Statistics Norway

Statistisk sentralbyrå

Yngve Vogt

Innføring i FAME

Forskningsavdelingen/Gruppe for administrasjon

INNFØRING I FAME

Av Yngve Vogt,
September 1994

Det har vært et overordnet mål å finne et databaseverktøy som kan håndtere tidsserier. Byrået har valgt å satse på FAME som et slikt verktøy. FAME kjøres på UNIX-arbeidsstasjoner.

Frem til idag har det vært en omstendelig prosess å lage og analysere en tidsserie. I FAME finnes det en del fikse kommandoer som kan brukes til å bearbeide og analysere tallserier og databaser. Det er også enkelt å lage grafer og rapporter i FAME.

De som skal bruke FAME fra NYE TROLL vil oppdage at det er en enkel sak for TROLL å lese FAME-databaser.

I samarbeid mellom Den Sentrale EDB-gruppa og Forskningsavdelingen er det blitt utviklet et windowsbasert databaseverktøy som organiserer store datamengder. Forutsatt at man kjenner til oppbyggingen av databasestrukturen er dette et meget greit verktøy å bruke for å finne frem til tidsserier. Dette systemet er programmert i FAME level-1. Ønsker du mer informasjon om dette referansedatabasesystemet kan du kontakte Cecilie Alnæs eller Jon Erik Lindberg.

Vanlig FAME kalles for Level-0. Hva er forskjellen mellom level-0 og level-1? Alt hva du lærer i dette kursheftet er level-0 kommandoer. Altså VANLIG FAME. Dette er altså kommandoer for å bearbeide tidsserier og databaser, lage rapporter, grafer osv. Ønsker du å analysere noen tidsserier er det meget smart å kunne enkle knep i programmeringsspråket FAME level-0.

Level-1 er IKKE et programmeringsspråk. Derimot er LEVEL-1 et prosedyrebibliotek skrevet i vanlig FAME (Level-0) som brukes til å bygge opp og organisere store mengder databaser. Level-1 er utviklet ved Eurostat i Luxembourg. Du behøver ikke lære deg level-1.

INNHOLDSFORTEGNELSE:

START AV FAME:

Du starter FAME ved å skrive

fame

For å avslutte FAME skriver du i FAME

* exit

Har du X-vision (Et Windowslignende system på UNIX-arbeidsstasjon), vil det lønne seg å skrive

fame &

når du starter. Da vil du ha to aktive vinduer samtidig; Et FAME-vindu og et UNIX-vindu. Lurer du på hva &-tegnet betyr, anbefales det å lese kursheftet til Yngve Vogt; Første møte med UNIX.

DATABASE I FAME.

Har du en mengde med tallserier som tilhører samme emne (for eksempel fiskeristatistikken), kan du legge disse tallseriene inn i en felles database.

Du åpner en database ved å skrive

* open demo

Når du avslutter FAME vil databasen automatisk lukkes, og hvis du har gjort forandringer underveis, vil den bli oppdatert.

Databaser i FAME har alle etternavnet .db
I dette eksemplet heter databasen: demo.db

GRUNNLEGGENDE KOMMANDOER

Vil du skrive ut innholdet i en tallserie, kan du bruke "display"

* display inntekt

```
INNTEKT

Jan 93      13.00
Feb 93      12.00
Mar 93      14.00
Apr 93      15.00
May 93      16.00
Jun 93      15.00
Jul 93      14.00
Aug 93      13.00
Sep 93      16.00
Oct 93      15.00
Nov 93      14.00
Dec 93      16.00
```

Vil du skrive ut FLERE tallserier samtidig, kan du skille disse tallseriene med komma:

* display inntekt, kostnad

I dette eksemplet er "display" en FAME-kommando.
"inntekt" og "kostnad" er kommando-argumenter.

Kjennetegnet på en FAME-kommando er at den øyeblikkelig produserer et eller annet resultat.

Man kan også putte inn opsjoner som ikke har noen øyeblikkelig effekt, men som influerer på hvordan FAME-kommandoer skal oppføre seg.

Ønsker du f.eks. at alle tallene bare skal skrives ut med ETT desimal, må dette legges inn som en opsjon.

* Decimal 1
* Display salg

```
INNTEKT

Jan 93      13.0
Feb 93      12.0
Mar 93      14.0
Apr 93      15.0
May 93      16.0
Jun 93      15.0
Jul 93      14.0
Aug 93      13.0
Sep 93      16.0
Oct 93      15.0
Nov 93      14.0
Dec 93      16.0
```

Man kan lage rapport av tidsserier:

* REPORT inntekt, kostnad, netto

	Jan 1993	Feb 1993	Mar 1993	Apr 1993	May 1993	Jun 1993	Jul 1993
INNTEKT	13.00	12.00	14.00	15.00	16.00	15.00	14.00
KOSTNAD	4.00	5.00	6.00	4.00	5.00	6.00	7.00
NETTO	9.00	7.00	8.00	11.00	11.00	9.00	7.00

(Av plasshensyn i kursheftet er det bare oppført data frem til juli 1993)

Ønsker du å lage en vertikal rapport istedet, kan dette styres med en opsjon:

* SHOW VERTICAL
* REPORT inntekt, kostnad, netto

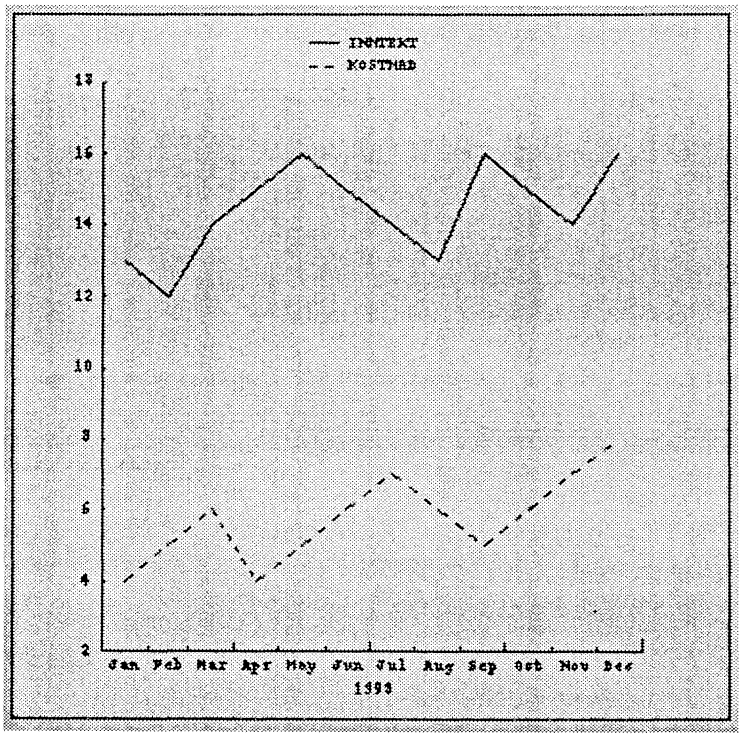
	INNTEKT	KOSTNAD	NETTO
Jan 93	13.00	4.00	9.00
Feb 93	12.00	5.00	7.00
Mar 93	14.00	6.00	8.00
Apr 93	15.00	4.00	11.00
May 93	16.00	5.00	11.00
Jun 93	15.00	6.00	9.00
Jul 93	14.00	7.00	7.00
Aug 93	13.00	6.00	7.00
Sep 93	16.00	5.00	11.00
Oct 93	15.00	6.00	9.00
Nov 93	14.00	7.00	7.00
Dec 93	16.00	8.00	8.00

Opsjonen settes tilbake igjen til horisontal ved å skrive:

* SHOW HORIZONTAL

Det er lett å lage grafer i FAME:

* graph inntekt, kostnad



FORKORTELSER AV FAME-KOMMANDOER

Det er unødvendig å skrive kommandoer og opsjoner fullt ut i FAME. Det er nok at kommandoen er entydig. Det trengs aldri mer enn 4 tegn for å identifisere en kommando.

eks:
* DISP inntekt
* DECI 1
* SHOW VER eller *SHO V
* REPO inntekt

FAME-KOMMANDOER SOM ER LENGRE ENN EN LINJE

Hvis et FAME-uttrykk går over flere linjer, må man bruke to &-tegn for å fortsette på neste linje.

eks:
```
* PLOT #1 DRAW BAR,   &&
          COLOR RED
```

Hva denne FAME-opsjonen gjør kan du lese om i kapitlet om grafer.

FLERE FAME-KOMMANDOER PÅ SAMME LINJE

Skal du skrive flere FAME-uttrykk på samme linje må du skille dem med et komma (;)

eks:
```
* PLOT #1 DRAW BAR, COLOR RED ; GRAPH inntekt, kostnad
```

JOKERNOTASJON

Jokernotasjonen i FAME er ikke like godt utviklet som i UNIX.
Men du kan jobbe med MANGE serier samtidig ved å bruke tegnene
? og ^

? : Erstatter en hvilken som helst tekstlengde
^ : Erstatter et hvilket som helst tegn.

Display hjort?
(Skriver ut alle seriene som starter med navnet hjort., f.eks:
hjort.norge, hjort.sverige, hjort.danmark)

Display ?kvinner?
(Skriver ut alle seriene som inneholder tekststrengen "kvinner". F.eks:
finnmark.kvinner.unge, finnmark.kvinner.gamle osv)

Display S^L
(Skriver ut alle serier på tre tegn og hvor 1. og 3. tegn er gitt. F.eks:
Sol, Sel, Sal osv.)

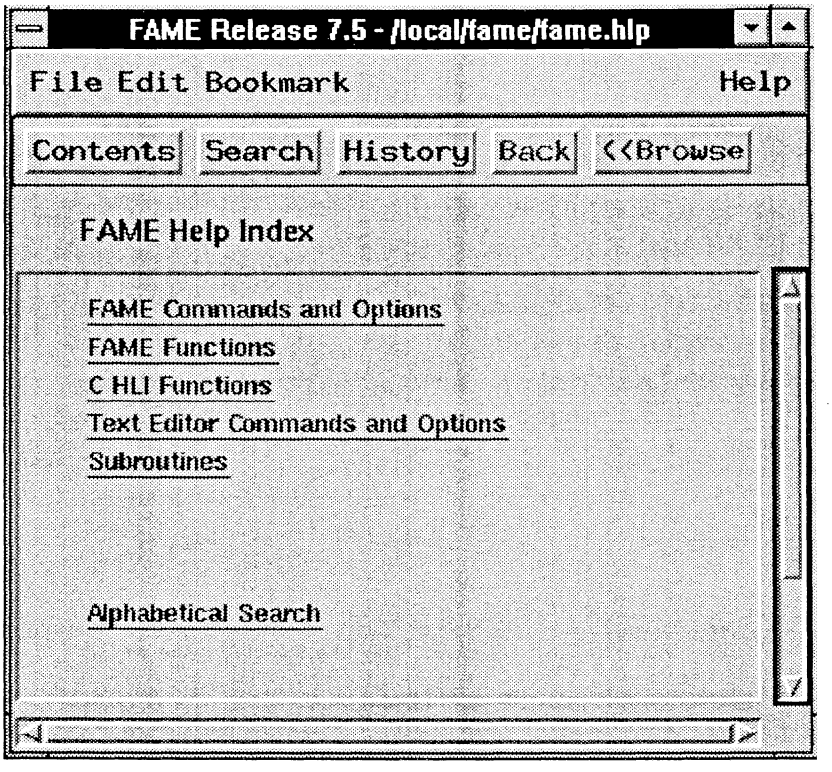
Display ^^^^mark?
(Skriver ut alle serier hvor de FIRE første tegn er ukjent og som har en vilkårlig mengde med tegn etter "mark: Eks: Finnmark.kvinner, Telemark.kvinner.
Noe som IKKE godtas er: Hedmark.kvinner)

For den avanserte:

For enkelte er denne jokernotasjonen svak. Ønsker man en jokernotasjon tilsvarende den man finner i UNIX, er det teknisk mulig å lage en kombinasjon av FAME-prosedyrer og UNIX-script som fikser dette.

HJELPEFUNKSJONER

Har du problemer kan du skrive
* HELP



Dette er et HYPERCARD system hvor man kan bla og klikke seg til den hjelpen man trenger. Finner du ikke den hjelpen du trenger, kan du kontakte din lokale FAME-kontakt. Det er også mulig å kontakte FAMEs "krisesenter" i LONDON. EMAIL-adressen dit er : hotline@fame.com

Men før du går løs på FAME i London vil vi anbefale deg å kaste deg over en av de lokale FAME-programmererne i SSB først.

FAME-MANUALER

FAME-manualer bør du ha. Ihvertfall de viktigste. Kontakt
Hans Kristian Torvbråthen og spør etter:
1) Reference-manual (uunværlig!)
2) USERs guide to FAME (bra for nybegynnere)
3) USERs guide to Graphics (egen manual bare for grafikk-delen!)
4) USERs guide to Reports (egen manual bare for rapporter)
5) Tips and Techniques (Et firesiders nyhetshefte utgitt av FAME. Inneholder smarte triks. Utgis noen ganger om året.

MER OM FAME

DATAOBJEKTER

I FAME kan du bruke TRE typer dataobjekter.

SERIES: Kan inneholde en MENGDE verdier.

SCALAR: Kan inneholde EN verdi.

FORMULA: Er et uttrykk laget av SERIER eller SKALARER.

Et DATAobjektnavn
- kan bestå av opptil 64 tegn
- De norske bokstavene Æ,Ø og Å kan ikke brukes.
- Må begynne med $, % eller en bokstav.
- Kan inneholde alle bokstaver, tall, $,%,#,@,_ og .
- Må IKKE være likt et FAMEreservert ord slik som DISPLAY, DATE...

MULIG NAVNEKONVENSJON

En mulig navnekonvensjon er:

Bruk _ (strek) for å skille navn som består av to eller flere ord:
Skatte_rate
Netto_inntekt
Konsum_pris_indeks

Bruk . (punktum) for å skille mellom dimensjoner:
Finnmark.kvinner.20-30
Volum.ujust.sn623
Danmark.hjortebestand

Det er utarbeidet en egen navnekonvensjon i LEVEl-1. Dette kan man få vite mer om ved å kontakte Cecilie Alnæs eller Jon Erik Lindberg.

DATA-TYPER:

En SERIE eller en SCALAR kan inneholde følgende DATATYPER:

NUMERIC	-	169.04 (Lagres i 16-bits format) [Korte tall]
PRECISION	-	65536256.12 (Lagres i 32-bits format) [Lange tall]
STRING	-	" Konsumprisindeksen " [Tekststreng]
BOOLEAN	-	TRUE [Boolske verdier kan enten være sann (TRUE) eller ikke sann (FALSE)]
DATE	-	19JUL61 [Datoformat kan f.eks. representeres i dagsformat, månedsformat, årsformat osv.]

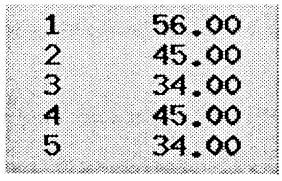
INDEKSERING

En SERIES eller SCALAR kan indekseres på TO forskjellige måter:
DATE eller CASE.

DATE: Dataene er indeksert etter et datoformat. Eks: Månedlig frekvens.

```
Apr 93     15.00
May 93     16.00
Jun 93     15.00
Jul 93     14.00
Aug 93     13.00
```

CASE: Dataene blir IKKE indeksert etter et dato-format. En CASE-serie kan oppfattes som en ARRAY.



1	56.00
2	45.00
3	34.00
4	45.00
5	34.00

HVORDAN MAN LAGER EN SCALAR

* SCALAR navn : type

type kan enten være NUMERIC, PRECISION, STRING, BOOLEAN eller DATE

Eksempler:

* SCALAR skatte_rate : Numeric

* SCALAR skatte_rate

* SCALAR bedrift : string = "Statistisk Sentralbyrå"

DEFAULT for type er NUMERIC.

HVORDAN MAN LAGER EN SERIE

SERIES navn : type INDEXED BY index

index kan enten være DATE eller CASE

type kan enten være NUMERIC, PRECISION, STRING, BOOLEAN eller DATE

DEFAULT-verdier er
type: numeric og index: DATE

Eksempler:

* SERIES inntekt : numeric INDEXED BY DATE
Pga. DEFAULT-verdiene er dette det samme som
* SERIES inntekt

* SERIES fylkesnavn : string INDEXED BY CASE

Før man kan lage en serie må man ha kjennskap til FREQ og DATE / CASE-opsjonene som beskrives på de neste sidene.

FORMLER:

Du kan lage formler i FAME ved å lage kombinasjoner / beregninger av forskjellige serier:

FORMULA netto = inntekt - kostnad

Når man lager en formel behøver ikke "inntekt" og "kostnad" bli definert FØR "netto".

Skal man forandre på en EKSISTERENDE formel kan man bruke MODIFY:
MODIFY skatt = skatte_rate * (inntekt - kostnad)

Det finnes blant annet følgende matematiske muligheter i FAME som kan brukes i en formel:

+	addisjon
-	subtraksjon
*	multiplikasjon
/	divisjon
**	Opphøyd i
DIV	Divisjon hvor resten etter komma blir glemt
MOD	Divisjon hvor man bare beholder resten etter komma

FREKVENS-OPSJONEN:

Før man lager en serie (i date-format) er det viktig å fortelle hvilken frekvens man ønsker.

Eksempler på frekvenser:

FREQ DAILY
FREQ D

FREQ BUSINESS
FREQ B

FREQ WEEKLY(MONDAY)
FREQ W(M)

FREQUENCY MONTHLY
FREQ M

FREQ QUARTERLY
FREQ Q

FREQ ANNUAL
FREQ A

DATE OPSJONEN

Når man skal lage en tidsserie må man også fortelle i hvilket tidsområde man ønsker å legge inn data.

Skal du se på deler av innholdet i en tidsserie, må man også bruke DATE-opsjonen.

SYNTAKS:

* DATE område

EKSEMPLER:

* DATE jan94 to dec94
* DATE 94:1 to 94:12
* DATE jan92 to dec94 step 2
* DATE 1986 to 1976 step -1
* DATE 19jul61,13aug67 to 23sep67 step 2
* DATE jan94 to *
* DATE * to *

Det er viktig at det er samsvar mellom FREQ- og DATE-opsjonen.
Har man valgt FREQ QUARTERLY, kan man f.eks IKKE velge
* DATE jan94 to dec94

Har man valgt FREQ QUARTERLY betyr
* DATE 94:1 to 94:4
de fire kvartalene i 1994.

Har man valgt FREQ MONTHLY betyr
* DATE 94:1 to 94:4
de fire første månedene i 1994.

SE PÅ DATE- og FREQ-OPSJONEN

Lurer du på hvilken frekvens eller tidsområde-opsjon som er satt, kan du skrive:

* TYPE @DATE
* TYPE @FREQ
Har man hverken satt frekvens eller tidsområde er følgende DEFAULT:
* FREQ ANNUAL
* DATE * to *

HVORDAN LEGGE INN DATA I EN SERIE

* update serienavn

Tast inn alle dataene og avslutt med SAVE

Eksempel:

* DATE jan94 to dec94
* UPDATE inntekt
94:1> 24,34
94:3> 25 for 3, 31,32
94:8> 40
94:9> 26 for 4

Dette blir:

```
INNTEKT

Jan 94      24.00
Feb 94      34.00
Mar 94      25.00
Apr 94      25.00
May 94      25.00
Jun 94      31.00
Jul 94      32.00
Aug 94      40.00
Sep 94      26.00
Oct 94      26.00
Nov 94      26.00
Dec 94      26.00
```

SUMMA SUMMARUM AV HVORDAN MAN LAGER EN SERIE

Før man lager en tidsserie må man spesifisere frekvensen:

```
* FREQUENCY MONTHLY
* SERIES inntekt : numeric INDEXED BY date
```

Før man legger inn data i serien må man spesifisere tidsområdet:

```
* date jan94 to dec94
* update inntekt
```

Du kan også legge inn alle dataene momentant:

```
Frequency monthly
Date 1994
SERIES inntekt = 2,4,5,6,7,5,4,6,7,8,3,2
```

Eksempel på hvordan man lager en CASE-serie:

```
SERIES fylker : string INDEXED BY case
CASE 1 to 19
Update fylker
```

Man kan også legge inn alle fylkene momentant. Det er viktig å huske på at tekststrenger skal skrives i apostrof:

```
CASE 1 to 19
SERIES fylker: string INDEXED BY case = "Østfold", "Akershus", "Oslo", osv,
```

CASE-opsjonen fungerer på samme prinsipp som DATE-opsjonen:

```
CASE 1 to 19
CASE 19 to 1 step -1
CASE 1 to 50 step 5
```

NEW kommandoen

Med NEW-kommandoen kan du fysisk lage en ny serie som er beregnet utifra eksisterende serier:

* NEW netto = inntekt - kostnad

FORSKJELLEN PÅ NEW OG FORMULA

Hva er forskjellen på NEW og FORMULA?

NEW netto = inntekt - kostnad
Med NEW blir "netto" beregnet der og da. Skulle "inntekt" eller "kostnad" forandres underveis, vil dette IKKE ha noen effekt på "netto".

FORMULA netto = inntekt - kostnad
FORMULA lager ingen ny serie som heter "netto". Først når man senere ønsker å bruke "netto" i et uttrykk vil FAME gjøre de nødvendige beregninger.

Det er ikke nødvendig at "inntekt" og " kostnad eksisterer FØR man bruker netto:

FORMULA netto = inntekt - kostnad
series inntekt = 1,2,3,4, osv.
series kostnad = 0,1,2,3 osv.
DISP netto

Fordelen med FORMULA er at man oppnår et fleksibelt system, og at dette tar LITEN plass på databasen.

Fordelen med NEW er at hvis beregningene tar LANG tid, kan man gjøre disse beregningene en gang for alle.

FJERNING AV DATAOBJEKT

Skal du fjerne et dataobjekt kan du bruke DELETE-kommandoen:

* DELETE inntekt

KOPIERING AV DATAOBJEKT

Syntaksen for å kopiere er:

COPY dataobjekt(er) AS nytt_navn TO hvilken_database

Eksempler:

COPY inntekt AS utgift

COPY inntekt AS utgift TO demo
Denne kommandoen kopierer "inntekt" til databasen "demo" og kaller så dette dataobjektet for "utgift".

COPY ? to ny_database
Kopierer alle dataobjektene og legger dem i "ny_databse".

FORANDRING AV ET DATAOBJEKTNAVN

Skal du endre navnet på et dataobjekt kan du bruke RENAME-kommandoen:

* RENAME salg AS underskudd

KONSTANTER / UTTRYKK

Man kan bruke SET til å lage konstanter / uttrykk:

SET skatt = skatte_rate * inntekt

SET skatt[jan94] = 34

SET skatte_rate = 34.50

Skal du tilordne en del av en serie lik en annen serie kan du også bruke SET:
Eksempel:
Date 1980 to 1991
Series gml = 1,2,3,4,5,6,7,8,9,10,11,12
Date 1988 to 1991
Series ny = 0,0,0,0
SET gml = ny
Du vil da se at det bare er tidsenhetene innenfor DATE-området i serien "gml" som forandres.

Prøver du å sette:
gml = ny
vil du observere at "gml" blir identisk med "ny". Du mister derfor all den gamle informasjonen i serien "gml". Prøv og se forskjellen. Dette er meget viktig.

Beregning i en serie, hvor ett ledd er avhengig av et annet:

Har du f.eks. at
* x[jan94] = 112

og ønsker at alle månedene i dette året skal økes med 2 prosent i forhold til forrige måned, kan du bruke SET-kommandoen til dette:

* DATE feb93 TO dec93
* SET x = x[T-1]*1.02

FORSKJELLEN PÅ TYPE OG DISPLAY

Du har helt sikkert sett at det hittil i heftet både er brukt TYPE og DISPLAY. TYPE brukes for SKALARER, mens DISPLAY brukes for SERIER.

BESKRIVELSE AV SERIE

Til hver serie kan man legge inn en beskrivelse:

DESCRIPTION(salg) = "Informasjon om salget av statistisk sentralbyrå"

Hvis du vil se på hva slags beskrivelse du har lagt i en serie kan du skrive:
* TYPE DESCRIPTION(salg)

INFORMASJON OM EN SERIE

Lurer du på hva slags type dataobjekt du har kan du bruke WHATS-kommandoen:
* WHATS inntekt

```
                        INNTEKT

          En byråansatts skrøpelige inntekt

Class:  SERIES                          DB name:  DEMO
Type:   NUMERIC                         Created:  13-Apr-94
Index:  DATE:MONTHLY                    Updated:  13-Apr-94

First Value at: Jan 93                  Observed: SUMMED
Last Value at:  Dec 93                  Basis:    DAILY
```

INNHOLD I DATABASE

Med CATALOG-kommandoen kan du få en liste over ALLE dataobjektene du har i en database:

* CATALOG demo

```
                        DEMO

              /ssb/neumann/h1/ynv/demo.db

Created:  13-Apr-94                     Updated:  13-Apr-94

                      Contents

HJORT.DANMARK -- SERIES (NUMERIC by DATE:MONTHLY)

HJORT.NORGE -- SERIES (NUMERIC by DATE:MONTHLY)

HJORT.SVERIGE -- SERIES (NUMERIC by DATE:MONTHLY)

INNTEKT -- SERIES (NUMERIC by DATE:MONTHLY)
 En byråansatts skrøpelige inntekt

JERV.NORGE -- SERIES (NUMERIC by DATE:MONTHLY)

JERV.SVERIGE -- SERIES (NUMERIC by DATE:MONTHLY)

KOSTNAD -- SERIES (NUMERIC by DATE:MONTHLY)

NETTO -- SERIES (NUMERIC by DATE:MONTHLY)
```

MER OM OPSJONER

Det skjer ingenting momentant når man setter på en opsjon. Poenget med en opsjon er at den har effekt på visse andre KOMMANDOER.

Det er verdt å legge merke til at alle opsjoner har en DEFAULT-verdi.
Lurer du på hva en opsjon er lik, kan du skrive:
* type @opsjonen
eksempel:
* type @freq

Man kan sette opsjoner på 4- nivåer:
- Globale opsjoner
- Block-opsjoner
- Kommando-opsjoner
- Argument-opsjoner

GLOBALE OPSJONER:

opsjoner som settes globalt gjelder på alle nivåer frem til de forandres igjen.

eksempler:

* Date jan94 to dec94
* Title "En byråkrats problematiske omgang med skatt"
* Caption "Undertegnet av Yngve Vogt"
* Report inntekt, kostnad, netto

forklaring: Title kommer øverst i rapporten
Caption kommer som en fotnote i rapporten

Hver gang det lages en rapport heretter, vil den automatisk få den overskriften og fotnoten som ble valgt her.

BLOCK-OPSJONER

Man kan lage blokker i FAME, hvor de opsjoner man setter i en blokk bare gjelder lokalt. Dette er et triks som er spesielt lurt å bruke når man skal programmere prosedyrer og funksjoner.

Eksempel:

```
* date jan94 to dec94
* decimal 2
* block
*   decimal 5
*   date mar94 to sep94
*   display inntekt        --   Inntekt vil bare bli skrevet ut for sommerhalvåret
                           --   og med 5 desimaler
* end block
* display inntekt         --   Inntekt vil bli skrevet ut for hele 1994 og med
                           --   to desimaler.
```

Man kan bare lage ett nivå med blokker.

KOMMANDO-SPESIFISERTE OPSJONER

Kommando-spesifiserte opsjoner gjelder lokalt i en kommando. De globale opsjonene som er satt vil fortsatt være gyldige etter at kommandoen din er utført.

```
* report < show vertical ; decimal 1 > inntekt, kostnad, netto

* display < date jan94 to jun94 > inntekt, kostnad, netto
-- Her vil alle seriene kun bli skrevet ut for første halvdel 1994.
```

ARGUMENT-SPESIFISERTE OPSJONER

Argument-spesifiserte opsjoner gjelder bare for et bestemt argument i en kommando. Slike opsjoner må stå ETTER hvert enkelt argument.

```
* date jan94 to dec94
* display inntekt < date mar 94 to sep 94 > , kostnad, netto
-- Her vil "inntekt" bli skrevet ut for sommerhalvåret, mens "kostnad" og "netto" -
--  blir skrevet ut for hele 1994.
```

```
* report < decimal 2 > inntekt < decimal 1>, kostnad,netto
-- Til de argumentene hvor det ikke er gitt en spesifikk opsjon, vil det bli laget
-- rapport med to desimaler ( jfr: kostnad og netto). "Inntekt" har fått en spesifikk
-- opsjon og vil få 1 desimal.
```

```
* graph inntekt < plot color green > , &&
        kostnad < plot color red >
-- grafen av "inntekt" blir grønn mens grafen av "kostnad" blir rød.
```

DATA-BASER

I en database kan man lagre en mengde serier, skalarer og formler.
Man kan ha flere databaser åpne samtidig. Har du ikke åpnet noen databaser vil alle de dataobjektene du lager bli lagret i en database som kalles for WORK. DenneWORK-databasen er temporær (midlertidig) og blir slettet når du går ut av FAME.

Lage en database:
CREATE min_database

En database får automatisk etternavnet .db

Skal man åpne en allerede eksisterende database skriver man:
* open demo

Skal man åpne en database som ligger på en annen katalog enn fra der man startet FAME, må man spesifisere katalogen:
* OPEN "/local/fame/level1/DEMO/DB/demo" AS egen_database
Når man kaller på denne databasen henviser man til "egen_database"

OPPDATERING AV DATABASE

Hvis maskinen går ned risikerer du å miste alle de oppdateringene du har gjort. Så ønsker du å lagre disse uten å avslutte databasen, kan du bruke POST-kommandoen.

* post demo1,demo2
* post all

LUKKING AV DATABASE

Når du lukker en database vil den samtidig la seg oppdatere med de forandringene du har gjort.

* close demo
* close all

Når du avslutter FAME med "exit" vil du samtidig lukke og oppdatere alle databasene som var åpne.

ANGRING

Angrer du på de oppdateringer du har gjort siden siste CLOSE eller POST-kommando, kan du bruke RESTORE-kommandoen.
* RESTORE demo

FLERE BRUKERE AV SAMME DATABASE

Når du åpner en database har du følgende tre muligheter:
1) Du gir deg selv bare leserettigheter
2) Du gir deg rettigheter til å oppdatere databasen, samtidig som du sperrer databasen for alle andre mens du er inne i den.
3) Du gir deg rettigheter til å oppdatere databasen, samtidig som du lar andre brukere få anledning til å se på den.

En foreløbig DEFAULT er punkt 2, dvs at du får alle rettigheter til å lese og oppdatere dataobjektene som ligger der, samtidig som INGEN andre får anledning til å bruke databasen mens du har den åpen. Dette kan være en grei ordning når du åpner en database på ditt eget filområde, mens det er høyst upraktisk hvis databasen ligger på et fellesområde.

Til OPEN-kommandoen bør du kjenne til følgende opsjoner:

Har du en egen database som du ikke ønsker skal forandres på ved en feiltagelse, kan du bruke READ-opsjonen. Andre brukere kan da se på databasen samtidig.
* OPEN < ACCESS READ > min_database

Gir du SHARED-opsjonen får du tillatelse til å skrive på databasen, samtidig som andre brukere får LESErettigheter.
* OPEN < ACCESS SHARED > felles_database
Anta at en annen bruker samtidig er inne i denne databasen, mens du oppdaterer den. Brukeren vil da ikke få tilgang til oppdateringene dine før du har lukket databasen, og han har lukket og gjenåpnet den.

Vil du både lese og oppdatere en database kan du skrive:
* OPEN < ACCESS UPDATE> min_database

UPDATE er DEFAULT, så skriver du
* open min_database
vil du få alle rettigheter til denne databasen.

Det vurderes å legge inn READ som DEFAULT opsjon. Det farlige med dette er at hvis du da åpner en database uten å legge til en annen ACCESS-opsjon, vil alle de dataobjektene du lager havne på WORK-området, hvilket betyr at alt det arbeidet du har gjort forsvinner når du går ut av FAME, såfremt du ikke er påpasselig.

SØK I DATABASER

Har du åpnet en hel haug med databaser og ønsker å finne et dataobjekt, vil FAME lete igjennom alle databasene i den rekkefølge de er åpnet.

Mener du at det ville være mer tidsmessig optimalt at FAME leter gjennom databasene i en annen rekkefølge, kan du bruke SEARCH-opsjonen:
* SEARCH min_database, felles_database
Dette er spesielt lurt i de tilfeller hvor du oftest trenger å hente frem dataobjekter i min_database og hvor felles_database er umådelig stor.

For å se hvilken rekkefølge du har valgt skriver du:
* TYPE @SEARCH

LAGRING I DATABASER

Har man flere databaser åpne, vil de nye dataobjektene man genererer, automatisk bli lagret i den FØRSTE databasen du åpnet.

Ønsker du at de nye dataobjektene skal lagres på en bestemt database skriver du:
* STORE viktig_database

Angrer du på dette, kan du skrive:
* STORE AUTO
Du vil da heretter lagre alle dataobjektene i den første databasen du åpnet.

Vil du lagre en ny serie i en bestemt database kan du skrive:
* SERIES < STORE viktig_database > profitt : numeric indexed by date
ELLER
* SERIES < STORE viktig_database > profitt

REFERERING TIL DATABASE

Vil du referere til en bestemt database, kan du skrive databasenavnet rett foran dataobjektet:
* DISPLAY min_database'inntekt
* SERIES min_database'kuer.sverige
Refererer du ikke til en bestemt database og du har flere dataobjekter med samme navn, vil FAME finne frem til det første dataobjektet den finner i henhold til det søkekriteriet du har gitt.

WORK-DATABASE

I FAME finnes det en temporær database som heter WORK. Denne slettes automatisk når du går ut av FAME. Har du ikke åpnet en database med skrivetillatelse vil alle de dataobjektene du genererer automatisk bli lagret i WORK. Trenger du å lage mellomregninger, er det smart å legge dem i WORK.

Følgende TRE kommandoer er LIKE, og alle lagrer følgende uttrykk i WORK:
* / netto = inntekt - kostnad
* < STORE WORK > netto = inntekt - kostnad
* WORK'netto = inntekt - kostnad

FJERNING AV DATABASE

Man kan fjerne en database direkte fra FAME ved å bruke UNSAVE:
* UNSAVE min_database
* UNSAVE "/ssb/lynx/h1/ynv/prosjekt/demo"
Dette kan selvfølgelig også gjøres direkte fra UNIX med rm-kommandoen.

FORANDRING AV DATABASENAVN

Man kan forandre navnet på en database ved å skrive:
* REFILE gammeltnavn AS nyttnavn
Dette kan også gjøres direkte fra UNIX med mv-kommandoen.

LAGS

LAGS henviser til faste og relaterte indekser i en tidsserie.

* Konsumprisindeks = (konsum / konsum[jan70])*100
Her vil konsumprisindeksen bli regnet ut iforhold til 1970-verdien.

* REPORT inntekt, inntekt[T-12],inntekt[T+12]
Dette blir en rapport hvor du får med verdiene for gjeldende år, året foran og året etter. Dette forutsatt at inntekt er en månedsserie, slik at 12 står for 12 måneder.

* prosent = ((inntekt - inntekt[T - 1])/inntekt[T-1])*100
Her beregnes prosentvis økning for hver tidsenhet for serien "inntekt".

INPUT-FILER I FAME

Skal du gjøre en hel haug med kommandoer er det smart å legge dem på en input.fil. Da kan du forandre på enkelte kommandoer/opsjoner uten å måtte trykke inn alle en gang til.

Input-filer er også meget velegnet hvis du ofte skal gjøre bestemte arbeidsoppgaver, slik som å åpne en mengde med databaser eller å initialisere en hel haug med opsjoner.
En INPUT-fil skriver du i en teksteditor (f.eks. emacs) i UNIX.

En INPUT-fil må ha etternavnet .inp

Eksempel på en inputfil som vi har kalt for rapport.inp:

```
BLOCK
-- Alt som kommer etter to streker blir tolket som en kommentar.
-- Etter en TRY-setning kan man åpne en database hvis man ikke er
-- sikker på om den allerede er åpnet. Hadde databasen allerede vært åpnet og du
-- hadde prøvd å åpne den en gang til uten å bruke TRY-kommandoen, ville
-- hele programmet ha stoppet opp.
TRY
OPEN demo
END TRY
-- I denne databasen skal det skrives rapport om seriene inntekt,
-- kostnad og netto. Det ønskes INGEN desimaler, og vi ønsker
-- bare rapport fra sommerhalvåret.
freq monthly
date mar93 to sep93
Title "Økonomiske problemer"
Decimal 0
show vertical
Report inntekt, kostnad, netto
END BLOCK
-- Hvorfor legges dette inn i en blokk? Jo, fordi de opsjonene som settes her da
-- ikke vil forandre på de globale opsjonene som er satt.
```

For å kjøre dette programmet er det nok å skrive
* input rapport

FUNKSJONER / PROSEDYRER

I FAME er det fullt mulig å lage egne prosedyrebibliotek. Et bibliotek kan inneholde så mange prosedyrer du vil.

Det er mulig å gi argumenter (= parametre) til en prosedyre.

Før du kan bruke prosedyren, må prosedyrebiblioket både kompileres og loades. Man kompilerer biblioteket for å finne syntaksfeil. Når kompileringen er vellykket kan man "LOADE" biblioteket. Du får da tilgang til alle prosedyrene i biblioteket. Disse kan så brukes på lik linje med hvilke som helst andre FAME-kommandoer.

Som en navnestandard er det vanlig skikk at alle prosedyrer starter med et $-tegn. Dette for at det senere skal bli enklere å lese programmene dine.

Eksempel på en prosedyre:

Du ønsker å lage en prosentvisberegning av serien inntekt. Dette eksemplet blir brukt som et utgangseksempel i kapitlet om prosedyrer og funksjoner.

```
PROCEDURE $prosent
BLOCK
TRY
        OPEN demo
END TRY
        freq monthly
        date mar93 to sep93
        Decimal 0
        Local prosent = ((inntekt - inntekt[T-1])/inntekt[T-1])*100
        -- Local betyr at dette dataobjektet bare eksisterer så lenge
        -- man befinner seg innenfor en blokk.
        disp prosent
END BLOCK
END PROCEDURE
```

For å kjøre denne prosedyren må man først kompilere prosedyrefilen:
* COMP bibliotek

Dernest må prosedyrefilen loades:
* LOAD bibliotek

Først da kan man kjøre prosedyren $rapport:
* $prosent

ARGUMENTER TIL EN PROSEDYRE

Det er mulig å gi argumenter til en prosedyre.

PROCEDURE $rapport
ARGUMENTER argument1,argument2,.....,argument_n

kommandoer

eks:
DISPLAY argument1, argument2,.....,argument_n
osv.

END PROCEDURE

For å kalle opp denne prosedyren skriver man:
* $rapport argument1,argument2,.....,argument_n

Forandrer du innholdet i argumentene inne i prosedyren, vil argumentene beholde disse verdiene etter at prosedyren er kjørt.

Eksempel:

Du vil gjerne modifisere prosedyren $prosent som ble laget i forrige avsnitt til å gjelde generelt for en hvilken som helst tidsserie. Slik som den er laget nå kan du bare beregne prosentvise forandringen av dataobjektet "inntekt". For å generalisere den kan man lage et argument til prosedyren:

```
PROCEDURE $prosent
Argument serienavn
BLOCK
TRY
        OPEN demo
END TRY
        freq monthly
        date mar93 to sep93
        Decimal 0
        Local prosent = (( serienavn - serienavn[T-1])/serienavn[T-1])*100
        disp prosent
END BLOCK
END PROCEDURE
```

For å beregne prosentvis endring i "inntekt" skriver du:
* $prosent inntekt

FUNKSJONER

En funksjon skiller seg fra en prosedyre ved at funksjonen returnerer et dataobjekt.

Anta at du er fornøyd med beregningsrutinene i forrige prosedyre, men at du ønsker å være fri med HVA du skal gjøre med resultatet. Av og til vil du kanskje bruke DISPLAY, andre ganger lage en rapport eller graf. Denne fleksibiliteten kan man ikke få til i prosedyreformen.

Løsningen er da å bruke FUNCTION. Der gjøres utelukkende beregningene. En function returnerer så resultatet som deretter brukes videre i en annen kommando.

Funksjoner kan legges i akkurat de samme biblioteksfilene som prosedyrer. Og de kan gjerne legges om hverandre. Eneste betingelse er at de ikke får samme navn.

```
FUNCTION $prosent
Argument serienavn
BLOCK
TRY
        OPEN demo
END TRY
        freq monthly
        date mar93 to sep93
        Decimal 0
        Local prosent = (( serienavn - serienavn[T-1])/serienavn[T-1])*100
        RETURN prosent
END BLOCK
END FUNCTION

* Disp $prosent(inntekt)
* Report $prosent(inntekt), $prosent(kostnad), $prosent(netto)
* Graph  $prosent(inntekt), $prosent(kostnad)
```

Eksempel 2:

Anta at du har lyst til å lage en funksjon som beregner summen av alle tallene fra januar til en gitt måned.

```
FUNCTION $sum_fra_januar_til_denne_maaned
ARGUMENT serienavn, mnd
BLOCK
 date mnd-period(mnd)+1 to mnd
 LOCAL x=sum(serienavn)
 RETURN x
END BLOCK
END FUNCTION


* TYPE  $sum_fra_januar_til_denne_maaned(inntekt,jul93)
```

DIRIGERING AV OUTPUT I FAME

OBS!
Dette avsnittet er HELT avhengig av om du har X-vision installert.

Som du ser har du følgende vinduer i FAME:
- DIALOG (Her taster du inn kommandoene dine)
- OUTPUT (Her ser du resultatet av hva du gjør)
- INFO (Her kommer feilmeldingene)
- PICTURE (Her kommer grafene)

Hvis du vil omdirigere alle resultatene fra OUTPUTvinduet til en fil kan du skrive:
* OUTPUT resultatfil.txt
(Skriver du ikke etternavnet txt, vil filen automatisk få dette.)

Anta at denne filen allerede finnes på forhånd. Da må du gi tillatelse til å overskrive denne filen:
* OUTPUT < ACCESS OVERWRITE > resultatfil.txt

Når du vil dirigere resultatene tilbake til skjermen skriver du:
* OUTPUT TERMINAL

Ønsker du at feilmeldingene skal dirigeres til OUTPUT-vinduet eller at du ikke er interessert i "ADVARSLER" = (Warnings) er dette fullt mulig. Det finnes også muligheter i FAME for å ta vare på alle de kommandoene du har gjort.

OMDIRIGERINGER

* CHANNEL AUDIT NONE
(AUDIT betyr at alle de kommandoene du gjør i FAME blir lagret på en fil. DEFAULT er NONE. Dvs at de IKKE blir lagret.)

Ønsker du å lagre kommandoene må du først åpne en tekstfil, og så omdirigere kommandoene dit.
* OPEN < ACCESS OVER; KIND TEXT > husk.alt
* CHANNEL AUDIT husk.alt

* CHANNEL ECHO NONE
(EDIT betyr at alle kommandoene som blir lest inn fra INPUT-fil blir lagret. DEFAULT er NONE, dvs at de IKKE lagres.)
Ønsker du å lagre dette må du:
* OPEN < ACCESS OVER; KIND TEXT > husk.input
* CHANNEL ECHO husk.input

* CHANNEL REPORTS OUTPUT
(All rapportgenerering dirigeres til OUTPUT. Vil du ha det på en fil må du:
* OPEN < ACCESS OVER; KIND TEXT > husk.report
* CHANNEL REPORTS husk.report

* CHANNEL RESULTS OUTPUT
(Betyr at resultatet av alle kommandoene unntatt REPORT blir sendt til RESULTS. I dette tilfelle er OUTPUT-vinduet default.

* CHANNEL ERRORS INFO
(Betyr at alle feilmeldingene blir sendt til INFO-vinduet. Ønsker du å ha dem til OUTPUT-vinduet kan du skrive: CHANNEL ERRORS OUTPUT)

* CHANNEL WARNINGS INFO
(Betyr at alle advarslene (= WARNINGS) blir sendt til INFO-vinduet. Ønsker du ingen advarsler kan du skrive: CHANNEL WARNINGS NONE)

KONSTANTER

Skal du skrive FAME-programmer er det greit å bruke konstanter for å få programmene mer leselige.

```
* SET fradato = JAN93
* SET tildato  = DEC93
```

Så kan man senere sette tidsrammen slik:

```
* DATE fradato TO tildato
```

Man kan koble sammen tekststrenger i FAME med +-tegnet:

```
* type " Bedrift " + " Statistisk Sentralbyrå.

* SET bedrift = "Statistisk Sentralbyrå"
* TITLE #1 bedrift + TODAY
```

(I en rapport kan man ha 6 titler. #1 betyr tittel 1)

INNLESING AV VERDI FRA TERMINAL

Fra et FAME-program kan man underveis stoppe programmet og be brukeren taste inn en verdi.

```
* SCALAR bedrift : STRING
* ACCEPT "Hvor jobber du?, Tast inn svaret her: " : bedrift
* CAPTION "Dette er en report for " + bedrift
* REPORT inntekt
```

BOOLSKE VERDIER

I IF-tester som du kan lese om i neste avsnitt, trenger man kjennskap til BOOLSKE uttrykk.

Man kan bruke de logiske uttrykk: AND, OR, NOT
Og relasjonsuttrykkene:

EQ	(Er lik)
NE	(Ikke lik)
LT	(Mindre enn)
LE	(Mindre enn eller lik)
GT	(Større enn)
GE	(Større enn eller lik)

Eksempler:

* Date jan93 to jul93
* Display inntekt

```
INNTEKT

Jan 93      13.00
Feb 93      12.00
Mar 93      14.00
Apr 93      15.00
May 93      16.00
Jun 93      15.00
Jul 93      14.00
```

* Display inntekt GT 12 AND inntekt LT 15

```
INNTEKT GT 12 AND INNTEKT LT 15

Jan 93       TRUE
Feb 93      FALSE
Mar 93       TRUE
Apr 93      FALSE
May 93      FALSE
Jun 93      FALSE
Jul 93       TRUE
```

IF-TESTER

Har du et utrykk som er avhengig av en enten / eller- betingelse , kan du bruke IF-kommandoen:

Syntaks:
```
IF boolsk uttrykk
   kommandoer
ELSE IF boolsk uttrykk
   kommandoer
ELSE
   kommandoer
END IF
```

Eksempel:

```
SCALAR avdeling : NUMERIC
ACCEPT "Tast inn avdelingsnummeret ditt: " : avdeling
IF avdeling GE 500 AND avdeling LT 600
    CAPTION "Forskningsavdelingen " + TODAY + NOW
ELSE
    CAPTION "Resten av SSB" + TODAY + NOW
END IF
```

IF-TEST I TILORDNING

Den generelle syntaksen for IF-test i tilordning er:

```
dataobjekt = IF boolsk uttrykk THEN kommandoer
             ELSE IF boolsk uttrykk THEN kommandoer
             ELSE kommandoer
```

Eksempel:
```
* skatt = IF inntekt LT 10000 THEN lav_skatteprosent*inntekt &&
         ELSE normal_skatteprosent * inntekt
```

LØKKER

Syntaksen for løkker er:

LOOP FOR uttrykk = fra TO til
kommandoer
END LOOP

Eksempler:

LOOP FOR maaned = jan93 to dec93
  kommandoer
END LOOP

LOOP for indeks = 1 to 12 step 2
kommandoer
END LOOP

LOOP WHILE indeks GT 0
kommandoer
END LOOP

FUNKSJONER I FAME

Det finnes en mengde funksjoner i FAME. Du finner dem alle i referansemanualen. Dette er bare en oversikt over noen meget få av dem:

* TYPE AVE(inntekt)
(Finner gjennomsnittsinntekten)

* DISPLAY LOG(inntekt)
(Finner logaritmen av inntekten)

DIFFERANSE

Ønsker du å se på differansen mellom hvert ledd i en serie kan du bruke kommandoen: DIFF

* DIFF(inntekt)

```
DIFF(INNTEKT)

Feb 93      -1.00
Mar 93       2.00
Apr 93       1.00
May 93       1.00
Jun 93      -1.00
Jul 93      -1.00
Aug 93      -1.00
Sep 93       3.00
Oct 93      -1.00
Nov 93      -1.00
Dec 93       2.00
```

* DISPLAY PCT(inntekt)

(Finner prosentforskjellen fra verdi til verdi i serien)

```
PCT(INNTEKT)

Feb 93      -7.69
Mar 93      16.67
Apr 93       7.14
May 93       6.67
Jun 93      -6.25
Jul 93      -6.67
Aug 93      -7.14
Sep 93      23.08
Oct 93      -6.25
Nov 93      -6.67
Dec 93      14.29
```

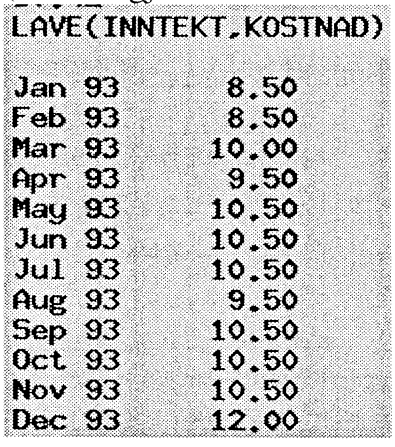
* Display MAVE(inntekt,2)

(Tar for hver verdi i serien, gjennomsnittet av denne og forrige verdi)

```
MAVE(INNTEKT,2)

Feb 93      12.50
Mar 93      13.00
Apr 93      14.50
May 93      15.50
Jun 93      15.50
Jul 93      14.50
Aug 93      13.50
Sep 93      14.50
Oct 93      15.50
Nov 93      14.50
Dec 93      15.00
```

LISTE-GJENNOMSNITT:
*LAVE(inntekt, kostnad)
(Finner gjennomsnittet for tidsenhet i "inntekt" og "kostnad")



LAVE(INNTEKT,KOSTNAD)	
Jan 93	8.50
Feb 93	8.50
Mar 93	10.00
Apr 93	9.50
May 93	10.50
Jun 93	10.50
Jul 93	10.50
Aug 93	9.50
Sep 93	10.50
Oct 93	10.50
Nov 93	10.50
Dec 93	12.00

Skal man finne gjennomsnittet av ALLE tallene i FLERE serier kan man kombinere LAVE og AVE:
* TYPE AVE(LAVE(serie1,serie2,serie3, osv))

Tilsvarende finnes det funksjoner for LSUM og SUM
(LSUM : Finner summen, måned for måned av en del serier)
(SUM: Gir summen av alle tallene i EN serie)

Prøv også LMAX, MAX, LMIN og MIN

KONVERTERING AV DATAOBJEKTNAVN TIL TEKST OG OMVENDT

NAME-funksjonen forandrer et dataobjektnavn til en streng-variabel:
* TITLE NAME(inntekt) + " i Statistisk Sentralbyrå"

ID-funksjonen forandrer en streng-variabel til et dataobjektnavn:
* SCALAR mittnavn : STRING = "Inntekt"
* DISPLAY ID(mittnavn)

SERIER MED LIKE TALL

SERIES-funksjonen gir LIKE tall til en serie:
* nyserie = SERIES(12)
* disp nyserie

```
NY

Jan 93      12.00
Feb 93      12.00
Mar 93      12.00
Apr 93      12.00
May 93      12.00
Jun 93      12.00
Jul 93      12.00
```

TILFELDIGE TALL:

UNIFORM-funksjonen lager tilfeldige tall mellom 0 og 1:
* nyserie = UNIFORM*100
* disp nyserie

```
NY

Jan 93      83.80
Feb 93      83.34
Mar 93      20.64
Apr 93      13.01
May 93      24.27
Jun 93      13.62
Jul 93      90.36
```

NAVNELISTER

I en navneliste kan man samle flere serier:
* samlet = { inntekt, kostnad, netto }
* report samlet

Da vil det bli laget rapport om alle de seriene som ble lagt inn i gruppen "samlet".

Viktig forskjell:

Bruker man ! (utropstegn) unngår man ekspansjon:
* display samlet
(skriver ut alle seriene som er i gruppen samlet)
* display !samlet
(skriver bare ut NAVNENE på de seriene som er med i gruppen samlet)

```
SAMLET

INNTEKT, KOSTNAD, NETTO
```

*whats !samlet

```
                        SAMLET

Class:  SCALAR                          DB name:  DEMO
Type:   NAMELIST                        Created:  16-Apr-94
                                        Updated:  16-Apr-94

{ INNTEKT, KOSTNAD , NETTO }
```

*delete !samlet
(fjerner bare GRUPPEN "samlet". Seriene inne i gruppen vil IKKE fjernes)

* delete samlet
(fjerner ALLE seriene i gruppen "samlet")

KOMBINASJON AV GRUPPER / NAMELIST

Har du FLERE grupper med serier, kan man kombinere gruppene, trekke ut utvalg av dem osv.

For å kombinere NAMELIST/GRUPPER finnes følgende 4 kommandoer:

UNION
INTERSECT
+
EXCEPT

For å illustrere hvordan dette virker, brukes følgende GRUPPER:

```
abc = { a,b,c )          [ Gruppen "abc" inneholder seriene a,b og c ]
cef  = { c,e,f }
ba   = { b,a }
ab   = { a,b }
cba  = {c,b,a }
aab  = { a,a,b }
```

```
* type !abc + cef          BLIR     { a,b,c,c,e,f }
* type !abc union cef      BLIR     { a,b,c,e,f }
* type !abc intersect cef  BLIR     { c }
* type !abc except cef     BLIR     { a,b }
* type !cef except abc     BLIR     {e.f }
```

TESTUTTRYKK I NAMELIST

SUBSET	(en del av en annen gruppe)
EQ	(identisk lik rekkefølgen til en annen gruppe)
NE	(omvendt av EQ)
EQL	(Det er nok at seriene i gruppen er den samme. Rekkefølgen er likegyldig)
NEL	(omvendt av EQL)

```
* type ab subset abc                    BLIR    TRUE
* type abc subset ab                    BLIR    FALSE
* type ab eq ba                         BLIR    FALSE
* type ab eql ba                        BLIR    TRUE
* type ab eql aab                       BLIR    TRUE
* type (abc except cef ) subset abc     BLIR    TRUE
```

```
ALPHA( grupper )      ( lager en alfabetisk liste av gruppenavnene)
REVERSE ( grupper)    ( reverserer listen av gruppenavnene)
UNIQUE ( grupper)     ( Fjerner eventuelle dubletter )
```

```
type ! alpha ( cba)              BLIR    {a,b,c}
type ! reverse (aab)             BLIR    {b,a,a}
type ! unique (aab)              BLIR    {a,b}
type ! reverse(unique(aab))      BLIR    ( b,a}
```

BRUK AV WILDLIST I NAMELIST:

Med WILDLIST kan man bruke en slags jokernotasjon for å lage grupper/namelist:

```
liste = wildlist(demo,"inntekt")
!type liste              BLIR    {inntekt }

liste = wildlist(demo,"?")
!type liste              BLIR    { ALLE seriene som finnes i demo.db}

liste = wildlist(demo."?NORGE?")
!type liste              BLIR    { ALLE de serier som har NORGE i
                                 navnet sitt}
```

Hvis du ønsker den jokernotasjonen som finnes i UNIX, er dette mulig ved å lage en kombinasjon av FAME- og SCRIPT-programmer.

BRUK AV LØKKER I NAMELIST/GRUPPER

Anta at man har følgende SERIER:

kuer.danmark
kuer.norge
kuer.sverige
rein.sverige
rein.norge

Ønsker du å skrive ut alle seriene vha. av en LOOP kan du gjøre følgende:

Først må du definere DYR og LAND i vha. NAMELIST:
dyr = { kuer,rein}
land={norge,sverige,danmark}

&-tegnet blir oppfattet som LIM og kalles i FAME for GLUE. Bruker du punktum for å skille mellom dimensjonene i filnavn, må du si at limet skal være punktum. Dette gjøres ved å sette:
GLUE DOT

```
GLUE DOT
LOOP FOR x IN dyr
   LOOP FOR y IN land
     if exists(x&y)
        display x&y
     end if
   END LOOP
END LOOP
```

Bruker man ESCAPE inne i en LOOP, kastes man helt ut av løkken.

Hvis alle kombinasjonene av LAND og DYR finnes, kan man bruke CROSSLIST:

* DISPLAY !CROSSLIST (DYR, LAND)
Skriver ut alle serienavnene du har i kombinasjonene fra DYR og LAND.

* DISPLAY !crosslist (DYR, LAND EXCEPT {Danmark})
Skriver ut alle kombinasjonene unntatt de seriene som inneholder delnavnet Danmark.

RAPPORT

En av FAMEs sterke sider er muligheten for å lage rapporter.

Eksempel på en enkel rapport er:
* date jan93 to mar93,jul93
* report inntekt, kostnad, inntekt - kostnad AS "netto"

	Jan 1993	Feb 1993	Mar 1993	Jul 1993
INNTEKT	13.00	12.00	14.00	14.00
KOSTNAD	4.00	5.00	6.00	7.00
netto	9.00	7.00	8.00	7.00

* show vertical
* justify column heading center
* report inntekt, kostnad, inntekt - kostnad as "Netto!etter!kostnad"

	INNTEKT	KOSTNAD	netto etter kostnad
Jan 93	13.00	4.00	9.00
Feb 93	12.00	5.00	7.00
Mar 93	14.00	6.00	8.00
Jul 93	14.00	7.00	7.00

Det finnes en mengde opsjoner man kan bruke i REPORT. For å lære om dem bør du kaste deg over REPORT-manualen.

Eksempler på opsjoner er:
COMMA, DECIMAL, PREFIX, SUFFIX, JUSTIFY COLUMN, HEADING, CAPTION, DESCRIPTION, TITLE

Vil du lage mer strukturerte rapporter anbefales det at du bruker blokksystemet i report.

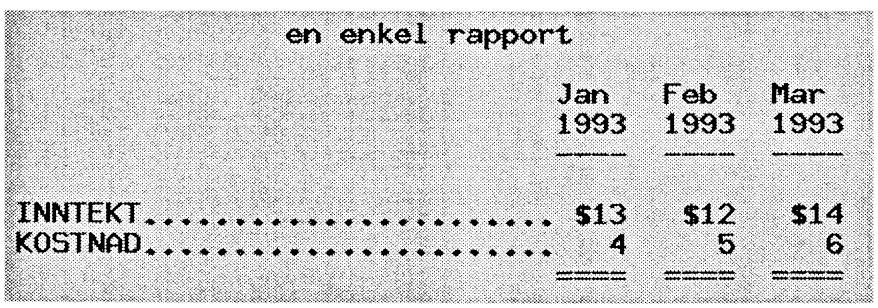
Følgende to kommandosekvenser er like:

* REPORT inntekt, kostnad

* REPORT
* SELECT DATE (skriv inn område)
* PRINT inntekt, kostnad
* END REPORT

Prøv følgende:

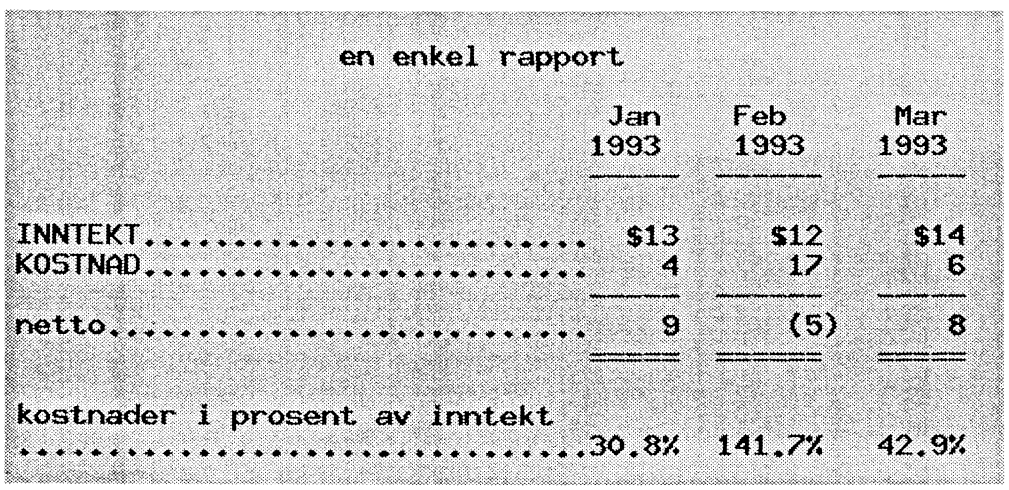
```
REPORT
 TITLE "en enkel rapport"
 DECIMAL 0
 DESCRIPTION WIDTH 30, FILL DOT, GAP 0
 SELECT DATE ( JAN93 to MAR93)
 PRINT inntekt < PREFIX "$" >, kostnad
 SCORE "="
END REPORT
```



WIDTH betyr hvor mange tegn som skal reserveres til beskrivelsesfeltet.
FILL DOT betyr at ledige plasser skal fylles ut med punktum. Man kan også skrive FILL "."
GAP 0 betyr at det skal være 0 plasser mellom beskrivelsesfeltet og tabellen.
SCORE "=" betyr at det skal skrives en linje med likhetstegn.

Prøv følgende eksempel:

```
REPORT
 FREQ MONTH
 NEGATIV PARENS
 TITLE "en enkel rapport"
 DECIMAL 0
 DESCRIPTION WIDTH 32, FILL DOT, GAP 0
 SELECT DATE ( JAN93 to MAR93)
 PRINT inntekt < PREFIX "$">
 PRINT kostnad
 SCORE
 PRINT inntekt - kostnad AS "netto"
 SCORE "="
 BLANK
 TEXT "kostnader i prosent av inntekt "
 SUFFIX "%"
 DECIMAL 1
 PRINT 100*kostnad/inntekt AS ""
END REPORT
```



en enkel rapport

	Jan 1993	Feb 1993	Mar 1993
INNTEKT	$13	$12	$14
KOSTNAD	4	17	6
netto	9	(5)	8
kostnader i prosent av inntekt	30.8%	141.7%	42.9%

NEGATIV PARENS betyr at det skal legges parantes rundt negative tall.
SUFFIX legger et bestemt tegn etter tallet.
BLANK legger inn en blank linje

Prøv følgende eksempel

```
REPORT
JUSTIFY COLUMN HEADING CENTER
SELECT DATE ( jan93, mar93)
SELECT VALUE @STUB [mar93] - @STUB[jan93] AS "økning fra!jan til mar"
SELECT VALUE 100 * ( @STUB[mar93] / @STUB[jan93] - 1 ) AS  &&
       "prosent!økning" < DECIMAL 1 >
PRINT inntekt, kostnad
END REPORT
```

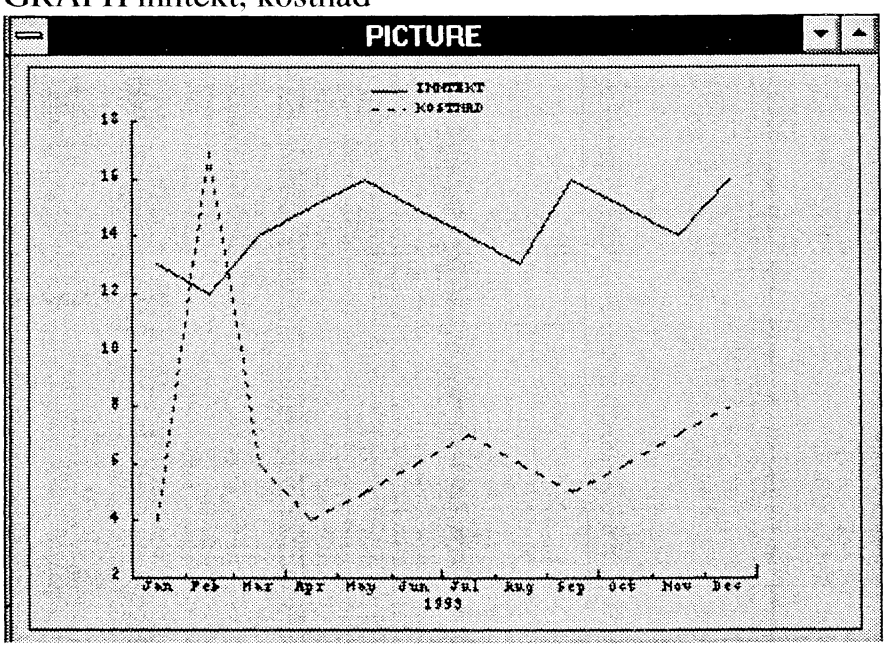
	Jan 1993	Mar 1993	økning fra jan til mar	prosent økning
INNTEKT	13.00	14.00	1.00	7.7
KOSTNAD	4.00	6.00	2.00	50.0

@STUB referer til kolonne. Hadde man ikke brukt SELECT VALUE måtte man først ha laget en formel for dette.

GRAFIKK

En av FAMEs sterke sider er grafikkdelen. Ønsker du å lære mer enn hva du kan finne her, må du kaste deg over grafikk-manualen.

GRAPH inntekt, kostnad



graph inntekt ,mave(inntekt) < plot red >

Man kan legge inn 6 nivåer med titler i GRAPH.
TITLE #1 TEXT "trist regnskap", BLUE, GOTHIC,MEDIUM

trist regnskap

Ønsker du å lage spesialiserte grafer utenfor den standard du får ved å bruke GRAPH, må du lære deg hvordan man lager grafer i et eget blokksystem for grafer.

Følgende to sekvenser er like:

* graph inntekt

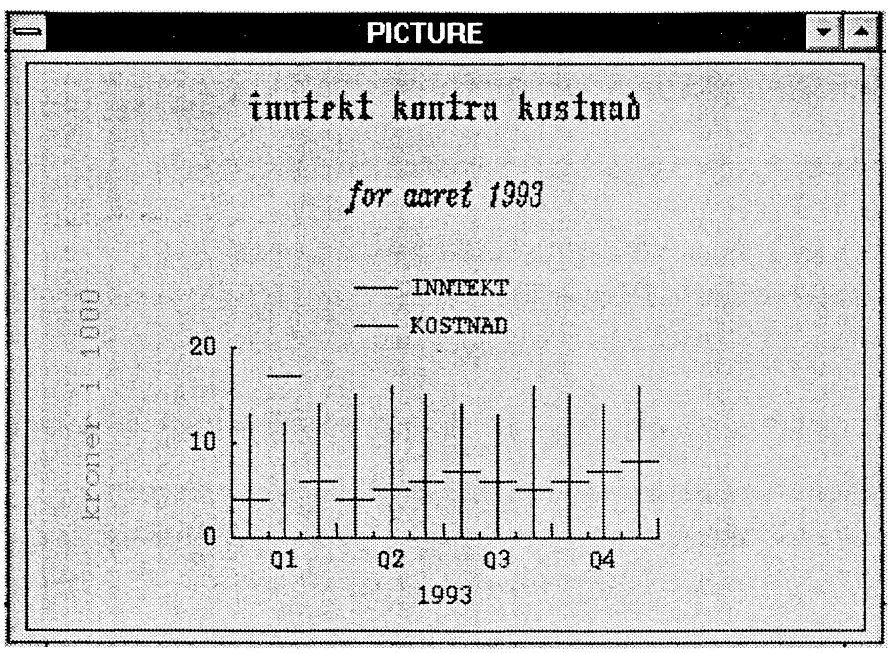
* graph
* data inntekt
* end graph

Linjediagram er DEFAULT.
* PLOT DRAW LINE

Ønsker man et historisk diagram (søylediagram) kan man skrive:
* PLOT DRAW BAR

eksempel:

```
GRAPH
FREQ M
TITLE #1 "inntekt kontra kostnad"
TITLE #2 BLANK
TITLE #3 " for aaret 1993 ", FONT ITALICS, SIZE SMALL
LABEL LEFT "kroner i 1000", GREEN
TITLE #4 BLANK
LEGEND ON
DATE JAN93 to DEC93
DATA inntekt < plot vertical, solid, blue> , &&
     kostnad < plot horizontal, solid, black>
end graph
```



LEGEND FRAME ON
(legger en ramme rundt forklaringsvariablene)

TICK LABEL RIGHT ON
(Skriver også aksetallene på høyre side av grafen)

TICK LEFT ZERO
(Starter aksen fra null)

PLOT COLOR #1 green, #2 blue, #3 black
(setter på bestemte farger på grafene 1,2 og 3)

CAPTION #1 " tekststreng", SMALL
(legger på en undertekst til grafen som skal være med liten skrift)

PLOT #1 THICKNESS THICK
(Plotter ut graf 1 med tykk strek)

UNIX FRA FAME

Du kan kalle opp en UNIX-kommando direkte i FAME ved å bruke utropstegn foran kommandoen.

```
! unix-kommando
! ls -l
```

ELLER

```
system(unix-kommando)
system("ls -l")
```

UTNYTTELSE AV UNIX I FAME

UNIX-kunnskaper kan forenkle arbeidet ditt i FAME.

For sakte OUTPUT-vindu?

Synes du at utskriften til Output-vinduet går for sakte? Man kan da omdirigere dette til UNIX-vinduet, og så bruke en UNIXkommando for å se på resultatet:

I FAME:	OUTPUT fil.txt
I UNIX:	tail -f fil.txt
	Alt hva du gjør i FAME vil nå komme på UNIX-vinduet.
	Gidder du ikke lenger stopper du
	UNIX-kommandoen med CTRL-C.
I FAME:	OUTPUT Terminal

Fordelen med dette er at det går LYNRASKT! Ulempen er at man da IKKE kan bla seg bakover for å se på tidligere resultater.

Innhold i biblioteksfil:

Har du en biblioteksfil med en hel haug prosedyrer og funksjoner, kan man fort miste oversikten over hva man har. I UNIX kan man bruke egrep-kommandoen for å få en rask oversikt.

Anta at biblioteksfilen din heter bibliotek.pro. Med følgende kommando får du listet ut navnet på alle prosedyrene og funksjonene du har tilgjengelig i filen:

```
egrep -ni  '^PROC|^FUNC'  bibliotek.pro
```

Miljøvariable

Skal du åpne en famedatabase som ligger på et annet område enn der du startet opp fame, må man spesifisere hele søkestrengen der famedatabasen ligger.

eks:

```
open "/local/fame/level1/DEMO/DB/demo" AS demo
```

Har du ofte behov for å åpne databaser som ligger på en slik katalog, kan man lage en miljøvariabel i UNIX. Denne kan så brukes i FAME-kommandoen:

```
i UNIX:     set KATALOG = "/local/fame/level1/DEMO/DB"

i FAME:     open "$KATALOG/demo" AS demo
```

Lager man en miljøvariabel direkte i UNIX vil den forsvinne med en gang du logger av. For å lage en miljøvariabel som skal være der "for alltid", kan du legge set-kommandoen inn i oppstartfilen din. Den heter .cshrc

MANIPULERING MED VINDUER

Ønsker du å legge bestemte resultater til egne "analyse"-vinduer er dette fullt mulig. Du kan også styre størrelsen på vinduer og hvor de skal være plassert på skjermen.

Output < window > dialog
-- Resultatene legges rett i dialogvinduet.

Create < window > analyse
-- Lager et eget analysevindu.

Create < window top 0, left 500, width 200, height 200 &&
        background red, foreground blue > analyse
-- top og left sier hvor på skjermen vinduet skal plasseres. Windows deler
-- skjermen opp i 500 * 500 punkter.
-- width og height forteller noe om størrelsen på vinduet.
-- background og foreground gir de fargene som ønskes.

hide < window > analyse
-- Gjemmer vinduet

reveal < window top 0, left 0> analyse
-- Henter frem vinduet igjen og forteller hvor det skal plasseres.

close < window > analyse
-- Lukker vinduet fullstendig.

create < window; kind graphics > grafikkvindu
-- Grafikk har en egen standard. Dette må spesifiseres. KIND TEXT er default.

UTSKRIFT TIL SKRIVER

Skal du ha skrevet ut grafikk og rapporter til skriver må du du først lage postscriptfil av det du skal ha skrevet ut:

Device Graph postp
Picture < access over > filnavn
channel reports picture

For å stille dette tilbake til "normalen" igjen, må du gjøre følgende:

Device Graph x
Picture terminal
Channel reports output

For å sende postscriptfilen til skriver er det nok å skrive (i UNIX)
lpr filnavn
eller
f.eks: ps720 filnavn

EKSEMPEL PÅ PROCEDURE som skriver ut en graf til skriver:

```
procedure $grafut
argument navn
block
overwrite on
device graph postp
picture < access over > skrivfil.txt
graph navn
device graph x
picture terminal
system ("til_skriver")
end block
end procedure
```

scriptet "til_skriver" ser slik ut:

```
#!/bin/sh
lpr skrivfil.txt
```

MEGET KORT OM FORBINDELSEN MELLOM FAME OG TROLL

Man starter NYE TROLL i UNIX ved å skrive:
troll

Man avslutter NYE TROLL ved å skrive:
trexit

Trykker man ENTER i TROLL får man hjelp

Enhver kommando i TROLL avsluttes med semikolon. Trykker man ENTER får man et hint om hvordan man skal fortsette å skrive inn kommandoen.

Adgang til famedatabase:
access aa type famedata id /ssb/frisch/fame/kvarts/kvdata87.db mode read ;

aa	:	Betyr at famedatabasen i TROLL skal hete aa
type	:	Dette sier om i hvilket dataformat databasen er lagret. I dette tilfellet er det FAMEdata.
id	:	Etter "id" må man skrive i hvilken UNIXkatalog databasen ligger og hva den heter.
mode	:	står det mode read kan man bare lese databasen. Man kan også velge "create" eller "write".

for å kjøre en UNIX-kommando fra TROLL kan man skrive:
host "unixkommandoen"
eks:
host "ls -l "
host "fame"

Henting av en tidsserie:
search data aa;
do prt.(c11);

ØNSKER DU INFORMASJON OM NYE TROLL MÅ DU KONTAKTE JØRGEN OUREN.

Statistisk sentralbyrå

Oslo
Postboks 8131 Dep.
0033 Oslo

Tlf.: 22 86 45 00
Fax: 22 86 49 73

Kongsvinger
Postboks 1260
2201 Kongsvinger

Tlf.: 62 88 50 00
Fax. 62 88 50 30

Statistisk sentralbyrå
Statistics Norway