Lab 1 R Basics

Srujana Vanka

18/01/2024

Lab 1 Lab Manual Exercise

copy and paste your work by following each example from the lab manual for this exercise

```
rm(list = setdiff(ls(), lsf.str()))
# Vectors and Factors
# Create a vector as input.
data <- c("East","West","East","North","North","East","West","West","West","Eas
t","North")

print(data)
```

```
##  [1] "East"  "West"  "East"  "North" "North" "East"  "West"  "West"  "West"
## [10] "East"  "North"
```

```
print(is.factor(data))
```

```
## [1] FALSE
```

```
# Apply the factor function.
factor_data <- factor(data)

print(factor_data)
```

```
##  [1] East  West  East  North North East  West  West  West  East  North
## Levels: East North West
```

```
print(is.factor(factor_data))
```

```
## [1] TRUE
```

```
# Data frames
# Create the vectors for data frame.
height <- c(132,151,162,139,166,147,122)
weight <- c(48,49,66,53,67,52,40)
gender <- c("male","male","female","female","male","female","male")

# Create the data frame.
input_data <- data.frame(height,weight,gender)
print(input_data)
```

```
##   height weight gender
## 1    132     48   male
## 2    151     49   male
## 3    162     66 female
## 4    139     53 female
## 5    166     67   male
## 6    147     52 female
## 7    122     40   male
```

```
# Test if the gender column is a factor.
print(is.factor(input_data$gender))
```

```
## [1] TRUE
```

```
# Print the gender column so see the levels.
print(input_data$gender)
```

```
## [1] male   male   female female male   female male
## Levels: female male
```

```
# # Function Syntax
#
# function_name <- function(arg_1, arg_2, ...) {
#    Function body
# }


# Create a function with arguments.
new.function <- function(a,b,c) {
   result <- a * b + c
   print(result)
}

# Call the function by position of arguments.
new.function(5,3,11)
```

```
## [1] 26
```

```
# Call the function by names of the arguments.
new.function(a = 11, b = 5, c = 3)
```

```
## [1] 58
```

```
# From Mariam Aly's tutorial
##############################################################################
## Factors
##############################################################################
# A factor is a vector object used to specify a discrete classification (groupin
g) of the components of other vectors of the same length.
# can be ordered or unordered

## Example for 'ragged arrays', which can have subclasses of different sizes
# say you have 6 subjects and 3 conditions in your experiment
# this is a list of the condition that each subject took part in
condition=c('faces','scenes','objects','faces','scenes','objects')
# can create a factor for condition
conditionf=factor(condition) #use the factor() function
# print to screen
print(conditionf)
```

```
## [1] faces   scenes  objects faces   scenes  objects
## Levels: faces objects scenes
```

```
# produces:
# faces   scenes  objects faces   scenes  objects
# Levels: faces objects scenes
# can ask specifically for the levels of the factor
levels(conditionf) # returns "faces"   "objects" "scenes"
```

```
## [1] "faces"   "objects" "scenes"
```

```
# you can then use the tapply() function to calculate things like the mean for a variable you have for each of your factors
# continued from above
accuracy=c(90,88,72,84,81,94) # accuracy for each of you 6 subjects, in the same order in which you input the conditions (i.e. f,s,o,f,s,o)

# now calculate the mean accuracy for each condition using tapply()
# this function takes this form: tapply(data,factor/index variable,function), where factor/index is the factor variable you created and function is what you want to do on the data

# so if you want to see mean age for males and females, data=age, factor/index=gender, function=mean
    # looks at data in the first variable as a function of different levels of the second variable
# note that tapply() will work even if the second argument is not a factor, because the argument will be coerced into a factor when necessary (using as.factor())

# e.g. calculate the mean
condaccmeans=tapply(accuracy,conditionf,mean)
print(condaccmeans)
```

```
##   faces objects  scenes
##    87.0    83.0    84.5
```

```
# returns :
# faces objects  scenes
  # 87.0    83.0    84.5
    # would work if you use tapply(accuracy,condition,mean) because condition would be coerced into a factor

# The function tapply() is used to apply a function, here mean(), to each group of components of the first argument, here accuracy, defined by the levels of the second component,here conditionf, as if they were separate vector structures. The result is a structure of the same length as the levels attribute of the factor containing the results.
```

Lab 1 Generalization exercises

use the code from above to attempt to solve the extra things we ask you do for this assignment

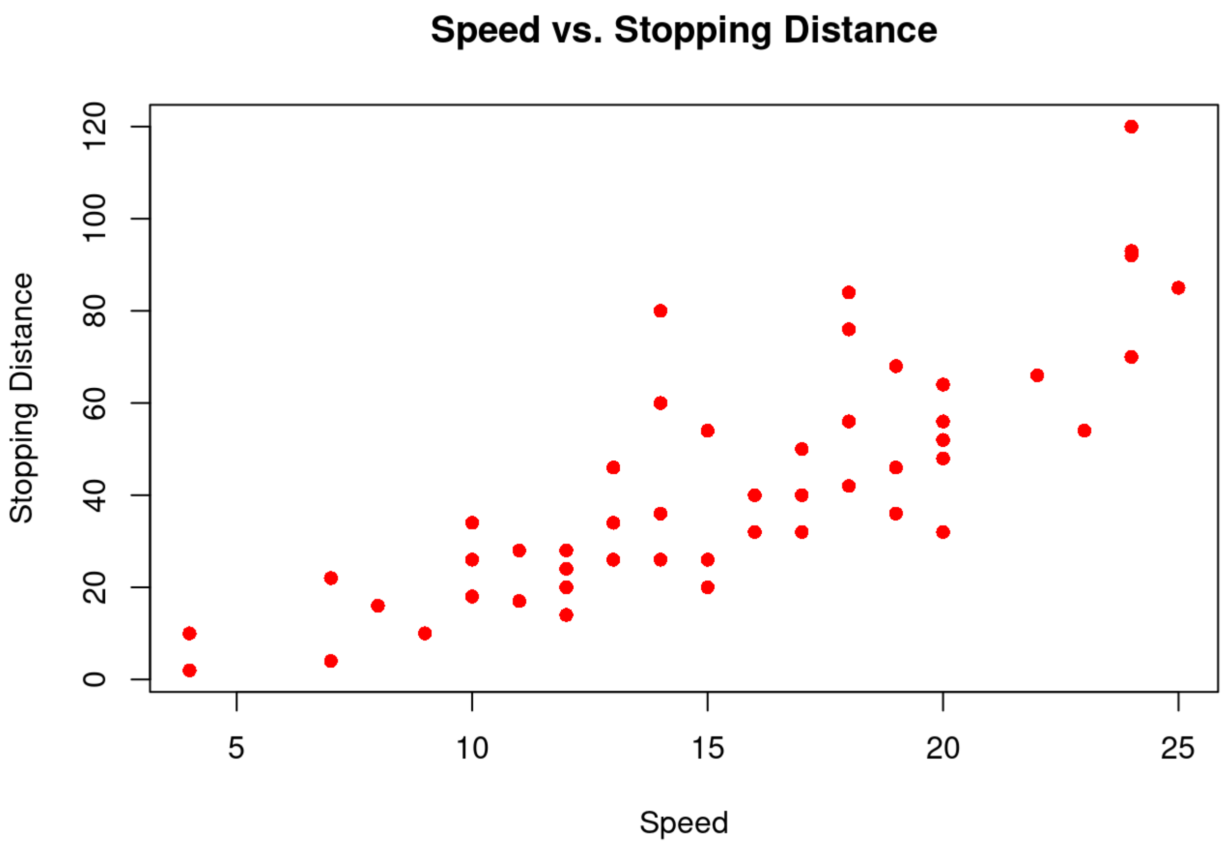
```
View(cars)
```

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

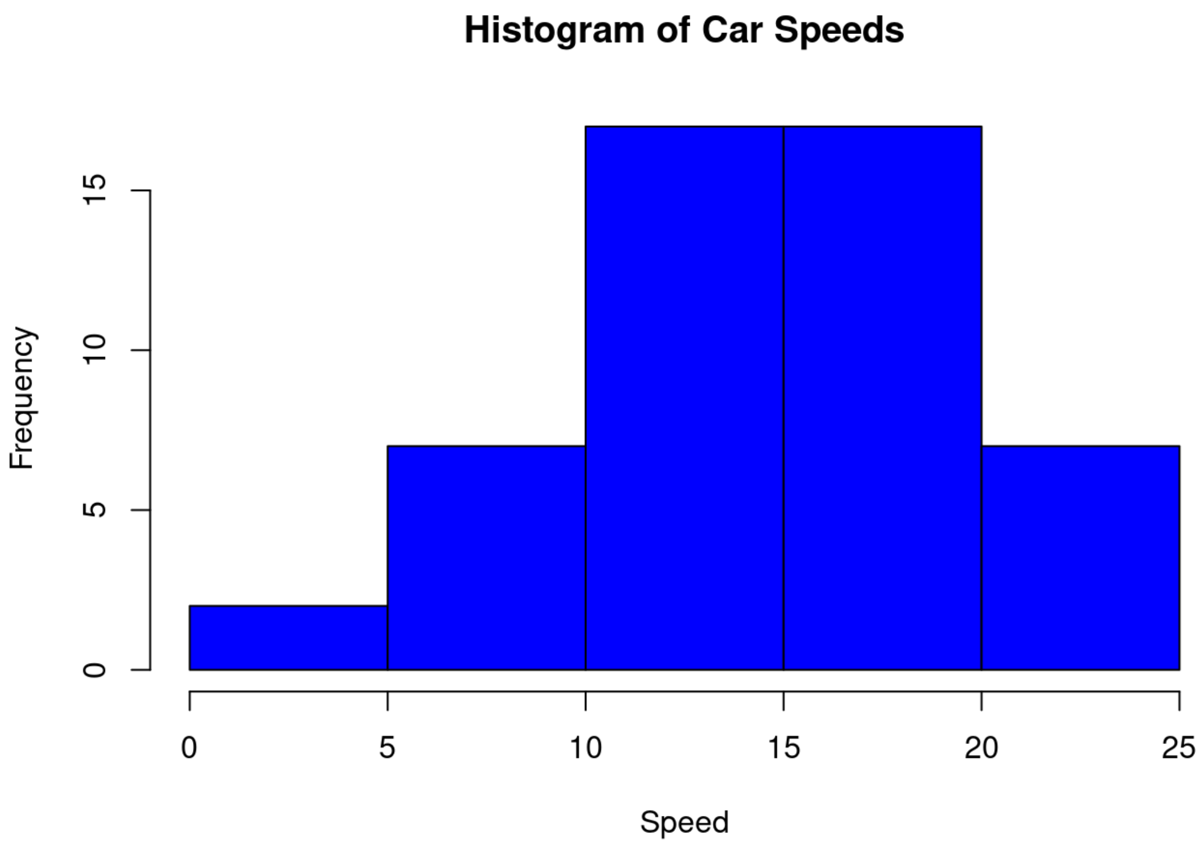
```
View(cars)
```

```
# Q1: what do you think is the relationship between speed and stopping distance
based on the scatterplot? +, -, or no relationship? (use the plot function)
# Use the plot function to visualize the relationship
plot(cars$speed, cars$dist, main = "Speed vs. Stopping Distance", xlab = "Spee
d", ylab = "Stopping Distance", pch = 16, col = "red")
```



1. The relationship between speed and stopping distance is directly proportional. As speed increases, stopping distance increases.

```
# Q2: plot a histogram of car speeds (use hist)
# Use the hist function with customized parameters

hist(cars$speed, main = "Histogram of Car Speeds", xlab = "Speed", col = "blue",
border = "black", xlim = c(0, max(cars$speed)))
```

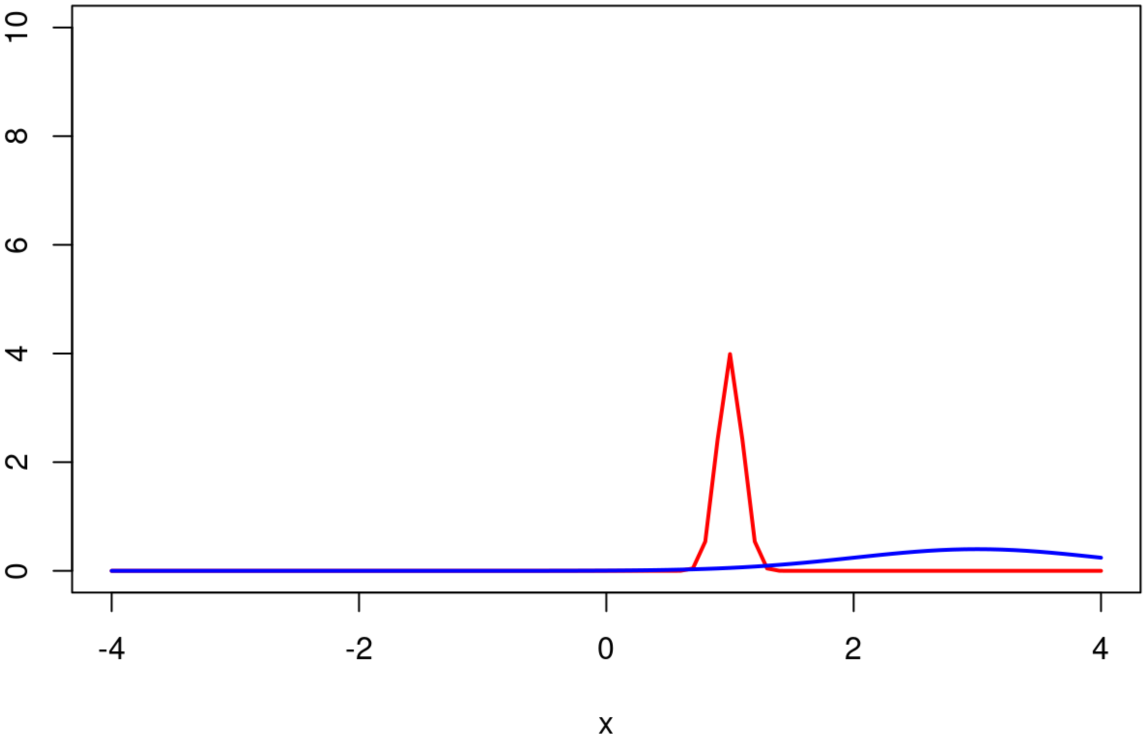



```
# Q3: what is the most frequent stopping distance in this dataset (an approx bin
of distances is fine)?
table_dist <- table(cars$dist)
most_frequent_dist <- as.numeric(names(table_dist)[which.max(table_dist)])

# Most frequent stopping distance
print(paste("The most frequent stopping distance is approximately", most_frequen
t_dist))
```

```
## [1] "The most frequent stopping distance is approximately 26"
```

Lab 1 Written answer question

```
# Grid of X-axis values
x <- seq(-4, 4, 0.1)

#-----------------------------------------
# Same standard deviation, different mean
#-----------------------------------------
# Mean 0, sd 1
plot(x, dnorm(x, mean = 1, sd = 0.1), type = "l",
     ylim = c(0, 10), ylab = "", lwd = 2, col = "red")
# Mean 3, sd 1
lines(x, dnorm(x, mean = 3, sd = 1), col = "blue", lty = 1, lwd = 2)
```


10
8
6
4
2
0
-4
-2
0
2
4
x