Homework 02 - STAT440

Joseph Sepich (jps6444)

09/04/2020

Problem 1

Which of the following is an appropriate variable name?

- (a) 1st_var
- (b) first_var
- (c) first.var

first_var or choice b is the appropriate variables name of the three choices. Variables cannot start with a number and using a dot in the variable name can be confused with function syntax.

Problem 2

Recall that if $x := (x_1, ..., x_d) \in R^d$, then the euclidean norm of x is $||x||_2 = \sqrt{\Sigma_{i=1}^{d} x_i^2}$. Let

$$V = [v_1, v_2, v_3, v_4, v_5] = \begin{vmatrix} 1 & 2 & 4 & -1 & 0 \\ 2 & 1 & -4 & 1 & 3 \\ 3 & 0 & 1 & -1 & 5 \end{vmatrix}$$

Create matrix V in R:

```
mat_v <- matrix(c(1, 2, 3, 2, 1, 0, 4, -4, 1, -1, 1, -1, 0, 3, 5), nrow = 3, ncol=5)
mat_v
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    4   -1    0
## [2,]    2    1   -4    1    3
## [3,]    3    0    1   -1    5
```

Use R to do the following

2a

Create a matrix D made out of the norm of all pairwise distances of the column vectors of V. That is, the ij^{th} entry of D is $||v_i - v_j||_2$.

```r
l2_norm <- function(vec) {
  sqrt(sum(vec^2))
}

num_cols <- dim(mat_v)[2]
mat_d <- matrix(1:25, nrow = num_cols, ncol = num_cols)
for (i in 1:num_cols) {
  for (j in 1:num_cols) {
    mat_d[i, j] <- l2_norm(mat_v[,i] - mat_v[,j])
  }
}
mat_d
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 0.000000 3.316625 7.000000 4.582576 2.449490
## [2,] 3.316625 0.000000 5.477226 3.162278 5.744563
## [3,] 7.000000 5.477226 0.000000 7.348469 9.000000
## [4,] 4.582576 3.162278 7.348469 0.000000 6.403124
## [5,] 2.449490 5.744563 9.000000 6.403124 0.000000
```

2b

Use D to compute the average and standard deviation of these distances. Be careful not to double count.

```r
dists <- mat_d[upper.tri(mat_d,diag=TRUE)]
# Average
print(mean(dists))
```

```
## [1] 3.63229
```

```r
# Standard Deviation
print(sd(dists))
```

```
## [1] 3.140712
```

2c

Find vectors y_j so that the j^{th} of D_{y_j} is the average distance from v_j to all other points. Report these numbers.

Problem 3

3a

Build a simple linear regression function using ordinary least squares that takes two inputs x and y, fits y to x, and returns the slope and intercept. Use it to fit the **iron** column to the **calcium** column in the **nutrient** dataset.

```r
ols_regress <- function(x, y) {
  slope_numerator <- cov(x, y)
  slope_denom <- var(x)
  slope <- slope_numerator / slope_denom
  inter <- mean(y) - slope * mean(x)
  return(list("slope" = slope, "intercept" = inter))
}
```

```r
# load dataset
nutrient_df <- read.csv('./data/nutrient.csv')
```

```r
# perform regression
model <- ols_regress(nutrient_df$calc, nutrient_df$iron)
# Slope
print(model$slope)
```

```
## [1] 0.005956363
```

```r
# Intercept
print(model$intercept)
```

```
## [1] 7.412836
```

3b

Learn how to use the R function **lm** and use it to fit iron to calcium. Use the **summary** function on the output of **lm** and compare it to the output of your function in (a).

```r
model <- lm(iron~calc,data=nutrient_df)
summary(model)
```

```
##
## Call:
## lm(formula = iron ~ calc, data = nutrient_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.029  -3.432  -0.799   2.401  45.907
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.4128358  0.3774502   19.64   <2e-16 ***
## calc        0.0059564  0.0005103   11.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.5 on 735 degrees of freedom
## Multiple R-squared:  0.1564, Adjusted R-squared:  0.1552
## F-statistic: 136.2 on 1 and 735 DF,  p-value: < 2.2e-16
```

The output of the lm function regression of fitting **iron** to **calcium** has the same estimate for the intercept and slope.

Book Problems

Chapter 2 Problem 1

Instead of copying the table in the book, use the full dataset Deer.txt, available in Canvas. Use $ instead of c to extract the appropriate columns and give the average for all animals, not just the seven that are shown. Hint: If you need to tell a function not to include NA values. use na.rm=TRUE as an argument.

```
# read dataset
deers <- read.delim('./data/Deer.txt')
```

```
# create length var
Length <- deers$LCT
Tb <- deers$Tb

# Average length
print(mean(Length, na.rm = TRUE))
```

```
## [1] 161.5138
```

Chapter 2 Problem 2

```
Farm <- deers$Farm
Month <- deers$Month

Boar <- cbind(Month, Length, Tb)

# Number of animals
print(nrow(Boar))
```

```
## [1] 1182
```

```
# Number of variables
print(ncol(Boar))
```

```
## [1] 3
```

```
# Number of animals, Number of variables
print(dim(Boar))
```

```
## [1] 1182    3
```

Chapter 2 Problem 5

```
# Confirm data type
print(str(deers))
```

```
## 'data.frame':    1182 obs. of  9 variables:
##  $ Farm   : chr  "AL" "AL" "AL" "AL" ...
##  $ Month  : int  10 10 10 10 10 10 10 10 10 10 ...
##  $ Year   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Sex    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ clas1_4: int  4 4 3 4 4 4 4 4 4 4 ...
##  $ LCT    : num  191 180 192 196 204 190 196 200 197 208 ...
##  $ KFI    : num  20.4 16.4 15.9 17.3 NA ...
##  $ Ecervi : num  0 0 2.38 0 0 0 1.21 0 0.8 0 ...
##  $ Tb     : int  0 0 0 0 NA 0 NA 1 0 0 ...
## NULL
```

```r
deers$sqrtLength <- sqrt(deers$LCT)
deers$sqrtLength[1:5]
```

```
## [1] 13.82027 13.41641 13.85641 14.00000 14.28286
```

```r
deer_list <- list(length = deers$LCT, Farm = Farm)
print(str(deer_list))
```

```
## List of 2
##  $ length: num [1:1182] 191 180 192 196 204 190 196 200 197 208 ...
##  $ Farm  : chr [1:1182] "AL" "AL" "AL" "AL" ...
## NULL
```

```r
deer_list$sqrtLength <- sqrt(deer_list$length)
deer_list$sqrtLength[1:5]
```

```
## [1] 13.82027 13.41641 13.85641 14.00000 14.28286
```

There was no real difference in performing the operation in the list versus the data.frame. This holds true, because the data.frame data structure is merely a list with certain rules imposed such as each element/column must be the same length.

Chapter 2 Problem 6

```r
data_file <- './data/ISIT.txt'
bio_read <- read.table(data_file)
# bio_scan <- scan(data_file, what="character")
bio_scan <- scan(data_file, what = list("", "", "", "", "", "", "", "", "", "", "", "", "", ""))

str(bio_read)
```

```
## 'data.frame':    790 obs. of  14 variables:
##  $ V1 : chr  "SampleDepth" "517" "582" "547" ...
##  $ V2 : chr  "Sources" "28.73" "27.9" "23.44" ...
##  $ V3 : chr  "Station" "1" "1" "1" ...
##  $ V4 : chr  "Time" "3" "3" "3" ...
##  $ V5 : chr  "Latitude" "50.1508" "50.1508" "50.1508" ...
```

```
##  $ V6 : chr  "Longitude" "-14.4792" "-14.4792" "-14.4792" ...
##  $ V7 : chr  "Xkm" "-34.106" "-34.106" "-34.106" ...
##  $ V8 : chr  "Ykm" "16.779" "16.779" "16.779" ...
##  $ V9 : chr  "Month" "4" "4" "4" ...
##  $ V10: chr  "Year" "2001" "2001" "2001" ...
##  $ V11: chr  "BottomDepth" "3939" "3939" "3939" ...
##  $ V12: chr  "Season" "1" "1" "1" ...
##  $ V13: chr  "Discovery" "252" "252" "252" ...
##  $ V14: chr  "RelativeDepth" "3422" "3357" "3392" ...
```

```r
str(bio_scan)
```

```
## List of 14
##  $ : chr [1:790] "SampleDepth" "517" "582" "547" ...
##  $ : chr [1:790] "Sources" "28.73" "27.9" "23.44" ...
##  $ : chr [1:790] "Station" "1" "1" "1" ...
##  $ : chr [1:790] "Time" "3" "3" "3" ...
##  $ : chr [1:790] "Latitude" "50.1508" "50.1508" "50.1508" ...
##  $ : chr [1:790] "Longitude" "-14.4792" "-14.4792" "-14.4792" ...
##  $ : chr [1:790] "Xkm" "-34.106" "-34.106" "-34.106" ...
##  $ : chr [1:790] "Ykm" "16.779" "16.779" "16.779" ...
##  $ : chr [1:790] "Month" "4" "4" "4" ...
##  $ : chr [1:790] "Year" "2001" "2001" "2001" ...
##  $ : chr [1:790] "BottomDepth" "3939" "3939" "3939" ...
##  $ : chr [1:790] "Season" "1" "1" "1" ...
##  $ : chr [1:790] "Discovery" "252" "252" "252" ...
##  $ : chr [1:790] "RelativeDepth" "3422" "3357" "3392" ...
```

```r
is.data.frame(bio_read)
```

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

```r
is.data.frame(bio_scan)
```

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

```r
is.matrix(bio_read)
```

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

```r
is.matrix(bio_scan)
```

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

The read.table function will read the text file directly into a data frame object while the scan function will create a single long vector containing each value in the text file. You can also scan each column into separate elements of a list by specifying a list in the what parameter of the scan function.