HW2_Song

Due September 16, 2020

Xinyi Song

2020-09-15

For each assignment, turn in by the due date/time. Late assignments must be arranged prior to submission. In every case, assignments are to be typed neatly using proper English in Markdown.

In Lectures 2 and 3, we spoke about Reproducible Research, R and version control, getting, cleaning, munging and 'tidying' data and finally, summarizing data. Again, we are focusing on Reproducible Analysis which, for us, is accomplished by mixing code, figures and text into a cohesive document that fully describes both the process we took to go from data to results and the rational behind our data driven conclusions. In this exercise we begin creating tidy data sets. While others have proposed standards for sharing data with statiticians, as practicing data scientists, we realize the often onerous task of getting, cleaning and formatting data is usually in our hands. From here on out, we will use GitHub to retrieve and turn in the homework assignments.

Problem 1

Work through Primers titled "Work with Data" and "Tidy Your Data".

For a different take on how to use R, try swirl:

```
library(swirl)
install_course("The R Programming Environment")
install_course("Exploratory_Data_Analysis")
swirl()
```

These can be a little cheezy, but are overall a pretty decent intro. Each one takes 5-10 min. Pick and choose as you like. Don't worry about submitting anything at the end, these were used for credits in a Coursera Data Science Course.

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_pid, i.e. for me it would be HW2_rsettlag

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

Problem 3

In the lecture, there were two links to StackOverflow questions on why one should use version control. In your own words, summarize your thoughts (2-3 sentences) on version control in your future work. No penalties here if you say, useless!

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 dataset, you should perform the cleaning 2x: first with base R functions (ie no dplyr, piping, etc), second using tidyverse function. Make sure you weave your code and text into a complete description of the process and end by creating a tidy dataset describing the variables, create a summary table of the data (summary, NOT full listing), note issues with the data, and include an informative plot.

a. Sensory data from five operators. – see video, I am doing this one
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

`Solution`

For the sensory dataset, there are five operators, and I noticed that it has NA values once for every three observations. And since the first line is irrelevant to our analysis, I use 'skip' parameter to avoid reading it.

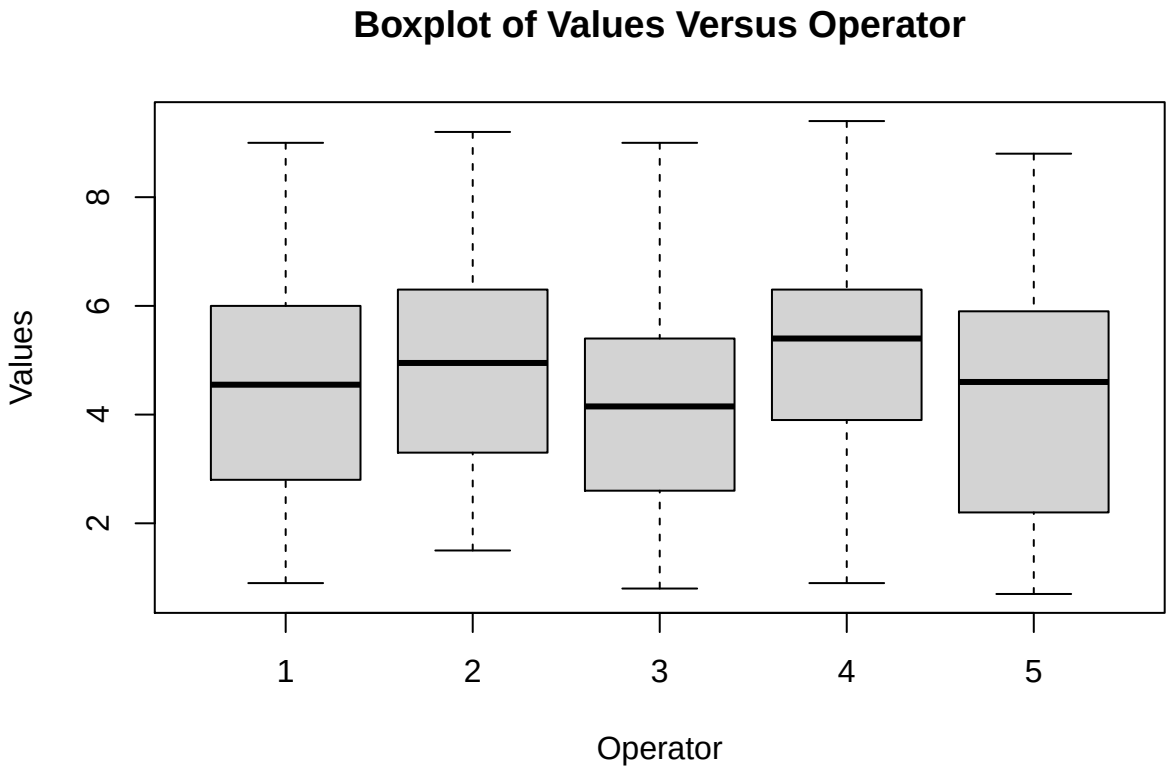
Since I did not find detailed summary of this dataset, from my comprehension, I think for each observation, it has three columns which indicates five operators. And for item, for each line without NA value, the first column of the line corresponds to 'item', after sorting, I found 10 items.

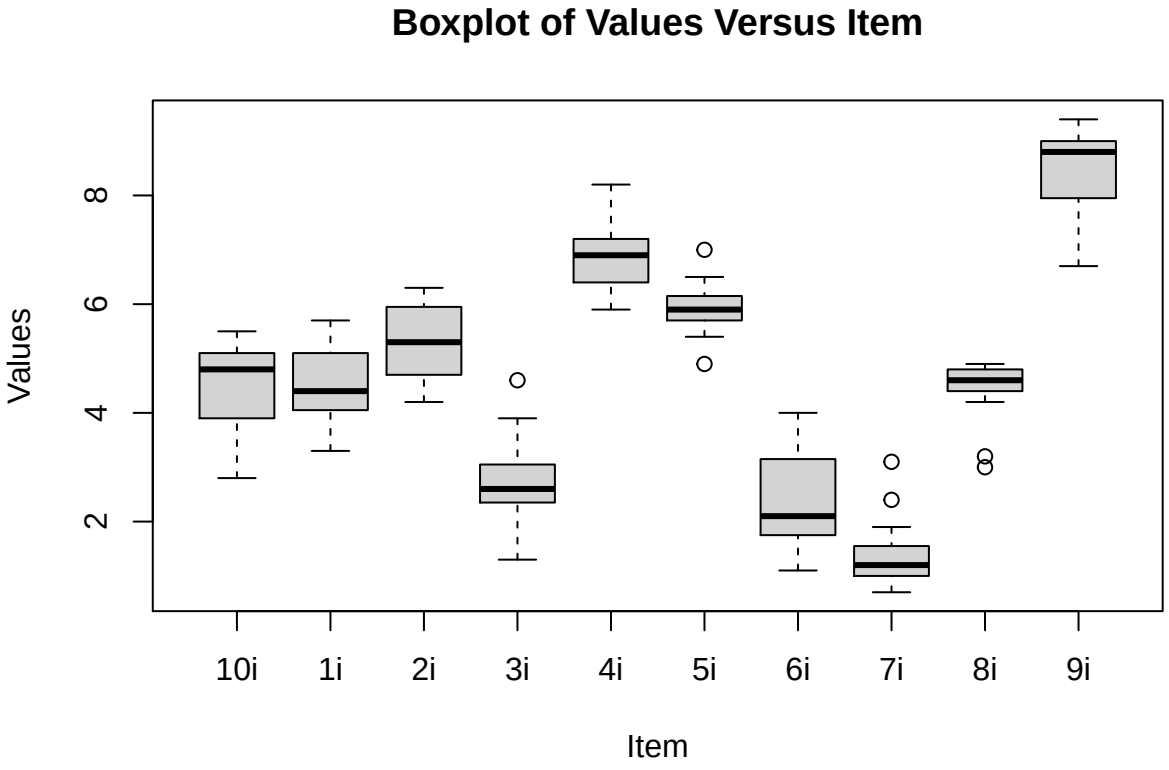
In summary, each observation has columns (five operators), and there are total 10 items, for each item, it has three observations.

```
library(data.table)
# Sensor Data
# Stack Method
url_sensor <- 'http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat'
lines<- fread(url_sensor,fill = TRUE, skip =1, header = TRUE)
# decide num of items
item_kind = NULL
dat = matrix(0, nrow(lines), ncol(lines)-1)
for (i in 1:nrow(lines)){
  if (sum(is.na(lines[i,]))>0) {
    dat[i,] = unlist(lines[i,1:5])
  } else {
    dat[i,] = unlist((lines[i,2:6]))
    item_kind = c(item_kind,lines[i,1])
    }
}
dat = as.data.frame(dat)
item = rep(seq_len(length(item_kind)), each = nrow(lines)/length(item_kind))
dat$item = paste(item, 'i',sep='')
colnames(dat) = c('1','2','3','4','5','item')
sensor_data_tidy = data.frame(item=dat[,6],stack(dat[,-6]))
colnames(sensor_data_tidy) = c('item', 'values', 'operator')
sensor_data_tidy$item = as.factor(sensor_data_tidy$item)
sensor_data_tidy$operator = as.factor(sensor_data_tidy$operator)
```

We have converted the data frame to tidy data frames using the base functions. Here is the summary of data:

item	values	operator
10i :15	Min. :0.700	1:30
1i :15	1st Qu.:3.025	2:30
2i :15	Median :4.700	3:30
3i :15	Mean :4.657	4:30
4i :15	3rd Qu.:6.000	5:30
5i :15	Max. :9.400	NA
(Other):60	NA	NA





The following method is by using package tidyverse:

```
# Stack and Fix Columns using tidyverse
library(data.table)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

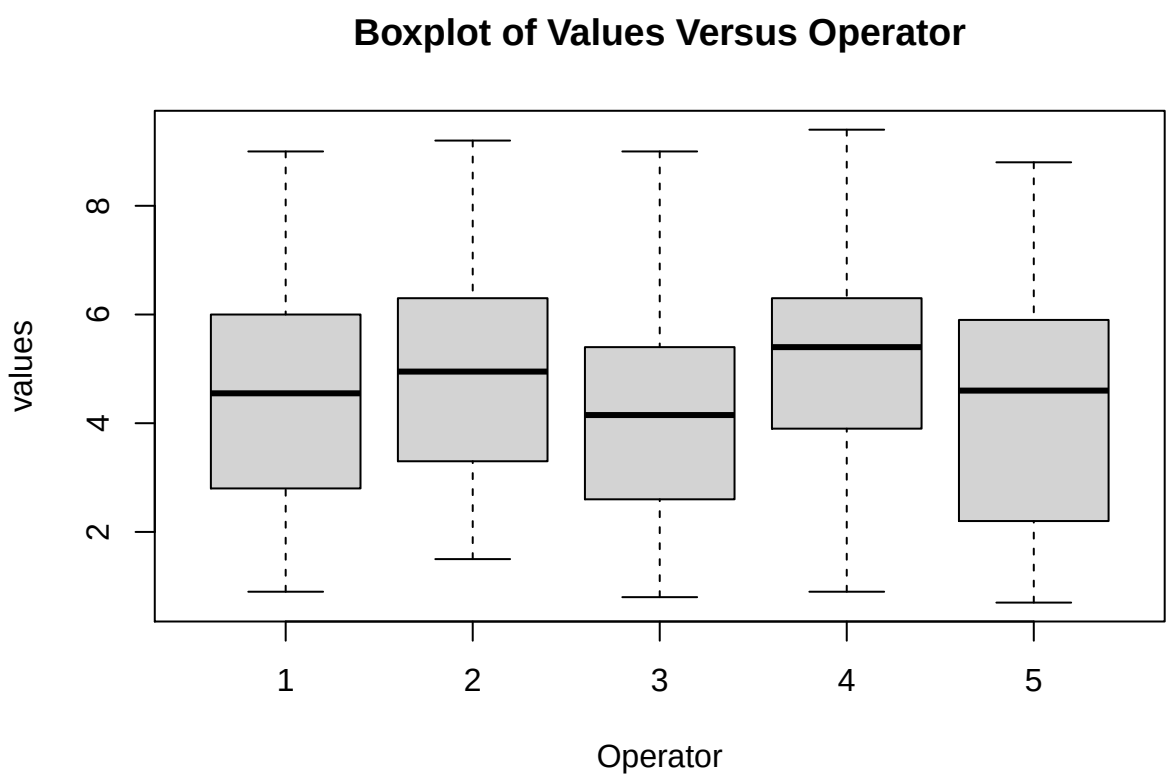
```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()
```

```
dat$item = as.factor(dat$item)
sensor_tidy_tv <- dat %>% gather(key = 'method', value = 'value', 1:5)
summary(sensor_tidy_tv)
```

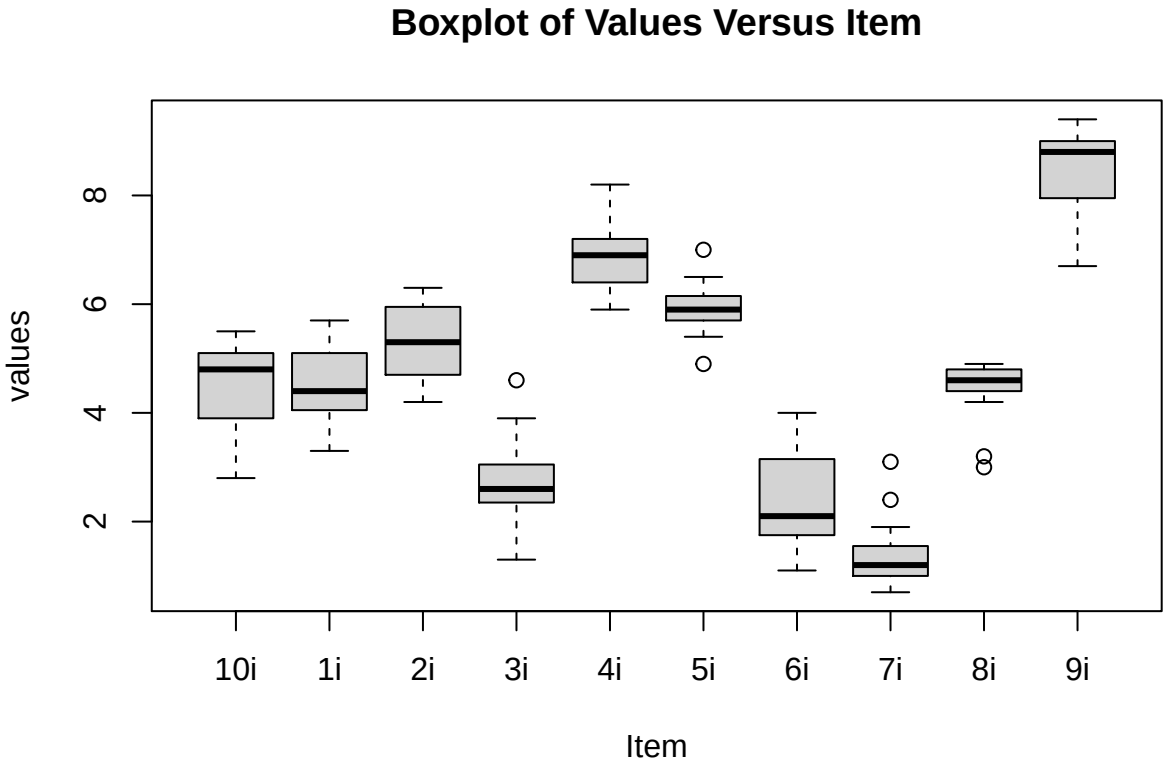
```
##       item       method              value
##  10i    :15   Length:150         Min.   :0.700
##  1i     :15   Class :character   1st Qu.:3.025
##  2i     :15   Mode  :character   Median :4.700
##  3i     :15                      Mean   :4.657
##  4i     :15                      3rd Qu.:6.000
##  5i     :15                      Max.   :9.400
##  (Other):60
```

item	method	value
10i :15	Length:150	Min. :0.700
1i :15	Class :character	1st Qu.:3.025
2i :15	Mode :character	Median :4.700
3i :15	NA	Mean :4.657
4i :15	NA	3rd Qu.:6.000
5i :15	NA	Max. :9.400
(Other):60	NA	NA

```
boxplot(sensor_tidy_tv$value ~ sensor_tidy_tv$method, xlab = 'Operator', ylab = 'values', main = 'Boxpl
```




```
boxplot(sensor_tidy_tv$value ~ sensor_tidy_tv$item, xlab = 'Item', ylab = 'values', main = 'Boxplot of '
```



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

Solution

For this dataset, in addition to missing values (NA value), when reading data with fread function, it automatically let 'long jump' variable become 'long' and jump, which leads to extra columns with NA values.

For cleaning this dataset, I just select the column and rename the colmun of dataframe, also, remove the observations with missing value.

Also, the year here is coded as 1900 = 0, I just add 1900 for year of each observation to reset it.

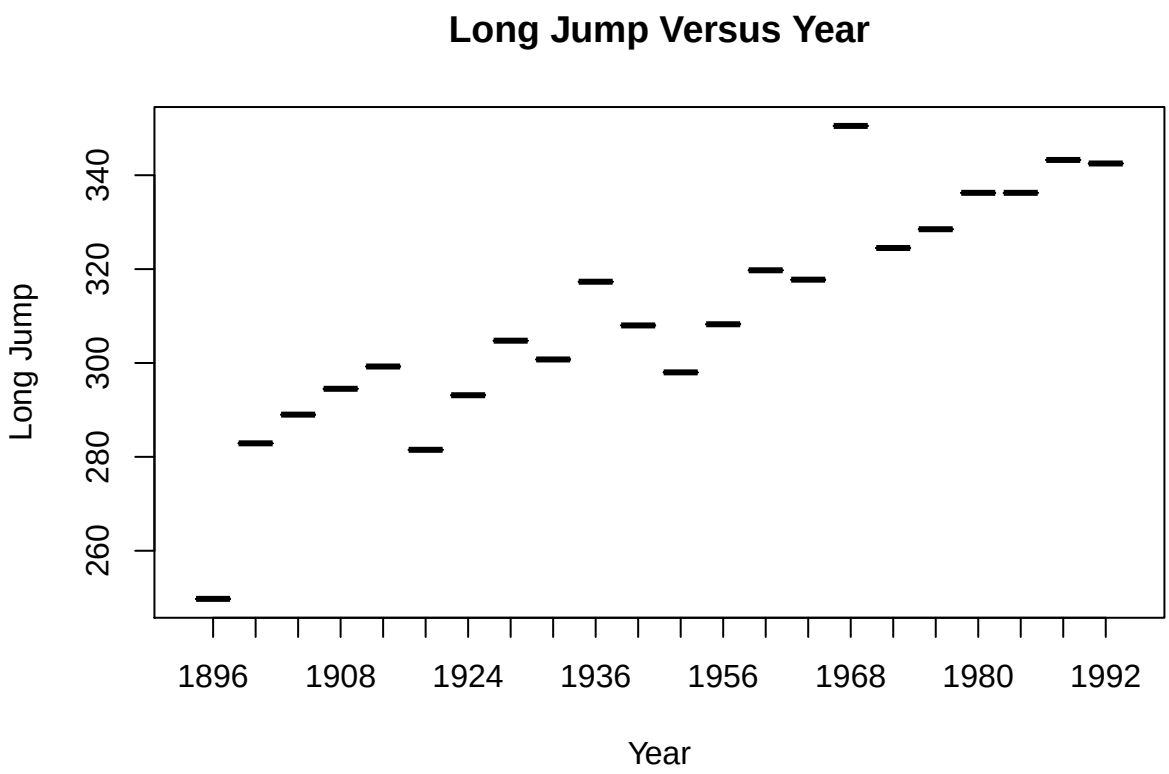
```
long_jump <- 'http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/LongJumpData.dat'
lines<- fread(long_jump,fill = TRUE, header = TRUE)
lines = lines[,-(9:12)]
colnames(lines) = rep(c('Year','Long Jump'),4)
S_1 = as.data.frame(cbind(as.matrix(lines[[1]]),as.matrix(lines[[2]])))
colnames(S_1) = c('Year', 'Long Jump')
S_2 = as.data.frame(cbind(as.matrix(lines[[3]]),as.matrix(lines[[4]])))
colnames(S_2) = c('Year', 'Long Jump')
S_3= as.data.frame(cbind(as.matrix(lines[[5]]),as.matrix(lines[[6]])))
colnames(S_3) = c('Year', 'Long Jump')
S_4= as.data.frame(cbind(as.matrix(lines[[7]]),as.matrix(lines[[8]])))
colnames(S_4) = c('Year', 'Long Jump')
DAT_longjump_base = rbind(S_1,S_2,S_3,S_4[-(nrow(S_4)-1):-nrow(S_4),])
DAT_longjump_base$Year = DAT_longjump_base$Year + 1900
```

```
knitr::kable(summary(DAT_longjump_base))
```

Year	Long Jump
Min. :1896	Min. :249.8

Year	Long Jump
1st Qu.:1921	1st Qu.:295.4
Median :1950	Median :308.1
Mean :1945	Mean :310.3
3rd Qu.:1971	3rd Qu.:327.5
Max. :1992	Max. :350.5

```
boxplot(DAT_longjump_base$`Long Jump`~DAT_longjump_base$Year, xlab = 'Year', ylab = 'Long Jump', main =
```



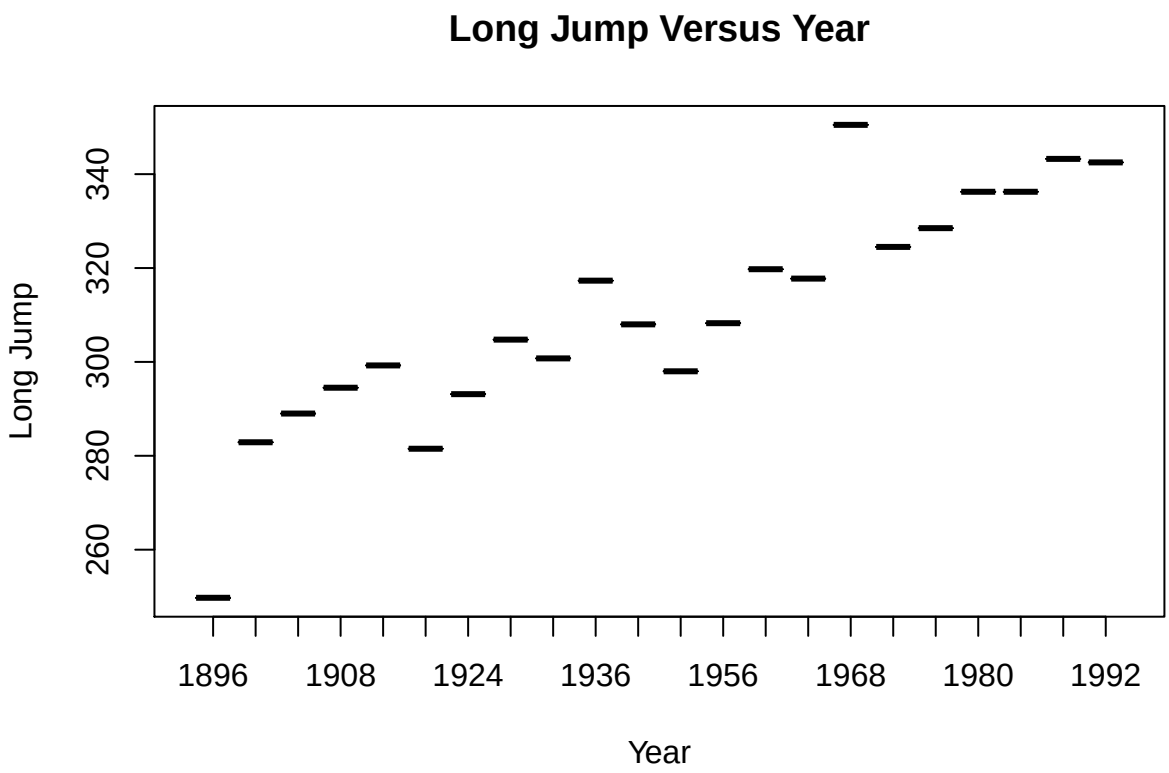
The following method is by using tidyverse method:

```
library(data.table)
library(tidyverse)
longjump_tv_S1 <- S_1%>% gather(key = 'Year', value = 'Long Jump')
longjump_tv_S2 <- S_2%>% gather(key = 'Year', value = 'Long Jump')
longjump_tv_S3 <- S_3%>% gather(key = 'Year', value = 'Long Jump')
longjump_tv_S4 <- S_4%>% gather(key = 'Year', value = 'Long Jump')
longjump_tv <- as.data.frame(rbind(longjump_tv_S1, longjump_tv_S2, longjump_tv_S3, longjump_tv_S4[(-nrow
longjump_tv$Year = longjump_tv$Year + 1900
```

```
knitr::kable(summary(longjump_tv))
```

Year	Long Jump
Min. :1896	Min. :249.8
1st Qu.:1921	1st Qu.:295.4
Median :1950	Median :308.1
Mean :1945	Mean :310.3
3rd Qu.:1971	3rd Qu.:327.5
Max. :1992	Max. :350.5

```
boxplot(longjump_tv$`Long Jump`~longjump_tv$Year, xlab = 'Year', ylab = 'Long Jump', main = 'Long Jump V
```



Based on the output above, we can see that as time goes by, the values of long jump increases although there still exist fluctuations.

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

Solution

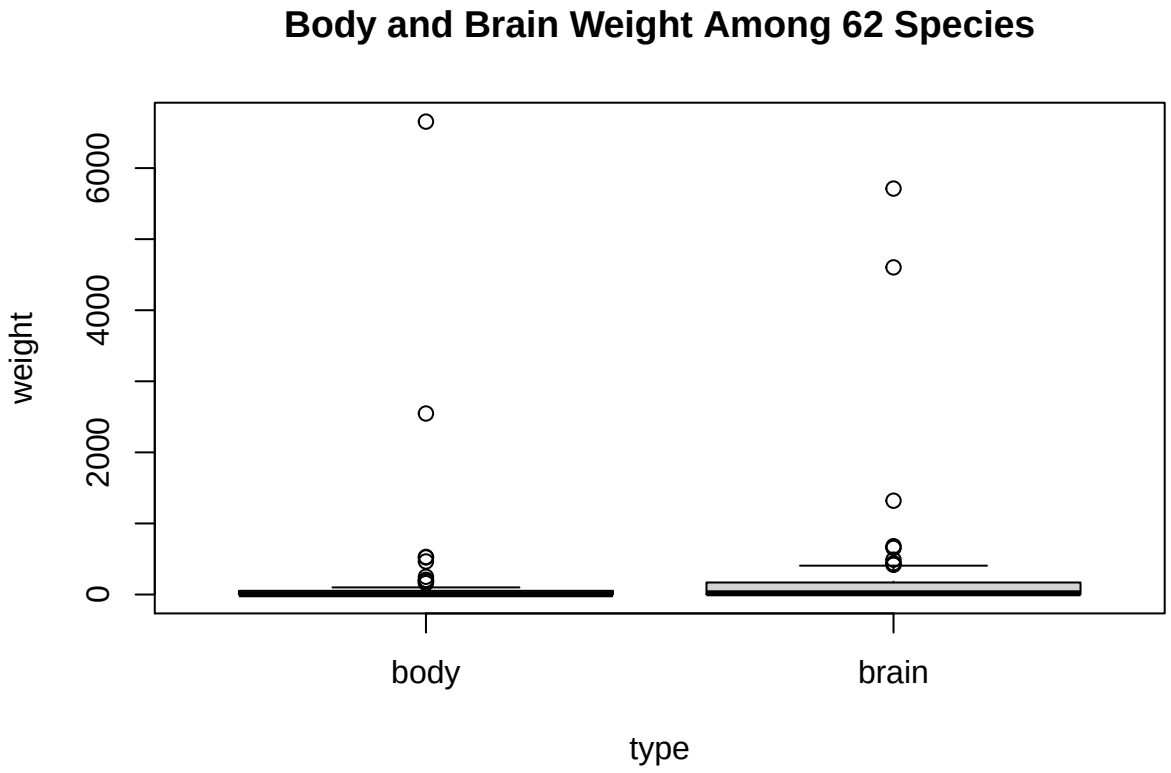
For this dataset, the fread function automatically let Wt become separate column which leands to six more extra columns with NA value. The column name should be 'Body Wt' and 'Brain Wt'. After removing observation with missing values and extra six columns, I used stack function to deal with each two columns corresponding to Body and Brain and then put them together.

```
body_weight <-'http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/BrainandBodyWeight.dat'
lines<- fread(body_weight ,fill = TRUE, header = TRUE)
lines = lines[,(-7):(-12)]
S_1 = as.data.frame(cbind(as.matrix(lines[[1]]),as.matrix(lines[[2]])))
colnames(S_1) = c('body', 'brain')
S_1 = stack(S_1)
S_1$species = c(rep(1:(nrow(S_1)/2),2))
S_2 = as.data.frame(cbind(as.matrix(lines[[3]]),as.matrix(lines[[4]])))
colnames(S_2) = c('body', 'brain')
S_2 = stack(S_2)
S_2$species = c(rep((nrow(S_1)/2+1):((nrow(S_1)/2+1)+nrow(S_2)/2-1),2))
S_3 = as.data.frame(cbind(as.matrix(lines[[5]]),as.matrix(lines[[6]])))
colnames(S_3) = c('body', 'brain')
S_3 = stack(S_3[-21,])
S_3$species = c(rep(43:62,2))
bodyweight_base = rbind(S_1, S_2, S_3)
```

```
knitr::kable(summary(bodyweight_base))
```

values	ind	species
Min. : 0.005	body :62	Min. : 1.0
1st Qu.: 1.388	brain:62	1st Qu.:16.0
Median : 7.450	NA	Median :31.5
Mean : 240.962	NA	Mean :31.5
3rd Qu.: 98.650	NA	3rd Qu.:47.0
Max. :6654.000	NA	Max. :62.0

```
boxplot(bodyweight_base$values~bodyweight_base$ind, xlab = 'type', ylab = 'weight', main = 'Body and Bra
```



The following method is by using package tidyverse:

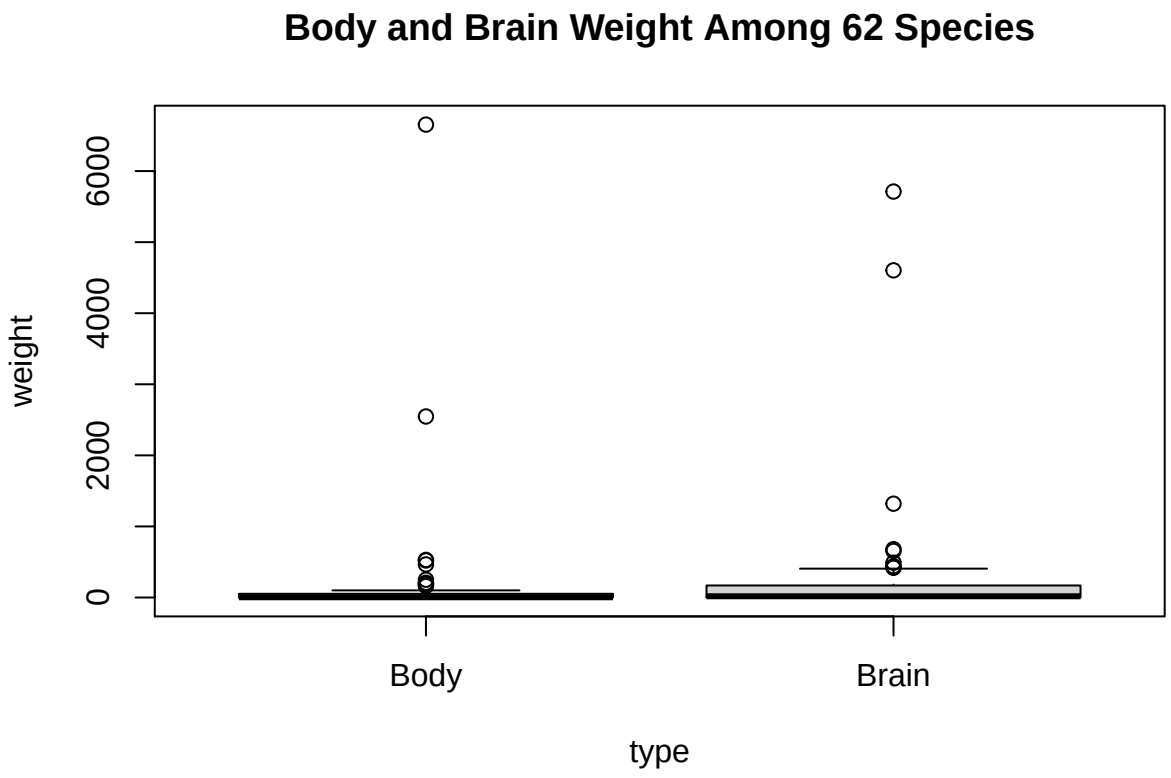
```
lines<- fread(body_weight ,fill = TRUE, header = TRUE)
lines = lines[,(-7):(-12)]
colnames(lines) = rep('names',6)
Body = as.matrix(rbind(lines[,1],lines[,3], lines[,5]))
colnames(Body) = 'val'
Body = as.matrix(Body[-nrow(Body)])
Brain = as.matrix(rbind(lines[,2], lines[,4], lines[,6]))
Brain = as.matrix(Brain[-nrow(Brain)])
colnames(Brain) = 'val'
dat =as.data.frame(rbind(Body,Brain))
dat$type = c(rep('Body', length(Body)), rep('Brain', length(Brain)))
colnames(dat) = c('value', 'method')
dat$method = as.factor(dat$method)
bodyweight_tidy_tv <- dat%>% gather(key = 'method', value = 'value')
```

```
bodyweight_tidy_tv$species = rep(1:62,2)
summary(bodyweight_tidy_tv)
```

```
##      value            method      species
##  Min.   :   0.005   Body :62   Min.   : 1.0
##  1st Qu.:   1.388   Brain:62   1st Qu.:16.0
##  Median :   7.450              Median :31.5
##  Mean   : 240.962              Mean   :31.5
##  3rd Qu.:  98.650              3rd Qu.:47.0
##  Max.   :6654.000              Max.   :62.0
```

```
knitr::kable(summary(bodyweight_tidy_tv))
```

value	method	species
Min. : 0.005	Body :62	Min. : 1.0
1st Qu.: 1.388	Brain:62	1st Qu.:16.0
Median : 7.450	NA	Median :31.5
Mean : 240.962	NA	Mean :31.5
3rd Qu.: 98.650	NA	3rd Qu.:47.0
Max. :6654.000	NA	Max. :62.0



Based on the output above, we can see that among the 62 species in the dataset, there does not exsit much difference of weight among the brain and body.

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

`Solution`

For the dataset 'tomato', there are two varieties of tomatoes: 'Ife\#1' and 'PusaEarlyDwarf'. The three

planting densities corresponds to '10000', '20000' and '30000'.

By using split function, I tried to stack it to a tidy data and do dexcriptive analysis.

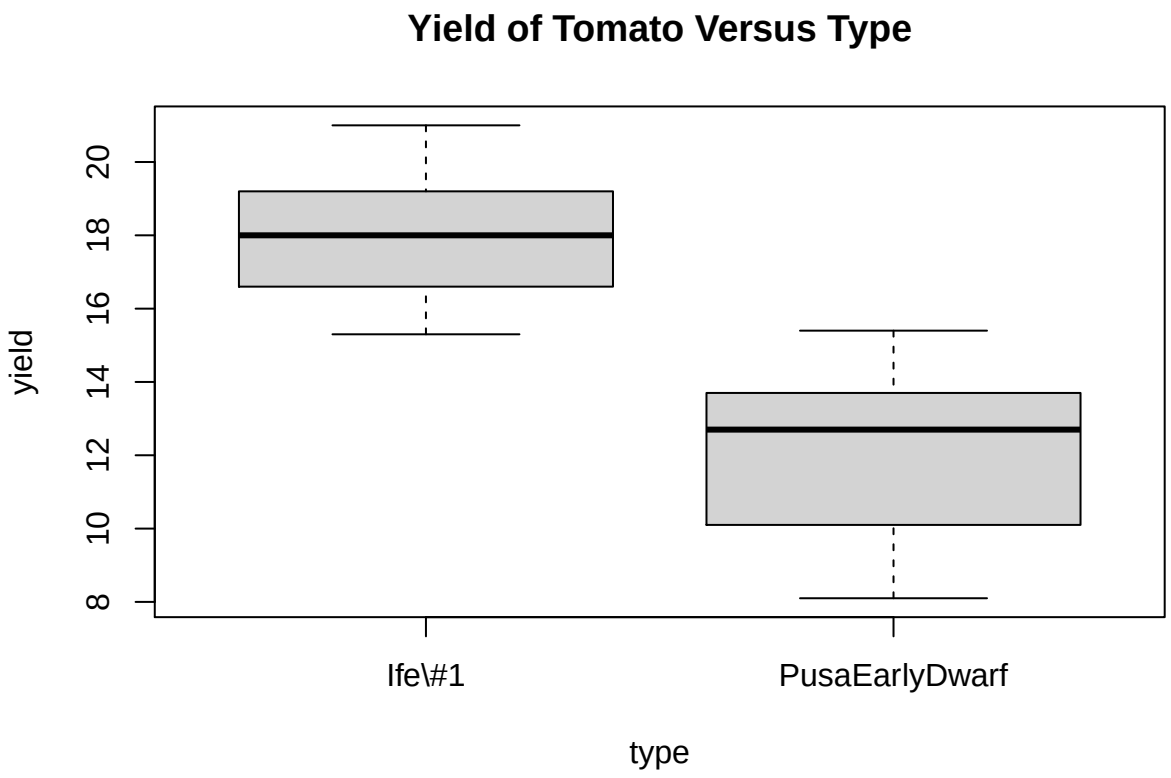
```
# Base
url_tomato <- 'https://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/tomato.dat'
lines<- fread(url_tomato, header = TRUE,strip.white=TRUE,skip = 1, fill = TRUE)
Ife <- split(lines, lines[,1])[[1]][,-1]
PusaEarlyDwarf <- unlist(split(lines, lines[,1])[[2]][,-1])
Ife = strsplit(unlist(split(Ife, ' ')),',')
PusaEarlyDwarf = strsplit(unlist(split(PusaEarlyDwarf, ' ')),',')
IFE_DAT = matrix(0,3,3)
PUSA_DAT = matrix(0,3,3)
for (i in 1:length(Ife)){
  tmp = strsplit(unlist(Ife[i]), ',')
  tmmp = strsplit(unlist(PusaEarlyDwarf[i]), ',')
  IFE_DAT[,i]= cbind(as.numeric(tmp[[1]]), as.numeric(tmp[[2]]), as.numeric(tmp[[3]]))
  PUSA_DAT[,i] = rbind(as.numeric(tmmp[[1]]), as.numeric(tmmp[[2]]), as.numeric(tmmp[[3]]))
  }
DAT = as.data.frame(rbind(IFE_DAT,PUSA_DAT))
DATA_TOMATO_BASE = matrix(0,3*3*2,3)
DATA_TOMATO_BASE[,1] = rbind(as.vector(DAT[,1]),as.vector(DAT[,2]),as.vector(DAT[,3]))
DATA_TOMATO_BASE = as.data.frame(DATA_TOMATO_BASE)
DATA_TOMATO_BASE[,2] = rep(c('10000', '20000','30000'), 3*3*2/3)
DATA_TOMATO_BASE[,3] = c(rep('Ife\\#1', 9), rep('PusaEarlyDwarf', 9))
colnames(DATA_TOMATO_BASE) = c('values', 'density', 'type')
DATA_TOMATO_BASE$density = as.factor(DATA_TOMATO_BASE$density)
DATA_TOMATO_BASE$type = as.factor(DATA_TOMATO_BASE$type)
summary(DATA_TOMATO_BASE)
```

```
##      values       density             type
##  Min.   : 8.10   10000:6   Ife\\#1       :9
##  1st Qu.:12.95   20000:6   PusaEarlyDwarf:9
##  Median :15.35   30000:6
##  Mean   :15.07
##  3rd Qu.:17.88
##  Max.   :21.00
```

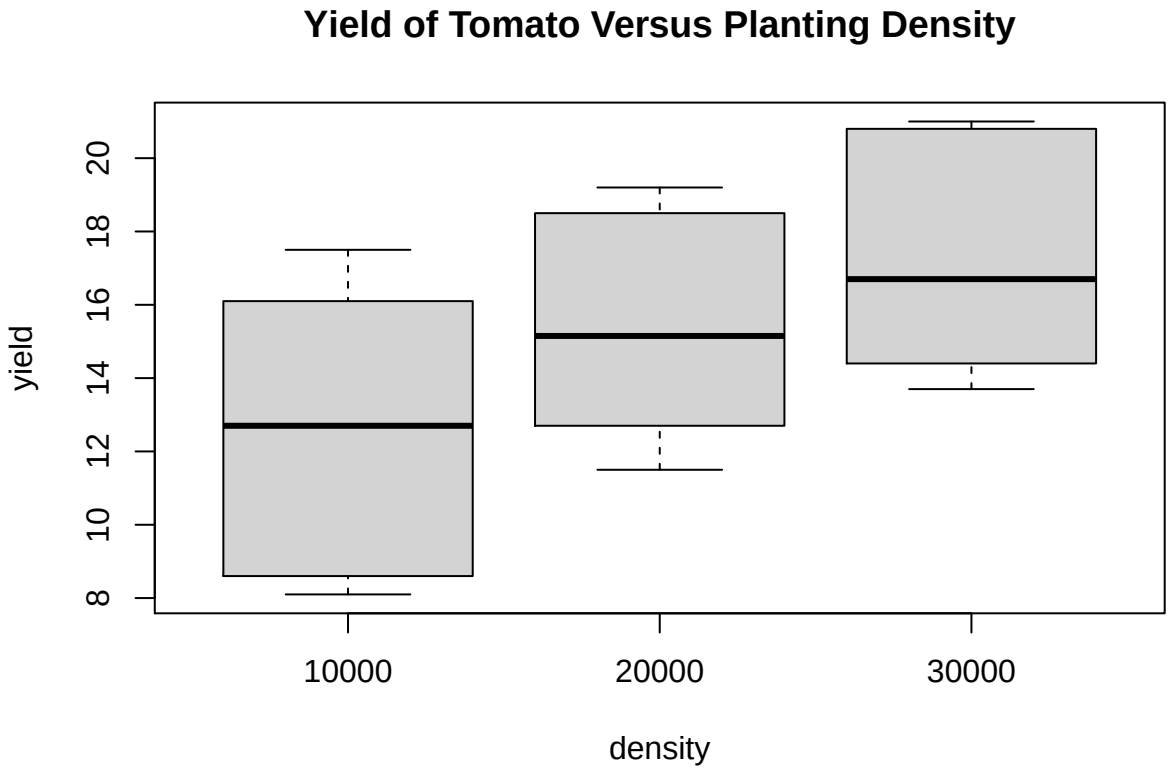
```
knitr::kable(summary(DATA_TOMATO_BASE))
```

values	density	type
Min. : 8.10	10000:6	Ife#1 :9
1st Qu.:12.95	20000:6	PusaEarlyDwarf:9
Median :15.35	30000:6	NA
Mean :15.07	NA	NA
3rd Qu.:17.88	NA	NA
Max. :21.00	NA	NA

```
boxplot(DATA_TOMATO_BASE$values~DATA_TOMATO_BASE$type, xlab = 'type', ylab = 'yield', main = 'Yield of '
```




```
boxplot(DATA_TOMATO_BASE$values~DATA_TOMATO_BASE$density, xlab = 'density', ylab = 'yield', main = 'Yie
```



The following method is by using package tidyverse:

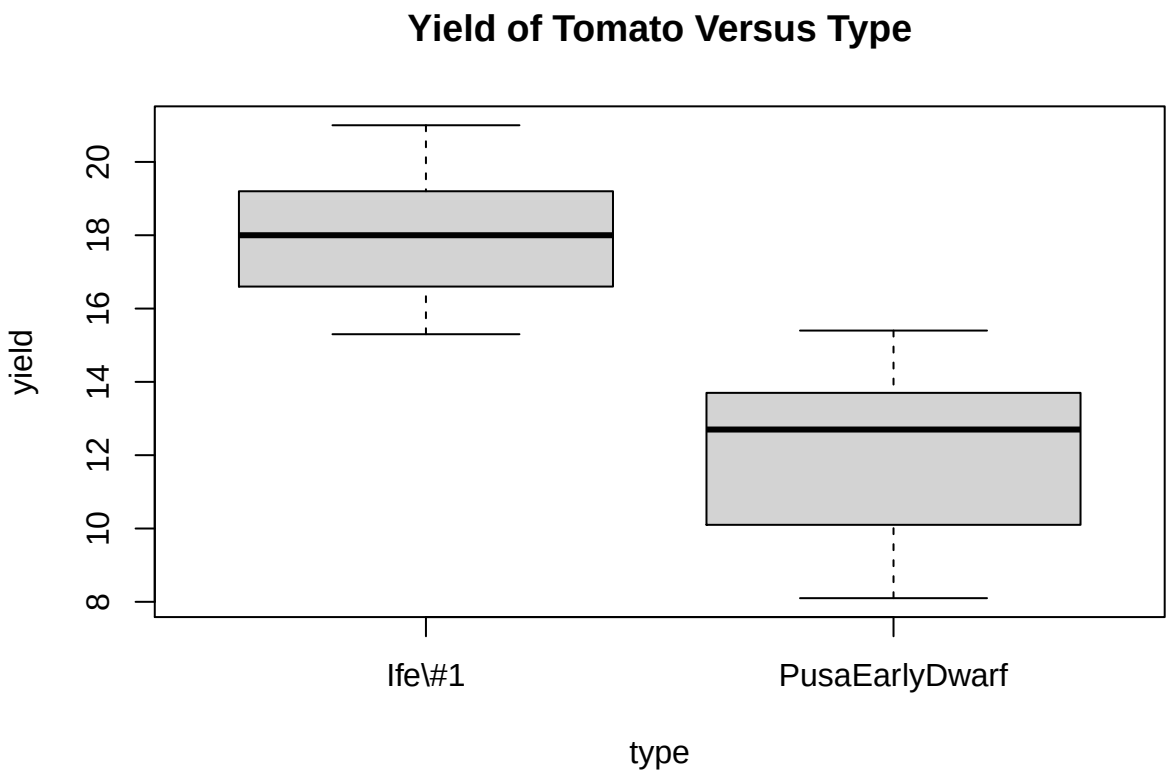
```
DAT = as.data.frame(DAT)
DAT$type = c(rep('Ife\\#1',3), rep('PusaEarlyDwarf', 3))
```

```
colnames(DAT) = c('10000', '20000', '30000','type')
DAT_TOMATO_RAW = data.frame(rep(DAT$type), stack(DAT[,-4]))
colnames(DAT_TOMATO_RAW) = c('type', 'yield', 'density')
DAT_TOMATO_RAW$type = (DAT_TOMATO_RAW$type)
DAT_TOMATO_RAW$density = (DAT_TOMATO_RAW$density)
DAT_TOMATO_RAW$yield = as.numeric(DAT_TOMATO_RAW$yield)
DAT_TOMATO <-DAT_TOMATO_RAW%>%group_by(density)%>%gather(key = 'density',value='yield',-type) %>%slice(
DAT_TOMATO_density <-DAT_TOMATO_RAW%>%group_by(density)%>%gather(key = 'density',value='yield',-type) %
DAT_TOMATO['density'] = DAT_TOMATO_density['yield']
DAT_TOMATO_TV = DAT_TOMATO
DAT_TOMATO_TV[['yield']] = as.numeric(DAT_TOMATO_TV[['yield']])
DAT_TOMATO_TV[['type']] = as.factor(DAT_TOMATO_TV[['type']])
DAT_TOMATO_TV[['density']] = as.factor(DAT_TOMATO_TV[['density']])
```

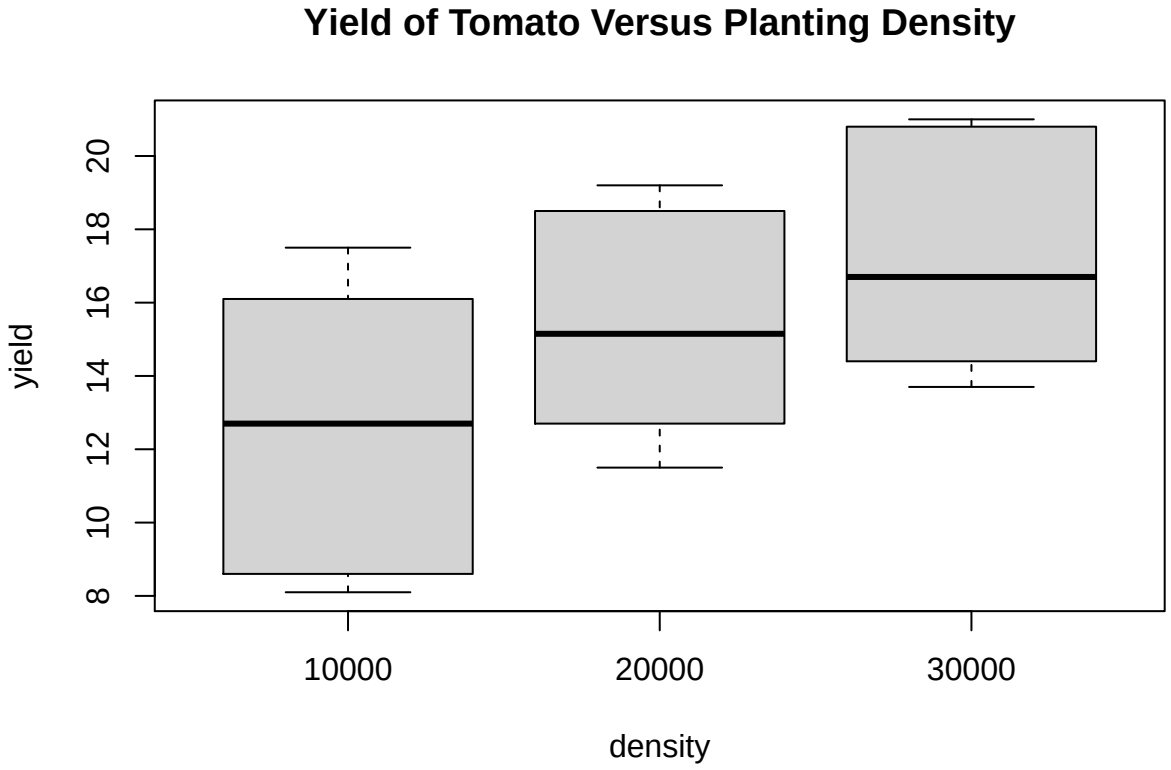
```
knitr::kable(summary(DAT_TOMATO_TV))
```

type	density	yield
Ife#1 :9	10000:6	Min. : 8.10
PusaEarlyDwarf:9	20000:6	1st Qu.:12.95
NA	30000:6	Median :15.35
NA	NA	Mean :15.07
NA	NA	3rd Qu.:17.88
NA	NA	Max. :21.00

```
boxplot(DAT_TOMATO_TV$yield~DAT_TOMATO_TV$type, xlab = 'type', ylab = 'yield', main = 'Yield of Tomato
```




```
boxplot(DAT_TOMATO_TV$yield~DAT_TOMATO_TV$density, xlab = 'density', ylab = 'yield', main = 'Yield of T
```



Based on the results above, we can see that other things being equal, on average, the `Ife\#1` type of tomato has higher yield with less variance than that of 'PusaEarlyDwarf' tomato.

Also, it is obvious that other things being equal, the yield of tomatoes increases as the planting density increases.

Problem 5

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

If you have difficulty with steps 1-5, git is not correctly or completely setup. See me for help.

Only submit the .Rmd and .pdf solution files. Names should be formatted HW2_lastname.Rmd and HW2_lastname.pdf

Optional preperation for next class:

TBD