SAS Guide With Examples

Table of Contents

Section 1 Preliminaries

1.1 Getting Started With SAS

One of the first things to consider prior to creating a data set is to know what information you would like conveyed. The biggest problem that researchers have is coming up with a clear and concise question and sticking to it. Under the assumption that the data of interest has already been obtained through a primary source (your own data collection technique) or a secondary source (another investigators data or maybe data on the internet), you now must be able to bring this data into a usable format. This is referred to as creating a data set. The following steps are helpful:

Using SAS

First, name the data set you would like to create.

```
data one;
```

The following data set is given the name, one. The data set, one, is empty for the moment until actual raw data can be brought in.

Before going any further, one important thing to remember when using SAS is to put a semicolon after each statement. This can get tedious at times, but is something that should be kept in mind before starting anything in SAS.

In order to bring in your data a variety of possibilities are available. One method is to keypunch your data directly into SAS. Before you start you must use the input command as in the following line:

```
input Sub_name $ weight age;
```

The input command lets SAS know not only the variable name for each of the columns of your data but also the type of data you will be bringing in. The two types of data that we will be concerned with is character and numeric. From the input line, it is easily noticed that Sub_name is your first column variable, and will refer to the subject's name. Since names are character data, we need to let SAS know this. This is accomplished with the use of the ($). Now SAS knows it will be reading in character or alpha numeric data for the variable Sub_name. The other variables: height, weight, and treat, will be considered numeric. One thing to note is that SAS is capable of reading in format types other than character and numeric data. It can read in dates for example, but we will leave that issue for a later discussion. Once SAS knows what type of data you will be using, and before the keypunching process can begin a datalines statement is required.

```
datalines;
```

The datalines statement lets SAS know where the raw data exists and will be read from. After the datalines statement, you can begin the keypunching process using a space as a delimiter, which allows SAS to distinguish between the different variables you are punching in data for. The following is an example of some observation lines:

```
JaySmith    125 22
MaryCarr    115 20
JoeJones    177 46
;
```

Notice that the data after being keypunched in was followed by a semicolon. This indicates that there will be no more records entered and that SAS has reached the end of the raw data set. Once these steps have been completed procedure statements called proc statements can be used to view, manipulate, or analyze the data. Keep in mind that a "run" statement usually follows all proc statements. You can use the following proc statements to view the contents of what SAS has read into data set one:

```
proc print;
run;
```

If you are using the windows version of SAS, just point to the running man on the toolbar and click. This is an execute command that runs the print procedure.

One thing to be aware of is that SAS by default will read only up to eight character values. Some names longer than eight characters will get cut off at the eighth letter or character. This problem is easily handled by specifying the number of characters that should be allotted to accommodate the length of the longest observation being entered. For example, assume that you would like to read in the following data set:

```
Jay Smith      125 22
Mary Carr      115 20
Joe Jones      177 46
Rachel Lonely 105 27
;
```

Now we have two problems that must be addressed. The first is that the subjects' names are more than eight characters in length. The other is that there are spaces between the first and last names of each of the subjects. This is a problem because SAS uses a space as a delimiter. When SAS encounters the space between the names, it assumes that the next variable is being addressed which is not the case in our example. The following input statement corrects our problem.

```
input Sub_name $ 1-13 weight age;
```

By specifying the number of spaces needed to accommodate the longest name, our problem is solved. Specifying the number of spaces needed to read in data can be used for both character and numeric data as follows:

```
input patient $ 1-13 weight 15-17 age 19-20;
```

Now try executing the following statements:

```
data one;
  input patient $ 1-13 weight 15-17 age 19-20;
datalines;
Jay Smith     125 22
Mary Carr     115 20
Joe Jones     177 46
Rachel Lonely 105 27
;
proc print;
run;
```

This is the output you should have obtained:

The SAS System

Obs	Sub_name	weight	age
1	Jay Smith	125	22
2	Mary Carr	115	20
3	Joe Jones	177	46
4	Rachel Lonely	105	27

Now let's explore some other ways of bringing in data.

Assume that the same information is saved as a text file, "example.txt", in ASCII format in the following location "C:\Temp" on your PC. This data set can be read using an ASCII editor like notepad. The following is a copy of the contents stored in the file "example.txt":

```
Jay Smith     125 22
Mary Carr     115 20
Joe Jones     177 46
Rachel Lonely 105 27
```

Instead of copying and pasting this information from notepad to SAS, the data stored in "example.txt" can be brought in using the following set of statements:

```
data one;
  infile 'C:\Temp\example.txt';
  input patient $ 1-13 weight 15-17 age 19-20;
proc print;
run;
```

The infile statement gives SAS the location of text file where the data of interest in stored. Notice that the same input statement is used as if the data were keypunched directly into SAS.

Now, assume that the same information is stored as an EXCEL file, example.xls, in the following location "C:\Temp" on your PC. Although EXCEL is a good spreadsheet

program, it is not very accommodating if complex analysis of you data is required. Therefore importing your data from EXCEL into SAS is often necessary, and can be done easily with the following procedure statements:

```
proc import datafile = 'C:\Temp\example.xls' out = exdata1
  replace;

proc print;
run;
```

The statement proc import allows the SAS user to import data from an EXCEL spreadsheet into SAS. The datafile statement provides the reference location of the file. In this case, the file "example" with the extension ".xls" to denote an EXCEL file is the file we would like to import. The out statement is used to name the SAS data set that has been created by the import procedure. Notice that print procedure has been utilized once more in order to view the contents of the SAS data set exdata1.

Although we have mentioned character and numeric data types, we have yet to discuss formatting numeric data, and how to work with variables that have dates as observations.

Looking at the following data set may help us understand the necessary steps involved when working with dates as observations and formatting numeric data. The columns in this data refers to the name of the subject, the date they were observed, their date of birth, and their weight in pounds, which is to consist of two digits behind a decimal point.

```
Mary   11/12/2002 06/05/78 12567
Joe    05/14/2001 07/08/67 15634
James  01/09/2002 02/28/64 16790
```

In this case the researcher would like to create a SAS data set using this information. Later, the researcher would like to utilize the information consisting of the date of observation and the date of birth to acquire the age of each subject. The researcher also notices that the decimal point in the data for the weight of the subjects is missing. Although these problems may seem complex to the novice SAS user, they can easily be remedied with the use of "informat" statements. Informat statements allow the user to specify the format of the variables at the input stage of the SAS code. This basically lets SAS know what type of data it is reading in before it even reads it in.

The following is an input statement that contains informats and other symbols that we have yet to discuss:

```
input sub $ 1-5 @8 obs_date MMDDYY10. @19 dob MMDDYY8. @28 weight d5.2;
```

The informat "MMDDYY10." is commonly used when reading in dates. Notice that the number of characters for the date of observation is ten, eight numeric values and the two

forward slashes. For the date of birth, there are eight characters, six numeric and the two forward slashes. The variable weight uses the informat "d5.2" which informs SAS to read in five values with two behind a decimal point. Something else that should look different is the "@" followed by a number in front of the variable names. The "@" indicates to SAS where to begin reading the variable observations that are being inputted. For example "@8", refers to character column 8, so SAS will begin reading "obs_date" data at column eight.

At this point, we are ready to create a data set; let's call it testing, and it should look something like the following:

```
data testing;
  input sub $ 1-5 @8 obs_date MMDDYY10. @19 dob MMDDYY8. @28 weight
d5.2;
datalines;
Mary   11/12/2002 06/05/78 12567
Joe    05/14/2001 07/08/67 15634
James  01/09/2002 02/28/64 16790
;
proc print;
run;
```

After running this code, you should obtain the following output:

The SAS System

Obs	sub	obs_date	dob	weight
1	Mary	15656	6730	125.67
2	Joe	15109	2745	156.34
3	James	15349	1519	167.90

One thing that should be readily noticeable is that numeric values are given in place of the dates. Those numeric values represent the number of days after January 1, 1960, which is the year cutoff for SAS. Also note that the weight of the subjects is in the desired format.

The calculation of the age for each subject is now relatively easy. By adding one line to the previous SAS code, we can subtract the observation date from the date of birth and divide by 365.25. I use 365.25 because it takes into account the fact that every four years is a leap year. The results we desire should be provided with the following SAS code:

```
data testing;
  input sub $ 1-5 @8 obs_date MMDDYY10. @19  dob MMDDYY8. @28 weight
d5.2;
age = (obs_date - dob)/365.25;
datalines;
```

```
Mary   11/12/2002 06/05/78 12567
Joe    05/14/2001 07/08/67 15634
James  01/09/2002 02/28/64 16790
;
proc print;
run;
```

Notice that the line containing age is the only difference between this piece of SAS code and the one previously written. The following is the output that should have been obtained:

The SAS System

Obs	sub	obs_date	dob	weight	age
1	Mary	15656	6730	125.67	24.4381
2	Joe	15109	2745	156.34	33.8508
3	James	15349	1519	167.90	37.8645

1.2 More on Data Management by Example

The following tutorial allows you to go through some steps that will help you get better acquainted with data management using SAS. First locate the file "anthros.txt" on the diskette provided or on your hard drive. The file should contain the following lines of text:

```
The following is a list of five individuals along with their
corresponding height in inches and weight in pounds.

Bruce Burks   66 128
Janie Johnson 62 135
Joe Jacobs    72 157
Jim Lewis     70 186
Linda Carol   60 119
```

There are quite a few things to consider before reading in this raw data. The first thing that you should notice is that the first two lines give you information about your data set. The third line is a blank line, and the data of interest starts on the fourth line. Unfortunately, SAS cannot distinguish between text that is giving information about the data and the data itself; however, this dilemma can easily be resolved in the following manner:

```
data anthro;
  infile 'C:\Temp\anthros.txt' firstobs = 4;
  input Name $ 1-13 Height 15-16 Weight 18-20;
```

Here the data set that was created was named bmi. Next we used an INFILE statement to bring in the raw data from either a diskette or our hard drive. Notice the option in the INFILE statement "firstobs". The "firstobs" option in the INFILE statement basically alerts SAS to the fact that the data that should be read begins on the "nth" line. In our example, the nth line is the fourth line, where "n" is restricted to the positive integers. The "firstobs" option is extremely useful especially when you acquire data sets that have extraneous information and want to jump directly into the analysis phase.

For the sake of completeness, let's take another example. The following text file contains the exact same information as previously given the only difference is that it's arranged differently.

```
Bruce Burks   66 128
Janie Johnson 62 135
Joe Jacobs    72 157
Jim Lewis     70 186
Linda Carol   60 119

This is a list of five individuals along with their corresponding
height in inches and weight in pounds.
```

Here the first line contains the data we are interested in; however, if we try to "infile" this data into SAS, the problem occurs in the last two lines. Without specifying where SAS should stop reading the data, SAS will treat all the lines as if they were data that you would like read. We can take care of this problem without modify the text file by using the following command lines:

```
data anthro;
  infile 'C:\Temp\anthros.txt' obs = 5;
  input Name $ 1-13 Height 15-16 Weight 18-20;
```

In this case we are using the "obs" option, which alerts SAS to stop from reading any more lines of data after the nth line has been read. In our example, we want SAS to stop reading data after the fifth line has been read. You can easily view the contents of what SAS has read in by using the "proc print" command followed by a "run" command. The block of code should look like the following:

```
data anthro;
  infile 'C:\Temp\anthros.txt' obs = 5;
  input Name $ 1-13 Height 15-16 Weight 18-20;
proc print;
run;
```

In order to appreciate the significance of what has been learned so far, let's extend our example into one that is a bit more complex. Let's assume that the same researcher has assigned two employees into the field in order to gather more data on the anthropometrics of the individuals eating at fast food restaurants.

The first employee returns with the following data set stored in the text file "anthros1.txt":

```
I went to the following places: BurgerKing, McDonalds, and Pizza Hut.
I randomly chose five individuals and collected their name, height in
inches, and weight in pounds.  These are my results:

Brian Long    72 208
Jim Branda    69 200
Brenda Green  59 157
Jo Ling       62 112
Daren Woods   68 156
```

The researcher, after opening the file, knows that the following SAS code is needed in order to read in the data:

```
data one;
  infile 'C:\Temp\anthros1.txt' firstobs = 5;
  input Name $ 1-12 Height 14-15 Weight 17-19;
```

Now the second employee arrives and gives the researcher his data set named "anthros2.txt" containing the following information:

```
Jim Jacobs   69 165
Wendall Lee  70 220
Pat Jones    60 143
Nel Henry    73 210
Jeremy Jacks 67 182

I went to Wendy's and Jack-in-the-Box.  Five individuals were chosen at
random.
```

Again, the researcher, after opening the file, knows that the following SAS code is needed in order to read the data in:

```
data two;
  infile 'C:\Temp\anthros2.txt' obs = 5;
  input Name $ 1-12 Height 14-15 Weight 17-19;
```

Now that the researcher has these two data sets, she would like to bring them together to form one big data set. This is called concatenation and SAS is capable of concatenating data sets provided the variable names and content coincide among data sets. The following code illustrates how this is done:

```
data one;
  infile 'C:\Temp\anthros1.txt' firstobs = 5;
  input Name $ 1-12 Height 14-15 Weight 17-19;
run;


data two;
  infile 'C:\Temp\anthros2.txt' obs = 5;
  input Name $ 1-12 Height 14-15 Weight 17-19;
run;


data three;
  set one two;
proc print;
run;
```

The "set" command is critical here because it allows you to set data sets one and two into a third data set called three. The results can be seen in the output after running "proc print":

```
                          The SAS System

Obs    Name            Height    Weight
  1    Brian Long        72        208
```

```
 2    Jim Branda        69        200
 3    Brenda Green      59        157
 4    Jo Ling           62        112
 5    Daren Woods       68        156
 6    Jim Jacobs        69        165
 7    Wendall Lee       70        220
 8    Pat Jones         60        143
 9    Nel Henry         73        210
10    Jeremy Jacks      67        182
```

Finally, we will end this discussion with merging data sets. Let's assume that a third employee hired by the researcher had accompanied the other two employees and recorded the ages of the individuals who were interviewed. However, this information was not put into data set three, the previous data set that was formed. First, let's look at the contents of the text file named "ages.txt" submitted by the third employee:

```
Brian Long   23
Jim Branda   64
Brenda Green 51
Jo Ling      20
Daren Woods  33
Jim Jacobs   47
Wendall Lee  55
Pat Jones    22
Nel Henry    19
Jeremy Jacks 44
```

Since the goal is to merge data set three with data set ages, the merge can be accomplished but a "by variable" must be specified. The "by variable" is usually information on observations that is common to both data sets. In this case, the only choice for a "by variable" would be the variable "Name", since the names are given in both sets of data. Before we proceed, it should be mentioned that the data sets must be sorted by the "by variable", in this case "Name", prior to the merge. The following command lines illustrate this procedure:

```
data one;
  infile 'C:\Temp\anthros1.txt' firstobs = 5;
  input Name $ 1-12 Height 14-15 Weight 17-19;
run;

data two;
  infile 'C:\Temp\anthros2.txt' obs = 5;
  input Name $ 1-12 Height 14-15 Weight 17-19;
run;

data three;
  set one two;
proc sort;
  by Name;
run;

data four;
  infile 'C:\Temp\ages.txt';
  input Name $ 1-12 Age 14-15;
```

```
proc sort;
  by Name;
proc;
run;

data five;
  merge three four;
  by Name;
proc print;
run;
```

If you will notice, the data sets of interest are three and four since they contain the information that we would like merged. The previous two data sets, data sets one and two, were also included for completeness since they were utilized in the creation of data set three. After running this procedure, the output obtained should appear as follows:

The SAS System

Obs	Name	Height	Weight	Age
1	Brenda Green	59	157	51
2	Brian Long	72	208	23
3	Daren Woods	68	156	33
4	Jeremy Jacks	67	182	44
5	Jim Branda	69	200	64
6	Jim Jacobs	69	165	47
7	Jo Ling	62	112	20
8	Nel Henry	73	210	19
9	Pat Jones	60	143	22
10	Wendall Lee	70	220	55

Notice that the last column contains the information on the ages of each of the individuals observed.

Section 2 Parametric Tests

2.1 Getting Acquainted with the T-test

This tutorial will introduce you to t-tests, ANOVA (Analysis of Variance), and basic linear regression. Let's begin with a t-test.

One common application of the t-test is to test whether the means of two different variables, measured over the same set of observations, have a significant difference. This is basically the same thing as saying does the mean of the difference between the two variables of interest, over the same set of observations, differ significantly from zero. This can best be illustrated via an example. Take for instance the data collected on individual's heights and weights from different fast food restaurants. The following information comes from the first employee:

```
Brian Long   72 208
Jim Branda   69 200
Brenda Green 59 157
Jo Ling      62 112
Daren Woods  68 156
```

And the next piece of information comes from the second employee:

```
Jim Jacobs   69 165
Wendall Lee  70 220
Pat Jones    60 143
Nel Henry    73 210
Jeremy Jacks 67 182
```

Let's assume that we have read the tutorial on data set management, or have some familiarity with SAS already. We use our techniques to concatenate the two data sets and obtain the following data set:

```
Brian Long   72 208
Jim Branda   69 200
Brenda Green 59 157
Jo Ling      62 112
Daren Woods  68 156
Jim Jacobs   69 165
Wendall Lee  70 220
Pat Jones    60 143
Nel Henry    73 210
Jeremy Jacks 67 182
```

Now we would like to distinguish between the two groups. One way to do this is by adding another variable called Group. Group can be labeled "1" for the first set of five observations and "2" for the second set.

Before we continue we should save the information we have as text in a temporary file in our C-drive with the name "onettest". The ".txt" is implied so when saving you shouldn't have to add it on. Now, the following SAS statements can be used to create the data set "ttesting":

```
data ttesting;
  infile 'C:\Temp\onettest.txt';
  input Name $ 1-12 Height 14-15 Weight 17-19 Group 21-21;
```

Let's assume that the researcher would like to see if there is a significant difference in the mean weight of individuals interviewed from each of the restaurants by his employees. The following commands can be used to run a t-test:

```
proc ttest;
  class Group;
  var Weight;
run;
```

The "class" command contains the variable that we are using to distinguish between the two groups of observations collected by each employee. Since we are interested in comparing the weights of the two groups, we use the "var" command with the specified variable of interest. In this case, the variable "Weight" as shown above. One more thing that should be noted is that although we have the same number of individual observation for each group this is not a paired t-test. You should run a pair t-test only when there is some logical reasoning for pairing the variables of interest. One case when pairing is required is when one observation is dependent on another observation in order to obtain the correct information desired. For example, Assume you have a group of student and that you give them a pre-test at the beginning of the semester and a post-test at the end. Now, you are interested to see if the course increased their knowledge. The best way to do this is to pair the post-test to the pre-test of each student, and examine the difference in his or her scores. An example will be given later to demonstrate the details.

Now getting back to our example, we would like to see what kind of output SAS will provide us with. Therefore after running the previous commands we acquire the following results:

The SAS System
The TTEST Procedure

Statistics

Variable	Class	N	Lower CL Mean	Mean	Upper CL Mean	Lower CL Std Dev	Std Dev	Upper CL Std Dev	Std Err
Weight	1	5	118.45	166.6	214.75	23.234	38.779	111.43	17.342
Weight	2	5	144.65	184	223.35	18.989	31.694	91.074	14.174
Weight	Diff (1-2)		-69.05	-17.4	34.249	23.921	35.414	67.845	22.398

T-Tests

Variable	Method	Variances	DF	t Value	Pr > |t|
Weight	Pooled	Equal	8	-0.78	0.4596
Weight	Satterthwaite	Unequal	7.7	-0.78	0.4605

Equality of Variances

Variable	Method	Num DF	Den DF	F Value	Pr > F
Weight	Folded F	4	4	1.50	0.7054

One of the problems that most individuals complain about when using SAS is that it gives too much output, or output that most individuals find extraneous. Although this is true most of the time, the output displayed here is very useful. However, we are interested in the Weight Diff, which is the difference in the means of the two weights. We would like to know if the difference is significantly different from zero. Under the null hypothesis, we are assuming no significant difference. Also, notice that SAS provides the statistics for the assumption of equality of variance. In this case with a p-value of 0.7054, we cannot reject the null hypothesis that the variances for the two groups are equal. Now, observing the t-test with the pooled method, we obtain a p-value of 0.4596 and thus cannot reject the null hypothesis of no difference between the mean weights of the two groups. One more thing that should be mentioned is that the hypothesis test we conducted was a two-tailed test. Basically what this means is that we began with the null hypothesis of the mean difference being equal to zero, and the alternative hypothesis that the mean difference is not equal to zero. Since no directionality is implied meaning that one group is not expected to have a higher mean than the other, we refer to our test as a two-tailed test rather than a one-tailed test.

Now let's explore another scenario. The next data set was formed after the individuals who were interviewed regarding their height and weight were exposed to a six-week diet intervention. Their weights were recorded prior to the six-week period, and after they completed the intervention. The following is a copy of the data set after this information was collected:

```
Brian Long   72 208 200
Jim Branda   69 200 198
Brenda Green 59 157 160
Jo Ling      62 112 110
Daren Woods  68 156 150
Jim Jacobs   69 165 168
Wendall Lee  70 220 219
Pat Jones    60 143 140
Nel Henry    73 210 204
Jeremy Jacks 67 182 183
```

Now the researcher is interested to see if the intervention may have had an impact on the weight of the individuals. The following SAS code can be used to run a paired t-test in order to acquire the desired information:

```
data ttesting;
  infile 'C:\Temp\pairttest.txt';
  input Name $ 1-12 Height 14-15 Weight1 17-19 Weight2 21-23;
```

```
proc ttest;
  paired Weight1*Weight2;
run;
```

The corresponding output looks like the following:

```
                                The SAS System
                            The TTEST Procedure

                                Statistics

                        Lower CL           Upper CL  Lower CL           Upper CL
Difference            N      Mean    Mean      Mean   Std Dev  Std Dev   Std Dev  Std Err
Weight1 - Weight2    10    -0.607     2.1    4.8072    2.6031   3.7845     6.909   1.1968

                                T-Tests

                    Difference             DF    t Value    Pr > |t|
                    Weight1 - Weight2       9       1.75      0.1132
```

Notice here that once again we cannot reject the null hypothesis, and must conclude that the intervention had no significant impact on weight.

2.2 ANOVA by Example

Next, we look at example demonstrating the use of ANOVA. Let's assume that the researcher has data on individuals from three different diet camps. All the researcher is concerned with is seeing whether the mean weights of the individuals in each camp are significantly different from one another. Since we are comparing three different means, we must employ the use of ANOVA. The proceeding lines represent the SAS code and data for this example.

```
data expanova;
  input group weight;
datalines;
1 223
1 234
1 254
1 267
1 234
2 287
2 213
2 215
2 234
2 256
3 234
3 342
3 198
3 256
3 303
;

proc anova;
  class group;
  model weight = group;
  means group;
run;
```

The following is the corresponding output to the previous lines of code.

```
                                The SAS System
                              The ANOVA Procedure

Dependent Variable: weight

                                        Sum of
        Source                  DF        Squares     Mean Square    F Value    Pr > F
        Model                    2     2071.60000      1035.80000       0.69    0.5201
        Error                   12    17998.40000      1499.86667
        Corrected Total         14    20070.00000

                      R-Square     Coeff Var      Root MSE    weight Mean
                      0.103219      15.49124      38.72811       250.0000

        Source                  DF        Anova SS     Mean Square    F Value    Pr > F
        group                    2     2071.600000     1035.800000       0.69    0.5201
```

Here we see that because the model is not significant (p-value = 0.5201), we must conclude that the mean weights of the three groups are not significantly different.

2.3 Linear Regression by Example

Finally, let's end our discussion with basic linear regression. The researcher has information on the distance individuals live from fast-food places and the number of times they eat at those places per week. The researcher wants to see if there is a relationship between these to variables in order to assess if location of residence is a good predictor of the number of times individuals eat at fast-food places per week. The following is a view of the raw data saved as "reg.txt" in a Temp file on the C-drive. The first column represents the number of miles from the fast-food place (miles) and the second column is the number of times per week the individual eats at the fast-food place (eat_times). Next, you will see the SAS code used to access the data and run the linear regression.

```
0.34 5
1.23 3
6.78 1
5.54 1
0.43 4
1.18 4
2.34 2
0.98 4
3.34 2
1.20 4
```

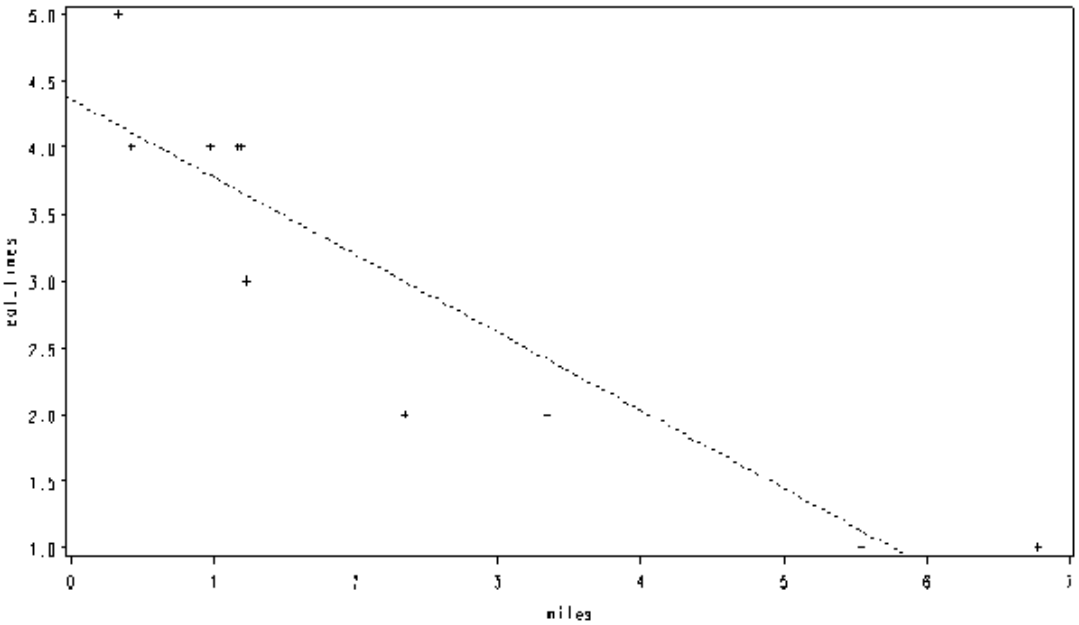
```
data fastfood;
  infile 'C:\Temp\reg.txt';
  input miles eat_times;
proc reg;
  model eat_times = miles;
  plot eat_times*miles;
run;
```

Finally, the SAS output is shown with a plot of the data.

```
                                The SAS System
                              The REG Procedure
                               Model: MODEL1
                        Dependent Variable: eat_times

                                Analysis of Variance

                                    Sum of           Mean
Source                      DF      Squares         Square    F Value    Pr > F

Model                        1      15.07435       15.07435      41.22    0.0002
Error                        8       2.92565        0.36571
Corrected Total              9      18.00000


                Root MSE              0.60474    R-Square     0.8375
                Dependent Mean        3.00000    Adj R-Sq     0.8171
                Coeff Var            20.15788
```

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	4.36223	0.28564	15.27	<.0001
miles	1	-0.58315	0.09083	-6.42	0.0002



In this example the number of times individuals ate at the fast-food places decreased with the number of miles they live from the fast-food place. The model is significant with a p-value = 0.0002 and the relationship is strong since R-squared = 0.8375.

Section 3 NonParametric Tests

3.1 Wilcoxon Rank Sum Test and Kruskal-Wallis Test by Example

This is a tutorial on nonparametric tests. The biggest difference between a parametric and nonparametric test is the fact that a parametric test assumes that the data under investigation is coming from a normal distribution. The SAS software provides several nonparametric tests such as the Wilcoxon rank-sum test, which is equivalent to the Mann-Whitney test for two samples, and the Kruskal-Wallis test when dealing with two or more samples.

Consider the following example, Researcher Bob is interested in testing the difference between the effectiveness of two allergy drugs out on the market. He would like to administer drug A to a random sample of study subjects and then drug B to another random sample who suffer from the same symptoms as those individuals taking drug A. Researcher Bob would like to see if there is a difference between the two groups in the time, in minutes, for subjects to feel relief from their allergy symptoms.

Table 1. The time, in minutes, until relief is felt after drug has been administered for study groups A and B.

Subject	Drug A	Drug B
1	43	28
2	40	33
3	32	48
4	37	37
5	55	40
6	50	42
7	52	35
8	33	43

In order to do some analysis the first step would be to form a SAS data set as follows:

```
data drugtest;
input subject drug_group $ time;
datalines;
1  A 43
2  A 40
3  A 32
4  A 37
5  A 55
6  A 50
7  A 52
8  A 33
9  B 28
10 B 33
11 B 48
12 B 37
13 B 40
14 B 42
15 B 35
16 B 43
```

```
;
proc means median min max;
  by drug_group;
  var time;

proc npar1way wilcoxon;
  class drug_group;
  var time;
run;
```

In order to perform these nonparametric tests, we use the procedure "npar1way" and the option "wilcoxon" to request that particular test. Note that "wilcoxon" is not the only option available. Other options are "median", "mood", and "exact", but for our purposes we will not discuss these options at this time. In the case above, we have demonstrated the Wilcoxon Rank Sum Test in order to test if a significant difference exists between the drug groups in median time to relief from allergy symptoms. The following is the output obtained from the SAS code:

```
---------------------------------------- drug_group=A
                                  Analysis Variable : time

                        Median         Minimum         Maximum
                     ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
                       41.5000000      32.0000000      55.0000000
                     ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

---------------------------------------- drug_group=B
                                  Analysis Variable : time

                        Median         Minimum         Maximum
                     ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
                       38.5000000      28.0000000      48.0000000
                     ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ


                                The NPAR1WAY Procedure

                    Wilcoxon Scores (Rank Sums) for Variable time
                          Classified by Variable drug_group

          drug_                 Sum of      Expected       Std Dev          Mean
          group       N         Scores      Under H0      Under H0         Score
          ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
          A           8          77.0          68.0      9.493858        9.6250
          B           8          59.0          68.0      9.493858        7.3750

                         Average scores were used for ties.
                              Wilcoxon Two-Sample Test

                                Statistic             77.0000

                                Normal Approximation
                                Z                      0.8953
                                One-Sided Pr >  Z      0.1853
                                Two-Sided Pr > |Z|     0.3706

                                t Approximation
                                One-Sided Pr >  Z      0.1924
                                Two-Sided Pr > |Z|     0.3848

                           Z includes a continuity correction of 0.5.
                                    Kruskal-Wallis Test
                                Chi-Square             0.8987
```

DF	1
Pr > Chi-Square	**0.3431**

The first thing that you should notice is that some descriptive statistics where given prior to the test of median differences. This is mainly because procedure "npar1way" displays the mean not the median for each drug group. Finally, we are interested in observing the output given under the heading Normal Approximation. The probabilities given are those associated with the Mann-Whitney test. Because of the way this hypothetical problem is presented, a two-tailed test should be employed mainly because the directionality of the outcome is not known (you don't know which drug works better). In any case, because the probabilities are above 0.05, you cannot reject the null hypothesis and must conclude that there is no difference between the median times to relief for both drug groups.

Section 4 Logistic Regression

4.1 Logistic Regression by Example

The following is a tutorial on logistic regression. There are several books that discuss logistic regression in detail, but for our purposes we'll just skim the surface. Logistic regression is a powerful tool; however, knowing when to use it is the most important thing. Fundamentally, it is said that logistic regression can be used when you have an outcome variable that is dichotomous. Although that is usually the case, it isn't the only time. You can use logistic regression with polychotomous and ordinal data also.

Before we begin, let's discuss the linear regression model in order to gain a better understanding of the differences between linear and logistic regression. The standard assumptions of linear regression using ordinary least squares are as follows:

- The model has the form: $y_i = b_0 + b_1x + e_i$
- The expected value of $e_i = 0$
- The variance of e_i is constant
- The e_i's are independent and identically distributed as a Normal distribution

For logistic regression, these assumptions change in the following manner:

- The expected value of $e_i = p_i$
- The variance of $e_i = p_i(1 - p_i)$
- The e_i's are binomially distributed

It is critical that we look at these assumptions in order to get a better understanding of our analysis. Now let's begin with a simple example in order to introduce the topic in better detail. Assume that Researcher Smith has conducted a study involving suicide and depression. Researcher Smith is interested in this relationship between suicide and depression because he would like to show that individuals who are depressed are more likely to attempt suicide. The following data was collected via surveys from a psychiatric ward in smalltown. Note that "1" for depression indicates that the individual was considered clinically depressed because they scored above a "5" on a depression criteria scale. A "1" for suicide indicates that the individual attempted suicide. Data on 20 individuals is given below.

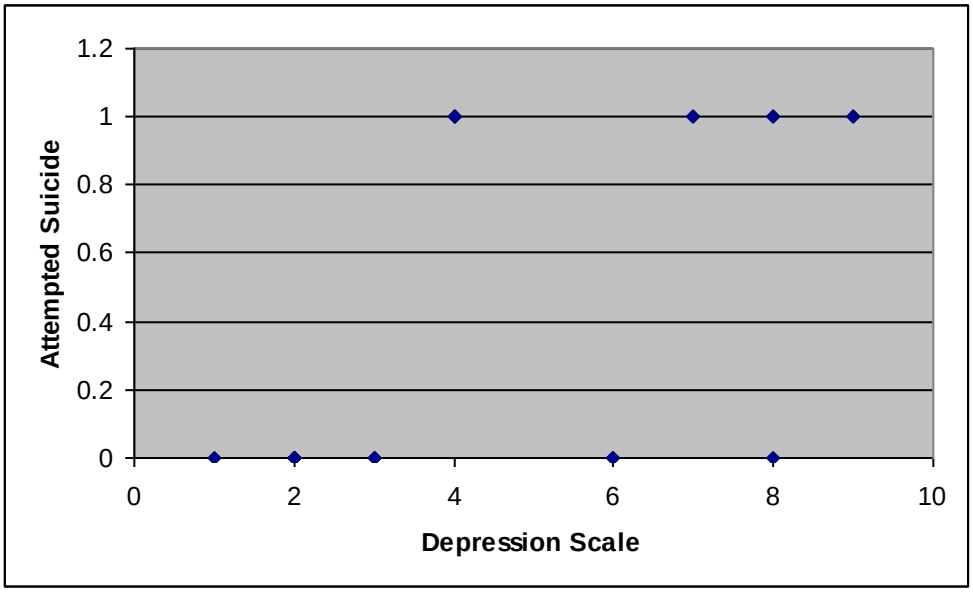
Subject	Suicide	DepressionCriteria	Depression
1	1	4	0
2	0	2	0
3	1	4	0
4	0	3	0
5	0	6	1
6	0	8	1
7	1	4	0
8	0	2	0
9	0	2	0
10	0	3	0
11	0	2	0

```
12 1 4 0
13 0 1 0
14 0 6 1
15 1 7 1
16 1 9 1
17 1 7 1
18 1 8 1
19 1 9 1
20 1 8 1
```

Proceed to view your data with the following commands:

```
data suicide;
  infile 'C:\Temp\suicide.txt' firstobs = 2;
  input subject suicide depcritera depression;
proc print;
run;
```

One thing that can be done now is a plot of the data. At the moment we will not be discussing graphics using SAS. Therefore we will continue the discussion in Excel just to obtain a plot of the data and resume with SAS. For the moment, we will assume that you have some experience with Excel and are capable of manipulating a spreadsheet and making charts. Proceed by making a chart that plots Suicide vs Depression. Your plot should look like the following:



At this point, it should be relatively easy to see that a straight line will not model this type of data very well. However, an S-shaped curve or sigmoid curve would do the trick. For that reason, we would employ logistic regression. In order to make our example easier

we will consider the variables suicide and depression which are both characterized as 1's and 0's. With the following SAS code, we can run a logistic regression as follows:

```
data suicide;
  infile 'C:\Temp\suicide.txt' firstobs = 2;
  input subject suicide depcriteria depression;
proc logistic descending;
  model suicide = depression;
run;
```

The only thing that is different in this block of code is the "proc logistic" which initiates the logistic procedure. Notice the option "descending" after the procedure statement. This tells SAS to use 1 as the outcome of interest for the dependent variable, "suicide". After the proc statement comes the statement containing the model format, you should always put the dependent variable first, an equal sign, and finally the independent variable(s).

The following output is obtained once these commands have been executed:

The LOGISTIC Procedure

Model Information

Data Set	WORK.SUICIDE
Response Variable	suicide
Number of Response Levels	2
Number of Observations	20
Link Function	Logit
Optimization Technique	Fisher's scoring

Response Profile

Ordered Value	suicide	Total Frequency
1	1	10
2	0	10

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	29.726	29.878
SC	30.722	31.869
-2 Log L	27.726	25.878

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	1.8480	1	0.1740
Score	1.8182	1	0.1775
Wald	1.7574	1	0.1850

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	-0.5596	0.6268	0.7971	0.3720
depression	1	1.2526	0.9449	1.7574	0.1850

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
depression	**3.500**	**0.549**	**22.300**

Association of Predicted Probabilities and Observed Responses

Percent Concordant	42.0	Somers' D	0.300
Percent Discordant	12.0	Gamma	0.556
Percent Tied	46.0	Tau-a	0.158
Pairs	100	c	0.650

Now that you have had a chance to view the output pay special attention to the section that is in bold titled Odds Ratio Estimates. From this information we get an estimate for the OR = 3.5. Basically this tells us that individuals who attempted suicide are 3.5 times more likely to be depressed. However, you must pay close attention to the 95% Confidence Interval of (0.549, 22.300) that accompanies the OR. Because the 95% Confidence Interval contains 1.000, the estimate for the OR is not statistically significant.

Now let's take the following scenario. Assume that Researcher Smith believes that Race is a key factor in suicide attempts, and he would like to explore his hypothesis by looking at data on suicide and depression coming from Whites, Blacks, and Hispanics. We will be using the text file 'suicide2.txt'. You should have been given a diskette where this data is stored. The researcher knows that is it common practice to use Whites as the reference group. In this example, we would like to compare Blacks and Hispanics to Whites, but in order to do so we need to create dummy variables. Table 1 refers to how this is to be done.

Table1.

	dummy1	dummy2
Whites (ref)	0	0
Blacks	1	0
Hispanics	0	1

At the moment, this may seem confusing; however, the following SAS code should give you some insight when creating dummy variables:

```
data suicide2;
  infile 'A:\suicide2.txt' firstobs = 2;
  input Suicide Depression Race;
if Race = 2 then dummy1 = 1;
```

```
   else dummy1 = 0;
if Race = 3 then dummy2 = 1;
   else dummy2 = 0;
proc logistic descending;
  model Suicide = dummy1 dummy2;
run;
```

After the program is run, the following output is obtained:

The LOGISTIC Procedure

Model Information

Data Set	WORK.SUICIDE2
Response Variable	Suicide
Number of Response Levels	2
Number of Observations	105
Link Function	Logit
Optimization Technique	Fisher's scoring

Response Profile

Ordered Value	Suicide	Total Frequency
1	1	42
2	0	63

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	143.332	142.199
SC	145.986	150.160
-2 Log L	141.332	136.199

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	5.1339	2	0.0768
Score	5.0000	2	0.0821
Wald	4.8431	2	0.0888

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	-1.0609	0.3867	7.5244	0.0061
dummy1	1	0.7732	0.5160	2.2454	0.1340
dummy2	1	1.1180	0.5138	4.7357	0.0295

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
dummy1	**2.167**	**0.788**	**5.957**
dummy2	**3.059**	**1.117**	**8.373**

Association of Predicted Probabilities and Observed Responses

Percent Concordant	46.0	Somers' D	0.238
Percent Discordant	22.2	Gamma	0.349
Percent Tied	31.7	Tau-a	0.115
Pairs	2646	c	0.619

Once again, let's look at the Odds Ratio estimates. Since dummy1 refers to Blacks, we can conclude that Blacks are 2.167 times more likely to attempt suicide compared to Whites. Since dummy2 refers to Hispanics, we can conclude that Hispanics are 3.059 times more likely to attempt suicide compared to Whites. Note that the 95%Confidence Interval is significant only for Hispanics because it does not contain 1.000.