# Homework 2 STAT 5014

*David Edwards*

*9/10/2019*

## Problem 3

Question: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.

Answer: The biggest area that I feel version control can help will be by simply keeping a backup of the code that I create. This will help serve as a hedge against hardware malfunction. The second way that I can see version control as being useful is by allowing me to easily revert back to previously working code in the event that make changes that result in the code either not compiling or not working correctly. This will allow me the ability to more easily experiment without the risk of breaking what has worked in the past.

## Problem 4

Question: 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

Answer: ### Part a For this data set the the columns and rows give values to the variables Operator and Item, with the actual values embeded in the table. What I did was read in the data line by line and the loop through each line to pick out the values. Then I added a row to a data frame for each "Operator", "Item", "Value" combination. This resulted in a data frame with 150 rows and 3 variables.

```r
library(readr)
sensory <- readLines("http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat")
```

```
## Warning in readLines("http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/
## data/Sensory.dat"): incomplete final line found on 'http://
## www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat'
```

```r
df <- data.frame("Operator" = integer(), "Item" = integer(), "Value" = double())
for (i in seq(3,32,3)){
  line1 <- sensory[i]
  line1 <- strsplit(line1, split = " ")
  item_num <- parse_integer(line1[[1]][1])
  for(j in 2:6){
    row <- c(j-1,item_num, parse_double(line1[[1]][j]))
    df <- rbind(df, row)
  }
  line2 <- sensory[i+1]
  line2 <- strsplit(line2, split = " ")
  for(j in 1:5){
    row <- c(j,item_num, parse_double(line2[[1]][j]))
    df <- rbind(df, row)
  }
  line3 <- sensory[i+2]
  line3 <- strsplit(line3, split = " ")
  for(j in 1:5){
    row <- c(j,item_num, parse_double(line3[[1]][j]))
    df <- rbind(df, row)
  }
}
names(df)[names(df) == "X1"] <-"Operator"
names(df)[names(df) == "X1.1"] <-"Item"
names(df)[names(df) == "X4.3"] <-"Value"
df
```

```
##     Operator Item Value
## 1          1    1   4.3
## 2          2    1   4.9
## 3          3    1   3.3
## 4          4    1   5.3
## 5          5    1   4.4
## 6          1    1   4.3
## 7          2    1   4.5
## 8          3    1   4.0
## 9          4    1   5.5
## 10         5    1   3.3
## 11         1    1   4.1
## 12         2    1   5.3
## 13         3    1   3.4
## 14         4    1   5.7
## 15         5    1   4.7
## 16         1    2   6.0
## 17         2    2   5.3
## 18         3    2   4.5
## 19         4    2   5.9
## 20         5    2   4.7
## 21         1    2   4.9
## 22         2    2   6.3
## 23         3    2   4.2
## 24         4    2   5.5
## 25         5    2   4.9
## 26         1    2   6.0
## 27         2    2   5.9
## 28         3    2   4.7
## 29         4    2   6.3
## 30         5    2   4.6
## 31         1    3   2.4
## 32         2    3   2.5
## 33         3    3   2.3
## 34         4    3   3.1
## 35          5    3   2.4
## 36          1    3   3.9
## 37          2    3   3.0
## 38          3    3   2.8
## 39          4    3   2.7
## 40          5    3   1.3
## 41          1    3   1.9
## 42          2    3   3.9
## 43          3    3   2.6
## 44          4    3   4.6
## 45          5    3   2.2
## 46          1    4   7.4
## 47          2    4   8.2
## 48          3    4   6.4
## 49          4    4   6.8
## 50          5    4   6.0
## 51          1    4   7.1
## 52          2    4   7.9
## 53          3    4   5.9
## 54          4    4   7.3
## 55          5    4   6.1
## 56          1    4   6.4
## 57          2    4   7.1
## 58          3    4   6.9
## 59          4    4   7.0
## 60          5    4   6.7
## 61          1    5   5.7
## 62          2    5   6.3
## 63          3    5   5.4
## 64          4    5   6.1
## 65          5    5   5.9
## 66          1    5   5.8
## 67          2    5   5.7
## 68          3    5   5.4
## 69          4    5   6.2
## 70          5    5   6.5
## 71          1    5   5.8
## 72          2    5   6.0
## 73          3    5   6.1
## 74          4    5   7.0
## 75          5    5   4.9
## 76          1    6   2.2
## 77          2    6   2.4
## 78          3    6   1.7
## 79          4    6   3.4
## 80          5    6   1.7
## 81          1    6   3.0
## 82          2    6   1.8
## 83          3    6   2.1
## 84          4    6   4.0
## 85          5    6   1.7
## 86          1    6   2.1
## 87          2    6   3.3
## 88          3    6   1.1
## 89          4    6   3.3
## 90          5    6   2.1
## 91          1    7   1.2
## 92          2    7   1.5
## 93          3    7   1.2
## 94          4    7   0.9
## 95          5    7   0.7
## 96          1    7   1.3
## 97          2    7   2.4
## 98          3    7   0.8
## 99          4    7   1.2
## 100         5    7   1.3
## 101         1    7   0.9
## 102         2    7   3.1
## 103         3    7   1.1
## 104         4    7   1.9
## 105         5    7   1.6
## 106         1    8   4.2
## 107         2    8   4.8
## 108         3    8   4.5
## 109         4    8   4.6
## 110         5    8   3.2
## 111         1    8   3.0
## 112         2    8   4.5
## 113         3    8   4.7
## 114         4    8   4.9
## 115         5    8   4.6
## 116         1    8   4.8
## 117         2    8   4.8
## 118         3    8   4.7
## 119         4    8   4.8
## 120         5    8   4.3
## 121         1    9   8.0
## 122         2    9   8.6
## 123         3    9   9.0
## 124         4    9   9.4
## 125         5    9   8.8
## 126         1    9   9.0
## 127         2    9   7.7
## 128         3    9   6.7
## 129         4    9   9.0
## 130         5    9   7.9
## 131         1    9   8.9
## 132         2    9   9.2
## 133         3    9   8.1
## 134         4    9   9.1
## 135         5    9   7.6
## 136         1   10   5.0
## 137         2   10   4.8
## 138         3   10   3.9
## 139         4   10   5.5
## 140         5   10   3.8
## 141         1   10   5.4
## 142         2   10   5.0
## 143        3   10   3.4
## 144        4   10   4.9
## 145        5   10   4.6
## 146        1   10   2.8
## 147        2   10   5.2
## 148        3   10   4.1
## 149        4   10   3.9
## 150        5   10   5.5
```

### Part b

For this part I did essentially the samething as above, read in the data line by line and clean it up so that we have a data frame with 2 variables and 22 observations. The table read in on 7 lines and then the first line wasn't needed and the next 4 lines had 4 observations each and the final two lines had 3 observations each.

```
lj <- readLines("http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/LongJumpData.dat")
```

```
## Warning in readLines("http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/
## data/LongJumpData.dat"): incomplete final line found on 'http://
## www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/LongJumpData.dat'
```

```
df <- data.frame("Year" = integer(), "Long Jump"= double())
line2 = lj[2]
line3 = lj[3]
line4 = lj[4]
line5 = lj[5]
line6 = lj[6]
line7 = lj[7]
line2 <- strsplit(line2, split = " ")
line3 <- strsplit(line3, split = " ")
line4 <- strsplit(line4, split = " ")
line5 <- strsplit(line5, split = " ")
line6 <- strsplit(line6, split = " ")
line7 <- strsplit(line7, split = " ")
for(i in c(1,3,5, 7)){
  row <- c(parse_integer(line2[[1]][i]), parse_double(line2[[1]][i+1]))
  df <- rbind(df, row)
  row <- c(parse_integer(line3[[1]][i]), parse_double(line3[[1]][i+1]))
  df <- rbind(df, row)
  row <- c(parse_integer(line4[[1]][i]), parse_double(line4[[1]][i+1]))
  df <- rbind(df, row)
  row <- c(parse_integer(line5[[1]][i]), parse_double(line5[[1]][i+1]))
  df <- rbind(df, row)
}
for(i in c(1,3,5)){
  row <- c(parse_integer(line6[[1]][i]), parse_double(line6[[1]][i+1]))
  df <- rbind(df, row)
  row <- c(parse_integer(line7[[1]][i]), parse_double(line7[[1]][i+1]))
  df <- rbind(df, row)
}
names(df)[names(df) == "X.4"] <-"Year"
names(df)[names(df) == "X.4.1"] <-"Long Jump"
df
```

```
##    Year X249.75
## 1    -4  249.75
## 2     0  282.88
## 3     4  289.00
## 4     8  294.50
## 5    24  293.13
## 6    28  304.75
## 7    32  300.75
## 8    36  317.31
## 9    56  308.25
## 10   60  319.75
## 11   64  317.75
## 12   68  350.50
## 13   80  336.25
## 14   84  336.25
## 15   88  343.25
## 16   92  342.50
## 17   12  299.25
## 18   20  281.50
## 19   48  308.00
## 20   52  298.00
## 21   72  324.50
## 22   76  328.50
```

### Part c

I'm guessing from the little I have to go on that values are in the correct order, that is alternating between Body WT and Brain WT, as the table suggests. From looking at the data it does seem that the values may have been reversed (as it would see that the brain weights more than the body) but it could also be that they are in different units.

Given the assumption above it was just a matter of looping through all the lines and pairing up the data to put into a data frame with two variables and 62 observations. The last only had 2 observations while the other 20 had 3 observations each.

```r
wt <- readLines("http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/BrainandBodyWeight.dat")
df <- data.frame("Body Wt" = double(), "Brain Wt"= double())
for(i in 2:21){
  line = wt[i]
  line <- strsplit(line, split = " ")
  for(j in c(1,3,5)){
    row <- c(parse_double(line[[1]][j]), parse_double(line[[1]][j+1]))
    df <- rbind(df, row)
  }
}
line <- wt[22]
line <- strsplit(line, split =" ")
for(j in c(1,3)){
  row <- c(parse_double(line[[1]][j]), parse_double(line[[1]][j+1]))
  df <- rbind(df, row)
}
names(df)[names(df) == "X3.385"] <-"Body WT"
names(df)[names(df) == "X44.5"] <-"Brain WT"
df
```

```
##     Body WT Brain WT
## 1     3.385    44.50
## 2   521.000   655.00
## 3     2.500    12.10
## 4     0.480    15.50
## 5     0.785     3.50
## 6    55.500   175.00
## 7     1.350     8.10
## 8    10.000   115.00
## 9   100.000   157.00
## 10  465.000   423.00
## 11    3.300    25.60
## 12   52.160   440.00
## 13   36.330   119.50
## 14    0.200     5.00
## 15   10.550   179.50
## 16   27.660   115.00
## 17    1.410    17.50
## 18    0.550     2.40
## 19   14.830    98.20
## 20  529.000   680.00
## 21   60.000    81.00
## 22    1.040     5.50
## 23  207.000   406.00
## 24    3.600    21.00
## 25    4.190    58.00
## 26   85.000   325.00
## 27    4.288    39.20
## 28    0.425     6.40
## 29    0.750    12.30
## 30    0.280     1.90
## 31    0.101     4.00
## 32   62.000  1320.00
## 33    0.075     1.20
## 34    0.920     5.70
## 35 6654.000  5712.00
## 36    0.122     3.00
## 37    1.000     6.60
## 38    3.500     3.90
## 39    0.048     0.33
## 40    0.005     0.10
## 41    6.800   179.00
## 42  192.000   180.00
## 43    0.060     1.00
## 44   35.000    56.00
## 45    3.000    25.00
## 46    3.500    10.80
## 47    4.050    17.00
## 48  160.000   169.00
## 49    2.000    12.30
## 50    0.120     1.00
## 51    0.900     2.60
## 52    1.700     6.30
## 53    0.023     0.40
## 54    1.620    11.40
## 55 2547.000  4603.00
## 56    0.010     0.30
## 57    0.104     2.50
## 58    0.023     0.30
## 59    1.400    12.50
## 60    4.235    50.40
## 61  187.100   419.00
## 62  250.000   490.00
```

**Part d**

I honestly have no idea what is going on with this data set. Without any context this is nearly impossible to determine the approprate way to clean this data set. So, here is the assumption that I'm working with to complete the assignment.

1. I'm going to assume that the 3 values in the second row are some amount of treatment.
2. I'm going to assume that values in the first column are the two different treatments.
3. I'm going to assume the values in the table are the actual results.

Given that I have looped through the two lines that contain informaiton and added that to a data frame containing 3 variable and 18 observations. For some reason I was never able to add the labels for the treatments "Ife\#1" or "PusaEarlyDwarf" do the data frame, so I call "PusaEarlyDwarf" treatment 2 and "Ife\#1" treatment 1.

```
t <- readLines("http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/tomato.dat")
df <- data.frame("Amount" = integer(), "Treatment"= integer(), "Value" = double())
line3 <- t[3]
line4 <- t[4]
line3 <- strsplit(line3, split = "\\s+")
line4 <- strsplit(line4, split = "\\s+")
line3[[1]][1] ### Treatment 1
```

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

```
line4[[1]][1] ### Treatment 2
```

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

```
for(i in 2:4){
  values <- strsplit(line3[[1]][i], split = ",")
  for(j in 1:3){
    row <-  c((i-1)*10000, 1, parse_double(values[[1]][j]))
    df<- rbind(df,row)
  }
}
for(i in 2:4){
  values <- strsplit(line4[[1]][i], split = ",")
  for(j in 1:3){
    row <-  c((i-1)*10000, 2, parse_double(values[[1]][j]))
    df<- rbind(df,row)
  }
}

names(df)[names(df) == "X10000"] <-"Amount"
names(df)[names(df) == "X1"] <-"Treatment"
names(df)[names(df) == "X16.1"] <-"Value"

df
```

```
##    Amount Treatment Value
## 1   10000         1  16.1
## 2   10000         1  15.3
## 3   10000         1  17.5
## 4   20000         1  16.6
## 5   20000         1  19.2
## 6   20000         1  18.5
## 7   30000         1  20.8
## 8   30000         1  18.0
## 9   30000         1  21.0
## 10  10000         2   8.1
## 11  10000         2   8.6
## 12  10000         2  10.1
## 13  20000         2  12.7
## 14  20000         2  13.7
## 15  20000         2  11.5
## 16  30000         2  14.4
## 17  30000         2  15.4
## 18  30000         2  13.7
```

## 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.

```r
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.
```

```r
# 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')
```

Answer: To clean up the data I put the 3 variables of interest into a new data frame and this used this new data frame to eleminate the rows with NA's. That way we deal with only the data that we have for these 3 variables.

It wasn't really clear to me what statistic we could use from the pH_min and pH_max, so I just used there average.

Given that it doesn't really make sense for a factor variable to be the response for a linear model I have considered a model with Foliage color as the input and average ph as the response. Given the summary below there are two colors overall that seem to have a significant variation in ph levels from the mean and they are gray-green and green.

```r
p <- data.frame(plants$Foliage_Color, plants$pH_Min, plants$pH_Max)
p<- na.omit(p)
p$pH_Avg <- (p$plants.pH_Min+p$plants.pH_Max)/2
mod <- lm(p$pH_Avg~p$plants.Foliage_Color)
summary(mod)
```

```
##
## Call:
## lm(formula = p$pH_Avg ~ p$plants.Foliage_Color)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.63750 -0.33410  0.00061  0.31590  2.01590
##
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          5.99939    0.05951 100.810  < 2e-16
## p$plants.Foliage_ColorGray-Green     0.41261    0.12312   3.351 0.000841
## p$plants.Foliage_ColorGreen          0.18471    0.06294   2.935 0.003430
## p$plants.Foliage_ColorRed            0.16311    0.27594   0.591 0.554617
## p$plants.Foliage_ColorWhite-Gray     0.44505    0.18924   2.352 0.018914
## p$plants.Foliage_ColorYellow-Green -0.06189    0.13440  -0.461 0.645275
##
## (Intercept)                         ***
## p$plants.Foliage_ColorGray-Green    ***
## p$plants.Foliage_ColorGreen         **
## p$plants.Foliage_ColorRed
## p$plants.Foliage_ColorWhite-Gray    *
## p$plants.Foliage_ColorYellow-Green
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5389 on 826 degrees of freedom
## Multiple R-squared:  0.0234, Adjusted R-squared:  0.01749
## F-statistic: 3.958 on 5 and 826 DF,  p-value: 0.00149
```