Chapter 2

First set of tools: SAS running under unix (including linux)

The SAS language is the same regardless of what hardware you use or what operating system is running on the hardware. SAS programs are simple text files that can be transported from one machine to another with minimal difficulty. In this course, everything will be illustrated with SAS running under the unix operating system, but it's not a problem even if the next place you go only has PCs. The adjustment to SAS-PC should be fast and fairly painless.

2.1 Unix

Unix is a line-oriented operating system. Well, there's X-windows (a graphical shell that runs on top of unix), but we won't bother with it. Basically, you type a command, press Enter, and unix does something for (or to) you. It may help to think of unix as DOS on steroids, if you remember DOS. The table below has all the unix commands you will need for this course. Throughout, *fname* stands for the name of a file.

A Minimal Set of unix Commands

exit Logs you off the system: ALWAYS log off before leaving!

passwd Lets you change your password. Recommended.

man *command name* Online help: explains *command name*, (like `man sort`).

ls Lists names of the files in your directory.

less *fname* Displays fname on screen, one page at a time. Spacebar for next page, `q` to quit.

lpr *fname* Prints hard copy on a laser printer. lpr stands for line printer. These physical devices no longer exist in most installations.

rm *fname* Removes *fname*, erasing it forever.

cp *fname1 fname2* Makes a copy of *fname1*. The new copy is named *fname2*.

mv *fname1 fname2* Moves (renames) *fname1*

emacs *fname* Starts the `emacs` text editor, editing *fname* (can be new file).

R Gets you into the `R` implementation of the S environment.

sas *fname* Executes `SAS` commands in the file ***fname*`.sas`**, yielding ***fname*`.log`** and (if no fatal errors) ***fname*`.lst`**.

ps Shows active processes

kill -9 # Kills process (job) number `#`. Sometimes you must do this when you can't log off because there are stopped jobs. Use `ps` to see the job numbers.

mail yourname@yourisp.com ¡ *fname* Email a file to yourself. Very handy for getting files to your home computer for printing.

This really is a minimal set of commands. The unix operating system is extremely powerful, and has an enormous number of commands. You can't really see the power from the minimal set of commands above, but you can see the main drawback from the standpoint of the new user. Commands tend to be terse, consisting of just a few keystrokes. They make sense once you are familiar with them (like `ls` for listing the files in a directory, or `rm` for remove), but they are hard to guess. The `man` command (short for manual) gives very accurate information, but you have to know the name of the command before you can use `man` to find out about it.

Just for future reference, here are a few more commands that you may find useful, or otherwise appealing.

A Few More unix Commands

mkdir *dirname* Makes a new sub-directory (like a folder) named *dirname*. You can have sub-directories within sub-directories; it's a good way to organize your work.

cp *fname dirname* Copies the file *fname* into the directory *dirname*.

cd *dirname* Short for Change Directory. Takes you to the sub-directory *dirname*.

cd .. Moves you up a directory level.

cd Moves you to your main directory from wherever you are.

ls > *fname* Sends the output of the `ls` command to the file *fname* instead of to the screen.

cat *fname* Lists the whole file on your screen, not one page at a time. It goes by very fast, but usually you can scroll back up to see the entire file, if it's not too long.

cat *fname1 fname2* > *fname3* Concatenates *fname1* and *fname2* (sticks them together) and re-directs the output to *fname3*

grep ERROR cartoon1.log Searches for the string ERROR in the file cartoon1.log. Echos each line containing the string. Silent if ERROR does not occur. Case sensitive.

alias chk "grep ERROR *.log ; grep WARN *.log" Makes a new command called `chk`. It checks for the string ERROR and the string WARN in any log file.

cal Displays a calendar for this month

cal 1 3002 Displays a calendar for January 3002.

unset noclobber

rm *fname1 fname2*

rm -f *fname*

alias rm "rm -f"

rm -r *dirname*

R –vanilla < *fname1* > *fname2*

Printing files at home This is a question that always comes up. Almost surely, the printer connected to your printer at home is not directly connected to the university network. If you want to do something like print your SAS output at home, you have to transfer the file on the unix machine to the hard drive of your home computer, and print it from there. One way is to use some kind of `ftp` (file transfer protocol) tool to get the file in question onto your hard drive. For short files, you can also use the `less` or `cat` command to list the file on your screen, select it with your mouse, copy it, paste it to a word processing document, and print it from there. It is a good idea to use a fixed-width font like courier, and not the Times or Times Roman font. Everything will be lined up better.

Perhaps easiest of all is to email yourself the file. This is illustrated in the first set of unix commands. To repeat, `mail yourname@yourisp.com < fname`.

2.2 Introduction to SAS

SAS stands for "Statistical Analysis System." Even though it runs on PCs as well as on bigger computers, it is truly the last of the great old mainframe statistical packages. The first beta release was in 1971, and the SAS Institute, Inc. was spun off from North Carolina State University in 1976, the year after Bill Gates dropped out of Harvard. This is a serious pedigree, and it has both advantages and disadvantages.

The advantages are that the number of statistical procedures SAS can do is truly staggering, and the most commonly used ones have been tested so many times by so many people that their correctness and numerical efficiency is beyond any question. For the purposes of this class, there are no bugs. The disadvantages of SAS are all related to the fact that it was *designed* to run in a batch-oriented mainframe environment. So, for example, the SAS Institute has tried hard to make SAS an "interactive" program, but the interface still basically file and text oriented, not graphical.

2.2.1 The Four Main File Types

A typical SAS job will involve four main types of file.

- **The Raw Data File**: A file consisting of rows and columns of numbers; or maybe some of the columns have letters (character data) instead of numbers. The rows represent observations and the columns represent variables, as described at the beginning of Section 1.1. In the first example we will consider below, the raw data file is called `drp.dat`.

- **The Program File**: This is also sometimes called a "command file," because it's usually not much of a program. It consists of commands that the SAS software tries to follow. You create this file with a text editor like `emacs`. The command file contains a reference to the raw data file (in the `infile` statement), so SAS knows where to find the data. In the first example we will consider below, the command file is called `reading.sas`. SAS expects program files to have the extension `.sas`, and you should always follow this convention.

- **The Log File**: This file is produced by every SAS run, whether it is successful of unsuccessful. It contains a listing of the command file, as well any error messages or warnings. The name of the log file is automatically generated by SAS; it combines the first part of the command file's name with the extension `.sas`. So for example, when SAS executes the commands in `reading.sas`, it writes a log file named `reading.log`.

- **The List File**: The list file contains the output of the statistical procedures requested by the command file. The list file has the extension `.lst` — so, for example, running SAS on the command file `reading.sas` will produce `reading.lst` as well as `reading.log`. A successful SAS run will almost always produce a list file. The absence of a list file indicates that there was at least one fatal error. The presence of a list file does not mean there were no errors; it just means that SAS was able to do *some* of what you asked it to do. Even if there are errors, the list file will usually not contain any error messages; they will be in the log file.

2.2.2 Running SAS from the Command Line

There are several ways to run SAS. In this text, all the examples will be run from the unix command line (`terminal`). In my view, this way is simplest and also the best way to start. Also, it is by far the easiest way to use SAS from home, assuming that SAS is running on a remote server and not your home computer.

The following illustrates a simple SAS run from the command line (using an application called `terminal` in some unix and linux environments). Initially, there are only two files in the directory — `reading.sas` (the program file) and `drp.dat` (the raw data file). The command `sas reading` produces two additional files — `reading.log` and `reading.lst`. In this and other examples, the unix prompt is `appsrv01.srv` (the name of the unix machine used to produce the examples), followed by a `>` sign.

```
appsrv01.srv> ls
drp.dat         reading.sas
appsrv01.srv> sas reading
appsrv01.srv> ls
drp.dat         reading.log     reading.lst     reading.sas
```

2.2.3 Structure of the Program File

A SAS program file is composed of units called *data steps* and *proc steps*. The typical SAS program has one data step and at least one proc step, though other structures are possible.

- Most SAS commands belong either in data step or in a proc step;they will generate errors if they are used in the wrong kind of step.
- Some statements, like the `title` and `options` commands, exist outside of the data and proc steps, but there are relatively few of these.

The Data Step The data step takes care of data acquisition and modification. It almost always includes a reference to the raw data file, telling SAS where to look for the data. It specifies variable names and labels, and provides instructions about how to read the data; for example, the data might be read from fixed column locations. Variables from the raw data file can be modified, and new variables can be created.

Each data step creates a **SAS data set**, a file consisting of the data (after modifications and additions), labels, and so on. Statistical procedures operate on SAS data sets, so you must create a SAS data set before you can start computing any statistics.

A SAS data set is written in a binary format that is very convenient for SAS to process, but is not readable by humans. In the old days, SAS data sets were always written to temporary scratch files on the computer's hard drive; these days, they may be maintained in RAM if they are small enough. In any case, the default is that a SAS data set disappears after the job has run. If the data step is executed again in a later run, the SAS data set is re-created.

Actually, it is possible to save a SAS data set on disk for later use. We won't do this here, but it makes sense when the amount of processing in a data step is large relative to the speed of the computer. As an extreme example, one of my colleagues uses SAS

to analyze data from Ontario hospital admissions; the data files have millions of cases. Typically, it takes around 20 hours of CPU time on a very strong unix machine just to read the data and create a SAS data set. The resulting file, hundreds of gigabytes in size, is saved to disk, and then it takes just a few minutes to carry out each analysis. You wouldn't want to try this on a PC.

To repeat, SAS data *steps* and SAS data *sets* sound similar, but they are distinct concepts. A SAS data *step* is part of a SAS program; it generates a SAS data *set*, which is a file – usually a temporary file.

SAS data sets are not always created by SAS data steps. Some statistical procedures can create SAS data sets, too. For example, `proc univariate` can take an ordinary SAS data set as input, and produce an output data set that has all the original variables, and also some of the variables converted to z-scores (by subtracting off the mean and dividing by the standard deviation). `Proc reg` (the main multiple regression procedure) can produce a SAS data set containing residuals for plotting and use in further analysis; there are many other examples.

The `proc` Step "Proc" is short for procedure. Most procedures are statistical procedures; the most noticeable exception is `proc format`, which is used to provide labels for the values of categorical variables. The `proc` step is where you specify a statistical procedure that you want to carry out. A statistical procedures in the `proc` step will take a SAS data sets as input, and write the results (summary statistics, values of test statistics, p-values, and so on) to the list file. The typical SAS program includes one data step and several `proc` steps, because it is common to produce a variety of data displays, descriptive statistics and significance tests in a single run.

2.2.4 A First Example: `reading.sas`

Earlier, we ran SAS on the file `reading.sas`, producing `reading.log` and `reading.lst`. Now we will look at `reading.sas` in some detail. This program is very simple; it has just one data step and one proc step. More details will be given later, but it's based on a study in which one group of grade school students received a special reading programme, and a control group did not. After a couple of months, all students were given a reading test. We're just going to do an independent groups t-test, but first take a look at the raw data file. You'd do this with the unix `less` command.

Actually, it's so obvious that you should look at your data that nobody ever says it. But experienced data analysts always do it — or else they assume everything is okay and get a bitter lesson in something they already knew. It's so important that it gets the formal status of a **data analysis hint**.

Data Analysis Hint 1 *Always look at your raw data file. It the data file is big, do it anyway. At least page through it a screen at a time, looking for anything strange. Check the values of all the variables for a few cases. Do they make sense? If you have obtained the data file from somewhere, along with a description of what's in it, never believe that the description you have been given is completely accurate.*

Anyway, here is the file `drp.dat`, with the middle cut out to save space.

Treatment	24
Treatment	43
Treatment	58
⋮	⋮
Control	55
Control	28
Control	48
⋮	⋮

Now we can look at `reading.sas`.

```
/****************** reading.sas *********************
*  Simple SAS job to illustrate a two-sample t-test    *
*******************************************************/

options linesize=79 noovp formdlim='_';
title 'More & McCabe (1993) textbook t-test Example 7.8';

data reading;
     infile 'drp.dat';
     input group $ score;
     label group = 'Get Directed Reading Programme?'
           score = 'Degree of Reading Power Test Score';
proc ttest;
     class group;
     var score;
```

Here are some detailed comments about `reading.sas`.

- The first three lines are a comment. Anything between a `/*` and `*/` is a comment, and will be listed on the log file but otherwise ignored by SAS. Comments can appear anywhere in a program. You are not required to use comments, but it's a good idea.

  The most common error associated with comments is to forget to end them with `*/`. In the case of `reading.sas`, leaving off the `*/` (or typing `\*` by mistake) would cause the whole program to be treated as a comment. It would generate no errors, and no output — because as far as SAS would be concerned, you never requested any. A longer program would eventually exceed the default length of a comment (it's some large number of characters) and SAS would end the "comment" for you. At exactly that point (probably in the middle of a command) SAS would begin parsing the program. Almost certainly, the first thing it examined would be a fragment of a legal command, and this would cause an error. The log file would say that the command caused an error, and not much else. It would be *very* confusing, because probably the command would be okay, and there would be no indication that SAS was only looking at part of it.

- The next two lines (the `options` statement and the `title` statement) exist outside the proc step and the data step. This is fairly rare.

- All SAS statements end with a semi-colon (`;`). SAS statements can extend for several physical lines in the program file (for example, see the `label` statement). Spacing, indentation, breaking up s statement into several lines of text – these are all for the convenience of the human reader, and are not part of the SAS syntax.

- The most common error in SAS programming is to forget the semi-colon. When this happens, SAS tries to interpret the following statement as part of the one you tried to end. This often causes not one error, but a cascading sequence of errors. The rule is, *if you have an error and you do not immediately understand what it is, look for a missing semi-colon.* It will probably be *before* the portion of the program that (according to SAS) caused the first error.

- Cascading errors are not caused just by the dreaded missing semi-colon. They are common in SAS; for example, a runaway comment statement can easily cause a chain reaction of errors (if the program is long enough for it to cause any error messages at all). *If you have a lot of errors in your log file, fix the first one and don't waste time trying to figure out the others.* Some or all of them may well disappear.

- `options linesize=79 noovp formdlim='_';`

  These options are highly recommended. The `linesize=79` option is so highly recommended it's almost obligatory. It causes SAS to write the output 79 columns across, so it can be read on an ordinary terminal screen that's 80 characters across. You specify an output width of 79 characters rather than 80, because SAS uses one column for printer control characters, like page ejects (form feeds).

  If you do not specify `options linesize=79;`, SAS will use its default of 132 characters across, the width of sheet of paper from an obsolete line printer you probably have never seen. Why would the SAS Institute hang on to this default, when changing it to match ordinary letter paper would be so easy? It probably tells you something about the computing environments of some of SAS's large corporate clients.

- The `noovp` option makes the log files more readable if you have errors. When SAS finds an error in your program, it tries to *underline* the word that caused the error. It does this by going back and *overprinting* the offending word with a series of "underscores" (_ characters). On many printers this works, but when you try to look at the log file on a terminal screen (one that is *not* controlled by the SAS Display Manager), what often appears is a mess. The `noovp` option specifies `no overprinting`. It causes the "underlining" to appear on a separate line under the program line with the error. If you're running SAS from the unix command line and looking at your log files with the `less` command or the `cat` command, you will probably find the `noovp` option to be helpful.

- The `formdlim='_'` option specifies a "form delimiter" to replace most form feeds (new physical pages) in the list file. This can save a lot of paper (and page printing charges). You can use any string you want for a form delimiter. The underscore

(the one specified here) causes a solid line to be printed instead of going to a new sheet of paper.

- `title` This is optional, but recommended. The material between the single quotes will appear at the top of each page. This can be a lifesaver when you are searching through a stack of old printouts for something you did a year or two ago.

- `data reading;` This begins the data step, specifying the name of the SAS data set that is being created.

- `infile` Specifies the name of the raw data file. The file name, enclosed in single quotes, can be the full unix path to the file, like `/dos/brunner/public/senic.raw`. If you just give the name of the raw data file, as in this example, SAS assumes that the file is in the same directory as the command file.

- `input` Gives the names of the variables.

  - A character variable (the values of `group` are "Treatment' and "Control") must be followed by a dollar sign.
  - Variable names must be eight characters or less, and should begin with a letter. They will be used to request statistical procedures in the `proc` step. They should be meaningful (related to what the variable *is*), and easy to remember.
  - This is almost the simplest form of the `input` statement. It can be very powerful; for example, you can read data from different locations and in different orders, depending on the value of a variable you've just read, and so on. It can get complicated, but if the data file has a simple structure, the input statement can be simple too.

- `label` Provide descriptive labels for the variables; these will be used to label the output, usually in very nice way. Labels can be quite useful, especially when you're trying to recover what you did a while ago. Notice how this statement extends over two physical lines.

- `proc ttest;` Now the proc step begins. This program has only one data step and one proc step. We are requesting a two-sample t-test.

- `class` Specifies the independent variable.

- `var` Specifies the dependent variable(s). You can give a list of dependent variables. A separate univariate test (actually, as you will see, *collection* of tests is performed for each dependent variable.

reading.log Log files are not very interesting when everything is okay, but here is an example anyway. Notice that in addition to a variety of technical information (where the files are, how long each step took, and so on), it contains a listing of the SAS program — in this case, `reading.sas`. If there were syntax errors in the program, this is where the error messages would appear.

```
appsrv01.srv> cat reading.log
1                                                          The SAS System
                          11:40 Thursday, September 2, 2004

NOTE: Copyright (c) 1999-2001 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Release 8.2 (TS2M0)
      Licensed to UNIVERSITY OF TORONTO/COMPUTING & COMMUNICATIONS, Site 0008987001.
NOTE: This session is executing on the Linux 2.6.8.1-smp-athlon-bk platform.



This message is contained in the SAS news file, and is presented upon
initialization.  Edit the files "news" in the "misc/base" directory to
display site-specific news and information in the program log.
The command line option "-nonews" will prevent this display.



NOTE: SAS initialization used:
      real time           0.08 seconds
      cpu time            0.01 seconds

1          /****************** reading.sas *********************
2          *  Simple SAS job to illustrate a two-sample t-test    *
3          ******************************************************/
4
5          options linesize=79 noovp formdlim='_';
6          title 'More & McCabe (1993) textbook t-test Example 7.8';
7
8          data reading;
9               infile 'drp.dat';
10              input group $ score;
11              label group = 'Get Directed Reading Programme?'
12                    score = 'Degree of Reading Power Test Score';

NOTE: The infile 'drp.dat' is:
      File Name=/homes/students/u0/stats/brunner/drp.dat,
      Owner Name=brunner,Group Name=stats,
      Access Permission=rw-r-----,
      File Size (bytes)=660

NOTE: 44 records were read from the infile 'drp.dat'.
      The minimum record length was 14.
      The maximum record length was 14.
NOTE: The data set WORK.READING has 44 observations and 2 variables.
NOTE: DATA statement used:
```

```
      real time           0.01 seconds
      cpu time            0.01 seconds


13         proc ttest;
14              class group;
15              var score;
NOTE: There were 44 observations read from the data set WORK.READING.
NOTE: The PROCEDURE TTEST printed page 1.
NOTE: PROCEDURE TTEST used:
      real time           0.08 seconds
      cpu time            0.01 seconds


NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414

2                                The SAS System
                                              11:40 Thursday, September 2, 2004

NOTE: The SAS System used:
      real time           0.24 seconds
      cpu time            0.03 seconds
```

reading.lst Here is the list file. Notice that the title specified in the `title` statement appears at the top, along with the time and date the program was executed. Then we get statistical output — the t-test we want, and also a bunch of other stuff, whether we want it or not. This is typical of SAS, and most other mainstream statistical packages as well. The default output from any given statistical procedures will contain more information than you wanted, and probably some things you don't understand at all. There are usually numerous options that can add *more* information, but almost never options to reduce the default output. So, you just learn what to ignore. It is helpful, but not essential,to have at least a superficial understanding of everything in the default output from procedures you use a lot.

```
appsrv01.srv> cat reading.lst

--------------------------------------------------------------------------------

              More & McCabe (1993) textbook t-test Example 7.8               1
                                          11:40 Thursday, September 2, 2004

                              The TTEST Procedure

                                  Statistics

                                 Lower CL          Upper CL  Lower CL
  Variable  group           N      Mean    Mean      Mean   Std Dev  Std Dev

  score     Control        23    34.106  41.522    48.937    13.263   17.149
  score     Treatmen       21    46.466  51.476    56.487    8.4213   11.007
  score     Diff (1-2)           -18.82  -9.954    -1.091    11.998   14.551

                                  Statistics

                               Upper CL
       Variable  group         Std Dev    Std Err    Minimum    Maximum

       score     Control       24.271     3.5758         10         85
       score     Treatmen      15.895      2.402         24         71
       score     Diff (1-2)    18.495     4.3919

                                  T-Tests

    Variable    Method          Variances      DF    t Value    Pr > |t|

    score       Pooled          Equal          42      -2.27      0.0286
    score       Satterthwaite   Unequal      37.9      -2.31      0.0264

                          Equality of Variances

        Variable    Method      Num DF    Den DF    F Value    Pr > F

        score       Folded F        22        20       2.43    0.0507
```

Now here are some comments about `reading.lst`.

- The first part of the output is labelled "Statistics," containing confidence intervals and some descriptive statistics. There are three rows to this display: one for the

Control group, one for the Treatment group, and one for the difference between groups.

- `Variable: score` The first column, labelled "Variable," tells you what the dependent variable is – particularly useful if you have more than one.
- `group` The independent variable. Underneath are the values of the independent variable in the first two rows. The third row is for the difference, computed as Control minus Treatment (1-2).

  Well actually, if you look carefully, you see that we do *not* quite get the values of the independent variable under `GROUP`. The values of the (alphanumeric, or character-valued) variable `group` are `Control` and `Treatment`, but the printout says "`Treatmen`." This is not a printing error; it is a subtle error in the reading of the data. The default length of an alphanumeric data value is 8 characters, but "Treatment" has 9 characters. So SAS just read the first eight. No error message was generated and no harm was done in this case, but in other circumstances this error can turn a data file into a giant pile of trash, without warning. Later we will see how to override the default and read longer strings if necessary.
- `N` The third column gives sample sizes; n=23 for the control group, and n=21 for the treatment group.
- The next three columns contain means and their associated 95% confidence intervals. The middle column has the means. For the Control group, the sample mean score on the DRP test was 41.522; for the Treatment group, the sample mean was 51.476. The difference between means is -9.954 = 41.522-51.476 (One minus Two).

  To the left of the mean, labelled "Lower CL Mean," is the lower confidence limit of the 95% confidence interval for the population mean. Thus, the 95% confidence interval for the Treatment mean is from 46.466 to 56.487, and the 95% confidence interval for the *difference* between means is from -18.82 to -1.091. The fact that this last interval does not contain zero means that the usual two-tailed t-test will be statistically significantly at the 0.05 level. This lovely consistency between the classical tests and confidence intervals is actually of no interest in the present course. In fact, we won't pay much attention to confidence intervals at all, but anyway we will continue here in the interest of completeness.
- The next three columns give confidence intervals for the standard deviations. We have the lower confidence limit, the standard deviation, and the upper confidence limit in the continuation below.
- Then we get standard errors (estimated standard deviations of the sample mean or difference between means), and finally the minimum and maximum for each group.

- Then finally, under "T-Tests," we get what we want – a t-test for the difference between the means of the Control and Treatment groups. Two tests are given; as usual there seems to be more output than we were expecting or wishing for.

Probably we were looking for the first one, using the Pooled Method. This is the traditional test, which assumes equal population variances, and therefore is based on a *Pooled* estimate of the common within-groups standard deviation.

- The value of the test statistic is `-2.27`.
- The degrees of freedom $n_1 + n_2 - 2$ is given in the `DF` column.
- The column `Prob>|t|` gives the two-tailed (two-sided) p-value. It is less than the traditional value of 0.05, so the results are statistically significant.

Sample Question 2.2.1 *What do we conclude from this study? Say something about reading, using non-technical language.*

Answer to Sample Question 2.2.1 *Students who received the Directed Reading Program got higher average reading scores than students in the control condition.*

It's worth emphasizing here that the main objective of doing a statistical analysis is to draw conclusions about the data — or to refrain from drawing such conclusions, for good reasons. The question "What do we conclude from this study?" will always be asked. The right answer will always be either "Nothing; the results were not statistically significant," or else it will be something about reading, or fish, or potatoes, or AIDS, or whatever is being studied. Many students, even when they have been warned, respond with a barrage of statistical terminology. They go on and on about the null hypothesis and Type I error, and usually say nothing that would tell a reasonable person what actually happened in the study. In the working world, a memo filled with such garbage could get you fired. Here, it will get you a zero for the question, even if the technical details you give are correct.

Remember, the purpose of writing up a statistical analysis is not to sound impressive and technical, but to impart information. To say things in a simple way is a virtue. It shows you understand what is going on. Now back to the printout.

- The Satterthwaite method gives a sort of t-test that does not assume equal variances. Well, it's not really a t-test, because the test statistic does not really have a t distribution, even when the data are exactly normal. But, the (very unpleasant) distribution of the test statistic is well approximated by a t distribution with the right degrees of freedom — not $n_1 + n_2 - 2$, but something messy that depends on the data. See the odd fractional degrees of freedom? See [3] for details. In any case, it does not matter much in this case, because the p-value is almost the same as the p-value from the traditional test. They lead to the same conclusions, and there is no problem. What should you do when they disagree? I'd go with the test that makes fewer assumptions.

- Next we see a test for Equality of Variances. This "Folded" F is the traditional test for whether the variances of two groups are equal, and it's *almost* significant. This test is provided so people can test for differences between variances; if it is significantly different they can use the unequal variance t-test, and otherwise they can use the traditional test. This seems reasonable, except for the following.

  Both the two-sample t-test and the F-test for equality of variances assume that the data are normally distributed. However, the normality assumption does not

matter much for the t-test when the sample sizes are large, while for the variance test it matters a *lot*, regardless of how much data you have. When the data are non-normal, the test for variances will be significant more than 5% of the time even when the population variances are equal. If you have equal population variances and a large sample of non-normal data, the F-test for variances could easily be significant, leading you to worry unnecessarily about the validity of the t-test.

2.2.5 Background of the First Example

We don't do statistical analysis in a vacuum. Before proceeding with more computing details, let's find out more about the reading data. This first example is from an introductory text. It's Example 7.8 (p. 534) in More and McCabe's excellent *Introduction to the practice of statistics* [3]. We are interested in analyzing *real* data, not in doing textbook exercises. But we will not turn up our noses just yet, because

Data Analysis Hint 2 *When learning how to carry out a procedure using unfamiliar statistical software, always do a textbook example first, and compare the output to the material in the text. Regardless of what the manual might say, never assume you know what the software is doing until you see an example.*

More and McCabe do a great job of explaining the t-test with unequal variances, something SAS produces (along with usual t-test that assumes equal variances) without being asked when you request a t-test. Besides, the data actually come from someone's Ph.D. thesis, so there is an element of realism. Here is Moore and McCabe's description of the study.

> An educator believes that new directed reading activities in the classroom will help elementary school pupils improve some aspects of their reading ability. She arranges for a third grade class of 21 students to take part in these activities. A control classroom of 23 third graders follows the same curriculum without the activities. At the end of 8 weeks, all students are given a Degree of Reading Power (DRP) test, which measures the aspects of reading ability that the program is designed to improve.

Sample Question 2.2.2 *What's wrong with this study?*

Answer to Sample Question 2.2.2 *The independent variable was manipulated by the experimenter, but it is not an experimental study. Even if classrooms were assigned randomly to conditions (it is impossible to tell whether they were, from this brief description), a large number of unobserved variables are potentially confounded with treatment. The teacher in the classroom that received the treatment might be better than the teacher in the control classroom, or possibly there was a particularly aggressive bully in the control classroom, or maybe a mini-epidemic of some childhood disease hit the control classroom*

$\vdots$ *The list goes on. The point here is that there are many ways in which the classroom experiences of children in the treatment group differ systematically from the experiences of children in the control group.*

Sample Question 2.2.3 *How could the problem be fixed?*

Answer to Sample Question 2.2.3 *Assign classrooms at random to treatments. The unit of analysis should be the classroom, not the individual student.*

2.2.6 SAS Example Two: The statclass data

These data come from a statistics class taught many years ago. Students took eight quizzes, turned in nine computer assignments, and also took a midterm and final exam. The data file also includes gender and ethnic background; these last two variables are just guesses by the professor, and there is no way to tell how accurate they were. The data file looks like this. There are 21 columns and 62 rows of data; columns not aligned. Here are the first few lines.

```
appsrv01.srv>  less statclass1.dat
1  2  9  1  7  8  4  3  5  2  6  10  10  10  5  0  0  0  0  55  43
0  2  10  10  5  9  10  8  6  8  10  10  8  9  9  9  9  10  10  66  79
1  2  10  10  5  10  10  10  9  8  10  10  10  10  10  10  9  10  10  94  67
1  2  10  10  8  9  10  7  10  9  10  10  10  9  10  10  9  10  10  81  65
0  1  10  1  0  0  8  6  5  2  10  9  0  0  10  6  0  5  0  54  .
1  1  10  6  7  9  8  8  5  7  10  9  10  9  5  6  4  8  10  57  52
0  1  0  0  9  9  10  5  2  2  8  7  7  10  10  6  3  7  10  49  .
0  1  10  9  5  8  9  8  5  6  8  7  5  6  10  6  5  9  9  77  64
0  1  10  8  6  8  9  5  3  6  9  9  6  9  10  6  5  7  10  65  42
1  1  10  5  6  7  10  4  6  0  10  9  10  9  10  6  7  8  10  73  .
0  1  9  0  4  6  10  5  3  3  10  8  10  5  10  10  9  9  10  71  37

⋮
```

Notice the periods at the ends of lines 5, 7 and 10. The period is the SAS *missing value code*. These people did not show up for the final exam. They may have taken a makeup exam, but if so their scores did not make it into this data file. When a case has a missing value recorded for a variable, SAS automatically excludes that case from any statistical calculation involving the variable. If a new variable is being created based on the value of a variable with a missing value, the new variable will usually have a missing value for that case too.

Here is the SAS program textttstatmarks1.sas. It reads and labels the data, and then does a variety of significance tests. They are all elementary except the last one, which illustrates testing for one set of independent variables controlling for another set in multiple regression.

```
appsrv01.srv>  cat statmarks1.sas

                  /* statmarks1.sas */
options linesize=79 noovp formdlim='_';
title 'Grades from STA3000 at Roosevelt University:  Fall, 1957';
title2 'Illustrate Elementary Tests';

proc format; /* Used to label values of the categorical variables */
     value sexfmt    0 = 'Male'   1 = 'Female';
     value ethfmt    1 = 'Chinese'
                     2 = 'European'
                     3 = 'Other' ;
data grades;
     infile 'statclass1.dat';
     input sex ethnic quiz1-quiz8 comp1-comp9 midterm final;
     /* Drop lowest score for quiz & computer  */
     quizave = ( sum(of quiz1-quiz8) - min(of quiz1-quiz8) ) / 7;
     compave = ( sum(of comp1-comp9) - min(of comp1-comp9) ) / 8;
     label ethnic = 'Apparent ethnic background (ancestry)'
           quizave = 'Quiz Average (drop lowest)'
           compave = 'Computer Average (drop lowest)';
     mark = .3*quizave*10 + .1*compave*10 + .3*midterm + .3*final;
     label mark = 'Final Mark';
     diff = quiz8-quiz1; /* To illustrate matched t-test */
     label diff = 'Quiz 8 minus Quiz 1';
     mark2 = round(mark);
     /* Bump up at grade boundaries */
     if mark2=89 then mark2=90;
     if mark2=79 then mark2=80;
     if mark2=69 then mark2=70;
     if mark2=59 then mark2=60;
     /* Assign letter grade */
     if mark2=. then grade='Incomplete';
        else if mark2 ge 90 then grade = 'A';
        else if 80 le mark2 le 89 then grade='B';
        else if 70 le mark2 le 79 then grade='C';
        else if 60 le mark2 le 69 then grade='D';
        else grade='F';
     format sex sexfmt.;       /* Associates sex & ethnic    */
     format ethnic ethfmt.;    /* with formats defined above */

/* Now the proc steps */


proc freq;
      title3 'Frequency distributions of the categorical variables';
      tables sex ethnic grade;
```

```
proc means n mean std;
     title3 'Means and SDs of quantitative variables';
     var quiz1 -- mark;        /*  single dash only works with numbered
                                  lists, like quiz1-quiz8    */
proc ttest;
     title3 'Independent t-test';
     class sex;
     var mark;
proc means n mean std t;
     title3 'Matched t-test: Quiz 1 versus 8';
     var quiz1 quiz8 diff;
proc glm;
     title3 'One-way anova';
     class ethnic;
     model mark = ethnic;
     means ethnic;
     means ethnic / Tukey Bon Scheffe;
proc freq;
     title3 'Chi-squared Test of Independence';
     tables sex*ethnic sex*grade ethnic*grade / chisq;
proc freq; /*  Added after seeing warning from chisq test above */
     title3 'Chi-squared Test of Independence: Version 2';
     tables sex*ethnic grade*(sex ethnic) / norow nopercent chisq expected;
proc corr;
     title3 'Correlation Matrix';
     var final midterm quizave compave;
proc plot;
     title3 'Scatterplot';
     plot final*midterm; /* Really should do all combinations */
proc reg;
     title3 'Simple regression';
     model final=midterm;

/* Predict final exam score from midterm, quiz & computer */
proc reg simple;
     title3 'Multiple Regression';
     model final = midterm quizave compave / ss1;
     smalstuf: test quizave = 0, compave = 0;
run;

/* Note that the final run statement is not needed when
   running SAS from the unix command line. */
```

Noteworthy features of this program include

- `options`: Already discussed in connection with `reading.sas`.

- `title2`: Subtitle
- `proc format`: This is a non-statistical procedure – a rarity in the SAS language. It is the way SAS takes care of labelling categorical variables when the categories are coded as numbers. `proc format` defines *printing formats*. For any variable associated with the printing format named `sexfmt`, any time it would print the value "0" (in a table or something) it instead prints the string "Male." The associations between variables and printing formats are accomplished in the `format` statement at the end of the data step. The names of formats have a period at the end to distinguish them from variable names. Of course formats must be defined before they can be associated with variables. This is why `proc format` precedes the data step.
- `quiz1-quiz8`: One may refer to a *range* of variables ending with consecutive numbers using a minus sign. In the `input` statement, a range can be defined (named) this way. It saves typing and is easy to read.
- Creating new variables with assignment statements. The variables `quizave`,`compave` and `mark` are not in the original data file. They are created here, and they are appended to the end of the SAS data set in oder of creation. Variables like this should never be in the raw data file.

  Data Analysis Hint 3 *When variables are exact mathematical functions of other variables, always create them in the* ***`data`*** *step rather than including them in the raw data file. It saves data entry, and makes the data file smaller and easier to read. If you want to try out a different definition of the variable, it's easy to change a few statements in the* ***`data`*** *step.*

- `sum(of quiz1-quiz8)`: Without the word "of," the minus sign is ambiguous. In the SAS language, `sum(quiz1-quiz8)` is the sum of a single number, the difference between `quiz1` and `quiz8`.
- `format sex sexfmt.;` Associates the variable `sex` with its printing format. In questionnaire studies where a large number of items have the same potential responses (like a scale from 1 = Strongly Agree to 7=Strongly Disagree), it is common to associate a long list of variables with a single printing format.
- `quiz1 -- mark` in the first `proc means`
- Title inside a procedure labels just that procedure.
- `proc means n mean std t` A matched t-test is just a single-variable t-test carried out on differences, testing whether the mean difference is equal to zero.
- `proc glm`
  - `class` Tells SAS that the IV ethnic is categorical.
  - `model` Dependent variable(s) = independent variable(s)
  - `means ethnic`: Mean of `mark` separately for each value of `ethnic`.

  - `means ethnic / Tukey Bon Scheffe`: Post hoc tests (multiple comparisons, probing, follow-ups). Used if the overall F-test is significant, to see which means are different from which other means.

- `chisq` option on `proc freq`: Gives a large collection of chisquare tests. The first one is the familiar Pearson chisquare test of independence (the one comparing observed and expected frequencies).

- `tables sex*ethnic / norow nopercent chisq expected;` In version 2 of `proc freq`

- `proc corr`

- `proc plot; plot final*midterm;` Scatterplot: First variable named goes on the y axis.

- `proc reg`: `model` Dependent variable(s) = independent variable(s) again

- `simple` option on `proc reg` gives simple descriptive statistics. This last procedure is an example of multiple regression, and we will return to it later once we have more background.

statmarks1.lst

```
--------------------------------------------------------------------------------

          Grades from STA3000 at Roosevelt University:  Fall, 1957          1
                        Illustrate Elementary Tests
            Frequency distributions of the categorical variables
                                          10:10 Sunday, September 5, 2004

                          The FREQ Procedure

                                                Cumulative    Cumulative
            sex    Frequency     Percent     Frequency      Percent
           -----------------------------------------------------------
           Male            39       62.90            39        62.90
           Female          23       37.10            62       100.00


                    Apparent ethnic background (ancestry)

                                                Cumulative    Cumulative
             ethnic    Frequency     Percent     Frequency      Percent
            -----------------------------------------------------------
            Chinese           41       66.13            41        66.13
            European          15       24.19            56        90.32
            Other              6        9.68            62       100.00


                                                Cumulative    Cumulative
          grade         Frequency     Percent     Frequency      Percent
          -----------------------------------------------------------------
          A                     3        4.84             3         4.84
          B                     6        9.68             9        14.52
          C                    18       29.03            27        43.55
          D                    21       33.87            48        77.42
          F                    10       16.13            58        93.55
          Incomplete            4        6.45            62       100.00

--------------------------------------------------------------------------------
```

The MEANS Procedure

Variable	Label	N	Mean	Std Dev
quiz1		62	9.0967742	2.2739413
quiz2		62	5.8870968	3.2294995
quiz3		62	6.0483871	2.3707744
quiz4		62	7.7258065	2.1590022
quiz5		62	9.0645161	1.4471109
quiz6		62	7.1612903	1.9264641
quiz7		62	5.7903226	2.1204477
quiz8		62	6.3064516	2.3787909
comp1		62	9.1451613	1.1430011
comp2		62	8.8225806	1.7604414
comp3		62	8.3387097	2.5020880
comp4		62	7.8548387	3.2180168
comp5		62	9.4354839	1.7237109
comp6		62	7.8548387	2.4350364
comp7		62	6.6451613	2.7526248
comp8		62	8.8225806	1.6745363
comp9		62	8.2419355	3.7050497
midterm		62	70.1935484	13.6235557
final		58	50.3103448	17.2496701
quizave	Quiz Average (drop lowest)	62	7.6751152	1.1266917
compave	Computer Average (drop lowest)	62	8.8346774	1.1204997
mark	Final Mark	58	68.4830049	10.3902874

The TTEST Procedure

Statistics

Variable	sex	N	Lower CL Mean	Mean	Upper CL Mean	Lower CL Std Dev	Std Dev
mark	Male	36	65.604	68.57	71.535	7.1093	8.7653
mark	Female	22	62.647	68.341	74.036	9.8809	12.843
mark	Diff (1-2)		-5.454	0.2284	5.9108	8.8495	10.482

Statistics

Variable	sex	Upper CL Std Dev	Std Err	Minimum	Maximum
mark	Male	11.434	1.4609	54.057	89.932
mark	Female	18.354	2.7382	48.482	95.457
mark	Diff (1-2)	12.859	2.8366		

T-Tests

Variable	Method	Variances	DF	t Value	Pr > |t|
mark	Pooled	Equal	56	0.08	0.9361
mark	Satterthwaite	Unequal	33.1	0.07	0.9418

Equality of Variances

Variable	Method	Num DF	Den DF	F Value	Pr > F
mark	Folded F	21	35	2.15	0.0443

Grades from STA3000 at Roosevelt University: Fall, 1957 4
Illustrate Elementary Tests
Matched t-test: Quiz 1 versus 8
10:10 Sunday, September 5, 2004

The MEANS Procedure

Variable	Label	N	Mean	Std Dev	t Value
quiz1		62	9.0967742	2.2739413	31.50
quiz8		62	6.3064516	2.3787909	20.87
diff	Quiz 8 minus Quiz 1	62	-2.7903226	3.1578011	-6.96

Grades from STA3000 at Roosevelt University: Fall, 1957 5
Illustrate Elementary Tests
One-way anova 10:10 Sunday, September 5, 2004

The GLM Procedure

Class Level Information

Class	Levels	Values
ethnic	3	Chinese European Other

Number of observations 62

NOTE: Due to missing values, only 58 observations can be used in this analysis.

Grades from STA3000 at Roosevelt University: Fall, 1957 6
Illustrate Elementary Tests
One-way anova 10:10 Sunday, September 5, 2004

The GLM Procedure

Dependent Variable: mark Final Mark

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	1238.960134	619.480067	6.93	0.0021
Error	55	4914.649951	89.357272		
Corrected Total	57	6153.610084			

R-Square	Coeff Var	Root MSE	mark Mean
0.201339	13.80328	9.452898	68.48300

Source	DF	Type I SS	Mean Square	F Value	Pr > F
ethnic	2	1238.960134	619.480067	6.93	0.0021

Source	DF	Type III SS	Mean Square	F Value	Pr > F

```
ethnic                    2    1238.960134     619.480067      6.93   0.0021
```

```
                           The GLM Procedure

              Level of            -------------mark------------
              ethnic        N             Mean          Std Dev

              Chinese      37       65.2688224        7.9262171
              European     15       76.0142857       11.2351562
              Other         6       69.4755952       13.3097753
```

```
                           The GLM Procedure

                Tukey's Studentized Range (HSD) Test for mark

       NOTE: This test controls the Type I experimentwise error rate.


                Alpha                                    0.05
                Error Degrees of Freedom                   55
                Error Mean Square                     89.35727
                Critical Value of Studentized Range   3.40649


       Comparisons significant at the 0.05 level are indicated by ***.


                              Difference
                    ethnic       Between     Simultaneous 95%
                   Comparison      Means    Confidence Limits

             European - Other      6.539      -4.460   17.538
             European - Chinese   10.745       3.776   17.715  ***
             Other    - European  -6.539     -17.538    4.460
             Other    - Chinese    4.207      -5.814   14.228
             Chinese  - European -10.745     -17.715   -3.776  ***
             Chinese  - Other     -4.207     -14.228    5.814
```

```
                           The GLM Procedure

                    Bonferroni (Dunn) t Tests for mark

   NOTE: This test controls the Type I experimentwise error rate, but it
  generally has a higher Type II error rate than Tukey's for all pairwise
                              comparisons.


                        Alpha                      0.05
                        Error Degrees of Freedom     55
                        Error Mean Square       89.35727
                        Critical Value of t      2.46941
```

Comparisons significant at the 0.05 level are indicated by ***.

ethnic Comparison	Difference Between Means	Simultaneous 95% Confidence Limits		
European - Other	6.539	-4.737	17.814	
European - Chinese	10.745	3.600	17.891	***
Other - European	-6.539	-17.814	4.737	
Other - Chinese	4.207	-6.067	14.480	
Chinese - European	-10.745	-17.891	-3.600	***
Chinese - Other	-4.207	-14.480	6.067	

The GLM Procedure

Scheffe's Test for mark

NOTE: This test controls the Type I experimentwise error rate, but it generally has a higher Type II error rate than Tukey's for all pairwise comparisons.

Alpha	0.05
Error Degrees of Freedom	55
Error Mean Square	89.35727
Critical Value of F	3.16499

Comparisons significant at the 0.05 level are indicated by ***.

ethnic Comparison	Difference Between Means	Simultaneous 95% Confidence Limits		
European - Other	6.539	-4.950	18.027	
European - Chinese	10.745	3.466	18.025	***
Other - European	-6.539	-18.027	4.950	
Other - Chinese	4.207	-6.260	14.674	
Chinese - European	-10.745	-18.025	-3.466	***
Chinese - Other	-4.207	-14.674	6.260	

The FREQ Procedure

Table of sex by ethnic

sex ethnic(Apparent ethnic background (ancestry))

Frequency| Percent | Row Pct | Col Pct	Chinese	European	Other	Total
Male	27	7	5	39
	43.55	11.29	8.06	62.90
	69.23	17.95	12.82	

```
                |  65.85 |  46.67 |  83.33 |
         ---------+--------+--------+--------+
         Female   |     14 |      8 |      1 |     23
                  |  22.58 |  12.90 |   1.61 |  37.10
                  |  60.87 |  34.78 |   4.35 |
                  |  34.15 |  53.33 |  16.67 |
         ---------+--------+--------+--------+
         Total          41       15        6       62
                     66.13    24.19     9.68   100.00


              Statistics for Table of sex by ethnic

        Statistic                     DF       Value      Prob
        ------------------------------------------------------
        Chi-Square                     2      2.9208    0.2321
        Likelihood Ratio Chi-Square    2      2.9956    0.2236
        Mantel-Haenszel Chi-Square     1      0.0000    0.9949
        Phi Coefficient                       0.2170
        Contingency Coefficient               0.2121
        Cramer's V                            0.2170

         WARNING: 33% of the cells have expected counts less
                  than 5. Chi-Square may not be a valid test.

                           Sample Size = 62
```

```
                          The FREQ Procedure

                          Table of sex by grade

    sex       grade

    Frequency|
    Percent  |
    Row Pct  |
    Col Pct  |A       |B       |C       |D       |F       |Incomple|  Total
             |        |        |        |        |        |te      |
    ---------+--------+--------+--------+--------+--------+--------+
    Male     |      1 |      3 |     13 |     14 |      5 |      3 |     39
             |   1.61 |   4.84 |  20.97 |  22.58 |   8.06 |   4.84 |  62.90
             |   2.56 |   7.69 |  33.33 |  35.90 |  12.82 |   7.69 |
             |  33.33 |  50.00 |  72.22 |  66.67 |  50.00 |  75.00 |
    ---------+--------+--------+--------+--------+--------+--------+
    Female   |      2 |      3 |      5 |      7 |      5 |      1 |     23
             |   3.23 |   4.84 |   8.06 |  11.29 |   8.06 |   1.61 |  37.10
             |   8.70 |  13.04 |  21.74 |  30.43 |  21.74 |   4.35 |
             |  66.67 |  50.00 |  27.78 |  33.33 |  50.00 |  25.00 |
    ---------+--------+--------+--------+--------+--------+--------+
    Total          3        6       18       21       10        4       62
                4.84     9.68    29.03    33.87    16.13     6.45   100.00


                  Statistics for Table of sex by grade

        Statistic                     DF       Value      Prob
        ------------------------------------------------------
        Chi-Square                     5      3.3139    0.6517
        Likelihood Ratio Chi-Square    5      3.2717    0.6582
        Mantel-Haenszel Chi-Square     1      0.2342    0.6284
        Phi Coefficient                       0.2312
        Contingency Coefficient               0.2253
```

Cramer's V	0.2312

WARNING: 58% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 62

Illustrate Elementary Tests
Chi-squared Test of Independence
10:10 Sunday, September 5, 2004

The FREQ Procedure

Table of ethnic by grade

ethnic(Apparent ethnic background (ancestry)) grade

Frequency / Percent / Row Pct / Col Pct	A	B	C	D	F	Incomplete	Total
Chinese	0	2	11	17	7	4	41
	0.00	3.23	17.74	27.42	11.29	6.45	66.13
	0.00	4.88	26.83	41.46	17.07	9.76	
	0.00	33.33	61.11	80.95	70.00	100.00	
European	2	4	5	3	1	0	15
	3.23	6.45	8.06	4.84	1.61	0.00	24.19
	13.33	26.67	33.33	20.00	6.67	0.00	
	66.67	66.67	27.78	14.29	10.00	0.00	
Other	1	0	2	1	2	0	6
	1.61	0.00	3.23	1.61	3.23	0.00	9.68
	16.67	0.00	33.33	16.67	33.33	0.00	
	33.33	0.00	11.11	4.76	20.00	0.00	
Total	3	6	18	21	10	4	62
	4.84	9.68	29.03	33.87	16.13	6.45	100.00

Statistics for Table of ethnic by grade

Statistic	DF	Value	Prob
Chi-Square	10	18.2676	0.0506
Likelihood Ratio Chi-Square	10	19.6338	0.0329
Mantel-Haenszel Chi-Square	1	5.6222	0.0177
Phi Coefficient		0.5428	
Contingency Coefficient		0.4771	
Cramer's V		0.3838	

WARNING: 78% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 62

Illustrate Elementary Tests
Chi-squared Test of Independence: Version 2
10:10 Sunday, September 5, 2004

The FREQ Procedure

Table of sex by ethnic

sex ethnic(Apparent ethnic background (ancestry))

Frequency| Expected | Col Pct	Chinese	European	Other	Total
Male	27	7	5	39
	25.79	9.4355	3.7742	
	65.85	46.67	83.33	
Female	14	8	1	23
	15.21	5.5645	2.2258	
	34.15	53.33	16.67	
Total	41	15	6	62

Statistics for Table of sex by ethnic

Statistic	DF	Value	Prob
Chi-Square	2	2.9208	0.2321
Likelihood Ratio Chi-Square	2	2.9956	0.2236
Mantel-Haenszel Chi-Square	1	0.0000	0.9949
Phi Coefficient		0.2170	
Contingency Coefficient		0.2121	
Cramer's V		0.2170	

WARNING: 33% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 62

Illustrate Elementary Tests

Chi-squared Test of Independence: Version 2

10:10 Sunday, September 5, 2004

The FREQ Procedure

Table of grade by sex

grade sex

Frequency | Expected | Col Pct	Male	Female	Total
A	1	2	3
	1.8871	1.1129	
	2.56	8.70	
B	3	3	6
	3.7742	2.2258	
	7.69	13.04	
C	13	5	18
	11.323	6.6774	
	33.33	21.74	
D	14	7	21
	13.21	7.7903	
	35.90	30.43	

F	5	5	10
	6.2903	3.7097	
	12.82	21.74	
Incomplete	3	1	4
	2.5161	1.4839	
	7.69	4.35	
Total	39	23	62

Statistics for Table of grade by sex

Statistic	DF	Value	Prob
Chi-Square	5	3.3139	0.6517
Likelihood Ratio Chi-Square	5	3.2717	0.6582
Mantel-Haenszel Chi-Square	1	0.2342	0.6284
Phi Coefficient		0.2312	
Contingency Coefficient		0.2253	
Cramer's V		0.2312	

WARNING: 58% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 62

The FREQ Procedure

Table of grade by ethnic

grade ethnic(Apparent ethnic background (ancestry))

Frequency / Expected / Col Pct	Chinese	European	Other	Total
A	0	2	1	3
	1.9839	0.7258	0.2903	
	0.00	13.33	16.67	
B	2	4	0	6
	3.9677	1.4516	0.5806	
	4.88	26.67	0.00	
C	11	5	2	18
	11.903	4.3548	1.7419	
	26.83	33.33	33.33	
D	17	3	1	21
	13.887	5.0806	2.0323	
	41.46	20.00	16.67	
F	7	1	2	10
	6.6129	2.4194	0.9677	
	17.07	6.67	33.33	
Incomplete	4	0	0	4
	2.6452	0.9677	0.3871	
	9.76	0.00	0.00	
Total	41	15	6	62

Statistics for Table of grade by ethnic

Statistic	DF	Value	Prob
Chi-Square	10	18.2676	0.0506
Likelihood Ratio Chi-Square	10	19.6338	0.0329
Mantel-Haenszel Chi-Square	1	5.6222	0.0177
Phi Coefficient		0.5428	
Contingency Coefficient		0.4771	
Cramer's V		0.3838	

WARNING: 78% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 62

Grades from STA3000 at Roosevelt University: Fall, 1957 17
Illustrate Elementary Tests
Correlation Matrix
10:10 Sunday, September 5, 2004

The CORR Procedure

4 Variables: final midterm quizave compave

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
final	58	50.31034	17.24967	2918	15.00000	89.00000
midterm	62	70.19355	13.62356	4352	44.00000	103.00000
quizave	62	7.67512	1.12669	475.85714	4.57143	9.71429
compave	62	8.83468	1.12050	547.75000	5.00000	10.00000

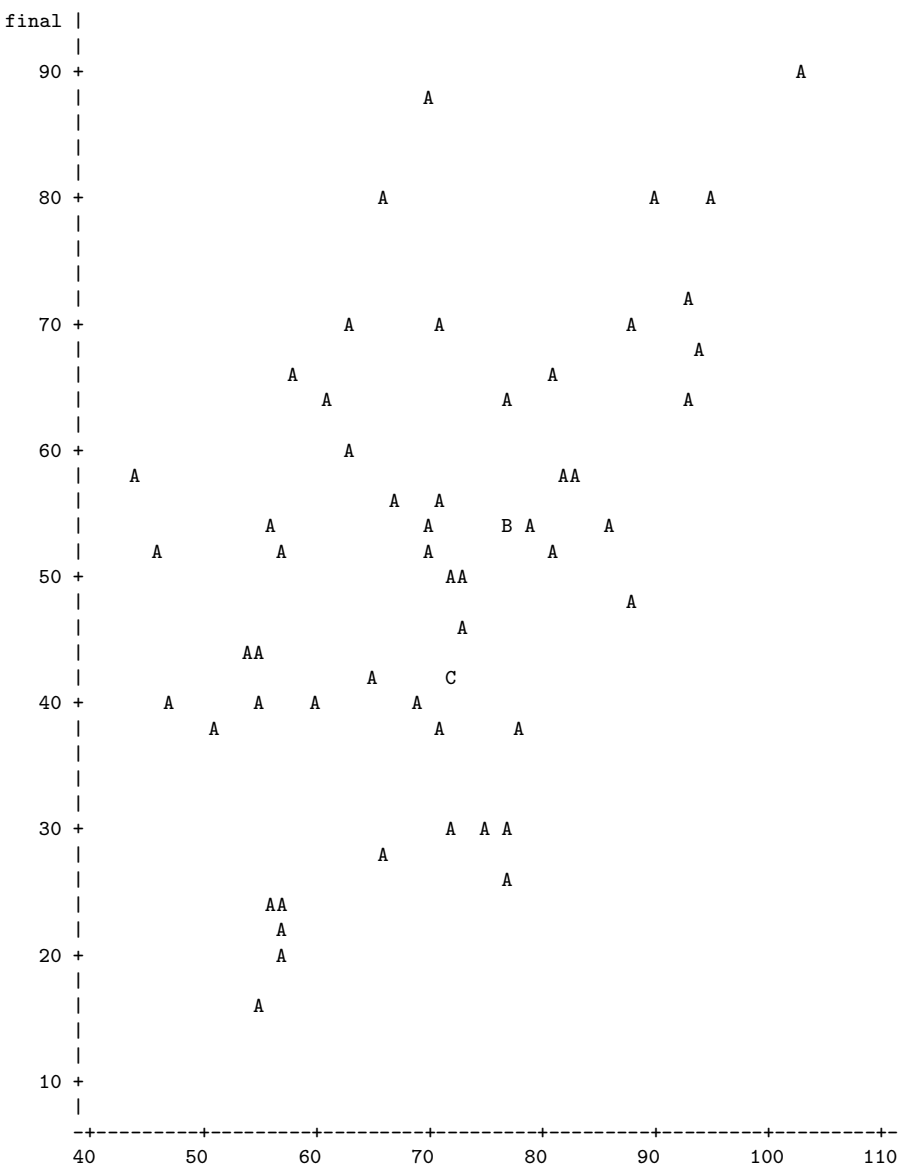
Simple Statistics

Variable	Label
final	
midterm	
quizave	Quiz Average (drop lowest)
compave	Computer Average (drop lowest)

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

	final	midterm	quizave	compave
final	1.00000	0.47963	0.41871	0.06060
		0.0001	0.0011	0.6513
	58	58	58	58
midterm	0.47963	1.00000	0.59294	0.41277
	0.0001		<.0001	0.0009
	58	62	62	62
quizave	0.41871	0.59294	1.00000	0.52649
Quiz Average (drop lowest)	0.0011	<.0001		<.0001
	58	62	62	62
compave	0.06060	0.41277	0.52649	1.00000
Computer Average (drop lowest)	0.6513	0.0009	<.0001	
	58	62	62	62

Plot of final*midterm. Legend: A = 1 obs, B = 2 obs, etc.



NOTE: 4 obs had missing values.

The REG Procedure
Model: MODEL1
Dependent Variable: final

Analysis of Variance

Sum of Mean

Source	DF	Squares	Square	F Value	Pr > F
Model	1	3901.64751	3901.64751	16.73	0.0001
Error	56	13059	233.19226		
Corrected Total	57	16960			

Root MSE	15.27063	R-Square	0.2300
Dependent Mean	50.31034	Adj R-Sq	0.2163
Coeff Var	30.35287		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	6.88931	10.80304	0.64	0.5263
midterm	1	0.61605	0.15061	4.09	0.0001

The REG Procedure

Descriptive Statistics

Variable	Sum	Mean	Uncorrected SS	Variance	Standard Deviation
Intercept	58.00000	1.00000	58.00000	0	0
midterm	4088.00000	70.48276	298414	180.35935	13.42979
quizave	451.57143	7.78571	3576.51020	1.06498	1.03198
compave	515.50000	8.88793	4641.50000	1.04862	1.02402
final	2918.00000	50.31034	163766	297.55112	17.24967

Descriptive Statistics

Variable	Label
Intercept	Intercept
midterm	
quizave	Quiz Average (drop lowest)
compave	Computer Average (drop lowest)
final	

The REG Procedure
Model: MODEL1
Dependent Variable: final

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	4995.04770	1665.01590	7.51	0.0003
Error	54	11965	221.58085		
Corrected Total	57	16960			

```
           Root MSE            14.88559    R-Square     0.2945
           Dependent Mean      50.31034    Adj R-Sq     0.2553
           Coeff Var           29.58754


                          Parameter Estimates

                                                    Parameter       Standard
  Variable   Label                             DF     Estimate          Error

  Intercept  Intercept                          1      9.01839       19.02591
  midterm                                       1      0.50057        0.18178
  quizave    Quiz Average (drop lowest)         1      4.80199        2.46469
  compave    Computer Average (drop lowest)     1     -3.53028        2.17562

                          Parameter Estimates

 Variable   Label                           DF  t Value  Pr > |t|    Type I SS

 Intercept  Intercept                        1     0.47    0.6374       146806
 midterm                                     1     2.75    0.0080   3901.64751
 quizave    Quiz Average (drop lowest)       1     1.95    0.0566    509.97483
 compave    Computer Average (drop lowest)   1    -1.62    0.1105    583.42537

-------------------------------------------------------------------------------

          Grades from STA3000 at Roosevelt University:  Fall, 1957          22
                        Illustrate Elementary Tests
                            Multiple Regression
                                        10:10 Sunday, September 5, 2004

                           The REG Procedure
                             Model: MODEL1

             Test smalstuf Results for Dependent Variable final

                                       Mean
          Source             DF         Square    F Value    Pr > F

          Numerator           2      546.70010       2.47    0.0943
          Denominator        54      221.58085
```

Data in fixed columns When the data values have at least one space between them, the variables are recorded in the same order for each case, and missing values are indicated by periods, the default version of the `input` statement (list input) does the job perfectly. It is a bonus that the variables need not always be separated by the same number of spaces for each case. Also, there can be more than one line of data for each case, and in fact there need not even be the same number of data lines for all the cases, just as long as there are the same number of variables.

Another common situation is for the data to be lined up in fixed columns, with blanks for missing values. Sometimes, especially when there are many variables, the data are *packed* together, without spaces between values. For example, the Minnesota Multiphasic Personality Inventory (MMPI) consists of over 300 questions, all to be answered True or False. It would be quite natural to code 1=True and 0=False, and pack the data together. There would still be quite a few data lines for each case.

Here is the beginning of the file `statclass2.dat`. It is the same as `statclass1.dat`, except that the data are packed together. Most of the blanks occur because two columns are reserved for the marks on quizzes and computer assignments, because 10 out of 10 is possible. Three columns are reserved for the midterm and final scores, because 100% is possible. For all variables, missing values are represented by blanks. That is, if the field

occupied by a variable is completely blank, it's a missing value.

```
appsrv01.srv>  less statclass2.dat
12 9 1 7 8 4 3 5 2 6101010 5 0 0 0 0  55 43
021010 5 910 8 6 81010 8 9 9 9 91010  66 79
121010 5101010 9 8101010101010 91010  94 67
121010 8 910 710 9101010 91010 91010  81 65
0110 1 0 0 8 6 5 210 9 0 010 6 0 5 0  54
1110 6 7 9 8 8 5 710 910 9 5 6 4 810  57 52
01 0 0 9 910 5 2 2 8 7 71010 6 3 710  49
0110 9 5 8 9 8 5 6 8 7 5 610 6 5 9 9  77 64
0110 8 6 8 9 5 3 6 9 9 6 910 6 5 710  65 42
1110 5 6 710 4 6 010 910 910 6 7 810  73
01 9 0 4 610 5 3 310 810 51010 9 910  71 37

⋮
```

Now we will take a look at `statread.sas`. It contains just the `proc format` and the `data` step; There are no statistical procedures. This file will be read by programs that invoke statistical procedures, as you will see.

```
                              /* statread.sas
Read the statclass data in fixed format, define and label variables. Use
with     %include 'statread.sas';   */

options linesize=79 noovp formdlim='_';
title 'Grades from STA3000 at Roosevelt University:  Fall, 1957';

proc format; /* Used to label values of the categorical variables */
     value sexfmt    0 = 'Male'   1 = 'Female';
     value ethfmt    1 = 'Chinese'
                     2 = 'European'
                     3 = 'Other' ;
data grades;
     infile 'statclass2.dat' missover;
     input (sex ethnic) (1.)
           (quiz1-quiz8 comp1-comp9) (2.)
           (midterm final) (3.);
     /* Drop lowest score for quiz & computer  */
     quizave = ( sum(of quiz1-quiz8) - min(of quiz1-quiz8) ) / 7;
     compave = ( sum(of comp1-comp9) - min(of comp1-comp9) ) / 8;
     label ethnic = 'Apparent ethnic background (ancestry)'
           quizave = 'Quiz Average (drop lowest)'
           compave = 'Computer Average (drop lowest)';
     mark = .3*quizave*10 + .1*compave*10 + .3*midterm + .3*final;
     label mark = 'Final Mark';
```

```
    diff = quiz8-quiz1; /* To illustrate matched t-test */
    label diff = 'Quiz 8 minus Quiz 1';
    mark2 = round(mark);
    /* Bump up at grade boundaries */
    if mark2=89 then mark2=90;
    if mark2=79 then mark2=80;
    if mark2=69 then mark2=70;
    if mark2=59 then mark2=60;
    /* Assign letter grade */
    if mark2=. then grade='Incomplete';
       else if mark2 ge 90 then grade = 'A';
       else if 80 le mark2 le 89 then grade='B';
       else if 70 le mark2 le 79 then grade='C';
       else if 60 le mark2 le 69 then grade='D';
       else grade='F';
    format sex sexfmt.;       /* Associates sex & ethnic    */
    format ethnic ethfmt.;    /* with formats defined above */

/*************************************************************/
```

The data step in `statread.sas` differs from the one in `statmarks1.sas` in only two respects. First, the `missover` option on the infile statement causes blanks to be read as missing values even if they occur at the end of a line and the line just ends rather than being filled in with space characters. That is, such lines are shorter than the others in the file, and when SAS `over`-reads the end of the line, it sets all the variables it would have read to missing. This is what we want, so you should always use the `missover` option when missing values are represented by blanks.

The other difference between this data step and the one in `statmarks1.sas` is in the `input` statement. Here, we are using *formatted* input. `sex` and `ethnic` each occupy 1 column. `quiz1-quiz8` and `comp1-comp9` each occupy 2 columns. `midterm` and `final` each occupy 3 columns. You can supply a list of formats for each list of variables in parentheses, but if the number of formats is less than the number of variables, they are re-used. That's what's happening in the present case.

The program `statread.sas` reads and defines the data, but it requests no statistical output; `statdescribe.sas` pulls in `statread.sas` using a `%include` statement, and produces basic descriptive statistics. Significance tests would be produced by other short programs.

Keeping the data definition in a separate file and using `%include` (the only part of the powerful *SAS macro language* presented here) is often a good strategy, because most data analysis projects involve a substantial number of statistical procedures. It is common to have maybe twenty program files that carry out various analyses. You *could* have the data step at the beginning of each program, but in many cases the data step is long. And, what happens when (inevitably) you want to make a change in the data step and re-run your analyses? You find yourself making the same change in twenty files. Probably you will forget to change some of them, and the result is a big mess. If you keep your data definition in just one place, you only have to edit it once, and a lot of problems are avoided.

```
                                /* statdescribe.sas */
%include 'statread.sas';
title2 'Basic Descriptive Statistics';

proc freq;
      title3 'Frequency distributions of the categorical variables';
      tables sex ethnic grade;

proc means n mean std;
     title3 'Means and SDs of quantitative variables';
     var quiz1 -- mark2;        /*  single dash only works with numbered
                                   lists, like quiz1-quiz8    */

proc univariate normal; /* the normal option gives a test for normality */
     title3 'Detailed look at mark and bumped mark (mark2)';
     var mark mark2;
```

2.2.7 SAS Reference Materials

This course is trying to teach you SAS by example, without full explanation, and certainly without discussion of all the options. If you need more detail, there are several approaches you can take. The most obvious is to consult the SAS manuals. The full set of manuals runs to over a dozen volumes, and most of them look like telephone directories. For a beginner, it is hard to know where to start. And even if you know where to look, the SAS manuals can be hard to read, because they assume you already understand the statistical procedures fairly thoroughly, and on a mathematical level. They are really written for professional statisticians. The SAS Institute also publishes a variety of manual-like books that are intended to be more instructional, most of them geared to specific topics (like *The SAS system for multiple regression* and *The SAS system for linear models*). These are a bit more readable, though it helps to have a real textbook on the topic to fill in the gaps.

A better place to start is a wonderful book by Cody and Smith [2] entitled *Applied statistics and the SAS programming language*. They do a really good job of presenting and documenting the language of the data step, and and they also cover a set of statistical procedures ranging from elementary to moderately advanced. If you had to own just one SAS book, this would be it.

If you consult *any* SAS book or manual (Cody and Smith's book included), you'll need to translate and filter out some details. Here is the main case. Many of the examples you see in Cody and Smith's book and elsewhere will not have separate files for the raw data and the program. They include the raw data in the program file in the data step, after a `datalines` or `cards` statement. Here is an example from page 3 of [2].

```
data test;
    input subject 1-2 gender $ 4 exam1 6-8 exam2 10-12 hwgrade $ 14;
    datalines;
10 M  80  84 A
 7 M  85  89 A
```

```
 4 F  90  86 B
20 M  82  85 B
25 F  94  94 A
14 F  88  84 C
;
proc means data=test;
run;
```

Having the raw data and the SAS code together in one display is so attractive for small datasets that most textbook writers cannot resist it. But think how unpleasant it would be if you had 10,000 lines of data. The way we would do this example is to have the data file (named, say, `example1.dat`) in a separate file. The data file would look like this.

```
10 M  80  84 A
 7 M  85  89 A
 4 F  90  86 B
20 M  82  85 B
25 F  94  94 A
14 F  88  84 C
```

and the program file would look like this.

```
data test;
    infile 'example1.dat'; /* Read data from example1.dat */
    input subject 1-2 gender $ 4 Exam1 6-8 exam2 10-12 hwgrade $ 14;
proc means data=test;
```

Using this as an example, you should be able to translate any textbook example into the program-file data-file format used in this course.

Bibliography

[1] Bickel, P. J., Hammel, E. A., and O'Connell, J. W. (1975). Sex bias in graduate admissions: Data from Berkeley. *Science*, **187**, 398-403.

[2] Cody, R. P. and Smith, J. K. (1991). *Applied statistics and the SAS programming language.* (4th Edition) Upper Saddle River, New Jersey: Prentice-Hall.

[3] Moore, D. S. and McCabe, G. P. (1993). *Introduction to the practice of statistics.* New York: W. H. Freeman.

[4] Neter, J., Kutner, M. H., Nachhtscheim, C. J. and Wasserman, W. (1996) *Applied linear statistical models.* (4th Edition) Toronto: Irwin.