Use R for Climate Research

January 10, 2009

James B. Elsner & Thomas H. Jagger

This tutorial gives you an introduction to the R programming language for statistical analysis and modeling. Examples are taken from our research into hurricane climate. It was developed with support from the Florida State University, the U.S. National Science Foundation, and the Risk Prediction Initiative of the Bermuda Institute of Ocean Sciences. You should follow along with an open R session.

The key strokes are given in red font. If you're not fast at typing, they can be copied from the pdf and pasted into your R session. Output from an R command is given in `blue` Courier New font. Questions for which answers are sought and practice problems are given in green font.

The tutorial is divided into 5 sections. We begin with some motivations for giving R a try and some information on how to download and install R on your computer. We show you how to use R as a calculator. In section 2 we show you how to enter data into R, how to query your data, and how to obtain some statistical information about your data including correlation. In section 3 we provide an introduction to using R for graphing your data. In section 4 we show you how R can be used to model data. Topics include linear regression, Poisson regression, and quantile regression. In section 5 we list additional resource material.

1. Introduction

Benefits and drawbacks of R
Free, open-source, runs on unix, linux, windows, and macs. Built-in help system. Excellent graphing capabilities. Powerful, extensible, and easy to learn syntax. Thousands of built-in functions.

Limited graphical user interface meaning it is harder to learn at the outset. No commercial support. It's a programming language so you need to appreciate syntax issues. Common metaphors for working with computers are: browsers, iTunes, and perhaps Excel. R is nothing like these. Irregular users forget commands. There are no visual cues: A blank screen is intimidating.

Why switch to R?
If you've already mastered a statistical package and if you only carry out a limited range of statistical tests with no intention of doing something different, then don't switch. The main reason for switching to R is to take advantage of its unrivaled coverage and availability of new, cutting edge applications in emerging statistical fields such as generalized mixed effects and generalized additive models. Another reason is to be able to understand the literature. More people are reporting results in the context of R, and it is important to know what they're talking about. Many of the world's leading statisticians use and contribute to R.

"...the massive collective international effort represented by the R project exceeds the most idealistic Marxist imagination." --- Roger Koenker

What is R?
R is an open-source statistical environment modeled after S and S-Plus (http://www.insightful.com). The S language was developed in the late 1980s at AT&T labs. The R project was started by Robert Gentleman and Ross Ihaka of the Statistics Department of the University of Auckland in 1995. It now has a large audience. It's currently maintained by the R core-development team; an international group of volunteer developers.

R is the `lingua franca' of data analysis and statistical computing.

Since climate is the statistics of weather, climate scientists need to be steeped in statistics.

R is similar to other programming languages (e.g., Java, C++) in that it helps you perform a variety of computing tasks by giving you access to various commands.

For statisticians R is particularly useful because it contains a number of built-in mechanisms for organizing data, running calculations on the information and creating graphical representations of data sets.

To get to the R project web site, open a browser and in a search window, type the key words "R project".

The R project web page is http://www.r-project.org/

Directions for obtaining the software, accompanying packages and other sources of documentation are provided at the site.

Downloading and starting R
Click on the above url (universal resource locator) and browse to the site. Click on CRAN (Comprehensive R Archive Network) to download the latest version of R. Scroll to the nearest mirror site. Click on the appropriate pre-compiled binary distribution. Click on the base distribution to download. Only the base directory is needed at this time.

On your computer, click on the download icon and follow the instructions to install R.

Once installed, click on the icon to start R. To open R in Linux, from a command window, type R.

R is most easily used in an interactive manner. You ask it a question and it gives you an answer. Questions are asked and answered on the command line.

The > (greater than symbol) is used as the prompt. In what follows, it is the character that is NOT typed.

If a command is too long, a + (plus symbol) is used for the continuation prompt. If you get completely lost on a command, you can use Ctrl c or Esc to get the prompt back and to start over.

To find out how to cite R in your publications, type:

```
citation()
```

What appears on the screen is a reference citation for R and a BibTeX entry for LaTeX users.

Most commands are functions and most functions require parentheses. If the function has all default options, then the function is applied by typing two side by side parentheses.

Packages from the R repository
Lots of user-developed packages are available for doing more specific statistic. To install and load a package. Determine the name of the package (http://cran.r-project.org/). Then type:

```
chooseCRANmirror()
```

For this tutorial, it will help to install the UsingR and quantreg packages. Select a site, then type:

```
install.packages(c("UsingR","quantreg"))
library(UsingR); library(quantreg)
```

A google-like search portal for everything R is available at http://www.rseek.org/

Close to 1,600 different packages reside on just one of the many Web sites devoted to R, and the number of packages has grown exponentially.

One package, called BiodiversityR, offers a graphical interface aimed at making calculations of environmental trends easier. Another package, called Emu, analyzes speech patterns, while GenABEL is used to study the human genome.

Calculating
R evaluates commands typed at the prompt. It returns the result of the computation on the screen or saves it to an object. For example, to find the sum of the square root of 25 and 2, type:

```
sqrt(25)+2
[1] 7
>
```

The [1] says "first requested element will follow". Here there is only one element the answer 7. The > prompt that follows indicates R is ready for another command.

```
12/3-5
```

How would you calculate the 5th power of 2?

How about the amount of interest on $1000, compounded annually at 4.5% a.p. for 5 years.

```
1000*(1+0.045)^5-1000
```

Functions
Many mathematical and statistical functions are available in R. They are all used in a similar manner. A function has a name, which is typed, followed by a pair of parentheses (required). Arguments are added inside this pair of parentheses as needed.

For example,

```
sqrt(2)    # the square root
```

The # is the comment character. All text in the line following this is treated as a comment.

```
sin(pi)    # the sine function
[1] 1.224647e-16
exp(1)
log(10)    # log of 10 base e
```

Many functions have extra arguments that allow you to change the default behavior. For example, to use base 10 for the logarithm, we could use either of the following:

```
log(100,10)
[1] 2
log(10, base=10)
[1] 2
```

To understand the first function, log(10,10), we need to know that R expects the base to be the second argument of the function. The second example uses a named argument, of the type base=, to say explicitly that the base is 10. The first style contains less typing, but the second is easier to remember.

Warning and errors
When R finds a command it doesn't understand, it will respond with an error message. For example:

```
squareroot(2)
Error: could not find function "squareroot"

sqrt 2
Error: syntax error

sqrt(-2)
[1] NaN
Warning message:
In sqrt(-2) : NaNs produced

sqrt(2
+
```

The last command shows what happens if R encounters a line that is not complete. The continuation prompt, +, is printed, indicating more input is expected.

Assignments
It is often convenient to name a value so that you can use it later. Doing so is called assignment. Assignment is straightforward. You put a name on the left-hand side of the equals sign and the value on the right. Assignment does not produce any printed output.

```
x=2    # assignments return a prompt only
x+3    # x is now 2
[1] 5
```

Note the pound symbol (#) is used as a comment character. Anything after the # is ignored. Adding comments to your commands is a good way of recalling what you did and why.

Acceptable variable names
You are free to make variable names out of letters, numbers, and the dot or underline characters. A name starts with a letter or a dot (a leading dot may not be followed by a number). We cannot use mathematical operators, such as +, -, *, and /. The help page for make.names describes this in more detail (?make.names).

Some examples of legitimate names are:
```
x = 2
n = 25
a.really.long.number=123456789
AReallySmallNumber=0.00000001
```

Note that case is important. Some variable names are naturally used to represent certain types of data. For instance, often n is for a length; x or y stores a data vaector; and i and j are for integers and indices.

Variables that begin with the dot character are usually reserved for programmers. These conventions are not forced, but consistently using them makes it easier for you (and others) to look back and understand what you've done.

Getting help
The help manuals are available as html. To access them, type:

```
help.start()
```

To quit R, type q(save="no").

2. Data

Entering small amounts of data
Statistics is the analysis and modeling of data. The most useful R command for quickly inputting small amounts of data is the c function. This function combines (concatenates) items together. As an example, consider a set of hypothetical annual land falling hurricane counts over a ten-year period.

2 3 0 3 1 0 0 1 2 1

To enter these into your R session, type:

```
counts = c(2,3,0,3,1,0,0,1,2,1)
counts
[1] 2 3 0 3 1 0 0 1 2 1
```

Notice a few things. You assigned the values to an object called counts. The assignment operator is an equal sign (=). Values do not automatically print. They are assigned to an object name. They can be printed by typing the object name as we did on the second line. Finally, the values when printed are prefaced with a [1]. This indicates that the object is a vector and the first entry in the vector is a value of 2 (The number immediately to the right of [1]). More on this later.

You can save yourself a lot of typing if you learn that the arrow keys can be used to retrieve previous commands. Each command is stored in a history file and the up arrow key will move backwards through the history file and the down arrow forwards. The left and right arrow keys will work as expected.

You can also enter small amounts of data with the scan function. You enter data values one at a time line by line. When finished type enter.

```
counts2 = scan()
1: 2
2: 3
>
```

Applying a function
Once the data are stored in a variable, you can use functions on them. R comes with many built-in functions that you can apply to your counts data. One of them is the mean function for finding the average. To use it, type:

```
sum(counts)        # total number of hurricanes making landfall
length(counts)     # length of data vector
sum(counts)/length(counts)   # avg no. of hurricanes
mean(counts)
[1] 1.3
```

Note you can find the average as the total sum divided by the length, or you can use the mean function.

Other useful functions to know include, sort, min, max, range, diff, and cumsum. Try these on the landfall counts.

Data as vectors
The data is stored in R as a vector. This means that R keeps track of the order that the data were entered. There is a first element, a second element, and so on. This is good for several reasons.

1. Our simple data vector of counts has a natural order - year 1, year 2, etc. We do not want to mix these up.
2. We would like to be able to make changes to the data item by item instead of entering the entire data again.
3. Vectors are math objects so that math operations can be performed on them.

Let's see how these concepts apply to our landfall counts example. Suppose counts contain the annual landfall count from the first decade of a longer record. We want to keep track of counts over other decades. This could be done by the following, example.

```
d1 = counts      # make a copy
d2 = c(0,5,4,2,3,0,3,3,2,1)   # enter a new set of counts
```

Note the two different object names. Unlike many programming languages, the period in R is only used as punctuation. You can't use an _ (underscore) to punctuate names as the underscore is actually another assignment operator although its use is fading.

Performing arithmetic in R is made easier by the vectorization of functions. That is, most

functions will do their operation on each entry of the data vector at the same time.

```
d1 + d2
d1 - d2
```

Here the first year of the first decade is added (and subtracted) from the first year of the second decade and so on.

```
d1-mean(d1)
```

In this case a single number (the mean of the first decade) is subtracted from a vector. The result is from subtracting the number from each entry in the data vector. This is an example of data recycling. R repeats values from one vector so that its length matches the other. So here, the mean is repeated 10 times.

Variance
Suppose you are interested in the variance of the set of hurricane landfall counts. The formula is given by:

$$\operatorname{var}(x) = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_n - \bar{x})^2}{n - 1}$$

Although the var function will do this work for you in the future, here you see how you could do this directly using the vectorization of functions. The key is to find the squared differences and then add up the values.

```
x=c(2,3,5,7,11)
xbar=mean(x)
x-xbar
(x-xbar)^2
sum((x-xbar)^2)
n=length(x)
n
sum((x-xbar)^2)/(n-1)
[1] 12.8
```

To verify:

```
var(x)
[1] 12.8
```

Data vectors have a type
One restriction on data vectors is that all the values have the same type. This can be numeric, as in counts, character strings, as in

```
simpsons = c("Homer","Marge","Bart","Lisa","Maggie")
simpsons
```

or one of the other types will will encounter later. Note that character strings are made with matching quotes, either double, ", or single, '.

If we mix the type within a data vector, the data will be coerced into a common type, which is usually a character. This can prevent arithmetic operations.

Giving data vectors named entries
A data vector can have its entries names. These will appear when the vector is printed. The names function is used to retrieve and set values for the names. This is done as follows:

```
names(simpsons)=c("Dad","Mom","Son","Daughter 1","Daughter 2")
simpsons
```

Note that the names function appears on the other side of the assignment, unlike most functions, as it modifies the attributes of the data vector.

Returning to our landfalling hurricanes, suppose the U.S. National Hurricane Center (NHC) reanalyzes a storm, and that the 6th year of the 2nd decade is a 1 rather than a 0 for the number of landfalls. In this case we can type:

```
d2[6] = 1    # assign the 6 year of the decade a value of 1 landfall
```

The assignment to the 6th entry in the vector d2 is done by referencing the 6th entry of the vector with square brackets []. It is very important to keep this in mind: Parentheses () are used for functions and square brackets [] are used for vectors (and arrays, lists, etc).

```
d2    #print out the values
[1] 0 5 4 2 3 1 3 3 2 1
d2[2]  # print the number of landfalls during year 2 of the second decade
[1] 5
d2[4]  # 4th year count
[1] 2
d2[-4]  # all but the 4th year
[1] 0 5 4 3 1 3 3 2 1
d2[c(1,3,5,7,9)]   # print the counts from the odd years
[1] 0 4 3 3 2
```

One way to remember how to use functions is to think of them as pets. They don't come unless they are called by name (spelled properly). They have a mouth (parentheses) that likes to be fed (arguments), and they will complain if they are not feed properly.

Finding help
Using R to do statistics requires knowing a lot of different functions---more than you will likely be able to keep in your head. Thankfully, R has excellent built-in help facilities. These can be consulted for information about what is returned by the function, for details on additional arguments, and for examples.

If you know the name of the function, you can type for example:

```
?var   Or   help(var)
```

This works great if you can remember the name of the desired function. If not, there are other ways to search. The function help.search will search each entry in the help system and returning matches (often many) of functions that mention the word "mean".

To match just function names, the well-named apropos function will search through the available function names and variables for matches. Try

```
apropos("mean")
```

Most help pages have examples. These can be run one-by-one by cutting and pasting into the console, or all at once by using the function example.

```
example(mean)
```

Working smarter not harder
R's console keeps a history of the commands entered. The history function shows the last 25. Individually the commands can be accessed using the up and down arrow keys. Repeatedly pushing the up arrow will scroll backward through the history. This is extremely useful, as we can reuse previous commands.

Many times we wish to change only a small part of a previous command, such as when a typo is made. With the arrow keys we can access the previous command then edit it as desired.

Creating structured data
Sometimes numbers have some structure or pattern. Take, for instance, the integers 1 through 99. To enter these into an R session one by one would be tedious.

The colon operator is used for creating simple sequences.

```
1:10
rev(1:10)
10:1
```

It is usually desirable to specify either the step size and the starting and ending points or the starting and ending points and the length of the sequence. The seq function allows us to do this.

```
seq(1,9, by=2)
seq(1,10,by=2)
seq(1,9,length=5)
```

When a vector of repeated values is desired, the rep function is used. The simplest usage is to repeat the first argument a specified number of times.

```
rep(1,10)
rep(1:3,3)
```

More complicated patterns can be repeated by specifying pairs of equal-sized vectors. In this case, each term of the first is repeated the corresponding number of times in the second.

```
rep(c("long","short"),c(1,2))
```

Asking questions
R lets us systematically examine our data. For example, to find the maximum number of landfalls in the first decade, type:

```
max(d1)
```

```
[1] 3
```

Which years had the maximum?

```
d1 == 3
[1] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

Notice the usage of double equals signs (==). This tests all the values of d1 to see if they are equal to 3. The 2nd and 4th answer yes (TRUE) the others no. Think of this as asking R a question. Is the value equal to 3? R answers all at once with a long vector of TRUE's and FALSE's.

Now the question is – how can we get the years corresponding to the TRUE values? Let's rephrase, which years have 3 landfalls? If you guessed that the command which will work, you're on your way to mastering R.

```
which(d1 == 3)
[1] 2 4
```

We might also want to know the total number of landfalls in each decade or the number of years in a decade without a landfall. Or how about the ratio of the mean number of landfalls over the two decades.

```
sum(d1); sum(d2)
[1] 13
[1] 24
```

Here we apply two functions on a single line by using the semicolon ;.

```
sum(d1 == 0); sum(d2 == 0)
[1] 3
[1] 1
```

```
mean(d2)/mean(d1)
[1] 1.85
```

So there is 85% more landfalls during the second decade. Is this statistically significant?

Before moving on, it is good practice to clean up your working directory. This is done using the rm (remove) function.

```
rm(d1,d2,counts)
```

Reading data

Most of your analyzes will be done on data that are read into R. First, determine what directory you are working in by typing:

```
getwd()
```

You can change your working directory by using the setwd() function and specifying the path.

Second, download the data (Save Link or Target As) into your working directory. The data

are available here (http://myweb.fsu.edu/jelsner/Data.html) under hurricane landfall counts.

Third, a simple way to read data is to use the read.table function.

```
ushur = read.table("UShur1851-2006.txt",header=T)
```

If the file is read in, you should get the > prompt. Note the quotes around the file name as you are referencing a file outside of R. If you can use full path names for files. The header=T argument means there is a header record in the data (T stands for true). Data read in with the read.table function are of class data.frame. To check, type:

```
class(ushur)
```

To see if the data are what you expect, try the following commands.

```
names(ushur)   # list the column names, US: US hurr, MUS: major US hur, G: Gulf coast, FL: Florida, E: East coast
head(ushur)  # list the first several rows
tail(ushur)  # list the last several rows
```

The function read.table has options for specifying the separator character or characters between columns used in the native file. Alternatives to the default use of a space are sep="," and sep="\t" for tab.

You can also control the choice of a missing value character, which by default is NA. If the missing value character in the native file is a 999, specify na.strings="999".

There are several variants of read.table that differ only in having different default parameter settings. Note in particular read.csv, which has settings that are suitable for comma delimited (csv) files that have been exported from an Excel spreadsheet.

You can also read directly from the web by specifying the complete URL instead of the local file name.

```
ushur=read.table("http://myweb.fsu.edu/jelsner/extspace/UShur1851-2006.txt",T)
```

If you want to change the names of the data frame, type:

```
names(ushur)[4]="GC"
```

This changes the 4th column name from G to GC.

More info: cran.r-project.org/doc/manuals/R-data.pdf

Summarizing data
Frequently all that's needed is a summary of your data.

```
mean(ushur$US)
[1] 1.711538
```

To access the columns of a data frame, you give the name of the data frame followed by the dollar sign ($) followed by the column name.

What is the median number of U.S. hurricanes over this time period? What is the variance?

The output carries too many digits. Since there are only about 150 or so years, the number of significant digits is 2 or 3. This can be changed using the signif function.

```
signif(mean(ushur$US),3)
[1] 1.71
```

Note that the signif function wraps around the mean function.

The mean or average number of counts over a time period is called a rate. How many years are in the data set?

```
dim(ushur)  # dimensions (rows by columns) of the data frame, which here is the number of years by the number of variables
[1] 156   6
```

So the annual hurricane rate is 1.71 hurricanes per year over the 156-year period.

To find out which year(s) have the maximum, type

```
ushur$Year[ushur$US==max(ushur$US)]
[1] 1886
```

Note how the command structure allows embedding. It is often best to read the expression from right to left. The first function on the right is max, so this returns a number giving the maximum annual count. Then we want the index(es) for that count, finally we want the year(s) corresponding to the index(es).

Similarly, to find the years with no hurricanes, type:

```
ushur$Year[ushur$US==min(ushur$US)]
 [1] 1853 1862 1863 1864 1868 1872 1884 1889 1890 1892 1895 1902 1905 1907
1914
[16] 1922 1927 1930 1931 1937 1951 1958 1962 1973 1978 1981 1982 1990 1994
2000
[31] 2001 2006
```

Note the output is indexed by the values inside the brackets on the left-hand side.

```
sum(ushur$US==0)  # How many years have zero hurricanes?
table(ushur$US)  # How many years have H hurricanes?
```

Correlation

Is there a trend in the U.S. landfall counts? One way to get an answer is to correlate the counts with the year.

```
cor(ushur$Year,ushur$US)
[1] -0.003594048
```

What if someone explains that since you are using count data, you should use the rank correlation. By default, R provides the Pearson product-moment correlation coefficient. To

change the default, type:

```
cor(ushur$Year,ushur$US,method="s")  # method="s" refers to the Spearman rank correlation
```

With two variables we are often interested in whether or not the correlation is significantly different from zero. In R we do that with the cor.test function.

```
cor.test(airquality$Ozone,airquality$Solar.R)
    Pearson's product-moment correlation
data:  airquality$Ozone and airquality$Solar.R
t = 3.8798, df = 109, p-value = 0.0001793
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.173194 0.502132
sample estimates:
      cor
0.3483417
```

Here the correlation between ozone and solar radiation is 0.35. Under the null hypothesis that the correlation is zero the t statistic defined as

$$t = \frac{r}{\sqrt{1-r^2}}\sqrt{n-2}$$

has a t distribution with df=n-2.

The p-value as evidence in support of the null hypothesis is very small (0.00018). Thus we conclude that there is a significant non-zero correlation between ozone and solar radiation.

The p-value
A p value is an estimate of the probability that a particular result, or a result more extreme than the result observed, could have occurred by chance if the null hypothesis were true. In short, the p value is a measure of the credibility of the null hypothesis.

However, the p value is often interpreted as evidence against the null hypothesis, thus a small p value indicates evidence to reject the null. The interpretation of the p value as evidence against the null hypothesis is:

p value range	interpreted as evidence against the null hypothesis
0-0.01	convincing
0.01-0.05	moderate
0.05-0.15	suggestive, but inconclusive
> 0.15	none

The p-value in this case is small so we reject the null hypothesis of a zero correlation between ozone and solar radiation based on this set of data.

Bootstrap samples
You've probably heard the old phrase about pulling yourself up by your own bootstraps. That is where the term 'bootstrap' in statistics comes from. You might only have a single sample but you can sample (bootstrap) from the original sample in many ways so long as you allow some values to appear more than once and other not to appear at all (sampling with replacement).

You calculate the mean of the values in each bootstrap sample and obtain a confidence interval by looking at the extreme highs and lows using the quantile function.

Let
```
US=rpois(150, 1.7)  #simulate 150 years of hurricane counts

a=c()
for(i in 1:1000) a[i]=mean(sample(US,10,replace=T))
hist(a,main="", xlab="Historical Annual U.S. Hurricane Rate")
quantile(a,c(0.025,0.975))
```

```
2.5% 97.5%
 1.2   2.8
```

Note the large range of bootstrapped hurricane rates. A 95% confidence interval spans a factor of 3. This is characteristic of small count data and should be kept in mind when describing periods as "active" and "inactive".

3. Graphs and Plots

"It is well and good to opine or theorize about a subject, as humankind is wont to do, but when moral posturing is replaced by an honest assessment of the data, the result is often a new, surprising insight."---*Levitt and Dubner (Freakonomics)*

If you do nothing else with your data you will likely at least make a graph. R has a wide range of plotting capabilities. It takes time to become a master, but a little effort goes a long way.

Although the commands are simple and somewhat intuitive, to get a publication quality figure usually requires some tweaking the default settings. This is true of all plotting software.

Several graphics device drivers are available including: On-screen graphics: Windows, X11, or quartz. Print graphics: postscript, pdf, png, jpeg, and wmf.

The on-screen devices are more commonly used. For publication-quality graphics, the postscript, pdf, or wmf devices are preferred because they produce scalable images. Use bitmap devices for drafts.

The preferred sequence is to specify a graphics device, then call graphics functions. If you do not specify a device first, the on-screen device is started.

Back to our hurricane data. Since we need access to the various columns in our data frame, it simplifies typing if we use the attach and detach functions. The attach function makes the columns available to use in commands without the need to preface the data frame name. For example, if you type:

```
US
Error: object "US" not found
```

The error tells you that there is no object with that name in your working directory. To list the objects, type ls(). Now type:

```
attach(ushur)
US
```

This makes things easier. It's good practice to use the detach function when you finish with the data frame to avoid masking objects with the same name from other data frames or objects.

Scatter plots
To start, let's look at the U.S. hurricane counts over time as a scatter plot.

```
plot(Year, US)
```

The default settings on plot give you a first look, but the plot needs to be modified before serious study.

```
plot(Year, US, type="l")    # Note the letter in quotes is a small "l" for line, not the number 1
```

This is often how you see count data plotted in the hurricane literature. Ignoring for now the axis labels, what's wrong with it? To help you with the answer, grab the lower right corner of the plot window and stretch it to the right. See it?

To avoid this problem, use the type="h" option.

```
plot(Year,US,type="h")
plot(Year,US,type="h",lwd=3)   # This increases the line width from a default of 1 pixel unit to 3
```

To make it prettier, type:

```
plot(Year,US,ylim=c(0,8),ylab="U.S. Hurricanes",type="h",lwd=3,las=1)
```

Is there any correlation between the covariates? One way to examine this is to graph a scatterplot of the data. First read the data from our website.

```
AMO=read.table("http://myweb.fsu.edu/jelsner/extspace/AMO1856-2006.txt",T)
SOI=read.table("http://myweb.fsu.edu/jelsner/extspace/SOI1866-2006.txt",T)
NAO=read.table("http://myweb.fsu.edu/jelsner/extspace/NAO1851-2006.txt",T)
```

Next create the covariate data.

```
amoAO=(AMO$Aug+AMO$Sep+AMO$Oct)/3
soiAO=(SOI$Aug+SOI$Sep+SOI$Oct)/3
naoMJ=(NAO$May+NAO$Jun)/2
```

```
plot(amoAO,naoMJ)
plot(amoAO,soiAO)
```

What do you conclude?

Distribution plots
Another useful data display for discrete data is the histogram. This provides information on the type of distribution. Knowing what kind of distribution your data conform to is important for modeling.

Let's start with the simplest thing to do.

```
hist(US)
```

Not bad, but the location of the abscissa labels makes it difficult to understand what bar belongs to what hurricane count. Moreover, since the values are discrete, the years with 0 landfalls are added to the years with 1 landfall. You can see this by looking at the output from table(US) as we did previously.

There are many options for improving the histogram. You can see these options by typing a question mark before the function name and leaving off the parentheses.

```
?hist  # brings up a help menu on the topic of histograms
```

Here is one that works by specifying the break points with the breaks argument. Note also the use of the seq function.

```
hist(US,breaks=seq(-0.5,7.5,1))
```

That's more like it. Now we can see that the mode is 1 (not 0) and that 2s are more frequent than 0s. The distribution is said to be "skew to the right" (or positively skew) because the long tail is on the right-hand side of the histogram.

To overlay a theoretical (model) distribution, type:

```
xs=0:7
ys=dpois(xs,1.71)
lines(xs,ys*156,col="red")
```

The data fits the theoretical distribution quite well. We will look at these theoretical distributions a bit later.

```
barplot(table(US),ylab="Number of Years",xlab="Number of Hurricanes")
barplot(table(US)/length(US),ylab="Proportion of Years",xlab="Number of Hurricanes")
```

Note that the command table(US)/length(US) produces proportions with the result handed off to barplot to make a graph. The graph has the same shape as the previous one, but the height axis is between 0 and 1 as it measures the proportion of years.

To examine the distribution of the covariates together with the distribution of US hurricanes, type the following sequence of commands:

```
par(mfrow=c(2,2))
```

```
hist(US,breaks=seq(-0.5,7.5,1))
hist(amoAO)
hist(soiAO)
hist(naoMJ)
```

The par function sets the graphical parameters. Here they are set by the argument mfrow, which should be read as "multiframe, rowwise, 2 x 2 layout". Thus each plot is added to the graph in lexicographical ordering (top to bottom, left to right).

For continuous variables like the covariates considered here, it is sometimes better to plot an estimate of the probability density function. The probability density function (pdf) can be seen as a smoothed version of the histogram. The probability of observing a value between any two locations [a, b] is given by the integral of the density function evaluated between a and b.

The plot of a probability density function first requires a call to the density function. By default an appropriate smoothing value (bandwidth) will be chosen as the standard deviation of the smoothing kernel (window). To examine the pdf for the default and other bandwidths, type:

```
par(mfrow=c(2,2))
plot(density(naoMJ))
plot(density(naoMJ,bw=0.15))
plot(density(naoMJ,bw=0.45))
plot(density(naoMJ,bw=0.7))
```

To add the locations of the values, use the rug function.

```
par(mfrow=c(1,1))
plot(density(naoMJ),main="",xlab="May-Jun NAO (sd)")
rug(naoMJ)
```

One purpose of plotting the pdf is to assess whether the data can be assumed normally distributed. Clearly in the case of the NAO, the data appears to be normal.

For a better assessment, you can plot the kth smallest NAO value against the expected value of the kth smallest value out of n in a standard normal distribution. In this way you would expect the points on the plot to form a straight line if the data come from a normal distribution with **any** mean and standard deviation. The function for doing this is called qqnorm.

```
qqnorm(naoMJ)
```

As the default title of the plot indicates, plots of this kind are also called "Q-Q plots" (quantile versus quantile). The sample of values has heavy tails if the outer parts of the curve are steeper than the middle part.

A "box plot", or more descriptively a "box-and-whiskers plot", is a graphical summary of a set of values. It is used to examine the distribution of your data. To get a box plot of the NAO values, type:

```
boxplot(naoMJ)
```

The box in the middle indicates that the middle 50% of the data (between the first and third

quartiles) lies between about -1 and +0.5 s.d. The median value is the thick horizontal line inside the box. The lines ("whiskers") show the largest/smallest value that falls within a distance of 1.5 times the box size from the nearest quartile. If any value is farther away, it is considered "extreme" and is shown separately.

Boxplots can also be used to look at conditional distributions. For instance, what is the distribution of the NAO values conditional on the upcoming season having few or many U.S. hurricanes? This is question that is important as we move toward modeling our data. In section 2 we noted a negative correlation between the number of U.S. hurricanes and the NAO thus we would expect to see lower values of NAO when the rate of U.S. hurricanes is above normal.

```
boxplot(naoMJ~ushur$US>3,ylab="NAO May-Jun (sd)")
```

The parallel box plots help us visualize the relationship between hurricanes and covariates.

```
label=factor(c(rep("On or Before 1928",78),rep("After 1928",78)))
boxplot(ushur$US~label,notch=T,xlab="",ylab="US Hurricanes")
```

Box plots can also be used to examine seasonality. The AMO covariate contains monthly values from 1856-2006. Suppose we want to look at a box plot of the AMO values by month. Note: Don't worry about memorizing this code.

```
amo2=na.omit(amo)   # omit rows with no data
amo2=stack(amo2[,2:13])   # stack the columns
ind=factor(as.character(amo2$ind),levels=month.abb)   # create an ordered factor
(months)
plot(ind,amo2$values)
```

Note that the examples as described will plot the figures on the screen. In order to create hard copy output, we need to specify an output device.
R supports several devices, but by far the most commonly used are the postscript and pdf devices. We first open the device explicitly. For
example, `postscript(file="filename")` or `pdf(file="filename")`.

All plotting commands that follow will be written to that device, until you close it with a `dev.off()` command. For example,

```
postscript(file="AMO.pdf") # file name for pdf file that is dropped outside of the R session in
your working directory
plot(ind,amo2$values)  # plotting command(s)
dev.off() # close postscript device
```

The width and height of the postscript plot is specified with the `height=` and `width=` arguments in the `postscript` function. Units on the values are inches.

Creating reports
Sweave is a tool that allows to embed the R code for complete data analyses in latex documents. The purpose is to create dynamic reports, which can be updated automatically if data or analysis change. Instead of inserting a prefabricated graph or table into the report, the master document contains the R code necessary to obtain it.

When run through R, all data analysis output (tables, graphs, etc.) is created on the fly and inserted into a final latex document. The report can be automatically updated if data or analysis change

See: http://www.statistik.lmu.de/~leisch/Sweave/

4. Data Models

Introduction
Theoretical distributions like the normal and the Poisson, are examples of data models. If your data conform to a theoretical distribution then the analysis is simplified, because the properties of the distributions are known.

Fitting models to data is a central function of R. The object is to determine a minimal adequate model from the large set of potential models that might be used to describe the given set of data.

Parsimony says that, other things being equal, we prefer:
- A model with n-1 parameters to a model with n parameters
- A model with k-1 explanatory variable to a model with k explanatory variables
- A linear model to a model that is curved
- A model without interactions to a model with interactions

We also prefer models containing explanatory variables that are easy to measure over variables that are difficult to measure or are derived (e.g., EOFs). And we prefer models that are based on a sound mechanistic understanding of the process over purely empirical functions.

All of the above are subject to the caveats that the simplifications make good scientific sense and do not lead to significant reductions in explanatory power. There is no perfect model.

Models in R follow the following formula: response variable ~ explanatory variable(s), where the ~ reads "is modeled as a function of".

There is a temptation to become personally attached to a particular model. You do well to remember the following truths about models:
- All models are wrong.
- Some models are better than others.
- Some models are useful.
- The correct model can never be known with certainty.
- The simpler the model, the better it is all else the same.

After fitting a model to your data it is essential that you investigate how well the model describes the data. In particular, are there any systematic trends in the goodness of fit? This is done by examining the model residuals which are computed as the difference between the observed response and the predicted response.

Linear regression
In section 2 we saw how to get the correlation between two variables. Correlation is nothing more than a statistical term that indicates whether two variables move together. It tends to be cold outside when it snows; these two factors are positively correlated. Sunshine and

rain, meanwhile, are negatively correlated.

Easy enough so long as there are only a couple of variables. With many variables things get harder. Regression analysis is the tool that enables a geographer to sort out piles of data with many variables. It does so by artificially holding constant every variable except the two he wishes to focus on, then showing how these two co-vary.

It has been said that economics is a science with excellent tools for gaining answers but a serious shortage of interesting questions. Geography is in the other boat. It has no shortage of interesting questions, but it tends to lack tools for gaining answers. Of all the tools in an economists toolbox, the most useful is regression.

Mathematically, regression attributes to a response variable y a normal distribution whose expected value depends on a covariate x, typically in the following way: $E[y] = a + b x$. The parameters a and b are estimated by a least-squares minimization procedure. It is called "linear" regression because the variables multiplied by their parameters are added together.

For linear regression, the function lm is used (linear model). To obtain a linear regression model of the August through October AMO on the May through June NAO, type:

```
lm(amoAO~naoMJ)
Call:
lm(formula = amoAO ~ naoMJ)

Coefficients:
(Intercept)        naoMJ
   22.79375     -0.02687
```

The argument to the lm function is a model formula. The variable to the left of the tilde is the dependent variable and the variable to the right is the independent variable.

Regression is a model of the response (dependent) variable (usually denoted by "y") ON the explanatory (independent) variable, (usually denoted by "x"). You regress y onto x. Unlike correlation, which is not a model for the data, it matters which variable is which and the choice is made by the investigator, not the software.

Continuous explanatory variables are called "covariates".

The statement "I regressed variable A and variable B" is vague.

The coefficients are the values of the intercept and slope of the straight line through the scatter of points when the NAO is on the horizontal axis. The equation for the line is read as:

The mean of Aug-Oct AMO in units of degrees C equals 22.7 minus 0.02687 times the May-Jun NAO in units of s.d. Thus the units on the intercept are the same as the units of the response variable, in this case degrees C, and the units on the slope are degrees C/s.d.

To see the line, type:

```
plot(naoMJ,amoAO)
abline(lm(amoAO~naoMJ))
```

The regression model is different if the NAO is regressed on the AMO. Try it. Also consider a regression of hurricane counts on year.

In its raw form, the output of lm is very brief. All you see is the estimated intercept and estimated slope values, but there are no tests of significance. The result of lm is a model object. This is a distinctive concept in R. Where other statistical programs focus on generating printed output that can be controlled by setting options, you get instead the result of a model fit encapsulated in an object from which the desired quantities can be obtained using other functions.

More output is obtained if you type:

```
summary(lm(amoAO~naoMJ))
Call:
lm(formula = amoAO ~ naoMJ)

Residuals:
     Min       1Q   Median       3Q      Max
-0.52648 -0.14007 -0.02344  0.12599  0.65141

Coefficients:
            Estimate Std. Error  t value Pr(>|t|)
(Intercept) 22.79375    0.02008 1135.221   <2e-16 ***
naoMJ       -0.02687    0.01920   -1.399    0.164
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2345 on 149 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-Squared: 0.01297,    Adjusted R-squared: 0.006343
F-statistic: 1.957 on 1 and 149 DF,  p-value: 0.1639
```

Model description
Let's dissect the output.

```
Call:
lm(formula = amoAO ~ naoMJ)
```

The output starts with a repeat of the function call. This is not very interesting, but it is useful if the result is saved in an object that is examined later.

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.52648 -0.14007 -0.02344  0.12599  0.65141
```

This gives us a quick look at the distribution of the residuals. The residuals are the key for understanding how well the model fits the data and whether there are problems. The average of the residuals is zero by definition, so the median should not be far from zero and the minimum and maximum should roughly be equal in absolute magnitude. Similar with the 1st (1Q) and 3rd (3Q) quartile values.

```
Coefficients:
            Estimate Std. Error  t valu  Pr(>|t|)
(Intercept) 22.79375    0.02008 1135.221   <2e-16 ***
naoMJ       -0.02687    0.01920   -1.399    0.164
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Here we again see the regression coefficients including the intercept and slope, but this time with accompanying standard errors, t values and p values. Note that the slope value and its statistics are given to the right of the covariate, but they refer to the coefficient that is multiplied by the covariate values, not the covariate values themselves. To the right of the p values is an indicator of the level of significance. Here we note that the NAO is not a very good predictor of the AMO.

```
Residual standard error: 0.2345 on 149 degrees of freedom
```

This is the residual variation, an expression of the variation of the observations around the regression line, estimating the model parameter sigma. The degrees of freedom is the number of observations minus 2 (since there are two model parameters).

```
(5 observations deleted due to missingness)
```

This indicates that there are 5 missing values in either the response or covariate or both. Rows with missing values are not included in the model (deleted). Care must be exercised when comparing models built with differing data sizes.

```
Multiple R-Squared: 0.01297,    Adjusted R-squared: 0.006343
```

The first item is the R squared value. When multiplied by 100%, it is interpreted as the variance explained by the covariate.

SSY=deviance(lm(amoAO~naoMJ))
SSE=deviance(lm(amoAO~1))
R squared = (SSY-SSE)/SSY

The adjusted R squared is an index that accounts for the loss in the degrees of freedom when adding a covariate to the model. It is important in the context of multiple regression.

```
F-statistic: 1.957 on 1 and 149 DF,  p-value: 0.1639
```

This is an F test for the hypothesis that the slope is zero. The F-statistic has a value of 1.957 on 1 and 149 degrees of freedom (DF). For the case of a one covariate model, it gives the same p-value as the one on the slope coefficient. It is more interesting in the case of more than one covariate.

Another example: airquality data set.

?airquality

Regress ozone concentration (Ozone) on solar radiation (Solar.R), wind speed (Wind), and temperature (Temp).

```
model=lm(Ozone~Solar.R+Wind+Temp, data=airquality)
summary(model)
```

The R-squared value can be manipulated simply by adding more explanatory variables. The adjusted R-squared value is an attempt to correct this short-coming by adjusting both the numerator and the denominator by their respective degrees of freedom.

adjusted R squared = 1- (1 - R2)[(n - 1)/(n - p - 1)]

R2 = R squared
n = number of observations
p = number of explanatory variables

Unlike the R-squared, the adjusted R-squared will decline in value if the contribution to the explained deviation by the additional variable is less than the impact due to the loss in degrees of freedom. Thus it is a better way to compare models than using R-squared.

Note: while the R-squared is a percent, the adjusted R-squared is not and should be referred to as an index value.

To obtain confidence intervals on the model coefficients type:

```
confint(model)
```

Model predictions
To get predictions from the model for each set of explanatory variables type:

```
predict(model)
```

To get point-wise confidence intervals on the predictions

```
predict(model,level=0.95,interval="confidence")
```

To get prediction intervals use interval="prediction".

To find the confidence bands The confidence bands would be a chore to compute by hand. Unfortunately, it is a bit of a chore to get with the low-level commands as well. The predict method also has an ability to find the confidence bands if we learn how to ask. Generally speaking, for each value of x we want a point to plot.

This is done as before with a data frame containing all the x values we want. In addition, we need to ask for the interval. There are two types: confidence, or prediction. The confidence will be for the mean, and the prediction for the individual. Let's see the output, and then go from there. This is for a 90% confidence level.

```
predict(lm.result,data.frame(x=sort(x)), level=.9, interval="confidence")
fit lwr upr
1 195.6894 192.5083 198.8705
2 195.6894 192.5083 198.8705
3 194.8917 191.8028 197.9805
... skipped ...
```

We see we get 3 numbers back for each value of x. (note we sorted x first to get the proper

order for plotting.)

To plot the lower band, we just need the second column which is accessed with [,2]. So the following will plot just the lower. Notice, we make a scatterplot with the plot command, but add the confidence band with points.

```
plot(x,y)
abline(lm.result)
ci.lwr = predict(lm.result,data.frame(x=sort(x)), level=.9,interval="confidence")[,2]
points(sort(x), ci.lwr,type="l") # or use lines
```

Alternatively, we could plot this with the curve function as follows

```
curve(predict(lm.result,data.frame(x=x), interval="confidence")[,3],add=T)
```

This is conceptually easier, but harder to break up, as the curve function requires a function of x to plot.

Model adequacy

By fitting a linear regression model, you are making several implicit assumptions including linearity, normally distributed residuals, constant variance, and independence.

- Linearity assumption: The means of the subpopulations fall on a straight-line function of the explanatory variables.
- Normality assumption: There is a normally distributed subpopulation of responses for each value of the explanatory variable.
- Constant variance assumption: The subpopulation standard deviation are all equal.
- Independence assumption: The selection of an observation from any of the subpopulation is independent of the selection of any other observation.

The first three assumptions are best checked using plots.

```
plot(model)
```

The fourth assumption can be checked by plotting the autocorrelation function of the model residuals.

```
acf(residuals(model))
```

It is also a good idea to check for collinearity among the explanatory variables. This can be done with the cor function as we saw earlier. If the correlation between two explanatory variables is between -0.6 and +0.6 then collinearity is usually not a big problem.

Poisson regression

Poisson distribution

The Poisson distribution is one of the most useful and important of the discrete probability models for describing count data. We know how many times something happened. The number of lightning strikes over Tallahassee or the number of hurricanes affecting the state of Florida, are examples.

The Poisson distribution is a one-parameter distribution with the interesting property that its variance is equal to its mean. A large number of processes show variance increasing with the mean, often faster than linearly.

The density function of the Poisson distribution shows the probability of obtaining a count x

when the mean count per unit is lambda. It is given by the exponential of minus lambda times lambda raised to the x power all divided by x factorial.

The probability of zero events is obtained by setting x to 0, which yields exp(-lambda). Thus the probability of zero hurricanes affecting the U.S. next year, given a mean rate of 1.7 hurricanes/year is 18%.

```
exp(-1.7)
[1] 0.1826835
```

Of course, this implies that the probability of at least 1 hurricane is 100-18 or 82%.

The dpois function evaluates the above equation for any count of interest. To determine the probability of observing exactly 1 hurricane when the rate is 1.7 hurricanes per year, type:

```
dpois(1,1.7)
```

Or the probability of 5 hurricanes when the rate is 1.7 expressed in percent.

```
dpois(5,1.7)*100
```

To answer the question, what is the probability of 1 or fewer hurricanes we use the cumulative probability function ppois. Thus to answer the question, what is the probability of more than 2 hurricanes, we type:

```
1-ppois(3,1.7)
ppois(3,1.7,lower.tail=F)
```

If we want to simulate 100 seasons of hurricane strikes on the U.S. coast with a rate that is commensurate to the long-term rate of 1.7 hurricanes per year, we type:

```
rpois(100,1.7)
```

To see the distribution, type:

```
table(rpois(100,1.7))
```

Repeat for another 100 years.

To plot the distribution, type:

```
barplot(table(rpois(100,1.7)))
```

Poisson regression model

What if the rate of hurricanes depends on time or climate variables? Note the way the question is worded. We are interested in the rate of hurricanes because given the rate, the counts follow a Poisson distribution. Thus we are not modeling counts directly.

There's a mistaken belief that Poisson regression is least squares regression on the logarithm of counts. It's not. It is a regression using the logarithm of the rate. It's nonlinear in the regression function, but linear in regression structure and the model coefficients are determined by the method of maximum likelihoods rather than the method of least squares.

Our interest here is to study the relationship between the climate variables and coastal hurricane numbers. We do this with a regression model that specifies that the logarithm of the annual rate is linearly related to the covariates.

The Poisson regression model attributes to a response variable y a Poisson distribution whose expected value depends on a covariate x, typically in the following way: log(E[y]) = a + b x. The parameters a and b are estimated via the maximum likelihood procedure.

```
summary(glm(ushur$US~naoMJ,family=poisson))
Call:
glm(formula = ushur$US ~ naoMJ, family = poisson)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1480  -0.8054  -0.1172   0.5346   2.8388

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.44195    0.07019   6.297 3.04e-10 ***
naoMJ        -0.21708    0.06336  -3.426 0.000612 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 207.16  on 155  degrees of freedom
Residual deviance: 195.26  on 154  degrees of freedom
AIC: 515.54

Number of Fisher Scoring iterations: 5
```

We note that the coefficient on the NAO covariate is highly significant (p value < 0.001). The negative sign on the coefficient indicates that the there are fewer U.S. hurricanes when the NAO is high.

The value of -0.217 indicates that for a one s.d. change in the NAO, the difference in the logarithm of expected counts is -0.217. If there are other covariates we must add "given the other covariates in the model are held constant". Since log(A)-log(B) = log(A/B), we can interpret the parameter estimate as the log of the ratio of expected counts.

The residual deviance replaces the residual standard error and is used to determine model fit.

With Poisson regression the drop in deviance (deviance of the null model minus the deviance of the full model) is used to test whether the model is significant against the null hypothesis that none of the covariates are useful. Since the model contains significant covariates the model itself is significant. To see this, type:

```
pchisq(207.16-195.26,1,lower.tail=F)
```

Model adequacy is examined with the residual deviance.

The residual deviance is 195.26 on 154 degrees of freedom. A p-value from a chi-squared distribution with this quantile value and this dof is evidence in favor of the null hypothesis that the model is adequate. A small p-value is evidence against model adequacy.

Type:

```
pchisq(195.26,154,lower.tail=F)
```

Here we find moderate evidence against model adequacy indicating that other covariates are missing from the model. NAO is not the entire story.

While a Poisson distribution for given x has variance of y equal to mean of y, this assumption may not be supported in the data. If variance tends to exceed the mean, the data is over-dispersed relative to the Poisson distribution; under-dispersion occurs when variance is smaller than the mean. If a Poisson model seems appropriate but doesn't fit well, over-dispersion or under-dispersion could be the reason.

The AIC is the Akaike information criteria which is equal to 2 times the number of model parameters minus 2 times the logarithm of the likelihood function. The AIC is an operational way of trading off the complexity of an estimated model against how well the model fits the data. The smaller the AIC, the better the model. Adding a covariate that captures the probability of hurricane genesis will likely improve the model.

Practice
Fit a Poisson regression model to the U.S. hurricane counts using all three covariates. Interpret the results.

Quantile regression
Introduction
Quantile regression extends ordinary least squares regression model to conditional quantiles (e.g., 90th percentile) of the response variable.

Recall that ordinary regression is a model for the conditional mean (the mean of the response is conditional on the value of the explanatory variables). It is a semi-parametric technique because it relies on the non-parametric quantiles, but uses parameters to assess the relationship between the quantile and the covariates.

Quantiles are points taken at regular intervals from the cumulative distribution function of a random variable. The quantiles mark a set of ordered data into equal-sized data subsets.

Download the set of per storm maximum wind speeds (Save Link or Target As) into your working directory. The data are available here under per storm maximum wind speeds.

Read the data into your current R session and check the values from the first six rows by typing:

```
StormMax=read.csv("http://myweb.fsu.edu/jelsner/extspace/
extremedatasince1899.csv",header=T)
head(StormMax)
```

```
    Yr Region     Wmax   nao   soi        sst      sstmda sun split
1 1899  Basin 96.64138 -0.64 -0.21 0.05193367 -0.03133333 8.4     0
2 1899   East 90.19791 -0.64 -0.21 0.05193367 -0.03133333 8.4     0
```

```
3 1899  Basin 90.35300 -0.64 -0.21 0.05193367 -0.03133333 8.4     0
4 1899   East 37.51933 -0.64 -0.21 0.05193367 -0.03133333 8.4     0
5 1899  Basin 51.06743 -0.64 -0.21 0.05193367 -0.03133333 8.4     0
6 1899  Basin 40.00000 -0.64 -0.21 0.05193367 -0.03133333 8.4     0
```

Here Wmax is the maximum estimated wind speed for each cyclone in the North Atlantic by region. Basin refers to the region away from the U.S. coast.

Quantiles and distributions
Attach the data frame, subset by Basin, and determine the quartiles (0.25 and 0.75 quantiles) of the wind speed distribution.

```
StormMaxBasin=subset(StormMax,Region=="Basin")
attach(StormMaxBasin)
quantile(Wmax,c(0.25,0.75))
     25%      75%
50.43998 95.57234
```

We see that 25% of the storms have a maximum wind speed less than 51 kt and 75% have a maximum wind speed less than 96 kt so that 50% of all storms have a maximum wind speed between 51 and 96 kt (interquartile range).

To see if there is a trend in maximum wind speed values over time, we can use the linear regression technology from a previous section. However, we note that the distribution of per cyclone maximum wind speed is not normal so it is best to transform the wind speeds before using linear regression. A logarithmic transform is often a good choice.

To examine the distributions of the raw and log transformed wind speeds, type:

```
par(mfrow=c(2,1))
hist(Wmax)
hist(log(Wmax),breaks=12)
```

Or

```
par(mfrow=c(1,2))
boxplot(Wmax)
boxplot(log(Wmax))
```

To see if there is a significant trend in the mean wind speed over time, type:

```
summary(lm(log(Wmax)~Yr))
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.5535805  0.7600147   4.676 3.33e-06 ***
Yr          0.0003521  0.0003880   0.907    0.364
```

The answer is no.

Conditional quantiles
However, this is a model for the conditional mean (here, conditional on year). What about higher wind speeds? The theory of maximum potential intensity refers to storms at their

maximum possible wind speed. The subset of upper quartile maximum wind speeds would be on average a subset of storms closer their maximum possible wind speed.

So what we need is a model for the conditional quantiles.

Here we restrict the dataset to the years since satellite data.

```
detach(StormMaxBasin)
StormMaxBasin=subset(StormMaxBasin,Yr>1977)
attach(StormMaxBasin)

x=boxplot(Wmax~as.factor(Yr),plot=F)
boxplot(Wmax~as.factor(Yr),ylim=c(35,175),xlab="Year",ylab="Intensity (kt)")
xx=1:29
abline(lm(x$stats[5,]~xx),col="red")
abline(lm(x$stats[4,]~xx),col="blue")
abline(lm(x$stats[3,]~xx),col="green")
```

The graph verifies the lack of trend in the central tendency of maximum cyclone intensities. It also shows a tendency for the strongest storms to get stronger over time. To quantify this trend and determine significance we turn to a quantile regression model.

Quantile regression models
The functionality is available in the quantreg package thanks to Roger Koenker.

```
install.packages("quantreg")
library(quantreg)
citation("quantreg")
```

We begin by running a median regression on year and checking if the parameter (slope) is significantly different from zero.

```
summary(rq(Wmax~Yr,tau=0.5),se="iid")
Call: rq(formula = Wmax ~ Yr, tau = 0.5)

tau: [1] 0.5

Coefficients:
            Value     Std. Error t value   Pr(>|t|)
(Intercept) 118.58813 437.70671    0.27093   0.78661
Yr           -0.02639   0.21955   -0.12022   0.90438
```

As expected we find no significant trend in the median storm intensity over time. How about trends in the upper quantiles?

```
summary(rq(Wmax~Yr,tau=c(0.75,0.9,0.95)),se="iid")
```

At the 75th and 90th percentiles, there is suggestive, but inconclusive evidence of a trend, but at the 95th percentile there is convincing evidence of an increase in the intensity of the strongest hurricanes in accord with MPI theory.

To show the results on a plot, type:

```
model=rq(Wmax~Yr,tau=seq(0.7,0.95,0.01))
plot(summary(model,alpha=.05,se="iid"),parm=2,pch=19,cex=1.2,mar=c(5,5,4,2)+0.1,ylab="Trend
(kt/yr)",xlab="Quantile",main="North Atlantic Hurricanes")
```

The dots with lines indicate the trend for quantiles in the range between 0.7 and 0.95 by increments of 0.01. The gray shading indicates the 95% confidence interval on the slope estimate. The red solid line is the least squares regression slope and the dotted lines are the 95% confidence interval about that estimate.

Quantile regression is not linear regression on the quantiles

```
globalTCmax4=read.table("http://myweb.fsu.edu/jelsner/extspace/
globalTCmax4.txt",T,stringsAsFactors=F,na.strings="NaN")   #load the data fix North
Atlantic
globalTCmax4$Wmax=globalTCmax4$WmaxST*0.514444444

IO=subset(globalTCmax4,Basin=="IO")
IO1=tapply(IO$Wmax,IO$Year,quantile, p=.9)
IO1.df=data.frame(Year=unique(IO$Year),Wmax=IO1,weights=tapply(IO$Year,IO$Year,length))

lm90=lm(Wmax~Year,data=IO1.df)
qr90=rq(tau=.9,Wmax~Year,data=IO)
qr90
lm90

plot(Wmax~Year,data=IO,pch=16,cex=.7)
abline(qr90)
points(Wmax~Year,data=IO1.df,pch=15,col="green",cex=.7)
abline(lm90,col="green")
title("Quantile regression fit (black) \n versus linear regression of quantiles (green) \n for
the 90% quantile")
text(x=c(1992,1991),y=c(46,41.2),c("slope=0.42 m/s/yr","m=.36 m/s/yr"))
rq90=round(resid(qr90),10)
cat("Quantile Regression Residuals\n")

r1=c(sum(rq90<0),sum(rq90<=0))/length(resid(qr90))  #.90 is between these
observations we check out o.k. here with quantile regression
cat("Percentage of residuals less than or less than or equal to 0\n")
round(100*r1,2)
quantile(prob=.90, rq90)

rl90=round(IO$Wmax-predict(lm90,newdata=IO),10)
r2=c(sum(rl90<0),sum(rl90<=0))/length(resid(qr90))  #This is too low we only have 84%
of observations below or equal fitted line.
cat("Linear Model Residuals using all data points for prediction\n")
cat("Percentage of residuals less than or less than or equal to 0\n")
round(100*r2,2)
quantile(prob=.90, rl90)
```

Summary

Quantile regression provides a more complete picture of how the distribution of the response variable is conditioned on the values of the covariates.

5. Additional Reading and References

Baclawski, K., 2008: *Introduction to Probability with R*, Chapman & Hall/CRC.
Crawley, M.J., 2007: *The R Book*, John Wiley & Sons.
Dalgaard, P., 2002: *Introductory Statistics with R*, Springer.
Maindonald, J.H., 2004: Using R for Data Analysis and Graphics: Introduction, Code and Commentary.
Murrell, P., 2006: *R Graphics*, Chapman & Hall/CRC.
Tufte, E.R, 2001: *The Visual Display of Quantitative Information*, 2nd Ed., Graphics Press.
Verzani, J., 2004: *Using R for Introductory Statistics*, Chapman & Hall/CRC.
R Reference Card
UCLA Academic Technology Services http://www.ats.ucla.edu/stat/r