Introduction to SAS

Shannon Pileggi

STAT 330

OUTLINE

Course overview

Intro to SAS

About me

Degrees

- BS Mathematics and Hispanic Studies
- MS Biostatistics
- PhD Biostatistics

Personal

- Married, have a 2 year old daughter and 2 dogs
- Enjoy: bike commuting, soccer, disc golf, hiking, board games

SAS versus R

- The two dominating statistical software packages are SAS and R
  - R is freeware (pros and cons)
  - SAS is not free (pros and cons)
- SAS is superior at performing certain tasks compared to R
- R is superior at performing certain tasks compared to SAS

About SAS

- SAS (**S**tatistical **A**nalysis **S**ystem) was developed in the 1970s by a couple graduate students at NC State University.
- The software is now simply known as 'SAS' since its application extends beyond statistical analyses.
- **THE WORLDWIDE LEADER** in industry as the main statistical toolbox software.
- SAS consistently lands near the top of Fortune's annual list of best places to work.
- statweb.calpoly.edu/jdoi/web/classes/stat330/articleSAS.pdf
- statweb.calpoly.edu/jdoi/web/classes/stat330/articleSAS2.pdf

About this course

- This course will be more like a computer science class, where you learn the basics of a computing language
- There is not a lot of statistical theory in this course, but the examples we use may build upon your previous statistics courses
- You will learn a *few* statistical analysis methods using SAS (e.g. t tests), but will focus *more* on programming in SAS

Learning objectives

1. Formulate a game plan that states the objective before coding.
2. Identify multiple ways to achieve the game plan; consider pros and cons of the options before coding.
3. Apply techniques to prepare data for analysis, including: merging or transposing data, creating new variables, and identifying data errors and "cleaning" data.
4. Prepare summary statistics of data; create graphical displays of data.
5. Execute various statistical analyses and interpret results.
6. Apply arrays, loops, and SAS macros for efficient coding.
7. Discuss how SAS's progam data vector operates.
8. Import data of various sources and formats into SAS.
9. Verify that code renders the desired result in the presence of missing data.

Course resources

- Online Documentation support.sas.com/documentation
- SAS Proceedings http://lexjansen.com/
- Other helpful online resources from SAS Support

  Knowledge Base support.sas.com/resources
  Support support.sas.com/techsup
  Training and Bookstore support.sas.com/learn
  Community support.sas.com/community

Acknowledgements

A special thanks to the following institutions and instructors who have allowed me to borrow course notes and examples:

- Rebecca Ottesen
- Hunter Glanz
- Jimmy Doi
- SAS

Other

- Review PolyLearn site
- Discuss syllabus
- Discuss ways to access SAS
- Demonstrate shared directory

Course overview

Intro to SAS

On your own:

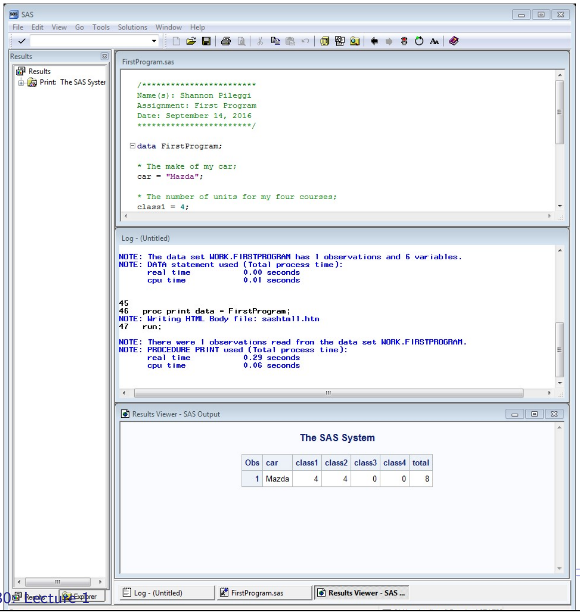
SAS Code

```
/* Input data */
DATA grades;
   INPUT name $ exam1 exam2 exam3;
   DATALINES;
   Shannon      96    82    83
   Lex          92    81    68
   Becky        92    75    73
   ;
RUN;


/* Print data */
PROC PRINT DATA = grades;
RUN;
```

SAS Code

SAS session: structure of windows



```
/************************
Name(s): Shannon Pileggi
Assignment: First Program
Date: September 14, 2016
************************/

data FirstProgram;

  * The make of my car;
  car = "Mazda";

  * The number of units for my four courses;
  class1 = 4;
```

```
NOTE: The data set WORK.FIRSTPROGRAM has 1 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds


45
46   proc print data = FirstProgram;
NOTE: Writing HTML Body file: sashtml1.htm
47   run;

NOTE: There were 1 observations read from the data set WORK.FIRSTPROGRAM.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.29 seconds
      cpu time            0.06 seconds
```

The SAS System

Obs	car	class1	class2	class3	class4	total
1	Mazda	4	4	0	0	8

Windows

1. **Editor**: SAS code is created here. Files saved from this window will have the `.sas` extension. Instead of using the regular "Program Editor", use the Enhanced Editor window (more user friendly).
2. **Log**: Messages pertaining to the successful (or unsuccessful) implementation of your code is listed here. Files saved from this window will have the `.log` extension.
3. **Results Viewer**: If your code led to any text output, it will be stored here. Files saved from this window will have the `.html` extension.
4. **Explorer**: Access libraries and SAS data sets

NOTE: If you want to save/print the contents of a particular window, be sure that it is 'active' by using the mouse to click on it.

SAS programs

SAS programs have two parts:

1. Data steps: read and modify data
2. Procedure (aka proc) steps: analyze data and produce reports

Our SAS program had ______ data step and ______ proc step.

1. 2; 0
2. 0; 2
3. 1; 1
4. 1; 2

SAS data sets

- SAS works with its own data sets - raw data sets must be a SAS dat set or converted to a SAS data set before you can work with it.
- Extensions for raw data sets can be almost anything - `.dat`, `.txt`, `.xls`, `.csv`, etc.
- Extensions for SAS data sets: `.sas7bdat`
- SAS can handle many observations (rows) and variables (columns), which depends on your computer's memory. (Prior to Version 9.1 SAS could accommodate 32,767 variables.)
- SAS data sets are self-documenting - they contain information about when it was created, the number of observations and variables, variable types, etc.

Data sets

Columns indicate **variables** ↓

Rows indicate **observations** →

Obs	name	exam1	exam2	exam3
1	Shannon	96	82	83
2	Lex	92	81	68
3	Becky	92	75	73

SAS data sets consist of two data types:

1. **Numeric**: any numeral including +/-, dates (e.g. 01/03/2005),, decimal and scientific notation
2. **Character**: Letters, numbers, special characters (up to 32,767, even in SAS v9.2)

Data types

- If a variable contains letters or special characters, its data type is *character*.
- If a variable contains all numbers, its data type can be either *numeric* or *character*.
- When deciding how to analyze your data, you should base your decision on what it represents in reality (eg, categorical versus quantitative).

Zip codes in the United States are 5 digit numbers (Cal Poly's is 93407). If I had a data set with a variable for zip code of counties the SAS data type would likely be ______________, and in reality this would represent a ______________ variable.

1. character; categorical
2. character; quantitative
3. numeric; categorical
4. numeric; quantitative

Variable and data set names

- Must be 32 characters in length or less
- Must start with a letter or underscore '_'
- Can only contain letters, numbers, or underscores
- Variable names are not case sensitive (the following variable names are equivalent: 'Gender', 'GENder', 'GENDER', ...).
- However, the values stored for a particular variable **are** case sensitive!

Which of the following are not valid variable names?

1. _age
2. Age.1
3. 1age
4. Age1
5. ThisIsTheAgeWeStudy
6. Age_1
7. AgE

Missing data

Many raw data sets contain missing observations. These are ‘stored’ in SAS data sets according to the data type.

- If the missing data type is numeric, it is stored as a period: {.}
- If the missing data type is character, it is stored as a blank: { }

Name	Age	Gender	Height	Weight
Max	33	male	.	204
Sally	21	female	68	143
Susan	25		65	142
Bob	.	male	73	215

Comments in SAS code

- Following general programming etiquette, it is **ESSENTIAL** that you include comments in your SAS code!
- This helps in making your code easier to read and reminds you why you may have coded things in a particular way.
- Comments can be invoked in two ways:

```
/* Anything placed here is a comment */
* Anything placed here is a comment ;
```

- Comments can span across multiple lines of code.
- To quickly comment/uncomment a section of code:
  - Select the code section by highlighting with the cursor and then use Ctrl + /
  - To uncomment any selected code simply use Ctrl + Shift + /

Common errors

- Misspelling a variable name or SAS key word
- Forgetting a semi-colon
- **Every SAS statement ends with a semi-colon!**
- Always check your log

On your own: Try removing a semi-colon or misspelling words to see what happens in your log.

Debugging your code

If you have an error in your code...

- It is VERY VERY helpful to use the `/* ... */` style of comments to *hide* sections of code
- Move the comment marker to sequentially reveal code **one portion at a time**
- By using this stepwise unveiling of code you should eventually be able to identify the error source(s)

On your own:

Copy this code into the SAS editor and identify the 3 errors.

SAS Code

```
DAT grades;
   INPUT name $ exam1 exam2 exam3;
   DATALINES;
   Shannon      96    82    83
   Lex          92    81    68
   Becky        92    75    73
   ;
RUN

PROC PRINT DATA = grade;
RUN;
```

SAS Code

Saving your SAS code:

- It is good practice to **regularly** save SAS code (even if it's still a work in progress)
- Be sure to have the Enhanced Editor window 'active', then save your file.
- Eventually, transfer all files to a personal storage device (flash drive, cloud storage, email, CD-Rom). The hard drives on the lab computers are wiped clean every evening!

Shortcut Keys

- Go to Tools → Options → Keys (or F9)
  Useful default keys:
  - F5 = `wpgm` = Enhanced Editor
  - F6 = `log` = Log Window
  - F7 = `output` = Output Window
  - F8 = `sub` = Submit SAS Code
- Create your own key in F12 - do this every time you come to class! This will greatly assist in identifying errors.

  `odsresults;clear;log;clear;wpgm;submit;log;top;`