Hydrologic Statistics©

A course by

William H. Asquith, Ph.D., P.G.

Center for Research into Water Resources
Austin, Texas

Fall 2007

Manuscript Version Fall 2007

CONTENTS

INTRODUCTION

A.1 Getting Started 1

A.2 A Graphical Example 3

A.3 Dataframes 5

BASIC SUMMARY STATISTICS

B.1 The Mean, Standard Deviation, Coefficient of Variation 7

B.1.1 Hodges-Lehmann Estimator of the Mean 7

B.1.2 Bias in the Standard Deviation 8

B.1.3 Confidence and Prediction Intervals 9

B.1.4 Confidence Intervals for the Mean 10

B.2 Median 11

B.3 Range and the Interquartile Range 11

B.4 Skew and Kurtosis 12

B.5 Boxplots in Hydrology 14

FLOW-DURATION CURVES

C.1 Flow-Duration Curves 18

C.2 Plotting Positions 18

C.3 Better Graphical Presentation of Flow-Duration Curves . . . 21

DISTRIBUTIONAL ANALYSIS–Classical Moments

D.1 Moments . . . 22

D.2 Method of Moments . . . 22

DISTRIBUTIONAL ANALYSIS–L-moments

E.1 Linear Moments (L-moments) . . . 25

E.1.1 L-moments . . . 25

E.1.2 R package `lmomco` . . . 26

E.2 The Method of L-moments . . . 26

HYPOTHESIS TESTINGS—Change of Location

F.1 Comparing Normally Distributed Data—Student's t-Test . . . 34

F.1.1 The Welch t-Test in R . . . 35

F.1.2 Wilcoxon Rank Sum Test in R . . . 37

REGRESSION

G.1 Line of Organic Correlation . . . 38

G.2 Multiple-Linear Regression . . . 43

G.2.1 Ordinary Least Squares Regression . . . 44

G.2.1.1 Evaulation of Multicollinearity of Variables . . . 47

G.2.1.2 Three Dimensional Scatter Plots . . . 49

G.2.2 Weighted Least Squares Regression . . . 50

G.3 Presentation of Regression in a Hydrologic Report 52

TREND EVALUATING

H.1 Kendall's τ 54
H.2 LOWESS and LOESS Trend lines 55
H.3 The Theil Line 57

SOME HYDROLOGIC STATISTICS IN TEXAS

FIGURES

A.1 Simple R session to compute the mean of some data, demonstrate built-in help features, and exit R 2
A.2 Simple R session to demonstrate that R is a bonafide programming language 3
A.3 Simple R session demonstrating generation of a plot 3
A.4 Plot from R code in figure A.3 4
A.5 Simple R session demonstrating generation of a plot as a PDF to the operating system 4
A.6 R code demonstrating simple operations on a built-in data frame 5
A.7 R code demonstrating row and column operations on a built-in data frame 5
A.8 R code demonstrating attachment and detachment of a built-in data frame 6
A.9 R code demonstrating how a data frame is constructed by hand 6

B.1 R code demonstrating computation of mean, standard deviation, and coefficient of variation 7
B.2 R code demonstrating the Hodges-Lehmann estimator of the mean for a contaminated data set 8
B.3 R code demonstrating bias in estimates of standard deviation 9

B.4 R code demonstrating computation of confidence intervals of the mean 10
B.5 R code demonstrating computation of mean and median 11
B.6 R code demonstrating computation of range 11
B.7 R code demonstrating computation of interquartile range and the summary command 11
B.8 R code demonstrating redirection of textual output to a file using the sink function 12
B.9 R code demonstrating computation of skew and kurtosis 12
B.10 R code demonstrating computation of classical and unbiased skew 13
B.11 R code demonstrating computation of classical, unbiased, and excess-unbiased kurtosis 14
B.12 R code demonstrating computation of mean and median 14
B.13 Plot from second call to boxplot() in figure B.12 15
B.14 Example of a boxplot description 16
B.15 Example of high-quality typesetting of boxplots 16

C.1 R code to generate a flow-duration curve for a data set 19
C.2 More complex R code to generate flow-duration curves for two data sets 20
C.3 Plot from R code in figure C.2 20
C.4 Example of a boxplot description 21
C.5 Example of high-quality typesetting of boxplots 21

D.1 R code to demonstrate method of mometns 23
D.2 R code to demonstrate method of moments using a large standard deviation 23
D.3 R code to demonstrate method of moments using a streamflow dataset 24
D.4 R code to demonstrate method of moments using a streamflow dataset 24

E.1 Partial listing of annual peak streamflow data 27

E.2 R code to load an visualize an annual peak streamflow dataset 27
E.3 R code to load an visualize an annual peak streamflow dataset 28
E.4 Plot from R code in figure E.3 but not including graphical code in figure E.5 28
E.5 R code to demonstrate computations of L-moments using annual peak streamflow dataset after the code in figure E.3 29
E.6 R code to demonstrate computations of L-moments using annual peak streamflow dataset 30
E.7 Plot from R code in figure E.6 31
E.8 R code to demonstrate development of an L-moment ratio diagram 31
E.9 Plot from R code in figure E.8 32
E.10 Plot of an outstanding L-moment ratio diagram used to assess distribution fit 33

F.1 R code to read in and plot annual mean streamflow values in preparation for a t-test. 35
F.2 Plot from R code in figure F.1 36
F.3 R code to perform a t-test using data from figure F.1 36
F.4 Listing of the results of the t-test from R code in figure F.3 36
F.5 R code to demonstrate application of the Wilcoxon rank sum test 37
F.6 Listing of the results of the Wilcoxon test from R code in figure F.5 37

G.1 Example of an excellent scatter plot 39
G.2 R code demonstrating computation of the line of organic correlation 41
G.3 Plot of line of organic correlation for first plot in example R code in figure G.2 42
G.4 R code demonstrating computation of the line of organic correlation 42
G.5 R code to visualize some annual peak streamflow data 43
G.6 Plot of four plot ensemble from R code in figure G.5 44
G.7 R code to compute three distrinct multiple-linear regression models from some annual peak streamflow data 45

G.8 Plot of four plot ensemble from R code in figure G.7 . . . 46
G.9 R code to compute show table of regression diagnostics . . . 47
G.10 R code to compute three distrinct multiple-linear regression models from some annual peak streamflow data . . . 48
G.11 Plot of `pairs()` function from example R code in figure G.10 . . . 48
G.12 R code to compute three-dimensionless scatter plot of watershed characteristics . . . 49
G.13 Plot of the first `scatterplot3d()` function from example R code in figure G.12 . . . 50
G.14 R code to compute three distrinct multiple-linear regression models from some annual peak streamflow data . . . 51
G.15 R code showing a suggested function to produce near ideal documentation of a regression equation . . . 53

H.1 R code to load and conduct Kendall's τ correlation on annual peak streamflow dataset . . . 55
H.2 Listing of the results of the Mann-Kendall from R code in figure H.1 . . . 55
H.3 R code to load an visualize and annual peak streamflow dataset . . . 56
H.4 Plot from R code in figure H.3 . . . 56
H.5 R code to compute and plot the Theil line on an annual peak streamflow dataset . . . 58
H.6 Plot from R code in figure H.5 . . . 58

LESSON A: INTRODUCTION

This course concerns a generalized overview of statistical analysis associated with hydrology. This course is an experiment of sorts for me and for you. I am intentionally avoiding heavy math and considerable theory. I aim to use our collective education, experience, and intuition to explore solutions to hydrologic problems. As a rule of thumb, I state that statistics generally tells us what a reasonable mind already knows. (Well some problems truly are too big to get the mind around.)

In this context, we are brought to the use of the R environment for statistical computing as a statistics teaching aid. Whereas I am not striving (necessarily) to teach R itself, as you will see in the next few hours, the command line oriented nature of R permits very detailed demonstration of numerous statistical concepts in a fashion that should prove beneficial to all and is portable. The R environment has become the standard language for teaching and research within the broader statistics community. The R environment is open-source and runs on a myriad of platforms and is fun to use. I am hopeful that the many examples in this course can be cut and pasted from the PDF version of this handout into R so you can proceed with your own followup as you see fit.

I also intend this course to be an opportunity to tell stories and share experiences related to my nearly 15 years of statistical practice. DISCLAIMER: I am trained twice over in civil engineering and my Ph.D. is in geosciences. I work primarily in surfacewater hydrology with a focus on hydrologic statistics and development of procedures for end users such as transportation engineers. My interests reside in the realm of computational methods and tricks to accomplish investigation and research goals. Although I have had several courses myself and have a robust statistics library, I am largely self/colleague taught in the statistical discipline. Therefore, there are statistical topics that some of you could be much more knowledgeable. For this course, I will to emphasize those areas that I am most adept, and use as part of my hydrologic research.

A.1 Getting Started

We will start by assuming that you have R installed and operational on your machine, but I do not assume that you have a machine with you today—I want you to grasp the statistical concepts and contribute to discussion. However, presentation of some R functionality is needed.

Let us begin by an introduction and discussion of basic operations of a small number of R commands. R generally is (and is recommended to be) deployed via a command line interface; you will shortly see the reasons that I find this attractive. I work primarily on MacOSX and Linux because they are my production platforms; however, I am hopeful that demonstration of some statistics under Windows will occur.

R is built around the concept of vectorized arithmetic. For example, to add vectors a and b (or "columns" in spreadsheet parlance), one simply says a+b. You will find that after the initial hurdle of basic syntax that the ability to document, communicate, and archive your statistical operations is greatly enhanced by vectorized arithmetic.

Lets begin—the following figure contains approximately six commands to give you a feel for an R session. We generate some fake data, compute the mean, and print the value to the screen. The printing step can be shortened by simply saying **mean**(some.fake.Data).

```
some.fake.Data <- rnorm(30) # 30 samples from standard normal
    # distribution
the.mean <- mean(some.fake.Data) # compute it
print(the.mean) # print it

help(mean) # to learn more about the function
help.search("mean") # fuzzy matching of the term

q() # exit R, you are rolling!
```

Figure A.1. Example of a simple R session to compute the mean of some data, demonstrate built-in help features, and exit R

The code illustrates several important details of the presentation for this course. Note that the code listing, and further code listings in this handout, are in a monospaced font and have syntax highlighting. Common commands built-in to R are shown in bold type as in the **mean**, comments are shown in oblique type *# comment*, and all other code is in regular type. The algorithm that performs the highlighting only recognizes patterns and not meaning as you can see the **mean** highlighted in the variable name the.**mean**. This is a minor inconvenience at the benefit of a markup system that should make material easier to follow. Finally, spaces within strings are shown with the "␣" character for clarity.

For the introductory code above, we generate some fake data, which by definition should have a mean of zero using the **rnorm**() (random normal distribution). However, because we are simulating the normal distribution by a finite sample, the mean will deviate slightly from zero.

The figure continues with a demonstration of the help system on your platform by the **help**(**mean**), which provides the help content related to the **mean** function. The example in figure A.2 ends by the quit function **q**(). You will see time and time again the use of paired () even for simple operations—this represents the idea of "function" and is not optional. The **q**() will largely become optional for the remainder of examples in this course, and thus will not typically be shown.

```
some.fake.Data <- rnorm(30)
n <- length(some.fake.Data) # hope n == 30
if(n == 30) {
  cat(c("GREAT!","xxxx␣IT␣DOES\n"))
}
q() # type this in separately
```

Figure A.2. Example of a simple R session to demonstrate that R is a bonafide programming language. The example computes the mean of some data and tests the length of the vector

A.2 A Graphical Example

Graphics are a fundamental part of statistical analysis. In fact, I argue that they are the most critical tool. Good graphics are powerful, but good graphics do take some effort, and the effort is difficult to fake. Out-of-the-box R provides generally ok graphics, but natively are insufficient for reports and papers requiring high-quality typesetting. I will scatter (pun intended) throughout the course my thoughts and techniques for good graphics style and generation. But first, let us continue with what R can do.

```
some.fakeX <- rnorm(30)
some.fakeY <- some.fakeX - 10 + rnorm(30)
plot(some.fakeX,some.fakeY)

plot(some.fakeX,some.fakeY, pch=14, col=2,
     xlab="X␣data", ylim=c(-15,5))
```

Figure A.3. Example of a simple R session demonstrating generation of a plot

The first `plot()` operation should largely be self evident. The second `plot()` operation includes the alteration of the plotting character to a solid circle (`pch=14`) with color red (`col=2`). We add an X label by `xlab="X␣data"` and specify both limits of the Y axis by `ylim=c(-15,5)`.

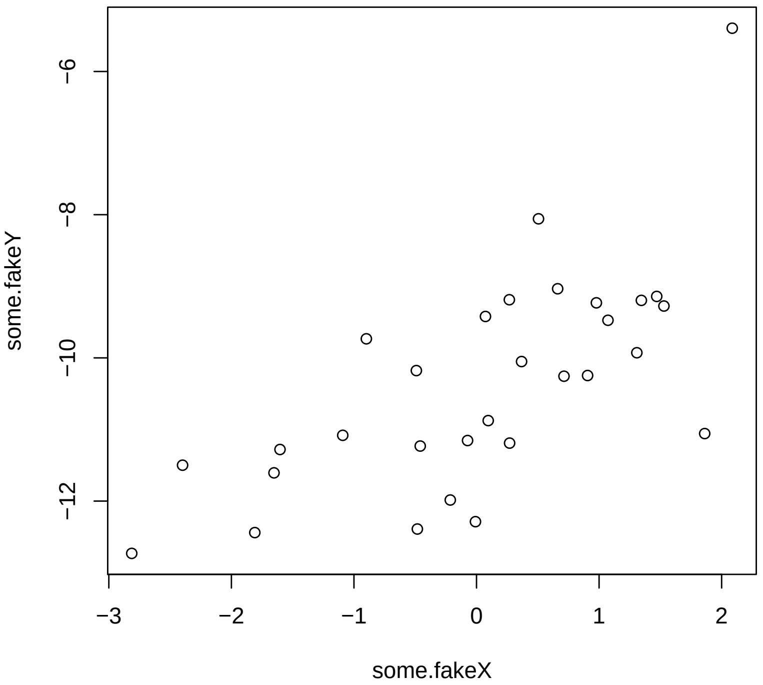

Figure A.4. Plot from R code in figure A.3

The use of the **c()** operator is introduced. It combines values into a vector or list structure that is fundamental to the vectorized operations of R. You will see **c()** used time and time again. In the context here, it combines the values -15 and 5 into a single value passed to the ylim argument of the **plot()** function. There are dozens of plotting functions and operations as well as numerous add-on packages deploying the capability of R in the graphics realm. Others will be seen throughout this course.

Often it is desirable to produce graphics, not to the screen, but as a stand alone file for inclusion in other documents. The following code example adds the pdf() and **dev.off()** functions. The pdf() function switches the graphics device from the screen to a subsystem that generates PDF files. The function takes several arguments, but the most important is the file name. The example uses yourfirst.pdf as the file name. The **plot()** call was shown earlier. The **dev.off()** command closes the device and reopens the screen and is not optional for this process to work.

```
some.fakeX <- rnorm(30)
some.fakeY <- some.fakeX - 10 + rnorm(30)
pdf("yourfirst.pdf")
  plot(some.fakeX, some.fakeY, pch=16, col=2,
       xlab="X␣data", ylim=c(-15,5))
dev.off()
```

Figure A.5. Example of a simple R session demonstrating generation of a plot as a PDF to the operating system

For high-quality typesetting, a PDF file such as that produced would be imported or otherwise opened in a drawing application and additional adjustments would be made. That processing is outside the scope of this course.

A.3 Dataframes

A data frame is a fundamental unit of information in R. It can be thought of as a simple spreadsheet with columns, rows, and headings. Data read from external files is automatically loaded into data frames. R provides many tools for advanced data frame manipulation that are beyond the scope of this course. R ships with many built-in datasets, let us use one to demonstrate a data frame. The following code has additional embedded comments to aid to our discussion.

```
data(airquality) # airquality is builtin, load it into your
    # workspace
print(airquality) # wow, that is a lot of data

# What are the columns in this data?
names(airquality) # 6 names will be shown

# Lets look at the wind data
airquality$Wind # the data is shown now in a horizontal multi-row
    # matrix

# Let us multiply the wind data by the temperature data.
# Do not know why one would do this, but . . .
a.product <- airquality$Wind*airquality$Temp
print(a.product)

# Now extract the 10th value of the product.
a.product[10]

# Print the 10th value of the Wind data
airquality$Wind[10]
```

Figure A.6. R code demonstrating simple operations on a built-in data frame

The example shows two techniques of variable referencing. First, airquality$Wind shows that the Wind is a named subset of the airquality data frame. Second the use of numeric or element lookup of the data vector a.product[10]. The example ends with a variation on each of these two methods.

Entire rows or columns of a data frame also are readily extracted:

```
data(airquality)
names(airquality)
airquality[10, ] # extract tenth row of the frame

airquality[, 3] # alternative way to extract the Wind data

airquality[10,3] # extract 10th Wind value
```

Figure A.7. R code demonstrating row and column operations on a built-in data frame

For this example, note the use of the commas in the square brackets and how to define the rows or columns. It is burdensome to type the name of the data frame, thus R offers the ability to attach the names of the data frame into your workspace.

```
data(airquality)
names(airquality) # it is a habit of mine to peak at the names
    # after a data frame is loaded.

attach(airquality)
  plot(Wind,Temp)
detach(airquality) # in some settings detachment is needed if other
    #  data frames will be attached as necessary. I am a little
    # vague on when this is mandatory.

# The plot call above is more curt than
plot(airquality$Wind,airquality$Ozone)
```

Figure A.8. R code demonstrating attachment and detachment of a built-in data frame

Data frames are easy to build by hand. Suppose we have four data vectors and desire to make a data frame out of them:

```
# First some random values
A <- c(123, 546, 345.2, 12,
       875, 321, 90, 800)

# Second multiple each value of A by unique random standard normal
    # values
B <- A*rnorm(length(A))

# Third generate a linear adjustment of A
C <- A*10 - 100 # note that negative 100 offset is cycled for all
    # values of vector A

# Finally build the data frame, demonstrating how names are made
    # and how c() can be used inline.
X <- data.frame(dataA=A, dataB=B, moredata=C,
                Ddata=c(12, 46, 32, 2, 85,
                        31, 0, 51))
print(X)
```

Figure A.9. R code demonstrating how a data frame is constructed by hand

LESSON B: BASIC SUMMARY STATISTICS

B.1 The Mean, Standard Deviation, Coefficient of Variation

The mean μ and standard deviation σ are well known and the most popular statistics of a data set. The ratio σ/μ is known as the coefficient of variation CV. The μ measures the location of the data on the real-number line and is further discussed in class. The σ measures the variation of the data about μ. The CV can be useful in some applications as it is an expression of relative variability. These two statistics also are further discussed in class.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma^2 = \frac{1}{(n-1)}\sum_{i=1}^{n}(x_i - \mu)^2$$

where n is the sample size, x_i is the ith observation. The -1 term in the σ^2 definition corrects for bias.

The following code demonstrates the computation of μ, σ, and CV for hand made data set.

```
madeupdata <- c(123, 546, 345.2, 12,
                875, 321, 90, 800)
mu <- mean(madeupdata)
sigma <- sd(madeupdata)
cv <- sigma/mu
cat(c(mu,sigma,cv,"\n"))
```

Figure B.1. R code demonstrating computation of mean, standard deviation, and coefficient of variation

This example reports all three values in the indicated order. A new function is used **cat()**. The function concatenates and prints a list of values—see how the **c()** is used to build this list. The "\n" value is a universal symbol used to denote a newline or return and enter keys on the keyboard.

B.1.1 Hodges-Lehmann Estimator of the Mean

A statistic is said to be *robust* if its sampling properties are resistant to the influence of extreme values. In particular if a data stream has the potential for erroneous data or the distribution of the phenomena understudy is heavy-tailed, then robust statistics are useful. If one is uncertain whether a distribution is symmetric, in which the arithmetic mean should perform well, then the best choice is the Hodges-Lehmann estimator (Good and Hardin, 2003). The estimator is defined as the median of the pairwise averages:

$$\check{\Delta} = \mathrm{median}_{i \leq j}(X_j + X_i)/2$$

The following code demonstrates the performance of $\check{\Delta}$. I define a function DELTA to compute the Hodges-Lehmann estimator. We simulate a sample of size 40 with **mean**=1500 and **sd**=2400. We subsequently contaminate the sample if a very large number—imagine that a decimal got off by accident. We compute the mean by **mean()** and then compute $\check{\Delta}$.

```
# The Hodges-Lehmann Estimator
DELTA <- function(x) {
  tmp <- vector(mode="numeric")
  n   <- length(x)
  count <- 0
  for(i in seq(1,n-1)) {
    for(j in seq(i,n)) {
      count <- count + 1
      tmp[count] <- (x[i]+x[j])/2
    }
  }
  DELTA <- median(tmp)
  return(DELTA)
}

# Begin the meat of the example. . .
# Run this portion again and again to see the performance of DELTA
    # . You will see again and again that DELTA is closer to 1500
    #  than mu.
X <- rnorm(40,mean=1500,sd=2400)
X[length(X)+1] <- 14000 # fake a hideously bad data value
mu  <- mean(X)
DELTA <- DELTA(X)
cat(c("Arithmetic␣Mean=",mu,"\n",
      "Hodges-Lehmann␣Estimator␣=",
       DELTA,"\n"),sep="")
```

Figure B.2. R code demonstrating the Hodges-Lehmann estimator of the mean for a contaminated data set

B.1.2 Bias in the Standard Deviation

The standard deviation σ that all of us should be very familiar with is computationally simple as shown in the previous section. The division by n in the mean seems reasonable enought, but why do we have the $(n-1)$ term? Does this mean that we do not compute the average square deviation? Yes, it does. Years ago I was simple told in class that well you give up a degree of freedom because the mean itself requires estimation. Ok—but what does that mean? I was never quite satisfied until many years later when I was studying L-moments and the idea of bias in a statistic that the true meaning of the $(n-1)$ term was made manifest. With a simple n term in the denominator, we under estimate σ^2. The following R code concretely demonstrates this idea. The code involves the idea of statistical simulation, sampling error, and statistics of statistics. We will discuss each element in detail in the classroom. Suffice to say that if you run the following code, you will confirm that the $(n-1)$ definition of standard deviation that is built-in to the **sd()** function provides a closer estimate to **sd**=10000.

```
 # two vector to hold sample estimates of standard deviation
 bias.tmp   <- vector(mode="numeric")
 unbias.tmp <- vector(mode="numeric")
 count <- 1 # a counter for the vectors
 sample.size <- 30 # a modest sample size
for(sim in seq(1,1000)) {
   X  <- rnorm(sample.size,mean=0,sd=10000) # a large standard
        # deviation?
   mu <- mean(X) # compute the sample mean of the count-th
        # simulation

   bias.sigma   <- sqrt(sum((X-mu)^2)/sample.size) # theoretical
        # definition
   unbias.sigma <- sd(X) # the unbiased estimate

   bias.tmp[count]   <- bias.sigma
   unbias.tmp[count] <- unbias.sigma
   count <- count + 1
}
# compute the summary of each vector of simulated standard
    # deviations
summary(bias.tmp)
summary(unbias.tmp)
```

Figure B.3. R code demonstrating bias in estimates of standard deviation

B.1.3 Confidence and Prediction Intervals

The sample mean is known as a point estimate. Alternatively, interval estimates posses a declared probability of containing the true population value (Helsel and Hirsch, 1992). Interval estimates can provide two bits of information that points estimates can not:

1. Declaration of probability that the interval contains to true population value—CONFIDENCE INTERVALS
2. Declaration of the probability that an individual value comes from the population under study—PREDICTION INTERVALS

The above concepts and terms are interrelated, they are not the same and can not be interchanged. Consult good statistical textbooks for clarity on the matter. Some of these books are listed in the Selected References of this course. We will briefly consider prediction intervals in the context of linear regression later in this course.

B.1.4 Confidence Intervals for the Mean

Interval estimates for the true mean μ are easy to compute. The symmetric intervals around the sample mean $\overline{X}$ are common. Samples with large n the symmetric intervals described here are adequate regardless of the distribution of the data. For small samples this is not true unless the distribution of the data is in fact normal.

The confidence intervals for the $1-\alpha$ interval require tables of the t-distribution or the R function **qt()**. For highly skewed data or data with outliers the classical statistics assumptions of the t-distribution are invalid, but for this course the following will suffice.

The confidence interval for the mean are:

$$\overline{X} - t_{[\alpha/2,n-1]}\sqrt{s^2/n} \leq \mu \leq \overline{X} - t_{[\alpha/2,n-1]}\sqrt{s^2/n}$$

where s is the sample standard deviation and $t_{[\alpha/2,n-1]}$ is the quantile function of the t-distribution. The computation of the intervals is shown in the following code.

```
madeupdata <- c(123, 546, 345.2, 12,
                875, 321, 90, 800)
n    <- length(madeupdata) # sample size
Xbar <- mean(madeupdata) # sample mean
sbar <- sd(madeupdata) # sample stand. dev.

# we want the 90-percent interval
alpha <- 1 - 0.90; a2 <- alpha/2
# notice how the semicolon permits more than one command per line

# need left and right tails of the distribution
the.t.LEFT  <- qt(a2,n-1)
the.t.RIGHT <- qt(1-a2,n-1)
tmp <- (sbar**2/n)**0.5

# the sign is explicitly attached to the.t.LEFT and the.t.RIGHT
    # values thus the use of + in both lower and upper intervals
lower <- Xbar + the.t.LEFT*tmp
upper <- Xbar + the.t.RIGHT*tmp
cat(c("␣␣␣␣␣␣␣␣␣",Xbar,"␣␣␣␣␣␣␣\n"))
cat(c("LOWER␣=␣",lower,"␣and␣",
      "UPPER␣=␣",upper,"\n"))
```

Figure B.4. R code demonstrating computation of confidence intervals of the mean

The essence of the above code is the application of the t-distribution. The function **qt()** returns the cumulative lower tail of the distribution. Hence special accommodation for this is made by a break in the verbatim application of the confidence interval equation shown above. One must be careful about how percentiles, tails, and αs are used in a particular function regardless of software package. You should also be able to readily verify that an arbitrary software package does in fact implement the computations correctly as we have here.

B.2 Median

The median is another well known statistic that measures the location of the data. The median is the 50th percentile. That is, it is expected that 50 percent of observations would be less than the median and 50 percent would be larger. The median is known as a nonparametric statistic that is based on the ranks of the data. The median is readily computed in R. For the example, the middle value is computed as the mean between 345.2 and 321 because the sample size is an even number.

```
madeupdata <- c(123, 546, 345.2, 12,
                875, 321, 90, 800)
mu <- mean(madeupdata)
med <- median(madeupdata)
cat(c(mu,med,"\n"))
```

Figure B.5. R code demonstrating computation of mean and median

B.3 Range and the Interquartile Range

The range is a statistic computed by the difference between the maximum and minimum values. The range is an alternative measure of data variability, but does not have particularly attractive properties. The interquartile range *IQR* is the difference between the 75th percentile (third quartile) minus the 25 percentile (first quartile). The range has a dedicated function

```
madeupdata <- c(123, 546, 345.2, 12,
                875, 321, 90, 800)
range(madeupdata)
```

Figure B.6. R code demonstrating computation of range

However, the *IQR* is readily computed using a higher level function **summary()** through value extraction. The computation of *IQR* is:

```
madeupdata <- c(123, 546, 345.2, 12,
                875, 321, 90, 800)
the.summary <- summary(madeupdata)
iqr <- the.summary[5] - the.summary[2]
print(iqr)

# How did I know to use the 2nd and 5th values?
# Well summary() reports five statistics.
print(the.summary)
# Even cooler, we can bipass the print()
the.summary
```

Figure B.7. R code demonstrating computation of interquartile range and the summary command

R is a command line oriented program, thus redirection of computational results should readily be pushed into other files. Indeed this is easy using the **sink()** function.

```
madeupdata <- c(123, 546, 345.2, 12,
                875, 321, 90, 800)
the.summary <- summary(madeupdata)
sink("a_text_file.txt") # has many additional parameters
print(the.summary)
sink() # close the file off
```

Figure B.8. R code demonstrating redirection of textual output to a file using the sink function

We can now open `a_text_file.txt` in a text editor or other appropriate software on your platform of choice. When files are produced in this fashion, they can readily be picked up by other software for further computation. R has much more powerful tools for data exporting than shown by the tiny example above. A starting point for adventurous folks is the **`write.table()`** function.

B.4 Skew and Kurtosis

Skew (γ) and kurtosis (κ) are known as higher moments. They have particularly poor properties in analysis of hydrologic data sets such as floods and other data exhibiting high outliers. R apparently lacks built-in functions for computation of these statistics, but the `moments` package that is available online does. The **`library()`** function is used to load in one or more of the nearly 1,000 contributed packages to R that are available online. To demonstrate by example, let us generate a random normal sample of size 30 with a $\mu = 100$ and $\sigma = 200$ and compute skew and kurtosis.

```
 madeupdata <- rnorm(30, mean=100, sd=200) # 30 normal samples
 library(moments) # your system will only have the this package if
      # you have downloaded and installed it. We need this package
      # for the skewness() and kurtosis() functions.
 g <- skewness(madeupdata)
 k <- kurtosis(madeupdata)

# Let us look at the definition of skewness and kurtosis
skewness
kurtosis
```

Figure B.9. R code demonstrating computation of skew and kurtosis

See how the open-source natre of R permits us to peak at how things are actually computed? Anyway the computations seem easy enough don't they? *No the don't.* Software packages and even some hard copy sources often are insufficiently versed in either statistical theory, practice, or documentation. As the above example shows the definition of γ and κ are simple enough to understand and cross check against text-book sources (Dingman, 2002). Using Dingman (2002, p. 559) lets role our own functions in R. The R codes for γ and κ are shown in the next two code examples.

The following figure shows development of a function `my.skewness()` to compute the classical definition of γ and the unbiased version.

The unbiased version is what we own to use in practice. You will note that the classical estimate is always less than the unbiased estimate. The difference prepresents the inherent bias in the mathematics. We want to use unbiased estimators in hydrologic studies. The example generates another random sample and computes γ using `skewness()` of the `moments` package and then continues on to compute `my.skewness ()`. Inspection of the output will show that the `skewness()` computes a biased version of γ. An easy mistake to make if one does not consult sources or the manual.

```
my.skewness <- function(x) {
    n   <- length(x)
    mu  <- mean(x)
    sig <- sd(x) # unbiased standard deviation

    M2  <- sqrt(sum((x-mu)^2) / n)
    M3  <- sum((x-mu)^3) / n
    classic.skew  <- M3 / M2^3
    unbiased.skew <-
                   n^2*M3
                      /
             ((n-1)*(n-2)*sig^3)

    return(list(classic  = classic.skew,
                unbiased = unbiased.skew))
 }
 madeupdata <- rnorm(30, mean=100, sd=200)
 library(moments)
 g <- skewness(madeupdata)
 print(g)
 z <- my.skewness(madeupdata)
 print(z)
```

Figure B.10. R code demonstrating computation of classical and unbiased skew

The following shows development of a function `my.kurtosis()` to compute the classical definition of κ, the unbiased version, and another κ known as excess kurtosis. The unbiased version or excess κ is what we need to use in practice. It is not always clear which κ or excess κ is needed for a particular situation involving application of distributions—This is a serious shortcoming in the statistical discipline. You will note that the classical estimate is always less than the unbiased estimate. The difference prepresents the inherent bias in the mathematics. We want to use unbiased estimators in hydrologic studies. The example generates another random sample and computes κ using `kurtosis()` from the `moments` package and then continues on to compute `my.kurtosis()`. Inspection of the output will show that `kurtosis()` computes a biased version of skew represent nonexcess κ. Another easy mistake to make if one does not consult sources or the manual.

```
my.kurtosis <- function(x) {
    n   <- length(x)
    mu  <- mean(x)
    sig <- sd(x) # unbiased standard deviation
```

```
   M2  <- sqrt(sum((x-mu)^2) / n)
   M4  <- sum((x-mu)^4) / n
   classic.kurt  <- M4 / M2^4
   unbiased.kurt <- n^3*M4
                      /
           ((n-1)*(n-2)*(n-3)*sig^4)

   # the excess kurtosis term is seen in some software packages
       # including your typical spreadsheet software
   exc <- (3*(n-1)^2)/((n-2)*(n-1))

   return(list(classic = classic.kurt,
               unbiased = unbiased.kurt,
               excess.unbiased =
                   unbiased.kurt - exc))
}
madeupdata <- rnorm(30, mean=100, sd=200)
library(moments)
k <- kurtosis(madeupdata)
print(k)
z <- my.kurtosis(madeupdata)
print(z)
```

Figure B.11. R code demonstrating computation of classical, unbiased, and excess-unbiased kurtosis

B.5 Boxplots in Hydrology

Boxplots are a highly useful tool for presentation of distributions of data. Particularly so when discrete groups or classifications of data exist. Box plots permit a compact visual representation of the distribution of the data. One or two boxplots are perhaps better shown as listed values in a table. However, as the number of boxes or catagories for a large sample becomes large greater clarity in the presentation is made through boxplots. Lets compute a view, discuss, and then I will show some really good ones.

```
madeupdata <- c(123, 546, 345.2, 12,
                875, 321, 90, 800)
boxplot(madeupdata)

help(boxplot)

data(airquality)
boxplot(airquality)

boxplot(airquality$Wind ~ airquality$Temp,
        xlab="Wind,␣in␣m/s",
        ylab="Temperature,␣in␣degC")
mtext("Temperature␣=␣func(wind␣speed)")
```

Figure B.12. R code demonstrating computation of mean and median

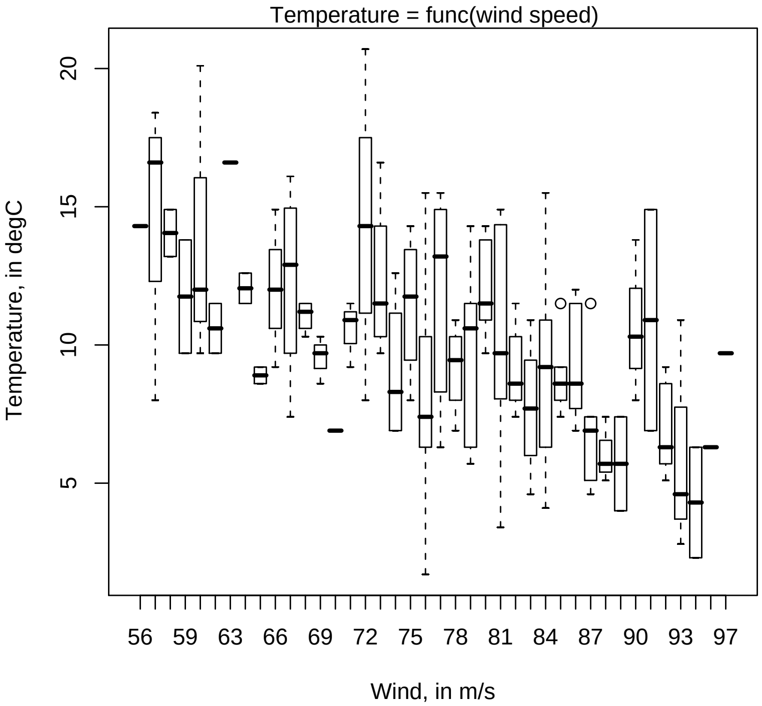

Figure B.13. Plot from second call to boxplot() in figure B.12

A major short coming in all software packages that I have seen, with the exception of my own Tkg2, is the insufficient graphical description of the actual boxes generated is made. Figure B.14 provides an outstanding example of comparatively simple boxes in which the conditional adjustment of the plotting is required to accommodate small sample sizes. The boxes described in figure B.14 are shown in figure B.15. The description required hand construction and placement of four unique types of boxplot explanation.

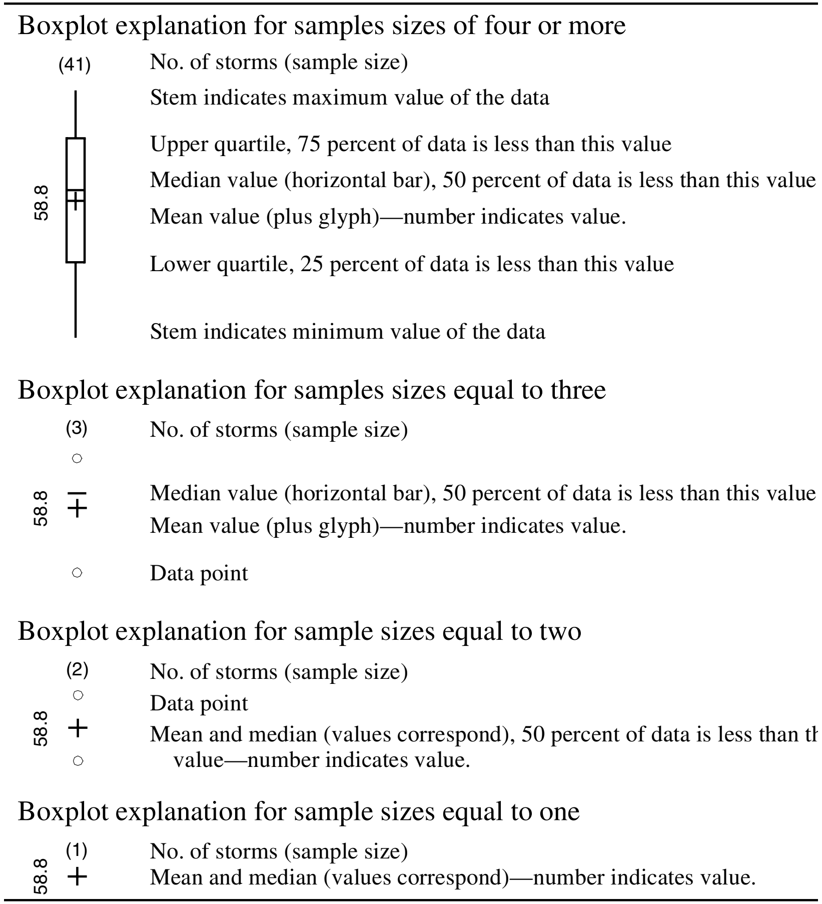
Boxplot explanation for samples sizes of four or more
(41)
No. of storms (sample size)
Stem indicates maximum value of the data
Upper quartile, 75 percent of data is less than this value
58.8
Median value (horizontal bar), 50 percent of data is less than this value
Mean value (plus glyph)—number indicates value.
Lower quartile, 25 percent of data is less than this value
Stem indicates minimum value of the data
Boxplot explanation for samples sizes equal to three
(3)
No. of storms (sample size)
58.8
Median value (horizontal bar), 50 percent of data is less than this value
Mean value (plus glyph)—number indicates value.
Data point
Boxplot explanation for sample sizes equal to two
(2)
No. of storms (sample size)
Data point
58.8
Mean and median (values correspond), 50 percent of data is less than tl
value—number indicates value.
Boxplot explanation for sample sizes equal to one
(1)
No. of storms (sample size)
58.8
Mean and median (values correspond)—number indicates value.

Figure B.14. Example of a boxplot description

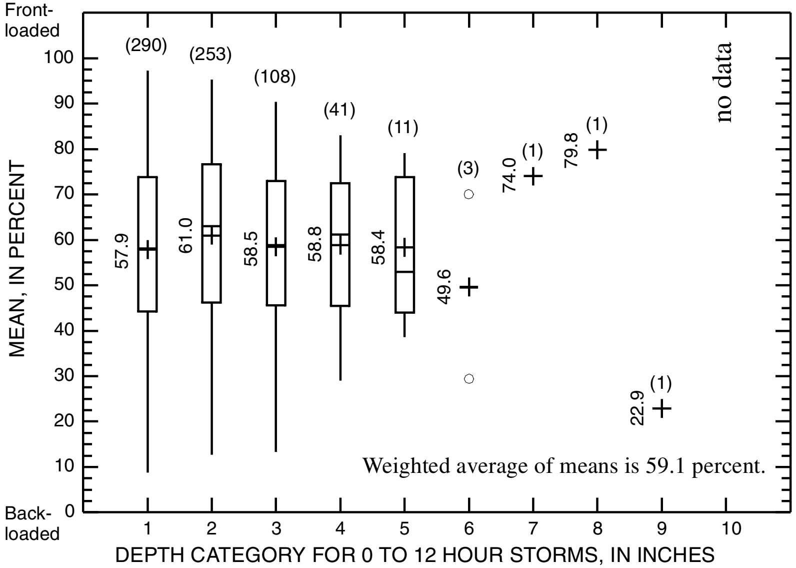

Figure B.15. Example of high-quality typesetting of boxplots

In conclusion, I would like to add a few comments about good graphics. Graphics in a report are incredible tools to convey information and greatly ease the burden of lengthy written discription. The following list provides some guidance that I felt compelled to mention as I write this course.

1. Seldom will a graphics package provide near perfect typesetting. Be prepared to post process your graphics in a vector editing software package.

2. Read sources with description of proper graphic style in scientific and technical writing. Common spreadsheets make abysmal graphics—do not use them.

3. Minimize ink usage. Ink alone conveys neither information or wisdom. The intersection of arbitrary ink lines and dots only conveys information by reader's previous exposure, context, and written description. *Our collective educational and cultural experiences bring meaning to scientific graphics.*

4. Virtually avoid color at all costs. Although I am using color liberally today, it is not good practice for scientific writing. Creativity stems from constraint, constrain yourself from color and your graphics will improve.

5. Do not use histogram, pie charts, and 3-D effects. (3-D plots are excluded.)

LESSON C: FLOW-DURATION CURVES

C.1 Flow-Duration Curves

Flow-duration curves are simple, yet highly informative graphical summaries of the variability of a (daily) time series. A flow-duration curve is a graph plotting the magnitude of a variable Q verses fraction of time the Q does not exceed a specified value $Q(F)$. The fraction of time can be thought of as probability and cumulative fraction of time is termed nonexceedance probability F. The probability refers to the frequency or probability of nonexceedance (or exceedance) in a "suitably long" period of time rather than probability of exceedance on a specific time interval (daily).

The area under the flow-duration curve is equal to the average for the period. Other statistics or statistical concepts visible include: median, quartiles, other percentiles, variability, and skewness. Steeper curves are associated with increasingly variable data. The slopes and changes in the slope of the curves can be important diagnostics of streamflow conditions in a watershed.

Flow-duration curves for neighboring stations yield valuable insights into hydrologic or hydrogeologic processes. For natural streams, the slope of the flow-duration curve for upper end is determined by regional climate and characteristics of large precipitation events. The slope of the lower end is determined by geology, soils, and topography. The slope of the upper end is relatively flat where snowmelt is the principal cause of floods and for large streams where floods are caused by long duration storms. Flashy watersheds and watersheds effected by short duration storms have steep upper ends. A flat lower end slope usually indicates that flows come from significant storage in ground water aquifers or frequency precipitation inputs.

C.2 Plotting Positions

Plotting positions are used to define the nonexceedance probability or cumulative percentages of data points in a sample. Plotting positions are used to define flow-duration curves, generate probability graph paper, and compare two distributions. To generate such plots:

1. Order data $x_1 \leq x_2 \leq \ldots \leq x_n$
2. Rank'em $1, 2, \ldots, i, \ldots, n$ (i is rank)
3. $F(x)$ = nonexceedance probability or just the percentile

4. $1 - F(x)$ = exceedance probability

The general formula for plotting positions is

$$F(x) = \frac{i-a}{n+1-2a}$$

where i is ascending rank, a is the plotting position coefficient, and n is the sample size.

The true probability associated with the largest (and smallest) observation is a random variable with mean $1/(n+1)$ and a standard deviation of nearly $1/(n+1)$. Hence, all plotting position formulas give crude estimates of the unknown probabilities associated with largest and smallest events. The plotting position coefficient can acquire several different values. But for our purposes $a = 0$ the so-called Weibull plotting position, which is the most(?) for magnitude and frequency analysis. You can convince yourself that as n becomes large that choice of a becomes unimportant.

```
D1 <- read.table("pp1.txt",fill=TRUE,
                 header=TRUE)
names(D1)
streamflow <- D1$dv_va
streamflow <- sort(streamflow)
n <- length(streamflow)
rank <- seq(1,n)
plot(rank/(1+n),streamflow,type="l")

# perform this command separately from above to see both plots
plot(rank/(1+n),
     log10(streamflow), type="l")

# add the qnorm to make normal probability paper on the
    # horizontal axis
plot(qnorm(rank/(1+n)),
     log10(streamflow), type="l")
```

Figure C.1. R code to generate a flow-duration curve for a data set

```
# not the addition of the comment character argument
# read.table defaults to comment.char="#", but read.csv does not.
D2 <- read.csv("pp2.csv",
               comment.char = "#",
               header=TRUE)
names(D2)

D3 <- read.csv("pp3.csv",
               comment.char = "#",
               header=TRUE)
names(D3)

makeFDC <-
  function(flow) {
    flow <- sort(flow) # ascending sort
    n <- length(flow) # how many data points
    rank <- seq(1,n) # sequence of increasing numbers
    pp <- rank/(1+n) # the plotting position
    return(list(PP=pp,FLOW=flow))
}

FDC2 <- makeFDC(D2$VALUE)
FDC3 <- makeFDC(D3$VALUE)

plot(qnorm(FDC2$PP), log10(FDC2$FLOW),
     type="l", xlim=c(-2,4), col=4,
     xlab="STANDARD␣NORMAL␣DEVIATE",
     ylab="log10␣STREAMFLOW,␣IN␣CFS") # BLUE LINE
lines(qnorm(FDC3$PP), log10(FDC3$FLOW),
      col=2) # RED LINE
```

Figure C.2. More complex R code to generate flow-duration curves for two data sets

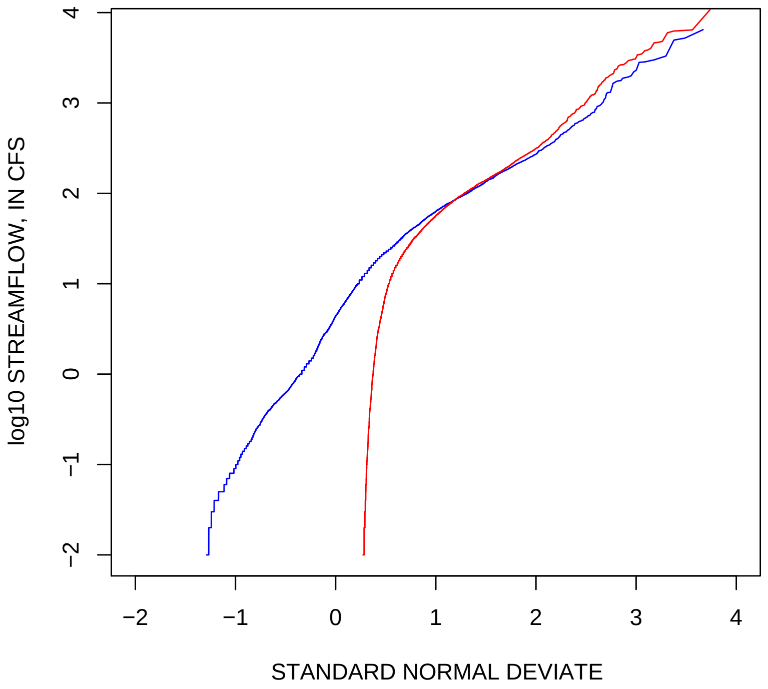

Figure C.3. Plot from R code in figure C.2

C.3 Better Graphical Presentation of Flow-Duration Curves

Graphical presentation of flow-duration curves is not a trivial task with most software. Years ago I wrote a Perl based program called Tkg2 just for this very task. The software runs extremely well under Unix, Linux, and MacOSX, milage will very on Windows because of inherent shortcomings in that operating system. However, the following two figures were generated for this course and depict the flow-duration curves for the subject daily streamflow data.

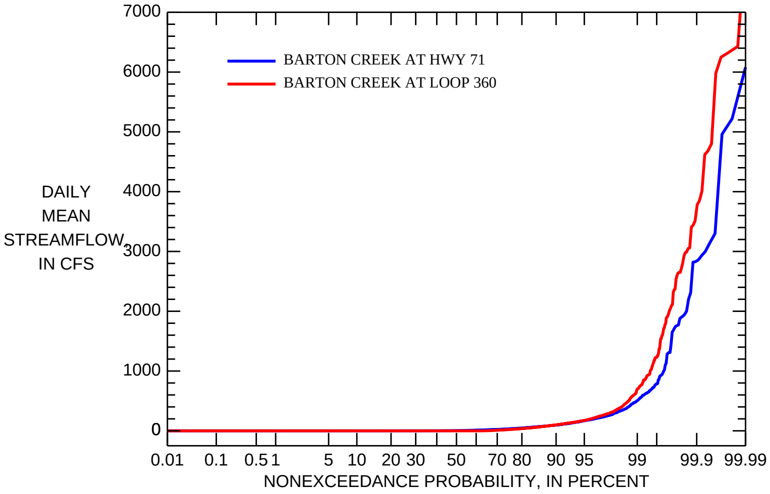

Figure C.4. Example of a boxplot description

Figure C.4 shows the flow duration curve with a properly built probability axis instead of the standard normal deviates provided by R. The streamflow axis is linear. Figure C.5 on the other hand depicts the streamflow axis in log10 space. What differences do you see between the two plots. How are one's interpretations of watershed behavior influenced by the choice of plotting method?

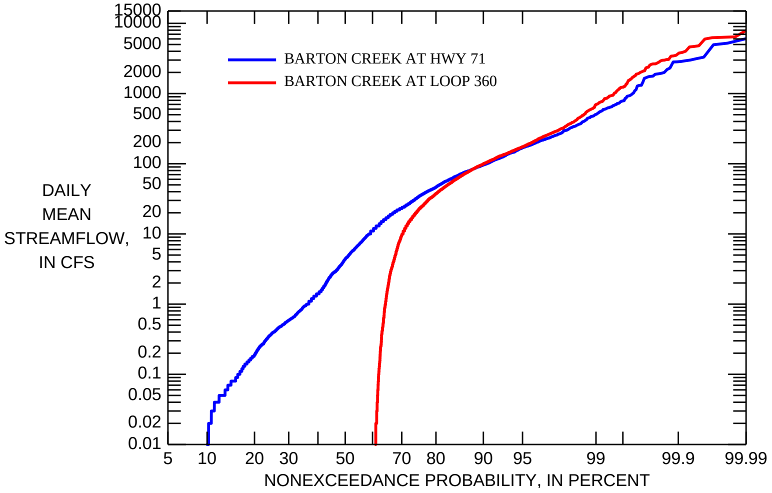

Figure C.5. Example of high-quality typesetting of boxplots

Now both figures C.4 and C.5 still need a few tweaks here and there. Primarily font rotation on the vertical axis label. But that is a job of post processing software like Illustrator or Inkscape.

LESSON D: DISTRIBUTIONAL ANALYSIS–Classical Moments

D.1 Moments

We have already played with classical or product moments in easlier lessons. These are the mean, standard deviation, skew, and kurtosis. Moments are powerful statistics that reduce data into a few values. The mean is the most important and it can be argued that higher and higher moments provide decreasingly important information of the distribution of the data.

The product moments raise data to a power—hence, "products" of the data are generated.

$$M_1 = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$M_2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - M_1)^2$$

$$M_3 = \frac{1}{n}\sum_{i=1}^{n}(x_i - M_1)^3$$

$$M_4 = \frac{1}{n}\sum_{i=1}^{n}(x_i - M_1)^4$$

where M_n is the nth moment. The higher moments are often standardized by division with $(\sqrt{M_2})^n$.

In hydrology, the application of the product moments is not always satisfactory and the L-moments are preferred.

D.2 Method of Moments

Distributions are specified by theoretical moments, and distributions of data are specified by sample moments. When the theoretical moments of the distribution are equated to the sample moments through solution of the distribution parameters, then the distribution is said to be "fitted" to the data. The normal distribution is a simple example.

The normal distribution is a two parameter distribution whose two parameters exactly equal the mean and the standard deviation. The following example demonstrates the fitting of the normal distribution to some data generated from a normal distribution. This is an example in which the data is known to be "drawn" from the normal.

```
just.some.data <- rnorm(24,mean=3400,sd=5000)
just.some.data <- sort(just.some.data)
n  <- length(just.some.data)
pp <- seq(1,n)/(n+1) # plotting positions

mu <- mean(just.some.data) # sample mean
sd <- sd(just.some.data)   # sample standard deviation

fitted.normal <- qnorm(pp,mean=mu,sd=sd)

plot(qnorm(pp),just.some.data)
lines(qnorm(pp),fitted.normal,col=2)
```

Figure D.1. R code to demonstrate method of mometns

The code draws a random sample from a normal distribution having **mean**=3400 and $\sigma =$**sd**=5000. The code computes the Weibull plotting positions for the simulated data. Next, we compute the first two moments and fit the normal distribution by computing the quantiles of this fitted distribution to the plotting positions of interest by the **qnorm** () function. Finally, we simply plot the quantiles of the data as open circles and overlay a red line represents the fit of the data. As seen in the plot generated, the fit is pretty darn good. Well it should be, the data was normally distributed to begin with.

However, as the sample size decreases or is otherwise small and the σ of the distribution becomes large the sampling properties of the product moments are problematic. The following code is illustrative if repeatedly cut and pasted into an R session. Note that I have added the **qnorm()** on the pp term in the graphics calls to change the horizonal axis to probability paper.

```
dat <- rnorm(24,mean=3400,sd=50000)
dat <- sort(dat)
n  <- length(dat); pp <- seq(1,n)/(n+1)

mu <- mean(dat); sd <- sd(dat)

fitted.normal <- qnorm(pp,mean=mu,sd=sd)

plot(qnorm(pp),dat,
     xlab="STANDARD␣NORMAL␣DEVIATE")
lines(qnorm(pp),fitted.normal,col=2)
```

Figure D.2. R code to demonstrate method of moments using a large standard deviation

What are your thoughts after watching several re-simulations and re-plotting? You will note that the data now plots as a straight line on the normal probability paper—a good sign that we know what we are doing. However, occasionally you will see outliers.

There are many distributions to choose from. It is not always easy—in fact often hard—to decide which distribution should be used in a particular circumstance. The Selected References section of this handout provides several authoritative resources regarding application of the method of moments.

Let us work another example. The following code loads in the annual streamflow values for a streamflow-gaging station in the Austin area and uses the normal distribution as a model of the distribution of the data. Is the fit any good? Is the sample size large enough to make statements about alternative distributions?

```
S <- read.table("loc1.txt",header=TRUE)
print(S);
Q <- sort(S$mean_va)
n <- length(Q); pp <- seq(1,n)/(n+1)
mu <- mean(Q); sd <- sd(Q)
plot(pp,Q)
lines(pp,qnorm(pp,mean=mu,sd=sd),col=2)

# Now let use replot the data on probability paper
# Do you see deviations better in the tails?
plot(qnorm(pp),Q)s
lines(qnorm(pp),qnorm(pp,mean=mu,sd=sd),col=2)
```

Figure D.3. R code to demonstrate method of moments using a streamflow dataset

The plot using the normal distribution in the previous example has some serious left and right tail fit problems after careful inspection. Perhaps the lognormal distribution would be better. The lognormal distribution is used in situations in which the logarithms of the data are normally distributed. Let us give it a try. The code listed in figure D.4 is to follow immediately from the code listed in figure D.3.

```
mu.L <- mean(log10(Q)); sd.L <- sd(log10(Q))
lines(qnorm(pp),
      10^qnorm(pp,mean=mu.L,sd=sd.L),
      col=4)
```

Figure D.4. R code to demonstrate method of moments using a streamflow dataset

The blue line appears to not provide a better fit to the data. We will return to this exact same data set in the next lesson.

LESSON E: DISTRIBUTIONAL ANALYSIS–L-moments

E.1 Linear Moments (L-moments)

The analysis of univariate distributions is a complex subject. This is particularly the case with heavy-tailed data. The L-moments, which have similar meaning as the ordinary (product or central) moments, have revolutionized many fields including statistical hydrology in which I participate. L-moments have many properties that make them extremely attractive. These properties include unbiasedness, efficiency, consistency, robustness, and others.

E.1.1 L-moments

The theoretical L-moments for a real-valued random variable X with a quantile function $X(F)$ are defined from the expectations of order statistics. The order statistics of X for a sample of size n are formed by the ascending order $X_{1:n} \leq X_{2:n} \leq \ldots \leq X_{n:n}$. The theoretical L-moments are

$$\lambda_r = \frac{1}{r}\sum_{k=0}^{r-1}(-1)^k \binom{r-1}{k} E[X_{r-k:r}],$$

where r is the order of the L-moment, and $E[X_{r-k:r}]$ is the expectation of the $r-k$ order statistic of a sample of size r. The expectation of an order statistic is

$$E[X_{j:r}] = \frac{r!}{(j-1)!(r-j)!}\int_0^1 X(F) \times F^{j-1}(1-F)^{r-j}\mathrm{d}F.$$

The first four theoretical L-moments are

$$\lambda_1 = \int_0^1 X(F)\mathrm{d}F,$$

$$\lambda_2 = \int_0^1 X(F) \times (2F-1)\mathrm{d}F,$$

$$\lambda_3 = \int_0^1 X(F) \times (6F^2-6F+1)\mathrm{d}F, \text{ and}$$

$$\lambda_4 = \int_0^1 X(F) \times (20F^3-30F^2+12F-1)\mathrm{d}F.$$

The L-moment ratios are the dimensionless quantities

$$\tau = \lambda_2/\lambda_1 = \text{coefficient of L-variation},$$

$$\tau_3 = \lambda_3/\lambda_2 = \text{L-skew},$$

$$\tau_4 = \lambda_4/\lambda_2 = \text{L-kurtosis},$$

and for $r \geq 5$, which are unnamed,

$$\tau_r = \lambda_r/\lambda_2.$$

The sample L-moments are computed from the sample order statistics $x_{1:n} \leq x_{2:n} \leq \cdots \leq x_{n:n}$. The sample L-moments are

$$\hat{\lambda}_r = \frac{1}{r}\sum_{i=1}^{n}\left[\frac{\sum\limits_{j=0}^{r-1}(-1)^j\begin{pmatrix} r-1 \\ j \end{pmatrix}\begin{pmatrix} i-1 \\ r-1-j \end{pmatrix}\begin{pmatrix} n-i \\ j \end{pmatrix}}{\begin{pmatrix} n \\ r \end{pmatrix}}\right] x_{i:n}.$$

Several L-moments, unlike conventional moments, are bounded (Hosking, 1990, Theorem 2). Two useful examples for L-moment ratios are

$$-1 < \tau_r < 1 \text{ for } r \geq 3 \text{ and}$$

$$\frac{1}{4}(5\tau_3^2 - 1) \leq \tau_4 < 1.$$

E.1.2 R package `lmomco`

The R package `lmomco` fully implements L-moments in the context of many probability distributions including the Exponential, Gamma, Gumbel, Normal, Generalized Extreme Value, Generalized Lambda, Generalized Logistic, Generalized Normal (log-Normal), Generalized Pareto, Pearson Type III, Kappa, Wakeby, and Weibull. The `lmomco` package provides core functions and numerous ancillary functions to help get the user started and to keep the user entertained by building complex analysis applications.

E.2 The Method of L-moments

Assuming that the `lmomco` package has been downloaded an installed on your machine, we can begin some statistical computations using L-moments.

First, the data file **peak.csv** contains the annual peak streamflow and gage height data for a streamflow-gaging station in the Hill Country. The flood data series for this site is characteristic of those in the area: (1) Periods of considerable lowflow in which the annual peak might not represent runoff, and (2) During periods of abundant rainfall the annual peaks can be several orders of magnitude higher. An abbreviated listing of the data file is shown in the following figure.

```
peak_dt,peak_va,gage_ht,wy
1869-07,missing,42.3,1869
1900-07-16,missing,38.4,1900
7/1/32,missing,38.4,1932
1939,3820,missing,1939
10/10/39,7520,14.79,1940
4/27/41,15400,19.14,1941
4/25/42,7010,14.5,1942
10/15/42,3870,12.1,1943
.. snippet ..
```

Figure E.1. Partial listing of annual peak streamflow data

Inspection of the data shows that some values are missing and these are identified with the string **missing**. Missing values are a chronic problem in most fields of science, and this is certainly true in hydrology. Many statistical software packages are well poised to handle missing values and have specific syntax. We get a flavor of that in the next figure, which lists the code needed to read the data in to the data frame S. The **na**="missing" tells R that any field in the file with "missing" is a missing or NA value. R can handle more than one missing value identifier at a time, but that is beyond the scope of this course. After the data is read in, the customary **print**ing of the data is made, the data frame **attach**ed, and the **names** in S shown. The code completes with two separate plots of the data to inspect the time series of annual peak streamflow and gage height at this location.

```
S <- read.csv("peak.csv", header=TRUE,
               na="missing")
print(S); attach(S); names(S);

plot(wy,gage_ht,xlab="WATER␣YEAR",
                ylab="GAGE␣HEIGHT,␣IN␣FEET")

plot(wy,peak_va,xlab="WATER␣YEAR",
                ylab="STREAMFLOW,␣IN␣CFS")
```

Figure E.2. R code to load an visualize an annual peak streamflow dataset

In distributional analysis of course we are interested in the quantiles of the data. This is readily accomplished in the next code listing that builds on our previous work with plotting positions and the method of moments. Two new topics are introduced here. The subselection of nonmissing streamflow values into the **Q** variable. The **! is.na**(peak_va) instructs R to find those entries that are not (**!**) is-a-missing (**na**) values. These nonmissing values are handed off to the sort function and finally loaded into **Q**. The lmomco library is loaded to gain access to a premade function pp that by default returns the Weibull plotting positions. The plotting positions are loaded into a variable named pp. Note that pp() is the function and pp is just another variable.

```
Q <- sort(peak_va[! is.na(peak_va)])

library(lmomco) # your system will only have the this package if
    # you have downloaded and installed it. We need this package
    # for the pp() function.
pp <- pp(Q)

mu  <- mean(Q)
sd  <- sd(Q)
cat(c("Estimated␣standard␣deviation",
      sd,"\n"))

mu.L <- mean(log10(Q))
sd.L <- sd(log10(Q))
```

```
plot(qnorm(pp),log10(Q),cex=2)
lines(qnorm(pp),
      log10(qnorm(pp,mean=mu,sd=sd)))
lines(qnorm(pp),
      qnorm(pp,mean=mu.L,sd=sd.L),col=2)
```

Figure E.3. R code to load an visualize an annual peak streamflow dataset

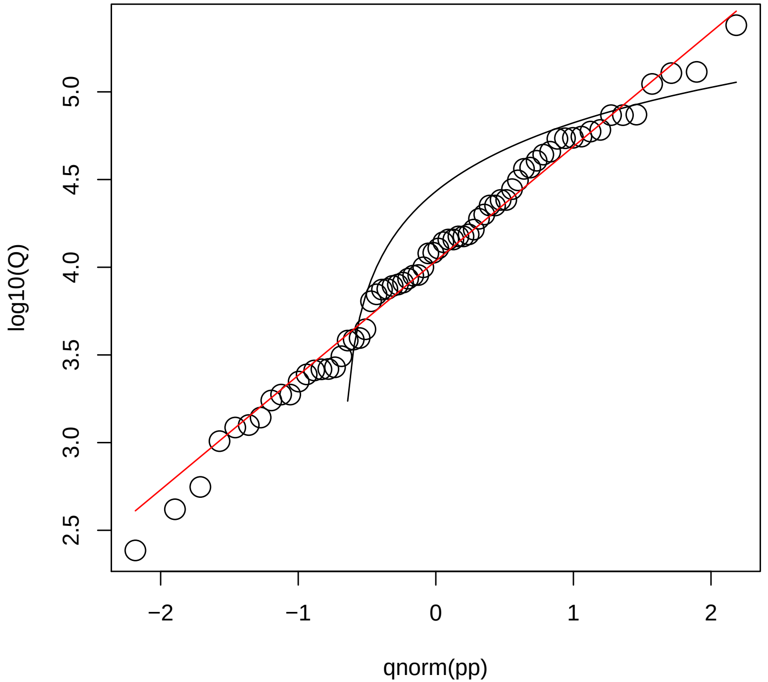

Figure E.4. Plot from R code in figure E.3 but not including graphical code in figure E.5

The code in figure E.3 continues with computation of the sample mean and standard deviation using nomenclature that we have used before. These are printed to the screen with the **cat** function. Next the sample mean and standard deviations of the **log10**() of the values are computed. The code terminates with a hopefully now familiar quantile plot. The data points are plotted by **plot**() using an open circle twice as big as the default (cex=2). Note that we again use **qnorm**() to generate probability paper and **log10**() to make the vertical axis in log space. The normal distribution fit by the method of moments is drawn as the thin black line by the first call to **lines**(). The log normal distribution is similarly drawn in red (**col**=2, color = 2). Notice immediately that the red line is straight on log-probability paper.

Careful inspection of the left and right tails suggests that the log-normal distribution (red line) over estimates the distribution. The over estimation is subtle and difficult to grasp without experience looking at such plots. However, the effects of sample size are sinister and the conclusion could simply be in error.

Lets continue and use the method of L-moment to fit two more selected distributions. Both the normal and log normal distributions used thus far are two parameter. There exists a three parameter generalized normal and four parameter kappa. For each parameter, another moment (or L-moment) is required to fit the distribution. Thus the first three L-moments are needed to fit the generalized normal and the first four L-moments are needed for the kappa. The following code listed in figure E.5 is intended to immediately follow figure E.3.

```
# library(lmomco) has already been loaded to gain access to the
    # lmoms(), pargno(), quagno(), parkap(), and quakap()
    # functions
lmr  <- lmoms(Q) # compute the L-moments of the streamflow Q
str(lmr) #

lmom.sd <- lmr$lambdas[2]*sqrt(pi)
cat(c("L-moment␣estimated␣standard␣deviation",
      lmom.sd,"\n"))

my.gno <- pargno(lmr)
lines(qnorm(pp),log10(quagno(pp,my.gno)),
      col=4,lwd=2)

my.kap <- parkap(lmr)
lines(qnorm(pp),log10(quakap(pp,my.kap)),
      col=6,lwd=4)
```

Figure E.5. R code to demonstrate computations of L-moments using annual peak streamflow dataset after the code in figure E.3

The `lmoms` returns a list of vectors containing the L-moments and L-moment ratios. The `str` function is built-in to R and provides for pretty printing of a variable—we have not seen the `str` function before. The next two lines involving the `lmom.`**sd** show a computation. Theory requires that $\sigma = \lambda_2 \times \sqrt{\pi}$. However, this estimate will not equal σ from a function such as **sd**. Hosking has reported to me that one should not estimate σ through the second L-moment (λ_2), but should instead just use λ_2.

The following pair of lines compute the parameters of the generalized normal distribution (pargno) and then draw a thicker (lwd=2) blue line (**col**=4). One can readily see how the generalized normal quantiles (quagno) generally fit the distribution better on the right tail, but do not fit as well on the left tail. This is very common and in many cases (paradoxically) we don't care about left tail fit in the context of flood magnitude. Now for lowflows this is another story; for lowflow situations, often the data is reversed by multiplying by -1 or some other reversing operation.

Finally, the four-parameter kappa distribution is shown. The parameters are estimated by the parkap function, and the quantiles are generated by the quakap function. The drawn line is very thick (lwd=4) and purple (**col**=6). Although several of the distributions drawn produced a NaNs produced in: **log**(x, base) warning message because logarithms of ≤ 0 values can not be made, a major advantage of the L-moments is that log transformation of the data prior to fitting of a distribution is not needed. This greatly simplifies analysis of data sets involving zero values or a mixture of negative and positive values.

```
S <- read.csv("peak.csv", header=TRUE,
               na="missing")
attach(S);
Q <- sort(peak_va[! is.na(peak_va)])

library(lmomco) # your system will only have the this package if
    # you have downloaded and installed it. We need this package
    # for the lmoms(), pargno(), parkap(), quagno(), and quakap()
    #  functions.
pp <- pp(Q)

lmr  <- lmoms(Q) # compute the L-moments
my.gno <- pargno(lmr) # fit the gen. normal
my.kap <- parkap(lmr) # fit the kappa

plot(qnorm(pp),Q,cex=2) # plot data
lines(qnorm(pp),quagno(pp,my.gno),
      col=4,lwd=2) # plot gen. normal
lines(qnorm(pp),quakap(pp,my.kap),
      col=6,lwd=4) # plot kappa
```

Figure E.6. R code to demonstrate computations of L-moments using annual peak streamflow dataset

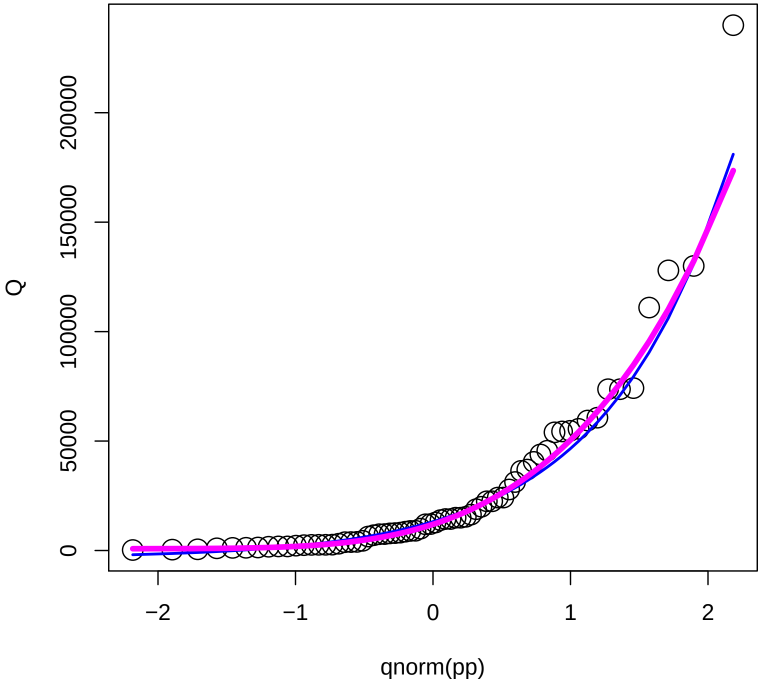

Figure E.7. Plot from R code in figure E.6

To conclude our discussion of L-moments, I want to discuss a very powerful component of the theory by which assessment of distribution fit can be made using a graphic known as an L-moment ratio (or just L-moment) diagram. This diagram plots the relation between L-skew and L-kurtosis of the data and comparison this relation to the theoretical L-skew and L-kurtosis relations for each distribution. The distributions that mimic the appearance of the data are deemed suitable. The `lmomco` package provides a couple of functions for plotting the base (background) of these plots. The R code is shown in figure E.8. The `lmrdia()` function returns the coordinates to draw the distributions, and `plotlmrdia()`, which can take numerous arguments, draws the plot. The graphic from the R code in figure E.8 is shown in figure E.9.

```
library(lmomco) # your system will only have the this package if
    # you have downloaded and installed it. We need this package
    # for the lmrdia() and plotlmrdia() functions.
lmr.diagram <- lmrdia()
plotlmrdia(lmr.diagram)
```

Figure E.8. R code to demonstrate development of an L-moment ratio diagram

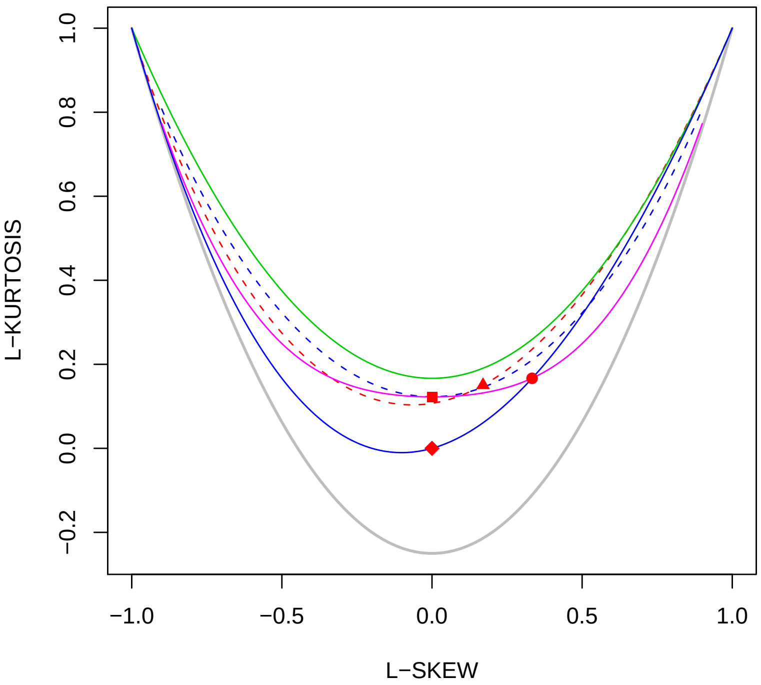

Figure E.9. Plot from R code in figure E.8

Finally, the L-moment ratio diagram in the previous figure is not particular interesting by itself. It lacks data and is not well type set. The L-moment ratio diagram in figure E.10 is very well done. This diagrams depicts the relation between the L-skew and L-kurtosis of observed distributions of storm depth and storm duration for 774 rainfall stations in Texas. I will further explain the figure in the classroom.

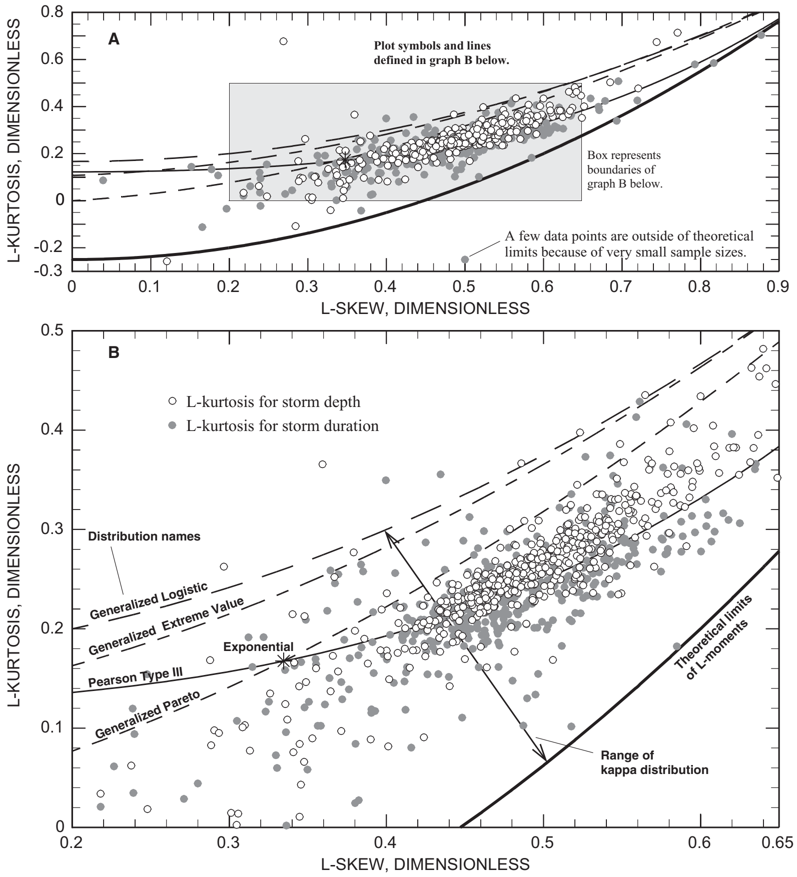

Figure E.10. Plot of an outstanding L-moment ratio diagram used to assess distribution fit

LESSON F: HYPOTHESIS TESTINGS—Change of Location

F.1 Comparing Normally Distributed Data—Student's t-Test

The t-test is used to evaluate the hypotheses about the mean of two data sets (populations), which are assumed to by normally distributed.

The so-called independent samples t-test is used to test a null hypothesis that the means of the populations sampled by the data are equal. The test statistic is a standardized difference between the means.

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{1/n_1 + 1/n_2}}$$

where $\overline{x_i}$ is the sample mean for the ith dataset and n_i is the sample size for the ith dataset. A pooled standard deviation is

$$s = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

where s_i is the sample standard deviation for the ith dataset.

By null hypothesis, the t-statistic listed above has a Student's t-distribution with $n_1 + n_2 - 2$ degrees of freedom. The $100(1 - \alpha)$ percent confidence interval for the difference of the means is constructed as

$$\overline{x_1} - \overline{x_2} \pm t_{[a,n_1+n_2-2]} s \sqrt{1/n_1 + 1/n_2}$$

where $t_{[a,n_1+n_2-2]}$ is the t-distribution with a cumulative distribution function of, $\mathcal{P}(t \leq t_{[a,n_1+n_2-2]}) = 1 - \alpha/2$.

If the two populations under study have differing standard deviations then the t-statistic is modified and becomes the Welch test:

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{s_1^2/n_1 + s_2^2/n_2}}$$

A "paired t-test" is used to compare sample means of two populations in which each individual of the first dataset has a paired value from the second dataset. Such a test might involve the simultaneous deployment of two water quality sensors, and one wants to evaluate the performance differences of each. For the paired t-test the t-statistics becomes

$$t = \frac{\overline{d}}{s/\sqrt{n}}$$

where $\overline{d}$ is the mean difference between the two datasets, s is the sample standard deviation of the differences, and n is the sample size.

F.1.1 The Welch t-Test in R

In R, the t-test is implemented by the `t.test` function. Lets work an example. The annual mean streamflow for Colorado River at Austin resides in data file `coloaustin.txt`. This is a tab-delimited file. In the following code, we read the data into the **D** data frame. Mansfield dam was completed about 1941/1942 as best I can remember. We want to know if the mean flow of the river is different today than it was prior to Lake Travis impoundment. Therefore, we define two variables containing the flow as in `Q.before]` and `Q.after`. For plotting purposes, we also define `Y.before` and `Y.after` to represent the water years. Finally, we compute the two sample means in `mu.before` and `mu.after`.

```
D <- read.table("coloaustin.txt",header=TRUE)
attach(D)
Q.before <- mean_va[year_nu <= 1941]
Q.after  <- mean_va[year_nu > 1941]
Y.before <- year_nu[year_nu <= 1941]
Y.after  <- year_nu[year_nu > 1941]
mu.before <- mean(Q.before)
mu.after  <- mean(Q.after)

plot(Y.before,Q.before,
     xlim=c( range(year_nu) ),
     ylab="ANNUAL␣STREAMFLOW,␣IN␣CFS",
     xlab="WATER␣YEAR" )
points(Y.after,Q.after,col=2,cex=2)

abline(mu.before,0)
abline(mu.after,0,col=2)
```

Figure F.1. R code to read in and plot annual mean streamflow values in preparation for a t-test.

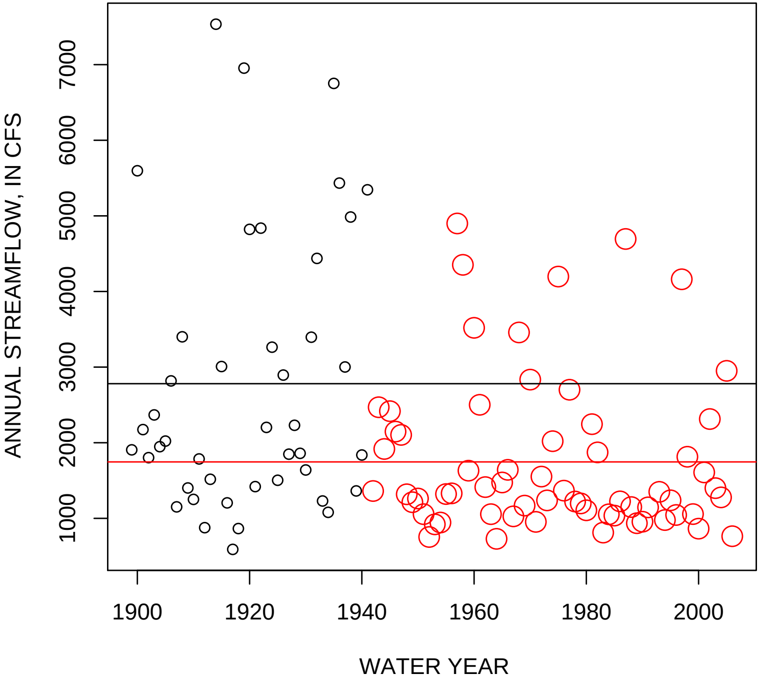

Figure F.2. Plot from R code in figure F.1

The example continues by **plot**ing each dataset in a different symbol size and color. Note the use of the `xlim=`**`c`**`(`**`range`**`(year_ne))`, which anticipates that we need the horizontal axis to encompass the entire record. We have not seen the need for resetting the default axis limits (as far as I know as I type.)

We are now prepared to perform the t-test using **`t`**`.test()` as shown in the following code, which is intended to follow the code in figure F.1. The code also launches the help page for the t-test so that we can learn precisely how the call is made and how to interpret the returned values.

```
help(t.test)
t.test(Q.before,Q.after)
```

Figure F.3. R code to perform a t-test using data from figure F.1

```
        Welch Two Sample t-test

data:  Q.before and Q.after
t = 3.4005, df = 60.229, p-value = 0.001199
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  425.8712 1642.4530
sample estimates:
mean of x mean of y
 2780.651  1746.489
```

Figure F.4. Listing of the results of the t-test from R code in figure F.3

The t-test is a fine test when it is appropriate to assume that the population from which the data results is normal. In practice this is not always appropriate. Hydrologic data in particular is often not normally distributed. Our data is bounded below by zero (negatives often are physically meaningless) and plagued by outliers on heavy right-tailed distributions. In these settings the Wilcoxon Mann-Whitney rank sum or Wilcoxon signed rank test (paired data) are more appropriate.

F.1.2 Wilcoxon Rank Sum Test in R

The Wilcoxon Mann-Whitney rank sum or Wilcoxon signed rank tests are widely used alternatives to the t-test. In either case the data is discarded—that is the original data is replaced—in favor of the ranks of the values. The test is implemented by the `wilcox.test`. The following code performs the test. An addition of the **`help.search()`** is made to this code because I could not remember the precise name of the test function (`wilcox.test`). The **`t.test`** defaults to showing the confidence interval—the `wilcox.test` does not, so the `conf.int=TRUE` is added.

```
D <- read.table("coloaustin.txt",header=TRUE)
attach(D)
Q.before <- mean_va[year_nu <= 1941]
Q.after  <- mean_va[year_nu > 1941]
help.search("rank␣sum")
help(wilcox.test)
wilcox.test(Q.before,Q.after,conf.int=TRUE)
```

Figure F.5. R code to demonstrate application of the Wilcoxon rank sum test

```
        Wilcoxon rank sum test with continuity correction

data:  Q.before and Q.after
W = 1942, p-value = 0.0003266
alternative hypothesis: true mu is not equal to 0
95 percent confidence interval:
  297.9999 1043.8000
sample estimates:
difference in location
              646.4282
```

Figure F.6. Listing of the results of the Wilcoxon test from R code in figure F.5

LESSON G: REGRESSION

Regression is a generic term encompassing well known and not so well known methods for describing and quantifying the statistical relation between two or more, usually continuous, variables. Statistical theory and operations related to the linear modeling of the relation is the most developed and readily available in numerous software packages. This lesson will consider several different "fronts" in the description of the linear relation between variables. The many references shown in the Selected References section of this document will provide a valuable exit for your own endeavors.

The practice of regression must encompass the graphical display of the relation. (FYI: USGS absolutely does not permit the term "relationship," as that is a feature exclusively of human *relationships*—thought that you would like to know.) The relations are extremely difficult to depict simultaneously if one is working in more than 3 dimensions. So careful exploratory analysis is needed. We will look at more plots later, but first I want to introduce an extremely well typeset scatter plot in figure G.1.

This scatter plot has nicely proportioned open circles for symbols. An explanation is lacking, but the caption would provide description. The line object in the plot is in fact a 1:1 line, but is precisely stated as an EQUAL VALUE LINE. You will notice that the line is thicker than the rest of the line work on the plot to draw the reader into the fact that the symbols reasonably scatter on either side of the line. Both scales are logarithmic. In fact these scales are extremely well done log-scales that popular spreads still(?) are incapable of depicting. Note that each scale has its own unique terminal points that are not coincident with an even log cycle (1, 10, 100, 1000, . . .). The ticking on the log scale is sufficient to show that the scale is in fact log, but the labeling is not so dense as to interfere with the "airiness" of the graphic. As mandatory, the units of each axis are well identified without abbreviation. Further, this particular figure was rendered with an aspect ratio that should permit ideal lifting into presentation software. The typeset page is tall, but the presentation projector is wide. Anyway, enough banter, let us move along to regression and the line of organic correlation.

G.1 Line of Organic Correlation

The line or organic correlation (*LOC*), in some geologic fields "reduced major axis," in some hydrologic fields "Maintenance of Variance–Extension," is an important method that we should know about in the

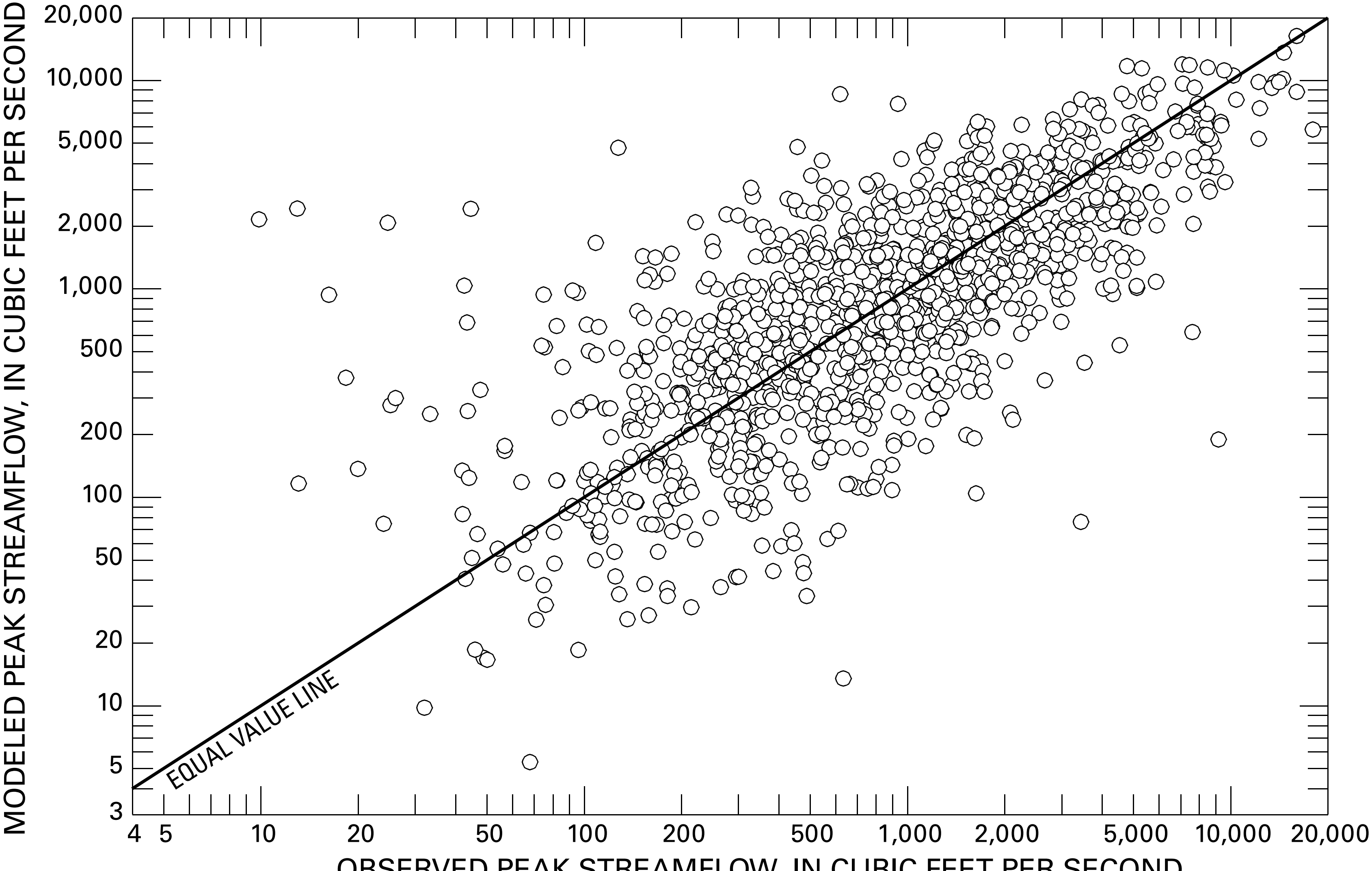

Figure G.1. Example of an excellent scatter plot

context of simple linear regression. Thus, I choose to discuss the *LOC* first.

The *LOC* possesses three characteristics (Helsel and Hirsch, 1992) preferable to simple linear regression in specific circumstances:

1. *LOC* minimized errors in both horizontal and vertical directions.
2. It provides a *unique* line whether X is regressed on Y or visa versa. This is important if one of the variables is not necessarily a physical predictor of the other (the cause and effect paradox of regression).
3. The cumulative distribution function of predictions from the regression have similar statistical properties of the actual data the predictions are generated to represent.

The *LOC* is highly suitable in situations of index prediction of streamflow or other hydrologic characteristics from one location to another based on previous history of pairwise values for the two locations. The equation for a line is

$$Y = mX + b + \epsilon$$

where Y is the regressor variable, X is the predictor variable, m is the slope, b is an intercept, and ϵ is an error term. *LOC* is designed to minimize the sum of the areas of right triangles formed by horizontal and vertical lines extending from observations to the fitted line (Helsel and Hirsch, 1992, p. 276).

LOC is computed by

$$m = \mathrm{sign}[r]\frac{s_y}{s_x}$$

where r is the correlation coefficient, s_y is the standard deviation of y, s_x is the standard deviation of x. The intercept is computed as:

$$b = \overline{Y} - \mathrm{sign}[r]\frac{s_y}{s_x}\overline{X}$$

The following R code provides a thorough demonstration on how the *LOC* is to be computed. We have two streamflow-gaging stations located in the Austin area, and the data is contained in the two data files `loc1.txt` and `loc2.txt`. We read the data in as usual. However, this time a couple of **`print`**`()`s are needed to show that the period of record is different between the two sites. This is a very common occurrence in hydrology—yuk! We seek to estimate the flow at the first station using the data from the second. The first station lacks data from 1983 through 1989; thus, it is necessary to add complexity to the code to deal with this situation. The `S2$year_nu <= 1982` and `S2$year_nu >= 1990` are used as conditional tests to subselect the data. The `|` symbol denotes an "or." Later in the code example an "and" will be used by the `&` symbol.

```
# Read two data files for an upstream (1) and downstream
    # streamflow station
S1 <- read.table("loc1.txt",header=TRUE)
S2 <- read.table("loc2.txt",header=TRUE)

print(S1); print(S2) # as usual peek at the data to confirm that
    # things were read in properly

Q1 <- S1$mean_va # extract the flow

# darn we have a gap in record at station 1
# so build a subset
Q2 <- S2$mean_va[S2$year_nu <= 1982 |
                 S2$year_nu >= 1990]
# R possess a high level function for the above, but I am at a
    # loss as what it is---easier to role one's own as I rush to
    # get this handout ready.

# this plot call would have failed unless Q2 built as in above
sd1 <- sd(Q1); sd2 <- sd(Q2)
rho.sign <- sign(cor(Q2,Q1))

m <- rho.sign*sd2/sd1 # slope
b <- mean(Q2) - m*mean(Q1) # intercept
# begin a few computations to get estimated streamflow for Q1
Q2.for.Q1.estimation <-
     S2$mean_va[S2$year_nu > 1982 &
                S2$year_nu < 1990]
```

```
# note the switch around of the math to accommodate the nature of
    # our variable definitions
Q1.est <- (Q2.for.Q1.estimation - b)/m


# FIRST PLOT IN EXAMPLE
plot(Q1,Q2) # take a peak
abline(b,m,col=2) # a built-in function to plot a line, which
    # happens to be colored red
points(Q1.est, Q2.for.Q1.estimation,
       pch=16, col=4, cex=2)

# SECOND PLOT IN EXAMPLE
plot(Q1,Q2,xlim=c(0,30),ylim=c(0,30))
abline(b,m,col=2)
 points(Q1.est, Q2.for.Q1.estimation,
       pch=16, col=4, cex=2)
```

Figure G.2. R code demonstrating computation of the line of organic correlation

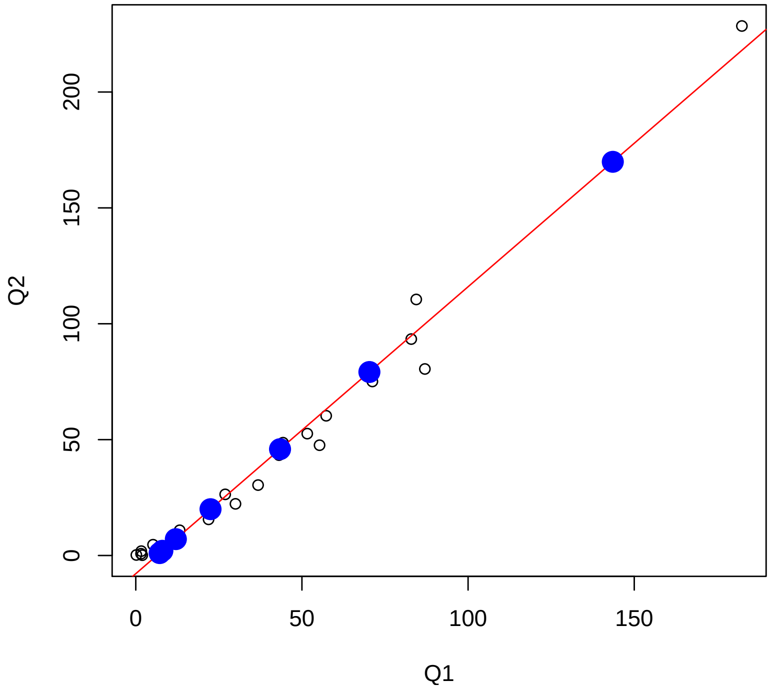

Figure G.3. Plot of line of organic correlation for first plot in example R code in figure G.2

The code in figure G.2 uses the **plot**() function to visualize the data, and the **abline**() function is used to draw the red **col**=2 line to visualize the organic relation between the annual flow between station 2 and station 1. The **sign**() function is used to compute the sign (−1 or 1) of the correlation coefficient between Q2 and Q1.

The example ends by extracting the flow values for years present at station 2 but not station 1 (Q2.**for**.Q1.estimation). The *LOC* for Q1.est is thus solved. Finally these the estimated data values are estimated by the **points**() function.

Is the plot from the previous code informative? Does station 1 have to operate? In other words is the statistical relation of annual streamflow so large or reliable that data from station 1 is redundant relative to station 1? These are serious questions with no straightforward answers that affect real taxpayer dollars. Perhaps in the annual flow context the station is not needed, but in other contexts such as lowflow or stormflow that station is? Perhaps there is critical curvature in the far left of the line. How well does *LOC* work? Lets take a look in figure G.4 that is to follow code in figure G.2 .

```
plot(Q1,Q2,xlim=c(0,10),ylim=c(0,10))
abline(b,m,col=2)
points(Q1.est, Q2.for.Q1.estimation,
         pch=16, col=4, cex=2)
```

Figure G.4. R code demonstrating computation of the line of organic correlation

After viewing the second plot with greater resolution at small flows, do you have a different impression? (Recall that I have purposefully hidden the names/locations of these stations.) These two stations from a pragmatic or large picture perspective demonstrate a flavor of the myriad of decisions involved in the design of a data collection program.

G.2 Multiple-Linear Regression

As mentioned earlier, regression is the statistical process of mathematically quantifying the relation between a regressor variable and one or more predictor variables. Regression is immensely popular and considerable research continues on the methods associated or otherwise related to it. I like regression because my customers typically require parameter—that is easy to formulate—models by which they can make predictions from the equations at locations for which no hydrologic data has ever been recorded.

Prior to multiple-linear regresssion, visualization of the data and the various relations between the data is extremely informative. So let us do that. The following example reads our dataset in to `F` from the file **`tx664.csv`**. This file for deep legacy reasons already has a $\log_{10}$ tranformation made no most of the columns. This is a common delimited file that I used to develop well-known regression equations to estimate flood frequency for ungaged and natural (rural) locations in Texas. Although the specific indent and implementation of that research is not the subject of our example per se.

The example continues by **`attach`**ing the data frame into the current work space. A first use in the course is the **`layout()`** function which permits more than one graphic on the screen device at a time. The ensemble of four **`plot()`** calls produce the observed relation between the 100-year peak streamflow and readily acquired basin characteristics.

```
F <- read.table("tx664.csv",sep=",",
                header=TRUE)
attach(F)
# Q100 = 100-year streamflow
# CDA = drainage area
# Slope = main channel slope
# Shape = CDA / squared channel length
# MAP = mean annual precipitation

layout(matrix(1:4,nrow=2))
plot(Q100~CDA); plot(Q100~Slope)
plot(Q100~Shape); plot(Q100~MAP)

pdf("fldfrqA.pdf")
  layout(matrix(1:4,nrow=2))
  plot(Q100~CDA); plot(Q100~Slope)
  plot(Q100~Shape); plot(Q100~MAP)
dev.off()
```

Figure G.5. R code to visualize some annual peak streamflow data

The example R code in figure G.5 terminates by recalling the four plots, but with the wrapping of the `pdf("fldfrqA.pdf")` and **`dev.off`** `()` functions, which product a PDF file named `fldfrqA.pdf`, which is shown in figure G.6.

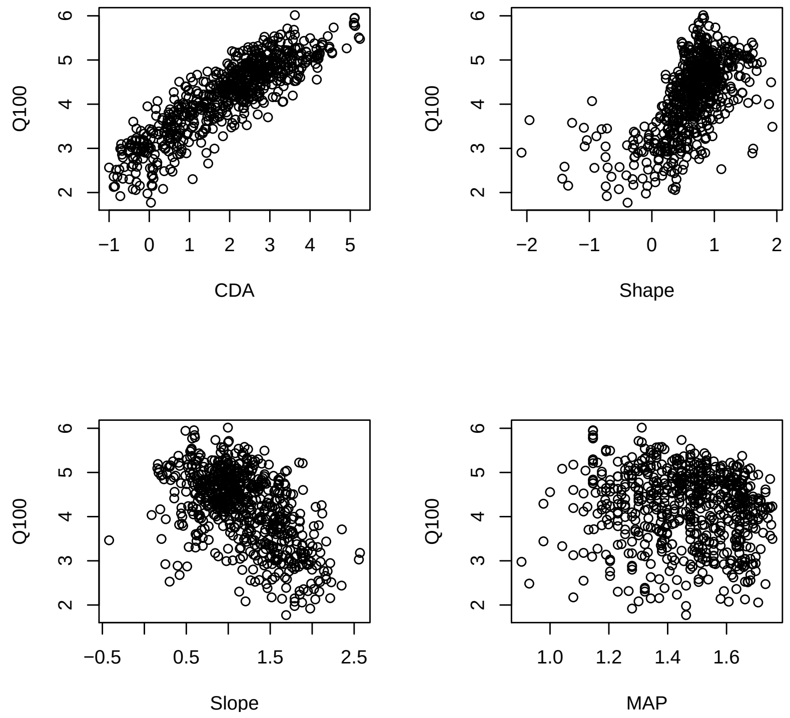

Figure G.6. Plot of four plot ensemble from R code in figure G.5

G.2.1 Ordinary Least Squares Regression

Ok, now that we have visualized the data, let us start by fitting an ordinary least squares (ols) regression. These examples are multiple-linear, but it should be self evident how a single Y and X model would be built. However, let us fully introduce the modeling syntax of R.

The R syntax for model building uses the ~ to separate the regressor variable from the predictors. The **lm**() or linear model is the function that calls the linear modeling subsystems in R. To regression Y on X one simply says **lm**(Y~X). Additional variables are added: **lm**(Y~X1+X1). One can add some mathematical transformations: **lm**(**log10**(Y)~X1+X2*X3), which says to regress the base-10 log of Y on X1 and the product of X2 and X3. If you have a darn good reason to eliminate the intercept term, you can say **lm**(Y~X1-1), where the -1 removes the intercept. The **lm**() has numerous additional arguments. We will see only the inclusion of two more in a later example.

```
F <- read.table("tx664.csv",sep=",",
                header=TRUE)
attach(F)
D <- data.frame(Q100=Q100,
                Area=CDA,Slope=Slope,
                Shape=Shape,MAP=MAP)
detach(F)
ols.modelB <- lm(Q100~.,data=D)
summary(ols.modelB) # conclude that shape is useless!
```

```
# The next two are our formal models for the topic
ols.modelA <- lm(Q100~Area+Slope+MAP,data=D)
ols.modelT <- lm(Q100~Area+MAP,data=D)

# Run the next two lines together and then run the following two
    # lines separately
layout(matrix(1:4,nrow=2))
plot(ols.modelA)

layout(matrix(1:4,nrow=2))
plot(ols.modelT)
```

Figure G.7. R code to compute three distrinct multiple-linear regression models from some annual peak streamflow data

From the ols regression seen in figure G.7, we see the re-expression of the `F` data frame into a smaller data frame with minor relabeling of the columns. This is done so that the **`lm`**`(Q100~.,`**`data`**`=`**`D`**`)` can be used to use the entire (`.`) data frame as the predictor variables—nice tidy notation. Anyway, the regression model returned from **`lm`**`()` is another data object to R, and we've loaded it into `ols.modelB`. The **`summary`** `(ols.modelB)` produces pretty output. (We will actually capture this information later.)

The **`summary`**`(ols.modelB)` output shows virtually all critical components of a regression and for many reports, it could be argued that this information is sufficient. (I have needs for other details, which we will cover shortly.) It is very important to stress that simply reporting the model and the so-called R^2 is considered in extremely poor taste and reflects poorly on the analyst presenting the data. Sorry for the digression.

The **`summary`**`()` output shows that `Shape` is not statistically significant in prediction of the 100-year streamflow. So we drop it and rerun the regression without it and load the results into `ols.modelA`. Extremely easy to do and the code in figure **??** would provide a fine archive as to what was done. Finally, I have included the `ols.modelT` as a regression using only drainage area and mean annual precipitation. I do this as a reminder to myself to discuss the purposes and abilities of end users. During my tenure, it has become useful to have "regional" equations for hydrologic variables to have only drainage area and other very easy to acquire variables (such as mean annual precipitation). Because drainage area is universally available for a stream location understudy such equations as contained in `ols.modelT` are very useful. Moral? Know your customer.

To finish discussion and give me a major jump off point for class discussion, the last two ensembles of code in figure G.7 show the very powerful high level and inheritance nature of some R functions. The **`plot`**`()` function when called with a linear modeling object dispatches to a function called **`plot.lm`**`()` (not shown), which in turn produces a sequence of four standard regression diagnostic plots. These plots for the **`plot`**`(ols.modelA)` are shown in figure G.8

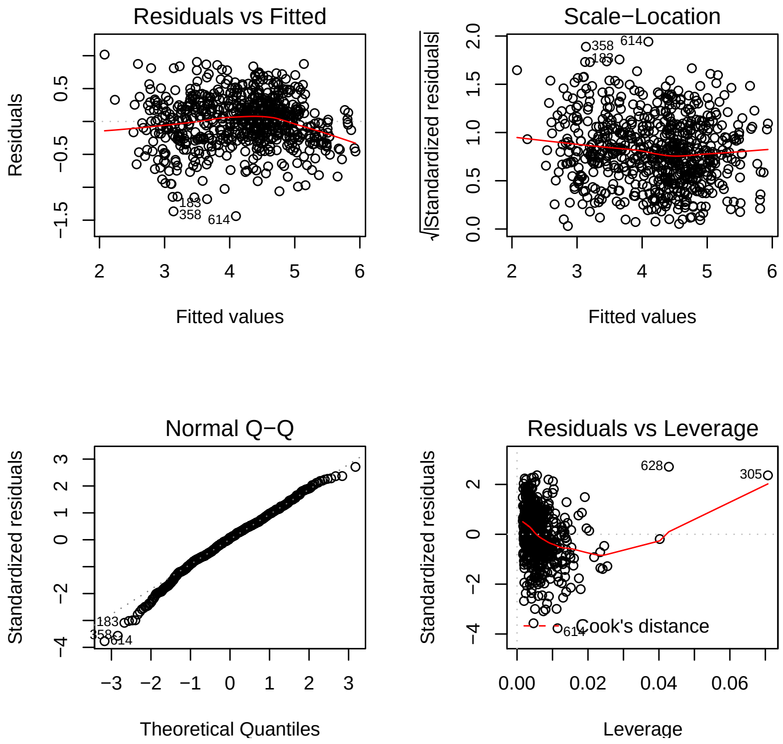

Figure G.8. Plot of four plot ensemble from R code in figure G.7

The diagnostics plots consist of a standard residual plot in which the residuals are plotted against the fitted values from the model. The red line is a smooth line. In this case some curvature in the residuals is evident, which is not a good thing. We will talk more about the residuals in the classroom. The next plot is the normal Q-Q plot in which normally distributed residuals would plot as a straight line. In this particular can the relation is not too bad. The standardized residual plot also is shown. Standardized residuals have a mean of zero and approximately unit variance. As a result, a large standardize residual, greater than about 3, potentially indicates an outlier. Finally, the residuals are plotted against leverage. Leverage is a measure of how far a particular combination of predictor variables are from the center of the cloud or parameter space. For the example, observations 305, 614, and 628 are shown as outliers.

High leverage points exert large influence on the regression. High leverage points might not be outliers. Outliers are points in which the model does not fit well. Outliers might not be high leverage points. Influence in a regression is measured by Cook's D. Cook's D is a deletion diagnostic—it measures the influence of the *i*th observation if it is removed from the sample. Cook's D is computed as

$$D_i = \frac{r_i^2}{p}\frac{h_{ii}}{1-h_{ii}}$$

where D_i is Cook's D, t_i^2 is the squared studentized residual, and h_{ii} is leverage for the data point. The r_i is

$$r_i = \frac{e_i}{\sigma(1-h_{ii}}$$

where e_i is the usual regression residual.

Cook's D, leverage, and other statistics are readily extracted in R using the **influence.measures()**. Leverage can be extracted by **hat**.values(). The **influence.measures()** function is demonstrated in figure G.9.

```
F <- read.table("tx664.csv",sep=",",
                header=TRUE)
attach(F)
D <- data.frame(Q100=Q100,
                Area=CDA,Slope=Slope,
                Shape=Shape,MAP=MAP)
detach(F)
ols.modelA <- lm(Q100~Area+Slope+MAP,data=D)
influence.measures(ols.modelA)
```

Figure G.9. R code to compute show table of regression diagnostics

Other common diagnostic statistics are reported by the **influence.measures()** function. These include DFBETAs and DFFITs. The DFBETAs is a measure by coefficient to the change in that coefficient based on the deletion of the ith value. The DFFITs statistic is the change in the fitted value for the ith observation if it is deleted. It has been suggested that if $|DFBETAs_{[j,i]}| > 2/\sqrt{n}$ then the ith observation needs examination. It has been suggested that if $|DFFITS_i| > 2\sqrt{p/n}$.

G.2.1 Evaulation of Multicollinearity of Variables

Often we pack variables into equations with little consideration that related variables contain redundant information. This can be assessed through variance inflation factors (VIF) on the regressor variables. Internal correlation between the regressor variables results in inflation of the variances of regression coefficients. This "multicollinearity" can seriously affect the precisions of regression coefficient estimation. The VIF for the jth regressor variable is

$$VIF_j = \frac{1}{1 - R_j^2}$$

where R_j^2 is the coefficient of determination obtained from regressing x_j on the other regression variables. If x_j is linearly dependent on the other variables, then x_j brings no information to the regression and R_j^2 will be near unity. It is suggest in literature that $VIF > 10$ implies a serious problem. My experience in our hydrologic problems related to streamflow regionalization is that $VIF > 5$ is worrisome. I demonstrate VIF computation using the vif() function from the DAAG package in the code listed in figure G.10. This code adds another plotting function to our toolbox; the **pairs()** function is used to produce a scatter plot matrix shown in figure G.11.

```
library(DAAG)# your system will only have the this package if you
    # have downloaded and installed it. We need this package for
    # the vif() function.
```

```
F <- read.table("tx664.csv",sep=",",
                header=TRUE)
attach(F)
D <- data.frame(Q100=Q100,
                Area=CDA,Slope=Slope,
                Shape=Shape,MAP=MAP)
detach(F)

pairs(D) # this is a powerful function that produces a complete
    # matrix of the data and produces a scatter plot for each

ols.modelC <- lm(Q100~Area+Slope+Shape+MAP,data=D) # fit a linear
    # model as before
vif(ols.modelC) # compute the variance inflation factors for the
    # regressor variables in the model. The output is shown below.

# The following is the output from vif()
#   Area  Slope  Shape    MAP
# 3.7337 3.0666 1.8810 1.7310
```

Figure G.10. R code to compute three distrinct multiple-linear regression models from some annual peak streamflow data

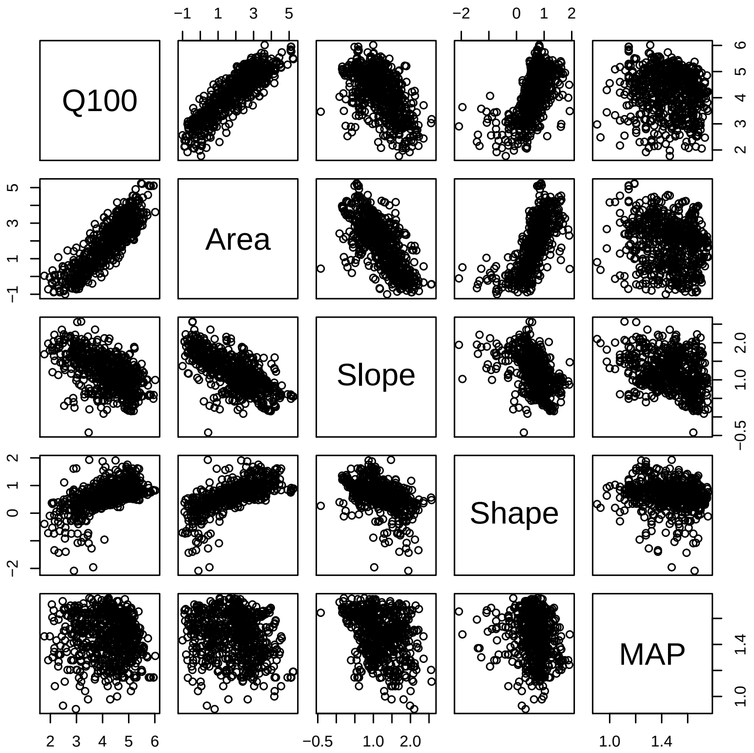

Figure G.11. Plot of `pairs()` function from example R code in figure G.10

G.2.1 Three Dimensional Scatter Plots

Because we have been discussing the interrelations between two and more variables. I think this is a good point in the course to provide an example of a three dimensional scatterplot. The R code is shown in figure G.12 and an example plot from this code is shown in figure G.13. The scatterplot3d() function obtained from the scatterplot3d package is used. I have provided a few examples of calling style to help give you a flavor for the capabilities of the function.

```
library(DAAG) # your system will only have the this package if you
    # have downloaded and installed it. We need this package for
    # the pause() function.
library(scatterplot3d) # your system will only have the this
    # package if you have downloaded and installed it. We need
    # this package for the scatterplot3d() function.
F <- read.table("tx664.csv",sep=",",
                header=TRUE)
attach(F)
D <- data.frame(Q100=Q100,
                Area=CDA,Slope=Slope,
                Shape=Shape,MAP=MAP)
detach(F)
scatterplot3d(Area,Slope,Shape)
scatterplot3d(Area,Slope,Shape,
              highlight.3d=TRUE) # used for figure in course
                  # handout

scatterplot3d(Area,Slope,Shape,
              highlight.3d=TRUE,type="h",
              angle=90)

# Loop the angles and require user to hit enter to see next plot
for(angle in seq(-180,180,by=5)) {
  scatterplot3d(Area,Slope,Shape,
                highlight.3d=TRUE,type="h",
                angle=angle)
  pause()
}
```

Figure G.12. R code to compute three-dimensionless scatter plot of watershed characteristics

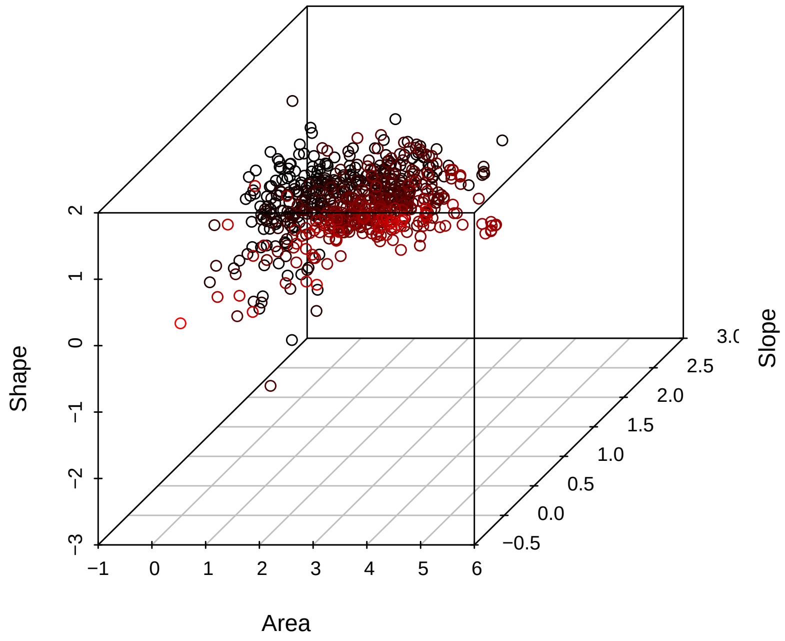

Figure G.13. Plot of the first `scatterplot3d()` function from example R code in figure G.12

G.2.2 Weighted Least Squares Regression

Weighted least squares is a regression variant in which weights are assigned that are inversely proportional to the uncertainty or variance of the data for a given data point. For the file `tx664.csv` there is a column that represents something called the equivalent years of record, which is tightly related to the number of years of record for a given streamflow-gaging station.

Weighted least squares can be a very tricky topic and software documentation is often lacking sufficient description. Usually the application of the weights is easy and the regression coefficients are invariant to transformations of the weights. That is the good new.

Now the bad. One can get very different standard errors from regression operations. You must grant me a highly technical discussion on this topic and in the context of the next few examples. I have used at least five major statistical packages (three packages on old mainframes or Unix) including R and remained unclear as the to specific properties of the weights. Through brute force and intuition I have determined that R requires that the sum of the weight factors equal the number of data points. This is accomplished through the `MLRweights()` function defined in the following example.

Weighted least squares is demonstrated in figure G.14. The code mimics that seen in figure G.7. You will note the inclusion of the `MLRweights()` function and the `Wgts` and `CorrectWgts` in the **`D`** data frame. The example fits ordinary least squares model `ols.modelA` as before. Then fits two weighted least squares models: `wls.modelA` and `wls.modelB`. Note that we can simply make a regression weighted by the inclusion of the **`weights`**= option to **`lm`**`()`.

```
F <- read.table("tx664.csv",sep=",",
                header=TRUE)
attach(F)

MLRweights <- function(avector) {
  tmp = length(avector)/sum(avector)
  return (tmp*avector)
}

D <- data.frame(Q100=Q100,
                Area=CDA,Slope=Slope,
                Shape=Shape,MAP=MAP,
                Wgts=EqYrs,
                CorrectWgts=MLRweights(EqYrs))

ols.modelA <- lm(Q100~Area+Slope+MAP,data=D)
wls.modelA <- lm(Q100~Area+Slope+MAP,
                 weights=Wgts,data=D)
wls.modelB <- lm(Q100~Area+Slope+MAP,
                 weights=CorrectWgts,data=D)

summary(ols.modelA)
summary(wls.modelA)
summary(wls.modelB)

O <- summary(ols.modelA)
A <- summary(wls.modelA)
B <- summary(wls.modelB)

z <- list(ols.sigma=O$sigma,
          bad.wls.sigma=A$sigma,
          good.wls.sigma=B$sigma)
str(z)
```

Figure G.14. R code to compute three distrinct multiple-linear regression models from some annual peak streamflow data

I show the triad of regressions by the **summary**() ensemble. Inspection and discussion of these results is left for the class room. My focus here is on the residual standard error (sigma). The second **summary**() ensemble loads the summary of each regression into O, A, and B. The sigma is readily extracted from each with syntax such as ols.sigma=O $sigma. The sigma for each model is placed into the **list**() z. Finally, the str(z) reports each value in a nice structured fashion.

What is the point? Close inspection of the ols.sigma and good.wls .sigma will show that the errors are of the same order with good.wls .sigma < ols.sigma as one would anticipate if the weights represent information content. (At least that is my interpretation—I would become alarmed/surprized if the relation was opposite.) However, note that the bad.wls.sigma is much much larger than the other two. Game playing will show that we can systematically change the sigma of the regression by simply multiplying the EqYrs by arbitrary constants—this is not a good thing. I will demonstrate this in class and discuss my historical context with this topic.

I make no apology for this discussion as you could very well encounter similar in your own field of study regardless of whether you ever use R or not.

G.3 Presentation of Regression in a Hydrologic Report

I want to nearly conclude our discussion of regression by returning briefly to my thoughts about documenting the regression process before finishing with regression trees. I have written the following function `PostRegressComps()`, which accepts an object by **`lm()`** and the vector of weights. The `PostRegressComps()` outputs the standard summary—no big deal here. The function continues on with the computation and then printing of the inverted-covariance matrix and the maximum leverage. We will talk about each of these during the course. The primal purpose of this discussion is related to the presentation of regression results by which end users can compute the prediction intervals for arbitrary predictions from the regression model.

```
PostRegressComps <- function(themodel,theweights) {
   print(summary(themodel))
   W  <- diag(theweights) # convert weight vector to diagonal of
      # matrix
   X  <- model.matrix(themodel) # extract the model matrix
   Xt <- t(X) # transposition of matrix X
   invcov    <- chol2inv(chol(Xt %*% X)) # Choleski Decomposition
      # and then inverted.
   invcov.W  <- chol2inv(chol(Xt %*% W %*% X))


   # This matrix is critical to document for end users to be able
      # to construct prediction intervals.
   cat(c("Inverted␣Covariance␣Matrix␣([XWXt]-1)␣=\n"))
   print(invcov.W)

   # This matrix is critical to document for end users to be able
      # to construct prediction intervals.
   cat(c("Inverted␣Covariance␣Matrix␣([XXt]-1)␣=\n"))
   print(invcov)



   # Maximum leverages
   hat.nowgt <- diag(X %*% invcov %*% Xt)

   hat.nowgt.W <- diag(X %*% invcov.W %*% Xt)

   hat.max.nowgt   <- max(hat.nowgt)
   hat.max.nowgt.W <- max(hat.nowgt.W)


   hat.max <- max(hatvalues(themodel)) # The maximum leverage with
      # for consideration of the regression weights.
   cat(c("Max␣hat␣(no␣weights␣with␣[XXt]-1)␣=",
              hat.max.nowgt,"\n"))
   cat(c("**␣Max␣hat␣(no␣weights␣with␣[XWXt]-1)␣=",
              hat.max.nowgt.W,"\n"))

   cat(c("Max␣hat␣=",hat.max,"\n"))

   sum.hat.max.notwgt   <- sum(hat.nowgt)
   sum.hat.max.notwgt.W <- sum(hat.nowgt.W)
   sum.of.hats <- sum(hatvalues(themodel))
   cat(c("Sum␣of␣hats␣(no␣weights␣with␣[XXt]-1)␣=",sum.hat.max.
       # notwgt,"\n"))
   cat(c("**␣Sum␣of␣hats␣(no␣weights␣with␣[XWXt]-1)=",sum.hat.max.
       # notwgt.W,"\n"))
   cat(c("Sum␣of␣hats␣=",sum.of.hats,"\n"))
}
PostRegressComps(wls.modelB,D$CorrectWgts)
```

Figure G.15. R code showing a suggested function to produce near ideal documentation of a regression equation

LESSON H: TREND EVALUATING

Hydrologic data is dominated by a particular data type—time series data. Although I have not utilized any “true” time series statistics in this course, there are some simple topics that we are prepared to discuss. That of trend testing, which from a certain perspective can be thought of as a regression like problem.

Helsel and Hirsch (1992, p. 323) report that procedures for trend analysis are built on topics of regression and hypothesis testing. The explanatory variable of interest is usual time, but this need not be the case.

Hydrologic data sets often exhibit seasonality. I leave that topic untouched in the current course. However, Helsel and Hirsch (1992, p. 337–346) provide an excellent and thorough discussion of seasonal evaluations.

H.1 Kendall’s τ

Kendall’s τ or more formally the Mann-Kendall test is a nonparametric or rank-based test that can be used to test significance of Kendall’s τ correlation. In lay words, the test can be stated most generally as a test of whether the variable Q increases or decreases with time (a monotonic change). τ is insensitive to monotonic transformations of Q, which in part makes the test attractive. As a result no *a priori* assumptions of the form of the trend (linear or nonlinear) is required.

Within R, correlation is computed through **cor**() and associated arguments and options. Kendall’s τ is a special correlation available in **cor**(). The test itself is performed by **cor**.test(). The following example reads in some annual peak streamflow data from the Texas Hill Country in data file peak.**csv**. As mentioned before, this file has some missing values so we accommodate this fact without further discussion. As good habit for a course like this, we **print**(S) the data frame.

```
# read in some annual peak streamflow data
S <- read.csv("peak.csv", header=TRUE,
              na="missing")
print(S);
S <- S[S$wy >= 1939, ] # desire only wateryears after 1939
Q <- S$peak_va; WY <- S$wy
cor(WY, Q, method="kendall") # compute tau
cor.test(WY, Q, method="kendall") # this time actually perform the
    #  test and compute significance of tau
```

Figure H.1. R code to load and conduct Kendall's τ correlation on annual peak streamflow dataset

Continuing, only data from 1939 on ward is useable in the τ context, and the condition is made by `S$wy >= 1939`. I then to tidy up the code set **`Q`** and `WY` equal to the streamflow and the water year, respectively. The **`cor`**`()` function is demonstrated by computation of τ between `WY` and **`Q`**. Finally, the Mann-Kendall test is applied with **`cor`**`.test()`. From the results would you conclude that there is a trend in annual peak streamflow for this location?

```
        Kendall's␣rank␣correlation␣tau

data:␣␣WY␣and␣Q
z␣=␣0.9211,␣p-value␣=␣0.357
alternative␣hypothesis:␣true␣tau␣is␣not␣equal␣to␣0
sample␣estimates:
␣␣␣␣␣␣␣tau
0.07648358
```

Figure H.2. Listing of the results of the Mann-Kendall from R code in figure H.1

H.2 LOWESS and LOESS Trend lines

Sometimes trend evaluation is all that we are interested in for a particular time series; we do not need/want an actual test to be performed because the test might imply understanding of the data that in reality is lacking. The smoothing technique LOWESS (LOcally WEighted Scatterplot Smooth) can be use to describe the relation between Y and X *without* assuming that the relation is linear or that the residuals are normally distributed (Helsel and Hirsch, 1992, p. 334).

R provides two functions for LOWESS generation. First, there is an older function titled **`lowess`**, and a second function called `loess`. They are functionally similar although each function by default produces different trend lines. I recommend consultation of the **`help`**`()` of each: **`help`**`(`**`lowess`**`)` and `loess`.

The R code listing in figure H.3 returns to the same data set used in the previous section. The period of record is plotted and shown in figure H.4. Two separate smooth lines are generated by the **`lowess`**`(`**`Q`**`)` and `loess(`**`Q`**`~WY)` functions. These lines will be different. The lines are superimposed on the plot by the two **`lines`**`()` commands. The line by **`lowess`**`()` is red, and the line by `loess()` is green.

```
S <- read.csv("peak.csv", header=TRUE,
               na="missing")
print(S);
S <- S[S$wy >= 1939, ]
Q <- S$peak_va; WY <- S$wy
plot(WY,Q,xlab="WATER␣YEAR",
          ylab="STREAMFLOW,␣IN␣CFS")
my.lowess <- lowess(Q)
my.loess  <- loess(Q~WY) # newer formula version of lowess, but
    # has different defaults
lines(WY,my.lowess$y,lwd=2,col=2)
lines(WY,fitted.values(my.loess),lwd=2,col=3)
```

Figure H.3. R code to load an visualize and annual peak streamflow dataset

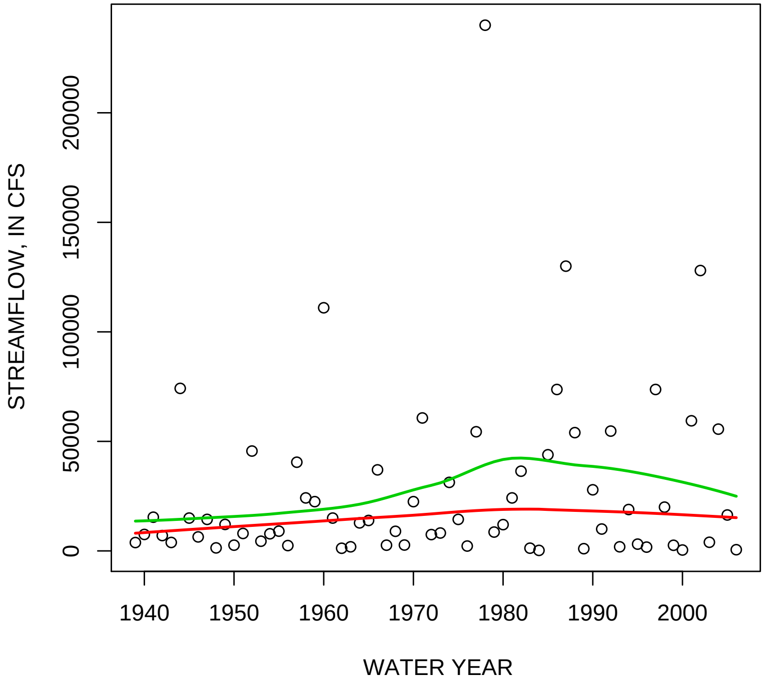

Figure H.4. Plot from R code in figure H.3

H.3 The Theil Line

The Theil line is an usual graphical tool to accompany Kendall's τ. Just as τ uses the ranks of the data, we can construct a line known as the Theil line using median statistics. The line will not depend on normality of residuals, as is very insensitive to outliers.

The slope of the Theil line is computed as the median of the $n(n-1)/2$ unique pairwise slopes computed from the data set. The intercept of the Theil line is computed as the difference between median streamflow and the slope multiplied by the median water year. The solid black dot in figure H.6 represents the **median** streamflow and **median** water year. It can be thought of as a pivot point for the Theil line.

$$\begin{aligned} S_{\mathcal{T}} &= \text{median}[\eta(i,j)] \text{ for } 1 \leq i \leq j, \\ B_{\mathcal{T}} &= \text{median}[\mathcal{FLOW}] - S_{\mathcal{T}} \times \text{median}[\mathcal{YEAR}] \end{aligned}$$

where $S_{\mathcal{T}}$ is the Theil slope, $B_{\mathcal{T}}$ is the Theil intercept, $\eta(i,j)$ is the slope between the jth and ith data point:

$$\eta(i,j) = \frac{\mathcal{FLOW}_j - \mathcal{FLOW}_i}{\mathcal{YEAR}_j - \mathcal{YEAR}_i}.$$

The R code listed in figure H.5 computes and plots the Theil line for the annual peak streamflow data used in the previous two sections. The graphical output from this code is seen in figure H.6.

```
S <- read.csv("peak.csv", header=TRUE,
               na="missing")
print(S);
S <- S[S$wy >= 1939, ]
Q <- S$peak_va; WY <- S$wy

slopes <- vector(mode="numeric")
counter <- 0
for(i in seq(1,length(Q)-1)) {
  for(j in seq(i+1,length(Q))) {
    denom <- WY[j] - WY[i]
    # we need to trap for division by zero
    if(denom != 0) {
      counter <- counter + 1
      my.slope <- ( Q[j] -  Q[i]) /
                       denom
      slopes[counter] <- my.slope
    }
  }
}
m <- median(slopes)
med.Q  <- median(Q)
med.WY <- median(WY)
b <- med.Q - med.WY*m
plot(WY,Q,xlab="WATER␣YEAR",
          ylab="STREAMFLOW,␣IN␣CFS")
abline(b,m,lwd=2)
points(med.WY,med.Q,pch=16,cex=2)
```

Figure H.5. R code to compute and plot the Theil line on an annual peak streamflow dataset

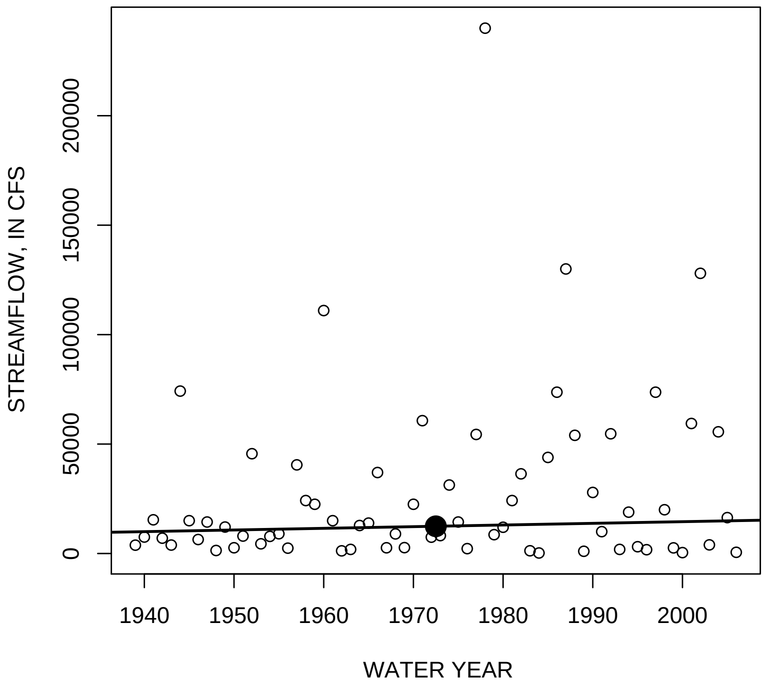

Figure H.6. Plot from R code in figure H.5

LESSON I: SOME HYDROLOGIC STATISTICS IN TEXAS

After developing the previous lessons, it is obvious that I do not have the luxury of time to describe how the lessons have influenced my research into hydrologic statistics in Texas. However, I have identified several salient references in the Selected References section.

SELECTED REFERENCES

Asquith, W.H., and Roussel, M.C., 2004, Atlas of depth-duration frequency of precipitation annual maxima for Texas: U.S. Geological Survey Water-Resources Investigations Report 2004–5041, 106 p.

Asquith, W.H., Roussel, M.C., Thompson, D.B., Cleveland, T.G., and Fang, Xing, 2004, Summary of dimensionless Texas hyetographs and distribution of storm depth developed for Texas Department of Transportation Research Project 0–4194: Texas Department of Transportation Research Report 0–4194–4, Center for Transportation Research, University of Texas at Austin, 68 p.

Asquith, W.H., and Thompson, D.B., 2005, Alternative regression equations for estimation of annual peak-streamflow frequency for undeveloped watersheds in Texas using PRESS minimization: Texas Department of Transportation Research Report 0–4405–2, Texas Tech Center for Multidisciplinary Research in Transportation, Lubbock, 27 p.

Asquith, W.H., Roussel, M.C., Thompson, D.B., Cleveland, T.G., and Fang, Xing, 2005, Unit hydrograph estimation for applicable Texas watersheds: Texas Department of Transportation Research Report 0–4193–4, Texas Tech University, Lubbock, Tex., 71 p.

Asquith, W.H., Roussel, M.C., Cleveland, T.G., Fang, Xing, Thompson, D.B., 2006, Statistical characteristics of storm interevent time, depth, and duration for eastern New Mexico, Oklahoma, and Texas: U.S. Geological Survey Professional Paper 1725, 299 p.

Dingman, S.L., 2002, Physical hydrology: 2nd ed., Prentice Hall, Upper Saddle River, NJ.

Evans M., Hastings, N.A.J., and Peacock, J.B., 2000, Statistical distributions, 3rd edition: John Wiley and Sons Inc., New York.

Faraway, J.J., 2006, Extending the linear model with R—Generalized linear, mixed effects and nonparametric regression models: Boca Raton, Fl., Chapman and Hall/CRC, 301 p.

Good, P.I., and Hardin, J.W., 2003, Common errors in statistics (and how to avoid them): Hoboken, New Jersey, John Wiley, 221 p.

Helsel, D.R., and Hirsch, R.M., 1992, Statistical methods in water resources—Studies in environmental science 49: New York, Elsevier. http://pubs.usgs.gov/twri/twri4a3/

Hosking, J.R.M., 1990, L-moments—Analysis and estimation of distributions using linear combinations of order statistics: Journal Royal Statistical Society B, v. 52, no. 1, p. 105–124.

Maindonald, J., and Braun, J., 2003, Data analysis and graphics using R—An example-based approach: Cambridge, United Kingdom, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press.

Montgomery, D.C., Peck, E.A., and Vining, G.G., 2001, Introduction of linear regression analysis: Wiley Series in Probability and Statistics, John Wiley.

R Development Core Team, 2006, R—A language and environment for statistical computing: R Foundation for Statistical Computing, Vienna, Austria, ISBN 3–900051–07–0, `http://www.R-project.org`

Therneau, T.M., and Atkinson, Beth, 2006, `rpart` package—Recursive partitioning: R port by Brian Ripley, R package version 3.1-29, S-PLUS 6.x original at `http://mayoresearch.mayo.edu/mayo/research/biostat/splusfunctions.cfm`

Williams-Sether, T., Asquith, W.H., Thompson, D.B., Cleveland, T.G., and Fang, Xing, 2004, Empirical, dimensionless, cumulative-rainfall hyetographs developed from 1959–86 storm data for selected small watersheds in Texas: U.S. Geological Survey Scientific Investigations Report 2004–5075, 125 p.