# HW2_Kim

*Jaiyool Kim*

*9/7/2019*

**Problem 1 : Work through the "R Programming E" lesson parts 4-7, 14**

**(optional 12 - only takes 5 min). From the R command prompt:**

```
install.packages("swirl")
library(swirl)
install_course("R_Programming_E")
swirl()
```

I have taken all of the coures you mentioned at PM 10:21 on sunday (Sep. 8th), and sent you e-mails related to the history I took.

**Problem 2 : Create a new R Markdown file within your local GitHub repo folder**

**(file–>new–>R Markdown–>save as).**

**The filename should be: HW2_lastname, i.e. for me it would be HW2_Settlage**

**You will use this new R Markdown file to solve problems 3-5.**

I made this R Markdown file whose names is HW2_Kim.

**Problem 3 : In the lecture, there were two links to StackOverflow questions on why one should use version control.**

**In your own words, summarize in 2-3 sentences how you think version control can help you in the classroom.**

Actually, I think version control can be very useful in a variety of ways in this class.

First, version control can make it easier for professor and me to communicate with each other via GitHub and share the information simultaneously using good tools such as Forking and rewinding and Collaboration. Even if there are some comments related to appropriateness and necessities of version control from solo data-analysts, I, as a novice of R & Github, also one of students learning the mechanisms of the infrastructure of collaborating the information among researchers, these kinds of version controls would be needed for me as I think.

Second, version control can allow myself to think numerous ways about solving problems or issues. It will be especially helpful when I consider doing different ways to conduct a certain kind of project or research as I think. Using Branches, I can keep track of all my branches of thinkings to do researches or any assignments, which can make me comfortable in terms of being arranged automatically even though I did not make the individual folder in my local repository (such as my MacBook).

In addition to the above pros, there will be so many useful and informative points when using version control as I expect !!.

**Problem 4**

**In this exercise, you will import, munge, clean and summarize datasets from Wu and Hamada's Experiments:**

**Planning, Design and Analysis book you will use in the Spring. For each one, please weave your code and text to describe both your process and observations. Make sure you create a tidy dataset describing the variables, create a summary table of the data, note issues with the data.**

a. Sensory data from five operators. http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat

b. Gold Medal performance for Olympic Men's Long Jump, year is coded as 1900=0. http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/LongJumpData.dat

c. Brain weight (g) and body weight (kg) for 62 species. http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/BrainandBodyWeight.dat

d. Triplicate measurements of tomato yield for two varieties of tomatos at three planting densities. http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/tomato.dat

## a. Sensory data from five operators.

```
library(knitr)

Sensory.data <- read.csv("Sensory.csv",header=FALSE,sep=" ",stringsAsFactors = FALSE)
kable(Sensory.data[c(1:8),] ,caption="Raw data")
```

Table 1: Raw data

| V1 | V2 | V3 | V4 | V5 | V6 |
|---|---|---|---|---|---|
| Operator | NA | NA | NA | NA | NA |
| Item | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 |
| 1 | 4.3 | 4.9 | 3.3 | 5.3 | 4.4 |
| 4.3 | 4.5 | 4.0 | 5.5 | 3.3 | NA |
| 4.1 | 5.3 | 3.4 | 5.7 | 4.7 | NA |
| 2 | 6.0 | 5.3 | 4.5 | 5.9 | 4.7 |
| 4.9 | 6.3 | 4.2 | 5.5 | 4.9 | NA |
| 6.0 | 5.9 | 4.7 | 6.3 | 4.6 | NA |

```
## It needs to be cleaned.

## Set the column names as the follow :
first.data.names <- c("ID","1st op.","2nd op.","3rd op.","4th op.","5th op.")

## Check the type of the given data
typeof(Sensory.data) # list
```

```
## [1] "list"
```

```
## Change the form of the given data to matrix.
Sensory.data <-as.matrix(Sensory.data)

## Assign the column names using the character vector we made at the above line.
colnames(Sensory.data) <- first.data.names
```

```r
## Eliminate the first two row
Sensory.data <- Sensory.data[-c(1:2),]

## Show the (a little bit arranged) data set again.
kable(Sensory.data[c(1:8),], caption="A little bit cleaned data")
```

Table 2: A little bit cleaned data

| ID | 1st op. | 2nd op. | 3rd op. | 4th op. | 5th op. |
|---|---|---|---|---|---|
| 1 | 4.3 | 4.9 | 3.3 | 5.3 | 4.4 |
| 4.3 | 4.5 | 4.0 | 5.5 | 3.3 | NA |
| 4.1 | 5.3 | 3.4 | 5.7 | 4.7 | NA |
| 2 | 6.0 | 5.3 | 4.5 | 5.9 | 4.7 |
| 4.9 | 6.3 | 4.2 | 5.5 | 4.9 | NA |
| 6.0 | 5.9 | 4.7 | 6.3 | 4.6 | NA |
| 3 | 2.4 | 2.5 | 2.3 | 3.1 | 2.4 |
| 3.9 | 3.0 | 2.8 | 2.7 | 1.3 | NA |

```r
## From the above shown data set, we need to fill the NA value with corresponding the value
## located in ID column.
## That is, for example, the (2,6)th element NA value
## should be exchanged with the (2,1)th element : 4.3
## Plus, the (2,1)th element should be filled with '2' (second ID number)

## We need to adjust the data set for making what I have mentioned above.

## Let's make the function that allows us to do the above thing.

NA.correction.function <- function(data){

  for(i in 1 : nrow(data) )
    for(j in 1 : ncol(data) )

      if(is.na(data[i,j])==TRUE) {

        data[i,j] <- data[i,1]

        data[i,1] <- i

      }

  return(data)

}


Arranged.data <- NA.correction.function(Sensory.data)

kable(Arranged.data[c(1:14),], caption="3rd cleaned data set")
```

Table 3: 3rd cleaned data set

| ID | 1st op. | 2nd op. | 3rd op. | 4th op. | 5th op. |
|---|---|---|---|---|---|
| 1 | 4.3 | 4.9 | 3.3 | 5.3 | 4.4 |
| 2 | 4.5 | 4.0 | 5.5 | 3.3 | 4.3 |
| 3 | 5.3 | 3.4 | 5.7 | 4.7 | 4.1 |
| 2 | 6.0 | 5.3 | 4.5 | 5.9 | 4.7 |
| 5 | 6.3 | 4.2 | 5.5 | 4.9 | 4.9 |
| 6 | 5.9 | 4.7 | 6.3 | 4.6 | 6.0 |
| 3 | 2.4 | 2.5 | 2.3 | 3.1 | 2.4 |
| 8 | 3.0 | 2.8 | 2.7 | 1.3 | 3.9 |
| 9 | 3.9 | 2.6 | 4.6 | 2.2 | 1.9 |
| 4 | 7.4 | 8.2 | 6.4 | 6.8 | 6.0 |
| 11 | 7.9 | 5.9 | 7.3 | 6.1 | 7.1 |
| 12 | 7.1 | 6.9 | 7.0 | 6.7 | 6.4 |
| 5 | 5.7 | 6.3 | 5.4 | 6.1 | 5.9 |
| 14 | 5.7 | 5.4 | 6.2 | 6.5 | 5.8 |

```
## We have completed what we wanted to do, however we have to align the ID number in order
## (1st column)
## So I decide to weave another function to do that.

ID.align.function <- function(data) {

  for (i in 1 : nrow(data) )
    if(data[i,1]!=1) data[i,1] <- i

    return(data)

}

## Let's apply this function to the data

Arranged.data <- ID.align.function(Arranged.data)

kable(Arranged.data[c(1:20),], caption="Finally arranged data set !")
```

Table 4: Finally arranged data set !

| ID | 1st op. | 2nd op. | 3rd op. | 4th op. | 5th op. |
|---|---|---|---|---|---|
| 1 | 4.3 | 4.9 | 3.3 | 5.3 | 4.4 |
| 2 | 4.5 | 4.0 | 5.5 | 3.3 | 4.3 |
| 3 | 5.3 | 3.4 | 5.7 | 4.7 | 4.1 |
| 4 | 6.0 | 5.3 | 4.5 | 5.9 | 4.7 |
| 5 | 6.3 | 4.2 | 5.5 | 4.9 | 4.9 |
| 6 | 5.9 | 4.7 | 6.3 | 4.6 | 6.0 |
| 7 | 2.4 | 2.5 | 2.3 | 3.1 | 2.4 |
| 8 | 3.0 | 2.8 | 2.7 | 1.3 | 3.9 |
| 9 | 3.9 | 2.6 | 4.6 | 2.2 | 1.9 |
| 10 | 7.4 | 8.2 | 6.4 | 6.8 | 6.0 |
| 11 | 7.9 | 5.9 | 7.3 | 6.1 | 7.1 |
| 12 | 7.1 | 6.9 | 7.0 | 6.7 | 6.4 |
| 13 | 5.7 | 6.3 | 5.4 | 6.1 | 5.9 |

| ID | 1st op. | 2nd op. | 3rd op. | 4th op. | 5th op. |
|---|---|---|---|---|---|
| 14 | 5.7 | 5.4 | 6.2 | 6.5 | 5.8 |
| 15 | 6.0 | 6.1 | 7.0 | 4.9 | 5.8 |
| 16 | 2.2 | 2.4 | 1.7 | 3.4 | 1.7 |
| 17 | 1.8 | 2.1 | 4.0 | 1.7 | 3.0 |
| 18 | 3.3 | 1.1 | 3.3 | 2.1 | 2.1 |
| 19 | 1.2 | 1.5 | 1.2 | 0.9 | 0.7 |
| 20 | 2.4 | 0.8 | 1.2 | 1.3 | 1.3 |

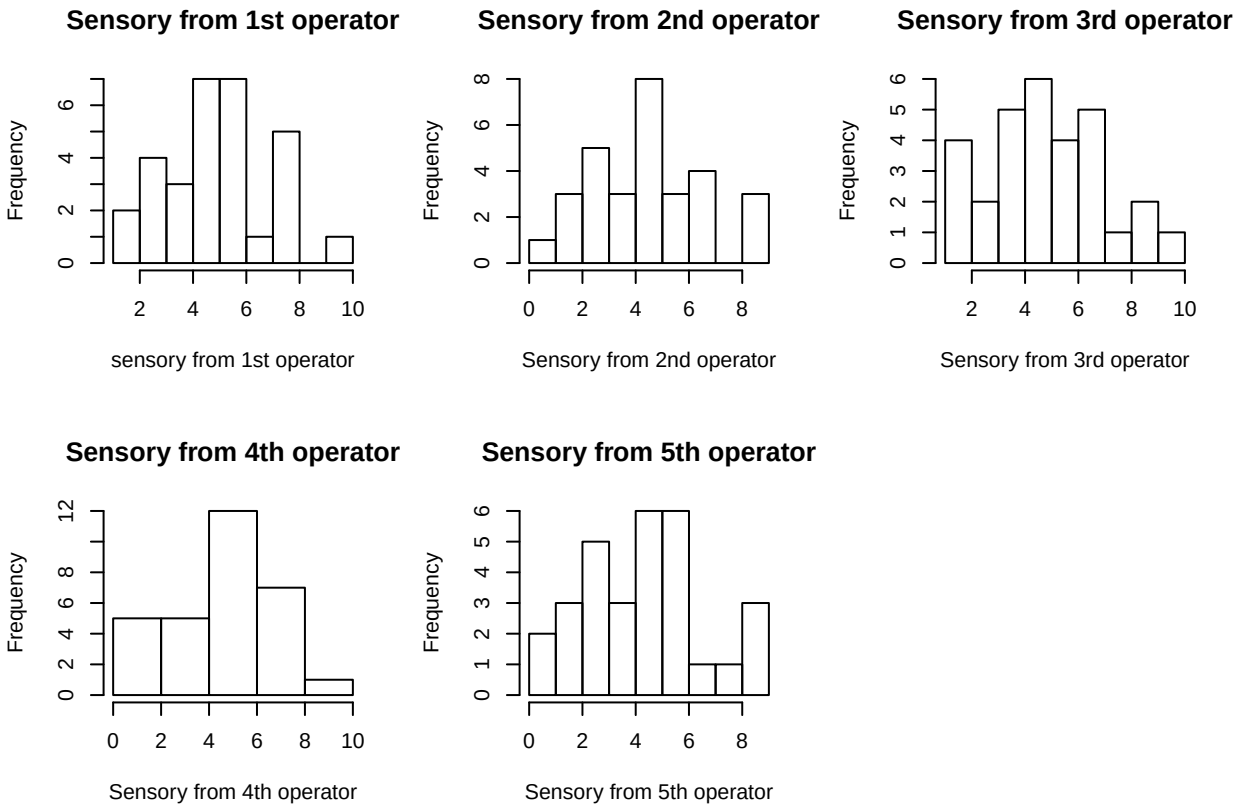
```
## Change the type of the elememt (from character to numeric) of arranged matrix
mode(Arranged.data) <- "numeric"


## Now, we can see the completely arranged data with the first column : ID,
## and other remaining columns meaning the each operator, row : obs.
## Naturally, each cell data means (except for first column data) sensory data
## corresponding to each operator.

## Summary table for arranged.data for each variable.

kable(summary(Arranged.data[,-1]), caption = "Quick summary of sensory data from each operator.")
```

Table 5: Quick summary of sensory data from each operator.

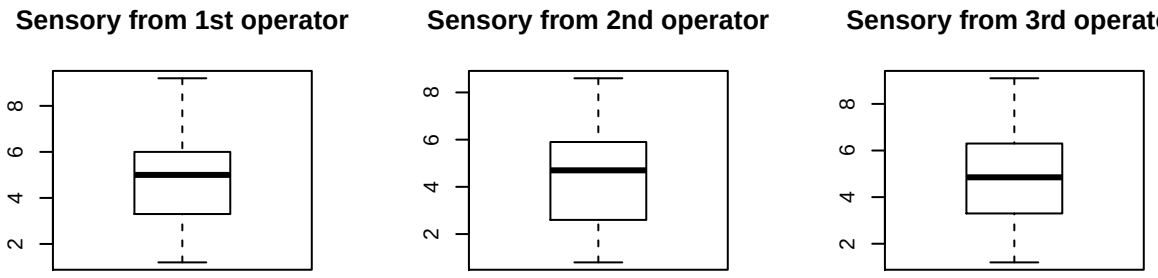
| 1st op. | 2nd op. | 3rd op. | 4th op. | 5th op. |
|---|---|---|---|---|
| Min. :1.200 | Min. :0.800 | Min. :1.200 | Min. :0.90 | Min. :0.700 |
| 1st Qu.:3.450 | 1st Qu.:2.650 | 1st Qu.:3.450 | 1st Qu.:3.15 | 1st Qu.:2.850 |
| Median :5.000 | Median :4.700 | Median :4.850 | Median :4.65 | Median :4.350 |
| Mean :4.967 | Mean :4.403 | Mean :4.900 | Mean :4.58 | Mean :4.433 |
| 3rd Qu.:6.000 | 3rd Qu.:5.775 | 3rd Qu.:6.275 | 3rd Qu.:6.05 | 3rd Qu.:5.875 |
| Max. :9.200 | Max. :8.600 | Max. :9.100 | Max. :9.40 | Max. :9.000 |

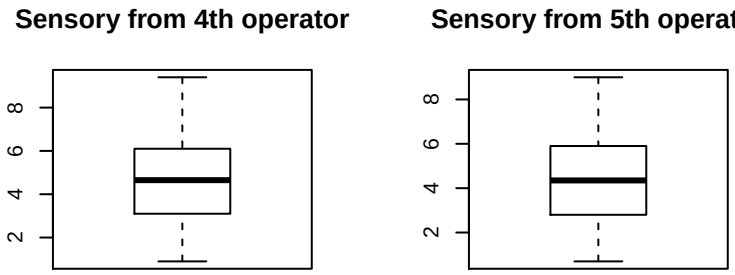
```
## Visualization for each variable.


par(mfrow=c(2,3))

# Histogram.

# kable(hist(Arranged.data[,-1]), caption = "Histogram of sensory data from each operator.")


hist(Arranged.data[,2],main="Sensory from 1st operator",xlab="sensory from 1st operator")

hist(Arranged.data[,3],main="Sensory from 2nd operator",xlab="Sensory from 2nd operator")

hist(Arranged.data[,4],main="Sensory from 3rd operator",xlab="Sensory from 3rd operator")

hist(Arranged.data[,5],main="Sensory from 4th operator",xlab="Sensory from 4th operator")

hist(Arranged.data[,6],main="Sensory from 5th operator",xlab="Sensory from 5th operator")
```

```r
par(mfrow=c(2,3))
```




```r
## Box plot.

boxplot(Arranged.data[,2],main="Sensory from 1st operator")

boxplot(Arranged.data[,3],main="Sensory from 2nd operator")

boxplot(Arranged.data[,4],main="Sensory from 3rd operator")

boxplot(Arranged.data[,5],main="Sensory from 4th operator")

boxplot(Arranged.data[,6],main="Sensory from 5th operator")

## There seems to be nothing special about each sensory data set obtained from each operator.
```




**b. Gold Medal performance for Olympic Men's Long Jump, year is coded as 1900=0.**

```
Longjump.data <- read.csv("LongJumpData.csv",header=T,sep=" ",stringsAsFactors = FALSE)
kable(Longjump.data[c(1:8),], caption="Raw data")
```

Table 6: Raw data

| | Year | Long | Jump | Year.1 | Long.1 | Jump.1 | Year.2 | Long.2 | Jump.2 | Year.3 | Long.3 | Jump.3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -4 | 249.75 | 24 | 293.13 | 56 | 308.25 | 80 | 336.25 | NA | NA | NA | NA |
| 2 | 0 | 282.88 | 28 | 304.75 | 60 | 319.75 | 84 | 336.25 | NA | NA | NA | NA |
| 3 | 4 | 289.00 | 32 | 300.75 | 64 | 317.75 | 88 | 343.25 | NA | NA | NA | NA |
| 4 | 8 | 294.50 | 36 | 317.31 | 68 | 350.50 | 92 | 342.50 | NA | NA | NA | NA |
| 5 | 12 | 299.25 | 48 | 308.00 | 72 | 324.50 | NA | NA | NA | NA | NA | NA |
| 6 | 20 | 281.50 | 52 | 298.00 | 76 | 328.50 | NA | NA | NA | NA | NA | NA |
| NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| NA.1 | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |

```
nrow(Longjump.data) # 6
```

```
## [1] 6
```

```
ncol(Longjump.data) # 12
```

```
## [1] 12
```

```
## We need to transform the given data set with the # row : 22 (# of obs), and
## # col : 2 (Year (1900=0 format), Long jump).

## First, we seperate the data by variables by Year, Long jump data.
```

```r
## By concatenating the 1st, 3rd, 5th, and 7th column data, we can get the vector of year data.
year.expected.data <- c(Longjump.data[,1],Longjump.data[,3],Longjump.data[,5],Longjump.data[,7])

length(year.expected.data)
```

```
## [1] 24
```

```r
## By concatenating the 2nd, 4th, 6th,and 8th column data, we can get the vector of Long jump data.
Long.jump.expected.data <- c(Longjump.data[,2],Longjump.data[,4],Longjump.data[,6],Longjump.data[,8])

length(Long.jump.expected.data)
```

```
## [1] 24
```

```r
## Make arranged data with matrix form (column : variable, row : observation)
Arranged.data2 <- matrix(c(year.expected.data,Long.jump.expected.data),nrow=24,ncol=2,byrow=FALSE)


## We arrange the given data, roughly, but we have to change the first column data into 'day'form.
## At first, we need to eliminate the last two row data because they are NAs which are useless.
Arranged.data2 <- Arranged.data2[-c((nrow(Arranged.data2)-1):nrow(Arranged.data2)),]

kable(Arranged.data2[c(1:8),], caption="firstly arranged data for Long jump data")
```

Table 7: firstly arranged data for Long jump data

| | |
|---|---|
| -4 | 249.75 |
| 0 | 282.88 |
| 4 | 289.00 |
| 8 | 294.50 |
| 12 | 299.25 |
| 20 | 281.50 |
| 24 | 293.13 |
| 28 | 304.75 |

```r
## Procedures to change the 1st column data to day data.
edates <- Arranged.data2[,1]

kable(edates[c(1:8)], caption = "need to be changed to date form")
```

Table 8: need to be changed to date form

| x |
|---|
| -4 |
| 0 |
| 4 |
| 8 |
| 12 |
| 20 |
| 24 |
| 28 |

```r
edates[edates>=60] <- edates[edates>=60]
edates <- as.Date(edates, origin="1900-01-01")

kable(edates[c(1:8)], caption = "Date")
```

Table 9: Date

| x |
|---|
| 1899-12-28 |
| 1900-01-01 |
| 1900-01-05 |
| 1900-01-09 |
| 1900-01-13 |
| 1900-01-21 |
| 1900-01-25 |
| 1900-01-29 |

```r
## Success !!

## Eliminate NA data in second column (Long jump data)data using the below code.
Long.jump.expected.data <- Long.jump.expected.data[-c((length(Long.jump.expected.data)-1):length(Long.ju

kable(Long.jump.expected.data[c(1:15)], caption="Finally cleaned Long jump data")
```

Table 10: Finally cleaned Long jump data

| x |
|---|
| 249.75 |
| 282.88 |
| 289.00 |
| 294.50 |
| 299.25 |
| 281.50 |
| 293.13 |
| 304.75 |
| 300.75 |
| 317.31 |
| 308.00 |
| 298.00 |
| 308.25 |
| 319.75 |
| 317.75 |

```r
## Now, we have to recombine this day data and Longjump data into matrix form.

Arranged.data2 <- data.frame(
   Id = c (1:22),
   Date=edates,

   Long.jump.record = Long.jump.expected.data,
```

```
   stringsAsFactors = FALSE
)

kable(Arranged.data2[c(1:15),], caption = "Finally arranged combined data set")
```

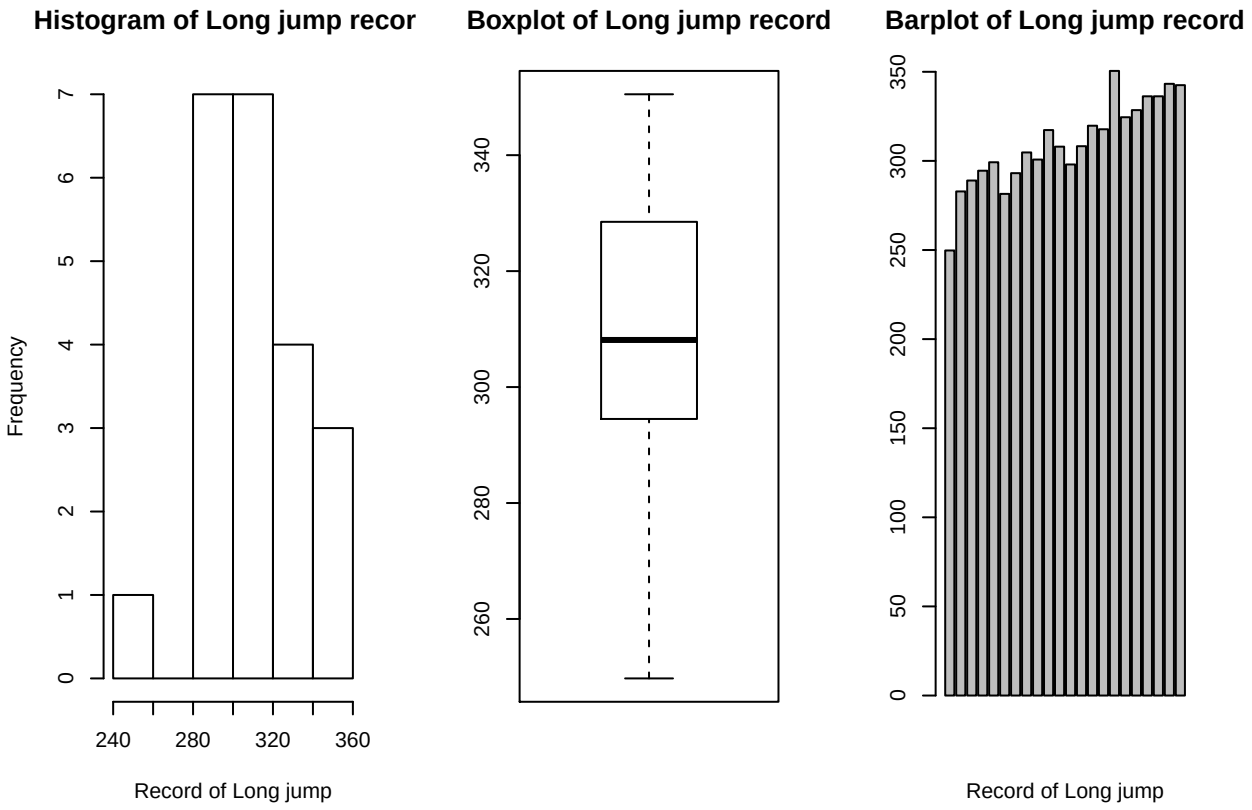
Table 11: Finally arranged combined data set

| Id | Date | Long.jump.record |
|---|---|---|
| 1 | 1899-12-28 | 249.75 |
| 2 | 1900-01-01 | 282.88 |
| 3 | 1900-01-05 | 289.00 |
| 4 | 1900-01-09 | 294.50 |
| 5 | 1900-01-13 | 299.25 |
| 6 | 1900-01-21 | 281.50 |
| 7 | 1900-01-25 | 293.13 |
| 8 | 1900-01-29 | 304.75 |
| 9 | 1900-02-02 | 300.75 |
| 10 | 1900-02-06 | 317.31 |
| 11 | 1900-02-18 | 308.00 |
| 12 | 1900-02-22 | 298.00 |
| 13 | 1900-02-26 | 308.25 |
| 14 | 1900-03-02 | 319.75 |
| 15 | 1900-03-06 | 317.75 |

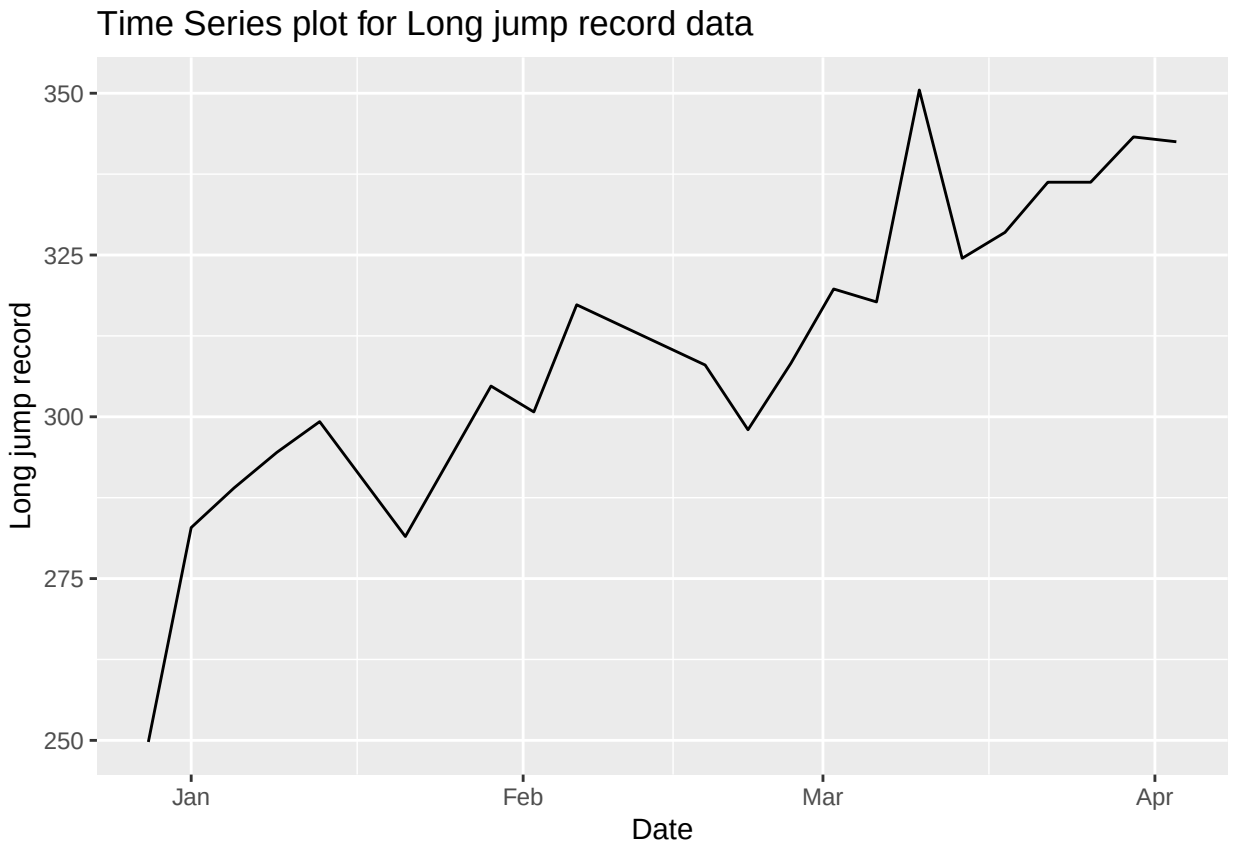
```
## Summary and Visualization

## Summary for Long jump record data.
summary(Arranged.data2$Long.jump.record)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   249.8   295.4   308.1   310.3   327.5   350.5
```

```
par(mfrow=c(1,3))

hist(Arranged.data2$Long.jump.record,main="Histogram of Long jump record",xlab = "Record of Long jump")

boxplot(Arranged.data2$Long.jump.record,main="Boxplot of Long jump record")

barplot(Arranged.data2$Long.jump.record,main="Barplot of Long jump record",xlab = "Record of Long jump")
```




```
par(mfrow=c(1,1))

## Time series plot

library(ggplot2)
Arranged.data2.ts <- ts(Arranged.data2[,-1])

Arranged.data2.ts <- data.frame(Long.jump.record.data = as.numeric(Arranged.data2$Long.jump.record),
                  Date = edates)

Arranged.data2.ts$Date <- as.Date( Arranged.data2.ts$Date, '%Y/%m/%d')

ggplot( data = Arranged.data2.ts, aes( Arranged.data2.ts$Date, as.numeric(Arranged.data2$Long.jump.reco
```




### c. Brain,body, and weight data

```
Brain.Body.weight.data <- read.csv("BrainandBodyWeight.csv",header=T,sep=" ",stringsAsFactors = FALSE)
kable(Brain.Body.weight.data, caption= "Raw data")
```

Table 12: Raw data

| Body | Wt | Brain | Wt.1 | Body.1 | Wt.2 | Brain.1 | Wt.3 | Body.2 | Wt.4 | Brain.2 | Wt.5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3.385 | 44.5 | 521.000 | 655.0 | 2.500 | 12.10 | NA | NA | NA | NA | NA | NA |
| 0.480 | 15.5 | 0.785 | 3.5 | 55.500 | 175.00 | NA | NA | NA | NA | NA | NA |
| 1.350 | 8.1 | 10.000 | 115.0 | 100.000 | 157.00 | NA | NA | NA | NA | NA | NA |
| 465.000 | 423.0 | 3.300 | 25.6 | 52.160 | 440.00 | NA | NA | NA | NA | NA | NA |
| 36.330 | 119.5 | 0.200 | 5.0 | 10.550 | 179.50 | NA | NA | NA | NA | NA | NA |
| 27.660 | 115.0 | 1.410 | 17.5 | 0.550 | 2.40 | NA | NA | NA | NA | NA | NA |
| 14.830 | 98.2 | 529.000 | 680.0 | 60.000 | 81.00 | NA | NA | NA | NA | NA | NA |
| 1.040 | 5.5 | 207.000 | 406.0 | 3.600 | 21.00 | NA | NA | NA | NA | NA | NA |
| 4.190 | 58.0 | 85.000 | 325.0 | 4.288 | 39.20 | NA | NA | NA | NA | NA | NA |
| 0.425 | 6.4 | 0.750 | 12.3 | 0.280 | 1.90 | NA | NA | NA | NA | NA | NA |
| 0.101 | 4.0 | 62.000 | 1320.0 | 0.075 | 1.20 | NA | NA | NA | NA | NA | NA |
| 0.920 | 5.7 | 6654.000 | 5712.0 | 0.122 | 3.00 | NA | NA | NA | NA | NA | NA |
| 1.000 | 6.6 | 3.500 | 3.9 | 0.048 | 0.33 | NA | NA | NA | NA | NA | NA |
| 0.005 | 0.1 | 6.800 | 179.0 | 192.000 | 180.00 | NA | NA | NA | NA | NA | NA |
| 0.060 | 1.0 | 35.000 | 56.0 | 3.000 | 25.00 | NA | NA | NA | NA | NA | NA |
| 3.500 | 10.8 | 4.050 | 17.0 | 160.000 | 169.00 | NA | NA | NA | NA | NA | NA |
| 2.000 | 12.3 | 0.120 | 1.0 | 0.900 | 2.60 | NA | NA | NA | NA | NA | NA |
| 1.700 | 6.3 | 0.023 | 0.4 | 1.620 | 11.40 | NA | NA | NA | NA | NA | NA |

| Body | Wt | Brain | Wt.1 | Body.1 | Wt.2 | Brain.1 | Wt.3 | Body.2 | Wt.4 | Brain.2 | Wt.5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2547.000 | 4603.0 | 0.010 | 0.3 | 0.104 | 2.50 | NA | NA | NA | NA | NA | NA |
| 0.023 | 0.3 | 1.400 | 12.5 | 4.235 | 50.40 | NA | NA | NA | NA | NA | NA |
| 187.100 | 419.0 | 250.000 | 490.0 | NA | NA | NA | NA | NA | NA | NA | NA |

```
ncol(Brain.Body.weight.data) # 12
```

```
## [1] 12
```

```
nrow(Brain.Body.weight.data) # 21
```

```
## [1] 21
```

```
## We need to transform the given data set with the # row : 62, and # col : 2 (Brain Weight (g), Body W
## First, we seperate the data by variables by Brain weight, Body weight


## By concatenating the 3rd, 4th, and 6th column data, we can get the vector of brain weight data.
Brain.Weight.expected.data <- c(Brain.Body.weight.data[,3],Brain.Body.weight.data[,4],Brain.Body.weight

## By concatenating the 1st, 2nd, and 5th column data, we can get the vector of Body weight data.
Body.Weight.expected.data <- c(Brain.Body.weight.data[,1],Brain.Body.weight.data[,2],Brain.Body.weight.c


## Make arranged data with matrix form (column : variable, row : observation)
Arranged.data3 <- matrix(c(Brain.Weight.expected.data,Body.Weight.expected.data),nrow=63,ncol=2,byrow=F

## Set the column names and row number
colnames(Arranged.data3) <- c("Brain.Weight(g)","Body.Weight(kg)")

rownames(Arranged.data3) <- c(1:63)

kable(Arranged.data3[c(1:17),], caption= "Cleaned data")
```

Table 13: Cleaned data

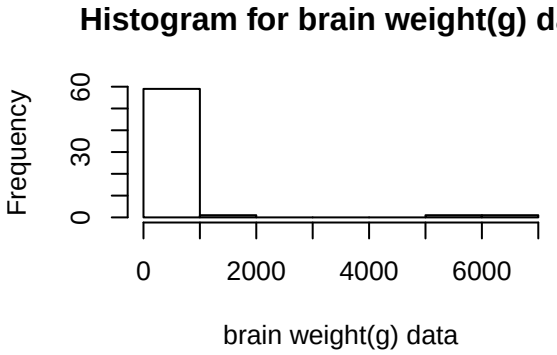
| Brain.Weight(g) | Body.Weight(kg) |
|---|---|
| 521.000 | 3.385 |
| 0.785 | 0.480 |
| 10.000 | 1.350 |
| 3.300 | 465.000 |
| 0.200 | 36.330 |
| 1.410 | 27.660 |
| 529.000 | 14.830 |
| 207.000 | 1.040 |
| 85.000 | 4.190 |
| 0.750 | 0.425 |
| 62.000 | 0.101 |
| 6654.000 | 0.920 |
| 3.500 | 1.000 |
| 6.800 | 0.005 |
| 35.000 | 0.060 |
| 4.050 | 3.500 |
| 0.120 | 2.000 |

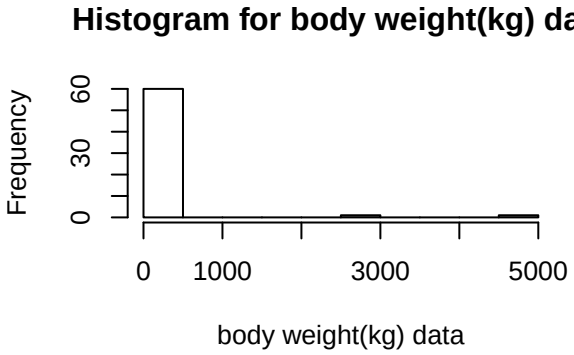
```r
## Summary table for arranged.data for each variable.
kable(summary(Arranged.data3), caption= "Summary table for Cleaned data")
```

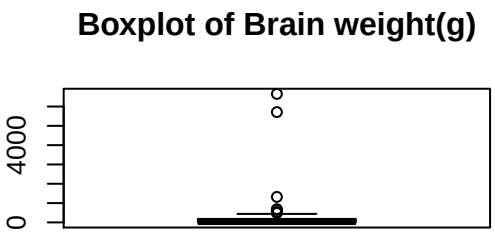
Table 14: Summary table for Cleaned data

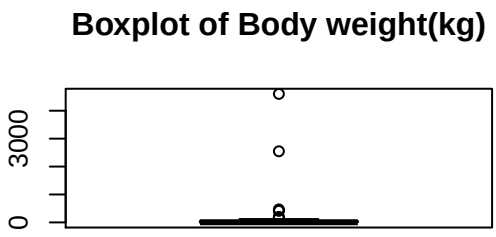
| Brain.Weight(g) | Body.Weight(kg) |
|---|---|
| Min. : 0.010 | Min. : 0.005 |
| 1st Qu.: 2.525 | 1st Qu.: 0.940 |
| Median : 17.250 | Median : 4.261 |
| Mean : 322.046 | Mean : 159.878 |
| 3rd Qu.: 178.000 | 3rd Qu.: 50.245 |
| Max. :6654.000 | Max. :4603.000 |
| NA's :1 | NA's :1 |

```r
## Visualization for each variable.

par(mfrow=c(2,2))
# Histogram.

hist(Arranged.data3[,1],main="Histogram for brain weight(g) data",xlab="brain weight(g) data") # for br

hist(Arranged.data3[,2],main="Histogram for body weight(kg) data",xlab="body weight(kg) data") # for bo

## From the each variable's histograms, we can find the fact that most of the observed data
## seems to aggregate in the interval (0,1000)


## Box plot for each variable
boxplot(Arranged.data3[,1],main="Boxplot of Brain weight(g)")# for brain weight

boxplot(Arranged.data3[,2],main="Boxplot of Body weight(kg)")# for body weight
```



```
## There seems to be some outliars within the each variable data sets.
```

### d. Triplicate measurements of tomato yield for two varieties of tomatoes at three planting densities.

```
library(stringr)
## In terms of this data set, I will use the 'readLines' Command for reading the raw data set.

Triplicate.meas.tomatoes.at3planting.den.data <- readLines(con="tomato.csv")

kable(Triplicate.meas.tomatoes.at3planting.den.data, caption = "Raw data")
```

Table 15: Raw data

| x |
|---|
| #this needs reformatting to read into Splus |
| 10000 20000 30000 |
| Ife#1 16.1,15.3,17.5 16.6,19.2,18.5 20.8,18.0,21.0 |
| PusaEarlyDwarf 8.1,8.6,10.1, 12.7,13.7,11.5 14.4,15.4,13.7 |

```
## Arrange the raw data set a little bit.

Triplicate.meas.tomatoes.at3planting.den.data <- Triplicate.meas.tomatoes.at3planting.den.data[-1]

Triplicate.meas.tomatoes.at3planting.den.data<-as.matrix(Triplicate.meas.tomatoes.at3planting.den.data,

## We need to handle with the repeated measure.

## The way I choose to deal with repeated measure is assigning the column vector for each kind of tomat

## Let's make the numeric vector of measure of first kind of tomato

tempo.first.tomato.repeated.measure <- Triplicate.meas.tomatoes.at3planting.den.data[2,]

## Split the 'tempo.first.tomato.repeated.measure' data for obtaining the individual data point.

tempo.first.tomato.repeated.measure2 <- str_split(tempo.first.tomato.repeated.measure, ",") # sep : com

tempo.first.tomato.repeated.measure2
```

```
## [[1]]
## [1] "Ife\\#1          16.1" "15.3"
## [3] "17.5   16.6"           "19.2"
## [5] "18.5   20.8"           "18.0"
## [7] "21.0"
```

```
## Even if we have done the above procedure, there seems to need more efforts to get the individual obs


## 2nd, 5th, 8th, and 9th obs. data from first kind of tomato can be received by just following indexin
Second.data.in.the.first.tomato.repeated.measure <- tempo.first.tomato.repeated.measure2[[1]][2]

Fifth.data.in.the.first.tomato.repeated.measure <- tempo.first.tomato.repeated.measure2[[1]][4]
```

```
Eightth.data.in.the.first.tomato.repeated.measure <- tempo.first.tomato.repeated.measure2[[1]][6]

Nineth.data.in.the.first.tomato.repeated.measure <- tempo.first.tomato.repeated.measure2[[1]][7]

## The thing is that we should break into multi parts what is composed of several things (such as combi
tempo.first.tomato.repeated.measure.first. <- str_split(tempo.first.tomato.repeated.measure2[[1]][1], "

tempo.first.tomato.repeated.measure.first.
```

```
## [[1]]
##  [1] "Ife\\#1" ""        ""        ""        ""        ""        ""
##  [8] ""        ""        ""        ""        "16.1"
```

```
First.data.in.the.first.tomato.repeated.measure <-  tempo.first.tomato.repeated.measure.first.[[1]][12]

First.data.in.the.first.tomato.repeated.measure
```

```
## [1] "16.1"
```

```
tempo.first.tomato.repeated.measure.third. <- str_split(tempo.first.tomato.repeated.measure2[[1]][3], "

tempo.first.tomato.repeated.measure.third.
```

```
## [[1]]
## [1] "17.5" ""     ""     "16.6"
```

```
Third.data.in.the.first.tomato.repeated.measure <- tempo.first.tomato.repeated.measure.third.[[1]][1]

Third.data.in.the.first.tomato.repeated.measure
```

```
## [1] "17.5"
```

```
Fourth.data.in.the.first.tomato.repeated.measure <- tempo.first.tomato.repeated.measure.third.[[1]][4]

Fourth.data.in.the.first.tomato.repeated.measure
```

```
## [1] "16.6"
```

```
tempo.first.tomato.repeated.measure.fifth. <- str_split(tempo.first.tomato.repeated.measure2[[1]][5], "


tempo.first.tomato.repeated.measure.fifth.
```

```
## [[1]]
## [1] "18.5" ""     ""     "20.8"
```

```
Sixth.data.in.the.first.tomato.repeated.measure <- tempo.first.tomato.repeated.measure.fifth.[[1]][1]

Sixth.data.in.the.first.tomato.repeated.measure
```

```
## [1] "18.5"
```

```
Seventh.data.in.the.first.tomato.repeated.measure <- tempo.first.tomato.repeated.measure.fifth.[[1]][4]

Seventh.data.in.the.first.tomato.repeated.measure
```

```
## [1] "20.8"
```

```r
# By weaving the above code, we can get the individual obs !!


## Now, we can concatenate repeated measures of first tomato (# : 9)
First.tomato.repeated.measure.vector <- c(First.data.in.the.first.tomato.repeated.measure, Second.data.

First.tomato.repeated.measure.vector <- as.numeric(First.tomato.repeated.measure.vector)


First.tomato.repeated.measure.vector
```

```
## [1] 16.1 15.3 17.5 16.6 19.2 18.5 20.8 18.0 21.0
```

```r
## It is the numeric vector composed of obs. of first kind of tomato.



## Do the same procedure to make the numeric vector of measure of Second kind of tomato

tempo.second.tomato.repeated.measure <- Triplicate.meas.tomatoes.at3planting.den.data[3,]



tempo.second.tomato.repeated.measure2 <- str_split(tempo.second.tomato.repeated.measure, ",")

tempo.second.tomato.repeated.measure2
```

```
## [[1]]
## [1] "PusaEarlyDwarf   8.1" "8.6"                 "10.1"
## [4] "    12.7"             "13.7"                "11.5   14.4"
## [7] "15.4"                 "13.7 "
```

```r
Second.data.in.the.second.tomato.repeated.measure <- tempo.second.tomato.repeated.measure2[[1]][2]

Third.data.in.the.second.tomato.repeated.measure <- tempo.second.tomato.repeated.measure2[[1]][3]


Fourth.data.in.the.second.tomato.repeated.measure <- tempo.second.tomato.repeated.measure2[[1]][4]


Fifth.data.in.the.second.tomato.repeated.measure <- tempo.second.tomato.repeated.measure2[[1]][5]

Eightth.data.in.the.second.tomato.repeated.measure <- tempo.second.tomato.repeated.measure2[[1]][7]

Nineth.data.in.the.second.tomato.repeated.measure <- tempo.second.tomato.repeated.measure2[[1]][8]



tempo.second.tomato.repeated.measure.first. <- str_split(tempo.second.tomato.repeated.measure2[[1]][1],

tempo.second.tomato.repeated.measure.first.
```

```
## [[1]]
## [1] "PusaEarlyDwarf" ""               ""               "8.1"
```

```
First.data.in.the.second.tomato.repeated.measure <-  tempo.second.tomato.repeated.measure.first.[[1]][4]

First.data.in.the.second.tomato.repeated.measure
```

```
## [1] "8.1"
```

```
tempo.second.repeated.measure.sixth. <- str_split(tempo.second.tomato.repeated.measure2[[1]][6], " ")

tempo.second.repeated.measure.sixth.
```

```
## [[1]]
## [1] "11.5" ""     ""     "14.4"
```

```
Sixth.data.in.the.second.tomato.repeated.measure <- tempo.second.repeated.measure.sixth.[[1]][1]

Sixth.data.in.the.second.tomato.repeated.measure
```

```
## [1] "11.5"
```

```
Seventh.data.in.the.second.tomato.repeated.measure <- tempo.second.repeated.measure.sixth.[[1]][4]

Seventh.data.in.the.second.tomato.repeated.measure
```

```
## [1] "14.4"
```

```
## Now, we can concatenate repeated measures of the second kind of tomato (# : 9)

Second.tomato.repeated.measure.vector <- c(First.data.in.the.second.tomato.repeated.measure, Second.data



Second.tomato.repeated.measure.vector <- as.numeric(Second.tomato.repeated.measure.vector)


Second.tomato.repeated.measure.vector
```

```
## [1]  8.1  8.6 10.1 12.7 13.7 11.5 14.4 15.4 13.7
```

```
## The next thing we should do is the arranging the data set can look comfortable.
## I will use the 'list' form to look this arranged data set nice to viewers.



## Assign each (repeated) data set to the each kind of tomato and each density 'list' space
## (refer to the following R code)


 First.Tomato.First.Density <- new.env()

 First.Tomato.First.Density$first.obs<- as.numeric(First.tomato.repeated.measure.vector[1])

 First.Tomato.First.Density$second.obs<- as.numeric(First.tomato.repeated.measure.vector[2])

 First.Tomato.First.Density$third.obs<- as.numeric(First.tomato.repeated.measure.vector[3])

 First.Tomato.First.Density <- as.list(First.Tomato.First.Density)
```

```
First.Tomato.Second.Density <- new.env()
First.Tomato.Second.Density$first.obs<- as.numeric(First.tomato.repeated.measure.vector[4])
First.Tomato.Second.Density$second.obs<- as.numeric(First.tomato.repeated.measure.vector[5])
First.Tomato.Second.Density$third.obs<- as.numeric(First.tomato.repeated.measure.vector[6])
First.Tomato.Second.Density <- as.list(First.Tomato.Second.Density)


First.Tomato.Third.Density <- new.env()
First.Tomato.Third.Density$first.obs<- as.numeric(First.tomato.repeated.measure.vector[7])
First.Tomato.Third.Density$second.obs<- as.numeric(First.tomato.repeated.measure.vector[8])
First.Tomato.Third.Density$third.obs<- as.numeric(First.tomato.repeated.measure.vector[9])
First.Tomato.Third.Density <- as.list(First.Tomato.Third.Density)




 Second.Tomato.First.Density <- new.env()
Second.Tomato.First.Density$first.obs<- Second.tomato.repeated.measure.vector[1]
Second.Tomato.First.Density$second.obs<- Second.tomato.repeated.measure.vector[2]
Second.Tomato.First.Density$third.obs<- Second.tomato.repeated.measure.vector[3]
Second.Tomato.First.Density <- as.list(Second.Tomato.First.Density)


Second.Tomato.Second.Density <- new.env()
Second.Tomato.Second.Density$first.obs<- Second.tomato.repeated.measure.vector[4]
Second.Tomato.Second.Density$second.obs<- Second.tomato.repeated.measure.vector[5]
Second.Tomato.Second.Density$third.obs<- Second.tomato.repeated.measure.vector[6]
Second.Tomato.Second.Density <- as.list(Second.Tomato.Second.Density)
```

```
Second.Tomato.Third.Density <- new.env()

Second.Tomato.Third.Density$first.obs<- Second.tomato.repeated.measure.vector[7]

Second.Tomato.Third.Density$second.obs<- Second.tomato.repeated.measure.vector[8]

Second.Tomato.Third.Density$third.obs<- Second.tomato.repeated.measure.vector[9]

Second.Tomato.Third.Density <- as.list(Second.Tomato.Third.Density)



Arranged.data4 <- list(First.Tomato.First.Density, First.Tomato.Second.Density, First.Tomato.Third.Dens
                  Second.Tomato.First.Density, Second.Tomato.Second.Density, Second.Tomato.Third.Densi


names(Arranged.data4)[[1]] <- "First Tomato obtained from 1st density"

names(Arranged.data4)[[2]] <- "First Tomato obtained from 2nd density"

names(Arranged.data4)[[3]] <- "First Tomato obtained from 3rd density"

names(Arranged.data4)[[4]] <- "Second Tomato obtained from 1st density"

names(Arranged.data4)[[5]] <- "Second Tomato obtained from 2nd density"

names(Arranged.data4)[[6]] <- "Second Tomato obtained from 3rd density"

library(tinytex)
library(devtools)
```

```
## Loading required package: usethis
```

```
Arranged.data4
```

```
## $`First Tomato obtained from 1st density`
## $`First Tomato obtained from 1st density`$third.obs
## [1] 17.5
##
## $`First Tomato obtained from 1st density`$first.obs
## [1] 16.1
##
## $`First Tomato obtained from 1st density`$second.obs
## [1] 15.3
##
##
## $`First Tomato obtained from 2nd density`
## $`First Tomato obtained from 2nd density`$third.obs
## [1] 18.5
##
## $`First Tomato obtained from 2nd density`$first.obs
## [1] 16.6
##
```

```
## $`First Tomato obtained from 2nd density`$second.obs
## [1] 19.2
##
##
## $`First Tomato obtained from 3rd density`
## $`First Tomato obtained from 3rd density`$third.obs
## [1] 21
##
## $`First Tomato obtained from 3rd density`$first.obs
## [1] 20.8
##
## $`First Tomato obtained from 3rd density`$second.obs
## [1] 18
##
##
## $`Second Tomato obtained from 1st density`
## $`Second Tomato obtained from 1st density`$third.obs
## [1] 10.1
##
## $`Second Tomato obtained from 1st density`$first.obs
## [1] 8.1
##
## $`Second Tomato obtained from 1st density`$second.obs
## [1] 8.6
##
##
## $`Second Tomato obtained from 2nd density`
## $`Second Tomato obtained from 2nd density`$third.obs
## [1] 11.5
##
## $`Second Tomato obtained from 2nd density`$first.obs
## [1] 12.7
##
## $`Second Tomato obtained from 2nd density`$second.obs
## [1] 13.7
##
##
## $`Second Tomato obtained from 3rd density`
## $`Second Tomato obtained from 3rd density`$third.obs
## [1] 13.7
##
## $`Second Tomato obtained from 3rd density`$first.obs
## [1] 14.4
##
## $`Second Tomato obtained from 3rd density`$second.obs
## [1] 15.4
```

```
## The above arranged data set has the form of list which can allow us to seize the structure of the g
```

```
## Summary table for arranged.data for each tomato from each density
```

```r
# kable(summary(Arranged.data4))

summary(as.numeric(Arranged.data4$`First Tomato obtained from 1st density`))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    15.3    15.7    16.1    16.3    16.8    17.5
```

```r
summary(as.numeric(Arranged.data4$`First Tomato obtained from 2nd density`))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.60   17.55   18.50   18.10   18.85   19.20
```

```r
summary(as.numeric(Arranged.data4$`First Tomato obtained from 3rd density`))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   19.40   20.80   19.93   20.90   21.00
```

```r
summary(as.numeric(Arranged.data4$`Second Tomato obtained from 1st density`))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.100   8.350   8.600   8.933   9.350  10.100
```

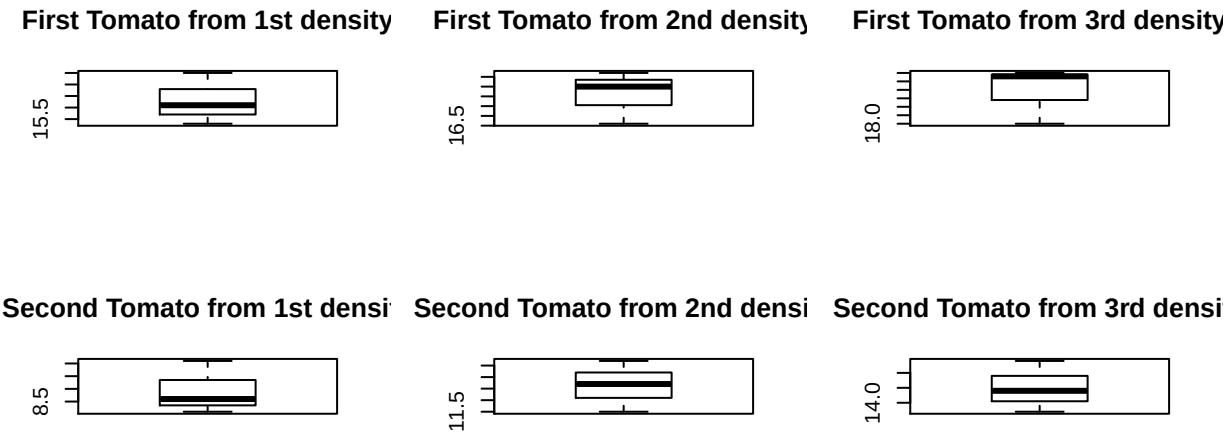
```r
summary(as.numeric(Arranged.data4$`Second Tomato obtained from 2nd density`))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.50   12.10   12.70   12.63   13.20   13.70
```

```r
summary(as.numeric(Arranged.data4$`Second Tomato obtained from 3rd density`))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.70   14.05   14.40   14.50   14.90   15.40
```

```r
## Visualization for each tomato from each density.

par(mfrow=c(3,3))

## Box plot for each tomato from each density

boxplot(as.numeric(Arranged.data4$`First Tomato obtained from 1st density`),main="First Tomato from 1st

boxplot(as.numeric(Arranged.data4$`First Tomato obtained from 2nd density`),main="First Tomato from 2nd

boxplot(as.numeric(Arranged.data4$`First Tomato obtained from 3rd density`),main="First Tomato from 3rd

boxplot(as.numeric(Arranged.data4$`Second Tomato obtained from 1st density`),main="Second Tomato from 1s

boxplot(as.numeric(Arranged.data4$`Second Tomato obtained from 2nd density`),main="Second Tomato from 2n

boxplot(as.numeric(Arranged.data4$`Second Tomato obtained from 3rd density`),main="Second Tomato from 3r

## There doensn't seem to be no outliar or weird points when we see the a variety of plots.
```




**Problem 5**

**In the swirl lessons, you played with a dataset "plants". Our ultimate goal is to see if there is a relationship between pH and Foliage_Color. Consider a statistic that combines the information in pH_Min and pH_Max.**

**Clean, summarize and transform the data as appropriate. Use function 'lm' to test for a relationship. Report both the coefficients and ANOVA results in table form.**

**Note that if you didn't just do the swirl lesson, it is now not available. Add the following code to your project to retrieve it.**

```
library(swirl)
```

```
##
## | Hi! I see that you have some variables saved in your workspace. To keep
## | things running smoothly, I recommend you clean up before starting swirl.
##
## | Type ls() to see a list of the variables in your workspace. Then, type
## | rm(list=ls()) to clear your workspace.
##
## | Type swirl() when you are ready to begin.
```

```
# Path to data
.datapath <- file.path(path.package('swirl'), 'Courses',
'R_Programming_E', 'Looking_at_Data','plant-data.txt')

# Read in data
plants <- read.csv(.datapath, strip.white=TRUE, na.strings="")

# Remove annoying columns
.cols2rm <- c('Accepted.Symbol', 'Synonym.Symbol')
plants <- plants[, !(names(plants) %in% .cols2rm)]

# Make names pretty
names(plants) <- c('Scientific_Name', 'Duration', 'Active_Growth_Period',
'Foliage_Color', 'pH_Min', 'pH_Max','Precip_Min', 'Precip_Max',
'Shade_Tolerance', 'Temp_Min_F')


kable(plants[c(1:15),], caption = "Raw data")
```

Table 16: Raw data

| Scientific_Name | Duration | Active_Growth_Period | Foliage_Color | pH_Min | pH_Max | P |
|---|---|---|---|---|---|---|
| Abelmoschus | NA | NA | NA | NA | NA | |
| Abelmoschus esculentus | Annual, Perennial | NA | NA | NA | NA | |
| Abies | NA | NA | NA | NA | NA | |
| Abies balsamea | Perennial | Spring and Summer | Green | 4.0 | 6.0 | |
| Abies balsamea var. balsamea | Perennial | NA | NA | NA | NA | |
| Abutilon | NA | NA | NA | NA | NA | |
| Abutilon theophrasti | Annual | NA | NA | NA | NA | |
| Acacia | NA | NA | NA | NA | NA | |
| Acacia constricta | Perennial | Spring and Summer | Green | 7.0 | 8.5 | |
| Acacia constricta var. constricta | Perennial | NA | NA | NA | NA | |
| Acalypha | NA | NA | NA | NA | NA | |
| Acalypha gracilens | Annual | NA | NA | NA | NA | |
| Acalypha rhomboidea | Annual | NA | NA | NA | NA | |
| Acalypha virginica | Annual | Spring, Summer, Fall | Green | 5.9 | 7.0 | |
| Acarospora | NA | NA | NA | NA | NA | |

```
## Raw data : too messy data set (because of too many NAs and columns)
## => need to be cleaned for appropriate analysis.

plants.NA.removed <- na.omit(plants)

kable(plants.NA.removed[c(1:15),], caption = "Refined data with NA removed")
```

Table 17: Refined data with NA removed

| | Scientific_Name | Duration | Active_Growth_Period | Foliage_Color | pH_Min | pH_Max | |
|---|---|---|---|---|---|---|---|
| 4 | Abies balsamea | Perennial | Spring and Summer | Green | 4.0 | 6.0 | |
| 9 | Acacia constricta | Perennial | Spring and Summer | Green | 7.0 | 8.5 | |
| 14 | Acalypha virginica | Annual | Spring, Summer, Fall | Green | 5.9 | 7.0 | |
| 17 | Acer negundo | Perennial | Spring and Summer | Green | 5.0 | 7.8 | |
| 19 | Acer nigrum | Perennial | Spring and Summer | Green | 4.5 | 7.3 | |
| 20 | Acer pensylvanicum | Perennial | Spring and Summer | Green | 4.4 | 6.5 | |
| 21 | Acer platanoides | Perennial | Spring and Summer | Green | 4.8 | 7.2 | |
| 22 | Acer pseudoplatanus | Perennial | Spring and Summer | Yellow-Green | 5.8 | 7.0 | |
| 23 | Acer rubrum | Perennial | Spring and Summer | Green | 4.7 | 7.3 | |
| 26 | Acer saccharinum | Perennial | Spring and Summer | Green | 4.0 | 7.3 | |
| 27 | Acer saccharum | Perennial | Spring and Summer | Green | 3.7 | 7.9 | |
| 29 | Acer spicatum | Perennial | Spring and Summer | Green | 4.8 | 7.0 | |
| 31 | Achillea millefolium | Perennial | Spring | Green | 6.0 | 8.0 | |
| 32 | Achillea millefolium var. occidentalis | Perennial | Spring | Green | 6.0 | 8.0 | |
| 43 | Acorus calamus | Perennial | Spring and Summer | Green | 5.2 | 7.2 | |

```
## Statistic : range = plants$pH_Max - plants$pH_Min
## We will use the above statistic value as Dependent Variable(Y)

# plants$pH_Min
# plants$pH_Max
```

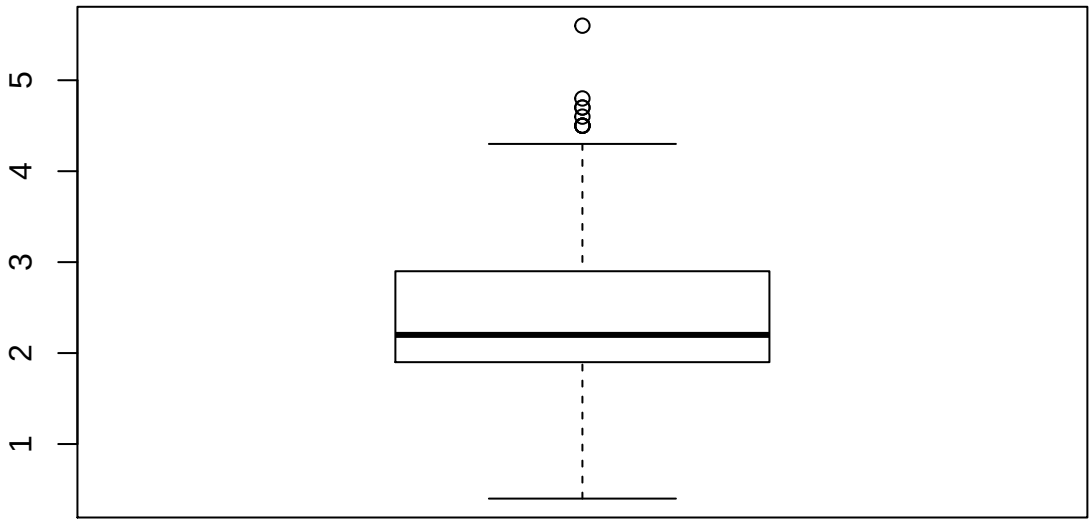
```r
plants.range <- plants.NA.removed$pH_Max - plants.NA.removed$pH_Min

plants.range[c(1:15)]
```

```
##  [1] 2.0 1.5 1.1 2.8 2.8 2.1 2.4 1.2 2.6 3.3 4.2 2.2 2.0 2.0 2.0
```

```r
summary(plants.range)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.400   1.900   2.200   2.347   2.900   5.600
```

```r
boxplot(plants.range)
```




```r
Foliage_Color <- plants.NA.removed$Foliage_Color

## We use this variable as factor or explanatory variable (X) for ANOVA

summary(Foliage_Color)
```

```
##   Dark Green   Gray-Green        Green          Red   White-Gray
##           82           24          675            3            9
## Yellow-Green
##           20
```

```r
## Data cleaning

## What we need is the set of pHs and Foliage_color data set

required.data.set <- data.frame(plants.range,Foliage_Color)

kable(required.data.set[c(1:15),], caption = "Data cleaning for fitting ANOVA")
```

Table 18: Data cleaning for fitting ANOVA

| plants.range | Foliage_Color |
|---|---|
| 2.0 | Green |
| 1.5 | Green |
| 1.1 | Green |
| 2.8 | Green |

| plants.range | Foliage_Color |
|---|---|
| 2.8 | Green |
| 2.1 | Green |
| 2.4 | Green |
| 1.2 | Yellow-Green |
| 2.6 | Green |
| 3.3 | Green |
| 4.2 | Green |
| 2.2 | Green |
| 2.0 | Green |
| 2.0 | Green |
| 2.0 | Green |

```
## With the above cleaned data set, we do the ANOVA.

library(stargazer)
```

```
##
## Please cite as:
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

```
# stargazer(required.data.set)

lm.ph.Foliage.color <- lm(plants.range ~ Foliage_Color,data=required.data.set)

# stargazer(lm.ph.Foliage.color, title="Fitting Liner model Results", align=TRUE, type = "html")

kable(anova(lm.ph.Foliage.color), caption = "ANOVA table")
```

Table 19: ANOVA table

| | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---|---|---|---|---|
| Foliage_Color | 5 | 8.971691 | 1.794338 | 2.974994 | 0.0114063 |
| Residuals | 807 | 486.734016 | 0.603140 | NA | NA |

```
## ANOVA TABLE
## From the above ANOVA table,
## the $H_0 : \beta_1=\beta_2=\beta_3=\beta_4=\beta_5 =0$ has been rejected
## based on the p-value : 0.001141 (based on significance level (alpha) = 0.05).
## It implies there is at least one of the Foliage colors has the significant
## relationship with plants_range



summary(lm.ph.Foliage.color)
```

```
##
## Call:
## lm(formula = plants.range ~ Foliage_Color, data = required.data.set)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7665 -0.4665 -0.1444  0.5250  3.2335
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 2.17683    0.08576  25.382   <2e-16 ***
## Foliage_ColorGray-Green     0.46484    0.18024   2.579   0.0101 *
## Foliage_ColorGreen          0.18969    0.09082   2.089   0.0371 *
## Foliage_ColorRed           -0.31016    0.45651  -0.679   0.4971
## Foliage_ColorWhite-Gray     0.46762    0.27271   1.715   0.0868 .
## Foliage_ColorYellow-Green -0.20183    0.19368  -1.042   0.2977
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7766 on 807 degrees of freedom
## Multiple R-squared:  0.0181, Adjusted R-squared:  0.01202
## F-statistic: 2.975 on 5 and 807 DF,  p-value: 0.01141
```

```
## Summary table
## From the above Summary table, we can figure out that
## between (Foliage-Color-Gray-Green and Plants_range),
## (Foliage-ColorGreen and Plants_range),
## there are significant relationship
## under the p-value 0.0101, 0.0371  respectively.
## (based on significance level (alpha) = 0.05)
```

**Problem 6**

**Finish this homework by pushing your changes to your repo. In general, your workflow for this should be:**

1. git pull – to make sure you have the most recent repo
2. In R: do some work
3. git add – this tells git to track new files
4. git commit – make message INFORMATIVE and USEFUL
5. git push – this pushes your local changes to the repo

I did the above procedure !!!