Statistisk sentralbyrå
Statistics Norway

Dokumentasjon av pre-system

Python-pakke med formel- og forsystemklasser, samt tidsseriefunksjoner

Magnus Kvåle Helliesen

TALL SOM FORTELLER
NOTATER / DOCUMENTS
2024/9

I serien Notater publiseres dokumentasjon, metodebeskrivelser, modellbeskrivelser og standarder.

Publisert: 8. februar 2024

ISBN 978-82-587-1911-0 (elektronisk)
ISSN 2535-7271 (elektronisk)

Standardtegn i tabeller	Symbol
Ikke mulig å oppgi tall Tall finnes ikke på dette tidspunktet fordi kategorien ikke var i bruk da tallene ble samlet inn.	.
Tallgrunnlag mangler Tall er ikke kommet inn i våre databaser eller er for usikre til å publiseres.	..
Vises ikke av konfidensialitetshensyn Tall publiseres ikke for å unngå å identifisere personer eller virksomheter.	:
Desimaltegn	,

Forord

Statistisk sentralbyrå er i ferd med å gå fra proprietær programvare som kjøres på egne servere til åpen kildekode-språk som kjøres på Google Cloud. Vi må derfor oversette alle programmer til Python- og/eller R-kode. Nasjonalregnskapet benytter et såkalt *forsystem* for å utarbeide sine korttidsregnskaper, der en sentral komponent er *formler* (helt generelt et objekt som tar imot input og returnerer output). Formlene sammenstiller data (Årlig nasjonalregnskap, indikatorer og vekter) med ulik frekvens, og ekstrapolerer nasjonalregnskapsvariable fra et basisår. Å implementere slike formler med funksjonaliteten som finnes i for eksempel Pandas vil være krevende og lede til svært mange linjer med kode. Statistisk sentralbyrå har derfor utviklet denne pakken med formelfunksjonalitet internt. Pakken inneholder en rekke *formel*objekter, og et *forsystem*objekt som organiserer formlene. I tillegg inneholder den funksjoner med tidsseriefunksjonalitet som brukes mye i dagens forsystem. Pakken vedlikeholdes på GitHub.

Statistisk sentralbyrå, 25.01.2024

Lasse Sandberg

Sammendrag

Dette notatet inneholder dokumentasjon av pakken *pre-system*. Pakken inneholder en rekke *formel*objekter som kan brukes til å bygge nødvendige formler som inngår i et *forsystem* i Nasjonalregnskapets korttidsstatistikker. Pakken inneholder også et *forsystem*objekt som kan organisere formelobjekter og data. Formel- og forsystemobjektene gjør det enklere å definere, inspisere og evaluere forsystemer bestående av formler. Pakken inneholder også tidsseriefunksjoner som er nyttige i byggingen og bruken av forsystemer.

Pakken er tenkt benyttet når Månedlig- og Kvartalsvis nasjonalregnskap, Kvartalsvis institusjonelt sektorregnskap og Varekonsumindeksen legges over til åpen kildekode-språk.

Notatet er organisert slik at først gjennomgås hvert formelobjekt, deretter forsystemobjektet, og avslutningsvis tidsseriefunksjonene. Hvert objekt eller funksjon beskrives, både med hensyn til innhold og bruk.

Innhold

1. Formula-klassen

Klassen `Formula` er en *foreldre*klasse som alle andre formler springer ut fra, og arver egenskapene (attributtene og metodene) til. `Formula`-klassen har attributtene

- `name`: tekststreng med formelens navn (kan ikke endres etter initialisering)
- `baseyear`: formelens basisår (kan endres etter initialisering)
- `what`: formelen skrevet som tekst (bestemmes av hvordan formelen er definert)
- `calls_on`: liste med formler som formelen kaller på (bestemmes av hvordan formelen er definert; listen er tom hvis formelen ikke kaller på andre formler)
- `indicators`: liste med indikatorer som inngår i formelen (bestemmes av hvordan formelen er definert)
- `weights`: liste med vekter som inngår i formelen (bestemmes av hvordan formelen er definert)

og metodene

- `info()`: skriver ut beskrivelse av hva formelen er, og sporer seg rekursivt bakover til hvilke andre formler som inngår i formelen (om noen) og skriver ut hva de er
- `indicators_weigths(trace=True)`: skriver ut indikatorer og vekter som inngår i formelen, og alle formler som formelen kaller på (om noen)
- `evaluate(annual_df, indicators_df[, weights_df, correction_df])`: evaluerer formelen, gjenstand for `DataFrame`'s med data. Dataene må ha Pandas `PeriodIndex` med årlig frekvens (for `annual_df` og `weights_df`) og kvartalsvis eller månedlig (for `indicators_df` og `correction_df`). Merk at `DataFrame`'ene i klammer er frivillige, og *må kun* spesifiserers dersom den aktuelle formelen inneholder vekter eller korreksjoner (se under)

`Formula`-klassen konverterer alle variabelnavn til minuskel. Det vil si at `evaluate`-metoden kun finner kolonner med kolonnenavn med små bokstaver i `DataFrame`'ene brukeren gir som input. Dette er hensyntatt i `PreSystem`-klassen (se under) ved at enhver `DataFrame` som lastes inn får kolonnenavn konvertert til små bokstaver.

`Formula`-klassen har en rekke *barne*klasser (under) som sammen *burde* dekke alle behov som melder seg når en bygger/definerer formler i et forsystem. Dersom funksjonaliteten ikke dekker behovet *kan* det være et signal om at den aktuelle formelen burde forenkles. Klassen krever med andre ord en viss grad av sparsommelighet (eller tarvelighet) av brukeren: man må spørre seg «kan denne formelen gjøres enklere?». Det kan også være at man må gjøre litt forarbeid i `Pandas`, slik at års- og indikatorinputen blir som den skal (for eksempel tidsforskyving, bevegelige snitt mv.).

1.1. Indicator-klassen

Definisjon

Klassen `Indicator` er en *barne*klasse av `Formula`-klassen. Den representerer formler på den generelle formen

$$x_t = \begin{cases} X_B \dfrac{k_t \sum_{i \in I} w_{i,B} \bar{I}_{i,t}}{\sum_{s \in B} k_s \sum_{i \in I} w_{i,B} \bar{I}_{i,s}} & \text{hvis } \textit{summen} \text{ av indikatoren skal være avstemt i basisåret,} \\ X_B \dfrac{k_t \sum_{i \in I} w_{i,B} \bar{I}_{i,t}}{\frac{1}{\lvert B \rvert} \sum_{s \in B} k_s \sum_{i \in I} w_{i,B} \bar{I}_{i,s}} & \text{hvis } \textit{gjennomsnittet} \text{ av indikatoren skal være avstemt i basisåret.} \end{cases}$$

der

$$\bar{I}_{i,t} = \begin{cases} I_{i,t} & \text{hvis indikatorene ikke skal normaliseres,} \\ I_{i,t} \Big/ \sum_{s \in B} I_{i,s} & \text{ellers.} \end{cases}$$

Formelen er altså en framskriving av et årsnivå X_B (fra basisåret B) ved hjelp av noen indikatorer $I_{i,t}$ som er/kan være vektet sammen med vekter $w_{i,B}$ (også fra basisåret). Forløpet kan korrigeres med korreksjonsfaktoren k_t. Formelen er avstemt mot X_B i basisåret «by construction» (dette kan man se ved å putte hele formelen inni en sum over alle $t \in B$ og observere at brøken alltid summerer til 1 i basisåret).

Initialisering

Klassen initialiseres ved

```
Indicator(name, annual, indicators[, weights, correction, **kwargs])
```

der de påkrevde argumentene er

- `name`: tekststreng med navnet på formelen (x i formelen over)
- `annual`: tekststreng med navnet på årsserien som skal fremskrives (X i formelen over)
- `indicators`: liste med tekststrenger med navn på indikator(er) som skal brukes til å fremskrive årsnivået (I_i i formelen over)

De valgfrie argumenter er

- `weights`: liste med tekststrenger med navn på vekter(er) *eller* tallverdier (w_i i formelen over)
- `correction`: tekststreng med navn på korreksjonsfaktor (k i formelen over)
- `**kwargs`:
  - `normalise`: sannhetsverdi (`False`, om ikke annet er gitt) som bestemmer om hver indikator normaliseres før de inngår i indikatorformelen
  - `aggregation`: er enten '`sum`' (om ikke annet er gitt) eller '`avg`', og bestemmer om summen eller gjennomsnittet av indikatoren skal være avstem i basisåret

Eksempel

Vi kan lage en variabel `x` som holder på indikatoren

$$x_t = X_B \frac{x_{1,t} + x_{2,t}}{\sum_{s \in B} x_{1,s} + x_{2,s}}$$

som følger:

```
x = Indicator('x', 'x', ['i1', 'i2'])
```

Her den første `x`'en navnet på formelen vi lager, og den andre navnet på årsserien formelen ekstrapolerer fra (begge er skrevet med små bokstaver, siden `Formula`-objektet, som sagt, kun benytter små bokstaver)[1].

Vi kan se hvordan formelen er definert som følger (her er kall og resultater vist sammen):

```
x.what

'x*<date None>*(i1+i2)/sum((i1+i2)<date None>)'
```

Basisåret er foreløpig ikke satt og vises som objektet `None`. Basisåret må settes før man kan kalle på metoden `evaluate()`.

Vi kan også kalle på attributter og metoder for å vise indikatorer og vekter som følger (her er kall og resultater vist sammen):

```
x.indicators

['i1', 'i2']

x.weights

[1, 1]

x.indicators_weights()

[('i1', 1), ('i2', 1)]
```

1.2. FDeflate-klassen

Definisjon

Klassen `FDeflate` er en *barne*klasse av `Formula`-klassen. Den representerer formler på formen

$$y_t = \left(\sum\nolimits_{s\in B} x_s\right) \frac{k_t\, x_t / \sum_{i\in I} w_{i,B} \bar{I}_{i,t}}{\sum_{s\in B} k_s\, x_s / \sum_{i\in I} w_{i,B} \bar{I}_{i,s}},$$

hvor $\bar{I}_{i,t}$ er definert som over. Formelen y_t er altså formelen x_t deflatert med indikatorene $I_{i,t}$ som er/kan være vektet sammen med vekter $w_{i,B}$ (også fra basisåret). Forløpet kan korrigeres med korreksjonsfaktoren k_t. Formelen er avstemt mot x (altså formelen den deflaterer) i basisåret «by construction» (dette kan man se på samme måte som over).

[1] Merk at variabelen som peker på `Indicator`-objektet (til venstre for likhetstegnet) også heter `x`; den kan i prinsippet hete hva som helst, siden Python-objekter uansett ikke kan vite hvilke variable som peker på den. Når/dersom formelobjektene organiseres i forsystemobjekter vil man uansett ikke ha bruk for variable som peker på formlene.

Initialisering

Klassen initialiseres ved

```
FDeflate(name, formula, indicators[, weights, correction, **kwargs])
```

der de påkrevde argumentene er

- `name`: tekststreng med navnet på formelen (y i formelen over)
- `formula`: formel som skal deflateres (x i formelen over)
- `indicators`: liste med tekststrenger med navn på indikator(er) som skal brukes til å fremskrive årsnivået (I_i i formelen over)

De valgfrie argumenter er

- `weights`: liste med tekststrenger med navn på vekter(er) *eller* tallverdier (w_i i formelen over)
- `correction`: tekststreng med navn på korreksjonsfaktor (k i formelen over)
- `**kwargs`:
  - `normalise`: sannhetsverdi (`False`, om ikke annet er gitt) som bestemmer om hver indikator normaliseres før de inngår i indikatorformelen

Eksempel

Vi kan lage en variabel `y` som holder på indikatoren

$$y_t = \left(\sum\nolimits_{s\in B} x_s\right) \frac{x_t/(w_{1,B}p_{1,t} + w_{2,B}p_{2,t})}{\sum_{s\in B} x_s/(w_{1,B}p_{1,s} + w_{2,B}p_{2,s})}$$

som følger:

```
y = FDeflate('y', x, ['p1', 'p2'], ['w1', 'w2'])
```

Vi kan se hvordan formelen er definert som følger (her er kall og resultater vist sammen):

```
y.what

'sum(x<date None>)*(x/(w1*p1+w2*p2))/sum(x/(w1*p1+w2*p2)<date None>)'
```

Vi kan også kalle på attributter og metoder for å vise indikatorer og vekter som følger (her er kall og resultater vist sammen):

```
y.what

'sum(x<date None>)*(x/(w1*p1+w2*p2))/sum(x/(w1*p1+w2*p2)<date None>)'

y.indicators

['p1', 'p2']

y.weights

['w1', 'w2']

y.indicators_weights(trace=False)

[('p1', 'w1'), ('p2', 'w2')]

y.indicators_weights(trace=True)

[('p1', 'w1'), ('p2', 'w2'), ('i1', 1), ('i2', 1)]
```

Merk at opsjonen `trace` (`True`/`False`) bestemmer om man skal spore tilbake indikatorer og vekter som inngår i formler lenger ned i hierarkiet.

1.3. FInflate-klassen

Definisjon
Klassen `FInflate` er en *barne*klasse av `Formula`-klassen. Den representerer formler på formen

$$y_t = \left(\sum\nolimits_{s\in B} x_s\right)\frac{k_t x_t \sum_{i\in I} w_{i,B}\bar{I}_{i,t}}{\sum_{s\in B} k_s x_s \sum_{i\in I} w_{i,B}\bar{I}_{i,s}},$$

hvor $\bar{I}_{i,t}$ er definert som over. Formelen y_t er altså formelen x_t inflatert med indikatorene $I_{i,t}$ som er/kan være vektet sammen med vekter $w_{i,B}$ (også fra basisåret). Forløpet kan korrigeres med korreksjonsfaktoren k_t. Formelen er avstemt mot x (altså formelen den inflaterer) i basisåret «by construction».

Initialisering
Klassen initialiseres som FDeflate.

1.4. FSum-klassen

Definisjon
Klassen `FSum` er en *barne*klasse av `Formula`-klassen. Den representerer formler på formen

$$x_t = x_{1,t} + x_{2,t} + \cdots + x_{N,t},$$

der $x_{i,t}$ er formler.

Initialisering
Klassen initialiseres ved

```
FSum(name, formula1, formula2, formula3, …)
```

der de påkrevde argumentene er

- `name`: tekststreng med navnet på formelen (x i formelen over)
- `formulae`: formler som skal summeres (x_i i formelen over)

1.5. FSumProd-klassen

Definisjon
Klassen `FSumProd` er en *barne*klasse av `Formula`-klassen. Den representerer formler på formen

$$x_t = w_{1,B}x_{1,t} + w_{2,B}x_{2,t} + \cdots + w_{N,B}x_{N,t},$$

der $x_{i,t}$ er formler og $w_{i,B}$ er koeffisienter.

Initialisering
Klassen initialiseres ved

```
FSumProd(name, formulae, weights)
```

der de påkrevde argumentene er

- `name`: tekststreng med navnet på formelen (x i formelen over)
- `formulae`: liste med formler som skal vektes sammen (x_i i formelen over)
- `weights`: liste med tekststrenger med navn på vekter(er) *eller* tallverdier (w_i i formelen over)

1.6. FMult-klassen

Definisjon
Klassen `FMult` er en *barne*klasse av `Formula`-klassen. Den representerer formler på formen

$$x_t = x_{1,t} \cdot x_{2,t},$$

der $x_{1,t}$ og $x_{2,t}$ er formler.

Initialisering
Klassen initialiseres ved

```
FMult(name, formula1, formula2)
```

der de påkrevde argumentene er

- `name`: tekststreng med navnet på formelen (x i formelen over)
- `formula1 og formula2`: formler som skal multipliseres sammen (hhv. x_1 og x_2 i formelen over)

1.7. FDiv-klassen

Definisjon

Klassen `FDiv` er en *barne*klasse av `Formula`-klassen. Den representerer formler på formen

$$x_t = \frac{x_{1,t}}{x_{2,t}},$$

der $x_{1,t}$ og $x_{2,t}$ er formler.

Initialisering

Klassen initialiseres ved

```
FDiv(name, formula1, formula2)
```

der de påkrevde argumentene er

- `name`: tekststreng med navnet på formelen (x i formelen over)
- `formula1 og formula2`: formler som skal divideres (hhv. x_1 og x_2 i formelen over)

1.8. FJoin-klassen

Definisjon

Klassen `FJoin` er en *barne*klasse av `Formula`-klassen. Den representerer formler på formen

$$x_t = \begin{cases} x_{1,t} & \text{hvis } t \in T \geq T_1, \\ x_{0,t} & \text{ellers.} \end{cases}$$

der $x_{1,t}$ og $x_{0,t}$ er formler. Dette betyr at x_t er lik $x_{1,t}$ for perioder t som er i år T_1 eller senere; ellers er x_t lik $x_{0,t}$.

Initialisering

Klassen initialiseres ved

```
FJoin(name, formula1, formula0, from_year)
```

der de påkrevde argumentene er

- `name`: tekststreng med navnet på formelen (x i formelen over)
- `formula1 og formula0`: formler som skal settes sammen (hhv. x_1 og x_0 i formelen over)
- `from_year`: året de formlene skal skjøtes sammen i (T_1 i formelen over)

1.9. MultCorr-klassen

Definisjon

Klassen `MultCorr` er en *barne*klasse av `Formula`-klassen. Den representerer formler på formen

$$\left(\sum\nolimits_{s\in B} x_s\right)\frac{k_t x_t}{\sum_{s\in B} k_s x_s},$$

der x_t er en formel og k_t er en korreksjonsfaktor. Resultatet er en korrigert formel som er avstemt mot x i basisåret.

Initialisering

Klassen initialiseres ved

```
MultCorr(formula, correction)
```

der argumentene er

- `formula`: formel som skal korrigeres.
- `correction`: tekststreng med navn på korreksjonsfaktor (k i formelen over).

`MultCorr`-instansen arver navnet til indikatoren den skal korrigere. Dermed kan man definere formelen og korreksjonen i én og samme operasjon.

Eksempel

Vi kan lage en variabel `x` som holder på indikatoren

$$x_t = x_{1,t} + x_{2,t} + x_{3,t},$$

der $x_{1,t}$, $x_{2,t}$ og $x_{3,t}$ er formler, korrigert med k_t som følger:

```
x = MultCorr(FSum('x', x1, x2, x3), 'k')
```

Merk at `x` da tar navnet fra `FSum`-formelen.

1.10. AddCorr-klassen

Definisjon

Klassen `MultCorr` er en *barne*klasse av `Formula`-klassen. Den representerer formler på formen

$$x_t + k_t - \frac{1}{|B|}\sum_{s\in B} k_s,$$

der x_t er en formel og k_t er en korreksjonsfaktor. Resultatet er en korrigert formel som er avstemt mot x i basisåret.

Initialisering er identisk som for `MultCorr`.

2. PreSystem-klassen

`PreSystem` er en klasse som holder på *instanser/forekomster* av `Formula`-klassen. Klassen initialiseres ved

```
forsystem = PreSystem(name)
```

der `name` er et beskrivende navn for forsystemet. Når forsystemet er initialisert, kan man se attributtene

- `name`: tekststreng med forsystemets navn (kan ikke endres)
- `baseyear`: forsystemets basisår (kan endres)
- `formulae`: ordbok (dictionary) med som inneholder formelnavn (nøkkel) og -instanser (verdi)
- `indicators`: liste med indikatorer som inngår i formlene i forsystemet (bestemmes av hvordan formlene er definert)
- `annuals_df`: Pandas `DataFrame` med alle årstall som brukes av formlene i forsystemet (kan endres)
- `indicators_df`: Pandas `DataFrame` med alle indikatorer som brukes av formlene i forsystemet (kan endres)
- `weights_df`: Pandas `DataFrame` med alle vekter som brukes av formlene i forsystemet (kan endres)
- `corrections_df`: Pandas `DataFrame` med alle korreksjoner som brukes av formlene i forsystemet (kan endres)

og kalle på metodene

- `info()`: skriver ut informasjon om forsystemet, herunder basisår, antall formler, hvorvidt og evt. når `DataFrame`'ene er lastet inn
- `add_formula(x)`: legger formelen som variabelen `x` «peker på» inn i forsystemet. Merk at alle formler som eventuelt inngår i `x` (dersom `x` er en deflatering/inflatering/sum etc.) *må* lastes i forsystemet *først*, ellers får man en feilmelding. Formelen lagres i forsystemet med navnet den er gitt (attributten `name`), hvilken variabel som peker på den (her `x`) har ingenting å si (dette har å gjøre med hvordan Python fungerer som programmeringsspråk)
- `formula(name)`: viser formelen ved navn `name`
- `evaluate()`: evaluerer alle formlene gjenstand for `DataFrame`'ene i forsystemet og returnerer en Pandas `DataFrame`
- `evaluate_formula(navn)`: evaluerer formelen ved navn `name` gjenstand for `DataFrame`'ene i forsystemet og returnerer en Pandas `Series`

Ikke alle attributtene og metodene har (meningsfullt) innhold før man har lastet inn formler.

`Formula`-klassen konverterer alle variabelnavn til små bokstaver. Det vil si at `evaluate`-metoden kun finner kolonner med kolonnenavn med små bokstaver. Dette er hensyntatt i `PreSystem`-klassen på den måten at enhver `DataFrame` som lastes inn får kolonnenavn konvertert til små bokstaver.

Eksempel
La oss bruke variablene fra eksemplene over:

```
x = Indicator('x', 'x', ['i1', 'i2'])

y = FDeflate('y', x, ['p1', 'p2'], ['w1', 'w2'])
```

Vi har nå to variable, `x` og `y`, som peker på henholdsvis en `Indicator`- og en `FDeflate`-formel. La oss videre opprette et forsystem:

```
forsystem = PreSystem('Test av PreSystem')
```

Vi kan nå laste inn formlene i forsystemet:

```
forsystem.add_formula(x)

forsystem.add_formula(y)
```

Merk at dersom vi hadde forsøkt å gjøre dette i motsatt rekkefølge ville vi fått feilmeldingen «`Register all dependencies: x`». Dette for å sikre at alle avhengigheter inngår i forsystemet.

Vi kan nå bruke metodene `formula(name)`, som returnerer formelen med navn `name`. Hvis formelen ikke finnes returneres `None`.

Vi kan også laste inn DataFrame's med data:

```
forsystem.annuals_df = <some DataFrame with PeriodIndex>

…
```

Vi kan nå evaluere hele forsystemet ved å bruke se på `evaluate`-attributten, eller en enkelt formel ved navn `name` ved å bruke `evaluate_formula(name)`-metoden.

Se https://github.com/statisticsnorway/ssb-pre-system/tree/main/examples/pre-system-demo.ipynb for eksempler i en "self contained" notebook.

3. Upsampling og overlay

Pakken `pre_system` inneholder også tre funksjoner, `convert`, `convert_step` og `overlay`. De to første konverterer en `DataFrame` eller `Series` til høyere frekvens (såkalt *upsampling*, for eksempel fra kvartal eller år til måned), mens `overlay` returnerer en `DataFrame` eller `Series` der eventuelle hull (`NaN`'s) lengts til venstre fylles med tallverdier lengre til høyre (etter regelen «venstre først»). `convert` lager en «glatt» serie, mens `convert_step` blir en «step»-funksjon.

Eksempel
La oss si vi har en `DataFrame` `df` med kvartalsvis eller årlig frekvens; vi kan konvertere denne til månedlig frekvens og lagre den i `df_m` med koden

```
df_m = convert(df, 'm')
```

Dersom `df` har årlig frekvens kan vi også konvertere til kvartalsfrekvens med argumentet `'q'`. `convert_step` kalles med nøyaktig samme syntaks. Dersom man sender en `Series` til disse funksjonene i stedet for en `DataFrame` er resultatet en `Series`.

Dersom vi har et antall `DataFrame`'s (eller `Series`') med huller (`NaN`'s), `df0`, `df1`, `df2`, kan vi fylle hullene med koden

```
df = overlay(df0, df1, df2)
```

Funksjonen virker som følger:

	df = overlay(df0, df1, df2)	df0	df1	df2
2020-01	1	1	2	3
2020-02	2	NaN	2	3
2020-03	3	NaN	NaN	3
2020-03	NaN	NaN	NaN	NaN

3.1. Mer om convert-funksjonen

La oss si vi har en kvartalsserie (resultatet generaliserer til andre frekvenser) $X = \{X_3, X_6, \dots, X_T\}$ (notasjonen må ikke forveksles med formelnotasjonen over). Her indikerer indeksen måned, slik at kvartalstallene altså indekseres med den siste måneden i kvartalet. Vi ønsker å finne en månedsserie $x = \{x_t\}_{t=1}^{T}$ som summerer seg til kvartalsserien, altså at $x_1 + x_2 + x_3 = X_3, x_4 + x_5 + x_6 = X_6, \dots, x_{T-2} + x_{T-1} + x_T = X_T$. I tillegg skal månedsserien være «glatt» i én eller annen forstand.

En mulig måte å måle om månedsserien er glatt, er å formulere en målfunksjon (tapsfunksjon) som «straffer» serier som er «ruglete». Én potensiell målfunksjon er

$$f(x_1, x_2, \dots, x_T) = \frac{1}{2}(x_2 - x_1)^2 + \frac{1}{2}(x_3 - x_2)^2 + \cdots + \frac{1}{2}(x_T - x_{T-1})^2.$$

Denne funksjonen «straffer» endringer i månedsserien, slik at det laveste mulige tapet er assosiert med en helt flat (konstant) månedsserie. Vi ønsker å minimere f gitt at månedsserien er avstemt mot kvartalsserien. Minimeringsproblemet er

$$\min_{x_1,x_2,\dots,x_T} \frac{1}{2}(x_2 - x_1)^2 + \frac{1}{2}(x_3 - x_2)^2 + \cdots + \frac{1}{2}(x_T - x_{T-1})^2$$
$$\text{s. t.}$$
$$x_1 + x_2 + x_3 = X_3,$$
$$x_4 + x_5 + x_6 = X_6,$$
$$\vdots$$
$$x_{T-2} + x_{T-1} + x_T = X_T.$$

Lagrangefunksjonen for minimeringsproblemet er

$$\begin{aligned}\mathcal{L}(x_1, x_2, \dots, x_T, \lambda_3, \lambda_6, \dots, \lambda_T) \\ = \sum_{t=2}^{T} \frac{1}{2}(x_t - x_{t-1})^2 + \lambda_3(x_1 + x_2 + x_3 - X_3) + \lambda_6(x_4 + x_5 + x_6 - X_6) + \cdots \\ + \lambda_T(x_{T-2} + x_{T-1} + x_T - X_T).\end{aligned}$$

Førsteordensbetingelsen er gitt ved

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_1} &= 0 = -(x_2 - x_1) + \lambda_3 = x_1 - x_2 + \lambda_3, \\ \frac{\partial \mathcal{L}}{\partial x_2} &= 0 = (x_2 - x_1) - (x_3 - x_2) + \lambda_3 = -x_1 + 2x_2 - x_3 + \lambda_3, \\ \frac{\partial \mathcal{L}}{\partial x_3} &= 0 = (x_3 - x_2) - (x_4 - x_3) + \lambda_3 = -x_2 + 2x_3 - x_4 + \lambda_3, \\ \frac{\partial \mathcal{L}}{\partial x_4} &= 0 = (x_4 - x_3) - (x_5 - x_4) + \lambda_6 = -x_3 + 2x_4 - x_5 + \lambda_6, \\ &\vdots \\ \frac{\partial \mathcal{L}}{\partial x_T} &= 0 = (x_T - x_{T-1}) + \lambda_T = x_T - x_{T-1} + \lambda_T, \\ \frac{\partial \mathcal{L}}{\partial \lambda_3} &= 0 = x_1 + x_2 + x_3 - X_3, \\ \frac{\partial \mathcal{L}}{\partial \lambda_6} &= 0 = x_4 + x_5 + x_6 - X_6, \\ &\vdots \\ \frac{\partial \mathcal{L}}{\partial \lambda_T} &= 0 = x_{T-2} + x_{T-1} + x_T - X_T.\end{aligned}$$

Disse kan skrives på matriseform som

$$\begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{X} \end{pmatrix},$$

der

$$\mathbf{A}_{1,1} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & -1 & 1 \end{pmatrix},$$

$$\mathbf{A}_{1,2} = \mathbf{A}'_{2,1} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{pmatrix},$$
$$\boldsymbol{\lambda} = \begin{pmatrix} \lambda_3 \\ \lambda_6 \\ \vdots \\ \lambda_T \end{pmatrix}, \text{og } \mathbf{X} = \begin{pmatrix} X_3 \\ X_6 \\ \vdots \\ X_T \end{pmatrix},$$

og $\mathbf{0}$ er en matrise/vektor med 0'er, som uten at vi spesifiserer det nærmere har riktige dimensjoner (slik at den passer inn i matrisen/vektoren den befinner seg). Løsningen er da gitt ved

$$\begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{0} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ \mathbf{X} \end{pmatrix}.$$

Dette kan forholdsvis enkelt programmeres ved at man bygger de ulike delmatrisene, setter dem sammen, inverterer og multipliserer.

En *annen* mulig målfunksjon er

$$\tilde{f}(x_1, x_2, \dots, x_T) = \frac{1}{2}(x_3 - 2x_2 + x_1)^2 + \frac{1}{2}(x_4 - 2x_3 + x_2)^2 + \cdots + \frac{1}{2}(x_T - 2x_{T-1} + x_{T-2})^2.$$

Denne straffer kurvatur (og ikke endring, slik som den forrige målfunksjonen), slik at det laveste tapet er assosiert med en lineær månedsserie (ikke nødvendigvis en konstant). Førsteordens-betingelsene kan fortsatt skrives på matriseform som vist over, men med $\mathbf{A}_{1,1}$ byttet ut med

$$\widetilde{\mathbf{A}}_{1,1} = \begin{pmatrix} 1 & -2 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ -2 & 5 & -4 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -4 & 6 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 6 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & -4 & 5 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & -2 & 1 \end{pmatrix},$$

mens alle de andre delmatrisene er uendret. `convert`-funksjonen benytter målfunksjonen $\tilde{f}$, siden denne ikke premierer flate serier fremfor serier med helling. Metoden er nevnt i for eksempel Boot et al. (1967), og således ikke nybrottsarbeid. Utledningen er likevel tatt med for enkelhets skyld.

Referanser

Boot, J. C. G., W. Feibes and J. H. C. Lisman (1967). Further Methods of Derivation of Quarterly Figures from Annual Data. Journal of the Royal Statistical Society. Series C (Applied Statistics), 1967, Vol. 16, No. 1 (1967), pp. 65-75. URL: https://www.jstor.org/stable/2985238