Supplerende noter til 'Kursus i brug af SAS'

Søren Højsgaard

Biometry Research Unit
Danish Institute of Agricultural Sciences
Research Centre Foulum
DK 8830 Tjele

Flemming Skjøth

Dansk Kvæg
Landbrugets Rådgivningscenter
DK 8200 Århus N

Indhold

Forord

Dette notesæt giver en simpel indføring i brugen af SAS systemet. Notesættet er udarbejdet med henblik på et kursus af 3 dages varighed. Der forudsættes ingen forudgående kendskab til SAS. Notesættet anvendes som supplement til bogen *Elementær indføring i SAS* af Andersen, A.T., Bedsted, T.V., Feilberg, M., Jakobsen, R.B. og Milhøj, A., Akademisk Forlag 2002. Der henvises til forlagets hjemmeside http://www.akademisk.dk for yderligere informationer, hvorfra de datasæt der benyttes i bogen også kan hentes.

Det datamateriale, samt de SAS programmer, der anvendes i dette notesæt kan hentes fra

```
http://www.jbs.agrsci.dk/biometri/SASmateriale/
```

En beskrivelse af datamaterialet samt organiseringen heraf findes i afsnittene 9.1 og 9.2. Det anbefales, at man indledningsvis læser disse afsnit.

Notesættet indeholder en del opgaver, som man opfordres til at arbejde med. I tillæg hertil forslås det at man i `SAS` hjælpefunktionen går ind under punktet "Sample SAS Programs and Applications". Herunder finder man, som navnet antyder, en række eksempler. Der opfordres til at man afprøver disse eksempler, og prøver at modificere dem efter eget ønske.

Kapitel 7 forudsætter et vist kendskab til statistiske metoder (variansanalyse, analyse af split–plot forsøg, analyse af blok forsøg, simpel regressionsanalyse og multipel regressionsanalyse).

Søren Højsgaard og Flemming Skjøth
November 2002

Kapitel 1

Introduktion til SAS systemet og dets opbygning

1.1 Mit første SAS program

Et eksempel på et meget simpelt SAS program er følgende:

```
/* Mit første SAS program */    /* 0: En kommentar                  */
DATA eksempel;                  /* 1: Start på DATATRIN             */
  a = "En tekst";               /* 2: danner en tekst variabel a    */
  x = 3.17;                     /* 3: danner en numerisk variabel x */
RUN;                            /* 4: Afslutter datatrin            */

TITLE 'Mit første SAS program '; /* 5: En lille tekst som overskrift */
PROC PRINT DATA=eksempel;       /* 6: Print data i output vinduet   */
RUN;                            /* 7: Udfør ovenstående             */
```

Eksemplet illustrerer, hvoledes SAS programmer generelt er bygget op af to slags komponenter:

Datatrin: Datatrin hvori data kan indlæses og manipuleres.

Proceduretrin: Proceduretrin hvori der foregår en behandling af data.

Linierne i eksemplet skal læses som følger:

/* **0** */ Dette er en kommentar. Tekst angivet mellem /* og */ ignoreres af SAS

/* **1** */ Et datatrin starter med ordet `DATA` og datasættet hedder EKSEMPEL

/* **2** */ Tekst variablen skal hedde a

/* **3** */ Tal variablene skal hedde x

/* **4** */ `RUN;` afslutter et datatrin

/* **5** */ `PROC PRINT` .... fortæller at SAS datasættet EKSEMPEL skal skrives ud på skærmen.

/* **7** */ `RUN;` fortæller at SAS skal udføre kommandoen specificeret med `PROC PRINT`

1.2 Hvad er et datasæt

For at procedurer i SAS skal kunne foretage beregninger på data, skal disse være lagret i et såkaldt SAS-datasæt. I et SAS-datasæt er data lagret binært i et bestemt format sammen med visse følgeoplysninger. Som bruger kan man betragte et SAS-datasæt som en navngiven tabel med et antal rækker og søjler.

- Hver søjle kaldes en variabel og indeholder alle data fra en bestemt egenskab, f.eks. en faktors betegnelse, et bloknummer o.s.v. En variabel har et entydigt navn og kan være enten numerisk eller alfanumerisk (character).
- Hver række i et SAS-datasæt kaldes en observation og indeholder typisk data fra en bestemt parcel, en bestemt plante eller et bestemt dyr.

I praksis har man ofte gemt data på en fil. Dette kan f.eks. enten være som

- som en almindelig tekstfil, også kaldet en ASCII fil, eller
- som et Excel regneark.

1.3 Hvad er en procedure?

En af SAS-systemets hovedbestanddele er procedurene. En procedure er et programmodul, som kan at udføre en bestemt opgave. Nogle af de ting man bruger procedurer til, ville man også kunne programmere selv i et datatrin, men det ville kræve megen programmering. Man kan opfatte SAS-procedurer som

> *Et kraftfuldt programmeringssprog som med meget få kommandoer kan udføre komplicerede opgaver.*

Prisen for dette kraftfulde sprog, som muliggør at man med få ordrer kan få tegnet grafer, udskrevet tabeller og udført komplicerede statistiske beregninger, er at man skal overholde nogle temmelig stive regler. Det gælder først og fremmest med hensyn til det datasæt, der arbejdes med. Næsten alle procedurer accepterer kun et SAS-datasæt som input. Det er her samspillet mellem DATA-trin og PROCEDURE-trin kommer ind: Med DATA-trinnet omformer man rådata til data som kan anvedes af SAS-procedurene. I proceduretrinet behandler vi disse data på forskellige måder.

Der vil ofte være mulighed for at komme med yderligere specifikationer, såkaldte ”options”, efter procedurenavnet. Specifikationerne afhænger af, hvilken type procedure

man anvender. En specifikation som kan anvendes i næsten alle procedure er dog: DATA=datasætnavn.

```
PROC PRINT DATA=sashelp.class;
```

Specificerer man ikke `DATA=` vil SAS anvende det senest oprettede datasæt. Af hensyn til læseligheden af sit SAS program anbefales det at man specifcerer `DATA= .... ;`.

Der kan derefter følge et antal linier med andre kommandoer, som giver yderligere specifikationer til proceduren. Disse specifikationer kan enten være valgfrie, ”optional”, eller nødvendige, ”required”, for at proceduren kan udføres.

1.3.1 PROC SORT

`PROC SORT` sorterer datasæt efter værdien af angivne variabler. Der kræves altid et `BY`-statement sammen med `PROC SORT`, der angiver de variabler, som datasættet skal sorteres efter. Sammen med `PROC SORT` kan der v.h.a. `DATA=`datasæt angives inputdatasæt, og v.h.a. `OUT=`datasæt kan der angives, hvor det sorterede datasæt skal anbringes. Hvis der ikke angives noget `OUT=`datasæt bliver det sorterede datasæt anbragt i det datasæt, der blev brugt som inputdatasæt.

```
TITLE 'PROC SORT - Data sorteret efter "køn (sex)"';

PROC SORT DATA=sashelp.class OUT=a;
   BY sex;
PROC PRINT DATA=a; RUN;
```

Med `BY`-statement er det muligt, at angive flere variable, som datasættet skal sorteres efter. Den første variabel, der angives bliver det primære sorteringskriterium; den anden det sekundære kriterium o.s.v. Hvis datasættet ønskes sorteret i omvendt rækkefølge for en bestemt variabel skal nøgleordet `DESCENDING` angives før den pågældende variabel.

```
TITLE 'PROC SORT - Data sorteret efter "køn (sex)"';
TITLE2 'og faldende værdi af "alder (age)"';

PROC SORT DATA=sashelp.class OUT=a;
   BY sex DESCENDING age;

PROC PRINT DATA=A; RUN;
```

I ovenstående eksempel bliver datasættet sorteret efter stigende værdi af optagn som det primære kriterium og faldende værdi af saatid som det sekundære kriterium.

1.3.2 PROC PRINT

`PROC PRINT` bruges, som vi tidligere har set, til udskrivning af datasæt. Sammen med `PROC PRINT`-ordren kan der angives forskellige options, f.eks. hvilket datasæt der ønskes udskrevet ved angivelse af `DATA=`datasætnavn.

```
TITLE 'Demo af PROC PRINT';
PROC PRINT DATA=sashelp.class; RUN;
```

En anden vigtig option, der kan angives sammen med `PROC PRINT`, er `LABEL`, som bevirker, at variabelbeskrivelserne, der blev angivet ved datasættets oprettelse, fremkommer som kolonneoverskrifter.

Med `VAR`-statement kan man udvælge en liste af variable, som ønskes udskrevet. Variablerne udskrives i den rækkefølge, de nævnes i `VAR`-ordren.

```
TITLE 'Demo af PROC PRINT med VAR';
PROC PRINT DATA=sashelp.class LABEL;
   VAR name age;
RUN;
```

Med `BY`-ordren angives, at datasættet skal skrives ud efter værdien af en bestemt variabel. Når `BY`-ordren benyttes kræves det generelt, at datasættet i forvejen er sorteret efter den variabel, der benyttes i `BY`-ordren.

```
TITLE 'Demo af PROC PRINT med BY-ordren';
PROC SORT DATA=sashelp.class OUT=a;
   BY sex;
PROC PRINT DATA=a;
   VAR name age;
   BY sex;
RUN;
```

Demo af PROC PRINT 1

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5

11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Demo af PROC PRINT med BY-ordren 2

---------- Sex=F ----------

Obs	Name	Age
1	Alice	13
2	Barbara	13
3	Carol	14
4	Jane	12
5	Janet	15
6	Joyce	11
7	Judy	14
8	Louise	12
9	Mary	15

---------- Sex=M ----------

Obs	Name	Age
10	Alfred	14
11	Henry	14
12	James	12
13	Jeffrey	13
14	John	12
15	Philip	16
16	Robert	12
17	Ronald	15
18	Thomas	11
19	William	15

1.3.3 PROC CONTENTS

For at få en oversigt over hvilke variable der indgår i et datasæt kan man benytte procedure `PROC CONTENTS`, der køres som i følgende eksempel:

```
PROC CONTENTS DATA=sashelp.class; RUN;
```

1.3.4 PROC GPLOT

`PROC GPLOT` bruges til grafisk afbildning af en variabel mod en anden. De dannede grafer kommer i GRAPH-vinduet.

`PROC GPLOT` kan anvendes på følgende måde:

```
TITLE 'Demo af PROC GPLOT';
PROC GPLOT DATA=sashelp.class;
   SYMBOL I=r1 V=dot C=blue;
   PLOT heigth * age;
RUN;
```

Her afbildes „height" som funktion af „age". Med `PLOT`-statement angives altså, hvilke variable, der ønskes plottet mod hinanden. `PLOT`-ordren kan for eksempel anvendes på disse måder:

```
PLOT Y*X;           *markerer observationer som punkter på grafen;
PLOT Y*X=variabel;  *viser grafen med forskellige symboler svarende til ;
                    *værdien af variablen, der står på højre side af    ;
                    *lighedstegnet.                                     ;
```

`BY`-ordren kan som i andre procedurer også anvendes sammen med `PROC GPLOT`.

```
TITLE 'Demo af PROC PLOT med BY-ordren';

PROC SORT DATA=sashelp.class OUT=a;
   BY sex;

PROC GPLOT DATA=A;
   PLOT height * age;
   BY sex;
RUN;
```

Der bliver ved anvendelse af `BY`-ordren dannet et plot for hver værdi af BY-variablen. Som sædvanligt kræves datasættet sorteret efter BY-variablen.

1.4 Øvelser

Opgave 1 *Gennemgå Skærmøvelse 2.3 i „Elementær inføring i SAS".*

Opgave 2 *I denne opgave skal SAS datasættet „class", der ligger i „sashelp", sorteres med „sex" som primært sorteringskriterium og „name" som sekundært kriterium.*

Det sorterede datasæt skal anbringes i et datasæt der skal hedde „klasse".

Opgave 3 *Udskriv datasættet „klasse“ dannet i opgave 2:*

1. *så kun variablerne „name age weigth“ udskrives.*
2. *sæt danske label på ved hjælp af „View Columns“ i Explorer vinduet.*
3. *udskriv så der kommer labels over kolonnerne.*

Opgave 4 *Tag igen udgangspunkt i datasættet „klasse“*

1. *Lav et plot med „alder“ ud ad x-aksen og „vægt“ op ad y-aksen.*
2. *Lav dernæst et plot for hver køn med de samme variable.*
3. *Lav et plot, hvor „vægt“ plottes mod „alder“ med „køn“ som symbol.*

Kapitel 2

Indlæsning og lagring af data

2.1 Indlæsning af data fra Excel regneark

Ofte har man sine data gemt som et Excel regneark. Man kan oprette et SAS datasæt ud fra Excel-filen ROER.xls ved hjælp af proceduren `PROC IMPORT`. Et eksempel der importerer arket 'ARK1' fra Excel regneark 'ROER.xls' er gengivet nedenfor.

```
PROC IMPORT OUT=work.roer
            DATAFILE= "c:\saskursus\raadata\roer.xls"
            DBMS=EXCEL2000 REPLACE;
     RANGE="Ark1$";
     GETNAMES=YES;
RUN;
```

Dette kan man med fordel gemme og genbruge hvis man skal indlæse mange Excel regneark. Alternativt kan man benytte sig af de makroer der omtales i Afsnit **??**.

En anden måde, der involverer lidt mere klikkeri er som følger:

1. I SAS, gå til Filer og herfra til Import
2. Ved pilen vælg 'Excel 97/2000 spreadsheet'. Klik derefter på Next
3. Klik på Browse for at finde Excelfilen roer.xls. Denne fil ligger i mappen `C:\SASKURSUS\RAADATA\`.
4. Klik eventuelt på Options for at sikre at SAS har valgt det rigtige ark i regnearket. Klik derefter på Next
5. I feltet Member skrives det navn man ønsker datasættet skal have i SAS. Klik derefter på Finish.
6. Gå nu til loggen for at sikre at datasættet faktisk er importeret.

Bemærk: Hvis regnearket stadig er åbent i Excel vil man få problemer med at importere regnearket.

Man kan nu se indholdet af det importede datasæt ved at køre programstumpen:

```
PROC PRINT DATA=roer;
RUN;
```

2.2 Indlæsning af data fra en tekstfil

For at få data fra en tekstfil omdannet til et SAS-datasæt, skal vi give forskellige oplysninger:

1. Hvad skal datasættet hedde?
2. Hvad hedder tekstfilen?
3. Hvilke variabelnavne vil vi benytte?
4. Hvordan læses disse fra tekstfilen?

Der er mange muligheder for indlæsning af data af denne type i SAS. Vi skal her beskrive nogle af de simpleste.

Data fra et forsøg med sukkerroer (se afsnit 9.2.1) kan indlæses med følgende 3 sætninger (statements):

```
TITLE 'Simpel indlæsning';

DATA roer;                                              /* 1 */
  INFILE 'c:\saskursus\raadata\roer.txt' FIRSTOBS=6; /* 2 */
  INPUT  optagn $ blok $ saatid $ kg  pct_sukk;      /* 3 */
RUN;                                                    /* 4 */

PROC PRINT DATA=roer;
RUN;
```

Disse linier læses som følger (idet der refereres til kommentarerne angivet i /* ... */):

/* 1 */: Her starter et datatrin og SAS-datasættet skal hedde ROER.

/* 2 */: Her angives navnet på den fil, der skal læses, dvs. den tekstfil hvori data ligger. Intruktionen `FIRSTOBS` specificerer at data først kommer i den 6. linie i filen (de første 5 linier er kommentarer).

/* 3 */: Her specificeres de 5 navne vi ønsker variablerne i datasættet skal have. Endvidere specificeres at variablerne der ønskes kaldt „optagn“, „blok“ og „saatid“ skal være tekststrenge (dette gøres ved at sætte et $-tegn efter navnet). Variablerne „kg“ og „`PCT_SUKK`“ skal begge være talvariable, hvilket de automatisk bliver når der ikke er sat et $-tegn efter hver af dem.

/* 4 */: Kommandoen `RUN;` fortæller at nu er datatrinnet slut.

Den sidste linie, `PROC PRINT ....; RUN;` har intet med datatrinnet at gøre men tjener alene til at printe data i output–vinduet.

Såvel et SAS datasæt som variablene i et sådant skal tildeles navne efter følgende regler:

1. Navnet skal starte med et bogstav (A-Z) eller understregningstegnet (_).
2. De efterfølgende tegn kan bestå af et bogstav (A-Z), understregningstegnet (_) eller et ciffer (0-9).
3. Der skelnes ikke mellem store og små bogstaver.
4. I SAS 8 kan navne højst være 32 tegn lange. I version 6.12 og tidligere versioner er grænsen 8 tegn.

Som det fremgår af tekstfilen ROER.txt er udbyttet `KG` angivet i Kilogram * 10 og sukkerindholdet `PCT_SUKK` i sukkerprocent * 10. Der er flere måder at få SAS til at læse data ind på således at tallene får den rigtige betydning. Dette vil vi komme ind på i kapitel 5.8.

2.3 Filename

Man kan bruge `FILENAME` til at referere til en bestemt tekstfil med. Dette kan være hensigtsmæssigt i store programmer. Et eksempel på brugen af `FILENAME` ved indlæsning af tekstfilen „roer.txt“ er følgende:

```
FILENAME indfil 'c:\saskursus\raadata\roer.txt';     /* 1 */
TITLE 'Brug af filename';

DATA roer;
   INFILE indfil FIRSTOBS=6;                         /* 2 */
   INPUT  optagn $ blok $ saatid $ kg pct_sukk;
RUN;

PROC PRINT DATA=roer;
RUN;
```

/* 1 */: Der specificeres at SAS ved navnet „INDFIL“ skal forstå filen

`c:\saskursus\raadata\roer.txt`.

/* 2 */: Ovenstående udnyttes her, hvor der blot refereres til INDFIL. SAS ved så hvad dette betyder. Dette er relevant hvis man i samme SAS program flere gange skal have fat i samme fil, eller hvis SAS programmet er meget stort.

Det anbefales at FILENAME–sætninger sættes i starten af SAS programmet.

2.4 Tilføjelse af Labels og Formats

Alle variabler i et datasæt kan tildeles ekstra oplysninger med sætningerne LABEL og FORMAT i et datatrin. Labels bruges til at få mere læsevenlige udskrifter med, mens formater bruges til at specificere hvorledes variablene skal udskrives. Man kan også angive en LABEL til selve datasættet.

```
TITLE 'Tilføjelse af labels og formater';
FILENAME indfil 'c:\saskursus\raadata\roer.txt';

DATA roer(LABEL='Data fra forsøg med roer');
   INFILE indfil FIRSTOBS=6;
   INPUT  optagn $ blok $ saatid $ kg pct_sukk;

   LABEL OPTAGN = ' Optagningstid'
           BLOK = ' Bloknummer'
         SAATID = ' Såtid'
             KG = ' Optagne roer i kg/parcel'
       PCT_SUKK = ' Sukkerprocent';                /* 1 */
   FORMAT OPTAGN BLOK SAATID $6. KG 4. PCT_SUKK 4.1;   /* 2 */
RUN;

PROC PRINT DATA=roer LABEL;                            /* 3 */
RUN;
```

/* 1 */: Med LABEL-sætningen tildeles alle variable beskrivelser. Disse kan være op til 40 tegn lange, og må indeholde alle tegn (dog er der særlige regler for at få apostroffer ind i teksten). Brugen af labels øger læseligheden af udskrifter fra SAS.

/* 2 */: Med FORMAT-sætningen tildeles KG et felt på 4 tegn (og afrundes til hele kg), mens PCT_SUKK udskrives med 1 decimal i et felt på 4 tegn. Bemærk: FORMAT-sætningen får ikke decimaler til at forsvinde fra datasættet. Det har udelukkende virkninger for det man ser i sine udskrifter, og har således ingen betydning for den præcision hvormed SAS foretager sine beregninger. De øvrige tekstvariable får tildelt 6 felter.

/* 3 */: Ved at tilføje ordet LABEL i kaldet af PRINT proceduren, opnås at det bliver variablenes labels der printes, i stedet for deres navne.

Formater

Formater anvendes til at bestemme, hvordan data skal indlæses/udskrives. SAS Systemet arbejder med to typer af formater: Indformater og udformater. Indformater fortæller SAS Systemet, hvordan værdier skal opfattes, når disse indlæses. Udformater fortæller SAS systemet, hvordan værdier skal udskrives, fx. i en rapport. Her er en kort oversigt over nogle af de mange formater der er tilrådighed.

Karakterformat

Et karakterformat bestemmer, hvordan karakterværdier skal behandles. En karaktervariabel hvor værdierne kun antager værdier med 3 bogstaver, f.eks. værdien 'HEL' kan gives formatet $3. : $-tegnet angiver, at formatet er et karakterformat. '3' angiver variablens længde. Dermed bruges mindst mulig plads i datasættet til at opbevare datene, samt til at skrive dem ud. Punktummet er altid en del af et formatnavn.

Numerisk format

Et numerisk format bestemmer, hvordan numeriske værdier skal behandles. Som standard anvender SAS Systemet formatet w.d, dvs. ingen tusindeadskillelse og punktum foran decimaler. Værdien 131.422,00 har formatet `COMMAX10.2`. `COMMAX` er et format, hvor værdierne vises med europæisk kommatering, dvs. punktum for tusinder og komma foran decimaler. 10 angiver variablens totale længde inkl. punktum og komma og 2 angiver decimalantallet.

Værdien 131.422,00 vist med andre formater:

`w.d` Eksempel:	samlet bredde.antal decimaler `9.2` viser 131422.00
`COMMAw.d` Eksempel:	amerikansk format `COMMA10.2` viser 131,422.00
`COMMAXw.d` Eksempel:	europæisk format `COMMAX10.2` viser 131.422,00
`Zw.d` Eksempel:	foranstiller nuller `Z11.2` viser 00131422.00

Datoformat

SAS Systemet gemmer datoværdier som tal. Dette letter bl.a. beregning af antallet af dage mellem to datoer.

Et dato-indformat bestemmer, hvordan SAS Systemet indlæser datoværdier. Disse omregnes til det antal dage, der er gået siden den 1. januar 1960. Formatet `DATE7.` omregner datoen, som brugeren indtaster, fx. 03MAR91, til tallet 11384.

Et dato-udformat bestemmer, hvordan datoværdier udskrives. `DATE7.` omregner tallet 11384 til datoen 03MAR91 og udskriver denne, fx. i en rapport.

Værdien 030391 vist med andre formater:

`DDMMYYw.` Eksempel:	DDMMYY (dato, måned og år som tal) `DDMMYY8.` viser 03/03/91
`DATEw.` Eksempel:	DDmånedsnavnÅÅ `DATE7.` viser 03MAR91
`WORDDATXw.` Eksempel:	DD månedsnavn ÅÅÅÅ `WORDATX18.` viser 3 March 1991
`TlMEw.` Eksempel:	timer:minutter:sekunder `TIME8.` viser 11:15:00

Et format kan f.eks. bruges i forbindelse med udskrifter:

```
PROC PRINT DATA= .. ;
VAR xdato y x;
FORMAT y x COMMAX10.1 xdato DDMMYY8.;
RUN;
```

2.5 Lagring af data - Libname

Ovenfor har vi set hvorledes man kan danne SAS-datasæt ved navn „roer“ ved enten at indlæse en tekstfil eller et Excel-regneark. SAS-datasættet „roer“ er rent fysisk gemt på harddisken i et specielt katalog under SAS-programmet. Når man lukker SAS slettes indholdet af dette katalog. Det betyder at SAS-datasættet „roer“ ikke vil være tilgængeligt næste gang man starter SAS. Man siger derfor at „roer“ er et midlertidigt datasæt eller temporært datasæt.

Hvis man senere skal benytte de samme data igen, vil det være en fordel at slippe for at indlæse og evt. omregne sine data igen. Et SAS-datasæt kan derfor også gemmes som et permanent datasæt som man har liggende på sin harddisk ligesom alle andre filer. Som med alle andre filer er det hensigtsmæssigt at organisere sine SAS-datasæt. Således vil vi igennem dette notesæt gemme alle permanente SAS-datasæt i biblioteket
`C:\SASKURSUS\SASDATA\`

At “gemme” og “åbne” et SAS–datasæt foregår på en speciel måde der IKKE ligner det man er vant til fra andre programmer, f.eks. Excel, hvor man direkte

1. åbner et regneark,
2. gør noget og
3. gemmer det igen.

Den hensigtsmæssige måde at gemme et datasæt permanent på er følgende. Med sætningen `LIBNAME` tildeles biblioteket `C:\SASKURSUS\SASDATA\` et navn som vi selv kan vælge – vi har valgt navnet `MSAS`, men kunne ligeså godt have kaldt det SVENDIB, PROJEKT eller lignende.

Dernæst startes et datatrin, hvor der på datasættets plads står et to-delt navn:

- Den første del af navnet, er det valgte navn på biblioteket (her `MSAS`).
- Den anden del er selve datasættets navn (her `ROER`).

```
LIBNAME  msas 'c:\saskursus\sasdata';

DATA msas.roer;
     SET roer;
RUN;

PROC PRINT DATA=msas.roer;
RUN;
```

Sætningen `SET` angiver, at det permanente datasæt „msas.roer“ skal dannes ud fra det midlertidige datasæt „roer“. Der herefter to ens datasæt: Datasættet „roer“ der ligger i `C:\SASKURSUS\SASDATA\` og datasættet „roer“ der ligger i et andet katalog hvis indhold slettes når SAS lukkes. (Alternativt kunne man have dannet det permanente datasæt „msas.roer“ straks uden først at danne et midlertidigt SAS-datasæt. Overvej selv hvordan!)

Kaldet af `PROC PRINT` tjener alene til at sikre at det ønskede permanente datasæt er dannet.

Næste gang man starter SAS-programmet, så har man adgang til det permanente SAS-datasæt „msas.roer“ med følgende programstump:

```
LIBNAME  msas 'c:\saskursus\sasdata';

PROC PRINT  DATA=msas.roer;
RUN;
```

Man bør bemærke sig følgende:

- Som det ses skal man altså ikke ”åbne”et datasæt på samme måde som man skal med et Excel-regneark. Man har umiddelbart adgang til et SAS-datasæt i det øjeblik man med et `LIBNAME` har fortalt SAS hvor det ligger.
- Når man starter SAS op på ny, skal man altså køre `LIBNAME`–linien. Dette behøver man kun gøre een gang i en SAS session.
- Det anbefales at `LIBNAME` sætninger sættes i starten af programmet.

Libnames er meget bekvemme og fleksible at arbejde med. Hvis man beslutter sig for at alle sine SAS datasæt skal flyttes fra eet katalog til et andet, så skal man kun ændre een linie, idet man blot skal angive det nye katalog som data er flyttet til.

Det skal understreges at man kan have flere forskellige Libnames, der refererer til det samme bibliotek, samt at selve navnet på et libname blot er noget der “findes internt i SAS”.

Eksempelvis kan man skrive:

```
LIBNAME  minedata 'c:\saskursus\sasdata';
LIBNAME  msas 'c:\saskursus\sasdata';

PROC PRINT  DATA=minedata.roer;
PROC PRINT  DATA=msas.roer;
RUN;
```

Resultatet af de to `PRINT`–sætninger er det samme idet „minedata“ og „msas“ begge refererer til det samme katalog på harddisken nemlig `C:\SASKURSUS\SASDATA\`.

2.6 SAS datasæt i forskellige versioner

I skrivende stund er SAS version 8.2 den seneste SAS version. Man vil imidlertid ofte komme ud for at skulle arbejde med SAS data gemt i en tidligere version af SAS, eksempelvis SAS 6.12. Dette er der ikke nogen problemer i, idet SAS 8.2 sagtens kan læse data gemt i SAS 6.12, mens SAS 6.12 ikke kan læse datasæt gemt i version 8.2.

Dog skal man være opmærksom på følgende: Antage at vi har brugt `LIBNAME` sætningen som beskrevet ovenfor. Hvis der i `C:\SASKURSUS\SASDATA\` alene ligger data gemt i version 6.12 og man laver et nyt datasæt som man også forsøger at gemme i `C:\SASKURSUS\SASDATA\`, så er vil dette nye datasæt også blive gemt som værende i version 6.12.

Skulle man nu i det nye datasæt have lavet en variabel med et navn på mere end 8 tegn, så vil SAS komme med en meddelelse om at det kan man ikke. Omvendt, dersom `C:\SASKURSUS\SASDATA\` indeholder mindst eet datasæt gemt i version 8.2 så vil efterfølgende datasæt også blive gemt i version 8.2. Dette kan være uheldigt hvis man skal videregive datasættet til en kollega, der kun har version 6.12.

Løsningen på disse problemer er angivet nedenfor, hvor det i `LIBNAME` sætningerne angives i hvilket format data skal gemmes:

```
LIBNAME  msas6 v612 'c:\kursus\sasdata';
LIBNAME  msas8 v8   'c:\kursus\sasdata';

DATA msas6.roer6; SET roer;  RUN;
DATA msas8.roer8; SET roer;  RUN;
```

Datsættet „ROER6“ er nu i version 6.12 formatet mens „ROER8“ er i version 8.2 formatet.

2.7 Øvelser

Opgave 5 *Tag udgangspunkt i datasættet „sashelp.class“ vi tidligere har set på.*

1. *Lav en midlertidig version kaldet klasse og udvid programmet så der sættes passende beskrivelser på og tilføj formater på højde og vægt så de udskrives uden decimaler.*
2. *Lad også programmet fremstille et permanent SAS-datasæt, som placeres i biblioteket* `C:\SASKURSUS\SASDATA\`
3. *Gem programmet igen.*

Opgave 6 1. *I filen* `KEPA-NAVNE.XLS` *findes data fra Opgave 1 som et Excel regneark.*
2. *Importer dette regneark til SAS, og giv datasættet navnet KEPAEXC.*
3. *Lav et program der gemmer datasættet KEPAEXC i et permanent SAS-datasæt, som placeres i biblioteket* `C:\SASKURSUS\SASDATA\`
4. *Importer igen regnearket, men nu således at kun rækkerne 6 til 10 indlæses (Tip: se skærmøvelse 3.7 i „Elementær indføring i SAS“.*

Opgave 7 *Prøv med udgangspunkt i følgende eksempel*

```
DATA a1;
  INPUT  dato DATE9. tekst & $ 30. Note $;
  FORMAT dato DDMMYY.;
CARDS;
04jan1999 Den 4. januar 1999     Tirsdag
;

PROC PRINT DATA=a1;
RUN;
```

at eksperimentere med nogle af de datoformater og numeriske formater, der er omtalt på side 12, prøv også at flytte `FORMAT` *instruktionen ned til* `PROC PRINT` *instruktionen. Prøv også at fjerne „&“-tegnet i* `INPUT` *ordren.*

Kapitel 3

PROC MEANS - en udvidet lommeregner og ODS

3.1 PROC SUMMARY og PROC MEANS

`PROC MEANS` beregner beskrivende statistik (middelværdi, median, standardafvigelse o.s.v.) på numeriske variabler i et SAS datasæt og udskriver som standard resultatet i et nyt SAS-datasæt. Proceduren `PROC SUMMARY` er identisk med `PROC MEANS` med undtagelse af, at den som standard ikke udskriver resultatet til skærmen. I begge procedurer kan man ændre standard udskrivningsmåden.

Et simpelt eksempel på brugen af `PROC MEANS` er følgende:

```
PROC MEANS DATA=saskurs.roer;
    VAR kg pct_sukk;
RUN;
```

Man angiver, at man vil benytte `PROC MEANS` proceduren ved at skrive dette først, derefter angiver man det datasæt, der skal benyttes (medmindre man vil benytte det sidst oprettede - (_LAST_). `NOPRINT` optionen angives for at undgå at få skrevet resultatet ud på skærmen. Endelig angives i `VAR` ordren hvilke variable man vil regne på.

I dette eksempel har vi ikke specifikt angivet hvilke beregninger vi vil have udført, så `PROC MEANS` udskriver som standard for hver variabel i `VAR` ordren: N, navn på variablen, minimum værdi, maximum værdi, middelværdi og spredning.

Vil man bestemme, hvilke beregninger man vil have, angives de ønskede beregninger ved hjælp af nøgleord. Udskrives til skærmen skal nøgleordene skrives i `PROC MEANS` ordren. Ønsker vi f.eks. beregnet middelværdi, spredning, sum og variationskoefficient får vi følgende program:

```
PROC MEANS DATA=saskurs.roer PRINT MEAN STD SUM CV;
    VAR kg pct_sukk;
RUN;
```

For at få resultaterne gemt som et SAS-datasaet, skal man tilføje en OUTPUT-ordre, hvor man angiver output datasættets navn, hvilke beregninger man vil have foretaget, og navn på de nye variable indeholdende resultaterne.

```
PROC MEANS DATA=saskurs.roer;
    VAR kg pct_sukkkg pct_sukk;
    OUTPUT OUT=roersum  MEAN =
                        STD  = stdhkg stdsuk
                        SUM(kg) = sumhkg;
RUN;
PROC PRINT DATA=roersum; RUN;
```

Resultatet af beregningerne skrives nu ud i det temporære datasæt „roersum“ (som forsvinder, når man 'går ud' af SAS), med nøgleordene angiver man de ønskede beregninger, og efter lighedstegnene skrives de nye variable, der skal indeholde den beregnede statistik. For MEAN står der ingen variabelnavne, her bruger SAS de oprindelige variabelnavne dvs. her HKG_HAog PCT_SUKK, denne mulighed kan dog kun benyttes for et af statistik nøgleordene, (ellers ville man få flere variable med samme navn). Der er to variable i VAR-ordren, derfor skal der også angives to variabelnavne for hver statistisk beregning; for SUM har vi dog her angivet, at vi kun vil have beregninger for HKG_HAved at skrive variabelnavnet i parentes, derfor er det nok med et variabelnavn.

Indtil nu har beregningerne givet et resultat beregnet på samtlige tal i datasættet. Ofte vil man dog gerne have resultater beregnet for undergrupper i datasættet, f.eks. i det foreliggende datasæt roer beregnet middelværdi spredning mv. for hver optagningstid og såtid. I CLASS-ordren angives de variable, hvis værdi man vil bruge til at opdele datasættet i undergrupper.

```
PROC MEANS DATA=saskurs.roer;
    CLASS optagn;
    VAR kg pct_sukk;
    OUTPUT OUT=roersum
        MEAN =    STD = stdhkg stdsuk   SUM(kg) = sumhkg;
RUN;
PROC PRINT DATA=roersum; RUN;
```

I dette tilfælde får vi middelværdi, spredning og sum for hver af de to optagningstider, men også for det samlede datasæt, dvs ialt 2 typer af observationer. SAS-variablen _TYPE_ har værdien 0 i observationen med gennemsnit over hele datasættet og 1 ved gennemsnit for hver klassevariabels værdier.

Man kan have flere klassevariabler:

```
PROC MEANS DATA=saskurs.roer;
```

```
    CLASS optagn saatid;
    VAR kg pct_sukk;
    OUTPUT OUT=roersum
        MEAN =    STD = stdhkg stdsuk   SUM(kg) = sumhkg;
RUN;
PROC PRINT DATA=roersum; RUN;
```

Der vil være 4 'typer' resultater ved 2 klassevariable:

TYPE = 0 beregninger over hele datasættet,

TYPE = 1 beregninger for hver værdi af sidste klassevariabel – her f.eks. middelværdi af saatid 1 over alle optagningstider, af saatid 2 over alle optagningstider osv,

TYPE = 2 beregninger for hver værdi af næstsidste klassevariabel,

TYPE = 3 beregninger for hver kombination af værdier for de to sidste klassevariable – her middelværdi af saatid 1 optagningstid 1, middelværdi af saatid 2 optagningstid 1 osv.

Man kan dog begrænse antallet af 'typer' til netop de man er interesseret i, ved at benytte `TYPES`-ordren hvor man angiver præcis de kombinationer af de 2 eller flere variable man har listet efter `CLASS`-ordren.

```
PROC MEANS DATA=saskurs.roer;
    CLASS optagn saatid;
    VAR kg pct_sukk;
    TYPES optagn saatid;
    OUTPUT OUT=roersum
        MEAN =    STD = stdhkg stdsuk   SUM(kg) = sumhkg;
RUN;
PROC PRINT DATA=roersum; RUN;
```

Ønsker man kun den sidste type resultater, altså beregninger for hver kombination af værdier for klassevariable kan man i `PROC`-ordren angive optionen `NWAY` således:

```
PROC MEANS DATA=saskurs.roer NWAY;
    CLASS optagns aatid;
    VAR kg pct_sukk;
    OUTPUT OUT=roersum
        MEAN =    STD = stdhkg stdsuk   SUM(kg) = sumhkg;
RUN;

PROC PRINT DATA=roersum; RUN;
```

NWAY vil bevirke at kun observationer med den højeste værdi af _TYPE_ udskrives.
Dette sidste datasæt kan også skabes ved at bruge BY-ordren i stedet for CLASS-ordren.

```
PROC SORT DATA=saskurs.roer;
   BY optagn saatid;

PROC MEANS DATA=saskurs.roer;
   BY optagn saatid;
   VAR kg pct_sukk;
   OUTPUT OUT=roersum
        MEAN =    STD = stdhkg stdsuk   SUM(kg) = sumhkg;
RUN;

PROC PRINT DATA=roersum; RUN;
```

Bruger man BY-ordren skal datasættet dog sorteres først med PROC SORT.

3.2 ODS

Output Delivery System benyttes til at omdirigere og styre output fra SAS til forskellige andre typer dokumenter - end standard output fra SAS. Et eksempel på brugen af Output Delivery System (ODS) hvor outputtet fra sidste kald til PROC MEANS sendes til et rtf-dokument (der kan læses af Microsoft Word), er følgende:

```
ODS RTF FILE='c:\saskursus\meanstabel.rtf';
PROC MEANS DATA=saskurs.roer;
   BY optagn saatid;
   VAR kg pct_sukk;
   OUTPUT OUT=roersum
        MEAN =    STD = stdhkg stdsuk   SUM(kg) = sumhkg;
RUN;
ODS RTF close;
```

Der henvises i øvrigt til Kapitel 7 'ODS: Eksport af tekst og grafik til andre programmer' i 'Elementær indføring i SAS'.

3.3 Øvelser

Opgave 8 *Anvend datasættet ROER.*

1. *Beregn gennemsnit og standardafvigelse for hver for udbytte og sukkerprocent for hver kombination af SAATID og OPTTID, for hver af disse faktorer for sig, og for data som helhed.*

2. *Gem (til senere brug) disse resultater i et permanent datasæt.*

Opgave 9 *Gennemfør skærmsøvelse 7.3 i bogen 'Elementær indføring i SAS'.*

Kapitel 4

Byt om med PROC TRANSPOSE og få HJÆLP !

4.1 Proc Transpose

Har man brug for at bytte om på rækker og variable i sit datasæt kan man benytte sig af proceduren `PROC TRANSPOSE`. Antag vi har brug for at have en kolonne med alle informationer for hvert barn i sashelp.class datasættet:

```
PROC PRINT DATA=sashelp.class; RUN;
```

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
.					
.					
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

Ved at benytte følgende program kan vi få det ønskede resultat.

```
PROC TRANSPOSE DATA=sashelp.class OUT=transclass NAME=response;
   VAR age height weight;
   ID name;
RUN;
PROC PRINT; RUN;
```

Obs	response	Alfred	Alice	Barbara	Carol	Henry	James	Jane	Janet	Jeffrey
1	Age	14.0	13.0	13.0	14.0	14.0	12.0	12.0	15.0	13.0
2	Height	69.0	56.5	65.3	62.8	63.5	57.3	59.8	62.5	62.5
3	Weight	112.5	84.0	98.0	102.5	102.5	83.0	84.5	112.5	84.0

Obs	John	Joyce	Judy	Louise	Mary	Philip	Robert	Ronald	Thomas	William
1	12.0	11.0	14.0	12.0	15.0	16	12.0	15	11.0	15.0
2	59.0	51.3	64.3	56.3	66.5	72	64.8	67	57.5	66.5
3	99.5	50.5	90.0	77.0	112.0	150	128.0	133	85.0	112.0

Ønsker vi derimod at have, værdierne for højde og vægt i samme variabel, da kan vi benytte følgende program:

```
PROC TRANSPOSE DATA=sashelp.class OUT=transclass2
               NAME=response PREFIX=size;
     VAR height weight;
     BY name age;
RUN;
PROC PRINT; RUN;
```

hvorved vi får:

Obs	Name	Age	response	size1
1	Alfred	14	Height	69.0
2	Alfred	14	Weight	112.5
3	Alice	13	Height	56.5
4	Alice	13	Weight	84.0
5	Barbara	13	Height	65.3
6	Barbara	13	Weight	98.0
7	Carol	14	Height	62.8
8	Carol	14	Weight	102.5
.				
.				
33	Ronald	15	Height	67.0
34	Ronald	15	Weight	133.0
35	Thomas	11	Height	57.5
36	Thomas	11	Weight	85.0
37	William	15	Height	66.5
38	William	15	Weight	112.0

4.2 Få hjælp til SAS

SAS/HELP findes på SAS's hjemmeside: http://v8doc.sas.com/sashtml, for at benytte hjælpefunktionen skal man tilmeldes som bruger, hvorefter man får adgang med login 'onlinedoc' og adgangskode 'sas'.

Der henvises derudover til afsnit 2.4 i 'Elementær indføring i SAS'.

4.3 Øvelser

Opgave 10 *Find dokumentationen til* `PROC TRANSPOSE` *på internet versionen af SAS/HELP, og gennemgå eksempel 1, 2 og 4 som øvelse.*

Kapitel 5

Datahåndtering

Man har ofte behov for at foretage omregninger af data. Dette kan omfatte dannelse af nye variabler eller modifikation af allerede eksisterende variabler. Sådanne beregninger sker altid i et datatrin. Beregningerne kan ofte med fordel laves direkte i forbindelse med indlæsning af data.

Hvis vi gerne vil have parcelnummeret sat på, have beregnet rod- og sukkerudbytte i hkg/ha, kan vi danne tre nye variabler, som indeholder disse værdier således (parcellerne er indtastet i rækkefølge og er hver på 25 m^2). Undertiden har man behov for at transformere data, før de analyseres statistisk. Sådanne transformationer skal i så fald foretages i datatrinnet. Eksempelvis vil vi tage 10-tals logaritmen af sukkerprocenten og kalde den transformerede variable for `LPCT_S`. Dette kan gøres på følgende måde:

```
TITLE 'BEREGNINGER OG DANNELSE AF NYE VARIABLER';

DATA NYROER;
  SET ROER;                                  /* 1 */
  PARC    = _N_;                             /* 2 */
  HKG_HA  = KG/25/100*10000;                 /* 3 */
  SUKK_HA = HKG_HA*PCT_SUKK/100;
  LPCT_S  = LOG10( PCT_SUKK );
RUN;
```

/* 1 */: Her specificeres at man vil danne datasættet `NYEROER`.

/* 2 */: Med `SET` kommandoen fortælles at `NYROER` skal dannes ud fra det allerede eksisterende datasæt `ROER`.

/* 3 */: Variablen PARC i sættes lig den specielle variabel `_N_`, der er en intern variabel i et datatrin, der angiver hvilken række SAS er nået til at behandle i datatrinnet.

Det nye datasæt `NYROER` vil herefter indeholde fire søjler mere end det oprindelige datasæt `ROER`, nemlig søjlerne `PARC`, `HKG_HA`, `SUKK_HA` og `LPCT_S`.

Ofte vil man i stedet for at lave et nyt datasæt blot modificere et allerede eksisterende. Man kan gøre dette med:

```
DATA ROER; SET ROER;                    /* 1 */
  PARC    = _N_;                        /* 2 */
  HKG_HA  = KG/25/100*10000;            /* 3 */
  SUKK_HA = HKG_HA*PCT_SUKK/100;
  LPCT_S  = LOG10( PCT_SUKK );
RUN;
```

Bemærk: Den naturlige logaritme hedder blot LOG.

5.1 Beregninger samtidig med indlæsning

Som nævnt tidligere er udbyttet angivet i Kilogram * 10 og sukkerindholdet i sukkerprocent * 10 i tekstfilen ROER.txt. En alternativ måde at få de to variabler lavet om til de rette enheder på er ved at dele de indlæste værdier med 10. Dette kan gøres i et ny data trin som illustreret ovenfor eller direkte i forbindelse med indlæsningen af data som det er illustreret nedenfor:

```
TITLE 'BEREGNINGER SAMTIDIG MED INDLÆSNING';

DATA ROER;
  INFILE 'C:\SASKURSUS\RAADATA\ROER.TXT' FIRSTOBS=6;
  INPUT  OPTAGN $ BLOK $ SAATID $ KG   PCT_SUKK;
  KG       = KG / 10;                          /* 1 */
  PCT_SUKK = PCT_SUKK / 10;
RUN;
```

/* 1 */: I denne sætninger skal lighedstegnet (=) læses "sættes lig med", så der står at variablen `KG` (på venstre side) "sættes lig med"variablen `KG` (på højre side) divideret med 10. (Det er vigtigt at læse lighedstegnet på denne måde. For ellers kan man jo ende med en ligning hvor der der står $X = X + 2$, hvilket jo er noget vrøvl!)

5.2 Betingelser, løkker, sammensætning og udeladelse af data

I behandlingen af ovenstående emner vil vi gøre brug af data fra et fodringsforsøg med svin (se afsnit 9.2.2).

5.3 Betingelser

I mange tilfælde har man behov for at lade de beregninger, der udføres, være afhængige af andre værdier. Vi er interesseret i at beregne det gennemsnitlige daglige foderindtag for

hvert dyr, nemlig

$$\frac{kgweek2 + \ldots + kgweek12}{11 \times 7}$$

Dette kan gøres som følger:

```
DATA EK; SET SASKURS.EVITCU;
   ADF=SUM(OF KGWEEK2-KGWEEK12)/(11*7);

PROC PRINT DATA=EK;
   VAR ADF LITTER PIG CU EVIT;
RUN;
```

Vi ser en anvendelse af funktion `SUM(OF ...)` der er hensigtsmæssig når man skal summere variabler, der er navngivet systematisk, sådan som tilfældet er ovenfor.

Det viser sig at på nogle dyr er foderindtaget i uge 12 ikke registreret. Derfor må man modificere beregningerne af det gennemsnitlige foderindtag. Dette kan gøres som følger:

```
DATA EK; SET SASKURS.EVITCU;
    ADFTMP=SUM(OF KGWEEK2-KGWEEK12)/7;
    IF KGWEEK12=. THEN
        ADF=ADFTMP/10;
    ELSE
        ADF=ADFTMP/11;
RUN;
```

Vi ser her en anvendelse af `IF`–konstruktionen

"`IF` betingelse `THEN` gør sådan; `ELSE` gør sådan;"

Det kan være hensigtsmæssigt at indikere denne slags specielle forhold i sit datasæt. Dette kan f.eks. gøres som følger:

```
DATA EK; SET SASKURS.EVITCU;
   ADFTMP=SUM(OF KGWEEK2-KGWEEK12)/7;
   IF KGWEEK12=. THEN
   DO
       ADF=ADFTMP/10;
       NOTE = 'WEEK 12 MISSING';
   END;
   ELSE
       ADF=ADFTMP/11;
RUN;
```

Har man brug for at få udført mere end een sætning efter THEN eller ELSE, må man sætte en slags "parentes" rundt om disse sætninger. Denne "parentes" er netop `DO ....END;`-konstruktionen.

Betingelser benyttes også ofte ved data kontrol. Vi kan således benytte `IF`-sætningen til at danne et datasæt, som indeholder de observationer, der ikke opfylder visse betingelser.

Vi ønsker at lave to nye datasæt, et med normale data og et med observationer, der eventuelt kræver nærmere undersøgelser. Det drejer sig om observationer,

- hvor der er lavet en bemærkning om at uge 12 er manglende, og
- hvor det gennemsnitlige daglige foderforbug er større end 2 kg.

```
DATA gode suspekte; SET ek;
   IF note NE '' OR adf>2 THEN
      OUTPUT suspekte;
   ELSE
      output gode;
RUN;
```

Bemærk her brugen af `OR` i en `IF`–sætning.

5.4 Ens behandling af flere variabler

I eksemplet ovenfor kunne det være relevant at udregne, hvor meget grisene vokser uge for uge. En brute force måde at gøre dette på er

```
DATA EK; SET SASKURS.EVITCU;
   GAIN2  = WEIGHT2-WEIGHT1;
   GAIN3  = WEIGHT3-WEIGHT2;
   /* og så videre ...*/
RUN;
```

Denne metode er ouverkommeligt hvis der er mange variabler, og risikoen for at man kommer til at lave fejl er for stor. En mere elegant og fejlsikker måde er følgende:

```
DATA EK; SET SASKURS.EVITCU;
   ARRAY A1{*} WEIGHT1-WEIGHT12;
   ARRAY A2{*} GAIN1-GAIN12;
   DO I = 2 TO 12;
      A2{I} = A1{I}-A1{I-1};
   END;
   FORMAT GAIN1-GAIN12 3.1;
RUN;
```

5.5 Beregninger på tværs af observationer

Normal køres datatrinnet igennem for hver observation (dvs. hver række) i ind-datasættet. Hvis man laver nye variabler i datatrinnet nulstilles (eller rettere "missing" stilles) disse hver gang datatrinnet gennemløbes for en ny observation. Dette kan undertrykkes ved et `RETAIN` statement. Derved beholder variablen den værdi den havde ved afslutningen af et datastep gennemløb til næste gennemløb. Man kan bruge dette til at udføre beregninger på tværs af observationer i et datasæt.

Betragt nedenstående datasæt, som illustrerer at man på nogle dyr har målt deres vægt uge for uge:

```
TITLE 'BEREGNINGER PÅ TVÆRS AF OBSERVATIONER';

DATA EKS;
   INPUT DYR UGE VAEGT;
   CARDS;
   1  1  12
   1  2  16
   1  3  19
   2  1  11
   2  2  16
   2  3  20
   3  1  12
   3  2  16
   3  3  21
;
```

Målet er at beregne den ugentlige tilvækst. Dette kan gøres med følgende program:

```
PROC SORT DATA=EKS;
   BY DYR UGE;

DATA EKS2; SET EKS;
   BY DYR ;
   RETAIN FORRIGE 0;
   IF FIRST.DYR THEN
      TVAEKST = .;
   ELSE
      TVAEKST = VAEGT - FORRIGE;
   FORRIGE = VAEGT;

PROC PRINT DATA=EKS2;   RUN;
```

Sætningen `RETAIN` bevirker at værdien af variablen `FORRIGE` ikke nulstilles når en række i datasættet er behandlet. Værdien af variablen `FORRIGE` fra den forrige række er således tilgængelig i den aktuelle række.

Det samme resultat kan opnås med brug af funktionen `LAG`, der returnerer værdien af en given variabel fra den foregående observation.

```
PROC SORT DATA=EKS;
   BY DYR UGE;

DATA EKS3; SET EKS;
   BY DYR ;
   LAGVAEGT = LAG(VAEGT);
   TVAEKST = VAEGT - LAGVAEGT;
   IF FIRST.DYR THEN
      TVAEKST = .;

PROC PRINT DATA=EKS3;   RUN;
```

5.6 Udeladelse af data

Da SAS-procedurerne regner på alle observationer i det SAS-datasæt, der benyttes, er det nødvendigt at fjerne de observationer, som af den ene eller den anden grund ikke ønskes benyttet i beregningerne. Der findes mange måder at gøre dette på, og vi skal her vise nogle af dem.

Vi ønsker kun at se på de data hvor E-vitamin og/eller kobber er på niveauerne 1 og 2. Nedenfor vises fire forskellige metoder at opnå dette resultat på:

```
TITLE 'UDELADELSE AF DATA';

/* METODE 1 */
DATA EK; SET SASKURS.EVITCU;
   IF EVIT=3 THEN DELETE;
   IF CU=3   THEN DELETE;
RUN;
PROC PRINT DATA=EK; RUN;

/* METODE 2 */
DATA EK; SET SASKURS.EVITCU;
   IF EVIT IN (1 2) AND CU IN (1 2)
   THEN
      OUTPUT;
```

```
   ELSE
      DELETE;
RUN;
PROC PRINT DATA=EK; RUN;

/* METODE 3 */
DATA EK; SET SASKURS.EVITCU;
   WHERE EVIT IN (1 2) AND CU IN (1 2);
RUN;
PROC PRINT DATA=EK; RUN;

/* METODE 4 */
DATA EK;
   SET SASKURS.EVITCU(WHERE=(EVIT IN (1 2) AND CU IN (1 2)));
PROC PRINT DATA=EK; RUN;
```

Vi ser at med IF–sætningen fjernes en eller flere rækker i datasættet. (Søjler kan fjernes med sætningen DROP.)

5.7 Sammensætning af data

Ofte vil man have behov for at sammensætte flere SAS-datasæt før beregning. Der er to principielt forskellige måder at sætte SAS-datasættet sammen på.

- Man kan sætte datasættene sammen, så det nye datasæt får lige så mange observationer, der er tilsammen i de datasæt, der benyttes
- Man kan flette datasæt sammen, så de observationer, der matcher, placeres i samme række i det nye datasæt.

I begge tilfælde bliver antal variable i det nye datasæt lig med antal forskellige variabler i de datasæt, der indgår. De to måder at sammensætte SAS-datasæt på udføres med henholdsvis `SET` og `MERGE` sætningerne.

Betragt følgende datasæt:

```
TITLE 'SAMMENSÆTNING AF DATA MED SET OG MERGE';

DATA B1;
   INPUT tekniker $ dyrid kuld tid y;
   CARDS;
   John 1 12 1 23
   Linda 1 12 2 34
```

```
   Linda 2 23 1 31
   John 2 23 2 55
 ;
DATA B2;
   INPUT DYRNR KULD DIET;
   CARDS;
   1 12 999
   2 23 777
   3 37 888
 ;
RUN;
```

Disse datasæt ønskes flettet sammen efter dyreidentifikationen. Denne er givet som henholdsvis DYRID og DYRNR i de to datasæt. Een mulighed er at omdøbe f.eks. DYRNR til DYRID i B2, og herefter sætte datasættene sammen i et nyt datatrin med `MERGE` kommandoen. Dette er dog ikke nødvendigt idet vi kan gøre følgende:

```
PROC SORT DATA=b1; BY dyrid;
PROC SORT DATA=b2; BY dyrnr;
DATA B12MERGE;
        MERGE b1(drop=tekniker)  b2(rename=(DYRNR=DYRID));
        BY DYRID;

PROC PRINT DATA=B12MERGE; RUN;
```

Resultatet bliver da følgende:

Obs	Dyrid	Kuld	Tid	Y	Diet
1	1	12	1	23	999
2	1	12	2	34	999
3	2	23	1	31	777
4	2	23	2	55	777
5	3	37	.	.	888

Tilsvarende kan de to datasæt sættes efter hinanden med `SET` kommandoen:

```
DATA B12SET;
   SET B1 B2;

PROC PRINT DATA=B12SET; RUN;
```

Dette giver følgende resultat:

Obs	Dyrid	Kuld	Tid	Y	Dyrnr	Diet
1	1	12	1	23	.	.
2	1	12	2	34	.	.
3	2	23	1	31	.	.
4	2	23	2	55	.	.
5	.	12	.	.	1	999
6	.	23	.	.	2	777
7	.	37	.	.	3	888

5.8 Øvelser

Opgave 11 *Tag udgangspunkt i datasættet* `EvitCu`*.*

1. *Lav et datasæt med de observationer, hvor der ikke er foretaget måling af vægt eller foderindtaget i uge 12. Gem dette som et permanent datasæt.*
2. *Lav et nyt datasæt, der kun indeholder observationer hvor* `Evit` *er lig med 3 og* `Cu` *er lig med 1 eller 3, og brug dette i resten af denne opgave. Prøv at lave dette datasæt ved hjælp af* `OUTPUT` *og* `DELETE` *og også ved hjælp af* `WHERE`*.*
3. *Sæt formater på variablerne således at de kun udskrives med eet decimal.*
4. *Beregn det ugentlige foderudnyttelse, givet som foderindtag divideret med tilvæksten, uge for uge for hvert dyr.*
5. *Beregn, uge for uge, forholdet mellem tilvæksten og foderindtaget.*
6. *Beregn den gennemsnitlige ugentlige tilvækst for hele forsøgsperioden.*
7. *Gem det nye datasæt du har lavet i kataloget* `C:\SASKURSUS\SASDATA\`*. Gem også SAS–programmet i* `C:\SASKURSUS\SASPROG\`*.*

Opgave 12 *Gennemfør skærmøvelse 4.5 i 'Elementær indføring i SAS', start fra side 68 midt.*

Kapitel 6

Grafik med SAS

6.1 PROC GPLOT

PROC GPLOT bruges til grafisk afbildning af en variabel mod en anden. De dannede grafer kommer i GRAPH-vinduet. Vi så i afsnit 6.1 at vi kunne lave en figur med følgende program:

```
TITLE 'Demo af PROC PLOT med BY-ordren';

PROC SORT DATA=sashelp.class OUT=a;
   BY sex;

PROC GPLOT DATA=A;
   PLOT height * age;
   BY sex;
RUN;
```

Her afbildes "højde"som funktion af "alder". Med PLOT-statement angives altså, hvilke variable, der ønskes plottet mod hinanden. GPLOT-ordren kan anvendes på 3 måder:

PLOT Y*X; Markerer punkter på grafen som A = 1 obs, B = 2 obs o.s.v.

PLOT Y*X='symbolnr'; Viser grafen med den symbolinstruktion der angives med SYMBOL instruktionen

PLOT Y*X=variabel; Viser grafen med forskellige symboler svarende til værdien af variablen, der står på højre side af lighedstegnet.

BY-ordren kan som det ses af eksempel også anvendes sammen med PROC GPLOT. Der bliver ved anvendelse af BY-ordren dannet et plot for hver værdi af BY-variablen. Som sædvanligt kræves datasættet sorteret efter BY-variablen.

Når Y*X='symbolnr' ordren anvendes er det ofte med henblik på at vise flere kurver i den samme graf, her skal man benytte SYMBOL instruktionen. Ved at bruge OVERLAY-optionen kan variablene afbildes i den samme graf.

```
TITLE 'DEMO AF PROC GPLOT MED BY-ORDREN OG OVERLAY';

PROC GPLOT DATA=sashelp.class;
   SYMBOL1 VALUE=plus COLOR=blue;
   SYMBOL2 VALUE=dot  COLOR=red;
   PLOT height * age=1 weight * age =2 / OVERLAY;
   BY sex;
RUN;
```

I dette tilfælde er det nok bedre at benytte `PLOT2` instruktionen:

```
TITLE 'DEMO AF PROC GPLOT MED BY-ORDREN OG OVERLAY';

PROC GPLOT DATA=sashelp.class;
   SYMBOL1 VALUE=plus COLOR=blue;
   SYMBOL2 VALUE=dot  COLOR=red;
   PLOT height * age=1 ;
   PLOT2 weight * age =2 ;
   BY sex;
RUN;
```

Hvis der ønskes en speciel akseinddeling kan der i `PLOT`-statement v.hj.a. haxis = og vaxis = angives, hvordan hhv. den horisontale og vertikale akse skal se ud. Dette sker ved hjælp af `AXIS` instruktionerne. Som i dette eksempel hvor der dannes en logaritmisk vertikal akse (dog er dette næppe relevant i dette tilfælde).

```
TITLE 'DEMO AF PROC GPLOT MED LOGARITMISK AKSE';

PROC GPLOT DATA=A;
   AXIS1 LOGBASE=10 LOGSTYLE=expand;
   PLOT height * age  / VAXIS = AXIS1;
RUN;
```

Instruktioner der gives med `AXIS` og `SYMBOL` er gældende indtil de bliver ændret, og altså således også på tværs af flere kald til for eksempel `PROC GPLOT`. For at nulstille alle disse instruktioner kan man køre følgende linie:

```
GOPTIONS RESET=ALL;
```

For kun at nulstille `SYMBOL` instruktionerne kan man istedet køre:

```
GOPTIONS RESET=SYMBOL;
```

For at få danske bogstaver til at fungere skal man benytte følgende instruktion.

```
GOPTIONS  KEYMAP=WINANSI DEVMAP=WINANSI;
```

Her et lidt mere udvidet eksempel på dannelsen af en graf:

```
PROC GPLOT DATA=a;
   SYMBOL1 I=none V=dot C=blue H=.5;
   SYMBOL2 I=none V=dot C=red  H=.5;
   AXIS1 ORDER=10 TO 16 BY 1 LABEL=('Alder');
   AXIS2 ORDER=50 TO 80 BY 10 LABEL=(ANGLE=90 H=1.5 'Højde (in)');
   PLOT height * age=sex/  HAXIS=AXIS1 VAXIS=AXIS2 NOLEGEND;
RUN;
QUIT;
```

Et andet lidt mere udvidet eksempel er baseret på et datasæt om højdeaf fyrretræer, hvor vi her ønsker en figur hvor symbolet er værdien af en variabel i datasættet:

```
GOPTIONS RESET=all;
TITLE  'TRÆHØJDE';
PROC GPLOT DATA=SASKURS.traer;
   SYMBOL POINTLABEL=("#trae" POSTITION=middle JUSTIFY=center)
          VALUE=none REPEAT=18;
   PLOT hoejde*diameter=trae;
RUN;
```

6.2 PROC GCHART

`PROC GCHART` bruges til at danne søjlediagrammer. Der kan dannes både liggende og stående søjler - i SAS kaldes de hhv. `HBAR` og `VBAR`. I `PROC GCHART` kaldes den akse, som søjlerne står på for midpointaksen, og den anden akse kaldes responseaksen (se nedestående figur).

Liggende søjler dannes med `HBAR`, og lodrette søjler dannes med `VBAR`. De options, der kan bruges for de to typer søjler, er præcis de samme. Denne gennemgang begrænses derfor til `VBAR`.

Med `VBAR` angives midpointvariablen, og med `SUMVAR` angives responsevariablen. Af options til `VBAR` skal vi her berøre `SUMVAR TYPE DISCRETE` og `GROUP`.

I `PROC GCHART` kan der beregnes gennemsnit, sum eller frekvens, som kan vises i et søjlediagram. Med `SUMVAR`=variabel angives den variabel, som man ønsker at lave søjler for. `TYPE`=type angiver, hvilken type søjler, der skal dannes. Hvis vi ønsker at lave gennemsnit for `SUMVAR`=variabel, så skal `TYPE` være lig med `MEAN`. Ønsker vi at vise summerede værdier, så skal `TYPE` være lig med `SUM`.

```
TITLE 'Lav et simpelt histogram';
```

```
PROC GCHART DATA=saskurs.roer;
  VBAR kg /TYPE=FREQ;
RUN;
```

`PROC GCHART` opfatter midpointvariablen som kontinuert, når intet andet er angivet. Proceduren laver så sin egen inddeling af midpointaksen. Når vi vil have en søjle for hver værdi af midpointvariablen (såtid), så skal optionen `DISCRETE` angives til `VBAR` statementet.

Man kan selv specificere klasseinddelingen i for histogrammerne, f.eks. med følgende:

```
PROC GCHART DATA=saskurs.roer;
  VBAR kg /TYPE=FREQ MIDPOINTS=100,110,120,130,140,150;
  VBAR kg /TYPE=FREQ LEVELS=8;
RUN;
```

Man kan danne histogrammer for gennemsnittene for hver såtid med følgende:

```
PROC GCHART DATA=saskurs.roer;
  VBAR saatid /SUMVAR=kg TYPE=MEAN ;
RUN;
```

Et `BY`-statement kan bruges sammen med `PROC GCHART`, men kræver en forudgående sortering af datasættet efter `BY`-variablen.

```
PROC SORT DATA=saskurs.roer;
  BY optagn;

PROC GCHART DATA=saskurs.roer;
  BY optagn;
  VBAR saatid /SUMVAR=kg TYPE=MEAN DISCRETE;
RUN;
```

Med `GROUP=`variabel kan datamaterialet inddeles i grupper, som vises side om side i søjlediagrammet. På midpointaksen markeres, hvilken gruppe søjlerne tilhører.

```
PROC SORT DATA=saskurs.roer;
  BY optagn;

PROC GCHART DATA=saskurs.roer;
  VBAR saatid / SUMVAR=kg TYPE=MEAN GROUP=optagn;
  VBAR saatid / SUMVAR=kg TYPE=MEAN GROUP=optagn SUBGROUP=blok;
RUN;
```

Med `BLOCK`-statement kan der dannes blokdiagrammer:

```
PROC SORT DATA=saskurs.roer;
  BY optagn;

PROC GCHART DATA=A;
  BLOCK saatid /SUMVAR=kg TYPE=MEAN GROUP=optagn;
  BLOCK saatid /SUMVAR=kg TYPE=MEAN GROUP=optagn SUBGROUP=blok;
RUN;
```

Lagkagediagrammer kan dannes med `PIE`-statement. Her er det vist, hvor stor en del kg for hvert saatid udgør af det samlede udbytte for alle 5 saatid.

```
TITLE 'Lav et lagkagediagram';

PROC SORT DATA=saskurs.roer;
  BY optagn;

PROC GCHART DATA=A;
  BY optagn;
  PIE saatid /SUMVAR=kg TYPE=MEAN;
RUN;
```

6.3 Øvelser

Opgave 13 *I denne opgave skal datasættet KEPA bruges.*

1. *Lav et plot med optagningstid ud ad x-aksen og udbytte ud ad y-aksen.*
2. *Lav dernæst et plot for hver blok med de samme variable, hvor symbolet, der vises i plottet er '+'. Akseinddelingen skal være den samme for hver blok.*
3. *Lav et plot, hvor udbytte plottes mod optagningstid med bloknummeret som symbol.*

Opgave 14 *Gennemfør skærmøvelse 2.5 i 'Elementær Indføring i SAS'.*

Opgave 15 *Anvend datasættet KEPA.*

1. *Lav et histogram med samlet udbytte op ad den vertikale akse og optagelsestid ud ad den horisontale akse.*
2. *Underopdel søjlerne, så det fremgår af søjlerne, hvormeget hver blok bidrager med.*
3. *Opdel søjlerne således, at data også grupperes efter bloknummer, men stadig optræder i en og samme graf.*

Kapitel 7

Statistiske analyser med SAS

7.1 Variansanalyse

Der findes flere procedurer, som kan fremstille variansanalyser. De mest benyttede er procedurerne `PROC ANOVA` og `PROC GLM` (der står for General Linear Model). Proceduren `PROC ANOVA` kan kun benyttes til beregning af variansanalyser i balancerede forsøg, mens proceduren `PROC GLM` desuden kan benyttes, når data er ubalanceret samt benyttes til beregning af regressionsanalyser og kovariansanalyser. Derfor er det i det følgende kun `PROC GLM` der omtales.

Det antages i udgangspunktet at observationerne er uafhængige, normalfordelt, alle med samme varians, og at alle effekter er systematiske (i modsætning til tilfældige).

Vigtige statements i `PROC GLM` er:

CLASS	specifikation af faktorer.
MODEL	specifikation af responsvariabel, model og options.
MEANS	specifikation af ønskede gennemsnitstabeller.
TEST	specifikation af tests, hvor nævneren ikke skal være ”ERROR”.
LSMEANS	specifikation af ønskede tabeller med ”korrigerede/marginale” gennemsnitsværdier.
OUTPUT	Specifikationer for beregning af bl.a. predikterede værdier og residualer.

7.1.1 Monofaktorielt blokforsøg

Lad os se på, hvorledes vi kan få beregnet en variansanalyse for udbyttet `HKG_HA` af sukkerroer for optagningstidspunktet op1 i datasættet ROER. Når vi kun ser på data fra optagningstidspunkt op1 er det rimeligt at benytte følgende basis-model:

$$Y_{bs} = \mu + \alpha_b + \beta_s + \epsilon_{bs}$$

hvor Y_{bs} er udbyttet i parcellen med såtid s i blok b, μ er niveauet, α_b er effekten af blok b, β_s er effekten af saatid s og ϵ_{bs} antages normalfordelt med middelværdien 0 og konstant

varians σ^2.

Vi benytter følgende program:

```
TITLE 'SUKKERROER';

DATA OP1DATA; SET MSAS.ROER;
   IF OPTAGN EQ 'op1' THEN OUTPUT;
RUN;

PROC GLM DATA=OP1DATA;
   CLASS BLOK SAATID;
   MODEL KG= BLOK SAATID ;
   MEANS SAATID;
RUN;
```

Dette program giver følgende udskrift:

```
                                SUKKERROER                                 1
                             The GLM Procedure
                          Class Level Information
                Class         Levels    Values
                BLOK               3    1 2 3
                SAATID             5    saa1 saa2 saa3 saa4 saa5

                         Number of observations    15
Dependent Variable: KG
                                      Sum of
  Source                     DF        Squares    Mean Square   F Value   Pr > F
  Model                       6    3176.566667     529.427778     62.90   <.0001
  Error                       8      67.333333       8.416667
  Corrected Total            14    3243.900000

                R-Square     Coeff Var      Root MSE       KG Mean
                0.979243      2.383853      2.901149      121.7000

  Source                     DF      Type I SS    Mean Square   F Value   Pr > F
  BLOK                        2       2.500000       1.250000      0.15   0.8643
  SAATID                      4    3174.066667     793.516667     94.28   <.0001

  Source                     DF    Type III SS    Mean Square   F Value   Pr > F
  BLOK                        2       2.500000       1.250000      0.15   0.8643
  SAATID                      4    3174.066667     793.516667     94.28   <.0001

                 Level of           --------------KG-------------
                 SAATID       N             Mean          Std Dev
                 saa1         3       136.333333       1.25830574
                 saa2         3       131.000000       3.00000000
                 saa3         3       128.000000       2.00000000
                 saa4         3       118.000000       2.50000000
                 saa5         3        95.166667       3.75277675
```

7.1.2 Split-plot forsøg

Hvis vi inkluderer data fra begge optagningstider i analysen får vi et split–plot. Vi benytter nu følgende basis-model,

$$Y_{bso} = \mu + \alpha_b + \gamma_o + U_{bo} + \beta_s + \delta_{os} + \epsilon_{bso}$$

hvor Y_{bso} er udbyttet i parcellen med såtid s og optagningstidspunkt o i blok b, μ er niveauet, α_b er effekten af blok b, γ_o er effekten af optagningstidspunktet o, β_s er effekten af saatid s, og δ_{os} er vekselvirkningen mellem optagnings– og såtidspunkt. Endelig antages $U_{bo} \sim N(0, \tau^2)$ og $\epsilon_{bso} \sim N(0, \sigma^2)$.[1]

Vi kan da benytte følgende program:

```
PROC GLM DATA=MSAS.ROER;
   CLASS OPTAGN BLOK SAATID;
   MODEL KG = BLOK OPTAGN BLOK*OPTAGN
                  SAATID SAATID*OPTAGN ;
   TEST H=OPTAGN E=BLOK*OPTAGN;
RUN;
```

Med statementet `TEST` specificeres det at testet af faktoren `OPTAGN` skal ske mod den tilfældige effekt `BLOK*OPTAGN`.

Ved at bruge et `RANDOM` statement med option `TEST` sørger `PROC GLM` selv for at testet for `OPTAGN` sker mod den relevante tilfældige effekt `BLOK*OPTAGN`:

```
PROC GLM DATA=MSAS.ROER;
    CLASS OPTAGN BLOK SAATID;
    MODEL KG = BLOK OPTAGN BLOK*OPTAGN
                  SAATID SAATID*OPTAGN ;
    RANDOM BLOK*OPTAGN / TEST;
RUN;
```

Vi ønsker desuden 3 tabeller med gennemsnitstal - en for hver af de to hovedvirkninger og en for vekselvirkningen. Dette fås ved at inkludere et `MEANS` statement i programmet

```
PROC GLM DATA=MSAS.ROER;
    CLASS OPTAGN BLOK SAATID;
    MODEL KG = BLOK OPTAGN BLOK*OPTAGN
                  SAATID SAATID*OPTAGN ;
```

[1] At man kan analysere et split–plot forsøg (der jo har to tilfældige effekter) i `PROC GLM` skyldes at de relevante teststørrelser m.v. hørende til den tilfældige helplot–effekt kan beregnes udfra visse kvadratsummer når forsøget er balanceret. Er forsøget ikke balanceret, laver `PROC GLM` visse approximationer. Disse er for nogle aspekter af analysen rimelige (desom forsøget kun er "moderat ubalanceret"), mens det for andre aspekter (herunder beregningen af variansen på visse kontraster) ikke er optimalt. I sådanne tilfælde kan proceduren `PROC MIXED` anvendes i stedet. Denne procedure omtales dog ikke nærmere her.

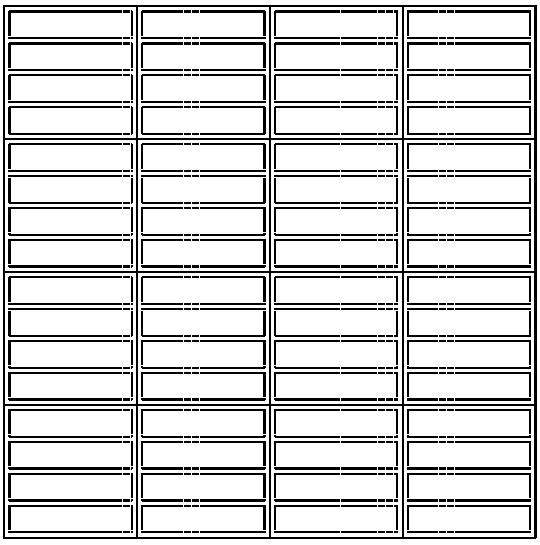
```
    RANDOM BLOK*OPTAGN / TEST;
    MEANS OPTAGN SAATID SAATID*OPTAGN;
RUN;
```

7.1.3 Ufuldstændigt blokforsøg

I forsøg med mange behandlinger (f.eks. sorter) bliver blokke, som indeholder alle behandlinger ofte så store, at parcellerne i en blok ikke kan fås tilstrækkelig ensartede. Man benytter da ofte at anlægge forsøget med ufuldstændige blokke, det vil sige at ikke alle behandlinger forekommer i alle blokke.

Vi vil her analysere et sortforsøg med 16 sukkerroesorter i 4 gentagelser. Inden for hver gentagelse var de 16 parceller inddelt i 4 blokke hver bestående af 4 parceller.

En skitse af forsøget er vist nedenfor. Kvadrater indenfor en gentagelse angiver blokke og stiplede rektangler indenfor hver blok angiver en parcel.



Ved analysen vil vi bruge følgende basis-model

$$Y_{rbs} = \mu + \alpha_r + \beta_{b(r)} + \gamma_s + \epsilon_{rbs}.$$

Her er Y_{rbs} udbyttet for sort s i blok b i gentagelse r, μ er niveauet, α_r er effekten af gentagelsen, $\beta_{b(r)}$ er effekten af den b'te ufulfstændige blok indenfor en gentagelse, α_s er sortseffekten og ϵ_{rbs} er residualen, og her antages der at $\epsilon_{rbs} \sim N(0, \sigma^2)$.

Vi vil analysere tørvægten af toppen. Data fra forsøget er placeret som de første 64 observationer i datasættet `TOERSTOF`. For at tage hensyn til at nogle sorter kan have ligget i blokke som er bedre/dårligere end gennemsnittet må vi benytte korrigerede gennemsnit i stedet for simple gennemsnit (LSMEANS i stedet for MEANS). De enkelte sorter ønskes

endvidere sammenlignet parvist med hensyn til om der er en signifikant forskel på deres tørstofindhold. Dette kan gøres at tilføje optionen `PDIFF` til `LSMEANS` statmentet.

Vi vil desuden lave lidt modelkontrol ved at plotte residualerne mod de predikterede værdier og parcelnummer.

Vi benytter da følgende program:

```
TITLE 'FORSØG MED SUKKERROER - ANALYSE AF TOPUDBYTTE (TØRSTOF)';

PROC GLM DATA=SASKURS.TOERSTOF(OBS=64);
   CLASS REP BLOK KENDENR;
   MODEL TOPT=REP BLOK(REP) KENDENR;
   LSMEANS KENDENR / STDERR PDIFF;
   OUTPUT OUT=B RESIDUAL=RTOPT PREDICTED=PTOPT STUDENT=STOPT;
RUN;

PROC GPLOT DATA=B;
   PLOT RTOPT*(PTOPT PARC)/VREF=0;
   PLOT STOPT*(PTOPT PARC)/VREF=0 1.96 -1.96;
RUN;
```

7.2 Regressionsanalyse

Der findes en række procedurer der kan udføre regressionsanalyser, heriblandt GLM og REG, og det er kun disse, der vil blive beskrevet det følgende.

Disse procedurer forudsætter, som nævnt i forrige afsnit, at observationerne er normalfordelte, uafhængige, alle med samme varians og at alle effekter er systematiske.

7.2.1 Simpel regression

Vi har indsamlet samhørende værdier af højde og diametermålinger på 18 fyrretræer.

Vi starter med at plotte højden mod diameteren (data ligger i datasættet TRAER) med programmet fra afsnit 6.1.

```
GOPTIONS RESET=all;
TITLE  'TRÆHØJDE';
PROC GPLOT DATA=saskurs.traer;
   SYMBOL POINTLABEL=("#trae" POSTITION=middle JUSTIFY=center)
          VALUE=none REPEAT=18;
   PLOT hoejde*diameter=trae;
RUN;
```

På baggrund heraf antager vi, at højden tilnærmelsesvist kan beskrives ved hjælp af diameteren og vil gerne estimere parametrene i den rette linie som "bedst"beskriver højden.

Vi derfor antager følgende model:

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

hvor Y_i er højden af træ i, x_i er diameteren for træ i ϵ_i antages normalfordelt med middelværdien nul og varians σ^2 og β_0 og β_1 er parametre.

Vi vil altså gerne beregne estimater for β_0, β_1 og σ^2. Samtidig vil vi beregne spredningen på estimaterne for β_0 og β_1 samt teste om disse kan være nul.

Vi benytter følgende program:

```
PROC GLM DATA=SASKURS.TRAER;
   MODEL HOEJDE=DIAMETER;
   OUTPUT OUT=B RESIDUAL=RHOEJ PREDICTED=PHOEJ STUDENT=SHOEJ;
RUN;
```

hvilket giver følgende udskrift:

```
                                TRÆHØJDE                                  8
                              The GLM Procedure
                        Number of observations    18

                              The GLM Procedure
Dependent Variable: HOEJDE   Højde
                                        Sum of
  Source                     DF        Squares    Mean Square   F Value   Pr > F
  Model                       1    86.84964627    86.84964627    263.66   <.0001
  Error                      16     5.27035373     0.32939711
  Corrected Total            17    92.12000000

               R-Square     Coeff Var      Root MSE    HOEJDE Mean
               0.942788      2.893771      0.573931       19.83333

  Source                     DF      Type I SS    Mean Square   F Value   Pr > F
  DIAMETER                    1    86.84964627    86.84964627    263.66   <.0001

  Source                     DF    Type III SS    Mean Square   F Value   Pr > F
  DIAMETER                    1    86.84964627    86.84964627    263.66   <.0001

                                       Standard
        Parameter         Estimate           Error    t Value    Pr > |t|
        Intercept      13.22305821      0.42898240      30.82      <.0001
        DIAMETER        0.31229646      0.01923282      16.24      <.0001
```

Vi kan benytte følgende program til at lave modelkontrol;

```
GOPTIONS RESET=SYMBOL;
PROC GPLOT DATA=B;
   SYMBOL I=rq v=dot c=black;
   PLOT RHOEJ*PHOEJ/VREF=0 NOLEGEND;
   PLOT SHOEJ*PHOEJ/VREF=0 1.96 -1.96  NOLEGEND;
RUN;
```

7.2.2 Polynomial regression (multipel regression)

Ovenstående plot antyder, at der ikke var en retliniet sammenhæng mellem højden og diameteren i hele det undersøgte interval. Vi vil derfor udvide vores model, så den bedre kan beskrive sammenhængen, og vi antager nu i stedet følgende model:

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$$

hvor vi nu yderligere skal estimere parameteren β_2. Ud over spredningerne på estimaterne β_0, β_1 og β_2 vil vi nu også gerne have beregnet korrelationerne mellem disse estimater. Vi vil stadig benytte residualer til modelkontrol samt have fremstillet figurer, hvor vi visuelt kan vurdere rimeligheden af de estimerede parametre.

Vi benytter nu følgende program:

```
DATA TRAER2; SET SASKURS.TRAER;
   D2=DIAMETER**2;
RUN;

PROC GLM DATA=TRAER2;
   MODEL HOEJDE=DIAMETER D2;
   OUTPUT OUT=B RESIDUAL=RHOEJ PREDICTED=PHOEJ STUDENT=SHOEJ;
RUN;

PROC GPLOT DATA=B;
   PLOT RHOEJ*PHOEJ=TRAE/VREF=0 NOLEGEND;
   PLOT SHOEJ*PHOEJ=TRAE/VREF=0 1.96 -1.96  NOLEGEND;
RUN;
```

7.3 Modelkontrol - PROC UNIVARIATE

Til kontrol af, om den benyttede model er rimelig, benyttes ofte at plotte residualerne mod relevante variable f.eks. de predikterede værdier. For at få beregnet disse må vi tilføje et `OUTPUT`-statement til `PROC GLM` og kalde proceduren `PROC GPLOT` for at fremstille de ønskede plots.

`PROC GLM` kan danne dels de rå residualer (observeret - predikteret) og dels de "studentiserede" residualer, der er er rå residualer divideret med residual standardafvigelsen. De studentiserede residual skal derfor, såfremt modellen er rimelig tilnærmelsesvist have varians 1. Følgelig skal skal omtrent 95% af residualerne ligge indenfor intervallet [-1.96,1.96].

Tager vi udgangspunkt i analysen af udbytte af effekten af såtider og optagningstider på udbytte af roer, da bliver programmet:

```
PROC GLM DATA=MSAS.ROER;
```

```
   CLASS OPTAGN BLOK SAATID;
   MODEL KG = BLOK OPTAGN BLOK*OPTAGN
              SAATID SAATID*OPTAGN;
   TEST H=OPTAGN E=BLOK*OPTAGN;
   MEANS OPTAGN SAATID SAATID*OPTAGN;
   OUTPUT OUT=B RESIDUAL  = R_KG
                PREDICTED = P_KG
                STUDENT   = SR_KG;
RUN;

PROC GPLOT DATA=B;
   PLOT R_KG  * P_KG / VREF=0;
   PLOT SR_KG * P_KG / VREF=0 1.96 -1.96;
RUN;
```

Med disse plots kan vi undersøge om variationen er konstant uanset værdien af det forventede niveau. Vi kan yderligere undersøge om det er rimeligt at antage at residualer er udfald fra en normalfordeling ved at benytte proceduren `PROC UNIVARIATE`. `PROC UNIVARIATE` kan benyttes til at undersøge fordelingen af en given numerisk variabel. Et visuelt test for normalitet er at undersøge et såkaldt Q-Q-plot, eller med et histogram med et indlejret figur af den teoretiske fordeling. Følgende program danner disse figurer:

```
PROC UNIVARIATE DATA=B NOPRINT;
   HISTOGRAM SR_KG / NORMAL(MU=EST SIGMA=EST);
   QQPLOT SR_KG / NORMAL(MU=EST SIGMA=EST);
RUN;
```

Det skal dog bemærkes at man i nuværende version af SAS (version 8.2) får forkerte resultater med `PROC UNIVARIATE` hvis der er manglende observationer i datasættet. Disse observationer skal derfor fjernes fra datasættet inden `PROC UNIVARIATE` benyttes, som f.eks.

```
DATA B; SET B;
WHERE SR_KG NE .;

PROC UNIVARIATE DATA=B NOPRINT;
   HISTOGRAM SR_KG / NORMAL(MU=EST SIGMA=EST);
   QQPLOT SR_KG / NORMAL(MU=EST SIGMA=EST);
RUN;
```

7.4 Øvelser

7.4.1 Øvelser til Variansanalyse

Opgave 16 *1. Lav en variansanalyse på sukkerindholdet i ROER. Beregn også tabeller med gennemsnitstal.*

2. *Endvidere ønskes gennemført modelkontrol ved at residualer og predikterede værdier plottes mod hinanden.*

3. *Lav en regressionsanalyse, der belyser om der er sammenhæng mellem sukkerindholdet og udbyttet.*

4. *Lav et plot af sukkerindholdet mod udbyttet (brug evt.* `GPLOT` *til dette formål). Hvad kan man heraf konkludere om at lave en regressionsanalyse inden man har undersøgt data grafisk?*

5. *Gennemfør en modelkontrol afregressionsanalysen.*

Opgave 17 *1. Datasættet ROERSORT indeholder resultatet af fra et sortsforsøg med sukkerroer. Analyser friskvægten af roden (variablen RODF).*

2. *Gennemfør en passende modelkontrol.*

7.4.2 Øvelser til regressionsanalyse

Opgave 18 *Dan datasættet „A“ ved hjælp af følgende programstump:*

```
%LET BETAA=1;
DATA A;
  DO GRP=1 TO 2;
    DO TID=0.1 TO 2.0 BY 0.1;
      Y = &BETAA * GRP - &BETAA * GRP * TID + &BETAA * TID *TID
          + 0.5 * NORMAL(0);
      OUTPUT;
    END;
  END;
RUN;
```

Undersøg med `PROC GLM` *følgende hypotese for hver gruppe:* $Y_i = \alpha + \beta t_i$*, altså at* Y *som funktion af tiden* t *kan beskrives som en ret linie, med skæringspunkt og hældningskoefficient som afhænger af gruppen* i*. Lav også en detaljeret modelkontrol.*

Opgave 19 *Datasættet* `LAERK` *indeholder registreringer af højden samt N–, P–, K– og askeindhold af 26 japanske lærketræer.*

1. *Fremstil en figur, hvor højden plottes mod N–indholdet.*

2. *Vurdér om det er rimeligt at antage en lineær sammenhæng mellem højde og N–indhold.*

3. *Beregn en lineær regressionsanalyse efter modellen*

$$Y_i = \alpha + \beta x_i + \epsilon_i$$

hvor Y_i *er højden for observation* i*,* x_i *er N–indholdet for observation* i

4. *Foretag modelkontrol.*

5. *Plot også residualerne fra ovenstående model mod variablerne P–, K– og askeindhold.*

6. *Giver dette anledning til at modificere modellen? Hvis ja, så gør dette og lav en ny modelkontrol.*

Opgave 20 *Datasættet* `PIGCLASS` *indeholder registreringer af kødprocenten* `LMP` *på 344 slagtesvin, samt målinger af tykkelsen af kødlaget* `MXX`*, tykkelsen af fedtlaget* `FXX` *samt nogle fysiske karakteristika* `PXX`

1. *Gennemfør en multipel regression af* `LMP` *med de øvrige variabler som forlarende med proceduren* `PROC GLM`*.*

2. *Gennemfør en modelkontrol*

3. *Plot eventuelt residualerne mod nogle af de udeladte variabler.*

Kapitel 8

Tabellering af data

8.1 PROC TABULATE

`PROC TABULATE` er en (meget fleksibel) procedure til udskrivning af tabeller. I `PROC TABULATE` er der 3 vigtige statements.

`CLASS`-ordren bruges til diskrete identifikationsvariable, hvis værdier skal bruges som kolonne- og række-værdier,

`VAR`-ordren bruges til de målte variable, hvis værdier skal stå 'inde i' tabellen som funktion af CLASS variablerne.

`TABLE`-ordren hvori det specificeres hvilke tabeller, der skal laves.

Betragt igen data fra fordringsforsøget omtalt i afsnit 9.2.2. Vi ønsker at lave en tabel, der viser gennemsnit og standardafvigelse af `Weight12` og `kgweek12` for hver kombination af `EVIT` og `CU` og en anden tabel med antallet af svin for hver kombination af `EVIT` og `CU`:

```
LIBNAME KSAS 'C:\SASKURSUS\KURSUSSASDATA';

TITLE 'MY TWO FIRST TABLES';

PROC TABULATE DATA=KSAS.EVITCU FORMAT=6.2;
    CLASS EVIT CU;
    VAR WEIGHT12 KGWEEK12;
    TABLE EVIT , CU / BOX="MY FIRST TABLE - NUMBER OF OBSERVATIONS";
    TABLE EVIT * CU, (WEIGHT12 KGWEEK12) * (MEAN STD) /
        BOX="MY SECOND TABLE - MEANS AND STANDARD DEVIATIONS";
RUN;
```

Optionen `format=6.2` specificere at hver celle skal bestå af 6 pladser og at tallene skal vises med 2 decimaler.

Der kan være mange TABLE statements i et kald af PROC TABULATE. I det første TABLE statement betyder Evit, Cu at der skal laves en tabel med EVIT som rækker og CU som søjler. TABLE ordren indeholder ingen specifikation af hvad der skal være i tabellens celler og derfor vil tabellen blot indeholde antallet af observationer. Optionen box=... bruges til at angive et navn for tabellen.

I det næste TABLE statement betyder Evit * Cu (som står til venstre for kommaet (,)) at en tabel med en række for hver kombination af EVIT og CU skal laves. Efter kommaet (,) følger en specifikation af hvad der skal være i søjlerne, nemlig gennemsnit og standardafvigelse for Weight12 og kgweek12. Dette opnås ved at "anvende" funktionerne MEAN and STD på variablerne Weight12 og kgweek12.

Dernæst ønskes lavet en tabel for hver værdi af EVIT. Hver tabel skal indeholde gennemsnittet af Weight12 og kgweek12 for de forskellige værdier af CU:

```
TITLE 'CREATING SEPARATE TABLES';

PROC TABULATE DATA=KSAS.EVITCU FORMAT=6.2;
    CLASS EVIT CU;
    VAR WEIGHT12 KGWEEK12;
    TABLE EVIT , CU , (WEIGHT12 KGWEEK12) * MEAN / BOX=_PAGE_;
    TABLE EVIT , CU , (WEIGHT12 KGWEEK12) * MEAN / BOX=_PAGE_ CONDENSE;
RUN;
```

I det første TABLE statement er der 3 led adskilt at komma. Dermed laves en tabel for hvert niveau af EVIT. Hver sådan tabel vil have niveauerne af CU som rækker og gennemsnittet af Weight12 og kgweek12 som søjler. Optionen box=_PAGE_ bevirker at værdien af EVIT sættes i det øverste venstre hjørne af hver tabel.

Optionen condense i det næste TABLE statement bevirker at tabellerne printes ud i en mere kompakt form med flere tabeller på hver side.

8.1.1 Tilføjelse af gennemsnit m.v. til tabellen

Dernæst ønsker vi at lave en tabel med gennemsnit og standardafvigelse for hvert niveau af EVIT men også det totale gennemsnit og standardafvigelse:

```
TITLE 'CREATING TABLES WITH SUMMARY STATISTICS';

PROC TABULATE DATA=KSAS.EVITCU FORMAT=6.2;
    CLASS EVIT CU;
    VAR WEIGHT12 KGWEEK12;
    TABLE EVIT , (WEIGHT12 KGWEEK12) * (MEAN STD) ;
    TABLE EVIT ALL , (WEIGHT12 KGWEEK12) * (MEAN STD) ;
    TABLE EVIT ALL="AVERAGE" , (WEIGHT12 KGWEEK12) * (MEAN STD) ;
RUN;
```

Den første TABLE ordre laver tabellen med gennemsnit og standardafvigelse for hvert niveau af EVIT. Optionen ALL den næste TABLE ordre bevirker at funktionerne MEAN og STD anvendes på tværs af niveauerne for EVIT. I den 3. TABLE ordre er vist en måde at tilskrive navne til de beregnede størrelser på.

Den næste opgave er at lave en tabel som i tillæg til gennemsnit og standardafvigelse for hver kombination af EVIT og CU også indeholder disse størrelse beregnet på tværs af EVITog af CU, og af begge.

```
TITLE 'CREATING TWO-WAY TABLES WITH SUMMARY STATISTICS';

PROC TABULATE DATA=KSAS.EVITCU FORMAT=6.2;
    CLASS EVIT CU;
    VAR WEIGHT12 KGWEEK12;
    TABLE EVIT ALL="AVERAGE EVIT" ,
          (CU ALL="AVERAGE CU") * (WEIGHT12) * (MEAN STD) ;
    TABLE EVIT ALL="AVERAGE EVIT" ,
          (CU ALL="AVERAGE CU") * (WEIGHT12 KGWEEK12) * (MEAN STD) ;
RUN;
```

8.1.2 LABELS og KEYLABELS

I tillæg til at skrive labels i en TABLE ordre (som det er vist tidligere) kan lavels også specificeres direkte med LABEL og KEYLABEL ordrene:

```
TITLE 'CREATING TWO-WAY TABLES WITH SUMMARY STATISTICS AND LABELS';

PROC TABULATE DATA=KSAS.EVITCU FORMAT=6.2;
    CLASS EVIT CU;
    VAR WEIGHT12 KGWEEK12;
    TABLE EVIT ALL, (CU ALL) * (WEIGHT12) * (MEAN STD) ;
    LABEL EVIT = 'VITAMIN E' CU = 'COPPER';
    KEYLABEL    MEAN = 'AVERAGE'
                STD  = 'STD ERROR'
                ALL  = 'AVERAGE';
RUN;
```

8.1.3 Kontrol af cellebredde med FORMAT

PROC TABULATE anvender et standard format for tabellen med mindre andet specificeres. Standard er at cellerne har formatet 12.2, hvilket betyder at cellen består af 12 felter og tallene angives med 2 decimaler. Titelcellen til venstre vil som standard bruge 1/4 af linielængden

Ovenfor har vi set that celler kan gøres mindre ved brug af optionen `FORMAT`, f.eks. `FORMAT=6.2`. Bredden af titelcellen kan specificeres med optionen `RTS`:

```
TITLE 'CHANGING THE FORMATS OF THE TABLE';

PROC TABULATE DATA=KSAS.EVITCU FORMAT=7.2;
    CLASS EVIT CU;
    VAR WEIGHT12 KGWEEK12;
    TABLE EVIT ALL, (CU ALL) * (WEIGHT12) * (MEAN*FORMAT=6. STD) / RTS=12;
    LABEL EVIT = 'VITAMIN E' CU = 'COPPER';
    KEYLABEL    MEAN = 'AVERAGE'
                STD  = 'STD ERROR'
                ALL  = 'AVERAGE';
RUN;
```

`FORMAT` optionen i `PROC TABULATE` specificeres at alle felter skal være 7 pladser brede og have 2 decimaler. `FORMAT` optionen i `TABLE` ordren "overskriver" denne standard for gennemsnittet, idet dette udskrives uden decimaler. Bredden af titlecellen sættes til 12 felter med optionen `RTS=12`.

8.2 PROC FREQ

Her henvises til afsnit 6.1 i 'Elementær indføring i SAS'.

8.3 Øvelser

Opgave 21 *Brug datasættet ROER.*

1. *Lav en tabel, der indeholder informationen fra `PROC SUMMARY` i opgave 8. Det vil sige gennemsnit og standardafvigelse for alle kombinationer af faktorerne SAATID og OPTTID, for hver faktor for sig og for datamaterialet som helhed.*

2. *Sæt passende labels og formater på tabellen således at tabellen ser pæn ud.*

Opgave 22 *Gennemfør skærmøvelse 6.5 i 'Elementær indføring i SAS'.*

Kapitel 9

Andre emner

Dette kapitel omhandler mindre emner der ikke passede ind i den forudgående fremstilling.

9.1 Organisering af data i dette kursusmateriale

Det er altid hensigtsmæssigt at organisere sine data (og sine SAS programmer) på en struktureret måde. De data der arbejdes med i dette notesæt og i de tilhørende øvelser er organiseret på følgende måde:

`C:\SASKURSUS\RAADATA\` Dette katalog indeholder, som navnet antyder, rådata. Det kan være data gemt som Excel regneark og data gemt som en almindelig tekstfil.

`C:\SASKURSUS\SASDATA\` Dette katalog indeholder SAS datasæt der bruges som illustration og i opgaverne i kursusmaterialet. En oversigt over disse datasæt er givet i afsnit 9.2. Disse datasæt er gemt som skrivebeskyttede filer, således at man ikke fejlagtigt kommer til at slette dem. I dette katalog bør man gemme de SAS datasæt man selv laver gennem kursusforløbet.

`C:\SASKURSUS\SASPROG\` I dette katalog bør man gemme de SAS programmer man selv laver gennem kursusforløbet. Endvidere kan man også heri gemme outputtet fra SAS.

`C:\SASKURSUS\SASMACRO\` I dette katalog findes nogle SAS makroer (hjemmelavede SAS programmer), der omtales i dette notesæt

Bemærk: I alle eksempler hvor `LIBNAME` anvendes vil libname'et `saskurs` referere til `C:\SASKURSUS\SASDATA\`

Det vil sige, at for at kunne køre disse eksempler forudsættes det, at man een gang har kørt linien

```
LIBNAME saskurs 'C:\SASKURSUS\SASDATA';
```

Se i øvrigt afsnit 2.5 for mere information om `LIBNAME`

9.2 Oversigt over datamaterialet

9.2.1 ROER: Så- og optagningstider i sukkerroer

```
Forsøgsplan

Såtider:                1        4. april
                        2       12. april
                        3       21. april
                        4       29. april
                        5       18. maj

Optagningstider:        1        2. oktober
                        2       21. oktober

Parcelfordeling:
               Blok  1    Blok  2    Blok  3
             +---------------------------------+
     Parcel | 1 1 1 1 1  2 2 2 2 2  1 1 1 1 1 | Optagningstid
       1-15 | 3 4 5 2 1  3 2 4 5 1  5 2 3 4 1 | Såtid
             +---------------------------------|
     Parcel | 2 2 2 2 2  1 1 1 1 1  2 2 2 2 2 | Optagningstid
      16-30 | 2 1 5 4 3  4 1 3 2 5  1 4 3 2 5 | Såtid
             +---------------------------------+
Data:
                                          Såtid
                              1       2       3       4       5
   Blok   Optagningstid    kg roer / parcel
   1             1         136     131     128     118      95
   1             2         154     147     141     139     100
   2             1         135     134     130     116      92
   2             2         156     150     136     140     100
   3             1         138     128     126     120      99
   3             2         155     151     142     140     102
```

Data fra ovenstående forsøg findes i tekstfilen `ROER.txt` og i Excel filen `ROER.xls`, der begge ligger i kataloget `C:\SASKURSUS\RAADATA\`

Indholdet af filerne er gengivet nedenfor hvor søjlerne angiver

Optagningstid Blok Såtid Udbytte Sukkerindhold

Parcellerne er indtastet i rækkefølge og er hver på 25 m^2.

```
op1 1 saa3  1280   171
op1 1 saa4  1180   169
op1 1 saa5   950   166
op1 1 saa2  1310   170
op1 1 saa1  1365   170
op2 2 saa3  1365   170
op2 2 saa2  1500   170
op2 2 saa4  1400   167
op2 2 saa5   995   164
op2 2 saa1  1560   168
op1 3 saa5   990   166
op1 3 saa2  1280   169
op1 3 saa3  1260   171
op1 3 saa4  1205   168
op1 3 saa1  1375   169
op2 1 saa2  1470   170
op2 1 saa1  1535   168
op2 1 saa5  1000   165
op2 1 saa4  1390   167
```

```
op2 1 saa3  1410   170
op1 2 saa4  1155   168
op1 2 saa1  1350   169
op1 2 saa3  1300   170
op1 2 saa2  1340   170
op1 2 saa5   915   165
op2 3 saa1  1550   167
op2 3 saa4  1405   166
op2 3 saa3  1420   169
op2 3 saa2  1510   169
op2 3 saa5  1020   164
```

BEMÆRK: 1280 skal læses som 128.0 kg og 171 som 17.1 procent

9.2.2 EVITCU: Fodringsforsøg med svin

I et forsøg med svin er E-vitamin of kobber tilsat i forskellige mængder til svinenes foder (begge faktorer har 3 niveauer, benævnt 1,2 og 3 i datasættet (niveau 1 er kontrollen). Formålet er at undersøge om de to stoffer har nogen gavnlig virkning på svinenes tilvækst, foderforbrug og foderudnyttelse.

Hvert svin er vejet en gang om ugen i 12 uger og man har registreret det ugentlige foderindtag. Den første vejning er sket i den uge hvor svinet er sat ind i forsøget. Data findes i datasættet `EvitCu` i `C:\SASKURSUS\KURSUSSASDATA\`, hvor `weight1 ... weight12` angiver vægten mens `kgweek1 ... kgweek12` angiver foderindtaget.

9.2.3 Øvrige datasæt

CARCASS: Målinger af fedt og spæktykkelse, samt kødprocent for 344 slagtesvin.

KEPA: Forsøg med kepaløg

TOERSTOF: Sortsforsøg med sukkerroer 1980 og 1981. 16 sorter i ufuldstændigt blokforsøg. Kun toptørvægten er angivet.

TRAER: Samhørende værdier af højde og diameter for 18 fyrretræer.

ROERSORT: Som TOERSTOF men 13 sorter fra 1982, kun rodfriskvægt angivet.

LAERK: Samhørende analyser af N-P-K- og askeindhold samt højde for 26 træer af Japansk Lærk.

9.3 Options i SAS systemet

I SAS kan man kontrollere dele af outputtet i outputvinduet med `OPTIONS`. Et eksempel på brug af options er

```
options linesize=94 pagesize=58 pageno=1;
```

Options er IKKE en del af hverken data trin eller procedure trin. En options–line som den ovenfor skrives blot for sig selv og submittes.

En oversigt over nogle ofte brugte options er givet i Tabel 9.1.

Tabel 9.1: Nogle ofte brugte options.

Option	Betydning
`linesize, ls`	Definerer liniestørrelsen.
`pagesize, ps`	Definerer sidestørrelsen.
`date, nodate`	Definerer om der kommer dato på siden
`center, nocenter`	Definerer om udskriften centreres eller ej.
`pageno`	Sætter sidenummeret
`formdlim`	Definerer hvorledes et sideskift skal se ud. For eksempel `options formdlim='';` (som er standard) definerer at hver udskrift starter på en ny side. `options formdlim='-';` definerer at hver udskrift adskilles med linien `----------`

Et eksempel på brug af options er følgende:

```
options  ps=58 ls=94 formdlim='-' date pageno=1000 nocenter;
title1 'Forskelligt output med OPTIONS';
title2 'CLASS data fra sashelp';

proc print data=sashelp.class;
run;
```

9.4 Konvertering af mellem dataformater

For at konvertere en numerisk variabel til en tekstvariabel kan man benytte funktionen `PUT()`, som i følgende eksempel, hvor der også vises hvordan man konverterer et tal i en tekstvariabel til en numerisk variabel:

```
DATA A;
 STR = '37';
 NUM = 42;
 NUM_STR = PUT(NUM,$8.); * FRA TAL TIL TEKST;
 STR_NUM = 1.0 * STR;    * FRA TEKST TIL TAL;
RUN;
```

9.5 Andre Statistikprogrammer

Til visse opgaver kan man med fordel anvende andre statistiske programmer end SAS. Eet sådant skal omtales her, nemlig programmet der slet og ret hedder "R". I R kan man gennemføre mange af de samme beregninger som i SAS. Nogle ting er lettere at gøre i SAS, andre er lettest i R. Det kan derfor være en god ide at vænne sig til at arbejde med begge programmer. Blandt andet er det en del nemmere at lave avanceret grafik i R end i SAS.

Programmet R er gratis og kan hentes fra

```
http://www.R-project.org
```

9.6 Valg af navne

Det er en god idé at overveje hvilke navne man giver sine variabler i sine SAS datasæt. I eksemplet med sukkerroer er der valgt navne, der er sigende - også for udenforstående. Det er fristende at bruge korte navne, f.eks. kunne `SAATID` være kaldt for `S` mens `BLOK` kunne være kaldt for `B` o.s.v. Men hvis man senere vender tilbage til sit program eller hvis man vil diskutere sit program med en kollega, så kan det være svært at overskue betydningen af sådanne navne!