HW2_Sauer_Annie

Annie Sauer

9/5/2018

Problem 4

Even though I am a student that primarily works on projects independently, version control is a useful tool that will allow me to back up and easily edit my codes. With version control I can save multiple versions of a project and easily track the changes I make to it. This will come in handy if I need to revert changes or see if minor changes actually had profound impacts on my code. I may also collaborate with others on projects in the future, and I will already have a collaboration framework in place.

Problem 5

(a) Sensory data from five operators

```
# Import data
Sensory <- read_csv("https://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat",
                    col_names = "Data")
```

```
## Parsed with column specification:
## cols(
##   Data = col_character()
## )
```

```
head(Sensory)
```

```
## # A tibble: 6 x 1
##   Data
##   <chr>
## 1 "\tOperator"
## 2 Item 1 2 3 4 5
## 3 1 4.3 4.9 3.3 5.3 4.4
## 4 4.3 4.5 4.0 5.5 3.3
## 5 4.1 5.3 3.4 5.7 4.7
## 6 2 6.0 5.3 4.5 5.9 4.7
```

```
dim(Sensory)
```

```
## [1] 32  1
```

```
# This data is made up of 32 rows and one column.  Each row is a character string.
# There is a row indicator integer in every third row (ranging from 1 to 10).  First
# we need to separate each single row of characters into six separate columns (the
# first one representing the item and the other five representing the operators).

sensory_new <- separate(Sensory, Data, c('Item','Operator 1','Operator 2',
                                         'Operator 3','Operator 4','Operator 5'),
                        sep=" ",fill="left")
sensory_new
```

```
## # A tibble: 32 x 6
```

```
##    Item  `Operator 1` `Operator 2` `Operator 3` `Operator 4` `Operator 5`
##    <chr> <chr>        <chr>        <chr>        <chr>        <chr>
##  1 <NA>  <NA>         <NA>         <NA>         <NA>         "\tOperator"
##  2 Item  1            2            3            4            5
##  3 1     4.3          4.9          3.3          5.3          4.4
##  4 <NA>  4.3          4.5          4.0          5.5          3.3
##  5 <NA>  4.1          5.3          3.4          5.7          4.7
##  6 2     6.0          5.3          4.5          5.9          4.7
##  7 <NA>  4.9          6.3          4.2          5.5          4.9
##  8 <NA>  6.0          5.9          4.7          6.3          4.6
##  9 3     2.4          2.5          2.3          3.1          2.4
## 10 <NA>  3.9          3.0          2.8          2.7          1.3
## # ... with 22 more rows
```

```
# The first two rows are now repeats of our header, and we can delete them.
sensory_new <- sensory_new[-c(1,2),]
# Next we create a vector to fill the "Item" column which contains many NA values.
item = rep(1:10, each = 3)
sensory_new['Item'] <- item
# Finally we gather the five Operator columns into two columns.
sensory_new <- sensory_new %>%
  gather(Operator, Data,-Item) %>%
  mutate(Operator = parse_number(Operator))
# Change Data values to numeric values
sensory_new$Data <- as.numeric(sensory_new$Data)
# Produce summary table
summary(sensory_new)
```

```
##       Item         Operator      Data
##  Min.   : 1.0   Min.   :1   Min.   :0.700
##  1st Qu.: 3.0   1st Qu.:2   1st Qu.:3.025
##  Median : 5.5   Median :3   Median :4.700
##  Mean   : 5.5   Mean   :3   Mean   :4.657
##  3rd Qu.: 8.0   3rd Qu.:4   3rd Qu.:6.000
##  Max.   :10.0   Max.   :5   Max.   :9.400
```

Now we have a tidy data set with three columns - Item, Operator, and Data. Each column represents a single variable and each row represents a single observation. The third column is titled 'Data' simply becasue we were not given information on the units or importance of these values. In cleaning up this data, we assumed that each original row was in the proper order of Operators 1-5 and that the Items 1-10 carried through three rows each. We may not have had enough information to assume this layout from the original data set.

(b) Gold Medal Performance for Olympic Men's Long Jump (year coded as 1900 = 0)

```
# Import data
Jump <- read_csv("https://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/LongJumpData.dat"
                 ,col_names = "Data")
```

```
## Parsed with column specification:
## cols(
##   Data = col_character()
## )
```

```
head(Jump)
```

```
## # A tibble: 6 x 1
##   Data
##   <chr>
## 1 Year Long Jump Year Long Jump Year Long Jump Year Long Jump
## 2 -4 249.75 24 293.13 56 308.25 80 336.25
## 3 0 282.88 28 304.75 60 319.75 84 336.25
## 4 4 289.00 32 300.75 64 317.75 88 343.25
## 5 8 294.50 36 317.31 68 350.50 92 342.50
## 6 12 299.25 48 308.00 72 324.50
```

```
dim(Jump)
```

```
## [1] 7 1
```

```
# We have a data set with seven rows and one column.  In the first row the 'Year' and
# 'Long Jump' headers are repeated four times each.  First we need to separate
# each row into the eight different columns that correspond to these initial headers.

# Remove the first row of header information (we will soon assign new headers)
jump_new <- Jump[-c(1),]
# Separate into columns with new headers
jump_new <- separate(jump_new, Data, c('Year 1','LJ 1','Year 2','LJ 2','Year 3',
                                       'LJ 3','Year 4','LJ 4'), sep=" ")
```

```
## Warning: Expected 8 pieces. Missing pieces filled with `NA` in 2 rows [5,
## 6].
```

```
jump_new
```

```
## # A tibble: 6 x 8
##   `Year 1` `LJ 1` `Year 2` `LJ 2` `Year 3` `LJ 3` `Year 4` `LJ 4`
##   <chr>    <chr>  <chr>    <chr>  <chr>    <chr>  <chr>    <chr>
## 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>
```

```
# Stack all year and jump columns
years <- stack(jump_new, c('Year 1','Year 2','Year 3','Year 4'))
jumps <- stack(jump_new, c('LJ 1','LJ 2','LJ 3','LJ 4'))

# Combine stacked columns into new data frame, remove the unnecessary indicators,
# and rename column headers
jump_df = data.frame(years,jumps)
jump_df <- jump_df[,-c(2,4)]
names(jump_df) <- c('Year','Long Jump')
# Remove the two rows of NA at the end
jump_df <- na.omit(jump_df)
# Convert to numeric values
jump_df$Year <- as.numeric(jump_df$Year)
jump_df$`Long Jump`<- as.numeric(jump_df$`Long Jump`)

# Produce summary table
summary(jump_df)
```

```
##       Year         Long Jump
##  Min.   :-4.00   Min.   :249.8
##  1st Qu.:21.00   1st Qu.:295.4
##  Median :50.00   Median :308.1
##  Mean   :45.45   Mean   :310.3
##  3rd Qu.:71.00   3rd Qu.:327.5
##  Max.   :92.00   Max.   :350.5
```

We now have a tidy data set with two variables represented in two columns. In rearranging this data we assumed that consecutive 'Year' and 'Long Jump' values were paired data, which should be a reasonable assumption.

(c) Brain weight (g) and body weight (kg) for 62 species.

```
# Import data
weight <- read_csv("https://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/BrainandBodyWeight.dat"
                   ,col_names = "Data")
```

```
## Parsed with column specification:
## cols(
##   Data = col_character()
## )
```

```
head(weight)
```

```
## # A tibble: 6 x 1
##   Data
##   <chr>
## 1 Body Wt Brain Wt Body Wt Brain Wt Body Wt Brain Wt
## 2 3.385 44.5 521.000 655.0 2.500 12.10
## 3 0.480 15.5 0.785 3.5 55.500 175.00
## 4 1.350 8.1 10.000 115.0 100.000 157.00
## 5 465.000 423.0 3.300 25.6 52.160 440.00
## 6 36.330 119.5 0.200 5.0 10.550 179.50
```

```
dim(weight)
```

```
## [1] 22  1
```

```
# Our data set is made up of 22 rows and 1 column.  In the first row we have
# the column headers 'Body Wt' and 'Brain Wt' repeated three times each.  First
# we must break up each row into the six separate columns that correspond to
# these headers.

# Remove first row of header information (we will replace with new headers)
weight_new <- weight[-c(1),]
# Separate into columns and create new headers
weight_new <- separate(weight_new, Data, c('Body 1','Brain 1','Body 2',
                                           'Brain 2','Body 3','Brain 3'), sep=" ")
```

```
## Warning: Expected 6 pieces. Missing pieces filled with `NA` in 1 rows [21].
```

```
weight_new
```

```
## # A tibble: 21 x 6
##    `Body 1` `Brain 1` `Body 2` `Brain 2` `Body 3` `Brain 3`
##    <chr>    <chr>     <chr>    <chr>     <chr>    <chr>
```

```
##  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
## # ... with 11 more rows
```

```
# Stack all body weight and brain weight columns
body <- stack(weight_new, c('Body 1','Body 2','Body 3'))
brain <- stack(weight_new, c('Brain 1','Brain 2','Brain 3'))

# Combine stacked columns into new data frame
weight_df = data.frame(body,brain)
weight_df
```

```
##       values    ind values.1   ind.1
## 1      3.385 Body 1     44.5 Brain 1
## 2      0.480 Body 1     15.5 Brain 1
## 3      1.350 Body 1      8.1 Brain 1
## 4    465.000 Body 1    423.0 Brain 1
## 5     36.330 Body 1    119.5 Brain 1
## 6     27.660 Body 1    115.0 Brain 1
## 7     14.830 Body 1     98.2 Brain 1
## 8      1.040 Body 1      5.5 Brain 1
## 9      4.190 Body 1     58.0 Brain 1
## 10     0.425 Body 1      6.4 Brain 1
## 11     0.101 Body 1      4.0 Brain 1
## 12     0.920 Body 1      5.7 Brain 1
## 13     1.000 Body 1      6.6 Brain 1
## 14     0.005 Body 1      0.1 Brain 1
## 15     0.060 Body 1      1.0 Brain 1
## 16     3.500 Body 1     10.8 Brain 1
## 17     2.000 Body 1     12.3 Brain 1
## 18     1.700 Body 1      6.3 Brain 1
## 19  2547.000 Body 1   4603.0 Brain 1
## 20     0.023 Body 1      0.3 Brain 1
## 21   187.100 Body 1    419.0 Brain 1
## 22   521.000 Body 2    655.0 Brain 2
## 23     0.785 Body 2      3.5 Brain 2
## 24    10.000 Body 2    115.0 Brain 2
## 25     3.300 Body 2     25.6 Brain 2
## 26     0.200 Body 2      5.0 Brain 2
## 27     1.410 Body 2     17.5 Brain 2
## 28   529.000 Body 2    680.0 Brain 2
## 29   207.000 Body 2    406.0 Brain 2
## 30    85.000 Body 2    325.0 Brain 2
## 31     0.750 Body 2     12.3 Brain 2
## 32    62.000 Body 2   1320.0 Brain 2
## 33  6654.000 Body 2   5712.0 Brain 2
## 34     3.500 Body 2      3.9 Brain 2
```

```
## 35    6.800 Body 2    179.0 Brain 2
## 36   35.000 Body 2     56.0 Brain 2
## 37    4.050 Body 2     17.0 Brain 2
## 38    0.120 Body 2      1.0 Brain 2
## 39    0.023 Body 2      0.4 Brain 2
## 40    0.010 Body 2      0.3 Brain 2
## 41    1.400 Body 2     12.5 Brain 2
## 42  250.000 Body 2    490.0 Brain 2
## 43    2.500 Body 3    12.10 Brain 3
## 44   55.500 Body 3   175.00 Brain 3
## 45  100.000 Body 3   157.00 Brain 3
## 46   52.160 Body 3   440.00 Brain 3
## 47   10.550 Body 3   179.50 Brain 3
## 48    0.550 Body 3     2.40 Brain 3
## 49   60.000 Body 3    81.00 Brain 3
## 50    3.600 Body 3    21.00 Brain 3
## 51    4.288 Body 3    39.20 Brain 3
## 52    0.280 Body 3     1.90 Brain 3
## 53    0.075 Body 3     1.20 Brain 3
## 54    0.122 Body 3     3.00 Brain 3
## 55    0.048 Body 3     0.33 Brain 3
## 56  192.000 Body 3   180.00 Brain 3
## 57    3.000 Body 3    25.00 Brain 3
## 58  160.000 Body 3   169.00 Brain 3
## 59    0.900 Body 3     2.60 Brain 3
## 60    1.620 Body 3    11.40 Brain 3
## 61    0.104 Body 3     2.50 Brain 3
## 62    4.235 Body 3    50.40 Brain 3
## 63     <NA> Body 3     <NA> Brain 3
```

```r
# Remove indicator columns (they are not necessary)
weight_df <- weight_df[,-c(2,4)]
# Rename columns
names(weight_df) <- c('Body Weight','Brain Weight')
# Omit NA row
weight_df <- na.omit(weight_df)
# Convert all values to numeric
weight_df$`Body Weight` <- as.numeric(weight_df$`Body Weight`)
weight_df$`Brain Weight` <- as.numeric(weight_df$`Brain Weight`)
# Produce summary table
summary(weight_df)
```

```
##   Body Weight        Brain Weight
##  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.:  48.203   3rd Qu.: 166.00
##  Max.   :6654.000   Max.   :5712.00
```

We now have a tidy data set with two variables represented in two columns. Again, we had to assume that consecutive 'Brain Weight' and 'Body Weight' values were paired.

(d) Triplicate measurements of tomato yield for two varieties of tomatos at three planting densities

```
# Import data
tomato <- fread("http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/tomato.dat"
                ,sep=" ")
```

```
## Warning in fread("http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/
## tomato.dat", : 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.
```

```
# Note - I used fread in this example so I could already use 'sep' and split the data
# set into multiple columns.
tomato
```

```
##                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
```

```
dim(tomato)
```

```
## [1] 2 4
```

```
# Our data set is made up of 2 rows and 4 columns.  We first need to separate the data
# for each yield then separate the data for each variety.

# First separate each row by density (nine columns, three of each density)
tomato_new <- tomato %>%
  separate('10000',into= c('10000.1','10000.2','10000.3'),sep=",") %>%
  separate('20000',into= c('20000.1','20000.2','20000.3'),sep=",") %>%
  separate('30000',into= c('30000.1','30000.2','30000.3'),sep=",")
```

```
## Warning: Expected 3 pieces. Additional pieces discarded in 1 rows [2].
```

```
tomato_new
```

```
##                V1 10000.1 10000.2 10000.3 20000.1 20000.2 20000.3 30000.1
## 1:        Ife\\#1    16.1    15.3    17.5    16.6    19.2    18.5    20.8
## 2: PusaEarlyDwarf     8.1     8.6    10.1    12.7    13.7    11.5    14.4
##    30000.2 30000.3
## 1:    18.0    21.0
## 2:    15.4    13.7
```

```
# Now gather Variety and Density into their own columns.
tomato_new <- tomato_new %>%
  gather(Density, Yield,-V1) %>%
  mutate(Density = as.integer(Density))
tomato_new
```

```
##                V1 Density Yield
## 1         Ife\\#1   10000  16.1
## 2  PusaEarlyDwarf   10000   8.1
## 3         Ife\\#1   10000  15.3
## 4  PusaEarlyDwarf   10000   8.6
## 5         Ife\\#1   10000  17.5
```

```
## 6  PusaEarlyDwarf   10000  10.1
## 7          Ife\\#1   20000  16.6
## 8  PusaEarlyDwarf   20000  12.7
## 9          Ife\\#1   20000  19.2
## 10 PusaEarlyDwarf   20000  13.7
## 11         Ife\\#1   20000  18.5
## 12 PusaEarlyDwarf   20000  11.5
## 13         Ife\\#1   30000  20.8
## 14 PusaEarlyDwarf   30000  14.4
## 15         Ife\\#1   30000  18.0
## 16 PusaEarlyDwarf   30000  15.4
## 17         Ife\\#1   30000  21.0
## 18 PusaEarlyDwarf   30000  13.7
```

```
# Rename first column and change 'Yield' to numeric
colnames(tomato_new)[1] <- "Variety"
tomato_new$Yield <- as.numeric(tomato_new$Yield)

# Produce summary table
summary(tomato_new)
```

```
##    Variety             Density          Yield
##  Length:18          Min.   :10000   Min.   : 8.10
##  Class :character   1st Qu.:10000   1st Qu.:12.95
##  Mode  :character   Median :20000   Median :15.35
##                     Mean   :20000   Mean   :15.07
##                     3rd Qu.:30000   3rd Qu.:17.88
##                     Max.   :30000   Max.   :21.00
```

Now we have a tidy data set with three columns representing three variables. We did not have to assume anything when rearranging the data set - the original data set was clearer than the previous ones.

Problem 6

```
# 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')

# Ignore NA columns
plants <- na.omit(plants)
```

```r
# Create a new column - mean of pH_min and pH_max
plants <- mutate(plants, pH_mean = (pH_Min+pH_Max)/2)

# Goal - Determine if there is a relationshp between Foliage_Color and pH_mean
# Group data by Foliage Color and summarize pH by color
by_color <- group_by(plants, Foliage_Color)
summarise(by_color, pH_avg = mean(pH_mean),n=n())
```

```
## # A tibble: 6 x 3
##   Foliage_Color pH_avg     n
##   <fct>          <dbl> <int>
## 1 Dark Green      6.00    82
## 2 Gray-Green      6.37    24
## 3 Green           6.18   675
## 4 Red             6.4      3
## 5 White-Gray      6.44     9
## 6 Yellow-Green    5.94    20
```

```r
# Run a linear model
model <- lm(plants$pH_mean ~ plants$Foliage_Color)
summary(model)
```

```
##
## Call:
## lm(formula = plants$pH_mean ~ plants$Foliage_Color)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.63750 -0.37083 -0.02511  0.32489  2.02489
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      5.99939    0.05940 101.001  < 2e-16 ***
## plants$Foliage_ColorGray-Green   0.37144    0.12483   2.976  0.00301 **
## plants$Foliage_ColorGreen        0.17572    0.06290   2.793  0.00534 **
## plants$Foliage_ColorRed          0.40061    0.31618   1.267  0.20551
## plants$Foliage_ColorWhite-Gray   0.44505    0.18888   2.356  0.01870 *
## plants$Foliage_ColorYellow-Green -0.06189    0.13414  -0.461  0.64465
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5379 on 807 degrees of freedom
## Multiple R-squared:  0.02189,    Adjusted R-squared:  0.01583
## F-statistic: 3.613 on 5 and 807 DF,  p-value: 0.003077
```

Problem 7 - omitted