HW3

2024-09-22

Homework 3 – Due 28 September 2024

The total points on this homework is 175. Five points are reserved in total for clarity of presentation, punctuation and commenting with respect to the code.

1.

Consider the dataset available in the Excel file at **wind.xls** which contains measurements on wind direction taken at Gorleston, England between 11:00 am and noon on Sundays in the year 1968 (Measurements were not recorded for two Sundays). Note that the data are in angular measurements, and also that the file is in Microsoft Excel format. Therefore, we will need for a way to read the file in a different format.

(a)

The R package readxl1is one providing functionality to read in MS Excel files. Install (if needed) and load the library. Then, read in the file, and assign to a dataframe. *5 points*

```
require(readxl)
```

```
## Loading required package: readxl
```

```
windDf <- as.data.frame(readxl::read_excel("C:/Users/samue/OneDrive/Desktop/Iowa_State_PS/STAT 5790/PS/
```

(b)

Provide descriptive summaries of the measurements such as means, standard deviations, medians, quartiles and inter-quartile ranges. *10 points*

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
seasonsSummary <- sapply(windDf, summary)

iqrSeason <- function(season) {
    quartile1 <- quantile(windDf[[season]], 0.75)
    quartile2 <- quantile(windDf[[season]], 0.25)
    iqr <- quartile1 - quartile2
    iqr[[1]]
}

iqrVals <- sapply(c("Spring", "Summer", "Autumn", "Winter"),
    iqrSeason)
endDf <- rbind.data.frame(seasonsSummary, iqrVals)
rownames(endDf)[rownames(endDf) == "7"] <- "IQR"

endDf
```

```
##           Spring    Summer Autumn   Winter
## Min.      0.0000  10.00000     30  50.0000
## 1st Qu.  55.0000  20.00000    155 205.0000
## Median  185.0000  35.00000    215 255.0000
## Mean    176.6667  80.83333    200 238.3333
## 3rd Qu. 275.0000 150.00000    260 297.5000
## Max.    350.0000 190.00000    350 340.0000
## IQR     220.0000 130.00000    105  92.5000
```

(c)

Given that these are angular data, do any of these descriptive measures above make sense? Why/why not? Think about the average between 1 $\circ$ and 359 $\circ$. *5 points*

The above summary statistics do make sense, but they require a certain degree of understanding or transformation in order to be relevant for a particular research question. For example: Knowing angular measurements are between 1 and 359 does not give us an idea of **angular dispersion** or **circular mean**, which tend to be used when studying angular measurements. So transforming these using a triometric method would likely aid in interpretability. But I would argue that we have summary statistics of "raw measurements", there is some value to the above table, insomuch as it tells us that the angular measurements between seasons are different. However, we cannot surmise much beyond that without undergoing some transformations, either to a particular statistic or to the underlying data.

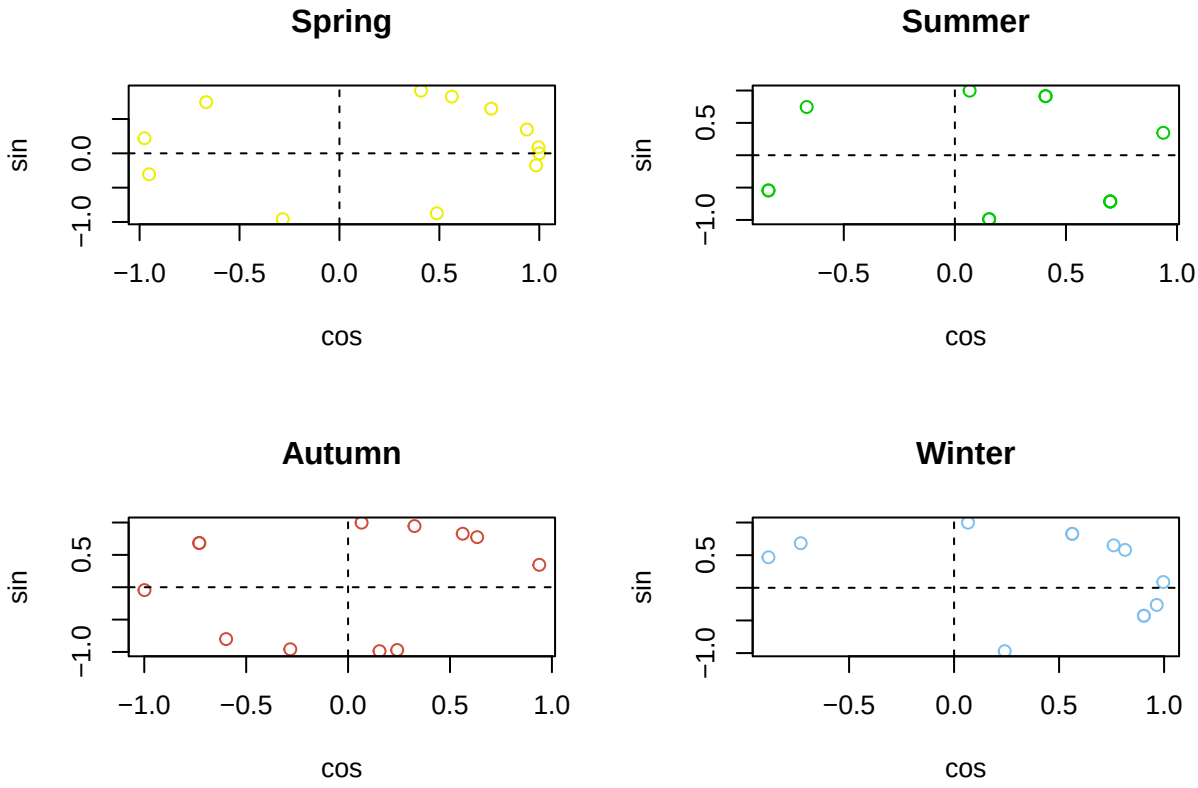
(d)

Plot, in one figure, the angular measures, using color for the season. (Note that to obtain a meaningful plot, we need to display angle in terms of a bivariate plot. One way to do so is to use a bivariate direction vector given by (

$$cos(\theta), sin(\theta)$$

) for each angle.) Comment on seasonal differences, if any. *10 points*

```r
par(mfrow = c(2, 2))
plot(sin(windDf$Spring) ~ cos(windDf$Spring), xlab = "cos", ylab = "sin",
    main = "Spring", col = "yellow2")
abline(v = 0, lty = "dashed")
abline(h = 0, lty = "dashed")
plot(sin(windDf$Summer) ~ cos(windDf$Summer), xlab = "cos", ylab = "sin",
    main = "Summer", col = "green3")
abline(v = 0, lty = "dashed")
abline(h = 0, lty = "dashed")
plot(sin(windDf$Autumn) ~ cos(windDf$Autumn), xlab = "cos", ylab = "sin",
    main = "Autumn", col = "tomato3")
abline(v = 0, lty = "dashed")
abline(h = 0, lty = "dashed")
plot(sin(windDf$Winter) ~ cos(windDf$Winter), xlab = "cos", ylab = "sin",
    main = "Winter", col = "skyblue2")
abline(v = 0, lty = "dashed")
abline(h = 0, lty = "dashed")
```



Observation: The distributions of directions between seasons appear similar upon initial inspection, though some differences become apparent. In particular, if we look at the counts of points in each quadrant of the plot, e.g. upper right quadrant of points between $cos(\theta) \in (0, 1)$ and $sin(\theta) \in (0, 1)$, Autumn and Winter have 5 observations, whereas Summer has only 3 and Spring has at least 6.

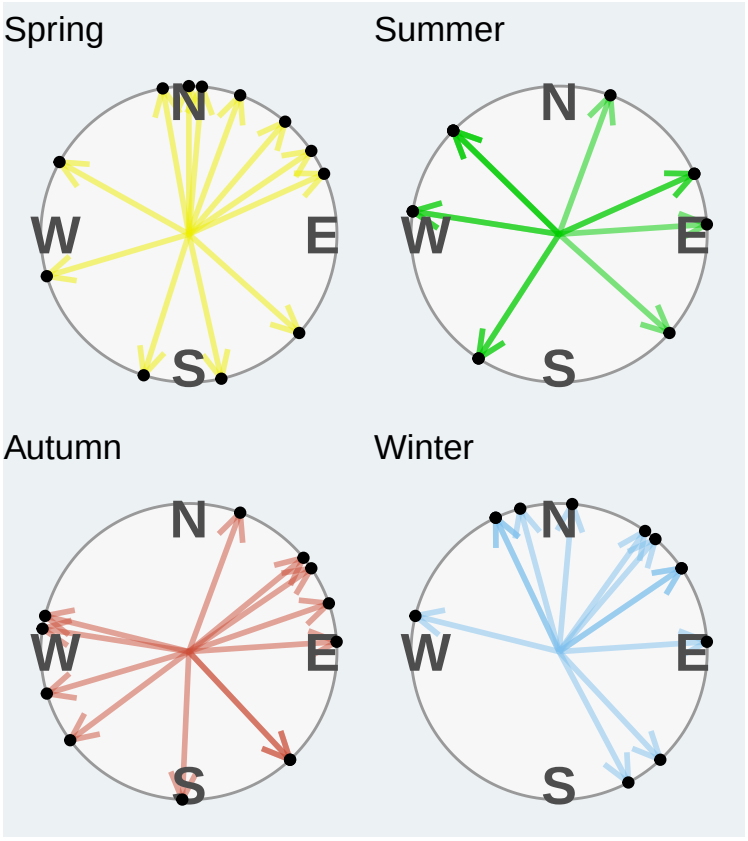
Using an alternative formulation, noted below, we can observe these differences more easily, and see there are marked differences between the seasons.

```
library(ggplot2)
library(patchwork)

x1 <- ggplot(data.frame(a = windDf$Spring%%(2 * pi))) + annotate("rect",
    xmin = -Inf, xmax = Inf, ymin = 0, ymax = 1, fill = "gray97") +
    geom_hline(yintercept = 1, color = "gray60") + geom_segment(aes(x = a,
    xend = a, y = 0, yend = 1), color = "yellow2", size = 1,
    alpha = 0.5, arrow = arrow(angle = 25, length = unit(4, "mm"))) +
    annotate("text", x = 0:3 * pi/2, y = 0.9, label = c("N",
        "E", "S", "W"), size = 7, fontface = 2, color = "gray30") +
    geom_point(aes(x = a, y = 1)) + scale_x_continuous(limits = c(0,
    2 * pi)) + coord_polar() + theme_void() + theme(plot.background = element_rect(color = NA,
    fill = "#ecf1f4")) + ggtitle("Spring")
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

```
x2 <- ggplot(data.frame(a = windDf$Summer%%(2 * pi))) + annotate("rect",
    xmin = -Inf, xmax = Inf, ymin = 0, ymax = 1, fill = "gray97") +
    geom_hline(yintercept = 1, color = "gray60") + geom_segment(aes(x = a,
    xend = a, y = 0, yend = 1), color = "green3", size = 1, alpha = 0.5,
    arrow = arrow(angle = 25, length = unit(4, "mm"))) + annotate("text",
    x = 0:3 * pi/2, y = 0.9, label = c("N", "E", "S", "W"), size = 7,
    fontface = 2, color = "gray30") + geom_point(aes(x = a, y = 1)) +
    scale_x_continuous(limits = c(0, 2 * pi)) + coord_polar() +
    theme_void() + theme(plot.background = element_rect(color = NA,
    fill = "#ecf1f4")) + ggtitle("Summer")

x3 <- ggplot(data.frame(a = windDf$Autumn%%(2 * pi))) + annotate("rect",
    xmin = -Inf, xmax = Inf, ymin = 0, ymax = 1, fill = "gray97") +
    geom_hline(yintercept = 1, color = "gray60") + geom_segment(aes(x = a,
    xend = a, y = 0, yend = 1), color = "tomato3", size = 1,
    alpha = 0.5, arrow = arrow(angle = 25, length = unit(4, "mm"))) +
    annotate("text", x = 0:3 * pi/2, y = 0.9, label = c("N",
        "E", "S", "W"), size = 7, fontface = 2, color = "gray30") +
    geom_point(aes(x = a, y = 1)) + scale_x_continuous(limits = c(0,
    2 * pi)) + coord_polar() + theme_void() + theme(plot.background = element_rect(color = NA,
    fill = "#ecf1f4")) + ggtitle("Autumn")

x4 <- ggplot(data.frame(a = windDf$Winter%%(2 * pi))) + annotate("rect",
    xmin = -Inf, xmax = Inf, ymin = 0, ymax = 1, fill = "gray97") +
    geom_hline(yintercept = 1, color = "gray60") + geom_segment(aes(x = a,
    xend = a, y = 0, yend = 1), color = "skyblue2", size = 1,
    alpha = 0.5, arrow = arrow(angle = 25, length = unit(4, "mm"))) +
    annotate("text", x = 0:3 * pi/2, y = 0.9, label = c("N",
        "E", "S", "W"), size = 7, fontface = 2, color = "gray30") +
    geom_point(aes(x = a, y = 1)) + scale_x_continuous(limits = c(0,
    2 * pi)) + coord_polar() + theme_void() + theme(plot.background = element_rect(color = NA,
    fill = "#ecf1f4")) + ggtitle("Winter")
```

```
x1 + x2 + x3 + x4
```



2.

The Central Limit Theorem (CLT). CLT is considered to be one of the most important results in statistical theory. It states that means of an arbitrary finite distribution are always distributed according to a normal distribution, provided that the sample size, n, for calculating the mean is large enough. To see how big n needs to be we can use the following simulation idea:

(a)

Generate $m = 1000$ samples of size $n = 2$ from a $Uniform(0, 1)$ distribution and storage the samples in a matrix of dimensions 2×1000. (Hint: runif() generates values from a random uniform distribution between 0 and 1.) *6 points*

```
seeds <- 1:1000  # 10 seeds
count <- 0
UniformNums <- replicate(1000, {
    count <<- count + 1
    set.seed(seeds[count])
    runif(2, min = 0, max = 1)
})
matrixDf <- matrix(UniformNums, nrow = 2, ncol = 1000)
dim(matrixDf)
```

```
## [1]    2 1000
```

(b)

Calculate the mean X {Some Math my Knitr Didn't Like} for each sample. (Hint: consider using matrix operations.) *6 points*

```
# Overall
top <- sum(matrixDf)
bottom <- length(matrixDf)
top/bottom
```

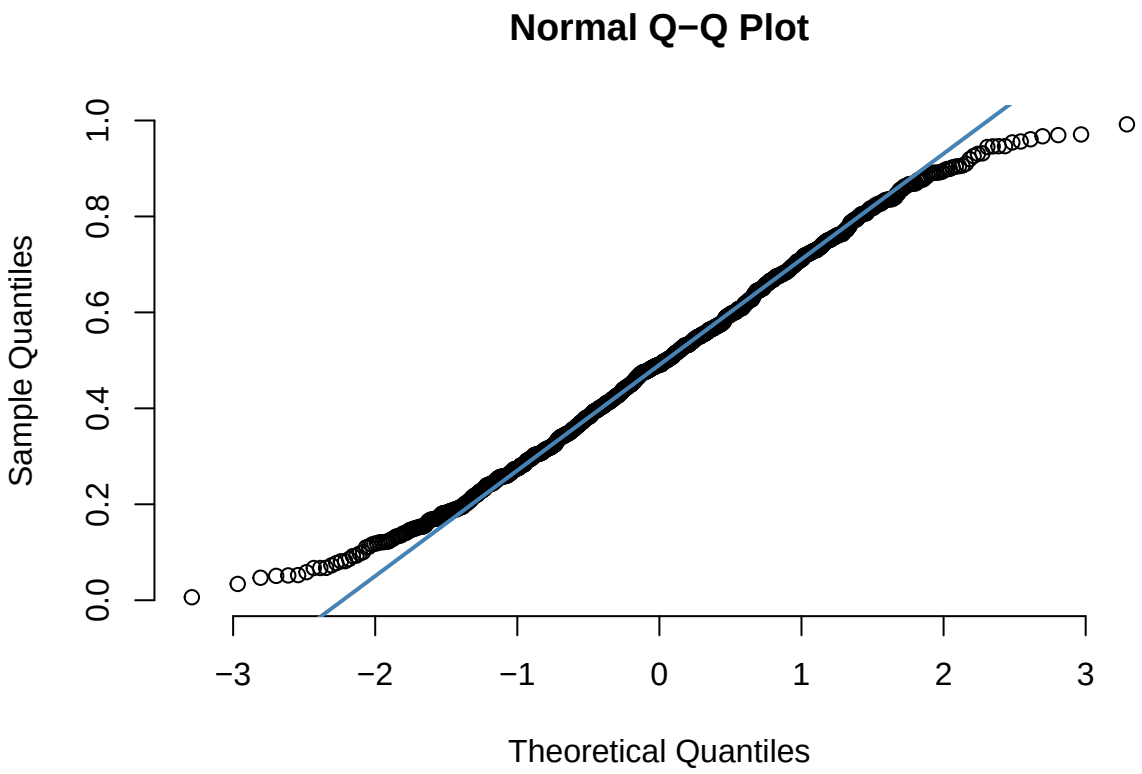
```
## [1] 0.4930478
```

```
# By Sample By column
XsDf <- colMeans(matrixDf)

# colMeans(t(matrixDf))
```

(c)

Draw a QQ-plot for the 1000 Xs to judge the normality. Comment. *5 points*

```
qqnorm(XsDf, pch = 1, frame = FALSE)
qqline(XsDf, col = "steelblue", lwd = 2)
```



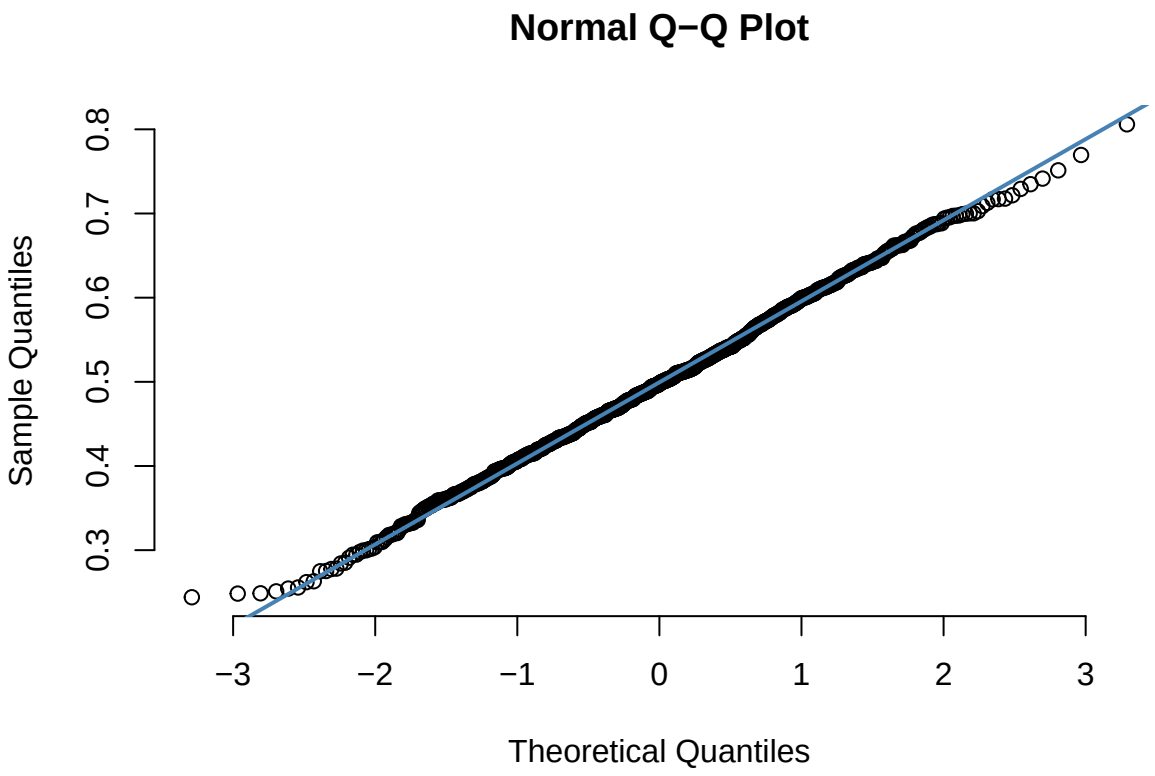

```
# hist(XsDf)
```

We generally have a good fit of normality, but the tails deviate a bit from what we'd empirically/theoretically expect, as in we appear to have the opposite of a heavy tail/potentially negative kurtosis.

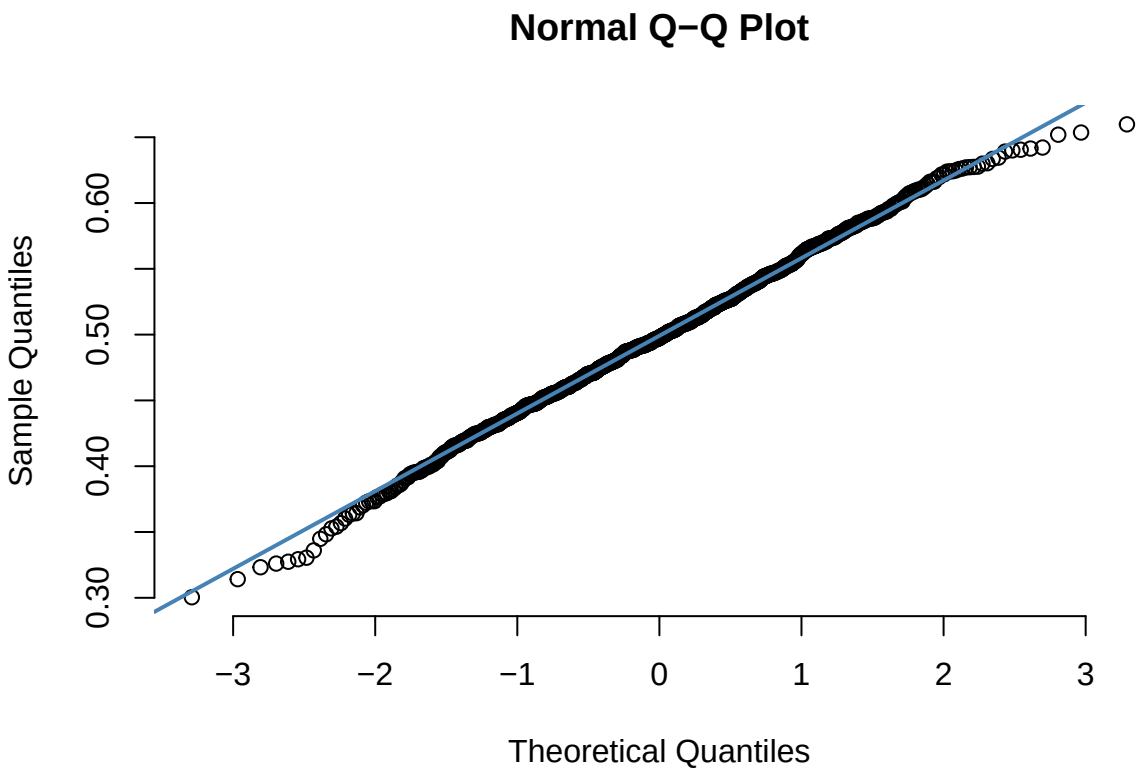
(d)

Repeat the procedure ((a),(b) and (c)) for $n = 10$, 25, and 100 with $m = 1000$. Turn in the QQ-plots and the R code. *10 points*

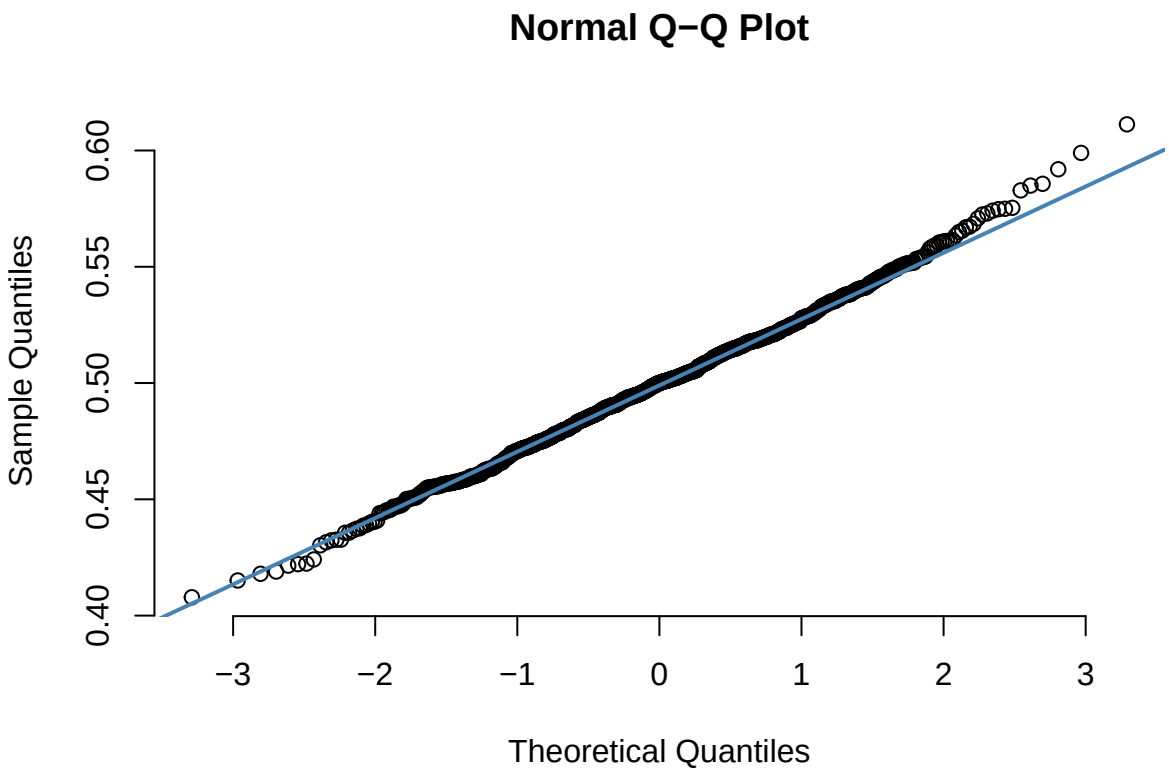
```
seeds <- 1:1000  # 10 seeds
count <- 0
UniformNums <- replicate(1000, {
    count <<- count + 1
    set.seed(seeds[count])
    runif(10, min = 0, max = 1)
})
matrixDf <- matrix(UniformNums, nrow = 10, ncol = 1000)

XsDf <- colMeans(matrixDf)

qqnorm(XsDf, pch = 1, frame = FALSE)
qqline(XsDf, col = "steelblue", lwd = 2)
```




```r
seeds <- 1:1000  # 10 seeds
count <- 0
UniformNums <- replicate(1000, {
    count <<- count + 1
    set.seed(seeds[count])
    runif(25, min = 0, max = 1)
})
matrixDf <- matrix(UniformNums, nrow = 25, ncol = 1000)

XsDf <- colMeans(matrixDf)

qqnorm(XsDf, pch = 1, frame = FALSE)
qqline(XsDf, col = "steelblue", lwd = 2)
```




```r
seeds <- 1:1000  # 10 seeds
count <- 0
UniformNums <- replicate(1000, {
    count <<- count + 1
    set.seed(seeds[count])
    runif(100, min = 0, max = 1)
})
matrixDf <- matrix(UniformNums, nrow = 100, ncol = 1000)

XsDf <- colMeans(matrixDf)

qqnorm(XsDf, pch = 1, frame = FALSE)
qqline(XsDf, col = "steelblue", lwd = 2)
```



(e)

What conclusions can you draw? *3 points*

"Boy, R sure can do some fast simulations!"

Pithy comments aside, the above plots don't look very different from one another. To me, this indicates two things: The first being that having a large m may be more of a contributing factor to achieving normality in the observations, which in turn means that we achieve something "sufficiently normal" for relatively small n. Contrapositively, this to me indicates that we will continue to see (or at least expect to see) fairly normal observations nonetheless contain some possible deivations or lack of a perfect fit even for larger and larger n.

3.

Do you ride a bicycle? (If not, you should seriously consider it.) Bike-sharing is the idea that you can rent a bike at one station and ride it to another where you drop it off. Users are charged by the amount of blocks of time that they have the bike. The data set **bikes.csv** in the Datasets section of Canvas contain information about a bike sharing service in Washington DC. Each row in the dataset is a record on one rental/trip.

For all the answers, provide the R code necessary for a complete solution.

```
knitr::include_graphics("memes.png")
```

(a)

Load the data into R. How many trips were there overall?. How many factor variables are in the data, how many variables are of other kinds? *4 points*

```
library(readr)
bikes <- read_csv("C:/Users/samue/OneDrive/Desktop/Iowa_State_PS/STAT 5790/PS/PS3/bikes.csv")
```

```
## Rows: 77186 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): Start.date, wday, Start.Station, End.Station, Subscriber.Type, Bike.
## dbl (2): Duration, hour
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
nrow(bikes)
```

```
## [1] 77186
```

```
summary(bikes)
```

```
##     Duration       Start.date            wday                hour
##  Min.   :     0   Length:77186       Length:77186       Min.   : 0.00
##  1st Qu.:   420   Class :character   Class :character   1st Qu.:10.00
##  Median :   720   Mode  :character   Mode  :character   Median :15.00
##  Mean   :  1094                                        Mean   :14.28
##  3rd Qu.:  1140                                        3rd Qu.:18.00
##  Max.   :775560                                        Max.   :23.00
##  Start.Station      End.Station        Subscriber.Type       Bike.
##  Length:77186       Length:77186       Length:77186       Length:77186
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
```

There were 77,186 trips contained in the bikes dataset.

There appear to be 6 Factor Variables in the bikes dataset: - wday - hour - Start.Station - End.Station - Subscriber.Type - Bike.

And there appears to be 2 non-Factor Variables in the bikes dataset: - Duration - Start.date

(b)

The variable Duration contains the length of the bike rental in seconds.

i

How long was the longest rental (converted into days)?. *2 points*

```
# 86400 seconds in a day
max(bikes$Duration)/86400
```

```
## [1] 8.976389
```

8.976389 days, or roughly speaking: 8 days, 23 hours, 26 minutes long.

ii

What other information is available in the data on this trip? *5 points*

```
bikes |>
    filter(Duration == max(Duration))
```

```
## # A tibble: 1 x 8
##   Duration Start.date     wday   hour Start.Station  End.Station Subscriber.Type
##      <dbl> <chr>          <chr> <dbl> <chr>          <chr>       <chr>
## 1   775560 6/4/2014 10:19 Wed      10 Metro Center ~ 3rd & H St~ Casual
## # i 1 more variable: Bike. <chr>
```

This trip started at 10am on a Wednesday, using a W21208 while beginning at the Metro Center, and ended at a different station (3rd & H St NW). This was also a "Casual" subscriber to the service.

iii

How many trips (of all trips) lasted more than one day? *2 points*

```
# 86400 seconds in a day
oneDay <- 86400
bikes |>
    filter(Duration > 86400) |>
    nrow()
```

```
## [1] 7
```

7 trips lasted longer than one day.

(c)

Start.Station describes the start of each trip.

i

From which station did most trips originate? How many trips? *4 points*

```
bikes |>
    group_by(Start.Station) |>
    summarize(count = n()) |>
    arrange(desc(count)) |>
    head()
```

```
## # A tibble: 6 x 2
##   Start.Station                          count
##   <chr>                                  <int>
## 1 Massachusetts Ave & Dupont Circle NW  1595
## 2 Columbus Circle / Union Station       1587
## 3 Lincoln Memorial                      1541
## 4 Jefferson Dr & 14th St SW             1312
## 5 15th & P St NW                        1150
## 6 Thomas Circle                         1073
```

Massachusetts Ave & Dupont Circle NW has most trips originate from it.

ii

Is that the same station at which most trips ended (End.Station)? *3 points*

```
bikes |>
    group_by(End.Station) |>
    summarize(count = n()) |>
    arrange(desc(count)) |>
    head()
```

```
## # A tibble: 6 x 2
##   End.Station                            count
##   <chr>                                  <int>
## 1 Massachusetts Ave & Dupont Circle NW  1736
## 2 Columbus Circle / Union Station       1645
## 3 Lincoln Memorial                      1506
## 4 Jefferson Dr & 14th St SW             1331
## 5 15th & P St NW                        1255
## 6 Thomas Circle                         1063
```

Massachusetts Ave & Dupont Circle NW also was the most frequent end station, so yes, it is both the most common start and stop location.

(d)

When a bike is not returned, the End.Station is marked as "". How often do bikes not get returned? What is reported for the duration of those trips? Change the value of Duration to NA for these records. *4 points*

```
bikes$Duration[is.na(bikes$End.Station)] <- NA

bikes |>
    filter(is.na(End.Station))
```

```
## # A tibble: 1 x 8
##   Duration Start.date     wday   hour Start.Station  End.Station Subscriber.Type
##      <dbl> <chr>          <chr> <dbl> <chr>          <chr>       <chr>
## 1       NA 6/4/2014 18:47 Wed      18 Georgia & New~ <NA>        Registered
## # i 1 more variable: Bike. <chr>
```

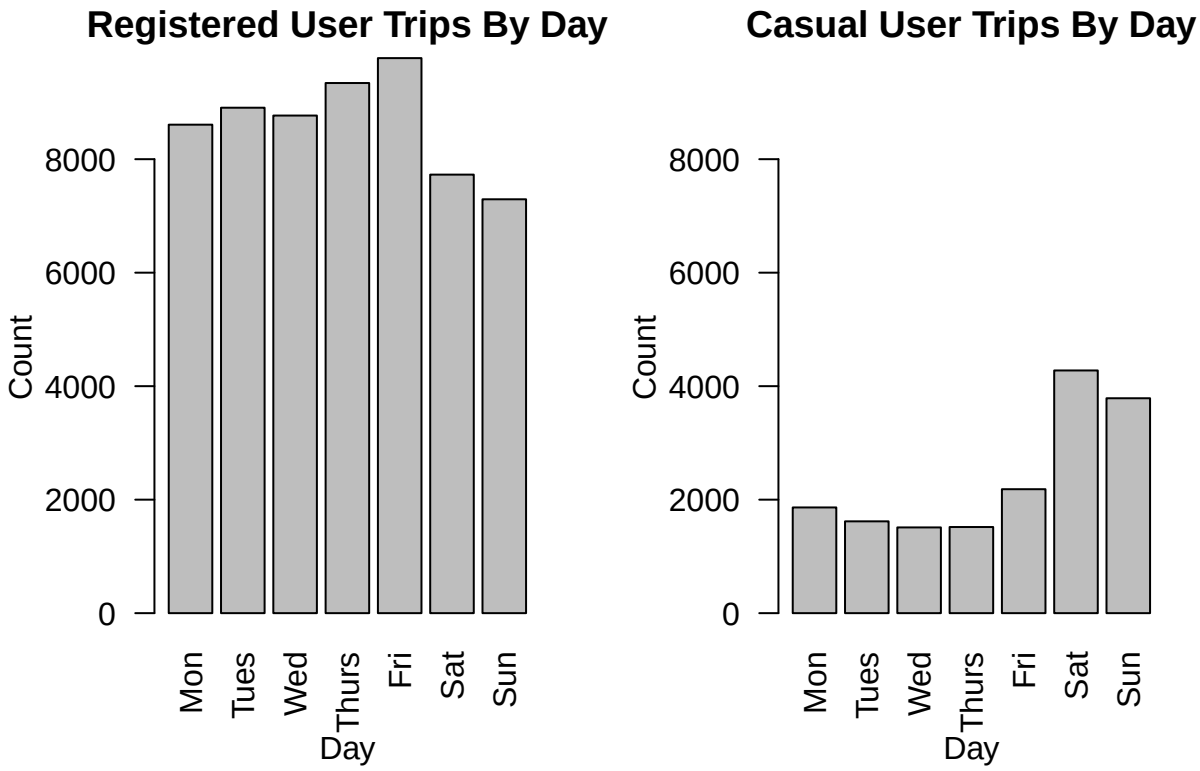
(e)

Plot barcharts of the number of trips on each day of the week, for each Subscriber.Type. Note that one can divide the plotting area using par(mfrow = c(....)) if needed. Make sure that the days of the week (wday) are in the usual order (Start with Mondays). Describe any patterns you see. *6 points*

```
dailySummaries <- bikes |>
    group_by(wday, Subscriber.Type) |>
    summarize(count = n())
```

```
## `summarise()` has grouped output by 'wday'. You can override using the
## `.groups` argument.
```

```
subs <- dailySummaries |>
    filter(Subscriber.Type == "Registered")

casuals <- dailySummaries |>
    filter(Subscriber.Type == "Casual")

days_of_the_week <- c("Mon", "Tues", "Wed", "Thurs", "Fri", "Sat",
    "Sun")
par(mfrow = c(1, 2))

barplot(subs$count ~ factor(subs$wday, days_of_the_week), main = "Registered User Trips By Day",
    ylim = c(0, 9000), xlab = "Day", ylab = "Count", las = 2)
barplot(casuals$count ~ factor(casuals$wday, days_of_the_week),
    main = "Casual User Trips By Day", ylim = c(0, 9000), xlab = "Day",
    ylab = "Count", las = 2)
```



Observations: Between both subscriber types, most Casual subscriber usage spikes during the weekend (Saturday-Sunday), whereas Registered subscriber usage decreases during the weekend. Also, perhaps unsurprisingly, the population of Registered subscribers goes on a trip more often than casual users regardless of the day of the week.

4.

The Titanic. The dataset, available as a comma-separated file on the WWW at **titanic.txt** provides the survival status of passengers on the Titanic, together with their names, age, sex and passenger class. (Note that a good portion of the ages for the 3rd Class passengers is missing.)

Variable	Description
Name	Recorded name of passenger\
PClass	Passenger class: 1st, 2nd or 3rd\
Age	Age in years\
Sex	male or female\
Survived	1 = Yes, 0 = No

(a)

Read in the dataset, using R. Note that the fields are comma-delimited. *3 points*

```
titantic <- read.table(file = "C:/Users/samue/OneDrive/Desktop/Iowa_State_PS/STAT 5790/PS/PS3/titanic.t
    sep = ",", header = TRUE)
```

(b)

Using for instance the function table, cross-classify the passengers by gender and passenger class. Do a further cross-tabulation, perhaps using the same function additionally stratified by survival status. Comment on your findings. *3 + 3 + 3 points*

```
table(titantic$PClass, titantic$Sex)
```

```
##
##       female male
##   1st    143  179
##   2nd    107  173
##   3rd    212  499
```

```
table(titantic$PClass, titantic$Sex, titantic$Survived)
```

```
## , ,  = 0
##
##
##       female male
##   1st      9  120
##   2nd     13  148
##   3rd    132  441
##
## , ,  = 1
##
##
##       female male
##   1st    134   59
##   2nd     94   25
##   3rd     80   58
```

Females in 1st class tended to survive than not, and likewise for females in the 2nd class. However, this trend is inverted for 3rd class females, which is to say more 3rd class females did not survive than survive. Looking at men, we observe more men tended not to survive than survive regardless of class, though the proportion of 1st class men who survived is greater than the proportion of 3rd class men who didn't survive.

(c)

Is there any preliminary evidence that there is a difference in ages among people who survived and those that did not? To answer this question, calculate the mean difference in ages (separately, for both men and women) and the standard errors of the means. What assumptions do you need to make in order to answer this question? *(2 + 3 + 2)× 2 + 4 points*

```
gone <- titantic |>
    filter(Survived == 0) |>
    na.omit() |>
    group_by(Sex) |>
    summarize(ageAvg = mean(Age), stdErrAge = sd(Age)/sqrt(length((Age))),
        nObs = n())

survivor <- titantic |>
    filter(Survived == 1) |>
    na.omit() |>
    group_by(Sex) |>
    summarize(ageAvg = mean(Age), stdErrAge = sd(Age)/sqrt(length((Age))),
        nObs = n())

gone <- cbind.data.frame("Survived == 0", gone)
survivor <- cbind.data.frame("Survived == 1", survivor)

gone
```

```
##   "Survived == 0"    Sex   ageAvg stdErrAge nObs
## 1   Survived == 0 female 24.90141  1.548272   71
## 2   Survived == 0   male 32.32078  0.684307  372
```

```
survivor
```

```
##   "Survived == 1"    Sex   ageAvg stdErrAge nObs
## 1   Survived == 1 female 30.86714  1.019990  217
## 2   Survived == 1   male 25.95188  1.578879   96
```

Preliminary evidence indicates that women who survived tended to be older on average, and that men who survived tended to be younger on average.

An assumption one should make when supporting the validity of this claim is that not other variables could possibly upend this presumption. In statistical terms, we suppose that observations are independent between groups, i.e. between men and women, and that each of the variables in question is independent of some other variable. For example, one would need to suppose that no other factors could have caused this phenomenon to occur, which is rather difficult to support given past analysis and general consensus over the fact that people in the upper class were more likely to survive compared to lower class. Combine this information with a potential demographic skew of, say, older women tending to belong to the upper class of passengers, and we start to suspect some other factors playing a role in survival.

5.

This problem is a short exercise to get you into manipulating matrices, columns, entries, and some preliminary approaches when dealing with text data (character strings). The objective is to understand how to use available tools to perform our analysis.

The 109th US Congress, comprising the Senate and the House of Representatives, was the legislative branch of the US government from January 3, 2005 to January 3, 2007. During this period, 542 bills were voted on by the US Senate. Each of 100 Senators either voted in favor or against or failed to record their vote on each of these bills. Details voting preferences of the 100 senators on these bills is provided in the file available on Canvas in **senate-109.txt**.

(a)

Read in the file, *noting that the fields are tab-delimited.* Also, *there are apostrophe quotes in some of the field names.* The first column of the file contains the name of the bill and its type, the second column contains the number of missing votes for each bill and the remaining 100 columns contain the votes of each senator (a vote in favor = 1, a vote against = -1, and a no vote = 0). *5 points*

```
senateVote <- read.table(file = "C:/Users/samue/OneDrive/Desktop/Iowa_State_PS/STAT 5790/PS/PS3/senate-
    sep = "\t", quote = "\"", header = TRUE)
```

(b)

Bill type. The field **bill_type_bill_name_bill_ID** contains details on the bills. (Make sure that this field is a vector of character strings.) The first part of this field (before the " ") contains the type of the bill. We will now proceed with obtaining the bill type only, and in doing so, perform a series of operations on character strings.

i

Our objective is to take the above vector of character strings, and for each element, to only keep the portion that contains the string preceding the first " ". There are a few ways to do this, but you can use the function sub() and its allies (see ?sub for examples) which replaces the first time a desired string matches in each element with our choice. Use this to create a new vector of character strings containing only the bill type. *10 points*

```
require(dplyr)
TypeOnlyDf <- senateVote |>
  mutate(
    bill_type_bill_name_bill_ID = gsub("\\_.*","",bill_type_bill_name_bill_ID)
    # the below will extract using a different method
    # and create a new column
    # BillTypeOnly = str_extract(senateVote$bill_type_bill_name_bill_ID, "^[^_]+(?=_)")
  )
```

ii

Tabulate the frequency for each type of bill. *5 points*

```
TypeOnlyDf |>
    group_by(bill_type_bill_name_bill_ID) |>
    summarize(count = n()) |>
    arrange(desc(count)) |>
    mutate(prop = paste(round(count/sum(count), 3) * 100, "%"))
```

```
## # A tibble: 36 x 3
##    bill_type_bill_name_bill_ID   count prop
##    <chr>                        <int> <chr>
##  1 Appropriations                 111 20.5 %
##  2 Budget, Spending and Taxes      75 13.8 %
##  3 Executive Branch                44 8.1 %
##  4 Defense                         29 5.4 %
##  5 Health Issues                   22 4.1 %
##  6 Immigration                     22 4.1 %
##  7 Abortion Issues                 19 3.5 %
##  8 Foreign Aid and Policy Issues   19 3.5 %
##  9 Energy Issues                   18 3.3 %
## 10 Education                       16 3 %
## # i 26 more rows
```

(c)

Data Quality. The second field contains the number of votes which were not recorded for each senator. We will evaluate if there is any discrepancy. To do so, note that if $\boldsymbol{X}$ is the matrix of votes (only), then the diagonal elements of $\boldsymbol{XX}'$ plus the column of missing votes should match the total number of senators. (Note that there is an "easier"way to do this, using the function apply(), but you are not asked to try that here, since we will be encountering this in detail later.) *5 points*

```
ncol(TypeOnlyDf) - 2
```

```
## [1] 100
```

We expect 100 senators in our dataset.

```
missingMatrix <- matrix(TypeOnlyDf$missing_votes, ncol = 1)
tMissing <- t(missingMatrix)
diag(missingMatrix %*% tMissing) + missingMatrix
```

```
##        [,1]
##   [1,]    0
##   [2,]    0
##   [3,]    0
##   [4,]    0
##   [5,]    0
##   [6,]    0
##   [7,]    0
##   [8,]    0
##   [9,]    0
##  [10,]    0
##  [11,]    0
```

```
##  [12,]    0
##  [13,]    0
##  [14,]    0
##  [15,]    0
##  [16,]    2
##  [17,]    2
##  [18,]    2
##  [19,]    2
##  [20,]    2
##  [21,]    2
##  [22,]    2
##  [23,]    2
##  [24,]    2
##  [25,]    2
##  [26,]    2
##  [27,]    2
##  [28,]    2
##  [29,]    2
##  [30,]    2
##  [31,]    2
##  [32,]    2
##  [33,]    2
##  [34,]    2
##  [35,]    2
##  [36,]    2
##  [37,]    2
##  [38,]    2
##  [39,]    2
##  [40,]    2
##  [41,]    2
##  [42,]    2
##  [43,]    2
##  [44,]    2
##  [45,]    2
##  [46,]    2
##  [47,]    2
##  [48,]    2
##  [49,]    2
##  [50,]    2
##  [51,]    2
##  [52,]    2
##  [53,]    2
##  [54,]    2
##  [55,]    2
##  [56,]    2
##  [57,]    2
##  [58,]    2
##  [59,]    2
##  [60,]    2
##  [61,]    2
##  [62,]    2
##  [63,]    2
##  [64,]    2
##  [65,]    2
```

```
##  [66,]    2
##  [67,]    6
##  [68,]    6
##  [69,]    6
##  [70,]    6
##  [71,]    6
##  [72,]    6
##  [73,]    6
##  [74,]    6
##  [75,]    6
##  [76,]    6
##  [77,]    6
##  [78,]    6
##  [79,]    6
##  [80,]    6
##  [81,]    6
##  [82,]    6
##  [83,]    6
##  [84,]    6
##  [85,]    6
##  [86,]    6
##  [87,]    6
##  [88,]    6
##  [89,]    6
##  [90,]    6
##  [91,]    6
##  [92,]    6
##  [93,]    6
##  [94,]    6
##  [95,]    6
##  [96,]    6
##  [97,]    6
##  [98,]    6
##  [99,]    6
## [100,]    6
## [101,]    6
## [102,]    6
## [103,]    6
## [104,]    6
## [105,]    6
## [106,]   12
## [107,]   12
## [108,]   12
## [109,]   12
## [110,]   12
## [111,]   12
## [112,]   12
## [113,]   12
## [114,]   12
## [115,]   12
## [116,]   12
## [117,]   12
## [118,]   12
## [119,]   12
```

```
## [120,]   12
## [121,]   12
## [122,]   12
## [123,]   12
## [124,]   12
## [125,]   12
## [126,]   12
## [127,]   12
## [128,]   12
## [129,]   12
## [130,]   12
## [131,]   12
## [132,]   12
## [133,]   12
## [134,]   12
## [135,]   20
## [136,]   20
## [137,]   20
## [138,]   20
## [139,]   20
## [140,]   20
## [141,]   20
## [142,]   30
## [143,]   30
## [144,]   30
## [145,]   30
## [146,]   30
## [147,]   30
## [148,]   30
## [149,]   30
## [150,]   42
## [151,]   42
## [152,]   42
## [153,]   42
## [154,]   42
## [155,]   42
## [156,]   42
## [157,]   56
## [158,]   56
## [159,]   56
## [160,]   56
## [161,]   90
## [162,]   90
## [163,]   90
## [164,]   90
## [165,]  110
## [166,]  110
## [167,]  110
## [168,]  110
## [169,]  110
## [170,]  110
## [171,]  110
## [172,]  132
## [173,]  132
```

```
## [174,]  132
## [175,]  132
## [176,]  132
## [177,]  132
## [178,]  132
## [179,]  132
## [180,]  132
## [181,]  132
## [182,]  156
## [183,]  156
## [184,]  156
## [185,]  156
## [186,]  156
## [187,]  156
## [188,]  156
## [189,]  156
## [190,]  156
## [191,]  156
## [192,]  156
## [193,]  156
## [194,]  156
## [195,]  156
## [196,]  182
## [197,]  182
## [198,]  182
## [199,]  182
## [200,]  182
## [201,]  182
## [202,]  182
## [203,]  182
## [204,]  182
## [205,]  182
## [206,]  182
## [207,]  182
## [208,]  182
## [209,]  210
## [210,]  210
## [211,]  210
## [212,]  210
## [213,]  210
## [214,]  210
## [215,]  210
## [216,]  240
## [217,]  240
## [218,]  272
## [219,]  272
## [220,]  306
## [221,]  380
## [222,]  462
## [223,]  462
## [224,]  462
## [225,]  462
## [226,]  462
## [227,]  462
```

```
## [228,]  462
## [229,]  462
## [230,]  462
## [231,]  462
## [232,]  462
## [233,]  462
## [234,]  462
## [235,]  462
## [236,]  462
## [237,]  462
## [238,]  462
## [239,]  462
## [240,]  462
## [241,]  462
## [242,]  462
## [243,]  462
## [244,]  462
## [245,]  462
## [246,]  462
## [247,]  462
## [248,]  506
## [249,]  506
## [250,]  506
## [251,]  506
## [252,]  506
## [253,]  506
## [254,]  506
## [255,]  506
## [256,]  506
## [257,]  506
## [258,]  506
## [259,]  552
## [260,]  552
## [261,]  552
## [262,]  552
## [263,]  552
## [264,]  552
## [265,]  552
## [266,]  552
## [267,]  600
## [268,]  600
## [269,]  600
## [270,]  600
## [271,]  600
## [272,]  600
## [273,]  600
## [274,]  650
## [275,]  650
## [276,]  650
## [277,]  650
## [278,]  870
## [279,]  870
## [280,]  870
## [281,]  870
```

```
## [282,]  870
## [283,]  870
## [284,]  930
## [285,]  930
## [286,]  930
## [287,]  930
## [288,]  930
## [289,]  930
## [290,]  930
## [291,]  930
## [292,]  930
## [293,]  930
## [294,]  930
## [295,]  930
## [296,]  930
## [297,]  930
## [298,]  930
## [299,]  930
## [300,]  930
## [301,]  930
## [302,]  930
## [303,]  992
## [304,]  992
## [305,]  992
## [306,]  992
## [307,]  992
## [308,]  992
## [309,]  992
## [310,]  992
## [311,]  992
## [312,]  992
## [313,]  992
## [314,]  992
## [315,]  992
## [316,]  992
## [317,]  992
## [318,]  992
## [319,]  992
## [320,]  992
## [321,]  992
## [322,]  992
## [323,]  992
## [324,]  992
## [325,]  992
## [326,]  992
## [327,]  992
## [328,]  992
## [329,]  992
## [330,] 1056
## [331,] 1056
## [332,] 1056
## [333,] 1056
## [334,] 1056
## [335,] 1056
```

```
## [336,] 1056
## [337,] 1056
## [338,] 1056
## [339,] 1056
## [340,] 1056
## [341,] 1056
## [342,] 1056
## [343,] 1056
## [344,] 1056
## [345,] 1122
## [346,] 1122
## [347,] 1190
## [348,] 1190
## [349,] 1190
## [350,] 1190
## [351,] 1260
## [352,] 1332
## [353,] 1332
## [354,] 1332
## [355,] 1332
## [356,] 1332
## [357,] 1332
## [358,] 1332
## [359,] 1332
## [360,] 1332
## [361,] 1332
## [362,] 1332
## [363,] 1332
## [364,] 1332
## [365,] 1332
## [366,] 1332
## [367,] 1332
## [368,] 1332
## [369,] 1332
## [370,] 1332
## [371,] 1332
## [372,] 1332
## [373,] 1332
## [374,] 1332
## [375,] 1332
## [376,] 1332
## [377,] 1332
## [378,] 1332
## [379,] 1332
## [380,] 1332
## [381,] 1332
## [382,] 1332
## [383,] 1332
## [384,] 1332
## [385,] 1332
## [386,] 1332
## [387,] 1332
## [388,] 1332
## [389,] 1332
```

```
## [390,] 1332
## [391,] 1406
## [392,] 1406
## [393,] 1406
## [394,] 1406
## [395,] 1406
## [396,] 1406
## [397,] 1406
## [398,] 1406
## [399,] 1406
## [400,] 1406
## [401,] 1406
## [402,] 1406
## [403,] 1406
## [404,] 1406
## [405,] 1406
## [406,] 1406
## [407,] 1406
## [408,] 1406
## [409,] 1406
## [410,] 1406
## [411,] 1406
## [412,] 1406
## [413,] 1406
## [414,] 1406
## [415,] 1406
## [416,] 1406
## [417,] 1406
## [418,] 1406
## [419,] 1482
## [420,] 1482
## [421,] 1482
## [422,] 1482
## [423,] 1482
## [424,] 1482
## [425,] 1482
## [426,] 1482
## [427,] 1482
## [428,] 1482
## [429,] 1482
## [430,] 1482
## [431,] 1482
## [432,] 1482
## [433,] 1482
## [434,] 1482
## [435,] 1560
## [436,] 1560
## [437,] 1560
## [438,] 1560
## [439,] 1640
## [440,] 1640
## [441,] 1640
## [442,] 1722
## [443,] 1722
```

```
## [444,] 1722
## [445,] 1722
## [446,] 1722
## [447,] 1806
## [448,] 2352
## [449,] 2352
## [450,] 2352
## [451,] 2352
## [452,] 2352
## [453,] 2352
## [454,] 2352
## [455,] 2352
## [456,] 2352
## [457,] 2352
## [458,] 2352
## [459,] 2352
## [460,] 2352
## [461,] 2352
## [462,] 2352
## [463,] 2352
## [464,] 2352
## [465,] 2352
## [466,] 2352
## [467,] 2352
## [468,] 2352
## [469,] 2352
## [470,] 2352
## [471,] 2352
## [472,] 2352
## [473,] 2352
## [474,] 2352
## [475,] 2352
## [476,] 2352
## [477,] 2450
## [478,] 2450
## [479,] 2450
## [480,] 2450
## [481,] 2450
## [482,] 2450
## [483,] 2450
## [484,] 2450
## [485,] 2450
## [486,] 2450
## [487,] 2450
## [488,] 2450
## [489,] 2450
## [490,] 2450
## [491,] 2450
## [492,] 2450
## [493,] 2450
## [494,] 2450
## [495,] 2450
## [496,] 2450
## [497,] 2450
```

```
## [498,] 2450
## [499,] 2450
## [500,] 2450
## [501,] 2450
## [502,] 2450
## [503,] 2450
## [504,] 2450
## [505,] 2450
## [506,] 2450
## [507,] 2450
## [508,] 2450
## [509,] 2450
## [510,] 2550
## [511,] 2550
## [512,] 2550
## [513,] 2550
## [514,] 2550
## [515,] 2550
## [516,] 2550
## [517,] 2550
## [518,] 2550
## [519,] 2550
## [520,] 2550
## [521,] 2550
## [522,] 2550
## [523,] 2550
## [524,] 2550
## [525,] 2550
## [526,] 2550
## [527,] 2550
## [528,] 2550
## [529,] 2550
## [530,] 2550
## [531,] 2652
## [532,] 2652
## [533,] 2652
## [534,] 2652
## [535,] 2652
## [536,] 2652
## [537,] 2652
## [538,] 2652
## [539,] 2756
## [540,] 2756
## [541,] 2756
## [542,] 2862
```

```
sum(TypeOnlyDf$missing_votes)
```

```
## [1] 12888
```

(d)

One issue here is whether we can discern voting trends on different issues. However, every vote here is recorded as a-1/0/1 vote, regardless of whether it is for/neutral/against on a conservative/moderate/liberal issue. For this reason, we will analyze the datasets according to whether senators voted with or against the (majority) leader, Senator Bill Frist (a Republican whose votes are recorded in the last field). Thus, we will convert all the votes of the other senators relative to whether they voted with or against Senator Frist. Do so, using a set of appropriate matrix and vector operations, after eliminating those Bills from consideration where the leader did not record a vote. *15 points*

```
filterDf <- TypeOnlyDf |>
    filter(William.H...Bill..Frist..TN. != 0)

fristOnly <- filterDf |>
    select(William.H...Bill..Frist..TN.) |>
    as.matrix(nrow = 539, ncol = 1)

otherSenators <- filterDf |>
    select(-c(William.H...Bill..Frist..TN., bill_type_bill_name_bill_ID,
        missing_votes)) |>
    as.matrix(nrow = 539, ncol = 99)
```

```
M <- otherSenators
v <- fristOnly
# positive entries when agreed (-1 * -1 = 1) negative if
# voted against 0 if they voted neutral

result <- M * matrix(v, nrow = nrow(M), ncol = ncol(M), byrow = TRUE)
analysisData <- cbind.data.frame(billType = filterDf$bill_type_bill_name_bill_ID,
    result)
```

(e)

For each bill type identified earlier, tabulate the average number of times senators voted with, against, or indifferently from the Senate majority leader. *10 points*

Senator Averages

```
resultsNeutralInd <- analysisData |>
    group_by(billType) |>
    # Get a summary of a senators votes by billType
summarize(across(everything(), ~sum(. == 0, na.rm = TRUE), .names = "zero_{col}"),
    ) |>
    arrange(desc(billType))


resultsForInd <- analysisData |>
    group_by(billType) |>
    summarize(across(everything(), ~sum(. == 1, na.rm = TRUE),
        .names = "for_{col}"), ) |>
    arrange(desc(billType))
```

```
resultsAgainstInd <- analysisData |>
    group_by(billType) |>
    summarize(across(everything(), ~sum(. == -1, na.rm = TRUE),
        .names = "against_{col}"), ) |>
    arrange(desc(billType))
```

```
averageCountsperType <- TypeOnlyDf |>
    group_by(bill_type_bill_name_bill_ID) |>
    summarize(count = n()) |>
    arrange(desc(bill_type_bill_name_bill_ID))

names(averageCountsperType)[1] <- "billType"

# compNeutral <- mutate(averageCountsperType,
# resultsNeutralInd ) compFor <-
# mutate(averageCountsperType, resultsForInd ) compAgainst
# <- mutate(averageCountsperType, resultsAgainstInd )
```

```
m <- as.matrix(averageCountsperType$count, nrow = 36, ncol = 1)
mSenateNeutral <- as.matrix(resultsNeutralInd[, 2:100], nrow = 36,
    ncol = 99)
mSenateFor <- as.matrix(resultsForInd[, 2:100], nrow = 36, ncol = 99)
mSenateAgainst <- as.matrix(resultsAgainstInd[, 2:100], nrow = 36,
    ncol = 99)

neutralTable <- round(sweep(mSenateNeutral, 1, m, "/"), 3)
forTable <- round(sweep(mSenateFor, 1, m, "/"), 3)
againstTable <- round(sweep(mSenateAgainst, 1, m, "/"), 3)
```

```
neutralInfo <- cbind(resultsNeutralInd$billType, neutralTable)
# neutralInfo
```

```
forInfo <- cbind(resultsForInd$billType, forTable)
# forInfo
```

```
againstInfo <- cbind(resultsAgainstInd$billType, againstTable)
# againstInfo
```

Overall Averages

```
resultsNeutral <- analysisData |>
    group_by(billType) |>
    # Get a summary of a senators votes by billType
summarize(across(everything(), ~sum(. == 0, na.rm = TRUE), .names = "zero_{col}"),
    ) |>
    # mutate so we keep billType for use later now we sum
    # the total of senators across senators
mutate(total_zero = rowSums(across(starts_with("zero_")))) |>
    select(c(billType, total_zero)) |>
    arrange(desc(billType))
```

```
resultsFor <- analysisData |>
    group_by(billType) |>
    summarize(across(everything(), ~sum(. == 1, na.rm = TRUE),
        .names = "for_{col}"), ) |>
    mutate(total_for = rowSums(across(starts_with("for_")))) |>
    select(c(billType, total_for)) |>
    arrange(desc(billType))

resultsAgainst <- analysisData |>
    group_by(billType) |>
    summarize(across(everything(), ~sum(. == -1, na.rm = TRUE),
        .names = "against_{col}"), ) |>
    mutate(total_against = rowSums(across(starts_with("against_")))) |>
    select(c(billType, total_against)) |>
    arrange(desc(billType))
```

```
combinedCounts <- mutate(resultsNeutral, resultsAgainst, resultsFor)

averageVoting <- combinedCounts |>
    mutate(totalCount = total_zero + total_against + total_for,
        averageNeutral = round(total_zero/totalCount, 3), averageFor = round(total_against/totalCount,
            3), averageAgainst = round(total_for/totalCount,
            3))

averageVoting
```

```
## # A tibble: 36 x 8
##    billType         total_zero total_against total_for totalCount averageNeutral
##    <chr>                 <dbl>         <dbl>     <dbl>      <dbl>          <dbl>
##  1 Welfare and Pov~        195            75       126        396          0.492
##  2 Veterans Issues           6           158       133        297          0.02
##  3 Transportation ~        421           261       605       1287          0.327
##  4 Trade Issues            226           425       537       1188          0.19
##  5 Technology and ~        211           126       356        693          0.304
##  6 Social Issues            58            87       152        297          0.195
##  7 Senior and Soci~         87           297       309        693          0.126
##  8 Science and Med~          0            44        55         99          0
##  9 Regulatory Issu~        139            62        96        297          0.468
## 10 National Securi~        219           492       675       1386          0.158
## # i 26 more rows
## # i 2 more variables: averageFor <dbl>, averageAgainst <dbl>
```