Homework 2

Ankur Patel

9/16/2020

Problem 3:

In my future work, I think version control will be useful mostly in collaborative settings because it will enable my team members and myself to keep track of each other's changes. However, if I am the sole person working on a script I may not make use of it as I will keep track of changes through comments directly on the script. I think that it is mostly useful when a script is being worked on by at least 2 people and is a good safeguard against fatal coding changes.

Problem 4:

(a) We are looking at the sensory data from five operators from the Wu and Hamada book. First, we will get the raw data using the URL.

```
#Code to get "http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat"
sensory_url <- "http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat"
sensory_data_raw <- fread(sensory_url,skip = 2,data.table = FALSE,header = FALSE, fill = TRUE)
```

Now we will tidy the sensory data using base R.

```
#create an items_column which will be on the left side of the dataframe
items_column <- vector(length = 0)
for (i in 1:10)
{
  items_column <- append(items_column, rep(i,3))
}
#the item numbers are woven into the first column so we replace them with NA's #using the two lines of
first_col_clean <- sensory_data_raw$V1
first_col_clean[seq(1,30,3)] <- NA
sensory_data_tidy_baseR <- cbind(first_col_clean,sensory_data_raw[,2:6])
#we store the rows to fix in rows_to_fix
rows_to_fix <- seq(1,30,3)
#the for loop below shifts the rows to fix one entry to the left and the last entry is NA, ensuring tha
for (i in 1:length(rows_to_fix))
{
  curr_row <- rows_to_fix[i]
  cleaned_row <- c(sensory_data_tidy_baseR[curr_row,2:6],NA)
  sensory_data_tidy_baseR[curr_row,] <- cleaned_row
}
#rename the first column
names(sensory_data_tidy_baseR)[1] <- "V1"
#drop the last column since it is all NA
sensory_data_tidy_baseR <- sensory_data_tidy_baseR[,1:5]
```

```
#bind the items_column and rename the columns
sensory_data_tidy_baseR <- cbind(items_column,sensory_data_tidy_baseR)
names(sensory_data_tidy_baseR)<- c("Item","1","2","3","4","5")
#the current form of sensory_data_tidy_baseR is what we will use for the tidyverse version so we make a
sensory_data_tidy_tidyverse <- copy(sensory_data_tidy_baseR)
#operator_column repeats the operator values in a sequence 1,...,5 30 times which will be one of the co
operator_column <- rep(seq(1,5),30)
#the measurement column turns the  data in the rows into a vector
measurement_column <- vector(length = 0)
for (i in 1:dim(sensory_data_tidy_baseR)[1])
{
  #unlist is used to turn the rows into vectors
  measurement_column <- append(measurement_column,unlist(sensory_data_tidy_baseR[i,2:6],use.names = FALS
}
#the final items column should have 150 entries
items_column_final <- vector(length = 0)
for (i in 1:10)
{
  items_column_final <- append(items_column_final, rep(i,15))
}
sensory_data_tidy_baseR <- data.frame(cbind(items_column_final,operator_column,measurement_column))
names(sensory_data_tidy_baseR) <-c("Item", "Operator","Data")
print(head(sensory_data_tidy_baseR))
```

```
##   Item Operator Data
## 1    1        1  4.3
## 2    1        2  4.9
## 3    1        3  3.3
## 4    1        4  5.3
## 5    1        5  4.4
## 6    1        1  4.3
```

Now we will tidy the sensory data using tidyverse

```
sensory_data_tidy_tidyverse <- sensory_data_tidy_tidyverse %>% gather(key = "Operator", value = "Data",
print(head(sensory_data_tidy_tidyverse))
```

```
##   Item Operator Data
## 1    1        1  4.3
## 2    1        1  4.3
## 3    1        1  4.1
## 4    2        1  6.0
## 5    2        1  4.9
## 6    2        1  6.0
```

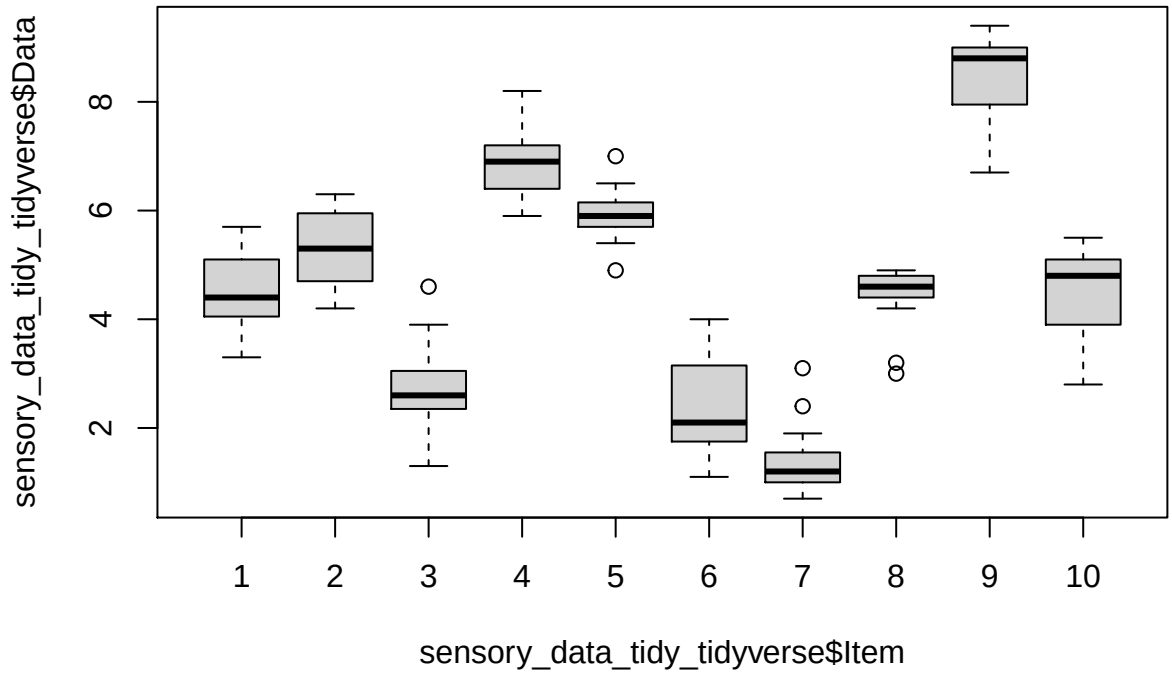
Now we will do a summary and plot of the sensory data.

```
knitr::kable(summary(sensory_data_tidy_tidyverse))
```

Item	Operator	Data
Min. : 1.0	Length:150	Min. :0.700
1st Qu.: 3.0	Class :character	1st Qu.:3.025
Median : 5.5	Mode :character	Median :4.700
Mean : 5.5	NA	Mean :4.657
3rd Qu.: 8.0	NA	3rd Qu.:6.000

Item	Operator	Data
Max. :10.0	NA	Max. :9.400

```
boxplot(sensory_data_tidy_tidyverse$Data~sensory_data_tidy_tidyverse$Item)
```



(b)

Now we will look at the olympic data. First we will read in the raw data from the URL.

```
#Assign the url to a variable
olympic_url <- "http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/LongJumpData.dat"
#Read in the raw data
olympic_data_raw <- fread(olympic_url,skip = 1,data.table = FALSE,header = FALSE, fill = TRUE)
```

Next we will tidy the olympic data using base R.

```
#years_columns are the positions of the columns that have year values
years_columns <- c(1,3,5,7)
#jump_columns are the positions of the columns that have values for the long jump
jump_columns <- c(2,4,6,8)
#standard_year is the standardized year (by adding 1900) of the stacked Year columns using years_columns
standard_year <- 1900+stack(olympic_data_raw[,years_columns])[1]
stacked_jump <- stack(olympic_data_raw[,jump_columns])[1]
olympic_data_tidy_baseR <- data.frame(cbind(standard_year, stacked_jump))
names(olympic_data_tidy_baseR) <- c("Year", "Long Jump")
print(head(olympic_data_tidy_baseR))
```

```
##   Year Long Jump
## 1 1896    249.75
## 2 1900    282.88
## 3 1904    289.00
## 4 1908    294.50
## 5 1912    299.25
## 6 1920    281.50
```

Next we will tidy the olympic data using tidyverse

```r
#use dplyr select to get the year and long jump columns separately into dataframes
years_frame <-  select(olympic_data_raw,1,3,5,7)
jump_frame <-  select(olympic_data_raw,2,4,6,8)
#stack the dataframes
years_stacked <- 1900+stack(years_frame)[1]
jump_stacked <- stack(jump_frame)[1]
#bind the stacked dataframes and restore the names
olympic_data_tidy_tidyverse <- data.frame(cbind(years_stacked,jump_stacked))
names(olympic_data_tidy_tidyverse) <- c("Year","Long Jump")
print(head(olympic_data_tidy_tidyverse))
```

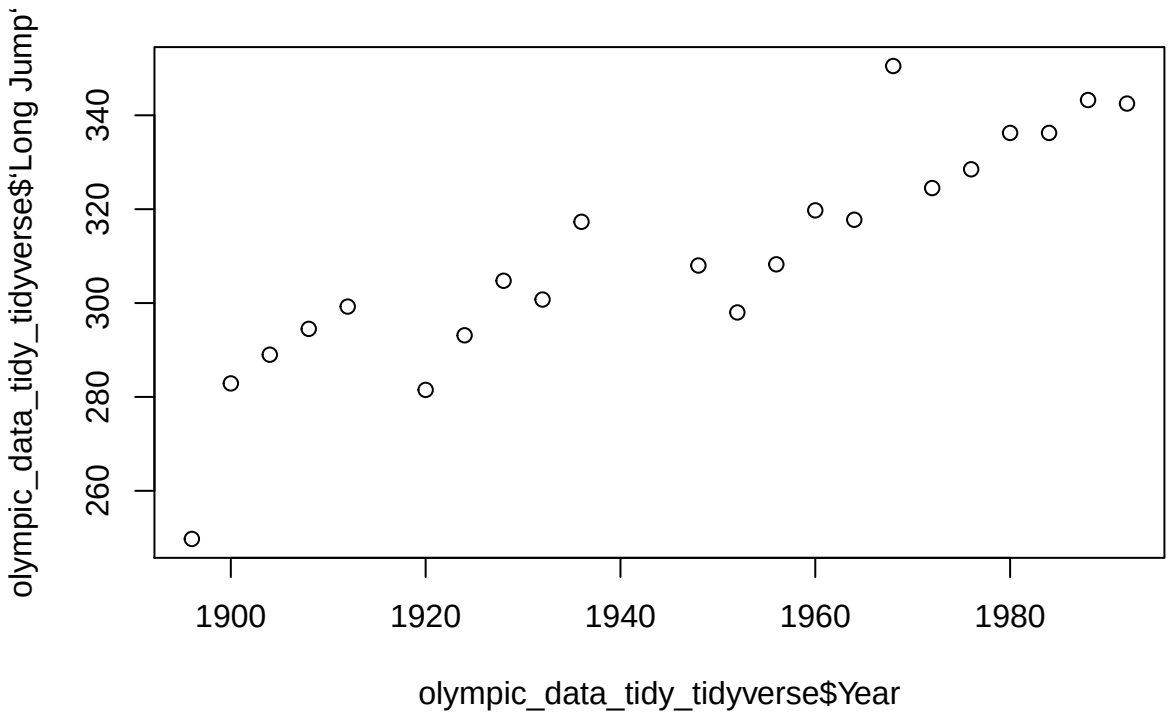
```
##   Year Long Jump
## 1 1896    249.75
## 2 1900    282.88
## 3 1904    289.00
## 4 1908    294.50
## 5 1912    299.25
## 6 1920    281.50
```

Finally we create a summary and plot of the olympic data

```r
knitr::kable(summary(olympic_data_tidy_tidyverse))
```

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
NA's :2	NA's :2

```r
plot(olympic_data_tidy_tidyverse$Year,olympic_data_tidy_tidyverse$`Long Jump`)
```



(c)

Now we will look at the brain and body weight data for various species. First we will import the raw data from the URL.

```
#Specify and assign the url to a variable
species_url <- "https://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/BrainandBodyWeight.dat"
#Read in the raw data
species_data_raw <- fread(species_url,skip = 1,data.table = FALSE,header = FALSE, fill = TRUE)
```

Next we will tidy the species data using base R.

```
#body_columns and brain_columns are the position of the columns in the raw species data for body weight
body_columns <- seq(1,6,2)
brain_columns <- seq(2,6,2)
#body_vector and brain_vector are the stacked vectors of body and brain weight
#extracted from species_data_raw by using body_columns and brain_columns
body_vector <- stack(species_data_raw[,body_columns])[1]
brain_vector <- stack(species_data_raw[,brain_columns])[1]
#the id corresponding to a unique species
species_id <- seq(1,62)
#Now we combine the stacked vectors (the first 62 entries since there are only 62 species) into a data
species_data_tidy_baseR<-data.frame(cbind(species_id,body_vector[1:62,1],brain_vector[1:62,1]))
names(species_data_tidy_baseR) <- c("Species ID","Body Weight(Kg)", "Brain Weight(g)")
print(head(species_data_tidy_baseR))
```

```
##   Species ID Body Weight(Kg) Brain Weight(g)
## 1          1           3.385            44.5
## 2          2           0.480            15.5
## 3          3           1.350             8.1
## 4          4         465.000           423.0
## 5          5          36.330           119.5
## 6          6          27.660           115.0
```

Now we will tidy the species data using tidyverse

```r
#use dplyr to select the brain and body weight columns into their own data frame
#brain_frame and body_frame contain the brain and body weight columns
brain_frame <- select(species_data_raw,seq(2,6,2))
body_frame <- select(species_data_raw,seq(1,6,2))
#stack the dataframes
brain_stack <- stack(brain_frame)
body_stack <- stack(body_frame)
#build the tidy dataframe using data.frame and cbind from the stacked dataframes and restore the origin
species_data_tidy_tidyverse <- data.frame(cbind(species_id,body_vector[1:62,1],brain_vector[1:62,1]))
names(species_data_tidy_tidyverse) <- c("Species ID", "Body Weight (Kg)", "Brain Weight (g)")
print(head(species_data_tidy_tidyverse))
```

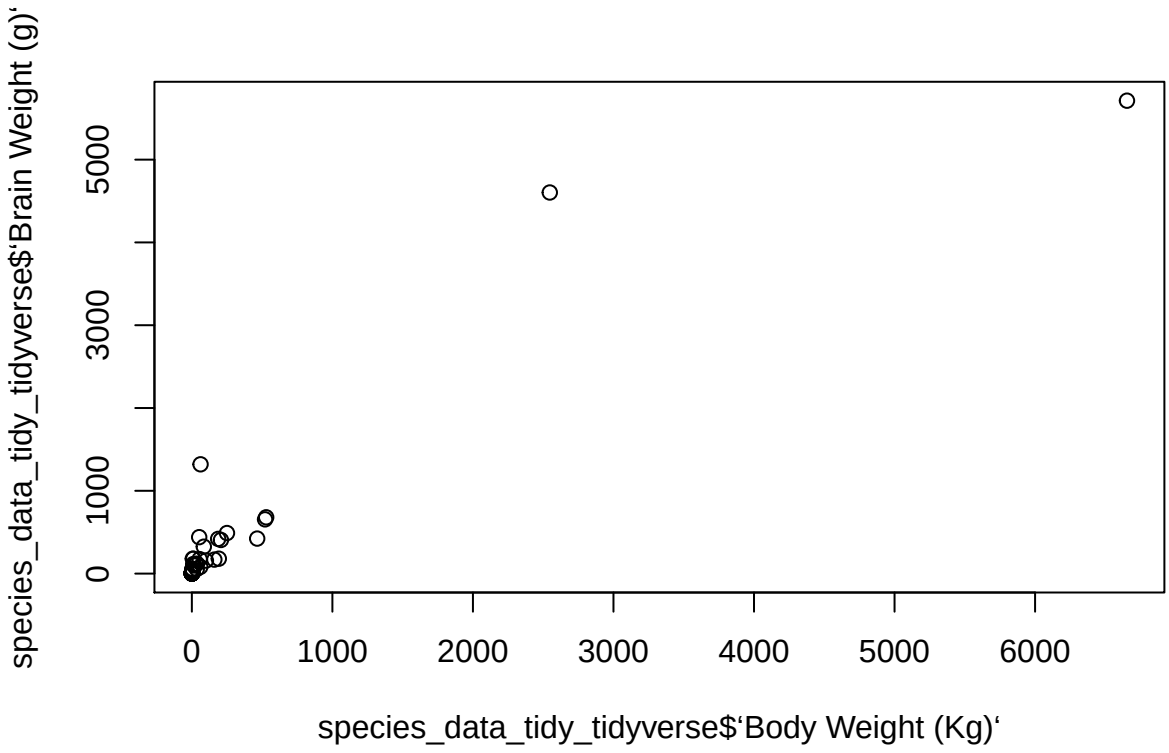
```
##   Species ID Body Weight (Kg) Brain Weight (g)
## 1          1            3.385             44.5
## 2          2            0.480             15.5
## 3          3            1.350              8.1
## 4          4          465.000            423.0
## 5          5           36.330            119.5
## 6          6           27.660            115.0
```

Now we will do a summary and plot of the species data

```r
knitr::kable(summary(species_data_tidy_tidyverse[,2:3]))
```

Body Weight (Kg)	Brain Weight (g)
Min. : 0.005	Min. : 0.10
1st Qu.: 0.600	1st Qu.: 4.25
Median : 3.342	Median : 17.25
Mean : 198.790	Mean : 283.13
3rd Qu.: 48.202	3rd Qu.: 166.00
Max. :6654.000	Max. :5712.00

```r
plot(species_data_tidy_tidyverse$`Body Weight (Kg)`,species_data_tidy_tidyverse$`Brain Weight (g)`)
```



(d)

Now we will look at the tomato yield data. We will first assign the URL to a variable and then import the raw data.

```
tomato_url <- "https://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/tomato.dat"
#Read in the raw data
tomato_data_raw <- fread(tomato_url,skip = 1,data.table = FALSE,header = TRUE, fill = TRUE)
```

Now we will tidy the data using base R.

```
#make a copy of the raw data
tomato_data_tidy_baseR <- copy(tomato_data_raw)
#rename the columns and apply it to the copy
names(tomato_data_tidy_baseR) <- c("Variety", "10000", "20000", "30000")
#there are three measurements for each of the planting densities ("10000","20000","30000"). So for each
Ife_yield <- vector(length = 0)
PusaEarlyDwarf <- vector(length = 0)
#populate Ife_yield and PusaEarlyDwarf by unlisting, string splitting and converting to numeric the dat
for (j in 2:4)
{
  Ife_yield <- append(Ife_yield,as.numeric(unlist(strsplit(tomato_data_tidy_baseR[1,j], split = ","))))
  PusaEarlyDwarf <- append(PusaEarlyDwarf,as.numeric(unlist(strsplit(tomato_data_tidy_baseR[2,j], split
}
measurements <- c(Ife_yield, PusaEarlyDwarf)
#set up the column for planting density
plant_density <- c(rep(10000,3),rep(20000,3),rep(30000,3))
plant_density <- rep(plant_density,2)
#set up the column for variety names
variety_names <- c(rep(tomato_data_tidy_baseR[1,1],9), rep(tomato_data_tidy_baseR[2,1],9))
#construct the dataframe
tomato_data_tidy_baseR <-  data.frame(cbind(variety_names,plant_density,measurements))
#give appropriate names
names(tomato_data_tidy_baseR) <- c("Variety", "Planting Density", "Measurement")
```

```
print(head(tomato_data_tidy_baseR))
```

```
##   Variety Planting Density Measurement
## 1 Ife\\#1            10000        16.1
## 2 Ife\\#1            10000        15.3
## 3 Ife\\#1            10000        17.5
## 4 Ife\\#1            20000        16.6
## 5 Ife\\#1            20000        19.2
## 6 Ife\\#1            20000        18.5
```

Now we will tidy the tomato data using tidyverse

```
#copy the raw data as tomato+data_tidy_tidyverse and reassign column names
tomato_data_tidy_tidyverse <- tomato_data_raw
names(tomato_data_tidy_tidyverse) <- c("Variety", "10000", "20000", "30000")
#densityi_col corresponds to column i+1. So for example density1_col corresponds to the column header 1
density1_col <- c(as.numeric(unlist(strsplit(tomato_data_tidy_tidyverse[1,2], split = ","))),as.numeric
density2_col <- c(as.numeric(unlist(strsplit(tomato_data_tidy_tidyverse[1,3], split = ","))),as.numeric
density3_col <- c(as.numeric(unlist(strsplit(tomato_data_tidy_tidyverse[1,4], split = ","))),as.numeric
#variety_col will list the variety names for each measurement
variety_col <- c(rep(tomato_data_tidy_tidyverse[1,1],3),rep(tomato_data_tidy_tidyverse[2,1],3))
#get the untidy dataframe ready for using the gather function
tomato_data_tidy_tidyverse <- data.frame(cbind(variety_col,density1_col,density2_col,density3_col))
names(tomato_data_tidy_tidyverse) <- c("Variety", "10000", "20000", "30000")
#use the tidyverse gather function to get the final tidy dataframe
tomato_data_tidy_tidyverse <- tomato_data_tidy_tidyverse %>% gather(key = "Planting Density", value = "
tomato_data_tidy_tidyverse$`Planting Density` <- as.integer(tomato_data_tidy_tidyverse$`Planting Density
tomato_data_tidy_tidyverse$Measurement <- as.numeric(tomato_data_tidy_tidyverse$Measurement)
print(head(tomato_data_tidy_tidyverse))
```

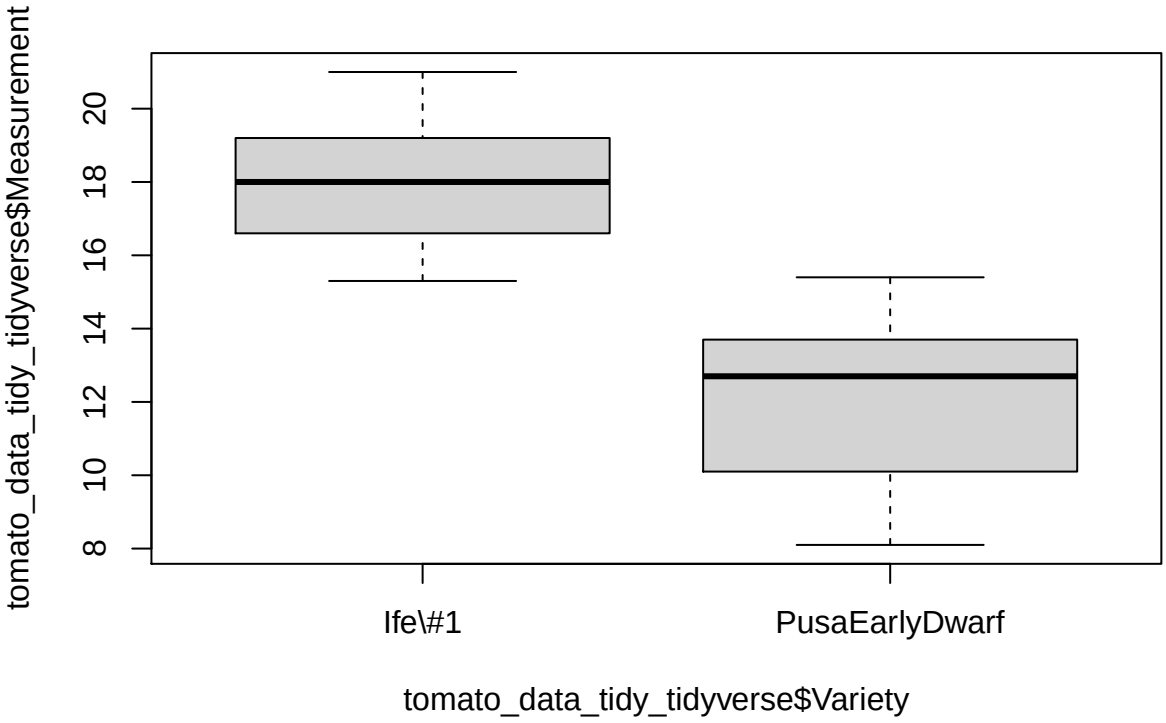
```
##          Variety Planting Density Measurement
## 1        Ife\\#1            10000        16.1
## 2        Ife\\#1            10000        15.3
## 3        Ife\\#1            10000        17.5
## 4 PusaEarlyDwarf            10000         8.1
## 5 PusaEarlyDwarf            10000         8.6
## 6 PusaEarlyDwarf            10000        10.1
```

Now we will do a summary and plot of the tomato yield data

```
#gives a summary table and a boxplot of measurement by variety
knitr::kable(summary(tomato_data_tidy_tidyverse[2:3]))
```

Planting Density	Measurement
Min. :10000	Min. : 8.10
1st Qu.:10000	1st Qu.:12.95
Median :20000	Median :15.35
Mean :20000	Mean :15.07
3rd Qu.:30000	3rd Qu.:17.88
Max. :30000	Max. :21.00

```
boxplot(tomato_data_tidy_tidyverse$Measurement ~ tomato_data_tidy_tidyverse$Variety)
```


tomato_data_tidy_tidyverse$Measurement
8
10
12
14
16
18
20
Ife\#1
PusaEarlyDwarf
tomato_data_tidy_tidyverse$Variety