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)]
print(paste0('Average: ', mean(dists)))
```

```
## [1] "Average: 3.63228997170899"
```

```r
print(paste0('Standard Deviation: ', sd(dists)))
```

```
## [1] "Standard Deviation: 3.14071242397252"
```

2c

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

```r
mat_d %*% c(0.2,0.2,0.2,0.2,0.2)
```

```
##          [,1]
## [1,] 3.469738
## [2,] 3.540138
## [3,] 5.765139
## [4,] 4.299289
## [5,] 4.719435
```

```r
for (i in 1:5) {
  print(mean(mat_d[i,]))
}
```

```
## [1] 3.469738
## [1] 3.540138
## [1] 5.765139
## [1] 4.299289
## [1] 4.719435
```

The vector $y_j = (\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$ will make the vector Dy_j the average distance from each column vector to the other vectors, because there are 5 total columns, and matrix vector multiplication will multiply this value by the values in the row (which represent all the distances) and then sum them.

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)
print(paste0('Slope: ', model$slope))
```

```
## [1] "Slope: 0.00595636285775166"
```

```r
print(paste0('Intercept: ', model$intercept))
```

```
## [1] "Intercept: 7.41283579661136"
```

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.

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

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

print(paste0('Average length: ', mean(Length, na.rm = TRUE)))
```

```
## [1] "Average length: 161.513821892393"
```

Chapter 2 Problem 2

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

Boar <- cbind(Month, Length, Tb)

print(paste0('# of animals: ', nrow(Boar), ' same as ' ,dim(Boar)[1]))
```

```
## [1] "# of animals: 1182 same as 1182"
```

```r
print(paste0('# of vars: ', ncol(Boar), ' same as ', dim(Boar)[2]))
```

```
## [1] "# of vars: 3 same as 3"
```

Chapter 2 Problem 5

```r
# 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, header = TRUE)
# bio_scan <- scan(data_file, what="character")
bio_scan <- scan(data_file, what = list("", "", "", "", "", "", "", "", "", "", "", "", "", ""))

str(bio_read)
```

```
## 'data.frame':    789 obs. of  14 variables:
##  $ SampleDepth  : num  517 582 547 614 1068 ...
##  $ Sources      : num  28.7 27.9 23.4 18.3 12.4 ...
##  $ Station      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Time         : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Latitude     : num  50.2 50.2 50.2 50.2 50.2 ...
##  $ Longitude    : num  -14.5 -14.5 -14.5 -14.5 -14.5 ...
##  $ Xkm          : num  -34.1 -34.1 -34.1 -34.1 -34.1 ...
##  $ Ykm          : num  16.8 16.8 16.8 16.8 16.8 ...
##  $ Month        : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ Year         : int  2001 2001 2001 2001 2001 2001 2001 2001 2001 2001 ...
##  $ BottomDepth  : int  3939 3939 3939 3939 3939 3939 3939 3939 3939 3939 ...
##  $ Season       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Discovery    : int  252 252 252 252 252 252 252 252 252 252 ...
##  $ RelativeDepth: num  3422 3357 3392 3325 2871 ...
```

```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.

Chapter 3 Problem 2

```r
# extract data from station 1
station_1 <- bio_read[which(bio_read$Station == 1),]
summary(station_1$SampleDepth)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     517    1528    2520    2549    3652    3939
```

```r
# extract data from station 2
station_2 <- bio_read[which(bio_read$Station == 2),]
summary(station_2$SampleDepth)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     501    1821    3290    2760    3602    3916
```

```r
# extract data from station 3
station_3 <- bio_read[which(bio_read$Station == 3),]
summary(station_3$SampleDepth)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     516    1340    2169    2311    3733    3965
```

```r
# find low sample size stations
station_counts <- table(bio_read$Station)
station_counts
```

```
##
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
## 38 44 27  5 12 27 35 34 54 55 53 40 56 58 56 51 47 48 49
```

Stations 4 and 5 have considerably fewer observations, so we will omit them.

```r
# remove stations 4 and 5
bio_sub <- bio_read[which((bio_read$Station != 4) & (bio_read$Station != 5)),]
unique(bio_sub$Station)
```

```
##  [1]  1  2  3  6  7  8  9 10 11 12 13 14 15 16 17 18 19
```

```r
# extract 2002 data
data <- bio_read[which(bio_read$Year == 2002),]
paste0('# of rows: ', nrow(data))
```

```
## [1] "# of rows: 405"
```

```r
paste0('Unique years in data: ', unique(data$Year))
```

```
## [1] "Unique years in data: 2002"
```

```r
# extract April data
data <- bio_read[which(bio_read$Month == 4),]
paste0('# of rows: ', nrow(data))
```

```
## [1] "# of rows: 126"
```

```r
paste0('Unique months in data: ', unique(data$Month))
```

```
## [1] "Unique months in data: 4"
```

```r
# extract measurements greater than 2000m depth
data <- bio_read[which(bio_read$SampleDepth > 2000),]
paste0('# of rows: ', nrow(data))
```

```
## [1] "# of rows: 387"
```

```r
paste0('Min depth of data: ', min(data$SampleDepth))
```

```
## [1] "Min depth of data: 2003"
```

```r
# show data by increasing depth values
data <- bio_read[order(bio_read$SampleDepth),]
data[1:20,]
```

```
##     SampleDepth  Sources Station Time Latitude Longitude      Xkm     Ykm
## 39       501.00 21.53000       2    3  50.0910  -14.4665  -33.294  10.112
## 427      505.00 28.57000      13    2  49.8567  -13.9620    2.722 -15.890
## 83       516.00 24.43000       3    1  50.1337  -14.4992  -35.543  14.890
## 694      516.00 31.63000      18    3  49.4647  -15.5700 -113.383 -59.450
## 1        517.00 28.73000       1    3  50.1508  -14.4792  -34.106  16.779
## 541      518.70 59.55335      15    2  49.8070  -14.0643   -4.590 -21.447
## 112      522.00 26.45000       4    1  49.8358  -11.4977  179.313 -18.224
```

```
## 115      526.00 26.83000       5    2  49.8842  -11.6330  169.599  -13.067
## 775      531.00 18.83000      19    2  49.7792  -13.6275   26.693  -24.558
## 425      543.00 33.34000      13    2  49.8567  -13.9620    2.722  -15.890
## 3        547.00 23.44000       1    3  50.1508  -14.4792  -34.106   16.779
## 695      549.00 31.42000      18    3  49.4647  -15.5700 -113.383  -59.450
## 84       550.00 22.41000       3    1  50.1337  -14.4992  -35.543   14.890
## 540      554.97 77.93401      15    2  49.8070  -14.0643   -4.590  -21.447
## 657      556.00 16.72000      17    2  48.7772  -16.4845 -181.965 -135.902
## 110      559.00 27.56000       4    1  49.8358  -11.4977  179.313  -18.224
## 116      561.00 25.66000       5    2  49.8842  -11.6330  169.599  -13.067
## 755      567.00 29.59000      19    2  49.7792  -13.6275   26.693  -24.558
## 426      580.00 32.57000      13    2  49.8567  -13.9620    2.722  -15.890
## 693      580.00 36.33000      18    3  49.4647  -15.5700 -113.383  -59.450
##     Month Year BottomDepth Season Discovery RelativeDepth
## 39      4 2001        3981      1       252       3480.00
## 427     3 2002        3901      1       260       3396.00
## 83      4 2001        3977      1       252       3461.00
## 694    10 2002        4728      2       266       4212.00
## 1       4 2001        3939      1       252       3422.00
## 541     3 2002        3993      1       260       3474.30
## 112     4 2001         740      1       252        218.00
## 115     4 2001        1035      1       252        509.00
## 775    10 2002        2927      2       266       2396.00
## 425     3 2002        3901      1       260       3358.00
## 3       4 2001        3939      1       252       3392.00
## 695    10 2002        4728      2       266       4179.00
## 84      4 2001        3977      1       252       3427.00
## 540     3 2002        3993      1       260       3438.03
## 657    10 2002        4808      2       266       4252.00
## 110     4 2001         740      1       252        181.00
## 116     4 2001        1035      1       252        474.00
## 755    10 2002        2927      2       266       2360.00
## 426     3 2002        3901      1       260       3321.00
## 693    10 2002        4728      2       266       4148.00
```

```r
# show data at depths > 2000 in April
data <- bio_read[which((bio_read$SampleDepth > 2000) & (bio_read$Month == 4)),]
data[1:20,]
```

```
##    SampleDepth Sources Station Time Latitude Longitude     Xkm    Ykm Month
## 14        2003    3.80       1    3  50.1508  -14.4792 -34.106 16.779     4
## 15        2034    3.63       1    3  50.1508  -14.4792 -34.106 16.779     4
## 16        2068    2.81       1    3  50.1508  -14.4792 -34.106 16.779     4
## 17        2444    2.48       1    3  50.1508  -14.4792 -34.106 16.779     4
## 18        2504    1.98       1    3  50.1508  -14.4792 -34.106 16.779     4
## 19        2477    1.32       1    3  50.1508  -14.4792 -34.106 16.779     4
## 20        2536    1.32       1    3  50.1508  -14.4792 -34.106 16.779     4
## 21        3722    0.83       1    3  50.1508  -14.4792 -34.106 16.779     4
## 22        3446    0.66       1    3  50.1508  -14.4792 -34.106 16.779     4
## 23        3630    0.66       1    3  50.1508  -14.4792 -34.106 16.779     4
## 24        3660    0.66       1    3  50.1508  -14.4792 -34.106 16.779     4
## 25        3939    0.66       1    3  50.1508  -14.4792 -34.106 16.779     4
## 26        3414    0.50       1    3  50.1508  -14.4792 -34.106 16.779     4
## 27        3505    0.50       1    3  50.1508  -14.4792 -34.106 16.779     4
```

```
## 28         3534    0.50       1     3  50.1508  -14.4792 -34.106 16.779     4
## 29         3912    0.50       1     3  50.1508  -14.4792 -34.106 16.779     4
## 30         3568    0.33       1     3  50.1508  -14.4792 -34.106 16.779     4
## 31         3600    0.33       1     3  50.1508  -14.4792 -34.106 16.779     4
## 32         3697    0.33       1     3  50.1508  -14.4792 -34.106 16.779     4
## 33         3853    0.33       1     3  50.1508  -14.4792 -34.106 16.779     4
##    Year BottomDepth Season Discovery RelativeDepth
## 14 2001        3939      1       252          1936
## 15 2001        3939      1       252          1905
## 16 2001        3939      1       252          1871
## 17 2001        3939      1       252          1495
## 18 2001        3939      1       252          1435
## 19 2001        3939      1       252          1462
## 20 2001        3939      1       252          1403
## 21 2001        3939      1       252           217
## 22 2001        3939      1       252           493
## 23 2001        3939      1       252           309
## 24 2001        3939      1       252           279
## 25 2001        3939      1       252             0
## 26 2001        3939      1       252           525
## 27 2001        3939      1       252           434
## 28 2001        3939      1       252           405
## 29 2001        3939      1       252            27
## 30 2001        3939      1       252           371
## 31 2001        3939      1       252           339
## 32 2001        3939      1       252           242
## 33 2001        3939      1       252            86
```

Sampling Basics

Problem 1

Obtain 1000 samples from the chi-squared distribution with 1 degree of freedom by first sampling 1000 samples Z_i from a standard normal distribution, and then applying the appropriate transformation. Plot a histogram of the results. Overlay a curve onto the histogram denoting the true density.

```r
# standard normal sample
num <- 1000
samples <- rnorm(num)
```

Recall the definition of a chi-square distribution with k degrees of freedom:

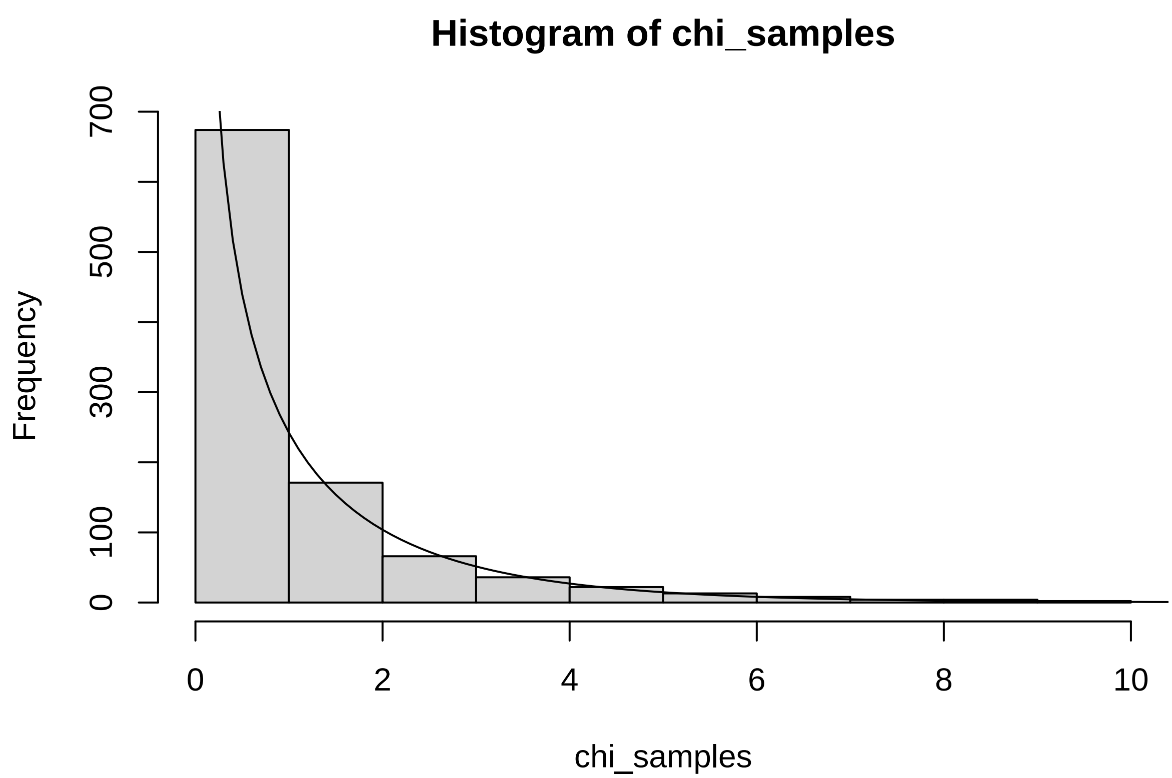
$$Q = \Sigma_{i=1}^{k} Z_i^2$$

For one degree of freedom this would merely be squaring each sample.

$$Q_i = Z_i^2$$

```r
# transform
chi_samples <- samples^2
# histogram
hist(chi_samples)

# overlay true density
df <- 1
x <- seq(0, 12, 0.1)
y <- dchisq(x, df) * num
lines(x,y)
```



Problem 2

Repeat the previous question, but produce t-distributed random variables with 5 degrees of freedom. Do this by only generating standard normals (then transforming them the appropriate way).

Recall the Student's t distribution follows the formula below where the distribution has n degrees of freedom.

$$\frac{Z}{\sqrt{\chi^2/n}}$$

```r
df <- 5
t <- vector(mode='numeric', length=num)
z_samples <- rnorm(df * num)
```

```r
samples <- rnorm(num)
test <- c(0)
for (i in seq(1, df * num, df)) {
  start <- i
  end <- i + df -1
  sub_samp <- z_samples[start:end]
  test <- append(test, sub_samp)
  chi <- sum(sub_samp^2)
  t[i/df] <- samples[i/df] / (sqrt(chi / df))
}
hist(t)

# overlay true sample
x <- seq(-20, 20, 0.1)
y <- dt(x, df) * num
lines(x,y)
```