STAT 210
Applied Statistics and Data Analysis: Homework 2

Due on Sep. 21/2025

You cannot use artificial intelligence tools to solve this homework.
Show complete solutions to get full credit. Writing code is not enough to answer a question. Your comments are more important than the code. Do not write comments in chunks. Label your graphs appropriately

Question 1

You will need the file `weights.txt`. This file has (simulated) data on an experiment to test the effects of two different types of diet. The set has 100 observations (50 males and 50 females), and there are three types of `diet`, coded as 1, 2, and 3. The first type (coded 1) corresponds to the control group and has subjects who did not change their usual diet during the eight weeks that the experiment lasted. The other two groups (coded 2 and 3) correspond to the diets that are being tested. The weight is measured in kilograms and the height in centimeters.

(a) Read the file `weights.txt` and store it in an object called `data1`. Look at the structure of `data1` using the function `str`. Check whether there are missing values in the file using the function `is.na`.

```r
#reads the file, header assumes the first line is column name
data1 <- read.table("weights.txt", header = TRUE)
str(data1)
```

```
## 'data.frame':    100 obs. of  7 variables:
##  $ subject    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ gender     : chr  "F" "F" "F" "F" ...
##  $ age        : int  50 30 43 36 43 41 24 24 57 45 ...
##  $ height     : int  169 170 175 162 173 157 170 153 161 168 ...
##  $ pre_weight : int  76 70 73 65 75 42 68 41 43 66 ...
##  $ diet       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ post_weight: num  74.4 73.6 75.3 65.6 77.7 ...
```

```r
# is.na returns a matching size matrix of the arguement with true values for missing spots.
# any() returns true if at least one element in the arguement is true.
any(is.na(data1))
```

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

(b) The columns `pre_weight` and `post_weight` have the weight for the subjects before starting the diet and eight weeks after following the diet, respectively. Create a new column in `data1` called `diff` that has the difference in weights (final weight minus initial weight).

```
#adds a new difference column that is the after - before weight.
data1$diff <- data1$post_weight - data1$pre_weight
head(data1)
```

```
##   subject gender age height pre_weight diet post_weight       diff
## 1       1      F  50    169         76    1    74.40019 -1.5998062
## 2       2      F  30    170         70    1    73.64465  3.6446476
## 3       3      F  43    175         73    1    75.25867  2.2586682
## 4       4      F  36    162         65    1    65.64755  0.6475503
## 5       5      F  43    173         75    1    77.66597  2.6659703
## 6       6      F  41    157         42    1    44.36541  2.3654054
```

(c) Using the function `subset`, create a new data frame named `data1b` that has the variables `gender`, `age`, `height`, `diet`, and `diff` in `data1`. Using `data1b` and the function `tapply`, calculate mean and standard deviation for `diff` according to `diet`. Compare these results and comment.

```
#makes a subset from the main data frame with only specific columns
data1b <- subset(data1, select = c(gender, age, height, diet, diff))
str(data1b)
```

```
## 'data.frame':    100 obs. of  5 variables:
##  $ gender: chr  "F" "F" "F" "F" ...
##  $ age   : int  50 30 43 36 43 41 24 24 57 45 ...
##  $ height: int  169 170 175 162 173 157 170 153 161 168 ...
##  $ diet  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ diff  : num  -1.6 3.645 2.259 0.648 2.666 ...
```

```
#calculate the means of the 3 different diet groups
tapply(data1b$diff, data1b$diet, mean)
```

```
##          1          2          3
##  0.3076043  0.2291316 -1.2873649
```

```
# calculate the standarad deviations of the 3 different diet groups
tapply(data1b$diff, data1b$diet, sd)
```

```
##         1         2         3
## 2.5301875 3.6408409 0.9576384
```

the average change of diets 1 and 2 is an increase of weight which means its not really working. also their standard deviation is high which means it might work for a specific group and doesn't work for another group. diet 3 had the expected result of a weight losing diet, where the average difference is lower weight (negative) and the standard deviation is low which means it works in a similar way on the entire sample.

(d) Using `data1b` and the function `tapply`, calculate mean and standard deviation for `diff` according to `diet` and `gender`. Compare these results and comment.

```r
tapply(data1b$diff, list(data1b$diet, data1b$gender), mean)
```

```
##            F          M
## 1  1.0094958 -0.3529995
## 2  0.2901713  0.1642768
## 3 -1.3280988 -1.2466309
```

```r
tapply(data1b$diff, list(data1b$diet, data1b$gender), sd)
```

```
##          F         M
## 1 2.076769 2.7935578
## 2 4.058366 3.2713625
## 3 1.006437 0.9354238
```

According to the data diet 1 works for male individuals, diet 2 seems to not have a major difference, and diet 3 works for both genders. the standard deviations shows larger deviations for the first 2 diets and a low deviation for the 3rd one. this could mean that the first 2 diets are not as strict as diet 3, allowing for more individual variability which increases the SD.

(e) Use the function `split` on the file `data1` with argument `gender` and store the result in an object called `d1`. What type of object is `d1`? Use the function `quantile` on the variable `height` on each of the components of `d1` to get a summary of the height for the different genders classes. Store the results in two vectors named `ql1` and `ql2`. Calculate `ql2/ql1` and interpret the result.

```r
#splits the data frame into a list of multiple data frames according to the number of gender options.
d1 <- split(data1, data1$gender)
str(d1)
```

```
## List of 2
##  $ F:'data.frame':   50 obs. of  8 variables:
##   ..$ subject    : int [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ gender     : chr [1:50] "F" "F" "F" "F" ...
##   ..$ age        : int [1:50] 50 30 43 36 43 41 24 24 57 45 ...
##   ..$ height     : int [1:50] 169 170 175 162 173 157 170 153 161 168 ...
##   ..$ pre_weight : int [1:50] 76 70 73 65 75 42 68 41 43 66 ...
##   ..$ diet       : int [1:50] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ post_weight: num [1:50] 74.4 73.6 75.3 65.6 77.7 ...
##   ..$ diff       : num [1:50] -1.6 3.645 2.259 0.648 2.666 ...
##  $ M:'data.frame':   50 obs. of  8 variables:
##   ..$ subject    : int [1:50] 51 52 53 54 55 56 57 58 59 60 ...
##   ..$ gender     : chr [1:50] "M" "M" "M" "M" ...
##   ..$ age        : int [1:50] 50 39 48 46 54 44 42 27 42 49 ...
##   ..$ height     : int [1:50] 193 172 182 167 188 181 181 177 163 179 ...
##   ..$ pre_weight : int [1:50] 100 62 92 43 112 81 88 84 41 66 ...
##   ..$ diet       : int [1:50] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ post_weight: num [1:50] 97.3 61.4 91.9 42.7 108.3 ...
##   ..$ diff       : num [1:50] -2.691 -0.6187 -0.0856 -0.3228 -3.7141 ...
```

```r
#splits the data by cutting it into percentiles with 0% and 100% being the minimum and maximum.
ql1 <- quantile(d1$F$height)
ql2 <- quantile(d1$M$height)
#divides male height percentiles over the female.
ql2 / ql1
```

```
##       0%      25%      50%      75%     100%
## 1.053691 1.037267 1.066265 1.076923 1.066298
```

The division shows the factor between the percentile values of each groups height. it shows that all percentile values are >1 which means the average height, minimum, and maximum height values.

(f) The body mass index (BMI) is defined as a person's weight in kilograms divided by the square of height in meters. Add a column named `bmi` to the data frame `data1` with the value of this index for each subject using the weight before the experiment started. Count how many subjects have BMI above 30.

```
#adds a bmi column using its calculation formula.
data1$bmi <- data1$pre_weight / (((data1$height) / 100)^2)
# counts the amount of entires with bmi higher than 30.
sum(data1$bmi > 30)
```

```
## [1] 10
```

Question 2

(a) Create two matrices. The first, called `m1`, has dimension 3×5 and the components are values simulated from a binomial distribution with size 15 and probability 0.25. The second, called `m2`, has dimensions 5×3 and the components are values simulated from a Poisson distribution with parameter 3. Create also a vector `v1` of length three from a negative binomial distribution with parameters `size = 2` and `prob = 0.25`.

```
#Creates a matrix of size 3x5, each entry corresponds to the number
#of successes in 15 trials with prob 0.25
m1 <- matrix(rbinom(15, size = 15, prob = 0.25), nrow = 3, ncol = 5)
head(m1)
```

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

```
#creates a similar matrix with flipped sizes
#but with a poission dist using lambda = 3
m2 <- matrix(rpois(15, lambda = 3), nrow = 5, ncol = 3)
head(m2)
```

```
##      [,1] [,2] [,3]
## [1,]    3    3    4
## [2,]    2    4    2
## [3,]    2    3    2
## [4,]    1    3    0
## [5,]    1    3    6
```

```
#creates a vector of negative binom distribution
v1 <- rnbinom(3, size = 2, prob = 0.25)
```

(b) Create a list named `hwlist` that has as a first component `m1`, second component `m2`, and third component `v1`. The names of these components should be `item1`, `item2`, and `item3`, respectively. Use the function `rm` to remove `m1`, `m2`, and `v1` from the working environment.

```
#create a list using the two matrices and vector.
hwlist <- list(item1 = m1, item2 = m2, item3 = v1)
#remove the original variables to not store them twice
rm(m1, m2, v1)
```

(c) Using matrix multiplication, multiply `item1` times `item2` and store the result in `hwlist` under the name `item4`. Multiply `item1` and `item2` also in the reverse order, but do not store the outcome

```
#matrix multiply the two matrices into another entry (produces 3x3)
hwlist$item4 <- hwlist$item1 %*% hwlist$item2
#matrix multiply in reverse order (produces 5x5)
hwlist$item2 %*% hwlist$item1
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   27   37   50   38   47
## [2,]   20   28   40   28   38
## [3,]   18   25   35   25   33
## [4,]    9   13   20   12   19
## [5,]   27   37   50   42   49
```

```
head(hwlist)
```

```
## $item1
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    3    4    5    3    4
## [2,]    2    3    5    3    5
## [3,]    3    4    5    5    5
##
## $item2
##      [,1] [,2] [,3]
## [1,]    3    3    4
## [2,]    2    4    2
## [3,]    2    3    2
## [4,]    1    3    0
## [5,]    1    3    6
##
## $item3
## [1] 5 4 7
##
## $item4
##      [,1] [,2] [,3]
## [1,]   34   61   54
## [2,]   30   57   54
## [3,]   37   70   60
```

(d) Denote the transpose of a matrix M by M^t. Verify that

$$(\texttt{item1}\star\texttt{item2})^t = (\texttt{item2})^t \star (\texttt{item1})^t$$

where $\star$ denotes standard matrix multiplication, and verify this relation also for the product of the matrices in the reverse order

```
#Checks if both sides are equal and that the given formula is correct.
all.equal(t(hwlist$item1 %*% hwlist$item2), t(hwlist$item2) %*% t(hwlist$item1))
```

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

```
#same check for reverse order
all.equal(t(hwlist$item2 %*% hwlist$item1), t(hwlist$item1) %*% t(hwlist$item2))
```

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

(e) The matrix `item4` has dimension 3×3. Add an identity matrix of dimension 3 to `item4` and store it in the same position.

```
#adds a 3x3 identity matrix to item 4.
hwlist$item4 <- hwlist$item4 + diag(3)
```

(f) Solve the system of equations `item4`$\star x =$ `item3`. Verify that you have obtained the correct solution.

```
#solves item4 * x = item3 (skeleton Ax = b)
x <- solve(hwlist$item4, hwlist$item3)
#verifies correct solution, while using the matrix as vector
all.equal(as.vector(hwlist$item4 %*% x), hwlist$item3)
```

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

(g) Find the inverse of `item4` and call it `item4_inv`. Verify that `item4_inv` is indeed the inverse of `item4`, and that multiplying `item4_inv` $\star$ `item3` gives the solution to the system of equations in (f). 2

```
#get the inverse given that A times its inverse gives the identity of that size.
hwlist$item4_inv <- solve(hwlist$item4, diag(3))
#Check if the inverse is correct
all.equal(hwlist$item4 %*% hwlist$item4_inv, diag(3))
```

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