Linear Statistical Modeling Methods with SAS

Introduction to SAS

Xuemao Zhang
East Stroudsburg University

January 17, 2024

Outline

- Course Topics
- Introduction to SAS
- Data entry/import

Course Topics

- Course topics
  - Review of elementary statistics
  - Simple and multiple linear regression models
  - Generalized Linear Models
  - Supervised learning
- Software
  - SAS programming
  - SAS Viya https://support.sas.com/en/software/sas-viya-programming.html

Introduction to SAS

- SAS (https://en.wikipedia.org/wiki/SAS_(software)) was developed in the early 1970s at North Carolina State University. It is originally intended for management and analysis of agricultural field experiments and now it is the most widely used statistical software. SAS is used to stand for "Statistical Analysis System", but now it is not an acronym for anything. SAS is pronounced "sass", not spelled out as three letters.
- SAS is the most widely used statistical software due to the following several reasons:
  - It is often better with very large datasets and memory;
  - It can deal with multiple datasets at the same time.
  - It is better for data manipulation.
  - It is better on the Job Market (big company).

Introduction to SAS

- The SAS software suite has more than 200 components (https://support.sas.com/software/). Some of the SAS components include:
  - Base SAS – Basic procedures and data management
  - SAS/STAT – Statistical analysis
  - SAS/IML – Interactive matrix language
  - SAS Viya https://support.sas.com/en/software/sas-viya-programming.html

Introduction to SAS

- We will use Base SAS and SAS/STAT in this course to manipulate data and conduct basic statistical data analysis and machine learning. SAS/IML will be used a little bit in our course.
- SAS Viya will be used for machine learning.
- SAS/IML (https://support.sas.com/) software, is like statistical software R, gives you access to a powerful and flexible programming language (Interactive Matrix Language) in a dynamic, interactive environment. The fundamental object of the language is a data matrix. You can use SAS/IML software interactively (at the statement level) to see results immediately, or you can store statements in a module and execute them later. The programming is dynamic because necessary activities such as memory allocation and dimensioning of matrices are done automatically.

Introduction to SAS

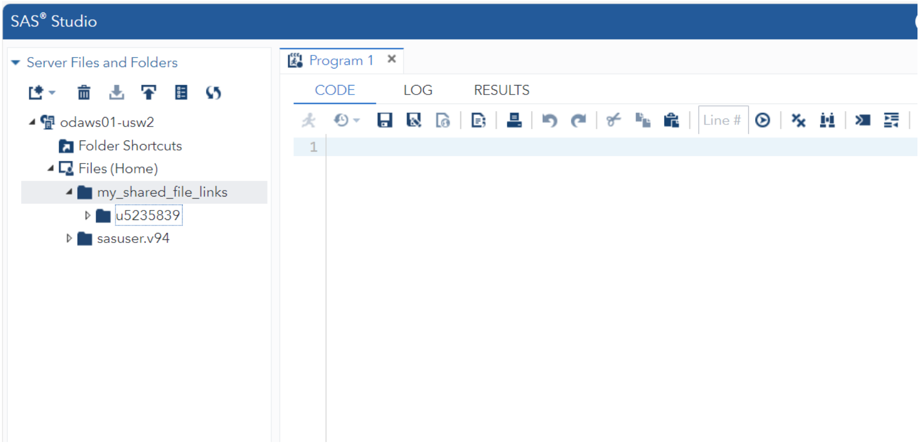
- R is a matrix-based programming language that allows you to program statistical methods reasonably quickly. It's open source software, and many add-on packages for R have emerged, providing statisticians with convenient access to new research. Many new statistical methods are first programmed in R.
    - SAS/IML Studio (https://support.sas.com/rnd/app/studio/) also provides the capability to interface with the R language.
    - R will be used in our course MATH 318- Exploratory Data Analysis.

Introduction to SAS

- SAS OnDemand for Academics
  - It is free for institution use.
  - Please create an account https://www.sas.com/profile/ui/#/create first.
  - Sign on the the Dashboard at https://welcome.oda.sas.com
  - Look for and select the *Enrollments* tab and select the `+ enroll in a course`" ' link to start the enrollment.
  - Enter the course code: **820ecb9c-8fc2-48af-83e6-d67504f0c3df**.
  - Submit the form.
- SAS Viya will be discussed later in the course.

Introduction to SAS

- SAS OnDemand for Academics



Introduction to SAS

- SAS OnDemand for Academics. The SAS Studio consists of the 4 main windows:
    - 1. Left panel: Upload data to the cloud etc.
    - 2. Code window – Used to create, edit and execute SAS programs
    - 3. Log Window – Reports on progress of SAS procedures (BLUE). Displays error messages (RED) and warnings (GREEN).
    - 4. Results Window – Displays output from SAS program. Can view, print, copy or save information in the output window

Introduction to SAS

- General Information
  - All statement lines must end with a semi-colon ;.
  - Comments are indicated in 2 ways:
    1. Start line with * and end with a semi-colon ;
    2. Enclose with as /* put comment text here */
  - End all procedures with RUN.
- To start our first SAS program, we need to get data into SAS first. There are several ways to get data into SAS: 1) Read in-stream data, 2) Use INFILE statement, and 3) Import the data from Excel. Let's use several examples to show how to read raw data into SAS.

Data entry/import

Problem: Reports of results from clinical trials often include statistics about "baseline characteristics", so we can see that different groups have the same basic characteristics. It is of interest to see the difference between the mean age of men (μ_1) and the mean age of women (μ_2). A sample of 40 men and a sample of 40 women are randomly and independently selected and the ages are recorded in the file BODY.csv or BODY.txt.

Data entry/import

- Method 1(copy the data to the SAS code using `data` step):
  - the symbole $ specifies that `gender` is categorical.

```
data age1;
input age gender$ @@;
cards;
18 M 20 M 43 M 39 M 60 M 18 M 57 M
27 M 20 M 18 M 63 M 20 M 24 M 46 M
29 M 63 M 21 M 45 M 40 M 50 M 48 M
64 M 18 M 50 M 20 M 20 M 47 M 19 M
55 M 23 M 21 M 19 M 64 M 30 M 43 M
23 M 64 M 40 M 23 M 44 M 60 F 24 F
49 F 62 F 53 F 18 F 41 F 21 F 21 F
19 F 19 F 58 F 44 F 52 F 48 F 36 F
48 F 34 F 22 F 61 F 21 F 33 F 32 F
37 F 19 F 51 F 35 F 18 F 60 F 58 F
60 F 48 F 31 F 29 F 46 F 18 F 50 F
20 F 56 F 18 F
;
run;
```

Data entry/import

- The `@@` prevents the input pointer from moving to the next line after reading values, allowing you to read **multiple values from the same line**.
- It is usually a good idea to print the first few (or all) cases of your dataset to check that things were read correctly.
- You can see a `OUTPUT DATA` window
- After reading in the data with a data step, We can print the first several cases of the data set.

```
proc print data= age1 (obs=10);
run;
```

Data entry/import

- Method 2(use INFILE statement in the data step):

```
data age2;
infile "/home/u5235839/my_shared_file_links/u5235839/BODY.txt"
firstobs=2; /*change the directory if necessary*/
input age gender$;
run;
```

```
proc print data= age2 (obs=10);
run;
```

Data entry/import

- Method 3 (use IMPORT procedure):

```
proc import
datafile="/home/u5235839/my_shared_file_links/u5235839/BODY.xlsx"
out=age3 dbms=xlsx replace;
/*
dataset with same name age3 will be replaced
dbms=xls specifies the type of database management system (DBMS)
*/
getnames=yes;
run;

proc print data= age3 (obs=10);
run;
```

Data entry/import

- Method 4: Right click a data set and choose `Import Data`
- Method 5: The SAS `Import Data` wizard can be used to access spreadsheets (Excel, Lotus) and database (Access) files as well.

Variable Transformations

- Transforming data is an important technique in exploratory data analysis. For example, centering and scaling are simple examples of transforming data. There is no need to transform your raw data outside of SAS. You can use Data step to transform your data.

```
data data1;
input x y gender$@@;
cards;
4.5 17.5 M
2.3 20.5 F
6.7 13.6 F
8.4 17.8 M
-2.0 14.7 M
;
run;

proc print data=data1;
run;
```

Variable Transformations

- Now add a variable z, a log transformation of y, to the data set.

```
data data2;
input x y gender$@@;
z=log(y);
cards;
4.5 17.5 M
2.3 20.5 F
6.7 13.6 F
8.4 17.8 M
-2.0 14.7 M
;
run;

proc print data=data2;
run;
```

Variable Transformations

- Equivalently, we can do this as well.

```
data data3;
set data1;  /* create data3 from data1 */
z=log(y);
run;
```

- `set data1` copies the observations from the existing dataset `data1` to the new dataset `data3`.

Variable Transformations

- Deleting rows: We can delete some rows based on a control condition.

```
data data4;
set data3;
   if x > 0;  /*  keep rows with x>0 only */
run;
```

Variable Transformations

- Deleting variables: We can delete a column as well.

```
data data5;
set data4;
   drop y;  /*  remove variable y*/
run;
```

do ... end loop

- do ... end loop: It is similar as the for loop in R which is used for iterating over a sequence.

```
data uniform;
   do i = 1 to 10;
      x = ranuni(0); /* random uniform number between 0 & 1 */
      output;
   end;

proc print data=uniform;
run;
```

- Note the use of a do loop, which is ended by an end; phrase. The output forces creation of a new case for each uniform number. Each case in set a will have the variables x and i.

do ... end loop

- Here is a list of random number generators:

```
x = ranuni(seed)        /* uniform between 0 & 1 */
   x = a+(b-a)*ranuni(seed);    /* uniform between a & b */
   x = ranbin(seed,n,p);    /* binomial size n prob p */
   x = rancau(seed);        /* cauchy with loc 0 & scale 1 */
   x = a+b*rancau(seed);    /* cauchy with loc a & scale b */
   x = ranexp(seed);        /* exponential with scale 1 */
   x = ranexp(seed) / a;    /* exponential with scale a */
   x = a-b*log(ranexp(seed));   /* extreme value loc a & scale b */
   x = rangam(seed,a);      /* gamma with shape a */
   x = b*rangam(seed,a);    /* gamma with shape a & scale b */
   x = 2*rangam(seed,a);    /* chi-square with d.f. = 2*a */
   x = rannor(seed);        /* normal with mean 0 & SD 1 */
   x = a+b*rannor(seed);    /* normal with mean a & SD b */
   x = ranpoi(seed,a);      /* poisson with mean a */
   x = rantri(seed,a);      /* triangular with peak at a */
   x = rantbl(seed,p1,p2,p3);   /* random from (1,2,3) with probs p1
```

License