Predict 401

Programming with R Test #1

Latifi, Zeeshan

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 the 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.

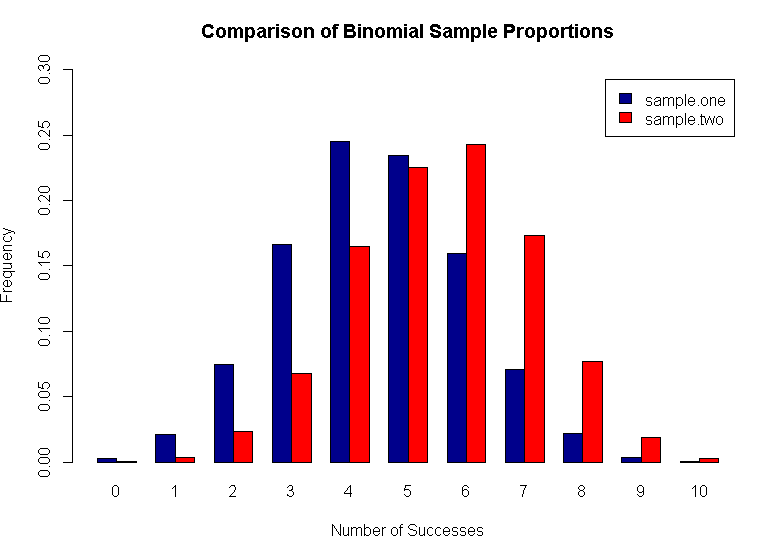
> counts <- rbind(sample.one, sample.two)

> 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 (25 points total)

1. (3 points) Create and print a vector that contains the following in this order:

* A sequence of integers from 6 to 10 (inclusive).
* A twofold repetition of the vector c(2, -5.1, -33).
* The value of the sum of 7/42 and 2.

> #R Test Number 1

>

> #sequence of integers 6 to 10

> integers <- seq(6,10,1)

> print(integers)

[1] 6 7 8 9 10

>

> #twofold repetition of the vector

> vec <- c(2, -5.1, -33)

> twovec <- rep(vec,2)

> print(twovec)

[1] 2.0 -5.1 -33.0 2.0 -5.1 -33.0

>

> #vector with a value of the sum 7/42 and 2

> val\_sum\_vector <- c(sum((7/42),2))

> print(val\_sum\_vector)

[1] 2.166667

>

> #combine the three vectors created

> final\_vec <- c(integers, twovec, val\_sum\_vector)

> print(final\_vec)

[1] 6.000000 7.000000 8.000000 9.000000 10.000000 2.000000 -5.100000 -33.000000 2.000000 -5.100000 -33.000000

[12] 2.166667

* 1. Extract the first and last elements of the vector you have created to form another vector using the extracted elements. Print this second vector.

> #extract first and last elements

> sec\_fv <- c(final\_vec[1], tail(final\_vec, n=1))

> print(sec\_fv)

[1] 6.000000 2.166667

* 1. Form a third vector from the elements not extracted in (a). Print this vector.

> #Form a third vector from the elements not extracted in (a)

> third\_vec <- final\_vec[2:11]

> print(third\_vec)

[1] 7.0 8.0 9.0 10.0 2.0 -5.1 -33.0 2.0 -5.1 -33.0

* 1. Use the vectors from (a) and (b) to reconstruct and print the original first vector.

> #reconstruct the original vector

> orig\_vec <- append(sec\_fv, third\_vec, after = 1)

> print(orig\_vec)

[1] 6.000000 7.000000 8.000000 9.000000 10.000000 2.000000 -5.100000 -33.000000 2.000000 -5.100000 -33.000000

[12] 2.166667

1. (5 points) The expression y = a + bx + cx2 is a quadratic function.   
   1. Write a function in R that accepts values for a, b, c and x and returns a value for y.

> #R Test Number 2

>

> #create quadratic function

> quad\_func <- function(x,a,b,c){

+ y <- a + b\*x + c\*(x^2)

+ return(y)

+ }

* 1. Create and save a vector, x, of 101 equally spaced values from -2 to 2 inclusive.

> #create vector of 101 values from -2 to 2

> x <- c(seq(-2,2,length=101))

* 1. Using the following values a = -1, b = 0, c = 1 with your function, calculate values for a vector, y, using the vector x from (b).

> #use values provided with quadratic function and use values from vector x

> y <- quad\_func(x,-1,0,1)

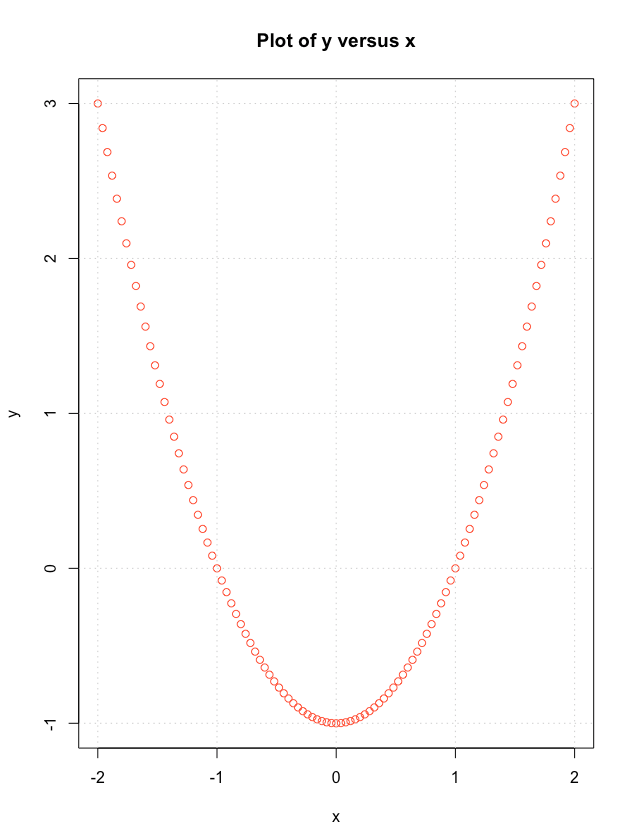
* 1. Plot y versus x in color. Add a title to the plot and other features to this plot.

> #plot y versus x

> myColor <- "#FF6347"

> plot(x,y,main = "Plot of y versus x", col = myColor)

> grid(nx = NULL, ny = NULL, col = "lightgray", lwd = par("lwd"), equilogs = TRUE)



* 1. Confirm the roots of the specified quadratic function are -1 and 1.

> #confirm the roots

> which(0.0000 == y)[[1]]

[1] 26

> which(0.0000 == y)[[2]]

[1] 76

> which(-1.0000 == x)[[1]]

[1] 26

> which(1.0000 == x)[[1]]

[1] 76

>

> x[26]

[1] -1

> y[26]

[1] 0

> x[76]

[1] 1

> y[76]

[1] 0

1. Use the trees data set (shown in the Quick Start Guide) for the following. This data set has three variables (Girth, Height, Volume) on 31 trees.
2. (2 points) Use apply() to present the median values for the three variables in trees. Using R and logicals, give the row number and print the three meplasurements for the tree that has a Volume measurement equal to the median Volume.

> #R Test Number 3

>

> #pull in data trees

> data(trees)

>

> #find median for three variables

> apply(trees,2,median)

Girth Height Volume

12.9 76.0 24.2

b. (2 points) Assume the Girth is the circumference of a circle. You may either use pi as   
 supplied in R as a library constant, or the value pi = 3.14159 to calculate the diameter  
 of each tree. Present a five-value summary of your results with summary().

> #print the row number and values that match the median of trees volume

> med\_val <- median(trees$Volume)

>

> if (med\_val == median(trees$Volume)){

+ print(trees[trees$Volume==med\_val, ])

+ }

Girth Height Volume

11 11.3 79 24.2

>

> #calculate diameter

> diameter <- trees$Girth/pi

> summary(diameter)

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

2.642 3.517 4.106 4.217 4.854 6.557

c. (2 points) Convert each diameter calculated in (b) to a radius r. Calculate the cross-  
 sectional area of each tree using pi\*r2. Present a five-value summary of your results  
 with summary().

> #calculate radius

> radius <- diameter/2

> area <- pi\*(radius)^2

> summary(area)

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

5.482 9.717 13.240 14.730 18.550 33.770

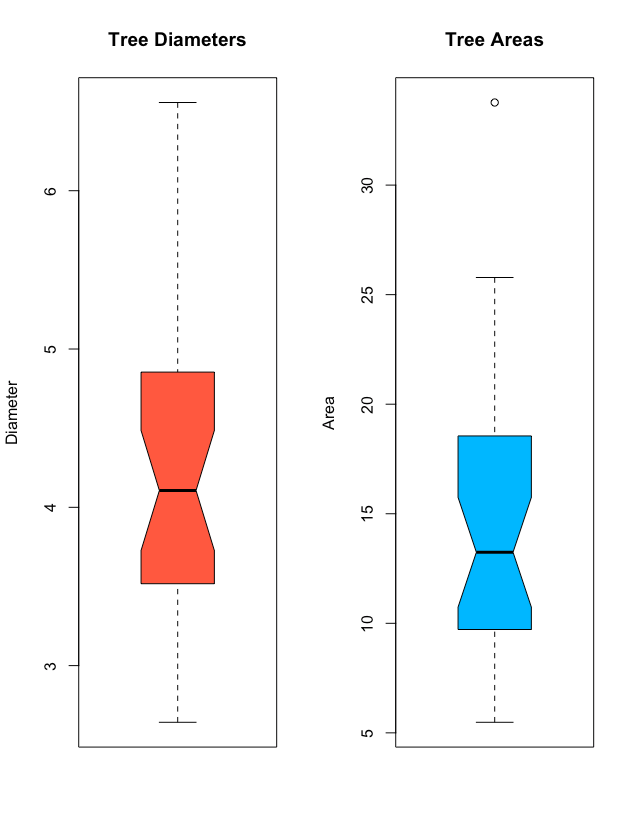
d. (2 points) Use par(mfrow = c(1,2)) and present notched, colored boxplots of the  
 diameters and areas calculated in (b) and (c). Label each accordingly.

> #create boxplots

> myColor <- "#FF6347"

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

> boxplot(diameter, main = "Tree Diameters", notch = TRUE, ylab = "Diameter", col = myColor)

> boxplot(area, main = "Tree Areas", notch = TRUE, ylab = "Area", col = "deepskyblue")  


e. (2 points) Using R as a calculator, demonstrate that the outlier revealed in the   
 boxplot of area is not an extreme outlier.

> mean(area)

[1] 14.72581

> mean(area, trim = .10)

[1] 14.14537

1. (3 points) Use matrix operations shown in the *Quick Start Guide* to solve the following system of linear equations. Display the R program and the numerical solution for *x*, y and z. Demonstrate your solution is correct by substituting values for x, y and z.

**x – y + z = 1, x + y – z = 1 and x + y + z = 3**

> #R Test Number 4

>

> #set up matrix

> lin\_equat <- matrix(c(1,-1,1,1,1,-1,1,1,1),nrow = 3, byrow = TRUE)

> lin\_equat

[,1] [,2] [,3]

[1,] 1 -1 1

[2,] 1 1 -1

[3,] 1 1 1

>

> #right side of the equal sign

> answers <- matrix(c(1,1,3), nrow = 3, byrow = TRUE)

> answers

[,1]

[1,] 1

[2,] 1

[3,] 3

>

> #use solve to solve the equation

> solution <- solve(lin\_equat, answers)

> solution

[,1]

[1,] 1

[2,] 1

[3,] 1

>

> #assign values to x y and z

> x = solution[1,1]

> y = solution[2,1]

> z = solution[3,1]

>

> #test solution

> x-y+z

[1] 1

> x+y-z

[1] 1

> x+y+z

[1] 3

1. (4 points) Use set.seed(123) and rnorm(), with mean = 0. Generate two different random samples each of size n = 1000. Designate the first sample as x and use a standard deviation of 2. Designate the second sample as y and use a standard deviation of 1.

> #R Test Number 5

>

> #generate random samples

> set.seed(123)

> x <- rnorm(1000, mean = 0, sd = 2)

> y <- rnorm(1000, mean = 0, sd = 1)

* 1. Generate a new object using cbind(x,y). Use apply() with this object to compute the sample standard deviation for each column (x and y).

> #create new object

> new\_obj <- cbind(x,y)

>

> #apply function to find sd

> apply(new\_obj,2,sd)

x y

1.983390 1.009674

* 1. Use par(mfrow = c(2,1)) and present two histograms in color with titles, one for x and the second for y. Maintain comparability of the x-axes with xlim = c(-6, 6).

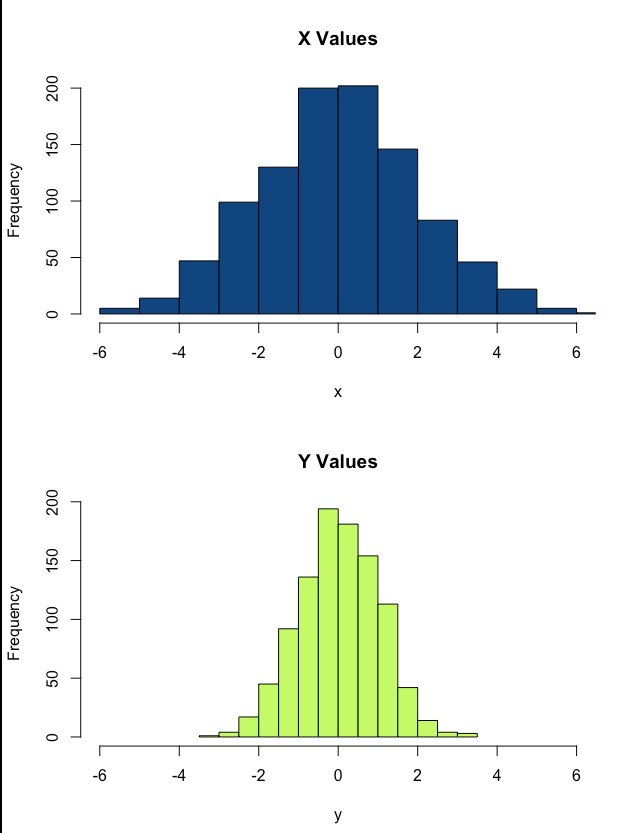
> #create histograms

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

>

> hist(x, main = "X Values", col = "dodgerblue4", xlim = c(-6,6))

> hist(y, main = "Y Values", col = "darkolivegreen1", xlim = c(-6,6))



* 1. The two vectors, x and y, are not correlated Use the vectors x and y without sorting or any other manipulation to produce a colored and titled scatterplot of their values with x on the x-axis. (In other words, plot consecutive pairs of data points (x1, y1), (x2, y2), etc.). Position a legend in the lower left corner which indicates what the standard deviation is for each variable.

> #create scatter plot for x and y values

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

> plot(x,y, main = "X Versus Y Scatterplot", col = "firebrick1")

> legend("bottomleft", legend=c(print(paste("x sd = ", round(sd(x), digits = 4), print(paste("y sd = ", round(sd(y), digits = 4)))))), cex = 0.75)

[1] "x sd = 1.9834 y sd = 1.0097"

