SAS language elementaries

Use of SAS
March 2009

SAS is a programming language

- A program is a “recipe”: A series of instructions to be executed in a specified sequence
- Notice: SAS is not a spreadsheet. Output is output, and does not change automatically if data are changed
- Some rules and conventions are necessary for SAS to be able to interpret its instructions
- Separators (semicolon, /)
- Parentheses and quote symbols must be matched

A simple SAS program

```
data new;                                     |
  set original;                               |
  age=1997-birthyr;                           |  Data Step
  bmi=weight/(height*height);                 |
run;                                          |



proc print data=new;                          |
  var id age bmi;                             |
run;                                          |
                                              |  Proc Steps
proc means data=new;                          |
  var age bmi;                                |
run;                                          |
```

DATA steps and PROC steps

- Roughly speaking, SAS programs consist of two kinds of *steps* (= blocks of instructions):
- DATA steps define datasets. E.g. by reading raw data, computing transformed variables, selecting cases, etc.
- PROC steps contain standard procedures that operate *on* datasets. You can't, e.g., transform variables in a PROC step.
- Normal arrangement of a SAS program is to put DATA steps at the beginning, but they can occur intermixed
- There are a a few SAS *statements* in addition to the DATA and PROC steps. They typically set up definitions for later use: LIBNAME, OPTIONS, AXIS, and SYMBOL statements are the most common ones

Basic things about the SAS language

- Almost everything starts with a keyword and ends with semicolon (exception being that there is no keyword before computations in a DATA step)

- **Statements** are pieces of code separated by semicolon

```
OPTIONS ls=80;
PROC GPLOT data=sasuser.fitness;
   PLOT maxpulse * age;
RUN;
PROC GLM data=sasuser.fitness;
   MODEL maxpulse = age / solution;
RUN;QUIT;
```

- Keywords: OPTIONS PROC PLOT MODEL RUN QUIT

- Some statements belong together in blocks (**steps**).

Things to notice

```
OPTIONS ls=80;
PROC GPLOT data=sasuser.fitness;
   PLOT maxpulse * age;
RUN;
PROC GLM data=sasuser.fitness;
   MODEL maxpulse = age / solution;
RUN;QUIT;
```

- The slash symbol (/) is often used to introduce options for a statement
- Separators like semicolons and slashes are necessary to avoid ambiguity: `solution` is not a variable name, `run` is not an option.
- SAS detects the end of a step when there is a new DATA or PROC statement (RUN is not always needed).

Formatting of code

- SAS generally doesn't care about whitespace and line breaks

  ```
  data
            work.cohort;
            set course.males98;
  run;
  ```

- is the same as

  ```
  data work.cohort; set course.males98; run;
  ```

- Good practice is to have at most one statement per line.

Indentation

- Enhances readability considerably. (You *will* have to read your own old code!)
- DATA and PROC steps are entered starting at the left edge. Likewise `OPTIONS` statements and `RUN` and `QUIT`.
- Any subordinate statements are indented by 2-4 blanks
- In statements which do not fit on one line, subsequent lines are also indented.
- This creates visual groups, so that you can easily see where one thing ends and the next begins.

Example of good indentation

```
data new;
  set original;
  age=1997-birthyr;
  bmi=weight/(height*height);
run;


proc print data=new;
  var id age bmi;
run;

proc means data=new;
  var age bmi;
run;
```

Ingredients of a DATA step

```
data new;
  set original;
  age=1997-birthyr;
  bmi=weight/(height*height);
run;
```

- Specification line (name of new data set)
- Data source (here: name of old SAS data set)
- Computation
- Assignment

Variables

- Columns of a dataset
- Can be numerical (usual case)
- – or character (text strings)
- Values of a character variable are given in quotes: `’male’` or `"male"`
- A dot (.) denotes a missing value for a numerical variable and is the lowest number in SAS.
- Calculations involving a missing will result in a missing (most of the times)
- A ”” or ” denotes a missing value for a character variable.

Names of variables

- SAS is case-insensitive (`SEX`, `sex`, `Sex` all refer to the same variable)
- Names can be up to 32 characters long (older SAS: max 8)
- Names can consist of (english) letters, numbers and underscore (`_`)
  - but can *not* start with a number

Comments

Two ways of making comments in SAS programs:

```
/* Comments */

 * Comments ;
```

Example:

```
/* Here I make a new data
with new variables age and bmi*/
data new;
  set original;
  age=1997-birthyr;
  bmi=weight/(height*height);
run;

*Here I print age and bmi;
proc print data=new;
```

```
  var id age bmi;
run;

*Here I calculate means;
proc means data=new;
  var age bmi;
run;
```