EPBI 414

Unit 8 - Introduction to SAS

The Recap - Unit 7

- Using SQL in more advanced ways
  - Outer joins, ON vs. WHERE
  - UNION, UNION ALL

- Using SQL to do basic analytic processes
  - Aggregation, summarization, functions

- Very simple database administration
  - `INSERT UPDATE DELETE`
  - `CREATE ALTER DROP`

Unit 8 Overview

- Introduction to SAS
  - What is SAS? History and features
  - Server and Windows versions of SAS
  - General overview of SAS - syntax and programs
  - Setting and manipulating options
  - Datasets; temporary vs. permanent; libraries

What is SAS?

- Short for **S**tatistical **A**nalysis **S**ystem

- Developed originally at North Carolina State University in the 1960s

- Spun out into its own company in the 1970s
  - Steadily grown and expanded ever since

- An entire ecosystem of analytic products

Why SAS?

- SAS is still industry standard in many fields
  - Clinical trials (especially FDA regulated)
  - Contract research organizations
  - Insurance industry
  - Public health departments
- You can make a good career out of being a SAS programmer
  - Sometimes easier than being an R programmer!

Why SAS? (continued)

- Large datasets

- Reliability
  - SAS is very well-tested and is reliable
  - The flaws are well-known and documented

- Backwards-compatible
  - SAS code often works for many generations without much adjustment

Why SAS? (finale)

- Clear and extensive documentation of how things work

- Large, active community with huge amounts of published guidance
  - SUGI papers (SAS Users Group International) are awesome (now SAS Global Users Group)

  - Google will bring up a ton of stuff for SAS questions

Why not SAS?

- Not always on the cutting edge
  - Active work in statistics is almost always done in R
  - R is the lingua franca of most practicing academic statisticians
    - Python is making steady in-roads, and is the subject of much debate
- Very expensive
- Not cross-platform capable

Why not SAS (contd)?

- Not really a full programming language
  - You can make it do a lot of things, but not always well
  - Lacks some abilities that are provided by other tools like R and Python
- Based on a very specific paradigm of tabular data
  - Non-standard data can be challenging in SAS

Base and extended SAS

- SAS is a system, and there are many different modules

- Base SAS forms the core of the program, and ties together modules

- The unifying feature is the "SAS programming language"
    - We will often just call this SAS

Installing SAS

- SAS is installed on the EPBI lab computers
- You may also obtain it from the CWRU Software Center for a nominal fee
- If you choose to install it...PLAN AHEAD
    - Installs can take a lot of time
    - As of a few years ago, required on-campus connection (VPN or Case Wireless)

Learning SAS

- This class is going to give you some basic skills in SAS

- Next steps would be training enough to pass the SAS Certified Base Programmer exam
  - Would probably require a full semester on SAS alone

- Other certifications exist - opinions vary on their utility

What does SAS do?

- Four key data-driven tasks that you can accomplish with SAS:
  - Access
  - Management
  - Analysis
  - Presentation

Data Access

- SAS natively has its own data format (.sas7bdat)

- But - you can read a lot of files using SAS!
  - Imports third-party files using various procedures (more on this later)

  - Also can read text data natively

Data Management

- SAS supports the process of data cleaning
    - Sometimes "cleansing", "munging", "scrubbing"...
    - The process of preparing data for analysis
    - Often better to design your data better (first part of the course)
- You can also use SAS to manipulate data
    - Psychometric scoring, coding of values, etc

Data Analysis

- SAS has powerful, ***highly stable***, and well-tested analysis packages

- It often takes many years for SAS to implement an analytic method in their system
  - Pro: it is robustly tested and documented very well
  - Con: you may not have access to newer methods

Data Presentation

- SAS includes numerous methods of presenting data

- We will cover basic outputs,and then using the Output Delivery System (or ODS)

- R is still arguably the best graphics
    - You can integrate R and SAS now! (advanced)

SAS Environments

- SAS can be deployed to individual workstations, or on a server
  - We have both options in EPBI - class will use workstation version

- The server version is accessed over the command line

- Workstation version limited to Windows

SAS on Unix

- Programs are edited using a text editor, submitted all at once ("batch" mode)

- Program output is written to a file (ending in `.lst`, for the "listing" output)

- Limited graphics and interactivity - but useful in certain cases
    - Regular processing, speed requirements, etc

SAS in Windows

- Allows for highlighting and submitting only ***portions*** of your code

- Provides a useful IDE (integrated development environment) with highlighting and other features

- Allows for full range of SAS graphics and outputs

Anatomy of a SAS program

In general, you create a ***SAS program*** by putting the SAS commands you would like to execute into a text file.

SAS generates two sets of output: a log of what was done and the program output (listing).

SAS commands

- A SAS command almost always starts with a SAS keyword, and ends with a semicolon (;)

- Examples:

```
DATA WORK.studydata;
   SET WORK.registry;
   IF study_id=‘cru1233’;
RUN;
PROC PRINT DATA=WORK.studydata;
RUN;
```

Where's my output?

- On the server, your output goes into two files, `program.lst` and `program.log`

- In Windows, this is integrated into the SAS IDE, not saved to a file

- You ***can*** save these outputs, however
    - Certain SAS modules allow for different outputs (case in point: ODS)

HTML vs. LISTING

- In previous versions of SAS, your output went to the LISTING
  - Basically, a text output

- Newer versions default to HTML-style output
  - This uses ODS

- For learning, it is easier to go back to LISTING

Enable LISTING output

- Tools -> Options -> Preferences...
- Results Tab
  - Uncheck "Create HTML"
  - Check "Create listing"
- We will revisit HTML (and other ODS stuff) in the future

SAS IDE
Overview

SAS program elements

- Two basic structures in a SAS program: **DATA** step and **PROC**EDURES

- The **DATA** step manipulates data sets in multiple ways

- A **PROC**EDURE generally performs an analysis or other task
  - Can create new data

Some other elements

- **OPTIONS** is used to set various program options

- **TITLE** is used when setting titles for outputs

- **ODS** is used when controlling the Output Delivery System

DATA step examples

- Reading in a CSV file
- Scoring a psychometric instrument
- Obtaining a mean value from multiple columns
- Transposing data from long to wide

PROC examples

- Summarize data (means, medians, etc)
- Generate linear models
- Produce figures or charts
- Many, many more...

Getting started with SAS

- The DATA step is very powerful, but we won't dive into it in depth until later units

- However, you will use it to do limited data manipulation over the next few weeks

- We start with working with nicely formatted data, then proceed to mashing it up!

How SAS works - broadly

- SAS starts by reading the whole program looking for errors
  - Syntax errors cause the program to abort before it runs
- If there are no errors, then it goes from top to bottom
- It compiles and executes each statement separately, one by one

How it often works

- Often, your programs will not work
  - Welcome to programming
- You will spend a lot of time reviewing your logs to find your errors
- You will get used to looking for syntax problems

A peek under the hood

- The DATA step involves loading each row of data into the ***program data vector (PDV)***

- You will hear more about this in the future - just know that it exists for now

- When data is being analyzed or manipulated, it is stored in the PDV

A note on SAS coding style

- We discussed coding conventions previously in SQL
  - Same basic idea applies in SAS
- As before, if you aren't sure, you can always look to my style
- SAS is flexible when it comes to whitespace, capitalization, et cetera

Basic SAS conventions

- Write yourself a preamble to document your code

- Capitalize the SAS keywords and leave other words lower case

- Use indentation wisely

- Comments are the best!

On comments, specifically

- Remember this from before?

- **Comment. Your. Code.**
    - It is good practice.
    - In this class, it will affect your grade.
    - It helps you understand what you were doing.
    - It helps others understand what you were doing.

SAS comments

- In SAS, you may comment in two ways

```
* inline comment;

/* block comment */
```

- Block comments are useful to make section notes; use inline to explain specific lines

Two "correct" commands

A poor way:

```
data work.surveys;set work.allsurveys;if study=123;run;
proc print data=work.surveys;run;
```

A better way:

```
DATA WORK.surveys;
   SET WORK.allsurveys;
   IF study=123;
RUN;
PROC PRINT DATA=WORK.surveys;
RUN;
```

SAS Options

- **OPTIONS** control the execution of your SAS program
- Default options are set each time SAS is started
- If you want to change them, you do so at the start of your program

OPTIONS syntax

```
OPTIONS <option1>=<value> <option2>;
```

- Some options take a value (like `ls`)
- Others are just statements (like `nocenter`)

Common SAS options

- PAGESIZE and LINESIZE
  - Control the number of lines, and their length, in outputs (less important using ODS)
- NOCENTER - Controls text centering
- NUMBER - Controls page numbering
- Many more...

Setting a few options

An example:

```
OPTIONS LS=80 PS=60 NOCENTER;
```

Find the current options:

```
PROC OPTIONS;
RUN;
```

Break

Time

SAS datasets

- SAS has two main types of dataset: ***temporary*** and ***permanent***

- ***Temporary*** datasets exist only during a session
  - You quit, they vanish (poof)

- ***Permanent*** datasets are, as you might imagine, more permanent

Dataset names

- SAS datasets always have ***two part names***
- The first part, before a dot (.), is the ***library name*** (we'll cover this in a second)
- The second part, after the dot (.), is the dataset name (and also the name of the file)
  - SAS datasets are stored in .sas7bdat files

Example dataset names

Good examples:

```
WORK.surveydata
CRU1233.patients
YRBSS.f01_consent
```

Technically correct examples:

```
john.hisdata
abcd.tables
```

Libraries in SAS

- SAS utilizes ***libraries*** to organize datasets
- A ***library*** in SAS is a location where SAS will look for SAS datasets
    - Will also look for other things there - like formats (more on this later)
- You define a ***LIBNAME*** to tell SAS where data is located

The WORK library

- By default, datasets in the WORK library are temporary

- At the end of the SAS session, they will disappear

- If you do not explicitly state a library, SAS assumes you are using temporary library
    - So, `WORK.dataset` and `dataset` are the same

Rules of naming

- SAS datasets must be named according to certain rules:
  - The length must be between 1 and 32 characters
  - The name must start with a letter or an underscore (_)
  - The name may contain letters, numbers, and underscores

Other rules of naming

- LIBNAMES are also constrained in naming
  - Partially a product of SAS's long development history
- Must be no more than 8 characters
- Follows the other conventions of a dataset

Declaring a LIBNAME

- Before you can access a library, you have to declare a LIBNAME

- The LIBNAME tells SAS where to find .sas7bdat files with data

- The syntax is:

  ```
  LIBNAME <libref> <path>;
  LIBNAME CRU1233 'C:\Trials\CRU1233';
  ```

SAS IDE
Show Libraries

What's in a data set?

- Columns and rows
  - Rows are often referred to as "cases"
  - Columns commonly labeled "variables"
- Also, some "metadata" which SAS uses
  - Literally means "data about data"
  - We will explore this in limited ways this week

Types of columns

- SAS has two primary distinctions, ***numeric*** and ***character***

- ***Character*** fields can contain up to 32,767 characters, including special characters
  - Avoid semicolons if possible...

- ***Numeric*** fields by default store 8 bytes
  - Pretty big from an integer standpoint

Other constraints

- When reading in ***numeric data***, commas, dollar signs, etc can cause problems
  - e.g. 100,000 vs 100000, $2.37 vs. 2.37

- Numeric data may be ***preceded*** by a + or - or an E for exponential

- A single dataset is always tabular
  - Multiple tables cannot be saved in one file

Missingness

- SAS has a built-in marker for missingness
- For character data, missing data is denoted by a blank space (" ")
- For numeric data, missing data is denoted by a dot (.)

Naming variables

- In the bad old days, names were limited to 8 characters
  - That's how you end up with F3RD2A as a variable name

- Now, variable names may be 1 - 32 characters
  - Must start with a letter or an underscore
  - Must only contain letters, numbers, underscores

Formats - a brief preview

- SAS lets you display data in many ways using formats

- This lets us do many useful things, like handle dates, show dollar signs and commas, etc

- We will discuss this much more next week
    - Also affects dates, etc

PROC CONTENTS

- To learn more about a dataset, use PROC CONTENTS

- As you might imagine, it tells you about a dataset's contents
  - Includes dimensions, types, encoding, and all manner of useful things

  - Try with VARNUM to see in normal ordering

Getting data into SAS

- Three major ways of getting data into a SAS session
  - Easiest: Use SET in the DATA step to copy an existing dataset
  - Medium: Use PROC IMPORT to read in data from an outside file
  - Harder: Use INPUT with INFILE and possibly DATALINES in the DATA step

Using SET

- Using SET in the DATA step lets you create a new dataset from another one:

```
DATA X;
  SET Y;
RUN;
```

- This can allow you to copy datasets easily
  - Other commands exist to do this too

Using INPUT in DATA step

- Reads data either from a text file or the program itself (INFILE and DATALINES)

- This allows us to use four styles:
  - List
  - Column
  - Formatted
  - Named

INFILE and DATALINES

- INFILE is a SAS keyword that permits the engine to read data from a file

- DATALINES allows us to read data from lines in the actual input itself (think back to STDIN and STDOUT)

- DATALINES is not very common in regular work

PROC IMPORT

- A procedure which allows importing data from multiple sources
    - Includes non-text sources like Excel files
    - Also, other statistics engines like SPSS
- Contains its own syntax and rules for how the command works

Why not PROC IMPORT?

- At one time, it was not part of Base SAS and so it cost money
    - Not sure if this is still the case
- Can be overkill for reading in text files
- It is important to understand how it works "under the hood"

Practical considerations

- For the bulk of all practical analysis work, you'll use PROC IMPORT

- If you are doing heavy data management, you will likely use more of INFILE and the DATA step

- You may also get into exotic things like XML and querying a RDBMS from SAS

For today...

- Focus on creating temporary data from permanent data, and vice versa

- This involves using the DATA step and SET
    - Remember: the WORK library is the temporary library

    - If your DATA step uses a dataset without a library name, it is temporary

Temporary from permanent

```
DATA somedata;
  SET CRU1233.data;
RUN;

DATA WORK.somedata;
  SET CRU1233.data;
RUN;
```

Permanent from temporary

```
DATA CRU1233.newdata;
  SET mydifferentdata;
RUN;

DATA CRU1233.newdata;
  SET WORK.mydifferentdata;
RUN;
```

Sorting and printing

- We will expand much more on this next week
- For now, some basics on printing and sorting data sets
- PROC PRINT and PROC SORT

PROC PRINT

- Prints data to the output device
  - This was once much more important before the IDE
- Can be customized to do many things
- Default print:

```
PROC PRINT DATA=<data>;
RUN;
```

PROC SORT

- PROC SORT is a procedure which can affect permanent datasets

- As such - it's good to be cautious with it
  - You ***do*** keep backups, right?

- Very basic syntax:

```
PROC SORT DATA=<data>;
  BY <vars>;
RUN;
```