Stuff Worth Knowing (and not much more)

Paul E. Johnson <pauljohn@ku.edu>

January 25, 2011

Paul E. Johnson

Department of Political Science

1541 Lilac Lane, Rm. 504

University of Kansas

Lawrence, Kansas

Foreward

I've been teaching advanced statistical methods courses for 20 years. In 2008, I volunteered to teach the elementary course with first year graduate students. As usual, I've been preparing handouts that include my opinions about how things ought to be done and posting them on a commercial Web hosting service:

http://pj.freefaculty.org/stat

For me, there's some deep personal satisfaction in the idea that an undergraduate in, for example, Brazil, can Google search for a topic, find one of my documents, and then send me a message pointing out a mistake. In case you doubt the impact of the Internet on academic life, I'd point to examples like that and say "you are completely wrong."

This manuscript reflects an effort to pull together those lectures and handouts for a first semester research course. My aim is to convey the fundamental concepts and skills that I think are truly necessary for people that are setting out on research careers at this time. Students want to know "what's really important *right now*?" and I intend to bend that curiosity to my purposes. I want to answer the question, "what skills will serve me well for the next ten or twenty years?"

I want students to learn how to manage data collections, how to collect and interpret statistical estimates, how to use R to achieve these purpose, and to get their feet firmly "on the ground" of the computer.

I probably put more emphasis on the computing end of things than most professors. That is partly a reflection of my personal strengths–I'd much rather work on the problem of combining data from the World Bank with data from the US Department of Agriculture, say, than bicker with somebody about how the World Bank gets those numbers in the first place. It is also a reflection of the fact that I'm afraid for the future of poorly trained academic researchers. Computer "users" are being pushed toward the brink of incompetence by the changes in operating systems. More and more students–at least the ones I deal with–have an astonishingly minimal understanding how information is stored and accessed.

They are encouraged to call a "geek squad" to fix the simplest things and rely on "tech support" for services that we used to handle on our own. And I have a conspiracy theory that this is an intended result. Companies would rather that users keep writing checks in order to do their work. Students who don't understand the basics about computers will not have much freedom of choice. They will only be able to do projects for which someone gives them software. I am afraid that graduate students may be "locked in" to a certain limited way of doing research.

I really like John Verzani's book *Using R for Introductory Statistics* and my class is working through that this semester. His writing is clear and the examples are just about perfect. That book comes as close as any book can get to answering the student's question, "what do I really need to know *right now*?" Maybe my purpose here is to provide the additional framework that will help for the next ten or twenty years.

These notes are intended to help people who want to use the statistical program R in their research, especially if they are using the Linux operating system, but also on other systems as well. I think that using Linux is the best way to preserve a person's freedom of thought and options for research, but I understand that many people don't have access to computers on which Linux is installed. R is available for Linux, Macintosh, and Windows, and the other programs I endorse are also available for all three platforms.

Most of the chapters here are in the "beta" stage of development–the outline is ready for general consumption and many of the details are OK, but there are some gaps, typos, and flaws in my LaTeX. If readers find mistakes, please email me.

It would be neat if a commercial publisher said, "hey, let us publish this as a book. We'll help you get it into finished form." I don't think that is likely to happen. The biggest reason is that I'm offering a lot of detailed explanations and examples that would have to be cut in order to fit this into a commercial format. In addition, there is always the problem that it is hard to find people who will pay money for something that I'm encouraging them to download for free. Nevertheless, I'm still working. I won't make money, but I'll have the rich feeling of satisfaction and email from students who want books for free. It could be worse.

Contents

1 There's a Terminal Under There Somewhere

There is a culture war going on in academics today. No, I don't mean the one between Christian intelligent design and Darwinian evolution. No, I don't mean that one between the forces of political correctness and people who call women "girls." And I don't mean the war between scientists and humanists.

Rather, I mean the culture war between people who think that "pointing and clicking" is computing and the rest of us who don't. We (people near 50) think you (people between 21 and 25) don't know as much about computing as you should. You probably don't know as much as you think you do. You can point-and-click, maybe even drag, but we are not impressed. We worry that your approach to computing problems is not clearly structured. When things go wrong, can you figure out why? Do you need to call a "geek" for service? Are you able to use software and concepts from the leading edge of science, or do you have to sit and wait for ten or twenty years until somebody digests those ideas into a collection of "icons" and menus and lets you buy it from them?

We (people near 50) don't want computers to keep going down the path of the automobile. There was once a time when ordinary people could understand how their cars worked and (I know it is hard to believe) actually fix things that went wrong. But there was no profit in that way of life, and cars became more and more complicated and expensive. A car "just works" most of the time, and when it does not, somebody has to write a big check to it fixed. I'm not saying that I would change that, but I do feel uncomfortable about it. When something goes wrong with an auto, I feel completely powerless in the battle of wills with the repair technician who says I need to spend $1500 on a new rack and pinion, whatever that is. I have no good way of knowing if he is telling the truth and offering the best (or least expensive) approach. The best I can do is hire another expert to double-check what the first one says. I have a colleague who will not let his children drive his cars until they prove that they know how to check the tires and change the oil. I applaud him for the effort. I can't say I've done the same with my kids.

At the current time, I am not at the mercy of any computer repair technician. If I want something done, I can usually figure out how to do it myself. I would like my students to have the same feeling of confidence. I am trying to impart that feeling of confidence by forcing students to try to understand a little bit about how to write computer programs. I am trying to stop them from being comfortable with the idea that a computer "just works"

as long as they keep writing checks. I want them to understand why it works and, when it does not work, how it can be fixed.

I'm not a programmer by trade, but I have written programs. I think almost everyone around my age has typed commands into a computer. In 1975, I learned to program in Basic when I was a junior in high school. We were all required to do it. My program calculated quartely compounded interest for a bank account. As I recall, it was about 80 lines long and it was saved on a long, 1 inch wide strip of paper that rolled up (like toilet paper). The paper had tiny holes that were scanned by a device that looked quite a bit like a credit card reader. In college, in 1979, we typed commands that were stamped onto paper "punch cards" and we carried stacks of cards to the computer center. After they were put in a hopper, we would wait by the mailboxes for a paper output. When I joined the faculty at the University of Kansas in 1987, the University provided a computer that ran the Zenith DOS operating system. The monitor displayed only letters and numbers and it had two colors, black and light green. In order to do statistical analysis, we would log into a remote system that ran the CMS operating system. We could type commands and save them in text files, and then we would submit them to the processor, which would write the results into text files. There were no beautiful color graphs.

People who start graduate school today have had a different experience, no doubt, rich in its own way, but, frankly speaking, not as good. Your generation, unlike mine, never had the sheer joy and exhiliration of typing commands and watching for error messages. If you can't remember an operating system before Windows 95, then you are one of the people I'm addressing at the moment. Most of you have never faced "the command line," most of you have never written a program. I'm not saying you are like the person who called Dell and complained that her "cup holder" would not open anymore. But I do think there is something missing in the fundamental framework of your brain.

One way to address that problem is to learn to do something useful that requires you to understand some basic elements of computers. It turns out that R, the leading-edge statistical program, is a great way to do just that. R is a command driven experience. People who use R are exposed to many of the basic elements that are common to all computer languages. And, best of all, in order to use R with any success whatsoever, it is necessary to understand how R interacts with the larger computer environment.

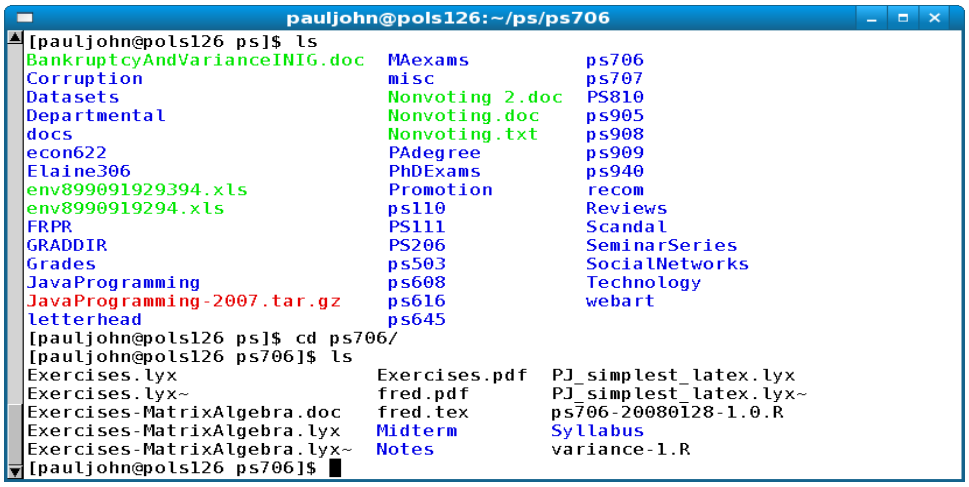
In this chapter, I will try to take you "under the hood" of a modern operating system. I can't make you into a geek without your willing consent and cooperation. But I can help you understand a geek, which is almost as good. I have some advice that applies to all kinds of modern operating systems, including Linux (and Unix), Macintosh, and Microsoft Windows. Linux is a PC-sized version of the older Unix system. I prefer to use the Linux operating system, mostly because it doesn't cost anything and it leaves me free to learn and try the newest, most exciting scientific software. The Apple Macintosh and Microsoft Windows operating systems are, on the surface, quite different, because they are so pointy-clicky. But that is only on the surface. A Macintosh runs its graphical interface on top of the BSD Unix operating system. Although the Windows designers try harder and harder to hide their roots, there is still a DOS framework underneath.

1.1 Start in a terminal.

Let's begin by considering Linux and Macintosh systems. After logging in, one will usually be presented with a graphical environment. That will have menus and icons and other pretty things. The first step deeper into the operating system is found in a "terminal emulator" program. These are called "terminal emulators" because their creators still remember the day when there were no graphical interfaces. In those days, one would sit in front of a terminal and type commands and read output in the form of letters and numbers. A terminal emulator program is supposed to re-create that spartan way of life, allowing users to be one step closer to the computer environment.

There are many different terminal emulator programs. In Figure 1.1, there is a snapshot of one of my favorites, the "mlterm".

Figure 1.1: The MLterm



In this image, you see I've typed some commands to display a list of directories and files on the system.

On a Linux (or Unix) system, there are other readily available terminal programs like "xterm" "gnome-terminal" "rxterm" or "kterm." They are all just variations on a theme. I like MLterm because it handles international characters very well and it is also has a highly customizable theme. An MLterm can be truly beautiful. This one is bland because I wanted to reproduce it in this document.

In Windows XP, there is a terminal emulator. One can click on the Start menu, and navigate to Accessories and find an icon "DOS Command Box" (or similar). Choosing that will open a terminal emulator in a window. That can also be obtained by opening the "run" menu and typing "cmd". On a Macintosh OSX, the terminal emulator is called "Terminal." It is installed by the factory. It can be found in the Applications folder, in a subfolder called Utilities.

All terminals have a prompt, an invitation for the user to type a command. The $ or symbol like > or # indicates that the terminal is ready for input.

A **shell** is a program with which one interacts in the terminal. There are many different shell programs. Most Linux systems today use the shell called Bash (Bourne Again Shell). Newer Macintosh also use Bash. Microsoft offers the "Command" shell, named after the "command" prompt.

Although there are many different shells in existence, they seem to be converging toward some common functionality.

1. All shells have some kind of **history, or command recall,** mechanism. It is not necessary to type a command over and over. On Linux, Mac, and Windows XP systems, the "up" and "down" arrow keys will cycle through previously typed commands. Those command can be executed again by hitting return, or they can be revised and then executed.

2. Decent shells will allow some form of "**word completion**." The goal is to make it faster to type commands by having the system "guess" the endings of the words or commands. If there is an executable command with a long name, such as "myReallyLongProgramNameIsBoringToType," then one could just type "myR" to get this started, and then hit the TAB key once or twice and the shell will look for completions of "myR." If there is only one valid completion, the shell will fill that in. There may be some systems on which the TAB key is not used for completion. On the DEC Unix systems at the University of Kansas, we used to have to hit Escape twice to ask for a completion, but I've not seen a system like that in a while. In January, 2008, I noticed that the tab key was the default for word completion on Linux, Windows, and Macintosh computers. When you are typing long names of files, the tab key will complete those words for you, or offer a list of possible fits.

   Tab completion has another important benefit. It reduces the number of typographical errors that users will make. The completion mechanism will search not only for program names, but also for valid data file names. Suppose you want to read a data file that is stored in "/home/pauljohn/DL/class2009fromJim/myGoodStuff.txt". To reduce the chance of a typographical error, type the first few letters and hit the TAB key, and the system will try to complete it for you. If the tab completion fails, it is a good hint that you have made a typographical error.

Another common feature of the modern user interface is that several programs can be "open" at once, and the windows in which they are displayed might be "stacked" on top of one another, or they might be hidden from view altogether. The most immediate aspects of the graphical interface are controlled by a program known as a **window manager**. To most users, the window manager is as much a part of the computing environment as the shell, because the window manager controls the way that things look on the screen. The window manager is the program that helps you find windows and re-arrange them. In my opinion, people rely too much on the mouse, causing carpal-tunnel syndrome, and I encourage them

to learn keyboard shortcuts for window management. Most users seem not to realize that Alt-Tab will cycles through open windows. On Mac, use the Command key in place of Alt.

The window manager controls where windows "show up" when programs start. It draws the "outer edges" around programs–it draws the "resize bars" and "close" or "minimize" buttons. It also handles the way that windows can be layered on top of each other and minimized.

If you become an expert computer user, you may be interested in customizing your choice of window manager to manipulate graphical effects or change desktops. I have seen this done on all operating systems, but it is easier in Linux or Mac than on Windows. In Figure 1.2, I proudly present a snapshot of my Ubuntu Linux 8.10 laptop running a new window manager called Compiz-Fusion. It is a fully 3 dimensional framework that is highly customizable. The ability to fundamentally change the appearance of the graphical interface by replacing the window manager is the key feature that convinced me to adopt Linux as my operating system of choice in 1997. I don't recommend that novices try to do too much fiddling with the window manager, but it is important to realize that it is possible to do so. [1]

1.2 Directories and files

file: A file is a "container" for information. It may hold information that we view as text, or as a photo, or a compressed "zip" archive.

dot file: a file whose name begins with a period. These are "hidden" files. Most programs do not display them by default.

file system: The "file system" is a well organized collection of directories.

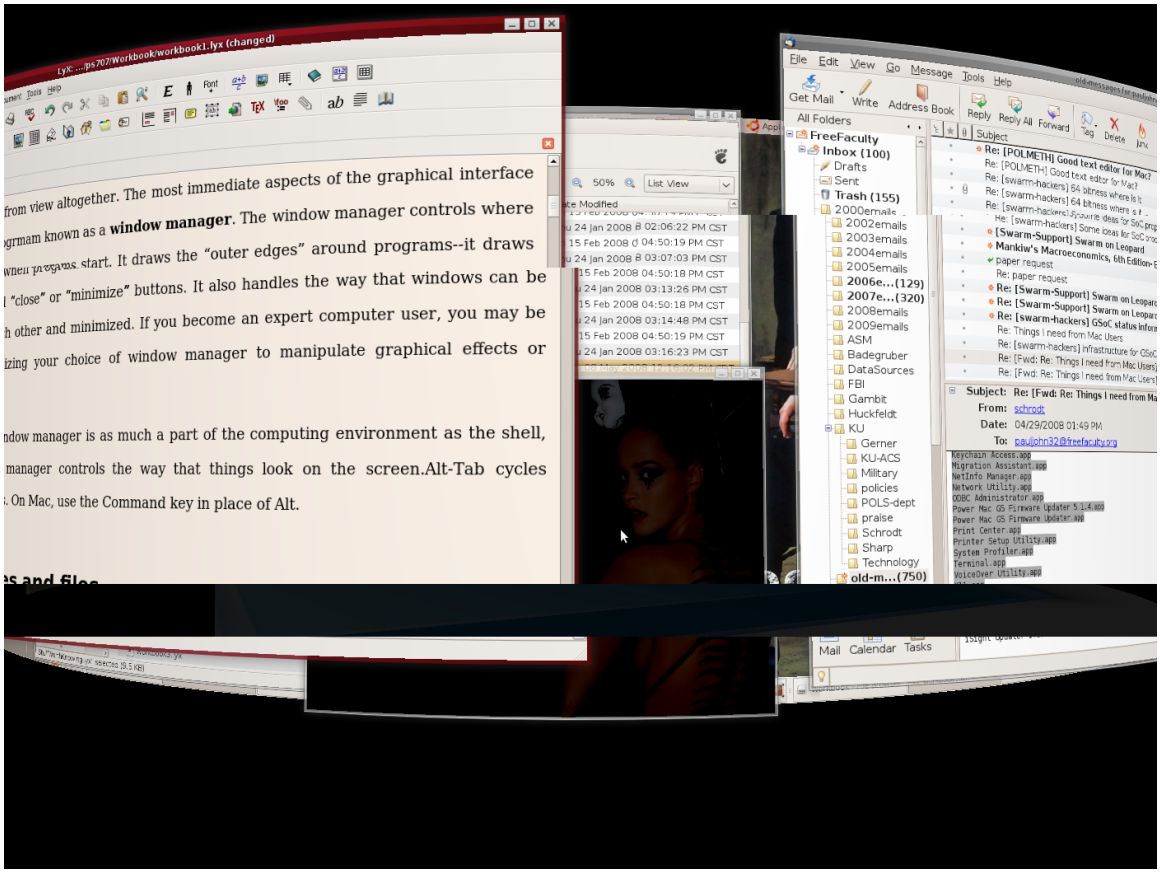
Directory: This is what the graphical interfaces on Mac and Windows call "folder": a collection of files or directories.

PATH: list of directories in which system looks for programs you want to run.

/ the "root" of the file system. This is a Linux/Unix/Mac concept. It is the "top level" address, a place from which one can go to access each and every file in the system. Conceptually, it is like the thing that Microsoft Windows calls "My Computer." It is the one place from which one can go anywhere else.

[1] It is important for the following reason. Sometimes things will go wrong. For example, students often say that windows disappear or vanish. While programs may die sometimes, it is much more likely that the programs are still running but that the window display has been moved to a different location. In those cases, a little bit of knowledge about the window manager will go a long way.

Figure 1.2: The Compiz-Fusion Window Manager



1.3 List the contents of a directory

Open a terminal and type the command that matches your operating system

Linux or Mac	Microsoft Windows
`$ ls`	`> dir`

This displays all files & directories in the current working directory (ls is abbreviation for "list").

Most operating systems have documentation that is available for basic commands like these. In Linux and Unix, the longest lived approach for documentation is the "man" system (short for manual). People who write programs are expected to provide "man" pages with them. To read the help for the ls command,

```
$ man ls
```

Use the SPACE BAR to scroll to the bottom, or hit the "q" key to quit.

In some Microsoft systems, there is relatively detailed help available. Type

```
> help dir
```

to read more about it.

On all of these systems, there are many options that control the amount of information that is provided. Most default to present only a list of file names, but the can also display file sizes, creation or modification times, ownership, and so forth.

1.4 The Environment

The environment is the framework of components upon which a program can call when it needs information or storage. If a program wants to save some output, for example, it may ask the environment where the user's HOME directory is, and it will use that as a default storage place.

To see all components of the current environment on a Linux or a Mac system, open the terminal and run the command

```
$ env
```

That command will cause the users "environment" to be displayed. Here's sample output from the env command on a Linux system. In my terminal, the prompt is a dollar sign.

```
$ env
SSH_AGENT_PID=2639
KDE_MULTIHEAD=false
HOSTNAME=pols126
GPG_AGENT_INFO=/tmp/gpg-iCTi3m/S.gpg-agent:2726:1
SHELL=/bin/bash
XDG_MENU_PREFIX=kde-
TERM=xterm
DESKTOP_STARTUP_ID=
HISTSIZE=1000
WINDOWID=46137414
QTDIR=/usr/lib/qt-3.3
QTINC=/usr/lib/qt-3.3/include
KDE_FULL_SESSION=true
USER=pauljohn
SESSION_MANAGER=local/unix:@/tmp/.ICE-unix/2760,unix/unix:/tmp/.ICE-unix/2760
XDG_CONFIG_DIRS=/etc/kde/xdg:/etc/xdg
MAIL=/var/spool/mail/pauljohn PATH=/usr/lib/qt-3.3/bin:/usr/kerberos/bin:/usr/
    lib/ccache:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/pauljohn/bin
    DESKTOP_SESSION=kde
GDM_XSERVER_LOCATION=local
INPUTRC=/etc/inputrc
PWD=/home/pauljohn
HOME=/home/pauljohn
CVS_RSH=ssh
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-VbEq0wlmse,guid=
    fb554b550b8e03710bd9b9004790cd12 XDG_DATA_DIRS=/usr/local/share/:/usr/
    share/:/usr/share/gdm/
LESSOPEN=|/usr/bin/lesspipe.sh %s
DISPLAY=:0.0 G_BROKEN_FILENAMES=1
XAUTHORITY=/tmp/.gdmAQZM5T
```

What is all of that stuff? Well, I don't worry about most of it. But I do know that USER, SHELL, HOME and PATH are vital components.

When you want to access values from the environment in a Linux or Mac system, use a dollar sign, as in $HOME or $CVS_RSH or such. The "echo" program will grab something and display it on the screen. Try the command

```
$ echo $HOME
```

or

```
$ echo $USER
```

to see what I mean.

The DOS command box also allows access to the environment. In Windows, the command "set" prints out the environment.

```
> set
```

If options are added, it can create variables in the environment. On a Windows system, the environment can be accessed through a graphical interface in the Control Panel (Look under System/Environment). But there's hardly any fun in that!

I recently asked a student who owns a new Macintosh to share the environment output with me, and it looks like this

```
$ env
MANPATH=/usr/share/man:/usr/local/share/man:/usr/local/man:/Library/TeX/
    Distributions/.DefaultTeX/Contents/Man:/usr/X11/man
TERM_PROGRAM=Apple_Terminal
TERM=xterm-color
SHELL=/bin/bash
TMPDIR=/var/folders/uB/uBGqU6vWGOiJNWZM-gymUk+++TI/-Tmp-/
Apple_PubSub_Socket_Render=/tmp/launch-QMA1pa/Render
TERM_PROGRAM_VERSION=240
USER=phyllis
COMMAND_MODE=unix2003
SSH_AUTH_SOCK=/tmp/launch-HsG0jU/Listeners
__CF_USER_TEXT_ENCODING=0x1F5:0:0
PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/texbin:/usr/X11/bin
PWD=/Users/phyllis
LANG=en_US.UTF-8
SHLVL=1
HOME=/Users/phyllis
LOGNAME=phyllis
DISPLAY=/tmp/launch-a5FMoz/:0
SECURITYSESSIONID=8b0380
_=/usr/bin/env
```

1.5 About the Path

The path can be examined by reviewing the environment, or one may ask for it directly.

Linux or Mac	Microsoft Windows

`$ echo $PATH`	`> echo %PATH%`
	or
	`> path`

That will display only the PATH component of the environment. The path, defined as "the list of directories in which the system looks for programs when you try to use them."

Here's what I see when I type that command on one Linux system.

```
$ echo $PATH
PATH=/usr/lib/qt-3.3/bin:/usr/kerberos/bin:/usr/lib/ccache:/usr/local/bin:/usr
    /bin:/bin:/usr/X11R6/bin:/home/pauljohn/bin
```

This means that if I type in a program name, say "whatever", the system will look in each colon-separated location before it fails and says "whatever" does not exist.

A file is called an *executable* if it can be "run." That is to say, typing the program's name will cause it to be submitted to the operating system. Otherwise, a file is just "text" or "data" or a "picture". In Windows systems, the tradition is that executable programs end with the suffices ".exe",".bat", or ".com". In Unix/Linux/Mac, there is no such requirement, and an executable file may have no suffix at all. The ability to execute a command is determined by file properties, to which we shall turn below.

If an executable file is in the PATH, it can be started simply by typing its name. Example, type "oowriter". If a program is not in the PATH, it must be started by including its full path, as in "/usr/bin/oowriter".

The command

```
$ pwd
```

(abbreviation of print working directory) shows "where you are now." The current working directory can be referred to by the shorthand symbol "." (the period).

The Linux system does not automatically add the current working directory to the path. This is considered to be a security feature, it prevents you from accidentally running harmful programs that hackers might try to drop into your directories. If there is an executable file "whatever" in the current working directory, then a "dot slash" must be prepended in order to run it. As in:

```
./whatever
```

If 'whatever' is in the current directory, and the current directory is in the path, then typing "whatever" would be sufficient.

which program_name Will ask the system "where is the executable program_name stored"? On some systems, the program "type" does the same thing as "which"

1.5.1 Setting environment variables

The shell offers a way to set values of environment variables. People who used Windows 95 will probably recall a text files called autoexec.bat and config.sys. Users often had to revise lines in that file in order to make devices work or to set environment variables. In Windows XP, the control panel has a System icon that has an environment tab in which the path can be changed.

The environment variable that I've changed most often is the PATH variable. By adding directories to the path, it makes it easier to launch programs. Suppose I'm working on a program that is stored in "/home/pauljohn/myFavorites/newTestProgram." If I want to run that program, I have to type out its full address. If I add "/home/pauljohn/myFavorites" to the path, then it is necessary to type only "newTestProgram."

We usually do not replace the path. We add components to it. On Linux and mac, the existing value of the path can be accessed by the symbol $PATH. To add a new directory "/home/pauljohn/myFavorites", a command like this would work:

```
$ export PATH=$PATH:/home/pauljohn/myFavorites
```

In Windows, instead of referring to the existing path as $PATH, the symbol %PATH% is used.

```
> set path=%PATH%;c:\Users\pauljohn\programs
```

In addition to changing existing variables, the shell also allows one to add new variables. When programs are being developed, it is often necessary to set variables in the environment to govern the way test programs run. For example, if a program has special duties to perform when the DEBUG variable is set, then we can set set that parameter

```
$ export DEBUG=1
```

Programs that can access the shell can check the value of $DEBUG.

To find out the current value of an environment variable, use the echo function, as in

```
$ echo $DEBUG
```

or, using the "pipe and grep" trick,

```
$ env | grep DEB
```

The pipe "|" transfers the output from env to the program "grep." Grep is a text scanner. It displays all lines from the output that have the letters DEB.

Is there a difference between "env | grep HOME" and "echo $HOME"? Well, just a little. If you know the name of something in the environment, you can get it exactly by asking for it by name. But if you are unsure of the full name, but you know a part, then try the grep trick. To see what I mean, try the command:

```
$ env | grep HO
```

That finds every environment variable that has the letters "HO".

1.6 HOME: A Very Special Writable Directory. Every User Has One.

Home is the user's "place" for saving files. On a Linux or Unix system, the custom has been to create user accounts within a directory that is called /home. Windows 95 had no "home directory" at all and users were allowed to write any files they wanted to write on any part of the operating system (bad). On the Macintosh OSX, the home directories of the users are saved under a directory called /Users. In a stunningly original move, Microsoft Vista introduced the new user home directory "C:\Users." While it is obviously immitating the Mac, it is a good move because the previous approach using "C:\Documents and Settings" was quite disastrous. The spaces in that directory name tended to confuse scripts that were fooled by the space into thinking that the directory name was just "C:\Documents".

When you log into a Linux system, and you open a terminal, it will generally display files in your home directory. My username is always "pauljohn" and so my HOME is /home/-pauljohn, and when I type

```
$ pwd
```

it returns the answer "/home/pauljohn".

The command:

```
$ env | grep HOME
```

is another way to find out what the HOME variable's value is.

If you type

Linux	Mac	Windows Vista	Windows XP
`ls /home`	`ls /Users`	`dir c:\Users`	`dir c:\Documents and Settings`

you should see a list of all the users that are defined on the system you are using.

On a Linux system, the user's HOME and the /tmp directory are the only directories where users can write files.

$\sim$ symbol that represents the user's HOME directory

View the non-hidden contents of the home directory while in any other directory:

```
$ ls ~
```

That lists the directories and files that are in HOME.

The symbol $\sim$ can be used in copy and move commands.

To use this in an example, we need to create a little file and dump it somewhere. Type like this in the terminal:

```
$ echo Professor Johnson Knows Something Useful > /tmp/some_file.txt
```

That creates a file, and drops it in the /tmp directory. That's a temporary directory, a place where all programs can write trash and scratch files and so forth. The system cleans it up often.

To copy a file out of the temporary space into my hope directory, I can either run

```
$ cp /tmp/some_file.txt  /home/pauljohn
```

or

```
$ cp  /tmp/some_file_I_want  ~/
```

Please don't be one of the student who comes to me with an error message saying that he/she is unable to access /home/pauljohn in the penultimate example. Of course, you have to put your HOME directory there. Not mine!

1.7 Read, Write, Execute, For User, Group, and Other users

On a Linux or Mac system, a detailed listing of all files, including hidden files, is obtained by adding the options "-la", as in

```
$ ls  -la
```

That displays files with more details. Here is some sample output, where I've found a directory called ".xine" as well as a file called "paul.conf"

```
drwxr-xr-x 2 pauljohn pauljohn 4096 2007-12-16 19:14 .xine
-rw-r--r-- 1 pauljohn pauljohn 4967 2006-06-01 22:20 paul.conf
```

The first part of the line has 10 symbols.

The first symbol indicates if the item is a directory (d means it is!).

The 9 following symbols give the permissions of the "user", the user's "group", and "others" in the system.

`r=read, w=write, x=execute.`

rwxrwxr-x means the user can read, write, and execute, and the same is true of group members. Others can only read and execute, but not write.

On the other hand,

`-rw-------`

means a file is readable and writable by the owner/user, but not executable, and neither the members of the group nor the other users have any rights at all.

The ls program accepts the asterix as a "wildcard", meaning "any combination of letters or numbers or symbols."

For example,

```
$ ls g*
```

That example lists all files in the working directory that begin with g. If one wants to see the hidden files that begin with g, a period must be inserted, as in

```
$ ls .g*
```

Recall that *hidden files* are the ones that start with periods. They do not show if you run "ls" but do show if you do "ls -la". These files are used for configuration files.

```
$ ls ~/usr/bin
```

displays all files in /usr/bin.

1.8 Take a look at the programs in your path

Double check your path. Run

```
env | grep PATH
```

The directory /usr/bin is the standard place where programs that you usually need are stored. That's where you find text editors like "emacs" and image viewers like "gqview". Go browse /usr/bin, you'll see a big list of programs.

Programs that are used for system administration are stored in /sbin or /usr/sbin. On some systems, those programs are in the path for ordinary users, but usually they are not.

The directory /usr/local/bin is supposed to contain programs that are built and installed for your particular computer, or for your particular computing lab.

The directory $HOME/bin is a place where users can install programs. On most systems, that directory is in the path.

If a user installs a new program in some other place, such as $HOME/whatever/some/directory/bin, and wants to add that to the path, it can be done easily. To append that to the existing path, the command to type is

```
$ export PATH=$PATH:$HOME/whatever/some/directory/bin
```

That does not change the path permanently. Only within that one shell is the path changed. To make a permanent change, the user should edit a configuration file. On my system, in $HOME/.bash_profile, I have

```
export PATH=$PATH:$HOME/bin:/usr/local/stata8/bin:/usr/local/netlogo/bin
```

Apparently, at one time, I had versions of Stat and Netlogo installed and I wanted them in the path.

The directory /etc is the place where configuration files for the system itself are stored. Go ahead and see if it will let you view and edit them. If the system allows you to damage these files, then your system administrator is not very sharp.

1.9 Run a program

Suppose for a moment you are on a Mac system. Here is a list of some files that are stored in a directory called /Applications/Utilities. That directory is (by default) in the path.

```
% cd /Applications/Utilities
[Scott-G5:/Applications/Utilities] jimmy% ls
Activity Monitor.app
Adobe Utilities.localized
AirPort Admin Utility.app
AirPort Setup Assistant for Graphite and Snow.app
AirPort Setup Assistant.app
Asia Text Extras
Audio MIDI Setup.app
Bluetooth File Exchange.app
ColorSync Utility.app
Console.app
DigitalColor Meter.app
Directory Access.app
Disk Utility.app
EarthLink TotalAccess
Grab.app
Grapher.app
Installer.app
Java
Keychain Access.app
Migration Assistant.app
NetInfo Manager.app
Network Utility.app
ODBC Administrator.app
Power Mac G5 Firmware Updater 5.1.4.app
Power Mac G5 Firmware Updater.app
```

```
Print Center.app
Printer Setup Utility.app
System Profiler.app
Terminal.app
VoiceOver Utility.app
X11.app
i-Installer.app
iDisk Utility.app
```

Notice, for example, Terminal.app. In your terminal, you can type "/Applications/Utilities/Terminal.app" and that should make another terminal "pop up" on the screen. The same is true for the other items in that list. The Mac Finder probably has menu icons that run those same programs, but you can run them from the command line.

1.10 The most commonly used commands are cd, cp, mv, rm, and mkdir.

mkdir x creates a directory called x in the current working directory

mkdir -p /home/pauljohn/some/directory The "-p" option will create a directorys "some" and then inside it "directory" will be created. If "-p" is not used, then an error results unless some exists in /home/pauljohn.

mkdir -p ~/some/directory. Note ~ works for all users, whereas /home/pauljohn only works for pauljohn

cd x changes the working directory to x

cd .. changes the working directory to the one "above" the current working directory

cd cd with no options changes the working directory to HOME

rm x remove a file x

rm -rf x remove a directory x and all of its contents (rf options mean "recursive" and "force")

mv x y moves a file or directory x to a new name y. There is no "rename" command. mv serves that purpose.

mv x ~ moves file x to the user's HOME

cp x y copies a file x to y. Does nothing if x is a directory.

cp -R x y the -R option means "recursive". It will cause the copy to go into directories and subdirectories.

Windows

I recently had to use the Windows shell quite a bit in a consulting project. After an initial shock of transition, I found I was able to do most of what I needed to do. According to this website, `http://www.ss64.com/nt/`, the Windows Command shell supplies a set of functions that are always available. Unlike Linux or Mac, Windows is not case sensitive (a source of much trouble and confusion), so "CD" is the same as "cd".

```
ASSOC    Change file extension associations
CALL     Call one batch program from another
CD       Change Directory - move to a specific Folder
CLS      Clear the screen
CMD      Start a new CMD shell
COLOR    Change colors of the CMD window
COPY     Copy one or more files to another location
DATE     Display or set the date
DEL      Delete one or more files
DIR      Display a list of files and folders
ECHO     Display message on screen
ENDLOCAL End localisation of environment changes in a batch file
ERASE    Delete one or more files
EXIT     Quit the current script/routine and set an errorlevel.
FOR /F   Loop command: against a set of files
FOR /F   Loop command: against the results of another command
FOR      Loop command: all options Files, Directory, List
FTYPE    Display or modify file types used in file extension asso-
ciations
IF       Conditionally perform a command
MD       Create new folders
MOVE     Move files from one folder to another
PATH     Display or set a search path for executable files
PAUSE    Suspend processing of a batch file and display a message
POPD     Restore the previous value of the current direc-
tory saved by PUSHD
PROMPT   Change the command prompt
PUSHD    Save and then change the current directory
REM      Record comments (remarks) in a batch file
REN      Rename a file or files.
RD       Delete folder(s)
SET      Display, set, or remove environment variables
SETLOCAL Control the visibility of environment variables
SHIFT    Shift the position of replaceable parame-
ters in a batch file
START    Start a program or command in a separate window.
TIME     Display or set the system time
```

```
TITLE    Set the window title for a CMD.EXE session
TYPE     Display the contents of a text file
VER      Display version information
VERIFY   Verify that files have been saved
VOL      Display a disk label
Some example usages of the most frequently used com-
mands are found on http://www.cs.unca.edu/~jkdawg/help/msdos.html.
```

dir Lists names of files in current directory (folder)

dir Hello* Lists all files whose names start with Hello.

cd C:\files Changes directory to C:\files ("absolute" pathname)

cd myfiles Changes directory to myfiles subdirectory of current directory ("relative" pathname)

cd .. Changes directory to "parent" of current directory

notepad Hello.java MS-DOS text editor, used to create and edit ASCII textfiles

type Hello.java Displays contents of ASCII textfile on screen

exit Ends command interpreter, makes console window go away...

copy Test1.java Test2.java Copies contents of Test1.java to new file Test2.java

ren Test1.java Test2.java Renames Test1.java to Test2.java

del Test1.java Deletes Test1.java (Be careful with this command...)

del Test1* Deletes all files with Test1 prefix (Be even more careful with this command...)

mkdir playpen Creates new directory playpen as subdirectory of current directory

rmdir playpen Removes directory playpen (must be empty)

1.11 Other things worth knowing on Linux & Mac

grep magic_word * grep will search for the text "magic_word" in all files found by *

ps aux The "ps" command lists system processes. With the options aux, it shows all processes launched by all users.

ps aux | grep whatever "|" is the pipe symbol, and grep is a filter that hides all lines except the ones that have the exact letters "whatever".

top Shows the programs that are using the most system resources (CPU & Memory). Hit Control-c to stop the display.

kill 1234123 The ps program displays programs by number. If you think one is "stuck" or "running out of control", then kill it by using its program number. If that fails, there's an industrial strength kill option, -9

killall whatever Will try to find all programs you have called "whatever" and kill them. Has only occasional success.

1.12 Media are Mounted and Umounted.

mount. Term meaning "to create access to files" by attaching them to the file system. You can't access a file on a device unless it is mounted first. One of the most frustrating things about moving from one operating system to another is that they all use their own ways of finding external media and making it available. Windows XP notices if a USB drive or CD is inserted and it prompts the user, asking if a program should run or a folder should be opened. If the user accidentally chooses the wrong thing, the system may remember the mistake and repeat it over and over. How frustrating?

Linux has a framework for external media that is based on the traditional Unix framework in which all files are referred to in relation to / , the root directory. The current custom is that CDRoms, floppy disks, USB drives, will be mounted under the directory /media. Until a few years ago, media would have been mounted under /mnt.

```
$ ls /media
```

The Linux operating system has been evolving toward a point where most file storage devices are automatically mounted by a framework known as the "hardware abstraction layer", also known as "hal". Not all devices mount automatically. Floppies never do. Audio CDs just play, they are not mounted.

If the automatic mount fails, then the command "mount" is used to get the job done. A physical device will be listed under the /dev directory, and to make that accessible, it is mounted by under the /media directory by a command like this.

```
$ mount -t iso9660 /dev/cdrecorder/media/cdrecorder
```

If you have proper permissions, you can tell the system to mount a CD-R device under the mount point /media/cdrecorder. The filesystem type is "iso9660".

Umount. Some smart alec decided to frustrate new users by making the command to detach mounted devices "umount" rather than "unmount". It is vital to umount devices, particularly USB flash devices, in order to make sure all files are finished in writing.

umount /media/usbdrive will remove a device called /media/usbdrive.

Some file manager programs, such as Nautilus, will try to help with the problem of unmounting devices.

1.13 Is it Supposed to be \ or /?

Windows is an outgrowth of DOS, and DOS was created as a small version of the Unix operating system for personal computers. The pioneers of DOS did not copy the Unix syntax exactly, however. On Windows, for example, path is printed somewhat differently. Where on Linux one would fidnd the path like this (a colon separated list)

```
/usr/bin:/usr/local/bin:/home/pauljohn/bin
```

on Windows one finds a semi-colon separated list like

```
C:\windows;C:\windows\system32;\Program Files\Microsoft Of-
fice;C:\Cygwin
```

In Windows XP, the forward slash is also accepted in most commands.

In Windows, there is no "root" folder comparable to / that which is in Linux or Mac. The closest you can get to / in a Windows system is the output of "My Computer" in the graphical interface. This is a source of much trouble for Windows users and programmers. Once I had several hard disk partitions and the Windows system named them C:\, D:\ and E:\. When I would insert a CD-ROM into the system, the CD would "grab" the letter D:\ and my two partitions would be bumped down one letter in the alphabet. The content that used to be under D:\ was then listed in E:\. Of course, that broke all programs that went looking for files in D:\. For me, this was quite a crippling weakness of Windows 95.

2 Running the Right Programs

In MS Windows and Mac systems, the user is presented with a more-or-less locked-in graphical interface. One can easily change color schemes, but it is quite difficult to make changes that fundamentally alter the user's computing experience. The Linux operating systems offer a broader variety of user interfaces because there are competing groups of programmers, research institutes, and corporations that are trying to put their own "stamp" on the Linux user experience.

All of the modern operating systems are based on the same concepts. The user is presented with a "workspace," possibly more than one. Programs are displayed in rectangular layers. The systems use different terminology and have different "widgets," but many of the core concepts are the same. The user has a menu from which programs can be started. Several programs can run at the same time and they can draw their displays on the screen within rectangular "windows" or "frames".

The legend holds that when Apple computing pioneer Steve Jobs visited the Xerox PARC research center in 1979, he saw a prototype computer that used a pointing device called a "mouse" and a color screen windowing system. The concepts that he gleaned were embellished and embodied in the Apple Lisa computer, and later the famously successful Macintosh, which hit the market in 1984. Microsoft leader Bill Gates was aware of Jobs's plans, and about the same time Microsoft introduced its graphical interface, which it called Windows. Early versions of Windows amounted to little more than "program launchers" that could initiate programs; there was no common "look and feel" across programs and programs did not 'work together' in any meaningful sense. For me, the first minimally functional version of Windows was version 3.1, which we were using in 1992, and not until Windows 95 did it seem to me that there was any real benefit in the graphical user interface.

On Linux systems, there is no "big company" that has the power to force through a single graphical user interfaces (GUI). All Linux GUIs are built on top of the X11 video windowing system. On top of that X11 (or xorg) framework, one adds a window manager and possibly some kind of "desktop" for the convenience of a user. I've seen at least 20 different window managers in action over the years. There are several competing "desktop" systems. Because almost all Linux programs are "open source," programmers are more free to try out new ideas about the graphical interface. A benefit is that new and interesting features pop up all the time. That's a disadvantage as well, however, in the sense that change makes new tools frustrating for users.

2.1 "Session" is a Catch-Phrase for a Bundle of Options

Leading efforts to build a complete "Linux desktop" in the Mac or MS style are called Gnome, KDE, and XFCE. Less ambitious efforts exist as well. In the 1990s, the Sun Corporation sold a graphical interface for Unix systems called the Common Desktop Environment, but because it was a closed/proprietary framework, it fell out of use when free alternatives became available that were functional and open source. The KDE project was the first effort to make a free, open source, integrated Linux desktop, but it was based on a closed, proprietary software library known as QT. Objecting to the reliance on the closed QT framework, a competing graphical interface called Gnome was offered (based on the free/open toolkit known as GTK). KDE and Gnome have gone through several iterations and both of them have very adamant proponents and detractors. Believing that both KDE and Gnome are annoying and tedious, smaller, groups of programmers have sought to develop more streamlined desktop interfaces, such as XFCE.

I have been using RedHat Linux based systems for more than 10 years, and the RedHat company was an original opponent of KDE and one of the supporters of development in Gnome. When the QT toolkit was made available in a free version, RedHat agreed to package KDE as a desktop option. RedHat (and its offspring, such as Fedora Linux) now generally offer several desktops and one can choose among these types of "sessions" before logging in to the computer. If your Linux system does not offer a choice, it may mean your installer elected to keep things simple by just installing one desktop interface.

The session type controls these kinds of parameters:

1. Menus for selecting programs
2. Icons on the desktop
3. A window manager
4. A "background manager"
5. A workspace switcher

2.2 How to change the type of session you have.

On the Linux workstations in our computer lab, the type of "session" desired can be selected. The default type is Gnome, but one might also try the others, especially KDE. KDE 3 has much to recommend it. If a WindowMaker session is available, try that for a "bare bones" Linux user experience.

On the login screen, there should be a "button" or menu item that allows you to choose your session. Today, the only sessions I really use are Gnome and KDE. In the old days, I used WindowMaker, which is simple and nice, and I also think XFCE is nice, but not better. By default, you get Gnome. Alternatively, you get KDE.

2.3 Does it make a Difference Which Session you Choose?

Well, it makes a big difference to some people. To many people, the differences are not very apparent because those people do not delve very deeply into the interface.

2.3.1 Maybe the Type of Session Does not Matter!

The desktop environnments make competing versions of programs that do exactly the same thing. Want evidence? Consider the file manager of each desktop. Open a terminal and type

```
$ nautilus . &
```

In Fedora, nautilus is "dumbed down" a bit, so use the pull down menus to smarten it up. Go to Edit | Preferences. Under the Views tab, tell it to use the List Mode (instead of Icon Mode) and click the button "Show backup and hidden files." Under the Behavior tab, tell it to "always open in browser windows." After you close the configurations, close nautilus, and re-start, you will see a thing that looks more like a file manager.

Then open a terminal (or from the same one if you remembered the &) type

```
konqueror . &
```

What do you have? **Two File Managers!**

There were about 20 competing file managers in the Linux community when I started. Now we are setting down to a competition of 8 or 10. Try running "Thunar" if you have it.

The Fedora Linux distribution (a descendant of RedHat Linux) that I use is crafted so that you can run either Gnome or KDE without noticing too many differences. The Gnome team writes some good programs, the KDE team writes some good programs, and we'd like to try some of each.

Without a doubt, the best Gnome program is "Gimp", the Graphics Image Manipulation Program. It is a photo editing program. Open a terminal, type

```
gimp
```

Once you get the hang of this, you will think it is the greatest program ever. It is totally awesome. I've used it to do a lot of photo work for my family's yearbook publication. I've had to remove old boyfriends from family photos with Gimp.

Without a doubt, the best KDE program is "k3b", a disk burning tool. It can write CD and DVD within a very pleasant and workable graphical interface. In the terminal, type

```
k3b
```

and you will be treated to a disk creation extravaganza!

2.3.2 When does the Type of Session Matter?

2.4 Why does Professor Johnson always run programs from the terminal?

Once students start to find their way around the lab, they notice that there are menus in the graphical interface so they can start Gimp and k3b without opening terminals. OK, fine. You are so clever.

I hardly ever do that, except when I run the web browser firefox or I start terminals. Instead, when I want to run a program, I open a terminal. Then I use "cd" to change to the desired directory in which I want to run the program, and then I type the name of the program. It is very liberating to be able to start programs without relying on icons or menus.

I start things from terminals because

1. It helps me to figure out what's wrong, when a program crashes or I make a mistake. Error message will show in the terminal where the program was started. That is especially important for LyX, which gives nice debugging information.

2. I can cd to a working directory in the terminal and programs like Emacs with ESS will notice and use that working directory.

2.5 What Programs Does a Windows Person Need?

I don't use Windows[1] when I can avoid it. If you are "stuck" with Windows, you can still get good software. The GNU community has been investing much effort to make available open source software for Windows. Consider the web browser, Firefox, probably the most famous example. One can obtain Windows versionf of the most famous Linux programs like Gimp, Emacs, R. While these Windows versions may not be quite as nice as their Linux versions, they "mostly work."

When I'm given administrative access to a Windows system, at the first opportunity I install the most basic vital fundamentals:

1. Firefox (browser) `http://www.mozilla.com/en-US/firefox/`

2. Thunderbird (email) `http://www.mozilla.com/en-US/thunderbird/`

3. 7zip (a package opener for zip and tar.gz files)`http://www.7-zip.org/`(*)

[1] because it is a closed source operating system that is expensive and over which I have very little control or ability for input

4. File Transfer Programs

    a) WinSCP (file transfer program capable of secure shell transaction). `http://winscp.net/eng/index.php`(*)

    b) FileZilla (same) `http://filezilla-project.org/` (*)

5. Putty (terminal emulator: just download and drop putty.exe in some folder that is in your path. Then run "putty" in the "Run" prompt or command box). `http://www.chiark.greenend.org.uk/~sgtatham/putty/`(*)

6. Acrobat reader.`http://get.adobe.com/reader/`

7. Flash Player.`http://www.adobe.com/products/flashplayer/`

8. LyX (a LaTeX interface, which installs MikTeX, a LaTeX processor).`http://www.lyx.org` (*)

9. R `http://www.r-project.org`(*)

10. Emacs (including the Emacs addons like ESS and AucTeX). Don't install the ordinary Emacs. Get the one that Vincent Goulet has customized: `http://vgoulet.act.ulaval.ca/ressources/emacs` Currently, the version is 22.3-modified-2. (*)

11. OpenOffice. This is a free replacement for Microsoft Office.`http://www.openoffice.org`(*)

12. Gimp, a great image editor. The Gimp homepage is here: `http://www.gimp.org` and there is a separate site where you get the Windows installer: `http://gimp-win.sourceforge.net/stable.html` and a separate package of help documentation.

(*) I'm making my collection of free Windows programs available. These are just things I've collected, not things I've built. I've also posted a few scripts. Look here for information

`http://pj.freefaculty.org/Software/WinProgs`

Installation Warning. Avoid spaces in file names wherever possible.

I install things into C:\ProgramFiles – note there is no space. The space causes all kinds of trouble when you start finding your way around in your computer.

After you have installed the basic programs, then it is time to turn your attention to the programs that are specially tailored to the research experience.

1. Install Emacs, the bestest editor ever!

   When I get a new MS Windows computer, I install Emacs first. Emacs is an old text editor program favored by hard core geeks like me. More beautiful editors are proposed all the time, but I always come back to Emacs because, well, it is always there for me. I've found a nice new version of Emacs is packages by one of the participants in the ESS email list. That version includes ESS and other components for editing LaTeX documents.

   Run Emacs from the Icon or start menu or by running the name of the program inside the DOS Command box. Recently, the program name seems to be "runemacs.exe". Put the full path together, and run it from the box.

   After you learn about Emacs, they you might want to customize it. A file I provide,

   ```
   emacs-ess-kups.el
   ```

   can be inserted into the Emacs file system and it will customize the start-up of emacs. Our computer lab uses this package.

   On the Windows platform, there are many other text editors that would probably do as well. Emacs has some special features that facilitate the use of R, but I've seen many people endorse a text editor called TINN-R and others.

2. Install R. From http://www.r-project.org

   I do not let R install itself in the default location. It proposes "C:\Program Files\R\R 2.6.1". I cut out the spaces, and cut the last part altogether. I set the install target as C:\ProgramFiles\R and so when R installs itself, it creates subdirectories C:\ProgramFiles\R. I do this because I do not want to allow for the possibility (and confusion) of installing several versions of R at once. I want R to install its executable programs into C:\ProgramFiles\R\bin. The executables include Rterm.exe, which is a plain R terminal that is similar to running R in a Linux terminal, and there is also a program that launches a graphical interface.

   After the install of R is finished, start it up and do some testing. Check what packages you have installed. Try to run "update.packages()". If it runs fine, great. You can consider installing many more packages. I have an R program called R_installFaves-2.R that has a long list of packages and in the computer lab, we run that program to try to make sure every system has everything that anybody wants. You can run that too on your system. Or you can study it, edit it to delete some packages, and then run it.

3. It may be that you have to customize Emacs so it knows where your Rterm.exe will be. There is a file in the ESS installation (under Emacs) that controls how ESS operates. That file is called "ess-site.el". In the version of Emacs that I recommend, the user's attention is drawn to this file by an item in the Start menu that is in the Emacs group. Note, when you customize that file, the path has FORWARD slashes.

4. Install the big LyX windows bundle. That installs the LaTeX graphical interface (like a word processor) called LyX as well as several other vital programs. Those are MikTeX, a latex publishing system, GhostScript, a program for creating and revising postscript files, and ImageMagick, a program that once called itself the Swiss Army Knife of Image Manipulation. Let the LyX installer put in those programs. No customization is needed.

5. I have been eagerly looking for a program with which to create line art in Windows. Preferrably, it would be able to create encapsulated postscript files as well as a format that can include mathematical symbols in a way that is consistent with LaTeX. In Linux, the program for making line art is xfig, and there have been efforts to create a Windows version.

   Xfig is a very handy for making line drawings that can be included directly in LyX documents. It is a huge time saver. The version of xfig I use in my windows systems is based on the Cygwin Toolkit, but that may be a difficult install for the ordinary Windows user. Cygwin is a full Linux style environment that runs within Windows, and it has many strengths and weaknesses. Rather than install Cygwin, I'd guess many new users might consider WinFig as an alternative. Find about that at `http://homepage.usask.ca/~ijm451/fig/winfig.html`.

3 Networking

Why use a network? Here are some reasons.

1. Some programs exist on remote servers, so it is necessary to transfer files to and from those systems. Most people don't run Web servers on their personal computers, so files must be uploaded to a server.

2. Users on mobile systems or in public laboratories may not want to lose their work if/when machines die, so storage on a permanent, separate, free-standing network may contribute to the user's sense of mind.

3. Users who have adequate desktop file storage may wish to "mirror" their files so that they, or others, can access them.

In short, researchers in the new era of Internet computing need to be able to ship files around, so as to share their data among systems and between research teams.

Everybody who uses a personal computer has to be a user and a system administrator (or hire someone to administer the system). The users run programs, write papers, and so forth, and the administrator installs programs, configures networking of shared services, and so forth. A personal workstation may run a program that shares files to the rest of the users in a local network or to the world. (Most teenagers found out about that in the late 1990s. Remember Napster?) As a result, there is no clear dividing line between a hardware that you might call a "personal workstation" or a "server."

3.1 Secure File Transfer

"Telnet" was the old standard for accessing one system from another. The old standard for shipping files from one computer to another was the "file transfer protocol" (ftp). Neither of these was secure–your password was sent in the "clear blue" (unencrypted) across the internet.

Today, telnet is generally not running on well administered systems and neither is ftp. Instead, programs that access systems through the "secure shell" protocol (ssh) are necessary. At the current time, we are using version 2 of the ssh protocol.

3.1.1 Graphical File Transfer Programs

In the "old days," ftp was the standard program for copying files from one system to another. However, it was an insecure program, and now we are copying files across systems using secure protocols, mainly "secure shell" (SSH). A remote system has to be running a secure shell daemon and, generally, those systems will "listen" on port 22 for secure connections. Systems may refuse connections if their administrators are security conscious; it may be necessary to contact the administrators to find out how you are supposed to communicate with their system.

gftp and other descendants of WSftp

If you are just geting started, perhaps the easiest way to begin is with one of the "two pane" graphical file transfer programs. Generally, these systems will provide a user interface that lists two systems, side by side, and it will generally be obvious how to transfer files from one system to the other. Some programs allow "drag and drop," while others offer arrows to indicate the direction of transfer.

On MS Windows, a very famous program called WSftp was introduced in 1995. It created the paradigm for programs of this sort that is now widely followed. On Linux systems, I tend to prefer to use the program "gftp" to access other systems, while in Windows systems I prefer one called "WinSCP". Many people I work with like the transfer program called Filezilla, which is available for Linux as well as Windows. At the current time, I prefer WinSCP on Windows because it is the only program that allows full control of the ownership of files on a remote system.

A screenshot of gftp is presented in Figure 3.1. Before I hit the "icon" of the 2 computers to start the connection, I type in I set

- the name of the remote system
- the user name
- the password
- Choose SSH2 as the protocol.

After setting those values, I hit the 2 computers icon and the connection is created.

The screen shows 2 directories. The left side of the screen shows the current working directory of the system at which I am sitting. The right side shows the remote system. The arrows indicate the direction of file transfer. After you experiment with this a bit, you will see it is very intuitive.

Pay attention to the small box in the bottom of the gftp display. It shows the status of the interaction between the two system. When things go wrong, that is the place to look.

Many of these graphical secure file transfer programs try to enrich the user experience by allowing user interaction with files on the remote system. All of these programs, for example, allow the user to select a file on the remote system and delete or rename it (usually a right-click menu is offered). There is usually some kind of simple menu or "shortcut" to change the attributes of files. A "right click" menu example is illustrated in Figure 3.2.

There is one other special feature of gftp that works very well. Suppose that one wants to revise a file on the remote system. Ordinarily, it would be necessary to launch a terminal, go onto that remote system, and edit the file. That may be inconvenient. Many of these graphical file transfer programs implement a way to "transparently" edit a file on a remote system. In all of these programs, this will not work without some customization. There will be an options or preferences menu in which one specifies a program on the user's workstation that will serve as a text editor. I always designate Emacs as my editor (runemacs.exe on Windows, emacs on Linux). This configuration in gftp is displayed in Figure 3.3. After you tell gftp about which editor to use, right click on the file in the remote computer and choose the "edit" option. That file will be downloaded to your local system, and it will be opened in Emacs. After you finish editing and try to save the file and close Emacs, then the gftp program will notice the change and it will ask if you want to transfer the result back to the other computer.

3.1.2 Synchronizer programs: Unison and others

Unison is a graphical program that is used to synchronize file directories. I use this to keep my research and course directories up to date across several computers. It is smart enough to look at both systems and see which files have been modified. Unison only works if it is installed by the administrator on both systems. Unison is available for MSWindows as well as Unix/Linux systems.

When the Unison program starts, the user is presented with a list of profiles that have been created. In my example displayed in Figure , you will see I'm synchronizing from my laptop, the "localhost," to the desktop computer in my office, "pols110.pols.ku.edu".

After you setup your profile to compare directories on 2 systems, then Unison will scan both systems and create a database. The database is used to see if files have changed and it tries to guess which version is "best" for you. The suggested transfers are represented by arrows in Figure 3.4. I prefer to start unison with the -t option, so that the file modification times are kept the same across the 2 systems. Otherwise, Unison will set the modification time to the current time on the system onto which the system is being transferred.

Unison has the interesting quality that it can work either across 2 computers (via an ssh link) or across 2 directories on a single computer. If you have a network file system mounted in one location, you can use Unison to sync a directory in your home with the server.

Unison or other programs like it generally will require some configuration on the remote server system. If that system's administor is not cooperative, then we have to find some

Figure 3.1: Screenshot of gftp

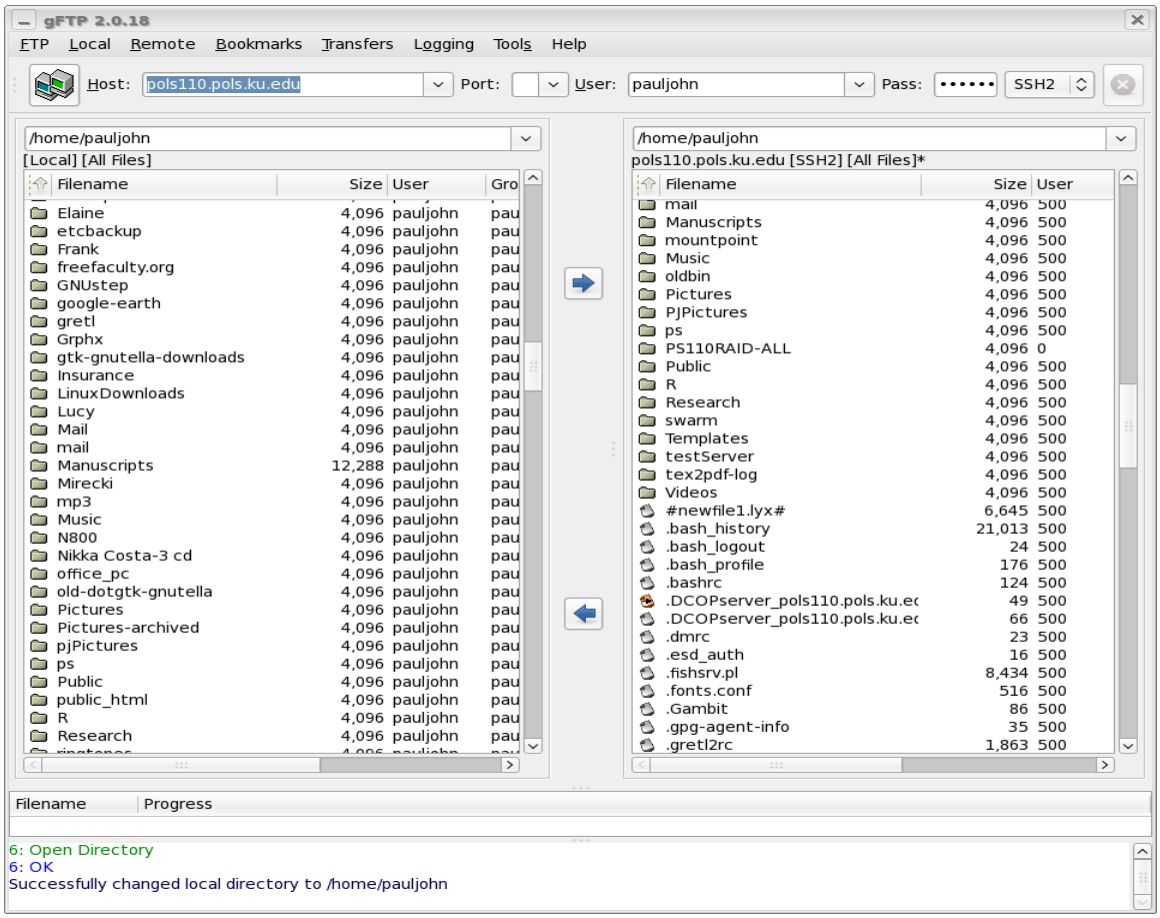


Figure 3.2: Right Click Menu in gftp

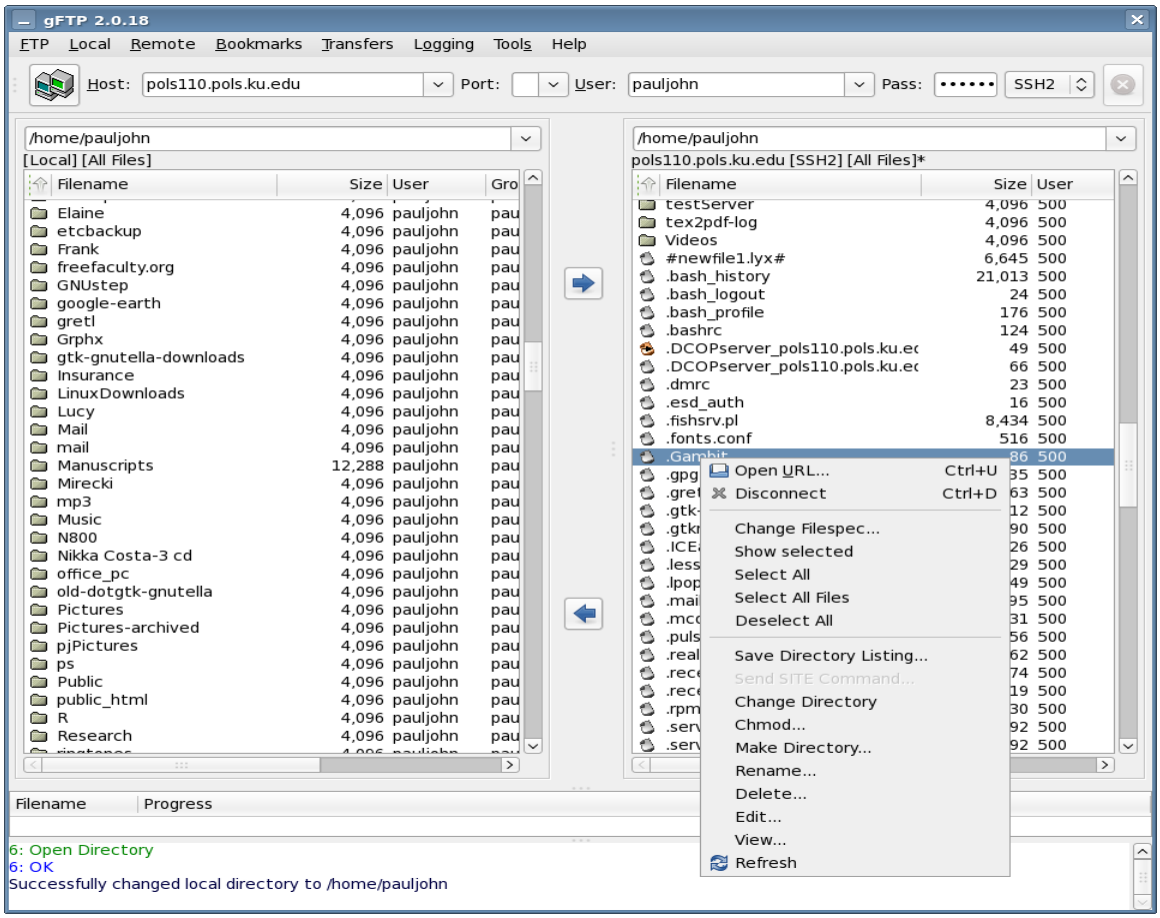


Figure 3.3: Editor Configuration in gftp

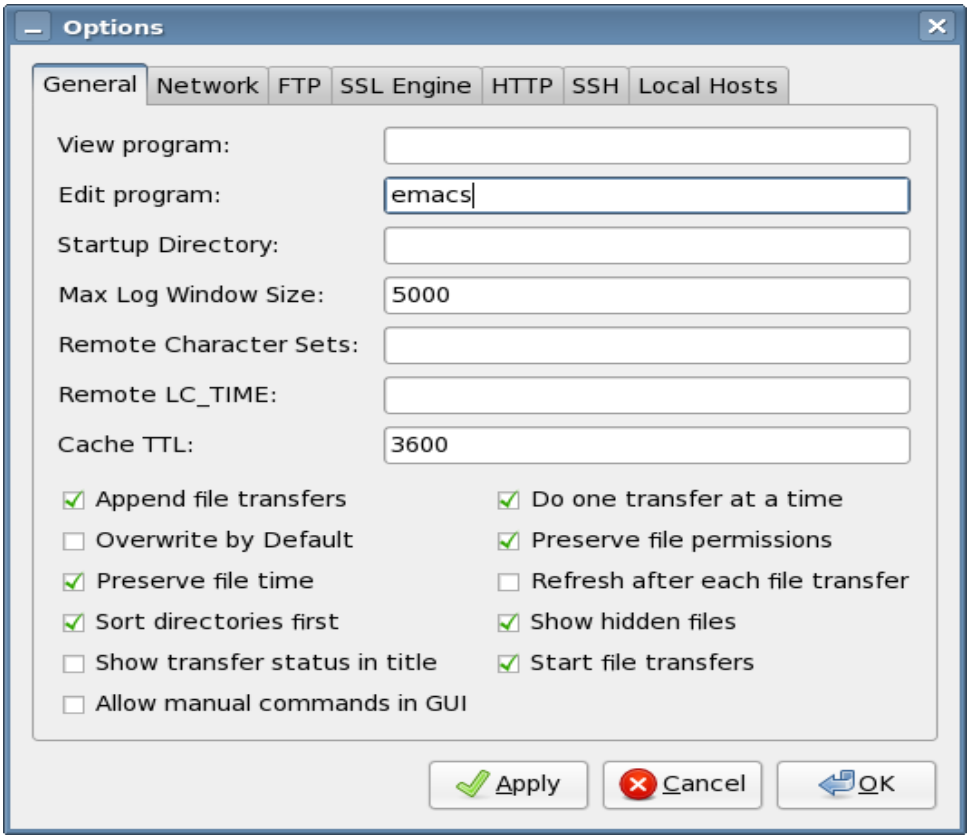


Figure 3.4: Unison Startup

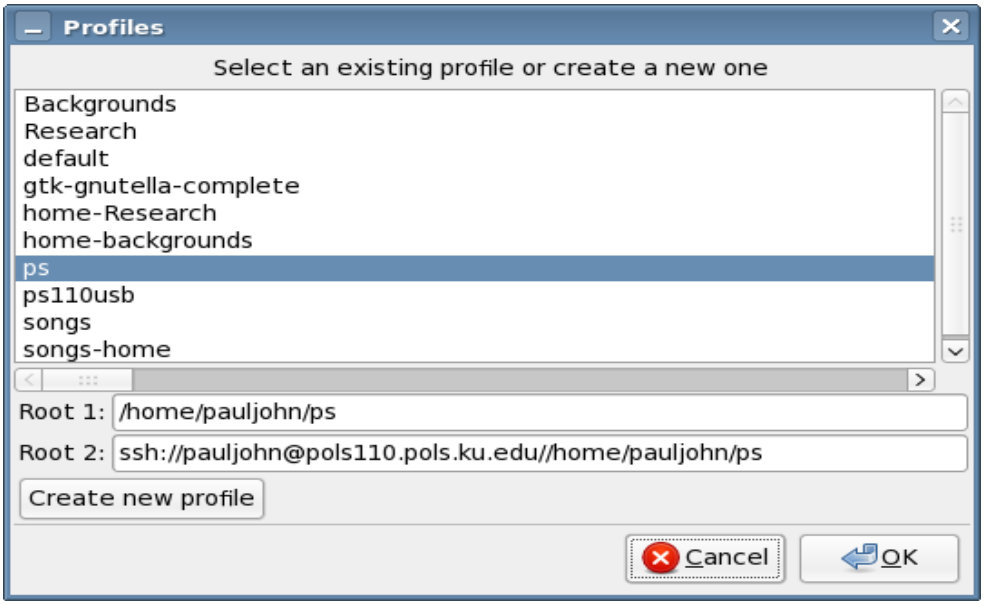
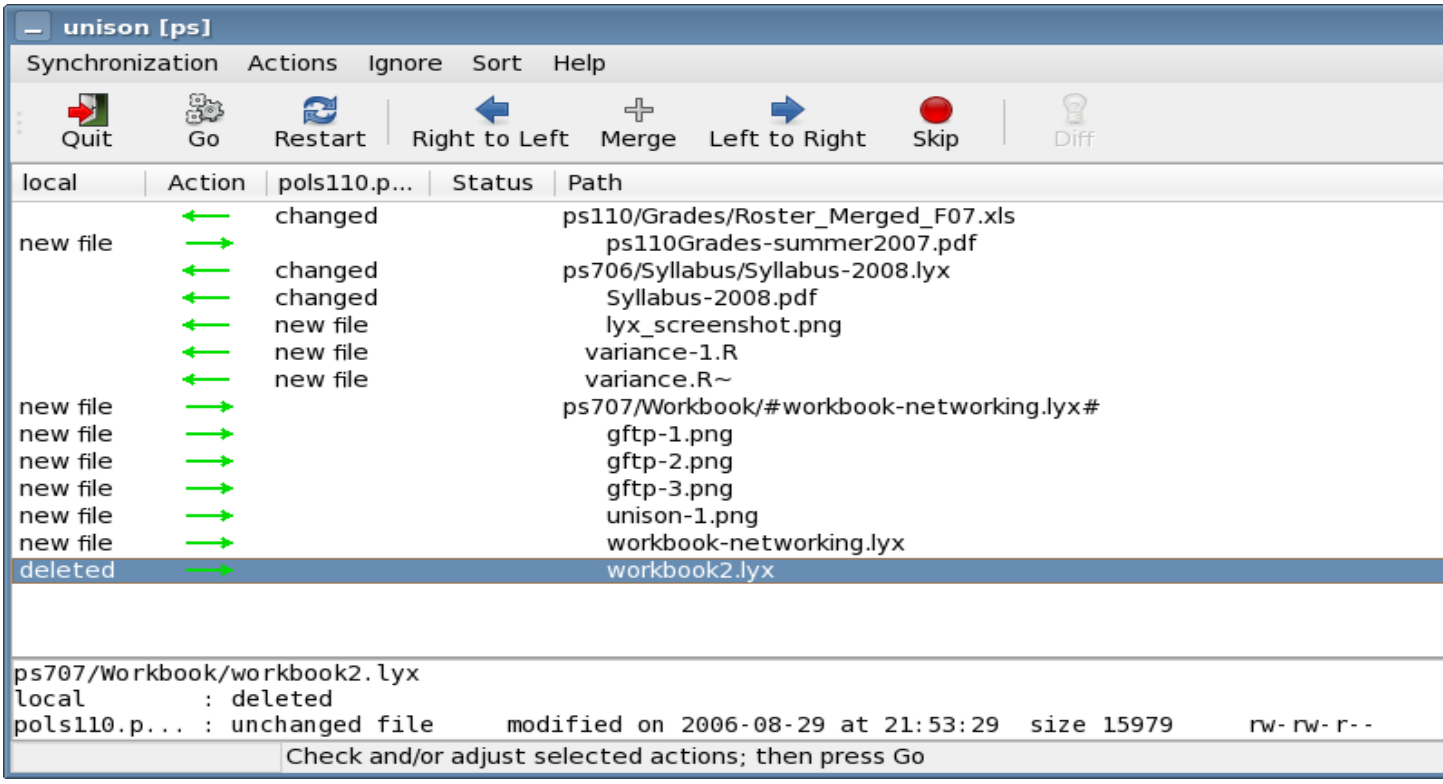


Figure 3.5: Unison Compares 2 Directories



work-arounds. A program that exists only on the user's system will have to get all of the work done. On the MS Windows 7 computers in our research laboratory, we have experimented with many other file synchronization programs. Many users were happy with a commercial product Alway Sync, but they were not eager to pay the license fees. A free program called DSynchronizer has been a reasonable substitute. It uses the graphical interface known as "GTK+", which must be installed separately. Generally, I recommend that users should install the free software editor called the Gimp, for which GTK was originally written. After the Gimp is installed, then DSynchronizer works fine.

3.1.3 Command Line File Transfer.

I get tired of pointing and clicking. The point-and-click programs like gftp and unison are nice, but they are somewhat tedious to use. There are equivalent programs that you can use from the command line that are very handy. In the linux terminal, the two file transfer programs I use the most often are "scp" and "rsync".

scp stands for "secure copy", a secure version of "cp" that works across systems. It is based on the SSH technology. Examples.

scp x my.pc.com: Copies file x into your home directory on "my.pc.com". Don't forget the

semicolon. If your username is different on my.pc.com, include your user name, as in: "scp x pauljohn@my.pc.com:"

scp x* my.pc.com: Copies all files that begin with x to "my.pc.com." * is a "wildcard"

scp x* my.pc.com:whatever/some/dir/is will drop all files that begin with x into a subdirectory ∼/whatever/some/dir/is if that directory exists. Error is returned otherwise.

scp -r pauljohn@my.pc.com:whatever . Copies a directory ∼/whatever into my current working directory (".").

scp -r dirx 129.237.61.11:whatever it will copy an entire directory dirx and all contents onto the machine with the IP number 129.237.61.11 into the directory ∼/whatever. If a directory whatever does not exist, this command will fail.

Rsync is a smarter program than scp. Like scp, it does "one directional" updates from one system to another. But it is smarter in that it saves bandwidth and avoids extra work. If a file is the same on both systems, it will not be transferred. And if a file is changed, an efficient algorithm is used to only tranfer the changed part of the file. If the –delete option is used, then files that are missing on the source system are erased on the target. In a sense, rsync is a little frustrating for synchronizing 2 systems because one would have to run 2 commands in order to completely synchronize 2 systems. One command would copy from the local system to the remote, and another to copy in the other direction.

Rsync is one of the few Linux programs that has a truly helpful manual page with great examples for beginners. In the terminal, run "man rsync" and you'll see what I mean. Use "q" to get out of that man page. The -e ssh option is necessary because it tells Rsync to use a secure shell. Most systems will refuse the insecure alternative.

rsync only works if it is installed by the administrator on both systems. Here are some example usages.

```
rsync -e ssh -rav directory_named_x  my.pc.com:
```

"r" stands for recursive, "a" stands for archive mode (preserving permissions and modification times), and "v" means print out a verbose report on file transfers. This copies my "directory_named_x" and all files inside it to the remote system in my home directory. The files end up in the home directory because I left a colon, and nothing else, after the name of the targett system. The files will end up in "/home/pauljohn/directory_named_x". When I do not put in a user name, it assumes the same user name on both systems.

```
rsync -e ssh -rav directory_named_x pauljohn44@my.pc.com:
```

On my.pc.com, my account is "pauljohn44", but on the local system, it is "pauljohn". So I have to give my account name.

```
rsync -e ssh -rav directory_named_x my.pc.com:whatever
```

This command updates from the current system's "directory_named_x" and it copies the whole thing and creates a subdirectory under the other system called ~/whatever/directory_named_x. R

```
rsync  -e ssh -rav my.pc.com:directory_named_x .
```

The period at the end means "current directory." This copies from "my.pc.com" to your current working directory.

```
rsync -e ssh -rav my.pc.com:directory_named_x/ .
```

Caution. Adding the / has a big effect. This will copy the contents to "." but it will not copy the directory name itself. That can be really bad. You want to grab a bunch of files from a directory_named_x, and you want them inside a directory. If all those files are just "splattered" over your current working directory, you have a mess on your hands.

The use of the slash at the end of the directory names is the worst "gotcha" in rsync.

Why is caution important? Well, I'll give you an example of what went wrong for one of my students. He once ran

```
rsync -e ssh -rav ps blake01.clas.ku.edu:
```

That successully copied the "ps" directory so that it ended up in his home directory on blake01.clas.ku.edu. On blake01, his files would have been saved in "/home/student/ps". Then the student returned to another session, and could not remember how to backup the contents of his ps directory. So he did this:

```
rsync -e ssh -rav  ps  blake01.clas.ku.edu:ps
```

Too bad. That created a new sub directory "/home/student/ps/ps" and it put a copy of ps inside ps.

Instead of that mistake, he could have done either of these to update the ps directory:

```
rsync -e ssh -rav ps/ blake01.clas.ku.edu:ps
```

or

```
rsync -e ssh -rav ps blake01.clas.ku.edu:
```

3.2 Logging in to a system remotely with Secure Shell (SSH2)

The whole point of the secure shell is that a user can open a terminal on a workstation and "go over to the other system" and have a terminal that works "as if" the user were directly connected to a system. Several years ago, my university's system administrators refused to allow me to run a Web program that required a MySQL database. As a result, I migrated my Web programs to a commercial service called Dreamhost.com. They serve up my Web pages and I can log into their systems by using a secure shell "terminal."

For Microsoft Windows computers, there are many competing terminal programs, some of which are expensive, others of which are free. The Putty program has always worked well for me and I install it in all of the Windows systems that I administer. When Putty opens, it appears as a black-and-white computer terminal would, except it is inside a small windows. On Linux or Unix systems, one can generally open any terminal program, such as xterm, gnome-terminal, Eterm, or MLTerm (or any of the 100s of terminal programs) and then run a program to log into a remote system. The program that actually runs the remote secure login is called "slogin", and one can use it like so:

$ slogin bigbopper@freefaculty.org

Many users had trouble remembering that the right command was "slogin" and so most systems also allow an alias, "ssh" to go from one system to another. That is to say, I usually type this instead

$ ssh bigbopper@freefaculty.org

because it is easier to remember.

After the secure connection is established, I can interact with the remote system "as if" I were in a terminal physically connected to it.

It seems obvious to me that most readers will realize the benefits of the ssh connection. They can go to a remote system and configure their Web pages. Because the connection is secure (encrypted), it is unlikely that people who monitor the wires between systems will be able to steal usernames and passwords.

There are some other scenarios that call for the use of an ssh client. Perhaps these will bring the point home.

Help! My computer is frozen

On my laptop computer, I have the Open SSH service running (which probably means my laptop is a server). Sometimes I get busy running programs and cause my laptop keyboard

to "lock up." It used to happen quite often, now it only happens about once per year. When the laptop is running, but unresponsive, the last option is to hold the power button down for several seconds in order to make the system turn off. The moving parts inside the laptop–especially the hard disk–don't like to be turned off so abruptly. There may be files that are lost or corrupted by the sudden shutdown.

When the laptop stops responding, I have learned that there is a work around. I can go to another computer and remotely log into my laptop. Supposing I know the IP number of the laptop, it is as simple as opening a terminal and running

$ ssh 192.168.0.1

For reasons I don't understand, the system will still answer the secure shell request, even though the keyboard is locked up. Once I'm logged in, I can use various commands to halt the misbehaving programs or initiate an orderly shutdown and restart cycle.

The "my files are on that other computer" emergency

Suppose you are using one computer and you save some files. You should copy them to a networked system so you can get them from anywhere when you need them. But you forget.

Lets consider an example network in which the machines have IP numbers 192.168.0.1 through 192.168.0.100. That last IP number is the one we will call the "server" system, it has the most hard disk space. All of the other systems are smaller, with ordinary desktop storage, and we refer to them by the last digits of their IP. Researchers may log in on one system and get a lot of work done, but they forget to transfer it onto the server. When they return to the lab, someone else is using that particular computer. In a lab of Windows workstations, that would be a horrible problem, we would have to ask the other user to log off in order to retrieve those files. But, since we have Linux systems that have the secure shell servers running, it is easily managed.

The researcher should go to an available system and log in. Suppose the vital information is stored on a system at 192.168.0.13 and the user is logged in at 192.168.0.18. If the user has a pefect recollection of the files that need to be retrived, this will be easy. A command like rsync will transfer a folder "MyVitalDirectory" from "13" to "18."

```
$ rsync -e ssh -rav 192.168.0.13:MyVitalDirectory .
```

The period at the end of the command means the "current working directory" on system "18."

What if the user's memory is not so clear. He doesn't know the names of the files that were saved on "13," so he has to go to "13" and do some inspection. This is possible, even while that other researcher is using system "13." From system "18", we can log into system "13" like so. Open a terminal and run the command

```
$ ssh 192.168.0.13
```

That assumes the username is the same on both systems. If it is different on "13," then the right command is

```
$ ssh other-user-name@192.168.0.13
```

The secure shell system will generally ask for permission to make a connections. We answer "yes" (in case there was any mystery).

After that succeeds, one is logged into 192.168.0.13. All commands like "ls" and "cd" and "cat" *as if* the user were sitting in front of system "13."

Once we find them, there are many ways to retrieve the files. Here's one possibility. While logged into "13", one can use a command line transfer program to send the files "back" to the system where the user is currently logged in. For example,

$ rsync -e ssh -rav MyVitalDirectory 192.168.0.18:

will copy the folder called MyVitalDirectory from the current location onto the other system.

It might be smarter to transfer the files to a safe place. Before we do that, lets start another terminal on system "18" in order to monitor what's going on. In that terminal, we want to open a secure connection with the "server" system like so

$ ssh 192.168.0.100

So there should be two terminals sitting side by side, one is displaying system "13" and the other is displaying "100". In the terminal that is displaying 13, copy the files onto the server system.

```
$ rsync -e ssh -rav MyVitalDirectory 192.168.0.100:
```

In the other terminal, the one displaying system "100", type "ls -la MyVitalDirectory" to verify that the files are appearing there. When the work is finished, the command "exit" will close the remote shell.

3.3 Graphical programs in the remote shell

The graphical display on a Unix/Linux system is usually presented by a program known as X11 that was developed at MIT in the days long before Macintosh or MS Windows existed. An "X server" takes messages from programs that you are running and it displays them as the familiar "windows". There is a free X server for Macintosh that is distributed, but not

installed, with the operating system disks. On Windows systems, there are competing X11 servers. We have good results with the free X server called "Xming."

One of the very special properties of the X server framework is that, when you are sitting in front of one computer and looking at its monitor, then it is possible to see programs that are running on other computers. It is "as if" you are running a GUI program on your current workstation, but actually it is running on another one.

On the systems that I configure, the SSH protocol configuration is set so that graphical interface programs are "forwarded" from one system to the other. When you log into another machine, and run a GUI program, then it will be displayed back to the terminal at which you are sitting. Open a terminal, and this time add the -X option to the ssh command.

```
$ ssh -X 192.168.0.14
```

The -X option makes it clear that you want graphical programs from system "14" to travel through the "ssh tunnel" and appear on your workstation. After you log in, type

```
$ emacs
```

Assuming that the X server is running on your computer, the graphical display of the Emacs program will be forwarded back to your current system. The work you do on the screen is saved onto the computer with the IP number 192.168.0.14.

If you do some work on one computer, and then you return to find someone else using "your computer", it is pretty obvious what you ought to do, isn't it? Log into an available system, and then connect up to the one that has "your files."

And then you should learn to copy "your files" onto a server somewhere so they will always be safely accessible from whatever system you use.

3.4 Network File Shares

Network File System. Files can exist on a remote system but appear "as if" they are in the computer in front of which you are sitting. Such files are part of a network file system.

Background. Some files are in your hard disk. Some are on memory sticks. Some are on servers in other buildings. The Linux philosophy is that we try to "attach" these resources in a way that is transparent to the users. Some servers allow their "shares" to be "mounted" so they appear to the user "as if" they are actually on the hard disk. There is no standard place where networked files will appear in the directory structure. Some servers will allow users to mount "shares" with particular commands, some will require that an administrator do it.

There are many competing formats for networked file systems. In the earlier versions of these notes, I have had instructions in more-or-less detail for connecting to Novell file servers, Samba file servers, and NFS file servers. It seems as though the folks who run these systems in my university decide they need to destroy them and get something else as soon as I learn how to use them.

As a result of that problem, I've come to realize that it does not do much good for me to write down a lot of detail about systems that may not exist next month or at any other system. Nevertheless, I can convey some benefits of interacting with these file server systems.

The networked file system concept can arise in two ways. First, which I would call "mobile user paradigm", has an administrator set up a file server and then design the individual workstation computers so that they seamlessly read and write on the central file server. That is to say, user "home" folders actually exist on the file server; there is no local hard disk that saves the users "home" folders. The Sun workstations at the Santa Fe Institute were set up that way. On any workstation, one could log in and all of the files, folders, and settings were instantly available. We have experimented with that in the political science department, with varying degrees of success. On a Windows 2000 server, we had a similar setup that was referred to as the "roaming user" concept. When the network does not respond quickly, or when the server is down, reliance on that kind of centralized storage may be unwise.

A less intrusive strategy is to allow the local hard disks to keep the user home information, but also configure a server so that users can easily copy files onto the server. Generally speaking, the term "mount" means that we take a remote file share and make it appear on the user's system as if it were a local hard disk. In an MS Windows computer, the network computer icon will usually have a right-click option "map network drive". As long as the network server offers a protocol that the user's workstation can manage, then access to the networked storage can become almost seamless.

I say "almost seamless" because mounted file systems may fail to record the ownership and permissions for files properly. A Windows file server generally will label all files as executable, but a Unix or Linux server will not. Bringing the operating systems of the file server and the workstation into complete harmony is a serious project.

Generally speaking, it is safest to use a protocol that is most similar to the file server's operating system. If a file server is a Linux system, one may have the choice of mounting its shared folders either as ssh shared file systems or through an MS Windows emulation known as a Samba (or CIFS) protocol. If your workstation has the tools installed to acces the files directly through the ssh shared file system (often called sshfs), it is more likely that the transfer of files between the systems will be trouble free. On the other hand, if the file server happens to be running MS Windows, the Samba (CIFS) protocol file service is the closest to the native operating system, so I would use a Samba client to mount the shares.

3.5 Put things on a web page

If you have an account on a web server, may be possible to upload files in a variety of ways. At the University of Kansas, our administrators created a Linux server that they called people.ku.edu and students who want to create web pages have the option to do so on that system. The preferred software for serving web pages at the current time is the Apache Web Server, and the system administrators are allowed to control the locations from which users are allowed to offer files. On a tightly administered system, the administrator may keep tight reigns by insisting that all files to be offered on the World Wide web must be stored in a folder /var/www/html and only the administrator is allowed to write files there.

On most systems that host Websites for more than a few users, the administrators have to be a bit more generous. At KU, as in many systems, the users are told to put their web materials in a folder called "public_html" in their home directories. Users then have the responsibility to set permissions on that folder so that the web server has permission to read the files. Depending on the generosity of the system administrators, those permission issues may have already been taken care of. But the chances are good that the user will have to take some steps to make sure that permissions are correct. If the user doesn't have public_html, he shoudl create it. All directories to be offered on the web have to be readable to the world. The shortcut to set those permissions is

```
$ chmod  755 public_html
```

Transfer files into that subdirectory with any tool you like (scp, rsync, gftp,...).

Then open a web browser and go to http://people.ku.edu/~your_user_name. If everything is set up correctly, your content should be presented.

Administering a Web page is, in some ways, very easy. I wrote some notes about setting up web pages in the early 1990s and they have held up over time surprisingly well. The directory contents of my html advice is still available, `http://pj.freefaculty.org/misc_support_docs/htmlhelp` The bare bones of this are still, roughly, the same as they were in 1988 when I started putting up Websites. The fine points and fancy details are generally optional.

One of the Web server features I like is automatic listing of directory contents. If I want to make files available for download, I transfer them into a folder on my Web server and then people who browse the directory automatically see the files. If the system administrators have turned off automatic directory listings, then it is more tedious to make files available because it is necessary to revise a web page to actually list the files and link them for the users. One of the reasons I don't use the university's web servers anymore is that they turned off automatic directory listings.

4 Starting With R

I'm going to suppose that R is installed in your system. Lets suppose also that you have the Emacs text editor installed. If those assumptions are incorrect, well, get to work. I have a collection of tips on setting up Emacs and ESS on the Emacs Wiki web site:

4.1 How to Run the R Program

4.1.1 Interact Directly with R (if you don't do anything important)

You can start R in many ways. Generally, for light work that will not need to be replicated, one can start R and directly interact with it. That is to say, open an R terminal and type commands into it.

In Linux, the name of the R program is "R", and generally, if one opens any terminal program and types "R", then R will appear inside that terminal. For the Linux OS, there is no widely accepted point-and-click graphical interface. MS Windows system, the terminal version of R is called "Rterm.exe." In a DOS Command Box, one can type "Rterm". If "Rterm.exe" is not in the system path, then it is a bit more difficult, since it will be necessary to type the full path to Rterm.exe. On the other hand, one can use the Windows Explorer to go into R's bin folder and double-click on Rterm.exe.

In Windows or on the Macintosh, the R packagers have done quite a bit of work to make the program more familiar to users. In R, the graphical interface is opened if the user runs the program called "Rgui.exe". The default R install leaves a desktop R icon that points toward Rgui.exe. That's basically an R terminal, with a few pulldown menus for managing packaging. There is a very primitive text editor offered with the current R program. It is not nearly as nice as Emacs for editing text, but it does work. On the Macintosh, on the other hand, R is delivered with quite a nice editor.

4.1.2 After You Log In, Try this first.

Most people will only run R in a terminal once or twice. Nevertheless, it is the way most of us get our feet wet. After R starts, the user will be prompted with ">". (Don't re-type the ">" when you follow my instructions.) Type some commands, such as

```
> library()
```

to get a listing of all installed packages. Then why not generate some random variables and create a plot or two:

```
> x ← rpois(1000, lambda=43)
> hist(x, main="My First Histogram")
> y ← rnorm(1000, mean=32, sd=10)
> plot(x, y, main="My First Scatterplot")
```

It is a good idea to stay up to date. If you are logged in as an administrator (in Windows, start R with a right-click menu "Run as Administrator"),

```
> update.packages()
```

To install a new package called lmtest, a person would run

```
> install.packages("lmtest")
```

4.2 Understand the Help system within R.

Use the help facility to read about linear regression with

```
> ?lm
```

Run the examples for lm

```
> example(lm)
```

Run this to fire up a web browser that displays R's online documentation:

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

Note that help.start() lists all of the packages, as well as displaying manuals. After running help.start(), what happens when you type

```
> ?lm
```

Guess what help.search does?

```
> help.search("regression")
```

If you are on the internet, watch this!

```
> RSiteSearch("regression")
```

4.3 The beauty of (). Or, "Yes, Virginia, you really do need those empty parentheses!".

Try to quit. What happens if you type try the obvious thing:

```
> quit
```

That did not quit. It printed out some crap on the screen. What is that stuff? It is the code of the function "quit". It is the script that R executes when you quit. There is a help page for the quit function. Until I started writing these notes, I had never run the command "?quit". But you might, for fun.

In order to tell R you really want to exit, do this:

```
> quit()
```

or

```
> q()
```

I almost always say N when it asks if I want to save the workspace. Shortcuts are allowed. q() works as well as quit().

The parentheses–()–are needed, even if you do not put any options in there. The parentheses are the way in which you tell R that you are using one of its functions. To use R functions, even if you do not want to set any options, it is necessary to include parentheses.

Why did R print code when you ran "quit" before? Well, typing the name of a thing in R implicitly calls the print() function. If you run

```
> x
```

R receives your command and shows you the output you would have gotten if you ran

```
> print(x)
```

When you run "quit" with no parentheses, R responds as if you had run

```
> print(quit)
```

4.4 Start R in your Working Directory.

It is VITAL for you to know "where you are" when you run R, because R needs to know where it can save files and where it should try to open files.

Different operating systems will ofter ways in which to create working directories and initiate R from within the right one. On Linux, I prefer the old fashioned approach. First, open a terminal. Second, create a working directory if it does not already exist.

```
> mkdir -p ps/stat/ex_4.4
```

Next change into that directory, so that it will be the working directory when R starts.

```
> cd /ps/stat/ex_4.4
```

After that, when R is started, I make sure it sees the working directory that I intend for it to use.

```
> getwd()
```

That asks R "where are you going to save output files? Where are you going to look for input files?"

One can manually set the working directory for R after it has started with the setwd() command. For example, if a terminal's current working directory is /home/pauljohn and I start R, but I really wish to use files that are stored in /home/pauljohn/ps/stat/ex_4.4, then I can run the command

```
> setwd("/home/pauljohn/ps/stat/ex_4.4")
```

The following command will create a list of files that exist in the current working directory.

```
> list.files()
```

This is very useful is one needs to verify whether a data file is already available in the working directory.

R will find files in the current working directory. If one wants R to access files that are not in the current working directory, then one must specify the full path to the files when. That's a difference between typing a single word, as in

```
> dat <- read.table("secrets.txt", header=TRUE)
```

and a long string of symbols, some of which you may get wrong:

```
> dat <- read.table("/home/pauljohn/ps/stat/ex_4.4",  header=TRUE)
```

All users should develop a directory-naming scheme and use it consistently. Subdivide projects in order to keep good records of what has been done and also to make it easy to share all files that are relevant with a project with another user.

For these purposes, as well as easing the creation of backups, I STRONGLY ENCOURAGE all students to start a directory in their HOME directory called "ps", and under that create subdirectories for different classes and projects. I encourage that because it makes it easy to find the things that need to be backed up. I wouldn't think this point should be made so emphatically, except that when I sit with students in order to help with their projects, we end up spending a good deal of time trying to figure out where they have stashed their files.

4.5 How I Usually Run R: Inside Emacs.

The editor I use is called Emacs. That stands for "Editor Macros." It has evolved from the early days of Unix terminals into a more-or-less modern graphical user interface. I recommend that all students should learn to use Emacs for the following reasons.

1. Emacs is available on just about any kind of operating system. Emacs is available for Linux, Windows, and Macintosh. As a result, one is not tied to a particular operating system by the selection of a specialized editing program.

2. Emacs has plenty of tools to make it easier to write good code. One of the strengths of Emacs is its "extensability," meaning that it is open to the addition of components. The addon component that I use for R is called ESS, "Emacs Speaks Statistics." It has handy "color highlighting" for code syntax along with context-sensitive indentation.

3. Because of its multiplatform availability, long-term proven track record of volunteer maintenance, and the wealth of addon packages for specialized projects, I believe that Emacs will always exist. And, because of the licensing, it will always be open to innovation and it will always be available free of charge.

4. In addition to co-operating well with R, Emacs is also a great development environment for other kinds of programs. People who program in R with Emacs can just as well write LaTeX or C with Emacs. Emacs has great tools to compare files for changes, for example, and it also is integrated with the GNU debugger, "gdb".

5. Emacs can be customized to behave like other point-and-click programs. In my opinion, Emacs has a reputation as a "difficult" program because the proponents usually try to teach us how to use it as if we did not own mouses or have a menu bar. Read the online Emacs tutorial, you'll see what I mean. There is a discussion that goes on for pages about how to move the cursor and select text with the keyboard. The user is asked to remember keystrokes like Control-x Control-f to open a file. That is 1) tedious and 2) not as easy as mouse clicking on File/Open. The default Emacs framework is still trying to manage the display as if it were a physical terminal, with no movable windows. I have used Emacs on a daily basis for at least a decade and I still don't remember the complicated key strokes. Instead, I have learned how to configure my startup file so that Emacs works like other programs. In particular, I want new content–new files and new popup menus–to appear in their own window frames, without covering over the material I'm working on. Hopefully, if you have installed and customized Emacs according to the Wiki instructions that I provided in the introduction of this essay, you will see what I mean.

I'm running a major risk by writing so frankly about Emacs. My smart friends who program for a living will now learn, once and for all, that I'm really not one of them. But, if you are willing to try things my way, then I will be one of you.

4.5.1 Run Emacs from a Terminal

Here's what happens when I want to get some R work done. I log into Linux, then I open a terminal. Second, I change into a directory in which the R files will be saved. If there is no directory, I make one.

$ mkdir −p ps/stats/exercise−4.5

$ cd ps/stats/exercise−4.5

Then create a new file "filename.R" by typing "emacs filename.R". The suffix R tells Emacs that you are going to use ESS mode. Continuing the example,

$ emacs exercise−4.5.R

That should open a new, blank file, and it should cause the ESS (Emacs Speaks Statistics) interface to start up. (Don't forget to run getwd() to find out where you are).

Note that the "Frame" on the screen (Emacs calls a "window" a "frame") is divided into several parts.

1. Pull down menus: File, Edit, Option, etc
2. Buttons
3. Main "buffer," the place where you type most of your commands.
4. Status bar: indicates the editor mode and other useful things, sometimes
5. Command mini-buffer: at the very bottom, this is a place where you can type some commands and read output.

If the start-up options have not been customized to prevent this, Emacs displays a large "spash screen," as shown in Figure 4.1. Any key stroke or mouse click will erase the spash.

One can type R commands into the open file, but nothing can be sent to R until R is actually started. In order to launch R, one can take several methods. Note that in the Emacs "button bar," the ESS module draws some R-specific items

The large R will start R and display it in an Emacs terminal. One can also launch R by the keystroke

```
M–x R (and Enter)
```

Figure 4.1: Emacs Start Screen

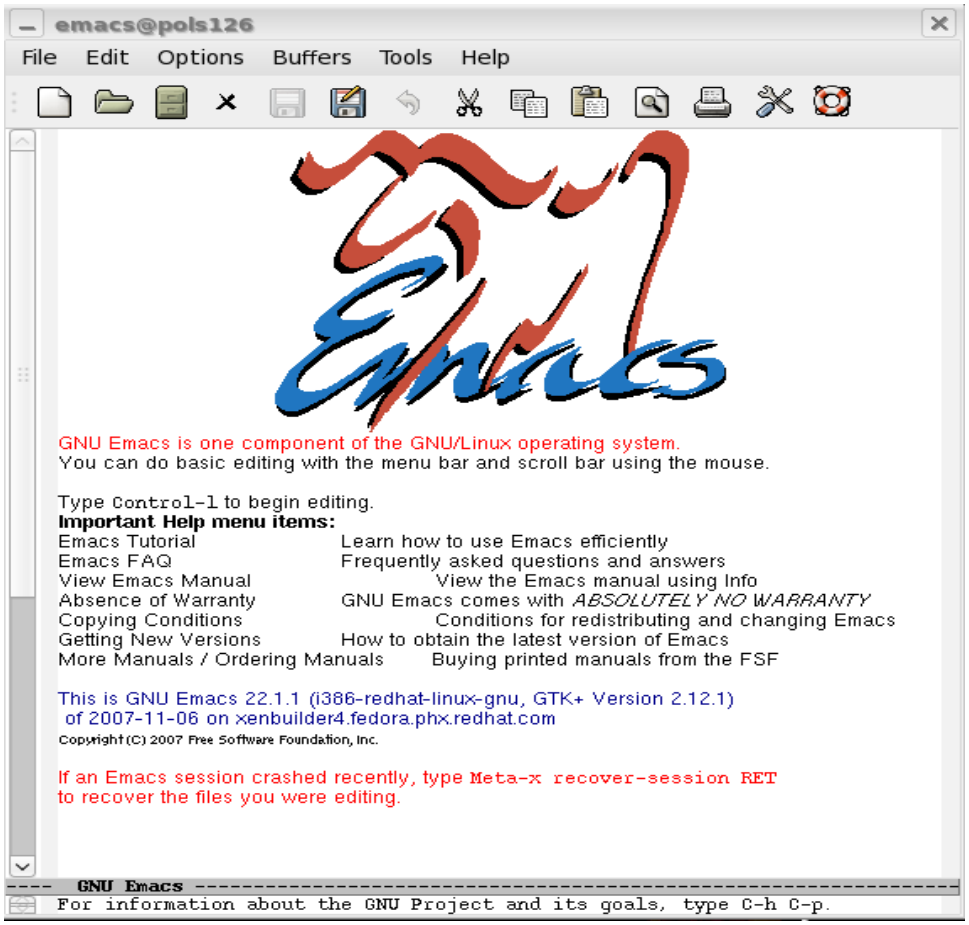
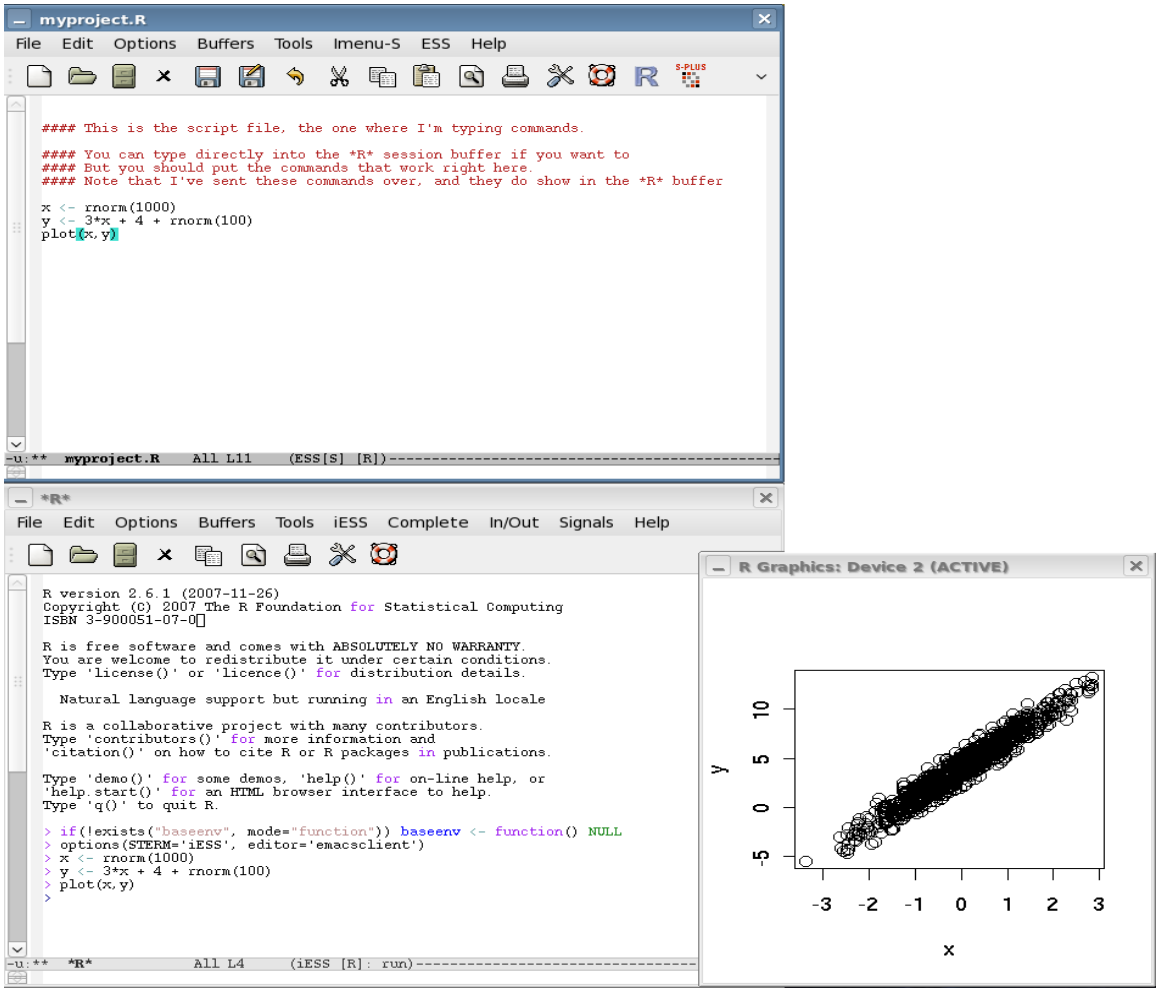


Figure 4.2: Emacs Running ESS



What's M? In the Emacs documentation, one will find three prominent modifier keys. Shift (S), Meta (M), and Control (C). On IBM style "personal computers," the "Alt" key is the Meta key. So M-x means Alt-x. After that keystroke, the minibuffer–in the very bottom of the Emacs frame–pops open and it says "M-x", which means "go ahead, I'm ready to execute what is typed next." Type R in that "command mini-buffer" and it return.

Be careful to resize Emacs before you try this. Sometimes the Emacs frame will be too large for the screen and the mini-buffer will fall off the bottom of the screen. Be careful to adjust, or else you will miss out on a chance to interact with Emacs.

In Figure 4.2, a display of Emacs with ESS in action is presented. These frames have been re-arranged on the screen so that one showing the R commands is on top, the R session that is running inside Emacs is in the bottom frame, and a small plot is displayed on the bottom right.

In the default configuration of ESS for Emacs, R startup will cause the Emacs frame to be horizontally divided ("window splits"), and the Emacs will pause for some user interaction

with the mini-buffer. There will be no R until the user designates a starting directory. And then, when you try to send a command to R, then the program will pause and ask you if you really want to send it to R.

On my systems, I've customized Emacs so that R does not bother us with so many questions. Systems are set so that:

1. R always starts in the current working directory, without asking permission, and
2. A new Emacs "frame" will be displayed–that's a separate "window" in the screen where R is running.

In Emacs, a **buffer** is a file or "thing" that is open for editing. They change the menu items all the time, but why don't you experiment. In the File pulldown, choose "New Frame" and watch what happens. Experiment with the File pulldown's options to split and unsplit the frame.

There are two very special things about ESS. First, Look for the Emacs buffer called *R*. That is the R session. In that frame, Emacs is displaying R in the same way that a terminal, displays R. One can type in any R commands. If everything is properly configured, ESS will allow "tab completion" of commands. As in an ordinary shell, if one types a few letters and hits tab, then the system will search for completions. If there are many possible matches, a new buffer will be opened and a list will appear. One of these can be selected with the mouse.

Second, look for the Emacs buffer exercise-4.5.R. In that buffer, one can type commands and build a script. Instead of "copy and paste" to take those commands to the R session, ESS offers some "shortcuts". Position the cursor at the top of the script. Clicking on the one-line arrow in the ESS icon bar. The same can be done with the combined keystroke C-x C-n. (To me that seems tedious. If your version of Emacs is setup with the options in emacs-ess-kups.el, shift-Enter will also send lines over to the R session.)

You can also send a "region" over to the R session from the buffer exercise-4.5. For example, try a pair of lines, such as

```
x ← rnorm(100)
hist(x)
```

then highlight both lines with the mouse, then send the commands to R.

You can send this small region of commands to R in several ways.

1. Button bar: Hit the menu icon with an arrow -> and three lines under it. If you hover the mouse over it, it will say "Eval Region". Hit that, it will send the command over.
2. Menu. After highlighting the region, click on the word ESS in the top menu bar, pull down to "ESS and go" and choose "Eval region". The only difference here is that since you chose the "go" option, the focus will change to the R window.

I'm not going to mention the keyboard sortcut, mostly because the Emacs "cua-mode" keyboard option (which I like!) interferes with some ESS keystroke combinations. But we could turn off cua-mode (in the Options menu item, uncheck the cua mode box). In case you wonder about the keyboard shortcuts, mouse click on the ESS menu, and the shortcuts are labeled.

At the end of a work session, one will generally save the script file as well as the output files. First, save the command file (the .R file). Second, save the "*R*" buffer as a text file. Click in that buffer and hit File->save as. I would choose a name like "exercise-4.5.Rout").

Note that the "*R*" buffer, the one that displays the commands and results, will allow you to type in the second space after the prompt ">". There is one big difference from the ordinary terminal, however. Inside Emacs, the up-arrow and down-arrow do not cycle through the command history. In Emacs, those keys move the cursor. In order to go through the command history, press the control key before hitting the up and down arrows.

4.5.2 If You Started Emacs from an Icon or a Menu

More and more people look at me like I'm crazy when I say "open a terminal and run emacs". Even on Linux operating systems, people want to use the "GUI desktop environment" click on the pretty Emacs icon or click on a file to open Emacs. That will open Emacs, of course, but the working directory will generally be the user's HOME directory.

This is easily fixed BEFORE STARTING R. If the working directory or R files already exists, one can simply open them with the File->Open dialog. If no R file exists, choose File->Visit New and use the directory navigator to change to the desired directory. If the desired working directory does not exist, it can be created from within Emacs. Choose the File->Open Directory menu option, and it opens the Emacs directory editor. There is a pulldown menu called Immediate that offers the option of creating a new directory.

After starting the *.R command script file, then one should start R. And then use getwd() to double-check the working directory.

4.6 There are Other Editors, of course.

When R is sitting open, and you are typing commands into it, and results are streaming out, everything seems fine. If you close that session, all of your hard work vanishes and you have to open R again and type it all over again.

Don't let that happen to you. Keep a complete record of commands that work in a separate file. By doing so, one can always reproduce a result.

Perhaps Emacs is too complicated or not available. In the worst case scenario, one can use any editor that allows cutting and pasting. Even Windows Notepad will provide the bare minimum ability to record commands that can be pasted into a running R session. The other features offered by Emacs are lacking, but something is better than nothing (I just invented that saying).

There is an editor called TINN that has been recommended by some Macintosh and Windows users.

4.7 Run R "Inside" a LaTeX document

LaTeX is a document preparation system. LyX is a program that is "like a word processor" that can prepare LaTeX documents. This document that you are reading was prepared with LyX.

There is a LaTeX document subsystem known by the nickname "literate programming". In a literate program, one writes words and sentences, and then when one needs a calculation, one inserts some code commands and a result "pops out." There is a variant of this approach called Sweave, and I have used it both in Emacs while editing LaTeX documents directly and also while using the LyX "like a word processor" program.

I think I will postpone the details of this for your further study, but in order to excite your enthusiasm, I refer you to a directory of documents that I created with past students.

`http://pj.freefaculty.org/stat/Distributions`

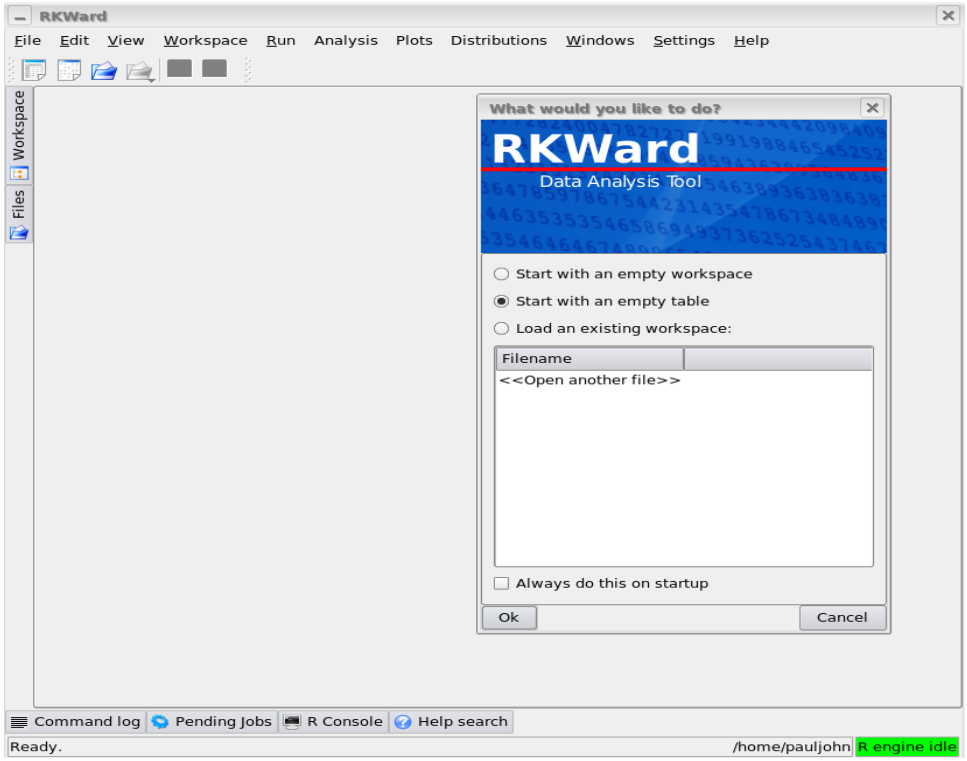
We wanted to understand different kinds of statistical distributions. The PDF documents show our results, and the lyx files show the input. I don't hesitate to say that the creation of such a wonderful, beautiful set of documents would have been impossible without Lyx and Sweave.

4.8 Graphical Interfaces

If one runs R in Windows or on the Macintosh, a sort of bare-bones graphical interface is started. In Linux, I don't usually wish I had an interface like that, except when it comes to saving the results of graphs into files. It seems to me that the Windows interface that allows one to point and click to save into a Windows Meta File is a valuable thing. Too bad you have use Windows to obtain that feature (insert smiley face here).

There are several efforts on-going to create a more pleasant, complete, graphical user interface for R. At the current time, I think they do some nice things, but do not encourage beginners to spend any effort on them. People should learn to use R directly first, rather than relying on these as a crutch. But you can try them as you please, as long as you promise me that you will never say "I can't do that assignment because there's no button for it in my program."

Figure 4.3: rkward's opening Screen



4.8.0.1 rkward

This is the newest of the R environments that I've found, and I found about it quite by accident. It is in the Fedora Linux distribution. rkward is a program that will start an R session, allow you to send commands to R, and it will keep track of results. It has some nice pull-down menus. If you need to do the particular exercises for which the rkward authors have planned, you will have some fun.

When rkward starts, it initiates R, and then it asks the user if a new "clean worksession" is desired, or if an existing worksession should be opened. That is illustrated in Figure 4.3.

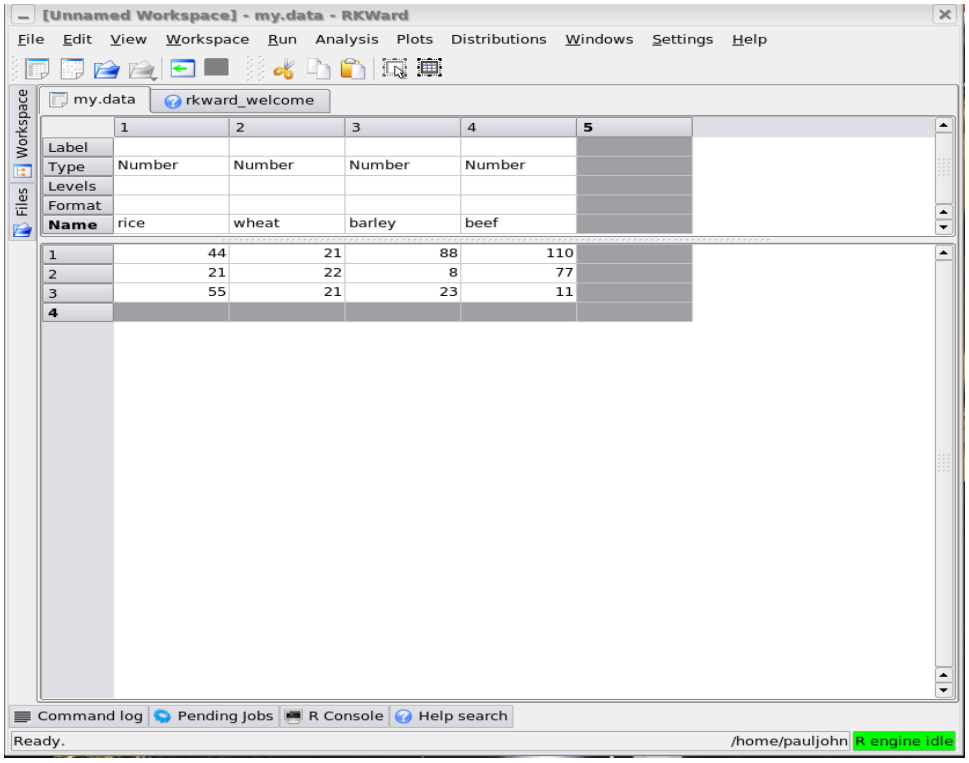
rkward has a very special feature, a nice data entry spreadsheet-style thing, that is illustrated in Figure 4.5. In the menu under Distributions, one can select the Beta and then vary the parameters and watch as the probabilities change.

In Figure , a snapshot of me experimenting with the Beta distribution:

4.8.0.2 R Commander

Inside R, run

Figure 4.4: rkward's Data Entry Table




```
> library(Rcmdr)
Loading required package: tcltk
Loading Tcl/Tk interface ... done
Loading required package: car
Rcmdr Version 1.3-11
```

A picture of the Rcmdr at start time is displayed in Figure 4.6. As luck would have it, something is wrong with my install of Rcmdr at the moment, because my mouse is not able to select variables inside the menu options. That's about par for the course, in my experience. These GUI things always let you down.

Sometimes that's handy. Often it is more trouble than it is worth. These days, the only time I ever run Rcmd is when I want to experiment with 3-D plots. After removing Rcmdr and re-installing it, I was able to make a 3-D graph, as illustrated in Figure

4.8.0.3 jgr

"JGR", pronounced "Jaguar", is a Java based graphical interface for R. At the current time, it runs, and many people like it. Personally, I don't use it so much. Like any other Java program, if it runs, it will probably be OK. If it doesn't, well, you want to gouge your eyes

Figure 4.5: rkwrap Feature to Illustrate Probability Distributions

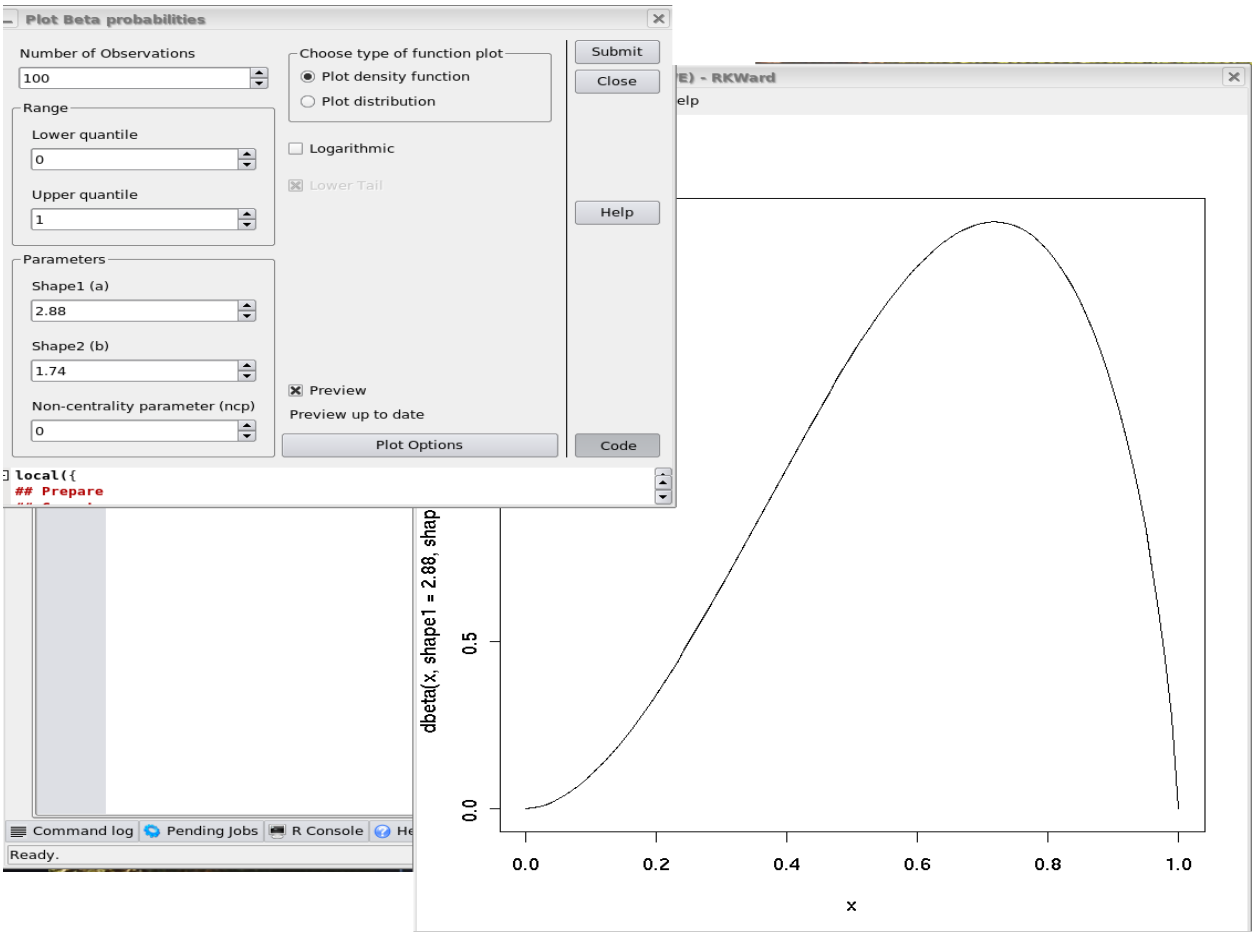


Figure 4.6: Rcmdr

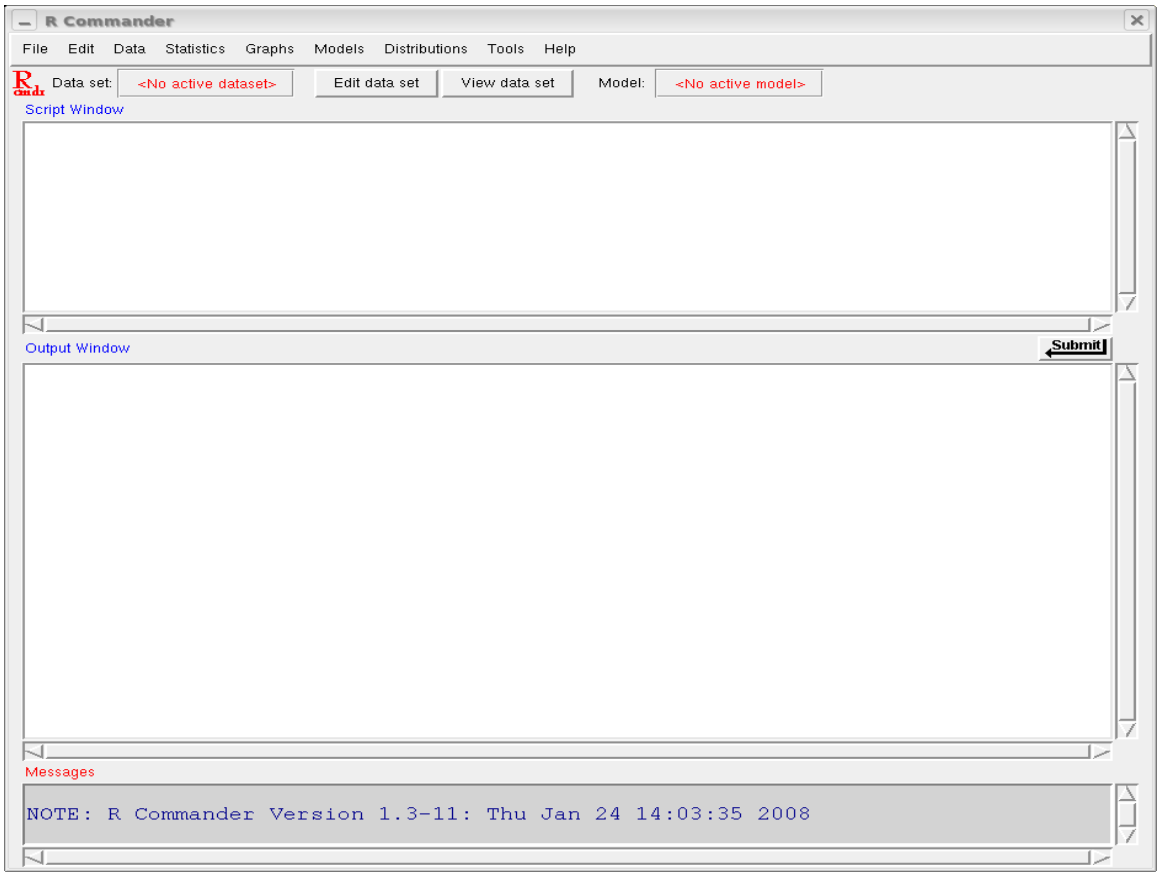
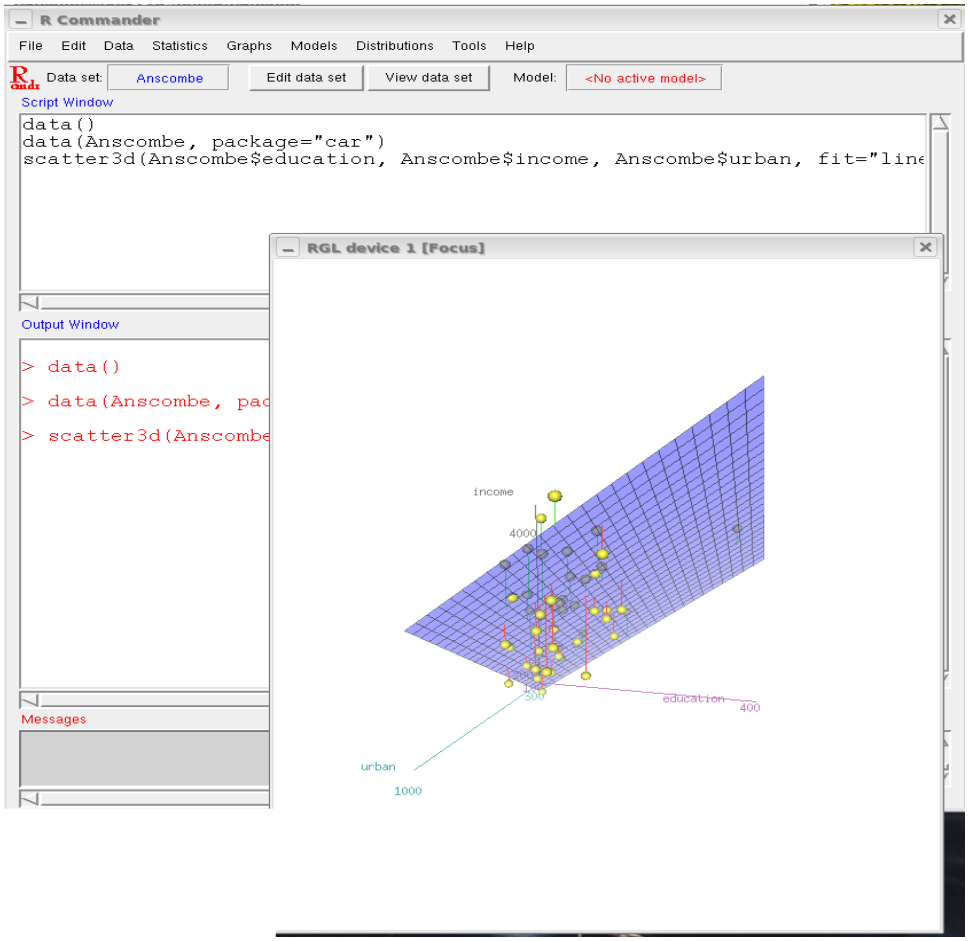


Figure 4.7: A 3-D Scatterplot from Rcmdr



out in disgust. You can run JGR either from within an R session or from the command line. Assuming it is installed in R, that is.

Inside R, run

```
> library(JGR)
```

That will spit out some instructions about how to start the JGR interface.

You can also run JGR as a program before starting R. I think that's what the JGR authors recommend as a part of the effort to hide R details from users. In the Blake lab systems, open a terminal and type "jgr" and it may work. I mean, I tried to set it up for you.

4.9 Understand the Environment that R Creates

When R starts, it creates an environment–a place that links you with R and R with your computer's environment.

On one occasion,

When I'm in the middle of an R session, and things start happening that I don't expect, I start to wonder if the environment is somehow damaged or corrupted. Here are some things worth knowing.

```
> objects()
```

gives a listing of R objects that exist at the current time. As far as I know, it is the same as running this command

```
> ls()
```

To remove a particular item, say X, from the environment, you use the rm command.

```
> rm(X)
```

If you want to kill all of the objects you have, this works.

```
> rm (list=ls())
```

If you read the rm help page, you will see that it can accept an R list as an option. So we specify the list option here, and for our list we give the output from ls(), which includes everything. After that, one can forget about his/her mistakes and work on a clean slate.

Suppose you run ls() and you see a bunch of objects, and you wonder what they are. For example, if you run commands like "demo(graphics) and "demo(persp)", you will end up with a bunch of objects in memory.

```
> ls()
[1] "fcol" "fill" "g" "i1" "i2" "lev"
[7] "n" "opar" "pie.sales" "pin" "rotsinc" "scale"
[13] "sinc.exp" "usr" "x" "xadd" "xdelta" "xscale"
[19] "xx" "y" "yadd" "ydelta" "yscale" "yy"
[25] "z" "z0" "zi"
```

Type an object's name, and it is displayed. To find out more about them, use the functions attributes() or summary().

```
> ydelta
[1] 600
> attributes(ydelta)
NULL
> g
   [1] 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  [25] 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  [snipped to save space]
 [961] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 [985] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
Levels: 1 2 3 4 5 6 7 8 9 10
> attributes(g)
$levels
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
$class
[1] "factor"
> attributes(fill)
$dim
[1] 88 62
> is.matrix(fill)
[1] TRUE
> fill[1,]
 [1] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[snipped to save space]
[61] "gray" "gray"
```

5 Data in R.

A data frame is a set of "columns" of the same length, but not necessarily the same "type" of data. One column might represent the height of survey respondents, another might indicate "Yes" or "No" responses to questions. Any valid R variable type can be included in a column of a data frame.

On the contrary, an R matrix must include columns that are all of the same type.

A list in R is a collection of objects of any types and lengths. So any data frame can be viewed as a list. But not all lists can be data frames.

5.1 Using data frames supplied with R packages: data()

If you have never run R before, this may be a surprise. You get some data sets for free! The command data() will access them, assuming you have used the library function to retrieve the package you need.

The command

```
> data()
```

will list all R data frames that are available in currently attached dara frames. There are many other data frames as well, the data() command has this tip at the end:

Use

```
> data(package = .packages(all.available = TRUE))
```

to list the data sets in all *available* packages.

Once you see one you want, use the library function to open the package.

Suppose, for example, you want to find out about the data frame called "Skye" that is in the MASS package. Try this:

```
> library(MASS)
> library(help=MASS)
```

Notice that the output divides the elements provided by the package MASS into categories, Functions and Datasets.

The dataset "Skye" is described as "AFM Compositions of Aphyric Skye Lavas." Everybody wants to know about larvas (larvae?), I always want to say.

```
> data(Skye)
> ls()
[1] "Skye"
```

```
> is.data.frame(Skye)
[1] TRUE
```

```
> str(Skye)
'data.frame': 23 obs. of 3 variables:
$ A: int 52 52 47 45 40 37 27 27 23 22 ...
$ F: int 42 44 48 49 50 54 58 54 59 59 ...
$ M: int 6 4 5 6 10 9 15 19 18 19 ...
```

```
> summary(Skye)
A F M
Min. :13.00 Min. :42.00 Min. : 4.00
1st Qu.:18.50 1st Qu.:51.00 1st Qu.:12.50
Median :23.00 Median :54.00 Median :20.00
Mean :26.83 Mean :53.74 Mean :19.43
3rd Qu.:32.00 3rd Qu.:57.50 3rd Qu.:22.50
Max. :52.00 Max. :62.00 Max. :39.00
```

5.2 Creating data frames.

Everybody should learn the basics about creating data frames in R. It is easy to load a pre-manufactured data frame from a package with the load command. But there's no fun in that.

5.2.1 Create a data frame interactively.

For small statistical exercises, it is convenient to create the data structures manually. Consider these commands.

```
james ← c( 0, 1, 0, 1, 0, 1)
bill ← c( 44, 232, 11, 99, 12, 19)
jane ← c( yes, no,yes,no,no,no)
myDF ← data.frame(james,bill,jane)
thoseNames ← c(var1,var2,var3)
names(mydf) ← thoseNames
```

This is the same as

```
myDF ← data.frame(var1=james, var2=bill,var3=jane)
```

It is easy to add a new variable to an existing data frame. For example, if a column of values is stored in a vector named emily, one can add that to the data frame with the variable name label4 as follows.

```
emily ← c("red","blue","green","red","blue","yellow")
mydf$var4 ← emily
```

5.2.2 Bring a text data file into R.

To import numbers from a text file, use read.table(). This is a fundamental R skill that all users should master. Please read the help page in R. Consult my Rtips sheet.

First, we need to create a text file that has some numbers that we can import. Any text editor will do, as long as it creates an ordinary "text" file without any fancy formatting stuff (no "underlining" or "bold" or paragraph styles or other formatting elements inserted by programs like Microsoft Word or OpenOffice writer).

Let's suppose we want to create the text file in Emacs. In Emacs, from the File pull down menu, click "open new".

In the bottom command line it will ask for a file name. For this example, I call it "testdata.txt".

In the Editor, create a silly text data set with the names in the first row, like:

```
x1 x2 x3 x4 x5
22 1 22 11 11
22 33 11 323 11
22 1 33 55 76
55 33 33 44 33
```

Don't forget to "hit return" at the end of the last one. You may get an error if you don't. The first row in that file is a "header" row, a list of variable names.

Save that. Then, in R, try to read that in.

```
> read.table("testdata.txt",header=TRUE)
```

R looks in the "current working directory," which (hopefully) is the same place where testdata.txt will be found (unless something went wrong). If you saved the data somewhere else, then you have three choices. First, copy the data file into the proper directory. Second, use the setwd() command in R to change the working directory to the directory where testdata.txt is saved. Third, give the path to the file in complete detail in the read.table command, as in:

```
> read.table("/home/paul/MyRStuff/testdata.txt", header=TRUE)
```

The data should be printed out in the R session. BUT IT IS NOT SAVED. To save it as a "data frame" you have to do this:

```
> paultest ← read.table("testdata.txt",header=TRUE)
```

(that's all on one line)

Now look over your new data frame:

```
> paultest
```

Ask for the column names

```
> names(paultest)
```

The dollar sign notation calls for particular variables.

```
> paultest$x1
> paultest$x2
```

Note you can access the first row only this way too:

```
> paultest[1,]
```

If you want the first column.

```
> paultest[,1]
```

Now try:

```
> str(paultest)
> attributes(paultest)
> summary(paultest)
> objects()
> help.search("data.frame")
```

Suppose you want to look only at cases for which the first variable, x1, equals 55.

```
> paultest[paultest$x1==55, ]
```

Note the double equal sign for "logical equal to". That's standard in many computer languages.

5.2.3 Excel.

We struggled with exchanging data from Excel for quite a while. One can export a "csv" (comma separated variables text) file from Emacs and use read.table() to import it. If the user customizes the text output, using a special separator like |, then there's a fairly good chance that this

```
read.table("whatever.csv", header=T, sep="|")
```

will work. That process is a bit tedious, however.

There are functions available that can read in data directly from an Excel spreadsheet. On my systems, I use a package called "gdata". It has a function called read.xls. That function works very well on Linux and Mac systems, but on Windows it is sometimes problematic because the helper program Perl is not installed. Perl is generally installed as a helper program for programs like ImageMagick or Gimp, so it is not really much of an obstacle. Another option for Windows users is a package called xlsReadWrite. This works without complication, but only within limits. (Its author offers an enhanced version, xlsReadWritePro, that is available for purchase).

It is necessary that the data in the spreadsheet must a rectangular–it should be in rows and columns. Load and read about read.xls like so:

```
> library(gdata) ## or library(xlsReadWrite)
> library(help=gdata)  ## or library(help=xlsReadWrite)
> ?read.xls
```

If "/home/pauljohn/blop/mydata.xls" is a rectangular set of numbers in Excel Worksheet 1 with single-word names in the top row, then bring that into R with a command like

```
> mydat <- read.xls("/home/pauljohn/blop/mydata.xls",sheet=1)
> mydat
> str(mydat)
> attributes(mydat)
```

If this does not work, there are some frequent sources of trouble to check. First, in the spread sheet program, check for the column formats (right click on the top of the column in most programs). If the format says "GENERAL", then that format is causing read.xls to fail. Change that column type to either "Numeric" or "Text" and save the spread sheet again. Second, Excel allows a single file to include several sheets. Make sure read.xls() is looking at the desired sheet. The read.xls assumes that you want to import the first sheet, and if you need a different sheet, it is necessary to specify it.

If it is necessary to create an Excel spreadsheet, there are several options. The most dependable option is to use R's write.table() function to write the data into a text file. I specify the separator in order to be absolutely sure to bring in the information correctly. In Excel, or in any other spreadsheet for that matter, one will then be careful to do a "configurable" text

import, so that the separator between columns can be set appropriately. It may also be possible to save an Excel spreadsheet directly from R. The package "dataframes2xls" (available for most platforms) promises that ability. It is also available in the xlsReadWrite package for Windows.

The read.xls import will fail if the spread sheet has functions or other unrecognizable components. The only option is to remove those components or work around them. If a sheet uses a formula to set values in some columns, I have found one effective approach. Highlight all of the columns that are desired, and then move to a clean spreadsheet and use "paste special" and choose the radio button "values." This will convert the functions to numerical values. This new sheet can be imported into R with read.xls().

If the Excel spread sheet is too complicated for R to import directly, or if you don't have a working version of read.xls(), (possibly because your system administrator did not install all of the required software), then you have to attack the problem in the old fashioned way. In the spread sheet program, figure out how to export your data as a text format, possibly in comma separated format. I tend to prefer to change the separator to a rare symbol like "|", in order to keep everything tidy. After the data is exported in some text format, use the Emacs editor to browse the file and inspect for telltale signs of trouble, such as hundreds of rows of gibberish. Spread sheet programs often insert phantom columns when they output to text. After verifying the quality of the text data file, be sure to specify the delimiter in the read.table().

```
> read.table("testdata.txt",header=TRUE, sep="|")
```

5.2.4 Importing data from Statistical Programs

Political Scientists who learned to do research in the 1970s used SPSS (Statistical Package for the Social Sciences) almost exclusively. Although SPSS today bears almost no similarity at all to the version that was used "back in the day," there are still plenty of people who have stayed with it. When they share data with R users, it is incumbent on the R user to import the information and verify the accuracy of the translation. In the 1980s, SAS was increasingly popular, and in the 1970s Stata gained a wide followership among young researchers. Many public archives, such as the University of Michigan's ICPSR (Interuniversity Consortium for Political and Social Research), will provide data sets in formats suitable for those older programs. The importation of SAS data into R is much less dependable than the importation of SPSS or Stata data, but at KU we are fortunate because we have access to SAS on widely accessible computer systems and we can use those systems to export data into manageable formats.

The foreign package in R has great functions for bringing Stata and SPSS data sets into R. Run

```
> library(foreign)
> library(help=foreign)
```

and look for "read.dta" and "read.spss".

A word of warning. One must be very careful to compare the results against the "codebook" provided by the original datasets.

5.2.5 SPSS data focus

When using read.spss, it is important to check the options you want. In this example, I'm reading an SPSS dataset that was created by a colleague, who drew some variables from Professor Robert Putnam's Social Capital Benchmark Survey which is available through the Roper Center. (`http://137.99.31.42/CFIDE/roper/collectioninterest/webroot/registration.cfm?subject=SCC`) .

```
mydat ← read.spss("SCBS.sav", use.value.labels=T,
                  max.value.labels=17, to.data.frame=T)
```

(I use the name "mydat" for almost everything I create on the fly...) If one forgets to add the option to.data.frame=T, the result will be a list of variables, not an R data frame. The value labels from the SPSS dataset are used to try to create R factors that represent categorical or qualitative variables (unordered "nominal" variables or ordered factors, i.e., variables for which differences in numerical scores are not substantively meaningful).

5.2.6 Stata Data Focus

If your University belongs to the InterUniversity Consortium for Political and Social Research (ICPSR), you will have access to a vast collection of data sets. Data will generally be available in a text file format, which is difficult to read correctly. It may also be available in an SPSS or Stata format. I recently downloaded all of the data files for the American National Election Study, 2004 from this address: `http://www.icpsr.umich.edu/cgi-bin/bob/newark?study=4245`. I downloaded a big zip file called "10243963.zip" that includes all the files. After unzipping, the data file that I want is down in a subdirectory, as you can see:

```
library(foreign)
mydta1 ← read.dta("10243963/ICPSR_04245/DS0001/04245-0001-Data.dta")
names(mydta1)
```

In that same subdirectory, there is a pdf formatted Codebook, in which one finds details on the definitions of the variables. In that file, one finds the single most important variable in American political behavior research:

```
V043116 J1x. Summary: R party ID
========================================
PRE-ELECTION SURVEY:
QUESTION:
```

```
---------
Generally speaking, do you usually think of yourself as a REPUBLI-
CAN, a DEMOCRAT, an INDEPENDENT, or what? Would you call your-
self a STRONG [Democrat/Republican] or
a NOT VERY STRONG [Democrat/Republican]? Do you think of your-
self as CLOSER to the Republican Party or to the Democratic party?
VALID CODES:
------------
0. Strong Democrat (2/1/.)
1. Weak Democrat (2/5-8-9/.)
2. Independent-Democrat (3-4-5/./5)
3. Independent-Independent (3/./3-8-9 ; 5/./3-8-
9 if not apolitical)
4. Independent-Republican (3-4-5/./1)
5. Weak Republican (1/5-8-9/.)
6. Strong Republican (1/1/.)
7. Other; minor party; refuses to say (9/./. ; 4/./3-8-9)
MISSING CODES:
--------------
8. Apolitical (5/./3-8-9 if apolitical)
9. DK
```

After importing the data, one should compare the information from the Codebook against the values that were actually imported. I often use commands like this:

```
> table(mydta1$V043116)
> attributes(mydta1$V043116)
```

The labels of the factor levels are tedious and too long for statistical output, and I wanted the output to look a little tighter. One should generally be cautious about making changes "on top" of an existing variable, but I throw caution to the wind by renaming the levels as:

```
> levels(mydta1$V043116) ← c("SD","WD","ILD","II","ILR","WR","SR","O","APol",
    "DK")
```

5.3 Extracting values from data frames

5.3.1 Access values with either [] or $ notation

With dataframes, one can obtain an individual variable by referring to its column name:

```
> myDataFrame$gnp
```

or by asking for a particular column, as in

```
> myDataFrame[ , 4]
```

supposing that the variable in the 4th column is "gnp", these commands yield the same result.

The use of the array style notation is most useful when one wants to "grab" a collection of columns. It would be faster to write

```
> myDataFrame[, 4:10]
```

to take the fourth through tenth columns than it would be to take them by name, as in

```
> myDataFrame[ , c(gnp,poverty,ed,health,energ,water,pollute)]
```

5.3.2 Subset to take particular rows

A cave man would hunt around and select rows like this

```
> myDataFrame[ c(1,5,6,8,10,22), ]
```

Someone who shops at Geico would choose rows in a more clever way, either by stating a logical condition in the braces,

```
> myDataFrame[ myDataFrame$pollut > 888.3, ]
```

Or by using the "subset" command.

5.4 Any matrix can become a dataframe. But the converse is not true.

Can you see why

```
> myDF <- as.data.frame(someMatrix)
```

will always do what you want, but

```
> aMatrix <- as.matrix(myDF)
```

may not? Try it and see with a data frame that has different types of columns. For example:

```
> df1 <- data.frame(x=c(1,2),y=c("a","b"))
> mat1 <- as.matrix(df1)
     x   y
[1,] "1" "a"
[2,] "2" "b"
> is.numeric(mat1[ , 1])
[1] FALSE
```

Don't read more until you figure what's wrong there.

5.5 Never use attach(). Just don't.

Instead, look for procedures that allow you to specify which data frame should be used, such as

```
> plot(y~x, data=someDataFrame)
```

or, in the alternative, use the generic approach

```
> with(someDataFrame, plot(y~x))
```

Why not? attach() creates confusion because it creates "copies" of variables in a data frame for your use. If you make some changes in variables, then the changes you make apply to the locally stored "copies", not the original data frame. Many students have lost their work to this quirk.

5.6 Dealing with Missing Values

R uses a special symbol NA for missing values. Our challenge is to figure out which observations should have scores set equal to NA, and then one must assign those values. The management of missing values is especially important when data is imported from other formats. In the case of data imported from Stata or SPSS, one will often find scores that appear valid, but in fact they are not.

5.6.1 Numerical variables and missing values.

This continues the SPSS data import example from section 5.2.5. In the Foreign package that is distributed with R 2.6.1, I notice some significant improvements in the read.spss function. It is better at differentiating variables that are intended to be seen as numerical from variables that are truly categorical variables. In previous versions of R, a numerical variable with valid values from 1 to 120, along with a labeled "missing value" represented by 999, would be treated as a factor. As you can see from the following, the variable AGE is rightly seen as a numerical variable. The table function shows the values in the first row and the number of respondents who gave that answer in the second row.

```
> table(mydat$AGE)
18  19  20  21  22  23  24  25  26  27  28  29  30  31
113 121 123 149 155 164 164 180 167 157 201 222 210 191
[snip to save space]
78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 115 998 999
44 42 49 38 27 32 19 14 13 15  6  5  5  5  4   1   2 230
> is.numeric(mydat$AGE)
[1] TRUE
```

The only problem is that the "missing values," ages 998 and 999, are still included in the numerical data. We could get rid with an unimaginative command like this (which finds all cases for which AGE is greater than 997, and then sets AGE for those cases to NA).

```
> mydat$AGE[mydat$AGE > 997] ← NA
```

That approach is not very clever, however, and it does not easily translate to other variables. We'd rather not fuss over each variable in such detail if we can avoid it. For this specific case, at least, I've found a way to grab the correct values for missing scores and set them as NA in the AGE variable. Note that read.spss has set an attribute on the AGE variable called value.labels.

```
> attributes(mydat$AGE)
$value.labels
Refused Don't Know
    999         998
```

The following grabs those attributes and then uses them to set the unwanted scores to NA.

```
> age.attributes ← attributes(mydat$AGE)
> mydat$AGE[mydat$AGE %in% age.attributes$value.labels] ← NA
```

In the end, we have no more observations that seem to indicate respondents are 998 or 999 years old.

```
> table(mydat$AGE)
 18  19  20  21  22  23  24  25  26  27 28  29  30  31
113 121 123 149 155 164 164 180 167 157 201 222 210 191
[SNIP to save space]
78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 95 115
44 42 49 38 27 32 19 14 13 15  6  5  5  5  4  4  2   1
```

Apparently, we have one respondent who is actually 115 years old. (So many smart alec comments occur to me that I elect to pass, on the grounds that it would be too easy.)

I expect there will soon be an automatic way to set those values as missing.

5.6.2 Qualitative Variables, Value Labels, and Factors

Consider a qualitative variable, such as "How much do you trust your neighbors." In the SPSS tradition, the responses are assigned numerical scores. Any numbers will do, of course, but suppose we have {1,3,5,7,8,9}. How are we supposed to know what those values indicate? The meanings are recorded in "value labels". Suppose, for example, the number:value label pairs in SPSS are:

```
1: None
3: A little
```

```
5: Some
7: A lot
8: Don't Know
9: Refused to answer
```

The S language is partly based on the idea that assigning numerical values to a qualitative indicator is altogether silly and we should stop. So, in R, one would create a variable with acceptable values like {"High", "A little", "Some", "A lot"} and unknown observations would be assigned an NA. The foreign package in R tries to help a little bit with the translation.

In R 2.6.1, the read.spss command takes notice of these value labels. If the option use.value.labels=T is set, then the values of the labels will be used as the values of an R factor.

```
> attributes(mydat$TRUST)
$levels
[1] "Not at all" "A little" "Some" "A lot"
$class
[1] "factor"
```

On the other hand, if the read.spss command is run with the option use.value.labels=F, as in:

```
> mydat2 ← read.spss("SCBS.sav", use.value.labels=F, to.data.frame=T)
```

then the variables will be numeric. However, even in this case, each numeric variable that has value labels in SPSS will have a special attribute in R. Observe the attributes:

```
> attributes(mydat2$TRUST)
$value.labels
      A lot    Some    A little    Not at all
          3       2           1             0
> is.numeric(mydat2$TRUST)
[1] TRUE
```

R does not think of NA, the special symbol for missing values, as a value label.

```
> table(dat2$TRUST)
 0    1    2    3
590 1203 3189 3061
```

The table command ignores missings, and forcing it to display the missing cases is something of a chore.

```
> table( factor(mydat2$TRUST, exclude=NULL) )
 0    1    2    3   <NA>
590 1203 3189 3061 1454
```

5.7 About that Warning in the factor manual page

Users who type

```
? factor
```

are treated to the usual kind of R help page. It describes how one can create factors and assign labels. Like many of the help pages, the writing is somewhat terse and one might read a single sentence many times before understanding all of its implications. At the end of the factor help page, there is one particular comment upon which I have gazed for many hours:

> Warning:
>
> The interpretation of a factor depends on both the codes and the '"levels"' attribute. Be careful only to compare factors with the same set of levels (in the same order). In particular, 'as.numeric' applied to a factor is meaningless, and may happen by implicit coercion. To transform a factor 'f' to its original numeric values, 'as.numeric(levels(f))[f]' is recommended and slightly more efficient than 'as.numeric(as.character(f))'.

This tidbit was buried in the end of the details section of the page. In the r-help email list, people regularly crop up saying "R wrecked my variable" and they are referred to this Warning. I don't recall seeing another help page in R that has a tidbit so dramatic that it is called a Warning. Anything so unusual deserves some attention.

If you understand this Warning, then I believe you are well on your way to understanding R on a very deep level. Before this Zen moment of clarity evaporates, I will jot this down.

Consider a numerical variable "age." Let's suppose we go to a father-daughter party at the sorority house and it turns out that all of the daughters are 21 years old and all of the fathers are 45 years old.

```
> age ← c( 21, 45, 21, 45, 21, 45, 21, 45)
```

In a statistical analysis, say a regression model, or in a plot, the variable age will be treated as a numerical scale. One might be inclined to make statements like "each additional year of age results in a change in predicted income of ..." This kind of statement may be unwarranted because the data only contrasts two particular ages, it does not support a claim of the form "if a 21 year old becomes 1 year older, then we expect something to happen."

As a result, we have an ideal case of the need to treat an apparently numeric variable as a factor. It is easy to create a new factor variable:

```
> ageFactor ← as.factor(age)
> ageFactor
[1] 21 45 21 45 21 45 21 45
Levels: 21 45
> attributes(ageFactor)
```

```
$levels
[1] "21" "45"
$class
[1] "factor"
> levels(ageFactor)
[1] "21" "45"
```

Now comes the big shock, the one that causes all of those puzzled messages to r-help. Watch what happens if one changes her mind, and decides that the ageFactor approach was a mistake. If the original age variable was somehow lost, one might try to convert ageFactor into a numerical variable again. Look what happens:

```
> as.numeric(ageFactor)
[1] 1 2 1 2 1 2 1 2
```

Why in the heck do we get 1 and 2 as outputs, when we expected to get 21 and 45? Quite simply, R gave you what you deserved. The fundamental idea of using factors is that the assigned scores are irrelevant. It shouldn't matter whether the assigned score is 1 or 21. The key is that all respondents in that group are given the same representation. All that matters is membership, not numerical scores.

Perhaps this table will clear this up a bit.

numerical value of ageFactor	The associated level is a text string	The level can be retrieved as a character string by	That text string can be converted to a number by
1	"21"	levels(ageFactor)[1]	as.numeric("21")
2	"45"	levels(ageFactor)[2]	as.numeric("45")

To be completely explict about this, the following R code shows the step by step process that could be followed to re-create the age variable with numerical values 21 and 45. This is not the most concise, efficient way, but I believe it is the most understandable way.

```
> ageFactorLevels ← levels(ageFactor)
> ageFactorLevels
[1] "21" "45"
> asNumericAgeFactor ← as.numeric(ageFactor)
> asNumericAgeFactor
[1] 1 2 1 2 1 2 1 2
> ageFactorTextValues ← ageFactorLevels[asNumericAgeFactor]
> ageFactorTextValues
[1] "21" "45" "21" "45" "21" "45" "21" "45"
> newNumericAge ← as.numeric(ageFactorTextValues)
> newNumericAge
[1] 21 45 21 45 21 45 21 45
```

But people don't actually do all of those steps. The factor help page has that Warning and it gives a one line recipe to get all of that work done.

```
> newNumericAge ← as.numeric(levels(ageFactor))[ageFactor]
```

This recommended approach is more efficient than the step-by-step approach I describe. My example applies as.numeric to the text string for each case, and I might be wasting a lot of computing effort by doing it that way. If I had instead applied as.numeric to the levels themselves, then the conversion of values from text to numeric would occur only for 2 values, “21” and “45”, rather than a string of text values, c(“21”,”45”,”21”,”45”,”21”,”45”).

To my eye, that factor Warning is magical, almost enchanting. If you understand it, you understand R.

5.8 Recoding Variables

5.8.1 Numerical variables

5.8.1.1 Mathematical re-calculations.

You can create new variables by doing any mathematical calculations available in R. There are tons of them.

```
> new1 ← log(paultest$x1)
```

Now ”new1” is a variable that exists in your worksheet. It is not part of the dataframe paultest, however. You have to tell it if you want that to happen:

```
> paultest$new2 ← log(paultest$x1)
```

5.8.1.2 Use ifelse

R offers a full programming language, with an extensive set of conditional operators. I won’t summarize them here. But I can demonstrate a few handy ones.

The ifelse function takes 3 arguments. To be verbose,

```
> newVariable ← ifelse(logical_condition, out_if_yes, out_if_no)
```

For example, we could create a new variable dichotomous variable “rich” by splitting a variable income at 100000

```
> rich ← ifelse( income > 100000, Rich, NotRich)
```

The ifelse can be nested in other ifelse commands, as in

```
> rich ← ifelse ( income > 100000, Rich, ifelse( income > 30000, MiddleClass,
    Poor))
```

This code uses the %in% command to achieve much the same purpose.

```
> dat$scale1 [dat$PLACE33num %in% c(9, 6, 29, 32)] ← 0
> dat$scale1 [dat$PLACE33num %in% c(22,27,33, 23, 7)] ← 1
> dat$scale1 [dat$PLACE33num %in% c(4 , 24, 26, 20, 21,
   5, 30, 31, 1,3, 28, 17, 16, 14, 2, 18)] ← 2
> dat$scale1 [dat$PLACE33num %in% c(8,12,10,11,19,25,13,15)] ← 3
```

5.8.1.3 Categorize a continous variable: use "cut" to break a continuous variable into ranges

Read ?cut, try it out on x <- rnorm(100). Cut returns a factor variable.

Here is an example how one can find the ranges (in this case the halfway points) and then use them to divide a sample.

```
qdividers ← quantile(CRIME98, probs=c(0.0, 0.5, 1.0))
nCRIME98 ← cut(dat$CRIME98, breaks=qdividers)
```

5.8.1.4 Convert a numeric variable into factor

Numerical variables can be translated literally into factors. The R framework would recommend that we ought to code a categorical variable using memorable names or initials, such as "M" and "F" for male and female. However, it is often the case that people collect data and code these scores as 1 and 2, for example. So we have the chore of translating the numeric variable into a factor.

On one level, this is a simple and easy problem. Each observed numeric value will be treated as a factor level. So if a variable is coded, for example, with values like

```
> x ← c(1, 3, 99, 1, 5, 3, 7, 5, 99)
```

the factor command will readily turn that into a categorical variable for which the levels are labeled with those numbers.

```
> xfac ← factor(x)
```

However, that doesn't really do any good. The new categorical variable doesn't have any new information:

```
> table(x,xfac)
    xfac
x     1 3 5 7 99
  1   2 0 0 0  0
  3   0 2 0 0  0
  5   0 0 2 0  0
  7   0 0 0 1  0
  99  0 0 0 0  2
```

More detail is required. The factor function needs to know the values we are "bringing in" and we assign a vector of labels to make them meaningful.

```
> xfac<-factor(x, levels=c(1,3,5,7,99), labels=c("Catholic","Protestant", "
    Muslim", "Jewish", "Refused"))
```

Note that the new variable "xfac" uses the correct labels when we tabulate it against the original.

```
> table(x,xfac)
    xfac
x     Catholic Protestant Muslim Jewish Refused
  1          2          0      0      0       0
  3          0          2      0      0       0
  5          0          0      2      0       0
  7          0          0      0      1       0
  99         0          0      0      0       2
```

Consider reading about the functions factor() and as.factor() or ordered() and as.ordered()

5.8.2 Recoding Factors

5.8.2.1 re-group levels of a factor variable

Here we consider a variable has sevaral different levels that are substantively identical. For example, a variable that is coded with levels "Red Fish", "Blue Fish", "Old Fish", "New Fish", and "Elephant" may have too much information about fish. We wish there were only two values, "Fish", and "Elephant". I have found this to be a surprisingly difficult chore. A person who masters it will have a deep understanding of factors and, for that matter, R in general.

Recently I've done a research project using a data set from the Social Capital Benchmark Survey, and it has many factor variables. Here are some examples of ways in which I've recoded factors. This code is ugly because the variables come from an SPSS dataset, and in SPSS the tradition has been to name all variables with capital letters.

First, there is a factor with 4 values representing trust in neighbors. We want to collapse the first 3 levels. This approach works to put together the three low levels of trust.

```
dat$NOTRNEI <- NA
dat$NOTRNEI [dat$TRNEI %in% levels(dat$TRNEI)[1:3] ] <-
"LittleOrMore"
dat$NOTRNEI [dat$TRNEI %in% levels(dat$TRNEI)[4] ] <- "NotAtAll"
dat$NOTRNEI <- factor(dat$NOTRNEI)
```

The following creates a new variable NEWTRNEI, and it puts together the respondents in the 2 lowest levels.

```
dat$NEWTRNEI ← NA
dat$NEWTRNEI [dat$TRNEI %in% levels(dat$TRNEI)[1:2] ] ← "LotOrSome"
dat$NEWTRNEI [dat$TRNEI %in% levels(dat$TRNEI)[3] ] ← "Little"
dat$NEWTRNEI [dat$TRNEI %in% levels(dat$TRNEI)[4] ] ← "NotAtAll"
```

The work is not finished. This has grouped together the lower categories, but it has not eliminated the labels of the now unused levels from the variable itself. That is to say, in addition to the levels we want, it also has "A lot" or "Some" in the list of possible scores, but there are no cases left that are assigned those values. There are several ways to get rid of these unused levels, I find the easiest is this:

```
dat$NEWTRNEI ← factor(dat$NEWTRNEI)
```

Running the variable through the "factor machine" cleans up the result. The new variable will have the valid levels "LotOrSome", "Little", and "NotAtAll". The factor function, by default, scans the levels that are actually used in the input variable and it purges the unused levels.

Most statistical procedures in R will know what to do when they encounter a factor variable; that's why it is worth the effort to code these properly. Most procedures will select one level as a "reference category" against which the others are compared. By default, the first level will be treated as the reference value. In order to change that, the relevel() function can be used to set "NotAtAll" as the reference level.

```
dat$NEWTRNEI ← relevel(dat$NEWTRNEI, "NotAtAll")
```

6 Emacs Tips

I once met a person who said Emacs was his operating system. He went to work, started Emacs, wrote in it, read mail in it, browsed the web in it, read Usenet in it, kept his calendar and diary in it and (mind you, I did not believe this part) he sometimes cooked lunch with it. Emacs can be a whole way of life, if you want it to be, because it is very "extensible." It is open to the creation of modules to do all kinds of things.

I do not live in Emacs and I have never used most of its features. I have read the Emacs tutorial many times and have repeatedly been amazed by the un-helpfulness of it. It reflects the fact that Emacs was written before there were mouses on computers and graphical user interfaces. If you have a mouse and a GUI, you can do just about anything with Emacs, so there is not much benefit in remembering lots of complicated keystroke sequences.

But there are a few that ***really help***.

In the following notes, we use the Emacs traditional notation

M the "alt" key on a standard keyboard, also known as the "meta" key.

C the "control" key

F1-F12 the "function" keys

Middle-mouse-click selects items

Left-mouse-click highlights selections

In the Emacs screen, a "**buffer**" is a separate "file" or a "text container". You have many buffers open at once, almost all the time. They are listed & selected under "Buffers." A **frame** is what we modern MS Windows users usually think of as a window–it shows content inside a rectangular box. I prefer to have one buffer show in each frame. Many Emacs old timers prefer to have several buffers showing at once by splitting their main frame into several smaller sections. Those frame sections are called **windows**. (See what I mean if you click "File" and then "Split Window".

Note there's a "Status" or "information" bar at the bottom, and below that there is the famous "mini-buffer." The mini-buffer is the place where Emacs interacts with you. It asks questions and you type answers.

Using those terms & symbols, here are the keys I remember from the top of my head. You can learn most of these by being observant. Most are listed in the menus, after all.

C-g cancels you out of the minibuffer. Sometimes you start doing something and make a mistake and just want to forget it and get back to the document. C-g is a "get out of jail free" card.

C-w cuts highlighted text. The w stands for "wipe". In MS Windows keyboards, we are used to C-x. But in Emacs, C-x is used for something else.

C-y pastes in the most recent cut or copy material. Remember the middle mouse button will "paste" the most recent selection.

M-w copies selected text. Note that this is not necessary if you select a section with the mouse. It is automatically selected. And the minibuffer warns you of that fact every time you highlight and hit M-w.

C-_ control-shift-underscore is "undo". You can hit that sequence over and over.

C-s control with small "s" starts a search. Watch the status bar and you should see "I-search:" appear, and then when you start typing letters, Emacs will immediately start fitting the letters you type. As soon as you finish typing, and it finds the first match, then hit C-s again to find the next one.

C-% control-shift-5 is Emacs version of search and replace. Very extremely handy. If you want to say yes to all, do exclamation mark "!"

tab key Emacs lets you get "auto completion" by hitting the tab key.

Shift and left mouse click make fonts bigger

There are giant manuals full of Emacs shortcuts ("macro key sequences") but I don't have much patience for them.

There are some handy things that most students never use. These are things you can do with the pull down menus:

File/NewFrame opens a new frame.

File/Open directory opens the current directory in a "file manager-ish" sort of mode. Can delete files, copy them. It is mostly good for viewing things and changing permissions.

Tools/Compare find differences between 2 files or buffers. Handy if you wrote 2 versions and wonder what the difference is.

Tools/Display Speedbar if you have a bunch of files and you want to look in them quickly, this is good. Don't forget the middle button is the selector.

There are also lots of jokes in Emacs, but most are not very funny. You can read a system manual ("man") page if you do "M-x man" and then tell it the name of the command you want to read about, such as "ls" or "cd". If you do "M-x woman" and do the same thing,

then you get a beautiful but more complicated version of the same. Don't forget, the middle mouse click selects items in Emacs.

Emacs reads a system configuration as well as your personal configuration in ~/.emacs when it starts. If you notice Emacs does something really weird, it probably means some funny thing got saved in ~/.emacs. Either edit that file or delete it to fix the problem.

7 Document Preparation: LaTeX and Lyx

7.1 Simplest Possible Introduction to LaTeX for people who will eventually use LyX.

Here's the smallest LaTeX document I can provide. Call it fred.tex.

```
\documentclass{article}
%%This is the preamble, where many options
%%can be specified for more complicated
%%documents

\makeatletter

\makeatother

\begin{document}

\author{Paul Johnson}

\date{January 16, 2008}

\title{Very Short Document in \LaTeX{}}

\maketitle
Here's the smallest \LaTeX{} document I can provide.

Type any crap you want here.

Use blank lines to separate paragraphs.


\end{document}
```

I want to demonstrate the step-by-step process that is used to translate "fred.tex" into a beautiful document.

You can follow allong if you download the file "fred.tex" from my course page `http://pj.freefaculty.org/stat/ps706` and save it in your working directory.

Open a terminal and navigate to the directory where you put fred.tex.

Open fred.tex in an editor, I recommend Emacs. (On Linux, it should be as easy as typing "emacs fred.tex"; on Windows, the Emacs executable is called "runemacs" and if it is in your path, then you type "runemacs frex.tex.") Go into the middle of the document, type in some crap. Any crap you want. Then Save and Close.

You are back in the terminal. You put to use the "backend" tools of the LaTeX system to process the source file "fred.tex." Hopefully, your LaTeX system provides you with the key programs like "latex" and "pdflatex".

- Use the latex program to process your prepared tex document, fred.tex:

  ```
  latex fred.tex
  ```

- List the files in the directory to be sure a new file "fred.dvi" was created. DVI is short for "device independent" file. DVI was a precursor to the more familiar PDF ("portable document format").

- To view the resulting DVI file, file.dvi, use a dvi viewing program, such as xdvi or kdvi (or, in Windows, the preferred DVI viewer seems to be yapp):

  ```
  xdvi fred.dvi
  ```

- You can send a dvi to a coauthor, but other readers generally want a pdf or a postscript file.

  To produce a PostScript file from the DVI file:

  ```
  dvips fred.dvi
  ```

- To view the resulting PS file, use a postscript viewer, such as evince, gsview32, gv (short for ghost view). On Linux, I type

  ```
  evince file.ps
  ```

- To print the PS file, use the lpr program to send it to a printer.

  ```
  lpr fred.ps
  ```

  That will go to the "default" printer in your system.

- Here, I tell it to go to the printer called "lab" with the -P option:

  ```
  lpr -P lab fred.ps
  ```

- The ps file can be converted to fred.pdf with a command like this:

```
ps2pdf fred.ps
```

That conversion may not be perfect if there are some fancy features in your ps file that the converter can't handle. fred.tex should come through just fine, but if I had inserted a "psfig" drawing or other postscript tricks, there could be trouble.

There are other ways to create PDF files.

- There's a program "pdflatex" that will do all the work in one step. From this command,

  ```
  pdflatex fred.tex
  ```

  one will receive a file called "fred.pdf".

Honestly, sometimes pdf output is not great looking. Sometimes there are details that are needed and I don't remember all of them. A script called "tex2pdf" seems to do a better job with lyx files. I can share it to you if you want. Or you can get your own here: `http://tex2pdf.berlios.de/`.

In case you want to write LaTeX documents that have non-roman characters (for example, Chinese or Japanese), there is another document processor you might try. The XeTeX package provides a replacement of pdflatex called "xelatex". I've written a HOWTO document for XeTeX and you should be able to find it either on the LyX website or in the folder `http://pj.freefaculty.org/latex`.

7.2 Knowing Just the Smallest Bit about LaTeX and a Lot about LyX

LaTeX is a format for preparing documents and a program that turns documents into beautiful output. LyX is a Graphical User Interface (GUI) to the LaTeX document preparation system. It is like a word processor, but is great for writing equations and preparing academic documents. The LyX program is something like a word processor, in that there are pull down menus that can be used to achieve most purposes. LyX saves documents in its own format. By default, files are saved with the suffix .lyx. LyX also has an Export option to create an actual LaTeX document, one that will be suffixed with .tex.

LaTex is a 'structured document' environment. Everything you type has a style environment associated with it, and so a publisher knows exactly how to turn your document into a pleasant looking thing.

The core LaTeX philosophy is that authors should not waste time on formatting. Authors should work on words, sentences, equations, paragraphs, and the like. No author should ever fuss over the question of whether a section title should be indented or not. Authors

should not fiddle with paragraph indentation, or anything else, because publishers choose those options. Authors should write documents in a more or less "generic" way so that they can be turned into an article or book in the style of any publisher. Authors are not supposed to fuss about trying to indent quotations "just so." LaTeX ignores most extra blank spaces in documents. It ignores tabs. The publisher styles are the "sty" and "cls" files that make up the collected LaTeX hierarchy of "packages" that is stored in /usr/share/texmf/tex/latex.

There are various ways to work in LaTeX documents. One alternative is to actually stay on the "ground floor" and write in an environment where you actually see the LaTeX markup. The editor Emacs has an optional mode called AucTeX that provides some convenience features for people writing in LaTeX. There are also editors like "TeXmaker" or "kile" that are specilized for working with LaTeX documents. In those editors, the LaTeX code is edited without aid of any simplifying representation. For example, whenever you want the symbol LaTeX to appear, you write

```
\LaTeX{}
```

That is a "macro" that tells the processor to produce that particular symbol.

If you write in LaTeX, on the ground floor, your documents are full of "mark up" like this:

```
\begin{equation}
e = mc^2
\end{equation}
```

That markup produces beautiful output like this:

$$e = mc^2 \tag{7.1}$$

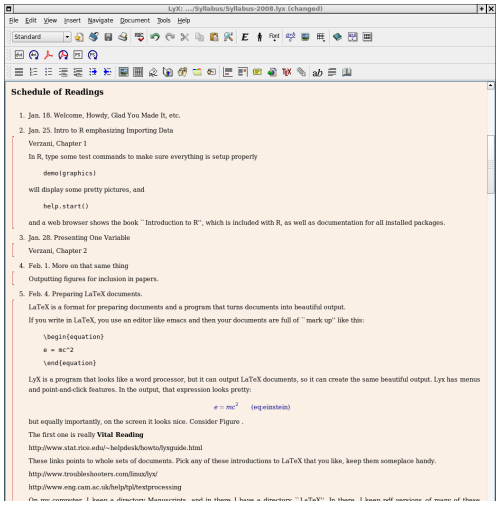
In-line mathematical symbols are surrounded in dollar signs, $\alpha \times \beta$ would print out as $\alpha \times \beta$. Subscripts are triggered by underscores and superscripts by the carot, as in β_{ij} or $\Gamma^{whatever}$.

A mathematics professor that I know uses LaTeX in that "ground floor" way. He edits the files in Emacs and uses some Emacs addons to assist in the processing and viewing of the document. I have done that, but it is hard work.

To the "hard core traditionalist," the argument I am about to make is wrong. I don't want to hide from that fact. I wear their scorn with a badge of honor (apologies to Dan Quayle). I believe that most people that I teach can make more headway running the program LyX, which is now available for Unix/Linux, Macintosh, and MS Windows. Here's my argument for taking the easy road.

LaTeX is great, but difficult to manage if you are not really, really devoted to it. You also need to be in an academic environment where many people use it. A document preamble can have pages and pages of options, most of which are difficult to master.

Figure 7.1: Screenshot of the LyX Program



LyX is a program that looks like a word processor, but it can output LaTeX documents, so it can create the same beautiful output. Lyx has menus and point-and-click features. Consider the screenshot in Figure 7.1.

In LyX, when I want LaTeX to appear in my LyX document, I simply type `LaTeX` and LyX does the rest of the work.

The main selling point for LyX is that it makes it easier, possibly even simple, to create complicated, professional looking mathematical expressions. I believe the only way to learn about that is to actually try it, so I've designed some elaborate exercises that appear in the end of this chapter.

LyX is provided with a voluminous amount of documentation. When you run LyX for the first time, LyX provides you with a list of things worth knowing. Under the Help menu item, one should find a Tutorial, a User Guide, and Extended User Guide, a Customization manual, and other things as well. I am one of the people who tries to understand instructions when I build bicycles or repair furnaces, so I have to admit I did spend several hours pointing and clicking in the LyX online guides. I realize that many people do not read instructions, and instead they prefer to try, and when they fail, to either write to the lyx-help email list or ask a professor what to do. For people who don't read manuals, the usual answer is "RTFM", which stands for "Read the Fine Manual." The word Fine may be replaced by other words that start with F, of course.

7.3 The LyX Document Framework.

When I open a LyX document, I try to use a "template" that has the settings I want. To create the template, I experiment until I have everthing set up in the way want, then I delete

the content of the document except for, say, my name, and then I save it. When I create documents in the future, I specify that saved version as the template.

Sometimes I forget to start with a template. Too bad. Then I have to re-do the settings to fit my taste. I've made that mistake on purpose and kept careful notes on what is necessary to convert the standard LyX document into a workable document.

Most of the work is done in the Document menu, under the Settings item. After you explore that a bit, you may think it is overwhelming in its detail. I don't have a comprehensive knowledge of all options, even after using LyX for a decade. However, there are just a few really vital things to check.

1. The Document Type is the first big choice. This book that you are reading was set with the KOMA script style for books, but most of the time I use the "article" format without any special settings for the postscript driver.

2. Change margins. Document → Settings → Page Margins. Set all margins to 1 inch.

3. Change Fonts. For me, the Latin Modern fonts make the best PDF output. Document →Settings→Fonts. For Roman, Sans Serif, and Typewriter, I choose the Latin Modern. I generally change the font base size to 12.

4. Make sure your paper is the correct size. Document → Settings → Page Layout. I set the item "Format" at "US Letter" (because I'm in America, obviously).

5. In the Document → Settings → Language, I usually change the input encoding to utf8 if I'm writing a document that will be produced in English.

   There is some danger here. If I cut-and-paste from other programs into lyx, and they use some other encoding, then there may be problems later. Sometimes characters will look like noise on the screen, sometimes the latex processor will complain.

6. I prefer ragged-right edges in output, not right-justified output as in books. That is achieved by putting this into the Document preamble. In Documents→Settings→LaTeX Preamble, put in this:

   ```
   \usepackage{ragged2e}
   \RaggedRight
   \setlength{\parindent}{1 em}
   ```

7. This next thing is optional for new users, but I include it because it demonstrates the customizability of LaTex and LyX. LaTeX does not automatically center graphics inside figures. You can manually center each graphic. Instead of manually centering each I have a preamble element that is supposed to do that for me. It goes like this.

```
%I forget where I got this, but it works, usually :)
\usepackage{ifthen}
\makeatletter
\renewenvironment{figure}[1][]{%
 \ifthenelse{\equal{#1}{}}{%
   \@float{figure}
 }{%
   \@float{figure}[#1]%
 }%
 \centering
}{%
 \end@float
}
\renewenvironment{table}[1][]{%
 \ifthenelse{\equal{#1}{}}{%
   \@float{table}
 }{%
   \@float{table}[#1]%
 }%
 \centering
}{%
 \end@float
}
```

If you do not use this preamble item, then it is necessary to manually center every graphic inside every float in your document. That is not a horrible problem, it is as simple as putting the cursor on the left edge of the graphic, hittting the Edit menu, choosing Paragraph Settings and then choose center. That can get tedious if you have a lot of figures.

7.4 Exercises

Exercise 1 offers the "bare minimum" LyX user introduction. Exercise 2 shows a little R program that creates some images, that you can then insert in your document. Exercise 3 is the big whopper, complete come-full-circle, super-great thingie that shows how the giant LaTeX experience should integrate references.

1. Create a bare bones document that uses a little math. I have a document that can serve as a "template" for your effort: `http://pj.freefaculty.org/stat/LyX-template-01.lyx`. I created that with an older version of LyX, but I believe it will work with newer version as well. Download that, and then re-name it and open it up. I always run LyX in a terminal, because error messages sometimes flash there. Some lazy people start LyX from an icon launcher. Those people get what they deserve (trouble).

a) Put your name at the top. Then with the cursor in your name (anwhere in your name), click on the Environment chooser on the left side (where it probably says "standard") and change it to "author". Then hit return to start a new line, type in a title, and then change that paragrph style to title.

b) Type something and then mark it with the title environment.

c) Then write whatever you want.

d) Test an in-line mathematical expresion.

C means the control key. Type C-m (Control and m at the same time)
That creates a "math entry box." Inside that blue math entry box, type this, including the back slash, and put spaces at the end of the words:

\beta (hit space bar) \alpha (hit space bar) \gamma (hit space bar)

That should cause these symbols to "pop up" on the screen:
$\beta\alpha\gamma$
Notice that when you hit the space bar, the Greek letter appears. The space bar has the effect of telling the program to translate \beta into β.

The "math entry box" will close as soon as you hit the space bar a second time (since you don't have any \ symbols that are still "open")

There is a graphical interface for choosing mathematical symbols, but it has changed in appearance over the years and I'm hesitant to tell you where exactly it might be. In earlier versions, there is a "toggle math panel" item under the Insert->Math option. In LyX version 1.6, the approach is to choose View->Toolbars, and choose the ones you want. I do the math(auto). After that, when you hit C-m, then a pointy-clicky math menu will appear around the edge of your screen.

If you have a really old version of LyX, you can bring up a panel of mathematical symbols called the "math panel". It floats separate from the document. That was my favorite, but it is apparently gone from the new versions.

e) Start an indented display equation with Shift-Control-m. That's the same kind of math entry box as before, except this one is indented and centered. Tiy can click to put in symbols.
When you are done choosing symbols from the math panel, you must click in your document again to tell LyX you want to type more.

f) Try out **cross-referencing.** One of the REALLY BIG features about LyX is that items can be labeled and then when you put in cross-references, the LaTeX processing system will make all of the numbers match up. For example, if you insert Figures, Lyx numbers them for you, and if you insert a label inside the

figure's title, then that label can be used as a cross reference. Similarly, labels can be inserted inside equations or other things.

First: let's label a mathematical equation. Choose the display equation (a centered math thing you get from C-M). Position the cursor at the end of the math insert box, hit the Insert menu and choose "label". The prompt will give the default "eq:" and you can put any word you like after that. On the screen, there will be a marker like #eqn, but just wait–the equation will be properly numbered in the output.

Second: insert a reference to that equation. Suppose you are typing a sentence like "In equation XXX, we find blah blah". You want the XXX to be replaced by the correct number. Go the the XXX in the text, hit the Insert button, choose Cross-Reference, and you should see a menu with all of the labels you have created so far. Choose the right one, and then view the DVI output of your document.

You can put in labels anywhere, inside equations, figure and table titles, as well as in chapter, section, or other headings or enumerated lists. For example, in item 1c above, I inserted a label, and I just used the cross-reference feature to refer to that item.

As a result, NEVER manually number equations, figures, tables, or whatever. Let LyX/LaTeX do the numbering, and you will be happier!

g) Review your document. Hit the View menu, and choose DVI. That will show you the "device independent interface" result. Usually, that is almost exactly the same as what the postscript output would be. Next, choose View-> PDF(pdflatex). That will display a PDF file on the screen (assuming you have a pdf viewer).

   If you don't get a pleasant looking document on the screen, then you made some kind of mistake. We can usually fix those, I don't want to bother too much about it in this document. You get a mistake if you get really busy pointing and clicking and you create some mess that LaTeX can't understand. You can also get errors if you inadvertently insert an international character that LaTeX can't understand.

   The LyX team members advise that, if you want to print a document, you should do so from within the viewer, not from the File-> Print menu item in LyX. If you print from the viewer, you are most likely to get what you really expect from the printer.

h) Use LyX to export your document in PDF format. File→Export→PDF(pdflatex).

i) If you have a web page and know how to post files, then put your pdf file up there and email me a link to let me know where the file is. If you don't know how to post files on a web page, you should learn, but that's not my problem right now. Just email it to me. In either case, please put a clear subject heading on your email (Name-lyx-exercise.pdf or such) so I don't mistake it for spam.

j) Do this next step only for your own information. In LyX, export the TeX document. File→Export→LaTeX(pdflatex). This is the file that publishers want (along with all your graphics and insert files, of course). Use an editor like Emacs to inspect the output file, which will have the same file name as your LyX document, but the suffix will be .tex. Then try to use "latex" and "pdflatex" to process that tex file.

2. The second LyX exercise is mainly focused on exploring the process of creating "floating graphics." I've uploaded an example directory on my website in a file called "LyxExercise-2.tar.gz". Download that file and untar it (Linux: "tar xzvf LyxExercise-2.tar.gz"). It will create its own directory.

   One important lesson is that you need to keep track of the many different files the belong to a project. A good habit is to work in a separate directory for every project that you do.

   a) Run the R file called "MakeSillyPlots.R" that is in the tar.gz file. A copy of that program can be reviewed in the Algorithm 7.1. If it works, that program should dump out 2 postscript files (*.eps), 1 pdf file, 2 png files, and 1 xfig. If the computer you are using refuses to create a pdf or a fig file, you should just give up and fight with that configuration problem another day. My computer really does make all those output types.

   a) Create a LyX document that includes the results of your R work. You might as well start with a clean document. I included a file called "templateEmpty.lyx" in the tar.gz package. You don't have to use it, but I think it might help.

      Type in any old stuff you want. Then insert a figure float for each eps or png output file that you created with that R program. I am confident those formats will work on all types of operating systems.

      We treat the eps, pdf, and png files as "graphics" in LaTeX. Proceed as follows for each one.

   b) Here's how graphics are inserted in LaTeX documents with LyX. Position the cursor on a new line in the document.

      i. Choose Insert, then Float, then Figure.

      ii. Type the title of your figure–this is the title to be displayed to readers.

      iii. After you type the title, while you are still inside the little title box, hit the Insert menu, choose Label. The system will guess a label for your, something like "fig: your title here". Usually, I accept those.

      Hit return when finished typing your title and inserting the label. The label does not show for the reader, but you can use it for cross referencing inside your document.

Algorithm 7.1 MakeSillyPlots.R

```
x<- rnorm(333)
y<- rnorm(333)
## run this only if you are in an interactive session, so
## you can see this on the screen.
## plot ( x,y, xlab="Input Variable", ylab="Output Variable")
postscript(file="testplot-1.eps", horizontal=F, height=6, width=6, family="Times", o
plot ( x,y, xlab="Input Variable", ylab="Output Variable")
dev.off()
postscript(file="testplot-2.eps", horizontal=F, height=4,
width=4, family="Times", onefile=F, paper="special")
plot ( x,y, xlab="Input Variable", ylab="Output Variable")
dev.off()
pdf(file="testplot-1.pdf", height=6, width=6, family="Times",onefile=F, paper="speci
plot ( x,y, xlab="Input Variable", ylab="Output Variable")
dev.off()
png(file="testplot-1.png", height=350, width=550, type="Xlib")
plot ( x,y, xlab="Input Variable", ylab="Output Variable")
dev.off()
png(file="testplot-2.png", height=350, width=550, type="cairo")
plot ( x,y, xlab="Input Variable", ylab="Output Variable")
dev.off()
### Now dump out an xfig file
xfig(file="testplot.fig", height=6,width=6, family="Times", horizontal=F, onefile=F)
plot ( x,y, xlab="Input Variable", ylab="Output Variable")
dev.off()
```

So far, you have only created an empty "floating box" for something. Finally, put stuff in there!

iv. The cursor should be on a line by itself inside your float. If not, hit return to get a new line. Choose Insert, then Graphics, and in that menu, navigate to the EPS or PNG or PDF file you want to display. Note you can re-scale the image if you want to by using the "insert graphic" panel options.

v. By default, your graphic will not be centered within the float. You treat the image as a paragraph and center it by using the Edit→paragraph menu. If you use my LyX document template, this is not strictly necessary because I have a preamble component that is supposed to center graphics inside floats when you view your document.

I want you to try to repeat this process for all of the graphics that are produced by MakeSillyPlots.R.

Please note, LyX does not "copy and incorporate the graphic" into the lyx file itself. LyX is not like the Borg in Star Trek: The Next Generation. Rather, LyX (and LaTeX) just make use of the file that is sitting there. If you delete the graphic file, then LyX won't have that figure anymore and your document won't work. If you want to share your LyX document to somebody, you also have to send them copies of the graphics. Get it?

Try to experiment a bit with repositioning your image files. You might hack in MakeSillyPlots. R to see what happens. I recall something bad but interesting happens if you create a pdf output image without the paper="special" option.

c) This is a special feature of LyX, but it may not work on MS Windows unless you have the editor Xfig installed. Try and see what happens. Create a figure float. Then, instead of inserting a graphic, choose to insert external material.

Recent versions of LyX have added an option under Insert → File → External material. The external material is supposed to be a specialized kind of file that LyX understands. On my system, I see a External type for things like "Chess diagrams" and "RasterImages." I also see an option for Xfig drawings. The R program I provided creates a file "testplot.fig". It is formatted for the drawing editor XFig, a popular and long-standing Unix program. I'm a bit curious to know what LyX on MS Windows will do with an XFig file (maybe it will bomb). After inserting the XFig file as an External Document, LyX should be able to display it, as if it were a graphic file. But there also will be an "Edit" option. If XFig is installed, then something magical should happen.

XFig is mainly useful for "line art," diagrams that appear in publications. In the LyxExercise-2.tar.gz package, I included a fig drawing that I made for a Math book. The figure is called "SmithSchwartz5.fig." You can use xfig to edit that and experiment with it in LyX.

d) View the pdf file that is generated. I'd like to see it. If you have a web page and know how to post files, then put your pdf file up there and email me a link to let me know where the file is. If you don't know how to post files on a web page, you should learn, but that's not my problem right now. Just email it to me with a clear subject heading like "Lastname-lyx-exercise-2.pdf".

3. The next step is the last big, new thing. Learn how to make an automatic list of references in your document that includes only the things you actually cite. This is the point where the power of LaTeX becomes visible. LaTeX works together with a program BibTeX that will keep track of citation and build a list of references when this document is finished.

   Where do you get the bibliography file? I included a file called "SocialChoice.bib" in the LyxExercise-2.tar.gz.

   Inside LyX, it is necessary to set the type of bibliography that is to be used and let LyX know where to insert the references. I almost always want social science style citations, the type that have in-text references like Lastname (1994) or (Lastname 1994, p. 34).

   Proceed as follows.

   a) In Document → Settings → Bibliography, choose "Natbib".

   b) Move the cursor to the end of the document, click Insert → List/TOC → BibTeX Bibliography. Do 2 things.

      i. First, tell it where your reference files are. Click "browse", choose the bib file you want, click ok, then choose it again from the list.

      ii. Second, tell LyX what format you want to use for references. I use a format file I created called "apsa2.bst." It is installed in the LaTeX hierarchy on my machines, but I think it will also work fine if you keep that bst file in the same directory as your document and your bibliography. So choose browse and choose that.

   c) Test out your bibliography setup by moving into the text of the paper inserting a sample reference. Click Insert->Citation and you should see a list of books and articles appear. Choose one, and hopefully it shows up in your output. Since you chose Natbib format, you should have incredible control over how you want each particular citation to appear in your document. You can make things appear like Black (1958) or Black (1958) or (Black 1958) or Black (1958, p. 44).

      Warning: I had some troublewhen I inserted some citations that had had the wrong "character encoding". I had cut and pasted some citations from a web browser into my bibliography file and I was "contaminated" by some strange character encodings. If you try to view your document and it fails because of "unknown characters," don't panic. In my case, I just used Emacs to open the file SocialChoice.bib and make some fixes.

d) After inserting a few citations, in LyX try View→pdflatex and check to see if the list of references if properly built.

e) Once you see this actually work, you should be persuaded to drop the old, silly habit of writing a long paper and then typing in the references when you are done. Instead, you should build a bib file as you go along and then let LaTeX make the references for you.

   I'm giving you the bibliography file, just use it. Do you wonder where that came from? A bibliography can be directly edited with an editor like Emacs (or TINN, or GVIM, or any other editor that does not try to "format" things for you, such as MS Word). The bib file is just a carefully formatted "flat text" file. If you inspect my bib file, you should be able to discern a certain rhyme and reason to creating complete citations. LaTeX allows names either in the format "James Q. Wilson and Roger P. Pettibone" or "Wilson, James Q; Pettibone, Roger P.") I tried to use Emacs to create bibliography entries (there is a helper component in Emacs for that), but I found the formatting too tedious. I only use Emacs for "bug shooting" bib files.

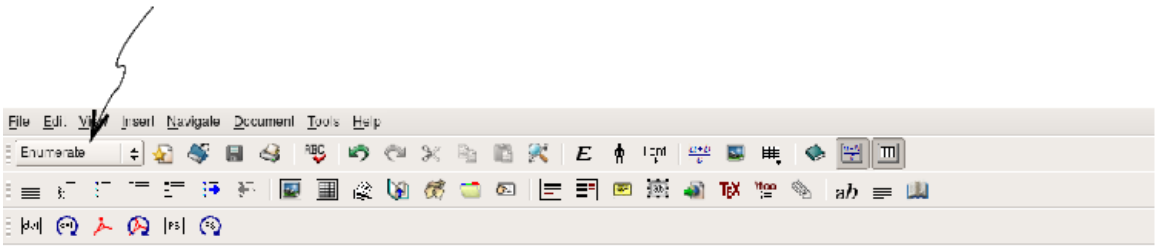
f) There is a pointy-clicky free program called pybliographic that is handy for this kind of thing. It works great in Linux. On Windows as well as Linux, I've also used a Java program called "JabRef" that is handy. Java programs run (more-or-less) on all operating systems that have Java, including Linux, Windows, and Macintosh. I'm told that people can also spend money on programs like Endnote that do this sort of work. EndNote can export a bibtex file that can be used with LaTeX.

In case you can't make the LyX file work with the bibliography, that's too bad. I have a working example file "pjCanCiteStuff.lyx" and if you put that in a folder with the bibliography and biblio-style file, you can test it on your system. If you open that file and view it in pdf, it should give you some encouragement.

7.5 LaTeX and LyX Paragraph Environments (Section)

In the title for this section, I have a marker `(Section)`, which I'm using to represent the idea that the words "LaTeX and LyX Paragraph Environments" are marked as a LaTeX Section in the document that I'm preparing. That tells LaTeX that the header is supposed to be numbered and set in a bold faced print with extra white space around it.

I started to wonder about all these different "environments." In LyX, there is a "menu" chooser that specifies the paragraph environment. I'm referring to that thing on the left side of the LyX graphical interface, where it currently shows "Enumerate" in this snapshot:



This chooser is used to specify the type of paragraph, e.g., a chapter title, numbered section, one item in a list, your name, and so forth. In the old MS Word days, we would call them "paragraph styles".

Why so many environments for paragraphs? What are they? What are they for?

In some special cases, like "Title", "Author" and "Abstract", the paragraph environment can have a dramatic effect of moving "text" into a desired position in the final document. If you put in your name, and mark that line as an "Author" environment, then your name will be moved to the front of the document when it is printed. The particular font and position is specified by the LaTeX document class.

In most other cases, the paragraph environment does not "move text around" so much as shape it in a desired way. By default, all ordinary text should be of the "Standard" type. Standard type will follow whatever conventions are established within the document. Most of the other paragraph types are for "section numbers" or "special formats".

I'm trying to make this a self documenting example, but it is tedious. I will use the `typewriter` font to insert markers for my paragraph environments, except when I'm using the "Standard" environment.

7.5.1 List types (`Subsection`)

Any `Section, Subsection, or Subsubsection` item is automatically numbered. A starred section item`, as in Section *` , is not numbered, but it is "set off" from the other text. That's what the star means–no number.

Enumerated List (`Subsubsection *`)

I now choose the paragraph style "Enumerated" (actually, I use the keyboard shortcut Alt-p e) and look what happens:

1. Enumerated list is automatically numbered. (`Enumerate`).

   Hit enter and LyX offers a second enumerated item

2. This is a second enumerated item. (`Enumerate`)

    a) If you hit return, and then do Alt-Shift-right arrow, then an "a" will pop up to start an indented enumerated item. (`Enumerate`)

    You can put this standard paragraph within the enumerated item if you want. It will "inherit" the indenting. After hitting return, change the paragraph type to "standard" and then hit the button to "increase list depth" so the paragraph "goes under" the enumerated item. If you can't find the button, then click "Edit" and choose "Increase List Depth". I use a shortcut key (shift-Alt-right arrow). (`Standard`).

    b) Enumeration in that sub-level continues until you make it stop (`Enumerate`).

    If you want to go to a higher enumeration level, you can hit enter, and then either do "Edit" and "Decreate List Depth" or use a shortcut key shift-Alt-left-arrow.

    To stop adding items, hit return and change the environment type to "standard", for example (Alt-p s).

    c) Just to be perfectly clear, when you are in an Enumerate item, and hit return, you get a new number in the sequence.(`Enumerate`)

        i. To go to a sub point, hit return then shift-alt-right-arrow, which is the same as Edit-> increase item depth.(`Enumerate`)

            A. Hit enter again, do shift-alt-right-arrow again, to get another subpoint(`Enumerate`)

        ii. Hit enter again and you are stuck adding more items. (`Enumerate`)

            A. To go to a higher level in the outline, do shift-Alt-left arrow (or decrease item depth).(`Enumerate`)

            B. To quit adding enumerated items, hit return and change the paragraph type.(`Enumerate`)

Itemized List (`Subsubsection *`)

- Itemized Items have bullets (`Itemize`).

  – To start a list, I use shortcut Alt-p i. But you can menu pulldown "Itemize"(`Itemize`).

    * One can put paragraphs within items. Hit return at the end of the item, then do Edit-> increase item depth or with a shortcut shift-Alt-right-arrow. Then change the paragraph type to Standard. In LyX, a red bracket should appear on the left to indicate you are inside another environment.(`Itemize`).

A Description List (`Subsubsection *`)

Thing the word on the left is described here. (`Description`)

Item a single word or Control-shift separated words can be on the left side of the description.(`Description`).

Here's a key term a description has a big bold faced phrase on the left and then a longer description on the right. Note ordinary spaces will separate your terms on the left side. Use Control-Space to hold them together. (`Description`)

Like all "list" environments, a description environment will keep making more description items when you hit enter. (`Description`)

I don't use many of these description lists because I don't need them too much. (`Description`)

7.5.2 Special Types of Paragraphs (`Subsection`)

Quote Me (`Subsubsection *`)

Two environments start with our favorite letter, Q!(`Standard`)

Quote (`Subsubsection *`)

> Here is a short quote. One wonders how a "quote" is different from a "quotation" in LaTeX? Many people have wondered the same thing. The answer: Indentation (`Quote`).
>
> > When you hit return in a "quote", it continues in the quote, with no indentation. Sometimes people want to have no indentation, even though it seems like they ought to.(`Quote`)
> >
> > See, the quote continues with no indentation.(`Quote`)

Quotation (`Subsubsection *`)

> Here is a short quotation. One wonders how a "quote" is different from a "quotation" in LaTeX? Many people have wondered the same thing. The answer: Indentation (`Quotation`).
>
> When you hit return in a quotation, it is indented (`Quotation`)

Verse (`Subsubsubsection *`**)**

On the other hand, the Verse environment might be used by a poet or a social scientist who had a peculiar style for the presentation(`Verse`)

of ideas.(`Verse`)

Lyx-Code (`Subsubsection *`**)**

```
In case  you want a more or less verbatim presentation of what you type, with a typ
It is often used for computer code examples, as in R-Code.(Lyx-Code)
The Document class will usually cause this to be set in a type writer font.  I susp
I'm getting some very peculiar line breaks in this example, and it makes me suspect
```

Paragraph (`Subsubsection *`**)**

More than one person has come to me in a tizzy because the "Paragraph" environment is not what they expect. (`Paragraph`**)**

I have to admit that I don't really know why a person wants the "Paragraph" environment." I've seen more than one student ruin a paper by using the paragraph environment. I never have used one. (`Paragraph`**)** In the LyX documentation, I think it says the paragraph environment is for some special purposes, and I don't know what they might be. I do notice a very strange thing. In my output, this "Standard" environment gets merged onto the "Paragraph" output in the line before.(`Standard`)

7.5.3 I am finished (`Subsection`).

In the article style (KOMA script), LyX offers the environments I've demonstrated, plus a few more for abstracts and such. But if you understand the role of "Standard", "Section", "Section *", and "Enumerat"ed lists, I expect you will understand the main piece.

7.6 Keybard Shortcuts I Remember Off the Top of my Head

In the very bottom of the LyX graphical user interface, there is a small notification bar that tells you what's going on. As I'm typing, it says "Font: Default." But if I choose a menu

item, then a keyboard shortcut for that menu item will flash up on the screen. Choose the "Enumerate" item from the paragraph style choser, and the notification bar pops up with (layout Enumerate: Alt+P E,Alt+P N).

That means that if I had typed Alt with P, and then after releasing all keys, then hit either "e" or "n", then I would have started an enumerated paragraph.

I learn most of the shortcuts in lyx by watching that little bar. Before today, it did not occur to me that the notification bar uses capital letters, but it should not, as far as I can tell. If you want to create a "description", then you pull down to "Description" in the menu chooser, the shortcut notification bar says "Alt+p D" but instead it shoudl say "Alt+p d". I may try to look into that, it seems like an obvious problem.

Math Mode shortcuts I remember

_

^

\sum

\int

\hat

\widehat

\bar

\alpha

\beta

\gamma

\Gamma

\infty

\times

\cdots

\ldots

\vdots

I usually have to look up the other symbols.

Nonmath Mode Shortcuts I remember.

shift+Alt+right_arrow

shift+Alt+left_arrow

Alt+p e

Alt+p s

8 R Plot and the Mystery of Object-Orientation Computer Programs

8.1 Introduction: The Mystical, Magical plot Command.

The plot command can create many different sorts of graphs. If only a few options are specified, plot "fills in" the rest with its best guesses. Users can add lines, boxes, circles, shaded smiley faces, and dirty words. In a way, an R plot is like an "electronic piece of graph paper" that has points, text, lines, and symbols. The output doesn't stop at the monitor, however. There are many possible output formats for R plots. One can draw a "picture" into a file in a format like like "png" or "jpg." One can create a vector graphics output file, such as "pdf" or "eps". One can even create output in formats that are intended for editing software, like "xfig" or "svg." Plot commands can be scripted, so that one can cycle through hundreds of datasets and save plots automatically for later inspection.

plot is one of the "old" S functions. It played a major part in the initial success of S. The idea that a user could generate a "publication quality" illustration without the help of a professional graphic artist was somewhat surprising. (For my first published article in 1986, it was necessary to pay a professional illustrator $166 prepare some "line art." I could generate the same figure with R today in about 10 minutes).

There are always new packages for plotting in R, but I have a fondness for plot. It makes nice graphs, but the newer plotting packages do too. I like plot because of what it demonstrates about the S and R as languages. The plot function has a lot of "old world charm," but it has a lot of modern conveniences. I believe that, if a person understands the 2-dimensional plotting framework in R, then he or she will have a pretty deep understanding of the way that R works.

The plot command has some qualities that make it an especially good example of the original intention of the S/R language framework. Simply put, plot tries to give you what you need, even if it is not exactly what you want, or even, what you think you are asking for. One may obtain a nice looking plot from a command as simple as :

```
plot(whatever)
```

where *whatever* is literally anything in R. Something entertaining may result from adding another object as well:

```
plot(whatever, something)
```

Different inputs to the plot command generate different types of output. The details will be explored later in this chapter, try not to worry about them too much at this point. The major point right now is that plot interprets what we give it.

For purposes of illustration, a selection of variables was gathered from the 2006 General Social Survey, a large national sample of American respondents. Compare the plots in Figures 8.1 through 8.3. All of these are the result of a simple command of the form:

```
plot(y ~ x)
```

where x is the independent (or "input") variable, the one that appears on the horizontal axis, and y is the dependent (or "output" variable), the one that appears on the vertical axis. The difference among the plots is driven purely by the type of variables that are included in that command. No other changes–no new options of any sort–are used to produce those three figures.

A scatterplot (Figure 8.1) results when we ask for a plot and x and y are both numeric.

```
plot(hrs1 ~ age)
```

In this case, we see the relationship between the number of hours worked per week (the variable named *hrs1*) against the respondent's *age* (in years). Some senior citizens work much more than one might expect, it appears.

A boxplot (Figure 8.2) results if we replace the numeric input variable *age* with a categorical variable *sex*.

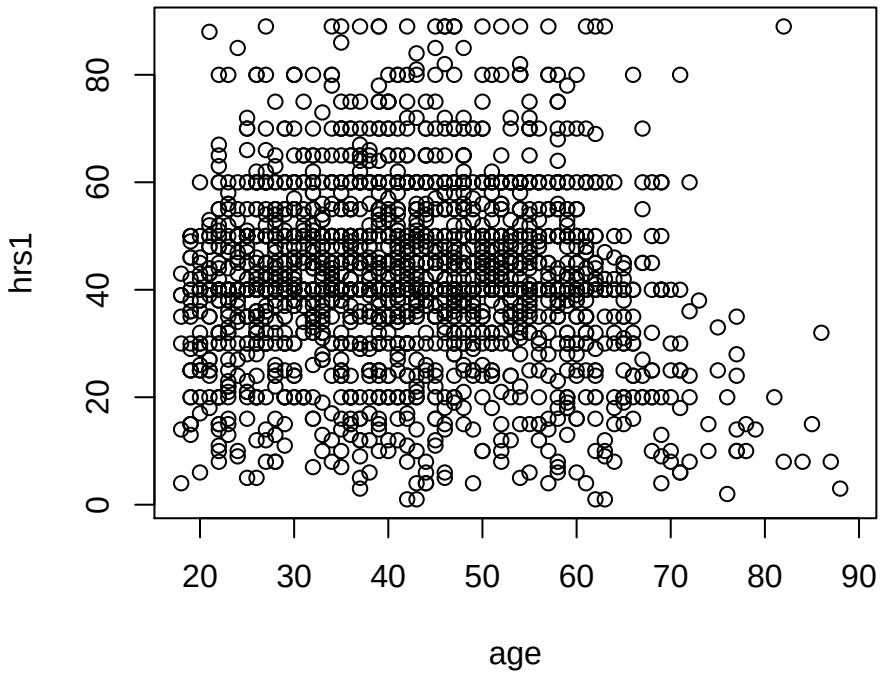
```
plot(hrs1 ~ sex)
```

For each value of *sex*, a vertical bar indicates the range of outcomes.

The illustration in Figure 8.3 is called a "spineplot." It results if we replace *hrs1* with a categorical variable, such as the respondent's favorite in the 2004 presidential election *(pres04)*.

```
plot(pres04 ~ sex)
```

Figure 8.1: Scatterplot: Plot two numeric variables



The spineplot is not my favorite type of plot; I find it difficult to read. Essentially, it is a "stacked barplot," but with a twist. The width of the bars represents the proportion of cases for each sex.

Instead of a set of stacked bars, I would much rather see a side-by-side barplot. In Figure 8.3, I include "beautified" bar plot for comparison. One point of emphasis here is that the default plot may be useful, but it will not always be your favorite. In a case like that, it is necessary to go a bit deeper into the details.

Why does this work? And why did they bother to make it work that way? These seemingly simple questions lead to some deep thoughts which eminate from–I hate to admit it–computer science.

Figure 8.2: Boxplot: Continuous output against a categorical input

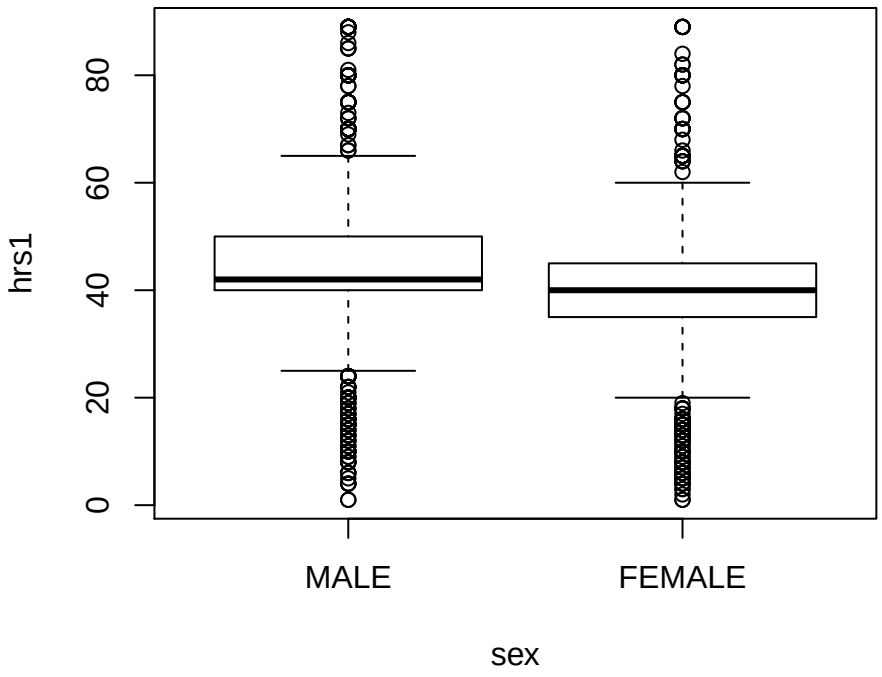


Figure 8.3: Spineplot: Categorical output against categorical input

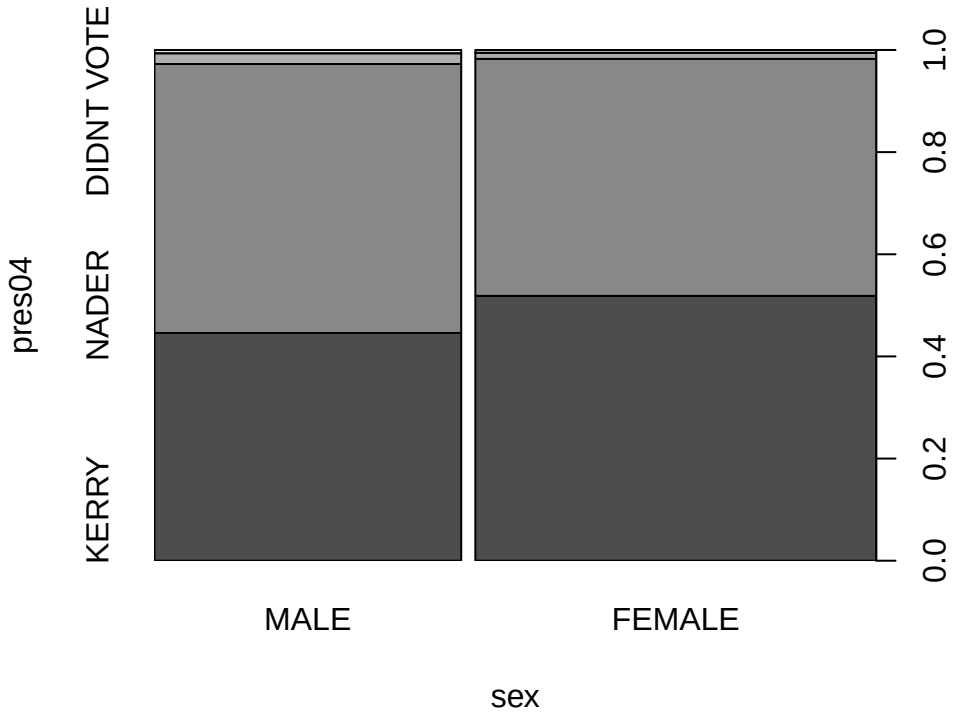
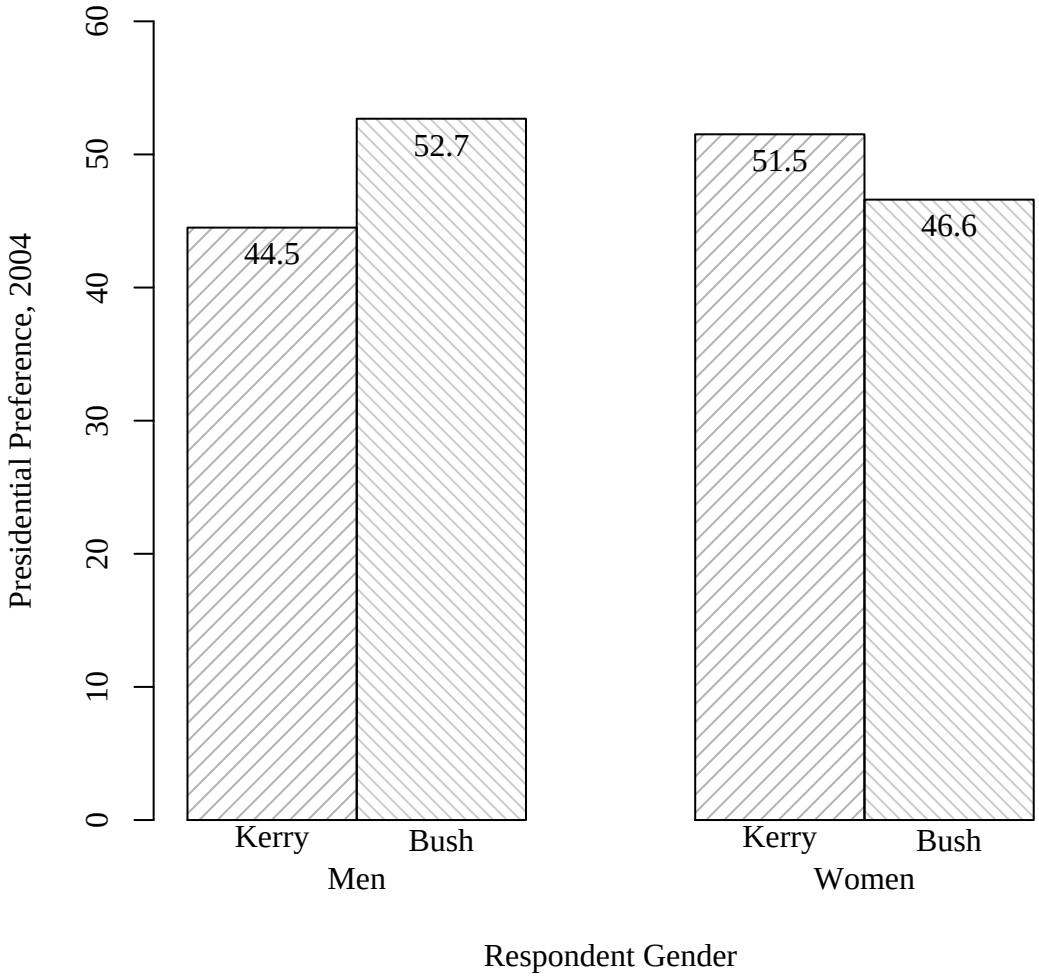


Figure 8.4: Customized bar plot



8.2 Some Computer Science Background

8.2.1 R is an interpreter, just like those folks at the United Nations

An exploration of the plot command brings to the forefront the fact that R is an interpreted computer interface. In a way that seems almost magical, the plot command translates user requests into a format that other R functions can understand. Those "other functions" are called "methods", for reasons which will be discussed below.

In R, a "generic function" is a command that translates user input into a formally structured collection and sends it somewhere else. It is very easy to miss the importance of the term "generic function" while reading the plot help page. While reading ?plot, most of us speed to the end to find out "how do I make a plot." The fact that plot is generic will not be important to most users until they think they are asking for one kind of plot and a completely different kind "pops up" on the screen.

The inner workings of plot() determine which plot routine will be put to use. We have some control over that selection, both in the type of variables we select for the plot and by specification of various plot options.

Here's a tip that I did not learn until I had used R for almost 10 years. If one wants a scatterplot, there is no need to bother with the generic `plot` command at all. One can instead use the command plot.default(), because that's where plot sends the instructions after it re-organizes them into the proper sequence. One should be cautious, however, because `plot.default()` will not allow user input in the formula format. Instead, one must submit a command of the form:

```
plot.default(x=age, y=hrs1)
```

Similarly, if one wants a bar chart, the `barplot` function can be be used directly. However, `barplot` does not accept a formula like $y \sim x$ and, even more inconveniently, `barplot` does not want x or y as "raw numbers." `Instead, barplot` requires the user to make a table object to summarize the data, a table which can then be sent to `barplot`. The generic function plot reformats the data as necessary to make the barplot for us.

In R, one will also find 2-dimensional plotting routines like boxplot(), hist(), and dotchart(). Each of these fulfills a purpose, and one can use them directly, or it may be that they will be receivers for a re-formatted set of instructions from the plot command.

When do we get something useful from commands like this?

```
whatever ← someRFunction(whichever)
plot(whatever)
```

Obviously, the result depends on what is inside "whatever." In this usage, we are hoping that someRFunction creates an output that can be plotted. That means the author of someRFunction() has to write a plot routine that displays a preferred set of illustrations.

Which R commands create items that plot will be able to manage? Upon re-reading ?plot, one finds that the reader is invited to run this command

```
methods(plot)
```

to view a selection of implementations that are available. In Table 8.1, the output of that command is presented. The items in that list are pairs joined by periods. The names of these commands are structured in a substantively meaningful format. Each combines a command, in this case "plot," with the name of the class of the object that is being plotted.

For example, consider the method "plot.lm". This means that the R function "lm" creates an object for which lm has also created a plot routine. If we want to, we can read about the details by running

```
?plot.lm
```

The lm() command creates a regression model, and plot can take that model, notice its type, and then pass the work on to the plot.lm() function. Since we know in this particlar case that plot() will simply re-route the regression object to plot.lm(), we can by-pass plot altogether if we want to. We could run

```
whatever <- lm(pres04 ~ sex, data = gss2006)
plot.lm( whatever )
```

If one were running this function thousands of times in a simulation, the program might run more quickly if we call plot.lm directly, thus avoid the additional work involved when the plot function has to scan the input and re-route the work to plot.lm.

As a hint to the budding R programmers who might read this, you can see what is "really getting done" by the function by typing this command (with no parentheses after it):

```
plot.lm
```

The code that does the actual work of plotting a regression model should display in the R terminal.

The list in Table 8.1, includes all of the commands that have created their own routines called "plot" that can be found by the generic function plot. If one's computer has many additional packages installed, one may see many more options listed.

Many of the functions in Table 8.1 are asterixed, meaning that the authors of the function do not intend that we try to use their plot method by calling its name. They want us to use the generic plot as a gateway. That is to say, because we see plot.isoreg*, we know that the authors of isoreg do not want us to try to take this approach.

Table 8.1: The output from methods(plot)

```
 [1] plot.acf*            plot.data.frame*     plot.decomposed.ts*
 [4] plot.default         plot.dendrogram*     plot.density
 [7] plot.ecdf            plot.factor*         plot.formula*
[10] plot.hclust*         plot.histogram*      plot.HoltWinters*
[13] plot.isoreg*         plot.lm              plot.medpolish*
[16] plot.mlm             plot.ppr*            plot.prcomp*
[19] plot.princomp*       plot.profile.nls*    plot.spec
[22] plot.spec.coherency  plot.spec.phase      plot.stepfun
[25] plot.stl*            plot.table*          plot.ts
[28] plot.tskernel*       plot.TukeyHSD

   Non-visible functions are asterisked
```

```
> whatever ← isoreg ( something interesting )
> plot.isoreg( whatever )
```

They want us to do the following instead.

```
> whatever ← isoreg ( something interesting )
> plot ( whatever )
```

We–the users–don't always have to do what they say, of course. If one wants to see the code that is executed by isoreg's plot routine directly, it can be done by asking for it by name, including the package from which it is delivered. This displays the code that actually does the work:

```
stats:::plot.isoreg
```

And, once we understand all of the required options, we can try to supply them ourselves, so instead of using the plot command to re-organize the input for us, we could just run:

```
whatever ← isoreg ( something interesting )
stats:::plot.isoreg ( be careful here, options must match needs of plot.isoreg
    )
```

Before you ask, let me explain about :::. If a method in a package is not askerixed, it is not hidden from the user interface, and it can be found with just two colons, as in stats::isoreg. This finds the code for the lm routine in the stats package. Since stats is automatically loaded whenever R starts, it is not necessary to type the prefix "stats::". (Nevertheless, it seems to me that more and more R authors are being explicit by including that "namespace" information in their code.) The authors of packages usually hide functions for a reason, so

R won't find those functions if only :: is used. Rather, ::: is needed. Usage of three colons is a user's "oath of personal responsibility." The user is saying, "I know what I'm doing, I won't yell at any programmers when this fails because I have not done this in the way the programmers recommend." Most of the time, ::: is not needed, but there have been some urgent cases in which I have had to use it. It usually arises when we want to extract information from an object and the author of the package did not anticipate we would want to extract just that one bit. If the element that is holding the information is hidden, we have no option but to force it into the light.

8.2.2 Object-Oriented Terminology

This section offers my summary of object-oriented programming, or OOP, for short. The terminology is important not only for R, but, well, for just about any modern computer program. Since my purpose in this book is to offer an enduring foundation for people who are starting on a career in research, I can hardly resist the temptation to introduce the hugely important ideas like "method" and "implementation." These will be unfamiliar to students who have not taken a course or two in computer programming; the main goal here is to demystify the language. It is not a programmer's guide; it is a guide to help people understand programmers. I have learned a lot about this by trying to write programs and hanging around with expert programmers, but I am not formally trained as a programmer. As a result, I may be a good translator between "us" and "them."

I first came across the terms "object", "method" and "implementation" when I decided I wanted to write agent-based computer simulation models with the Swarm simulation libraries. Because I was working on a model in which there were many autonomous people, I wanted to represent my agents as separate, self-contained entities. To my surprise and delight, I found out that computer scientists had developed a terminology of "objects" and "object oriented programming" that meshed perfectly with my substantive objective. When I started programming, "objects" were actually quite new in computer science. Since I did not do much programming in the "old school," the beauty and newness of objects was lost on me, at least at first, because I did not realize how bad the old way wa.

Swarm simulation models are written in Objective-C, one of the earliest object-oriented languages. The object-oriented programming languages that are most frequently encountered in the day-to-day practice of research are C++ and Java. They exist within (dominate?) a lively ecosystem of competing object-oriented languages. C++ and Objective-C can be thought of as "C with objects," and Java implements those same ideas, but it is supposedly designed "from the ground up" in an object-oriented way. Java was originally proposed as language that is more comprehensive in its adoption of object orientation; many of the components that C++ programmers have to write for themselves or find in addon libraries are offered by the Java language itself. C++ is touted as the high-performance ("fast") language, while Java is offered as a less-error prone (and "slower") language that is more easily translated from one kind of computer to another. By the early 1990s, the OO craze had swept across almost all parts of the computer language ecosphere. Object terminology

was "bolted on" to virtually every surviving computer language, including Perl, Python, Basic, Pascal, and so forth.

What's so different about object-oriented programming? The main idea is that we try to keep conceptually clear boundaries between different parts of projects and we communicate between parts of projects in a clear, formalized way. Instead of re-writing functionality in many different routines, we try to organize our ideas so that code can be "re-used." If we can solve a problem that arises in many different contexts, we should have one solution and find a way to put it to use in all those contexts.

The difference between the old school and the new school is difficult to convey unless one has actually tried to write a program in the old style. Perhaps I can illustrate the difficulty with a silly little example. Suppose we have research assistants collect data on food preferences. They ask people if they like each of 7 foods. A score of 1 indicates that a person likes a type of food, while a 0 indicates the opposite. Sort the data for each person into a vector.

$$Jane: \quad (0,1,1,0,1,1,0) \tag{8.1}$$

$$Bill: \quad (1,1,0,1,1,0,1) \tag{8.2}$$

Smart computer programmers in the days of yore (the same guys[1] who caused the "Y2K problem"), see an opportunity to "optimize this". In the "olden days," before object-orientation, they would want to stack together the observations about many people into a two dimensional array, allowing them to access information by row and column number. For example,

$$Preferences: \quad \begin{matrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{matrix} \tag{8.3}$$

One would access information by referring to row (person) and column (food), such as $Preferences[1,2]$. In case you are wondering, the third row represents the tastes of my daughters, who only like macaroni and cheese, while the fourth row represents the tastes of our dog, who likes everything.

Next, a clever programmer will treat this as one big row of numbers. It is faster for a computer to go to the 24th item in a single row than it is move "down" to the 4th row and "over" to the 3rd column. They might rephrase the whole thing like so:

$$Preferences = \begin{matrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \dots \\ & & & & & & & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{matrix} \tag{8.4}$$

[1] These are the well intentioned people who caused all that worry at the change of the millenium because they had converted symbols like 1967 to 67 in order to save memory.

When we want the data on the second row's third column, we have to do a little math. If we start counting at 0 for the first element, then we'd use $Preferences[9]$ to get the right number.

We could make this one step more "efficient" by converting these into "bits." A "bit" is the smallest amount of computer storage, either 0 and 1. Memory is made up of bits. We can take that string of 0's and 1's, and record it in a single integer.

That probably comes as a shock, so I should explain. To a computer, an integer 0 is not really 0. Instead, it is

$$00000000000000000000000000000000 \tag{8.5}$$

if we are on a 32 bit computer system. Any integer in base 10 is be represented by 0's and 1's. In a computer memory system, an integer like 0 or 1 takes up the same amount of storage as an integer like 344,432. These will be 32 bits or 64 bits of storage. In a 32 bit system, here is how the computer sees integers:

Integer in base 10	32 *bit storage*
0	00000000000000000000000000000000
1	00000000000000000000000000000001
2	00000000000000000000000000000010
3	00000000000000000000000000000011
4	00000000000000000000000000000100
8	00000000000000000000000000001000
89	00000000000000000000000001011001

(8.6)

Obviously, we are wasting a lot of space if we treat 0 as a 32 bit integer, because we really only need one bit to record that information. However, computer memory is accessed in chunks, usually 8 bits at a time, and usually we think of an integer as either 4 or 8 of those chunks (hence, the terms "32 bit" or "64 bit" operating system). A vector of preferences $(1, 0, 1, 1, 0, 0, 1)$ as a collection of 7 separate integer numbers uses up 7×32 bits of storage. We can compress the string of 0's and 1's as a single integer. The preference values $(1, 0, 1, 1, 0, 0, 1)$ are represented by just one integer, 89. The number 89 is, of course, not meaningful within the context of the research problem, but programmers have ways to convert back to bits when they need the 0's and 1's. I suppose I'm sensitive about this because the process of storing information into "bit vectors" and getting it back out again is the most challenging element in code of the Santa Fe Artificial Stock Market (???).

The danger of ordinary "old fashioned" programming is starting to become apparent. We grouped together the data on all the people and all the foods. Accessing the elements of a list of numbers in a single line is easy for the computer, and changing a 1 to a 0 is a fast operation. What if we make a mistake? While trying to change one person's data, we might accidentally hit another one. All of the information about everyone is being passed about, exposing it to corruption each time. One of the common sources of confusion in C programming, for example, is that the rows and columns are numbered beginning at 0.

When you want the second row, you ask for row 1. (I've made the "off by one" mistake more times than I'd like to admit.)

This is one of the mistakes that object-oriented programming is intended to fix. We should keep the data about separate people in separate containers. Instead of passing all of the data about everyone about and letting it be changed by functions, we instead want to safeguard the privacy of that data. We leave the data hidden, isolated, so that an effort to change the score for one case cannot damage another. In fact, after a while, we reach a point where we don't think of changing the data at all. Rather, we think of asking the "object" that is holding our data to change it for us.

The following are core principles that are held by all of the object-oriented languages (so far as I know).

1. Data should be encapsulated into separate "objects."

2. The programmer should interact with objects through a structured "interface."

3. Objects should be organized conceptually into a "hierarchy of classes," so that widely held characteristics and behaviors are represented in a "top level class" and objects with more specific characteristics are drawn from a "lower level class."

What is an object? An object is defined by two characteristics:

1. information (stored in variables)

2. capabilities (carried out by methods)

The information might be stored as integers, real numbers, true/false indicators, text strings, or as complicated combinations of these types. The main idea is that the object has access to information about itself, but it does not automatically have access to information about other objects.

The methods are capabilities–the things the object can do. In non-object-oriented programs, the term "function" is used to refer to a capability that can accept input and return output. In one of those older languages, such as C, a function would have a name like *calculate* and it would receive the entire data string about all of the cases. In the object-oriented approach, each different type of object can have it's own ability to *calculate.* That is referred to as the "implementation". We might have 10 different types of objects floating about, each of which has the ability to answer a request like *calculate* and these objects may be able to sort through a variety of types of input in order to discern what action is called for. The term function refers to a disembodied capability, something that is done to values. There's no recognition of privacy in functions. The term method is used to highlight the fact that each object's special capability can be keyed to the nature of the object itself. Honestly, I'm not sure method is the best possible word, but they needed something different from function. And it is too late to change it now.

All OO programming languages of which I am aware use the idea of a "class" to describe the features that are common across a collection of objects. Suppose we want to create a program in which there are lots of separate people represented as objects. We could create a "top level" class called Person (by custom, the class name is capitalized), and inside that class, there could be variables like height, weight, hair color, favorite food, and so forth (usually variable names are not capitalized). Each object created from the class will have memory set aside to hold values of those variables. There could be methods to respond to instructions like "go to the store," "eat," and "tell me your favorite food".

Subclasses are used to create specialized types of objects. Subclasses might be Student, Professor, Chef, GangMember, CorporateExecutive and so forth. Person holds common attributes and functions of people. Subclasses represent the special attributes. Each subclass inherits the variables and methods of the classes above it. A subclass might introduce new methods or it might replace (technically "override") a method. The subclass Student has variables for the "name of college," "major," "foreign language," and "living arrangement". The GangMember subclass may need to have separate data on "affiliation" and "weapons" while PeaceCorpsWorker may need special data on "language training" and "preferred region of service." All of these objects need to go to the store or eat, but only students need a method to "do the homework" and only gang members need a method to "initiate new members."

When a section of memory is set aside to create an object from a class, we are said to have created an "instance" from that class. It is quite common to create (formally, say that we "instantiate") many objects and add them into a container. A "container" is another object, of course. One of the headaches for newcomers to programming is that there are many different kinds of containers, and some are much better for some purposes than others. Some are "fast" to insert new items, some are fast to find specific items. We often also need "iterator" objects. An iterator can "grab" the first element out of a collection, and then the next, and so-forth. This is one of the ways in which we process a collection. If we want to know whether or not the people in our survey like broccoli, then we use an interator to go to each person's data object and ask if it likes that horrible stuff (I mean tasty, healthy green vegetable).

This seems strange to newcomers, but we are encouraged to respect the privacy of objects. This principle of "information hiding," or "object encapsulation," is widely accepted. It represents the inclusion of "real life" into the programming experience, once you think about it. Suppose you go to a cocktail party. Do you instantly know everything about everybody there? How do you find out about them. Do you ask people for their names? I do! Do you pull out your tape measure to assess their height? (If so, do people like it?) Or would you instead ask, "how tall are you?" A computer program should not grant itself global access to the information about the heights of the people objects. There should be a method *getHeight* that we can address to an object. When the program creates a Person object, there should be a method with a name like *setHeight* that asks the object to make note of its height. Because the information is set and stored away, it sometimes seems like a hassle. If we want a sub-collection of all females that are shorter than 5 feet, we have to cycle through the objects in a collection and ask each one, "are you a female shorter than 5 feet"?

We reduce the danger of accidentally changing a piece of data if we interact with the object through a formalized protocol. The formalized protocol is called an "interface." An interface is a list of instructions that an object can understand and respond to.

Programmers in R sometimes use the term "method function" to refer to methods. This seems somewhat oxymornic to me, but it differentiates the implementation, the code that actually does work, from the "generic function." As far as I know, the usage of the term "generic function" is unique to R, but I have heard the term "method function" used with quite different meanings in other contexts.

I think my point here is that R comes with a lot of object-oriented terminology, but it is not quite like any other object-oriented language. Even the syntax with which we interact with objects betrays the distinctiveness of R. Suppose we have a statistical model saved in an object we have called *reg1*. We want to tell that object to plot itself on the screen. In Objective-C, the command would be:

```
[reg1 plot];
```

While in C++ or Java, it would look like this

```
reg1.plot();
```

In both cases, the object is sent a message. We tell the object to do something.

In R, however, one does not address objects with messages in that way. In R, one would tell a function to manage an object.

```
plot(reg1)
```

Notice the difference. In the other languages, one takes the object *reg1* and addresses an instruction to it. On the other hand, R (in this case, at least) has us tell the plot function to do something with the *reg1* object. One has a function *plot*, which has to discern the nature of the object *reg1* and then figure out what to do with it.

The S language has been a work in progress, absorbing new ideas about statistics and software for more than 30 years. It is only natural that S, which existed before object-oriented programming, would adapt in the face of changes in computer science. The first version of S took shape in the mid 1970s at Bell Labs, well before work on C++ was initiated, also at Bell Labs in 1980. By the time C++ was published for commercial use (1985), S had already had two major released versions. The third generation of the S language, known as S3, was documented in *The New S Language* (1988). S3 incorporated some object-oriented ideas and terminology, but not in a tightly structured way. The fourth edition, S4, includes a stricter set of guidelines on the use of objects in S. Most of the core functionality of R is

written with S3 classes, and many doubt that will ever change. Nevertheless, developers of new packages are encouraged to use S4 classes.

There is much more to say about object-oriented programming languages, but I believe this has been a sufficient foundation. The reader needs just enough background to not be bothered by terms like "object", "method", "class", and "implementation." The R-specific term "generic function" is vitally important, of course, because it helps us to see the difference between a general functionality and the specific implementation.

I would summarize this section as follows. The substance of our research problem should break down into smaller pieces, individual bits of information and capability on which we can separately focus. The capability of each object is represented by its interface, a list of methods that it is able to carry out. We try to "hide the details" of the implementation behind the interface. We should try to think of the abstract features that are common across many types of entities and organize those common features as a class. Specialized features are introduced by the addition of subclasses.

8.3 R 2-D (Not R2D2 or C3PO)

plot() is the ring master that harmonizes the work of a whole family of plotting routines. Over the years, I've made boxplots, barplots, histograms, and so forth, but I did not appreciate the idea that these things are "objects" in the sense that they all have a basic set of variables that determine their size and display. A substantial part of plot's work is in the translation between the general interface and the specific details that are needed to make a particular kind of plot. This is the point at which the object-orientation of the family of 2-dimensional plots is meshed with the idea that R is an interactive ("interpreted") computing experience.

8.3.1 More about syntax and R's interpreter

This command:

```
plot (x = dat$sex, y = dat$hrs1)
```

has the same effect as this command (which is known as the "formula interface"):

```
plot (formula = hrs1 ~ sex, data = dat)
```

This will also have the same effect:

```
plot(dat$sex, dat$hrs1)
```

As will this:

```
plot(hrs1  sex, data=dat)
```

As long all arguments are explicitly named, they can be entered in any order:

```
plot(y = dat$hrs1, x = dat$sex)
```

One can specify a type of plot as well, but in this case:

```
plot(y = dat$hrs1, x = dat$sex, type="p")
```

the setting of type="p" has no effect because p (for "points") is the default type of plot.

And, finally, note that plot does not have a data option, except when using the formula interface. Hence, if we do not use the formula interface, it is necessary either to specify the data frame with each variable, as in dat$hrs1, or use the with() function to wrap the things together:

```
with(dat, plot(y=hrs1, x=sex))
```

When people say that "R is an interpreted language," they are referring to the fact that the R program sorts through the input it receives and it tries to "make sense" out of it. Somewhere, deep down, far "undeR the hood," there are specific, structured functions that expect options in a particular order and format. But, between that "deep down layer" and the user sits the interpreter, transforming the user's request into a format that the system can understand. If options are not named, but they match the types of arguments that would be expected in a particular order, then R will make use of them. If the options given don't make sense to the interpreter, then an error message will be offered.

By far, the most common source of confusion among my students is that plot allows both the formula $y \sim x$ interface and the x, y interface. A lot of scatterplots turn out backwards. People become familiar with the format of the formula interface (y first)

```
plot(y~x, data=dat)
```

and they then accidentally give their inputs in the wrong order when they use the other interface (which wants x first). It would be correct to write

```
plot (dat$x, dat$y)
```

but students are often inclined to reverse the x and y to match the ordering in the formula interface. In some ways, I have found R's tolerance for a variety of syntax to be annoying and distracting. I often wish there were just one correct way to do something and I could teach just one way. But the R developers feel differently, and they probably know best (and they are doing all of the heavy lifting).

In case the reader wonders why this interpreter issue fits together with object-oriented programming in my mind, here is the answer. One concept in some object oriented languages called "polymorphism." In Java and C++, for example, a class can have a number of methods with the same name, and the choice among them while the program runs depends on the type of input. From this perspective, plot is not really one function, rather it is a family of functions, and which one is actually called depends on the type of input.

There are several hints about the nature of the 2-D plot family in the help page, ?plot. One of this most intriguing hints is the ... Argument:

```
...: Arguments to be passed to methods, such as graphical parameters
    (see 'par'). Many methods will accept the following arguments:
```

That's a bit vague. "Many methods?" not all? What is that par thing it mentions? Don't worry about the details now, we will return to that later.

8.4 A Power Tour of the plot framework

The plot framework is a tightly interwoven set of tools in a package called "graphics". That package is part of the R base. That means it is maintained by the R Core Team; I expect it will always be in R, as long as R exists. That differentiates it from volunteer-donated packages on the CRAN network, which may be abandoned at any time.

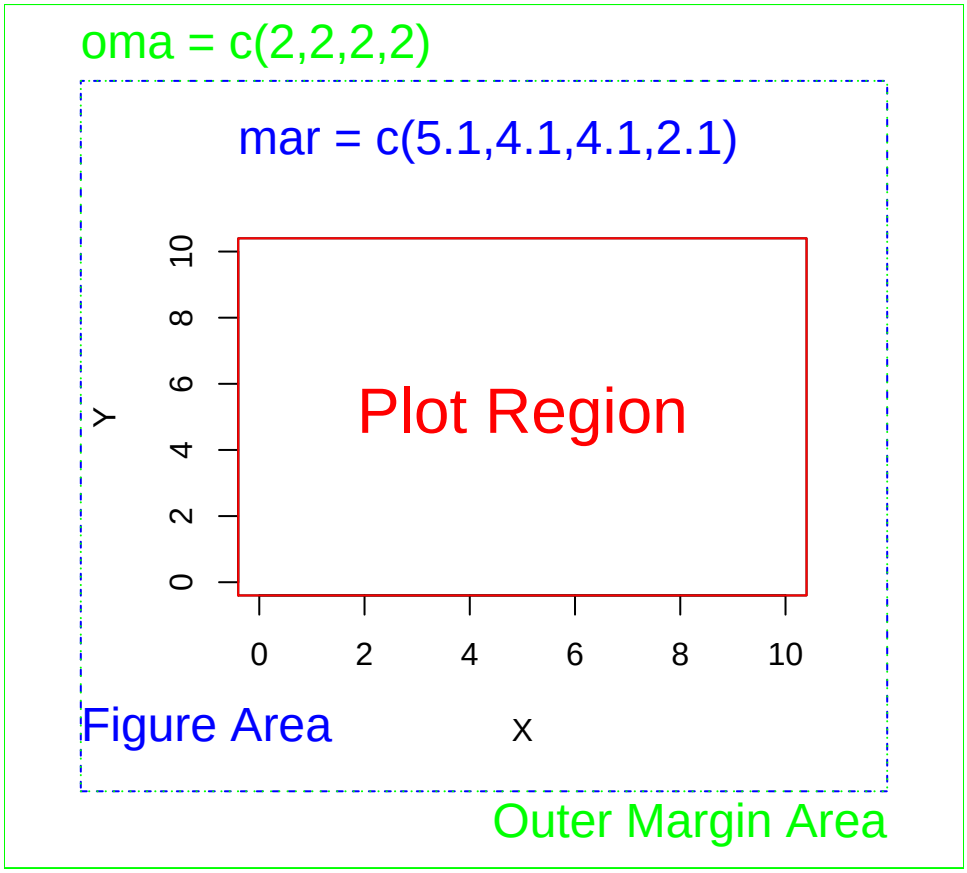
This framework can produce simple figures with too much trouble. It can produce extremely complicated figures, if one knows the ins-and-outs of it. There is so much that can be done, it is a bit difficult to know where to start. Perhaps a concrete example is in order. The next section illustrates the way in which the pieces work together.

8.4.1 Plot and the art of figure maintenance

This command draws a plot of nothing. Run it for yourself and see!

```
plot(x = 0:10, y = 0:10, type = "n", bty = "n", axes = F,
    xlab = "", ylab = "")
```

Figure 8.5: Plot region and margins



The option type="n" means nothing is plotted. We suppress the printing of the box around the figure area (bty="n"), don't let R draw axes (axes=F), and set the labels on the x axis (xlab="") and y axis (ylab="") as strings with no characters. The axes are still there, but they are "invisible".

The scale of the invisible axes structures the display. The invisible structure is brought into the light in Figure 8.5. The entire area of the figure can be divided into three parts. The "plot region" in the center is the part inside the axes; that is where we will place the substance of our illustration. The plot region is something of a "leftover," it is what we have after the margins have been imposed. There are two kinds of margins. The outer margin (oma=c(2,2,2,2)) is set to be 2 "lines" wide on all four sides, while the inner margin is set with a bit more space on the bottom and a bit less space on the right (mar=c(5.1, 4.1, 4.1, 2.1), where the dimensions refer to the bottom, left, top, and right sides). The axes are placed exactly on the edge that divides the margin area from the plot area, and the labels for the axes are written in the inner margin area. In this case, x and y are both integer sequences from 0 to 10, so the plot region in which we can write corresponds to coordinates in $(0, 10) \times (0, 10)$. The "figure area" equals the "plot region" plus the "inner margin" area.

There are several "low level" commands that can be used to doodle on that blank piece of paper. The ones that I use most commonly to write inside the plot region are:

- text: for writing inside the plot area
- points: puts plot characters in plot area
- abline: draws straight lines
- lines: draws a smooth curve through indicated points
- legend: for annotation of lines, points, or bars
- segments: draws straight lines between indicated points
- polygon: for a shaded region in the plot area

To write in the inner margin, the only commands that I often use are:

- mtext: for writing margin text
- axis: customize axes

I believe there is no substitute for experience. So before you continue reading, open R and execute the following commands one at a time:

```
plot(x = 0:10, y = 0:10, type = "n", bty = "n", axes = F,
    xlab = "", ylab = "")
axis(1)
axis(2)
box(bty = "L")
points(c(6, 1, 4, 8), c(7, 9, 3, 2), pch = 6, cex = 0.7)
arrows(x0 = 3, y0 = 3, x1 = 7, y1 = 8)
points(3, 3, pch = 20)
arrows(x0 = 7, y0 = 9.4, x1 = 9, y1 = 9.4, angle = 90,
    code = 3)
text(c(1, 1.3, 2, 2.3, 2.6), c(7, 7, 7, 7, 7), labels = c("O",
    "K", "", "G", "O"))
abline(h = 5, lty = 2)
abline(v = 5, lty = 2)
mtext("This is margin text written by mtext", side = 1,
    line = 2)
```

The result is illustrated in Figure 8.6.

8.4.2 Things I've done in the plot region (and lived to tell about it)

As mentioned above, a command like

Figure 8.6: Doodling on the Blank Plot

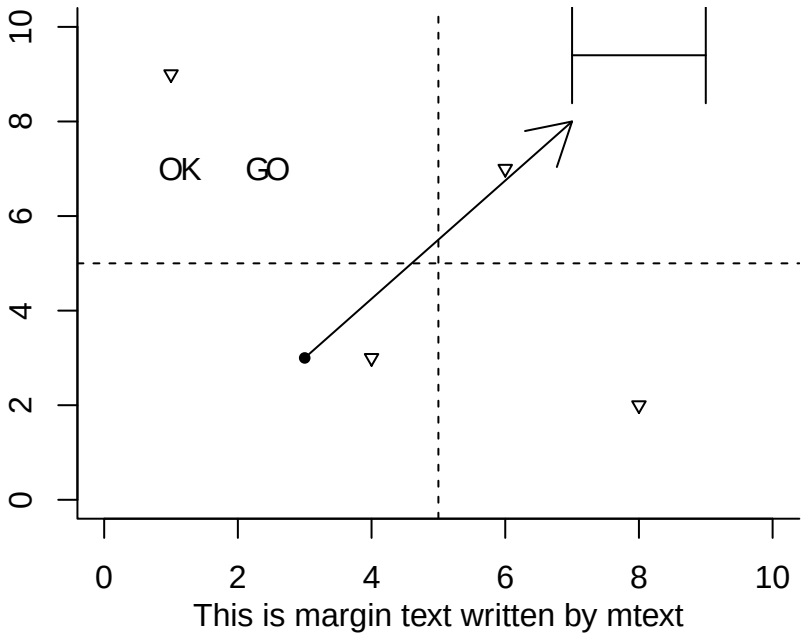
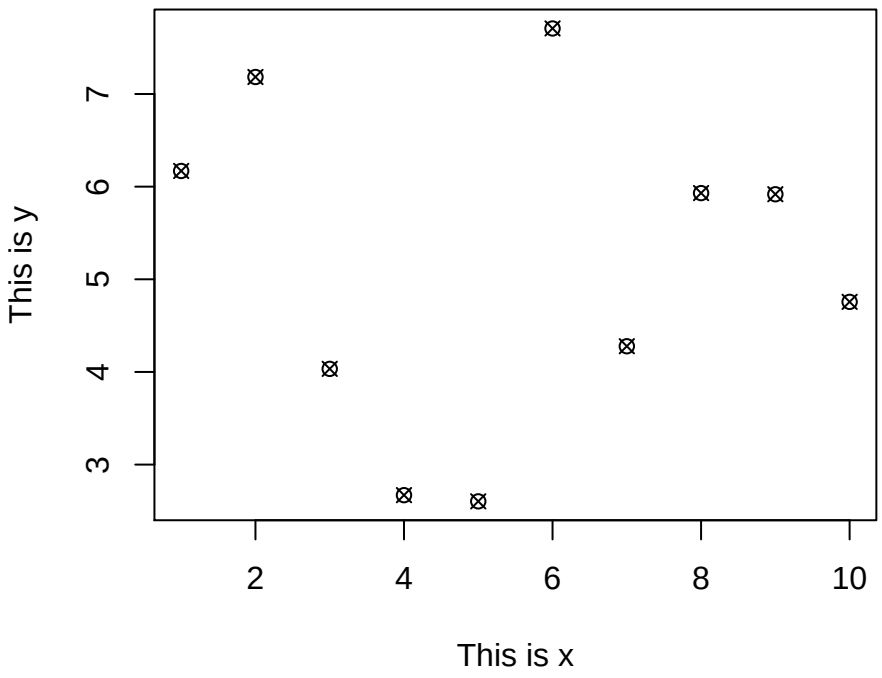


Figure 8.7: Using plot character 13



> plot(x,y)
will create a scatterplot if x and y are numeric variables. By default, the plotting character is a little circle.

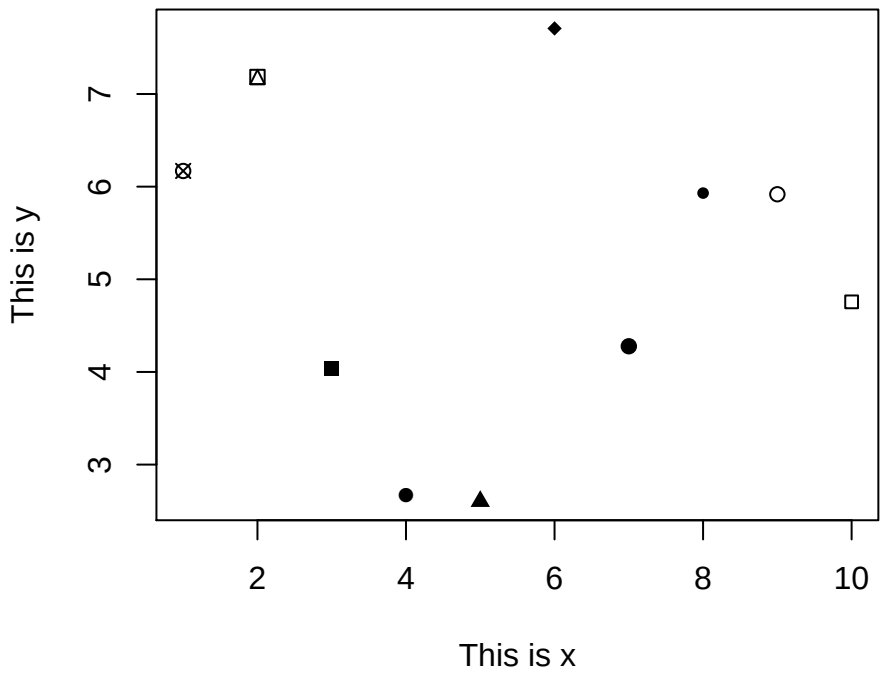
We don't have to put up with that little circle.

The first variation on the theme involves a change of the plotting character. In my experience, the most dependable way to do this is to plot a figure with an empty middle, and then add the points like so:

```
x ← 1:10
y ← rnorm(10, mean = 5, sd = 1.5)
plot(x, y, main = "", xlab = "This is x", ylab = "This is y",
    type = "n")
points(x, y, pch = 13)
```

Considering the explanation in the help page ?points, we know that there are many symbols that might be used. Figure 8.7 illustrates character 13, which I selected completely at random.

Figure 8.8: Plot characters 13-22



It is not necessary to use the same symbol for every point. In fact, if there are 10 "cases" being observed, then we can ask for a different symbol for each one. That is to say, we can use a formula involving x and y to specify the plotting character. For example, run the plot command again, but now select plot characters 13 through 22:

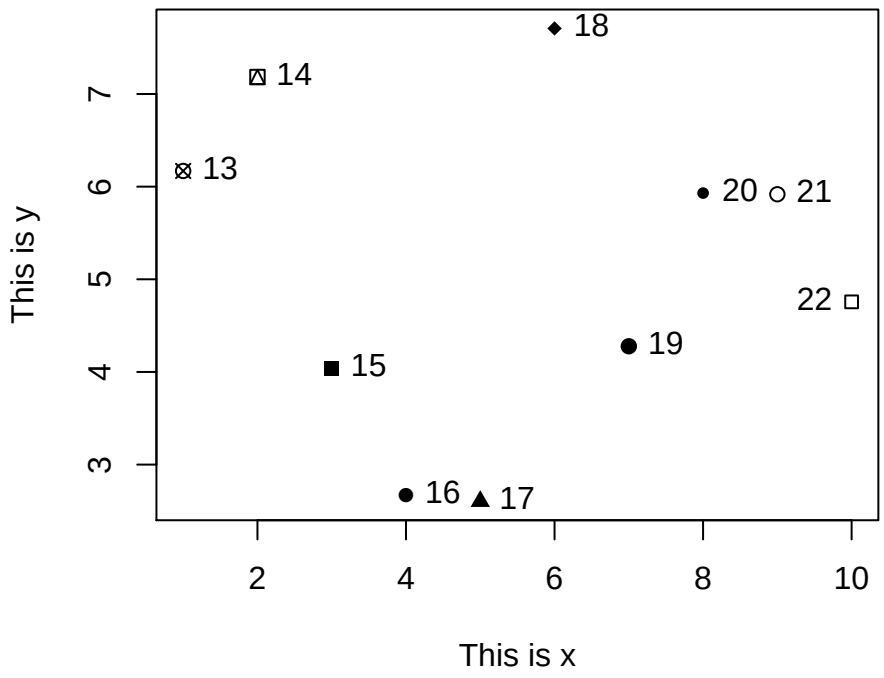
```
plot(x, y, main = "", xlab = "This is x", ylab = "This is y",
    type = "n")
points(x, y, pch = x + 12)
```

We get 10 characters, from 13 through 22, because $x = 1, 2, \ldots, 10$ and the plotting character equals $x + 12$. Examine the result in Figure 8.8.

I suppose the reader is thinking "how can I tell which one is which?" I applaud your initiative to ask! Why not label the points with some text? These commands produce Figure 8.9.

```
plot(x, y, main = "", xlab = "This is x", ylab = "This is y",
    type = "n")
points(x, y, pch = x + 12)
text(x, y, label = as.character(x + 12), pos = c(rep(4,
    9), 2))
```

Figure 8.9: Text labels on characters 13-22



Note the use of the text command places the numerals on the right side of the points, except for the last one because that would run off the edge of the figure region. We need that one numeral on the left of its symbol. This requires a bit of fancy footwork. The option "pos" is used to set the first 9 labels at position "4," the right, while the 10th label is at position 2, the left.

The only option for the points() function for which I have had much use is "cex", which adjusts the size of the character. Adding an option like

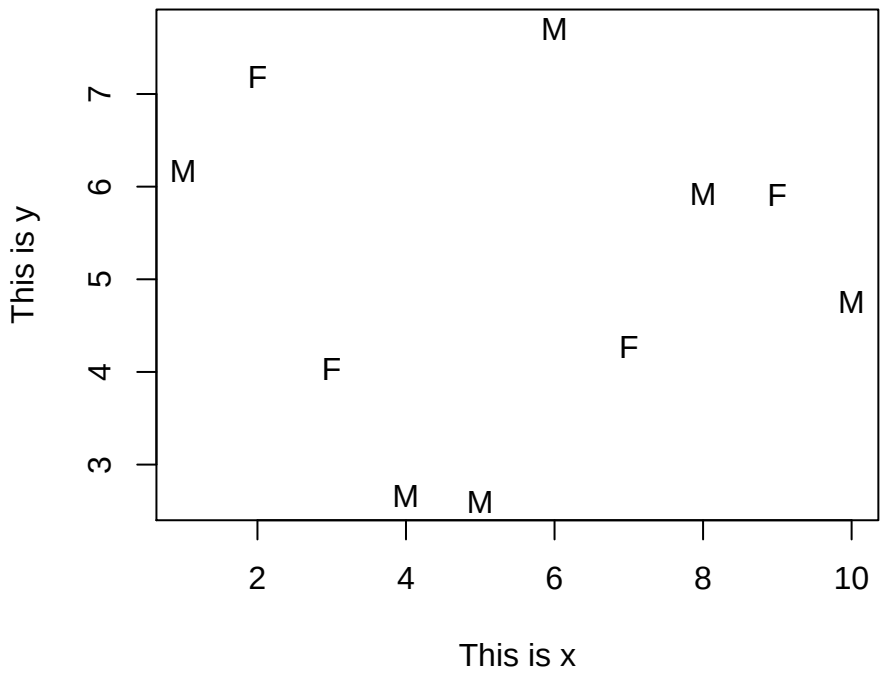
cex=0.5

in the points command will shrink the points to one half of their normal size, while setting the value to 2.0 will double their size. (Try it and see, please.)

The next variation is the use of text to represent a third variable. Let's add a factor variable representing the respondent's sex and then use it as the plotting character.

```
sex ← c("M", "F", "F", "M", "M", "M", "F", "M",
    "F", "M")
sex ← factor(sex)
plot(x, y, main = "", xlab = "This is x", ylab = "This is y",
```

Figure 8.10: M is for male, F is for female




```
     type = "n")
  text(x, y, label = sex)
```

I feel fortunate that the text command is able to use the variable sex as it is. In the first version of this code, I had included "as.character(sex)" as the option, but this way is cleaner. Please inspect the beautiful M's and F's in Figure 8.10.

It might be nice to include a legend that will help the reader remember what the letters stand for. The legend() command is used for that purpose. The legend() command has a format like this:

```
legend (x, y, legend=c("First thing", "Second thing", "Third thing"))
```

Don't be confused that the word "legend" is used twice (I was confused by that many times). The first is the name of the function, the second is an option inside that specifies the names of the things that are referred to. After the legend option, one can add many details to spice up the legend, such as small colored rectangles to refer to certain bars in a bar plot or plotting characters. We have the option of specifying the position with x,y coordinates, or

we can replace x and y with a symbol like “top”, “bottom”, or “bottomright”, which I have used in this case.

```
legend("bottomright", legend = c("Male", "Female"),
    pch = c("M", "F"))
```

By default, it produces a box drawn around each item specified by the legend= option, and there are quite a few ways to customize what else is included. Here I add only the plotting characters as labels.

Sometimes I want to add horizontal or vertical “reference lines” or a grid to help readers inspect the scatterplot. For that, I have had good luck with the command abline(). The abline() command draws one line, but it is “vectorized” so it can be told to draw several lines with a single command. That can be used to “section off” a plot. A command like

```
abline(v = 4)
```

will cause a solid vertical line to be drawn where the x coordinate is 4 (guess what h=4 would do). But I don’t want a solid black line, I want a nice light gray dotted line. So change the command to:

```
abline(v=4, lty=3, lwd=1.1, col=gray75)
```

That is nice, but I also want the light lines where x=2, 6, and 8. Lucky for me, the abline command accepts a vector. In this case, I use a sequence that skips from 2 to 8 by 2 units. Figure 8.11 displays the result from adding this command:

```
abline(v = seq(2, 8, by = 2), lty = 3, lwd = 1.1,
    col = "gray75")
```

```
mod <- lm(y ~ x)
abline(mod, lty = 2, lwd = 1.5, col = "green")
```

The abline() function can be used in a variety of ways. If the v or h options are specified, it will draw vertical or horizontal lines. It also allows one to specify the intercept and slope of a line, which it will then place into a figure. It even has a customized subroutine that will accept a regression model. That means that if we fit a straight line predictive model with R’s lm() function, then there is a method “abline.lm” that is capable of taking the lm output object, asking that model for its information about the intercept and slope, and then the suitable line is drawn into the figure. For example, this code:

produces Figure 8.12

Figure 8.11: Add sections and a legend

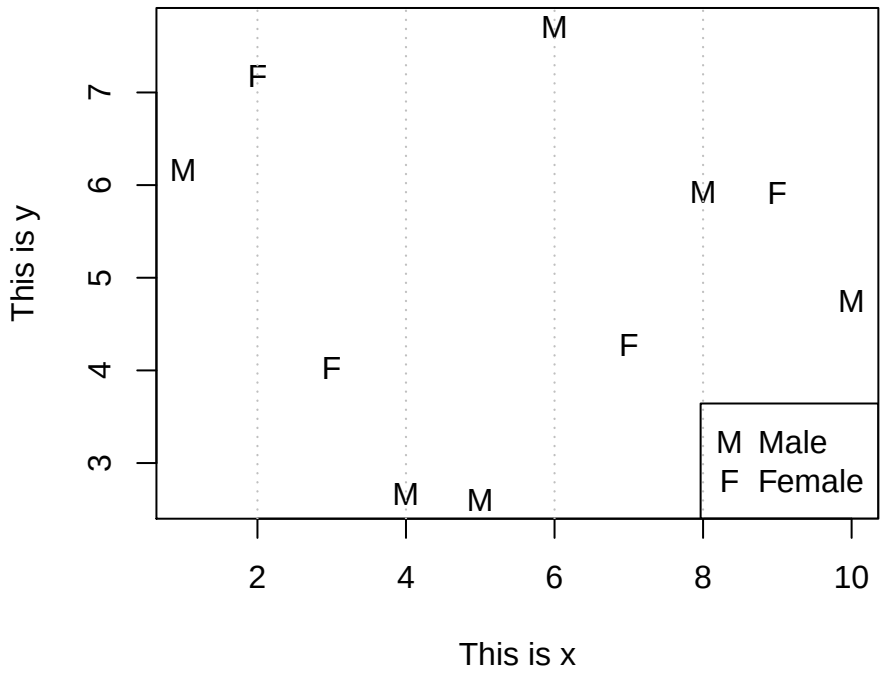
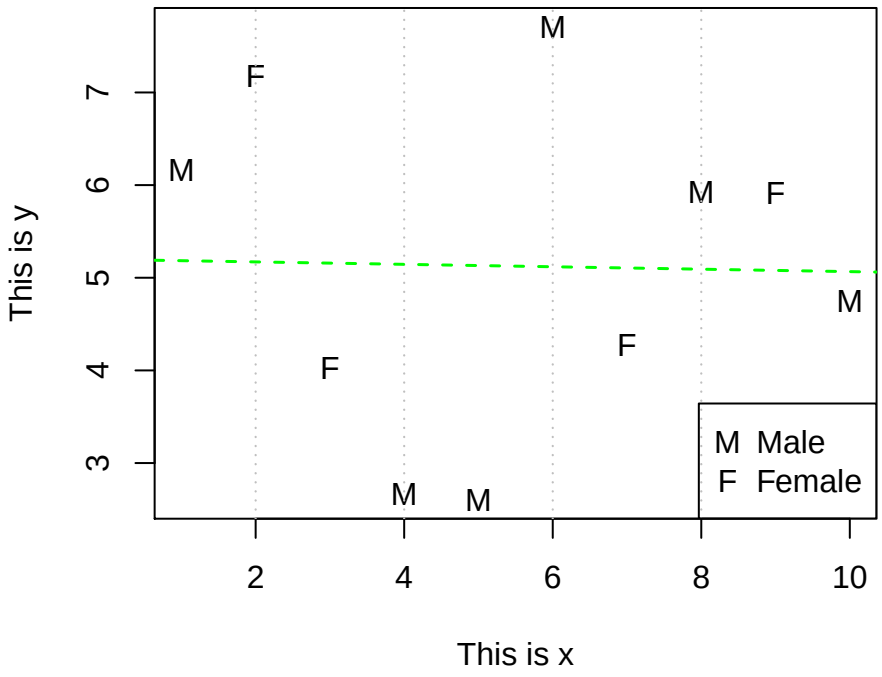


Figure 8.12: Use abline to superimpose a regression line



8.5 Where's my object orientation in there?

8.5.1 If functions have the same syntax and options, they appear to be object-oriented.

Even though the R functions are not written as formal subclasses with an object-oriented language like C++ or Java, the user's interaction with them has many object-oriented qualities. The designers of the 2 dimensional plot routines have clearly aimed to give the "look and feel" of an abstract class of 2D plots that provides standard elements (axes, labels, etc). Specialized drawing routines are usually going to be able to respond to all of those instructions that use generalized features.

It is easy to demonstrate this. Execute the command

```
?plot.default
```

and examine the results. You should see a usage summary in which the full list of options is summarized.

```
plot(x, y = NULL, type = "p", xlim = NULL, ylim = NULL,
log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
ann = par("ann"), axes = TRUE, frame.plot = axes,
panel.first = NULL, panel.last = NULL, asp = NA, ...)
```

Next, consider the barplot and its help page.

```
?barplot
```

```
barplot(height, width = 1, space = NULL, names.arg = NULL,
legend.text = NULL, beside = FALSE, horiz = FALSE, density = NULL,
angle = 45, col = NULL, border = par("fg"), main = NULL,
sub = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
xpd = TRUE, log = "", axes = TRUE, axisnames = TRUE,
cex.axis = par("cex.axis"), cex.names = par("cex.axis"),inside = TRUE, plot =
   TRUE, axis.lty = 0, offset = 0, add = FALSE, args.legend = NULL, ...)
```

Finally, consider the histogram. The help page

```
?hist
```

includes a similar usage guide:

```
hist(x, breaks = "Sturges", freq = NULL, probability = !freq,
include.lowest = TRUE, right = TRUE, density = NULL,
angle = 45, col = NULL, border = NULL, main = paste("Histogram of" , xname),
xlim = range(breaks), ylim = NULL, xlab = xname, ylab,
axes = TRUE, plot = TRUE, labels = FALSE, nclass = NULL, ...)
```

The options that all of these share, such as axes, xlim, ylim, xlab, ylab, main, are evidence that they are subclasses from a larger type of object. Any object that is going to fit into this 2-D plotting framework has to allow the user to specify those details.

I hate to belabor the obvious, but perhaps a few highlights of these common options are worth pointing out.

xlab, ylab These options are used with quoted strings, as in

```
plot(..., xlab=My Giant X Label Here)
```

Once you try this, it will be easy to understand.

One is allowed to define a string ahead of time, and then put it to use:

```
myInteger ← 8
mylabel ← paste("Really Long Variable Name With the Integer ", myInteger
   ," toward the end")
plot(..., xlab = mylabel)
```

Inside the plot command, use no quotes, because mylabel is already a string.

xlim, ylim Specify a 2-valued vector of the form c(minimum, maximum) if you need to "focus" your plot on a particular range of scores. If your observations on the horizontal axis range from -10 to 1000, but you only want to demonstrate a range between 755 and 801, then try the option xlim = c(755,801).

Each type of R plot has its own private algorithm to decide how wide the display range ought to be, given the data that is provided. In my experience, the range for plot.default is about right, but the range in barplot is usually too small. I almost always have to specify the vertical axis range in barplots.

The ranges are especially important if you have several figures that are being compared. In such cases, consider putting the limits to the same values.

There is a function called range that returns the minimum and maximum value of a numeric variable. Inside functions like plot.default, range is used to set the values for xlim and ylim, unless the plot command specifies xlim and ylim explicitly.

I often find myself asking variables for their ranges and then using them to set the limits in the desired way. For example, suppose I only want to display the top one-half of the possible values of x. There are many ways to get this done, all of which seem tedious to me. This approach grabs the range and then adjusts the first coordinate so that it is one half of the way between the minimum and maximum.

```
rx ← range(rnorm(100))
newrx ← c(rx[1] + 0.5*(rx[2]-rx[1]), 1) * rx
newrx
plot(..., xlim=newrx)
```

?????Move below to section on adding text to barplots

There's one other "gotcha." Some plotters, such as the barplot, use their own internal numerical scale for the horizontal axis. Thus, in my experience, it is usually necessary to run a barplot once and catch the created object, so I can then ask barplot what range it used for the axes. Sometimes this seems like a hassle. For example, follow along with these commands:

```
#manufacture x data

x <- as.factor( c(rep("Male",50), rep("Female",50)))

#manufacture y data

#Note plot sends work to barplot

#To find x axis coordinates, we have to work directly with barplot.

#Barplot requires a table, so make one and plot it.

mytab <- table(y,x)

b1 <- barplot(mytab)

b1
```

Each figure type has some unique qualities that are controlled by specialized options. A histogram is different from a scatterplot. The hist function needs to know how wide the bars will be and how to place the break points. It allows a user to either specify breakpoints with a vector or to choose an algorithm with which they are determined. A histogram's bars can be colored or illustrated in various ways. Do we want the histogram to display raw counts on the horizontal axis, or do we want proportions? The barplot also has bars, of course, and so some of its options are the same as the histogram (col, density, angle), but it is also different because a barplot does not have breakpoints.

Each type of figure also shares the mysterious ... option, which is actually a source of much confusion. I had originally thought that "..." was just a lazy writer's way of saying "some other stuff." But it isn't. Actually, "..." is a symbol. It might as well be a word like "other_options". Any option the user specifies that does not match the formally listed options will be sent along as part of "...". In the documentation for most funcitons, the author will say "..." represents options for implementing methods or graphical parameters.

The par() function is often mentioned in that connection. I've been putting that topic off, but can't avoided it any longer.

8.5.2 par(). Do you love it, or hate it?

You can specify everything. But does it have to be such a hassle?

There is a big collection of 2-D graphical settings that is created every time we start a new plot. This collection–an object, really–is floating about in memory and most of the time we don't notice it. Its rather like your heart or lungs: as long as everything works as expected, most of us don't think about them very often.

If a plot device is currently active, meaning you have run plot(), hist(), barplot() and so forth, then this command causes a listing of all the current graphical parameters:

```
par()
```

If no device is currently active, that command causes a new on-screen device to be created and the settings are then listed.

That's a pretty long list of options. If you want to read about them, set aside an hour or so and run

```
?par
```

That is tough reading, with not much benefit. I'd say there is no benefit at all, but once in a blue moon it is very important. If a plot comes out wrong, and you need to fix it, well, par may be your only friend.

To find just one particular setting, ask for it specifically, in a quoted string. This finds the inner margins:

```
par("mar")
```

You can replace a setting with syntax like this:

```
par(mar = c( 1, 1, 1, 1) )
```

to change margins to 1 line on all sides. That change applies only to the current "window", or "plot device." It will disappear as soon as you close that device. This is important to remember. Every time a new graphic device is started, a whole new set of parameters is used, based on the defaults.

The changes to the plot device parameters are forgotten when the device is closed, so there is no danger of permanently damaging anything by fiddling about with them. However, there is a danger that you will find "just the right" combination of settings and then forget it. R examples have been appearing lately that try to help with that by adopting this kind of idiom.

```
old.par ← par(no.readonly = T)
par(mar = c(1,1,1,1) )
plot(x,y)
new.par ← par(no.readonly = T)
par(old.par)
```

This saves the adjustable parameters into an object old.par, for "safe keeping". One it adjusts the settings, and creates whatever plots are needed. Before closing the device, save the adjusted settings as new.par. The object new.par is remembered, so it is safe to tell par to restore the original settings. If necessary at a later point, one can run

par(new.par)

to restore the special recipe. There is absolutely no need to bother with this unless we plan to restore the graphical parameters at a later point. If we ignore this, we can go on our merry way, make whatever changes to par we need, and then close the device in order to re-start with the default parameters.

8.5.3 Need larger margins? par is your friend

Generally speaking, I avoid par. I only think about par when things go wrong. Over the years, quite a few things have gone wrong, so I've become a bit more accustomed to par. But I can't say it seems like a "natural" approach to the problem. But I don't have a better suggestion to offer, either.

As an example, consider this sequence of commands which creates a vector to be used for the heights of bars in a bar plot. This is the same kind of vector produced by the table function, which plot uses to summarize a variable before sending it to barplot. The barplot option "las=2" can be specified inside the barplot so that the names are printed vertically under the bars (I like verbose plots, I suppose). In order to beautify the presentation, the colors for the bars are selected by asking the function grey.colors() for the first 6 shades of grey.

```
x ← c(3, 5, 4, 3, 4, 5)
casename ← c("Barbara Boxer", "Joseph Williams",
    "Roger Hughes Uppington-Smythe", "Billy Krystol",
    "Sandra Dickinson", "Joe P. McGillicutty")
barplot(x, names = casename, las = 2, col = grey.colors(6))
```

The result, Figure 8.13, is almost right, except that the labels are "cut off" on the bottom. The margins are not large enough to allow such long labels. The plot margins can be adjusted to add 10 lines more space with the following approach (which sets the plot margins back to the original settings after the work is finished).

```
old.mar ← par("mar")
par(mar = old.mar + c(10, 0, 0, 0))
barplot(x, names = casename, las = 2, col = grey.colors(6))
par(mar = old.mar)
```

That produces a nice enough barplot with beautifully long lables in Figure 8.14.

A similar sort of "not enough space" problem can happen in the top part of a barplot. The legend for the figure may "get in the way" of the bars. This stanza produces Figure 8.15.

```
old.mar ← par("mar")
par(mar = old.mar + c(10, 0, 0, 0))
barplot(x, names = casename, las = 2, col = grey.colors(6))
legend("top", legend = c("5'3\"", "5'10\"", "6'1\"",
    "5'11\"", "5'11\"", "5'7\""), fill = grey.colors(6))
par(mar = old.mar)
```

That figure is not completely horrible, but I wish the legend did not obscure the bars. In order to fix that, I need to make two changes. First, I add 5 lines of space on the top margin parameter. Second, I set the parameter "xpd" to TRUE. This latter is necessary because the default setting for xpd will not let our legend be written in the margin area.

```
old.mar ← par("mar")
par(mar = old.mar + c(10, 0, 5, 0))
par(xpd = T)
barplot(x, names = casename, las = 2, col = grey.colors(6))
legend(3, 7, legend = c("5'3\"", "5'10\"", "6'1\"",
    "5'11\"", "5'11\"", "5'7\""), fill = grey.colors(6))
par(mar = old.mar)
```

(see ?par concerning "xpd").

8.5.4 Many plots in a single figure: easy layout with par

There are options mfrow and mfcol that will tell the plotter to divide the plot region into sections and then place the following plots into the small areas. The option mfrow has the effect of filling in the multiple plots in row-order. For example, this command

> par(mfrow = c(2,2))

creates a square matrix of plots with 2 rows and 2 columns. The next four plot commands will fill in the figure in "rows first" order, as follows:

Figure 8.13: Barplot labels don't fit

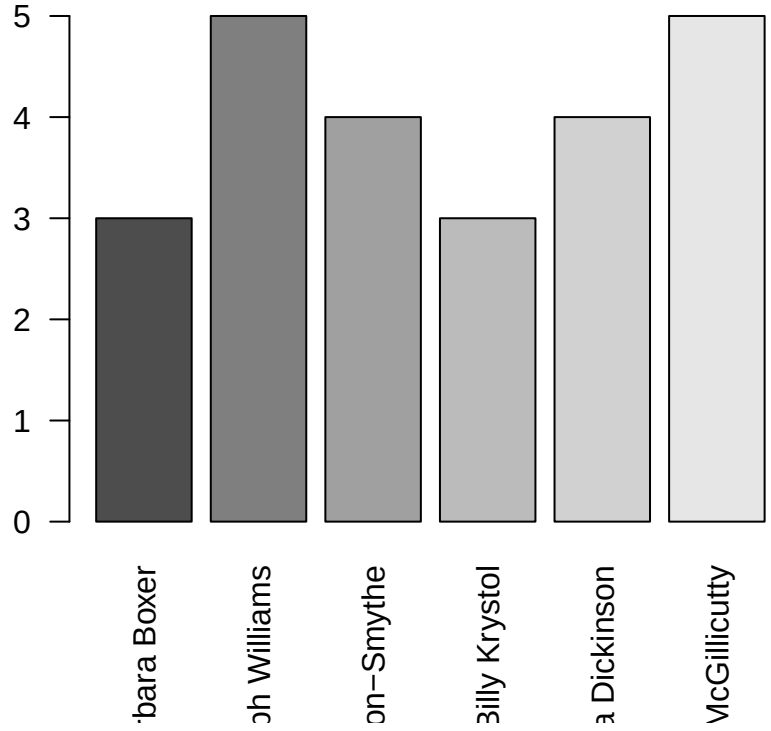


Figure 8.14: Barplot labels do fit

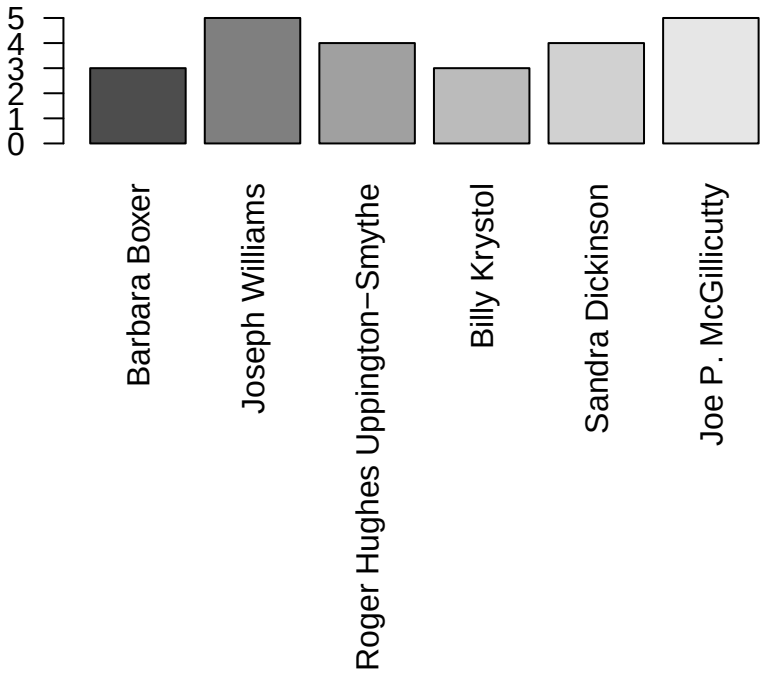


Figure 8.15: Legend does not fit

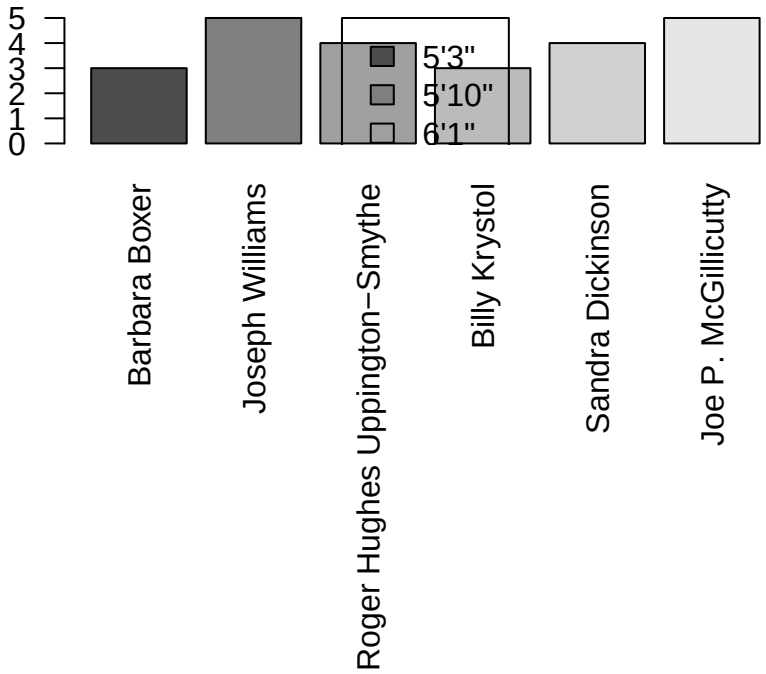
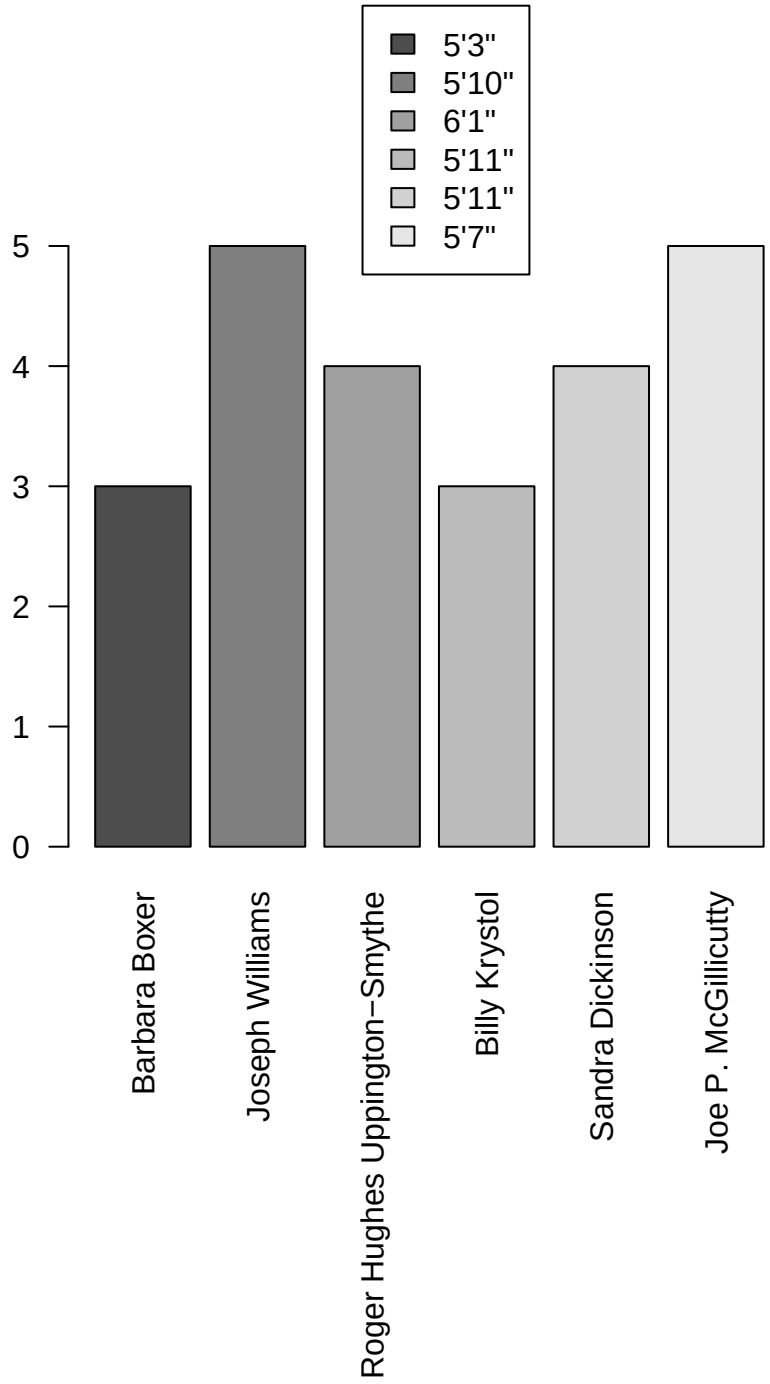


Figure 8.16: Push the legend into the margin area



$$\begin{array}{ll} plot\,1 & plot\,2 \\ plot\,3 & plot\,4 \end{array} \tag{8.7}$$

On the other hand, the command

> par(mfcol = c(2,2))

will fill in the columns first, as in

$$\begin{array}{ll} plot\,1 & plot\,3 \\ plot\,2 & plot\,4 \end{array} \tag{8.8}$$

Using the mfcol command, the following creates Figure 8.17 with three rows and two columns of plots.

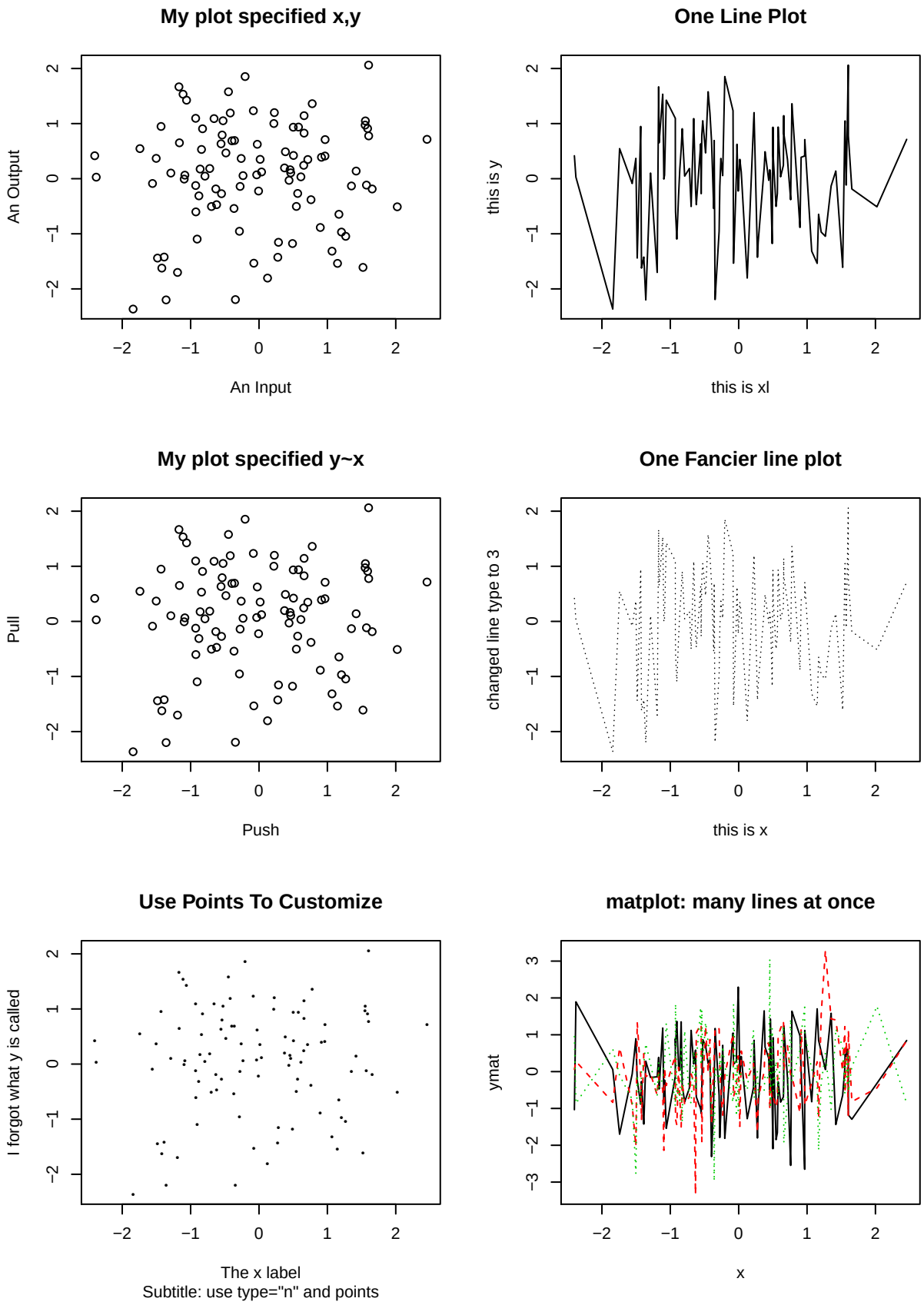
8.6 How are you saving your plots into files?

This section explains how to transfer image results from routines like plot, barplot and histogram, into image files that can be used in documents. In my experience, saving graphs in R is the most frustrating of all R experiences. Possibly, maybe, recoding factors is more frustrating, but I don't think so. In my Rtips web FAQ, I wrote a lot about this, so much that it discourages me to read it.

So here is the short and sweet version. If you create a graph on the screen, then you have drawn something on the screen. A file format would be a different device entirely, and it is not necessarily going to be easy to convert your screen "device" display into a file format that you can review later. In Linux, the screen device is probably the X11 device, or perhaps it is Cairo. On Windows, it is the Windows device. On a Macintosh, it is probably a Quartz device.

You can transfer the image on the screen into a file in a particular format. R has many output devices, some for creating "bitmap" pictures (like jpg or png), scalable vector graphics (svg), or even formatted line drawings for the xfig program (fig). Or you can run the plot command so it does not write on the "screen device" at all, but rather it goes straight into a file device. The advantage of writing straight into a file is that R will try to adjust the image so it "looks right" in the format of the file type you specified.

Figure 8.17: Many plots in one figure with mfcol



8.6.1 Write on the screen, copy to a file device

It may be easiest to begin by drawing on screen devices. The first thing to know is that you can start as many devices as you want. On Linux, execute this command a few times:

```
X11()
```

On MS Windows, try this:

```
window()
```

a few times. Then ask R to tell you which is the current device, meaning the one that your command will draw on if you run your command right now.

```
dev.cur()
```

There is a family of commands that can make a device the current one, such as "dev.next()" and "dev.set()". And when you want to close a device, run

```
dev.off()
```

Now draw a 2-D plot on the screen. Any kind of ordinary plot will work, meaning we could have something drawn by "plot", "hist", "barplot", "spineplot", and "dotplot". Those commands always write onto the current device.

Now suppose you want to save that in a file. There are many formats to choose from. Suppose for starters we choose to save an encapsulated postscript image file (eps). Encapsulated postscript is a postscript file that includes the "bounding box" information so that other programs can look at the file and discern its space requirements. The following would copy the screen display into a postscript file:

```
dev.copy(device = postscript, file = myExample.eps,
    onefile = F, horizontal = F, height = 5, width = 5,
    paper = "special")
dev.off()
```

The height and width are in inches. The option onefile=F tells R that I want the bounding box included. That makes this an encapsulated postscript (EPS) file, as opposed to an ordinary postscript file (ps). The option horizontal=F tells R I want a figure that is portrait oriented (not landscape). We don't want our figure centered in a large sheet of paper, we want

the document processing software to handle that, so we include the option paper="special". We may end up with huge white margins if we forget that.

It is necessary to run dev.off() at the end of all of this. Otherwise, that postscript file device remains open, waiting for you to draw some more material into the figure. dev.off() tells R to go ahead and write the information into the file. The resulting file, "myExample.eps," is saved in the current working directory.

When we are making figures for direct inclusion in LaTeX documents, the two preferred formats are encapsulated postscript and portable document format (PDF). For many years, R's default storage device was postscript, but recently that has changed to PDF. I have used both of these devices, depending on the desired output from a project. Ordinary LaTeX documents require EPS, while documents that are processed by the program "pdflatex" want PDF.

Because there are so many options with EPS and PDF devices, it is easy to forget them. It is possible to set defaults. I try to remember to have this "boiler plate" at the top of every R program.

```
ps.options(horizontal=F, onefile=F, family="Times", paper="special", height=4,
    width=6 )
pdf.options(onefile=F, family="Times", paper="special", height=4, width=6)
options(papersize="special")
```

By setting these options as default, then I need not enter them again when I actually use the device.

Some other document formats, such as web pages, do not want ps or pdf files. Instead, web browsers want image files. A "bitmapped" picture image format, say "jpg" or "png", will always work with web browsers, and the newer browsers will also accept images in the newer scalable vector graphics (svg) format.

If you decide to make a bitmapped "picture" file, try the png device:

```
dev.copy(device = png, file = "myExample.png", height = 800,
    width = 1000)
dev.off()
```

That produces a picture file that you can view with an image viewer. It always produces a "transparent" background when you use dev.copy().

The output from dev.copy() may not be very good. One problem is that devices on the screen display tend to be square, while the drawings we want to write into files are usually rectangular. If you tell R to write a 10x10 inch screen image into a 6x4 inch file format, the result will probably be. Fonts will get crushed, lines will be squished.

To avoid that problem, there are two approaches. The first approach is to create a fresh screen device of the right size. Then, when you draw on that, you are creating an image of the correct size, and saving it does not do so much damage. In MS Windows, R calls the screen device windows() while in Linux/Unix, it is x11(). So you start a new device with

```
> X11(height=6, width=4)
```

or

```
> windows(height=6, width=4)
```

Then you can go ahead and create the image with plot (or hist or whatever). Then use dev.copy to send it into a postscript, pdf, png, jpeg, or other file, specifying the height and width to match the screen device.

The second approach to the problem is described in the next section.

8.6.2 Write the plot directly into a file (bypass the screen altogether).

At first, this seemed so counter-intuitive to me that I could scarcely comprehend it. I notice many students strain against it as well. It is only natural to think of creating an image on the screen and then saving it into a file. But we have to think of the process differently. In this approach, we create an output device (pdf, png, jpg, png, etc), and then run the plot commands so that the output goes *straight into the file device*, without ever drawing on the screen. That's the counter-intuitive part. One runs the graphics commands and sends the results straight into a file, *without knowing for sure what they will look like.*

We are not blinding writing image files, however. We can fiddle around with interactive plot commands as long as we want. As soon as those commands are perfected, then they can be run again to produce a plot without displaying it on the screen. There is, no doubt, an element of uncertainty in this. What if the image that goes into the file is not right? Then one must open the code, fix it up, and run it over.

This way seems strange at first, but the beauty of it will quickly become apparent. Suppose one needs to create histograms for hundreds or thousands of variables. It is relatively easy to program a loop that will cycle through variables and save output automatically. We are sure that result files can be systematically regenerated any number of times. If one is writing a script that will create tens or hundreds of output files, it would be silly to design a system

that presupposes we will sit in front of the screen and save each image one-by-one. This way does not require any interactive pointing-and-clicking.

Suppose we have a set of plot commands that work perfectly with an on screen plot device. After those commands make an image that is "just right," with the correct labels and so forth, then we follow a 3 step procedure.

- Step 1. Open an 'output device'

```
postscript(file="myFigure.eps",height=4, width=6, horizontal=F, onefile=F
    , paper="special")
```

- Step 2. Run the commands to create the plot, as you would have done before

```
x ← 30 + 10 * rnorm(1000)
hist(x, breaks = 19, xlab = "Air speed of unladen sparrows",
     prob = T, main = "")
```

- Step 3. Close the output device.

```
dev.off()
```

The result is presented in Figure 8.18.

You can save yourself the trouble of re-running all of those postscript options if you have specified "ps.options" in the first part of your code. It is also possible to add specifications for the font and the size of the type, so that the figure will match a publication's style as closely as possible. The ps.options() will apply throughout an R session, until they are overridden by another use of ps.options().

Here is an example that creates a png output file, and the reader will agree that the background in this image *is green*.

```
png(file="mynew.png", height=300,width=400,bg="green")
hist(x, breaks=23, xlab="Air speed of unladen sparrows", prob=T, main=""))
dev.off()
```

Figure 8.18: A histogram saved in EPS format

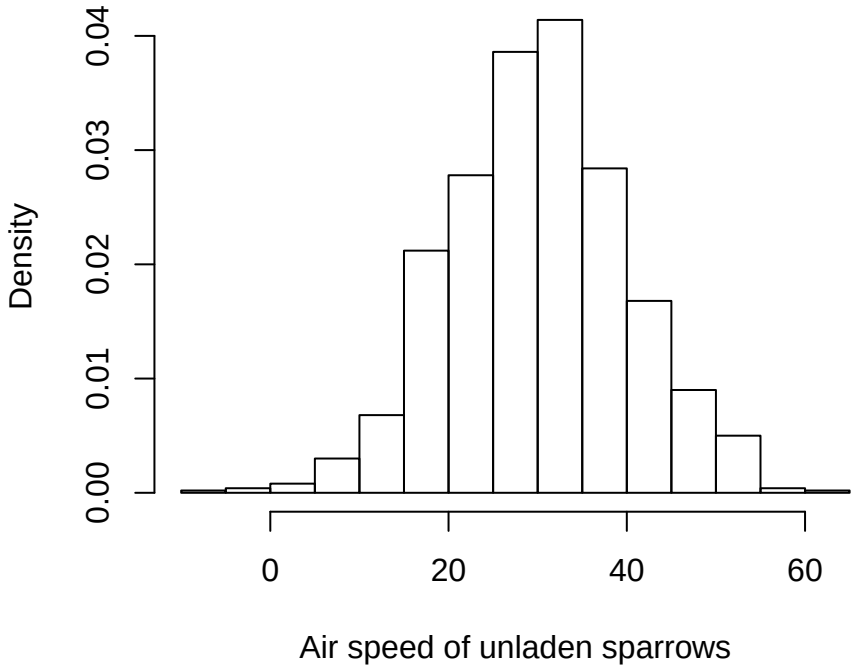
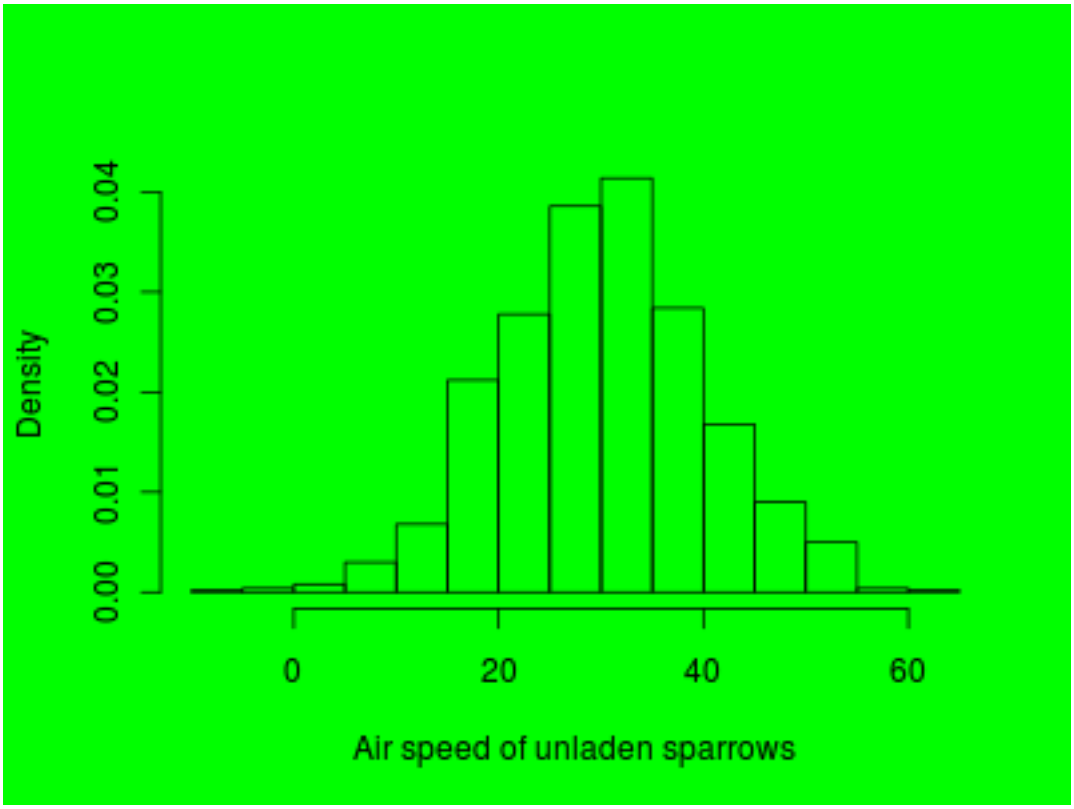


Figure 8.19: A png file



9 Cross Tabulation Tables

9.1 The Iron Laws of Crosstabs. (I inherited this from Gerhard Lowenberg, U Iowa).

The best possible crosstab has these properties.

1. The independent variable is on the top (as a consequence, the dependent variable is on the side).
2. Column percentages are calculated
3. Determine the impact of the independent variable by comparing across!

Here is what I mean:

Column Percentages		RESPONDENT	SEX	
The Spike Network is		male	female	
really horrible		25%	80%	
OK		50%	18%	
really great		25%	2%	

Or, for another example, concerning the 1980 presidential election:

		Party Identification			
	Republican	Independent	Democrat		
Percentages voting for					
Reagan	86%	55	26		
Carter	9	30	67		
Anderson	4	12	6		

The importance of the flag in the 1988 election. This was compiled from the American National Election Study of 1988.

How does seeing the American flag make you feel?

	Major Party Voters			
	Extremely good	Very good	Somewhat good	Not very good
Percentages voting for				
Bush	60.7%	50	30	8
Dukakis	39.3	50	70	92

It does not hurt, and usually helps, if you add Marginal Values Like so

Column Percentages		RESPONDENT	RESPONDENT	Total %
The Man Show is		male	female	
really horrible		25%	80%	50.6%
OK		50%	18%	38.0%
really great		25%	2%	11.4
Total N		1000	600	1600

In order to interpret a cross tabulation table, it is necessary to compare across, and a difference in values indicates that a respondent's position among the columns "makes a difference."

How big is the difference? There are two meanings to this question.

1. Is there a "numerical index" to tell us how big the difference is, so that we can compare tables by using a numerical score, and

2. Is there a way to tell if the difference you observe is big enough to lead you to conclude that it is not simply a result of "random sampling."

The first question leads down a very tedious path in which people propose different "correlation coefficients" to summarize the relationships observed. Many different coefficients have been proposed, Phi, Gamma, tau. These are hard to justify, except in some specific cases. Otherwise, they make me tired, but I've used them. Here's an easy case.

		Male	Female
	Yes	100%	0%
	No	0%	100%

And this is also an easy case:

		Male	Female
	Yes	88%	88%
	No	12%	12%

It seems to me that, in the first example, the relationship between Gender and the Magic Question is very powerful, and so it should have the maximum score, say 1.0. The second one seems to indicate that Gender does not predict anything about the Magic Question, and so the score should be 0.

Unfortunately, in "real data" we see more hard cases than easy ones. That's why people bicker back and forth about ways to numerically summarize the power of a relationship like this:

		Male	Female
	Yes	88%	44%
	No	12%	56%

For the second question, we have a better answer, the χ^2 statistic. This "chi-square" statistic follows the usual reasoning of inferential statistics. How far do the observed results differ from what we would expect if the data were created by a "completely random" or "completely non-patterned" process? We figure out what we would Expect for each cell, and then we note that the Observed value differs, and we square those differences, and weight them, and add them up. If the resulting number is really huge, it means the table is far from what we would expect if the process generating the data were "random."

9.2 Using R to make Cross Tabulations

In R, the table() and prop.table() functions are in the base statistics package. These can be used to obtain raw numbers and column percentages. The function xtabs() can be used to obtain more sophisticated cross tabulations.

It is common to call these Crosstabs, but the correct term is "cross tabulation table." As far as I know, the computer program SPSS, the Statistical Package for the Social Sciences, is the source of the slang Crosstabs. Crosstabs was the name of the routine that produced cross tabulations in SPSS. In the gmodels package, there is a function called CrossTable that creates a very informative table almost exactly like the tables that were produced by SPSS. Like the SPSS tables, they usually have too much information. There are options where you can tell it to omit some percentages, so you only see the numbers you really want.

I've generated two variables, x and y. Let's suppose x represents "Gender", coded 1 for Female and 2 for Male and y represents "Voted in 2008", coded 1 for No and 2 for Yes.

```
> x <- ifelse( rnorm(100) > 0, 1, 2)
> y <- ifelse( rnorm(100) > 0, 1, 2)
```

Each cell in the output table has the number of cases as well as a number of other values. The output from the R table function gives a simple set of "counts" for each combination of values.

```
> table(y,x)
   x
y    1  2
  1 28 23
  2 24 25
```

If we want column proportions, we can wrap the table command with prop.table, letting R know that we want to use the 2nd dimension, the columns, for calculating proportions:

```
> prop.table ( table(y,x), 2)
   x
y           1         2
  1 0.5384615 0.4791667
  2 0.4615385 0.5208333
>
```

We are still pretty far from a completely informative table, at least as far as an SPSS user would be concerned. I'd like to know row percents, total percents, and so forth, all in one easy SPSS style table. The more familiar output is found from the CrossTable command in the gmodels package.

```
> library(gmodels)
> CrossTable(x,y)

   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|
Total Observations in Table:  100
Total Observations in Table:  100

             | x
           y |         1 |         2 | Row Total |
-------------|-----------|-----------|-----------|
           1 |        28 |        23 |        51 |
             |     0.083 |     0.089 |           |
             |     0.549 |     0.451 |     0.510 |
             |     0.538 |     0.479 |           |
             |     0.280 |     0.230 |           |
-------------|-----------|-----------|-----------|
           2 |        24 |        25 |        49 |
```

```
             |     0.086 |     0.093 |           |
             |     0.490 |     0.510 |     0.490 |
             |     0.462 |     0.521 |           |
             |     0.240 |     0.250 |           |
-------------|-----------|-----------|-----------|
Column Total |        52 |        48 |       100 |
             |     0.520 |     0.480 |           |
-------------|-----------|-----------|-----------|
```

When I prepared the tables in the previous section, I used the LaTeX tabular as an organizer and I typed in the numbers by hand. That is a pretty tedious process. I've tried various approaches, but they generated LaTeX output that was so inadequate that I thought it was easier to simply type up a table (here I'm referring to the latex() function in Hmisc and the xtable package). In 2009, however, I did find a workable approach, which is described in the next section.

Getting Nice LaTeX tables out of R

The R package "memisc" offers a wealth of functions that will facilitate social science research presentations. I was drawn to that by a post in r-help that described its facilities to generate LaTeX tables for statistical reports. At the current time, I am persuaded that memisc is worthy of consideration because it offers tools to make not only nice looking cross tabulation tables in LaTeX format, but it also has a nice tool for collection and presentation of regression tables. Before I knew of memisc, I had used other techniques to generate LaTeX tables in R. A package called "xtable" can generate a latex table. In the Hmisc package, there is a latex function that generates some tables. I've experimented with several packages for making regression tables. It appears to me that memisc is the only one so far that offers all of these things within one coherent vocabulary.

For cross tabulation, the memisc package provides a generating function called genTable() that uses a pleasant, simple syntax:

```
genTable(percent(y)~x, data=mydata)
```

The memisc package follows the syntax of the xtabs function in R. The output is not exactly what one expects, but it is very powerful. A table that is consistent with the Iron Law of Crosstabs can be obtained in this way. If one wants LaTeX output, one can either do

```
mytable <- genTable(percent(y) ~ x, data=mydata)
toLateX(mytable)
```

or in one step,

```
toLatex(genTable(percent(y) ~ x, data=z))
```

The percent function is just one of the possible summary function that might be used on the left hand side. This is the point at which a study of xtabs will be helpful. The documentation on genTable clearly assumes that the reader understands xtabs. The right hand side can accept additional variables,

```
genTable(percent(y) ~ x + z, data=mydata)
```

so that summary values for all possible combinations of the input variables will be represented.

Here is an example of the output from toLatex() in the memisc package.

```
\begin{tabular}{lD{.}{.}{0}D{.}{.}{0}}
\toprule
 & \multicolumn{1}{c}{F} & \multicolumn{1}{c}{M} \\
\midrule
C & 8 & 12 \\
T & 22 & 18 \\
\bottomrule
\end{tabular}
```

This LaTeX markup uses some unusual options, \toprule, as well as an unusual syntax for column alignment (the D in tabular). These unusual pieces from toLatex will cause the latex processor to "crash" unless some additional packages are inserted into features will cause the latex processor to "crash" unless some additional packages are inserted into the LaTeX document's preamble. The required packages are dcolumn and booktabs (more on that in a moment).

While I was trying to solve that problem, I stumbled on some literature about standards for the publication of tables. Consider the essay "Publication quality tables in LaTeX," which can be found online at `http://www.ctan.org/tex-archive/macros/latex/contrib/booktabs/booktabs.pdf`. It was written as a part of the development for a LaTex package called "booktabs" that is intended to facilitate professional looking tables in LaTeX documents. Another essay by Lapo Fillipo Mori, called "Tables in LaTeX 2_ε : Packages and Methods" offers a broader overview (obtain that at:`http://www.tug.org/pracjourn/2007-1/mori/mori.pdf`). The first two rules are 1) never include any vertical lines in tables, and 2) never use a "double horizontal" line.

Some people do not agree with all of these guidelines, but most seem to agree with the broad strokes. A table that looks "just right" cannot be obtained unless some additional latex packages are used. Some R to LaTeX approaches avoid these additional packages, even at the expense of generating tables that are not quite optimal. These commands create a table that does not quite conform:

```
library (Hmisc)
mt <- with(reaction.time, table( control, gender))
latex(addmargins(mt,1) file="")
library(memisc)
```

In the LaTeX output, one will notice a double horizontal line produced by two \hline uses.

```
\begin{center}
\begin{tabular}{lrr}\hline\hline
\multicolumn{1}{l}{with}&
\multicolumn{1}{c}{F}&
\multicolumn{1}{c}{M}
\\ \hline
C&$ 8$&$12$\\
T&$22$&$18$\\
\hline
\end{tabular}
\end{center}
```

In LaTeX, that generates a table that looks like this:

	F	M
C	8	12
T	22	18

The output from memisc does not use \hline, but it will be be viewable at all unless two new packages are available in the LaTex system and those packages are inserted in the LaTeX document's preamble.

Step 1. In the document preamble, use either "\usepackage{ctable}" or "\usepackage{booktabs}". These packages provide the table separators \toprule, \midrule, and \bottomrule. (I don't know the difference between ctable and booktabs, both seem to work. It seems the latter may be more popular.)

Step 2. The preamble needs "\usepackage{dcolumn}".

The LaTeX package "dcolumn" facilitates the positioning of columns in which some values have decimal values. After inserting the preamble option \usepackage{dcolumn}, the output from memisc looks fine.

After re-configuring the document's preamble, the result of that LaTeX markup from toLatex will be:

	F	M
C	8	12
T	22	18

That output is acceptable, as far as I can tell.

If the dcolumn package for LaTeX is not available, there is a work-around. The standard syntax for a tabular in LaTeX uses "l", "c", or "r" (for "left", "center", or "right"). Replace that complicated looking D{}{} stuff with either "l", "c", or "r". This modified code for a tabular environment removes the usage of D.

```
\begin{center}
\begin{tabular}{lrr}
\toprule
 & \multicolumn{1}{c}{F} & \multicolumn{1}{c}{M} \\
\midrule
C & 8 & 12 \\
T & 22 & 18 \\
\bottomrule
\end{tabular}
\end{center}
```

It generates the following table.

	F	M
C	8	12
T	22	18

A table that includes column percentages can be obtained with this command:

```
toLatex(genTable(percent(control) ~ gender, data=reaction.times))
```

The LaTeX markup that results is

```
\begin{tabular}{lD{.}{.}{0}D{.}{.}{0}}
\toprule
 & \multicolumn{1}{c}{F} & \multicolumn{1}{c}{M} \\
\midrule
C & 27 & 40 \\
T & 73 & 60 \\
N & 30 & 30 \\
\bottomrule
\end{tabular}
```

That leads to a table object that looks like this;

	F	M
C	27	40
T	73	60
N	30	30

I googled "multicolumn latex" for 2 minutes and understood how to modify this so that the column structure and labels match my preferences. I also inserted \% for percent signs. The following is coming pretty close to the ideal:

	Respondent Sex	
Classification	Female	Male
Control	27%	40%
Test	73%	60%
N. of Cases	30	30

I think the following output is even more pleasant, but it requires more hacking on the LaTeX code and there is more danger of making a mistake that causes latex to crash.

		Respondent Sex	
		F	M
Classification	Control	27%	40%
	Test	73%	60%
	N.of Cases	30	30

Here's the R code that generates most of this example output

```
 library(UsingR)
## Ordinary table syntax wants the row variable first, column second
## So, if we are thinking of "control" in the reaction.time as the output
with(reaction.time, table( control, gender) )
library (Hmisc)
mt <- with(reaction.time, table( control, gender))
prop.table(addmargins(mt,1))
latex(addmargins(mt,1) file="")
library(memisc)
genTable(gender~control, data=reaction.time)
genTable(percent(gender)~control, data=reaction.time)
## Maybe you view "control" as the output?
## This generates total counts for each column:
toLatex(genTable(control~gender, data=reaction.time))
## want percents?
toLatex(genTable(percent(control)~gender, data=reaction.time))
```

10 Summation Signs

10.1 Definition

The summation sign $\sum$ is ever present in statistics.

$$\sum_{i=1}^{N} x_i \tag{10.1}$$

It means that N terms are added together and the index variable i counts up from 1 to N.

$$x_1 + x_2 + x_3 + \ldots + x_{N-1} + x_N \tag{10.2}$$

We are not required to use i as the index variable. We could, for example, use j

$$\sum_{j=1}^{N} x_j \tag{10.3}$$

or even a word like *case*

$$\sum_{case=1}^{N} x_{case} \tag{10.4}$$

Using a word as the index variable is uncommon, mainly because it requires so much more typing!

In many social science projects, we thinking of scores on a survey, such as the ages of respondents. Consider $x_1 = 9$, $x_2 = 22$, $x_3 = 29$, $\ldots, x_{59} = 101$. With those scores in mind, the average, which is customarily referred to as $\bar{x}$, would be defined as

$$\bar{x} = \frac{1}{59}\sum_{i=1}^{59} x_i = \frac{x_1 + x_2 + x_3 + \ldots + x_{59}}{59} = \frac{9 + 22 + 29 + \ldots + 101}{59} \tag{10.5}$$

Most statistics books will "abstract away" from the explicit number of cases, replacing 59 by N.

$$\frac{1}{N}\sum_{i=1}^{N} x_i \;\; or \;\; \frac{\sum_{i=1}^{N} x_i}{N} \tag{10.6}$$

In graduate research, we take the next step to think about sums of more abstract things. If we transform each x_i by a function f, (that is to say, calculate $f(x_i)$ for x_1, x_2, and so forth, the sum of the transformed values would be represented as:

$$\sum_{i=1}^{N} f(x_i) \tag{10.7}$$

It means that N terms are added together and the variable i counts up from 1 to N.

$$f(x_1) + f(x_2) + \ldots + f(x_N) \tag{10.8}$$

Any function can be included.

Many sums will involve the combination of several variables, as in

$$\sum_{i=1}^{N} x_i + z_i^2 = x_1 + z_1^2 + x_2 + z_2^2 + x_3 + z_3^2 + \ldots + x_N + z_N^2 \tag{10.9}$$

or

$$\sum_{i=1}^{N} (x_i - \bar{x})^2 = (x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_N - \bar{x})^2 \tag{10.10}$$

In the previous examples, the terms within the sum have used i only as the index variable, referring to specific elements x_i or z_i. It is, however, perfectly legitimate to have i as an element in the calculation. That is, f may depend on the value of the 'index' variable i . To be explicit, write

$$\sum_{i=1}^{N} f(x_i, i) \tag{10.11}$$

It should be easy to spot the pattern here, for example:

$$1 + 4 + 9 + 16 \tag{10.12}$$

is the same as

$$\sum_{i=1}^{4} i^2 \tag{10.13}$$

Sometimes the data is presented to us in a "grouped" format. We do not have data on the individuals who were surveyed (no x_i's as in equation 10.5). Instead, we are only told that there are 6 observations for people aged 1, 3 for people aged 2, 5 for people aged 3, and so forth. To calculate the sum of those scores, we could just write those out:

$$1 + 1 + 1 + 1 + 1 + 1 + 2 + 2 + 2 + 3 + 3 + 3 + 3 + 3 \tag{10.14}$$

Writing those scores in that way would be very tedious, however. Suppose we create a new variable to hold the observed counts, $g_1 = 6$, $g_2 = 3$, $g_3 = 5$. Then we could say the sum is

$$\sum_{j=1}^{3} j \cdot g_j \tag{10.15}$$

I used the index variable j to remind myself that I am summing over groups, not individuals.

One of the common uses of summations in which the terms depend on i arises in game theory. It is sometimes important to consider the value of a stream of payoffs over time. Suppose a person is to receive 3 "units" today, and 3 tomorrow, and so on. A payment of 3 units today is actually worth 3, but is a payment in 5 years equally valuable? Not in the eyes of most people. It seems likely the person will "discount" future payments. A payment at a future period might be down-weighted by multiplying by 0.9, for example. And a payment two days from now might be downgraded by 0.9^2. The "present value" of an infinite stream of 3's would be represented by this sum, which is know as a "geometric series":

$$\sum_{i=0}^{\infty} 3 \cdot 0.9^i = 3 + 3 \cdot 0.9 + 3 \cdot 0.9^2 + 3 \cdot 0.9^3 + \ldots + 3 \cdot 0.9^{\infty} \tag{10.16}$$

Usually, we would treat the so-called "weighting factor" in an abstract way, calling it w, and we would write

$$\sum_{i=0}^{\infty} 3 \cdot 0.9^i = 3 + 3 \cdot w + 3 \cdot w^2 + 3 \cdot w^3 + \ldots + 3 \cdot w^{\infty} \tag{10.17}$$

We could make that more abstract by allowing the payoffs to vary. Suppose the amount received at time i is represented by P_i, then we would have

$$\sum_{i=0}^{\infty} P_i w^i \tag{10.18}$$

The lower index and higher index values can be any integers, as long as the top is bigger than the bottom:

$$\sum_{i=14}^{17} i = 14 + 15 + 16 + 17 \tag{10.19}$$

10.2 Linearity: The most important property

First, observe

$$\sum_{i=1}^{n} af(x_i) = a\sum_{i=1}^{n} f(x_i) \tag{10.20}$$

Second, observe

$$\sum_{i=1}^{n}\{f(x_i) + g(y_i)\} = \sum_{i=1}^{n} f(x_i) + \sum_{i=1}^{n} g(y_i) \tag{10.21}$$

Put those 2 things together

$$\sum_{i=1}^{n}\{ax_i + by_i\} = a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} y_i$$

10.3 The Importance of Linearity: proportionality

Suppose the mean of a variable *education* is 8.5 years. If each year of school is composed of two semesters, what is the mean number of semesters? Obviously, it is 17. Each school year has 4 quarters. If education is measured in quarters, what will the mean be? Obviously, 34.

Since the mean is calculated as a summation of scores, linearity means that if each element is multiplied by a constant, then the mean is changed similarly. It is not necessary to actually "go through" and multiply the data in years by 2 and then re-sum. We do not have to go term-by-term:

$$\frac{1}{N}\sum_{i=1}^{\infty}(2 \cdot education_i) = \frac{2 \cdot 6 + 2 \cdot 10 + 2 \cdot 14 + 2 \cdot 8 \ldots}{N} \tag{10.22}$$

We don't need to do that calculation because it is obvious that if each score is multipled by 2, the sum is multiplied by two:

$$\frac{1}{N}\sum_{i=1}^{\infty}(2 \cdot education_i) = 2 \cdot \frac{1}{N}\sum_{i=1}^{N} education_i \tag{10.23}$$

10.4 The Importance of Linearity:Simplicification

Linearity also allows us to simplify some calculations. The fact that "the summation of the sum is the sum of the summations" can help us to see things in a new way. The quantity in equation 10.10 is known as the "sum of squares". It can be simplified by the application of the principle of linearity. Observe that

$$\sum_{i=1}^{N}(x_i - \bar{x})^2 = \sum_{i=1}^{N}(x_i - \bar{x}) \cdot (x_i - \bar{x}) \tag{10.24}$$

$$= \sum_{i=1}^{N}(x_i^2 - 2x_i\bar{x} + \bar{x}^2) \tag{10.25}$$

$$= \sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} 2x_i\bar{x} + \sum_{i=1}^{N} \bar{x}^2 \tag{10.26}$$

The first sum cannot be simplified, but we can combine the last two. Recall that $\bar{x}$ is a "constant" in this case: $\bar{x}$ is the same for all observations (the average height of American men is 5'10" for all men). That means that

$$\sum_{i=1}^{N} 2x_i\bar{x} = 2\bar{x} \cdot \sum_{i=1}^{N} x_i \tag{10.27}$$

Since $\bar{x} = \frac{1}{N}\sum x_i$, then $\sum x_i = N\bar{x}$. So we replace the sum on the right with

$$\sum_{i=1}^{N} 2x_i\bar{x} = 2\bar{x} \cdot \bar{x} = 2\bar{x}^2 \tag{10.28}$$

Similarly, it should be obvious that

$$\sum_{i=1}^{N} \bar{x}^2 = N \cdot \bar{x}^2 \tag{10.29}$$

As a result, equation 10.26 becomes

$$\sum_{i=1}^{N} x_i^2 - 2N\bar{x}^2 + N\bar{x}^2 = \sum_{i=1}^{N} x_i^2 - N\bar{x}^2 \tag{10.30}$$

The first term cannot be simplified further, but we have made a lot of progress. In words, the previous result indicates that the sum of squares is equal to the sum of squared observations minus N times the mean squared. Well, that does not seem much simplier, but it does help quite a bit.

10.5 Exercises

1. Use summation notation to simplify this:

$$2 + 4 + 6$$

2. Use summation notation to simplify this:

$$5 * x_1 + 5 * x_2 + 5 * x_3 + 5 * x_4 + 5 * x_5$$

3. Use summation notation to simplify this:

$$5 * x_1 + 25 * x_2 + 125 * x_3$$

4. Use summation notation to simplify this:

$$2^{14} * x_{14} + 2^{15} * x_{15} + 2^{16} * x_{16} + 2^{17} * x_{17} + 2^{18} * x_{18}$$

5. Sometimes you can't simplify expressions with sums. Sometimes you can. Can this be simplified?

$$4 * x_i + 4 * x_i^2 + 4 * x_i^3 + 4 * x_i^4$$

6. Sometimes you can't simplify expressions with sums. Sometimes you can. Can this be simplified?

$$4 * x_i + 4 * x_i^2 + 8 * x_i^3 + 19 * x_i^4$$

11 Curves in theory

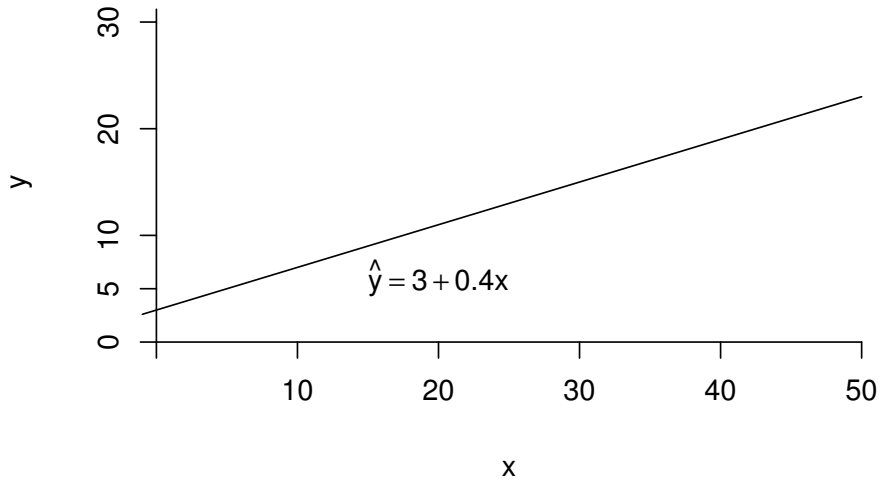
Recall main idea. You are allowed to make "one prediction" for each combination of input variables. That "one prediction" is the thing we call the "regression line", assuming it is a line.

11.1 Straight lines.

11.1.1 One Simple Line

Plot the curve

$$\hat{y}_i = 3 + 0.4 * x_i \tag{11.1}$$



Question: What is "3" representing?

Answer: That's the "constant" (or "intercept")

Question: What is "0.4" representing?

Answer: That's the "slope," the number of units that $\hat{y}_i$ goes up for each 1 unit increase in x_i.

Does it bother you to refer to this as

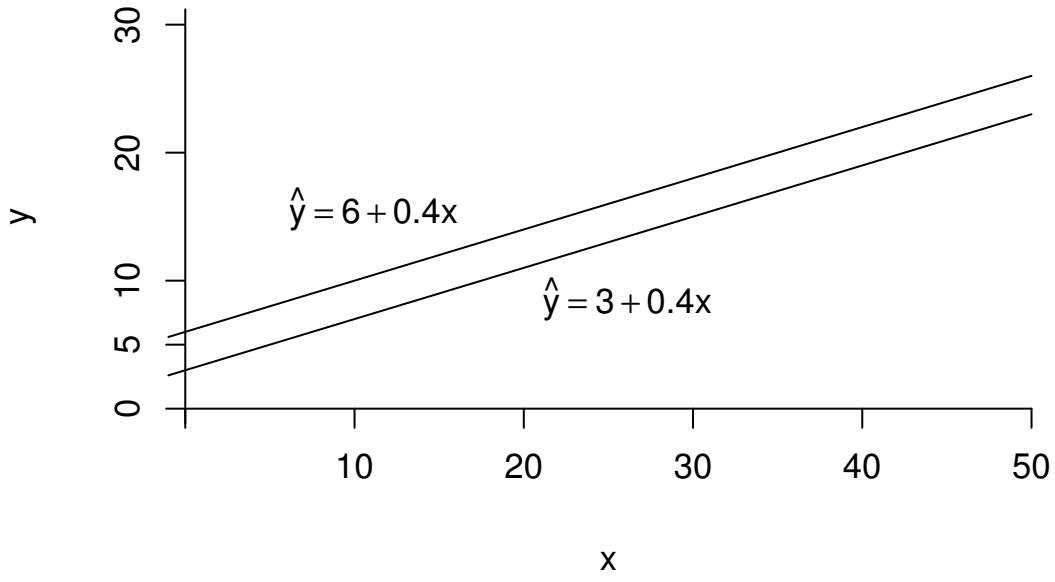
$$\hat{y}_i = b_0 + b_1 x_i \tag{11.2}$$

where $b_0 = 3$ and $b_1 = 0.4$?

It may bother you if somebody hit you over the head lots and lots of times with $y = mx + b$. But you have to try to get past that trauma.

11.1.2 An Intercept Shift?

Put a gap between two lines. What would that mean? It might mean there are 2 social groups, and each is represented by one line. The respondents within the two groups have the "same response" to an increase in x, but they do not begin at the same starting point. There are different y-intercepts, in other words.



Try to figure out a nice looking equation to describe that relationship.

$$\begin{aligned} Let\, z_i &= 0 \quad if\, x_i < 10 && (11.3)\\ &= 1 \quad if\, x_i \geq 10 && (11.4) \end{aligned}$$

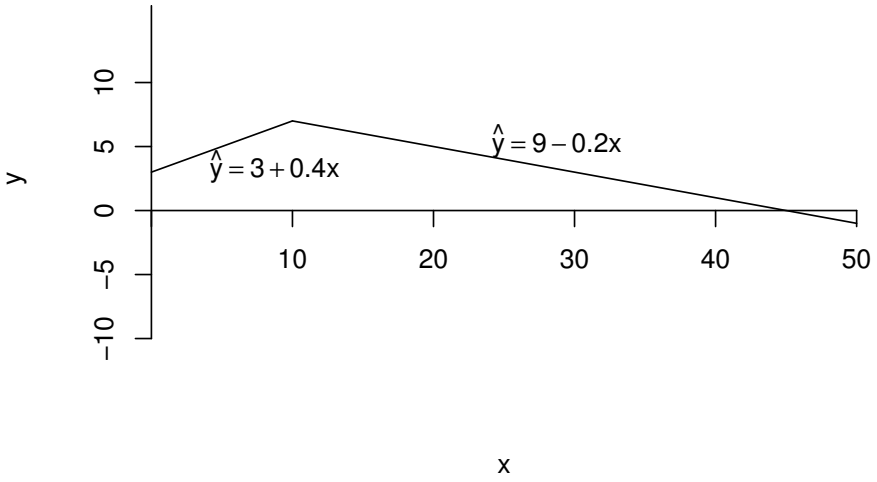
Then

$$\hat{y}_i = 3 + 3z_i + 0.4x_i \tag{11.5}$$

11.1.3 A segmented line

What if I said that if $x_i < 10$, then $b_0 = 3$ and $b_1 = 0.4$, but if $x_i \geq 10$, then there is a different relationship where $b_1 = -0.2$. For the moment, we will insist that the curve is not "broken," you can draw it without lifting your pencil.

Plot that idea:



Does the formula for the line on the right side of your picture have a different equation than the line on the left side? How does changing b_1 imply a change in b_0?

We try to represent the relationship across the whole domain by one equation. This seems simple, but it took me about 30 minutes to remember how to do it.

$$\begin{aligned} Let\, z_i &= 0 \quad if\, x_i < 10 && (11.6)\\ &= 1 \quad if\, x_i \geq 10 && (11.7) \end{aligned}$$

$$\begin{aligned} \hat{y}_i &= 3 + 0.6 \cdot 10 \cdot z_i + 0.4x_i - 0.6x_i z_i && (11.8) \\ &= 3 + 0.6 \cdot 10 \cdot z_i + (0.4 - 0.6z_i)x_i && (11.9) \end{aligned}$$

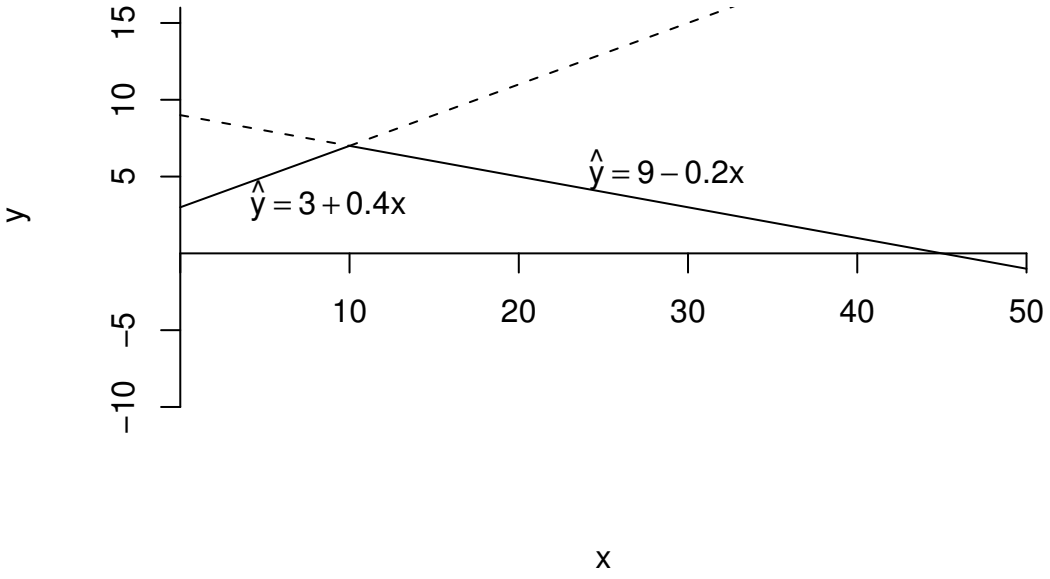
How could you describe the idea of a "slope shift" with this setup? Maybe it represents a "change point" where x_i has a different effect, after, say, a change of presidential administration. If something significant happens at $x_i = 10$, then it may be interesting to consider the possibility that before the change, our prediction is

$$\hat{y}_i = 3 + 0.4x_i$$

while after the shift, our prediction is

$$\hat{y}_i = 9 - 0.2x_i$$

Note that although the "y-intercept" in equation 11.10 may appear to be unaffected, the two lines in the graph really do have different y-intercepts. That's easier to visualize if dotted lines are added to the figure, so we see there really are two separate lines being plotted:



It is just not possible to "tilt" a line and change its slope without changing the intercept. Even though I've tried to do so in many handouts over the years.

11.1.4 Spline: You can break up a line many times.

A Spline is the result of "piecing together" several lines. It is sometimes called a "segmented linear" relationship.

11.1.5 Add an intercept shift and a slope shift.

In the segmented line we have drawn, there is no visible "gap" in the graph when it changes direction. The two lines being plotted "cross" at $x_i = 10$. The impact of x_i on y_i simply appears to change direction. One could allow a gap as well, however, by combining the intercept-shift and the slope shift.

$$\begin{aligned} \hat{y}_i &= 3 + 3z_i + 0.4x_i - 0.6x_i z_i && (11.10) \\ &= (3 + 10z) + (0.4 - 0.6z_i)x_i && (11.11) \end{aligned}$$

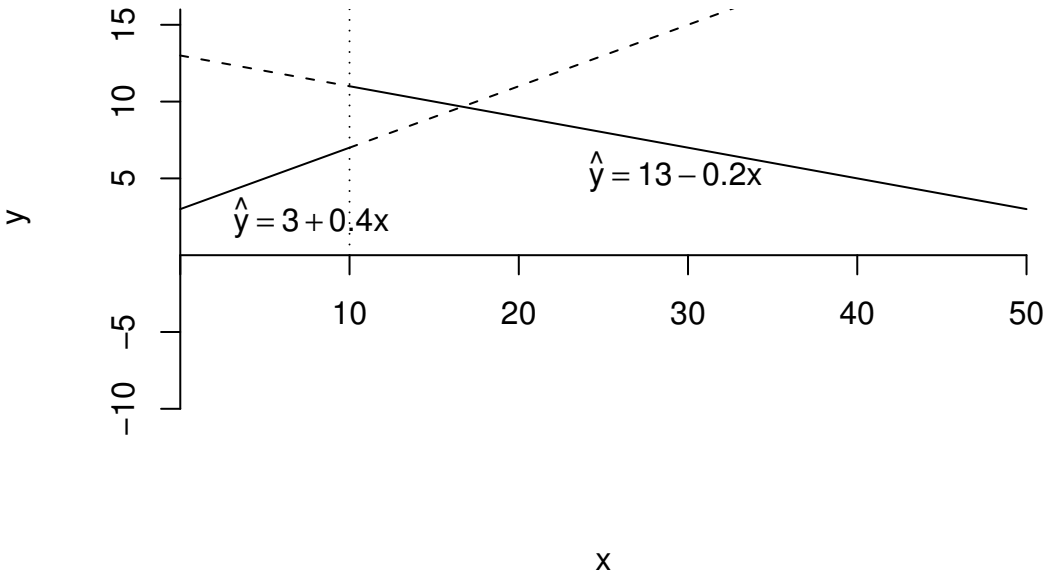
The line being graphed jumps from one value to another when $x = 10$.

$$\hat{y}_i = 3 + 0.4x_i \tag{11.12}$$

to this:

$$\hat{y}_i = 13 - 0.2x_i \tag{11.13}$$

If you draw these two lines, you will see that, where $x_i = 10$, there is a "gap" between the two lines.



11.2 Polynomial

After you get tired of

$$\hat{y}_i = b_0 + b_1 x_i \tag{11.14}$$

then you wonder if, perhaps, the effect of x_i is affected by the level of x_i itself. Suppose

$$b_1 = c_1 x_i \tag{11.15}$$

If we replace the "variable slope" b_i in equation 11.14 by the equation 11.15, then the result would be a nonlinear equation:

$$\begin{aligned} \hat{y}_i &= b_0 + (c_i x_i) x_i & (11.16) \\ &= b_0 + c_1 x_i^2 & (11.17) \end{aligned}$$

The x_i^2 term creates "curvature" in the graph of the relationship.

The introduction of the squared term is not the end of the process, however. Suppose your theory about $b_1$were more elaborate, say

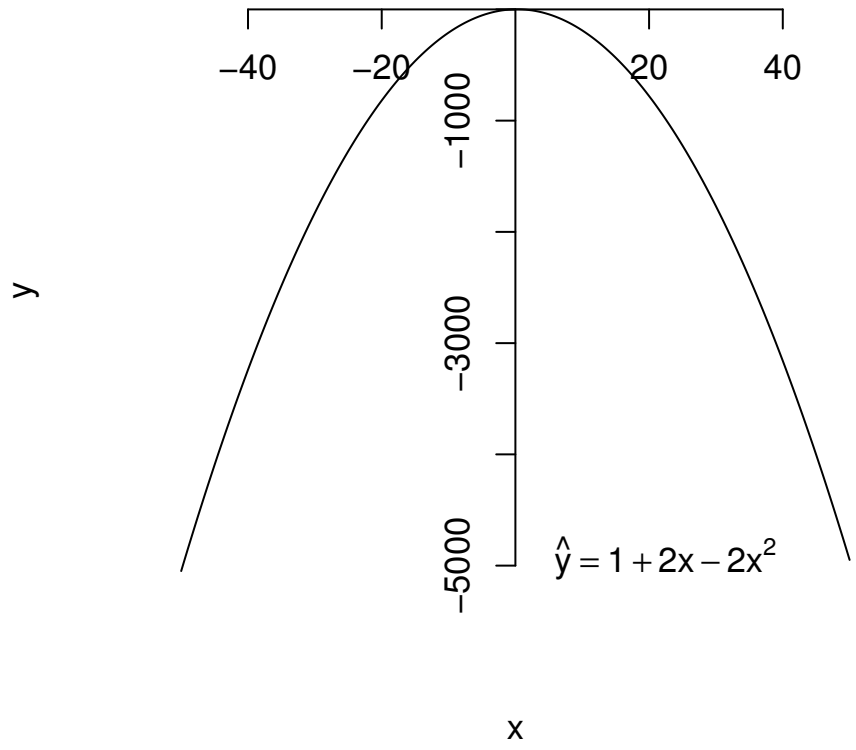
$$b_1 = c_0 + c_1 x_i \tag{11.18}$$

then when we insert that into the original expression, the result would be

$$\hat{y}_i = b_0 + (c_0 + c_1 x_i) x_i \tag{11.19}$$
$$\hat{y}_i = b_0 + c_0 x_i + c_1 x_i^2 \tag{11.20}$$

Usually, you'd re-label the parameters in some pleasant way, just using b_0, b_1 and b_2, or a,b, and c, but this is your friend "the quadratic equation." (If $y = ax^2 + bx + c$ doesn't make you think $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, then you didn't go to the same high school that I did.)

Suppose $b_0 = 1$, $c_0 = 2$, and $c_1 = -2$. Make a little sketch:



11.2.1 Go ahead, add more powers of x_i

An equation that has x_i and x_i can have at most one "bend". It has either a peak or a valley, never more than one of either. If the relationship you have in mind has a more complicated

shape, then perhaps you can add a cubic component, as in:

$$\hat{y}_i = b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3 \tag{11.21}$$

An equation like this can have at most two "bends."

There's a theorem that says that if you keep adding higher powers, like x_i^4, x_i^5 and so forth, then you can make a line bend and closely approximate any desired shape. We don't end up doing that kind of thing very often because, as we add powers of x_i, then it becomes more and more difficult to differentiate the influence of the various terms. Although it is mathematically correct to pursue a curvature problem by fitting a polynomial with more and more powers, it is not statistically tractible to do so.

Instead of trying to formulate one giant equation to get the desired curvature, we are much more likely to use an approach that is based either on a spline (divide x_i up into segments and fit a curve on each one) or a locally weighted regression (loess) that makes a prediction for a point x_0 with a model that puts most of its weight on points closest to x_0.

11.3 Logarithm. You can call it log for short.

11.3.1 Definition.

$$y = log_{base} x \tag{11.22}$$

means "to what power y would we need to raise *base* in order to get x?" That is, what value of y would make the following true:

$$base^y = x \tag{11.23}$$

Some people object that using logs is "thinking backwards." I agree.

For example, we know that

$$2^3 = 8$$

and so

$$log_2 8 = 3$$

What power of 2 is required to produce a result of 8.

11.3.2 An increasing relationship

If you want a relationship in which y goes up with x, but it goes up more and more slowly, then you want a logarithm.

The base of the natural logarithm, e, is about 2.7. e is called "Euler's Constant" (Euler is pronounced "oiler").

$$\hat{y}_i = log_e(x_i) = ln(x_i) \tag{11.24}$$

has a slope of $1/x_i$. Note that "ln" means "log_e". This is called a "natural logarithm."

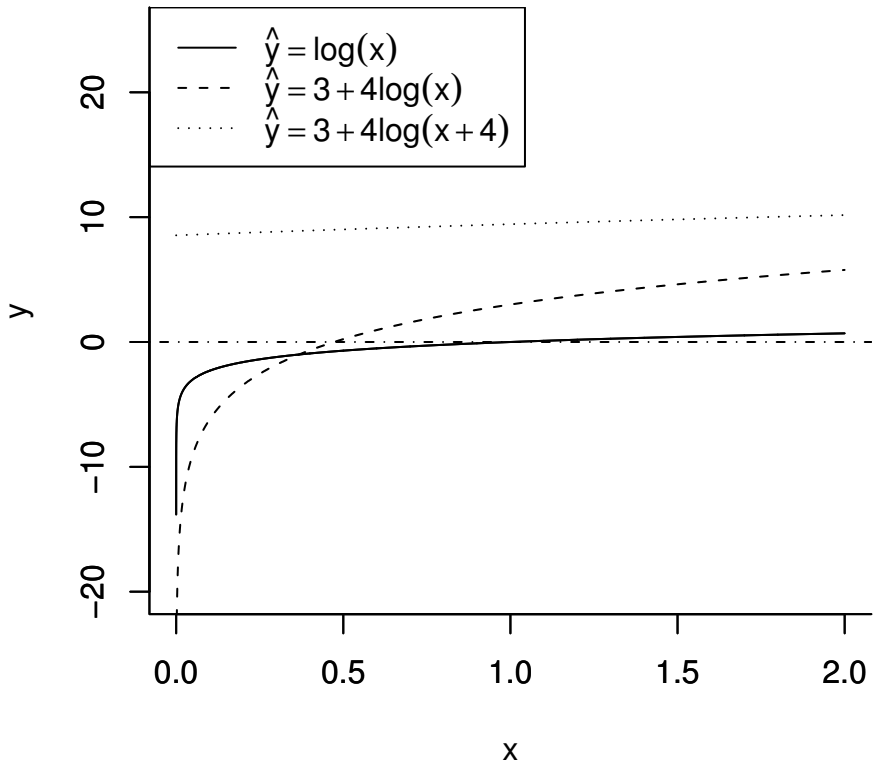
It is always true, no matter what the *base* is, that

$$log_{base}(1) = 0.$$

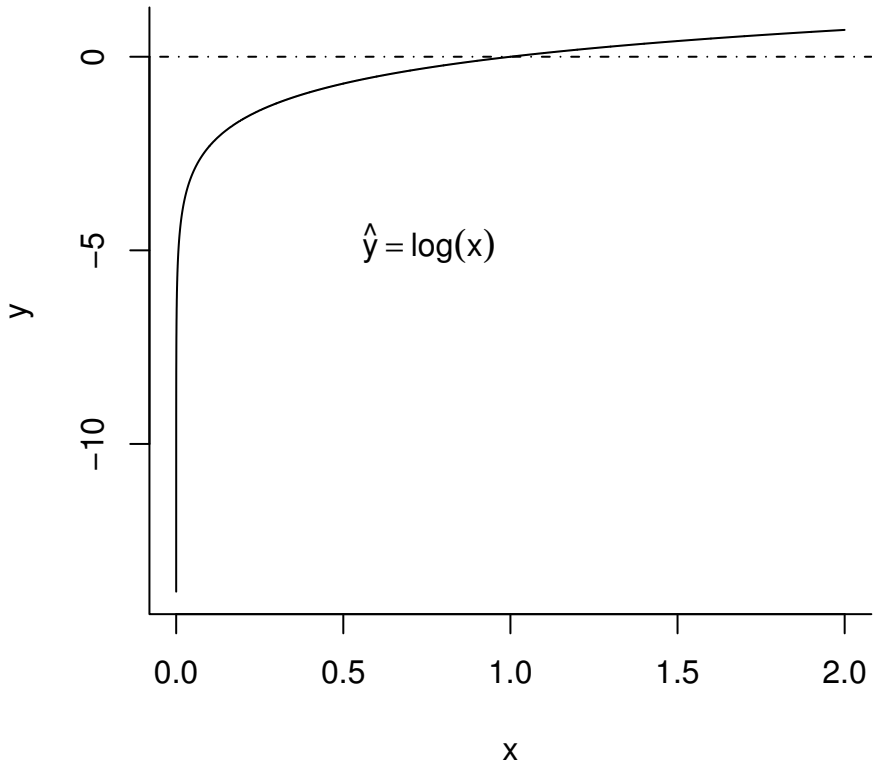
You can add an intercept and slope:

$$\hat{y}_i = b_0 + b_1 log(x_i) \tag{11.25}$$

Draw a log, and then imagine the effect of adding b_0 and multipling by b_1:



The log function has "a lot of curvature" only when x is very small.

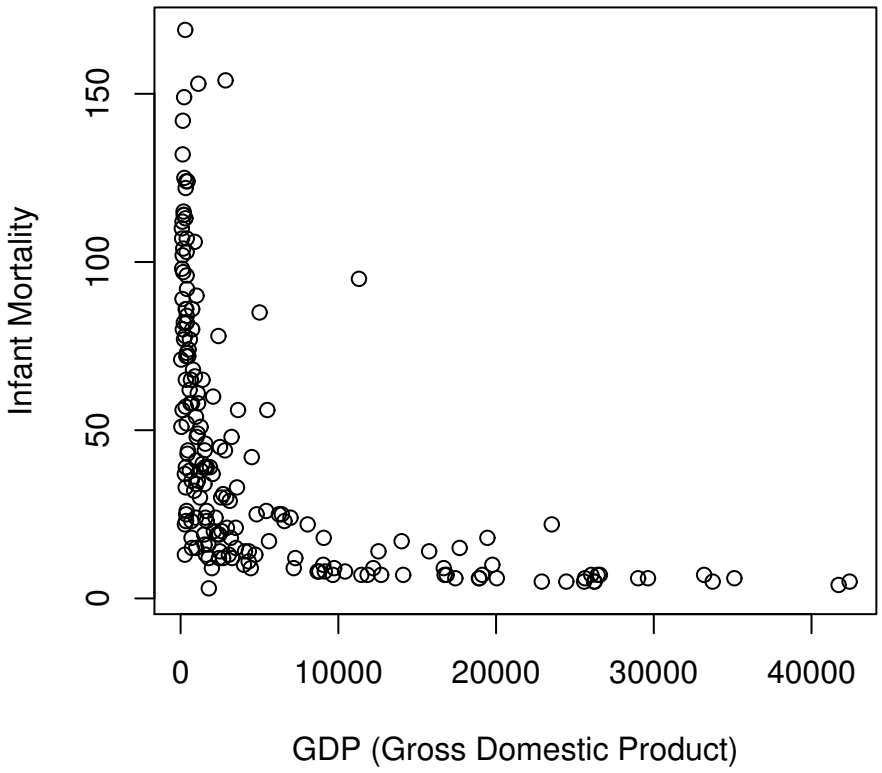


So it makes sense that when you the log plays an important role, it is either

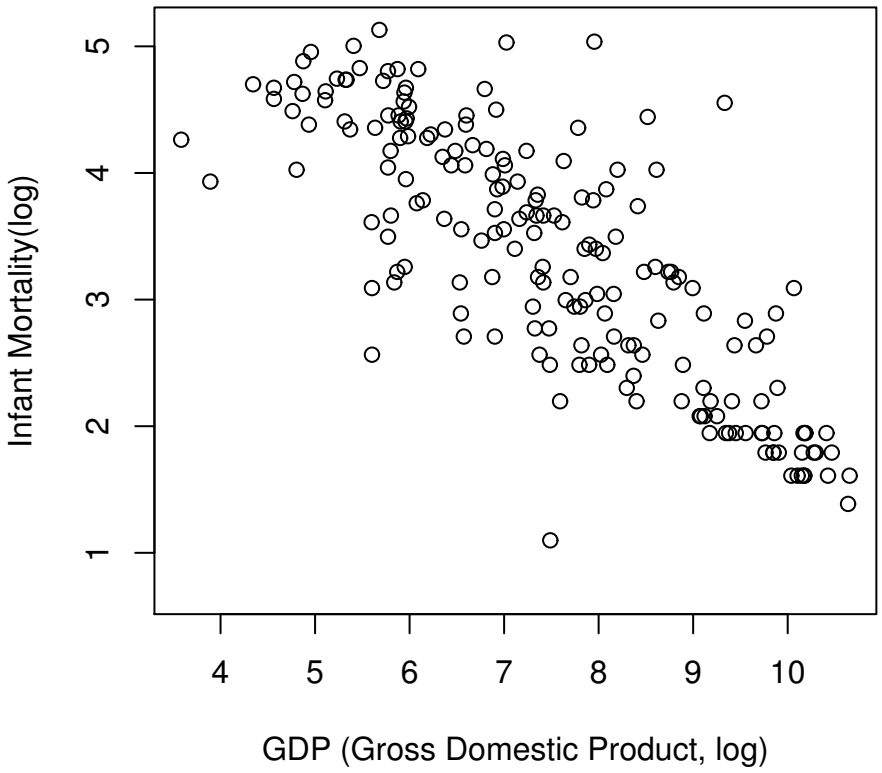
1. differentiating small values and "pulling them apart" or

2. eliminating differences among large values by "squishing them together."

To illustrate the impact of the logarithm, consider a classic example of a scatterplot based on "skewed" data. The data frame UN from the car package has information on infant mortality and gross domestic product (an indicator of wealth). It is an unfortunate fact of life that there are more poor countries than rich countries. This is a scatterplot that sets off all kinds of alarm bells:



The logarithm has a way of pulling the extreme cases toward the center, so the low values don't appear quite so low and the highs are not quite so high. A linear prediction would be much more meaningful with that data.



A clever student quickly responds, "but how do I interpret those logarithm values." There is plenty to say about this, but at this point, I only want to respond "translate your predictions back into the original units." If the predicted value of logged infant mortality for a country is 3.3, then the predicted infant mortality is

$$exp(3.3) = e^{3.3} = 27.11$$

That is to say, after "anti logging" the logged values, we retrieve a prediction that is meaningful in the original units. I have not yet explained the meaning of *exp* and the sense in which it "undoes" the logarithm, but that is a major point in a later section.

11.3.3 Laws of Logs.

Logs have some weird properties. This is called the "product rule."

$$log(x_i \times z_i) = log(x_i) + log(z_i) \tag{11.26}$$

That is true for as many objects in parentheses as you want.

$$log(x_1 \times x_2 \times \ldots \times x_N) = log(x_1) + log(x_2) + \ldots + log(x_N) \tag{11.27}$$

And this is also true:

$$log(x_i^{\alpha}) = \alpha \cdot log(x_i) \tag{11.28}$$

Dig this

$$log(\alpha/x_i) = log(\alpha) - log(x_i) \tag{11.29}$$

Note, you should be able to see that these last two things are not really different from each other. If you remember that

$$x_i^{-1} = \frac{1}{x_i}$$

you will see that 11.29 and 11.28 are the same thing.

11.3.4 Log knowledge has long term benefit to students

These findings are very important in "maximum likelihood estimation." We often end up with the idea of finding the best value of an estimate that maximizes the prodcut of a lot of numbers multiplied together. If somebody gives you a big long string of things multiplied together, you can simplify it by taking the logarithm.

Recall:

$$\Pi_{i=1}^{N} x_i = x_1 \times x_2 \times \ldots \times x_N$$

and

$$\Pi_{i=1}^{N} f(x_i) = f(x_1) \times f(x_2) \times \ldots \times f(x_N)$$

Using the "product rule" for logarithms:

$$log(\Pi_{i=1}^{N} f(x_i)) = \sum_{i=1}^{N} log(f(x_i)) \tag{11.30}$$

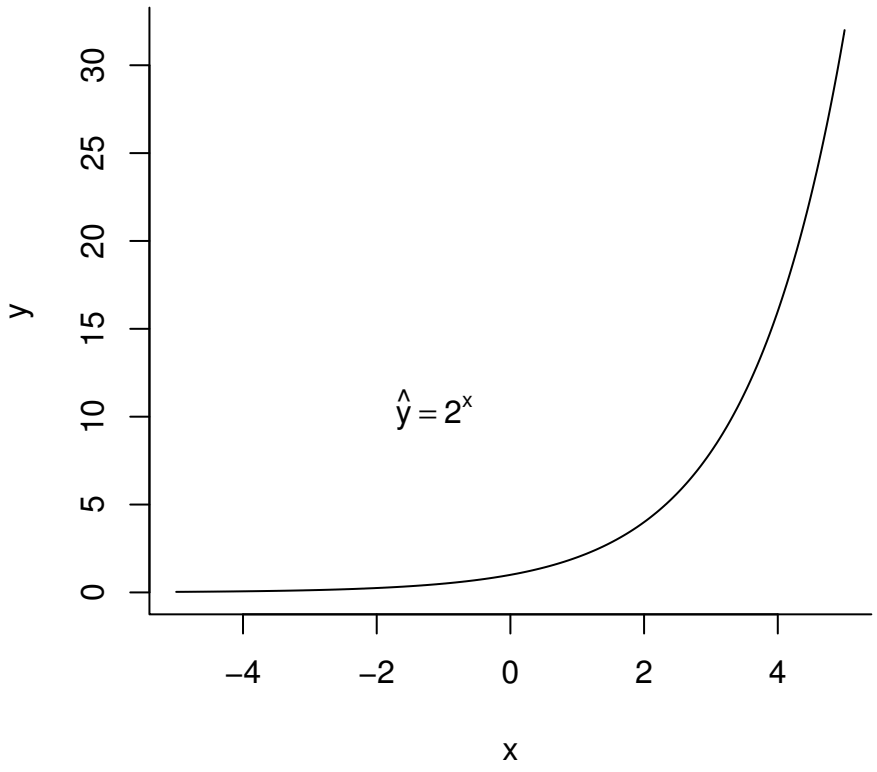
In words, "the log of the product is the sum of the logs." The sum is *much (much)* simpler for mathematical work. In a product, any change in any term has a multiplicative ripple effect. In a sum, the effect of a change in one value is simply added to the total.

11.4 Exponential

Do you want a relationship that goes up faster and faster?

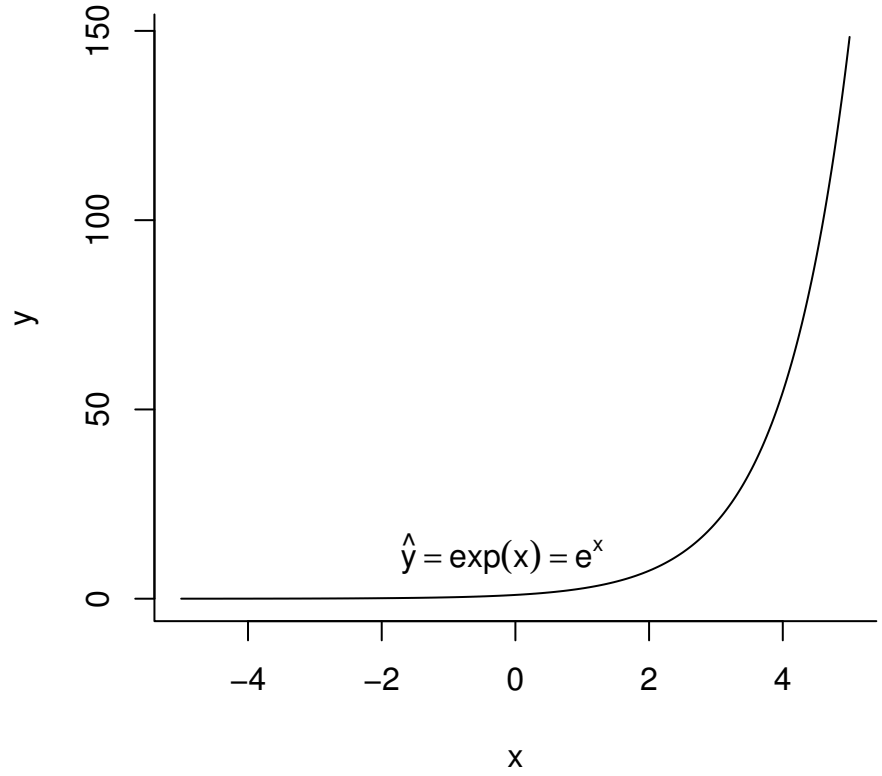
Graph this:

$$\hat{y}_i = 2^{x_i} \tag{11.31}$$



Now, imagine that instead of 2, I put in the base of the natural log:

$$\hat{y}_i = e^{x_i} \tag{11.32}$$



This is sometimes represented

$$\hat{y}_i = exp(x_i) \tag{11.33}$$

The *log* and *exp* perform the opposite effect. If a variable is logged, then exponentiating it "undoes" the logarithm. That is to say,

$$exp(ln(x_i)) = x_i \tag{11.34}$$

Similarly, the log "undoes" the exponential.

$$ln(exp(x_i)) = x_i$$

Note if you take the log of both sides of equation 11.33:

$$ln(\hat{y}_i) = ln(exp(x_i)) \tag{11.35}$$

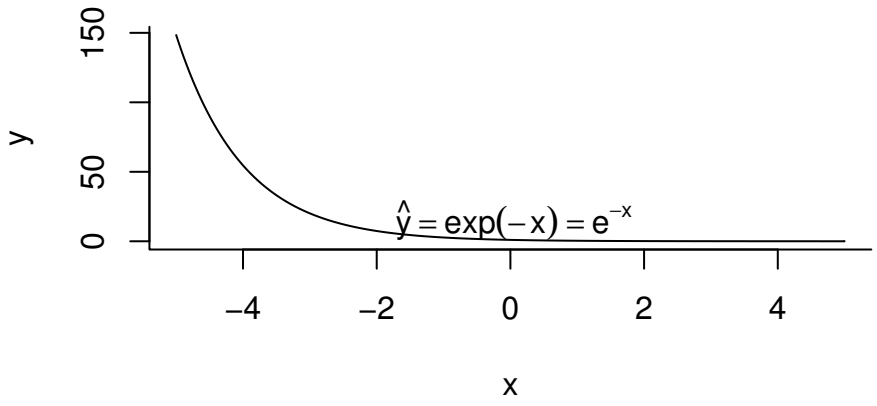
$$ln(\hat{y}_i) = x_i \tag{11.36}$$

This is a "log-linear" equation. The log of y is linear in x.

One of the most important uses of an exponential relationship is in the conversion of scores from negative to positive numbers. Note in the graph that no matter whether we put in a positive number or a negative number, the exponential always returns a positive value. There are many models where we use $exp()$ as an intermediate step. If we have an input z that might take on a range of values from negative to positive, then we might apply $exp(z)$ in order to assure that we only have positive values, and then a final calculation, which allows only positive inputs, can be applied.

Another important use of the exponential function is in the description of probabilities. A probability value must be greater than or equal to 0, and so exp is an obvious way to assure that the calculation ends up with a positive value.

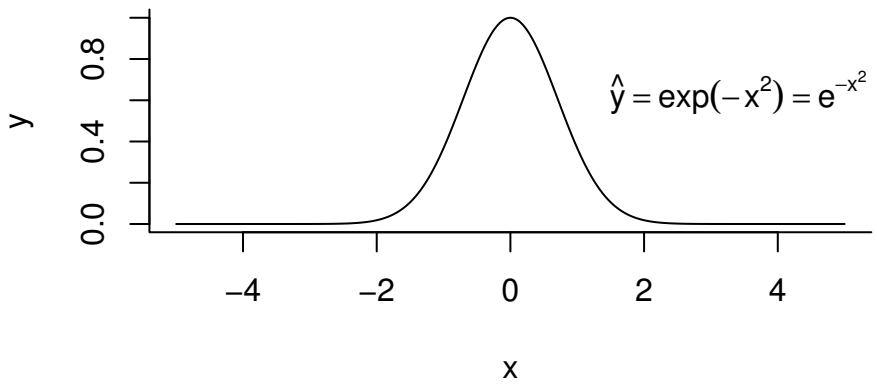
Without going into too much detail about probability theory, I believe I can still demonstrate an interesting use of the exponential function. In the first example, consider the graph of $exp(-x)$, which is shaped like a "ski jump".



The highest value is observed on the far left, and the output value shrinks as x_i increases. The predicted value of y_i is decreasing all across the spectrum of x_i.

As a contrast, consider the relationship $exp(-x^2)$. This is a summetric graph that is unimodal.

Figure 11.1: Sneaking in the Foundation of a Normal Distribution

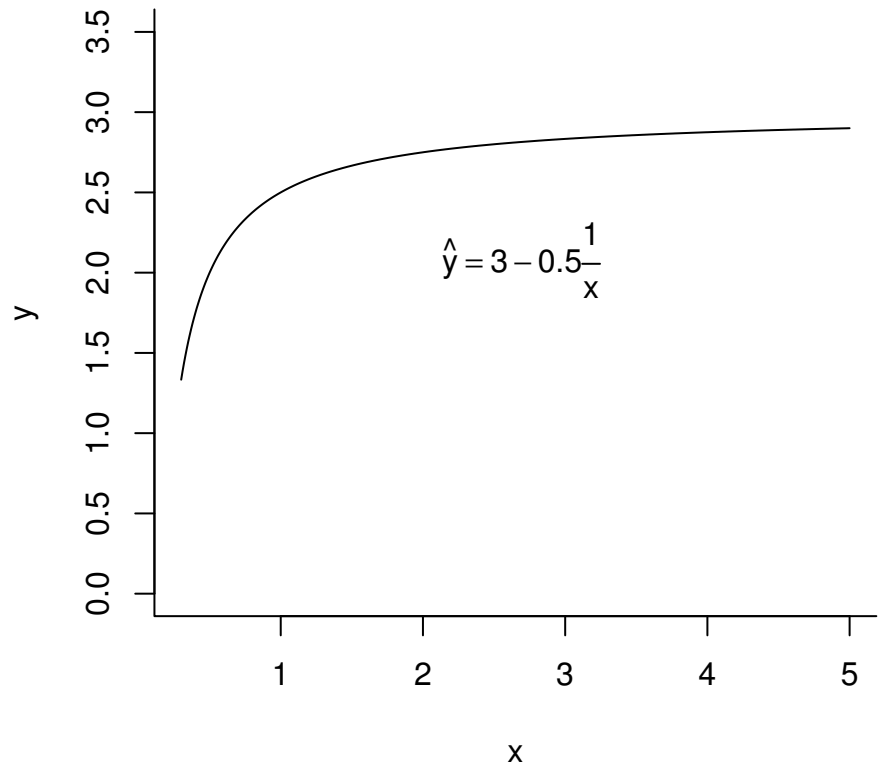


11.5 Reciprocal

Suppose

$$\hat{y}_i = b_0 + b_1 \times frac(1/x_i)$$

Graph this for $b_0 = 3$ and $b_1 = -0.5$ for $x_i > 1$.



11.6 Exercises: We love straight lines

I've run all of this code and am sure it does run, but that doesn't mean there aren't other mistakes. If you think you see a typographical error, let me know.

Assignment. Make a LyX document that includes the answers to the following exercises.

1. This R code will plot 3 lines. Their constants are included in the vector a and the slopes are in the vector b.

```
a <- c( -1, 4, 14)
b <- c( 2, 0.22, -0.8)
x <- seq(from=0, to=50, length.out=200)
y1 <- a[1] + b[1]*x
y2 <- a[2] + b[2]*x
y3 <- a[3] + b[3]*x
newy <- cbind(y1,y2,y3)
```

```
matplot (x, newy, main="LinearEquations", type="l",xlim=c(0,50))
```

Run this once the way it is, and then change those numbers and run again to customize it to fit the story you want to tell about these 3 lines.

Now, about your story. Imagine 3 political units (countries, groups, cities, whatever). There are 2 variables that are linked, but the linkages are different in the 3 cities.

a) Write one paragraph about your story. Be sure to use an equation in the beautiful LyX way to show what you are talking about.

b) The matplot output should be customized for the story you tell. Make a pleasant looking graph: Insert labels in your graphs to replace "x" and "y", insert nicer title. If you are able to, label the 3 lines (I showed you how to do that on the first day we went to the lab). Save that to an EPS file.

Don't forget to use LyX to insert the floating figure with the graphic inside. If you insert a label with your figure, you can use the LyX "cross reference" feature.

2. Here's more code that makes a figure with 2 line segments.

```
a<- 3
b <- 1.2
c <- -2
d <- 2
x <- seq(from=0, to=50, length.out=200)
z <- ifelse(x < 30, 0, 1)
y <- a + b*x + c*z + d*x*z
plot (x, y, main="LinearEquation: With Breaks?",
type="p",pch=2,xlim=c(0,50))
```

a) Customize the coefficients and the graph to fit some idea you might have.

b) Write a one paragraph story about it. Be sure to include a nice looking LyX equation, such as

$$Nasa\,Funding = a + b \times GNP + (c + d \times GNP) \times Democrats$$

i. or

ii.

$$poverty = \beta_0 + \beta_1 Employment + (\beta_2 + \beta_3 Employment) \cdot Democrats$$

c) In this code, the variable z is a "dummy variable." (In the example equations above, it's role is played by "*Democrats*". Can you describe its role in this relationship?

d) In what sense is it fair to say that the coefficient c is an "intercept shifter" and d is a "slope shifter"?

3. The following code will make a similar, but different figure. Customize it however you think appropriate, output to EPS, then include in your document. Write one paragraph and include an equation. Hint: the equation governing this process is something like

$$y = a_1 + b_1 x + (c_2 + d_2 x) \cdot z_1 + (c_3 + d_3 x) \cdot z_2$$

```
a <- c( -1, 4, 14)
b <- c( 2, 0.22, -0.8)
x <- seq(from=0, to=50, length.out=200)
y <- ifelse(x < 10, a[1] + b[1]*x, ifelse(x<24,a[2]+b[2]*x,a[3]+b[3]*x))
plot (x, y, main="LinearEquation: With Breaks?",type="p",pch=2,xlim=c(0,50))
```

12 Truly Useful Information about Matrix Algebra and Vectors

This is an effort to provide the linear algebra and matrix algebra terminology that is truly necessary for a social science graduate students who hope to become active empirical researchers. The focus is on conveying the most vital concepts. There is, of course, no substitute for one or more formal, rigorous courses in a mathematics department.

This is the first publicly shared draft. One will notice that examples are more plentiful in the beginning than the end, and also that exercises are not inserted in most places. Those enhancements may come someday.

Caution: This essay will not provide a sufficient background for people who intend to 1) do formal rational choice theory, especially the "spatial model" of elections, or 2) develop software for estimation of parameters. However, if people think they might like to become modelers of various types, then this might be the right place to start. The main aim is to prepare people to read articles that use sophisticated methods and terminology and possibly to conduct some sophisticated analysis after a suitable amount of coursework is completed.

12.1 What's it all about?

12.1.1 Bookkeeping

Go to the grocery store and pick up 2 pounds of hamburger, 1 bag of buns, 3 cans of beans, and one-half gallon of milk. Write that succinctly as:

$$(2, 1, 3, 0.5) \tag{12.1}$$

If a vegan goes to the store, he might need to get

$$(0, 1, 5, 0) \tag{12.2}$$

Your uncle might be a meat-a-tarian, so he gets

$$(5, 0, 0, 0) \tag{12.3}$$

As long as you remember which item is first, which is second, and so forth, this "vector" of numbers is a much more succinct way to write down your shopping list.

It may not be necessary to keep a succinct list if one is shopping only for one family. What if one goes to an apartment complex where senior citizens live and collects grocery orders from 100 families? There is at least a chance that we will fill the orders correctly if we number the families and stack up their orders into a matrix like so:

$$\begin{bmatrix} 2 & 1 & 3 & 0.5 \\ 0 & 1 & 5 & 0 \\ 5 & 0 & 0 & 0 \\ & \cdots & \cdots & \\ & \cdots & \cdots & \end{bmatrix} \tag{12.4}$$

Of course, we are social scientists, so we don't usually have to spend much time collecting orders for groceries. (Like most professors, I have a staff graduate assistants that handles those mundane chores for me, actually.) But we do need to keep track of records from surveys or other sorts of data collection exercises. It is customary to keep the information gathered from a single respondent in a row with separate columns that represent the information. Some of the large surveys that are administered by the University of Michigan's Survery Research Center or the University of Chicago's National Opinion Research Center will collect hundreds–or thousands–of columns of information. It is hard to conceive of a method for tracking that information that does not use structured rows and columns.

Along these same lines, we also use vectors and matrices to describe our theories of human behavior. Today's political scientist inherits about 50 years of experience with mathematical models of decision-making. One leading approach to choice is found in the "spatial model," which is based on the idea that we choose among bundles of attributes. Suppose the members of a city commission argue about how the winter weather budget ought to be allocated. One member believes the smartest approach is to buy extra salt for the roads, while another thinks that salt is not as important as sand. Another person may favor spending the money on a new snow plow. And another would rather hire workers with shovels to clean the sidewalk. A person's preferences are represented by an ideal point vector, a set of values for the most desirable allocation of resources.

We might gather data and try to estimate a person named Fred's most preferred allocation. We often call these ideal points, which we might write that down as:

$$x^*_{Fred} = [10000, 50000, 100000, 2000] \tag{12.5}$$

for the dollar costs of salt, sand, trucks, and shovels.

$$x^*_{Fred} = [x1^*_{Fred}, x2^*_{Fred}, x3^*_{Fred}, x4^*_{Fred}] \tag{12.6}$$

We could collect up the preferences of the city commission members in a vector,

$$[x^*_{Fred}, x^*_{Bev}, x^*_{Jane}, x^*_{Bob}, x^*_{Ron}] \tag{12.7}$$

12.1.2 Make calculations in a systematic way

Sometimes we don't actually "need" matrices and vectors. Sometimes, we can find other ways to describe problems and solve them. On the other hand, it will often be easier and less error-prone to use matrices and vectors.

Consider the problem of finding the point at which two lines intersect. In Figure 12.1 there are two lines. If one wanted to find the intersection, one could get a very good ruler and a magnifying glass. The point where the two lines cross is at the intersection of the two dotted lines. That point is a vector, $(3.5, -0.93)$.

A ruler does not offer the most precise solution. There are better ways to find the exact value of the coordinates for the intersection. The equations for these two lines are

$$x_2 = 8 - 2.5x_1 \tag{12.8}$$

and

$$x_2 = -7 + 1.7x_1 \tag{12.9}$$

The following strategy for solving this problem might be characterized as the mathematical equivalent of typing by "hunting and pecking." I learned this as a junior in high school. Solve one equation for x_1, and then insert that solution into the other equation. This approach relies on the fundamental algebraic rule that an equation remains valid if all of its terms are multiplied by a constant. To solve this example, take the first equation and divide all terms on both sides by 2.5

$$\frac{1}{2.5}x_2 = \frac{8}{2.5} - \frac{2.5}{2.5}x_1 \tag{12.10}$$

$$\frac{1}{2.5}x_2 = \frac{8}{2.5} - x_1 \tag{12.11}$$

We can also add or subtract something from both sides and maintain the equality. We use that principle to re-arrange thus:

$$x_1 = \frac{8}{2.5} - \frac{1}{2.5}x_2 \tag{12.12}$$

Insert that expression for x_1 into the equation 12.9

$$x_2 = -7 + 1.7(\frac{8}{2.5} - \frac{1}{2.5}x_2) \tag{12.13}$$

After a few algebraic re-arrangements[1], we find

$$x_2 = -0.9285 \tag{12.14}$$

Insert that value for x_2 into either equation and obtain $x_1 = 3.5714$.

We may find answers that are close enough with the ruler or by "hunting and pecking." Neither approach is optimal, however, for two reasons. First, we would like to learn from experience. If we have to solve many problems of this type, we should find a way to standardize the solution process. Second, it is often the case that the problems we need to solve involve many more variables. A system of equations that has 8 or 10 variables, all of which must be adjusted simultaneously to bring the equations into harmony, will not yield its answer so easily. We can use the notation of matrix algebra to represent the system more abstractly in order to allow the development of a systematic solution strategy.

12.2 Elementary notation and concepts

scalar. That's a number. Either an integer like 1 or a real number like 3.2.

vector. That's a column of numbers. If somebody says x is a vector, you are supposed to think "aha, a column of numbers."

Example 1. A vector might be most immediately recognizable to students because it is like a column of numbers in a spreadsheet. It might represent scores collected by observing the weights of monkeys in Madagascar or the heights of fourth grade students in Peoria, Illinois. (One is often struck by the apparent interchangeability of monkeys and fourth graders.)

Here is a vector that might represent the grade point averages of 7 students:

[1]

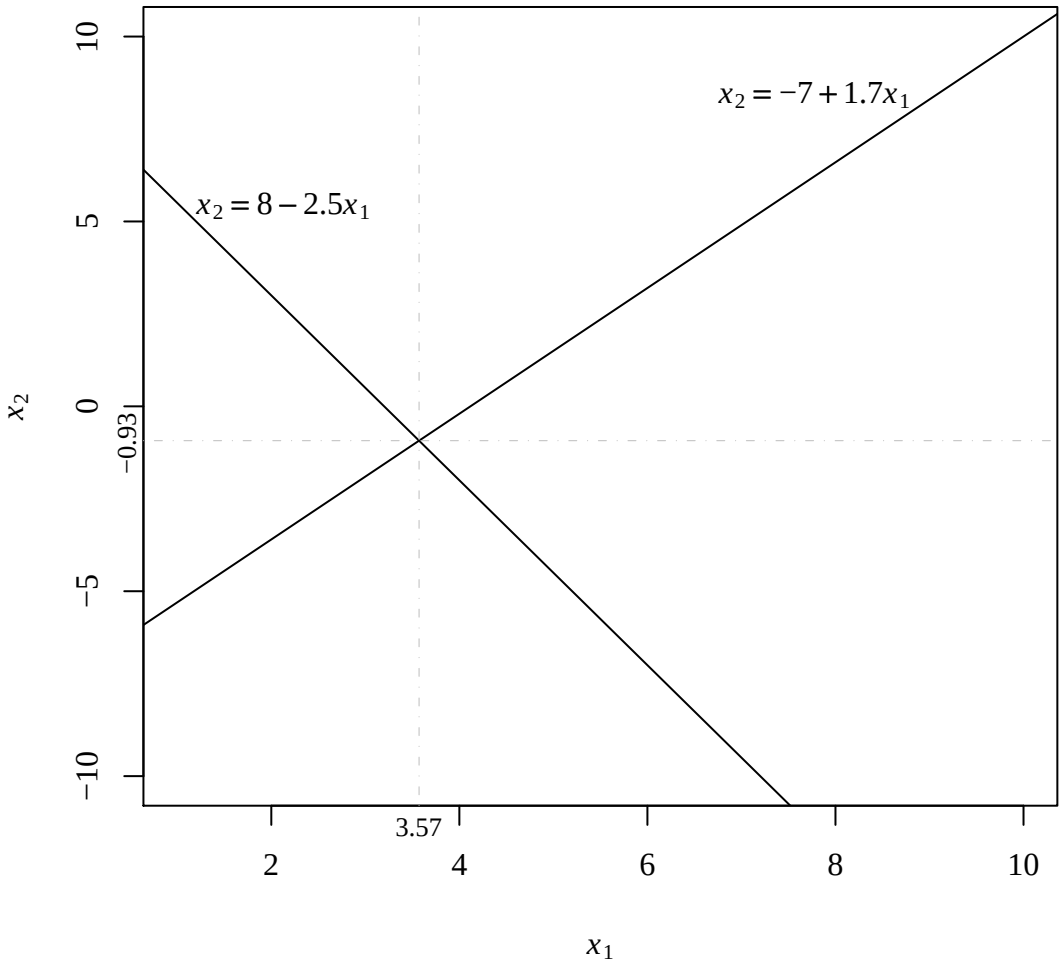
$$x_2 = -7 + \frac{1.7 \times 8}{2.5} - \frac{1.7}{2.5}x_2$$

$$x_2 + \frac{1.7}{2.5}x_2 = -7 + \frac{1.7 \times 8}{2.5}$$

$$\frac{4.2}{2.5}x_2 = -7 + \frac{1.7 \times 8}{2.5}$$

$$y_i = \frac{-7 \times 2.5}{4.2} + \frac{1.7 \times 8 \times 2.5}{4.2 \times 2.5}$$

Figure 12.1: Two Lines Cross



$$gpa = \begin{bmatrix} 3.3 \\ 4.0 \\ 3.3 \\ 3.7 \\ 2.2 \\ 1.1 \\ 2.2 \end{bmatrix} \tag{12.15}$$

Suppose those are grade point averages, for example. For people who have finished high school during the last 10 years or so,

We use a notation like y_j or $y[j]$ to refer to a particular j'th member of a vector y. The subscript or bracket notations appear to be equally acceptable, but within a particular article, one should try to be consistent. The first element, for example, is 3.3, and we could refer to it as in gpa_1 or $gpa[1]$. Ordinarily, in mathematics, the elements in a vector are numbered $\{1, 2, 3, \ldots, N\}$. In some computer languages one must be cautious because the first item in a vector is numbered 0 and the index of items will begin there, as in$\{0, 1, 2, \ldots, N-1\}$.

Example 2. A vector might represent a set of "unknowns" in a theory or a model. If we hypothesize that educational performance reflects a student's socio-economic status, the marital status of that student's parents, the quality of the teacher, and the quality of the school, then our theory might hold that that there are four effects that need to be taken into account. The estimates for those unknown effects might be written out as vectors, as in

$$\hat{\theta} = \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \\ \hat{\theta}_4 \end{bmatrix} \text{ or } \hat{\beta} = \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \\ \hat{\beta}_4 \end{bmatrix} \tag{12.16}$$

or, if you don't like symbols drawn from non-English languages, you can always make yourself feel at home with:

$$\hat{b} = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \\ \hat{b}_4 \end{bmatrix} \tag{12.17}$$

row vector. A row of numbers is a "row vector" (duh!).

transpose. To "transpose" a column vector, turn it on its side. This converts a column vector into a row vector, or vice versa.

Sometimes it is represented by a prime symbol, as in y', or sometimes as y^T. I think T is easier to remember, but many publications seem to prefer the apostrophe. So

$$y^T = \begin{bmatrix} 3.3 & 4.0 & 3.3 & 3.7 & 2.2 & 1.1 & 2.2 \end{bmatrix} \tag{12.18}$$

In computer programs, the transpose is thought of as a function, as in $t(y)$.

matrix. A matrix is a rectangular array of numbers.

In this example, X is a matrix that has 5 rows and 8 columns.

$$X = \begin{bmatrix} 1 & 2 & 44 & 6 & 1 & 8 & 1 & 1 \\ 1 & 1 & 33 & 4 & 1 & 7 & 0 & 1 \\ 1 & 1 & 21 & 6 & 2 & 8 & 1 & 1 \\ 1 & 0 & 43 & 2 & 1 & 7 & 0 & 2 \\ 1 & 1 & 66 & 1 & 4 & 7 & 1 & 2 \end{bmatrix} \tag{12.19}$$

This is a 5 x 8 matrix–5 rows, 8 columns.

More generally, we use the notation $m \times n$ to refer to rows and columns in matrices. Let m be the number of rows, and let n be the number of columns. Then we have an $m \times n$ matrix. It is important to remember "rows, then columns" when using matrices. I remember this according to a trick mentioned in Russell Hardin's book Collective Action. In case you forget whether the columns come first or second, remember the mnemonic "Roman Catholic." RC stands for "Row, then Column."

The element in the second row, third column, could be referred to as X_{23} or $X_{2,3}$ or $X[2,3]$. In some books, the author will refer to the particular element as $[x_{23}]$, where the lower case x is supposed to remind one of the fact that

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} \end{bmatrix} \tag{12.20}$$

If I were writing "the book," I suppose I would refer to the individual points as X_{23} because then I would not need to write (distracting) brackets around my symbols. But there is something to be said in favor of the notation $[x_{23}]$ because it reminds the reader that the element is drawn from a larger array of numbers.

It seems to me that there are many different acceptable conventions, and readers are expected to "stay on their toes."

If someone wrote $X_{2,.}$, they probably mean to refer to the second row. Similarly, $X_{.,5}$ refers to the fifth column.

You can think of a matrix as either a collection of m rows or n columns. See?

matrix transpose. Matrices can be transposed, just as vectors can be. A matrix is transposed by "rotating" the matrix in the same direction that we rotated the vector to transpose it. The first column becomes the first row, the second column becomes the second row, and so forth.

A picture is worth a thousand words in this context. If

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{bmatrix} \tag{12.21}$$

then the transpose is

$$X' = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 \\ 2 & 4 & 6 & 8 & 10 \end{bmatrix} \tag{12.22}$$

As in the case of vectors, the matrix transpose might be labeled X', X^T, X^t or even $t(X)$.

More formally, a transpose of an $(m \times n)$ matrix with elements $[x_{ij}]$ is the $(n \times m)$ matrix with elements $[x_{ji}]$. That statement is needed for completeness.

Example 1. A matrix represents a "data set," meaning a collection of columns, one for each variable measured on a set of subjects.

$$Students = \begin{bmatrix} 1 & 6 & 12 & 3.2 & 500 \\ 1 & 5 & 12 & 2.8 & 888 \\ 1 & 7 & 9 & 2.4 & 777 \\ 1 & 8 & 15 & 4.0 & 575 \\ 1 & 7 & 17 & 2.2 & 432 \\ 1 & 8 & 13 & 1.9 & 1199 \end{bmatrix} \tag{12.23}$$

A line of data is a "record", a collection of scores for an individual observation. The columns are "variables". For example, the columns might represent

1. A score of "1" is assigned if a record was completed.
2. A socio-economic status indicator on a 10 point scale.
3. The highest year of school finished by the student's father.
4. The teacher's grade point average in undergraduate school.
5. The number of dollars per year spent in the school that is attended by the student.

It is equally true to say that the 5 columns representing variables are "put together" to make a matrix, or that the rows representing the students are "stacked together" to create a matrix.

12.3 You can add, but you can't divide

Let me hit you with this right off. The world of matrices and vectors has its own set of rules, and some of the things you expect are just flat-out wrong.

For example, it does not make sense to try to do $\sqrt{X}$ or b/X. Those are "undefined" operations.

12.3.1 You can add, however.

Addition of 2 vectors requires that they have the same length. You just add the individual elements, one-by-one:

Suppose

$$y = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 3 \end{bmatrix}$$

and

$$x = \begin{bmatrix} 6 \\ 1 \\ 2 \\ 3 \\ 0 \end{bmatrix}$$

then the sum is:

$$y + x = \begin{bmatrix} 7 \\ 2 \\ 4 \\ 6 \\ 3 \end{bmatrix} \tag{12.24}$$

It is the same with matrices. If they have the same number of rows and columns, you can add their elements one by one to get the sum. Oh, gosh, I don't want to type all that in. But here's a little example:

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$Y = \begin{bmatrix} 3 & 5 \\ 6 & 99 \end{bmatrix}$$

$$X + Y = \begin{bmatrix} 4 & 7 \\ 9 & 103 \end{bmatrix} \tag{12.25}$$

Please remember. Matrix addition is not defined if the two matrices do not have exactly the same number of rows and the same number of columns.

From this, it should be easy for you to see how to add 2 row vectors, right? (Hint: a row vector is a (1xn) matrix!

12.3.2 Matrices and Vectors are scalable, in a sense.

You can multiply vectors and matrices by scalars.

If you have a scalar, say 2.1, then it is easy to multiply it with $x' = [2, 1, 4]$. Simply multiply 2.1 against each individual element.

$$2.1 \times \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 4.2 \\ 2.1 \\ 6.3 \end{bmatrix}$$

This is so pathetically easy, I hate to belabor it.

Example 1. I can only think of one instance in which this fact has an important substantive meaning. We start talking about covariance matrices, we might wonder if several variables share a given level of variance. The idea that there might be a common amount of variance–a homogeneous set of distributions–across observations is important in some statistical models. In such a situation, then we might see authors write either of the following.

$$\sigma^2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sigma^2 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 \\ 0 & 0 & 0 & \sigma^2 \end{bmatrix} \tag{12.26}$$

12.4 More Complicated Multiplication.

You can only multiply vectors or matrices if they "conform" (meaning "match up with the correct number of elements") against each other.

12.4.1 Multiplying Vectors

The first vector must have the same number of columns as the second one has rows.

Example 1. Multiply a (1×5) vector and a (5×1) vector.

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{bmatrix} = 1 \cdot 6 + 2 \cdot 7 + 3 \cdot 8 + 4 \cdot 9 + 10 \cdot 5 = 130 \tag{12.27}$$

To describe this in words: "Take the second vector and turn it on its side and multiply its elements one-by-one against the elements in the first vector. Then sum the resulting numbers."

Observe: $(1 \times 5) \cdot (5 \times 1)$ gives a result that is (1×1). So the 5's on the "inside" of $(1 \times 5) \cdot (5 \times 1)$ must be equal in order to multiply, and the result has a size equal to the values of the outside numbers.

12.4.2 Multiplying Matrices

Multiplication of matrices follows the same principle as multiplication of vectors. Because there are more numbers involved, one may become confused, but the idea is just as simple. The columns in the second matrix are taken one-by-one, "turned on their sides," and multiplied by the rows in the first matrix. This is particularly easy to see in a case where the second matrix has only one column.

Example 1. Multiply a matrix with 3 rows and 5 columns by a vector with 5 rows and 1 column.

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{bmatrix} \begin{bmatrix} 21 \\ 22 \\ 23 \\ 24 \\ 25 \end{bmatrix} = \begin{bmatrix} 21 \cdot 1 & +22 \cdot 2 & +23 \cdot 3 & +24 \cdot 4 & +25 \cdot 5 \\ 21 \cdot 6 & +22 \cdot 7 & +23 \cdot 8 & +24 \cdot 9 & +25 \cdot 10 \\ 21 \cdot 11 & +22 \cdot 12 & +23 \cdot 13 & +24 \cdot 14 & +25 \cdot 15 \end{bmatrix} \tag{12.28}$$

$$= \begin{bmatrix} 345 \\ 930 \\ 1505 \end{bmatrix} \tag{12.29}$$

Multiply each column of the matrix on the right against each row of the matrix on the left.

Note that's multiplying (3×5) against $(5x1)$. Like vectors, matrices must conform. The number of columns in the first matrix must equal the number of rows in the second. In this case, the 5's "match up" on the inside, so the multiplication is possible. And, you were paying attention in the previous section, you should expect that the result will be (3×1).

Example 2. Multiply a (3×5) matrix by a (5×2)matrix.

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{bmatrix} \begin{bmatrix} 21 & 26 \\ 22 & 27 \\ 23 & 28 \\ 24 & 29 \\ 25 & 30 \end{bmatrix} = \tag{12.30}$$

$$= \begin{bmatrix} 21 \cdot 1 & +22 \cdot 2 & +23 \cdot 3 & +24 \cdot 4 & +25 \cdot 5 & 26 \cdot 1 + 27 \cdot 2 + 28 \cdot 3 + 29 \cdot 4 + 30 \cdot 5 \\ 21 \cdot 6 & +22 \cdot 7 & +23 \cdot 8 & +24 \cdot 9 & +25 \cdot 10 & 26 \cdot 6 + 27 \cdot 7 + 28 \cdot 8 + 29 \cdot 9 + 30 \cdot 10 \\ 21 \cdot 11 & +22 \cdot 12 & +23 \cdot 13 & +24 \cdot 14 & +25 \cdot 15 & 26 \cdot 11 + 27 \cdot 12 + 28 \cdot 13 + 29 \cdot 14 + 30 \cdot 15 \end{bmatrix} \tag{12.31}$$

$$= \begin{bmatrix} 345 & 430 \\ 930 & 1130 \\ 1505 & 1830 \end{bmatrix} \tag{12.32}$$

You end up with as many rows as the matrix on the left and as many columns as the matrix on the right. Generally, to multiply an $(m_1 \times n_1)$ matrix against an $(m_2 \times n_2)$ matrix, then it is required that $n_1 = m_2$. And the result will be $(m_1 \times n_2)$.

Example 2. Multiply a (5×1) matrix (a column vector) and a (1×5) matrix (a row vector).

$$\begin{bmatrix} 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix} = \begin{bmatrix} 6 & 2 \cdot 6 & 3 \cdot 6 & 4 \cdot 6 & 5 \cdot 6 \\ 7 & 2 \cdot 7 & 3 \cdot 7 & 4 \cdot 7 & 5 \cdot 7 \\ 8 & 2 \cdot 8 & 3 \cdot 8 & 4 \cdot 8 & 5 \cdot 8 \\ 9 & 2 \cdot 9 & 3 \cdot 9 & 4 \cdot 9 & 5 \cdot 9 \\ 10 & 2 \cdot 10 & 3 \cdot 10 & 4 \cdot 10 & 5 \cdot 10 \end{bmatrix} = \begin{bmatrix} 6 & 12 & 18 & 24 & 30 \\ 7 & 14 & 21 & 28 & 35 \\ 8 & 16 & 24 & 32 & 40 \\ 9 & 18 & 27 & 36 & 45 \\ 10 & 20 & 30 & 40 & 50 \end{bmatrix} \tag{12.33}$$

Note the "insides" match up here, 1 and 1, and the result is 5×5.

It is important to note the obvious here. Multiplying $x' \cdot y$ generally gives a completely different answer than multiplying $y \cdot x'$.

Exercise: Care to test your understanding by filling in the blanks?

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} a+5b+9c & - & - & - \\ - & 2d+6e+10f & 3d+7e+11f & 4d+8e+12f \\ - & - & 3g+7h+11i & 4g+8h+12i \end{bmatrix} \tag{12.34}$$

12.5 Actually Useful Applications

I said this manuscript would be useful and immediately applicable for social scientists, and, at this point, a reader might think I was lying. I'm sorry, I'm really trying.

12.5.1 Your Predictions are $X\hat{b}$

You may have seen a journal article that included a regression model. A regression is a model in which one or more variables, called "predictors" (or "inputs" or "explanatory variables" or "independent variables") are used to calculate a "predicted value" for an output. Usually, we start with a simple model, one in which we hypothesize that the output is equal to a simple expression like

$$\widehat{gpa} = 2 + 0.1 \cdot hours\, spend\, studying \tag{12.35}$$

The idea here is that students who don't study at all are expected to have a *gpa* of 2, while hours spent studying are expected to raise the observed *gpa*.

In this example, there are two hypothesized values that are combined with data to make a prediction. The first hypothesized value is 2, which is a baseline or "constant" value, and 0.1, which represents the effect of studying. An additional hour of study time is hypothesized to raise the grade point average by one-tenth.

$$\hat{b} = \begin{bmatrix} 2 \\ 0.1 \end{bmatrix} \tag{12.36}$$

We think of the data as being organized like this, with a 1 for all respondents in the first column and the hours spent studying in the second column.

$$X = \begin{bmatrix} 1 & 10 \\ 1 & 8 \\ 1 & 14 \\ 1 & 22 \\ 1 & 4 \end{bmatrix} \tag{12.37}$$

Then the predicted grade point averages for the 5 students can be calculated by the matrix multiplication

$$X\hat{b} = \begin{bmatrix} 1 & 10 \\ 1 & 8 \\ 1 & 14 \\ 1 & 12 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2.8 \\ 3.4 \\ 3.2 \\ 2.4 \end{bmatrix} \tag{12.38}$$

I bet you you are thinking "it would have been faster to calculate those 5 *gpa*'s than to learn about multiplying matrices." That's true, but only if you never work with more than a few observations. I suppose that 14th century monks, the same ones that were charged with writing the Bible by hand over and over, might have relished a break afforded by some mathematical calculations. When asked to calculate some predicted grade point averages for 10,000 students, they would probably think it is light work. Begin with the first person being considered. Calculate

$\widehat{gpa}_1 = b_0 + \hat{b}_1 X_{12}$
Thank you, sir, may I have another:

$\widehat{gpa}_2 = b_0 + \hat{b}_1 X_{22}$
And another:

$\widehat{gpa}_3 = b_0 + \hat{b}_1 x_{32}$
Most of us would get tired of this after calculating 3,000 or so of these estimates.

Isn't it easier to write down

$$\hat{y} = X\hat{b} \tag{12.39}$$

and let a computer handle it? (This is when you think "yes, he's right.")

Just one more comment about notation. If an author is trying to describe the calculation of these predictions without matrices, he will usually say that the predictions are calculated as

$$\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i \; for \; i = 1, 2, 3, \ldots, N$$

To some editors, this is less intimidating than 12.39, but it is exactly the same thing. The only difference is that 12.39 is more succinct, and, generally, better (and more fun).

12.5.2 $X'X$ can be used to represent variance and covariance

The matrix product $X'X$ is very important in statistics, not only at the elementary level, but also in advanced levels. In elementary statistics, it is used to represent the components that can be used to calculate variance and correlation. Sometimes it is called the "matrix of sums of squares and cross-products." Recall the definition of variance (which I refer to as the "average of squared deviations"):

$$Var(x) = \frac{\sum_{i-1}^{N}(x_i - \bar{x})^2}{N} \quad . \tag{12.40}$$

In order to make this calculation, we would have to make one pass through the data to calculate the average ($\bar{x}$) and then we have to make another pass calculating the squared deviations. Statistical textbooks, especially the ones that were written during the days in which students would actually calculate variances "by hand," emphasize the fact that expression 12.40 can be algebraically re-arranged so that one needs to make only one pass through the data. If we calculate x_i and x_i^2 for each observation, then the variance can be calculated as

$$Var(x) = \frac{1}{N}\sum_{i=1}^{N} x_i^2 - \left(\frac{\sum x_i}{N}\right)^2 \tag{12.41}$$

In words, this says "the variance is equal to the mean of values squared minus the square of the mean." It is not difficult to convince oneself that this formula is correct. [2]

In matrix algebra, we are looking for ways to make representations more succinct. Begin by considering a couple of little details.

First, note that $x'x$ is simply the "sum of squares" for x,

[2] Note that $(x_i - \bar{x})^2 = (x_i - \bar{x}) \cdot (x_i - \bar{x}) = x_i^2 - 2x_i\bar{x} + \bar{x}^2$, so

$$Var(x) = \frac{\sum_{i=1}^{N}\left\{x_i^2 - 2x_i\bar{x} + \bar{x}^2\right\}}{N}.$$

Because the summation is a linear operator, it can be distributed over the individual terms:

$$= \frac{\sum_{i=1}^{N} x_i^2}{N} - \frac{\sum_{i=1}^{N} 2x_i\bar{x}}{N} + \frac{\sum_{i=1}^{N} \bar{x}^2}{N}.$$

It is always true that $\sum_{i=1}^{N} k \cdot x_i = k\sum_{i=1}^{N} x_i$ and $\sum_{i=1}^{N} \bar{x}^2 = N \cdot \bar{x}^2$,so the previous reduces to

$$= \frac{\sum_{i=1}^{N} x_i^2}{N} - \frac{2\bar{x}\sum_{i=1}^{N} x_i}{N} + \frac{N \cdot \bar{x}^2}{N}$$

$$= \frac{\sum_{i=1}^{N} x_i^2}{N} - 2\bar{x}\bar{x} + \bar{x}^2$$

$$= \frac{\sum_{i=1}^{N} x_i^2}{N} - \bar{x}^2$$

$$x'x = \sum_{i=1}^{N} x_i^2 = x_1^2 + x_2^2 + x_3^2 + \ldots + x_N^2 \tag{12.42}$$

This "sum of squares" is one component in the variance.

Second, recall that a mathematical average is $\bar{x} = \frac{\sum x_i}{N}$, or $\frac{1}{N}\{x_1 + x_2 + x_3 + \ldots + x_N\}$. In matrix algebra, we can create this same value by multiplying x by a row of 1's, as in

$$\frac{1}{N} \times [1, 1, 1, \ldots, 1] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \frac{1}{N}[x_1 + x_2 + x_3 + \ldots + x_N] \tag{12.43}$$

Write down the matrix X like this:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \tag{12.44}$$

So $X'X$ is

$$X'X = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_N \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} = \tag{12.45}$$

$$= \begin{bmatrix} 1 + 1 + 1 + \ldots + 1 & x_1 + x_2 + x_3 + x_N \\ x_1 + x_2 + x_3 + \ldots + x_N & \Sigma_{i=1}^{N} x_i^2 \end{bmatrix} \tag{12.46}$$

$$X'X = \begin{bmatrix} N & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \tag{12.47}$$

In other words, if you had $X'X$, you would have all of the information necessary to calculate the mean and the variance.

But wait! There's more! The covariance of two variables is also simplified in this way. Recall that covariance between two variables x and z

$$Covar(x, z) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(z_i - \bar{z})}{N} \tag{12.48}$$

The covariance between two variables x and y can be rewritten in the same way that the variance was revised in 12.41 to allow calculation with one pass through the data:

$$Covar(x,z) = \frac{1}{N}\sum_{i=1}^{N} x_i z_i - \left(\frac{\sum x_i}{N}\right)\left(\frac{\sum z_i}{N}\right) = \frac{1}{N}\sum_{i=1}^{N} x_i z_i - \bar{x}\bar{z} \tag{12.49}$$

We would make one pass, calculating the sum of the x's, z's, and $x \cdot z$, and then we would have all of the required components.

If one compares expression 12.49 with 12.41, one should immediately see that the covariance of x with itself, $Covar(x,x)$, is equal to the variance of x, $Var(x)$. One should also expect that we could use $X'X$ to represent the information needed to calculate $Covar(x,y)$.

Insert the column vector z onto the X matrix, as in:

$$X = \begin{bmatrix} 1 & x_1 & z_1 \\ 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \\ \vdots & \vdots & \vdots \\ 1 & x_N & z_N \end{bmatrix} \tag{12.50}$$

After including z as a third column, $X'X$ would give all of the information needed to calculate the covariance:

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_N \\ z_1 & z_2 & z_3 & \cdots & z_N \end{bmatrix} \begin{bmatrix} 1 & x_1 & z_1 \\ 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \\ \vdots & \vdots & \vdots \\ 1 & x_N & z_N \end{bmatrix} = \begin{bmatrix} N & \sum x_i & \sum x_i \\ \sum x_i & \sum x_i^2 & \sum x_i z_i \\ \sum z_i & \sum x_i z_i & \sum z_i^2 \end{bmatrix} \tag{12.51}$$

From this one matrix of numbers, it is possible to calculate $Var(x)$, $Var(z)$, and $Covar(x,z)$.

But wait! There's even more! The sum of squares and cross products also includes information needed to calculate the correlation between any two columns of data in the matrix. And, as all good statistics students know, the Pearson Correlation Coefficient is the ratio of the Covariance to the product of the standard deviations (the square roots the variances):

$$r = \frac{Covar(x,z)}{\sqrt{Var(x)} \cdot \sqrt{Var(z)}} \tag{12.52}$$

In conclusion, a statistician who has $X'X$ should be a happy person, a confident person, a more-well-liked person. It is the starting point for most happy endings in life.

For purposes of trivia, we hasten to point out that $(X'X)$ is always a square, symmetric matrix. If you multiply an $(m \times n)$ matrix by an $(n \times m)$ matrix, you get an $(n \times n)$ matrix–a square matrix.

It is also important to note that $X'X$ simplifies analysis. If X is a big data matrix with 10000 rows and 3 columns, then $X'X$ is only a (3×3) matrix.

I hate to end this section on a bad note, but I'm compelled to confess something that has recently been brought to my attention. Computer calculations with floating point numbers impose a rounding effect. The error due to rounding is always present, and often it is not given careful consideration. When designing a computer program to make precise calculations, it turns out, surprisingly enough, that the most basic, simple two-pass through the data approach implied by expression 12.40 (or covariance by 12.48) yields a more accurate estimate than would the one-pass approach implied by expression 12.41 (or 12.49). This mismatch between algebra and computer rounding error is at the heart of complaints about the inaccurate calculations that come out of the Microsoft Excel Spreadsheet. In other words, before you get too excited about writing a computer code, there is some work in the field of "numerical and applied linear algebra" that should be scrutinized. This is one of the frustrating things about being a teacher. After you think you have found out something really great, then you find out that its greatness is more appealing in theory than in practice.

12.6 The Inverse of a Matrix

12.6.1 Finding the intersection of two lines.

Perhaps you took algebra in high school and there were problems like this:

$$\begin{array}{rcrcl} 3x & + & 4y & = & 7 \\ 1x & + & 2y & = & 3 \end{array} \tag{12.53}$$

Here we have 2 linear equations, and if you plot them, you see they "cross" at one point. That (x, y) combination is the "solution" to this system of equations.

There is a practical way to solve this without graphing it. We "massage" the equations my multiplying and dividing until we reshape the result. For example, do this:

1. Multiply the second equation by 2 and subtract the result from the first equation:

$$\begin{array}{rcrcl} 1x & + & 0y & = & 1 \\ 1x & + & 2y & = & 3 \end{array} \tag{12.54}$$

2. Gaze happily upon the result and then subtract the first equation from the second. You get:

$$\begin{array}{rcrcl} 1x & + & 0y & = & 1 \\ 0x & + & 2y & = & 2 \end{array} \tag{12.55}$$

3. Now multiply the second equation by (1/2)and you get::

$$\begin{array}{rcrcl} 1x & + & 0y & = & 1 \\ 0x & + & 1y & = & 1 \end{array} \tag{12.56}$$

Whoa! There's an answer! It says the solution is $x = 1$ and $y = 1$.

If you write this problem in matrix terminology, you have a starting point like this:

$$\begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 7 \\ 3 \end{bmatrix} \tag{12.57}$$

and we end up with this:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{12.58}$$

or

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{12.59}$$

One would usually not be solving for a vector $[x, y]'$, but rather it would be called $[x_1, x_2]$ or $[\hat{b}_1, \hat{b}_2]$.

The procedure of multiplying and adding rows to massage the data into this form is called Gaussian Elimination. It is a time-honored method of solving systems of equations.

12.6.2 When would this approach fail?

This approach would fail if the two lines did not intersect–that is, if they were parallel lines. For example, these are parallel lines:

$$\begin{array}{rcrcl} 6x & + & 4y & = & 7 \\ 3x & + & 2y & = & 3 \end{array} \tag{12.60}$$

One can try to find a solution using the approach in the previous section, but it only leads to nonsense. Multiply the second equation by two:

$$\begin{array}{rcrcl} 6x & + & 4y & = & 7 \\ 6x & + & 4y & = & 6 \end{array} \tag{12.61}$$

If we subtract the second equation from the first, we arrive at the apparently ridiculous result

$$\begin{array}{rcrcl} 0x & + & 0y & = & 1 \\ 6x & + & 4y & = & 6 \end{array} \tag{12.62}$$

This does not make sense. And it shouldn't. Parallel lines don't cross and there should be no "solution" for a point of intersection.

12.6.3 Inverse, Identity, and anything else cool starts with "i"!

12.6.3.1 Identity Matrix

A diagonal matrix has 0's everywhere but the **main diagonal**:

$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix} \tag{12.63}$$

There is a special diagonal matrix called the **identity matrix**:

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12.64}$$

The identity matrix fulfills a special role. In the same way that multiplying some number by 1 gives back that same number, multiplying a matrix by the identity matrix gives the same matrix back. It is thus always true that if it X conforms to I, then

$$I \times X = X \tag{12.65}$$

Note that in equation 12.59, I used this fact about the identity matrix when I reduced

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{12.66}$$

to

$$\begin{bmatrix} x \\ y \end{bmatrix} \tag{12.67}$$

12.6.3.2 Inverse. A^{-1} works like $1/A$.

From elementary mathematics, one should recall that $3 \times \frac{1}{3} = 1$. As long as x is not equal to 0, then $x \times \frac{1}{x} = 1$. Recall also that $1/x$ is also represented as x^{-1}. If $x = 0$, this expression is undefined. In the previous section, we have found the matrix equivalent of "1/0 is undefined" when we tried to find the intersection of parallel lines.

In a matrix setting, we want to solve the "systems of equations." We would like to find a way to know if they can be solved in the first place, and then to calculate the solution if it exists.

The advantage of using the matrix representation is mainly in simplifying notation for large problems. In a small problem, say with two unknowns, the benefit of putting the numbers into matrices is really quite minor. Suppose our problem is to find x:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{12.68}$$

We are given numerical values for A and y, and we want to find the solution, the value of $x = [x_1, x_2]$ that "solves" the system. For example

$$A = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad y = \begin{bmatrix} 7 \\ 3 \end{bmatrix} \tag{12.69}$$

or

$$Ax = y \tag{12.70}$$

If there were an inverse for A, which we call A^{-1}, one could multiply it by both sides:

$$A^{-1}Ax = A^{-1}y \tag{12.71}$$

$$x = A^{-1}b \tag{12.72}$$

This would give you an answer, if there is one. If you had the inverse, that would mean you could solve for the values of the vector x.

The approach that is spelled out below will help solve systems of equations of all sizes. If we are presented with a system with more equations,

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{12.73}$$

we are confident that we can solve this, if a solution exists. If somebody gives us the values for the A_{ij}'s and y_i's, then we would be able to state (with confidence) that there is a solution, a set of 4 values for x_1, x_2, x_3, and x_4, that can make all of these equations hold true at the same time. Or we might report back that this is impossible.

Division is not defined for matrices. But we can do the next best thing, multiply the inverse. A by the inverse of A, which is called A^{-1}. I realize it is strange that A^{-1} may exist and be defined, but we do not write the fraction $1/A$. It is just "not done."

The inverse is defined implicitly. The inverse is a matrix such that

$$A \times A^{-1} = I \tag{12.74}$$

it seems pretty obvious that the same will be true if we reverse the order of the matrices.

$$A^{-1} \times A = I \tag{12.75}$$

If the inverse exists, then we could pre-multiply 12.73 on both the left and right hand

sides of the equation by A^{-1},

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}^{-1} \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{12.76}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{12.77}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{12.78}$$

As long as A^{-1}exists, the happy ending is sure to follow. The values of x_1, x_2, x_3, and x_4 that will solve this system can be calculated.

And, if A^{-1} does not exist, then the system of equations cannot be solved. They are, roughly speaking, "parallel lines."

I know you are thinking, "he just side-stepped the question. He said he could solve the problem, he did not show us how." Or perhaps you are thinking, "big deal, where does A^{-1} come from?" We'll get back to that, eventually.

12.6.4 What is it good for?

Why would you ever want to invert a square matrix? Why would you have a square matrix in the first place?

The suspicious reader should already have guessed. $X'X$ is a square matrix. Maybe there is some magic to be had if we could invert it!

A detailed explanation would take us a little bit out of our way, but there's room for a brief explanation. Suppose you are given data that represents a column of N "outcome scores," y, and a matrix of M columns of input variables, X. I'm going to fill the first column of X with 1's, to represent the idea that a respondent is "present" for a survey or a unit can be

observed. Having those 1's in the first column is handy, as we have already seen.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 4.0 \\ 3.3 \\ 2.2 \\ \vdots \\ 3.4 \end{bmatrix} \tag{12.79}$$

and

$$X = \begin{bmatrix} 1 & X_{12} & X_{13} & \cdots & X_{1M} \\ 1 & X_{22} & X_{23} & & X_{2M} \\ 1 & X_{32} & X_{33} & & \\ \vdots & & & \ddots & \vdots \\ 1 & X_{N2} & X_{N3} & \cdots & X_{NM} \end{bmatrix} = \begin{bmatrix} 1 & 7 & 12 & \cdots & 734 \\ 1 & 8 & 11 & & 554 \\ 1 & 7 & 16 & & 335 \\ \vdots & & & \ddots & \vdots \\ 1 & 6 & 9 & \cdots & 550 \end{bmatrix} \tag{12.80}$$

You are allowed to pick M coefficients, $\hat{b} = [\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_m]$ that can be used to calculate a column of predictions like so:

$$\hat{y} = X \times \hat{b} \tag{12.81}$$

Just for clarity, that is:

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} 1 & X_{12} & X_{13} & \cdots & X_{1M} \\ 1 & X_{22} & X_{23} & & X_{2M} \\ 1 & X_{32} & X_{33} & & \\ \vdots & & & \ddots & \vdots \\ 1 & X_{N2} & X_{N3} & \cdots & X_{NM} \end{bmatrix} \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \\ \vdots \\ \hat{b}_M \end{bmatrix} \tag{12.82}$$

Over the years, several different ideas of the "best guess" for $\hat{b}$ have been offered. A traditional favorite is the so-called "least squares estimator," which chooses $\hat{b}$ so as to make the sum of squared errors as small as possible. The sum of squared errors, of course, results from finding the gap between the guess and the observed, $(y_i - \hat{y}_i)$, and then that gap is squared and summed across all observations.

$$\sum_{i=1}^{N} (y_i - \hat{y}_i) \tag{12.83}$$

whereas in matrix notation, it would be

$$(y - \hat{y})'(y - \hat{y}) \tag{12.84}$$

and, since $\hat{y} = X\hat{b}$, this is the same as

$$(y - X\hat{b})'(y - X\hat{b}) \tag{12.85}$$

A knowledge of calculus is required to derive the next result, but no calculus is required to understand its significance. If one allows the values of $\hat{b}_1$, $\hat{b}_2$, $\hat{b}_3$, and so forth to "float up and down" until the the sum of squared errors is as small as possible, the solution will satisfy this condition:

$$(X'X)\hat{b} = X'y \tag{12.86}$$

The values in X and y are 'given' and we are trying to solve for $\hat{b}$. Compare this expression against equation 12.73. Here, the role of A is being played by $(X'X)$ and the role of y is being played by $X'y$. If $(X'X)$ can be inverted, then we would have solution.

$$(X'X)^{-1}(X'X)\hat{b} = (X'X)^{-1}X'y \tag{12.87}$$

The best estimate for the coefficients is given by:

$$\hat{b} = (X'X)^{-1}X'y \tag{12.88}$$

$(X'X)^{-1}$ is a vital part of the standard error of parameter estimates.

A casual reader (?) might skip the rest of this section. For the sake of completeness, some comments about another role played by $(X'X)^{-1}$ is discussed. Because the estimate $\hat{b}$ depends on random error, we should be uncertain about it. The amount of uncertainty that we assign to $\hat{b}$ depends on $(X'X)^{-1}$. Continuing with expression 12.88, apply the variance operator to both sides.

$$Var(\hat{b}) = Var[(X'X)^{-1}X'y] \tag{12.89}$$

$$= Var[(X'X)^{-1}X'(Xb + e)] \tag{12.90}$$

$$= Var[(X'X)^{-1}X'Xb + (X'X)^{-1}X'e] \tag{12.91}$$

$$= 0 + Var[(X'X)^{-1}X'e] \tag{12.92}$$

$$= (X'X)^{-1}X' \cdot Var[e] \cdot X(X'X)^{-1} \tag{12.93}$$

This final step is due to the fact that $Var[Ze] = ZVar[e]Z'$. Note that the outside parts of this expression, $(X'X)^{-1}$, are the bread of a sandwich.

For reasons that will be described below, the different types of regression analysis are distinguished by their assumptions about $Var[e]$. Ordinary Least Squares (OLS) analysis imposes assumptions that yield a simple result:

$$Var(\hat{b}) = \hat{\sigma}^2 \cdot (X'X)^{-1} \tag{12.94}$$

On the other hand, if the error variances follow a more complicated structure, then a different approach is needed. One approach, called either Weighted or Generalized Least Squares (WLS or GLS), the structure of $Var[e]$ is considered. The matrix of weights is given by the inverse of the error variance matrix.

$$W = Var[e]^{-1} \tag{12.95}$$

Instead of minimizing the unweighted sum of squares in 12.85, we would choose estimates $\hat{b}$ to minimize this weighted sum of squares:

$$(y - X\hat{b})' \cdot W \cdot (y - X\hat{b}) \tag{12.96}$$

The generalized least squares estimator for the slope coefficients throws the 'weight' matrix into the middle of everything, but the formula is not hugely different from OLS:

$$\hat{b} = (X'WX)^{-1}X'WY \tag{12.97}$$

and

$$Var(\hat{b}) = \hat{\sigma}^2(X'WX)^{-1} \tag{12.98}$$

Another approach, which is described in a later section, is the effort to calculate "robust standard errors," estimates of our uncertainty about $\hat{b}$ that are intended to rely more on the data itself than on assumptions about error structure.

12.6.5 How do you know an Inverse exists?

Not all systems of equations can be solved. As we saw, if you have two "parallel" lines, there is no intersection and hence no solution.

In order for a unique solution to exist, the number of unknowns has to equal the number of equations. That means the matrix A must be square. And that's a happy thing, since we already saw that only square matrices can be inverted (and $X'X$ is square).

The matrix equivalent of "parallel lines" arises when you apply row operations as described in the previous section and you end up with one or more rows of 0's. Suppose you did the

Gaussian elimination and found a system of equations like this:

$$\begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \end{bmatrix} \tag{12.99}$$

Quite plainly, there is no choice of (x, y) that will satisfy both equations. There is NO SOLUTION. It happened because the two equations were not providing different information about their slopes. they were proportional to each other. That would mean "parallel" in a graph. There is some 'redundant' information between the two equations–they are pointing in the same direction.

TERMINOLOGY: the matrix on the left is "not full rank." It is also said to be "singular".

The "**rank**" of a matrix is the number of "linearly independent" rows (rows that can't be reduced to all 0's by any sequence of row operations).

In some statistical programs, one will find the error message "matrix is not full rank." That means there are some 'redundant' rows.

If a matrix is "**singular**," it means no inverse exists. The term "singular" enters the fray because each matrix can be characterized by a number called a "determinant". The determinant of A is referred to either as $det(A)$ or $|A|$. It can be shown that the inverse of a matrix A is proportional to $\frac{1}{det(A)}$. A fraction is singular (undefined) if the denominator is 0. If the determinant is 0, then $\frac{1}{det(A)}$is undefined (singular), meaning the inverse does not exist.

And if no inverse exists, there's no solution to the problem.

The **determinant** of a matrix is just one way to find out if a matrix can be inverted, but it is a prominent way.

For a 2x2 matrix:

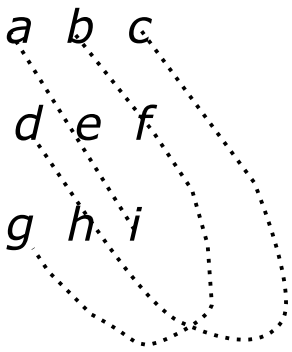
$$det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = \left|\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right| = a \cdot b - c \cdot b \tag{12.100}$$

Simple: multiply "down" diagonal, and subtract the result from multiplying "up" the other diagonal.

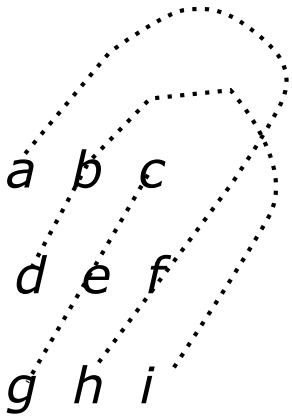
For a $3x3$ matrix,

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \tag{12.101}$$

the determinant is found by multiplying the numbers along the three 3 "downs"



and subtracting the product of the numbers on the 3 "ups":



The end result is

$$aei + bfg + chd - gec - bdi - afh$$

For a bigger matrix, calculating the determinant becomes unweildy. But there are plenty of math books floating around in the world and I'd urge you to consult one.

If the determinant is not exactly 0, but almost zero, then $1/det(A)$ will be a huge number, possibly tending towards infinity. In cases like that, it may be possible to calculate an inverse, but the result might not be meaningful. That is a case in which the so-called regression problem of "multicollinearity" would arise.

12.6.6 How do you actually find inverses?

The first third of a matrix algebra book will be about ways to find out if an inverse exists and how it can be found. It is somewhat of a disappointment to me (like most part time users) to find out that the most logical, practical ways to find inverses with pen-and-pencil are absolutely disastrous when you use those approaches in a computer. Computers round numbers, and there is a separate field "numerical linear algebra" that probes ways to optimally make calculations. I recently saw a presentation in which one section was called "Yes, Virginia, we don't actually invert $X'X$ anymore." There are more precise ways, like calculation of a "pseudoinverse" via the "singular value decomposition."

As a result of the complexities of numerical algebra, I don't want to go into a lot of detail about how to calculate inverses, maybe some other day it will be necessary to dig into that level of detail.

12.7 One use of the Kronecker Product

In this section, I display my devotion to the symbol $\otimes$. It is such a beautiful, mysterious symbol. When I first laid eyes on it, I knew for sure that, one day, I would know what it is good for.

12.7.1 Definition of the Kronecker product

The Kronecker product $A \otimes B$ means that one takes each term in the first matrix and multiplies it by the ENTIRE second matrix and then puts the result in place of the element of the first matrix.

Here is an example definition where A and B are both matrices:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & & & \\ a_{m1}B & a_{m2}B & & a_{mn}B \end{bmatrix} \tag{12.102}$$

The individual elements of A, which is an $m \times n$ matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{12.103}$$

are multiplied with the whole matrix B.

The kronecker product is especially useful when we are trying to represent "block diagonal" data structures. Block diagonal data structures are often encountered in regression models that are used to investigate cross-sectional time series data. We will return to that shortly.

12.7.2 An ordinary regression has a pleasant "random error"

In many of the models, we hypothesize that there are N observations. Our data may include some measurement error or other unpredictable variation. Represent that error as a vector,

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_N \end{bmatrix} \tag{12.104}$$

In a perfect world, we would like each of these bits of random error to be "independent" from each other bit. That is, we'd be very happy if there was no covariance between the errors.

We then hypothesize that this error is added, as in:

$$y = Xb + e \tag{12.105}$$

If you did not study regression yet, don't worry. Most of the focus here is not on the regression, per se, but rather on understanding the column of random errors, e.

The next step is to consider each error in isolation. Don't think of "the error" as being one variable, but rather a column of N random variables. This requires a conceptual leap for most students. We consider a particular error observed at time 1, e_1, as a draw from a set of possible outcomes. Then consider e_2 as a draw from some range of possibilities. And so forth. We think of e_1 and e_2 as draws from distributions because we want to entertain the possibility that the each error term is drawn from a different distribution. The variance of the error distribution for the first case, $Var(e_1)$, might be different from the other cases.

To make the problem simpler, most models are framed around the assumption that the errors are, on the average, neutral. Some are positive, some are negative, but, overall, they "average out." Formally speaking, the expected value of each error is 0, $E[e_i] = 0$.

Viewed in that way, the variance-covariance matrix of the error terms,$Var(e)$, is an NxN matrix:

$$Var(e) = Var\begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_N \end{bmatrix} = \begin{bmatrix} Var(e_1) & Cov(e_1,e_2) & \cdots & Cov(e_1,e_N) \\ Cov(e_1,e_2) & Var(e_2) & \cdots & Cov(e_2,e_N) \\ Cov(e_1,e_3) & Cov(e_2,e_3) & \cdots & Cov(e_3,e_N) \\ \vdots & \cdots & \cdots & \vdots \\ Cov(e_1,e_N) & Cov(e_2,e_N) & Cov(e_3,e_N) & Var(e_N) \end{bmatrix} \tag{12.106}$$

$$= \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} & \cdots & \sigma_{1N} \\ \sigma_{21} & \sigma_2^2 & \sigma_{23} & & \sigma_{2N} \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 & & \sigma_{3N} \\ \vdots & & & \ddots & \vdots \\ \sigma_{N1} & \sigma_{N2} & \sigma_{N3} & \cdots & \sigma_N^2 \end{bmatrix}$$

In the elementary "ordinary least squares" model, the strong simplifying assumption is invoked. We act as though the covariances are all 0 and the diagonal elements, the variances, are all equal to the same number, σ^2.

$$Var(e) = \begin{bmatrix} \sigma^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma^2 & 0 & & 0 \\ 0 & 0 & \sigma^2 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma^2 \end{bmatrix} \tag{12.107}$$

In words, each observation's random error is unrelated to the random errors that affect other observations, and, in addition, the variance of the random error for each observation is the same number, σ^2. In matrix terms, this reduces to an even simpler statement

$$= \sigma^2 \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} = \sigma^2 I \tag{12.108}$$

In most areas of advanced regression research, scholars are searching for ways to estimate models without that strong simplifying assumption.

12.7.3 Robust standard errors

In one approach to this problem, known as the "robust standard error" approach, one tries to estimate the variances of the error terms from the residuals in a fitted model. Suppose

the residuals from the fitted model are in a column vector $\hat{e}$. The theoretical variance of a variable is defined as

$$Var(e_i) = E[(e_i - E(e_i))^2].$$

Since the error terms are on average neutral, $E[e_i] = 0$, so this reduces to

$$Var(e_i) = E[e_i^2]$$

Similarly, covariance reduces to

$$Cov(e_i, e_j) = E[e_i \cdot e_j]$$

These theoretical (unobserved) values can be approximated by the observed errors $\hat{e}$ in the following sense.

$$\hat{e} \cdot \hat{e}' = \begin{bmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \\ \vdots \\ \hat{e}_N \end{bmatrix} \begin{bmatrix} \hat{e}_1 & \hat{e}_2 & \hat{e}_3 & \cdots & \hat{e}_N \end{bmatrix} \tag{12.109}$$

This matrix, which is $N \times N$, might be called $\widehat{Var(e)}$ or $\hat{\Omega}$.

$$\widehat{V(e)} = \hat{\Omega} = \frac{1}{N} \hat{e} \cdot \hat{e}'$$

$$\widehat{V(e)} = \hat{\Omega} = \frac{1}{N} \begin{bmatrix} \hat{e}_1^2 & \hat{e}_1\hat{e}_2 & \hat{e}_1\hat{e}_3 & \cdots & \hat{e}_1\hat{e}_N \\ \hat{e}_2\hat{e}_1 & \hat{e}_2^2 & \hat{e}_2\hat{e}_3 & \cdots & \hat{e}_2\hat{e}_N \\ \hat{e}_3\hat{e}_1 & \hat{e}_3\hat{e}_2 & \hat{e}_3^2 & & \\ \vdots & & & \ddots & \\ \hat{e}_N\hat{e}_1 & \hat{e}_N\hat{e}_2 & & & \hat{e}_N^2 \end{bmatrix} \tag{12.110}$$

Recall expression 12.93, in which one representation of our uncertainty about $\hat{b}$ was shown to "sandwich" the variance of the error term within a block of matrices. The robust standard error, also often called the Huber-White standard error, uses this estimate of $\widehat{V(e)}$ in place of the theoretical value. This approach is "robust" to theoretical errors in the sense that the data itself drives the exercise.

12.7.4 Cross Sectional Time Series Framework.

It is popular today to do data analysis with information gathered by repeated observations on different units of observation. One might have United Nations information for 10 years on

each of 120 countries, or data on 48 US (continental) states. To separate the information from different units and different times, it has become common to use a two-subscript approach. The first subscript refers to the unit–or "cluster"–the second to the time. The dependent variable is y_{it} (i= unit, t=time) and the set of independent variables observed for each country and time is x_{it}. The "cross sectional time series" (CXTS) model looks like an ordinary regression. The only difference is that data from N different units is "stacked" together:

$$\begin{array}{lcll} y_{11} & & x_{11}b & + \; e_{11} \\ y_{12} & & x_{12}b & + \; e_{12} \\ y_{13} & & x_{13}b & + \; e_{13} \\ \ldots & & & + \\ y_{1T} & & x_{1T}b & + \; e_{1T} \\ y_{21} & = & x_{21}b & + \; e_{21} \\ y_{22} & & x_{22}b & + \; e_{22} \\ \ldots & & & + \\ y_{2T} & & x_{2T}b & + \; e_{2T} \\ y_{31} & & x_{31}b & + \; e_{31} \\ y_{32} & & x_{32}b & + \; e_{32} \\ \ldots & & & + \\ y_{3T} & & x_{3T}b & + \; e_{3T} \end{array} \tag{12.111}$$

The data matrix X is a "stack" of smaller matrices, one for each cluster. Suppose there is an intercept and 3 variables to be estimated:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{N-1} \\ X_N \end{bmatrix} = \begin{bmatrix} 1 & x1_{11} & x2_{11} & x3_{11} \\ 1 & x1_{12} & x2_{12} & x3_{12} \\ 1 & x1_{1T} & x2_{1T} & x3_{1T} \\ 1 & x1_{21} & x2_{21} & x3_{21} \\ 1 & x1_{22} & x2_{22} & x3_{22} \\ 1 & x1_{23} & x2_{23} & x3_{23} \\ 1 & x1_{31} & x2_{31} & x3_{31} \\ 1 & & & \\ 1 & & & \\ 1 & x1_{N(T-1)} & x2_{N(T-1)} & x3_{N(T-1)} \\ 1 & x1_{NT} & x2_{NT} & x3_{NT} \end{bmatrix} \tag{12.112}$$

Similarly, the vector of observations:

$$y = \begin{bmatrix} y_{11} \\ y_{12} \\ \vdots \\ y_{1T} \\ y_{21} \\ \vdots \\ y_{2T} \\ \vdots \\ y_{N1} \\ \vdots \\ y_{NT} \end{bmatrix} \tag{12.113}$$

What is the variance of the error term? The variance matrix that is used in OLS would restrict all elements on the main diagonal to a single value, and all of the others would be set equal to 0, as in

$$Var(e) = \begin{bmatrix} \sigma^2 & ... & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & ... & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2 & ... & & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & & \vdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & & ... & \sigma^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2 & ... & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2 \end{bmatrix} \tag{12.114}$$

How can we "generalize" our theory, so that it better takes into account the variety that we might observe across units? There are a number of possibilities.

In the longitudinal analysis literature, it is common to assume the variance/covariance matrix is "block diagonal". That means there can be correlations of error terms within each cluster, but the observations of the clusters are not influenced by events in other clusters. In the i'th unit, at the j'th time, the variance is σ^2_{ij} and the covariance of errors within that unit at times s and t is $\sigma^2_{ist} = E(e_{is}, e_{it})$.

$$Var(e) = \begin{bmatrix}
\sigma^2_{11} & ... & \sigma^2_{11T} & 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 \\
\sigma^2_{112} & ... & \sigma_{12T} & 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 \\
 & ... & \vdots & 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 \\
\sigma^2_{11T} & ... & \sigma^2_{1TT} & 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 \\
0 & 0 & 0 & \sigma^2_{21} & ... & \sigma^2_{21T} & 0 & 0 & 0 & & 0 & 0 \\
0 & 0 & 0 & \vdots & & \vdots & 0 & 0 & 0 & & 0 & 0 \\
0 & 0 & 0 & \sigma^2_{21T} & ... & \sigma^2_{2T} & 0 & 0 & 0 & & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \sigma^2_{31} & ... & \sigma^2_{31T} & & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \vdots & \ddots & \vdots & & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \sigma^2_{31T} & & \sigma_{3T} & & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & \vdots & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & \sigma^2_{N(T-1)} & \sigma^2_{N(T-1)T} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & \sigma^2_{N(T-1)T} & \sigma^2_{NT}
\end{bmatrix} \tag{12.115}$$

Notice how the error variance matrix has two special properites. The errors for each specific unit are represented by the blocks along the main diagonal. All other values are set equal to 0 on the grounds that the units are not influencing each other.

12.7.5 Panel Corrected Standard Errors and the Kronecker product

In a highly influential article,"What to do (and not to do) with Time-Series Cross-Section Data," political scientists Neil Beck and Jonathan Katz (*American Political Science Review*, 1995) introduced an alternative conception of the structure of random errors in a cross-sectional time series analysis. To find their use of the Kronecker product, inspect page 646:

$$\hat{\Omega} = \frac{(E'E)}{T} \otimes I \tag{12.116}$$

Untangling this expression is my major objective in this section. The major result, the so-called "panel corrected standard error," is found by using this corrected estimate of the error variance/covariance matrix in the calculation of the estimated uncertainty about the model's estimates.

Beck and Katz are considering a situation in which the error structure follows a particular pattern. Within units, the variance of the errors is a fixed quantity. However, unlike the usual longitudinal model, the random effects on units may be correlated with each other. That is to say, there may be contemporaneous random affects that apply "across units" at

an instant in time. Consider this example in which there are 3 observations per unit. The hypothesized error variance matrix is.

$$\Omega = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \sigma_{12} & 0 & 0 & & \sigma_{1N} & 0 & 0 \\ 0 & \sigma_1^2 & 0 & 0 & \sigma_{12} & 0 & & 0 & \sigma_{1N} & 0 \\ 0 & 0 & \sigma_1^2 & 0 & 0 & \sigma_{12} & & 0 & 0 & \sigma_{1N} \\ \sigma_{12} & 0 & 0 & \sigma_2^2 & 0 & 0 & & \sigma_{2N} & 0 & 0 \\ 0 & \sigma_{12} & 0 & 0 & \sigma_2^2 & 0 & & 0 & \sigma_{2N} & 0 \\ 0 & 0 & \sigma_{12} & 0 & 0 & \sigma_2^2 & \cdots & 0 & 0 & \sigma_{2N} \\ & & \vdots & & & \vdots & \ddots & & & \\ \sigma_{1N} & 0 & 0 & \sigma_{2N} & 0 & 0 & & \sigma_N^2 & 0 & 0 \\ 0 & \sigma_{1N} & 0 & 0 & \sigma_{2N} & 0 & & 0 & \sigma_N^2 & 0 \\ 0 & 0 & \sigma_{1N} & 0 & 0 & \sigma_{2N} & & 0 & 0 & \sigma_N^2 \end{bmatrix} \tag{12.117}$$

How can the Kronecker Product play a role in simplifying our representation of this problem? Begin by writing a matrix of the variances within units and covariances across the units as,

$$V = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} & \cdots & \sigma_{1N} \\ \sigma_{21} & \sigma_2^2 & \sigma_{21} & \cdots & \sigma_{2N} \\ \sigma_{31} & & & & \\ \vdots & & & & \\ \sigma_{N1} & \sigma_{N2} & \cdots & \sigma_{N,N-1} & \sigma_N \end{bmatrix} \tag{12.118}$$

This is where the Kronecker magic enters. Observe the simplifying power of the Kronecker product:

$$\Omega = V \otimes I = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} & \cdots & \sigma_{1N} \\ \sigma_{21} & \sigma_2^2 & \sigma_{21} & \cdots & \sigma_{2N} \\ \sigma_{31} & & & & \\ \vdots & & & & \\ \sigma_{N1} & \sigma_{N2} & \cdots & \sigma_{N,N-1} & \sigma_N \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12.119}$$

In order to use this theory in practice, it is necessary to create estimates of the variance parameters in V. Beck and Katz observe that the residual vectors from a fitted model for each cluster can be used to create these estimates. The estimate of the variance of the error term for cluster 1 is

$$\hat{\sigma}_1^2 = \frac{1}{T}\{\hat{e}_{11}^2 + \hat{e}_{12}^2 + \cdots + \hat{e}_{1T}^2\} \tag{12.120}$$

This is the mean squared error for unit 1. A similar approach is used to create estimates of the covariances, σ_{ij}.

We might go crazy writing that down for each of the N clusters. In an effort to keep everything organized, we label the residual vectors from the clusters as $\hat{e}_1$, $\hat{e}_2$, ... , $\hat{e}_N$. These are vectors with T elements each, and they can be grouped together as a $T \times N$ matrix (the $\hat{e}_i$ are columns):

$$E = \left[\begin{array}{cccccc} \hat{e}_1 & \hat{e}_2 & \cdots & \hat{e}_{N-1} & \hat{e}_N \end{array} \right] \tag{12.121}$$

The data based estimates of the error variances represented by V in expression 12.118 are obtained in the following way:

$$\hat{V} = \frac{1}{T} E'E = \frac{1}{T} \left[\begin{array}{c} \hat{e}_1' \\ \hat{e}_2' \\ \vdots \\ \hat{e}_{N-1}' \\ \hat{e}_N' \end{array} \right] \left[\begin{array}{cccccc} \hat{e}_1 & \hat{e}_2 & \ldots & \hat{e}_{N-1} & \hat{e} & {}_N \end{array} \right] \tag{12.122}$$

On p. 646 they write:

$$\hat{\Omega} = \frac{(E'E)}{T} \otimes I \tag{12.123}$$

12.8 Orthogonality.

Orthogonality is a property of matrices and vectors.

Two vectors are orthogonal if they are "perpendicular" or are at "right angles". Orthogonality between x and y implies

$$x' \cdot y = 0 \tag{12.124}$$

Make a plot of $x = (1, 0)$ and $y = (0, 1)$. They form a right angle, and $x'y = 0$.

If x and y represent 2 vectors of data, each for N survey respondents, we could think of the correlation between them as a function of the angle between them as it depends on $x'y$. If the vectors are at a right angle, then they do not share any "direction", they are uncorrelated.

An **orthogonal matrix** is defined as a square matrix with the property that

$$X = (X')^{-1} \tag{12.125}$$

that's the same as:

$$X^{-1} = X' \tag{12.126}$$

In words, the inverse of an orthogonal matrix equals its transpose.

That definition is the formal one, but the most important thing is a provably equivalent property. An orthogonal matrix has the property that each of its rows is orthogonal to each of its other rows. Here is an example of an orthogonal matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12.127}$$

(There is also the subsidiary property that each vector has "length" one, meaning $v' \cdot v = 1$, but that's more important in linear algebra than statistics.)

Why would anybody ever want an orthogonal matrix? An orthogonal matrix would consist of rows of observations that "are not interrelated with each other" at all. When psychologists create experimental designs, they may strive to create orthogonal experimental conditions. If the outcomes are truly orthogonal, then it means that the separate effects can be clearly discerned.

12.9 Linear Algebra Pen and Paper Exercises

I found exercises like this on plenty of websites, so they must be good:

1. Given

$$A = \begin{bmatrix} \beta & \gamma & \phi \\ \pi & \lambda & \Theta - \eta \\ \mu & \rho & \Gamma \end{bmatrix}$$

   Find A'

   Answer:

$$A' = \begin{bmatrix} \beta & \pi & \mu \\ \gamma & \lambda & \rho \\ \phi & \Theta - \eta & \Gamma \end{bmatrix}$$

2. Given

$$B = \begin{bmatrix} 5 \\ 3 \\ 1 \\ 2 \end{bmatrix}$$

   Find B'

   Answer:

$$B' = \begin{bmatrix} 5 & 3 & 1 & 2 \end{bmatrix}$$

3. Given

$$C = \begin{bmatrix} 2 & 1 & 0 & x & \alpha \end{bmatrix}$$

Find C'

Answer:

$$C' = \begin{bmatrix} 2 \\ 1 \\ 0 \\ x \\ \alpha \end{bmatrix}$$

4. Given

$$D = \begin{matrix} z & y & x & w \\ a & b & c & d \end{matrix}$$

Find D'

Answer:

$$D' = \begin{matrix} z & a \\ y & b \\ x & c \\ w & d \end{matrix}$$

5. Is it possible to multiply $A \times B$? If so, do it.

Answer: NO

6. Is it possible to multiply $D \times B$? If so, do it.

Answer:

$$D \times B = \begin{bmatrix} 5z + 3y + x + 2w \\ 5a + 3b + c + 2d \end{bmatrix}$$

7. Let $A = \begin{bmatrix} -1 & 3 & 7 \\ 7 & 6 & 5 \\ 4 & 9 & 11 \end{bmatrix}$, $B = \begin{bmatrix} 2 & 1 & 0 \\ 8 & 7 & 3 \\ 11 & 2 & -1 \end{bmatrix}$

A) Find $A + B$

Answer:

$$A + B = \begin{bmatrix} 1 & 4 & 7 \\ 15 & 13 & 8 \\ 15 & 11 & 10 \end{bmatrix}$$

B) Find $A - B$

Answer:

$$A - B = \begin{bmatrix} -3 & 2 & 7 \\ -1 & -1 & 2 \\ -7 & 7 & 12 \end{bmatrix}$$

C) Find $A' + B'$

Answer:

$$A' + B' = \begin{bmatrix} -1 & 7 & 4 \\ 3 & 6 & 9 \\ 7 & 5 & 11 \end{bmatrix} + \begin{bmatrix} 2 & 8 & 11 \\ 3 & 7 & 2 \\ 0 & 3 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 15 & 15 \\ 6 & 13 & 11 \\ 7 & 8 & 10 \end{bmatrix}$$

8. Let $A = \begin{bmatrix} 4 & 6 & 2 \end{bmatrix}$ and $B = \begin{bmatrix} 2 \\ -2 \\ 3 \end{bmatrix}$

A) Find AB

Answer:

$$AB = [8 - 12 + 6] = 2$$

B) Find BA

Answer:

$$BA = \begin{bmatrix} 8 & 12 & 4 \\ -8 & -12 & -4 \\ 12 & 18 & 6 \end{bmatrix}$$

C) Find $A'A$

Answer:

$$A'A = \begin{bmatrix} 4 \\ 6 \\ 2 \end{bmatrix} \begin{bmatrix} 4 & 6 & 2 \end{bmatrix} = \begin{bmatrix} 16 & 24 & 8 \\ 24 & 36 & 12 \\ 8 & 12 & 4 \end{bmatrix}$$

D) Find $B'B$

Answer:

$$B'B = \begin{bmatrix} 2 & -2 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ -2 \\ 3 \end{bmatrix} = [17]$$

9. Let

$$A = \begin{bmatrix} 5 & 1 \\ 2 & 2 \\ 7 & -1 \end{bmatrix} \quad B = \begin{bmatrix} a & b & c & d \\ e & f & g & h \end{bmatrix}$$

a) Find AB

Answer:

$$AB = \begin{bmatrix} 5a+e & 5b+f & 5c+g & 5d+h \\ 2a+2e & 2b+2f & 2c+2g & 2d+2h \\ 7a-e & 7b-f & 7c-g & 7d-h \end{bmatrix}$$

b) Do you agree that you can't multiply $B \times A$?

Answer: Yes,$B \times A$ is undefined.

c) Calculate $Z = A'A$

Answer:

$$A'A = \begin{bmatrix} 78 & 4 \\ 3 & 6 \end{bmatrix}$$

d) Calculate $Y = BB'$

Answer:

$$BB' = \begin{bmatrix} a^2+b^2+c^2+d^2 & ae+bf+cg+dh \\ ae+bf+cg+dh & e^2+f^2+g^2+h^2 \end{bmatrix}$$

e) If you can calculate $ZY = (A'A) \times (BB')$, do it.

Answer:

$$ZY = \begin{bmatrix} 78(a^2+b^2+c^2+d^2)+4(ae+bf+cg+dh) & 78(ae+bf+cg+df)+4(e^2+f^2+g^2+h^2) \\ 3(a^2+b^2+c^2+d^2)+6(ae+bf+cg+dh) & 3(ae+bf+cg+df)+6(e^2+f^2+g^2+h^2) \end{bmatrix}$$

10. Let $C = \begin{bmatrix} 4 & 1 & 4 & 5 \end{bmatrix}$ and $D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

a) Find CD

Answer:

$$CD = \begin{bmatrix} 4 & 1 & 4 & 5 \end{bmatrix}$$

b) Find DC

Answer:

$$DC = \begin{bmatrix} 4 & 1 & 4 & 5 \end{bmatrix}$$

11. Let $X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$ and $Y = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$.

A) Find XY

Answer:

$$XY = \begin{bmatrix} (Ax_{11} + Cx_{12}) & (Bx_{11} + Dx_{12}) \\ (Ax_{21} + Cx_{22}) & (Bx_{21} + Dx_{22}) \\ (Ax_{31} + Cx_{32}) & (Bx_{31} + Dx_{32}) \end{bmatrix}$$

B) Find IX (where I represents the 3×3 identity matrix).

Answer:

$$IX = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

12. Let

$$B = \begin{bmatrix} 7 & -3 \\ 1 & 5 \\ 2 & 1 \\ -1 & 7 \end{bmatrix}.$$

A) Find $B'B$

Answer:

$$\begin{aligned} B'B &= \begin{bmatrix} 7 & 1 & 2 & -1 \\ -3 & 5 & 1 & 7 \end{bmatrix} \begin{bmatrix} 7 & -3 \\ 1 & 5 \\ 2 & 1 \\ -1 & 7 \end{bmatrix} \\ &= \begin{bmatrix} 55 & -21 \\ -21 & 84 \end{bmatrix} \end{aligned}$$

B) Find $B\,B'$

Answer:

$$\begin{aligned} BB' &= \begin{bmatrix} 7 & -3 \\ 1 & 5 \\ 2 & 1 \\ -1 & 7 \end{bmatrix} \begin{bmatrix} 7 & 1 & 2 & -1 \\ -3 & 5 & 1 & 7 \end{bmatrix} \\ &= \begin{bmatrix} 58 & -8 & 11 & -28 \\ -8 & 26 & 11 & 34 \\ 11 & 7 & 5 & 5 \\ -28 & 34 & 5 & 50 \end{bmatrix} \end{aligned}$$

From this, one should see that XX' and $X'X$ are always defined, for any matrix X.

13 Executive Summary: Calculus

When I was in high school, there was no calculus class. Trigonometry and Algebra were the ends of the road. Did I attend a poor, badly managed school in the ghetto? No, I went to a private "college preparatory" school that had competitive admissions and a good reputation. In college, I heard so many horror stories about math classes that I did not take calculus. When I started graduate school at the University of Iowa, most of the other students were in the same situation. I did not take my first calculus class until the second year of graduate school, when I enrolled in an honors section of calculus that was aimed at bright 18 year olds. If you are one of the people who never took calculus at all, I'm both sorry for you, but I'm also understanding.

How can I help? I asked my self. The answer came clear to me in a dream. Write something that helps students understand the terminology, notation, and concepts first, leaving the details until the end. In other words, prepare an "executive summary" of the sort that corporations prepare for their busy executives.

13.1 What is Calculus for?

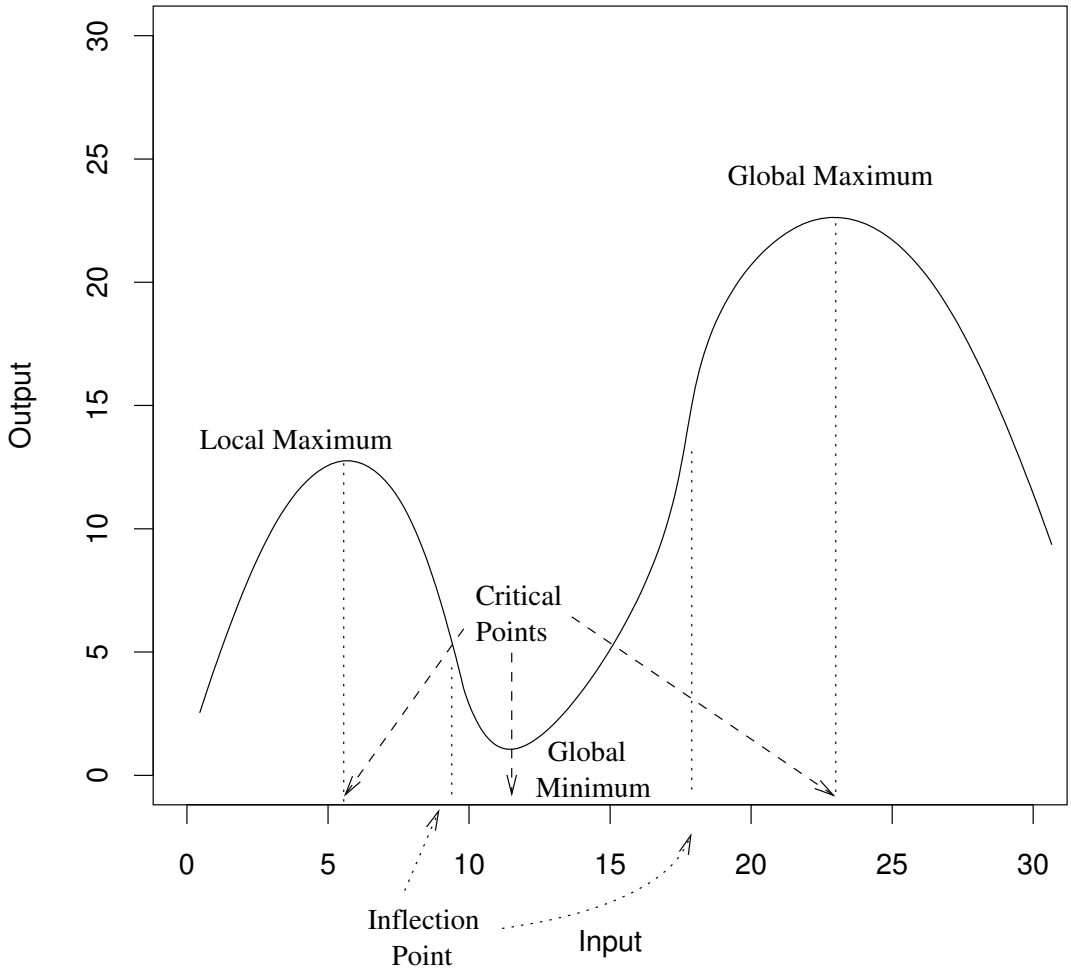
Calculus is the study of curves, change, and accumulation. That is what most social scientists are interested in, but they don't realize it from the outset. I'm trying to fix that by writing this.

13.1.1 We study relationships.

We want to make sense out of the world. We want to know if one thing causes another. We want to know if an increase in one variable causes another to go up or down. To represent this idea, we use the concept of a **function**. A function translates inputs into a single output. Consider the curve in Figure 13.1. The "independent variable," which is called the Input variable in the graph, seems to cause the "dependent variable," which is labeled Output, to go up, then down, then up again, and then down again. In the field of calculus, the emphasis is on diagnosing those ups and downs.

Calculus provides a terminology for describing functions. A **critical point** is a value of the input at which the output is either at the top of a peak or in the bottom of a valley. The

Figure 13.1: Describing a Curve: Terminology



only curve that does not have a critical point is a straight line. If a curve is not a straight line, then at any given point, its orientation is either "concave down" or "concave up." If a function is concave up, then its rate of increase–the slope–is increasing. In the vicinity of a maximum point, either a **local maximum** or a **global maximum**, the function is "concave down." In the vicinity of a minimum point, the function is "concave up." An **inflection point** is a value of the input at which the concavity changes.

In Figure 13.2, a number of different relationships are illustrated. The input variable, which is usually called X in the mathematics books, may cause the output, Y, to change in any number of ways. The set of all possible inputs is called the **domain** of the function, while the set of outputs is called the **range**. The graph in each part of the figure represents one possible "predicted" value of Y as it depends on X. The graphs in the first row of the figure show smooth, continuous change in the output, either increasing or decreasing. The graph illustrated in Figure 13.2c is different from the others because it is not **continuous**. Note how the value of the function "hops" at one point. There is no way to draw that curve without lifting the pencil from the paper. (The open loop indicates that the value of the function $f(x)$ is undefined). All of the other functions are continuous, however figure 13.2d deserves some special consideration. The function depicted there is made up of a sequence of straight lines. This is called a "**spline**." The points at which the slope changes are called knots. In many situations, splines are used to approximate smoothly changing functions.

The study of calculus usually begins with an effort to describe curvature of functions like these. Students are given a formula and asked to figure out, for example, if it is increasing or decreasing at a certain point. Students are supposed to be able to figure out if the slope is constant (or nearly so), indicating that a relationship is described by a straight line (or nearly so). Students are supposed to spot break points in relationships, and to find maximum and minimum points.

13.1.2 We want to understand the "range of possibilities."

Suppose we take the 40 yard dash times of nine 300 pound men from the University's football team. These times range from 5.9 seconds to 9.2 seconds.

Coaches want to know "how fast is this particular group of players?" As scientists, we are not usually asking the same question. We wonder, "how fast are players?" or "is this one group of players somehow different in performance from another?" We consider the possibilities that describe the larger set of "all possible players."

An exhaustive study of human physiology was performed on men that weigh 300 pounds or more and the relative likelihood of the various speeds is presented in Figure 13.3. One will note that 7 seconds is the most likely time, but there are plenty of times that are either slower or faster.

The head coach wonders if his assistants have done a good job of recruiting. Is his sample of players faster than one would expect than if we simply took big men at random and made them run 40 yards?

Figure 13.2: Some Curves

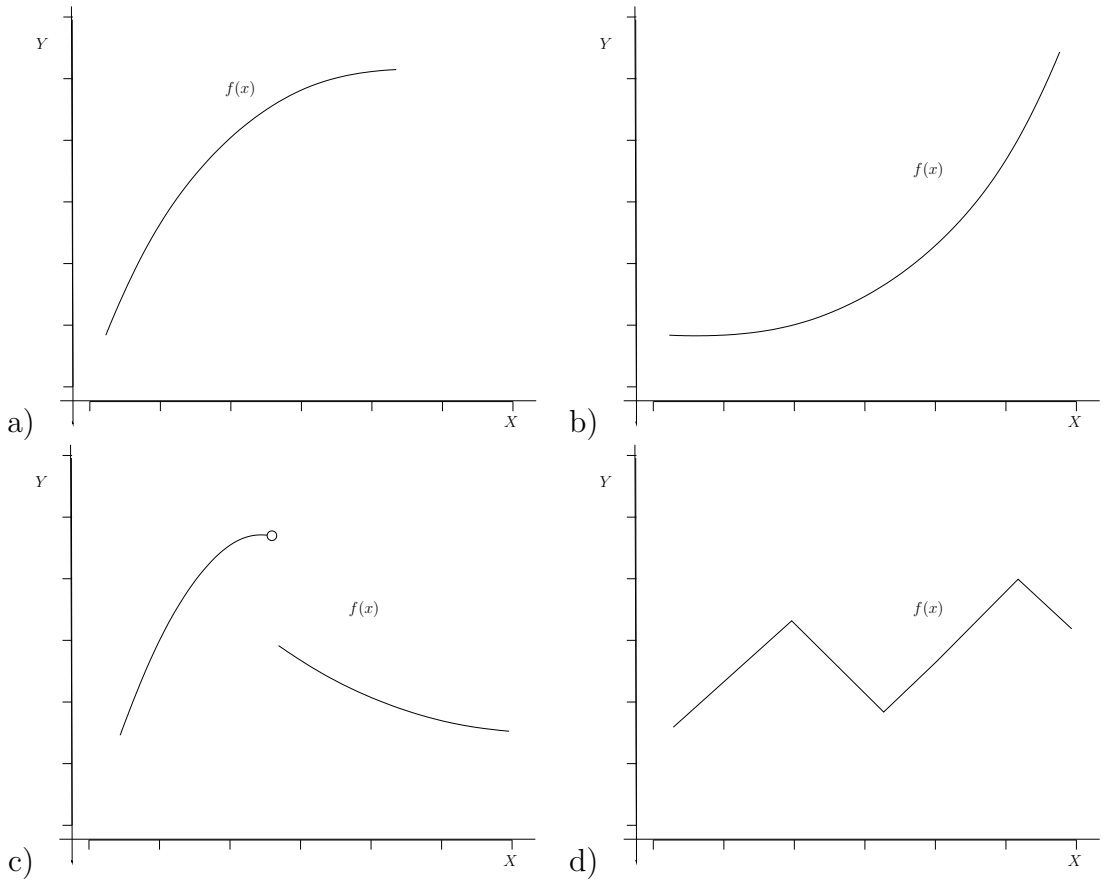


Figure 13.3: Speedy Big Men

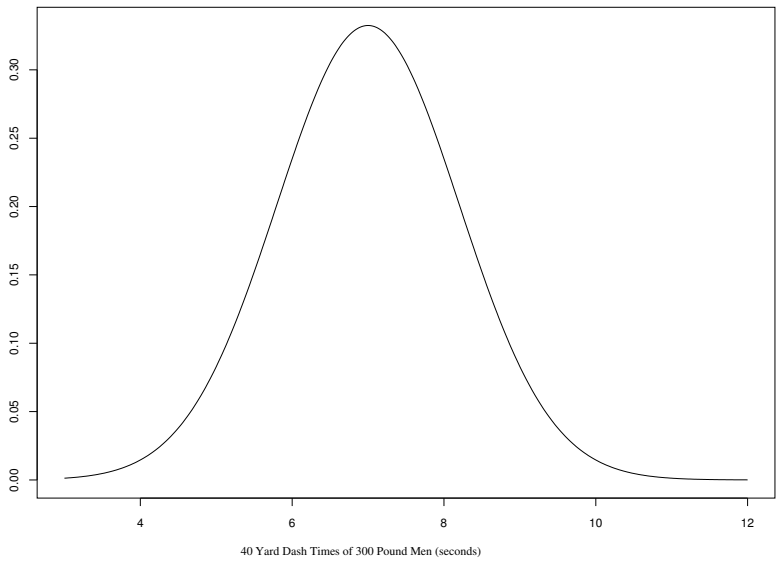
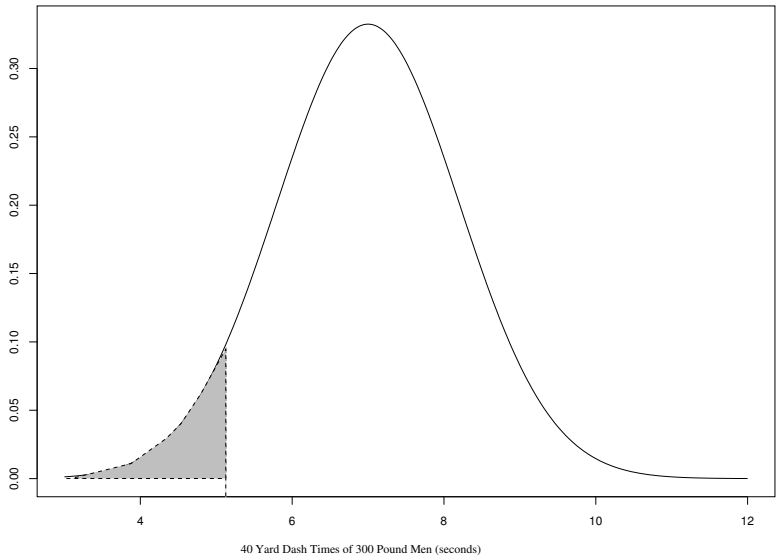


Figure 13.4: Chances of Finding a Fast Big Man



An integral is a measure of the "area under a curve." In the story of the big men, suppose we have one of the players run and the time measured is 5.3 seconds. That's certainly faster than average, but is that player a freak of nature? We really need to ask, "what is the chance that the 40 yard dash time of a randomly chosen big man would be smaller than 5.3 seconds?" That value would be the area under the curve in Figure 13.4.

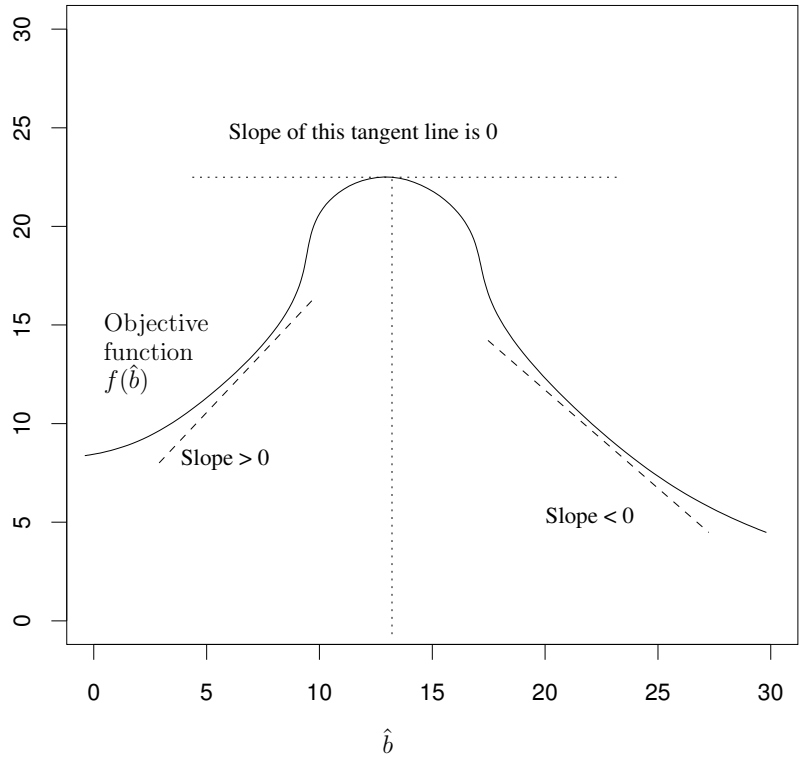
13.1.3 Optimizing things

One of the most fundamental uses of calculus is to find maximum or minimum points of functions. Maximum Likelihood statistical estimators are used to find the estimates that make the observed sample most likely to have occurred. A Least Squares regression line has coefficients that are chosen so as to make the squared errors as small as possible.

An optimization problem usually has 3 parts. First, there is an objective function. Second, there is a set of one or more variables that can be adjusted in order to achieve the desired maximum or minimum. In statistical problems, these adjustable variables are the "estimators." Third, there may be constraints that restrict the extent to which we can adjust the inputs.

An important characteristic of an optimum estimate is that the slope of the objective function is 0 at the optimum point. When the slope is 0, we are at the top of the hill. Once we have reached the top, we know that it is not possible to go any higher. In Figure 13.5, a problem of choosing the best value of $\hat{b}$ is illustrated. The best estimate is found by solving an equation that sets the slope of the function equal to 0. The slope of a tangent line is called a **derivative**.

Figure 13.5: Optimize: Choose the best $\hat{b}$



The essential nature of the solution of a one dimensional problem carries over to a problem in which we have more adjustable variables. Suppose we have an objective function that depends on two parameters, $\hat{b}_1$ and $\hat{b}_2$, as illustrated in Figure 13.6. The optimizing values of these estimates will be the ones that correspond to a tangent plane that is "flat" in both directions.

When an optimum point is found, it has the property that the slope of the function at that point is 0. At the very top of the hill, as it were, we are neither increasing nor decreasing. It turns out to be a little tricky sometimes because some functions have "flat spots" that are neither minima nor maxima, and so some double-checking is required to make sure we have actually found a critical point.

13.2 Slope and Derivative.

13.2.1 A straight line.

Consider the straight line illustrated in Figure 13.7.

This figure represents the Cartesian plane. A **point** is an ordered pair that is represented by a dot in the plane. For example, (x_1, y_1) is a point.

Figure 13.6: Choosing Optimal Values

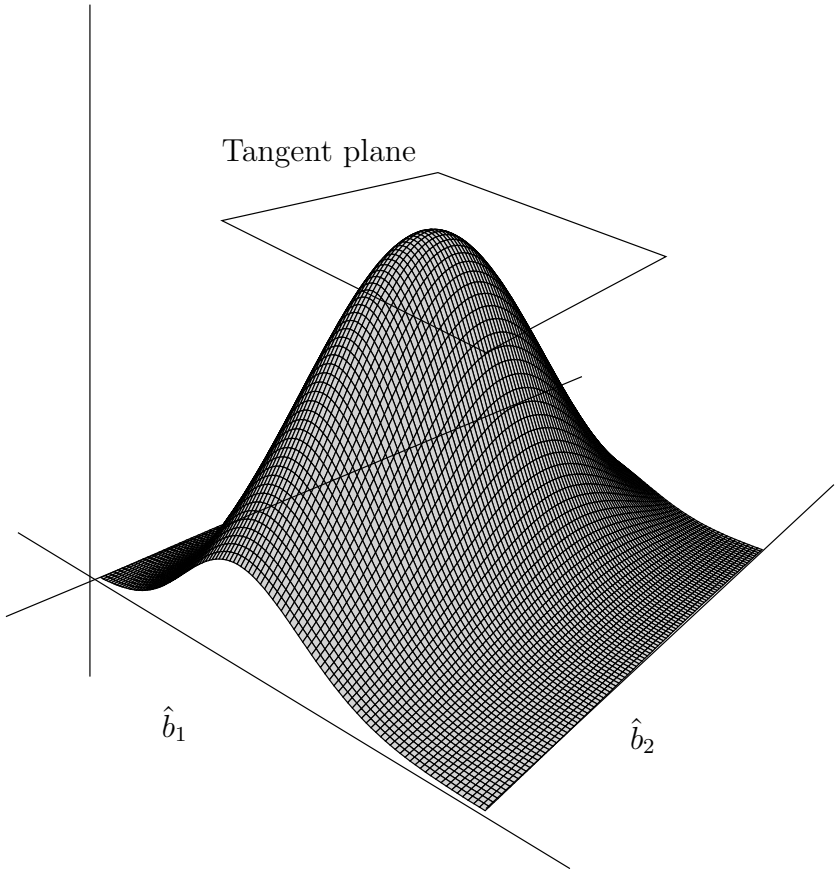
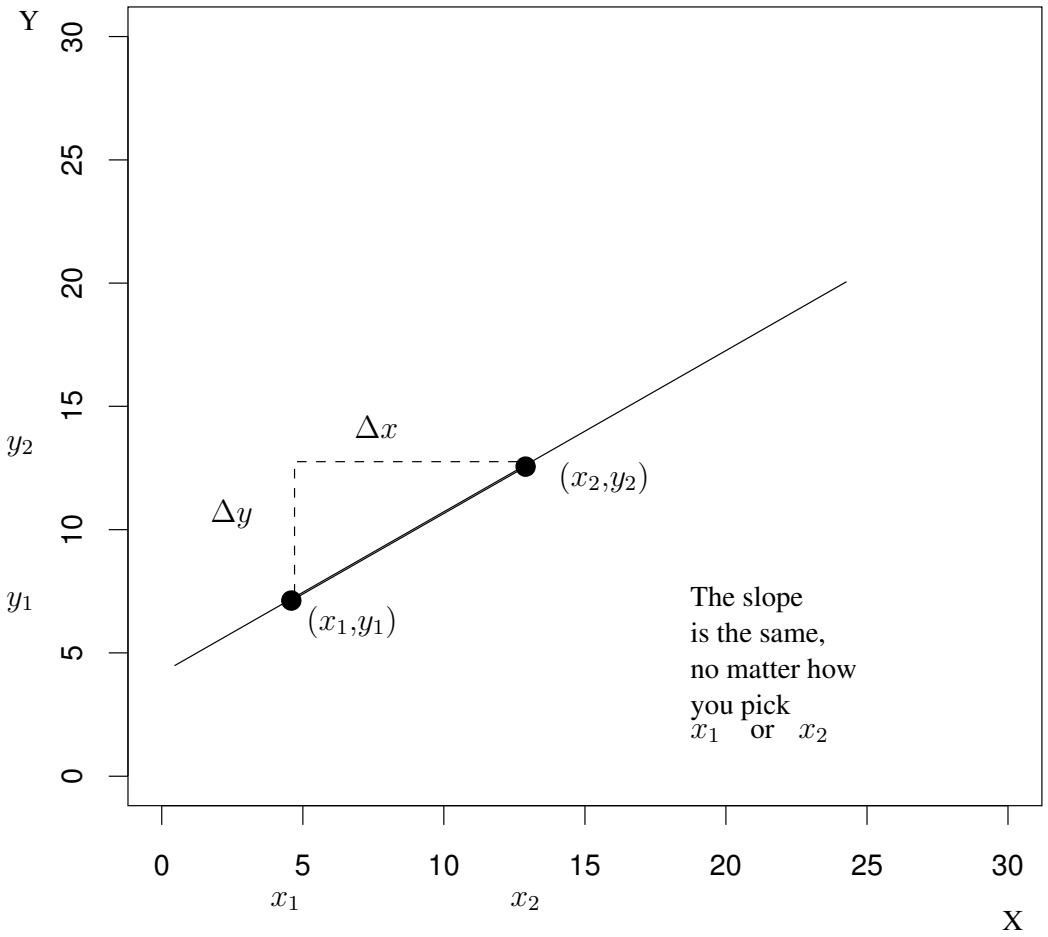


Figure 13.7: Slope of a straight line



Comparing two points (x_1, y_1) and (x_2, y_2):

the difference between them horizontally is $\Delta x = x_2 - x_1$.
the difference between them vertically is $\Delta y = y_2 - y_1$.

It is pretty easy to see the slope of the line is the ratio of the two changes.

$$slope = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} \tag{13.1}$$

This is "the rise over the run."

When you look at that figure, you see the slope of a straight line does not depend on how you select either x_1 or x_2. The slope is the same no matter where you measure it. That, of course, is the special feature of a straight line.

13.2.2 Slope is a local concept.

A "local" property is a property that is not "true everywhere." It is not a "global" property.

Consider the broken line in Figure 13.8.

The slope can't be meaningfully defined at the break point.

The calculation of the slope depends on "where you are," either on the left or the right of the break point.

13.2.3 Derivative: the slope of a "smooth curve".

Consider the smooth curve in Figure 13.9.

This figure is drawn so "x" is a particular point and values x_1, x_2, x_3 get closer and closer to x. Note that as x_i gets closer to x, the slope of the dotted line, $\frac{\Delta y}{\Delta x}$, gets steeper.

The term **derivative** is used to refer to the value of $\frac{\Delta y}{\Delta x}$ that is reached when Δx shrinks to nothing. As illustrated in part b of the figure, when Δx approaches 0, the dotted line keeps tilting until it is "just barely" touching $f(x)$. The dotted line is "tangent" to f at x. **The derivative is the slope of the tangent line**. In an example like this, the function is not jagged and it has no gaps, so the slope $\frac{\Delta y}{\Delta x}$ smoothly approaches a limiting value as Δx shrinks to zero. The limiting value of $\frac{\Delta y}{\Delta x}$ is the slope of the line that is tangent to $f(x)$ at x.

The the slope of this dotted line where Δx reaches 0, or comes "arbitrarily close" to it.

Figure 13.8: Slope is a Local Concept

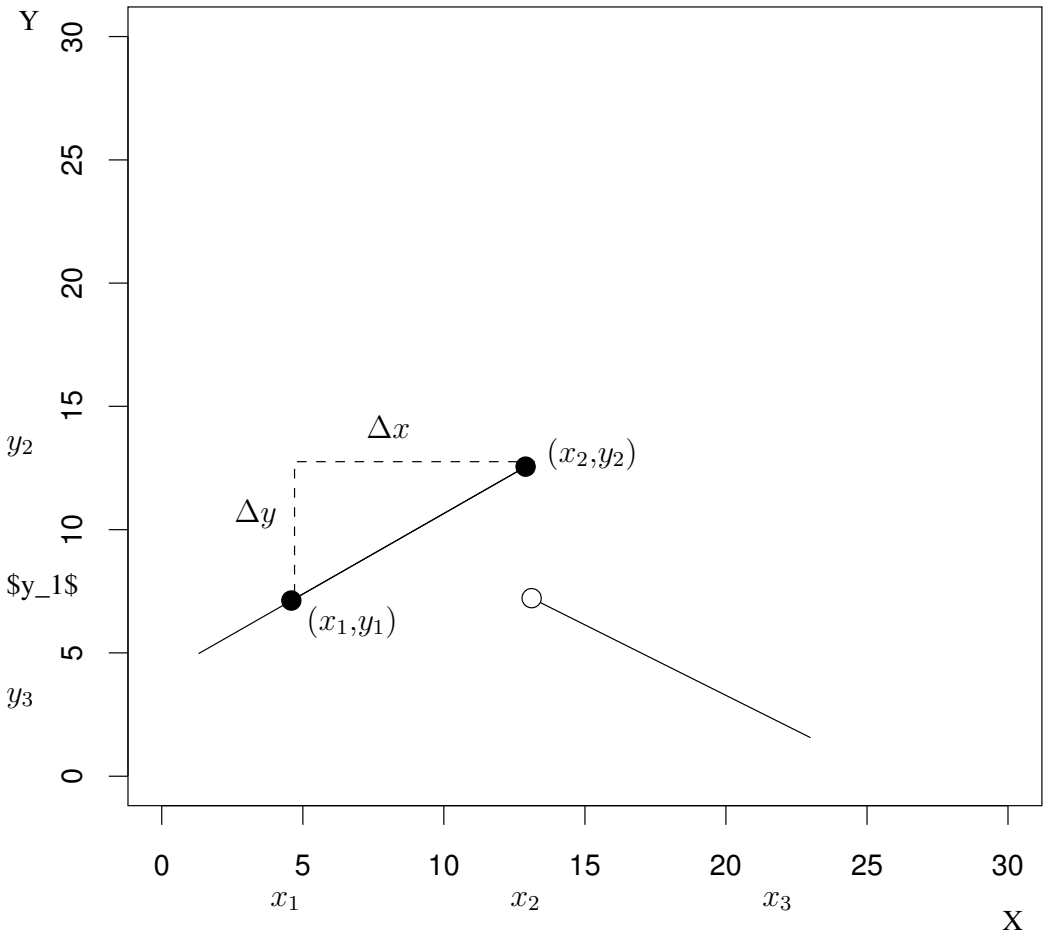
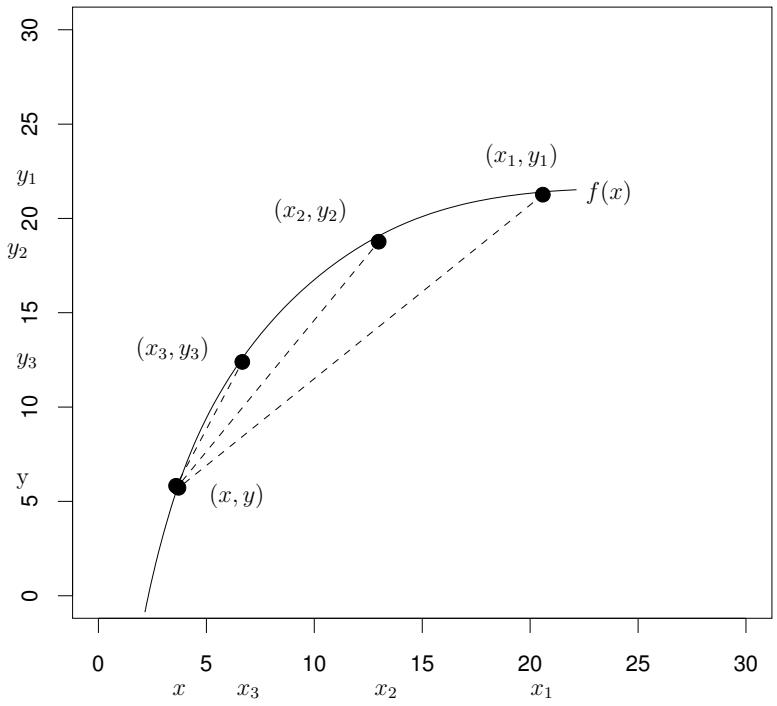
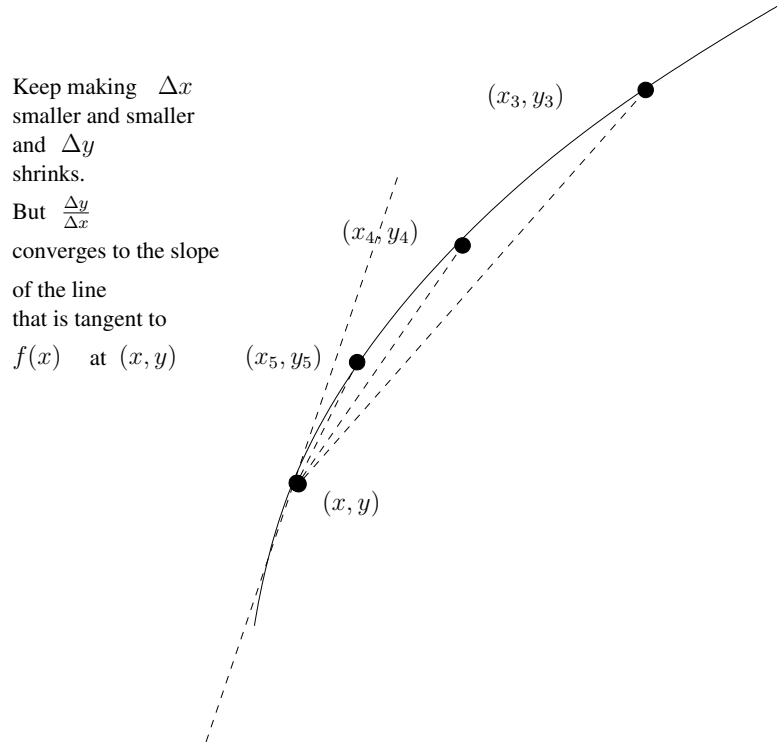


Figure 13.9: The Derivative

(a) Make Δx smaller and smaller



(b) Approaching Tangency



13.2.4 Derivative notation

There are various notations people use for derivatives. Since this is an executive summary, intended to assist with your "overall understanding," this may be the most important part of this chapter. Even a brief venture out into the literature will unearth a multitude of different kinds of notation for differential calculus. Since all of the different kinds of notation are not going to disappear, it is important to become accustomed to them.

One classic notation is:

$$\frac{df(x)}{dx} \tag{13.2}$$

or, if it is already known that $y = f(x)$ then:

$$\frac{dy}{dx} \tag{13.3}$$

or sometimes people don't want to refer to y, so they say "f prime of x", as in:

$$f'(x) \tag{13.4}$$

or if you are an "operator" minded person, think of D as the derivative operator:

$$Df(x) \tag{13.5}$$

13.2.4.1 Brief detour to formal definition

In order to formally define this concept, the idea of a limit must be introduced. A limit is used to describe the idea that when an input variable x gets "very close" to some particular value, x_0, then the output variable $f(x)$ may get "very close" to a particular value. The notation for this is

$$lim_{x \to x_0} f(x)$$

Of course, if $f(x_0)$ is defined–meaning it can be calculated–then the limit is easy to calculate. However, it may be that $f(x_0)$ can't be calculated, and yet the limit may exist. For example, everybody knows that $1/x$ gets smaller as x gets larger. The value $1/\infty$ is not defined. However, the limit of $1/x$ is defined, and it is equal to 0. It is the value toward which $1/x$ tends, even though it never actually "gets there" (because we never actually "reach" infinity).

$$lim_{x \to \infty} \frac{1}{x} = 0 \tag{13.6}$$

This example illustrates why the idea of a limit is useful. The value of $1/x$ is never exactly 0, but it gets closer and closer to 0. Thus, 0 is a "limiting value" of $1/x$ as x "goes to" infinity.

The formal definition of a derivative uses the concept of a limit. Consider a function at some point, $f(x)$. If we add "just a bit" to x, we arrive at $x + \Delta x$. So the change in y that results from that change is

$$\Delta y = f(x + \Delta x) - f(x) \tag{13.7}$$

The slope of the tangent line–the derivative–is defined only if the following limit exists:

$$lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{13.8}$$

13.2.5 Derivative factoids

Differential calculus is, by far, the funnest part. There are about 15 rules, and if you know these rules, you can get the derivative of any function. I've made a brief list of the most useful rules. I don't expect every executive who reads this memo will memorize them. Rather, I list them out on the off chance that a teacher somewhere will read this memo. I have to try to keep up appearances in front of my peers!

13.2.5.1 Linearity.

The derivative of a sum is the sum of the derivatives,

$$\frac{d}{dx}\{f(x) + g(x)\} = \frac{d}{dx} f(x) + \frac{d}{dx} g(x) = f'(x) + g'(x) \tag{13.9}$$

and the derivative of a constant times a function is equal to the constant times the derivative of the function.

$$\frac{d}{dx} a \cdot f(x) = a \cdot \frac{d}{dx} f(x) = a \cdot f'(x) \tag{13.10}$$

As a result, more complicated expressions can be "broken down" easily. For example

$$\frac{d}{dx}\{a \cdot f(x) + b \cdot g(x)\} = a \frac{d}{dx} f(x) + b \frac{d}{dx} g(x) \tag{13.11}$$

or

$$\frac{d}{dx}\sum_{i=1}^{N} f_i(x) = \sum_{i=1}^{N} \frac{d}{dx} f_i(x) = \sum_{i=1}^{N} f_i'(x) \tag{13.12}$$

13.2.5.2 Powers of x.

1. Slope of line.

   If

   $$y = bx \tag{13.13}$$

   then

   $$\frac{dy}{dx} = b \tag{13.14}$$

   Think backwards for a minute. You usually think of b as a constant, but change gears for a minute to notice
   $\frac{dy}{db} = x$
   (but that's just a digression)

2. Slope of a square.
   If

   $$y = x^2 \tag{13.15}$$

   then

   $$\frac{dy}{dx} = 2x \tag{13.16}$$

   That is one of the easiest ones to prove and "really believe." Use the definition:

   $$\frac{dy}{dx} = lim_{\Delta x \to 0} \frac{(x + \Delta x)^2 - x^2}{\Delta x} \tag{13.17}$$

   $$= lim_{\Delta x \to 0} \frac{x^2 + 2x\Delta x + (\Delta x)^2 - x^2}{\Delta x} \tag{13.18}$$

   $$= lim_{\Delta x \to 0} \frac{2x\Delta x + (\Delta x)^2}{\Delta x} \tag{13.19}$$

$$= lim_{\Delta x \to 0} \frac{(2x + \Delta x) \cdot \Delta x}{\Delta x} \tag{13.20}$$

$$= lim_{\Delta x \to 0} 2x + \Delta x \tag{13.21}$$

$$= 2x \tag{13.22}$$

Anyway, I believe in that, and take a lot of the rest on faith (insert smiley face here please).

3. Slope of a cube.
   If
   $$y = x^3$$
   then
   $$\frac{dy}{dx} = 3x^2$$

4. Slope of $y = x^4$.
   $\frac{dy}{dx} = 4x^3$

5. Slope of $y = x^{-1} = 1/x$
   $\frac{dy}{dx} = -1x^{-2}$

6. Slope of $\sqrt{x} = x^{1/2}$
   $\frac{dy}{dx} = \frac{1}{2}x^{-1/2}$

You start to notice a general pattern?

$$\frac{d}{dx}x^N = N \cdot x^{N-1} \tag{13.23}$$

That is true, whether N is a whole number or a fraction.

13.2.5.3 Logarithms

The only derivative I remember from the top of my head is the one for the natural logarithm.

$$\frac{d}{dx}ln(x) = \frac{1}{x} \tag{13.24}$$

It can't get any easier than that.

If you have a log to a different base, say $log_{10}()$, the derivative involves a "constant of proportionality." But I never remember it, I always have to look it up in the calculus book.

So wait a minute while I go look in the book.

Ah. Here:

$$\frac{d}{dx}log_b(x) = \frac{1}{ln(b)} \cdot \frac{1}{x} \tag{13.25}$$

13.2.5.4 Exponentials

Recall Euler's constant, e. It is the base of the natural logarithm. Instead of $ln(x)$ sometimes people write $log_e(x)$, just so you remember the base is the special number. One way to define Euler's constant is by declaring it to be the "magic number e" such that the derivative of $log_e(x)$ equals $1/x$.

Recall the notation:

$$e^x = exp(x) \tag{13.26}$$

The derivative is:

$$\frac{d}{dx}e^x = \frac{d}{dx}exp(x) = exp(x) \tag{13.27}$$

In other words, you "get the same thing back".

As in the case of the logarithm, if you are taking powers of some number besides e then a constant of proportionality enters the picture. The book says

$$\frac{d}{dx}b^x = ln(b) \cdot b^x \tag{13.28}$$

Note that $ln(e) = 1$, so if you set the base, b, equal to e, then this would reduce to the derivative of $exp(x)$.

13.2.5.5 Derivative of a product

$$\frac{d}{dx}\{g(x) \cdot h(x)\} = \frac{d}{dx}g(x) \cdot h(x) + g(x)\frac{d}{dx} \cdot h(x) \tag{13.29}$$

or, if you like primes,

$$\frac{d}{dx}\{g(x) \cdot h(x)\} = g'(x)h(x) + g(x) \cdot h'(x)$$

13.2.5.6 Function of a function

The **chain rule** states that:

$$\frac{d}{dx}\{f(g(x))\} = \frac{df}{dx}\mid_{g(x)} \cdot \frac{dg(x)}{dx} \tag{13.30}$$

That's the derivative of $f(x)$ calculated at the location given by the value $g(x)$, multiplied by the derivative of $g(x)$. Confusing enough? Probably.

Suppose, for example, you had

$$g(x) = x^2$$

and

$$f(x) = ln(x)$$

so

$$f(g(x)) = ln(x^2)$$

$$\frac{d}{dx}f(g(x)) = \frac{1}{x^2} \cdot 2x$$

13.3 Optimization

One of the most important uses for derivatives is in the search for optimal values of variables. Optimal may be a maximizing value or a minimizing value, depending on the way a problem is presented.

13.3.1 First Order Conditions

The single most important use of derivatives is to check to see if $f(x)$ is a local maximum or minimum. Consider the slope if we approach a maximum point from the left. The derivative is positive, but it gets smaller and smaller until it is equal to 0. When

$$f'(x) = 0 \tag{13.31}$$

it means we have found the exact "top of the hill."

Similarly, if we are searching for a local minimum, and we approach that minimum from the left, the derivative will be negative, but it will gradually become larger and eventually it will be exactly zero when we find the "bottom of the bowl."

This leads to an obvious kind of exercise. Formulate an objective function, look for a maximum or a minimium. This is the search for so-called "first order conditions." When the derivative is equal to 0, we have a "critical point." The derivative, by itself, does not tell us if we are at a minimum or maximum. Additional work is needed to address that detail (see below on "second order conditions").

In any kind of statistics or economics or physics book, a lot of time/effort is spent translating something into a function that depends on or more variables. Those variables, the "optimizing values" are assumed to exist where

$$f'(x) = 0.$$

13.3.2 Second-order conditions

When we have solved the "first order condition," found a point at which the slope is 0, we then face the challenge of finding out if we have found a maximum or minimum point.

13.3.2.1 Second derivatives.

The second derivative is the change in the slope if you start at x and go an "itty bitty" amount to the right. It is useful mainly because it offers us a clue about whether we are at the top of a hill or in the bottom of a bowl.

If the slope is **getting bigger**, that means the impact of x is "accelerating".

If the slope is **getting** smaller, that means the impact of x is "decelerating" or has "diminishing impact".

Sometimes notations for the change in the slope are:

$$f''(x) = D^2 f(x) = \frac{d^2 y}{dx^2} = \frac{d^2 f(x)}{dx^2} \tag{13.32}$$

I was trained by people who prefer this kind of notation, $f''(x)$.

13.3.3 Did you find a minimum or a maximum.

If you find the value of x for which $f'(x) = 0$, then you are either at a maximum or a minimum. (I mean "locally", in the immediate vicinity of x.)

Think for a minute about each of these claims.

1. If $f'(x) = 0$ and $f''(x) < 0$, then x is a local maximum. f is "concave down" at that point.

2. If $f'(x) = 0$ and $f''(x) > 0$, then x is a local minimum. f is "concave up" at that point.

If $f''(x) < 0$, then the function is "decelerating" as we reach the top of the hill. If $f''(x) > 0$, then the function heading downwards into the bottom of a bowl.

The terms "concave up" and "concave down" are used to describe points at which we have found minima or maxima.

13.4 Functions of several variables.

13.4.1 Functions of several variables.

Suppose you have 3 input variables, x_1, x_2, and x_3.

$$y = f(x_1, x_2, x_3)$$

The calculus of many variables can get really complicated, but most of the time it is really simple.

A **partial derivative** is the change in $f(x_1, x_2, x_3)$that results when all of the variables are being held constant except one. The most common notation for the partial derivative is

$$\frac{\partial y}{\partial x_1} \tag{13.33}$$

Because it is tedious to type that fraction all of the time, you sometimes see authors inventing convenient notation for partial derivatives. My personal favorite is to use a subscript to tell which variable I'm allowing to change:

$$f_1(x_1, x_2, x_3)$$

That is supposed to be the same as 13.33

13.4.2 Finding Optima

If you are given a function like $f(x_1, x_2, x_3)$ and you are instructed to find the maximum or mimimum, the first thing you do is find the place where ALL OF THE PARTIAL DERIVATIVES equal 0. That is, solve this system of equations:

$$\frac{\partial y}{\partial x_1} = 0$$

$$\frac{\partial y}{\partial x_2} = 0$$

$$\frac{\partial y}{\partial x_3} = 0$$

Note, you could as well think of this as a matrix of derivatives:

$$D = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \frac{\partial y}{\partial x_3} \end{bmatrix} = 0 \tag{13.34}$$

One tires quickly of writing down 3 rows of derivatives over and over, so one often just refers to this condition for a maximum or minimum as $D = 0$.

13.4.3 Second order conditions

In the calculus of one variable, it is easy to tell if one has found a maximum or a minimum from the second derivative.

It is not so simple in the calculus of several variables. It is easy to calculate a second partial derivative of $f()$with respect to x_1

$$\frac{\partial^2 y}{\partial x_1 \partial x_1} = \frac{\partial^2 y}{\partial^2 x_1}$$

And one can also find the partial of $\frac{\partial y}{\partial x_1}$with respect to another variable, say x_2.

$$\frac{\partial^2 y}{\partial x_1 \partial x_2}$$

I prefer short hand notation like $f_{11}()$ or $f_{12}()$ for these.

Anyway, suppose you begin with the matrix of first partials. You can differentiate each item by each of the 3 variables, so that means you can build up a 3x3 matrix of second partial

derivatives like so:

$$D' = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 y}{\partial x_1 \partial x_1} & \frac{\partial^2 y}{\partial x_1 \partial x_2} & \frac{\partial^2 y}{\partial x_1 \partial x_3} \\ \frac{\partial^2 y}{\partial x_2 \partial x_1} & \frac{\partial^2 y}{\partial x_2 \partial x_2} & \frac{\partial^2 y}{\partial x_2 \partial x_3} \\ \frac{\partial^2 y}{\partial x_3 \partial x_1} & \frac{\partial^2 y}{\partial x_3 \partial x_2} & \frac{\partial^2 y}{\partial x_3 \partial x_3} \end{bmatrix} \tag{13.35}$$

This is the so-called **Hessian matrix.** There are various conditions that can be set so that one can diagnose the question of whether a maximum, a minimum, or neither, has been found.

The one that sticks in my mind is the idea of a "positive definite" matrix. Take a $3x1$ column vector z. It is supposed to represent a small change from the location where one currently is, (x_1, x_2, x_3). Calculate the quantity:

$$z' \cdot D' \cdot z \tag{13.36}$$

or, more verbosely,

$$[z_1, z_2, z_3] \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{13.37}$$

There's a theorem that says:

If $z'D'z > 0$, then D' is a positive definite matrix.

If $z'D'z < 0$, then D' is a negative definite matrix.

Use these ideas to check if you have found a maximum or a minimum. Find the point where all the partials are 0, call it $x*$ and then evaluate the Hessian at that point.

If the Hessian is positive definite, you have found a minimum point.

If the Hessian is negative definite, you have found a maximum point.

Note this is entirely similar to the univariate case, where $f''(x) > 0$ means you have a minimum and $f''(x) < 0$ means you have a maximum.

13.5 The Integral

Even if you never took a calculus class, you may still have to read a book that uses integrals. The concept is not complicated, and as long as you don't actually have to solve some integrals, I expect a conceptual understanding is good enough for most people.

13.5.1 Definition

The elongated S symbol represents integration.

$$\int_a^b f(x)dx$$

This means "the total area under the curve $f(x)$ between a and b." This is illustrated in Figure 13.10.

The symbol dx represents the "dummy variable of integration." It is a signal that you are supposed to move along the x axis when you sum up from a to b.

13.5.2 Throw More Words and Beautiful Graphs at it!

Maybe that "area under a curve" thing is too blunt. Perhaps a "word problem" will help. There's no math topic that doesn't sound better with a problem that if it starts with "A car is driving Eastward at 50 miles per hour...."

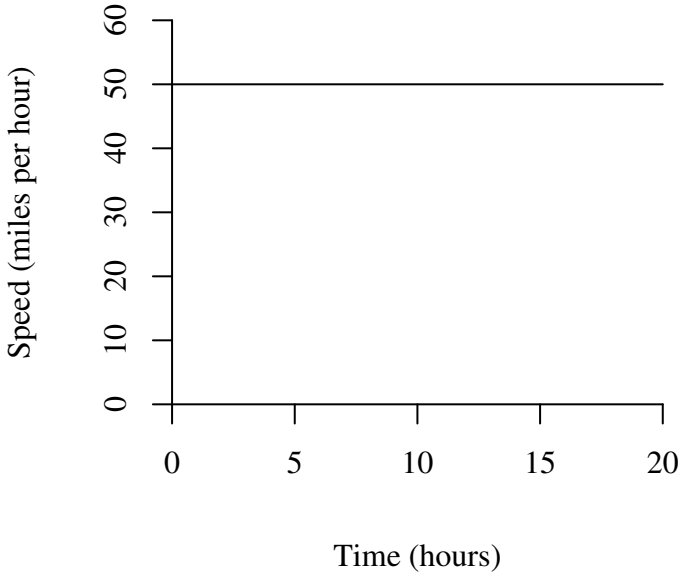
Trust me. I made some nice illustrations with R. The least you could do is to look at them.

Suppose you are driving on the highway going 50 miles per hour. Of course, after 1 hour, you have gone 50 miles. After 2 hourse, you have gone $2 \times 50 = 100$ miles. Generally, the "distance traveled" is t (which represents time) multiplied by the speed (in miles per hour). In Figure 13.11, the speed is depicted in the top part of the figure. That part is boring because the speed is constant. The bottom part of the figure, demonstrates the distance traveled.

Perhaps the reader has already noticed that the distance traveled, which I have labeled $F(t)$ in the bottom part of the figure, is equal to $t \times 50$, which is the area under the curve in the top part of the figure between time 0 and t. The value of $F(t)$ represents a "cumulative distance traveled." We know, for example, that at time 10, the distance traveled is 500 miles. And it is also easy to see that the distance traveled between hours 5 and 10 is $F(10) - F(5) = 250$.

The speed and distance graphs for a different car & driver are illustrated in Figure 13.12. The story is basically the same, except that the driver's speed, $f(t)$, is not constant. Rather, perhaps because of traffic or scenery, the driver's speed flucturates. If we wonder "how far has the car traveled before time t" it is not so easy to tell. But the car's odometer still works, and so the distance traveled can be seen in the bottom panel. The distance traveled after 10 hours, $F(10)$, is 446.02, and we can see that the trip's progress from 5 to 10 hours, $446.02 - 249.07 = 196.95$ miles, is quite a bit better than the progress betwen 10 and 15 hours, $543.61 - 446.02 = 97.59$ miles.

Figure 13.10: The Area Under a Curve between a and b



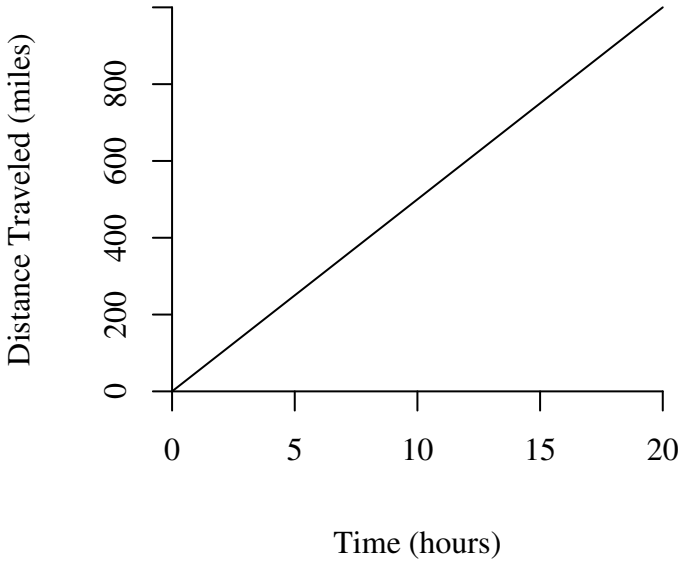
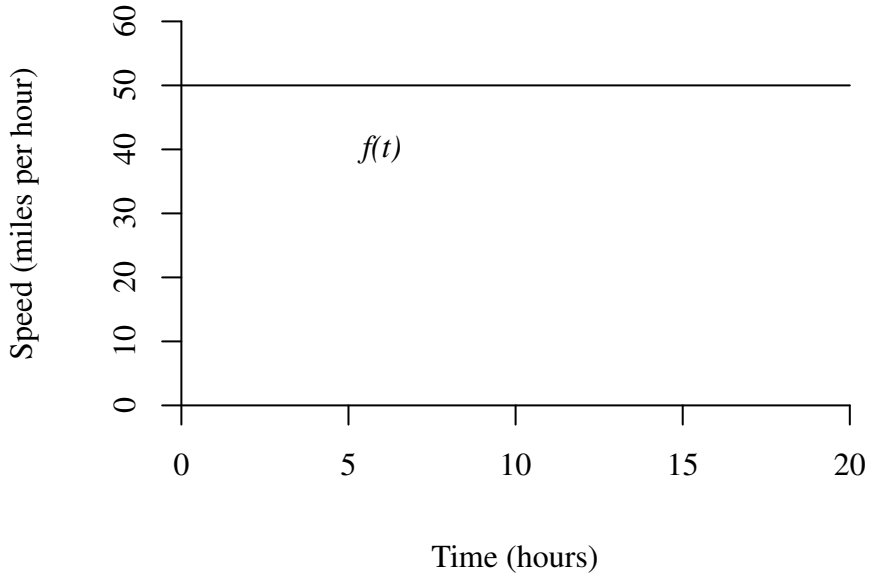


Figure 13.11: Driving at a Constant Speed

(a) The speed at time t is represented by f



(b) Accumulated speed is distance travleled, F

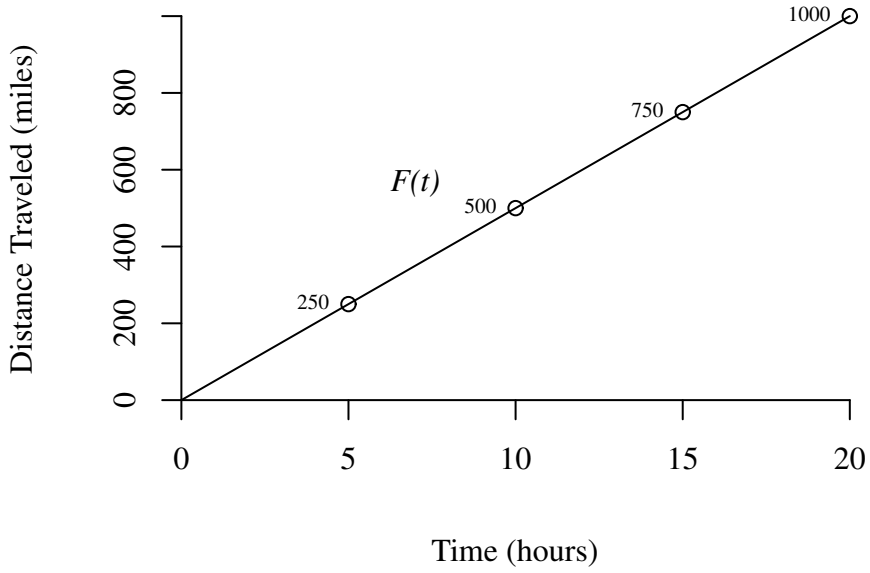
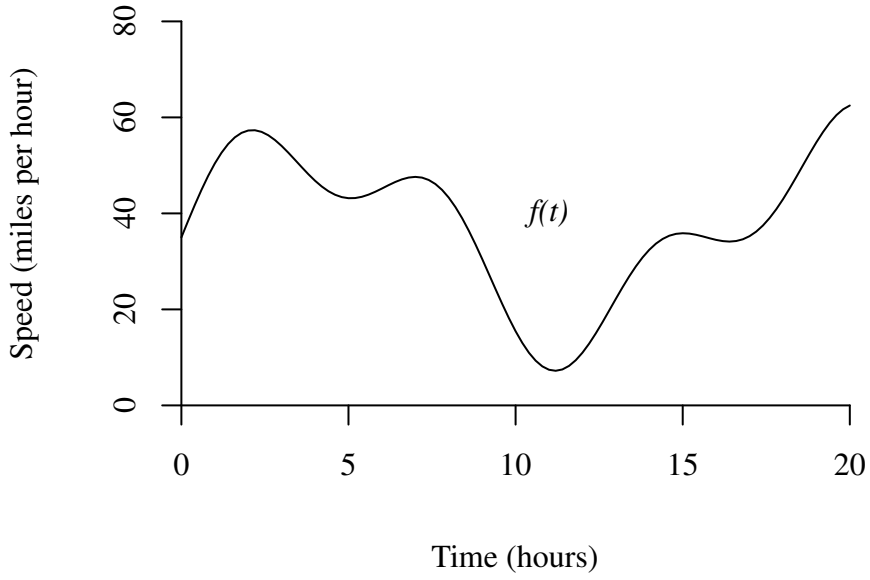


Figure 13.12: Driving at a Variable Speed

(a) The speed at time t is represented by f



(b) Accumulated speed is distance travleled, F

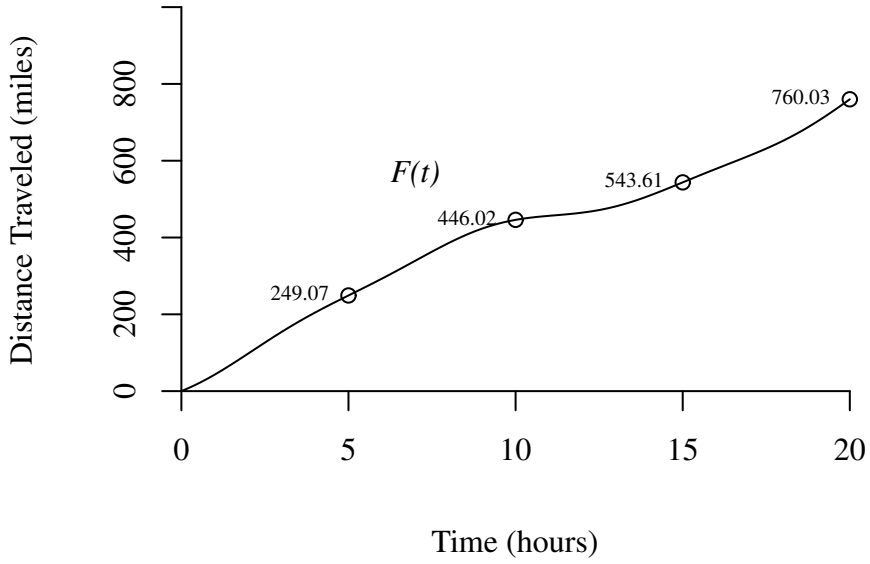
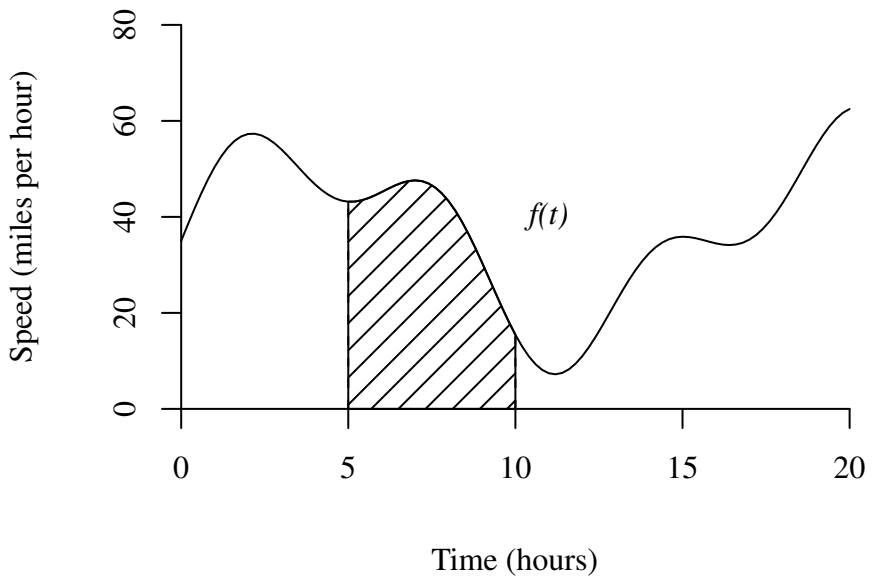


Figure 13.13: Distance Traveled Between Hours 5 and 10



The distance traveled between hours 5 and 10 is the focus of Figure 13.13. We are confident that the distance traveled, 196.95 miles, is the area under the curve displayed in this figure. Using the integral notation,

$$\int_5^{10} f(t)dt = 196.95 \tag{13.38}$$

And that area is equal to $F(10) - F(5)$.

In this way, the functions $f(t)$ and $F(t)$ are linked together. From the preceeding example, one expects the following would be generally true. For points a and b,

$$\int_a^b f(t)dt = F(b) - F(a) \tag{13.39}$$

The key to finding the area under a curve, then, hinges on finding the correct expression for the accumulator function $F(t)$. The Fundamental Theorem of Calculus states that the previous expression is true if $F(t)$ is differentiable and

$$\frac{dF(t)}{dt} = F'(t) = f(t)$$

.

I have struggled a bit in developing an understandable explanation this. It is important to

believe not only that the area under $f(t)$ between a and b can be calculated as $F(b) - F(a)$, but also that $f(t)$ is the derivative of $F(t)$. At this stage of my Executive Summary, I choose to convey the "flavor" rather than a rigorous proof. Consider the top panel of Figure 13.14, where the difference between the "area under the curve" and the "area of the rectangle below it" are compared. Begin by taking two points, a and $b = a + \Delta t$. The area of the rectangle $(\Delta t \cdot f(b))$ is easily seen to be smaller than the area under the curve. Similarly, in the bottom panel of the figure, the area of the rectangle $(\Delta t \cdot f(a))$ is greater than the area under the curve. For just a moment, pretend you are Goldilocks in the home of the three bears. If $f(a)$ gives a rectangle that is "too big," and if $f(b)$ gives a rectangle that is too small, then there must be some value in between $f(a)$ and $f(b)$ that would be "just right" (an application of the Mean Value Theorem). If f is continuous and defined between a and b, so we can fish for a value $a \leq c \leq b$ so that $\Delta t \cdot f(c)$ equals the true area under the curve.

The relationship between these areas is thus:

$$small\,rectangle \leq \quad area\,under\,f\,between\,a\,and\,b \quad \leq large\,rectangle \tag{13.40}$$

$$f(b) \times \Delta t \leq \quad F(a + \Delta t) - F(a) = f(c) \times \Delta t \quad \leq f(a) \times \Delta t \tag{13.41}$$

Now consider the changes that occur when Δt becomes smaller. As $\Delta t \to 0$, then $b \to a$ and $c \to a$. Similarly, $lim_{\Delta t \to 0} f(c) = f(a)$ and $lim_{\Delta t \to 0} f(b) = f(a)$. All three of the rectangles would converge to the same size.

Now consider the middle expression:

$$F(a + \Delta t) - F(a) = f(c) \times \Delta t \tag{13.42}$$

Divide both sides by Δt.

$$\frac{F(a + \Delta t) - F(a)}{\Delta t} = f(c) \tag{13.43}$$

The left hand side is starting to look like a derivative.

$$lim_{\Delta t \to 0} \frac{F(a + \Delta t) - F(a)}{\Delta t} = lim_{\Delta t \to 0} f(c) \tag{13.44}$$

$$\frac{dF(a)}{dt} = f(a)$$

In words, the derivative of $F(t)$ at point a equals the value of $f(t)$ at a.

Figure 13.14: Integral Area

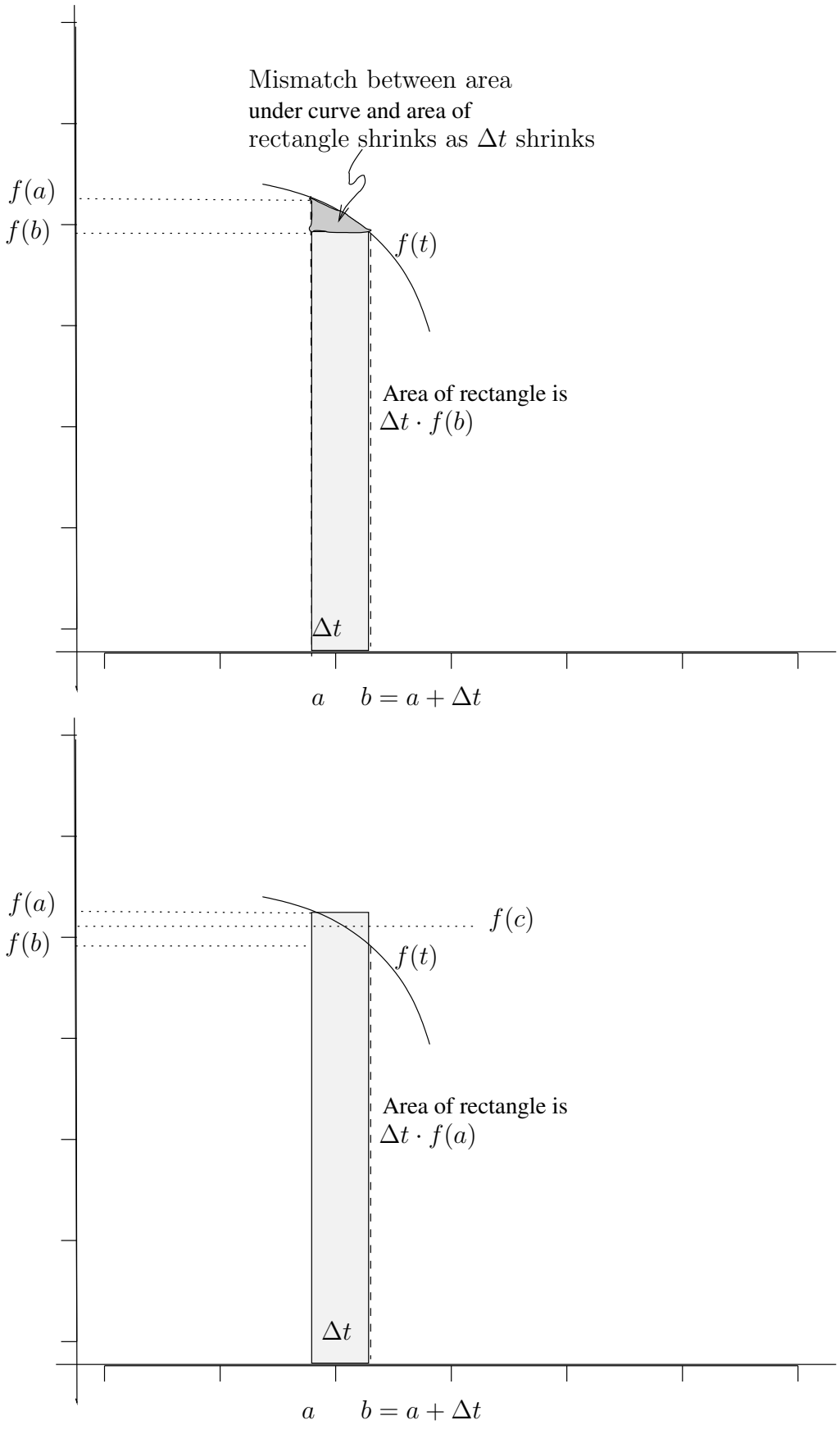
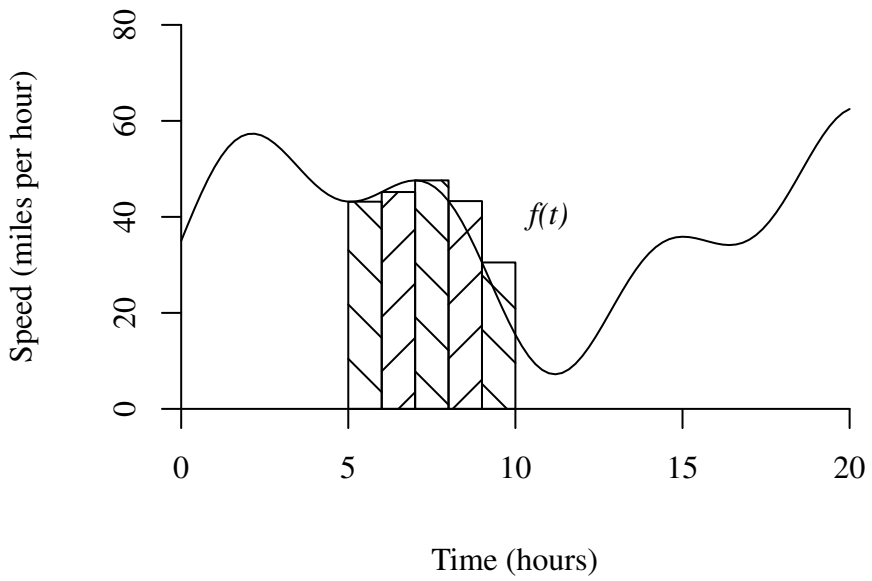


Figure 13.15: Approximating the Area by Rectangles



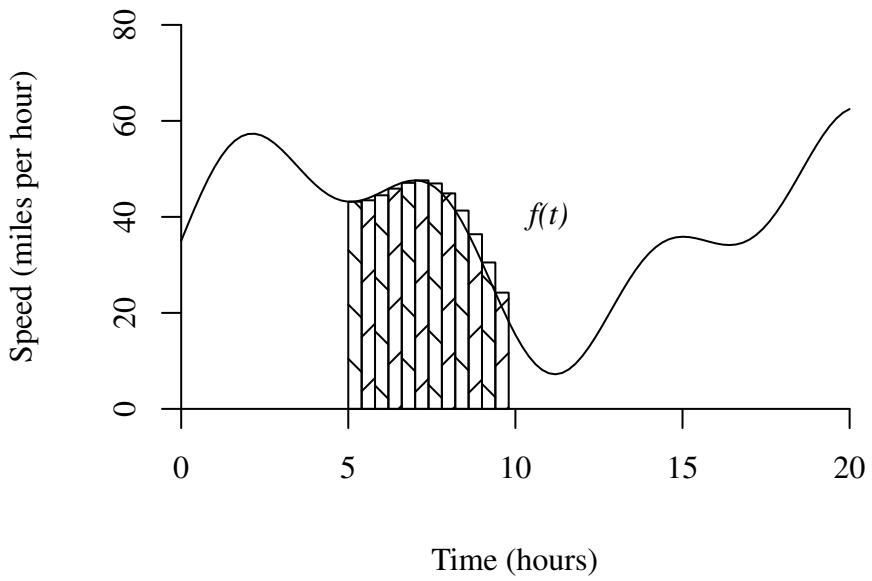
The conclusion is that we find the "magic function" F by thinking backwards. $F(t)$ is the function which, when differentiated, would be equal to $f(t)$. The slope of $F(t)$ is $f(t)$. For this reason, $F(t)$ is known as an "anti-derivative" of $f(t)$.

13.5.3 Approximating integrals numerically

I mentioned above that I think differentiation is the fun part of calculus. I think integration is the hard part. It is hard because it requires one to "think backwards," to try to deduce a function that, when differentiated, would yield another function. Some functions are simple, but most are not. Finding a simple, workable expression for F is not always possible. As a result, although the area under any curve is given exactly in theory by the formula in 13.39, it is necessary to find approximations for practical situations.

Fortunately, the high speed computing helps quite a bit. Almost everybody has the intuition to try to calculate the area under a curve by breaking the region into rectangles, and then summing their areas. Consider the example in Figure 13.15. Since the area of a rectangle is simply $width \times height$, it would be easy to use a ruler and a pencil to figure out an approximation for the area. One will observe that the rectangles do not always approximate the desired area closely, and one can make the rectangles smaller in order to obtain a more accurate estimate. See Figure 13.16.

Figure 13.16: Approximating the Area by More Precisely



As the width of the bars shrinks, the accuracy of the estimated area improves. However, that accuracy is bought at the expense of computational effort. There are several competing methods that attempt to make the approximation as close as possible to the correct value while minimizing the amount of computation required. The general idea is that we begin with wide bars, and then we replace them with smaller bars where the value of the function is rapidly changing. Because digital computers have been available only for a comparatively short time, the computer software for this is still developing rather rapidly. In case you are ever wading through a statistics manual and find references to the "Laplace approximation" of an integral or "adaptive Gaussian quadrature," then you will know you have run into this problem.

13.5.4 Integrals & Probability

If you are studying a continuous random variable, x, it means you are studying a variable that can take on real values in some domain, X. Suppose the "endpoints" of X are $left$ and $right$, where $left$ and $right$ can take on any real value, as well as infinity.

Real number digression: The symbol for the "real number line" is $\mathbb{R}$. The set of all real numbers is "all numbers that can be written down as numbers with decimals, including those with an infinite number of digits after the decimal point." The real numbers are a

closed system, in the sense that if a and b are real, then $a \times b$ is also real and if a/b is defined, then it is real as well.

If $f(x)$ is a **probability density function** (pdf) representing a probability distribution, it means these 2 conditions are true:

1. $f(x) \geq 0$ for all $x \in X$
2. $\int_{left}^{right} f(x)dx = 1$.

There is always some "tricky business" about the probability of a particular, individual point. The probability that a particular point will occur is 0 because a point is a thing with no "width." The probability that you could observe a particular outcome c is

$$\int_{c}^{c} f(x)dx$$

And that is always 0 by definition. So we are restricted to talking about outcomes in a particular range, say between c and d.

$$\int_{c}^{d} f(x)dx$$

The **cumulative distribution function** (cdf) is the probability that the outcome of a random draw will be smaller than some given value, say k. It is often symbolized by a capital letter corresponding to the pdf, in this case $F(k)$. Formally, it is the integral from $left$ up to the point k.

$$F(k) = \int_{left}^{k} f(x)dx$$

13.5.5 Integrals for Multivariate Probability

If the domain of outcomes is 2 or more dimensional, then the integral extends to represent it. For example, with 2 dimensions, $X = \{left_x, right_x\}$ and $Y = \{left_y, right_y\}$. The pdf is $f(x, y)$ and the probability that an observation will occur in a region in which $x \in \{c_x, d_x\}$ and $y \in \{c_y, d_y\}$ is represented as

$$\int_{c_x}^{d_x} \int_{c_y}^{d_y} f(x, y)dxdy$$

There's a theorem that say the order of integration does not affect the value, so you can swap the x and y things in there.

Generally, it is difficult to solve multivariate probability distributions, so we go searching about for simplifying results, such as independence,

$$f(x, y) = g(x) \cdot h(y)$$

This means that the joint observation of the pair (x, y) is just as likely as the separate observations of x and y .

[stopped here, will work on more next season]