Guide to Computing in Biostatistics

Corey McEntyre
Department of Biostatistics
University of North Carolina at Chapel Hill
Version 1.0

September 6, 1994

Printed at the University of North Carolina at Chapel Hill, for the Department of Biostatistics.

Contents

List of Figures

List of Tables

Preface

The Department of Biostatistics has developed into a computing environment in which the documentation for resources has become dispersed to the point of near non-existence. The point of *this* document is to compile all of the necessary information into one convenient source. It is intended as a simple guide; not as a technical document. Bruce Walter and Corey McEntyre have a large collection of books that cover a wide variety of subjects should you be interested in a more detailed discussion of a particular topic. We are willing to lend these books out on a first-come-first-serve basis, but this service depends upon the courtesy of those people who borrow the books.

Ralph DeMasi assisted in the writing of this document and gave technical advice on using LaTeX to make it both attractive and useful. He also contributed the sections on LaTeX and Emacs. Bruce Walter gave technical assistance on the structure of our computing environment and on various details of our software packages. My whole-hearted thanks to both of them.

The chapter on X Windows was adapted from chapter three of the "X Windows Users Guide".

The sections on GLIM, S-plus^{tm}, TeX, C, Pascal, FORTRAN 77, parts of the Glossary and the chapter on Information Resources were adapted from an original manual by Dr. Michael Conlon, University of Florida, dated September 20, 1992.

I will try to keep this document up to date with the ever changing environment of Biostatistics. You can help me make this guide more thorough and more useful through suggestions, comments and criticism. Please feel free to express yourself by sending ideas and encouragement to biosguide@biostat.sph.unc.edu. If you would like to see the most recent version of the guide, it is kept online in /distrib/guide/biosguide.ps. Use *ghostview /distrib/guide/biosguide.ps* to view it from an X Windows machine (or a PC with an X Windows server). Please do not print it — it is well over 100 pages long.

- Corey McEntyre

Chapter 1

Connecting to...

This chapter discusses the basic means of connecting to the Biostatistics Unix computing environment. If you are familiar with computers or have connected to the BIOS Unix network before then you can probably skip this chapter. It will take you through logging on to a terminal, connecting via telnet, and dialing in from home on your modem.

We currently run X11 Windows Release 5 (X11R5) on all of our Sun terminals. We were drawn to this interface for several key reasons:

- It's fast!
- It's small!
- It's free!
- There are many publicly available software packages written for it (they're free too).
- We can run X applications on our PCs running OS/2.
- It provides a very individualized work environment with options for every application:
  - Background and foreground colors.
  - Window geometry.
  - Text for the title of windows or icons.
  - Short cut keys.
  - The font in which text is displayed (size and style).

We think that X Windows will be a real benefit to your productivity and hope that you find it enjoyable to use. If you will be using the X Windows interface extensively, it is covered in detail in Chapter 2. If you find that the information in that chapter is not thorough enough, we have the entire O'Reilly & Associates series on X Windows, including library and programming manuals for programmers, the *very* detailed User's Guide, and an overview of X Windows: The X Window System in a Nutshell.

1.1 The Terminal

The Department of Biostatistics had (at the time this document was written) seven Sun terminals. Logging onto a Sun is a relatively simple task. The Suns have screen savers installed on them, so it may appear as though the computer is not on. A quick jiggle of the mouse or a couple of keystrokes should bring the computer screen up. Since we are running X Windows here, the first screen you see should look like Figure 1.1.

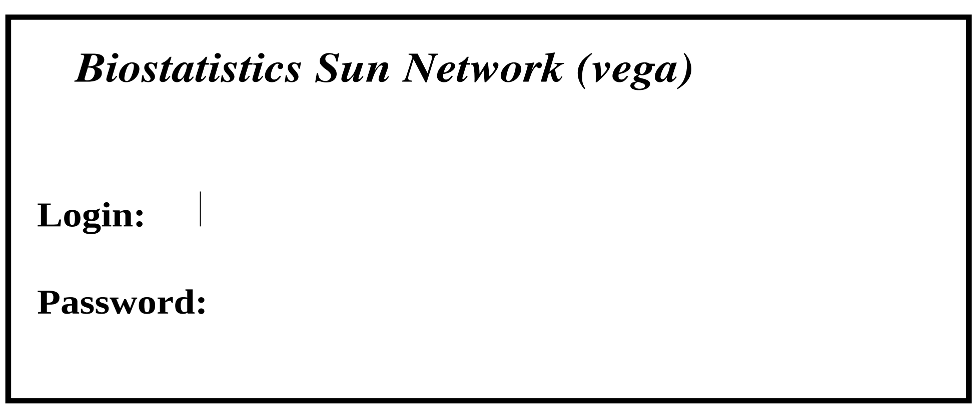

Figure 1.1: Login box for X Windows

To log on, simply type in your username and password. If you do not have a username, or if you have forgotten your password, check with Corey McEntyre or Bruce Walter in 3105J McGavran to set up an account or reset your password. Once you have correctly typed your username and password, X Windows will start with our default set-up.

This consists of two windows, a clock, and the TWM Icon Manager. The left mouse button will pull up a menu when pressed on the root window[1]. The menu is a pull-down type menu, which means that you need to hold the mouse button until you have highlighted your chosen action. Releasing the button executes the highlighted option. The logout option at the bottom of the menu is executed by holding down the left mouse button, pulling down to highlight it, moving to the right and highlighting "Yes". This was done to prevent accidental logouts. Move the mouse to the right to see the submenus[2]. If this is your first time using X Windows, we have set up a "Welcome screen" for your first login (see Figure 1.2).

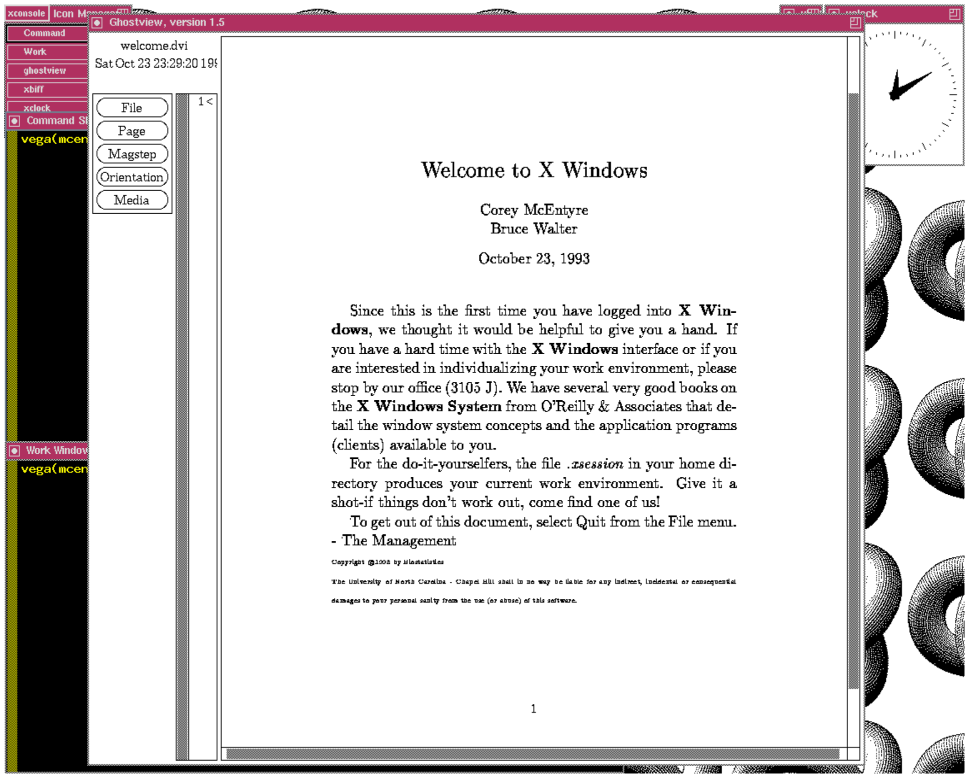

Figure 1.2: Initial screen for X Windows

[1] The root window is set as a montage of escher knots in the default configuration.
[2] They are denoted by little books next to the menu option.

This screen will not appear in subsequent logins. Play around some and try out the help utility on the main menu — if you have questions, we'll be glad to help.

1.2 Telnet

Using telnet to connect from one computer to another is as good as being there. Well, almost. You cannot[3] run graphics programs or open windows. But, you *can* access your files, use mail, or run any other text-based program. Telnet allows a user at one site to connect to another site via the internet. It passes the keystrokes from the user's terminal directly to the remote machine as if they had been typed there. The remote machine then sends the output to your terminal screen.

To use telnet you must know the address of your desired target. For local machines (machines on the same network) you can usually get away with simpler forms. For example, if you are on a Sun terminal connecting to another Sun terminal, it is as simple as typing `telnet biostat`. If you want to connect from the VAX, for another example, you would have to be more specific about the address: `telnet biostat.sph.unc.edu`. Once a connection has been established, you will see a prompt for username and password. You will not have a window interface, but a standard Unix prompt instead. Using telnet from a PC to open X windows on the PC will be covered in later chapters.

1.3 The Modem

For those of you who would like to connect from home, Computer And Information Services (CAIS) provides dial-up access to Xyplex (from which you can reach Biostatistics). The set-up for your software communications package is as follows:

Once there, you will see a menu consisting of twenty items. Item #16 is 'Connect to another address...'. After selecting it you will be prompted for an internet address. For example, if you wanted to connect to **biostat**, our file server, you would type `biostat.sph.unc.edu`. As soon as you have a connection, you will be prompted for your username and password. Again, you will not be able to run windowing

[3] Until the more advanced chapters of this manual.

Modem Set-up	
Parameter	Value
Parity	None
Data Bits	Eight
Stop Bit	One
Duplex	Full
Phone Number	962-9911

Table 1.1: Modem settings for Xyplex

applications. Instead, the standard Unix prompt will appear and you will be in your home directory. When you are finished, simply type `logout` and you will be back at the CAIS menu. Option #20 will exit you from Xyplex, and then you are free to hang up your modem or dial another number.

This may be confusing for a first-time modem user, so if you have questions don't be too shy to stop by and ask for help. Bruce Walter and Corey McEntyre both have experience with several different modem software packages (probably yours), so check with them if you run into trouble.

Chapter 2

X Windows

Chapter 1 touched on the basic reasons we feel X Windows is the best work environment for the department of Biostatistics. Also in that chapter was a quick description of the appearance of X Windows and an accompanying picture (Figure 1.2 showed you what your first session in X Windows would look like). The purpose of chapter 2 is to give a more detailed description of X Windows and how to get around in the X environment.

2.1 An Introduction to the X Window System

The X Window System, called X for short, is a network-based graphics window system that was developed at MIT in 1984. Several versions of X have been developed, the most recent of which is X Version 11 (X11), first released in 1987.[1]

X11 has been adopted as an industry standard windowing system. X is supported by a consortium of industry leaders such as DEC, Hewlett-Packard, Sun, IBM, and AT&T that have united to direct, contribute to, and fund its continuing development. In addition to the system software development directed by the X Consortium, many independent developers are producing application software specifically for use with X, including spread-sheets, database programs, and publishing applications.

[1] There have been several revisions to X since then. This guide describes the latest release, X11R5.

Before you plunge into Section 2.2, we suggest you read Section 1.1 in Chapter 1 so you know how to log on and what to expect.

2.2 Working in the X Environment

This chapter is designed to give you the basic skills needed to be productive in the X Windows environment. It will show you how to:

- Open an xterm window.
- Use the mouse to affect windows on the display.
- Iconify, deiconify, raise, lower, move, and resize windows using the mouse.
- Close an xterm window from the command line.
- Start additional client programs.

2.3 Creating Windows

Once you focus input on an xterm window, you can enter commands to open other client windows. For example, you can open an xterm window by typing this command at the prompt in an xterm window:

```
vega(mcentyre) 1% xterm &
```

After a few moments, the xterm will appear "pseudo-randomly". I say "pseudo" because after opening a few windows, you will easily be able to spot the pattern. The keyboard focus will not necessarily be in this window; it depends on the location of your pointer. The active window is the one with the pointer located inside of it — in this example, if your pointer is not in the new window just created, you will not be able to type in that window. To exit this window type `exit` at the Unix command prompt or select `close` from the root menu.

Notice how the titlebar, text cursor, and border in each xterm window are highlighted when you move the pointer to that window. Whatever you type will appear in the window with the highlighted titlebar.

2.3.1 Using the Mouse

The curser on the screen follows the mouse's movement on the desktop. Move your mouse around on the desk to get used to this. If you move to another display, the screen curser may move quicker or slower than the one you're used to.

You use the mouse to indicate a graphical element on the screen, such as a window, icon, or command button. Most mice have three buttons. For our purposes, we'll refer to these buttons as left, middle, and right. By placing the mouse on a particular element and performing some action (and possibly a mouse movement), you can invoke a variety of commands. The types of button actions and mouse movements you can perform are:

Click — To click a button, press the mouse button down and release it. A click is a rapid action; there is no pause between the press and release. A double click is two full clicks in succession, with no pause between clicks. A triple click is three clicks in succession.

Press — To press a button, push the button down and hold it down.

Release — After pressing a button down, release it by letting up on the button.

Drag — To drag a graphic object (such as a window or icon) from one location on the screen to another; place the mouse on the object; press one or more mouse buttons; move the mouse to another location (dragging the object); and release the button(s).

Keep in mind that some commands or actions are invoked by a simple click on a particular graphic element. Alternatively, some actions require a button press and a mouse movement (i.e., dragging).

When dragging is used to move an object, the actual object does not appear in the new location until you complete the movement and release the mouse button. Instead, you appear to drag an outline representing the object. When you release the mouse button, the actual object appears in the new location.

Dragging is also commonly used to change the size of a window. Again, an outline indicates that the window's size is changing. When the outline approximates the size you want, you release the mouse button and the actual window is redrawn using the selected dimensions.

The following sections describe how to perform the most basic window management functions, which require you to use the mouse in the ways we've discussed.

2.4 Managing Windows Using the Titlebar

The titlebar itself and its component parts are tools that allow you to manage the window using the mouse.

The titlebar spans the top of each window on the display. The titlebar contains the name of the application (e.g., *xterm*), two command buttons (used to iconify and resize the window) and a region that is highlighted when the window has the input focus.

Notice that whenever you move the mouse onto either one of the command buttons, the mouse changes to a small hand cursor, indicating that you can push the button. If you move the mouse onto any other part of the titlebar, it changes to an arrow cursor.

Though it's not immediately obvious, you can perform most window management functions by using the mouse on various parts of the titlebar. The following sections explain how to iconify, move, raise, lower, and resize windows using the titlebar. Should your titlebar become obscured by other windows, these same functions are available through the menu on your root window, under the heading "Window Utilities".

2.4.1 Converting a Window to an Icon

There are many circumstances in which it may be desirable to iconify a window:

- To prevent yourself from inadvertently terminating a window.
- While running a program whose progress you don't need to monitor; if you don't need to see it, the window is just taking up space.
- If you are only using an application occasionally. For example, you might be running the *xcalc* calculator program, but only using it every so often.

- If you want to use several application windows, but only display a few at a time; this arrangement can be somewhat less confusing than a display crowded with windows. Having some windows iconified also frees you from constantly shuffling the stacking order.

The *iconify* command button on the far left side of the titlebar allows you to convert a window to an icon. The iconify button is identified by a dot in its center. To iconify a window, use the following steps:

1. Place the mouse within the iconify command button. (The mouse simply has to rest within the button's outer border, not within the dot identifying it.) The mouse changes to a small hand symbol.

2. Click any mouse button. The window is iconified.

By default, an icon is displayed at the same coordinates as the corresponding window (that is, it has the same upper-left corner). Somewhat surprisingly, moving the window does not effect icon placement once the window has been iconified once. If you move the window (described later in this chapter) and then iconify it, the icon is still displayed at the window's original coordinates. However, if you move an icon on the display, this new location becomes the default for the icon.

You can also set up and organize icons within a window known as the Icon Manager (it is located in the upper-left corner of your root window). By clicking on the Icon Manager window, you can iconify and deiconify any client on the screen.

2.4.2 Converting an Icon to a Window

To convert an icon back to a window (deiconify it), place the mouse on the icon and click the left (or middle) mouse button. The window is redisplayed in the position it appeared before it was iconified.

If you want to type in the deiconified window, you need to move the mouse into it.

2.4.3 Raising a Window or Icon

We've already seen the necessity for raising windows on the display: frequently windows overlap. In order to work with a window that is all or partially covered by another window, you'll want to raise it to the top of the window stack. You raise a window with the following steps:

1. Place the mouse on any part of the titlebar, except the iconify and resize buttons. The mouse changes to the arrow cursor.

2. Click the left mouse button. The window is raised to the top of the stack.

If you leave the mouse on the titlebar, the raised window also has input focus. Once you have raised a window to the top of the stack, you should be able to enter input and read output easily.

Icons may be obscured by windows or other icons on the display. The only way to raise an icon using the mouse is to deiconify it, as described in the preceding section. When an icon is converted back to a window, it is also raised to the top of the stack

2.4.4 Lowering a Window

To lower a window (send it to the bottom of the window stack):

1. Place the mouse on any part of the titlebar, except the iconify and resize buttons. The mouse changes to an arrow cursor.

2. Click the middle mouse button. The window is lowered to the bottom of the stack

2.4.5 Moving a Window

In many cases you'll want to move a window from one location on the display to another. The largest part of the titlebar (which contains the application name and highlight region) allows you to move the window, using the following steps:

1. Place the mouse on any part of the titlebar, except the iconify and resize buttons. The mouse changes to the arrow cursor.

2. Press and hold down the left mouse button. The mouse changes to a cross arrow mouse and a window outline appears. This outline tracks the mouse's movement.

3. Move the window by dragging the mouse.

4. Drag the cross arrow mouse with the window outline to the desired location on your screen.

5. Release the left mouse button. The window will move to the selected location.

When you use the mouse to move a window, the mouse rests in the titlebar; if you don't move the mouse subsequently, input is always focused on the window you've moved.

Note that moving a window also raises it to the top of the stack.

2.4.6 Moving an Icon

Moving an icon is similar to moving a window. To move an icon:

1. Place the mouse on the icon.

2. Press the left mouse button.

3. Move the icon by dragging the mouse. When you begin to move the icon, the mouse changes to a cross arrow mouse and an icon outline appears. This outline tracks the mouse's movement.

4. Drag the cross arrow mouse with the icon outline to the desired location on your screen.

5. Release the left mouse button. The icon will move to the selected location.

2.4.7 Resizing a Window

You can resize a window horizontally, vertically, or simultaneously in both directions. Resizing is a bit trickier than any of the other window management functions we've already discussed, since the way you move the mouse determines the size of the window. It will probably take some practice. Regardless of the way you want to resize a window, you begin in the same way.

1. Place the mouse on the resize command button, which is on the right hand side of the titlebar and is defined by a group of nested squares. The mouse changes to a small hand symbol.

2. Press and hold down any mouse button. The mouse changes to the cross arrow mouse and a window outline (containing a nine partition grid) appears over the actual window. This outline tracks the mouse's movement and allows you to see how you are "stretching" the window. At the same time, a small box displaying the window's dimensions appears in the upper-left corner of the screen. (For most clients, dimensions are measured in pixels; for *xterm*, dimensions are measured in characters wide by lines high.)

3. This is a potentially tricky step. Move the mouse across the border you want to change—toward the outside of the window. You must move the mouse slightly outside the window, even if you want to drag a border *in* to create a smaller window.

   If you want to resize the window in both directions, move the mouse to a corner; again, you want to edge the mouse just outside the corner before resizing.

   If you don't cross a border or corner, you won't be able to resize the window.

4. Then drag the border or corner in any direction you want. (The other sides of the window will remain stationary.) You can drag a vertical border right or left; a horizontal border up or down. If you move the mouse to a corner, you can drag the corner out or in, changing both dimensions. As you resize the outline, the small box in the upper-left corner of the screen displays its changing dimensions.

5. When the window is the size you want, release the mouse button. The outline and resizing box disappear. The window is redrawn in the dimensions you selected.

Although we've suggested it in the preceding steps, you don't actually have to cross a *specific* border or corner to resize the way you want. Once you cross *any* part of the window frame (and the resizing function is initiated), you can drag the resize cursor around the perimeter of the outline and stretch any part.

You'll probably need to experiment a little to get used to this effect. Once you get the hang of moving the mouse outside the window when you begin, you'll find that the actual resizing is a bit like stretching a rubber band.

Clicking on the resize command button also raises the window to the top of the stack.

Note that resizing an *xterm* window will not change the dimensions of the text currently in the window. If you make the window smaller, for instance, some of the text may be obscured. On most operating systems, this should not be a problem. As you continue to work, perhaps starting a text editor, the text will be adjusted to display in the newly sized window. Problems are more likely to occur if you resize a window *during* a text editing session. Chances are that the text editing program will not know about the window's new size and will operate incorrectly. To solve this problem, simply quit out of the editor and start another session.

2.5 Starting Additional Clients

Now that you know the basics of managing windows, you can start other X clients just like you can start another instance of *xterm*. (As a note, ending an *xterm* session is accomplished by typing `exit` in the *xterm* you wish to leave.) The following sections describe how to open more client windows and place them on the display in convenient positions.

To start any client, at the command-line prompt in any *xterm* window, type the name of the client followed by an ampersand (&) to make the client run in the background. For example, by typing:

```
vega(mcentyre) 1% oclock &
```

you can start a clock application. The clock will appear on the screen at a "random" location. Move the clock to a more desirable position on the screen. Though moving windows on the display is fairly simple, manually positioning every window is not particularly convenient. For most clients, X provides a way to specify a window's location (and also its size) automatically—using an option when you run the command. A window's size and location is referred to as its *geometry* and you set these attributes using the `-geometry` option. The use of this option is discussed on page 16, in section "Window Geometry: Specifying Size and Location," later in this chapter.

2.5.1 Command-Line Options

Most X clients accept two powerful and extremely useful options: the `-geometry` option, which allows you to specify the size and location of a window on the screen; and the `display` option, which allows you to specify on which screen a window should be created. (Most commonly, you'd use the `-display` option to run a client on a remote machine and display the window locally, that is, on your display — the student computing room serves as a good example. It contains several PCs running OS/2 and an X server allowing you to run clients on vega (our fileserver), but display on the PC.)

The next section illustrates some typical uses of the `-geometry` option. In explaining how to use them, we introduce some new, perhaps somewhat involved concepts (such as the way distances can be measured on your screen). Bear with us. We feel that you need to master the `-geometry` and `-display` options in order to begin to take advantage of the powers of X. The DISPLAY environment variable will probably be set up for you, so you'll never need to use the option. We will therefore leave a detailed discussion of it to your investigation. The O'Reilly series on X windows provides in-depth help on altering the display screen of clients. Should you be interested in either a quick introduction or a detailed analysis, please stop by our office (3105 J McGavran) and borrow a book.

2.5.1.1 Window Geometry: Specifying Size and Location

The command-line option to specify a window's size and location has the form:

```
-geometry geometry
```

The `-geometry` option can be (and often is) abbreviated as `-g`, unless the client accepts another option that begins with "g."

The parameter to the geometry option (`geometry`), referred to as the "standard geometry string," has four numerical components, two specifying the window's dimensions and two specifying its location. The standard geometry string has the syntax:

```
width×height±xoff±yoff
```

Obviously, the first half of the string provides the `width` and `height` of the window. Many application windows are measured in pixels. However, application developers are encouraged to use limits that are meaningful in terms of the application. For example, *xterm's* dimensions are measured in columns and rows of font characters. More precisely, an *xterm* window is some number of characters wide by some number of lines high (80 characters wide by 24 lines high is usually the default).

The second half of the geometry string gives the location of the window relative to the horizontal and vertical edges of the screen. Imagining the screen to be a grid (where the upper-left corner is 0,0), `xoff` (x offset) and `yoff` (y offset) represent the x and y coordinates at which the window should be displayed. The x and y offsets are measured in pixels.

Many users may not be accustomed to thinking in terms of pixels. What exactly is a pixel? A pixel is the smallest element of a display surface that can be addressed by a program. You can think of a pixel as one of the tiny dots that make up a graphic image, such as that displayed by a terminal or a television. The number of dots (or pixels) per inch of screen determines the screen's *resolution.* The more dots per inch, the higher the resolution (and, hypothetically, the shaper the picture). This is, of course, not exactly true, but the details are not important for an understanding of this chapter.

Since a pixel is a tiny unit of measurement, gauging sizes and distances in pixels will take some practice. For instance, what are the dimensions of your screen in pixels (its resolution)?

So how do we use the geometry option to specify a window's size and location? First, be aware that you can specify any or all elements of the geometry string. Incomplete geometry specifications are compared to the application's default settings and missing elements are supplied from these values. All client windows have a default size. For example, if you run an *xterm* window with the geometry option and specify a location but no dimensions, the application default size of 80 characters by 24 lines is used.

If you don't specify the x and y coordinates at which to place a client window, X windows will place the window for you in a pseudo random position.

For now, let's just specify a window's location and let the size be the application default. The x and y offsets can be either positive or negative. If you specify positive offsets, you're positioning the left side and top side of the window. Negative offsets are interpreted differently. The possible values for the x and y offsets and their effects are shown in Table 2.1.

Offset Variables	Description
+xoff	A positive x offset specifies the distance by which the left edge of the window is offset from the left side of the display.
+yoff	A positive y offset specifies the distance by which the top edge of the window is offset from the top of the display.
-xoff	A negative x offset specifies the distance by which the right edge of the window is offset from the right side of the display.
-yoff	A negative y offset specifies the distance by which the bottom edge of the window is offset from the bottom of the display.

Table 2.1: Geometry Specification x and y Offsets

For example, the command line:

```
vega(mcentyre) 1% xclock -geometry -10+10 &
```

places a clock of the default size in the upper-right corner of the display, 10 pixels from the right and top edges of the screen.

If you want a window placed away from one or both edges of the screen, the guesswork starts. How many pixels away from the left side of the screen? How many pixels down from the top? You'll have to experiment with placing some clients on your screen to get a better idea of x and y offsets.

You can also place some windows in different places by dragging and then determine their geometry specifications using the *xwininfo* client.

Be aware that the geometry option is not necessarily the only means available to specify window size and location. Several clients allow you to set the size and location of a window using resource variables (in an .Xresources or other defaults file), but that's a topic too detailed for the scope of this chapter. For details on resources, check out the available literature in our office.

Chapter 3

Unix - An Introduction

Unix is unique in that it wasn't designed as a commercial operating system to run application programs, but as a hacker's toolset, by and for programmers. In fact, an early release of the operating system went by the name PWB (Programmer's Work Bench).

When Ken Thompson and Dennis Ritchie first wrote Unix at AT&T Bell Labs, it was for their own use, and for their friends and co-workers. Utility programs were added by various people as they had problems to solve. Because Bell Labs wasn't in the computer business, source code was given out to universities for a nominal fee. Brilliant researchers wrote their own software and added it to Unix in a spree of creative anarchy that hasn't been equalled since, except perhaps in the recent introduction of the X Window System.[1]

This Chapter will cover some of the basic utilities available with the Unix operating system and some of the "add ons" provided over the years. It is broken up into sections dealing with different aspects of the Unix environment; directories, files, printing, etc.... It does not cover the full range of options for each command, but it should be enough to get you started toward discovering the power of Unix.

3.1 User Commands

What follows is a basic description of the most commonly used Unix commands. For detailed help on these commands type `man` *command*. A useful command line option for `man` is the `-k` option. This causes the search to be executed as a keyword search instead of an exact match

[1] From "Unix Power Tools", O'Reilly & Associates.

search. For example, try typing `man -k rename`. This will find all commands that have `rename` in the command or in the description. The format of the output in this example is

```
mv (1)                    - move or rename files
```

The command `mv` is listed along with several others. The one refers to the section of the man pages where this command is located. Many of the available command line options for these commands will not be covered here. You can also use the **Help** option on the root window menu. Once in **Help** pull down the options menu and select search.

3.2 Directories

The most important thing to understand about Unix is its directory structure. When you are in a directory and you do a listing, you will *always* see the directories . and .. These represent the current directory (.) and the parent directory (..), or the directory above the current one. These special symbols are very useful when working with files as you will see in section 3.3. The basic structure of the Unix file system always starts with the root directory (/). All other (child) directories stem from the root. For example, your account is located in /home/users/<username>, where <username> is your user ID. What this represents is that off the root directory there is a directory called home. Off home there is one called users, and off users is your account, in directory <username>.

3.2.1 Print Working Directory - pwd

I thought it would be appropriate to start out with the command that tells you where you are. When you type **pwd** at the command prompt, Unix returns with the full path name of your current directory. For example, if you are in your home directory (where you are placed at login), and you type **pwd** you will get something like:

```
vega(mcentyre) 1% pwd
/home/users/mcentyre
vega(mcentyre) 2%
```

This sounds very simplistic at first, but you will find that there are many times that you have no idea where you are and pwd becomes quite useful.

3.2.2 Change Directory - cd

The **cd** command seems like the next logical command to learn. It allows you to change from one directory to another. It is necessary that you know where you are in order to use this command, unless you want to type in the full path. To use **cd**, you simply type in the name of the directory you wish to change to. For example, if you are in your home directory, and there is a directory named tryme, you would type:

```
vega(mcentyre) 1% cd tryme
```

The computer would respond by moving you to that directory and telling you where you are. Now if you type **pwd**, you'll see that you are in a new directory (viz., /home/users/<username>/tryme). You can also change to a directory by specifying the complete path. For example, if you are in your home directory, but you'd like to be in /usr/bin/X11 (DOS users: note the forward slash – not a backslash), you would type:

```
vega(mcentyre) 1% cd /usr/bin/X11
```

Since the parent of a directory is called **..**, typing:

```
vega(mcentyre) 1% cd ..
```

will move you up (or back) one directory. The only exception to this rule occurs when you are in the root directory (/). Its parent is itself, so typing **cd ..** when you are in the root directory just keeps you in the root directory.

3.2.3 Make Directories - mkdir

Now that you know how to find out which directory you are in and how to move around directories, you probably want to create some. Making a directory is a very simple process. Simply decide on a name and then type:

```
vega(mcentyre) 1% mkdir directoryname
```

where directoryname is the name you decided upon.

For creating a directory whose parent directories might be missing, you can use the command line option `-p`. The command works by moving down the specified directory and creating each directory that doesn't exist. For example, if I want to create a directory and subdirectory under my home directory called temp/paper. Since temp doesn't exist you have two choices. You can either make the directories seperately by typing:

```
vega(mcentyre) 1% mkdir temp
vega(mcentyre) 2% mkdir temp/paper
```

or

```
vega(mcentyre) 1% mkdir -p temp/paper
```

The second choice will create both. If you do use this option, be careful. Long path names could result in a lot of unnecessary directories if type-os are made, which brings us to our next section...

3.2.4 Remove Directory - rmdir

Once you have a few directories, you will eventually run into the need to get rid of some. A directory is really only a file that contains names and other information associated with the files within it, but it has its own special creation and deletion routines so people don't create files when they mean to create directories or destroy entire directories when they only mean to delete some files. The command **rmdir** is used to delete empty directories (for a tip on quickly removing the contents of a directory or several directories see Section 3.3.4 on removing files).

Once you have an empty directory that you'd like to remove, using **rmdir** is simple. For example, to remove an unwanted directory, you type:

```
vega(mcentyre) 1% rmdir unwanted
```

That's it! It's gone.

3.3 Files

Files are the basic building blocks of a Unix system. Even directories are really only files that contain all of the information about the directory. This section will show you how to manipulate your files as well as show you a few tips, tricks, and shortcuts to make you more productive as a file manager. This section covers the basics of copying, moving, renaming, listing, removing, viewing, and searching files. If you need a more detailed discussion, see the man pages or ask Bruce or Corey for assistance.

3.3.1 Copying - cp

There are basically three things you can do with the **cp** command:

- Copy filename1 to filename2.

  Example: `cp filename1 filename2`
  This will copy filename1 into filename2. If filename2 already existed it will be overwritten (unless you specify the `-i` option, in which case you will be asked if you wish to overwrite filename2). Otherwise, filename2 will be created.

- Copy the contents of directory1 to directory2 (either by creating directory2 if it doesn't exist, or by creating a subdirectory in directory2 called directory1, if it does exist).

  Example: `cp -R directory1 directory2`
  The -R is required (it means to copy recursively).

- Copy filename (or a list of filenames) to directory.

  Example: `cp filename directory`
  This copies the file *filename* to the *directory* directory (only if *directory* exists) and gives it the same name.

Beware of a recursive copy like this:

```
vega(mcentyre) 1% cp -r ~/src ~/src/bkup
```

which keeps copying files until it fills the entire file system. If you have difficulties, consult the man pages or ask Bruce or Corey for help.

3.3.2 Moving and Renaming - mv

mv moves files and directories around in the file system. A side effect of **mv** is to rename a file or directory. Like the **cp** command, the **mv** command has three basic forms:

- Move (change the name of) filename1 to filename2.

Example: `mv filename1 filename2`
If filename2 already exists, it is removed before filename1 is moved (unless you specified the `-i` option in which case you would be prompted to remove filename2).

- Change the name of directory1 to directory2.

  Example: `mv directory1 directory2`
  This works *only* if directory2 dosen't exist. If it does exist then this would follow the logic of the third example.

- Move one or more filenames (and/or directories) into the last directory on the list.

  Example: `mv filename(s) directory`
  This would move all files (and directories) in the filename(s) list to directory. In this form, **mv** does not alter any of the file names.

The command line options available for **mv** are:

`-f` Force. Override any mode restrictions and the -i option. The -f option also suppresses any warning messages about modes which would potentially restrict overwriting.

`-i` Interactive mode. **mv** displays the name of the file or directory followed by a question mark whenever a move would replace an existing file or directory. If you type a line starting with y, **mv** moves the specified file or directory, otherwise **mv** does nothing with that file or directory.

3.3.3 Listing - ls

To see the files in your current directory (i.e., the one displayed by a **pwd**) you can use the **ls** command. Generally, most people use an alias[2] to refer to a list command with some command line options. You see, **ls** dosen't really give you that much information. It lists both files and directories without distinguishing between them, and says nothing about the ownership or protection on a file. The following command line options are used to modify the information given by **ls**. You will have to choose which information format you like best.

[2] See Section 3.5 for help on defining and using aliases.

`-a` List all entries; in the absence of this option, entries whose names begin with a "." are not listed

`-C` Force multi-column output, with entries sorted down the columns; for **ls**, this is the default when output is to a terminal.

`-F` Mark directories with a trailing slash ("/") and executable files with a trailing asterisk ("*").

`-l` List in long format, giving mode, number of links, owner, size in bytes, and time of last modification for each file. If the time of last modification is greater than six months ago, it is shown in the format "month date year"; files modified within six months show "month date time". If the file is a symbolic link the pathname of the linked-to file is printed preceded by "->". /usr/5bin/ls will print the group in addition to the owner.

`-r` Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.

`-R` Recursively list subdirectories encountered.

`-t` Sort by time modified (latest first) instead of by name.

Use the `man ls` command to get more options.

3.3.4 Removing - rm

The command **rm** removes directory entries for one or more files. To remove a file, you must have write permission for its directory (see Section 3.2.4 for a discussion on how to remove directories). If you have write permission on the directory, but not on the file, **rm** will display the file's permissions and wait for you to respond to the question of whether or not you really want to delete the file. The following is a list of options available to use on the command line with **rm**:

`-f` Force files to be removed without displaying permissions, asking questions or reporting errors.

`-i` Ask whether to delete each file, and, under -r, whether to examine each directory. Sometimes called the interactive option.

`-r` Recursively delete the contents of a directory, its subdirectories, and the directory itself. Be *very* careful with this option...it's dangerous.

We have set up **rm** so it is aliased as "rm -i" automatically and would recommend you leave it that way until you have more experience with Unix. Especially if you plan on using file expansion characters or recursive deletes. You also might want to take a look at what you're deleting before giving it the axe. Thus, onto our next section...

3.3.5 Viewing Files

Below are four sections on ways to view a file. Each one has quite a range of options, but only the most useful are covered. For a better description of other available options, see the man pages on a particular command, or check out one of the available Unix books in our office.

3.3.5.1 A Whole Bunch At Once - cat

Generally, **cat** is used to show the whole file or send a file to another Unix utility[3]. It stands for concatenate and display. The reason for this is that you can **cat** several files on the command line and it will show them one after another (concatenated). The problem with **cat** is there is no way to slow down the output for large files or multiple files that result in output longer than one screen. Thats why the use of **cat** is usually reserved for redirects and pipes. Two useful command line options are `-b` and `-n` which number the lines. The first, `-b`, numbers lines of output, but it omits the blank lines. `-n` will number blank lines and non-blank lines. This is useful when you are working on a computer program and you'd like to check an error on line 1234 of your printout.

3.3.5.2 A Little at a Time - more

The command **more** is like **cat** in that you can specify several files on the command line, but it buffers its output to the length of your screen. This way you can move slowly through a file reading a screen at a time. **more** has some other very nice features. At the bottom of each screen is the line "–More–" and an indication of the percentage

[3]See section 3.6 on redirection and pipes.

of the characters displayed so far. **more** scrolls up to display one more line in response to a RETURN character; it displays another screenful in response to a SPACE character. Other commands are listed below. **more** sets the terminal to noecho mode, so that the output can be continuous. Commands that you type do not normally show up on your terminal, except for the / and ! commands. If the standard output is not a terminal, more acts just like cat, except that a header is printed before each file in a series. Since **more** has such a large range of options and features, I'll leave it to you to try them out. Either type `man more` at a prompt or type a `?` while using **more** to see a help screen. The `-c` option causes **more** to clear the screen instead of scrolling. This is a very useful option on slower machines as clearing and then displaying text is much faster than scrolling. The `-d` option is a very useful one for inexperienced users of **more**. Usually, when you enter an unrecognized command, **more** simply beeps at you. This option makes **more** display error messages and helpful hints.

3.3.5.3 Just a Little Off the Top - head

The format for **head** is `head [ -n ] [ filename... ]`, with the `-n` and the `filename` being optional. **head** copies the first n lines of each filename to the standard output. If no filename is given, head copies lines from the standard input. The default value of n is 10 lines. When more than one file is specified, the start of each file looks like:

```
==>filename<==
```

3.3.5.4 Just the End, Please - tail

A mirror image of **head**, **tail** simply does the same work except on the end of a file. You can specify starting points relative to the beginning or the end of the file using the +/-number option on the command line, where number can be in lines, blocks, or characters (you specify which by the command line options l, b, or c - lines is the default). I'm not sure why you would ever want to do this, but the `-r` option will copy the lines in reverse order (just for fun).

3.3.6 Searching - grep

grep is an amazingly powerful searching utility. In its most basic form, **grep** is used to locate string patterns in files. For example, to search a file called foo.bar for the string “find me” you would type

```
vega(mcentyre) 1% grep "find me" foo.bar
```

It is that simple. While **grep** can be used in this manner for simple searches, it can also perform searches for regular expressions and strings of variable length. You should use care when specifying searches for strings containing the characters “$”, “*”, “[”, “ˆ”, “—”, “(”, “)”, and “\” in the expression, as these characters are also meaningful to the shell. It is safest to enclose the entire expression argument in single quotes. When any of the grep utilities is applied to more than one input file, the name of the file is displayed preceding each line which matches the pattern. The filename is not displayed when processing a single file. Below are some of the more commonly used command line options:

`-c` Display a count of matching lines rather than displaying the lines which match.

`-i` Ignore the case of letters in making comparisons - upper and lower case are considered identical.

`-l` List only the names of files with matching lines (once) separated by NEWLINE characters.

`-v` Invert the search to only display lines that do *not* match.

The **grep** command has much more flexibility than we've led you to believe. If you're interested in learning more about this powerful tool, we would suggest trying the man pages. **grep** also has two related commands (**egreg** and **fgrep**) which have their own specific capabilities and features.

3.4 Printing

Our printing set-up consists of two printers, the IBM Lexmark (*lex_3107*) and the Hewlett-Packard LaserJet Series III (*hp3_3105*). Currently, the two computers in 3107 (byrgius and hecate) print to the Lexmark by default. vega is also defaulted to the Lexmark. deneb, tycho, and lyra are all defaulted to the Hewlett Packard. If you prefer one of these printers over the other and would like to ALWAYS print to the same printer, add one of the following lines to your .cshrc file:

```
setenv PRINTER hp3_3105
or
setenv PRINTER lex_3107
```

Otherwise, you will find your output at the specified default printer.

3.4.1 Print It - lpr

The **lpr** command will send your file (or standard input if no file is given) to the spooling area. Useful options include:

`-P` Send output to the named printer. Otherwise send output to the printer named in the PRINTER environment variable, or to the default printer, lp.

`-m` Send mail upon completion. (*Very* nice feature for large jobs when working in a different location than the printer)

`-d` The files contain data from TEX (DVI format from Stanford).

For example, if you wanted to print the TEX DVI file foo.bar on the HP printer while working from hecate, you would type:

```
vega(mcentyre) 1% lpr -Php3_3105 -d foo.bar
```

That is assuming you do not have the PRINTER environment variable set.

3.4.2 On Queue - lpq

The form of the command for **lpq** is:

```
vega(mcentyre) 1% lpq [ -Pprinter ] [ -l ] [ + [ interval ] ]
  [ job# ... ] [ username ...  ]
```

lpq displays the contents of a printer queue. It reports the status of jobs specified by job#, or all jobs owned by the user specified by username. **lpq** reports on all jobs in the default printer queue when invoked with no arguments.

For each print job in the queue, **lpq** reports the user's name, current position, the names of input files comprising the job, the job number (by which it is referred to when using **lprm** (see section 3.4.3)) and the total size in bytes. Normally, only as much information as will fit on one line is displayed. Jobs are normally queued on a first-in-first-out basis. If **lpq** warns that there is no daemon present (that is, due to some malfunction), the **lpc**[4] command can be used to restart a printer daemon.

Two of the more useful options follow:

`-P` Display information about the queue for the specified printer. In the absence of the -P option, the queue to the printer specified by the PRINTER variable in the environment is used. If the PRINTER variable isn't set, the queue for the default printer is used.

`-l` Display queue information in long format; includes the name of the host from which the job originated.

3.4.3 Removing From the Queue - lprm

lprm removes a job or jobs from a printer's spooling queue. Since the spool directory is protected from users, using **lprm** is normally the only method by which a user can remove a job.

Without any arguments, **lprm** deletes the job that is currently active, provided that the user who invoked **lprm** owns that job. You can remove a specific job by supplying its job number as an argument, which you can obtain using **lpq**. For example:

```
vega(mcentyre) 1% lpq -Plex_3107

Rank    Owner     Job  Files                           Total Size
active  mcentyre  385  standard input                  35501 bytes

vega(mcentyre) 2% lprm -Plex_3107 385
```

The `-P` option allows you to specify a printer. In the Biostatistics department, we currently have two printers connected to the Sun LAN. They have been given the names lex_3107 (a Lexmark in room 3107) and hp3_3105 (a Hewlett Packard LaserJet III in room 3105). Another nice option is a `-`. It will remove *all* jobs owned by you.

[4] See the man pages for information on how to use **lpc**.

3.5 Aliases

Aliases are the lazy man or woman's dream come true. They allow you to define a command using current Unix commands or current aliases with a name of your choice. To see currently defined aliases (you have some common aliases set up for you) simply type `alias` at the command prompt. Aliases are convenient because you can save long command lines in one name, or difficult to remember commands in more descriptive names. In fact, you could very easily create aliases to make your Unix environment look and feel like DOS.

The format for creating an alias contains three parts. First the Unix command **alias**, then the name you would like the alias to have, and finally, the command the alias represents. Here are some examples of aliases:

```
vega(mcentyre) 1% alias ll "ls -la"
vega(mcentyre) 1% alias rich "finger rich@rock.west.ora.com"
```

Now, if I type **alias**, I will get a listing that includes the two aliases I just defined. To remove an alias, you simply use the **unalias** command. For example:

```
vega(mcentyre) 1% unalias rich
```

Now if you try to type the command `rich`, all you'll get is an error message.

Adding these commands each time you log in could be quite tedious, especially if you have a lot of them. To make life easier on you, the Unix C shell has a configuration file which will allow you to define (among other things) aliases. You add the same line you would normally type in on the command line to your .cshrc file, and it will be executed for you each time you log in.

3.6 Pipes and Redirects

Perhaps the best reason to use Unix are these two powerful tools. Pipes are used to send the output of one command into the input of another.[5] Redirection is used to send the output of a command to a file, or to

[5]In the examples that follow, the commands **who**, **grep**, and **ls** are used to demonstrate pipes and redirects — these commands are described in this manual and in the man pages.

send a file into the input of a command. They take a little getting used to, but once you understand how to use the two commands, you'll find them invaluable. The pipe is denoted using the vertical bar (|). Redirection is denoted using the greater than and less than signs (> and <). Redirection can be used in two ways.

1. The most common way of using redirection is to send the output of a command to a file. This can be done by either creating the file, or by appending to the bottom of the file. For example:

   - Create the file and send the output to the file:

     ```
     vega(mcentyre) 1% who > filename
     ```

   - Append the output of the command to an existing file:

     ```
     vega(mcentyre) 1% who >> filename
     ```

2. Other times you might want to use a file as input to a command. Continuing with the example above, we could take the file that now contains the output of who and search for a friend of ours using the **grep** command.

   ```
   vega(mcentyre) 1% grep walter < filename
   ```

Pipes give you the ability to do things like the above example, but without the use of temporary files. For example, to accomplish the task of searching for a friend using the **grep** and **who** commands, you would need only type:

```
vega(mcentyre) 1% who | grep walter
```

This takes the output of **who** and pipes it directly into **grep** which uses it as input to search for walter. Here's another useful example:

```
vega(mcentyre) 1% ls -la | more
```

This gives a directory listing and pipes the output from the command into the **more** program which then displays it one screen at a time. One last example, just to show off the power of pipes:

```
vega(mcentyre) 1% who | grep ":0" | sort | more
```

This will take the output of the **who** command, search it for the string ":0", sort it alphabetically, and send it to **more** to show one screen at a time. While this command is not very useful, imagine the time and effort it would require to write and debug a program to do the same. And if it *was* a useful command, you could make it a nice short alias.

3.7 History Commands

This is *the* section for lousy typists (like myself) and lazy people (ditto). The Unix history mechanism keeps a list of all the commands you type on the command line. We have set up a few defaults to help you get started right away using history commands. First off, we specified (as a default) that the C shell should store your last 40 commands. When you type `history`, you will see that list with each command numbered. We have also set up you default prompt (the vega(mcentyre) 1%) you see each time you are prompted for another command. The number before the percent sign is there to tell you your current command number. These numbers are useful when referencing previous commands. The most common use of the history commands is the simple "repeat last command". It is denoted by using the "double-bang" (!!). This will execute the last command on the history list. The second most common use is to execute a particular command (say number 23). This is done by typing:

```
vega(mcentyre) 58% !23
```

That's it! The 23rd command will execute just as if you had typed the whole thing out. The history mechanism can also search based on a string. For example, suppose you do not remember the number of the command you want to execute (a *very* common occurance), but you do remember that it started with the word latex. You could simply type:

```
vega(mcentyre) 1% !latex
```

This will find the last command that started with "latex" and execute it.

There are also editing commands that you can use should you make a mistake on a long command. I will only be discussing two of the editing commands here, but there are many more. For those interested in becoming truly lazy, we have several books which cover the topic in detail. The easiest to use is the search and replace command. To use it, you simply specify the string to search for and the string to replace. It will grab the last command and try to execute it with your changes. For example, suppose you typed:

```
vega(mcentyre) 1% cat fname1 fname2 f-oops-name3 | more
```

when you wanted to type:

```
vega(mcentyre) 1% cat fname1 fname2 fname3 | more
```

Instead of retyping the entire line (and possibly making a different mistake), you could type:

```
vega(mcentyre) 2% ^f-oops-na^fna^
```

and it's fixed. Another version of this same command allows you to specify any command to modify. It uses the "bang-number" facility *and* a search and replace string. Here is an example of it being used on the previous mistake (take note of the number used):

```
vega(mcentyre) 3% !1:s/f-oops-/f/
```

This translates into "take the command numbered 1 and substitute (:s) the first string for the second" (where the / is the string delimiter).

A useful option that can be placed on the end of any command line is the :p. It has the effect of placing the command on the history list without actually executing it. This is sometimes useful when correcting many mistakes on a very long command line. I usually refer to it as the print command option.

Chapter 4

User Software - Unix and X Windows

4.1 X Windows Software

As was mentioned in previous sections, X has a large amount of software written for it that is freely available to the public. Corey McEntyre and Bruce Walter combed through all of the applications available, and created a menu structure for the most useful ones. This is the menu that appears on your root window. We also have a great deal of software that was not written for X windows, but is still available for your use on the Unix machines.

For detailed descriptions of the options available for each of these packages, see the man pages. For more complex questions, come by 3105J McGavran - we've probably got a book on the subject.

4.1.1 Emacs

Emacs is mouse aware in X and the menu is fairly self explanatory. There are also three hidden menus that are activated by holding down the control key while pressing one of the mouse buttons. These three menus are a buffer selector, a series of help menus, and a font selector for the text (this is *very* handy for people working on the smaller Sun workstations).

4.1.1.1 Introduction

This section describes the Emacs text editor, a very widely used and accepted alternative editor to *vi* for unix operating systems. Emacs was originally written by Richard Stallman of MIT in the 1970s and continues to evolve today. The current version as of this writing is GNU Emacs 19.19.2, a free version of Emacs distributed by the Free Software Foundation. For more information about the Free Software Foundation see the *GNU Manifesto* available through the *Info* browser in Emacs. The GNU Emacs reference manual provides a complete description of GNU Emacs and the MG reference manual by Sandra Loosemore contains an excellent treatment of the basic commands. These are available as TEX files at several ftp sites along with the *mg* program, a slim yet very extensive and fast Emacs clone for the pc.

4.1.1.2 Getting Started

To get into Emacs you simply type `emacs` at the system prompt followed by the name of the file you wish to edit. If no file by the specified name exists Emacs will create a new file by that name. For new Emacs users, it is strongly recommended that you work through the tutorial provided with the standard installation of Emacs. To do this just enter Emacs without specifying a file name and follow the directions on the screen. Emacs has an online information and help utility which can be entered through help. There's also a reference card (as a TEX file) which contains the basic commands in the Emacs directory `/home/emacs/lib/etc` called refcard.tex for those who wish to have a convenient list of the common commands.

When you enter Emacs you are in *insert* mode, meaning that anything you type will be inserted in the file at the cursor's location; any text present will be pushed to the right. Actually, unlike *vi* in Emacs you are *always* in insert mode; commands are entered by appropriate keystrokes and entering text into the *minibuffer*.

4.1.1.3 Basic Emacs Concepts

Command	A command in Emacs is a series of keystrokes that execute a specific task. Most commands start with the control or escape key, which we will hereafter abbreviate as C- and E-, respectively. Alternatively, on the sun terminals the diamond key also serves as the escape character, but in this case it's referred to as the *meta* key. A command such as `C-v` means to hold the control key down and hit the letter `v`. `E-v` means press and release the escape key and then hit the letter `v`. `META-v` means to hold the meta key down and hit the letter `v`. Almost all Emacs commands are case-insensitive.
Extended Commands	These commands are not bound to any specific keys; instead the user must type in the command by typing `E-X` and then entering the command in the area at the bottom of the screen.
Buffer	A block of text (not necessarily a file).
Mini-Buffer	This is the line at the bottom of the screen where you enter commands; this region is also the *echo area*, the place where you view warnings, prompts, and error messages.
Window	An area on the screen.
File	A block of text that can be copied to or from a buffer.
Kill-Buffer	The buffer contains the most recently deleted block of text. Text in the kill-buffer can be retrieved with a *yank* command.
Point	The current cursor location in the active buffer.
Mark	An invisible pointer in a buffer.
Region	The area between *point* and *mark*.
Mode	A special editing scheme which allows many additional commands; for example, in LaTeX mode the user can match beginning and ending statements with a command.
Text Area	The part of the screen where the current buffer's contents are displayed.
Mode Line	The second to last line at the bottom of the window. It displays the current buffer name, the editing mode and other information.

In the next few sections many frequently used commands are listed.

4.1.1.4 Moving Around in a File

If your terminal is set up correctly the arrow keys should move left, right, up, and down in a file. The following commands may also be of use. Note that depending on how your kill key is defined, backspace, delete, or `C-h` will delete the previous character. Also note that `M-x` key sequence must be followed by the appropriate text entered in the minibuffer.

C-f	forward one character
C-b	backward one character
return	next line (pushes text to right)
C-n	next line
C-p	previous line
C-a	beginning of line
C-e	end of line
E-f	forward word
E-b	backward word
C-v	scroll down one screen
E-v	scroll back one screen
E-x <	beginning of buffer
E-x >	end of buffer

4.1.1.5 Changing Text

One of the most useful things about Emacs is its ability to perform block copy and deletes (cuts). To copy a region of text you need to mark the beginning of the region with `C-@` and then move to the end of the region and hit `E-w`. Then move the cursor to the place where you want the marked region to be copied to and type `C-y` to yank it back from the kill-buffer. To move the marked text use `C-w` instead of `E-w`.

C-@	set mark (used for block moves/deletes)
C-w	cut region to kill buffer
E-w	copy region to kill-buffer
C-d	delete character
E-d	delete forward word

E-x over	change to overwrite mode
C-k	kill text to end of sentence
C-y	yank text from kill buffer

4.1.1.6 Searches

Emacs implements an *incremental search*, meaning that it moves to the next occurrence of the characters as they are typed. To perform a forward search type `C-s` followed by the text you want to find. (NOTE: For some terminals the `C-s` sequence is the XON/XOFF signal that freezes the screen; if this happens you need to enter `C-q` to resume.) To implement a reverse search use `C-r` and the search string. A search and replace is done by entering `E-%` and following the directions. Much more about searching and replacing can be found in the aforementioned references.

4.1.1.7 Manipulating Files, Buffers, and Windows

Emacs has a useful mode called directory editor which allows you, among other things, to move around directories and choose the file you wish to edit. For more on the directory editor use the help facility. The following commands for manipulating files, buffers, and windows are useful:

C-x d	load directory editor
C-x C-f	load file
C-x C-r	load file in read-only mode
C-x k	kill buffer
C-x C-s	save buffers
C-x b	switch to another buffer
C-x o	switch to other window
C-x 1	zoom current buffer
E-C-v	scroll other buffer

4.1.1.8 Miscellaneous

Obviously this is a bare bones version of the Emacs manual and should only be used as a quick reference to get you started. Nevertheless, most of the common editing commands have been covered here and will be sufficient for most people. User preferences and defaults can be set up by specifying the appropriate commands in a .emacs file stored in the user's home directory. Emacs also has a built-in *vi* emulator which is especially useful for people who have a working knowledge of *vi*. For further details see the GNU Emacs reference manual or use the *info* browser.

C-x C-x	exit Emacs
C-z	exit to unix shell (type `fg` to return)
C-g	cancel any command in progress (clear minibuffer)
C-l	redraw display
E-x set-fill-column	set fill column to current column
E-q	rejustify text in current paragraph
M-x go	goto line number
C-x =	print file information in minibuffer
E-x vip	enter *vi mode*
C-h	Help
C-h i	run info browser

4.1.2 XArchie

Before we can explain XArchie, you need to understand what Archie is and why it is so useful. The Archie information system is a network-based information tool offering proactive data retrieval and indexing for widely distributed collections of data.

Perhaps the best known application of the Archie system is to maintain the Internet Archives database. This database, already available from a number of service providers across the Internet, currently contains the names of over 2,100,000 files at over 1,000 anonymous ftp archive sites. Using this database, users can rapidly locate needed files without the need to log onto dozens or even hundreds of machines. Archie servers offering this database currently receive over 50,000 queries per day. It is this database that the Xarchie client accesses, as described in the next section, using one of the many independently owned and operated Archie servers around the world.[1]

XArchie is an X11 browser interface to the Archie Internet information system using the Prospero virtual filesystem protocol. Archie provides information about files available for ftp anywhere on the Internet;

[1]From the help pages in XArchie.

XArchie displays this information using an easy-to-use, point-and-click interface. XArchie allows you to further explore ftp sites by examining directories returned as query matches, and allows you to retrieve files located this way. XArchie is designed (like most X applications) to be highly customizable, allowing you to tailor the look-and-feel to your own preferences.[2]

If you decide to use XArchie to download files, be sure to set your ftp local directory under the Settings>, Other... menu to your desired directory. You may also want to set your ftp email address to usernamebiostat.sph.unc.edu under the same menu option.

4.1.3 XGopher

XGopher is an X window system client interface to the gopher information server. XGopher provides access to tremendous amounts of information which may be accessed from a local system or a remote information server. The source of the information is generally transparent, with data supplied from world-wide locations as easily as from a local on-campus server.[3]

For assistance with XGopher, see the on-line man pages — they are very thorough.

4.1.4 XTerm

The XTerm is the terminal emulator for X. It has three hidden windows on it that can be activated by holding down the control key and pressing one of the three mouse buttons. They control fonts, TeKtronix vs. vt terminal display, and window options.

4.1.5 XMan

XMan is the application that is called from the menu option Help on your root window menu. The manual page contains three sections. In the upper left corner are two menu buttons. When the mouse button is held on either of these options a new menu pops up. Directly to the right of the menu buttons is an informational display. This display usually contains the name of the directory or manual page being displayed. It is also used to display warning messages and the current

[2] From the XArchie man pages.
[3] From the XGopher man pages.

version of XMan. The last and largest section is the information display. This section of the application contains either a list of manual pages to choose from or the text of a manual page.[4]

4.1.6 Other Clients

As was mentioned earlier, X Windows has a great deal of software available for it. What follows are some of the other options available on the root window menu that you might find of interest.

The TWM Icon Manager is used to iconify or deiconify windows. It is also useful when a window is hidden and you want to locate it quickly. Figure 4.1 is an example of my icon manager running several windows. The small x in the box denotes that the window is iconified.

Figure 4.1: TWM Icon Manager

XBiff is a handy feature to show you that new mail has arrived. If you decide not to use xmailtool (which, if you leave it iconified, will also notify you of new mail), XBiff will beep and change from black on white to white on black. In figure 4.2, is xbiff waiting for new mail.

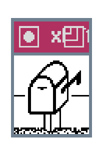

Figure 4.2: Mail Notification Icon

X Windows also has a nice calendar interface for keeping track of appointments and meetings. The calendar appears in figure 4.3.

[4]From the XMan Help screen.

Figure 4.3: X Windows Calendar

The X Windows calculator is not very powerful, but it is very easy to use. It is bound to the number pad on your keyboard for convienence. Its appearance is not the most attractive, but it gets the job done.

The clock that appears in the upper right corner of your display (see Figure 4.4) can also be set up to display in digital mode. Other than that, its only real purpose is to keep you from staring at your watch.

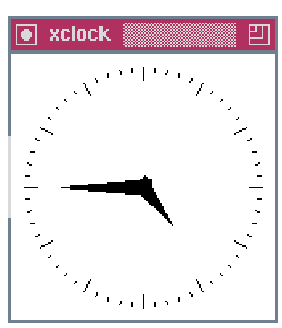

Figure 4.4: X Windows Analog Clock

For help with any of these applications, try the XMan tool located as Help on your root window. Or ask to borrow the X Window System Users Guide on reserve in our office (3105J McGavran).

4.2 TeX

TeX continues to be the world-wide standard for the production of mathematical text. It is capable of running on every computer system at UNC, supports every laser printer, and can perform every mathematical formatting operation. More and more books are being formatted with TeX. TeX documents are stored in simple ASCII files that are perfect for e-mail, making TeX the perfect choice for long distant collaboration. TeX "knows" mathematics. Mathematical expressions are specified using a simple set of operators that TeX uses to make positioning and sizing decisions.

TeX was developed at Stanford by Donald Knuth, and is in the public domain. The source of the TeX program has been published and is available from Addison Wesley.

This document was prepared using TeX. Actually, the document was prepared using LaTeX , a macro-package for TeX users that makes producing books with indices, bibliographies, and table of contents and much easier.

4.2.1 Using TeX

TeX is not a word processor. To use TeX, you create a file containing a TeX document by using your favorite editor. You do not need to learn a new editor to use TeX. A good choice, if you don't already have a favorite editor, is `Emacs`. Suppose you create a TeX document in a file called `sample.tex`. You would then follow the steps below.

Use the `tex` or `latex` command to convert your TeX document into a `dvi` file. To TeX a document stored in a file `sample.tex`, use:

```
vega(mcentyre) 1% tex sample
```

If your file name ends with `.tex` leave it off (as in the example above. If it doesn't, type the whole name.

If the file contains LaTeX source, type (for more information on LaTeX see section 4.3):

```
vega(mcentyre) 1% latex sample
```

If there are no errors, a `dvi` type file will be produced. A `dvi` file is a device-independent non-printable file which can be converted into a printable file by using a device driver for an available printer. `xdvi` is a graphics previewer for TEX and LATEX documents that will display the document on a graphics screen just as it will appear on paper (to the limits of the graphics screen). `xdvi` has commands to move around and magnify various portions of the document. We can also preview TEX output by converting it to PostScript^{tm} and previewing the PostScript^{tm} file.

To convert a `dvi` file to PostScript^{tm} use:

```
vega(mcentyre) 1% dvips sample
```

`dvips` takes a `dvi` file and creates PostScript^{tm} . See the man pages for `dvips` for details.

4.2.2 Metafont

The Metafont system for constructing fonts for use with TEX and LATEX has been installed and is ready for use. Metafont is a described in the "The Metafont Book" by Donald Knuth.

Metafont is used automatically by dvips (the TEX/LATEX file printing utility) to generate fonts that could not be found. This means, for example, if you attempt to use a large version of the `\tt` font, and dvips can't find it, dvips will call Metafont automatically to construct the font, then use the constructed font to complete the document. In the past, fonts that were not found by dvips generated "font substitutions" where a "close" font was used for the one that could not be found. This could produce output that was less than perfect. The new Metafont installation fixes this problem.

To use automatically constructed fonts, you must have a declaration for `TEXFONTS` in your .login. One is provided for you by the system defaults.

4.2.3 Learning TEX

TEX is a wondrous program — it makes the most beautiful mathematical output of any available software. But the program is large and ultimately flexible. This means that you need to go slow and build a solid foundation of TEX principles to be able to use it effectively in the long-run. Here are some steps that will help you learn about TEX:

1. Start with LATEX, not TEX itself. LATEX is much easier to use, and you can produce high-quality documents without seeing much of the TEXnical details. This manual was produced in LATEX.

2. To start learning LaTeX, read the document "Essential LaTeX", by Jon Warbrick. In just 17 pages, the basic principles are discussed and enough information is provided to enable you to produce memos, letters, and reports. No mathematical typesetting is discussed.

3. Add simple mathematics to your repertoire. Read the LaTeX book chapters 1 to 3. Chapter 3 discusses math mode and is quite readable.

4. Read "A Gentle Introduction to TeX," by Michael Doob. This will refine your TeXniques. Chapter 5 adds more details about mathematics.

5. If you still hunger for more, read the TeXbook. This witty, but incredibly detailed book is the definitive reference for TeX, written by its creator, Donald Knuth.

To whet your appetite for TeX, and LaTeX, the examples below show how to produce common typographical constructs. The examples include material from plain text typing as well as some common elements from statistical typing.

4.2.4 `xdvi`

Once you have created a `dvi` type file with LaTeX, or TeX you can view the results on the screen before committing them to paper. This can save a lot of paper.

Suppose you have a document called `sample.tex`. You run `latex sample` to create `sample.dvi`. Then you type:

```
vega(mcentyre) 1% xdvi sample
```

An X window appears (see Figure 4.5).

You can move around by using the push buttons along the right-hand side of the display. Use the first mouse button to activate the screen controls. You can also move around in the document by pressing simple keystrokes. To exit the previewer, click on Quit, or type `q`.

The mouse is used in `xdvi` to locate a "magnifying glass." You merely place the mouse cursor on the text you wish to see "close-up"

Examples of Common TEX and LATEX commands			
Type:	To get:	Type:	To get:
`$\beta$`	β	`90\%`	90%
`regular text`	regular text	`a\&b`	a&b
`souffl\'es`	soufflés	`\copyright`	©
` ``Hi'' `	“Hi”	`Dr.\ Smith`	Dr. Smith
`{\em italics\/}`	*italics*	`{\bf Bold}`	**Bold**
`\$500.`	\$500.	`$P(X\ge x)$`	$P(X \ge x)$
`$p_1$`	p_1	`$\alpha/2$`	$\alpha/2$
`$\hat p$`	$\hat p$	`$x^2$`	x^2
`$\sqrt{x^2+y^2}$`	$\sqrt{x^2+y^2}$	`$f'(x)$`	$f'(x)$
`$\sum$`	$\sum$	`$2 \times 2$`	2×2
`$\sum_j$`	$\sum_j$	`$\chi^2_{(.05)}$`	$\chi^2_{(.05)}$
`$\sum^\infty$`	$\sum^\infty$	`$z_{\alpha/2}$`	$z_{\alpha/2}$
`$\sum^n_{i=0}$`	$\sum^n_{i=0}$	`$H_0: p=.5$`	$H_0: p = .5$
`$\int_a^b xdx$`	$\int_a^b xdx$	`$\{x\}$`	$\{x\}$
`$P(Y \in \cal L)$`	$P(Y \in \mathcal{L})$	`$\log f$`	$\log f$
`--- -- -`	— – -	`$\frac{n}{k}$`	$\frac{n}{k}$
`$y_1,\dots,y_n$`	$y_1, \dots, y_n$	`$(X'X)^{-1}$`	$(X'X)^{-1}$
`$\theta \in \Theta$`	$\theta \in \Theta$	`$(m+n)/2$`	$(m+n)/2$
`$\Gamma(n+1)=n!$`	$\Gamma(n+1) = n!$	`$\lim_{n\to\infty}$`	$\lim_{n\to\infty}$

Table 4.1: Example TEX and LATEX commands

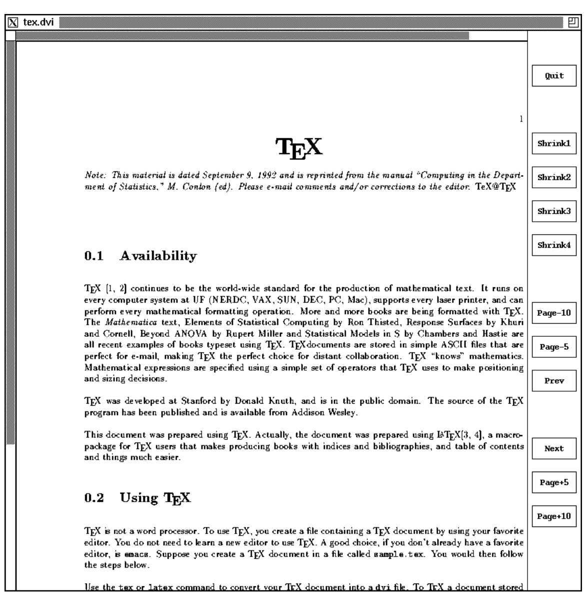

Figure 4.5: `xdvi`

and press one of the three mouse buttons. Each produces a different magnification.

`xdvi` is much faster than the PostScript^{tm} Previewers supplied with UNIX. You also don't need to create the PostScript^{tm} before you can preview. A drawback to `xdvi` is that you can't preview material that exists only as PostScript^{tm}, since it *is* a `dvi` viewer, not a PostScript^{tm} viewer. This means you can't preview portions of documents that use Adobe fonts. It also means you can't use `xdvi` to preview files that contain PostScript^{tm} graphic images.

See the man pages for a more detailed description of the features of `xdvi`. There are many command line options for changing default actions, setting colors, and much more.

4.2.5 Graphics in TEX documents

It is possible to include graphs stored in PostScript files into TEX and LATEX documents. The `epsf` macro provided with the `dvips` device driver is the piece of software that does the job. It can be used to include any PostScript graphic (from S, Mathematica, even home-made) into TEX documents.

The sequence of steps for including a graph into a document is:

1. Create a PostScript graphic and put it into a file using S, or Mathematica, for example. See notes about S below.

2. Create your TEX, or LATEX document.

3. After the `\begin{document}` command, (in LATEX), or near the beginning of your TEX file, enter

   ```
   \input{epsf}
   ```

4. At the point in your text where you want the graph to appear, include your graph using:

   ```
   \epsfysize=4truein
   \epsffile{mygraph.ps}
   ```

Note that the height of any graph included in your document can be altered (rescaled) at the time of inclusion into the text.

Once you run LATEX, or TEX and `dvips`, the resulting Postscript file for your document has the graph inside it, and can be previewed as a mixed text and graphics document using the `ghostview` previewer. `xdvi` is not equipped to preview documents with mixed text and graphics. If you try to use `xdvi` on such a document you will get a polite message that `\special` commands are not implemented, and it will leave a blank space in the preview screen where the graph will eventually appear in printed output.

Figure 4.6 contains a sample graph produced from S-plus^{tm}, using the `postscript()` device driver in S-plus^{tm}.

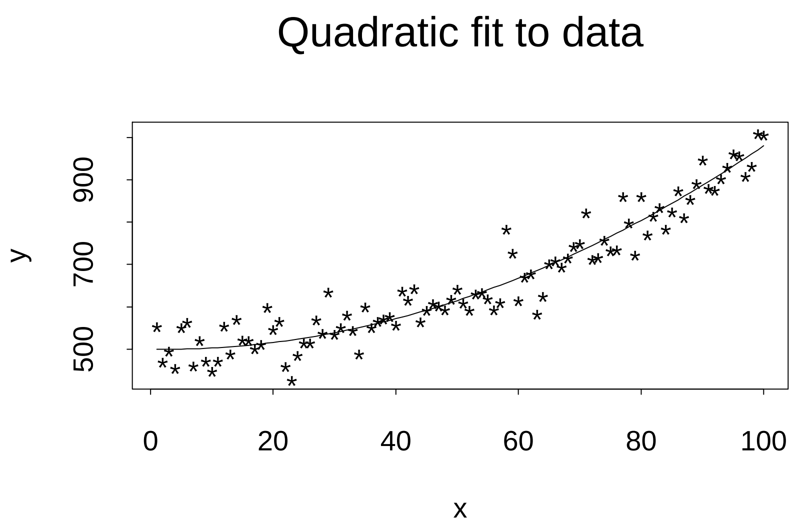

Figure 4.6: A sample S graph

There are several ways to create a PostScript file from an S graphic. A good way is to use the `postscript` device driver to generate your graphics into a file that you name. For example:

```
postscript(``b.ps'')
```

There are two good reasons to get the PostScript in this fashion: 1) the default orientation is portrait, not landscape. Landscape orientation would come out sideways in a document. 2) This driver puts a Bounding Box PostScript comment in the file. This is necessary for the TEX macro to work. If you know enough about Bounding Boxes to put one in yourself, you didn't need this note.

4.2.6 `detex`

`detex` is a filter to strip TEX commands (or LATEX) from a `.tex` file. It can be used to produce a text file for input to style or spell or other text-processing commands. For example, if you wanted to check the style of a file called `my.tex` you could use:

```
vega(mcentyre) 1% detex my.tex | style
```

See the manual page for details on its use.

4.2.7 BIBTEX

BIBTEX is a system for creating collections of bibliographic entries in a clear-text database which can automatically be extracted by LATEX to build a reference list for a particular document. LATEX formats the list according to the a specified bibliographic style, so documents can be changed from one style to another by rerunning LATEX on the document. BIBTEX also handles citation order and numbering.

Several bibliographic styles in addition to those described in the LATEX book are available for BIBTEX users:

`acm`	Association for Computing Machinery standards.
`theapa`	American Psychological Association standards.
`ieeetr`	IEEE Transactions format.
`jasa`	JASA format. (ugly, but faithful to the journal).
	by Brett Presnell, UF Dept. of Statistics.
`siam`	Society for Industrial and Applied Mathematics format.
`statmed`	Statistics in Medicine.

All BIBTEX styles are available to LATEX users without special commands. A complete and current list of styles can be obtained by using:

```
vega(mcentyre) 1% ls /home/tex314/inputs/*.bst
```

4.3 LATEX

This report demonstrates how to produce a technical document using LATEX. Some new macros and style files are provided for those who are not satisfied with certain aspects of LaTeX's stock styles. The production and inclusion of figures and tables, and the titlepage are demonstrated. Various aspects of producing both inline and displayed mathematics are explained, including vectors and matrices, theorems, and so forth. Examples of simple user macros are provided. Use of LaTeX's internal cross-referencing capabilities are demonstrated. The style is intentionally very casual and it is assumed that the reader has a working knowledge of a text editor such as *vi* or *Emacs*. Additionally, it may be helpful to have a copy of the `sample.tex` file provided with the standard implementation of LaTeX. It is located in /home/tex314/doc.

4.3.1 Introduction

This is a small document to give some idea of how one might write a paper using LaTeX. Many, but by no means all of the features of LaTeX are demonstrated, including labeling and referencing of equations, sections, tables, and figures. For more detail and for explanations you should consult the references. To fully understand this primer you should read the file /home/tex314/doc/primer.tex along with the the LaTeX'd output. For your convenience, the file (primer.tex) is located in Appendix C starting on Page 83.

In Section 4.3.2 you will see how to create an ASCII text file and process it by LaTeX. Also, viewing the output and routing it to a printer are discussed.

In Section 4.3.6, you will see how to obtain some of the common mathematical symbols and expressions, and how to typeset both inline and displayed equations. Equation numbering and labeling will also be demonstrated here. You will also see how to format, label, and reference theorems and their proofs.

In Section 4.3.5, we show how to insert, label, and reference tables and figures. Here we assume that any figures you wish to insert are in encapsulated PostScript format. Please note that some software produces plain (not encapsulated) postscript files as output, and using the *epsf* macro with these files may produce unexpected results. Another macro for use with non-encapsulated postscript files is *psfig*, which is available at many tex archives.

Some examples of bibliography entries are shown at the end. However, bibliographies are most conveniently handled through BibTeX. LaTeX can automatically generate a table of contents, list of figures, list of tables, and index. Although those also are not covered here, the references describe how to do handle these more advanced features.

Note that this sampler just provides a few simple examples. Please do not get the idea that this is all you need to know. You should have a copy of the LaTeX book, and you should read at least the first three chapters before you get too far along using LaTeX. If you really get interested and want to know more about TeX and how it works, the TeX book, by the creator of TeX, Don Knuth, is the final authority. However, for almost all purposes, the LaTeX manual will suffice.

4.3.2 Running LaTeX

The first thing you want to do when you use LaTeX is to become proficient in a text editor. Unlike many word processors, a LaTeX file is a standard ASCII text file with certain control codes embedded in it that LaTeX understands. Once you create your LaTeX file you need to process it using the `latex` command from the system prompt. For example, from the unix prompt you would type:

`latex` *filename*

followed by return, where filename is the root name (without the extension) of your LaTeX file. Your LaTeX file must end in `.tex`. After you process the LaTeX file, and assuming you received no error messages, you can view the LaTeX'd output on an X-terminal by typing the command

`xdvi` *filename*

followed by return. An `xterm` pop-up window appears. The commands available to you appear on the right of the screen. If you need to make changes to your file, quit the *xdvi* previewer, go back and edit the LaTeX file, re-process it, and then preview it again using *xdvi*. Alternatively, you can keep one window open for your text file, one for entering commands, and one for the *xdvi* previewer. This way you can make changes and run LaTeX on your file without exiting the text editor and re-invoking the *xdvi* previewer. When you want to print you need to convert the output file (*filename.dvi*) to a file format which can be understood by the printer. The `dvips` command converts the .dvi file into postscript format which can then be printed. To do this type

`dvips` *filename*

followed by return. The file *filename.ps* will be created and then can be printed using the `lpr` command:

`lpr` *filename.ps*

followed by return. For more information on running LATEX see the LATEX manual. You can also simply send your .dvi file to the printer, but it will envoke dvips for you and you will not be able to use the dvips options.

4.3.3 LATEX File Structure

As already stated, a LATEX file is simply an ASCII text file which contains certain formatting codes. Every LATEX file *must* contain a *preamble*, a section at the beginning of the file devoted to declaring style parameters such as font size, margins, etc... Also, every LATEX file must contain a *documentstyle* command and `\begin{document}` and `\end{document}` commands. The text between the `\begin{document}` and `\end{docucment}` commands is the body of the text, the text that you want to type. There are several document styles, such as article, report, and book; also, many *documentstyle* options are available, such as specifying font sizes and bibliographic data base options. See the references for further details.

4.3.4 Labeling and Referencing

You will notice throughout this file, that sections, equations, theorems, tables, and figures usually have a `\label` command associated with them. These are never required, but can be very useful for referencing the associated object or environment at other places in the document. Using a `\ref` command allows insertion of the number associated with an equation, say, without knowing a priori the number, and without having explicitly assigned the number. I already used this idea in Section 4.3.1. And if I wanted to say something about the Chebychev's inequality, I could refer to Theorem 1 or equation (4.2). This is especially useful when a document is being revised, as it saves searching through the document and correcting all references to an equation, section, or whatever.

4.3.5 Floating Bodies

4.3.5.1 Tables

It's not too hard to produce tables in LaTeX, and if you enclose them in a table environment you can give a caption and use the labeling and referencing capabilities to refer to them elsewhere in the document. It is also possible to produce a list of tables (and a list of figures) automatically. As an example, Table 4.2 is taken from Agresti (1990).

	Myocardial Infarction		
	Fatal Attack	Non-Fatal Attack	No Attack
Placebo	18	171	10,845
Aspirin	5	99	10,933

Table 4.2: Cross-Classification of Aspirin Use and Myocardial Infarction

4.3.5.2 Figures

Figures will generally be PostScript files produced by some other software, although LaTeX does have it's own somewhat primitive drawing capability. Anyway, using the encapsulated PostScript file macros (epsf.sty) provided with the dvips program, it's not too hard to do. For example, Figure 4.7 was produced by the following Splus commands:

```
postscript(file="scatter.ps",horizontal=F)
plot(lottery.number,lottery.payoff)
lines(lowess(lottery.number,lottery.payoff,f=.2))
graphics.off()
```

To include a graph using SAS/GRAPH© you would use the same LaTeX commands but the SAS commands would be something like the following:

```
filename gsasfile 'plot1.ps' ;


goptions gsflen=80 dev=epsf gaccess=gsasfile gprotocol=sasgpasc
```

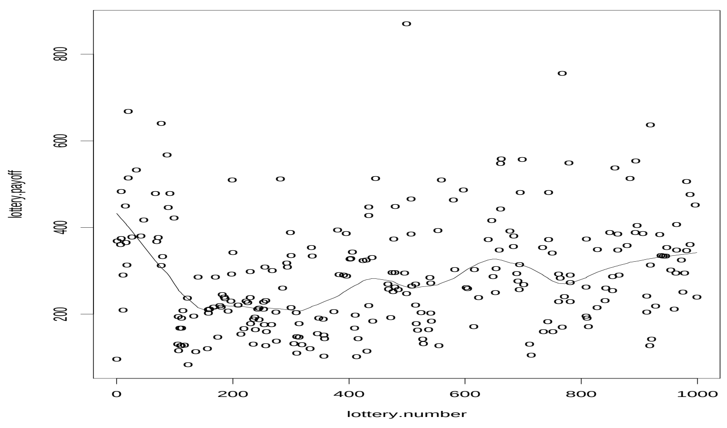

Figure 4.7: Scatter Plot taken from the Splus Book

```
noprompt handshake=none trantab=gtabcms gsfmode=replace gend='0d0a'x;
```

```
proc gplot ;
plot x*y ;
run ;
```

The graph will be output in encapsulated PostScript format and then can be included in the document exactly as shown above.

4.3.6 Typesetting Mathematics

4.3.6.1 General

It's easy to type inline equations in LaTeX. These should be surrounded in your LaTeX file either by $'s or by a `\( ...\)` pair. So, as you can see, $E(X) = \int_{-\infty}^{+\infty} xf(x)dx$ and $E(X) = \int_{-\infty}^{+\infty} xf(x)dx$ both produce the same thing.

Tables of many of the symbols available in LaTeX are given on pages 43–46 of the LaTeX book. A few things are worth mentioning here. A very common "mistake" seen in LaTeX documents from beginners is something like $M = min(X,Y)$. Notice the difference between this and $M = \min(X,Y)$. The latter is the correct form. Similarly, you should type $f(t) = A\cos(\omega t)$ as opposed to $f(t) = Acos(\omega t)$.

Another common question is how to type vectors. Most people are used to writing a letter with a $\sim$ underneath to represent vectors, but typeset mathematics rarely does this. Instead vectors and matrices are represented by bold-faced math characters. This is accomplished in LaTeX by using the `\boldmath` command. This leads to a construction that is long enough that it would be annoying to have to type it more than once, and since it will probably be used many times, it's a good idea to write a macro to produce various bold math symbols. In the macro file `my.mac`, there is a little macro that does this, so that I can type something like $\boldsymbol{Ax}$ to represent the product of the matrix $\boldsymbol{A}$ with the vector $\boldsymbol{x}$. Oh, it's also useful to know how to produce something like $\boldsymbol{x}' = (x_1, \ldots, x_d)$, and now you do. If you'll be using this kind of thing a lot in your paper, you might want to write a simple macro, so that you can just type something like $\boldsymbol{x}' = (x_1, \ldots, x_d)$.

4.3.6.2 Displayed Math

A displayed formula is one that is centered on a line by itself. For example, suppose you wanted to produce a binomial coefficient. Somehow this was left out of the LaTeX book, but it's provided by plain TeX, and is given in the TeX book. It's easy to write a binomial probability, just type:

$$p(k) = \binom{n}{k} p^k q^{n-k}.$$

For simple displayed equations that require no equation numbers, all you need to do is surround the equation with $$'s (double dollarsigns), though this is not mentioned in the LaTeX book. Thus for example, you can type:

$$E(X) = \int_{-\infty}^{+\infty} x f(x) dx.$$

If you want a numbered equation, you can use the equation environment, like so:

$$F(x) = \sum_{k=0}^{x} \frac{\lambda^k}{k!} e^{-\lambda} \tag{4.1}$$

Now I can refer to equation (4.1) anywhere in the document (even before the equation) using the `\ref` command.

You will notice that the limits on sums and integrals are aligned differently in display math and inline math. The reasons for this are obvious I think. For the same reasons, you should be careful about overusing `\frac`, especially when typing inline mathematics. For example, to write the last formula in line, you might consider typing $F(x) = \sum_{k=0}^{x} \lambda^k e^{-\lambda}/k!$, as opposed to $F(x) = \sum_{k=0}^{x} \frac{\lambda^k}{k!} e^{-\lambda}$.

Often you will wish to typeset a series of equations or inequalities with the equal signs or inequalities aligned. In LaTeX this is done by using the eqnarray and eqnarray* environments. The former numbers the equations, the latter doesn't. These environments are often used in writing a proof, as in the following, taken from Sen and Singer (1993):

Theorem 1 [Chebychev Inequality] *Let U be a non-negative r.v. with a finite mean $\mu = E(U)$. Then, for every $t > 0$,*

$$P\{U > t\mu\} \leq t^{-1}. \tag{4.2}$$

Proof: Note that

$$\begin{aligned} \mu = E(U) &= \int_0^{+\infty} udP(U \leq u) \\ &= \int_0^{t\mu} udP(U \leq u) + \int_{t\mu}^{+\infty} udP(U \leq u) \\ &\geq \int_{t\mu}^{+\infty} udP(U \leq u) \\ &\geq t\mu \int_{t\mu}^{\infty} dP(U \leq u) \\ &= t\mu P(U > t\mu) \end{aligned}$$

and, hence, dividing both sides through by $t\mu$, we get the desired result. $\square$

For matrices, arrays, and vectors, you use the array environment. As an example, consider:

$$A = LU = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 2 & 1 & 1 & \\ 3 & 2 & 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 & 1 & 2 & 1 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The LaTeX book gives more complex examples.

4.4 S-plus^{tm}

The new S language, is implemented on our system as S-plus^{tm}[5]. S-plus^{tm} is a full implementation of the new S language and conforms in

[5] S-plus^{tm} is a trademark of Statistical Sciences, Inc. who markets adaptations of the S system under license from AT&T.

every way to the language as described in the S book.

S is a modern system for data analysis and graphics. It has an extremely powerful, flexible and extensible programming language which can be used to construct graphs, manage data, do statistical analysis, explore new statistical techniques and much more.

4.4.1 Using S

To run S-plus^{tm}, type:

```
vega(mcentyre) 1% Splus
```

To start the graphics system on the Suns, type:

```
> motif()
```

To exit S-plus^{tm}, type:

```
> q()
```

It is convenient to have an editor window open while you use S. In this way, you can edit a function, write it to a file, for example `my.s`, click on the S session and type:

```
> source("my.s")
```

S will read the file and perform the S statements it finds there. It doesn't have to be a function definition — it could be any collection of S statements. Existing functions can be dumped to an ASCII file with the S function `dump` and then brought into an editor.

A second method for using an editor with S provides a very simple way to move back and forth — provided your function fits on one screen (many S functions are quite short since S is such a powerful language). Write your function. Then highlight the entire function using the mouse. Move to the S window and click left to copy the entire function into S. For help on highlighting text and using the mouse in the X Window environment, see chapter 2.

Printing graphics from S is very simple. To print graphs, just click on the Print Graph button in the S graphics window. A sample S-plus graphics window is shown in Figure 4.8.

To print text material produced by S, use the `sink` function to specify the destination of output, then run the functions that produce

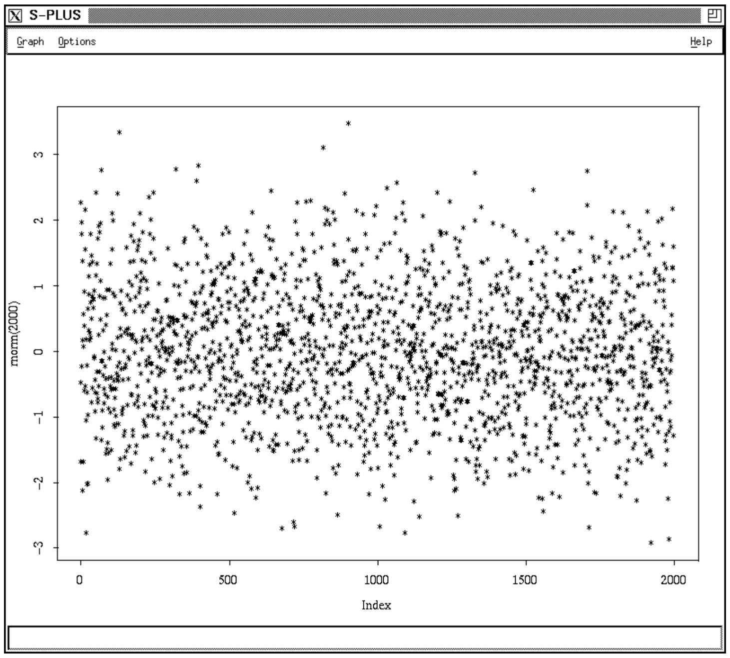

Figure 4.8: The S-plus^{tm} `motif` graphics window

the output, then restore printing to the screen using `sink("")`. Or you can use the UNIX `script` command to record the entire S session (both input and output). In either case, you produce a text file which can be printed with a standard printing command, such as `lpr`.

4.4.2 Help

S-plus^{tm} has two extensive on-line help systems. To get help with any function, type

```
> help("name")
```

S-plus^{tm} uses the `less` utility for displaying help information. See the man pages for details. When you issue an S help request, such as:

```
> help("ls")
```

S displays the help in the window it is using for the S dialog. To finish reading the help, press the `q` key (you are remined of this on the last line of the window when help appears).

S-plus^{tm} also has a graphical help system. Type:

```
> help.start()
```

to start the help system. The help system then has graphical buttons and text boxes for you to select subjects and have them pop-up on the screen (see Figure 4.9).

You can leave the help window on the screen while you use S and go to it whenever you want. The right hand side is used to select major categories, the left side to select individual help topics. Browsing the categories, you can find all the S-plus^{tm} functions associated with particular tasks. Figure 4.9 shows the result of clicking on "Data Manipulation" in the right panel. You can use the text field on the bottom of the window to enter specific names of functions you want help with.

4.4.3 Customizing

The starting position of the S-plus^{tm} graphics window can be customized by adding a line to your `.Xdefaults` file.

```
splus*geometry:    =450x450+150+150
```

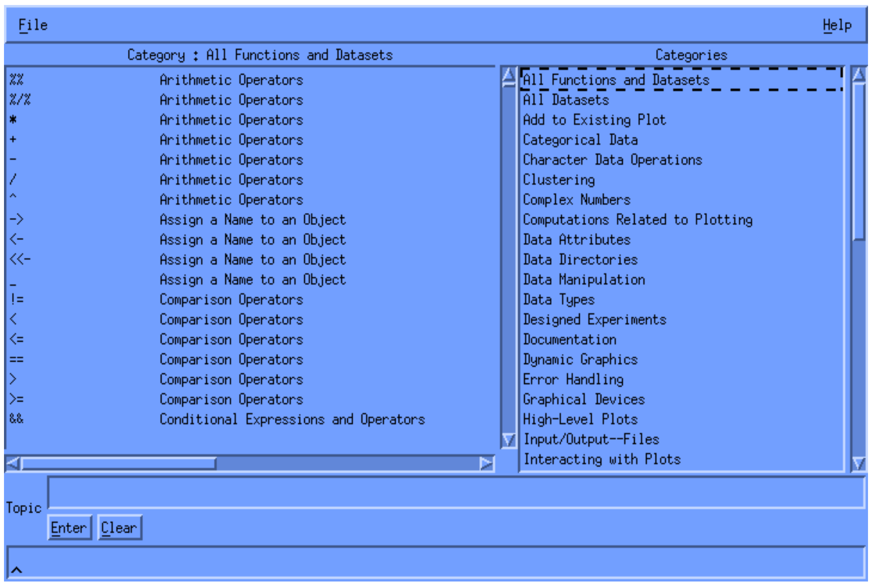

Figure 4.9: The S-plus^{tm} help window

creates a reasonably sized window (450 pixels square) located at pixel (150,150) from the upper left corner. Experiment to get a window position you like. Each time you modify `.Xdefaults` you must run

```
vega(mcentyre) 1% xrdb .Xdefaults
```

to have the `.Xdefaults` take effect, rather than logging out and logging back in.

4.4.4 Batch

S-plus^{tm} can easily be run in batch. This means that you can start an S-plus^{tm} "job" from the command line and it will create objects and a listing file in the background. You can set the batch job to start immediately — in which case it will run at a lower priority than what you are doing, so you won't notice it, or you can set it to start at a specific time, or at a time chosen by the system.

Follow the steps below to run an S-plus^{tm} batch job.

1. Create a file of the S-plus^{tm} commands you want to execute. Suppose you create a file called `in.s` containing your program.

2. Create a file with an editor, suppose you call it `run-S` that contains the single line:

   ```
   Splus BATCH in.s out.lis
   ```

   This command line instructs S-plus^{tm} to run in batch mode, to read its program statements from `in.s` and to write its output to `out.lis`. As always, any objects S-plus^{tm} creates are stored in the `./.Data` directory.

3. Make the file executable by typing:

   ```
   vega(mcentyre) 1% chmod +x run-S
   ```

4. Choose one of the two methods for batch submission:

   (a) To start immediately at a reduced priority, type:

   ```
   vega(mcentyre) 1% nice run-S &
   ```

   (b) To run at a specific time:

   ```
   vega(mcentyre) 1% at 2100 run-S
   ```

   where `2100` indicates 9:00pm. See `man at` for a description of this very powerful scheduling utility.

4.5 GLIM

GLIM, the Generalised Interactive Modeling System is a widely available, rather famous system for fitting a wide class of linear models. The models are characterized by expressing a relationship between the explanatory variables and a function which links the expected values of the data with the explanatory variables.

GLIM is a command driven program which was first introduced in 1972. It has a macro language, and a collection of pre-written macros is available (see below). We are running version 3.77 of GLIM, a licensed product of the Numerical Algorithms Group.

To start GLIM, type:

```
vega(mcentyre) 1% glim
```

To exit GLIM, type:

```
? $stop
```

The dollar sign ($) is very important. All GLIM commands (also known as directives) begin with a dollar sign. GLIM can run in batch mode. To run GLIM in the background, create a file of GLIM directives (say `my.glm`) and use:

```
vega(mcentyre) 1% glim < my.glm > my.out &
```

The `&` puts any command in the background, freeing your terminal while the work is being done. When the work is done, you will have a file called `my.out` containing the GLIM session.

By putting options on the command line to GLIM you can perform several useful tasks. The option `-info` displays the GLIM manual pages. For example:

```
vega(mcentyre) 1% glim -info
```

When you are finished reading the manual pages, the program will end. No GLIM session is started.

The option `-maclib` can be used to print a copy of the GLIM macro library documentation. The option `-help` is used to print a copy of the GLIM help files.

GLIM sessions can be started with arguments on the command line. Arguments tell GLIM the names of files to be used for various purposes during the GLIM session. Each argument is of the form *keyword=filename*, where *keyword* is one of the following.

pic Primary input.

poc Primary output.

sic Secondary input.

soc Secondary output.

pdc The `$DUMP/$RESTORE` file.

trc The transcript file (`GLIM.LOG`)

plc The macro library file to be used in place of the standard one.

GLIM users should note that there is an excellent function for fitting generalized linear models in the newest version of S. See section 4.4. The function is called `glm`.

4.6 The C Programming Language and Derivitives

The C programming language is available as a standard component of UNIX. More workstation and PC software is written in C than in any other language. In fact, Unix is written in C.

To compile a C language program, first create a source file with an editor, having a file name extension of `.c`, such as `hello.c`. Then use:

```
vega(mcentyre) 1% cc hello.c
```

If your program compiles without errors, a file named `a.out` will be created. This file is the executable image of your program. You will probably want to rename it to `hello` by typing:

```
vega(mcentyre) 2% mv a.out hello
```

You can do this in one step by using the `-o` command line option:

```
vega(mcentyre) 3% cc -o hello hello.c
```

In this case, no `a.out` file is produced. Instead, the output is stored in the file called `hello`. The output is a non-printable executable image.

Then you can run your program by typing its name:

```
vega(mcentyre) 4% hello
```

Standard input and standard output are directed to the screen by default, and redirection is on by default. See the man pages for details on these topics.

To compile a program which uses any math routines (even such simple things as square root), use the `-lm` command line switch to the linker, as in:

```
vega(mcentyre) 5% cc -o hello hello.c -lm
```

4.6.1 GNU C

We also have another C compiler, GNU C. GNU C generates large code that runs slower than the default C compilers. However, it is occasionally useful to debug programs by passing source code through more than one compiler. To run GNU C, follow the steps above, but instead of `cc` use `gcc`. Everything else stays the same.

4.6.2 C++

C++ is a recent extension to C that adds the concepts of object oriented programming (OOP). C++ is a superset of C, so everything you learn about C carries over to C++. C++ provides a means for defining classes of objects and operators for those classes.

To run C++, create a C++ program in a file with a file type of `.cc`. Then follow the instructions for `cc` above, but using `c++` rather than `cc`.

4.6.3 GNU C++

We have also installed GNU C++. You run it by using `g++` rather than `c++`. See the man pages for the gory details on differences between the two compilers (They are small, few, and far between, but if you're interested).

4.7 Make

A very powerful and useful utility for writing and managing any medium to large project is the `make` utility. `make` determines the parts of your program that need to be recompiled and linked to bring your program up-to-date. `make` is unnecessary for programs that exist in a single source file, but for larger projects with functions in many files, `make` is a real time-saver. Most software that we receive and install from the Internet uses the `make` utility to automate the executable image construction process. Check `man make` at a unix prompt for a complete description of this utility.

4.8 Pascal

You will create programs using your favorite editor and give them an extension of `.p`, as in `first.p`. A sample Pascal program is shown below.

```
program hi;
begin
  writeln('Hi!');
end.
```

You compile your program by typing:

```
vega(mcentyre) 1% pc hi.p
```

If your program compiles without errors, a file named `a.out` is created. This file is the executable image of your program. You will probably want to rename it to `first` by typing:

```
vega(mcentyre) 1% mv a.out hi
```

You can do this in one step using the `-o` compiler command line option:

```
vega(mcentyre) 1% pc -o hi hi.p
```

In this case, no file `a.out` is produced. Instead, the output is stored in a file named `hi`. The output file is a non-printable executable image. You can run your program by typing its name:

```
vega(mcentyre) 1% hi
```

The output is displayed on the terminal:

```
Hi!
```

If your program uses any mathematical functions, even simple ones like `sin` and `sqrt`, you must add the `-lm` option to the command line to include the math libraries during the linking phase:

```
vega(mcentyre) 1% pc -o first first.p -lm
```

Pascal is not currently gaining in popularity. It had its hey-days in the 1980's when it was the darling of Computer Science and the PC world was dominated by Turbo Pascal. Since then, Pascal has continued to be a good choice for novice programmers being introduced to the wonders of pointers and dynamic data structures, but the real work is being done in object oriented languages like C++ and special function languages like S-plus^{tm}, ILM, and GLIM.

4.9 FORTRAN

You will create programs using your favorite editor and give them an extension of `.f`, as in `first.f`.

You compile your program by typing:

```
vega(mcentyre) 1% f77 first.f
```

If your program compiles without errors, a file named `a.out` is created. This file is the executable image of your program. You will probably want to rename it to `first` by typing:

```
vega(mcentyre) 1% mv a.out first
```

You can do this in one step using the `-o` compiler command line option:

```
vega(mcentyre) 1% f77 -o first first.f
```

In this case, no file `a.out` is produced. Instead the output is stored in a file named `first`. The output file is a non-printable executable image. You can run your program by typing its name:

```
vega(mcentyre) 1% first
```

If your program uses any mathematical functions, even simple ones like `sin` and `sqrt`, you must add the `-lm` option to the command line to include the math libraries during the linking phase:

```
vega(mcentyre) 1% f77 -o first first.f -lm
```

You really should not use FORTRAN 77 unless you have to. Coding FORTRAN programs is very unproductive. You code hundreds of lines of FORTRAN to get the work done you could do in 1 line of S-plus^{tm}. The workstations are here to make you more productive. A much better way to spend your time is to read the S book and try examples. You'll be surprised how easily you reach a level of proficiency which will exceed your ability to make things happen in FORTRAN. But if you really want to see your productivity rise, learn C or C++, or IML. Then the sparks will fly!

A collection of software has been established just for statisticians. `StatLib` is run by folks at Carnegie-Mellon University and contains many useful pieces of statistical software, in FORTRAN, S, C, as well as other data sets. To get started with `StatLib` send a one-line e-mail message to `statlib@lib.stat.cmu.edu` with a message of `send index`. You will receive, by return e-mail, a description of `statlib` holdings and how to access them.

A larger collection of mathematical software is available on the net and maintained by Bell Labs. The `NetLib` is similar to the `statlib`, but contains many more routines. To get started with `NetLib` send a one-line e-mail message to `netlib@research.att.com` with a message of `send index`. You will receive, by return e-mail, a description of `netlib` holdings and how to access them.

Chapter 5

Information Resources

A networked computer system provides a unique opportunity to access information resources — from within and across the department, the University, the nation, and the world. This chapter includes information on a variety of resources.

5.1 The Internet

The Internet contains many sites offering special services, collections, documents, software and other resources. A good introduction to the Internet can be found in Zen and the Art of the Internet available at your local book store.

5.1.1 Statistical/Mathematical resources

Statistical analysis

(FTP) Information on UICSTAT SAS archive HOST: uicvm.uic.edu (128.248.2.50) DIRECTORY: uicstat FILES: uicstat.readme uicstat.filelist

(MAIL) Information on UICSTAT SAS archive MAIL TO: listserv@uicvm COMMANDS: get uicstat.readme get uicstat.filelist

(MAIL) Information about searching SAS Usage Notes and Zaps MAIL TO: listserv@uicvm COMMAND: get sasusage listing

(MAIL) SAS (MVS) programs for processing STF census data MAIL TO: listserv@umslvma COMMANDS: get stf1prof package get stf901c package

(FTP) Information on UIUC mathematical software archive HOST: ux1.cso.uiuc.edu (128.174.5.59) DIRECTORY: pub/math FILE: README

(FTP) MATHLIB software HOST: emx.utexas.edu (128.83.1.33) DIRECTORY: pub/mathlib

(MAIL) NetLib numerical analysis software server MAIL TO: netlib@ornl.gov COMMAND: send index

(MAIL) StatLib Statistical analysis software server MAIL TO: statlib@lib.stat.cmu.edu COMMAND: send index

(FTP) StatLib Statistical Analysis Software archive HOST: lib.stat.cmu.edu (128.2.241.142) USER: statlib

5.1.2 General Internet resources

The references below indicate the sources of many different services on the Internet.

-Archie telnet archie.mcgill.ca or 132.206.2.3 (Can./USA) telnet archie.funet.fi or 128.214.6.100 (Finland/Europe) telnet archie.au or 128.184.1.4 (Aussie/NZ) telnet cs.huji.ac.il or 132.65.6.5 (Israel) telnet archie.doc.ic.ac.uk or 146.169.3.7 (UK/Europe) telnet archie.sura.net or 128.167.254.179 telnet archie.unl.edu (Login: archie, pasword: archie1) telnet archie.ans.net or 147.225.1.2 (North America) offers: Internet anonymous FTP database. (Login: archie)

-Archie Mail Servers mail archie@¡INSERT ONE OF ABOVE ADDRESSES HERE¿ Subject: help Offers: alternative Archie access to those w/o ftp or telnet.

-CARL telnet pac.carl.org or 192.54.81.128 offers: Online database, book reviews, magazine fax delivery service.

-Cheeseplant's house telnet orchid.csv.warwick.ac.uk 2001 or 137.205.192.5 offers: Online chat service in a very unique format.

-Cleveland Freenet telnet freenet-in-a.cwru.edu or 129.22.8.82 offers: USA Today Headline News, Sports, etc...

-C64 Archive Server mail twtick@corral.uwyo.edu Subject: Mail-Archive-Request Body-of-letter: help (hit return) end

-Dante Project telnet library.dartmouth.edu or 129.170.16.11 offers: Divine Comedy and reviews. (Login: connect dante)

-Distance Educat. Data telnet sun.nsf.ac.uk or telnet 128.86.8.7 (Login: janet Hostname: uk.ac.open.acs.vax Username: icdl)

-Document Site ftp 128.32.184.254 offers: Many docs, including 5 puritytests, the Bible, Dec. of Ind, lyrics..

-Earthquake Info. finger quake@geophys.washington.edu offers: Recent quake info (location, time, magnitude, etc.)

-FTP Mail mail ftpmail@decwrl.dec.com Subject:(hit return) Body-of-letter: help (return) quit Offers:ftp via email

-Genetics Bank mail gene-server@bchs.uh.edu Subject: help Offers: genetic database accessible via email.

-Geographic Server telnet martini.eecs.umich.edu 3000 or 141.212.100.9 3000 offers: Info by city or area code (Population, Lat./Long., Elevation, etc).

-Gopher telnet consultant.micro.umn.edu Login: gopher Offers: access to many interesting features.

-Graf-Bib mail graf-bib@decwrl.dec.com Subject: help Body-of-letter: send index Offers: graphics bibliography.

-Ham Radio Callbook telnet marvin.cs.buffalo.edu 2000 or 128.205.32.4 2000 offers: National ham radio call-sign callbook.

-INFO - Rutgers CWIS telnet hangout.rutgers.edu 98 or 128.6.26.25 98 offers: Dictionary, thesaurus, CIA world fact book, quotations database.

-Internet Resource Guide ftp nnsc.nsf.net offers: compressed/tar'd list of net resources in /resource-guide.txt.tar.Z

-IRC Telnet Client telnet bradenville.andrew.cmu.edu or 128.2.54.2 offers: Internet Relay Chat access, like a CB on the computer.

-Library of Congress telnet dra.com or 192.65.218.43 offers: COPY of Library of Congress (Assumes terminal is emulating a vt100).

-List of Lists ftp ftp.nisc.sri.com or ftp 192.33.33.22 mail mlol-request@wariat.nshore.ncoast.org offers: List of interest groups/email lists in /netinfo/interest-groups.

-Lyric Server ftp cs.uwp.edu offers: Lyrics (/pub/music/lyrics/files) in text files for anonymous ftp.

-Mail Srver/Usr Lookup mail mail-server@pit-manager.mit.edu usage: in body of mail message: send usenet-addresses/[name searching for]

-Melvyl telnet melvyl.ucop.edu or 31.1.0.1 offers: access to various libraries. Type 'other' at prompt to see others.

-NASA Headline News finger nasanews@space.mit.edu offers: Daily press releases from NASA.

-NASA SpaceLink telnet spacelink.msfc.nasa.gov or 128.158.13.250 offers: Latest NASA news, including shuttle launches and satellite updates.

-NED telnet ipac.caltech.edu or telnet 131.215.139.35 offers: NASA Extragalactic Database. (Login: ned)

-NetLib mail netlib@ornl.gov Subject:(hit return) Body-of-letter: send index Offers: Math. software.

-Oceanic Info. Center telnet delocn.udel.edu or telnet 128.175.24.1 (Login: info)

-Oracle mail oracle@iuvax.cs.indiana.edu offers: The Usenet Oracle! Mail with subject as "help" for more info.

-PENpages telnet psupen.psu.edu or telnet 128.118.36.5 offers: Agricultural info (livestock reports, etc.) (Login: PNOTPA)

-SDDAS telnet espsun.space.swri.edu 540 offers: SW Research Data Display & Analysis Center. Or telnet 129.162.150.99

-SERVICES telnet wugate.wustl.edu or 128.252.120.1 offers: Access to nearly every listed service! (Login: luis)

-Software Server telnet charlie.secs.csun.edu 5742 or 130.166.2.150 5742 offers: Similar to Archie. Type help for a list of commands.

-StatLib Server mail statlib@lib.stat.cmu.edu Mail with line: send index. Offers: Prgms, Datasets, etc. for statisticians.

-STIS telnet stis.nsf.gov or 128.150.195.40 offers: Science & Technology Information System. (Login: public)

-Supreme Court Rulings ftp ftp.cwru.edu offers: ASCII files of Supreme Court rulings in directory /hermes

-Usenet News MailServr mail [newsgroup]@ucbvax.berkeley.edu Allows you to post to a Usenet newsgroup via email. Useful if you have read- only access to Usenet news. Note: .'s become -'s Ex. alt.test -> alt-test

-UNC BBS telnet samba.acs.unc.edu or 128.109.157.30 offers: Access to Library of Congress and nationwide libraries (Login: bbs)

-WAIStation telnet quake.think.com or 192.31.181.1 offers: Wide Area Info. Service. (Login: wais) FTP think.com for more info.

-Weather Service telnet madlab.sprl.umich.edu 3000 or 141.212.196.79 3000 offers: City/State forecasts, ski conditions, earthquake reports, etc.

-World-Wide Web telnet info.cern.ch or telnet 128.141.201.74 offers: Info. service with access to various documents, lists, and services.

Appendix A

Comming Attractions

This Appendix is here to list the planned improvements for future versions of the guide. If you have an idea for a chapter or topic, please consult this section first. Ideas can be submitted to me directly, but I would prefer to have them in written format through email - I dont want to lose any of these ideas. The address to submit comments and suggestions to is biosguide@biostat.sph.unc.edu. You will be credited in the next version of the guide if your suggestion is used or if you supply text for a chapter or section. My only requirement is that all submissions become property of the guide - meaning I can edit and distribute them without penalty.

1. Chapter on DOS.
2. Chapter on Microsoft Windows.
3. Chapter on IBM OS/2.
4. Chapter on advanced topics with sections for ftp, X Windows on the PC, displaying X Windows clients from other hosts, writing Unix C-shell scripts.
5. Chapter on Novell Netware and networks.
6. Chapter on computing in a networked environment - how to get the most done without walking all over other users.

Appendix B

Glossary

Files

.cshrc The .cshrc file is the configuration file used by Unix to set up aliases and environment defaults. We provide a default .cshrc which calls a system wide Cshrc file to set up standard aliases and environmental variables. You can override any of these by following the directions contained in your .cshrc. This file is run each time you start a new c shell.

.login The .login file runs when you log in and sets your terminal characteristics and other environmental variables.

.logout The .logout file runs when you log out and does any house cleaning you specify. It, like the .login file, calls a system wide default.

.Xresources The .Xresources file defines which resources (colors, fonts, title bars) your X Windows will have and how they will appear. It is used for defaults for various applications. If no default is listed for a particular application, a system default Xresources file is used.

.xsession The .xsession file is the equivilent to the autoexec.bat on a PC for X Windows. It starts all the X applications you would like on your root window. The default .xsession file provided to all users contains comments to help them should they wish to make changes. All applications added to the .xsession file should end in an ampersand (&) and be placed before the call to twm (twm, by the way, should NOT have an ampersand after it).

Terms

anonymous ftp An access method whereby users without accounts can copy software put in predesignated places. Anonymous ftp is one of the principle methods for obtaining public domain UNIX software. Our system uses many pieces of software obtained via anonymous ftp. In addition, we support anonymous ftp for external access of material on our systems, that is, we are an archive site (see below).

application A program which a user might run to perform a particular kind of work. For example, a word processor is an application program.

architecture The arrangement of devices in a computer network (network architecture) or a computer's central processing unit (CPU architecture).

archive sites Sites which permit anonymous ftp access to their holdings.

ASCII American Standard Code for Information Interchange. A numbering system for symbols that defines what numeric code in the range 0–127 will be used to represent each symbol in a character set. The ASCII codes are readily available.

ASCII file A file in which each character is one of the ASCII characters. In DOS, each line is terminated by a carriage return (ASCII character 13) and a line feed (ASCII character 10). In UNIX, ASCII files have lines terminated only with a line feed. Thus you can have a carriage return (line wrap) without ending a line. Lines can be as long as you want.

backbone The central thread of a network. Used to describe the section of a network that all other parts are dependent upon.

backup A process of storing information on a daily basis to prevent loss in the case of catastrophy. A backup procedure consists of hardware (tape drive and tapes), software (the tape archive utility — tar), communications (the network) and people (the system manager. The backup is run by an automatic process each night so that all user files are stored safely on tape.

BITNET An international network of research computers organized to provide electronic mail, file transfer and remote login across the entire network. BITNET is an acronym for "Because It's Time" Network.

C A programming language developed at Bell Labs in the late 60's. C is intended to be suitable for system programming and numerical computing. C is the most popular programming language today for software development.

C++ An object oriented extension of the C programming language. C++ can be used to create powerful software objects that can be used in higher-level programming.

client A computer on a network that uses files and programs stored on other systems (servers) on the network.

CPU Central Processing Unit. The part of a computer which performs instructions. The workstations use RISC CPUs.

device Any piece of equipment connected to a network or computer. Usually reserved for storage devices such as disks and tapes.

disk Hard disk storage. A machine for storing large amounts of information.

diskette Small removable disk storage. Diskettes are not often used in workstation computing.

ed A very old-fashioned primitive editor. Stay away from this. There are far better choices. This is the UNIX equivalent of edlin on the PC.

Emacs A very powerful editing, programming development environment for UNIX. Distributed at no charge from the GNU group at the Free Software Foundation at MIT.

e-mail Electronic mail. Mail which can be sent anywhere on the Internet and received without interrupting the recipient. A good mail system will notify the recipient that mail has arrived.

ethernet A cabling system and low-level protocol for networking.

ftp File transfer protocol. A standard definition of the arrangement of data for sending and receiving files across a network. ftp is implemented by telnet.

Free Software Foundation A grant sponsored group at MIT which develops and distributes free software for UNIX operating systems. The Free Software Foundation has developed such products as Emacs, X-windows, and a C++ compiler.

FORTRAN 77 A programming language developed in the late 1950's for scientific computing. Used primarily for numerically intensive computations. Not well-suited for modern software development.

GLIM Generalized Linear Models. Software to fit and estimate the generalized linear model.

GNU A group of programmers at the Free Software Foundation at MIT. GNU stands for GNU's Not UNIX. The G in GNU is *not* silent. We use GNU Emacs.

Gopher A protocol for distributed document index and retrieval across the Internet. Machines which run Gopher can make documents available to users on remote machines using a simple to use program called gopher. The gopher program accesses information on cooperating computers without the user signing on, or even needing to be aware of where the information is begin served from.

GUI Graphical User Interface. A collection of windows, scroll bars, buttons and other graphics objects and that define a way in which a user can interact with a computer.

host A computer on a network that provides computational services and boot procedures to client systems and terminals.

icon A small symbol which represents a window containing an application. Icons appear in the icon box for closed windows, and in the upper left-hand corner of each open window.

IMSL The International Mathematical and Statistical Libraries — a collection of several hundred functions and subroutines written in FORTRAN.

Internet A national network of research computers organized to permit electronic mail, file access and remote login across the country.

Lisp A computer language known for its very flexible expression syntax. Popular with artificial intelligence researchers due to its inherent ability to interchange data and code dynamically.

makefile A set of commands that instructs the UNIX make system how to compile, link and install a piece of UNIX software.

mail Software to facilitate the management of electronic mail. Messages can be composed, sent, forwarded, carbon copied, and mailed to all users on a list. Received messages can be read, answered, forwarded, filed and organized into folders.

man pages A collection of short documents available on-line on UNIX systems that describe the commands available on the system. Accessed via the man command.

memory Synonymous with RAM memory. Silicon based memory chips. The term memory is never used to refer to disk storage.

mouse A small hand-held device used to move a cursor on the workstation screen. A workstation mouse has three buttons — although the left one is the most commonly used.

multi-processing The ability of a computer operating system to perform several tasks simultaneously. UNIX is a multi-processing system. DOS is not.

NCSA The National Center for Supercomputing Activities. An NSF funded center which develops and distributes public domain network access tools. In particular, NCSA developed telnet and ftp for the PC and for the Macintosh.

NetLib An archive site for mathematical software.

network A collection of hardware and software components that enable workstations to share files and programs, as well as permitting remote sign-on and access to shared devices. The hardware of a network consists of a wire and interface boards. The wire we use is called thin-wire Ethernet. The interface boards are standard equipment for workstations, optional in PCs and Macs.

Pascal A programming language useful for procedure oriented application programming.

password A word assigned by a user using the **passwd** command that prevents other people from using a given username. A password must be at least 6 characters long.

PC Personal computer. Any of the family of IBM- compatible computers, including the XT, AT, and x86 class machines.

Perl A very powerful, very flexible, fully capable programming language for processing files. Perl can be used to write scripts and is portable across SUN, DEC and the PC.

PostScript^{tm} PostScript^{tm} is a page description language and is a trademark of Adobe Systems, Inc. A PostScript^{tm}printer is a printer with a built-in CPU capable of receiving PostScript commands and preparing graphical output from them. S-plus^{tm}, TEX, and SAS all produce PostScript^{tm}. PostScript^{tm}files are ASCII files.

RAM Random Access Memory. The computer chips that make up the memory that can contain user and system programs. Workstations typically have 8 or more megabytes of RAM.

RISC Reduced Instruction Set Computer.

S A powerful, interactive, graphic, statistical programming environment developed at Bell Labs. As one reviewer said "If you have UNIX and you don't have S, get S. If you don't have UNIX, get UNIX, then get S." Old-S has been around for 10 years. New S is 2 years old. We have new S.

S-plus^{tm} An implementation of new S for workstations. Produced and licensed by Statistical Sciences, Inc. of Seattle, Washington under a distribution agreement with AT&T.

SAS The Statistical Analysis System. One of the largest and most common of the conventional statistics packages.

sed The stream editor. A batch version of ed. Same comments here as per ed. But used and useful in many shell scripts.

server A system on a network that provides files and programs to other systems (clients) on the network.

shell The program that interprets your UNIX commands and decides how to execute them. There are several shells available in the UNIX world, and our workstations have two of them — the Bourne shell and the C-shell (cute). The Bourne shell is denoted sh and the C-shell is denoted csh.

shell script A UNIX command file written for a particular shell. While similar, UNIX shells are not identical, so scripts written for one may not be compatible with the other.

SPARC Scalable Processor Architecture with RISC CPU. SUN's trademark name for its line of RISC based processors.

SPARCStation SUN's trademark name for its line of UNIX workstations.

StatLib An archive site for statistical software and data.

SUN A company which markets a line of UNIX workstations. SUN markets only UNIX workstations.

SunOS SunOS is the SUN version of UNIX.

superuser A special account on each workstation which has powers to read and write any file on the system. The superuser account is used by the Unix system manager to install software, fix problems and provide backups.

TCP/IP A communication protocol for the Ethernet that is used by the workstations to transfer information across the wires. TCP/IP is used across the Internet to communicate between campuses.

telnet Software developed at NCSA to enable network users to sign on to remote hosts. Incorporates ftp for file transfer between network computers.

TeX The world-wide standard for mathematical text formatting, TeX for UNIX is in the public domain. TeX is official software of the American Mathematical Society.

Unix A multi-user, multi-processing operating system developed in the early 1970's at Bell Labs. It has grown in popularity over 20 years of evolution to become the premier operating system for workstation computing.

UNIX News Also known as Usenet News. A very large collection of open forums (over 3000) where Internet users can post questions, comments, and information and get feedback from other users.

username To use the workstations, you must have a username created by the system manager which authorizes use on a particular workstation. A username generally looks like your own name, and is the name that you will be known by in mail messages. A popular choice for a username is your first initial followed by your last name. Your username must be 8 characters long or less.

VAX Digital Equipment Corporation's brand-name for its mini-computer architecture introduced in the mid 1970's. VAX computing has dominated University minicomputing for 15 years.

vi The universally available character oriented UNIX editor with cursor control. Does not know about the mouse. A little clumsy, and unusual for PC users.

VT220 A character-based ASCII terminal. 80 by 24 display. No graphics.

window A rectangle on the screen in which an application runs. There can be many windows on the screen at a single time. Windows can be moved, resized, closed (changed to icons) and opened with very simple mouse commands. Having multiple windows on the screen makes doing several things at one time very natural.

workstation A powerful, multi-tasking, multi-user, graphically oriented computer designed to facilitate the computing needs of research and technical users by working in a network of connected machines.

X11 The current version of the X-Windows system. The current release, X11R5, is used on our machines.

X Windows A software system that performs standardized windowing operations on a large variety of UNIX computers. X Windows was developed by the Free Software Foundation at MIT.

Appendix C

LaTeX Reference

```
This report demonstrates how to produce a technical document
using \LaTeX.  Some new macros and style files are provided
for those who are not satisfied with certain aspects of
\LaTeX's stock styles.  The production and inclusion of
figures and tables, and the titlepage are demonstrated.
Various aspects of producing both inline and displayed
mathematics are explained, including vectors and matrices,
theorems, and so forth.  Examples of simple user macros are
provided.  Use of \LaTeX's internal cross-referencing
capabilities are demonstrated.  The style is intentionally
very casual and it is assumed that the reader has a working
knowledge of a text editor such as {\em vi} or {\em Emacs}.
Additionally, it may be helpful to have a copy of the
{\tt sample.tex} file provided with the standard
implementation of \LaTeX.  It is located in
/home/tex314/doc.


\subsection{Introduction}
\label{sec:intro_latex}
This is a small document to give some idea of how one might
write a paper using \LaTeX.  Many, but by no means all of
the features of \LaTeX\ are demonstrated, including labeling
and referencing of equations, sections, tables, and figures.
For more detail and for explanations you should consult the
references.  To fully understand this primer you should read
```

```
the file /home/tex314/doc/primer.tex along with the the
\LaTeX'd output.  For your convenience, this file is located
in Appendix~\ref{ch:latexref} starting on
page~\pageref{ch:latexref}.


In section~\ref{sec:running} you will see how to create an
ASCII text file and process it by \LaTeX.  Also, viewing
the output and routing it to a printer are discussed.


In Section~\ref{sec:math}, you will see how to obtain some of
the common mathematical symbols and expressions, and how to
typeset both inline and displayed equations.  Equation
numbering and labeling will also be demonstrated here.  You
will also see how to format, label, and reference theorems
and their proofs.


In Section~\ref{sec:floats}, we show how to insert, label,
and reference tables and figures.  Here we assume that any
figures you wish to insert are in encapsulated PostScript
format.  Please note that some software produces plain (not
encapsulated) postscript files as output, and using the
{\em epsf} macro with these files may produce unexpected
results.  Another macro for use with non-encapsulated
postscript files is {\em psfig}, which is available at many
tex archives.


Some examples of bibliography entries are shown at the end.
However, bibliographies are most conveniently handled
through \BT\index{BiBTeX@\BT}.  \LaTeX\ can automatically
generate a table of contents, list of figures, list of
tables, and index.  Although those also are not covered here,
the references describe how to do handle these more
advanced features.


Note that this sampler just provides a few simple examples.
```

```
Please do not get the idea that this is all you need to know.
You should have a copy of the \LaTeX\ book, and you should
read at least the first three chapters before you get too
far along using \LaTeX.  If you really get interested and
want to know more about \TeX\ and how it works, the \TeX
book, by the creator of \TeX, Don Knuth, is the final
authority.  However, for almost all purposes, the \LaTeX\
manual will suffice.


\subsection{Running \LaTeX}
\label{sec:running}


The first thing you want to do when you use \LaTeX\ is to
become proficient in a text editor.  Unlike many word
processors, a \LaTeX\ file  is a standard ASCII text file
with certain control codes embedded  in it that \LaTeX\
understands.  Once you create your \LaTeX\ file you need to
process it using the {\tt latex} command from the system
prompt.  For example, from the unix prompt you would type:
$$
{\tt latex \;}{\em filename}
$$
followed by return, where filename is the root name (without
the extension) of your \LaTeX\ file.  Your \LaTeX\ file must
end in {\tt .tex}.  After you process the \LaTeX\ file, and
assuming you received no error messages, you can view the
\LaTeX 'd output on an X-terminal by typing the
command\index{xdvi}\index{LaTeX@\LaTeX!previewing}
$$
{\tt xdvi \;}{\em filename}
$$
followed by return.  An {\tt xterm} pop-up window appears.
The commands available to you appear on the right of the
screen.  If you need to make changes to your file, quit the
{\em xdvi} previewer, go back and edit the \LaTeX\ file,
re-process it, and then preview it again using {\em xdvi}.
Alternatively, you can keep one window open for your text
file, one for entering commands, and one for the {\em xdvi}
```

```
previewer.  This way you can make changes and run \LaTeX\ on
your file without exiting the text editor and re-invoking
the {\em xdvi} previewer.  When you want to print you need
to convert the output file ({\em filename.dvi}) to a file
format which can be understood by the printer.  The {\tt
dvips}\index{dvips}\index{LaTeX@\LaTeX!converting to \ps}
command converts the .dvi file into postscript format which
can then be printed.  To do this type
$$
{\tt dvips \;}{\em filename}
$$
followed by return.  The file {\em filename.ps} will be
created and then can be printed using the {\tt lpr} command:
$$
{\tt lpr \;}{\em filename.ps}
$$
followed by return.  For more information on running \LaTeX\
see the \LaTeX\ manual.  You can also simply send your .dvi
file to the printer, but it will envoke dvips for you and
you will not be able to use the dvips options.


\subsection{\LaTeX\ File Structure}\index{LaTeX@\LaTeX!file structure}


As already stated, a \LaTeX\ file is simply an ASCII text
file which contains certain formatting codes.  Every \LaTeX\
file {\em must} contain a {\em preamble}, a section at the
beginning of the file devoted to declaring style parameters
such as font size, margins, etc...  Also, every \LaTeX\ file
must contain a {\em documentstyle} command and
\verb+\begin{document}+ and \verb+\end{document}+ commands.
The text between the \verb+\begin{document}+ and
\verb+\end{docucment}+ commands is the body of the text, the
text that you want to type.  There are several document
styles, such as article, report, and book; also, many
{\em documentstyle} options are available, such as
specifying font sizes and bibliographic data base options.
See the references for further details.
```

```
\subsection{Labeling and Referencing}
\label{sec:labels}
You will notice throughout this file, that sections,
equations, theorems, tables, and figures usually have a
\verb+\label+ command associated with them.  These are never
required, but can be very useful for referencing the
associated object or environment at other places in the
document.  Using a \verb+\ref+ command allows insertion of
the number associated with an equation, say, without knowing
a priori the number, and without having explicitly assigned
the number. I already used this idea in
Section~\ref{sec:intro_latex}.  And if I wanted to say
something about the Chebychev's inequality, I could refer to
Theorem~\ref{thm:chebychev} or equation~(\ref{eq:chebychev}).
This is especially useful when a document is being revised,
as it saves searching through the document and correcting
all references to an equation, section, or whatever.


\subsection{Floating Bodies}
\label{sec:floats}


\subsubsection{Tables}\index{tables}\index{LaTeX@\LaTeX!tables}
\label{sec:tables}


It's not too hard to produce tables in \LaTeX, and if you
enclose them in a table environment you can give a caption
and use the labeling and referencing capabilities to refer
to them elsewhere in the document.  It is also possible to
produce a list of tables (and a list of figures) automatically.
As an example, Table~\ref{tab:heart} is taken from
Agresti~(1990).


\begin{table}[htbp]
```

```
  \leavevmode
  \begin{center}
    \begin{tabular}{lccc}
      \hline
      & \multicolumn{3}{c}{Myocardial Infarction} \\
      \cline{2-4}
      & Fatal & Non-Fatal & No \\ & Attack & Attack & Attack \\
      \hline
      Placebo & 18 & 171 & 10,845 \\ Aspirin & 5 & 99 & 10,933 \\
      \hline
    \end{tabular}
    \caption{Cross-Classification of Aspirin Use and Myocardial Infarction}
    \label{tab:heart}
  \end{center}
\end{table}\subsection{Figures}
\label{sec:figures}


Figures will generally be PostScript files produced by some
other software, although \LaTeX\ does have it's own
somewhat primitive drawing capability.  Anyway, using the
encapsulated PostScript file macros (epsf.sty) provided with
the dvips program, it's not to hard to do.  For example,
Figure~\ref{fig:scatter} was produced by the following Splus
commands:
\begin{center}
  \begin{verbatim}
  postscript(file="scatter.ps",horizontal=F)
  plot(lottery.number,lottery.payoff)
  lines(lowess(lottery.number,lottery.payoff,f=.2))
  graphics.off()

  \end{verbatim}

\end{center}
\begin{figure}[htbp]
  \begin{center}
    \leavevmode
     \epsfxsize=5.00in
     \epsfysize=3.00in
     \epsfbox{scatter.ps}
```

```
    \caption{Scatter Plot taken from the Splus Book}
    \label{fig:scatter}
  \end{center}
\end{figure}

\index{LaTeX@\LaTeX!importing graphs}
\index{SAS!including graphs in \LaTeX}
To include a graph using SAS/GRAPH$^\copyright$ you would
use the same \LaTeX\ commands but the SAS commands would be
something like the following:


\begin{verbatim}
filename gsasfile 'plot1.ps' ;


goptions gsflen=80 dev=epsf gaccess=gsasfile gprotocol=sasgpasc
noprompt handshake=none trantab=gtabcms gsfmode=replace gend='0d0a'x;


proc gplot ;
plot x*y ;
run ;

\end{verbatim}


The graph will be output in encapsulated PostScript format
and then can be included in the document exactly as shown
above.




\subsection{Typesetting Mathematics}\index{typesetting mathematics}
\index{LaTeX@\LaTeX!mathematics}
\label{sec:math}


\subsubsection{General}
\label{sec:gen_math}
```

```
It's easy to type inline equations in \LaTeX.  These should
be surrounded in your \LaTeX\ file either by \$'s or by a
\verb+\(+ \dots \verb+\)+ pair.  So, as you can see,
$E(X)=\int_{-\infty}^{+\infty}x f(x)dx$
and \(E(X) = \int_{-\infty}^{+\infty}x f(x)dx \) both
produce the same thing.


Tables of many of the symbols available in \LaTeX\ are given
on pages 43--46 of the \LaTeX\ book.  A few things are worth
mentioning here.  A very common ``mistake'' seen in \LaTeX\
documents from beginners is something like $M = min(X,Y)$.
Notice the difference between this and $M = \min(X,Y)$.  The
latter is the correct form.  Similarly, you should type
$f(t) = A\cos(\omega t)$ as opposed to
$f(t) = A cos(\omega t)$.


Another common question is how to type vectors.  Most people
are used to writing a letter with a $\sim$ underneath to
represent vectors, but typeset mathematics rarely does this.
Instead vectors and matrices are represented by bold-faced
math characters.  This is accomplished in \LaTeX\ by using
the \verb+\boldmath+ command.  This leads to a construction
that is long enough that it would be annoying to have to
type it more than once, and since it will probably be used
many times, it's a good idea to write a macro to produce
various bold math symbols.  In the macro file {\tt my.mac},
there is a little macro that does this, so that I can type
something like $\bold{A}\bold{x}$ to represent the product
of the matrix $\bold{A}$ with the vector $\bold{x}$.  Oh,
it's also useful to know how to produce something like
$\bold{x}' = (x_1, \ldots, x_d)$, and now you do.  If you'll
be using this kind of thing a lot in your paper, you might
want to write a simple macro, so that you can just type
something like $\bx'=\rvecx$.
```

```
\subsubsection{Displayed Math}
\label{sec:display}


A displayed formula is one that is centered on a line by
itself.  For example, suppose you wanted to produce a
binomial coefficient.  Somehow this was left out of the
\LaTeX\ book, but it's provided by plain \TeX, and is given
in the \TeX\ book.  It's easy to write a binomial probability,
just type:
$$
p(k)={n\choose k} p^k q^{n-k}.
$$


For simple displayed equations that require no equation
numbers, all you need to do is surround the equation with
\$\$'s (double dollarsigns), though this is not mentioned in
the \LaTeX\ book.  Thus for example, you can type:
$$
E(X) = \int_{-\infty}^{+\infty} x f(x)dx.
$$
If you want a numbered equation, you can use the equation
environment, like so:
\begin{equation}
F(x) = \sum_{k=0}^x \frac{\lambda^k}{k!}e^{-\lambda}
\label{eq:example}
\end{equation}
Now I can refer to equation (\ref{eq:example}) anywhere in
the document (even before the equation) using the
\verb+\ref+ command.


You will notice that the limits on sums and integrals are
aligned differently in display math and inline math.  The
reasons for this are obvious I think.  For the same reasons,
you should be careful about overusing \verb+\frac+,
especially when typing inline mathematics.  For example, to
write the last formula in line, you might consider typing
```

```
$F(x) = \sum_{k=0}^x \lambda^k e^{-\lambda}/k!$, as opposed
to $F(x) = \sum_{k=0}^x \frac{\lambda^k}{k!}e^{-\lambda}.$


Often you will wish to typeset a series of equations or
inequalities with the equal signs or inequalities aligned.
In \LaTeX\ this is done by using the eqnarray and eqnarray
\verb+*+environments.  The former numbers the equations, the
latter doesn't.  These environments are often used in
writing a proof, as in the following, taken from Sen and
Singer (1993):
\begin{theorem}\label{thm:chebychev}
{\bf [Chebychev Inequality]} Let $U$ be a non-negative r.v.
with a finite mean $\mu=E(U).$  Then, for every $t>0,$
\begin{equation}
  \label{eq:chebychev}
   P\{U > t \mu \} \leq t^{-1}.
\end{equation}
\end{theorem}
\begin{rproof}
Note that
\begin{eqnarray*}
\mu = E(U) & = & \int_0^{+\infty} u dP(U \leq u)\\
           & = & \int_0^{t\mu} u dP(U \leq u)+\int_{t\mu}^{+\infty} u dP(U \leq u)\\
           & \geq & \int_{t\mu}^{+\infty} u dP(U \leq u)\\
           & \geq & t\mu\int_{t\mu}^{\infty}dP(U\leq u) \\
           & =& t\mu P(U > t\mu)
\end{eqnarray*}
and, hence, dividing both sides through by $t\mu,$ we get
the desired result.
\end{rproof}


For matrices, arrays, and vectors, you use the array
environment.  As an example, consider:
$$
A = LU = \left[ \begin{array}{cccc}
         1 \\
         2 & 1\\
         2 & 1 & 1\\
```

```
        3 & 2 & 4 & 1\\
        \end{array} \right ]
        \left[ \begin{array}{cccccc}
        1 &2 &0 &1 &2 &1\\
        0 &0 &2 &2 &0 &0\\
        0 &0 &0 &0 &1 &1\\
        0 &0 &0 &0 &0 &1\\
        \end{array} \right ].
$$
The \LaTeX\ book gives more complex examples.
```

Index