksjon.....	208
PROBIT	
SAS funksjon.....	210
PROBNEGB	
SAS funksjon.....	209
PROBNORM	
SAS funksjon.....	209,210
PROBPLOT	
proc univariate setning.....	185
PROBSIG=	
system option.....	37
PROBT	
SAS funksjon.....	209,210
statistisk nøkkelord.....	127,188,223
PROC=	
system option.....	37

PROCLEAVE=		
system option.....	37	
PROC-steget	5,56,96	
prosedyrer	96	
Proc append.....	96	
Proc cimport	7,8	
Proc compare	97	
Proc contents	11,103	
Proc copy.....	11,86,106	
Proc cport	7,8	
Proc datasets	107,216	
Proc download.....	6,7,111	
Proc format	11,21,75,113,176	
Proc freq.....	121,165,176,215	
Proc fslst.....	124	
Proc fsview	124	
Proc means	115,120,126,151,215	
Proc options.....	34,134	
Proc print	11,135,215	
Proc printto.....	137	
Proc rank	138	
Proc report	140,220	
Proc sort.....	156	
Proc sql.....	19,158,216,217	
Proc tabulate	33,75,115,120,139,145,165,204,220	
Proc template	88,89,92,219	
Proc transpose.....	180	
Proc univariate.....	185	
Proc upload.....	6,191	
prosentiler	172,185,186,188,189	
prosentuering	121,142,172,176	
PRT		
statistisk nøkkelord	142,172,223	
PUT		
SAS funksjon.....	12,21,22,149,205	
SAS-setning.....	9,71	
PUTC, SAS funksjon	205	
PUTN, SAS funksjon.....	205	
PW=, SAS datasett option.....	40	
PWREQ=, SAS datasett option.....	40	
Q		
Q1		
statistisk nøkkelord	127	
Q3		
statistisk nøkkelord	127	
QQPLOT		
proc univariate setning.....	185	
QRANGE		
statistisk nøkkelord	127	
QTR, SAS funksjon	201	
QUOTE, SAS funksjon.....	195	
R		
RANBIN		
SAS call rutine.....	48	
SAS funksjon.....	207	
RANCAU		
SAS call rutine.....	48	
SAS funksjon.....	207	
RANEXP		
SAS call rutine.....	48	
SAS funksjon.....	207	
RANGAM		
SAS call rutine.....	48	
SAS funksjon.....	207	
RANGE		
SAS funksjon.....	199	
statistisk nøkkelord	127,142,172,188,223	
rangere data etter verdier.....	138	
RANK		
prosedyre	138	
SAS funksjon.....	195	
RANKS		
proc rank setning	138,139	
RANNOR		
SAS call rutine.....	48	
SAS funksjon.....	207	
RANPOI		
SAS call rutine.....	48	
SAS funksjon.....	207	
RANTBL		
SAS call rutine.....	48	
SAS funksjon.....	207	
RANTRI		
SAS call rutine.....	48	
SAS funksjon.....	208	
RANUNI		
SAS call rutine.....	48	
SAS funksjon.....	208	
RBREAK		
proc report setning.....	147	
RCHANGE		
SAS kommando	25	
READ=, SAS datasett option.....	40	
RECALL		
SAS kommando	24	
RENAME		
proc datasets setning.....	107,110	
SAS-setning.....	73	
RENAME=, SAS datasett option.....	40	
REPAIR		
proc datasets setning.....	107,110	
reparere ødelagte SAS datasett.....	107,110	
REPEAT, SAS funksjon	195	
REPEMPTY=, SAS datasett option.....	40	
REPLACE=		
SAS datasett option	40	
system option.....	37	
REPORT, prosedyre	140,220	
RESET		
proc sql setning.....	158,162	
RESHOW		
SAS kommando	26	
RESIZE		
SAS kommando	26	
RESOLVE, SAS funksjon	205	
RETAIN, SAS-setning.....	73	
RETURN, SAS-setning	74,147	
REUSE=		
SAS datasett option	40	
system option.....	37	
REVERSE, SAS funksjon	195	
RFIND		
SAS kommando	25	
RGET		
SAS-setning.....	93	

RIGHT	
SAS funksjon.....	196
SAS kommando.....	26
ROUND, SAS funksjon	203,222
RSASUSER=	
system option.....	37
RSUBMIT	
SAS kommando.....	24,111,191,212
SAS-setning.....	6,7,84,93,111,191,212
RUN, SAS-setning.....	41,65,74
RXCHANGE	
SAS call rutine.....	48,196
RXFREE	
SAS call rutine.....	48
RXMATCH	
SAS funksjon.....	196
RXPARSE	
SAS funksjon.....	196
RXSUBSTR	
SAS call rutine.....	48,196

S

S=	
system option.....	37
S2=	
system option.....	38
sammenhekting	28,44
sammenligne to SAS datasett.....	97
sammenlignende operatorer	28
sannsynlighetsfunksjoner	208
SAS call rutiner.....	48,207
EXECUTE.....	48
LABEL	10,48
RANBIN.....	48
RANCAU	48
RANEXP	48
RANGAM	48
RANNOR	48
RANPOI	48
RANTBL	48
RANTRI	48
RANUNI.....	48
RXCHANGE.....	48,196
RXFREE.....	48
RXSUBSTR	48,196
SOUND	48
SYMPUT.....	48,81,95
SYSTEM	49
VNAME	49
SAS dataformater	30
SAS datasett	
aggregere	129
endre	25,124
endre attributter til	107,110
endre navn på.....	107
endre variabelnavn.....	110
hekte sammen	96
indeksere.....	40,103,104,107,109,111,156,162,191
innhold i	103,109
koble	69
kopiere	106,107,109
lage	41,50,90
lagring av	86
lese.....	76

navnsetting	23
oppdatere	78
reparere.....	107,110
sammenligne.....	97
se på.....	25,124
skrive ut	135
slette	107,109
transponere	180
SAS datasett options	39
SAS dato	12,28,30,32,35,111,191,200
SAS editor.....	27
SAS expressions	5
SAS funksjoner	193
ABS	202
AIRY	210
ARCOS.....	210
ARSIN	210
ATAN.....	210
BAND.....	211
BETAINV	209
BOR.....	211
BXOR.....	211
BYTE	193
CEIL	203
CINV	209
COLLATE	193
COMPBL	193
COMPOUND.....	211
COMPRESS	12,194
COS	210
COSH	210
CSS.....	199
CV	199
DAIRY	210
DATDIF	201
DATE	201
DATEJUL.....	201
DATEPART	201
DATETIME.....	201
DAY	201
DEQUOTE	194
DHMS	201
DIGAMMA	210
DIM	9,42,51,202
ERF	210
ERFC.....	210
EXIST	203
EXP	210
FACT.....	210
FEXIST	203
FILEEXIST	203
FILENAME.....	203
FILeref	204
FINV	209
FLOOR.....	203
FUZZ.....	203
GAMINV.....	209
GAMMA	210
GETOPTION.....	204
HBOUND.....	202
HMS	201
HOUR.....	201
INDEX	194
INDEXC.....	195
INDEXW.....	195
INPUT	12,204

INT	203	SINH	210
INTCK	201	SKEWNESS	199
INTNX	201	SOUNDEX	197
JULDATE	201	SPEDIS	205
JULDATE7	201	SQRT	202
KURTOSIS	199	STD	199
LBOUND	202	STDERR	199
LEFT	195	SUBSTR	44,197,202,214
LENGTH	195	SUM	199
LGAMMA	210	SYMGET	205
LIBNAME	204	SYSGET	206
LIBREF	204	SYSPARM	38,206
LOG	210	SYSTEM	206
LOG10	210	TAN	210
LOG2	210	TANH	210
LOWCASE	195	TIME	201
MAX	199	TIMEPART	201
MDY	201	TINV	210
MEAN	199	TODAY	201
MIN	199	TRANSLATE	198
MINUTE	201	TRANWRD	198
MOD	202	TRIGAMMA	210
MONTH	201	TRIM	198
N	199	TRIMN	198
NMISS	199	TRUNC	203
NORMAL	207	UNIFORM	208
ORDINAL	200	UPCASE	198
PATHNAME	205	USS	199
POISSON	208	VAR	199
PROBBETA	208,209	VERIFY	198
PROBBNML	208	WEEKDAY	201
PROBCHI	208,209	YEAR	201
PROBF	208,209	YRDIF	201
PROBGAM	208,209	YYQ	201
PROBHYP	208	SAS keyword	5
PROBIT	210	SAS kommandoer	23,84
PROBNEGB	209	?	24
PROBNORM	209,210	ACCESS	25
PROBT	209,210	BFIND	25
PUT	12,21,22,149,205	BOT	26
PUTC	205	BYE	24
PUTN	205	CANCEL	26
QUOTE	195	CASCADE	26
QTR	201	CHANGE	25
RANBIN	207	CLEAR	24
RANCAU	207	COMMAND	26
RANEXP	207	CURPOS	26
RANGAM	207	CUT	24
RANGE	199	DIR	25
RANK	195	DLGFIN	25
RANNOR	207	DLGREPL	25
RANPOI	207	DOWN	26
RANTBL	207	END	24
RANTRI	208	FILE	24
RANUNI	208	FILENAME	25
REPEAT	195	FIND	25
RESOLVE	205	FOOTNOTE	25
REVERSE	195	FSLIST	25
RIGHT	196	FSVIEW	25
ROUND	203,222	HELP	24
RXMATCH	196	HOME	26
RXPARE	196	HSCROLL	26
SCAN	196	ICON	26
SECOND	201	INCLUDE	24
SIGN	202	KEYS	25
SIN	210	LEFT	26

LIB	25	DATA	41,50
LINESIZE	26	DATALINES	50
LOG	25	DATALINES4	50
MANAGER	25	DELETE	50
MARK	24	DISPLAY	50,80
NEXT	26	DM	6,7,84,147
NUM	24	DO	9,51,147
OPTIONS	25	DROP	52
OUT	25	END	52,147
PAGESIZE	26	ENDRSUBMIT	6,7,84,93,212
PASTE	24	ENDSAS	52
PCLEAR	24	ERROR	52
PGM	25	FILE	53
PMENU	26	FILENAME	6,11,84,111,191,203,205
PREVWIND	26	FOOTNOTE	54,85
RCHANGE	25	FORMAT	45,56,224
RECALL	24	GO TO	56,147
RESHOW	26	IF, subsetting	56
RESIZE	26	IF, THEN, ELSE	57,65,147
RFIND	25	INFILE	58
RIGHT	26	INFORMAT	45,61
RSUBMIT	24,111,191,212	INPUT	9,62
SIGNOFF	24	KEEP	65
SIGNON	24	kommentar	65,147
STORE	24	label	67
SUBMIT	24,84,93,111,191	LABEL	45,67
SUBTOP	26	LEAVE	68
TILE	26	LENGTH	10,45,68,147
TITLE	25	LIBNAME	11,86,203,205
TOP	26	LINK	68,147
UNDO	24	LIST	69
UNMARK	24	LOSTCARD	69
UP	26	MERGE	13,69
VAR	25	MISSING	70
VIEWTABLE	25	null	70
VSCROLL	26	ODS	87
WEDIT	25	OPTIONS	34,92
WPOPUP	26	OUTPUT	41,71
X	24	PAGE	93
ZOOM	24	PUT	9,71
SAS nøkkelord	5	RENAME	73
SAS operatorer	5,28	RETAIN	73
SAS statements	5	RETURN	74,147
SAS system options	34,92,134,204	RGET	93
SAS tidsverdier	28,30,32,35,200	RSUBMIT	6,7,84,93,111,191,212
sas7baud, filhale	108	RUN	41,65,74
sas7bdat, filhale	87	SELECT	75,147,149
sas7bndx, filhale	109	SET	11,20,76
SASAUTOS=		SKIP	77
system option	37	STOP	78
SASHELP=		sum	28,74,78,147
system option	37	SYSTASK	94
SAS-navn på et område	86	TITLE	55,95
SAS-setninger	5	UPDATE	15,78
%INCLUDE	58,147	WAITFOR	95
%LET	86	WHERE	79
%LIST	87	WINDOW	50,80
%PUT	93	X	95
ABORT	41	SASUSER=	
ARRAY	10,22,42	system option	37
assignment	28,44,73,78,147,193,197	SAS-uttrykk	5,28,44,48,51,56,57,63,72,78,79
ATTRIB	45	SAVE	
BY	13,33,45,69,77,78,79	proc datasets setning	107,110
CALL	48,147,207	SCAN, SAS funksjon	196
CARDS	49	se på eksterne filer	124
CARDS4	49	se på SAS datasett	124
CONTINUE	49		

SECOND, SAS funksjon	201	system option	38
sekvensielle filer	124	SORTSIZE=	
SELECT		system option	38
proc copy setning	106	SOUND, SAS call rutine	48
proc datasets setning	107,110	SOUNDEX, SAS funksjon	197
proc download setning	111,112	SOURCE=	
proc format setning	113,119	system option	38
proc sql setning	158,160,162	SOURCE2=	
proc upload setning	191,192	system option	38
SAS-setning	75,147,149	SPEDIS, SAS funksjon	205
selektare data	40,56,79	spesialfunksjoner	203
sentring av tekst	27,72	SPOOL=	
SEQ=		system option	38
system option	37	SPSS-filer	11,86
SERROR=		SQL	158
system option	37	SQL, prosedyre	19,158,216,217
SET, SAS-setning	11,20,76	SQRT, SAS funksjon	202
SET=		standard format for SSB	224
system option	37	standardavvik	186,199,223
SETINIT=		standardfeil	199,223
system option	37	starte et DATA-steg	50
sidestørrelse	26,37,55	statistiske beregninger	126,185
SIGN, SAS funksjon	202	statistiske funksjoner	199
SIGNOFF		statistiske nøkkelord	123,127,142,172,176,188,223
SAS kommando	24	STD	
SIGNON		SAS funksjon	199
SAS kommando	24	statistisk nøkkelord	127,142,172,188,223
SIN, SAS funksjon	210	STDERR	
SINH, SAS funksjon	210	SAS funksjon	199
sjekke om SAS-datasett eksister	83	statistisk nøkkelord	127,142,172,188,223
SKEWNESS		STIMER=	
SAS funksjon	199	system option	38
statistisk nøkkelord	127,188,223	STOP, SAS-setning	78
skilletegnseparerte filer	8	stoppe DATA-steget	35,41,55,61,78
SKIP, SAS-setning	77	stoppe et program	219
SKIP=		STORE	
system option	37	SAS kommando	24
skrive inndata til SASlog	69	Student's t	209,210,223
skrive til SASlog	71	SUBMIT	
skrive til sekvensiell fil	71	SAS kommando	24,84,93,111,191
skrive ut observasjoner	135	SUBSTR, SAS funksjon	44,197,202,214
skrive ut observasjoner til SAS datasett	71	SUBTOP	
slette SAS datasett	107,109	SAS kommando	26
SOLUTIONS=		SUM	
system option	37	proc print setning	135,136
SORT, prosedyre	156	SAS funksjon	199
SORTCUT=		statistisk nøkkelord	127,142,145,171,172,188,223
system option	38	sum, SAS-setning	28,74,78,147
SORTCUTP=		SUMBY	
system option	38	proc print setning	135,136
SORTDEV=		SUMWGT	
system option	38	statistisk nøkkelord	127,142,172,188,223
SORTDUP=		SYMBOLGEN=	
system option	38	system option	38
SORTEDBY=, SAS datasett option	40	SYMGET, SAS funksjon	205
sortere datasett	156	SYMPUT, SAS call rutine	48,81,95
sorteringsrekkefølge	226	SYSGET, SAS funksjon	206
SORTLIB=		SYSIN=	
system option	38	system option	38
SORTNAME=		SYSLEAVE=	
system option	38	system option	38
SORTPARM=		SYSPARM, SAS funksjon	38,206
system option	38	SYSPARM=	
SORTPGM=		system option	38,206
system option	38	SYSPRINT=	
SORTSEQ=		system option	38

SYSTASK		MSGCASE=	36
SAS-setning	94	MSGLEVEL=	37
SYSTEM		MSTORED=	37
SAS call rutine	49	NEWS=	37
system options	34,92,134,204	NOTES=	37
_LAST=	36	NUMBER=	37
ALTLOG=	34	OBS=	37
ALTPRINT=	34	PAGENO=	37
AUTOEXEC=	34	PAGESIZE=	37
BATCH=	34	PARM=	37
BUFNO=	34	PARMCARDS=	37
BUFSIZE=	34	PATH=	37
BYERR=	34	PRINT=	37
BYLINE=	34	PRINTCMD=	37
CAPS=	34	PROBSIG=	37
CATCACHE=	34	PROC=	37
CENTER=	34	PROCLEAVE=	37
CLEANUP=	34	REPLACE=	37
COMAMID=	6,34	REUSE=	37
COMDEF=	34	RSASUSER=	37
COMPRESS=	34	S=	37
CONFIG=	35	S2=	38
CONNECTREMOTE=	35	SASAUTOS=	37
CPUID=	35	SASHELP=	37
DATASTMTCHK=	35	SASUSER=	37
DATE=	35	SEQ=	37
DETAILS=	35	SERROR=	37
DEVICE=	35	SET=	37
DFLANG=	35	SETINIT=	37
DKRICOND=	35	SKIP=	37
DKROCOND=	35	SOLUTIONS=	37
DMS=	35	SORTCUT=	38
DSNFERR=	35	SORTCUTP=	38
ECHO=	35	SORTDEV=	38
ECHOAUTO=	35	SORTDUP=	38
EDITCMD=	35	SORTLIB=	38
ENGINE=	35	SORTNAME=	38
ERRORABEND=	35	SORTPARM=	38
ERRORCHECK=	35	SORTPGM=	38
ERRORS=	35	SORTSEQ=	38
FIRSTOBS=	36	SORTSIZE=	38
FMTERR=	36,56	SOURCE=	38
FMTSEARCH=	36	SOURCE2=	38
FONT=	36	SPOOL=	38
FORMCHAR=	36,140	STIMER=	38
FORMDLIM=	36	SYMBOLGEN=	38
FORMS=	36	SYSIN=	38
FULLSTIMER=	36	SYSLEAVE=	38
INITCMD=	36	SYS Parm=	38,206
INITSTMT=	36	SYS PRINT=	38
INVALIDDATA=	36	TERMINAL=	38
LABEL=	36	TOOLDEF=	38
LINESIZE=	36,212	USER=	38
LOG=	36	VALIDVARNAME=	38
MACRO=	36	VERBOSE=	38
MAPS=	36	VNFERR=	38
MAUTOSOURCE=	36	WORK=	38
MEMSIZE=	36	WORKINIT=	38
MERGENOBY=	36	WORKTERM=	39
MERROR=	36	XCMD=	39
MFILE=	36	XSYNC=	39
MISSING=	36	XWAIT=	39
MLOCIG=	36	YEARCUTOFF=	39
MPRINT=	36	SYSTEM, SAS funksjon	206
MRECALL=	36	søk og erstatt	25
MSG=	36	søke i tekst	25

T

T, statistisk nøkkelord	127,142,172,188,223
TABLE	
proc tabulate setning	165,172,173
TABLES	
proc freq setning	121
TABULATE, prosedyre	33,75,115,120,139,145,165,204,220
TAN, SAS funksjon	210
TANH, SAS funksjon	210
tegnsett	226
ansi	226
ascii	226
ebcdic	226
tekst istedenfor kode	113
tekst til variable	67,104,110,135,180,216
TEMPLATE, prosedyre	88,89,92,219
TERMINAL=	
system option	38
TEST	
proc freq setning	121,124
tidsvariable	11,30,32,35,200
tidsverdier	28,30,32,35,200
TILE	
SAS kommando	26
tilfeldige tallsfunksjoner	48,207
tilordnings-setningen	44,147
TIME, SAS funksjon	201
TIMEPART, SAS funksjon	201
TINV	
SAS funksjon	210
TITLE	
SAS kommando	25
SAS-setning	55,95
TOBSNO=, SAS datasett option	40
TODAY, SAS funksjon	201
TOOLDEF=	
system option	38
TOP	
SAS kommando	26
TRANSLATE, SAS funksjon	198
transponere SAS datasett	180
TRANSPOSE, prosedyre	180
TRANWRD, SAS funksjon	198
trekking med tilbakelegging	76
trekking uten tilbakelegging	207
TRIGAMMA, SAS funksjon	210
trigonometriske og hyperboliske funksjoner	210
TRIM, SAS funksjon	198
TRIMN, SAS funksjon	198
TRUNC, SAS funksjon	203
trunkeringsfunksjoner	203
tusenskilte	225
TYPE=, SAS datasett option	40
tømme innholdet i et vindu	24

U

UNDO	
SAS kommando	24
UNIFORM	
SAS funksjon	208
UNIVARIATE, prosedyre	185

UNMARK	
SAS kommando	24
UP	
SAS kommando	26
UPCASE, SAS funksjon	198
UPDATE	
proc sql setning	158,162
SAS-setning	15,78
UPLOAD, prosedyre	6,191
USER=	
system option	38
USS	
SAS funksjon	199
statistisk nøkkelord	127,142,172,188,223
utelate variable	39,52
utføre en operativsystem-kommando	95
utføre SAS kommando	84
utskriftsformater	31

V

VALIDATE	
proc sql setning	158,162
VALIDVARNAME =	
system option	38
VALUE	
proc format setning	113,119
VAR	
proc compare setning	97,100
proc fsview setning	124,125
proc means setning	126,128
proc print setning	135,136
proc rank setning	138
proc tabulate setning	165,171
proc transpose setning	180,181
proc univariate setning	185,189
SAS funksjon	199
SAS kommando	25
statistisk nøkkelord	127,142,172,188,223
VARDEF	127,142,168,187,223
varians	127,142,168,187,199,223
variasjonskoeffisient	199,223
veid kvadratavvik	199,223
veid kvadratsum	199,223
veiling	124,134,153,165,176,189,223
venstrejustering	27,31,72,104,113,114,117,195,214
VERBOSE=	
system option	38
VERIFY, SAS funksjon	198
VIEWTABLE	
SAS kommando	25
vise et vindu på skjermen	50
VNAME, SAS call rutine	49
VNFERR=	
system option	38
VSCROLL	
SAS kommando	26

W

WAITFOR	
SAS-setning	95
WEDIT	

SAS kommando	25
WEEKDAY, SAS funksjon	201
WEIGHT	
proc freq setning	121,124
proc means setning	126,134
proc report setning	140,153
proc tabulate setning	165,176
proc univariate setning	185,189
WHERE	
proc append setning	96
proc compare setning	97,100
proc download setning	111,112
proc freq setning	121
proc fsview setning	124,125
proc means setning	126,127
proc print setning	135,136
proc rank setning	138
proc report setning	140,142
proc sort setning	156,157
proc tabulate setning	165,168
proc transpose setning	180
proc univariate setning	185,187
proc upload setning	191,192
SAS-setning	79
WHERE=, SAS datasett option	40,76,77
WINDOW, SAS-setning	50,80
WITH, proc compare setning	97,100
WORK=	
system option	38
WORKINIT=	
system option	38
WORKTERM=	
system option	39
WPOPUP	
SAS kommando	26
WRITE=, SAS datasett option	40

X

X	
SAS kommando	24
SAS-setning	95
XCMD=	
system option	39
XSYNC=	
system option	39
XWAIT=	
system option	39

Y

YEAR, SAS funksjon	201
YEARCUTOFF=	
system option	39
YRDIF, SAS funksjon	201
YYQ, SAS funksjon	201

Z

ZOOM	
SAS kommando	24

Ø

ødelagte SAS datasett	107,110,219
-----------------------------	-------------

De sist utgitte publikasjonene i serien Statistisk sentralbyrås håndbøker

- | | |
|--|---|
| <p>45 Håndbok i datasikkerhet og fysisk sikring. Revidert utgave, november 1998. 1998. 83s.</p> <p>46 Telefonkatalog. 1998. 89s.</p> <p>47 EØS-avtalen. Det statistiske samarbeid og konsekvenser for Statistisk sentralbyrås statistikkproduksjon. 1994. 55s.</p> <p>48 Håndbok i tilsettingssaker. 1994. 32s.</p> <p>49 Oppgaveplikt og tvangsmulkt. 1995. 55s.</p> <p>50 Emneinndeling 1995. 1995. 43s.</p> <p>51 Intervju: EDB-arbeidsbok. 1995.</p> <p>52 Intervju: EDB-oppslagsbok. 1995.</p> <p>53 Intervju: Opplæring og administrasjon. 1995.</p> <p>54 Internkontroll: Revidert utgave 1997. 25s.</p> <p>55 Nordisk statistikk på CD-ROM: Veiledning. 20s.</p> <p>56 PC-Axis versjon 2.2: Brukerhåndbok. 69s.</p> <p>57 Produktregister versjon 4.0: Bruerveiledning. 49s.</p> | <p>58 Håndbok i prosjektstyring. 20s.</p> <p>59 Personalreglement for Statistisk sentralbyrå. 22s.</p> <p>60 Produktnummerkatalog pr. 28.02.1996. 55s.</p> <p>61 Innkjøpshåndbok. 1996.</p> <p>62 Timeplan versjon 3.0: Bruerveiledning. 16s.</p> <p>63 Håndbok i EDB-metode. 52s.</p> <p>64 Publiseringshåndbok: Regler og retningslinjer for publisering i Statistisk sentralbyrå. 93s.</p> <p>65 Håndbok i utvikling av statistikksystemer: Med vekt på IT-metode. 52s.</p> <p>66 Håndbok i datarevisjon. 48s.</p> <p>67 Arkivnøkkel for Statistisk sentralbyrå. 76s.</p> <p>68 Rapporteringshåndbok for KOSTRA-regnskap 1999: Oppslagshefte til hjelp ved filuttrekk for KOSTRA-rapportering. 52s.</p> <p>69 Yrkeskatalog for innrapportering av yrke til arbeidstakerregisteret. 86s.</p> <p>70 Håndbok for KOSTRA-rapportering 2000. Oppslagshefte til hjelp ved filuttrekk for KOSTRA-rapportering. 74s.</p> |
|--|---|

Statistisk sentralbyrå

Oslo:
Postboks 8131 Dep.
0033 Oslo

Telefon: 22 86 45 00
Telefaks: 22 86 49 73

Kongsvinger:
2225 Kongsvinger

Telefon: 62 88 50 00
Telefaks: 62 88 50 30

