Guide to Computing -- Quantitative Biology

The cheap computing power of the personal computer has completely changed the way that statistics are now done in biology. As the cost of calculation has dropped, more and better techniques have become widely available. An example is multivariate analysis, which became far easier with a personal computer. Other examples of good practice made possible by cheap computing power include diagnostics of residuals, randomization tests, and iterative estimation of parameters. Commonly used procedures in statistics are packaged together as a set of routines. Most statistical packages (for example, Minitab, SPlus, Systat) have easy to use graphical (menu-driven) interfaces. To complete the lab assignments and problem sets in this course you will need a statistical package. You can use a package that you already know, provided it meets the requirement in the table below.

Students at Memorial University have access to a recent version of Minitab with a graphics interface and pull-down menus. The SAS statistical package (line code only) is available to everyone on the University's PLATO operating system (Unix). SPSS is available in several computer labs at the University, but its general linear model procedure has limitations. The following table compares several packages with respect to procedures that are either convenient or necessary for this course.

	Spreadsheet	Statistical packages					
	Excel	Minitab	SAS	SPSS	Splus	Systat	Other
Spreadsheet visible	✓	✓	No	✓	No	✓	
Pull down menus	✓	✓	No	✓	No	✓	
*Basic statistical functions	✓	✓	✓	✓	✓	✓	
*Return a column of data in random order.	No	✓	✓	✓	✓	✓	
*General Linear Model	No	✓	✓	L	✓	✓	
*Residual analysis	✓	✓	✓	L	✓	✓	
*Logistic regression	No	✓	✓	✓	✓	✓	
Generalized Linear Model	No	No	✓	No	✓	✓	

L indicates limited capacity or problems with output.
* required for this course

How to use this guide.

This guide steps you through the basics you will need to use computers in this course. The guide contains tables and reference material that you can use as a reference later on, when you are completing the computer laboratories and problem sets. In the computer labs you will be given generic computational commands that will be explained and linked to specific examples. These generic commands will appear in boldface italics.

Define Data
from file

To see some examples, look for the boxes in Laboratory 3. We will work through an example later in this guide.

These are called generic commands because they describe a sequence of tasks, regardless of the details of the package you are using. Upon first occurrence these generic commands will have explanations and a sequence of actions (called pseudocode) to be executed regardless of the package. The pseudocode will be followed by examples of execution with a menu driven package (Minitab), with line code (Minitab), and in some cases in a spreadsheet (Excel). After several appearances the actions specific to a package will disappear. When the generic command or pseudocode appears again, you will have to decide how to carry it out in your statistical package.

Review:

How have computers changed the practice of statistics in Biology ?

What is a generic command ?

What is pseudocode ?

The Basics

Here is a checklist of basic tasks you need to know how to do, before starting the computer labs, or doing problem sets on the computer. As you work through this guide, check off each skill as you learn it.

If you are already familiar with some or all of these basics, you should still go through the list, check off the items you know how to accomplish, perform the rest, and check them off as you complete them.

_____ start the computer and log in if necessary

_____ list the names of files on your computer, or in your account if you have one

_____ create a data file on your computer, or in your account if you have one

_____ copy a data file from a server to your computer

_____ start up the statistical package

_____ read data into a package

_____ use the package to compute a mean

_____ send output from package to a file

_____ extract and print key pieces of this output file

_____ log off of your account

If you are using a package on your own personal computer or in a lab at the University, work through Section I. If you are using SAS on the University computer (Plato) work through section II.

Section I. Statistical packages on PCs.
- Part A. Meet the system
- Part B. Browsers, Email, Text editors.
- Part C. Meet the statistical package
  - Input
  - Descriptive statistics
  - Output
  - Saving your work

Section II. SAS on time-shared Unix system
- Part A. Meet the system
- Part B. Text editor (PICO) and email (Pine)
- Part C. Meet SAS

Section I. Statistical packages on PCs.
Part A. Meet the system.

The operating system takes care of house keeping on any computer you use. Examples of operating systems include:

Unix, Windows, Apple OS, Linux, MS-DOS

Operating systems perform tasks such as manipulating files, running programs, viewing file contents, and viewing lists of files (directories). Some system level commands are used frequently, while others are rarely used. In graphically oriented systems (Apple, Windows) the commonly used commands occur near the top of pull down menus.

The computers available in the labs at Memorial require you to have an account with userid and password. You can use this to log on to any computer in both teaching and open access labs across campus.

At this point, make sure you can
start the computer and log in if necessary;
list the names of files on your computer, or in your account if you have one;
create a data file on your computer, or in your account if you have one.

Section I. Statistical packages on PCs.
Part B. Browsers, Email, and Text editors.

Browsers are one of the most widely used programs on PCs. In this course we will use the browser to move data files from a central location (a server in Biology) out to the computer you are using. Here is the address.

www.mun.ca/biology/schneider/b4605/data

Using your web browser or ftp utility go to this server and find the following data file, and move it to your desktop.

Srbx9_5.dat

Email is another widely used program on most computers. There are many email programs.

Pegasus
Eudora
Access (Microsoft)
Pine (on Unix machines)
Communicator (in the Netscape Browser)

You can move files from one account to another with the email utility. In this course you can use email to ask questions about material in the lecture, about completing the labs, or about completing the problem sets. For this to happen, a class mailing list is needed. The most accurate way to assemble the list is for you to send a message to the prof:

A84DCS@Mun.Ca

so that you are on the mailing list for the course.

Section I. Statistical packages on PCs.
Part B. Browsers, Email, and Text editors.

Text editors complete the trio of most widely used programs on computers. In this course you will need a text editor to open data files and move data into the statistical package. You will also find a text editor useful in preparing reports with graphical and tabular display of statistical results. Text editors can be used to create data files, or edit them. You won't need to use a text editor to create or edit data files if your statistical package displays a spreadsheet. If you are using a statistical package that does not display a spreadsheet (such as SAS on Plato) you will find the text editor useful for creating and editing data sets.

All personal computers come with text editors, sometimes several. These vary in their complexity and sophistication. It doesn't matter which editor you use, but you do need to know how to save data as ASCII (unformatted) text. Some text editors automatically save files in ASCII format. Here are some examples:

pico, vi and edt (UNIX)
Notepad Wordpad(Windows)

Many word processors (WordPerfect, MS Word, etc.) save files in their own non-ASCII format. To save a file as ASCII, use the 'Save As' option under the file menu. One of the options under 'Save As' will be a standard (ASCII) text file.

The advantage of an ASCII file is that it can be read by any package on any system. ASCII was around before any of the currently used text editors. It will be around long after currently used text editors are gone. Many a researcher has lost data because they saved it only in a format used by a single package. If they had saved it as a text (ASCII) file they would still have their data.

ASCII (American Standard Code for Information Interchange) a nearly universal format for representing characters on a computer. It consists of 256 characters (essentially all those appearing on your keyboard, plus some extras that you won't need to use).

Now open the data file Srbx9_5.dat using a text editor. You can click directly on the file, which should open with a default text editor. Or you can open your preferred word processor, then open the data file from inside the program. Save the file in ASCII (text) format using the appropriate choice under the 'Save As' option in the word processor.

Section I. PCs.
Part C. Meet the statistical package. Input .

The previous sections covered the basic computer skills you will need for this course.
These were:

A. At the system level:
    find and move files,
    find commands and obtain help in using them,
    print a file.

B. Use the browser to move data files
    use the email utility to extract, save, and send files.
    use a text editor to open or create and save a data file.

At this point you should go back to the list of items to do (Basics) and make sure you are comfortable with the first 4 items in the list. With these skills, it is time to become acquainted with the statistical package.

Statistical packages with a graphics interface display at least two and sometimes three windows. One is for data and the others are for commands and output. The window for data looks and operates like a spreadsheet. If you already know how to use a spreadsheet (Excel, QuattroPro, Lotus, *etc.*), then you can use what you know to work with data in statistical packages with a spreadsheet interface (Minitab, SPSS, Systat, *etc.*).

There are several ways to bring data into a spreadsheet or spreadsheet interface. One is to type values in one at at time into the cells of the spreadsheet. For less than 10 values, it is often quicker to type the data than to bring it in from a file. For more than 10 values, it is better to move data with the cut and paste functions on the mouse. We'll call the process of bringing data into the statistical package 'data definition.' Here is a generic recipe (pseudocode) with generic command.

Generic recipe for bringing data into a spreadsheet or spreadsheet interface.

Define Data from file

```
Pseudocode (applies to any spreadsheet)
        Open file that has data, usually on the desktop.
        Use mouse to highlight the data
        Copy the highlighted data
        Paste onto the spreadsheet
        Name the columns
```

Once you have looked at the pseudocode, carry out the procedure in any package you like.
Use the data from Srbx9_5.dat, which shows age of *Daphnia* for genetic strains I and II.

Section I. PCs.
Part C. Meet the statistical package. Input .

Excel Spreadsheet.
- Paste to cell (data will appear below and to left of cell)
- If several columns of data appear in one spreadsheet column, then go back to the original data file and insert tabs after every number in every row. Copy and paste again.
- In the spreadsheet, insert a row above the data.
- In the first row of column 1 type *age(I)*
- In the first row of column 2 type *age(II)*

Minitab worksheet.
- Paste to cell (data will appear below and to left of cell).
- Click on top of column 1, type in variable name *age(I)*
- Click on top of column2 , type in variable name *age(II)*

To see or use the Minitab command lines, you may need to switch on the command line display. Once you have, you can use both command lines and the graphics interface. In fact all the graphics interface does is write the command lines which are then executed. All statistical packages with a graphics interface work this way.

Minitab Menu
- Editor
  - Enable commands

```
MTB > read 'srbx9_5.dat' c1 c2;
SUBC> nobs=7.
MTB > name c1 'age(I)'  c2 'age(II)'
```

This is called ***Data definition*** because we are assigning a name as well as values to the variable.

Section I. PCs.
Part C. Meet the statistical package. Descriptive statistics.

Next, we perform simple operations on the variables within the spreadsheet. These operations are performed by functions in the statistical package. One simple function is taking the mean of variable X, for which the functional expression is *mean(X)*. The mean value will be placed at a location in the spreadsheet, which we will call *f(x)*.

Here is the generic recipe (Pseudocode) with generic command.

Calculate statistic mean(X)

Generic recipe for calculating means of variables (columns) in a spreadsheet

```
Pseudocode (applies to any spreadsheet)
    Select a location to place $f(X) = mean(X)$.
    Find the $mean(X)$ function
    Apply the function to the variable (entire column).
    Make the calculation.
```

Now that you have looked at the pseudocode, compute the mean age for each strain of *Daphnia*, using the data from Srbx9_5.dat that you pasted into your spreadsheet.

```
Excel spreadsheet
    Click cell at bottom of column 1
        This will be the location of $f(X) = mean(X)$.
    Function
        Statistical
            Average
                A2:A8   (7 values in column A)
```

```
Minitab spreadsheet
    Calculate
        Column statistics
            Mean
            Input variable Age(I)
Highlight the result, copy and then paste onto the spreadsheet.
```

```
MTB > let k1 = mean(c1)
MTB > let k2 = mean(c2)
MTB > print k1 k2
MTB > desc c1-c2
```

Section I. PCs.
Part C. Meet the statistical package. Descriptive statistics.

Now that we have results, we need to communicate it to others in a format that is readily understood. For statistical results, this means explanations tied to the numbers and graphs. One easy way to accomplish this is to use cut and paste functions to move selected portions of the statistical results to a document in a word processor. Here is a generic command with recipe (pseudocode).

Output to document

Generic recipe for output of statistical results to a document for printing.

```
Pseudocode (applicable to almost any package).
        Open document in a new window.
        Return to spreadsheet or statistical package output
        Highlight output to be reported, click copy
        Return to document and paste at appropriate place.
        Repeat process for each selected portion of output.
        Add explanation.
        Print the document.
```

Now that you have looked at the pseudocode, try moving the *Daphnia* data and the means to a document in your preferred word processor.

As you move results from the package to your document consider how the document will look. Be sure to move only the results you need. Leave out the extraneous material. Be sure to check the format of the results after you paste them into the document. If the results were in columns, and they are no longer aligned, highlight the portion you transferred, then apply a format with uniform spacing (such as courier). This should bring mis-aligned columns back into alignment.

Here is an example. First with a scalable font that destroys alignment.

```
SOURCE      DF        SS        MS       F      p
Regression    1    0.096644    0.096644    23.64    0.005
Error       5    0.020442    0.004088
Total      6    0.117086
```

Now the same text displayed with a non-scalable font (courier) to restore the intended alignment.

```
SOURCE         DF          SS          MS         F        p
Regression      1    0.096644    0.096644     23.64    0.005
Error           5    0.020442    0.004088
Total           6    0.117086
```

Section I. PCs.

Part C. Meet the statistical package. Saving your work.

Once you have moved the results from the package to a document, you can either save the worksheet you have created or you can discard it. If you save the worksheet, be sure to use a name that is informative with the correct extension (.mtw .xls *etc.*). Examples are

Srbx9_5.mtw	Minitab worksheet
Srbx9_5.xls	Excel spreadsheet

In general it is a good idea to use the 'save as' command within a statistical package or spreadsheet to save the worksheet. That way, you choose the name and format for saving your work. At this point, click the 'save as' option under 'file' in the package you are using. Then look through the list of formats into which you can save your worksheet. You will see that each format has its own distinctive extension.

Table G2. Recommended extension for file names.

Command	extension	Type of file	Complementary commands
MTB > write	.dat	data file	MTB > read, > set, > insert
MTB > save	.mpj	worksheet	MTB > retrieve
MTB > store	.ctl	commands	MTB > execute
MTB > outfile	.out	listing	plato> lpr

Using extensions in a consistent way will make statistical analysis on the computer easier for you.

write the data as	'clam.dat'
save the worksheet as	'clam.mpj'
store repeated commands as	'clam.ctl'
outfile the results as	'clam.out' or 'clam_out.txt'

If you read data into a spreadsheet, find an error, and correct it in the spreadsheet, then you should correct the error in the source data file. Often this can be accomplished by highlighting the data columns, then saving these in ASCII (text) format back to the original file.

Section II. SAS on the Unix system.
z
Part A. Meet the system.

If you are using SAS you will need to know how to carry out basic tasks on the UNIX operating system. To use SAS you will need to use only a few system level commands, to accomplish the following tasks.

Obtaining a listing of files in a directory
Obtaining help
Deleting a file
Moving a file from one directory to another
Creating a new directory
Sending a file to a printer

If you are a new user of the UNIX system, start by logging onto your UNIX account, entering your username and your password. If you have logged on successfully, you will see some announcements and eventually you will have a symbol at the bottom left of the screen resembling 'plato>'. This is known as the 'prompt'; it is where system commands are entered to perform tasks.

Section I. SAS on the UNIX system.

Part A. Meet the system.

If you are used to operating systems with a graphical interface that uses a mouse, then the line oriented (typed) commands in UNIX will take some getting used to. Commands are not listed in a menu, so they have to be actively recalled, rather than recognized from a list. Commands are abbreviated (to save typing longer words frequently), which makes the commands harder to remember.

Table G2. System Commands

Command	Mnemonic	Equivalent Commands*	Explanation
`cd`	**c**hange **d**irectory	______	change from present directory up to a named subdirectory or down to the next directory (`cd ..`)
`cp`	**cop**y file	______	copy a file from one location to another
`dir`	**dir**ectory list (vertical)	______	list contents of a directory vertically
`exit`	**exit**	______	exit from your account
`ls`	directory **l**i**s**t (horizontal)	______	list contents of a directory horizontally (‘`ls -al`’ will list vertically with all file characteristics)
`man`	help **man**ual	______	obtain help about system commands and topics (i.e. ‘`man mkdir`’)
`mkdir`	**m**a**k**e **dir**ectory	______	create a new directory
`more`	see **more** of a file	______	read an ASCII text file (i.e. data file)
`mv`	**m**o**v**e file	______	move a file from one location to another
`rm`	**rem**ove file	______	delete a file (warning... this is permanent!)
`lpr`	to **l**ine **pr**inter	______	send file to printer

* If your operating system is not UNIX.

Now try using your system command to list the files in your account.

[Don't forget to check off this item on your list]

Then use a system command to find out how to create a new subdirectory.

Section II. SAS on the Unix system.
Part B. Meet the text editor (PICO).

If you are using SAS on the Unix computer, you will need to move data files from their storage place (the server in Biology) to your account on Plato, the Unix computer at MUN. You can do this with the ftp command from your account on Plato. Alternatively, you can move the file to your desktop, then email it to your account on Plato, then extract it to a file on Plato.

Once you have moved a file to your Plato account, you will need to use a text editor on Plato. The following narrative explains how to use the PICO text editor on UNIX. It explains how to start up the text editor, create a new file, and edit a pre-existing file.

At the system prompt type the following:

```
plato> pico                                              z
```

You are now in the PICO text editor, editing a new file that has yet to be named. The top of the screen will read 'new buffer', meaning this file is new and unnamed. You are now free to type any text you wish in the editing window. When you are finished typing your text, you can leave the editing program and save the file by holding down the <CTRL> key and pressing X. This is written in shorthand from now on as ^X, and appears as such in the commands at the bottom of the text editor. NOTE: Because PICO is user-friendly, many useful commands are displayed at the bottom of the screen, and you are encouraged to refer to those before asking questions, so you can try and answer your questions for yourselves. The text editor will ask you if you wish to save the changes made. Press Y if you do, and then type the name which you wish to call this new file. For consistency, we will call this file 'sample.txt'. Then press <ENTER> . This same procedure can be used to create a data file while at the system prompt (however you'd preferably use sample.dat as the extension, to let yourself know that it is a data file).

NOTE: *UNIX is a case-sensitive operating system, which means that a file named 'Sample.txt' is considered a completely different file than one named 'sample.txt'. The importance of this point will become more apparent when you are introduced to Minitab, the statistical package.*

We will now use that same file to edit a file that already exists. Type the same command at the prompt as before, however this time put a space after 'PICO' and type the name of the file we just created.

```
plato> pico sample.txt                                     z
```

You will notice this time that instead of ‘new buffer’ appearing at the top of the screen, the file name will appear. You can edit the file contents however you like, then save it in the same manner as before, except this time you won’t have to type the file name... it will automatically appear as, in this case, ‘sample.txt’. This same procedure can be used to edit any existing data file while at the system prompt. [score another for your list]

Section II. SAS on the Unix system.
Part B. Meet the email utility (Pine).

Another program that has commands independent of the operating system is the email utility. This section shows to use Pine to compose an e-mail message, mail a file, and extract a file. If you already know how to use these commands in E-mail, then skip ahead to the next part, Meet the statistical package.

Here is how to start up the Pine e-mail utility on UNIX, then use it to send a message, to extract a data file sent to you, and to send a data file to a friend.

To start the program from the UNIX system prompt:

```
plato> pine                                                  z
```

You will now see the main menu in front of you.

Let's start by composing a message.

*The first step is to press ‘C’ to compose (as on the screen). Now you are presented with several fields, the first being ‘To’, where you type the e-mail address of the person receiving the mail (**a84dcs@plato.ucs.mun.ca** is your prof’s e-mail address). You can skip the next two fields using the down arrow, and the ‘Re’ field is optional... it only provides a small phrase describing the content of the mail message. Using the arrow key again, you can move the cursor to the message body. This is where you type the message, or the message you are going to send. In this case, you need only type in your name.*

To send this message, use the ^X command, by holding down the control key and pressing X. The ^X send command is listed on the bottom of the screen. Then confirm that you want to send the message by pressing ‘Y’. Now to exit, press ‘Q’.

Congratulations, you've successfully sent an e-mail message with Pine!

[Another check on your list]

Section II. SAS on the Unix system

Part B. Meet the email utility (Pine).

A shortcut for sending an e-mail message is to type the Email addresss of the recipient of the message after the command at the prompt:

```
plato> pine a84dcs@plato.ucs.mun.ca                       z
```

If you use email rather than ftp to move a data file from the server to your Plato account you will need to extract the file and save it using the e-mail utility. To extract a file, start up the E-mail program, as you did in the previous example.

Instead of hitting 'C' for compose, hit 'I' for 'INBOX', the folder that contains new mail messages. You will see a list of e-mail messages that you have received. Using the up and/or down arrow keys on the keyboard, highlight this message. If there is more than one message, the rightmost field in this list (brief message contents) should contain an identifiable phrase concerning this data file (srbx9_5.dat). When you have highlighted the proper message, press <ENTER>.

You will see that the message is composed of two parts, or 'attachments'; the second one is the attached data file you wish to extract into your own directory on the system. To do so, press 'V' to view attachments, and then, as in the INBOX, use the up/down arrow keys to highlight the attachment containing the data file. Then press <ENTER> to see the file and ensure it is the proper one. Now press 'S' to save this attachment. The file name should automatically appear at the bottom of the screen when it asks you what you'd like to call the new file, so just press <ENTER>. Many e-mail systems make it easy to use the same file name that the sender used. In this course, this feature ensures that everyone has the same name on identical data files, saving everyone a lot of time. Now exit as you did earlier, by pressing 'Q' and acknowledging the confirmation.

To ensure that the file was correctly extracted, use your system command to view the file.

```
plato> more srbx9_5.dat                                   z
```

[Score another check mark on your list]

Section II. SAS on the Unix system
Part B. Meet the email utility (Pine).

If extracted correctly, the file should appear on your screen looking like this:

```
                                                                      z
 7.2  8.8
 7.1  7.5
 9.1  7.7
 7.2  7.6
 7.3  7.4
 7.2  6.7
 7.5  7.2

Data from Box 9.5 Sokal and Rohlf 1995.
Age (days) at first reproduction in two
 genetic groups of Daphnia longispina.
MTB > read 'srbx9_5.dat' c1 c2;
SUBC> nobs=7.
MTB > stack c1 c2     c3
MTB > set  c4
DATA> (0 1)7
DATA> end
MTB > name c3 'age'  c4 'group'
```

To e-mail a data file in your directory to a friend, start by entering the e-mail utility.

Now begin composing a Pine E-mail message. However in the 'Attachment' field toward the top of the screen, type the name of the data file (located in the directory from which you entered the mail program). If the file is in another directory in your account, press ^J (control key + J), then ^T to obtain a listing of file and directory names. Then using the arrow keys, manually choose the file name. Now send the message as you would a regular e-mail message.

Section II. SAS on the Unix System.
Part C. Meet SAS.

The statistical package SAS has a number of advantages. It can handle larger data sets, it can read complex data sets consisting of several types of records, and it contains a far larger repertoire of statistical procedures. SAS allows one to reorganize or generate new data sets. It can carry out the generalized linear model, one of the most important advances in statistics in the last 3 decades. This greater capacity and flexibility comes at a cost. SAS is harder to use, as one might expect of any complex instrument.

SAS on the Unix system (Plato) does not have a graphics interface. A SAS session typically works with several data sets each with its own columns (variables) and rows (cases). Input to the SAS package consists of data definition statements and procedures; these are organized into a series of commands that are typically submitted as a batch, rather than being typed in one line at a time from the terminal. The output is automatically directed to two different files. One is a list file with results. The other is a log file that reports on the comings and goings of data files, the success or failure of each command, and any errors made in executing the batch file.

Routines are built out of commands, these routines are modified to suite the situation and goals of computation. Assignments in this course can be accomplished by modifying one of the following routines, depending on the situation or goals of computation.

Sections of the routines have been assigned names, which are used in other parts of the lab manual. These generic commands occur in ***boldface italics*** to the right of the routine box.

To see how SAS works, we will read a data file into a structured SAS data set, print the structured data set, and compute descriptive statistics for variables in that data set. This is the same sequence used in the first section of this guide, for PCs. We will use the same data file, srbx9_5.dat, as in the PC section.

Section II. SAS on the Unix system
Part C. Meet SAS.

Here is a SAS command file that reads data into a a structured data set, lists data to an output file, and computes a variance. This set of commands is collected together into a file called srbx9_5.sas, which you will create with a text editor.

```
Title 'Daphnia longispina data.';
Options linesize=72;
Data A;  infile 'srbx9_5.dat' obs = 7;
  Input SeriesI  SeriesII;

Proc print;

Proc means;
   Var SeriesI SeriesII;
```

Define Data from file

Output to log file

Compute a variance

When you are done creating the file, check it carefully against the example in the box. Make sure that all of the semicolons are entered as shown in the example. Failure to place semicolons correctly is one the most frequent reasons for the failure of a SAS program to run.

The SAS commands are executed by turning the file over to the SAS package for processing.

```
plato> sas srbx9_5.sas
```

:

Section II. SAS on the Unix system
Part C. Meet SAS..

Before executing this set of commands, let's look at what they do. First, the commands that define the session and the SAS dataset.

Define data from file

```
Title 'Daphnia longispina data.';
```
This command puts labelling information into the output or list file.
```
Options linesize=72;
```
This command controls the width of the list file to 72 characters. The default is 132 characters, too wide to see easily on a standard computer screen.
```
Data A;  infile 'srbx9_5.dat' obs = 7;
  Input SeriesI  SeriesII;
```
This data definition command defines a data set called A. The Infile subcommand tells the SAS package that the data is to be read from an external file called srbx9_5.dat; only 7 lines are to be read from this file. The Input statement defines two variables, SeriesI and SeriesII.

Next, we'll look at the commands that carry out computations.

Output to log file

```
Proc print;
```
This command prints out the last data set, A, and sends it to the list or output file.
```
Proc means;
  Var SeriesI  SeriesII;
```
This procedural command computes descriptive statistics on two variables, SeriesI and SeriesII, from data set A. These results are sent automatically to the list or output file.

Compute variances

When the command file srbx9_5.sas is turned over to SAS, a log file called srbx9_5.log is automatically created. This has a record of the session, including a listing of any errors encountered in the command file. To view the log file, use a system command or a text editor.

Procedural commands send output to a list file, srbx9_5.lis, unless errors are encountered. If no list file is created, or if something is missing from the file, the log file must be checked to find the errors; the errors are then corrected in the control file (the file with the .sas extension). The corrected control file is turned over to SAS.

To see if a list file was created:

```
plato> ls *.lis
```

Section II. SAS on the Unix system.
Part C. Meet SAS..

To look at the results, view the file at the system prompt, i.e., with a system command. Or you can view it with the text editor, or send it to a printer. When you look at this list file, you will see that it contains a printout of the structured (SAS) dataset A, including variable names and a unique observation number for each case (row) in this data set. It also contains the formatted and labelled results from the Proc Means command. The SAS list file does not record the history of the session, only the results of computations.

If you are using SAS on a remote system rather than your own computer, you can move the list file from the system to computer using email or a file transfer program such as ftp. Once the SAS list file is on your computer, you can send it to your local printer or pull it into your local editor, in order to mark and extract pieces of it to display in reporting the results of your analysis with SAS. If you are going to print sections of a SAS list file, be sure to format these sections in a non-scalable font such as Courier 10. If you use a scalable font the results in the list file will be distorted and nearly unreadable. In this manual, a scalable font (Times Roman) is used outside the boxes, while a non-scalable font (Courier) is used inside boxes, to retain spacing and legibility.

* * *

The data definition commands in SAS permit extraordinary flexibility in reading and generating new data sets. To demonstrate this flexibility, let's re-organize the *Daphnia longispina* data into a new data set consisting of two variables: age at beginning of reproduction (days) and series (either I or II). This is the data structure that would be used to analyse this data with a General Linear Model, demonstrated later in the course.

Here are the SAS commands to read and re-organize the data. They are stored in a command file (either a revised version of srbx9_5.sas, or a new command file). The file is created with a text editor, either your own, or the one that some versions of SAS supply.

```
Title 'Daphnia longispina data.';
Options linesize=72;
Data A;  infile 'srbx9_5.dat' obs = 7;
  Input SeriesI SeriesII;

Data B; Set A;
  Age = SeriesI; Series = 'I'; output;
  Age = SeriesII; Series = 'II'; output;

Proc print;

Proc means; By Series;
   Var Age;
```

Define data from a file

Reorganize

Output to a log file

Compute a variance

Section II. SAS on the Unix system.
Part C. Meet SAS.

As before, let's look at the commands before executing them. First, the data statement used to reorganize the data.

```
Data B; Set A;
  Age = SeriesI; Series = 'I'; output;
  Age = SeriesII; Series = 'II'; output;
```

Reorganize

This creates a new data set labelled B. The Set command reads the variables from data set A into B, one line at a time. All subsequent commands occur within a loop that is executed for each line of the input data set. When the first line is read, the first value of SeriesI is assigned to the variable Age in the first line of data set B. The variable Series is assigned a value of I in this first line. The output statement forces output to the reorganized dataset B. The next line assigns the first value of SeriesII to the variable Age in the second line of data set B. The variable Series is assigned a value of II in this second line. SAS has now reached the end of the Data statement, so it goes back and executes the statement again. It reads the second line of dataset A. The first value is assigned to SeriesI then output to the third line of data set B. SAS continues in this fashion, looping repeatedly through the Data statement, until all of the data is read from data set A, or until SAS is otherwise told to stop.

These data definition statements create a new data set that stacks SeriesI and SeriesII into one column, then creates a new variable the identifies the source of the *Daphnia* age measurements.

Next, the Procedural statements that print the re-organized data set and compute summary statistics on it.

```
Proc print;
      This prints the contents of the last data set, B.
Proc means; By Series;
   Var Age;
      This computes summary statistics on the variable Age for each of
      the two source groups.
```

Output to log file

Compute a variance

The command file is created with the editor. This is a new version, so it will be given a slightly different name (srbx9_5b.sas), by adding the letter b to the file name. This command file is turned in for execution as before.

```
plato> sas srbx9_5b.sas
```

SAS automatically creates a log file called srbx9_5b.log . SAS creates a list file called srbx9_5b.lis, if there are no fatal errors in the command file.

Section II. SAS on the Unix system.
Part C. Meet SAS.

.

This reorganized data file in SAS can be saved as a system file, rather than being generated again and again. Here is a command file that places the reorganized data into a file called srbx9_5b.dat, created by SAS. Note the consistent use of the extensions .dat and .sas to distinguish data file from command files concerning the same data set.

```
Title 'Daphnia longispina data.';
Options linesize=72;
Data A;  infile 'srbx9_5.dat' obs = 7;
  Input SeriesI  SeriesII;

Data B; Set A;
  Age = SeriesI; Series = 'I'; output;
  Age = SeriesII; Series = 'II'; output;

Data C(Keep = Age Series);
  Set B;
  Filename srOUT 'srbx9_5b.dat';
  File srOUT;
  Put @6 Series  @2 Age;

Proc Print;
```

Define data from a file

Reorganize

Define data to a file

The only new code here is the third data statement, which creates dataset C. This new set contains only two variables (Age and Series) in contrast to dataset B, which contains four variables (SeriesI, SeriesII, Age, and Series). Both Age and Series have been written to a file. The Put statement is much like an input statement in form, except that it writes rather than reads data. This Put statement writes the variable Age starting at card column 6, then writes the variable Series starting at card column 2. This shows the flexibility of SAS in reading, reorganizing and writing data.

A longer list of SAS commands, which carry out computations needed for labs and problem sets, can be found elsewhere in the lab manual.