Introduction to SAS

How to Get SAS®

The University has a license agreement with SAS® for individual use for $35 a year. You can buy and download the software package at: https://software.ucdavis.edu/index.cfm. If you can have the software installed on your laptop and bring it with you to the seminar, we can work through the examples and exercises together. SAS will run in Windows® Vista, 7, 8, and 10, and on UNIX. To run SAS on a Mac® you can run SAS University, which is a limited version that is free and runs virtually on OS X

http://www.sas.com/en_us/software/university-edition/download-software.html#os-x

or you will need to log in remotely to a Windows or UNIX system or use a desktop virtualization program such as Parallels®. Your department can purchase a license for Parallels® using the link above for a discount. You can get a free 14 day trial of Parallels at:

http://trial.parallels.com/?lang=en&terr=us.

What is SAS®?

SAS is an integrated software system that enables you to perform a variety of tasks. For the purposes of this seminar series, we will be focusing on:

1) Data entry, retrieval, and management
2) Tables and graphics
3) Statistical analysis

For this session held on Sept 14 and 21 we will mainly discuss (1) and (2) as a start. The future seminars will discuss (3) in detail in the context of statistical analysis methods covered.

SAS stands for Statistical Analysis Software, but now we just use SAS as the registered trademark for the software package. When discussing SAS software we should always include the 'software' after 'SAS', according to SAS Institute. For example, we should say "SAS® software version 9.4 was used for all analyses for this seminar." Though we will use a less formal style from here forward, a full citation of the use of SAS software would be as follows:

"The data analysis for this paper was generated using SAS® software, Version 9.4 of the SAS System for Windows. Copyright © 2015. SAS Institute Inc. SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc., Cary, NC, USA."

Basic Rules for SAS Programming Statements

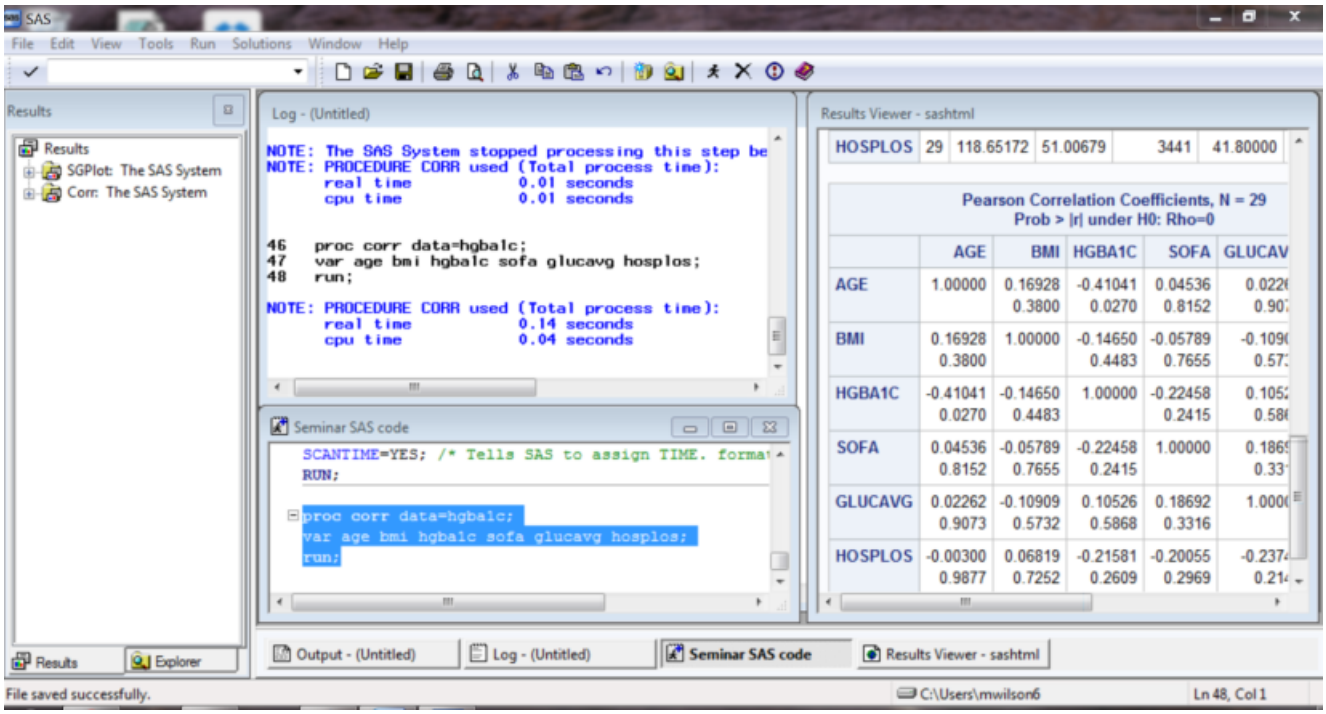
The code I show you here is a mixture of convention and hard rules for SAS syntax. Indentations, capitalization, extra spacing, and blank lines are all personal preference but are widely used to make the code easier to read. For example, SAS statements do not have to be on separate lines, but they do have to end with semicolons. Here is a list of what is required and what is flexible in the syntax:

1) SAS statements must end with a semicolon. [;]
2) No blanks can appear in SAS names. Variable names with more than one word should be joined by an underscore. For example, a variable indicating whether or not a good neurological outcome occurred should be named 'good_neuro' not 'good neuro' or 'good-neuro'.

3) Code is not case sensitive, but variable values are. That is, the word '**DATA**' in a line of code will be read the same as '**data**', but `'M'` will be read as a different value than `'m'`; and so SEX would be read as a three category variable with outcomes `'M'`, `'m'`, and `'F'`.
4) There can be more than one statement per line and a statement can be on more than one line.
5) We can add annotation and documentation to our SAS code using `/*comments*/`. Everything that appears in between the `/* */` will not be read as code.
6) A SAS name cannot be longer than 32 characters. (Because you will have to type these names you will want them shorter than that most of the time anyway!)
7) The first character of a SAS name or variable name must be a letter. Subsequent characters must be letters, numbers, or underscores (no non-alphanumeric characters allowed, other than underscore).
8) In the Windows® version of SAS, correct syntax is color coded as blue, and comments are green.

SAS in a Windowing Environment

SAS has three main windows that we use to write code, debug, and view results. The *program editor* is where we write code, the *enhanced program editor* helps us with color coding. We view information about how the code ran and look for warnings and error messages for debugging in the *log window*, and view the output and results of our executed program in the *results viewer*. To run our code, we hit the icon of the running man in the menu bar. To run just a portion of code, we select that portion and hit the icon.



Data Entry

The Data Step

For small data sets, we can enter data by hand. A SAS data set does not live beyond your SAS session unless you export it. However, if you save your program editor, the data set will instantly be recreated each time you run the code.

The DATA statement tells SAS that we are about to enter data to create a data set called SOFA. The INPUT statement identifies the variable names for the data being entered. The dollar sign indicates that SEX will be a character variable. The DATALINES statement indicates that data entry will begin in the next line. The following lines are the actual data with the variables entered in the order shown in the INPUT statement. The semicolon at the end of the data lines indicates that data entry has been completed, and RUN initiates the execution of the data step.

```
data sofa; /* Tells SAS that we want to create a data set called SOFA */
input PT SOFA GLUCAVG SEX $; /*variable names in order, $ means character
variable*/
datalines;
104    5      328    M
106    6      234    F
107    5      243    F
108    6      199    F
109    5      234    F
110    4      410    M
112    5      308    M
113    5      304    M
115    7      328    M
116    2      209    F
118    4      252    F
119    2      239    F
120    4      223    F
121    4      270    M
122    6      274    M
125    6      139    F
126    3      267    F
128    7      277    M
129    6      267    M
130    6      221    M
;
run;
```

VIEWTABLE: Work.Sofa

	PT	SOFA	GLUCAVG	SEX
1	104	5	328	M
2	106	6	234	F
3	107	5	243	F
4	108	6	199	F
5	109	5	234	F
6	110	4	410	M
7	112	5	308	M
8	113	5	304	M
9	115	7	328	M
10	116	2	209	F
11	118	4	252	F
12	119	2	239	F
13	120	4	223	F
14	121	4	270	M
15	122	6	274	M
16	125	6	139	F
17	126	3	267	F
18	128	7	277	M
19	129	6	267	M
20	130	6	221	M

From Excel

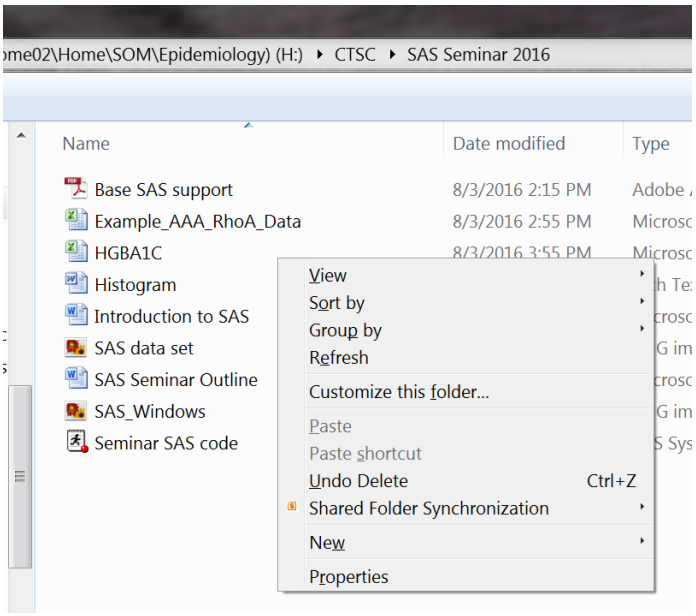
A lot of work that we do in SAS is done through *procedures*. For importing data from Excel into SAS we will use a procedure called IMPORT.

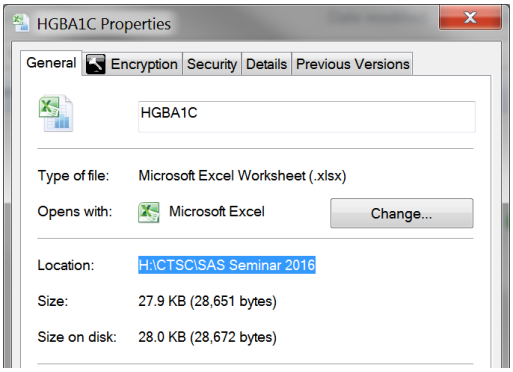
```
PROC IMPORT OUT=WORK.HGBA1C /*Call the procedure. Name Data set */
DATAFILE="H:\CTSC\SAS Seminar 2016\HGBA1C.8.25.16.xlsx" /*Path to file*/
DBMS=EXCEL REPLACE; /*File type, replace if already exists*/
RANGE="Data$"; /*Indicates name of the sheet within Excel workbook*/
GETNAMES=YES; /*Indicates first row contains variable names*/
MIXED=YES; /*Indicates both numeric and character variables in the data set*/
SCANTEXT=YES; /*Tells SAS to scan column to determine length of text*/
USEDATE=YES; /*Tells SAS to assign DATE format to date data*/
SCANTIME=YES; /* Tells SAS to assign TIME format to time data */
RUN;
```

You can also import .csv files in the same manner. The code is:

```
PROC IMPORT OUT= WORK.hgba1c;
DATAFILE= ="H:\CTSC\SAS Seminar 2016\HGBA1C.cvs"" DBMS=CSV REPLACE;
GETNAMES=YES; DATAROW=2;
RUN;
```

We can easily find the path for a file using a little trick in Windows. Go to the Excel file and right click on it. Then select "Properties". A new window will open that will show the path in a window that is copy-able. You can paste the path into your code, add a backslash, and then the name of the file:





Data Manipulation

Merging

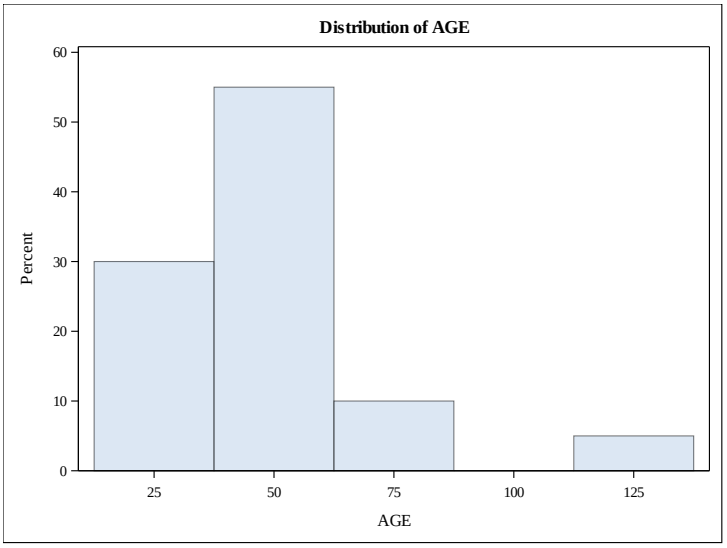
We may have to import data sets across several spreadsheets or data sources. We will want to be able to merge them. In this example we will merge the SOFA data set with the HGBA1C data set. In a merge, it is vital to use a unique identifier for each subject. That identifier must be both unique to each subject and have the same name across all data sets. In our example data sets, the unique identifier is the variable 'Pt'. When we are merging data sets, we also need to make sure they are sorted in order of the unique identifier.

```
proc sort data=sofa; /* sort data set */
by pt; /* by Pt variable */
run;
proc sort data=hgba1c;
by pt;
run;

data full; /* create new data set called FULL */
merge sofa hgba1c; /* by merging SOFA and HGBA1C */
by pt; /* using merging variable Pt */
run;
```

Subsetting and Outlier Removal

One important manipulation we may want to do frequently is subsetting of data or eliminating outliers. For example in a histogram of age we see a large outlier, and decide that this value is either a typo or this patient does not really fit with our study population and should be excluded. (PROC univariate gives us lots of output but we'll focus on the histogram and Extreme Observations table for now.)

```
proc univariate data=full;
var age;
histogram;
run;
```



Extreme Observations			
Lowest		Highest	
Value	Obs	Value	Obs
18	20	59	14
18	8	60	19
22	17	72	16
23	5	78	13
24	3	121	1

We can just set the age variable to missing for that subject.

```
data full; set full;
if age GT 78 then age= . ; /*set age values over 78 to missing */
run;
```

Or delete all subjects with extreme values.

```
data full; set full; /* keep original data set name */
if age GT 78 then delete; /*delete all obs with age > 78 */
run;
```

Other inequalities are: `GE`, `LT`, and `LE`. Similarly if you wanted to analyze only the men in the data set:

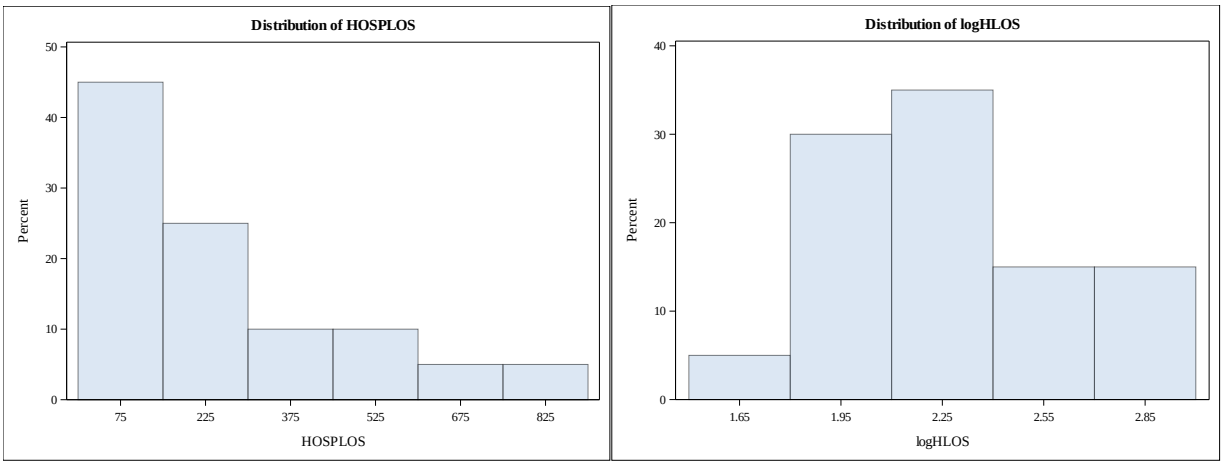
```
data full_male;
set full;
if sex = 'F' then delete;
run;
```

Numeric Functions

Often, we need to transform a variable to try to achieve an approximately normal distribution or for other reasons. We can create new variables by using SAS numeric functions.

```
data full; /* add variables to existing data set */
set full;
logHLOS = log10(hosplos); /* log base 10 transformation */
run;
```

Here are histograms of Hospital Length of Stay before and after using a log transformation:



Other SAS functions include:

```
data full;
set full;
lnHLOS = log(HospLOS); /* natural log */
HLOS_2 = hosplos**2; /* square */
sqrHLOS = sqrt(hosplos); /* square root */
expHLOS = exp(hosplos); /* exponentiated */
abs_HLOS = abs(hosplos); /* absolute value */
round_HLOS = round(hosplos, 1); /* round to nearest integer */
run;
```

Creating New Variables Using "If, Then" Statements

We can also create new variables from old ones using definitions that we are interested in. For example, perhaps we'd like to create a variable indicating whether or not the patient is overweight using the definition based on BMI. We may also want to use variables that are numerically coded, which is easier for some analyses, for example we can recode SEX to be a 0/1 variable called GENDER.

```
data full; set full;
if BMI LT 25 then overweight = 0; /* defining the NOT overweight group */
else if BMI GE 25 then overweight = 1; /* defining the overweight group */
if SEX = 'M' then Gender = 1; /* redefining SEX with numeric values */
else if SEX = 'F' then Gender = 0;
run;
```

Data Display

Continuous Variables

Once our data are in SAS we will next want to look at summary statistics and exploratory graphs (such as the histogram that we have already seen). For continuous and ordinal variables, we will want to look at means, standard deviations, medians, minimums, and maximums. Looking at a table of summary statistics will help us identify problems with the data, if any. We can also look at these summary statistics by the experimental group of interest to get a first approximation of our treatment effect. For this we will use the procedure PROC MEANS.

```
proc means DATA=full n mean median std min max; /* specifying statistics */
var age bmi hosplos loghlos glucavg sofa; /* specifying variables */
run;
```

Variable	Label	N	Mean	Median	Std Dev	Minimum	Maximum
AGE	AGE	19	43.7368421	41.0000000	17.5810571	18.0000000	78.0000000
BMI	BMI	20	26.6209998	25.4215523	6.7038277	18.4000000	41.6511705
HOSPLOS	HOSPLOS	20	257.4237611	176.6440140	208.9774855	43.7515114	800.0000000
logHLOS		20	2.2845431	2.2466915	0.3405333	1.6409931	2.9030900
GLUCAVG		20	261.3000000	259.5000000	57.4163923	139.0000000	410.0000000
SOFA		20	4.9000000	5.0000000	1.4473206	2.0000000	7.0000000

```
proc sort data=full; by sex; /* specifying sort by sex */run;

proc means DATA=full n mean median std min max;
var age bmi hosplos loghlos glucavg sofa;
by sex; /* specifying separate statistics by sex */
run;
```

SEX=F

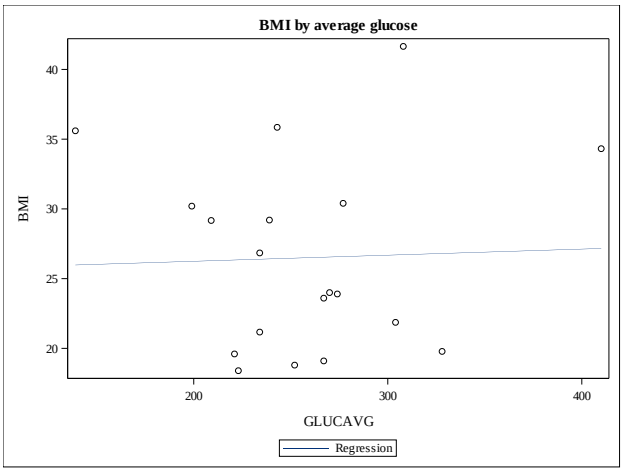
Variable	Label	N	Mean	Median	Std Dev	Minimum	Maximum
AGE	AGE	10	45.4000000	46.0000000	19.4662100	22.0000000	78.0000000
BMI	BMI	10	26.4328847	28.0064901	6.7204418	18.4000000	35.8493629
HOSPLOS	HOSPLOS	10	180.7850607	165.1217488	85.0056890	87.0651154	377.7440664
logHLOS		10	2.2189307	2.2148575	0.1885489	1.9398442	2.5771977
GLUCAVG		10	223.9000000	234.0000000	35.7380905	139.0000000	267.0000000
SOFA		10	4.3000000	4.5000000	1.5670212	2.0000000	6.0000000

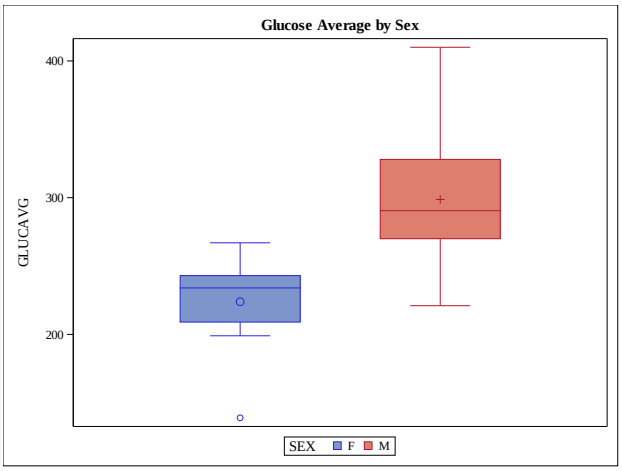
SEX=M

Variable	Label	N	Mean	Median	Std Dev	Minimum	Maximum
AGE	AGE	9	41.8888889	41.0000000	16.1821232	18.0000000	60.0000000
BMI	BMI	10	26.8091149	23.9500000	7.0450773	19.6000000	41.6511705
HOSPLOS	HOSPLOS	10	334.0624615	279.2757496	268.1742151	43.7515114	800.0000000
logHLOS		10	2.3501555	2.4092228	0.4468706	1.6409931	2.9030900
GLUCAVG		10	298.7000000	290.5000000	50.7347131	221.0000000	410.0000000
SOFA		10	5.5000000	5.5000000	1.0801234	4.0000000	7.0000000

We may also want to look at various plots. We can get a very large variety from **PROC SGPLOT**. We'll focus on scatter plots and boxplots for now.

```
title "BMI by average glucose"; /* adds a title to the plot */
proc sgplot data=full;
reg y=bmi x=glucavg; /*scatter plot with a regression line */
run;

title "glucavg by Sex"; proc sgplot data=full;
vbox glucavg / group=sex; /* specifies vertical boxplot, grouping by sex */
run;
```





Categorical Variables

For categorical variables we will want to look at proportions and frequencies (counts) for different groups for the variables in our data set. For this we use the procedure **FREQ**.

```
PROC FREQ data=full; table overweight*sex; run;
```

Contingency Table of Overweight by Sex

Table of overweight by SEX			
overweight	**SEX**		
Frequency Percent Row Pct Col Pct	**F**	**M**	**Total**
0	4 20.00 40.00 40.00	6 30.00 60.00 60.00	10 50.00
1	6 30.00 60.00 60.00	4 20.00 40.00 40.00	10 50.00
Total	10 50.00	10 50.00	20 100.00

We can make our tables look better by using **PROC FORMAT** and labeling. We create labels in the data step. These labels will then be used in tables and graphs.

```
proc format;
value Overweight 1='Yes' /* Changing numeric coding to words for tables */
                 0='No';
value Gender     0 = 'Female'
                 1 = 'Male'; run;
```

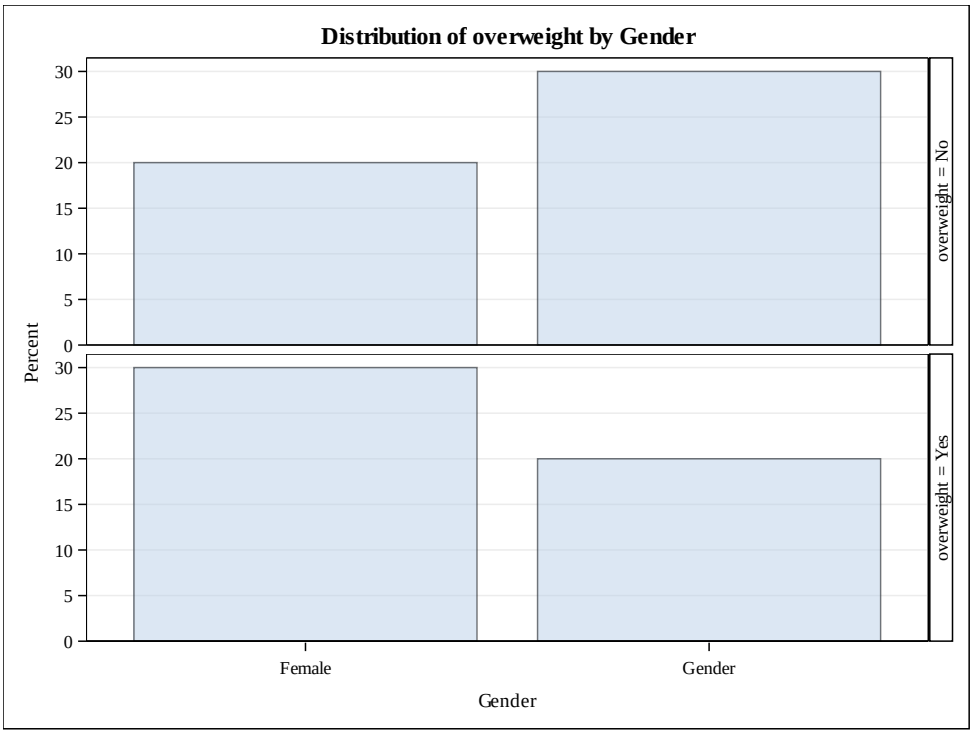
```
data full; set full;
label glucavg = "Average Glucose During Hospital Stay";
run;

proc means data=full; var glucavg; run;
```

Analysis Variable : GLUCAVG Average Glucose During Hospital Stay				
N	Mean	Std Dev	Minimum	Maximum
20	261.3000000	57.4163923	139.0000000	410.0000000

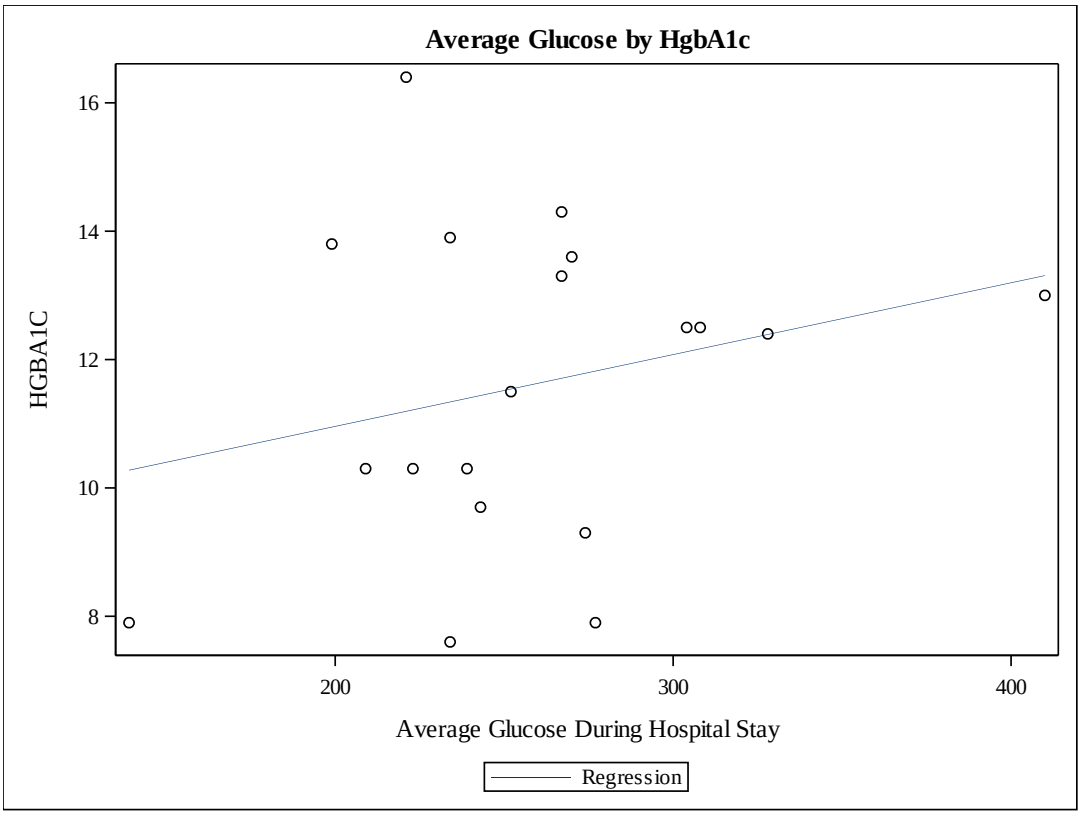
PROC FREQ will also give us frequency plots in addition to frequency tables:

```
proc freq data=full;
format Gender gender. overweight overweight.;
table overweight*Gender / plots(only)=(freqplot(scale=pct));
run;
```

Table of overweight by Gender			
overweight	Gender		
Frequency Percent Row Pct Col Pct	Female	Male	Total
No	4 20.00 40.00 40.00	6 30.00 60.00 60.00	10 50.00
Yes	6 30.00 60.00 60.00	4 20.00 40.00 40.00	10 50.00
Total	10 50.00	10 50.00	20 100.00




```
title "Average Glucose by HgbA1c";
proc sgplot data=full; reg y=hgba1c x=glucavg; run;
```



Getting Output Out of SAS

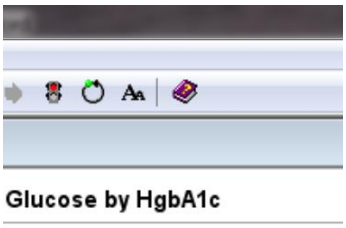
While we're running SAS we view all our results and output in the results viewer, but we will want to save it eventually for use in articles and reports. Getting the results of our analyses, graphs, and tables out of SAS and into Word® or a PDF file is easy.

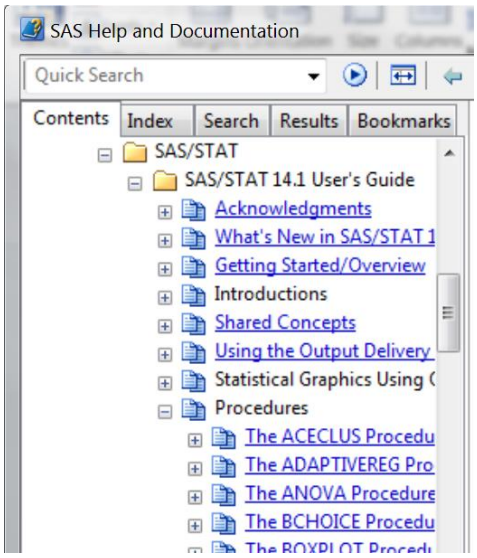
```
ods rtf file= "H:\CTSC\SAS Seminar 2016\Glucose_by_HgbA1c.rtf";
title "Average Glucose by HgbA1c";
proc sgplot data=full;
reg y=hgba1c x=glucavg;
run;
ods rtf close;
```

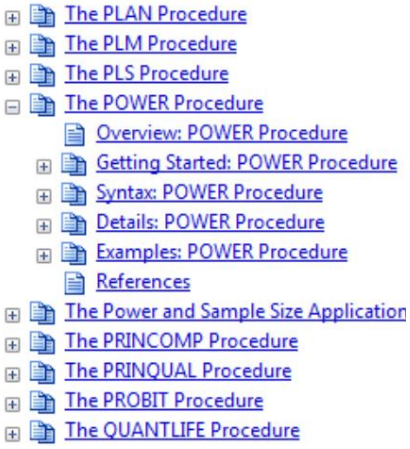
You will, of course, use the path to the directory you want to save your file to in place of the path shown. You can find the path using the simple trick for importing files, but just type what you'd like the name to be after the last back slash. After this code is run, an *rtf* file will open and be saved to the directory you specified. This file can then be converted to a Word document or a PDF file.

Getting Support for SAS

There is documentation for most procedures available right in SAS. If you click on the little purple book with the yellow question mark, then go to SAS Products, then scroll all the way down to SAS/STAT, then SAS/STAT 14.1 User's Guide, then Procedures, you can click on the procedure you'd like documentation for. For example, if you'd like to learn to use PROC POWER for sample size calculations, you would click on "The Power Procedure". You can then click on the items in the list below "The Power Procedure" to see an overview, the basic syntax, and examples.







Additionally, the University of California, Los Angeles provides primers on a variety of statistical analyses and software packages: http://www.ats.ucla.edu/stat/sas/

If you Google "sample size calculation SAS" for example, you will see the following link in the search results: http://www.ats.ucla.edu/stat/sas/dae/t_test_power2.htm as well as other links for ANOVA, etc.

There is also online documentation: https://support.sas.com/documentation/onlinedoc/stat/ as well as many white papers and reports published by SAS that will usually appear in the search results in links with sas.com in them. For example: http://www2.sas.com/proceedings/sugi29/195-29.pdf. These usually contain detailed explanations and sample code that goes beyond what is found in the SAS documentation in-package.

Finally, there are online support groups such as: https://communities.sas.com/ where you can ask questions and see what questions others have asked and answered.

References

Delwiche, Lora; Slaughter, Susan; **The Little SAS® Book**: **A Primer**, Fifth Edition, Copyright© 2012, SSA Institute Inc., Cary, North Carolina, USA. Also available online: https://www.sas.com/storefront/aux/en/splsb/65423_excerpt.pdf

SAS Institute Inc. 2001 *Step-by-Step Programming with Base SAS® Software*. Cary, NC: SAS Institute Inc. http://support.sas.com/documentation/cdl/en/basess/68381/PDF/default/basess.pdf

SAS® Support Online Documentation: https://support.sas.com/documentation/onlinedoc/stat/