Assignment 1

Wenhao Li 250959325

26/01/2020

# Question 1

#Find all possible R built-in functions related to exp distribution   
search()

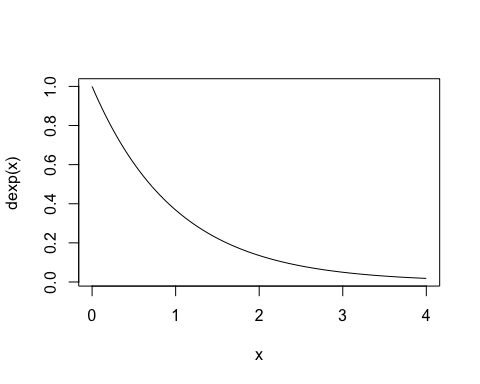
## [1] ".GlobalEnv" "package:stats" "package:graphics"   
## [4] "package:grDevices" "package:utils" "package:datasets"   
## [7] "package:methods" "Autoloads" "package:base"

objects(3)

## [1] "abline" "arrows" "assocplot" "axis"   
## [5] "Axis" "axis.Date" "axis.POSIXct" "axTicks"   
## [9] "barplot" "barplot.default" "box" "boxplot"   
## [13] "boxplot.default" "boxplot.matrix" "bxp" "cdplot"   
## [17] "clip" "close.screen" "co.intervals" "contour"   
## [21] "contour.default" "coplot" "curve" "dotchart"   
## [25] "erase.screen" "filled.contour" "fourfoldplot" "frame"   
## [29] "grconvertX" "grconvertY" "grid" "hist"   
## [33] "hist.default" "identify" "image" "image.default"   
## [37] "layout" "layout.show" "lcm" "legend"   
## [41] "lines" "lines.default" "locator" "matlines"   
## [45] "matplot" "matpoints" "mosaicplot" "mtext"   
## [49] "pairs" "pairs.default" "panel.smooth" "par"   
## [53] "persp" "pie" "plot" "plot.default"   
## [57] "plot.design" "plot.function" "plot.new" "plot.window"   
## [61] "plot.xy" "points" "points.default" "polygon"   
## [65] "polypath" "rasterImage" "rect" "rug"   
## [69] "screen" "segments" "smoothScatter" "spineplot"   
## [73] "split.screen" "stars" "stem" "strheight"   
## [77] "stripchart" "strwidth" "sunflowerplot" "symbols"   
## [81] "text" "text.default" "title" "xinch"   
## [85] "xspline" "xyinch" "yinch"

The R bulit-in functions related to exp distribution are dexp, pexp, qexp and rexp. Rexp generates random deviates. Dexp gives the density.

curve(dexp,from=0,to=4)



The code curve(dexp,from=0,to=4) plots Exp(1)’s distibution, where x is from 0 to 4 (domain of x is[0,4]).

# Question 2

# Use for loop for sum  
sum1<-0  
for (i in 1:100) {  
 sum1<-sum1+i  
}  
sum1

## [1] 5050

# Use for R function for sum  
sum(1:100)

## [1] 5050

sum1==sum(1:100)

## [1] TRUE

# Use for loop for mean  
sum1\_mean<-0  
for (i in 1:100) {  
 sum1\_mean<-sum1\_mean\*(i-1)/i+1  
}  
sum1\_mean

## [1] 50.5

# Use for R function for mean  
mean(1:100)

## [1] 50.5

sum1\_mean==mean(1:100)

## [1] TRUE

When we use R function and for loop to get the sum and mean, we can get the same result. However, using R function directly is better. Because it has less steps and it is more convenience to be read.

# Question 3

# n=10;Use for loop and compare with geometric series formula  
n\_10<-0  
for (j in 0:10) {  
 n\_10<- n\_10+1.08^j  
}  
n\_10

## [1] 16.64549

n\_10\_formula<-(1-1.08^11)/(1-1.08)  
n\_10\_formula

## [1] 16.64549

n\_10==n\_10\_formula

## [1] TRUE

#n=20;Use for loop and compare with geometric series formula  
n\_20<-0  
for (j in 0:20) {  
 n\_20<- n\_20+1.08^j  
}  
n\_20

## [1] 50.42292

n\_20\_formula<-(1-1.08^21)/(1-1.08)  
n\_20\_formula

## [1] 50.42292

n\_20==n\_20\_formula

## [1] TRUE

#n=30;Use for loop and compare with geometric series formula  
n\_30<-0  
for (j in 0:30) {  
 n\_30<- n\_30+1.08^j  
}  
n\_30

## [1] 123.3459

n\_30\_formula<-(1-1.08^31)/(1-1.08)  
n\_30\_formula

## [1] 123.3459

n\_30==n\_30\_formula

## [1] FALSE

#n=40;Use for loop and compare with geometric series formula  
n\_40<-0  
for (j in 0:40) {  
 n\_40<- n\_40+1.08^j  
}  
n\_40

## [1] 280.781

n\_40\_formula<-(1-1.08^41)/(1-1.08)  
n\_40\_formula

## [1] 280.781

n\_40==n\_40\_formula

## [1] TRUE

#n=10;Use sum function and compare with geometric series formula  
sum\_n10<-0  
sum\_n10<-sum(1.08^(0:10))  
sum\_n10

## [1] 16.64549

n\_10\_formula

## [1] 16.64549

sum\_n10==n\_10\_formula

## [1] TRUE

#n=20;Use sum function and compare with geometric series formula  
sum\_n20<-0  
sum\_n20<-sum(1.08^(0:20))  
sum\_n20

## [1] 50.42292

n\_20\_formula

## [1] 50.42292

sum\_n20==n\_20\_formula

## [1] TRUE

#n=30;Use sum function and compare with geometric series formula  
sum\_n30<-0  
sum\_n30<-sum(1.08^(0:30))  
sum\_n30

## [1] 123.3459

n\_30\_formula

## [1] 123.3459

sum\_n30==n\_30\_formula

## [1] TRUE

#n=40;Use sum function and compare with geometric series formula  
sum\_n40<-0  
sum\_n40<-sum(1.08^(0:40))  
sum\_n40

## [1] 280.781

n\_40\_formula

## [1] 280.781

sum\_n40==n\_40\_formula

## [1] TRUE

#Find difference when n=30  
n\_30-n\_30\_formula

## [1] 1.421085e-14

n\_30-sum\_n30

## [1] 1.421085e-14

c(sum\_n30,n\_30)

## [1] 123.3459 123.3459

Using sum function and geometric series formula can give the same result. When use for loop, it gives the same result with geometric series formula when n=10,20,40. When n=30, the result of for loop is different from the result of geometric series formula. However,when n=30, the difference between the result of for loop and the result of geometric series formula is small. The numerical rounding may leads to this difference. Also, using sum function is better. It has less step and easier to be read.

# Question 4

Sample size 1000

n<-1000  
set.seed(721)  
Sam1000<-numeric(15)  
for (i in 1:15) {  
 #Generate 1000 random numbers from Normal Distribution   
 sample1000<-rnorm(n,0,1)  
 #Absolute values  
 abs\_sample1000<-abs(sample1000)  
 #The absolute values that is less than 2  
 small\_abs\_sample1000<-abs\_sample1000<2  
 #The proportion  
 Sam1000[i]<-mean(small\_abs\_sample1000)  
}  
Sam1000

## [1] 0.949 0.949 0.953 0.945 0.966 0.952 0.952 0.958 0.958 0.947 0.957 0.957  
## [13] 0.943 0.955 0.942

Sample size 10000

n<-10000  
set.seed(721)  
Sam10000<-numeric(15)  
for (i in 1:15) {  
 #Generate 10000 random numbers from Normal Distribution   
 sample10000<-rnorm(n,0,1)  
 #Absolute values  
 abs\_sample10000<-abs(sample10000)  
 #The absolute values that is less than 2  
 small\_abs\_sample10000<-abs\_sample10000<2  
 #The proportion  
 Sam10000[i]<-mean(small\_abs\_sample10000)  
}  
Sam10000

## [1] 0.9529 0.9505 0.9564 0.9552 0.9537 0.9542 0.9543 0.9511 0.9523 0.9533  
## [11] 0.9535 0.9541 0.9543 0.9547 0.9561

Comparation

var(Sam1000)

## [1] 4.288571e-05

max(Sam1000)-min(Sam1000)

## [1] 0.024

var(Sam10000)

## [1] 2.653524e-06

max(Sam10000)-min(Sam10000)

## [1] 0.0059

When sample size is 1000, the variance is 4.288571e-05, and the range is 0.024. When sample size is 10000, the variance is 4.288571e-05, and the range is 0.0059. The sample which size is larger(10000) has smaller variance and range. As the sample size becomes larger,it will has smaller variance and range and the variation becomes smaller. the variation of sample size 10000 does become smaller.

#Question 5

col1<-seq(1:8)  
col2<-rep(1:2,each=4)  
col3<-rep(1:2,each=2,times=2)  
col4<-rep(1:2,each=1,times=4)  
Q5<-cbind(col1,col2,col3,col4)  
matrix(Q5,nrow=8,ncol=4,byrow=FALSE)

## [,1] [,2] [,3] [,4]  
## [1,] 1 1 1 1  
## [2,] 2 1 1 2  
## [3,] 3 1 2 1  
## [4,] 4 1 2 2  
## [5,] 5 2 1 1  
## [6,] 6 2 1 2  
## [7,] 7 2 2 1  
## [8,] 8 2 2 2

#Question 6

#Load build-in data frame cars   
data(cars)

#(a)

help(cars)

There are 50 observations and 2 variables in the datasets. The name of variables are speed and dist.

#(b)

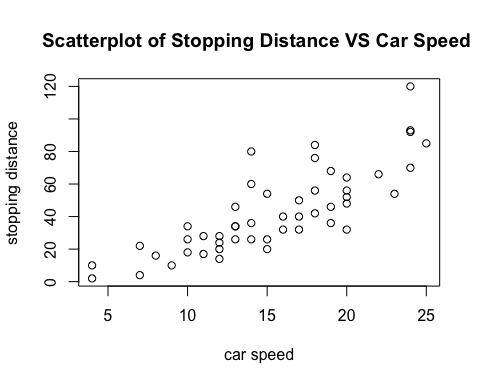
#Find the observations that the car speed is 20 miles per hour  
speed20<-cars[cars$speed==20,]   
#Get the mean of stopping distance of the car with 20 miles per hour speed   
mean(speed20$dist)

## [1] 50.4

So the mean stopping distance for all observations for which the speed was 20 miles per hour is 50.4ft.

#(c)

plot(x=cars$speed,y=cars$dist,main="Scatterplot of Stopping Distance VS Car Speed ",xlab="car speed",ylab="stopping distance")

 There is a positive relationship between Stopping Distance and Speed. It is may be a linear positive relationship.

#Question 7

#Load the built-in data frame USArrests  
data("USArrests")

#(a)

nrow(USArrests)

## [1] 50

ncol(USArrests)

## [1] 4

So there are 50 rows and 4 columns in this data frame.

#(b)

#Calculate the median of each columb  
median(USArrests$Murder)

## [1] 7.25

median(USArrests$Assault)

## [1] 159

median(USArrests$UrbanPop)

## [1] 66

median(USArrests$Rape)

## [1] 20.1

So the median of column Murder, Assault, UrbanPop and Rape are 7.25, 159, 66 and 30.1.

#(c)

#Find the observations that the UrbanPop exceeds 77%  
Urban\_77<-USArrests$Murder[USArrests$UrbanPop>77]  
#Get the average per capita murder rate  
mean(Urban\_77)

## [1] 8.5

#Find the observations that the UrbanPop less than 50%  
Urban\_50<-USArrests$Murder[USArrests$UrbanPop<50]  
#Get the average per capita murder rat  
mean(Urban\_50)

## [1] 8.25

As a result, the avergae per capita murder rate (Murder) in regions where the percentage of the population living in urban areas (UrbanPop) exceeds 77% is 8.5. The the avergae per capita murder rate (Murder) in regions where the percentage of the population living in urban areas (UrbanPop) less than 50% is 8.25. The the avergae per capita murder rate (Murder) in regions where the percentage of the population living in urban areas (UrbanPop) less than 50% is smaller than the the avergae per capita murder rate (Murder) in regions where the percentage of the population living in urban areas (UrbanPop) less than 50%.

#(d)

set.seed(721)  
#A sample with sampling 12 rows from USArrests data frame withous replacement  
Sam\_12\_row<-sample(1:50,12,replace = FALSE)  
#Corrspond the rows  
Sam\_12<-USArrests[Sam\_12\_row,]  
Sam\_12

## Murder Assault UrbanPop Rape  
## Missouri 9.0 178 70 28.2  
## Connecticut 3.3 110 77 11.1  
## Maryland 11.3 300 67 27.8  
## Alabama 13.2 236 58 21.2  
## New Mexico 11.4 285 70 32.1  
## Nevada 12.2 252 81 46.0  
## California 9.0 276 91 40.6  
## Michigan 12.1 255 74 35.1  
## Oregon 4.9 159 67 29.3  
## Massachusetts 4.4 149 85 16.3  
## Vermont 2.2 48 32 11.2  
## Rhode Island 3.4 174 87 8.3