*Latifi, Zeeshan*

*R Programming test 2*

Predict 401 – Spring 2017 – Northwestern University

**Instructions**: **Identify your test with your name, (last name followed by first name).** Use this WORD document to submit your test answers. I will add my comments directly to your .docx document. Enter your code solution below the problem statement along with any required output or displays. I prefer that you copy and paste results from the console. Be careful with the format of your report. Watch the margins and pagination. Depending on the problem, grading will be based on: 1) the correct result, 2) coding efficiency and 3) graphical presentation features (labeling, colors, size, legibility, etc.). I will be looking for well-rendered displays. Do not print and display the contents of vectors or data frames unless requested by the problem. You should be able to display each solution in fewer than ten lines of code.

**Example Problem with Solution:** Use rbinom() to generate two random samples of size 10,000 from a binomial distribution. For the first sample, use p = 0.45 and n =10. For the second sample use p = 0.55 and n = 10.

1. Convert the sample frequencies to sample proportions and compute the mean number of successes for each sample. Present these statistics.

> set.seed(123)

> sample.one <- table(rbinom(10000, 10, 0.45))/10000

> sample.two <- table(rbinom(10000, 10, 0.55))/10000

> successes <- (seq(0, 10))

> sum(sample.one\*successes)

[1] 4.4827

> sum(sample.two\*successes)

[1] 5.523

1. Present the proportions in a vertical side-by-side barplot color coding the two samples.

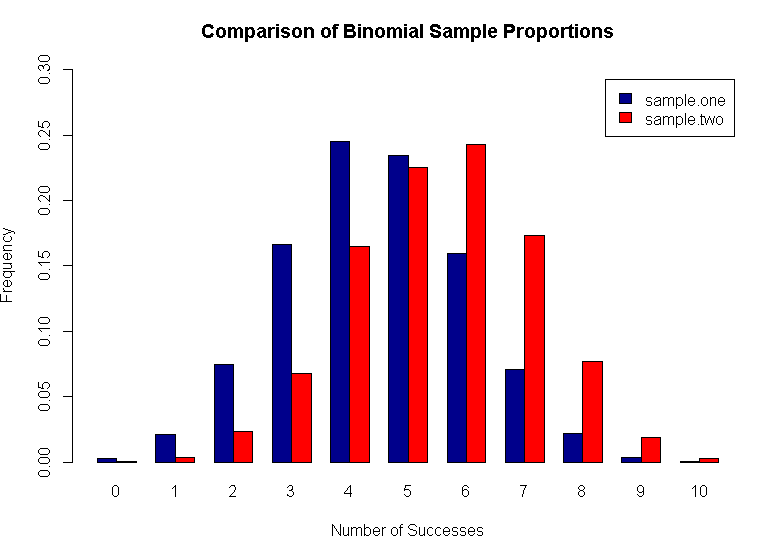
> counts <- rbind(sample.one, sample.two)

> barplot(counts, main="Comparison of Binomial Sample Proportions",

+ ylab = "Frequency", ylim = c(0,0.3),xlab="Number of Successes",

+ beside = TRUE, col=c("darkblue","red"),legend = rownames(counts),

+ names.arg = c("0","1","2","3","4","5","6","7","8","9","10"))



Test Questions (50 points total)

1. R has probability functions available for use (see Davies Chapter 16 and Kabacoff Section 5.2.3). Using one distribution to approximate another is not uncommon.

a) (3 points) Define the vector x <- c(0,1,2,3,4,5,6). Using this vector and the discrete probability functions shown below, calculate the probability of each outcome in x. Row bind the three sets of results, and generate a bar plot. The bar plot should have three color coded vertical bars for each outcome in x. Add additional features as you choose.

* + 1. dbinom(x, 100, 0.01)
    2. dpois(x, 1.0)
    3. dhyper(x, 20, 2000, 100)

> #create vector

> x <- c(0,1,2,3,4,5,6)

> x

[1] 0 1 2 3 4 5 6

> #create function for binomial, poisson, and hypergeometric distribution

> b1 <- dbinom(x, 100, 0.01)

> b2 <- dpois(x, 1.0)

> b3 <- dhyper(x, 20, 2000, 100)

>

> #combine the three distributions by row

> prob <- rbind(b1,b2,b3)

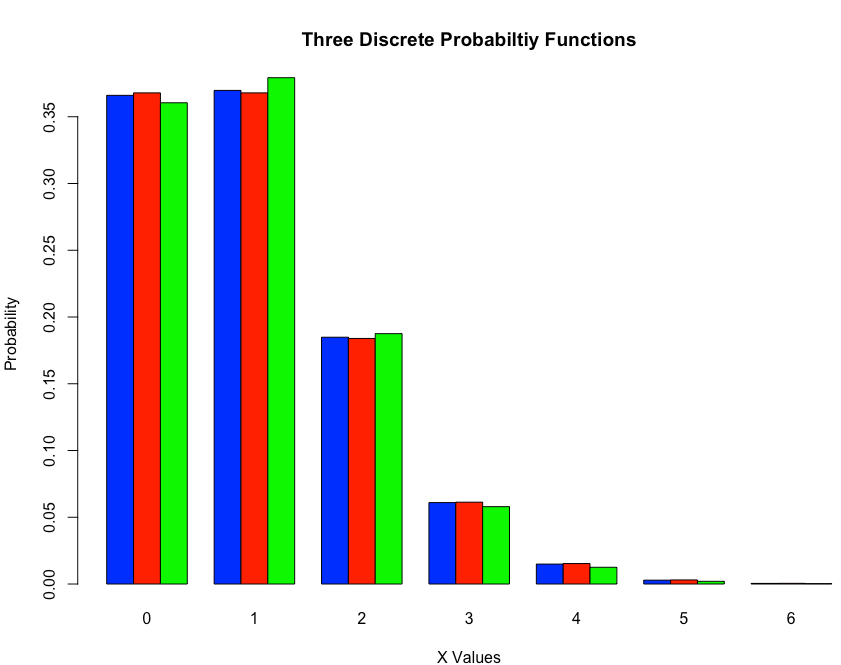
>

> #plot the combination of distributions

> barplot(prob, beside = TRUE, main = "Three Discrete Probabiltiy Functions",

+ ylab = "Probability", xlab = "X Values",

+ col = c("blue","red","green"),names.arg = c("0","1","2","3","4","5","6"))



b) (5 points) The normal distribution may be used to approximate the binomial   
distribution if np > 5 and np(1-p) > 5. Find the following binomial probabilities using   
dbinom() and pbinom() with a probability p = 0.5 and n = 100. Then estimate the same

probabilities using the normal approximation.

1. The probability of exactly 50 successes.
2. The probability of 42 or fewer successes.
3. The probability of 58 or more successes.

> n <- 100

> p <- 0.5

> n\*p #Test to see if np > 5

[1] 50

> n\*p\*(1-p) #Test to see if np(1-p) > 5

[1] 25

>

> #test probability p(50), p(42 or less), and p(58 or more)

> db\_p50 <- dbinom(50, size = n, prob = p)

> db\_p42less <- sum(dbinom(0:42, size = n, prob = p))

> db\_p58more <- sum(dbinom(58:100, size = n, prob = p))

>

> #display values

> db\_p50

[1] 0.07958924

> db\_p42less

[1] 0.06660531

> db\_p58more

[1] 0.06660531

>

> #Using pbinom for 3 probabilities calculated above using dbinom

> pb\_p50 <- pbinom(50, size = n, prob = p) - pbinom(49, size = n, prob = p)

> pb\_p42less <- pbinom(42, size = n, prob = p)

> pb\_p58more <- pbinom(100, size = n, prob = p) - pbinom(57, size = n, prob = p)

>

> #display values

> pb\_p50

[1] 0.07958924

> pb\_p42less

[1] 0.06660531

> pb\_p58more

[1] 0.06660531

>

> #calculate mean

> mu <- n\*p

> #calculate sd

> stdev <- sqrt(n\*p\*(1-p))

>

> #Using normal approximation with correction

> na\_p50 <- pnorm(50.5, mean = mu, sd = stdev) - pnorm(49.5, mean = mu, sd = stdev)

> na\_p42less <- pnorm(42.5, mean = mu, sd = stdev)

> na\_p58more <- 1-pnorm(57.5, mean = mu, sd = stdev)

>

> #display values

> na\_p50

[1] 0.07965567

> na\_p42less

[1] 0.0668072

> na\_p58more

[1] 0.0668072

c) (2 points) Use the binomial probabilities from dbinom() with n = 100 and p = 0.01 to calculate the expected value and variance for this distribution. (To do this you will need to use integer values from 0 to 100 as binomial outcomes with their corresponding probabilities.) Calculate the same using the formulas np and np(1-p). Compare.

> n <- 100

> p <- 0.01

> x <- 0:100

> q <- 0.99

>

> n\*p #calcuate expected value

[1] 1

> n\*p\*(1-p) #calculate variance

[1] 0.99

>

> #expected value using dbinom

> ev <- dbinom(x, n, p)

> sum(x\*ev)

[1] 1

>

> #variance using dbinom

> var <- dbinom(x, n, q) \* p

> sum(x\*var)

[1] 0.99

When using integer values as binomial outcomes the values come out the same as if they are done using the standard np and np(1-p) formulas.

1. A recurring problem in statistics is the identification of outliers. This problem involves plotting data to display outliers, and then classifying them.

a) (3 points) Use rexp(n=100, rate =1) to general a random sample of 100 values from an exponential distribution. Draw the random sample after using set.seed(123). Do not change this number as your sample will then differ from the answer sheet. If you must draw another sample, start the process with set.seed(123). Present these data in a side-by-side display showing a boxplot and a normal QQ chart. Add features to the plot as you choose.

> set.seed(123) #set seed

>

> #generate random exponential numbers

> randexp <- rexp(n = 100, rate = 1)

> #set window for graphs

> par(mfrow=c(1,2))

>

> #generate boxplot with random numbers

> boxplot(randexp, main = "Random Sample - Exponential Distribution",

+ ylab = "Random Values Distribution", col = "deepskyblue")

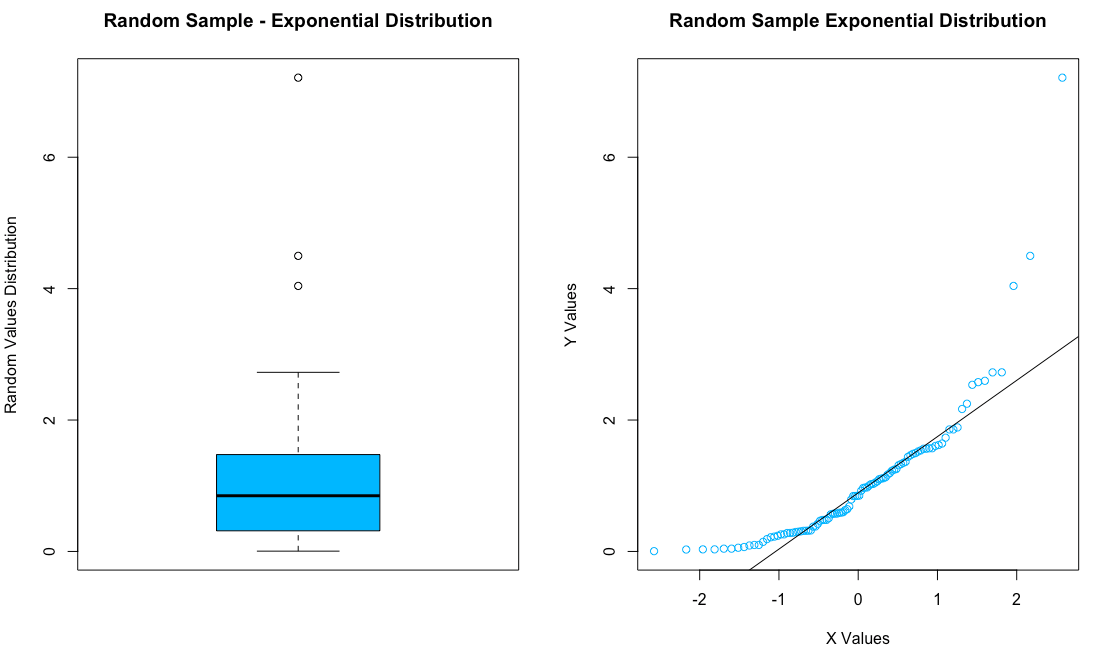
>

> #generate qq chart with random numbers

> qqnorm(randexp, main = "Random Sample Exponential Distribution", ylab = "Y Values",

+ xlab = "X Values", col = "deepskyblue")

> qqline((randexp))



b) (2 points) Identify the values which are outliers and which are extreme outliers. (Hint - Using boxplot.stats() is a convenient way to do this.)

> #boxplot values that are outliers listed as out

> boxplot.stats(randexp, coef = 1.5, do.conf = T, do.out = T)

$stats

[1] 0.004599127 0.313871774 0.847754047 1.474288177 2.726236464

$n

[1] 100

$conf

[1] 0.6644083 1.0310998

$out

[1] 4.041012 7.211008 4.498673

As shown above the outliers are 4.04, 4.49, and 7.21

c) (5 points) Repeat the steps in (a) and (b) on a transformed variable. Use the random  
 sample generated in (a), but transform it to a new variable using the following Box-Cox

Transformation: y = 3\*((x^1/3) – 1). (Note, x^1/3 is the cube root of x.) Address all the   
 requests and questions asked in (a) and (b).

> boxcox = 3\*((randexp^(1/3)-1))

> par(mfrow=c(1,2))

>

> #generate boxplot with random numbers

> boxplot(boxcox, main = "Random Sample - Exponential Distribution",

+ ylab = "Random Values Distribution", col = "salmon")

>

> #generate qq chart with random numbers along with qqline

> qqnorm(boxcox, main = "Random Sample Exponential Distribution", ylab = "Y Values",

+ xlab = "X Values", col = "salmon")

> qqline((boxcox))

>

> #test for outliers

> boxplot.stats(boxcox, coef = 1.5, do.conf = T, do.out = T)

$stats

[1] -2.5011005 -0.9612126 -0.1607023 0.4143867 1.9524041

$n

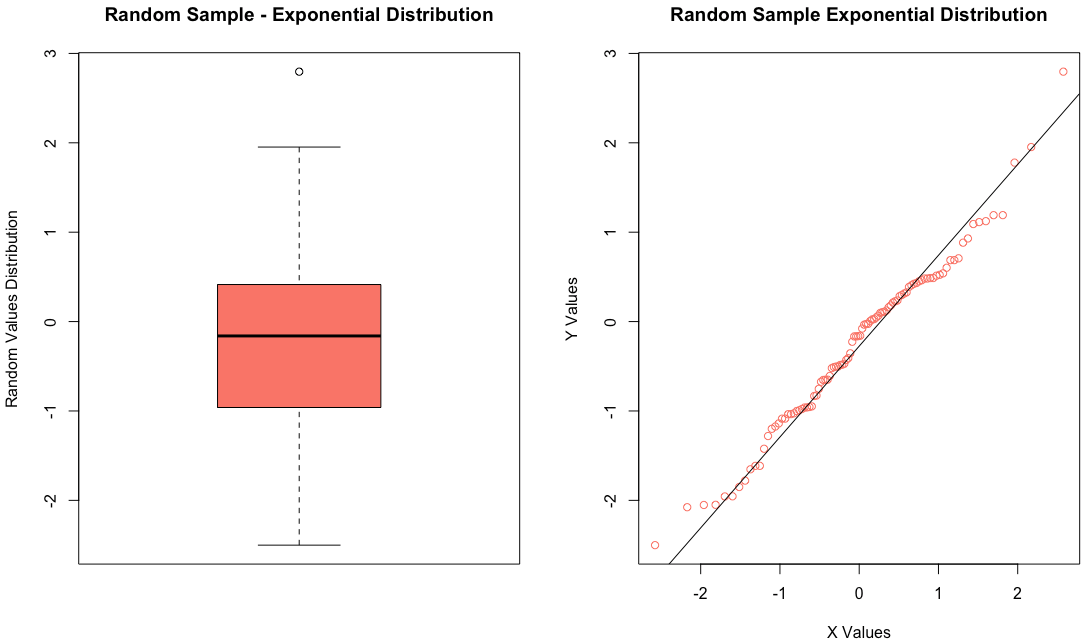
[1] 100

$conf

[1] -0.37804695 0.05664241

$out

[1] 2.795887



1. Performing hypothesis tests using random samples from two populations is fundamental to statistical inference. The first part of this problem deals with comparing two different diets based on sample means. The ChickWeight data available on R will be used. Execute the following code to prepare a data frame for analysis in this problem.

> data(ChickWeight)

> index <- (ChickWeight$Time == 21)&((ChickWeight$Diet == "1")|(ChickWeight$Diet == "3"))

> result <- subset(ChickWeight[index,], select = c(weight, Diet))

> result$Diet <- factor(result$Diet)

> str(result)

Classes ‘nfnGroupedData’, ‘nfGroupedData’, ‘groupedData’ and 'data.frame':26 obs. of 2 variables:

$ weight: num 205 215 202 157 223 157 305 98 124 175 ...

$ Diet : Factor w/ 2 levels "1","3": 1 1 1 1 1 1 1 1 1 1 ...

The file “result” will have chick weights for two diets identified as diet “1” and diet “3”. Using the file “result”, complete the following two items using R.

a) (4 points) Use the weight data for the two diets to test the null hypothesis of equal population weights for the two diets. Test at the 95% confidence level with a two-sided t-test. This can be done using t.test() in R. Assume equal variances. Report the results.

> #input chickweight data (code provided by professor)

> data(ChickWeight)

> index <- (ChickWeight$Time == 21)&((ChickWeight$Diet == "1")|(ChickWeight$Diet == "3"))

> result <- subset(ChickWeight[index,], select = c(weight, Diet))

> result$Diet <- factor(result$Diet)

> str(result)

Classes ‘nfnGroupedData’, ‘nfGroupedData’, ‘groupedData’ and 'data.frame': 26 obs. of 2 variables:

$ weight: num 205 215 202 157 223 157 305 98 124 175 ...

$ Diet : Factor w/ 2 levels "1","3": 1 1 1 1 1 1 1 1 1 1 ...

>

> #conduct t.test on result dataset, compare mean weights that are on diet 1 vs diet 3

> t.test(result$weight[result$Diet == "1"], result$weight[result$Diet == "3"], alternative = c("two.sided"), mu = 0,

+ paired = FALSE, var.equal = TRUE, conf.level = 0.95)

Two Sample t-test

data: result$weight[result$Diet == "1"] and result$weight[result$Diet == "3"]

t = -3.5955, df = 24, p-value = 0.001454

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-145.67581 -39.42419

sample estimates:

mean of x mean of y

177.75 270.30

>

> #prepare pre and post data for diet 3

> data(ChickWeight)

> index <- (ChickWeight$Diet == "3")

> pre <- subset(ChickWeight[index,], Time == 20, select = c(weight))$weight

> post <- subset(ChickWeight[index,], Time == 21, select = c(weight))$weight

As you can see the mean of x is 177.5 and the mean of y is 270.30

Working with paired data is another common statistical activity. The ChickWeight data will be used to illustrate how the weight gain from week 20 to 21 may be analyzed. Use the   
following code to prepare pre and post data from diet “3” for analysis.

> data(ChickWeight)

> index <- (ChickWeight$Diet == "3")

> pre <- subset(ChickWeight[index,], Time == 20, select = c(weight))$weight

> post <- subset(ChickWeight[index,], Time == 21, select = c(weight))$weight

b) (2 points) Plot the post data versus the pre data using a scatterplot. Compute the sample variances of pre, post and (post-pre). Compare the sample variances. What does this   
suggest regarding the advantages of performing paired t-tests?  
> #plot post data vs pre data

> par(mfrow=c(1,1))

> plot(post, pre, main = "Post versus Pre", xlab = "Post", ylab = "Pre", col = "deepskyblue")

>

> var(pre) #calculate variance of pre

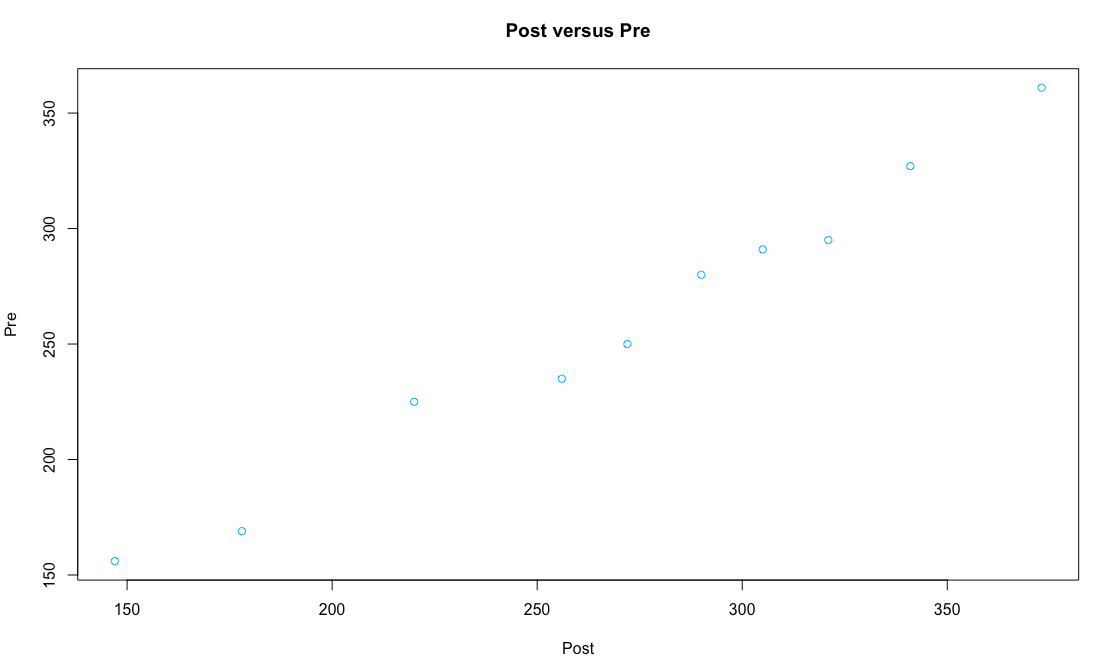
[1] 4256.767

> var(post) #calculate variance of post

[1] 5129.789

> var(post-pre) #calculate variance of the difference (post-pre)

[1] 124.9333



The advantage of a paired t-test is that it does not have additional variation that happens

when conducting a two sample t-test.

c) (4 points) Construct a 95% confidence interval for the average weight gain from week  
20 to week 21. Do not use t.test(). Write the code for the test including determination of the endpoints of the confidence interval. Present the average difference and the confidence interval. What do you conclude about weight gain?  
> avg <- mean(post-pre) #average difference

> avg

[1] 11.4

>

> standev <- sqrt(var(post-pre)) #standard deviation of the difference

> standev

[1] 11.17736

>

> #find endpoints

> qnorm(c(0.025, 0.975), avg, standev)

[1] -10.50722 33.30722

>

> zval <- round(qnorm(1 - (1 - 0.95)/2), 3) #find z score

>

> #confidence interval calculation

> meanlower <- avg-(zval\*(standev/sqrt(length(post))))

> meanupper <- avg+(zval\*(standev/sqrt(length(post))))

>

> #show confidence intervals

> confintervals <- c(meanlower,meanupper)

> confintervals

[1] 4.472202 18.327798

The confidence interval is pretty accurate for the average difference in weight gain.

However, based on the test run here, there is a significant variation in the mean weight gain

from week 20 to 21.

1. This problem deals with sampling distributions and the central limit theorem. Statistical inference depends on using a sampling distribution for a statistic in order to make confidence statements about unknown population parameters. The central limit theorem is a fundamental part of statistical inference. This problem illustrates how this comes about.

These data deal with the flow of the Nile from 1871 to 1970. Use the code below to prepare the data. This code may be helpful in crafting your solution.

> data(Nile)

> m <- mean(Nile)

> std <- sd(Nile)

> x <- seq(400, 1400,1)

> hist(Nile, freq = FALSE, col = "darkblue", xlab = "Flow",

+ main = "Histogram of Nile River Flows 1871 to 1970")

> curve(dnorm(x, mean=m, sd=std), col="orange", lwd=2, add=TRUE)

a) (3 points) Using the Nile River flow data, calculate skewness and kurtosis using the moments package. Using the Nile data, present a side-by-side display of a Quantile-Quantile plot (using qqnorm() and a boxplot (i.e. par(mfrow = c(1,2). Add features to these displays as you choose. Do these plots reveal any outliers?

> #prepare the Nile data (provided by professor)

> data(Nile)

> m <- mean(Nile)

> std <- sd(Nile)

> x <- seq(400, 1400,1)

> #histogram of the Nile

> hist(Nile, freq = FALSE, col = "darkblue", xlab = "Flow", main = "Histogram of Nile River Flows 1871 to 1970")

> curve(dnorm(x, mean=m, sd=std), col="orange", lwd=2, add=TRUE)

>

> library(moments)

> skewness(Nile) #check skewness of the Nile

[1] 0.3223697

> kurtosis(Nile) #check kurtosis of the Nile

[1] 2.695093

>

> par(mfrow = c(1,2)) #generate graph window

>

> #generate qq chart and box plot with random numbers

> qqnorm(Nile, main = "Nile QQ Plot", ylab = "Y Values",

+ xlab = "X Values", col = "tan1")

> qqline((boxcox))

>

> boxplot(Nile, main = "Nile Boxplot",

+ ylab = "River Flow", col = "tan1")

>

> #test for outliers

> boxplot.stats(Nile, coef = 1.5, do.conf = T, do.out = T)

$stats

[1] 456.0 798.0 893.5 1035.0 1370.0

$n

[1] 100

$conf

[1] 856.054 930.946

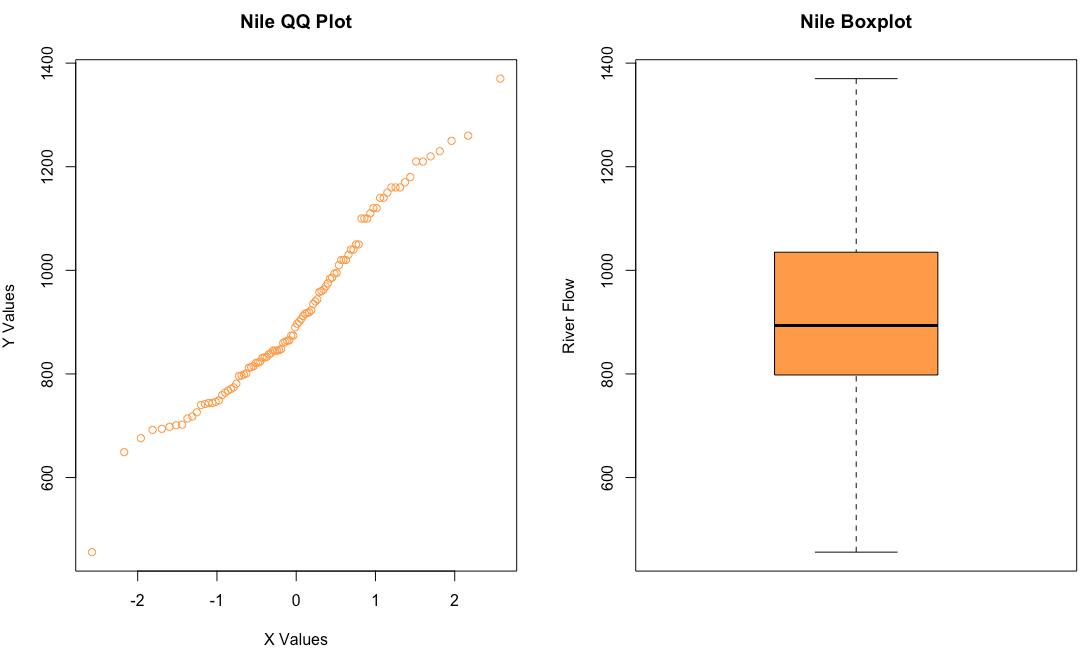
$out

numeric(0)

>

> par(mfrow = c(1,1)) #reset graph window

No outliers revealed by the plots that are shown here



b) (4 points) Using set.seed(124) and the data in Nile, generate 1000 random samples of size 25 with replacement. For each sample that is drawn, calculate and store the sample mean. This will require a “for” loop and use of the sample() function . Present a histogram of the sample means with the normal density function superimposed as shown above. Present the average of the sample means and the variance for the sample means shown in the histogram.

> set.seed(124) #set seed

>

> #generate for loop to have a 1000 sample of size 25 using Nile data

> mean.s1 <- numeric(0)

> N <- 1000

> for (k in 1:N){

+ mean.s1[k] <- mean(sample(Nile, 25, replace = TRUE))

+ }

>

> # Demonstration of convergence to a normal distribution.

> m <- mean(Nile)

> m

[1] 919.35

> std <- sd(Nile)/sqrt(25)

> std

[1] 33.8455

>

> var.s1 <- sd(mean.s1)^2 #variance of random sample

> var.s1

[1] 1095.874

> mean(mean.s1) #mean random sample

[1] 918.6558

>

> #generate histrogram with density distribution with curve

> x <- seq(700, 1100, 1)

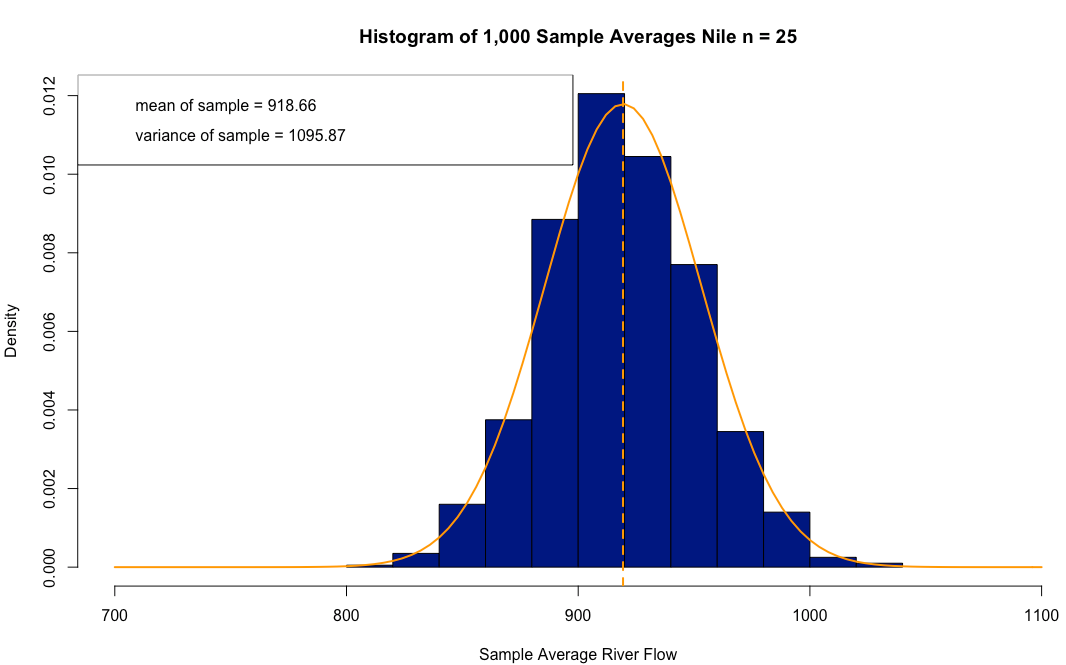
> hist(mean.s1, freq = FALSE, col = "darkblue", xlab = "Sample Average River Flow",

+ main = "Histogram of 1,000 Sample Averages Nile n = 25", xlim = c(700,1100))

> curve(dnorm(x, m, std), col = "orange", lwd = 2, add = TRUE)

> abline(v= m, col = "orange", lty = 2, lwd = 2)

> legend("topleft", legend = c("mean of sample = 918.66", "variance of sample = 1095.87"))



c) (3 points) Using set.seed(127) (a different starting point) and the Nile data, generate 1000 random samples of size 100 with replacement. For each sample, calculate and store the sample mean. Calculate the mean value and standard deviation of the 1000 generated means. Compare these summary statistics with the summary statistics calculated in b) above. Discuss the central limit theorem. Is the change in sample variances from b) to c) what would be expected?

> set.seed(127) #set seed

>

> #generate for loop to have a 1000 sample of size 100 using Nile data

> mean.s2 <- numeric(0)

> N <- 1000

> for (k in 1:N){

+ mean.s2[k] <- mean(sample(Nile, 100, replace = TRUE))

+ }

>

> #find mean and sd of the sample

> m2 <- mean(mean.s2)

> m2

[1] 918.9253

> sd2 <- sd(mean.s2)

> sd2

[1] 16.35126

>

> #summary of both samples

> summary(mean.s1)

Min. 1st Qu. Median Mean 3rd Qu. Max.

815.5 896.2 917.2 918.7 940.8 1033.0

> summary(mean.s2)

Min. 1st Qu. Median Mean 3rd Qu. Max.

859.9 907.8 919.0 918.9 930.7 972.3

Based on the change is size n, the variances also had a similar effect. Since we

changed our size from 25 to 100 (multiplied 25 by 4) we saw the same effect on the variance. If you took the variance of mean.s2 and multiplied by 4 you’d get a

number very close to the variance for mean.s.

1. This problem deals with 2x2 contingency table analysis. This is an example of categorical data analysis (see Kabacoff pages 145-151). This type of analysis is useful for determining if two categorical or nominal variables are associated. The method shown in this problem can be used to screen data for potential predictors that may be used in building a model.

The Seatbelts data set contains monthly road casualties in Great Britain 1969-84. A chi-square test of independence will be used (See Black Section 16.2) to determine if there is a trend in drivers killed over the period 1969-84. The first step is to organize the data for analysis. Use the code below to generate two variables: killed and month (starting in 1969).

> data(Seatbelts)

> Seatbelts <- as.data.frame(Seatbelts)

> Seatbelts$Month <- seq(from = 1, to = nrow(Seatbelts))

> Seatbelts <- subset(Seatbelts, select = c(DriversKilled, Month))

> summary(Seatbelts)

> killed <- factor(Seatbelts$DriversKilled > 118.5, labels = c("below", "above"))

> month <- factor(Seatbelts$Month > 96.5, labels = c("below", "above"))

a) (3 points) Using Seatbelts, generate a scatter plot of killed versus month. Show vertical and horizontal lines to indicate the median month and the median of killed. This is one way to evaluate variables pairwise for potential association. Label and add features as desired.

> #prepare the seatbelts data (provided by professor)

> data(Seatbelts)

> Seatbelts <- as.data.frame(Seatbelts)

> Seatbelts$Month <- seq(from = 1, to = nrow(Seatbelts))

> Seatbelts <- subset(Seatbelts, select = c(DriversKilled, Month))

> summary(Seatbelts)

DriversKilled Month

Min. : 60.0 Min. : 1.00

1st Qu.:104.8 1st Qu.: 48.75

Median :118.5 Median : 96.50

Mean :122.8 Mean : 96.50

3rd Qu.:138.0 3rd Qu.:144.25

Max. :198.0 Max. :192.00

>

> #generate two variables- killed & month

> killed <- factor(Seatbelts$DriversKilled > 118.5, labels = c("below", "above"))

> month <- factor(Seatbelts$Month > 96.5, labels = c("below", "above"))

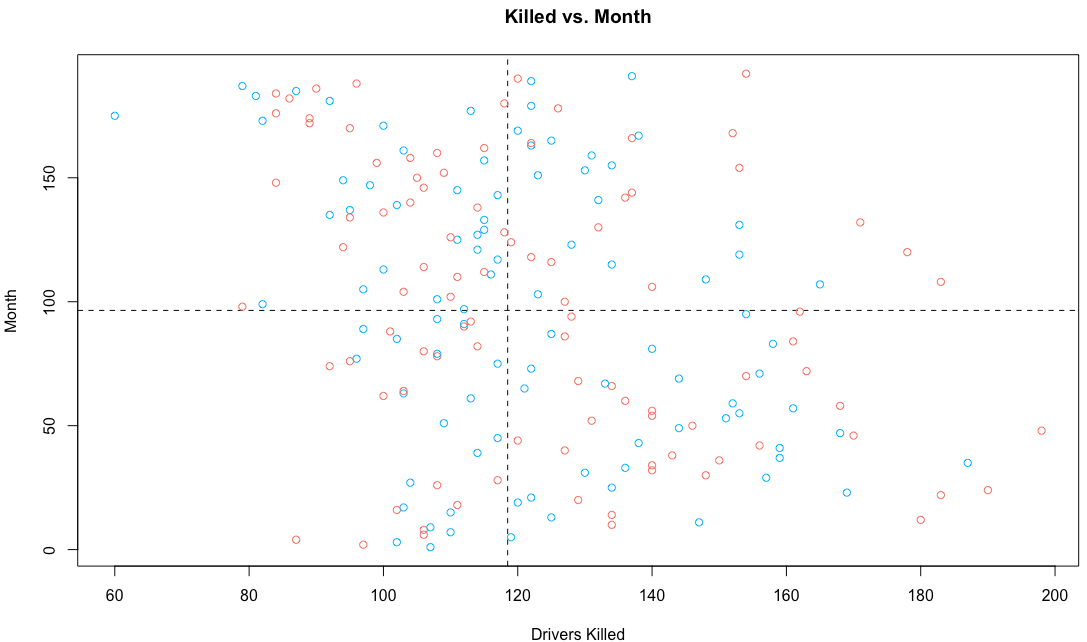
>

> #plot drivers killed vs month, show horizontal and vertical lines at median values for both variables

> plot(Seatbelts$DriversKilled, Seatbelts$Month, col = c("deepskyblue", "salmon"), main = "Killed vs. Month",

+ xlab = "Drivers Killed", ylab = "Month",

+ abline(v=c(median(Seatbelts$DriversKilled)), h=c(median(Seatbelts$Month)), lty = 2))



b) (2 points) Use table() to generate a 2x2 contingency table showing the fatality count classified by killed and month. Use the uncorrected chisq.test() to test at the 95% confidence level that killed and month are independent. State your conclusion.

> library(gmodels)

> #generate 2x2 contingency table

> my\_table <- table(killed, month)

> addmargins(my\_table)

month

killed below above Sum

below 37 59 96

above 59 37 96

Sum 96 96 192

>

> #generate 2x2 contigency table using gmodels package and crosstable function

> CrossTable(killed, month)

Cell Contents

|-------------------------|

| N |

| Chi-square contribution |

| N / Row Total |

| N / Col Total |

| N / Table Total |

|-------------------------|

Total Observations in Table: 192

| month

killed | below | above | Row Total |

-------------|-----------|-----------|-----------|

below | 37 | 59 | 96 |

| 2.521 | 2.521 | |

| 0.385 | 0.615 | 0.500 |

| 0.385 | 0.615 | |

| 0.193 | 0.307 | |

-------------|-----------|-----------|-----------|

above | 59 | 37 | 96 |

| 2.521 | 2.521 | |

| 0.615 | 0.385 | 0.500 |

| 0.615 | 0.385 | |

| 0.307 | 0.193 | |

-------------|-----------|-----------|-----------|

Column Total | 96 | 96 | 192 |

| 0.500 | 0.500 | |

-------------|-----------|-----------|-----------|

>

> #check critical value

> qchisq(.95, df=1)

[1] 3.841459

>

> #observed value = chi-square goodness of fit test on the my\_table

> chisq.test(my\_table, correct = FALSE)

Pearson's Chi-squared test

data: my\_table

X-squared = 10.083, df = 1, p-value = 0.001496

c) (5 points) Write a function that computes the uncorrected Pearson’s Chi-squared statistic based on a 2x2 contingency table with margins added. (Check Davies Section 11.1.1 pages 216-219, and Kabacoff Section 20.1.3 pages 473-474)*.* Add margins to the contingency table from (b) and submit to the function you have written. Compare the result with (b). You should be able to duplicate the X-squared value and the p-value.  
An example of code to include in the function is shown below. These statements calculate the expected value for each cell in the table. Using these statements the Pearson Chi Square Statistic may be calculated within the function and the value returned.

# To be used with 2x2 contingency tables that have margins added.

e11 <- x[3,1]\*x[1,3]/x[3,3]

e12 <- x[3,2]\*x[1,3]/x[3,3]

e21 <- x[3,1]\*x[2,3]/x[3,3]

e22 <- x[3,2]\*x[2,3]/x[3,3]

> #add margins to my\_table

> newtable <- addmargins(my\_table)

> newtable

month

killed below above Sum

below 37 59 96

above 59 37 96

Sum 96 96 192

> #define function

> cont\_table <- function(x){

+ total <- x[3,3]

+ e11 <- (x[1,3]\*x[3,1])/total

+ e21 <- (x[2,3]\*x[3,1])/total

+ e12 <- (x[1,3]\*x[3,2])/total

+ e22 <- (x[2,3]\*x[3,2])/total

+

+ ans1 <- ((x[1,1]-e11)^2)/e11

+ ans2 <- ((x[2,1]-e21)^2)/e21

+ ans3 <- ((x[1,2]-e12)^2)/e12

+ ans4 <- ((x[2,2]-e22)^2)/e22

+

+ answer <- sum(ans1, ans2, ans3, ans4)

+

+ return(answer)

+

+ }

>

> #call function

> cont\_table(newtable)

[1] 10.08333

As you can see the value of the uncorrected X2 value is the same when using the function created for a 2x2 contingency table and when using chisq.test function.