p8105\_hw1\_hs3163

Hao Sun

9/14/2019

## Problem 1

Task 1: Create a data frame comprised of:

1. a random sample of size 8 from a standard Normal distribution
2. a logical vector indicating whether elements of the sample are greater than 0
3. a character vector of length 8
4. a factor vector of length 8, with 3 different factor “levels”

Answer:

#Set the seed for the random Norma distribution  
set.seed(1)  
problem1\_df = tibble(  
 #the first vector is a random sample of size 8 from a standard Normal distribution  
 vec\_random\_sample = (x = rnorm(8) ),  
 #the second vector is the a character vector of length 8 indexing the sample from normal distribution  
 vec\_char = c("first", "second", "third", "forth","fifth","sixth","seventh","eighth"),  
 #the third vector is the logical vector that used to determined whether elements of the sample are greater than 0  
 vec\_logical = vec\_random\_sample > 0 ,  
 #the forth vector is the factor that comprised of three levels.  
 vec\_factor = factor(c("low", "low", "low", "middle","middle","high","high","high"))  
)  
   
problem1\_df

## # A tibble: 8 x 4  
## vec\_random\_sample vec\_char vec\_logical vec\_factor  
## <dbl> <chr> <lgl> <fct>   
## 1 -0.626 first FALSE low   
## 2 0.184 second TRUE low   
## 3 -0.836 third FALSE low   
## 4 1.60 forth TRUE middle   
## 5 0.330 fifth TRUE middle   
## 6 -0.820 sixth FALSE high   
## 7 0.487 seventh TRUE high   
## 8 0.738 eighth TRUE high

Task 2: Try to take the mean of each variable in your dataframe. What works and what doesn’t? Answer:

# Find the mean for each vector in the dataset  
sapply(problem1\_df, mean)

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA  
  
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:  
## returning NA

## vec\_random\_sample vec\_char vec\_logical vec\_factor   
## 0.1314544 NA 0.6250000 NA

# Means can be found for numeric or logical vector: vec\_random\_sample and vec\_logical

Task 3: In some cases, you can explicitly convert variables from one type to another. Write a code chunk that applies the as.numeric function to the logical, character, and factor variables (please show this chunk but not the output). What happens, and why? Does this help explain what happens when you try to take the mean?

Answer

#Try to see if the as.numeric function will work on the three vector  
sapply(select(problem1\_df,2:4),as.numeric)  
  
#It turns out that the as numeric.function works on the logical and factor vector but not the character one. This may be because for the logic vector, one can use number to illustrate the true/false state and for the factor vector, one can use number to indicate what level of the factor it is. However there are no sensible way to do it for the charactor vector. This explain why the charater can not be used to found mean, but did not explian why the factor cannot be used to find mean.

Task 4: In a second code chunk:

convert the logical vector to numeric, and multiply the random sample by the result convert the logical vector to a factor, and multiply the random sample by the result convert the logical vector to a factor and then convert the result to numeric, and multiply the random sample by the result

Answer:

#convert the logical vector to numeric, and multiply the random sample by the result  
Pb1\_Q4\_1<-as.numeric(problem1\_df$vec\_logical)\* pull( problem1\_df, vec\_random\_sample)  
Pb1\_Q4\_1

## [1] 0.0000000 0.1836433 0.0000000 1.5952808 0.3295078 0.0000000 0.4874291  
## [8] 0.7383247

#For the false result, the product become zero. For the postive result, the orignin values are retained,  
  
  
#convert the logical vector to a factor, and multiply the random sample by the result  
as.factor(problem1\_df$vec\_logical)

## [1] FALSE TRUE FALSE TRUE TRUE FALSE TRUE TRUE   
## Levels: FALSE TRUE

Pb1\_Q4\_2<-as.factor(problem1\_df$vec\_logical)\* pull( problem1\_df, vec\_random\_sample)

## Warning in Ops.factor(as.factor(problem1\_df$vec\_logical),  
## pull(problem1\_df, : '\*' not meaningful for factors

Pb1\_Q4\_2

## [1] NA NA NA NA NA NA NA NA

#Since factor is not a numbner, therefore multipilication does not make sense.  
  
#convert the logical vector to a factor and then convert the result to numeric, and multiply the random sample by the result  
Pb1\_Q4\_3<-as.numeric(as.factor(problem1\_df$vec\_logical))\* pull( problem1\_df, vec\_random\_sample)  
Pb1\_Q4\_3

## [1] -0.6264538 0.3672866 -0.8356286 3.1905616 0.6590155 -0.8204684  
## [7] 0.9748581 1.4766494

#Since, when converted to number, the levels of factors will be denoted by 1 and 2 instead of 0 and 1, the result will be different from the first command of this part.

## Problem 2

Task 1:

This problem focuses the use of inline R code, plotting, and the behavior of ggplot for variables of different types.

Create a data frame comprised of: x: a random sample of size 500 from a standard Normal distribution y: a random sample of size 500 from a standard Normal distribution A logical vector indicating whether x + y > 1 A numeric vector created by coercing the above logical vector A factor vector created by coercing the above logical vector Write a short description of your vector using inline R code, including: \* the size of the dataset (using nrow and ncol) \* the mean, median, and standard deviation of x \* the proportion of cases for which x + y > 1

Answer:

problem2\_df = tibble(  
 #the first vector is x, a random sample of size 500 from a standard Normal distribution  
 x = (x = rnorm(500) ),  
 #the second vector is y, a random sample of size 500 from a standard Normal distribution  
 y = (y = rnorm(500) ),  
 #the third vector is the logical vector that used to determined whether x+y >1  
 vec\_logical\_sum\_larger\_than\_1 = x+y > 1 ,  
 #the forth vector is the above logical vector coerced into numeric form  
 pb2\_logic\_num = as.numeric(vec\_logical\_sum\_larger\_than\_1),  
 #the fifth vector is the above logical vector coerced into factor form  
 pb2\_logic\_factor = as.factor(vec\_logical\_sum\_larger\_than\_1),  
 )  
  
problem2\_df

## # A tibble: 500 x 5  
## x y vec\_logical\_sum\_larger\_t… pb2\_logic\_num pb2\_logic\_factor  
## <dbl> <dbl> <lgl> <dbl> <fct>   
## 1 0.576 1.16 TRUE 1 TRUE   
## 2 -0.305 -1.11 FALSE 0 FALSE   
## 3 1.51 -2.53 FALSE 0 FALSE   
## 4 0.390 -0.936 FALSE 0 FALSE   
## 5 -0.621 -0.967 FALSE 0 FALSE   
## 6 -2.21 0.0475 FALSE 0 FALSE   
## 7 1.12 -0.404 FALSE 0 FALSE   
## 8 -0.0449 0.231 FALSE 0 FALSE   
## 9 -0.0162 -0.422 FALSE 0 FALSE   
## 10 0.944 0.374 TRUE 1 TRUE   
## # … with 490 more rows

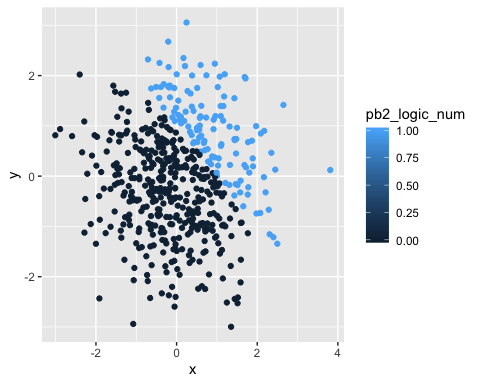
the size of the dataset is 5 variable each with the size of 500 The mean of x is 0.0196843 The median of x is -0.0384371 The standard diviation of x is 1.0137029 The proportion of cases for which x + y > 1 is 0.252

Task 2: Make a scatterplot of y vs x; color points using the logical variable (adding color = … inside of aes in your ggplot code should help). Make a second and third scatterplot that color points using the numeric and factor variables, respectively, and comment on the color scales.

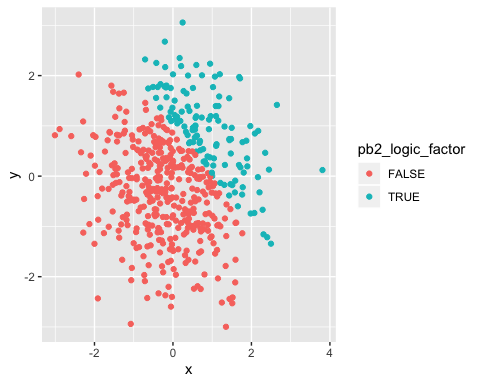
#scatterplot of y vs x; color points using the logical variable  
ggplot(problem2\_df, #Input dataset  
 aes(x = x, #Define variable that input  
 y=y,  
 color = vec\_logical\_sum\_larger\_than\_1))+geom\_point() # Define how color are devided and how datapot are arranged.



#scatterplot of y vs x; color points using the numeric variable  
ggplot(problem2\_df, #Input dataset  
 aes(x = x, #Define variable that input  
 y=y,  
 color = pb2\_logic\_num))+geom\_point() # Define how color are devided and how datapot are arranged.



#scatterplot of y vs x; color points using the factor variable  
ggplot(problem2\_df, #Input dataset  
 aes(x = x, #Define variable that input  
 y=y,  
 color = pb2\_logic\_factor))+geom\_point() # Define how color are devided and how datapot are arranged.



Comments on color scale: For the logic and factor vector, the color scale is binary, while for the numerice vector,the color scale is continuouse.

Task 3: Export your first scatterplot to your project directory using ggsave.