HW2

Tanner Glass

9/3/2020

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!

When it comes to version control I can see this to be a really great tool for collaborators. When working in a group there are times (most of the time actually...) where members have deferring ideas. Instead of having to have a vote on which way to go, it possible to use version control to create two possible routes off a base version. That way, the original code can be preserved and then we can have to paths to find out which way is optimal.

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.

Part A

a. Sensory data from five operators. – see video, I am doing this one
   http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat

We will read in the data first using the link above. Once read in, we will observe the first few operations.

```
#install.packages("data.table")
#fread used in lecture, similar to base r function read.table
library("data.table")

# url is here --> http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat

url <- "http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat"
#sensory_data_raw <- fread(url,fill= TRUE,header = TRUE, skip = 1)
#saveRDS(sensory_data_raw, "sensory_data_raw.RDS")
sensory_data_raw <-readRDS("sensory_data_raw.RDS")
head(sensory_data_raw)
```

```
##    Item   1   2   3   4   5
## 1:  1.0 4.3 4.9 3.3 5.3 4.4
```

```
## 2:  4.3 4.5 4.0 5.5 3.3  NA
## 3:  4.1 5.3 3.4 5.7 4.7  NA
## 4:  2.0 6.0 5.3 4.5 5.9 4.7
## 5:  4.9 6.3 4.2 5.5 4.9  NA
## 6:  6.0 5.9 4.7 6.3 4.6  NA
```

Yikes. From a pure pull, we see some major issues with the data frame. The data is intended to be given as 15 observations. Each Operator is supposed to have 3 observations per item. Operator 1 has the 1st, 6th, and 11th observation and so on for the four operators. The goal will be to extra the item numbers which are consisted within the first vector, and the compile the rest via each operator.

```
data<-c()
for (i in 1:nrow(sensory_data_raw))
{
  data<-c(data,as.numeric(sensory_data_raw[i,]))
}
data<-data[!is.na(data)] #for loop gets rid of NAs
remove<-c()
for (i in seq(from=0, to=length(data), by=16))
{
  remove<-c(remove,i+1) #removes item numbers
  #mixed in data
}
data<- data[-remove]
df<-data.frame(rep(0,15))
for (n in 0:((length(data)/15-1)))
{ #extracts 15 obs per item
  df<-cbind(df,data[(15*n+1):(15*(n+1))])
}
df<-df[,-1]
df2<- data.frame(rep(0,3))
for (k in 1:10)
{#assigns item and operator to each eval
for (j in 1:5)
{
  assign(paste0("item_",k,"_operator_",j),df[seq((0+j), to=15, by=5),k])
  df2<-cbind(df2,eval(as.name(paste0("item_",k,"_operator_",j))))
  names(df2)[(5*(k-1)+j+1)] <- paste0("Item_",k,"_Operator_",j)
}#names each column
}
df2<-df2[,-1]

df2
```

```
##   Item_1_Operator_1 Item_1_Operator_2 Item_1_Operator_3 Item_1_Operator_4
## 1               4.3               4.9               3.3               5.3
## 2               4.3               4.5               4.0               5.5
## 3               4.1               5.3               3.4               5.7
##   Item_1_Operator_5 Item_2_Operator_1 Item_2_Operator_2 Item_2_Operator_3
## 1               4.4               6.0               5.3               4.5
## 2               3.3               4.9               6.3               4.2
## 3               4.7               6.0               5.9               4.7
##   Item_2_Operator_4 Item_2_Operator_5 Item_3_Operator_1 Item_3_Operator_2
## 1               5.9               4.7               2.4               2.5
## 2               5.5               4.9               3.9               3.0
```

```
## 3              6.3              4.6              1.9              3.9
##   Item_3_Operator_3 Item_3_Operator_4 Item_3_Operator_5 Item_4_Operator_1
## 1               2.3               3.1               2.4               7.4
## 2               2.8               2.7               1.3               7.1
## 3               2.6               4.6               2.2               6.4
##   Item_4_Operator_2 Item_4_Operator_3 Item_4_Operator_4 Item_4_Operator_5
## 1               8.2               6.4               6.8               6.0
## 2               7.9               5.9               7.3               6.1
## 3               7.1               6.9               7.0               6.7
##   Item_5_Operator_1 Item_5_Operator_2 Item_5_Operator_3 Item_5_Operator_4
## 1               5.7               6.3               5.4               6.1
## 2               5.8               5.7               5.4               6.2
## 3               5.8               6.0               6.1               7.0
##   Item_5_Operator_5 Item_6_Operator_1 Item_6_Operator_2 Item_6_Operator_3
## 1               5.9               2.2               2.4               1.7
## 2               6.5               3.0               1.8               2.1
## 3               4.9               2.1               3.3               1.1
##   Item_6_Operator_4 Item_6_Operator_5 Item_7_Operator_1 Item_7_Operator_2
## 1               3.4               1.7               1.2               1.5
## 2               4.0               1.7               1.3               2.4
## 3               3.3               2.1               0.9               3.1
##   Item_7_Operator_3 Item_7_Operator_4 Item_7_Operator_5 Item_8_Operator_1
## 1               1.2               0.9               0.7               4.2
## 2               0.8               1.2               1.3               3.0
## 3               1.1               1.9               1.6               4.8
##   Item_8_Operator_2 Item_8_Operator_3 Item_8_Operator_4 Item_8_Operator_5
## 1               4.8               4.5               4.6               3.2
## 2               4.5               4.7               4.9               4.6
## 3               4.8               4.7               4.8               4.3
##   Item_9_Operator_1 Item_9_Operator_2 Item_9_Operator_3 Item_9_Operator_4
## 1               8.0               8.6               9.0               9.4
## 2               9.0               7.7               6.7               9.0
## 3               8.9               9.2               8.1               9.1
##   Item_9_Operator_5 Item_10_Operator_1 Item_10_Operator_2 Item_10_Operator_3
## 1               8.8                5.0                4.8                3.9
## 2               7.9                5.4                5.0                3.4
## 3               7.6                2.8                5.2                4.1
##   Item_10_Operator_4 Item_10_Operator_5
## 1                5.5                3.8
## 2                4.9                4.6
## 3                3.9                5.5
```

We have achieved a tidy data set using base R functions. Note that there are two factors in this data. Operator and Item, we simply create a variable for each of them and observed the three observations that they each cover. Now to use dplyr.

```
#install.packages('dplyr')
#install.packages('tidyr')
library("dplyr")
```

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

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

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

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

```
library("tidyr")


data2<-c()
for (i in 1:nrow(sensory_data_raw))
{
  data2<-c(data,as.numeric(sensory_data_raw[i,]))
}
data2<-data2[!is.na(data)] #for loop gets rid of NAs
remove2<-c()
for (i in seq(from=0, to=length(data), by=16))
{
  remove2<-c(remove2,i+1) #removes item numbers
  #mixed in data
}
data2<- data2[-remove]
df3<-dplyr::data_frame(rep(0,15))
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
for (n in 0:((length(data)/15-1)))
{ #extracts 15 obs per item
  df3<-cbind(df3,data[(15*n+1):(15*(n+1))])
}
df3<-df3[,-1]
df4<- dplyr::data_frame(rep(0,3))
for (k in 1:10)
{#assigns item and operator to each eval
for (j in 1:5)
{
  assign(paste0("item_",k,"_operator_",j),df3[seq((0+j), to=15, by=5),k])
  df4<-cbind(df4,eval(as.name(paste0("item_",k,"_operator_",j))))
  names(df4)[(5*(k-1)+j+1)] <- paste0("Item_",k,"_Operator_",j)
}
}
sensory_data_tidyr<-select(df4,2:51)

sensory_data_tidyr
```

```
##   Item_1_Operator_1 Item_1_Operator_2 Item_1_Operator_3 Item_1_Operator_4
## 1               4.3               4.9               3.3               5.3
## 2               4.3               4.5               4.0               5.5
## 3               4.1               5.3               3.4               5.7
##   Item_1_Operator_5 Item_2_Operator_1 Item_2_Operator_2 Item_2_Operator_3
## 1               4.4               6.0               5.3               4.5
```

```
## 2              3.3              4.9              6.3              4.2
## 3              4.7              6.0              5.9              4.7
##   Item_2_Operator_4 Item_2_Operator_5 Item_3_Operator_1 Item_3_Operator_2
## 1               5.9               4.7               2.4               2.5
## 2               5.5               4.9               3.9               3.0
## 3               6.3               4.6               1.9               3.9
##   Item_3_Operator_3 Item_3_Operator_4 Item_3_Operator_5 Item_4_Operator_1
## 1               2.3               3.1               2.4               7.4
## 2               2.8               2.7               1.3               7.1
## 3               2.6               4.6               2.2               6.4
##   Item_4_Operator_2 Item_4_Operator_3 Item_4_Operator_4 Item_4_Operator_5
## 1               8.2               6.4               6.8               6.0
## 2               7.9               5.9               7.3               6.1
## 3               7.1               6.9               7.0               6.7
##   Item_5_Operator_1 Item_5_Operator_2 Item_5_Operator_3 Item_5_Operator_4
## 1               5.7               6.3               5.4               6.1
## 2               5.8               5.7               5.4               6.2
## 3               5.8               6.0               6.1               7.0
##   Item_5_Operator_5 Item_6_Operator_1 Item_6_Operator_2 Item_6_Operator_3
## 1               5.9               2.2               2.4               1.7
## 2               6.5               3.0               1.8               2.1
## 3               4.9               2.1               3.3               1.1
##   Item_6_Operator_4 Item_6_Operator_5 Item_7_Operator_1 Item_7_Operator_2
## 1               3.4               1.7               1.2               1.5
## 2               4.0               1.7               1.3               2.4
## 3               3.3               2.1               0.9               3.1
##   Item_7_Operator_3 Item_7_Operator_4 Item_7_Operator_5 Item_8_Operator_1
## 1               1.2               0.9               0.7               4.2
## 2               0.8               1.2               1.3               3.0
## 3               1.1               1.9               1.6               4.8
##   Item_8_Operator_2 Item_8_Operator_3 Item_8_Operator_4 Item_8_Operator_5
## 1               4.8               4.5               4.6               3.2
## 2               4.5               4.7               4.9               4.6
## 3               4.8               4.7               4.8               4.3
##   Item_9_Operator_1 Item_9_Operator_2 Item_9_Operator_3 Item_9_Operator_4
## 1               8.0               8.6               9.0               9.4
## 2               9.0               7.7               6.7               9.0
## 3               8.9               9.2               8.1               9.1
##   Item_9_Operator_5 Item_10_Operator_1 Item_10_Operator_2 Item_10_Operator_3
## 1               8.8                5.0                4.8                3.9
## 2               7.9                5.4                5.0                3.4
## 3               7.6                2.8                5.2                4.1
##   Item_10_Operator_4 Item_10_Operator_5
## 1                5.5                3.8
## 2                4.9                4.6
## 3                3.9                5.5
```

We have accomplished organizing the data. Dyplr and Tidyr have similar commands for data frame building and selecting columns you need for organization. There is a rename function in the packages but using the for loop in Base R still proved to be more optimal.

Part B

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

```
url_b <-"http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/LongJumpData.dat"

#jump_data_raw<-fread(url_b,fill = TRUE,header = FALSE,skip = 1)
#saveRDS(jump_data_raw, "jump_data_raw.RDS")
jump_data_raw <-readRDS("jump_data_raw.RDS")
jump_data_raw
```

```
##    V1     V2 V3     V4 V5     V6 V7     V8
## 1: -4 249.75 24 293.13 56 308.25 80 336.25
## 2:  0 282.88 28 304.75 60 319.75 84 336.25
## 3:  4 289.00 32 300.75 64 317.75 88 343.25
## 4:  8 294.50 36 317.31 68 350.50 92 342.50
## 5: 12 299.25 48 308.00 72 324.50 NA     NA
## 6: 20 281.50 52 298.00 76 328.50 NA     NA
```

Since the header contained mulitple words for column names and mulptiple colomuns for different observations, it was best to read in without that top line and add in later on. We understand the two variables as year and the jump record. We will notw create a tidy data set in R.

```
year<-c(jump_data_raw$V1,jump_data_raw$V3,jump_data_raw$V5,jump_data_raw$V7[1:4])#concatenate all years
jump <-c(jump_data_raw$V2,jump_data_raw$V4,jump_data_raw$V6,jump_data_raw$V8[1:4])#concatenate all jump

jump_data_tidy<-data.frame(year,jump)
jump_data_tidy$year<-jump_data_tidy$year+1900
jump_data_tidy
```

```
##    year   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
## 7  1924 293.13
## 8  1928 304.75
## 9  1932 300.75
## 10 1936 317.31
## 11 1948 308.00
## 12 1952 298.00
## 13 1956 308.25
## 14 1960 319.75
## 15 1964 317.75
## 16 1968 350.50
## 17 1972 324.50
## 18 1976 328.50
## 19 1980 336.25
## 20 1984 336.25
## 21 1988 343.25
## 22 1992 342.50
```

By concatenation, we create a data frame that is set to standard. Now using dplyr and tidyr.

```
library("dplyr")
library("tidyr")

jump_data_tidyr<-dplyr::data_frame(Year=c(jump_data_raw$V1,jump_data_raw$V3,jump_data_raw$V5,jump_data_

#function does not work with pipelining,

jump_data_tidyr %>%
  slice(1:22) ->jump_data_tidyr


jump_data_tidyr$Year<-  jump_data_tidyr$Year+1900

jump_data_tidyr
```

```
## # A tibble: 22 x 2
##     Year  Jump
##    <dbl> <dbl>
##  1  1896  250.
##  2  1900  283.
##  3  1904  289
##  4  1908  294.
##  5  1912  299.
##  6  1920  282.
##  7  1924  293.
##  8  1928  305.
##  9  1932  301.
## 10  1936  317.
## # ... with 12 more rows
```

Part C

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

```
url_c <-"http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/BrainandBodyWeight.dat"

weight_data_raw<-fread(url_c,fill = TRUE,header = FALSE,skip = 1)

weight_data_raw
```

```
##           V1     V2       V3     V4      V5     V6
##  1:    3.385   44.5  521.000  655.0   2.500  12.10
##  2:    0.480   15.5    0.785    3.5  55.500 175.00
##  3:    1.350    8.1   10.000  115.0 100.000 157.00
##  4:  465.000  423.0    3.300   25.6  52.160 440.00
##  5:   36.330  119.5    0.200    5.0  10.550 179.50
##  6:   27.660  115.0    1.410   17.5   0.550   2.40
##  7:   14.830   98.2  529.000  680.0  60.000  81.00
##  8:    1.040    5.5  207.000  406.0   3.600  21.00
##  9:    4.190   58.0   85.000  325.0   4.288  39.20
## 10:    0.425    6.4    0.750   12.3   0.280   1.90
## 11:    0.101    4.0   62.000 1320.0   0.075   1.20
## 12:    0.920    5.7 6654.000 5712.0   0.122   3.00
```

```
## 13:    1.000    6.6    3.500    3.9   0.048   0.33
## 14:    0.005    0.1    6.800  179.0 192.000 180.00
## 15:    0.060    1.0   35.000   56.0   3.000  25.00
## 16:    3.500   10.8    4.050   17.0 160.000 169.00
## 17:    2.000   12.3    0.120    1.0   0.900   2.60
## 18:    1.700    6.3    0.023    0.4   1.620  11.40
## 19: 2547.000 4603.0    0.010    0.3   0.104   2.50
## 20:    0.023    0.3    1.400   12.5   4.235  50.40
## 21:  187.100  419.0  250.000  490.0      NA     NA
##           V1     V2       V3     V4      V5     V6
```

Similar set up to part b, I will concatenate each variable in which are related.

```
body_wt<-c(weight_data_raw$V1,weight_data_raw$V3,weight_data_raw$V5)

brain_wt<-c(weight_data_raw$V2,weight_data_raw$V4,weight_data_raw$V6)

weight_data_tidy<-data.frame(body_wt,brain_wt)

weight_data_tidy<- weight_data_tidy[-63,]
weight_data_tidy
```

```
##     body_wt brain_wt
## 1     3.385    44.50
## 2     0.480    15.50
## 3     1.350     8.10
## 4   465.000   423.00
## 5    36.330   119.50
## 6    27.660   115.00
## 7    14.830    98.20
## 8     1.040     5.50
## 9     4.190    58.00
## 10    0.425     6.40
## 11    0.101     4.00
## 12    0.920     5.70
## 13    1.000     6.60
## 14    0.005     0.10
## 15    0.060     1.00
## 16    3.500    10.80
## 17    2.000    12.30
## 18    1.700     6.30
## 19 2547.000  4603.00
## 20    0.023     0.30
## 21  187.100   419.00
## 22  521.000   655.00
## 23    0.785     3.50
## 24   10.000   115.00
## 25    3.300    25.60
## 26    0.200     5.00
## 27    1.410    17.50
## 28  529.000   680.00
## 29  207.000   406.00
## 30   85.000   325.00
## 31    0.750    12.30
## 32   62.000  1320.00
```

```
## 33 6654.000  5712.00
## 34    3.500     3.90
## 35    6.800   179.00
## 36   35.000    56.00
## 37    4.050    17.00
## 38    0.120     1.00
## 39    0.023     0.40
## 40    0.010     0.30
## 41    1.400    12.50
## 42  250.000   490.00
## 43    2.500    12.10
## 44   55.500   175.00
## 45  100.000   157.00
## 46   52.160   440.00
## 47   10.550   179.50
## 48    0.550     2.40
## 49   60.000    81.00
## 50    3.600    21.00
## 51    4.288    39.20
## 52    0.280     1.90
## 53    0.075     1.20
## 54    0.122     3.00
## 55    0.048     0.33
## 56  192.000   180.00
## 57    3.000    25.00
## 58  160.000   169.00
## 59    0.900     2.60
## 60    1.620    11.40
## 61    0.104     2.50
## 62    4.235    50.40
```

Now for using tidyverse.

```
weight_data_tidyr<- data_frame(body_wt=c(weight_data_raw$V1,weight_data_raw$V3,weight_data_raw$V5),brai
weight_data_tidyr<-slice(weight_data_tidyr,1:62)
weight_data_tidyr
```

```
## # A tibble: 62 x 2
##    body_wt brain_wt
##      <dbl>    <dbl>
##  1   3.38      44.5
##  2   0.48      15.5
##  3   1.35       8.1
##  4 465        423
##  5  36.3      120.
##  6  27.7      115
##  7  14.8       98.2
##  8   1.04       5.5
##  9   4.19      58
## 10   0.425      6.4
## # ... with 52 more rows
```

Easily enough, datasets using both base r and tidyverse functions are obtained.

Part D

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

```
url_d <-"http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/tomato.dat"

tomato_data_raw<-fread(url_d)
```

```
## Warning in fread(url_d): Detected 3 column names but the data has 4 columns
## (i.e. invalid file). Added 1 extra default column name for the first column
## which is guessed to be row names or an index. Use setnames() afterwards if this
## guess is not correct, or fix the file write command that created the file to
## create a valid file.
```

```
tomato_data_raw
```

```
##                V1          10000          20000          30000
## 1:        Ife\\#1 16.1,15.3,17.5 16.6,19.2,18.5 20.8,18.0,21.0
## 2: PusaEarlyDwarf  8.1,8.6,10.1, 12.7,13.7,11.5 14.4,15.4,13.7
```

We will now use base r funcions to clean up the data set.

```
tomato_data_tidy<- tomato_data_raw[,2:4]
tomato_data_tidy2<- data.frame(rep(0,3))
for (v in 1:3)
{
  for (w in 1:2)
  {
   new_measurements<- as.numeric(unlist((strsplit(as.character(tomato_data_tidy[w,]),split = ","))))[((v
   tomato_data_tidy2<-data.frame(tomato_data_tidy2,new_measurements)
  }
}



tomato_data_tidy2<-tomato_data_tidy2[,-1]
names(tomato_data_tidy2)<-c("10000_IFE","10000_Pusa","20000_IFE","20000_Pusa","30000_IFE","30000_Pusa")
tomato_data_tidy2
```

```
##   10000_IFE 10000_Pusa 20000_IFE 20000_Pusa 30000_IFE 30000_Pusa
## 1      16.1        8.1      16.6       12.7      20.8       14.4
## 2      15.3        8.6      19.2       13.7      18.0       15.4
## 3      17.5       10.1      18.5       11.5      21.0       13.7
```

We have successfully tidied the data using base R. Now to the Tidyverse.

```
tomato_data_raw %>%
  select("10000","20000","30000")->tomato_data_tidyr

tomato_data_tidyr2<- dplyr::data_frame(rep(0,3))
for (v in 1:3)
{
  for (w in 1:2)
  {
   new_measurements<- as.numeric(unlist((strsplit(as.character(tomato_data_tidyr[w,]),split = ","))))[(
   tomato_data_tidyr2<-data.frame(tomato_data_tidyr2,new_measurements)
  }
```

```
}
tomato_data_tidyr2<-select(tomato_data_tidyr2,2:7)
tomato_data_tidyr2<- rename(tomato_data_tidyr2,"10000_IFE"=new_measurements,"10000_Pusa"=new_measuremen

tomato_data_tidyr2
```

```
##   10000_IFE 10000_Pusa 20000_IFE 20000_Pusa 30000_IFE 30000_Pusa
## 1      16.1        8.1      16.6       12.7      20.8       14.4
## 2      15.3        8.6      19.2       13.7      18.0       15.4
## 3      17.5       10.1      18.5       11.5      21.0       13.7
```

We have achieved tidying the data using tidyverse funcions.