AARHUS
UNIVERSITET
BUSINESS AND SOCIAL SCIENCES

SAS Programming – Introduction

Analytics Group

See more at www.asb.dk/ag

Analytics Group

Agenda

- About SAS and pros/cons
- SAS 9.2 user interface
- Library references
- Proc contents / proc means
- Proc Print
- Proc sort
- Data steps
- Loops
- Importing csv. xls. And txt. Files to SAS
- Where can you learn more?

Practical information

- First a basic introduction to SAS
- Brief introduction to each topic.
- After each introduction – assignments for 10 minutes.
- Duration 3 hours – approximately

- Sign up for the SAS Institute - Student Resource Center

What is it?

- At first: SAS - Statictical Analysis System
  - Mainly a tool for statistical analysis of large amounts of data
- Now, it is just SAS and so much more than just statistics
- Used for business intelligence and business analytics within many different industries
  - Risk management
  - Forecasting and econometrics
  - Operations research
  - Supply chain management
  - IT management
  - Management accounting
  - ...so much more

Why learn SAS

- Almost every large company in Denmark (and the rest of the world) uses SAS for their data warehousing.
- Can generate standard reports and lists to be used in the daily, weekly or monthly analysis.
- Is very good at handling large amounts of data without using much space on servers.

Why not?

- Quite expensive
- Steep learning curve

SAS 9.2 User interface

- Three windows
  - Editor: F5
  - Output: F7
  - Log: F6
- Use CTRL + E to clear any of the three windows
- F3 to submit (run) your code
- Two tabs:
  - Results
  - Explorer

Data library

- SAS library references: Highway to data
- Default library references
  - SAS User
  - Work (temporary)
- User defined library references
  - The physical location on your computer/server where you want SAS to retrieve data sets from and store them in.

library reference

- In general

```
libname name 'path';
run;
```

- Examples:

```
libname asb ' C:\documents\school\class';
run;
```

```
libname mt 'M:\master thesis\datasets';
run;
```

The data set *Brown*

- Consists of seven variables
- Contains a total of 120 observations with monthly sales figures from five different regions from 1948 – 1957
- How?

Proc contents

- Proc contents can be used to get general information about the data set at hand

```
proc contents data=asb.brown;
run;
```

The CONTENTS Procedure

Data Set Name	ASB.BROWN	Observations	120
Member Type	DATA	Variables	7
Engine	V9	Indexes	0
Created	22. oktober 2007 mandag 16:39:53	Observation Length	56
Last Modified	22. oktober 2007 mandag 16:39:53	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label	Written by SAS		
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	4096
Number of Data Set Pages	3
First Data Page	1
Max Obs per Page	72
Obs in First Data Page	42
Number of Data Set Repairs	0
Filename	G:\SAS\Kursus\SAS PROG I\2009 - Forår\brown.sas7bdat
Release Created	9.0101M3
Host Created	XP_PRO

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
6	MONTH	Num	8
1	SALES1	Num	8
2	SALES2	Num	8
3	SALES3	Num	8
4	SALES4	Num	8
5	SALES5	Num	8
7	YEAR	Num	8

Sort Information

Sortedby	YEAR
Validated	YES
Character Set	ANSI

Proc Means

- The Means procedure computes descriptive statistics about one or more variables.

```
proc means data=asb.brown;
var sales1;
run;
```

Output

The SAS System 19:43 Wednesday, March 17, 2010 3

The MEANS Procedure

Analysis Variable : SALES1

N	Mean	Std Dev	Minimum	Maximum
120	100.7250000	23.4459731	35.0000000	167.0000000

Proc Means

- You can get confidence intervals for a variable by adding *cml* at the end of the Proc Means statement. The default level is 95%

```
proc means data=asb.brown clm;
var sales1;
run;
```

Output

The MEANS Procedure

Analysis Variable : SALES1

Lower 95% CL for Mean	Upper 95% CL for Mean
96.4869631	104.9630369

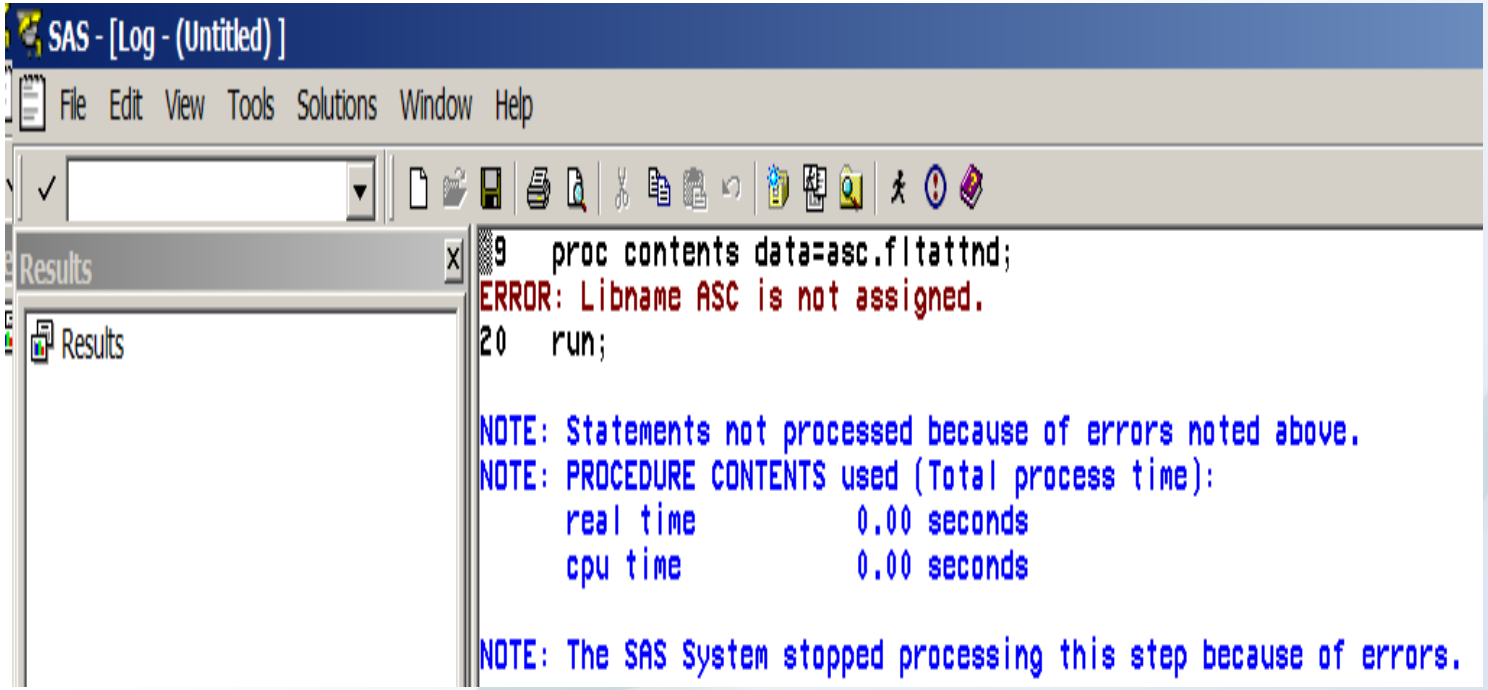
Debugging techniques

- The SAS log is very useful if problems occur when running your code.

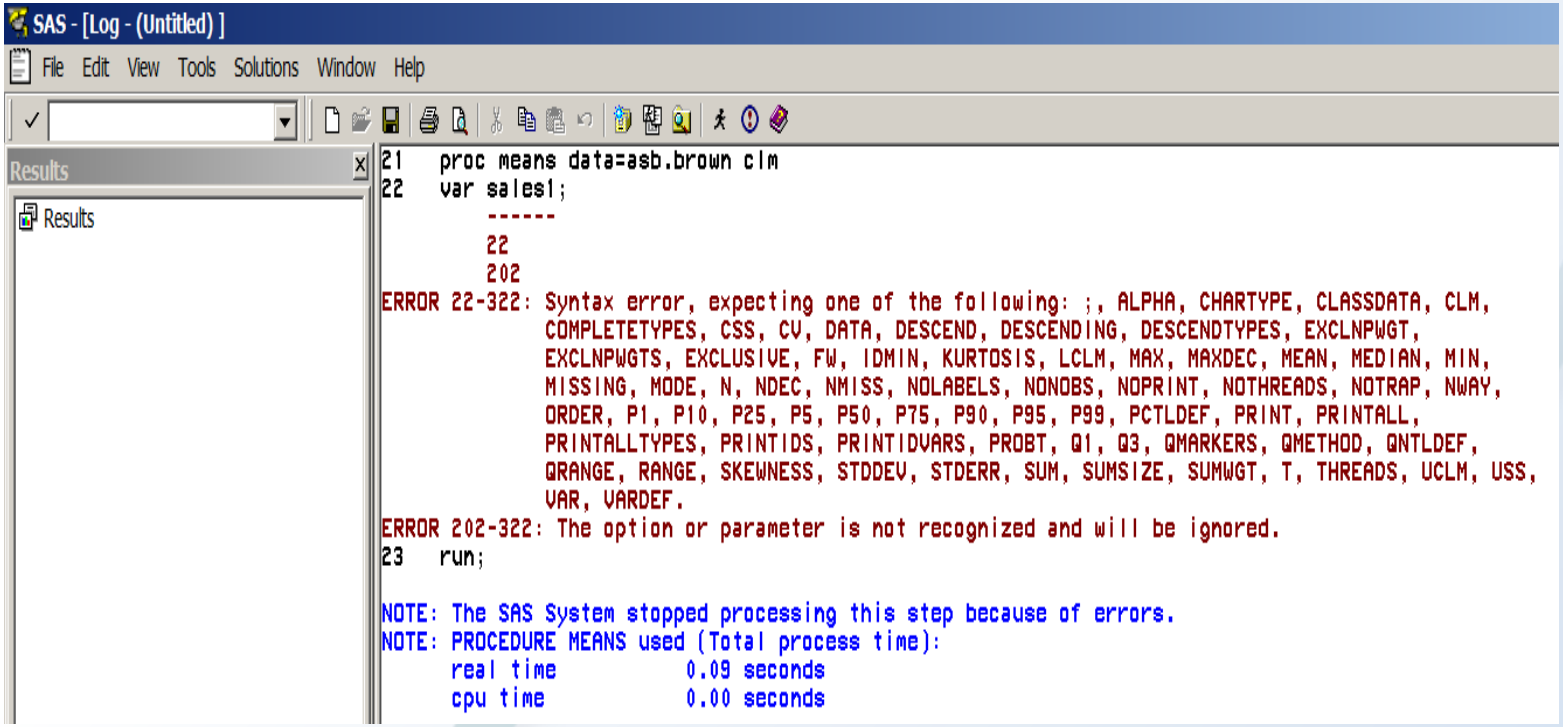
- You should ALWAYS check the log after submitting your code, and even before you check your output. Just because output has been generated does not mean your code has been submitted without errors. Also, your output results could actually be incorrect and If you do not check the log you might not notice incorrect results!

- Red = Bad
- Green = Warnings
- Blue = Good
- Black = The lines you submitted

Libname not assigned



Missing ; Very common mistake



Assignment 1

- Start by copying the data folder 'Programming' from \\okf-filesrv1\exemp\SAS*2010* to your M-drive

a) Assign a libname of your choice to this folder
b) Find out how many variables and how many observations the data set 'fltattnd' contains using SAS code
c) Find the mean, std. Deviation and confidence interval for the variable salary
d) Do the same as just above for the variable *jobcode* and figure out what the problem is
e) Do the same for the variable *hiredate*. What is the problem with the results?

Proc Print

- The Print procedure can be used to output basic information about a data set and the observations.

- Can control the following:
  - Excluding variables in the output
  - Sum together some of the variables
  - Page the output according to a variable
  - Include/exclude variables according to some conditions
  - And so on...

Proc Print examples

- Simple Proc Print statements
- In general

```
proc print data=libname.dataset;
<sas statements> …
run;
```

- An example

```
proc print data=asb.brown;
run;
```

Partial output

Obs	SALES1	SALES2	SALES3	SALES4	SALES5	MONTH	YEAR
1	103	105	105	191	29	1	1948
2	106	106	112	158	37	2	1948
3	105	109	75	184	59	3	1948
4	95	96	116	97	71	4	1948
5	123	95	154	124	95	5	1948
6	111	105	136	92	123	6	1948
7	78	101	108	99	127	7	1948
8	96	112	90	113	130	8	1948
9	110	106	80	100	106	9	1948
10	68	115	72	150	90	10	1948
11	115	99	106	228	49	11	1948
12	96	110	100	199	42	12	1948
13	84	112	92	182	37	1	1949
14	73	100	112	175	31	2	1949
15	110	103	137	155	76	3	1949
16	116	101	144	149	85	4	1949
17	87	85	112	84	91	5	1949
18	94	83	94	36	91	6	1949
19	107	65	68	129	65	7	1949
20	99	47	86	107	49	8	1949
21	114	49	109	135	24	9	1949
22	49	49	103	203	14	10	1949

Proc Print examples

- The *Var* statement can be used to control which variables you want to include in your output

```
proc print data=asb.brown;
var SALES1 MONTH YEAR;
run;
```

Partial output

Obs	SALES1	MONTH	YEAR
1	103	1	1948
2	106	2	1948
3	105	3	1948
4	95	4	1948
5	123	5	1948
6	111	6	1948
7	78	7	1948
8	96	8	1948
9	110	9	1948
10	68	10	1948
11	115	11	1948
12	96	12	1948
13	84	1	1949
14	73	2	1949
15	110	3	1949
16	116	4	1949
17	87	5	1949
18	94	6	1949
19	107	7	1949
20	99	8	1949
21	114	9	1949
22	49	10	1949

Proc Print examples

- The *Where* statement can be used to control which cases you want included in your output

```
proc print data=asb.brown;
var SALES1 MONTH YEAR;
where YEAR=1948;
run;
```

Partial output

Obs	SALES1	MONTH	YEAR
1	103	1	1948
2	106	2	1948
3	105	3	1948
4	95	4	1948
5	123	5	1948
6	111	6	1948
7	78	7	1948
8	96	8	1948
9	110	9	1948
10	68	10	1948
11	115	11	1948
12	96	12	1948

Proc Print examples

- You can use AND / OR / NOT to build up an expression

```
proc print data=asb.brown;
        var SALES1 YEAR MONTH;
        where YEAR=1948 and MONTH=1;
        run;
```

```
proc print data=asb.brown;
        var SALES1 YEAR MONTH;
        where YEAR=1948 or MONTH=1;
        run;
```

Outputs

- Where YEAR = 1948 and MONTH = 1:

Obs	SALES1	YEAR	MONTH
1	103	1948	1

- Where YEAR = 1948 or MONTH = 1:

Obs	SALES1	YEAR	MONTH
1	103	1948	1
2	106	1948	2
3	105	1948	3
4	95	1948	4
5	123	1948	5
6	111	1948	6
7	78	1948	7
8	96	1948	8
9	110	1948	9
10	168	1948	10
11	156	1948	11
12	196	1948	12
13	84	1949	1
25	84	1950	1
37	94	1951	1
49	125	1952	1
61	166	1953	1
73	105	1954	1
85	79	1955	1
97	52	1956	1
109	112	1957	1

Comparison operators

Mnemonic	Symbol	Definition
EQ	=	Equal to
NE	^= or ~=	Not equal to
GT	>	Greater than
LT	<	Less Than
GE	>=	Greater than or equal to
LE	<=	Less than or equal to
IN()		Equal to one of a list

Proc print examples

- You can use the IN() statement to select more than one condition

```
proc print data=asb.brown;
where MONTH in(1, 2, 3);
run;
```

- Outputs only month 1, 2, 3 (January, February, March)

- Conditioning on character cases in the IN() statement are case sensitive and needs to be embraced by '…'

Partial output

Obs	SALES1	YEAR	MONTH
1	103	1948	1
2	106	1948	2
3	105	1948	3
13	84	1949	1
14	73	1949	2
15	110	1949	3
25	84	1950	1
26	84	1950	2
27	109	1950	3
37	94	1951	1
38	78	1951	2
39	114	1951	3
49	125	1952	1
50	118	1952	2
51	115	1952	3
61	66	1953	1
62	92	1953	2
63	98	1953	3
73	105	1954	1
74	112	1954	2
75	127	1954	3
85	79	1955	1
86	42	1955	2
87	89	1955	3
97	52	1956	1
98	94	1956	2
99	48	1956	3
109	112	1957	1
110	133	1957	2
111	117	1957	3

Proc print examples

- You can sum a variable using the *Sum* function

```
proc print data=asb.brown;
var SALES1 MONTH YEAR;
where MONTH in(1, 2, 3,);
sum SALES1;
run;
```

Partial output

Obs	SALES1	YEAR	MONTH
1	103	1948	1
2	106	1948	2
3	105	1948	3
13	84	1949	1
14	73	1949	2
15	110	1949	3
25	84	1950	1
26	84	1950	2
27	109	1950	3
37	94	1951	1
38	78	1951	2
39	114	1951	3
49	125	1952	1
50	118	1952	2
51	115	1952	3
61	66	1953	1
62	92	1953	2
63	98	1953	3
73	105	1954	1
74	112	1954	2
75	127	1954	3
85	79	1955	1
86	42	1955	2
87	89	1955	3
97	52	1956	1
98	94	1956	2
99	48	1956	3
109	112	1957	1
110	133	1957	2
111	117	1957	3
	2868		

Proc Print examples

- A number of option can be added to the Proc Print statement. Here is a couple of examples
- 'noobs' suppresses the observation column in the output
- 'N' outputs the number of (non-missing) observation in the buttom of the output

```
proc print data=asb.brown noobs N;
Var SALES1 MONTH YEAR;
Run;
```

Partial output

SALES1	MONTH	YEAR
103	7	1956
82	8	1956
73	9	1956
123	10	1956
103	11	1956
59	12	1956
112	1	1957
133	2	1957
117	3	1957
112	4	1957
137	5	1957
139	6	1957
90	7	1957
84	8	1957
142	9	1957
112	10	1957
89	11	1957
143	12	1957

N = 120

Assignment 2

a) Open the data set *Weekrev* (same folder as the other data sets) to see the variables

b) Make a print of the data set where only the variables *FlightID*, *Origin*, *Date*, *CargoRev* and *PasRev* are included. Make sure there is no observation number

c) Alter the code so only flights from (origin) JFK and YYZ are in the output. Find the sum for *CargoRev* and *PasRev*.

d) Make a print of the data set *Sales121999* for only the route ids *0000002*, *0000024* and *0000083* if their EClassRev is above $70,000. The print should only contain FlightID, RouteID and EClassRev

Proc Sort

- The Proc Sort statement is used to sort a data set by a variable
- In general:

```
Proc sort data=input dataset
<out=output dataset>;
by <descending> by-variable(s);
run;
```

- Output data set is optional. Omitting it will alter the original data set.
- Ascending by default. Write Descending if that is what you want.

Proc Sort examples

```
proc sort data=asb.brown
out=asb.brown_sort;
by YEAR;
run;
```

Output

SALES4	SALES5	MONTH	YEAR
191	29	1	1948
158	37	2	1948
184	59	3	1948
97	71	4	1948
124	95	5	1948
92	123	6	1948
99	127	7	1948
113	130	8	1948
100	106	9	1948
150	90	10	1948
228	49	11	1948
199	42	12	1948
182	37	1	1949
175	31	2	1949
155	76	3	1949
149	85	4	1949
84	91	5	1949
36	91	6	1949

- Note: The Sort statement does not produce an output. You can check if the sorting was done by reading the Log, or by printing the sorted data set.

Proc Print examples

- You can arrange your output by sections or by pages. This sorts your output by some variable but does not sort and alter your data set.

```
proc print data=asb.brown;
by year;
pageby Year;
sum SALES1;
run;
```

- By: Creates section wise output for each year.
- Pageby: Creates page wise outputs for each year.
- You can use *By* without the *Pageby*, but not *Pageby* without *By*! The two variables (year) must be the same.

Output(s)

- By year:

---------- YEAR=1948 ----------

Obs	SALES1	SALES2	SALES3	SALES4	SALES5	MONTH
1	103	105	105	191	29	1
2	106	106	112	158	37	2
3	105	109	75	184	59	3
4	95	96	116	97	71	4
5	123	95	154	124	95	5
6	111	105	136	92	123	6
7	78	101	108	99	127	7
8	96	112	90	113	130	8
9	110	106	80	100	106	9
10	68	115	72	150	90	10
11	115	99	106	228	49	11
12	96	110	100	199	42	12
----	------					
YEAR	1206					

---------- YEAR=1949 ----------

Obs	SALES1	SALES2	SALES3	SALES4	SALES5	MONTH
13	84	112	92	182	37	1
14	73	100	112	175	31	2
15	110	103	137	155	76	3
16	116	101	144	149	85	4
17	87	85	112	84	91	5
18	94	83	94	36	91	6
19	107	65	68	129	65	7
20	99	47	86	107	49	8
21	114	49	109	135	24	9
22	49	49	103	203	14	10
23	75	64	100	200	0	11
24	113	58	104	193	0	12
----	------					
YEAR	1121					

---------- YEAR=1950 ----------

Obs	SALES1	SALES2	SALES3	SALES4	SALES5	MONTH
25	84	75	145	199	24	1
26	84	92	125	143	50	2
27	109	100	125	125	70	3

By- and Pageby Year:

---------- YEAR=1948 ----------

Obs	SALES1	SALES2	SALES3	SALES4	SALES5	MONTH
1	103	105	105	191	29	1
2	106	106	112	158	37	2
3	105	109	75	184	59	3
4	95	96	116	97	71	4
5	123	95	154	124	95	5
6	111	105	136	92	123	6
7	78	101	108	99	127	7
8	96	112	90	113	130	8
9	110	106	80	100	106	9
10	68	115	72	150	90	10
11	115	99	106	228	49	11
12	96	110	100	199	42	12
----	------					
YEAR	1206					

Assignment 3

a) Sort the data set *Sales121999* by *Destination* in descending order and make a print of it.

b) Sort the data set *Sales121999* by *FlightID* and make a print of it.

c) Make a page for each of the *FlightID's* where you sum *FClassRev* and *EClassRev* for each of the *FlightID's*. Make an output that only contains *FlightID*, *Destination*, *FClassRev* and *EClassRev*.

Data steps

- When you are working with variables, making calculations, creating new variables or new data sets, you are using data steps.

- A very general example

```
Data output_dataset;
set input_dataset;
additional sas statements;
run;
```

Data step examples

- Creating a new data set using an old one.

```
data asb.new_brown;
set asb.brown;
Keep SALES1 SALES2;
run;
```

- Instead of Keep you could use Drop.

Partial output

Obs	SALES1	SALES2
1	68	115
2	78	101
3	95	96
4	96	112
5	96	110
6	103	105
7	105	109
8	106	106
9	110	106
10	111	105
11	115	99
12	123	95
13	49	49
14	73	100
15	75	64
16	84	12
17	87	85
18	94	83
19	99	47

- Note: The datastep does not produce an output by itself. You have to either check the Log to check if your code was correct or make a print to see the actual output.

Data step examples

- Creating new variables by calculations

```
data asb.new_brown;
set asb.brown;
Total = SALES1 + SALES2;
keep SALES1 SALES2 Total;
run;
```

Partial output

Obs	SALES1	SALES2	Total
1	68	115	183
2	78	101	179
3	95	96	191
4	96	112	208
5	96	110	206
6	103	105	208
7	105	109	214
8	106	106	212
9	110	106	216
10	111	105	216
11	115	99	214
12	123	95	218
13	49	49	98
14	73	100	173
15	75	64	139
16	84	112	196
17	87	85	172
18	94	83	177

Assignment 4

a) Make a new data set called *Total_rev* in the *Work* library using the data set *Sales121999*. In the new data set, make a variable called *total* which adds the four revenue variables together (*FClassRev*, *BClassRev*, *EClassRev* and *CargoRev*). The new data set should only contain the variables *FlightID*, *RouteID* and the new variable *total*.

b) Make a print of the new data set.

c) Now make a new data set within your library called *new_total* that contains the variable *total* made before but only for the values between 10,000 – 20,000.

If statements

- If statements are quite similar to the ones in Excel.
- If statements can by used to manipulate with the output data in SAS. If statements are used within data steps.

- A general example

```
data out_dataset;
set input_dataset;
if expression then do;
executable statements;
end;
else do / else if;
executable statements;
end;
```

If statement example

- We only want information on the flights from Seattle

```
data seattle;
set asb.passngrs;
if Dest='SEA' then output;
keep FlightID Dest;
run;
```

Work library

Notice the ' ' surrounding SEA. These are necessary because its characters. Here SAS is case sensitive.

Output

Obs	Flight ID	Dest
1	IA01802	SEA
2	IA01804	SEA
3	IA01802	SEA
4	IA01804	SEA
5	IA01802	SEA
6	IA01804	SEA
7	IA01802	SEA
8	IA01804	SEA
9	IA01802	SEA
10	IA01804	SEA
11	IA01802	SEA
12	IA01804	SEA
13	IA01802	SEA
14	IA01804	SEA

If statement examples

- Deleting cases:
  - We do not want flights with less than 13 first class passengers

```
data new_passngrs;
set asb.passngrs;
if FClass lt 13 then delete;
run;
```

Output

Obs	Flight ID	Dest	Depart	FClass	BClass	EClass
1	IA02901	HNL	15101	13	24	138
2	IA03100	ANC	15101	13	22	150
3	IA03101	ANC	15101	14	.	133
4	IA02901	HNL	15102	14	25	132
5	IA03100	ANC	15102	16	26	143
6	IA03100	ANC	15103	14	18	137
7	IA02901	HNL	15104	13	22	150
8	IA03100	ANC	15104	14	17	144
9	IA03101	ANC	15104	13	.	142
10	IA02901	HNL	15105	13	14	145
11	IA03100	ANC	15105	15	22	99
12	IA02901	HNL	15106	13	24	137
13	IA03100	ANC	15106	15	16	137
14	IA02901	HNL	15107	13	19	144
15	IA03100	ANC	15107	15	23	105

If conditions

- If conditions can be used to control output and calculations under certain conditions.

- An example:
  - The flight company takes $1000 for a trip to HNL, $1500 for a trip to SEA and $2000 for a trip to ANC.
  - We want to assign the price for each trip in a new column using if-then-do statements.

If-Then-Do

```
data FClass_calc;
set asb.passngrs;
    if Dest='HNL' then do;
    Turnover=1000;
    end;
    else if Dest='SEA' then do;
    Turnover=1500;
    end;
    else if Dest='ANC' then do;
    Turnover=2000;
    end;
run;
```

Output

Obs	Flight ID	Dest	Turnover
1	IA01802	SEA	1500
2	IA01804	SEA	1500
3	IA02901	HNL	1000
4	IA03100	ANC	2000
5	IA03101	ANC	2000
6	IA01802	SEA	1500
7	IA01804	SEA	1500
8	IA02901	HNL	1000
9	IA03100	ANC	2000
10	IA01802	SEA	1500
11	IA01804	SEA	1500
12	IA02901	HNL	1000
13	IA03100	ANC	2000
14	IA01802	SEA	1500
15	IA01804	SEA	1500
16	IA02901	HNL	1000
17	IA03100	ANC	2000
18	IA03101	ANC	2000
19	IA01802	SEA	1500
20	IA01804	SEA	1500
21	IA02901	HNL	1000
22	IA03100	ANC	2000
23	IA01802	SEA	1500
24	IA01804	SEA	1500
25	IA02901	HNL	1000
26	IA03100	ANC	2000
27	IA01802	SEA	1500
28	IA01804	SEA	1500
29	IA02901	HNL	1000
30	IA03100	ANC	2000

Assignment 5

a) On the basis of the data set *pilots* create a new data set called *new_pilots.*The data set must only contain pilots who has the job code *PT2* and earn more than 84000 $

b) Make a print of *new_pilots* that includes the variables *IDNum* and *Salary*.

c) Make a new data set called *gendersal* which contains the two new variables *malesal* and *femsal*. The *malesal* variable must contain the salary for male employees and *femsal* must contain salary for female employees from the *pilots* data set.

d) Output the mean, std., min and max salary for the two genders in the new data set *gendersal*

Iterative data processing

- It is possible to get SAS to do calculations iteratively using loops.
- These calculations can for example be useful when making investment calculations.

Do loop

```
data <dataset name>;
    do i= 1 to 5;
    statement1;
    statement2;
end;
run;
```

When SAS reaches this point, it will loop as long as it is less than (or equal to) 5

Do loop example

Each year we deposit 50,000 dollars at an interest rate on 7.5%. If we start in 2000, how much do we have in 2005?

```
data invest;
        do Year=2000 to 2004;
        Capital + 50000;
        Capital + (Capital*0.075);
        end;
run;
```

Loops if Year is between 2000 and 2004

Output

Obs	year	capital
1	2005	312201.02

Do loop examples

- What if we want the sub calculations for each iteration?

```
data invest;
    do Year=2000 to 2004;
    Capital + 50000;
    Capital + (Capital*0.075);
    output;
    end;
run;
```

SAS outputs after each calculation.

Output

Obs	year	capital
1	2000	53750.00
2	2001	111531.25
3	2002	173646.09
4	2003	240419.55
5	2004	312201.02

Combining Do loops and If statements

- Do loops combined with If statements can be extremely useful. There is really nothing new to it. Just combine the statements in a data step.

Do While / Until

- If we want to know how many years we have to deposit $50,000 before we have $1,000,000 we use a do until loop.

Could be changed to:

do while (capital lt 1000000);

Adds one to the year in each loop.

```
data invest;
        do until (Capital ge 1000000);
        year + 1;
        Capital + 50000;
        Capital + (Capital*0.075);
        output;
        end;
run;
```

Output

Obs	capital	year
1	53750.00	1
2	111531.25	2
3	173646.09	3
4	240419.55	4
5	312201.02	5
6	389366.09	6
7	472318.55	7
8	561492.44	8
9	657354.37	9
10	760405.95	10
11	871186.40	11
12	990275.38	12
13	1118296.03	13

Assignment 6

a) A man buys a car today at a price of $30,000 and he borrows the money at an interest rate of 10%. He pays $5,000 upfront. How many years will he have to repay $5,000 before he has paid of his loan? Create a data set called calc where you do the calculations and sub calculations. The data set must contain the accumulated value of the loan.

b) Do the same calculations again but this time assume that after three years the interest rate falls to 7.5% and stays at that level for the rest of the loan period. How many years will he have to repay $5,000 before he has paid of his loan?

Importing data files

- It is possible to import data sets in other file formats by using the statement infile in a datastep (self study). Or by using the wizard under file → import data. It is possible to import xls. Csv. Txt. And many other file formats.

Where to learn more?

- Borrow *SAS books* at the library
- F1 – very useful. Place the cursor on a statement, frase or anything in your code, and press F1.
- Try to Google your question…
- Ask for help at analytics@asb.dk