Introduction to SAS

See SDA Chapters 1-3

LSB Chapters 1-5, 8

SAS is procedure-based

- R is a functional programming language
- SAS is more structured than R
  1. Data step
  2. Procedure step
- SAS has a macro language, but otherwise analysis is restricted to available procedures.

A simple SAS program

```
data first;
input income tax age state $;
datalines;
123546.75 03465 35 IA
234765.48 08956 45 IA
348578.65 05954 31 IA
345786.78 05765 41 NB
543567.51 12685 32 IA
;
run;
proc print;
title 'SAS Listing of Tax data';
run;
```

Rules and syntax

- Data are either numeric or character, e.g.
  - 71, .0038, –4., 8214.7221, 8.546E–2
  - MIG7, D’Arcy, 5678, South Dakota
- Other data (e.g. dates dd/mm/YYYY) can be stored using informats as numerics
- Data sets are rectangular with variables in columns
- Each variable has attributes associated with it:
  - Name, type, length, relative position, informat, format, label

SAS names

- Maximum length of 32 or 8.
- First character must be alphabetic
- Other characters can be alphabetic, numeric, or underscores
- Not case-sensitive
- Variables can also be specified using variable lists

  `var1-var6` is the same as

  `var1 var2 var3 var4 var5 var6`
- You can reference a sequence of variables, whether they are part of a list or not, using two dashes

  `var1--height`

SAS statements

- SAS keywords are reserved, often capitalized
  - `PROC UNIVARIATE;`
- SAS statements can begin and end in any column
- SAS statements must end with a semicolon
- More than one SAS statement can appear on a line, and one can stretch over multiple lines
- Items in SAS statements should be separated by blanks, except when they are connected by special symbols

SAS data sets

- Three steps to creating a SAS data set:
  1. The `data` statement (name the data set)
  2. Use `input`, `set`, `merge`, or `update` depending on the location of the information to be included in the data set
  3. (optional) Modify data before input using programming statements

The data step

- The first statement following the `data` statement is usually `input`.
  - The input statement defines the format of each data line
- List input: for data separated by blanks
- Formatted input: not separated by blanks
- Column input: read in by specifying columns

List input

- INPUT *variable_name_list*;
  - input age weight height;
  - input score1-score5;
- Character input
  - input name $ age height;

Formatted input

- Need to specify:
  - In which column the data value begins
    - `input @23 height @27 weight;`
  - How many columns to read
    - `input @23 height 4.;`
  - Whether the data value is numeric or character
    - `input @5 name $18. @23 height 4.;`
  - (optional) where a decimal point should be placed
    - `input @23 height 3.1;`
- `0001IA005040891349`
  - `input id 4. state $2. fert 5.2 percent 3.2 members 4.;`

Column input

- Similar to formatted input, but specify columns directly
- Blanks are ignored
- `0001IA 5.04 891349`
- `input id 1-4 state $ 5-6 fert 7-12 percent 13-15 .2 members 16-19;`

Data step programming

```
data sample;
input(x1-x7) (@5 3*5.1 4*6.2);
y1 = x1+x2**2;
y2 = abs(x3)
y3 = sqrt(x4+4.0*x5**2)-x6;
x7 = 3.14156*log(x7);
datalines;
...
;
```

Data step programming

- Conditional statements
  - `if score < 80 then weight=.67;`
    `else weight=.75;`
  - `weight=(score < 80) *.67`
    `+ (score >= 80) *.75;`
  - `if state= 'CA' | state= 'OR' then region='Pacific Coast';`
- An example using missing and delete
  - `if income= . then delete;`

Data step programming

- Conditional blocks

```
if score < 80 then do;
weight=.67;
rate=5.70;
end;
else do;
weight=.75;
rate=6.50;
end;
```

Procedure step

- `PROC procedure_name options_list;`
- If you are running a procedure and:
  - You are using the most recent data set
  - You are using all columns of the data set
  - You are using all rows of the data set

then you only need a simple PROC statement.

e.g. `proc print;`

Procedure step

- Specifying a data set
  - `proc print data=mydata;`
- Specifying a procedure option
  - `proc corr kendall;`
- Specifying a subset of the variables
  - `proc means data=store mean std;`
    `var bolts nuts screws;`
- Computing on subsets of the data
  - `proc print; by group;`

Formats and labels

- `format` can be used to specify a format used for printing in either a data step or proc step
  - `format expenses dollar10.2 ;`
- `label` can be used to specify a more descriptive name for a variable in either a data step or a proc step
  - ```
    label region='Sales Region'
    headcnt='Sales Personnel';
    ...
    proc print label;
    ```
    ```

Inserting comments and getting help

- You can get help from the help menu in SAS
- You can google for SAS documentation online
- Comments are inserted using

```
/* this is my
comment. SAS will
ignore it. */

* this is my comment.;
```

Running your code in SAS

- The `run;` statement
- Submitting code to run
- The log window
- The output window
- The libraries window
  - Libraries = ‘directories’ or ‘folders’
  - Defining with `libname mylib ‘path-to-lib’`
    - `mylib.mydataset`

Data statement: @ and @@

- Recall that the data statement has a hidden loop
- One line of input is read in, and the input statement tells SAS how to transform input stream into variables
- @ holds the loop so that an additional input statement can be executed
- @@ executes the input line repeatedly for multiple records on one line

Example: @

```
data mydata;
input category nrecs @;
do i=1 to nrecs;
  input value @;
  output;
end;
drop i nrecs;
datalines;
1 3 -2 2 7
2 1 8
3 6 -1 0 0 1 12 4
;
run;
```

Example: @@

```
data sat;
input name $ verbal math @@;
total= verbal + math;
datalines;
Sue 610 560 John 720 640 Mary 580 590
Jim 650 760 Bernard 690 670 Gary 570 680
Kathy 720 780 Sherry 640 720
;
run;
```

One data set from another

```
data athlete_2;
set athlete;
if abp >=100 & hr > 70;
run;
```

Reading data from a file: infile

```
data biology;
infile 'Lab2-data1.txt';
input id sex $ age year height
        weight;
run;
```

Modified list input, &, :, and ~

- When reading data using list input,
  - sometimes character variables will have a space (e.g. New York) – use & to specify. Two spaces will tell SAS that the end of the record has been reached.
  - sometimes input will require an informat, whether character or numeric (e.g. 2,014) – use : to specify.
  - sometimes you want to retain quotation marks and delimiters (e.g. "Green Hornets, Atlanta“) – use ~ to specify.

Modified list input, &, :, and ~

```
data lab2.world;
infile 'Lab2-data2.txt';
input country & $15. birthrat deathrat
      infmort lifeexp popurban
      percgnp : comma. levtech civillib @;
run;
```

```
NEW ZEALAND  16 8 13 74 83 7,410 66 1
```

Basic plotting in SAS

```
* Annotated scatterplot;
proc plot;
plot weight*height='*' $ sex;
run;

* Barplot of mean heights;
proc chart;
vbar sex/type=mean sumvar=height;
run;
```

Fancier plotting

```
proc sort data = insulin;
by week;
run;
proc boxplot data = insulin;
  plot insulin*week;
run;
```

Boxanno macro: 862 students only

- http://www.datavis.ca/books/sssg/boxanno.html