Simmons-Week9Lab.R

jesse

2024-03-19

#Set the directory (1 point)  
setwd("C:/Users/jesse/OneDrive/IntroToDataScience")  
getwd()

## [1] "C:/Users/jesse/OneDrive/IntroToDataScience"

#Task 1: Write, test, and submit the necessary code in R to accomplish the following  
# 1. Generate a normal distribution, or 1,000 samples, with a mean of 80.  
  
distribution <- (rnorm(1000, mean = 80, sd = 10 ))  
  
#2. Write a function that takes three variables – a vector, a min and a max – and returns the number of elements in the vector that are between the min and max (including the min and max).   
#\*\*\* HINT: Plug in XXX below to create the function.  
#myFunction <-function(vector, min, max)  
#{b <- length(XXX[(XXX>= XXX) & (XXX <= XXX) ])  
#return(b)  
#}  
  
  
myFunction <-function(vector, min, max)  
{b <- length(vector[(vector>= min) & (vector <= max)])  
return(b)  
}  
  
#3. Use the function to see how many of your normal distribution samples are within the range of 79 to 81. Pass the "distribution" as the vector parameter, 79 as the minimum parameter, and 81 as the maximum parameter.   
  
count<- myFunction(vector = distribution, min = 79, max = 81)  
  
#4. Repeat 3 times (creating a normal distribution and then calling your function) to see if the results vary.  
  
count

## [1] 80

distribution <- (rnorm(1000, mean = 80, sd = 10 ))  
count<- myFunction(vector = distribution, min = 79, max = 81)  
count

## [1] 96

distribution <- (rnorm(1000, mean = 80, sd = 10 ))  
count<- myFunction(vector = distribution, min = 79, max = 81)  
count

## [1] 76

#count<- myFunction(vector = distribution, min = 79, max = 81)  
#count  
  
#Task 2: Write, test, and submit the necessary code in R to accomplish the following:  
#Install “actuar” OR “VGAM” package and load the “actuar” OR “VGAM” package. Either one of these packages will work. Just use whatever works for you.  
  
  
require (VGAM)

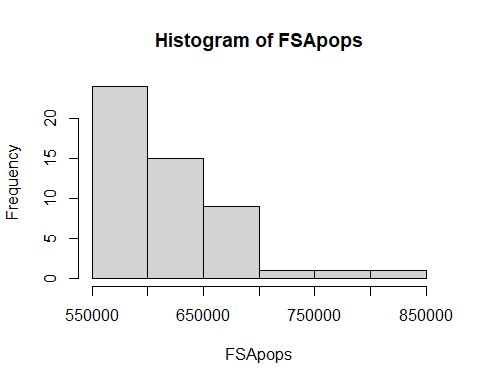
## Loading required package: VGAM

## Warning: package 'VGAM' was built under R version 4.3.3

## Loading required package: stats4

## Loading required package: splines

#Generate 51 random numbers in a Pareto distribution and assign them to a variable called “FSApops.” \*\* shape and scale arguments will be explained in 2.  
#2. Specify a “scale” and a “shape” for your Pareto distribution that makes it as similar as possible to the actual distribution of state populations on page 90 of the textbook.   
# rpareto(n, m, s): generating random numbers that fit a Pareto distribution  
# n -- generate 51 values; m -- location parameter (set it to be about the population size of Wyoming);   
# s -- vector of dispersion parameters.  
  
  
FSApops <- rpareto(51, 563626, 10)  
  
#3. Create a histogram that shows the distribution of values in FSApops.  
  
hist(FSApops)



#4. Use a command to report the actual mean and standard deviation of the 51 values stored in FSApops.  
  
mean(FSApops)

## [1] 617728

sd(FSApops)

## [1] 50803.77

#5. Use a command to report the minimum and maximum value of FSApops.  
min(FSApops)

## [1] 564498.7

max(FSApops)

## [1] 806140.4