# Homework #4 Submission

## R Script (Code)

#

# Course: IST687

# Name: Joyce Woznica

# Homework 4 - Samples HW

# Due Date: 2/5/2019

# Date Submitted:

#

# Step 1: Write a summarizing function to understand the distribution of a vector

# 1) The function, call it 'printVecInfo' should take a vector as input

# 2) The function should print the following:

# a) Mean

# b) Median

# c) Min & Max

# d) Standard Deviation

# e) Quantiles (at 0.05 and 0.95)

# f) Skewness

# install package for skewness function

install.packages("moments")

library(moments)

# install package for counting strings

install.packages("stringr")

library(stringr)

# This function creates a clean dataframe of the results from a quantile

# function call with probsV as a vector of the alpha and the 1-alpha

# I use this in the printVecInfo function

myQuantDF<-function(xVect,probsV)

{

quant<-quantile(xVect,probsV)

quantDF <- data.frame(id=names(quant), values=unname(quant), stringsAsFactors = FALSE)

return(quantDF)

}

# the printVecInfo function

printVecInfo <- function(vect)

{

# change these to print statements

# need to put these in a dataframe that gets returned

# what statistic, value

retv<-c(cat(sprintf("Mean is: %f\n", mean(vect))),

cat(sprintf("Median is: %f\n", median(vect))),

cat(sprintf("Minimum is: %f\n", min(vect))),

cat(sprintf("Maximum is: %f\n", max(vect))),

cat(sprintf("Standard deviation is: %f\n", sd(vect))))

# now I work with a nice way to print out the quantile function

# hardcoded to be alpha of 0.05

quantDF<-myQuantDF(vect,c(0.05,0.95))

index<-1 # for the index of the vector

while (index <= 2)

{

retv<-append(retv, cat(sprintf("Quantile for %s is: %f\n", quantDF$id[index], quantDF$values[index])))

index<-index+1

}

retv<-append(retv,cat(sprintf("Skewness is: %f\n", skewness(vect))))

}

# 3) Use this vector to test function

myVect<-c(1,2,3,4,5,6,7,8,9,10,50)

printVecInfo(myVect)

# Step 2: Creating Samples in a jar

# 4) Create a variable 'jar' that has 50 red and 50 blue marbles

# hint: the jar can have strings being 'red' and 'blue'

red<-"red"

blue<-"blue"

v.red<-replicate(50,red)

v.blue<-replicate(50,blue)

jar<-c(v.red,v.blue)

# 5) Confirm that there are 50 reds by summing the samples that are red

length(jar[jar=='red'])

# 6) Sample 10 marbles from the jar, how many are red?

# What percentage are red?

mySample<-sample(jar,10)

length(mySample[mySample==red])

# I decided to create a percent function

percentColor <- function(theVector,theColor)

{

thePercent<-length(which(theVector==theColor))/length(theVector)

return(thePercent)

}

sprintf("%1.2f%%", 100\*percentColor(mySample, red))

# 7) Do the sampling 20 times, using the 'replicate' command. This should generate

# a list of 20 numbers. Each number is the mean of how many reds there were in 10

# samples. Use the printVectInfo to see information of the samples. Also generate

# a histogram of the samples.

# note: use method #2

# sum up the 1's (matches) of 'red' in the size 10 sample from jar

# to determine how many reds in 10 samples of 20 numbers

sum(replicate(20,mean(str\_count(sample(jar,size=10,replace=TRUE),red))))

testvec1<-replicate(20,mean(str\_count(sample(jar,size=10,replace=TRUE),red)))

printVecInfo(testvec1)

hist(testvec1)

# 8) Repeat #7, but this time, sample the jar 100 times. You should get 20 numbers

# this time each number represents the mean of how many reds there are in the

# 100 samples.

# Use your PrintVecInfo to see information of the samples. Also generate a

# histogram of the samples.

testvec2<-replicate(20,mean(str\_count(sample(jar,size=100,replace=TRUE),red)))

printVecInfo(testvec2)

hist(testvec2)

# 9) Repeat #8, but this time, replicate the sampling 100 times. You should get 100

# numbers, this time each number represents the mean of how many reds there were

# in the 100 samples. use you printVecInfo to see information about the samples.

# Also generate a histogram of the samples.

testvec3<-replicate(100,mean(str\_count(sample(jar,size=100,replace=TRUE),red)))

printVecInfo(testvec3)

hist(testvec3)

# Step 3: Explore the airquality dataset

# 10) Store the 'airquality' dataset into a temporary variable

myairQ<-airquality

# 11) Clean the dataset by removing all the NAs

# JLJW NOTE: I elected to use the mean for each column for the NAs as opposed to

# removing all the observations that had an NA for one of the values

# create a function to replace each column NA with mean for that column

replaceNAwMeans<-function(vec)

{

numcols<-length(colnames(vec))

index<-1

while(index<=numcols)

{

theColV <- vec[,index]

theColV[is.na(theColV)]<-mean(theColV,na.rm=TRUE)

vec[,index]<-theColV

index<-index+1

}

return(vec)

}

# Update the dataframe with the modified frame with means applied to NA values

myairQ<-replaceNAwMeans(myairQ)

# 12) Explore the Ozone, Wind and Temp by doing a 'printVecInfo' on each as well as

# generating a histogram for each

# first for Ozone

printVecInfo(myairQ$Ozone)

hist(myairQ$Ozone)

# Next for Wind

printVecInfo(myairQ$Wind)

hist(myairQ$Wind)

# Next for Temp

printVecInfo(myairQ$Temp)

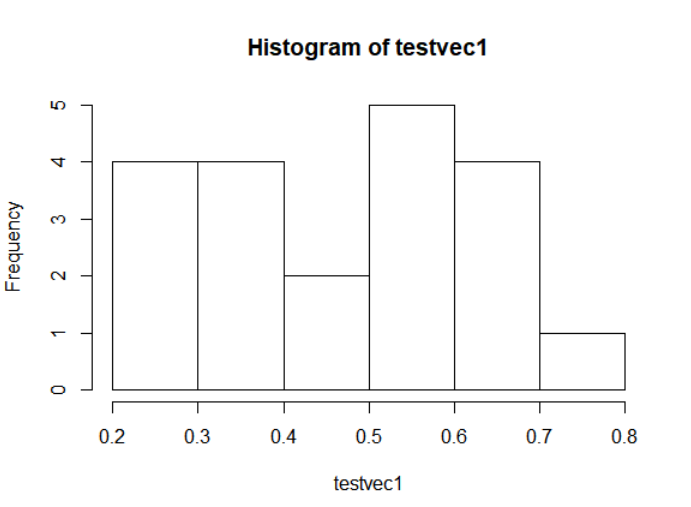
hist(myairQ$Temp)

## Console Log (Executed Code)

|  |
| --- |
| > #  > # Course: IST687  > # Name: Joyce Woznica  > # Homework 4 - Samples HW  > # Due Date: 2/5/2019  > # Date Submitted:  > #  > # Step 1: Write a summarizing function to understand the distribution of a vector  > # 1) The function, call it 'printVecInfo' should take a vector as input  > # 2) The function should print the following:  > # a) Mean  > # b) Median  > # c) Min & Max  > # d) Standard Deviation  > # e) Quantiles (at 0.05 and 0.95)  > # f) Skewness  >  > # install package for skewness function  > install.packages("moments")  Installing package into ‘C:/Users/Joyce/Documents/R/win-library/3.5’  (as ‘lib’ is unspecified)  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/moments\_0.14.zip'  Content type 'application/zip' length 55743 bytes (54 KB)  downloaded 54 KB  package ‘moments’ successfully unpacked and MD5 sums checked  The downloaded binary packages are in  C:\Users\Public\Documents\Wondershare\CreatorTemp\RtmpYvFhxv\downloaded\_packages  > library(moments)  Warning message:  package ‘moments’ was built under R version 3.5.2  >  > # install package for counting strings  > install.packages("stringr")  Installing package into ‘C:/Users/Joyce/Documents/R/win-library/3.5’  (as ‘lib’ is unspecified)  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/stringr\_1.3.1.zip'  Content type 'application/zip' length 194920 bytes (190 KB)  downloaded 190 KB  package ‘stringr’ successfully unpacked and MD5 sums checked  The downloaded binary packages are in  C:\Users\Public\Documents\Wondershare\CreatorTemp\RtmpYvFhxv\downloaded\_packages  > library(stringr)  Warning message:  package ‘stringr’ was built under R version 3.5.2  > # This function creates a clean dataframe of the results from a quantile  > # function call with probsV as a vector of the alpha and the 1-alpha  > # I use this in the printVecInfo function  > myQuantDF<-function(xVect,probsV)  + {  + quant<-quantile(xVect,probsV)  + quantDF <- data.frame(id=names(quant), values=unname(quant), stringsAsFactors = FALSE)  + return(quantDF)  + }  > # the printVecInfo function  > printVecInfo <- function(vect)  + {  + # change these to print statements  + # need to put these in a dataframe that gets returned  + # what statistic, value  + retv<-c(cat(sprintf("Mean is: %f\n", mean(vect))),  + cat(sprintf("Median is: %f\n", median(vect))),  + cat(sprintf("Minimum is: %f\n", min(vect))),  + cat(sprintf("Maximum is: %f\n", max(vect))),  + cat(sprintf("Standard deviation is: %f\n", sd(vect))))  + # now I work with a nice way to print out the quantile function  + # hardcoded to be alpha of 0.05  + quantDF<-myQuantDF(vect,c(0.05,0.95))  + index<-1 # for the index of the vector  + while (index <= 2)  + {  + retv<-append(retv, cat(sprintf("Quantile for %s is: %f\n", quantDF$id[index], quantDF$values[index])))  + index<-index+1  + }  + retv<-append(retv,cat(sprintf("Skewness is: %f\n", skewness(vect))))  + }  > # 3) Use this vector to test function  > myVect<-c(1,2,3,4,5,6,7,8,9,10,50)  > printVecInfo(myVect)  Mean is: 9.545455  Median is: 6.000000  Minimum is: 1.000000  Maximum is: 50.000000  Standard deviation is: 13.721251  Quantile for 5% is: 1.500000  Quantile for 95% is: 30.000000  Skewness is: 2.620396  > # Step 2: Creating Samples in a jar  > # 4) Create a variable 'jar' that has 50 red and 50 blue marbles  > # hint: the jar can have strings being 'red' and 'blue'  > red<-"red"  > blue<-"blue"  > v.red<-replicate(50,red)  > v.blue<-replicate(50,blue)  > jar<-c(v.red,v.blue)  > # 5) Confirm that there are 50 reds by summing the samples that are red  > length(jar[jar=='red'])  [1] 50  > # 6) Sample 10 marbles from the jar, how many are red?  > # What percentage are red?  > mySample<-sample(jar,10)  > length(mySample[mySample==red])  [1] 6  > # I decided to create a percent function  > percentColor <- function(theVector,theColor)  + {  + thePercent<-length(which(theVector==theColor))/length(theVector)  + return(thePercent)  + }  > sprintf("%1.2f%%", 100\*percentColor(mySample, red))  [1] "60.00%"  > # 7) Do the sampling 20 times, using the 'replicate' command. This should generate  > # a list of 20 numbers. Each number is the mean of how many reds there were in 10  > # samples. Use the printVectInfo to see information of the samples. Also generate  > # a histogram of the samples.  > # note: use method #2  >  > # sum up the 1's (matches) of 'red' in the size 10 sample from jar  > # to determine how many reds in 10 samples of 20 numbers  > sum(replicate(20,mean(str\_count(sample(jar,size=10,replace=TRUE),red))))  [1] 10.3  > testvec1<-replicate(20,mean(str\_count(sample(jar,size=10,replace=TRUE),red)))  > printVecInfo(testvec1)  Mean is: 0.515000  Median is: 0.550000  Minimum is: 0.200000  Maximum is: 0.800000  Standard deviation is: 0.169442  Quantile for 5% is: 0.295000  Quantile for 95% is: 0.705000  Skewness is: -0.173326  > hist(testvec1)  > # 8) Repeat #7, but this time, sample the jar 100 times. You should get 20 numbers  > # this time each number represents the mean of how many reds there are in the  > # 100 samples.  > # Use your PrintVecInfo to see information of the samples. Also generate a  > # histogram of the samples.  > testvec2<-replicate(20,mean(str\_count(sample(jar,size=100,replace=TRUE),red)))  > printVecInfo(testvec2)  Mean is: 0.480500  Median is: 0.485000  Minimum is: 0.390000  Maximum is: 0.560000  Standard deviation is: 0.051552  Quantile for 5% is: 0.409000  Quantile for 95% is: 0.560000  Skewness is: -0.029459  > hist(testvec2)  > # 9) Repeat #8, but this time, replicate the sampling 100 times. You should get 100  > # numbers, this time each number represents the mean of how many reds there were  > # in the 100 samples. use you printVecInfo to see information about the samples.  > # Also generate a histogram of the samples.  > testvec3<-replicate(100,mean(str\_count(sample(jar,size=100,replace=TRUE),red)))  > printVecInfo(testvec3)  Mean is: 0.494100  Median is: 0.490000  Minimum is: 0.390000  Maximum is: 0.590000  Standard deviation is: 0.048682  Quantile for 5% is: 0.410000  Quantile for 95% is: 0.560500  Skewness is: -0.235623  > hist(testvec3)  > # Step 3: Explore the airquality dataset  > # 10) Store the 'airquality' dataset into a temporary variable  > myairQ<-airquality  > # 11) Clean the dataset by removing all the NAs  > # JLJW NOTE: I elected to use the mean for each column for the NAs as opposed to  > # removing all the observations that had an NA for one of the values  >  > # create a function to replace each column NA with mean for that column  > replaceNAwMeans<-function(vec)  + {  + numcols<-length(colnames(vec))  + index<-1  + while(index<=numcols)  + {  + theColV <- vec[,index]  + theColV[is.na(theColV)]<-mean(theColV,na.rm=TRUE)  + vec[,index]<-theColV  + index<-index+1  + }  + return(vec)  + }  > # Update the dataframe with the modified frame with means applied to NA values  > myairQ<-replaceNAwMeans(myairQ)  > # 12) Explore the Ozone, Wind and Temp by doing a 'printVecInfo' on each as well as  > # generating a histogram for each  > # first for Ozone  > printVecInfo(myairQ$Ozone)  Mean is: 42.129310  Median is: 42.129310  Minimum is: 1.000000  Maximum is: 168.000000  Standard deviation is: 28.693372  Quantile for 5% is: 9.000000  Quantile for 95% is: 97.000000  Skewness is: 1.407648  > hist(myairQ$Ozone)  > # Next for Wind  > printVecInfo(myairQ$Wind)  Mean is: 9.957516  Median is: 9.700000  Minimum is: 1.700000  Maximum is: 20.700000  Standard deviation is: 3.523001  Quantile for 5% is: 4.600000  Quantile for 95% is: 15.500000  Skewness is: 0.344398  > hist(myairQ$Wind)  > # Next for Temp  > printVecInfo(myairQ$Temp)  Mean is: 77.882353  Median is: 79.000000  Minimum is: 56.000000  Maximum is: 97.000000  Standard deviation is: 9.465270  Quantile for 5% is: 60.200000  Quantile for 95% is: 92.000000  Skewness is: -0.374170  > hist(myairQ$Temp) |
|  |
| |  | | --- | |  | |

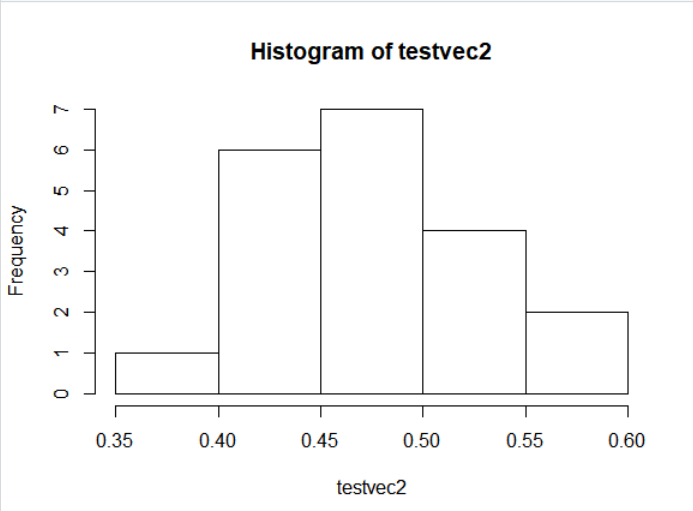
## Histogram for testvec1

This is the vector I used for #7: replicating 20 times a sample of 10 from jar



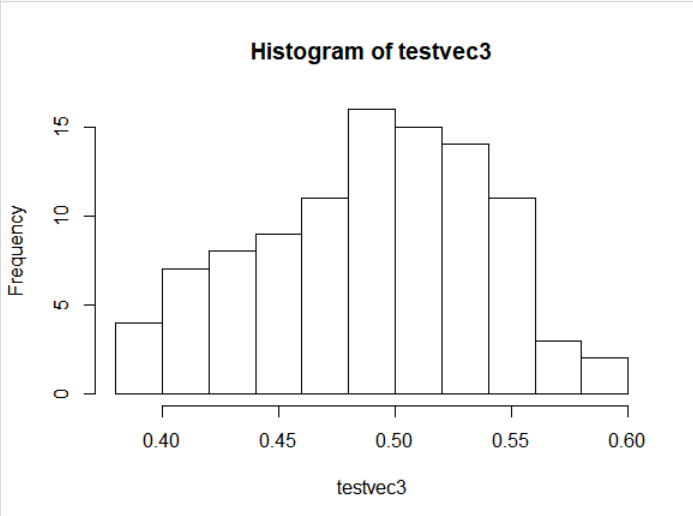
## Histogram for testvec2

This is the vector I used for #8: replicating 20 times a sample of 100 from jar

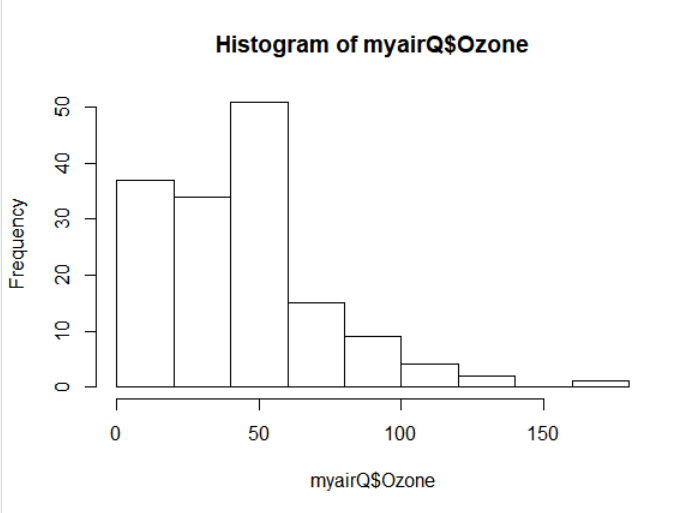


## Histogram for testvec3

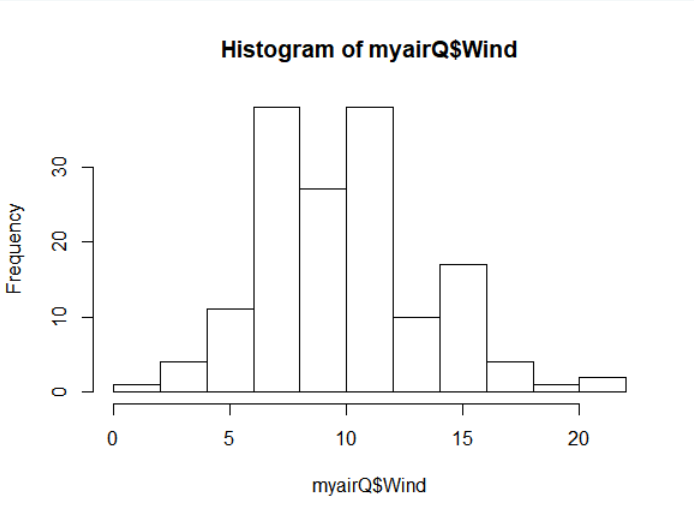
This is the vector I used for #9: replicating 100 times a sample of 100 from jar



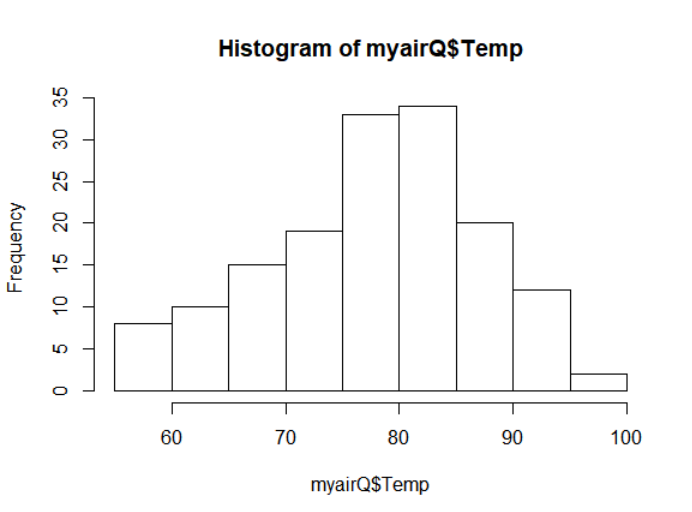
## Histogram for Ozone in Air Quality



## Histogram for Wind in Air Quality



## Histogram for Temp in Air Quality



## Global Data Screenshot

