Introduction to SAS
version 8.2 for Windows

Last Updated, Fall 2003
www.atlas.uiuc.edu
Table of Contents

What is SAS? ................................................................. 3
SAS at the University of Illinois ........................................ 3
Getting Help ............................................................... 3
SAS Environment ......................................................... 3
SAS Programs – The Basics ............................................. 5
  An example program  ................................................... 5
  Submitting programs ................................................... 6
Data Files and Libraries .................................................. 7
  Using an existing SAS dataset ....................................... 7
  Using an existing dataset in other formats ....................... 7
Working with Data ....................................................... 8
  Creating and deleting variables .................................... 8
  Recoding variables .................................................... 8
Running Procs .......................................................... 9
  proc print ............................................................. 9
  proc contents ....................................................... 9
  proc means ........................................................ 10
  proc univariate ..................................................... 11
  proc freq .......................................................... 11
  proc ttest .......................................................... 13
  proc glm ........................................................... 14
  proc corr ........................................................... 16
  proc reg ............................................................ 17
Graphical Output in SAS ............................................... 19
  proc gplot .......................................................... 20
  proc gchart ......................................................... 20
  proc g3d ............................................................ 22
Formatting Output .................................................... 23
Saving Your Work ..................................................... 25
Printing ................................................................. 26
Exiting SAS ............................................................. 26
What is SAS?

SAS, or “The SAS System” as its makers like to call it, is a software package for the management and analysis of data. Unlike the software that many computer users today are familiar with, you will interface with SAS primarily by writing short programs in a relatively straightforward code language specific to SAS. Though this can initially be intimidating, you just have to learn its language to properly communicate with the software. Learning SAS will let you access to one of the most powerful statistical software packages available, and give you a skill many employers look for. Go to http://www.sas-jobs.com/ to see some of the possibilities.

SAS at the University of Illinois

ATLAS, a division of LAS, has a number of computer classrooms with hours for public use. Room schedules are posted next to the classroom doors. SAS is installed on all computers in rooms 202 and 204 of Lincoln Hall (enter the complex through room 212) and in room G8 in the basement of the Foreign Language Building.

University of Illinois faculty, staff, and students may purchase yearly licenses to use SAS in their offices or on their home computers. The price of a license is usually about $50. You can check the current price and availability by clicking on SAS at http://www.ds.uiuc.edu/software.htm. (You’ll be asked for your netid and password before being redirected.) Purchasing SAS directly from the developer would cost several thousand dollars depending on the packages you chose.

Getting Help

LAS faculty, staff, and students can schedule appointments with ATLAS statistical consultants for free help. Consultants also keep regular hours that are posted outside 212 Lincoln Hall and at http://www.atlas.uiuc.edu/Stats/StatConsultingSched.htm. ATLAS can be reached by phone at 333-6750 and emailed at atlas@uiuc.edu. Users from outside LAS are charged $25/hour for help from a statistical consultant or the data archivist. University of Illinois staff and students can also find help at the Illinois Statistics Office, but will be charged an hourly fee (minimum $35). More information is available at http://www.stat.uiuc.edu/~iso/.

SAS has all its manuals in an online repository accessible at http://v8doc.sas.com/sashtml/. You will have to provide the username: onlinedoc and the password: sas. ATLAS has a number of older print editions of SAS manuals and other helpful books. SAS maintains a useful support site at http://support.sas.com/. Look on the ATLAS website at http://www.atlas.uiuc.edu/Stats/StatNewsandResources.htm and at a similar site from UCLA (http://www.ats.ucla.edu/stat/default.htm) for more help. These sites can be particularly useful sources for code fragments to use in your own SAS programs.

SAS Environment

Begin a SAS session by clicking on the start button and finding SAS under the Programs menu. The list is alphabetical but SAS is found in the folder labeled as “The SAS System.” Click on the button in the folder: to launch the program. SAS will open and look like this:
There are five main windows in the SAS environment, each of which is accessible from a button or tab near the bottom of the windows. Reading from left to right across the screen they are:

- **Results** – an outline of the output window to speed access to output that most interests you
- **Explorer** – view the contents of SAS data files and libraries
- **Output** – view the statistical or other results of your submitted editor code
- **Log** – gives messages about the code you submit for processing, including **ERROR MESSAGES**
- **Editor** – write and submit code for analysis

The output, log, and editor windows can be resized to whatever dimensions you prefer and can even be closed. If you close any of them, they can be opened again under the View menu at the top of the SAS window. The log and output windows can be cleared by pressing the white paper button while viewing the window you wish to clear. The results and explorer windows can only be changed in width.
**SAS Programs – The Basics**

A SAS program has two steps to it. The first is the **data** step in which SAS is told where to look for and how to read a file of data for analysis. In this step you can manually enter data if you want, or simply instruct SAS to open a file created by another program (Excel, SPSS, DBMSCOPY, etc.). You can also make new variables out of existing ones, transform variables, recode them, combine datasets, and a variety of other manipulations in the **data** step. The other kind of step is a **proc**. Proc, short for “procedure,” is a step that will actually examine your data and fill your output window with information.

**An example program**

On the next page is a sample SAS program to enter (hypothetical) data about people involved in public education, and then print it in the output window. Note the following things about a simple SAS program:

The program is color-coded by SAS. Data and proc statements are a **bold navy**, other statements and options interpreted by SAS are **bright blue**, actual data is filled in **yellow**, and annotations are **green**. If improper code is written SAS will highlight it in **red**. Variable and dataset names will remain black. You will see other color-coding as you work with SAS more. For example, code in quotations (" ") will be **purple**. It is a good idea to annotate you code so you know what it does. This is done by adding an asterisk (*) at the beginning of the statement.

Next, all lines in the program end with a semicolon (;). In SAS you must include these otherwise you will get errors and the program will probably not run. **The easiest mistake to make writing a SAS program is to leave out a semicolon.** It’s also too easy to misspell a variable or file name. Returns in the code (hitting “Enter”) do not matter to SAS: **proc print; run;** is read the same as **proc print;**

A data step begins with the declaration **data** and then provides a name for that dataset. In this case “educators.” Names can be up to 32 characters in length and can begin with a letter or an underscore (_). Following characters can be any combination of numbers, letters, or underscores. Special characters like *, ^, &, #, !, %, and $ are not allowed. Capitalization does not matter, as SAS is case insensitive. After the data set is named an **input** statement is given and the variables are named. The same rules for dataset names apply for variable names. SAS by default thinks all variables are numeric. Adding the $ **after** a variable name forces SAS to treat it as a character variable. A variable cannot, of course, be named “$”. Finally, SAS must be told that the data is going to be fed to it by a **cards** statement (a holdover from when computer data was stored by punching holes in card paper). The less archaic alternatives to **cards** are **datalines** and its shorter form **lines**. They all do the same thing.

Once data is available for analysis a **proc** can be run on it. The one in our program will just print out the dataset to display in the output window. Unless otherwise instructed, SAS will run the **proc** on the last dataset accessed. A **run** statement must follow every **proc** or data manipulation you code.
data educators;
input job $ sex marital house car salary political satisfac;
cards;
a 0 1 174063.87 15009.74 101392.78 1 6
a 1 3 129581.66 51075.57 84031.29 1 3
a 0 4  86812.48 76954.47 119737.28 1 5
a 1 2 170678.31 18565.79 108325.83 2 2
a 0 1 137080.61 78324.13 116396.80 0 4
t 1 3  66924.61 17729.13  69797.52 0 5
t 0 4 110585.64 .    73809.05 2 4
t 1 2  48985.12 11594.23  59736.72 2 2
t 0 1  62334.71 17419.86 53453.93 . 1
t 1 3  68601.52 15024.05  69797.52 0 0
s 0 4  35549.07 10636.65  20737.31 0 3
s 1 2  47871.16 6718.66  26835.67 0 2
s 0 1  31068.98  7191.43  25214.03 2 4
s . 3  56951.18 15822.16  29138.28 1 2
s 0 2  46018.44  8698.13  29165.59 0 4
;;;
* job: a=administrator t=teacher s=support (aid, secretary, janitor, etc.)
  sex: 0=male, 1=female marital: 1=married  2=single  3=divorced   4=widowed
  political: 0=democrat  1=republican  2=independent
  satisfac: 6=very happy <---> 0=completely unhappy ;
proc print;
run;

Submitting programs
To see the output of this program you will have to submit it. You do not need to type in the program, but can open the file “educators.txt” located in the folder S:SAS Training. Copy its contents with Edit > Select All (or Ctrl - a) from the top menu and then Edit > Copy (or Ctrl – c). Return to your SAS editor and paste what you have copied with Edit > Paste (or Ctrl - v). The program should look exactly like the one above. To submit it press the submit button that has an icon of a “running man” at the top of the screen. Note that you can only do this from the editor window. Once you have several pros in a program you may want to submit only a single one of them. You can do this by highlighting only the segment of code you want to submit with the mouse and then pressing the submit button.

What should have happened?
• Editor – nothing, only the user can change the editor
• Log – the program should have been read line by line and SAS gives you information about the dataset entered and the pros run on it
• Output – the dataset should be printed out in columns with each observation numbered
• Results – a small folder with “Print: The SAS System” next to it appeared
• Explorer – apparently nothing, but if you double-click on the filing cabinet icon for “Libraries” and then “Work” you should find a file called “Educators,” if you double-click on this icon you will be able to examine the dataset in a spreadsheet format
Data Files and Libraries

Though you now have a data set you can work with in SAS, “Educators,” it will vanish as soon as you close down SAS, whether you do this intentionally or not. A small dataset, such as the one above, can often just be included in your program and saved with it as a text or Word document, but larger datasets should not be. The best way to deal with such data sets is to make a permanent SAS data file. This is done by setting a two-level name for the file using a library name through the libname statement.

For example, if you wanted a permanent SAS data file of the “Educators” data set you would do the following:

- Set a destination library where you wished to save the file with the statement
  ```sas
  libname mystuff "D:\"
  ```

- Use a two-level name in the data statement with the part before the period as the libname
  ```sas
  data mystuff.educators;
  ```

- Submit the libname statement and entire data step, by clicking on the “running man”

Check that you now have a permanent SAS data file by browsing to D:\ (You can do this through Windows Explorer or by typing in D:\ at the run command.) The file should have the name after the period in the data step. The file extension for SAS data files is .SAS7BDAT . This part of the file name you will never need to type in SAS. You should also use the Explorer within SAS to check and see that there is, in fact, a library called “Mystuff” and it contains the “Educators” data set. It is often best to create a folder for your libname to reference rather than just a drive letter. This unfortunately cannot be done from within SAS. To do this using Windows Explorer browse to the location where you want a folder, right-click on the right pane of the window, and select “New > Folder.” Give it a name that you will remember, and then set the libname to reference it in SAS.

Using an existing SAS dataset

For the rest of this demonstration we will use a previously created permanent SAS data file. It is called “perm_ed” and is located in S:\SAS Training (S:\ is the drive for class data). For the purpose of the exercise set another libname to point to this location. Use a name that indicates it is the source of the data you are going be using and not where you will be saving your work. The code that follows would be an acceptable way of doing this.

```
*set libname for location of original data files;
libname datastor "S:\SAS Training";
```

Once the location of your permanent SAS data files are referenced by a properly set libname, you can run procs directly on them without any need of a data step. Before you continue, have two libnames set: 1 for your private data files, and 1 for the class data files. Here they have been called “mystuff” and “datastor,” respectively.

Using existing datasets in other formats

Data files come in many formats an SAS can read a number of them if you know the code to instruct it how to read the file properly. One easy way to get other data files into SAS for use is the infile command. If, for example, you had a comma-delimited file (one in which the columns of variables are separated by commas) named “many_ed.csv” at S:\SAS Training, you could import it using the code below. The dlm=',' tag lets SAS know that a comma separates entries. Comma-delimited files can be
created from Excel files by opening the .xls spreadsheet file and selecting “Save As ...” under the File menu and choosing “CSV (Comma-delimited)” under “Save as type:”. Excel files can also be imported through the “Import data” wizard under the File menu, or with a procedure called proc import that is part of SAS/ACESS (included in all UIUC distributions of SAS).

```sas
data ed;
infile "S:\SAS Training\many_ed.csv" dlm=',';
input job $ sex marital house car salary political satisfac;
cards;
```

Other formats will require you to know different code, similar to the dlm=',' tag, to import the data during the data step, but the general structure of the code will be the same. Check the Getting Help section above for where to find out about these coding differences. There are also standalone programs, such as DBMSCOPY, available on ATLAS computers, which allow you to directly create SAS data files from a variety of other formats.

**Working with Data**

**Creating and deleting variables**

If you wish to analyze variables in an existing SAS data file you do not need a data step. However, if you wish to recode or make new variables from that file you will have to include a data step in your editor code. If you wanted to create a new dataset with a log-transformed salary and combined measure of expenditures you could do the following using the educators data:

```sas
data mystuff.transed; *make a new file with transformed variables;
set datastor.perm_ed; *call the original file;
logsalary=log(salary); *transform/alter the desired variables;
expense=car+house;
drop salary house car; *get rid of the original data columns;
run;
```

By using the drop command at the end of the program SAS will eliminate the original data that it copied from the “perm_ed” data file, as it was instructed by the set command. The “transed” file will contain only our original codes, the log-transformed information about their salary, and the combined measure of spending. There are many functions available in SAS to transform, combine, and recode variables. Check the Getting Help section above for where to look for more information.

**Recoding variables**

Many times you may need to change the way a variable is coded or wish to change a continuous variable into a discrete one. Either of these can be done during the data step in SAS. Our educator data could be recoded to make salary a discrete variable with two categories (e.g. rich or poor), and marital could be recoded to two categories as well (e.g. married and unmarried). The following code would do this for you using if /then statements. Note that variables can easily be separated into more than two levels with greater than and less than signs such as has been done with the variable car here.
data mystuff.recode_ed;
set datastor.perm_ed;
if salary>80000 then saldisc=1;
if salary<80000 then saldisc=2;
if marital='2' or marital='3' or marital='4' then mardisc=1;
if marital='1' then mardisc=2;
if car<5000 then cardisc=0;
if 15000>car>=5000 then cardisc=1;
if 30000>car>=15000 then cardisc=2;
if car>=30000 then cardisc=3;
run;

If you need to do any kind of data manipulation, be aware that you will not be able to perform it on a permanent SAS data file located on an ATLAS server. You have “read-only” access to them. You must create your own data files from the class sets with by copying them in SAS with the set command.

**Running_procs**

SAS has many more procedures (statistical and otherwise) than you will probably ever use. Some of them are quite flexible and multipurpose, while others are dedicated to very specific applications. In this section a few of the most commonly used procs are outlined. To learn more about what else you can do with SAS check the Getting Help section above.

**proc print**

You already saw this proc in the example program above, but there are options that can be useful when printing out data or performing other procedures. Adding a `var` statement before `run` allows you to specify which variables you want printed. This will only print the variable salary. Typically, SAS will process all variables unless you specify them using `var`.

```
proc means data=datastor.perm_ed;
var salary;
run;
```

**proc contents**

This procedure will tell you a variety of information about a data set. The code and output are as follows:

```
proc contents data=datastor.perm_ed;
run;
```

The CONTENTS Procedure

<table>
<thead>
<tr>
<th>Data Set Name: DATASTOR.PERIM_ED</th>
<th>Observations: 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type: DATA</td>
<td>Variables: 8</td>
</tr>
<tr>
<td>Engine: V8</td>
<td>Indexes: 0</td>
</tr>
<tr>
<td>Created: 10:34 Friday, November 7, 2003</td>
<td>Observation Length: 64</td>
</tr>
<tr>
<td>Last Modified: 10:34 Friday, November 7, 2003</td>
<td>Deleted Observations: 0</td>
</tr>
<tr>
<td>Protection:</td>
<td>Compressed: NO</td>
</tr>
<tr>
<td>Data Set Type:</td>
<td>Sorted: NO</td>
</tr>
<tr>
<td>Label:</td>
<td></td>
</tr>
</tbody>
</table>

**Introduction to SAS**
-----Engine/Host Dependent Information-----

Data Set Page Size:         8192
Number of Data Set Pages:   1
First Data Page:            1
Max Obs per Page:           127
Obs in First Data Page:     15
Number of Data Set Repairs: 0
File Name:                  S:\SAS Training\perm_ed.sas7bdat
Release Created:            8.0202M0
Host Created:               WIN_PRO

-----Alphabetic List of Variables and Attributes-----

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>car</td>
<td>Num</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>house</td>
<td>Num</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>job</td>
<td>Char</td>
<td>8</td>
<td>56</td>
</tr>
<tr>
<td>3</td>
<td>marital</td>
<td>Num</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>political</td>
<td>Num</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>salary</td>
<td>Num</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>satisfac</td>
<td>Num</td>
<td>8</td>
<td>48</td>
</tr>
<tr>
<td>2</td>
<td>sex</td>
<td>Num</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

proc means

One of the most basic and useful procs is means. This proc gives you a set of simple descriptive statistics about any numeric variable in your dataset. As with proc print you should specify the variable(s) with the var statement; otherwise SAS will offer statistics on all the numeric variables in your data. There are other statistics you can instruct SAS to give you by adding tags after the data=xxx. Some of them are n, min, max, mean, std, median, q1, q3, nmiss, sum, range, stderr, t, probt, skew, kurt, cv, and clm. You can experiment with them or look in the manuals to find out what each one does. By default SAS will give you this first four of these. Also, if you don’t want absurdly precise statistics you can specify the number of decimals SAS will output with the tag maxdec=n.

```
proc means data=datastor.perm_ed;
var house car salary;
run;
```

The SAS System 10:27 Friday, November 7, 2003 1

The MEANS Procedure

<table>
<thead>
<tr>
<th>Variable</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>house</td>
<td>15</td>
<td>84073.82</td>
<td>47865.84</td>
<td>31068.98</td>
<td>174063.87</td>
</tr>
<tr>
<td>car</td>
<td>14</td>
<td>25054.57</td>
<td>24746.45</td>
<td>6718.66</td>
<td>78324.13</td>
</tr>
<tr>
<td>salary</td>
<td>15</td>
<td>63864.12</td>
<td>35484.58</td>
<td>20737.31</td>
<td>119737.28</td>
</tr>
</tbody>
</table>

Adding a by statement will give you statistics for each level of a discrete variable. However, for SAS to work properly the data must be sorted by the discrete variable first. Also, because you cannot sort the original class data file “perm_ed” you will have to create your own copy of it that you do have rights to manipulate. Do it with an out tag inside the proc sort as follows:
Our new file sort_ed in the mystuff library will be sorted by the variable job and available for a proc means by job. If you want SAS to sort in descending rather than the default, ascending, order, then you must add the tag `descending` between `by` and the variables in proc sort. You can use a `by` statement in many procs including print. Now, you can use proc means to get the statistics of continuous variables for each group of job:

```
proc means data=mystuff.sort_ed n mean median;
  var salary house;
  by job;
run;
```

It should be pointed out that SAS will not discriminate between ordinal, discrete, and continuous variables if they are entered as numeric. You must be smarter than SAS and not ask for “meaningless means" like the “mean of sex.”

**proc univariate**

When there is more information you want about a continuous variable but can’t get it out of means you should use proc univariate. This procedure is especially useful for checking the assumption that variables more or less fit a normal distribution and can safely be analyzed with common parametric statistical tests. Adding the tag `plot` after `data=xxx` will give you a stem-and-leaf and box-plot of the data along with a normal probability plot. The tag `normal` will provide four statistical tests of normality.

```
proc univariate data=datastor.perm_ed normal plot;
  var salary;
run;
```

A `by` statement can also be used with proc univariate to get statistics for each group of a discrete variable. The final portion of the output will also include a boxplot comparing all groups of the by variable. This is an easy way to graphically compare groups in SAS. Remember to sort the data by your discrete variable first if you wish to use a `by` statement.

**proc freq**

If you wish to do some basic analysis of discrete variables you will want to use proc freq. Freq works in a fashion similar to means but requires a `tables` rather than a `var` statement. For one-way tables the syntax is as follows:

```
proc freq data=datastor.perm_ed order=data;
  tables job sex marital;
run;
```

The tag `order=data` tells SAS to output the categories in each table in the order the data is entered rather than sorted alphabetically or ascending, which are the defaults. Note on the output for “sex” SAS has counted the number of missing cases. The cumulative percent and frequency columns reflect only valid cases (the missing cases have been removed from the calculation).
A two-way table is coded in a similar way except for the addition of an asterisk (*) between the two variables you wish to cross-tabulate in the `tables` statement. If there is an obvious independent-dependent relationship between variables (as there is in this case—sex could influence job but not vice versa) you should place the independent variable second in the `tables` statement.

```sas
proc freq data=datastor.perm_ed order=data;
  tables job*sex;
run;
```

Like the means procedure there are a number of options that can be specified for the freq output. All of these are placed after a forward slash (/) at the end of the `tables` statement. They include:

- `chisq`—calculates chi-square statistic for independence in a two-way table
- `expected`—computes the expected counts for a two-way table
- `nocum`—omits column percents
- `norow`—omits row percents
- `nofreq`—omits cell frequencies
- `nopercent`—omits basic percents

Here is a program that calculates the chi-square statistic to see whether job and sex are associated in the educators dataset. We would fail to reject the null hypothesis in this case. The chi-square statistic p-value (.5662) is located in the first row of the column labeled “Prob.” Unless otherwise noted all p-values SAS provides are 2-sided. Note that SAS has rightly given us a warning about how small our sample is and that the chi-square may be misleading here.

```sas
proc freq data=datastor.perm_ed order=data;
  tables job*sex /chisq expected norow nopercent;
run;
```
The FREQ Procedure

Table of job by sex

<table>
<thead>
<tr>
<th>job</th>
<th>sex</th>
<th>Frequency</th>
<th>Expected</th>
<th>Col Pct</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>3</td>
<td>2.8571</td>
<td>37.50</td>
<td>5</td>
</tr>
<tr>
<td>t</td>
<td>2</td>
<td>2</td>
<td>2.8571</td>
<td>25.00</td>
<td>5</td>
</tr>
<tr>
<td>s</td>
<td>3</td>
<td>3</td>
<td>2.2857</td>
<td>37.50</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>8</td>
<td>6</td>
<td></td>
<td></td>
<td>14</td>
</tr>
</tbody>
</table>

Frequency Missing = 1

Statistics for Table of job by sex

<table>
<thead>
<tr>
<th>Statistic</th>
<th>DF</th>
<th>Value</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-Square</td>
<td>2</td>
<td>1.1375</td>
<td>0.5662</td>
</tr>
<tr>
<td>Likelihood Ratio Chi-Square</td>
<td>2</td>
<td>1.1625</td>
<td>0.5592</td>
</tr>
<tr>
<td>Mantel-Haenszel Chi-Square</td>
<td>1</td>
<td>0.1387</td>
<td>0.7096</td>
</tr>
<tr>
<td>Phi Coefficient</td>
<td></td>
<td>0.2850</td>
<td></td>
</tr>
<tr>
<td>Contingency Coefficient</td>
<td></td>
<td>0.2741</td>
<td></td>
</tr>
<tr>
<td>Cramer's V</td>
<td></td>
<td>0.2850</td>
<td></td>
</tr>
</tbody>
</table>

WARNING: 100% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Effective Sample Size = 14
Frequency Missing = 1

**proc ttest**

Another common statistical test that can easily be performed in SAS is the *t* test. Proc *t*test syntax is similar to others you’ve seen but has an additional statement – `class`. This identifies the discrete variable that classifies a dependent variable you wish to analyze into two (and only two) categories.

```sas
proc ttest data=datastor.perm_ed; class sex;
var house;
run;
```
The SAS System     10:27 Friday, November 7, 2003  1

The TTEST Procedure

Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>sex</th>
<th>Lower CL Mean</th>
<th>Upper CL Mean</th>
<th>Lower CL Std Dev</th>
<th>Upper CL Std Dev</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>house</td>
<td>0</td>
<td>42242</td>
<td>85439</td>
<td>34163</td>
<td>51670</td>
<td>105162</td>
</tr>
<tr>
<td>house</td>
<td>1</td>
<td>36272</td>
<td>88774</td>
<td>31228</td>
<td>50028</td>
<td>122700</td>
</tr>
<tr>
<td>house</td>
<td>Diff (1-2)</td>
<td>-63337</td>
<td>-3335</td>
<td>56668</td>
<td>36566</td>
<td>50992</td>
</tr>
</tbody>
</table>

Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>sex</th>
<th>Std Err</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>house</td>
<td>0</td>
<td>18268</td>
<td>31069</td>
<td>174064</td>
</tr>
<tr>
<td>house</td>
<td>1</td>
<td>20424</td>
<td>47871</td>
<td>170678</td>
</tr>
<tr>
<td>house</td>
<td>Diff (1-2)</td>
<td>27539</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

T-Tests

| Variable | Method      | Variances | DF | t Value | Pr > |t| |
|----------|-------------|-----------|----|---------|------|---|
| house    | Pooled      | Equal     | 12 | -0.12   | 0.9056 |
| house    | Satterthwaite | Unequal  | 11.1 | -0.12 | 0.9053 |

Equality of Variances

<table>
<thead>
<tr>
<th>Variable</th>
<th>Method</th>
<th>Num DF</th>
<th>Den DF</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>house</td>
<td>Folded F</td>
<td>7</td>
<td>5</td>
<td>1.07</td>
<td>0.9779</td>
</tr>
</tbody>
</table>

If you wanted you could analyze more than one continuous variable at a time by listing them in the `var` statement (e.g. `var salary house car;`). Note that near the bottom of the output above, SAS has tested the equality of variances assumption, and given you an alternative test with a degrees of freedom adjustment (Satterthwaite). In this example the two rows of the t-test differ little because the variances of each group are so similar. If the equality of variances test at the bottom of the output is significant (“Pr>F” is is less than .05) then you should rely on the degrees of freedom adjusted test that is in the second row of the output under “T-Tests.”

**proc glm**

One of the most commonly used procedures in SAS is glm, which stand for general linear models. Many people use it for analysis of variance. We could do this with our educators dataset by specifying independent and dependent variables in a `model` statement. Like in ttest a `class` statement is also necessary, however the independent variable can now have more than 2 levels. The only other statement that is new is `means`, which is an optional statement in glm. The `/tukey` tag will use this method to separate the three mean house price for each of the three job categories.

```sas
proc glm data=datastor.perm_ed; class job;
model house=job; *predict house price with job;
means job /tukey;
run;
```
The output below shows that job significantly predicts house price (p=.0002). The tukey means separation technique identifies administrators as having more expensive houses than teachers and support workers, but teachers and support workers are not statistically different, though their means do differ by a good margin. Note in the middle part of the output the significance of job is tested twice with different sum of squares methods. They are identical in this case but would differ if we included a second independent variable (political, maybe). (This would be 2-way analysis of variance, which requires some different coding in the class and model statements.)
**proc corr**
The analysis of a set of continuous variables often begins by examining their correlations. SAS has a procedure for this type of analysis called proc corr. Its syntax and output are very simple:

```sas
proc corr data=datastor.perm_ed;
var salary house car;
run;
```

```
The SAS System 10:27 Friday, November 7, 2003 1
The CORR Procedure

3 Variables: salary house car

Simple Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Sum</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>salary</td>
<td>15</td>
<td>63864</td>
<td>35485</td>
<td>957962</td>
<td>20737</td>
<td>119737</td>
</tr>
<tr>
<td>house</td>
<td>15</td>
<td>84874</td>
<td>47866</td>
<td>1273107</td>
<td>31069</td>
<td>174064</td>
</tr>
<tr>
<td>car</td>
<td>14</td>
<td>25055</td>
<td>24746</td>
<td>350764</td>
<td>6719</td>
<td>78324</td>
</tr>
</tbody>
</table>

Pearson Correlation Coefficients
Prob > |r| under H0: Rho=0
Number of Observations

<table>
<thead>
<tr>
<th></th>
<th>salary</th>
<th>house</th>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td>salary</td>
<td>1.00000</td>
<td>0.82959</td>
<td>0.74219</td>
</tr>
<tr>
<td>house</td>
<td>0.82959</td>
<td>1.00000</td>
<td>0.42944</td>
</tr>
<tr>
<td>car</td>
<td>0.74219</td>
<td>0.42944</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

Correlation coefficients are given in the table at the bottom of the output with p-values just beneath them. If you are only interested in how a single variable correlates with others you should use the `with` statement as shown in the code below.

```sas
proc corr data=datastor.perm_ed;
var salary;
with house car;
run;
```
**proc reg**

Linear regression can be performed in several procs in SAS but the one designed for it is proc reg. In simple linear regression a continuous independent variable is chosen to predict a continuous dependent, or response, variable. The syntax is similar to glm but there is no class statement. A by statement can be used if you wish to do regression within groups. As an example, we could use salary to try to predict house price. From the proc corr output above we know that they are fairly tightly correlated (r=.82959, p<.0001), so regression would identify how, on average, house price changes with increasing salary. The syntax would be:

```
proc reg data=datastor.perm_ed;
model house=salary; *dependent=independent;
plot house*salary; *generate plot with regression line;
run;
```

The output begins with a test of the complete regression model (intercept and slope) significance in an analysis of variance table. It provides you with several other descriptors including the squared correlation coefficient (R-squared), and then gives the regression equation with standard errors on each parameter and tests of significance. In this case the intercept is not significant but salary is (p<.0001).

```
The SAS System          10:28 Friday, November 7, 2003  1

The REG Procedure
Model: MODEL1
Dependent Variable: house

Analysis of Variance

              Source    DF     Sum of Squares  Mean Square  F Value    Pr > F
Model              1    22075101605     22075101605    28.69    0.0001
Error             13    10000929563      769302274
Corrected Total  14    32076031168

Root MSE 27736    R-Square 0.6882
Dependent Mean 84874    Adj R-Sq 0.6642
Coeff Var 32.67943

Parameter Estimates

              Variable    DF  Parameter Estimate  Standard Error  t Value    Pr > |t|
Intercept       1    Intercept   13407        15142       0.89      0.3920
salary          1    salary      1.11905      0.20890       5.36      0.0001
```

Because we included the optional plot statement in proc reg, SAS will also output a high quality scatterplot of the data (shown below) with the calculated regression line, including annotation with the regression equation and other useful bits of information. It does this in a new window you may not have seen yet called “Graph.” In the past you had to code the output of such a graph manually, which could be time consuming and frustrating. Some other useful options in proc reg are:

- **p** - prints the observed values, predicted values, and residuals
- **r** - does everything p does plus giving several other statistics
- **clm** - prints 95% confidence intervals for the mean of each observation
- **cli** - prints 95% prediction intervals

*Introduction to SAS*
house = 13407 + 1.119 salary

- N = 15
- Rsq = 0.6882
- Adj Rsq = 0.6642
- RMSE = 27736
Graphical Output in SAS

**proc plot**
This is the old way to make quick and easy plots in SAS. They are not pretty to look at but get the job done in many cases. You can compare the plot below with the one on the previous page that proc reg made automatically.

```sas
proc plot data=datastor.perm_ed;
plot house*salary='S'; * yvar*xvar='pointsymbol' symbol optional;
run;
```

The SAS System          10:28 Friday, November 7, 2003 1
Plot of house*salary. Symbol used is 'S'.

Introduction to SAS
proc gplot
The way to make much more presentable plots (like the one proc reg output) is to use proc gplot. Gplot is a fairly recent addition to SAS so older books and manuals may not help you out with it. There is ample help online and a detailed manual on gplot available in the ATLAS Lincoln Hall labs. Figuring this proc out can take some time but the flexibility and control over your output you have in gplot will make it worthwhile. Here is an example of a simple scatterplot in gplot:

```sas
proc gplot data=datastor.perm_ed;
plot house*salary=2; *2 is red;
symbol value=dot;
run;
```

proc gchart
If you need to create histograms of continuous variables or charts of discrete variables, SAS can do that as well. Proc gchart outputs high quality histograms and bar, pie, doughnut, and star charts. Here are two examples:

```sas
proc gchart data=datastor.perm_ed; *for a histogram;
hbar3d salary; *vbar3d will make them vertical;
run;
```
**proc gchart data=datastor.perm_ed;**  *for a pie chart;*
**pie3d marital; run;**

---

**Introduction to SAS**
You can also do 3-dimensional plotting in SAS with a number of procedures. G3d is one of the most straightforward in which surfaces can be fit to data or a simple scatter can be produced. Here are the three continuous variables from the educator data:

```sql
proc g3d data=datastor.perm_ed;
scatter house*car=salary; * x*y=z ;
run;
```
Formatting Output

The normal SAS text output can be either difficult to read because of page breaks or may contain too much, or too little, information about the work you are doing. There are a number of ways you can change the way that SAS writes out results in the output window. Some of these are controlled by tags in an options statement usually given at the very beginning of your SAS editor program. Here are several commonly used options tags:

- `formdlim='x'` sets the division between pages in the output to the specified character in quotes rather than the default page break, the dash (--) is often nice
- `nodate` eliminates the date and time stamp SAS prints on every output page (useful if you don’t want your professors to know when you completed your assignment)
- `pageno=x` sets the first page of output from each submission to be x, 1 is usually a good choice, this will keep you from having page numbers in the hundreds and thousands, alternatively you can just use the nonumber tag and not have any numbering
- `nocenter` if you don’t want your output centered on the page use this tag
- `linesize=x` specifies the number of characters in a line in the output, you can play with this to find the exact number of characters to fit into a document of your specifications, for example a Word document with letter size paper (8.5”x11”) and 1” margins
- `pagesize=x` specifies the number of lines in a page, same benefits as linesize

```
options nodate pageno=1 linesize=97 pagesize=55 formdlim='-';
```

It is very easy to make descriptive notes to yourself that will be printed as headings on the SAS output window and in graphs. This is done with title statements. All output after a title statement is submitted will contain that title until a new title is specified. Multiple title lines can be added but require a number after the word title. Here is an example of how you might make headings for a homework output.

```
title1 'chapter 1 homework'; *remains constant;
title2 'problem #1';            *changes for each problem;

data mystuff.educators;
input job$ sex$ marital$ house car salary;
cards;
- - - lines of data - - -
;;;;
proc print;
run;

title2 'problem #2';
proc means n mean std median q1 q3;
var salary;
run;

title2 'problem #3';
proc corr ;
var house salary car;
run;
```
### Problem #1

<table>
<thead>
<tr>
<th>Obs</th>
<th>Job</th>
<th>Sex</th>
<th>Marital</th>
<th>House</th>
<th>Car</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>m</td>
<td>m</td>
<td>174064</td>
<td>15010</td>
<td>101393</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>f</td>
<td>d</td>
<td>129502</td>
<td>51076</td>
<td>84031</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>m</td>
<td>w</td>
<td>86812</td>
<td>76954</td>
<td>119737</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>f</td>
<td>s</td>
<td>170678</td>
<td>18566</td>
<td>108326</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td>m</td>
<td>m</td>
<td>137001</td>
<td>78324</td>
<td>116397</td>
</tr>
<tr>
<td>6</td>
<td>t</td>
<td>f</td>
<td>d</td>
<td>66925</td>
<td>17729</td>
<td>69798</td>
</tr>
<tr>
<td>7</td>
<td>t</td>
<td>m</td>
<td>m</td>
<td>110586</td>
<td>19091</td>
<td>73809</td>
</tr>
<tr>
<td>8</td>
<td>t</td>
<td>f</td>
<td>s</td>
<td>48985</td>
<td>11594</td>
<td>59737</td>
</tr>
<tr>
<td>9</td>
<td>t</td>
<td>m</td>
<td>m</td>
<td>62335</td>
<td>17420</td>
<td>53454</td>
</tr>
<tr>
<td>10</td>
<td>t</td>
<td>f</td>
<td>d</td>
<td>68602</td>
<td>15024</td>
<td>40190</td>
</tr>
<tr>
<td>11</td>
<td>s</td>
<td>m</td>
<td>s</td>
<td>35549</td>
<td>10637</td>
<td>20737</td>
</tr>
<tr>
<td>12</td>
<td>s</td>
<td>f</td>
<td>s</td>
<td>47871</td>
<td>6719</td>
<td>26836</td>
</tr>
<tr>
<td>13</td>
<td>s</td>
<td>m</td>
<td>m</td>
<td>31069</td>
<td>7191</td>
<td>25214</td>
</tr>
<tr>
<td>14</td>
<td>s</td>
<td>f</td>
<td>d</td>
<td>56951</td>
<td>15822</td>
<td>29138</td>
</tr>
<tr>
<td>15</td>
<td>s</td>
<td>m</td>
<td>s</td>
<td>46018</td>
<td>8698</td>
<td>29166</td>
</tr>
</tbody>
</table>

### Problem #2

#### The MEANS Procedure

**Analysis Variable: salary**

<table>
<thead>
<tr>
<th></th>
<th>Lower</th>
<th>Upper</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Median</th>
<th>Quartile</th>
<th>Quartile</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>15</td>
<td></td>
<td>63864</td>
<td>35484.57</td>
<td>59737</td>
<td>29138</td>
<td>101393</td>
</tr>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td>63864</td>
<td>35484.57</td>
<td>59737</td>
<td>29138</td>
<td>101393</td>
</tr>
</tbody>
</table>

### Problem #3

#### The CORR Procedure

**3 Variables: house salary car**

**Simple Statistics**

<table>
<thead>
<tr>
<th>Variable</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Sum</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>house</td>
<td>15</td>
<td>84074</td>
<td>47866</td>
<td>1273108</td>
<td>31069</td>
<td>174064</td>
</tr>
<tr>
<td>salary</td>
<td>15</td>
<td>63864</td>
<td>35485</td>
<td>957963</td>
<td>20737</td>
<td>119737</td>
</tr>
<tr>
<td>car</td>
<td>15</td>
<td>24657</td>
<td>23096</td>
<td>369855</td>
<td>6719</td>
<td>78324</td>
</tr>
</tbody>
</table>

**Pearson Correlation Coefficients, N = 15**

<table>
<thead>
<tr>
<th></th>
<th>house</th>
<th>salary</th>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td>house</td>
<td>1.0000</td>
<td>0.8295</td>
<td>0.4142</td>
</tr>
<tr>
<td>salary</td>
<td>0.8295</td>
<td>1.0000</td>
<td>0.7334</td>
</tr>
<tr>
<td>car</td>
<td>0.4142</td>
<td>0.7334</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
Saving Your Work

There are several ways in which you can permanently save your SAS programs and output to take out of the lab. You cannot save on ATLAS machines and expect it to remain there after you log off. It is suggested that you take your files with you in at least one (two is better) of the following ways:

- email your documents to yourself
- upload them to your Netfiles account, if you have one
- save them to a floppy disk, USB pen drive, etc.
- write to CD-R/RW media (available for purchase, ATLAS computers have built-in CD-burners)
- save to a ZIP disk, USB ZIP drives can be checked out at the front desk

The files you should take with you will include your editor program, your output, and perhaps your log. All of these can be saved within SAS, each in its own format. The editor is saved with the extension .sas, the log .log, and the output as .lst. All of these documents can be opened from a word processor like Microsoft Word, as long as the open file type is specified as “All files.” Unless your log has messages in it that you want to research you probably don’t need to save it after you have checked it over for errors.

If you would like to preserve the color-coding of your editor, it is recommended that you save your editor as a Word document. Do this by selecting the entire editor window, copying it, and pasting it into a blank text or Word document. This way you will be able to access your conveniently highlighted code on whatever computer you sit down in front of, whether it has SAS installed or not.

If your output is all text then you can save it in SAS, or copy and paste this into a text or Word document with your color-coded editor code. However, to save your output when it has graphics components (gplot or gchart output) it is recommended that you use SAS’s output delivery system (ODS) to generate an .html document with gif images to take with you. Do this by adding the line of code below at the beginning of your editor and the following one at the very end of your program. Submit the editor, with no text selected, and an html document of your entire program’s output will be created.

```
ods html file='A:\mySASoutput.htm'; *I go at the beginning;
---- program -----
ods html close; *I go at the end;
```

Be sure to take all of the graph window image files SAS created, as well. They should have names like “gchart3.gif” and be located in the same folder as the html document. The ODS statements can be added anywhere in the program so you can choose only certain portions of your program to have html output. If there are perhaps only one or two graphics that you would like to take with you, you can save them specifically by having them in the active window and choosing “Export as Image…” from the File menu. You can choose among image formats to save in; .bmp and .tif will give you high quality but large files while .jpg and .gif offer compression to smaller file size but will have some reduction in quality.

You will probably not need to save your datasets. If they are class datasets they will remain on ATLAS servers and you can’t overwrite them even if you try to; you have read-only access to them. If you want your own copy of a data file or you modified a dataset and want to save it to a new file you should do that on the lab computer you are working on (see the section on Data Files and Libraries above) and then take it with you by one of the methods noted above. ZIP disks, pen drives, and CD-R/RWs are probably best for this, as many data files that classes use are too large for a floppy disk.
**Printing**

You can print your editor, log, output, and any charts and graphics from within SAS under the File > Print menu. Be sure to run a “Print Preview” and alter the “Page Setup” if necessary before sending a print job. It often saves paper, ink, and money. If you click the printer icon at the top of the screen, SAS will immediately print the active window (editor, output, log, or graph) to the default printer, whether that is what you want it to do or not. When you print from an ATLAS computer lab in the Lincoln Hall suite, your print job should be printed on the laser printer that sits in the hallway near the front desk. You will be charged through the library printing system. If you wish to write in notes on your editor program or text outputs you can copy them into a text or Word document, add whatever information you like, and then print from the word processor as you would any other document.

**Exiting SAS**

You can quit SAS by clicking on the ❌ at the top right of the window or choosing “Exit” from the File menu. You will be prompted whether you want to end the SAS session. If you choose OK you will be prompted to save the editor. Do so if you haven’t already. SAS will then close. All temporary (1-level) datasets will be erased along with any unsaved editor, log, output, and graph windows.