# STAT:5400 Computing in Statistics

# Introduction to SAS

Lecture 18
Oct. 12, 2015

Kate Cowles
374 SH, 335-0727
kate-cowles@uiowa.edu

## SAS

- SAS is the statistical software package most commonly used in business, government, and many other applied research settings
  - has the best data management capabilities
  - can handle huge datasets
  - not as user-friendly or well-designed as R or S-PLUS in some other ways
  - R or S-PLUS are often preferred for research in theoretical and methodological statistics
- *preparing* data for analysis usually takes more time and work than doing the actual analysis

## Two ways to run SAS

- batch mode
  - use text editor to write a file containing SAS code
  - extension ".sas"; e.g. filename might by "conadp.sas"
  - run it in batch mode by entering "sas <filename>"; e.g. "sas conadp"
  - SAS produces two new output files
    - log file with .log extension
    - output file with .lst extension
- interactive mode
  - just enter "sas"
  - program editor, log, and output are three windows
  - online help available

## How SAS programs and commands are organized

- Use a *DATA step* to organize your data by creating a SAS dataset.
- Then use *PROC steps* to analyze your data using SAS procedures.
  - Once you have created a SAS dataset, you may apply any SAS procedures to it during the current SAS session without recreating the dataset.
- DATA and PROC steps consist of SAS *statements*.
  - Each statement must end with a semicolon.
  - Most statements include one or more *keywords* that must be spelled exactly as shown.

## The DATA step: Creating a SAS dataset

Before it can process data, SAS must read in the data in the form of a table with

- a row for each *observation*
- a column for each *variable*

You must choose a name for the entire dataset and a name for each variable.

SAS distinguishes between two types of variables:

- *numeric variables*, which contain only digits and decimal points and with which arithmetic operations may be done
- *character variables* (all other kinds of data).

## SAS data step

- *data* statement
  - must begin the data step
  - gives name by which to refer to dataset during this SAS run
- *infile* statement
  - tells SAS in what physical file to find data
  - may optionally give SAS information about file and how to read it

- *input* statement
  - gives names to all variables
  - variables are assumed to be numeric unless you put a $ after name in input statement
  - may contain informats (goes *after* variable name to which it applies)
  - may contain `@` <column-number> (goes *before* variable name to which it applies)
- may contain additional statements
  - formats for printing values of variables
  - calculating new variables
  - etc.
- to embed data right into the SAS program, *datalines* statement is used instead of *infile*
- ends with *run* statement

Starting SAS in different environments

- Running SAS in the ITC

  Click on Start/All Programs/SAS/SAS 9.1 (English). SAS will come up, but it will be running on the university's "Virtual Desktop," not on your local computer. That means that SAS will not be able to find files on the local drives. It should be able to find files in your H: drive, however.

  You can access SAS on the university's "Virtual Desktop" from other Windows computers, including from home. Go to itc.uiowa.edu

  and select "Virtual Desktop" on the left panel.
- Running SAS on the Linux network

  The Division of Mathematical Sciences has a SAS licence for only one of the computers on the Linux network. If you type "sas" at the Linux prompt, you will automatically be logged into that computer (you will be prompted for your password). SAS will come up in interactive mode.

To make Linux SAS work more like Windows SAS, do the following:

- Select the Program Editor window.
- Click "Tools" and "Options," and select "Preferences"
- Click the "Toolbox" tab and DEPRESS the three toggles (Display Tools Window, Display Command Window, Combine Windows)
- Click the "Editing" tab and UNPRESS the toggle "Automatically store selection"
- Click "OK"

## Entering commands and programs

However you access SAS, you will get a screen that shows:

- a menu bar
- a log window
- a program editor window

Click in the program editor window. You may now type commands and programs in this window.

## How SAS programs and commands are organized

Use a *DATA step* to organize your data by creating a SAS dataset. Then use *PROC steps* or automated features to analyze your data. Once you have created a SAS dataset, you may apply any SAS procedures or automated features to it during the SAS session without recreating the dataset.

DATA and PROC steps consist of SAS *statements*. Each statement must end with a semicolon. Most statements include one or more *keywords* that must be spelled exactly as shown.

## The DATA step: Creating a SAS dataset

Before it can process data, SAS must read in the data in the form of a table with

- a row for each *observation*
- a column for each *variable*

You must choose a name for the entire dataset and a name for each variable. SAS has the following rules for names:

- SAS names must begin with a letter or an underscore. The remaining characters in a SAS name can be letters, numbers, or underscores. There must be no embedded blanks.

SAS distinguishes between two types of variables: *numeric variables*, which contain only digits and decimal points and with which arithmetic operations may be done, and *character variables* (all other kinds of data).

## Controlling print width

Put this line at the beginning of every SAS program if you want output to print correctly on 8-1/2 by 11 inch paper:

```
options linesize = 75 ;
```

## Reading data in from an existing datafile

Suppose you have saved the file “billion.dat” in a directory called SASdata on your H: drive. Use an “infile” statement to tell SAS to use it.

```
data billion ;                      * gives dataset a name for SAS
infile 'h:\my documents\SASdata\billion.dat' ;  * tells SAS where the data is
input wlth age region $ ;           * names the variables in each row
                                    * $ after region identifies character vbl
run ;                               * end of data step
```

## Using *datalines*

**Alternatively, you may use a `datalines` statement instead of an `infile` statement, and copy and paste the data directly into the data step. This will avoid any issues with mapping local drives to virtual drives. You must put a semicolon on a line by itself after the last row of data. This would look like:**

```
data billion ;                      * gives dataset a name for SAS
input wlth age region $ ;           * names the variables in each row
                                    * $ after region identifies character vbl
datalines;
37 50 M
24 88 U
14 64 A
13 63 U
.   .  .
.   .  .
1 59 E
1 . E
1 . O
;
run ;                               * end of data step
```

Type these lines into the program editor window. To make SAS run these statements and create the dataset, use the mouse to highlight the block of statements and then click on the icon of the running man.

SAS will use the log window to tell you what it has done. Be sure to check the log window for any error messages. If any errors are reported, click in the program editor window to make it active. Correct the errors in the code and then rerun the block of code.

**Using SAS procedures to list and tabulate the dataset** Once the dataset is created, you may run SAS *procedures* to analyze it. To list the entire dataset:

```
proc print data = billion ;
run ;
```

To get a frequency distribution of the regions in which billionaires lived:

```
proc freq data = billion ;
tables region ;              * tables is a keyword; region is the name of
                             * the variable for which you want counts ;
run ;
```

The output is:

```
                    The FREQ Procedure

                                                 Cumulativ
region      Frequency        Percent        Frequenc
---------------------------------------------------
A                  38          16.31              38
E                  80          34.33             118
M                  22           9.44             140
O                  29          12.45             169
U                  64          27.47             233
```

## Proc univariate: SAS workhorse of descriptive statistics

Use *proc univariate* for quantitative variables when you want the following:

- means
- medians
- quartiles
- 5-number summary
- stem plots (for small datasets) or histograms (large datasets)
- boxplots

```
proc univariate plot data = billion ;
var wlth  ;
run ;
```

The output is:

```
                  The UNIVARIATE Procedure
                       Variable:  wlth

                           Moments

N                         233    Sum Weights                233
Mean               2.68154506    Sum Observations         624.8
Std Deviation      3.31884032    Variance            11.0147011
Skewness           6.57544276    Kurtosis            56.9655987
Uncorrected SS        4230.84    Corrected SS        2555.41064
Coeff Variation    123.765972    Std Error Mean      0.21742446


                 Basic Statistical Measures

       Location                    Variability

    Mean     2.681545     Std Deviation            3.31884
    Median   1.800000     Variance                11.01470
    Mode     1.000000     Range                   36.00000
                          Interquartile Range      1.70000


                 Tests for Location: Mu0=0

      Test           -Statistic-    -----p Value------

      Student's t    t  12.33323    Pr > |t|    <.0001
      Sign           M     116.5    Pr >= |M|   <.0001
      Signed Rank    S   13630.5    Pr >= |S|   <.0001


                  Quantiles (Definition 5)

                  Quantile      Estimate

                  100% Max          37.0
                  99%               14.0
                  95%                6.2
                  90%                4.5
                  75% Q3             3.0
                  50% Median         1.8
```

```
                  25% Q1             1.3
                  10%                1.1
                  5%                 1.0
                  1%                 1.0
                  0% Min             1.0


                    Extreme Observations

              ----Lowest----        ----Highest---

              Value      Obs        Value      Obs

                  1      233           13        4
                  1      232           13        5
                  1      231           14        3
                  1      230           24        2
                  1      229           37        1


                   Histogram                      #             Boxplot
  37+*                                            1                *
     .
     .
     .
     .
     .
     .*                                           1                *
     .
     .
  19+
     .
     .*                                           1                *
     .*                                           2                *
     .*                                           2                *
     .*                                           2                0
     .*                                           3                0
     .********                                  23                0
     .************************                  72             +--+--+
   1+******************************************  126             *-----*
     ----+----+----+----+----+----+----+----+--
     * may represent up to 3 counts
                 -2        -1         0        +1        +2
```