ECOG314 -- Suggested project guidelines -- A worked example using Wine dataset from UCI data repository

William Ampeh

September 12, 2016

Demo on how to analyze a multivariate dataset from UCI ML data repository

Reference materials

1. This R Data Import Tutorial Is Everything You Need -- R bloggers
2. Importing Data Into R - Part Two -- R bloggers

Data repositories

- Financial Accounts of the United States
- DATA.GOV
- UCI Data Repository -- Machine learning datasets

Financial Data repositories

- Federal Reserve -- few clicks required financial_accounts_of_the_us.png
- Data Market one of the user friendly Websites -- few clicks required
- INFORUM -- requires Inforum's database and regression package, G, to access that data

Big Data repositories

- Data Science Central
- Big data made simple
- Amazon Web Service (AWS)
- caesar0301/awesome-public-datasets

Steps:

1. Visit the UCI wine data repository for a detailed description, the data dictionary and maintainer of the data

   results of a chemical analysis of the qualities of wines grown in the same region in Italy but derived from 3 different cultivars (a plant variety that has been produced in cultivation by selective breeding)

   14 columns (dimensions, attributes)

   with 1st column containing the cultivar of a wine sample (labelled 1, 2 or 3)

   next 13 columns contain the concentrations of the 13 different chemicals in that sample

2. Setup your project workspace and cd into that direction (Session -> Set Working Directory -> Choose Directory (CNTRL+Shift+h)

3. Read dataset into R
   read.table(...)
   I always get a segment of the file first (for very large files)
   read.table("my_some_datatable_name.txt" header = TRUE, nrows = 25) # easier to work with smaller dataset

Step 1 of N: Point browser to the UCI wine dataset repository

- UCI ML wine dataset repository

Step 2 of N: Clear workspace, and set and cd into working directory

R-CODE

```
rm(list=ls())

# Set working directory and set Project-related global variables
setwd("C:/Users/William/Desktop/ECOG314/lecture_3")

# You do not have to, but if you need to share variable between different fun
ctions you can use
some_global_variable1 <<- 0                    # take note of the  <<-
some_global_variable2 <<- 0

max_steps <-  10

# alternatively you can use
assign("some_global_variable3", 0)

some_local_variable1 <- 0
```

Step 3 of N: Read the multivariate dataset into R

Data structure

- Data consists of 1 row per of record (observation) wine sample.
- First column, the cultivar of a wine sample is labelled 1, 2 or 3
- Next 12 columns contain the concentrations of the 13 different chemicals in that sample.
- Columns are separated by commas.

Segments from the wine dataset:

Reading the data

- We use read.table() function to read in the data into R
- We use sep="," argument in read.table() to specify that columns are separated by commas.

R-CODE

```
mysample <- 25     # Using nrows, even as a mild over-estimate, will help memory
usage
                   # -1 => skip record #1, i.e., the HEADER in the file,
resulting in 177 records (observations)

wine <- read.table(
         file = "http://archive.ics.uci.edu/ml/machine-learning-
databases/wine/wine.data",  # filename at the UCI repository
       header = TRUE,        # filehas a header
          sep = ",",         # rows separated by commas
       nrow = mysample       # just the first n=25 rows + 1 header = 26
       )
```

R-CODE

```
#--
# check dimension of the data you have read -- data sanity check 1


dim(wine)

Output: [1] 25 14
```

R-CODE

```
class(wine)

Output: [1] "data.frame"
```

R-CODE

```
head(wine, n=5)

Output:

   X1 X14.23 X1.71 X2.43 X15.6 X127 X2.8 X3.06 X.28 X2.29 X5.64 X1.04 X3.92 X1065
 1  1  13.20  1.78  2.14  11.2  100 2.65  2.76 0.26  1.28  4.38  1.05  3.40  1050
 2  1  13.16  2.36  2.67  18.6  101 2.80  3.24 0.30  2.81  5.68  1.03  3.17  1185
 3  1  14.37  1.95  2.50  16.8  113 3.85  3.49 0.24  2.18  7.80  0.86  3.45  1480
 4  1  13.24  2.59  2.87  21.0  118 2.80  2.69 0.39  1.82  4.32  1.04  2.93   735
 5  1  14.20  1.76  2.45  15.2  112 3.27  3.39 0.34  1.97  6.75  1.05  2.85  1450
```

BIG DATA files, import and extract a random sample into R, then offload master file onto hard drive to free up memory

- Use read.table(...) function to read in the data into R
- Use sample(...) function to get a random sample from the raw data
- Use saveRDS(...) and rm(...) functions to saVE file to hard drive and remove BIG DATA file to free up memory
- For Ubuntu and Linux users, this can be done at the Operating System (OS) level
  shuf -n 10 > small_wine_data.txt
  Use nl wine_data.txt | shuf -n 10 to verify that the lines in file has been shuffled (randomized)

R-CODE

```
# to take a random sample of 25 rows from the file

#
n <- sample(    x = c(1:178),      # range of rows
             size = 25,            # number of ranomly selected rows
          replace = FALSE          # sample WITHOUT replacement
       )

cat(sprintf("\nRows selected at random: %s\n", paste(n, collapse = ', ')))
```

Output:

```
## Rows selected at random: 175, 24, 97, 51, 5, 121, 104, 15, 149, 18, 78, 10
8, 146, 161, 111, 54, 26, 64, 166, 147, 58, 154, 49, 98, 60
```

R-CODE

```
# Read the data from source
big_wine.df <- read.table(          # we know from above that the class of ob
ject returned by read.table is a data.frame (.df extension)
         file = "http://archive.ics.uci.edu/ml/machine-learning-databases/win
e/wine.data",  # filename at the UCI repository
       header = TRUE,               # file has a header
          sep = ","                 # rows separated by commas.  NOTE:  We to
ok out the comma after at the end of the line
       # nrow = mysample            # we are ready every line in the file
       )

# column names
colnames(big_wine.df) <- paste("v", 1:dim(big_wine.df)[2], sep="")    #  v1,
v2, ....v13, v14
head(big_wine.df)
```

Output:

```
   v1    v2   v3   v4   v5  v6   v7   v8   v9  v10  v11  v12  v13  v14
1   1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28 4.38 1.05 3.40 1050
2   1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81 5.68 1.03 3.17 1185
3   1 14.37 1.95 2.50 16.8 113 3.85 3.49 0.24 2.18 7.80 0.86 3.45 1480
4   1 13.24 2.59 2.87 21.0 118 2.80 2.69 0.39 1.82 4.32 1.04 2.93  735
5   1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97 6.75 1.05 2.85 1450
6   1 14.39 1.87 2.45 14.6  96 2.50 2.52 0.30 1.98 5.25 1.02 3.58 1290
```

R-CODE

```
# check dimension of the data you have read -- data sanity check 1
dim(big_wine.df)
```

Output: [1] 177 14

R-CODE

```
small_wine.df <- big_wine.df[n, ]   #every row specified by n.  That is 86, 1
53, 78, 43, 13, 18 .... this changes if you do not specify a seed

dim(small_wine.df)
```

Output: [1] 25 14

R-CODE

```
small_wine.df          # see that indeed we are not using the entire dataset
```

Output:

```
     v1    v2   v3   v4   v5  v6   v7   v8   v9  v10   v11  v12  v13  v14
175   3 13.27 4.28 2.26 20.0 120 1.59 0.69 0.43 1.35 10.20 0.59 1.56  835
24    1 13.50 1.81 2.61 20.0  96 2.53 2.61 0.28 1.66  3.52 1.12 3.82  845
97    2 12.29 1.41 1.98 16.0  85 2.55 2.50 0.29 1.77  2.90 1.23 2.74  428
51    1 13.83 1.65 2.60 17.2  94 2.45 2.99 0.22 2.29  5.60 1.24 3.37 1265
5     1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97  6.75 1.05 2.85 1450
121   2 11.56 2.05 3.23 28.5 119 3.18 5.08 0.47 1.87  6.00 0.93 3.69  465
104   2 12.51 1.73 1.98 20.5  85 2.20 1.92 0.32 1.48  2.94 1.04 3.57  672
15    1 13.63 1.81 2.70 17.2 112 2.85 2.91 0.30 1.46  7.30 1.28 2.88 1310
149   3 13.08 3.90 2.36 21.5 113 1.41 1.39 0.34 1.14  9.40 0.57 1.33  550
18    1 14.19 1.59 2.48 16.5 108 3.30 3.93 0.32 1.86  8.70 1.23 2.82 1680
78    2 12.33 0.99 1.95 14.8 136 1.90 1.85 0.35 2.76  3.40 1.06 2.31  750
108   2 12.22 1.29 1.94 19.0  92 2.36 2.04 0.39 2.08  2.70 0.86 3.02  312
146   3 13.88 5.04 2.23 20.0  80 0.98 0.34 0.40 0.68  4.90 0.58 1.33  415
161   3 13.69 3.26 2.54 20.0 107 1.83 0.56 0.50 0.80  5.88 0.96 1.82  680
111   2 12.52 2.43 2.17 21.0  88 2.55 2.27 0.26 1.22  2.00 0.90 2.78  325
```

```
 54   1 13.74 1.67 2.25 16.4 118 2.60 2.90 0.21 1.62  5.85 0.92 3.20 1060
 26   1 13.39 1.77 2.62 16.1  93 2.85 2.94 0.34 1.45  4.80 0.92 3.22 1195
 64   2 12.17 1.45 2.53 19.0 104 1.89 1.75 0.45 1.03  2.95 1.45 2.23  355
 166  3 13.45 3.70 2.60 23.0 111 1.70 0.92 0.43 1.46 10.68 0.85 1.56  695
 147  3 12.87 4.61 2.48 21.5  86 1.70 0.65 0.47 0.86  7.65 0.54 1.86  625
 58   1 13.72 1.43 2.50 16.7 108 3.40 3.67 0.19 2.04  6.80 0.89 2.87 1285
 154  3 12.58 1.29 2.10 20.0 103 1.48 0.58 0.53 1.40  7.60 0.58 1.55  640
 49   1 13.94 1.73 2.27 17.4 108 2.88 3.54 0.32 2.08  8.90 1.12 3.10 1260
 98   2 12.37 1.07 2.10 18.5  88 3.52 3.75 0.24 1.95  4.50 1.04 2.77  660
 60   2 12.33 1.10 2.28 16.0 101 2.05 1.09 0.63 0.41  3.27 1.25 1.67  680
```

R-CODE

```
#change the column names
colnames(small_wine.df) <- paste("v", 1:dim(small_wine.df)[2], sep="")    #
v1, v2, ....v13, v14
head(small_wine.df, n=5)

Output:
```

```
     v1    v2   v3   v4   v5  v6   v7   v8   v9  v10   v11  v12  v13  v14
 175  3 13.27 4.28 2.26 20.0 120 1.59 0.69 0.43 1.35 10.20 0.59 1.56  835
 24   1 13.50 1.81 2.61 20.0  96 2.53 2.61 0.28 1.66  3.52 1.12 3.82  845
 97   2 12.29 1.41 1.98 16.0  85 2.55 2.50 0.29 1.77  2.90 1.23 2.74  428
 51   1 13.83 1.65 2.60 17.2  94 2.45 2.99 0.22 2.29  5.60 1.24 3.37 1265
 5    1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97  6.75 1.05 2.85 1450
```

Step 4 of N: Handle Missing Values in data

After reading the dataset into R, do a summary on your data and deal with missing values in the data, See Remove/Replace/Deal with NA entries

R-CODE

```
#maybe
t(summary(small_wine.df))        # t(..) to transpose


Output:
```

```
v1 Min.   :1.00     1st Qu.:1.00     Median :2.00     Mean   :1.92     3rd Qu.:3.00     Max.   :3.00
v2 Min.   :11.56    1st Qu.:12.37    Median :13.27    Mean   :13.09    3rd Qu.:13.72    Max.   :14.20
v3 Min.   :0.990    1st Qu.:1.430    Median :1.730    Mean   :2.193    3rd Qu.:2.430    Max.   :5.040
v4 Min.   :1.940    1st Qu.:2.170    Median :2.360    Mean   :2.368    3rd Qu.:2.540    Max.   :3.230
v5 Min.   :14.80    1st Qu.:16.50    Median :19.00    Mean   :18.88    3rd Qu.:20.00    Max.   :28.50
v6 Min.   : 80.0    1st Qu.: 92.0    Median :104.0    Mean   :102.7    3rd Qu.:112.0    Max.   :136.0
v7 Min.   :0.980    1st Qu.:1.830    Median :2.450    Mean   :2.361    3rd Qu.:2.850    Max.   :3.520
v8 Min.   :0.34     1st Qu.:1.09     Median :2.27     Mean   :2.25     3rd Qu.:2.99     Max.   :5.08
v9 Min.   :0.1900   1st Qu.:0.2900   Median :0.3400   Mean   :0.3608   3rd Qu.:0.4300   Max.   :0.6300
```

```
    v10 Min.   :0.410    1st Qu.:1.220    Median :1.480    Mean   :1.548    3rd Qu.:1.950    Max.   :2.760
    v11 Min.   : 2.000   1st Qu.: 3.400   Median : 5.850   Mean   : 5.808   3rd Qu.: 7.600   Max.   :10.680
    v12 Min.   :0.540    1st Qu.:0.860    Median :0.960    Mean   :0.968    3rd Qu.:1.120    Max.   :1.450
    v13 Min.   :1.330    1st Qu.:1.820    Median :2.780    Mean   :2.557    3rd Qu.:3.100    Max.   :3.820
    v14 Min.   : 312.0   1st Qu.: 550.0   Median : 680.0   Mean   : 817.5   3rd Qu.:1195.0   Max.   :1680.0
```

R-CODE

```
# what about
apply(small_wine.df, 2, function(x) sum(is.na(x)))
```

Output:

```
  v1  v2  v3  v4  v5  v6  v7  v8  v9 v10 v11 v12 v13 v14
   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

R-CODE

```
# what about big wine
t(summary(big_wine.df))
```

Output:

```
    v1 Min.   :1.000    1st Qu.:1.000    Median :2.000    Mean   :1.944    3rd Qu.:3.000    Max.   :3.000
    v2 Min.   :11.03    1st Qu.:12.36    Median :13.05    Mean   :12.99    3rd Qu.:13.67    Max.   :14.83
    v3 Min.   :0.74     1st Qu.:1.60     Median :1.87     Mean   :2.34     3rd Qu.:3.10     Max.   :5.80
    v4 Min.   :1.360    1st Qu.:2.210    Median :2.360    Mean   :2.366    3rd Qu.:2.560    Max.   :3.230
    v5 Min.   :10.60    1st Qu.:17.20    Median :19.50    Mean   :19.52    3rd Qu.:21.50    Max.   :30.00
    v6 Min.   : 70.00   1st Qu.: 88.00   Median : 98.00   Mean   : 99.59   3rd Qu.:107.00   Max.   :162.00
    v7 Min.   :0.980    1st Qu.:1.740    Median :2.350    Mean   :2.292    3rd Qu.:2.800    Max.   :3.880
    v8 Min.   :0.340    1st Qu.:1.200    Median :2.130    Mean   :2.023    3rd Qu.:2.860    Max.   :5.080
    v9 Min.   :0.1300   1st Qu.:0.2700   Median :0.3400   Mean   :0.3623   3rd Qu.:0.4400   Max.   :0.6600
   v10 Min.   :0.410    1st Qu.:1.250    Median :1.550    Mean   :1.587    3rd Qu.:1.950    Max.   :3.580
   v11 Min.   : 1.280   1st Qu.: 3.210   Median : 4.680   Mean   : 5.055   3rd Qu.: 6.200   Max.   :13.000
   v12 Min.   :0.480    1st Qu.:0.780    Median :0.960    Mean   :0.957    3rd Qu.:1.120    Max.   :1.710
   v13 Min.   :1.270    1st Qu.:1.930    Median :2.780    Mean   :2.604    3rd Qu.:3.170    Max.   :4.000
   v14 Min.   : 278.0   1st Qu.: 500.0   Median : 672.0   Mean   : 745.1   3rd Qu.: 985.0   Max.   :1680.0
```

R-CODE

```
#Elegant way to report missing values in a data.frame:  http://stackoverflow.com/questions/8317231/elegant-way-to-report-missing-values-in-a-data-frame
sapply(big_wine.df, function(x) sum(is.na(x)))
```

Output:

```
  v1  v2  v3  v4  v5  v6  v7  v8  v9 v10 v11 v12 v13 v14
   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Step 4 of N: Plotting Multivariate Data

After reading the dataset into R, the next step is usually to plot of the resulting data See Scatterplot matrices in R

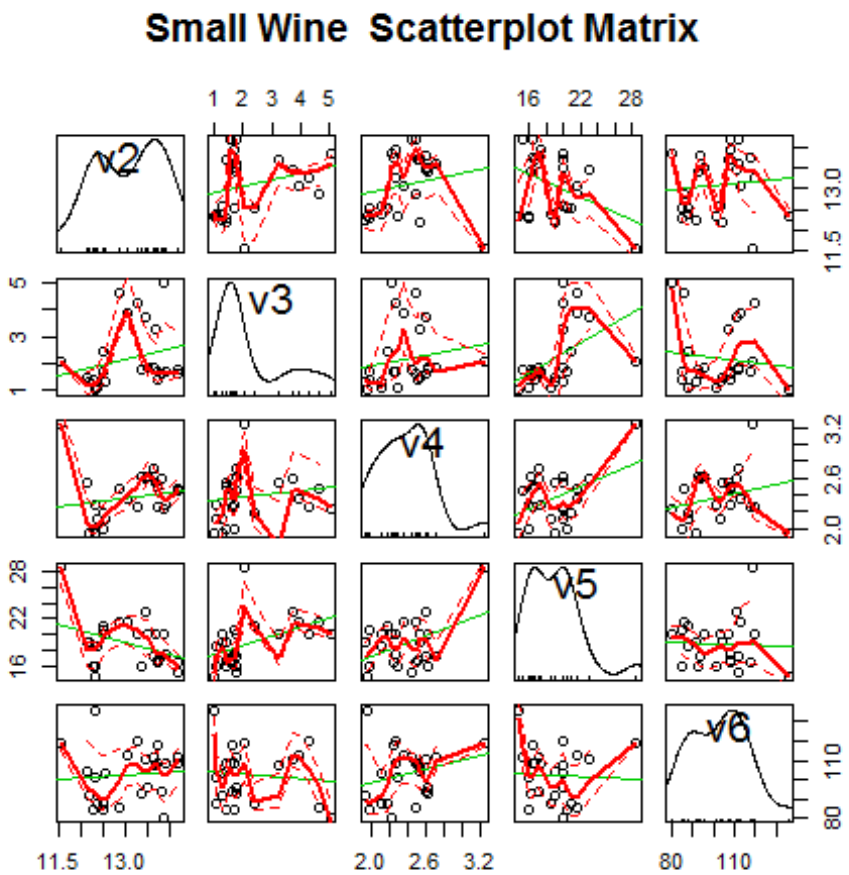
A Matrix Scatterplot

For multivariate data, one usually makes a matrix scatterplot, showing each pair of variables plotted against each other. The scatterplotMatrix(...) function is available in the car package. That is: install.packages("car"); library(car)

R-CODE

```
#-
# Plot #1: A matrix scatterplot
windows()
scatterplotMatrix(small_wine.df[, 2:6],                    # exclude column 1,
class label
                  main="Small Wine  Scatterplot Matrix"    # title of plot
                  )
```

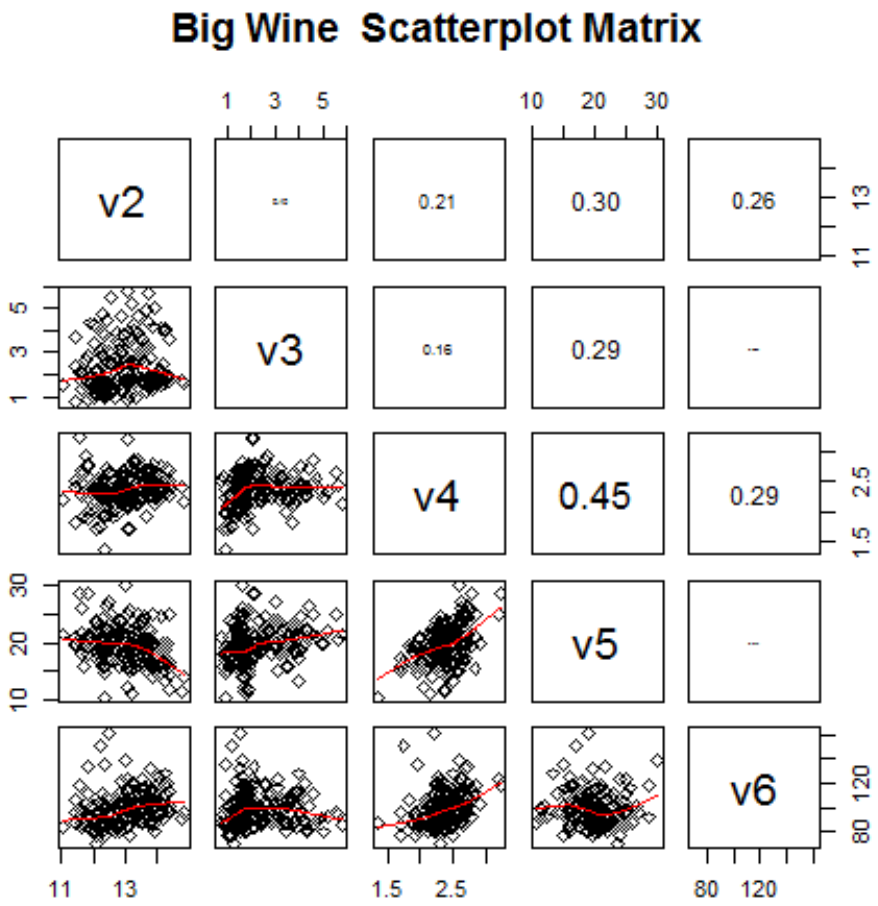
Output:



R-CODE

```
#--
# panel.smooth function is built in.
# panel.cor puts correlation in upper panels, size proportional to correlatio
n
panel.cor <- function(x, y, digits=2, prefix="", cex.cor, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits=digits)[1]
    txt <- paste(prefix, txt, sep="")
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}

# Plot #2: same as above, but add loess smoother in lower and correlation in
upper
windows()
pairs(big_wine.df[, 2:6],
      lower.panel=panel.smooth,   # lower portion of plot is the smooth plot
      upper.panel=panel.cor,     # uppper portion is a correlation plot
      pch=23,               # plot symbols, see http://www.statmethods.net/ad
vgraphs/parameters.html
      main="Big Wine  Scatterplot Matrix",
      na.action = na.omit       # causes cases with missing values in any of
the variables to be omitted entirely.
      )
```

Output:



Scatterplot with data Points labelled by group

Observation If you observe any interesting scatterplot for any two variables in the matrix scatterplot, plot that scatterplot in more detail, with the data points labelled by their group (their cultivar, i.e., v1 .. v14)

For example, the pair plot above, shows the 3rd column of the 4th row down is a scatterplot of V4 (x-axis) against V5 (y-axis). The figure shows a correlation of 0.45, that is a positive relationship between V4 and V5.

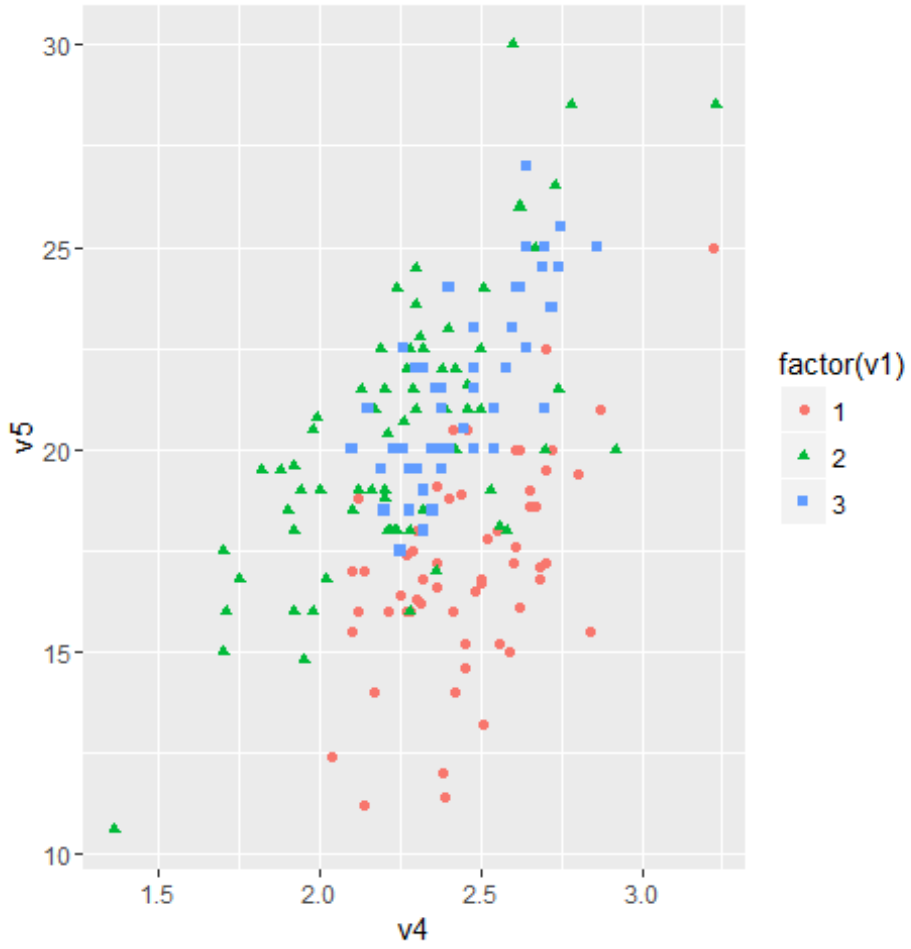
Zoom in on any interesting relationship with a plot

#Use plot(x= , y=) or qplot(x= , y=). I will use ggplot(data= ...)

R-CODE

```
# simplr plot
## define base for the graphs and store in object 'p'

v4_v5.plot <- ggplot(data = big_wine.df, aes(x = v4, y = v5, group=v1, color=
factor(v1), shape=factor(v1)))

## just plotting points (a.k.a., scatterplot)
windows()
v4_v5.plot + geom_point() # +  #geom_line()     # simple spaghetti plot
```

Output

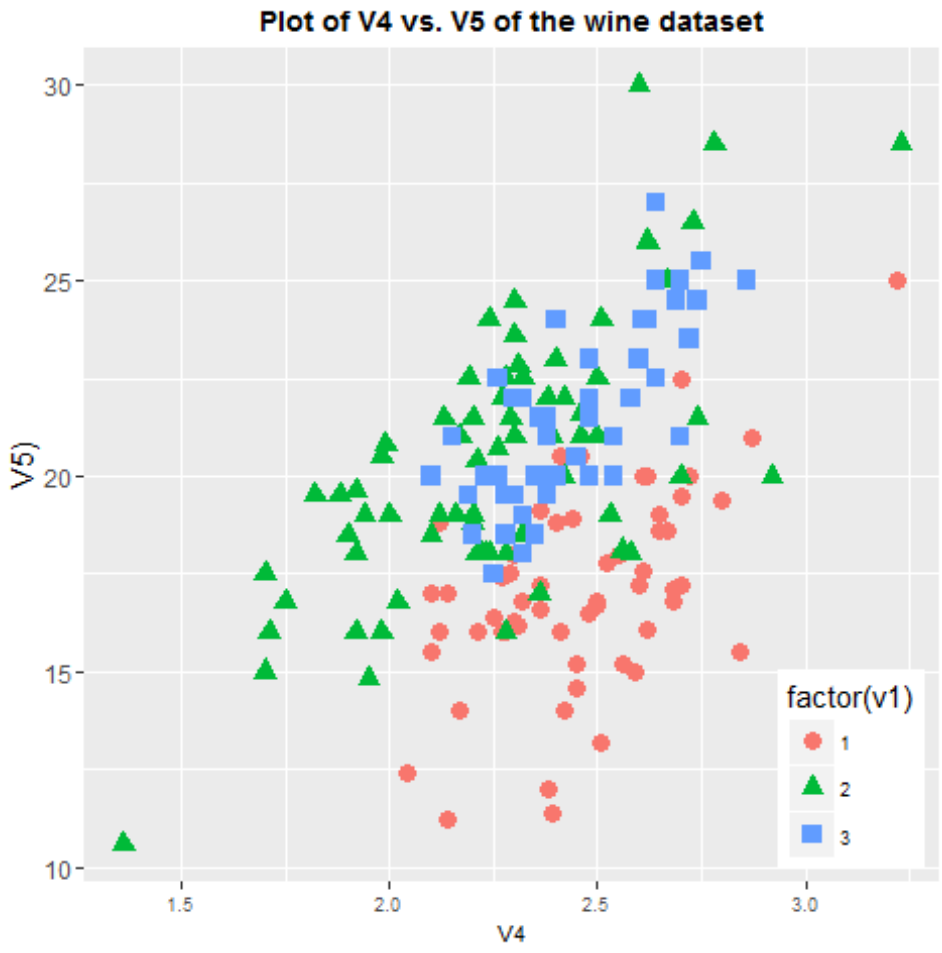


R-CODE

```
#---
# A more elaborate plot
# setup plot
v4_v5.plot <- ggplot(                    # cool plotting package
  data=big_wine.df,                      # dataset in this case a data.frame
  aes(
         x = v4,                         # x-axis
         y = v5,                         # y-axis
     colour= factor(v1),                 # color to use for the points
     shape = factor(v1)
    )) +
  geom_point( size = 3)   +              # plot points only, with increased
size
  xlab("V4")     +                       # x and y lables
  ylab("V5)")    +
  ggtitle("Plot of V4 vs. V5 of the wine dataset")



v4_v5.plot <- v4_v5.plot +
  theme(                        # the beauty of ggplot, you can add layers lat
er
           plot.title = element_text(lineheight=1.2, face="bold", size=10.5 )
,  #beautify title
           legend.position = c(1.01, 0.28), legend.justification=c(1,1),
#position the legend
      legend.position = "bottom", legend.justification=c(1,1),
#position the legend
          legend.text = element_text(size=6.5),
#format legend text
          axis.text.x = element_text(size=6.5),
#format x-axis text
         axis.title.x = element_text(size = rel(0.75))
#format x-axis title
  )

#--
```

```
#open a display window, else R will use RStudio's window
windows()
#-
# Add aesthetic mappings
#v4_v5.plot <- v4_v5.plot + geom_text(aes(colour = factor(v1), size=2))

# now display the plot
v4_v5.plot
```

Output



Observation fom detailed plot

We can see from the plot of x=V4 versus y=V5 that
wines from cultivar 1 seem to have lower values of V5 compared to the wines of cultivar 2 and 3
wines from cultivar 2 seem to have lower values of V4 compared to the wines of cultivar 1

A Profile Plot

Good reference for profile plot

Useful functions for Multivariate data analysis

Another useful plot is a profile plot which plots the value of each of the variables for each of the samples to show the variation in each of the variables.

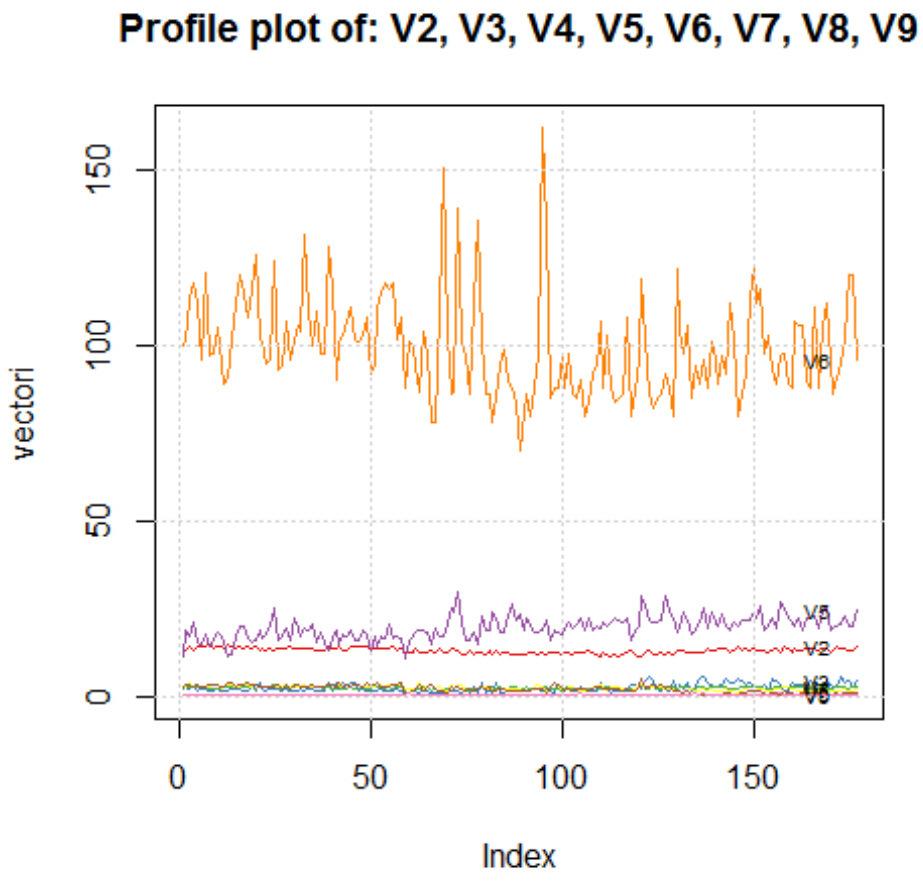
- Use the function makeProfilePlot(...) to make a profile plot. ++ Note: The makeProfilePlot(...) function requires the RColorBrewer library

A profile plot of the concentrations of the first 9 chemicals in the wine samples in columns V2, V3, V4, V5, V6, v7, v8, v9 of the big_wine dataset

R-CODE

```
source("helper_functions/pmr_tut_multivariatedataanalysis_edited_by_william.r
")

#  Gives us access to the following functions:
# 1: makeProfilePlot

chem.names <- paste("V", 2:9, sep="")

mylist <- list(big_wine.df$v2, big_wine.df$v3, big_wine.df$v4, big_wine.df$v5
,
               big_wine.df$v6, big_wine.df$v7, big_wine.df$v8, big_wine.df$v9
)

#--
windows()

makeProfilePlot(mylist, chem.names,
                title=sprintf("Profile plot of: %s", paste(chem.names, collap
se = ', '))
                )
grid()
```

```
Output
```



Observation fom detailed plot

We can see from the profile plot that * the mean and standard deviation for V6 is quite a lot higher than that for the other variables.

Calculating Summary Statistics for Multivariate Data

- For statistics on a dataset refer to the Mosaic package.
- Here we would calculate 2 summary statistics for each of the variables in the wine data set
- Use sapply
  mean
  standard deviation

R-CODE

```
sapply(big_wine.df[, 2:9], mean)        #mean and std of 2nd to 9th variables
```

Output

```
        v2         v3         v4         v5         v6         v7         v8         v9
12.9936723  2.3398870  2.3661582 19.5169492 99.5875706  2.2922599  2.0234463  0.3623164
```

R-CODE

```
sapply(big_wine.df[, 2:9], sd)
```

Output

```
        v2         v3         v4         v5         v6         v7         v8         v9
 0.8088084  1.1193144  0.2750804  3.3360711 14.1740185  0.6264651  0.9986576  0.1246529
```

R-CODE

```
# per sample type
group1_index = which(big_wine.df$v1 == 1); head(group1_index)   # get the ind
ex
```

Output

```
[1] 1 2 3 4 5 6
```

R-CODE

```
group2_index = which(big_wine.df$v1 == 2); head(group2_index)
```

Output

```
[1] 59 60 61 62 63 64
```

R-CODE

```
group3_index = which(big_wine.df$v1 == 3); head(group3_index)
```

Output

```
[1] 130 131 132 133 134 135
```

R-CODE

```
sapply(big_wine.df[ group1_index, 2:9], mean)          #mean and std of 2nd to 9th variables
```

Output

```
        v2         v3         v4         v5         v6         v7         v8         v9
13.7363793  2.0158621  2.4560345 17.0620690 105.9827586  2.8408621  2.9810345  0.2901724
```

R-CODE

```
sapply(big_wine.df[ group2_index, 2:9], mean)
```

Output

```
       v2        v3        v4        v5        v6        v7        v8        v9
12.278732  1.932676  2.244789 20.238028 94.549296  2.258873  2.080845  0.363662
```

R-CODE

```
sapply(big_wine.df[ group3_index, 2:9], mean)
```

Output

```
        v2         v3         v4         v5         v6         v7         v8         v9
13.1537500  3.3337500  2.4370833 21.4166667 99.3125000  1.6787500  0.7814583  0.4475000
```

Observation

- We are only able to compute the mean and standard deviation of the 2-9 chemicals' concentrations for just cultivar 1 samples, or for just cultivar 3 samples, in a similar way. statistics by variable for the entire group.
- What about by group with with just 1 line of command? Use google, e.g.: "r calculating standard deviation by group in a data"

Means and Variances Per Group

R-CODE

```
big_wine_groups.df <- t( big_wine.df %>% group_by(v1) %>% summarise_each(funs
(mean, sd)) )  # transpose
colnames(big_wine_groups.df) <- c("Group_1", "Group_2", "Group_3")

#--
# exclude row 1, V1
big_wine_groups.df <- big_wine_groups.df[-(1), ]
head(big_wine_groups.df)
```

Output

```
            Group_1   Group_2   Group_3
 v2_mean  13.736379 12.278732 13.153750
 v3_mean   2.015862  1.932676  3.333750
 v4_mean   2.456034  2.244789  2.437083
 v5_mean  17.062069 20.238028 21.416667
 v6_mean 105.982759 94.549296 99.312500
 v7_mean   2.840862  2.258873  1.678750
```

R-CODE

```
#show me rowsnames containing the word "mean"
# Use google: search for: "R show rownames that contain a word""
#http://stackoverflow.com/questions/13043928/selecting-rows-where-a-column-ha
s-a-string-like-hsa-partial-string-match

index_of_row_name_contain_mean <- grep("_mean", rownames(big_wine_groups.df))
```

Output

```
2  3  4  5  6  7  8  9 10 11 12 13 14
```

R-CODE

```
#-
# Now show the rows containing the word "mean"
group.means <- big_wine_groups.df[index_of_row_name_contain_mean, ]
group.means
```

Output

```
               Group_1    Group_2     Group_3
 v2_mean    13.7363793  12.278732  13.1537500
 v3_mean     2.0158621   1.932676   3.3337500
 v4_mean     2.4560345   2.244789   2.4370833
 v5_mean    17.0620690  20.238028  21.4166667
 v6_mean   105.9827586  94.549296  99.3125000
```

```
v7_mean     2.8408621   2.258873   1.6787500
v8_mean     2.9810345   2.080845   0.7814583
v9_mean     0.2901724   0.363662   0.4475000
v10_mean    1.8925862   1.630282   1.1535417
v11_mean    5.5263793   3.086620   7.3962500
v12_mean    1.0624138   1.056282   0.6827083
v13_mean    3.1446552   2.785352   1.6835417
v14_mean 1116.5862069 519.507042 629.8958333
```

R-CODE

```
#---
# do the same for variance
group.variance <- big_wine_groups.df[-c(index_of_row_name_contain_mean), ]
group.variance
```

Output

```
              Group_1     Group_2     Group_3
v2_sd      0.46163211   0.5379642   0.5302413
v3_sd      0.69340005   1.0155687   1.0879057
v4_sd      0.22912449   0.3154673   0.1846902
v5_sd      2.56137488   3.3497704   2.2581609
v6_sd     10.22465438  16.7534975  10.8904726
v7_sd      0.34187966   0.5453611   0.3569709
v8_sd      0.40083111   0.7057008   0.2935041
v9_sd      0.07064841   0.1239613   0.1241396
v10_sd     0.41241931   0.6020678   0.4088359
v11_sd     1.24930114   0.9249293   2.3109421
v12_sd     0.11746310   0.2029368   0.1144411
v13_sd     0.34550353   0.4965735   0.2721114
v14_sd   223.35276437 157.2112204 115.0970432
```

*** NEXT WE ARE GOING EXPLORE HOW WORK WITH A TOY DATASET ***

Reading from a toy dataset

Most times it is better doing the exploratory work using a smaller dataset and then use the code written on your big dataset. An example is shown below:

R-CODE

```
toy.data <- read.table(header = TRUE, text = "
Group_1 Group_2 Group_3
4.0 2.9 4.5
3.6 2.3 3.8
3.7 2.9 4.0
4.1 3.5 5.2
3.9 3.7 3.9
4.0 3.0 4.1
")

toy.data                                          #show the data
```

Output

```
  Group_1 Group_2 Group_3
1     4.0     2.9     4.5
2     3.6     2.3     3.8
3     3.7     2.9     4.0
4     4.1     3.5     5.2
5     3.9     3.7     3.9
6     4.0     3.0     4.1

#--
```

R-CODE

```
toy.df <<- melt(toy.data)

## Using  as id variables

colnames(toy.df) <- c("v1", "v2")
toy.df
```

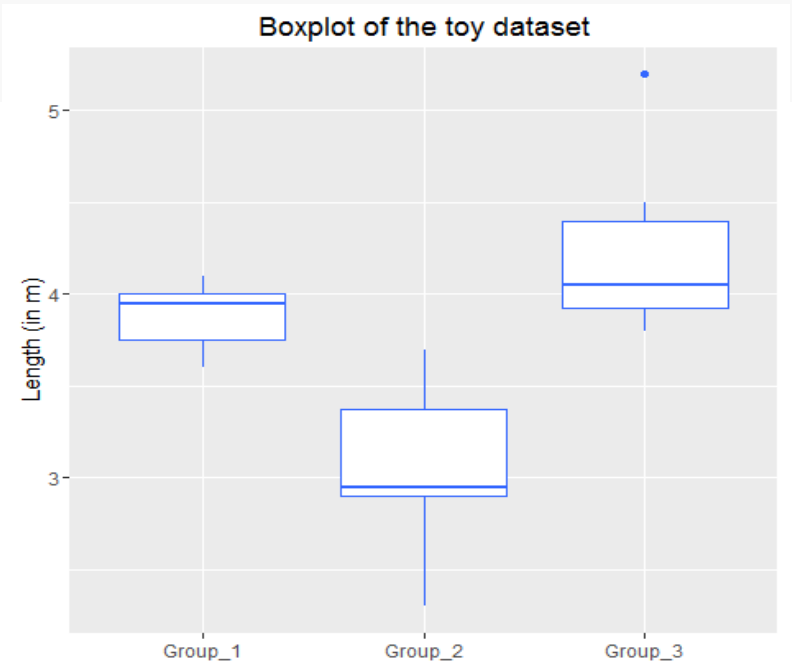
Output

```
         v1  v2
1  Group_1 4.0
2  Group_1 3.6
3  Group_1 3.7
4  Group_1 4.1
5  Group_1 3.9
6  Group_1 4.0
```

```
 7  Group_2 2.9
 8  Group_2 2.3
 9  Group_2 2.9
 10 Group_2 3.5
 11 Group_2 3.7
 12 Group_2 3.0
 13 Group_3 4.5
 14 Group_3 3.8
 15 Group_3 4.0
 16 Group_3 5.2
 17 Group_3 3.9
 18 Group_3 4.1


##### Boxplot
p <- ggplot(toy.df, aes(v1, v2))
p <- p + geom_boxplot(fill = "white", colour = "#3366FF") +
     ylab("Length (in m)") + xlab("") +
     ggtitle("Boxplot of the toy dataset")
p  # display plot
```

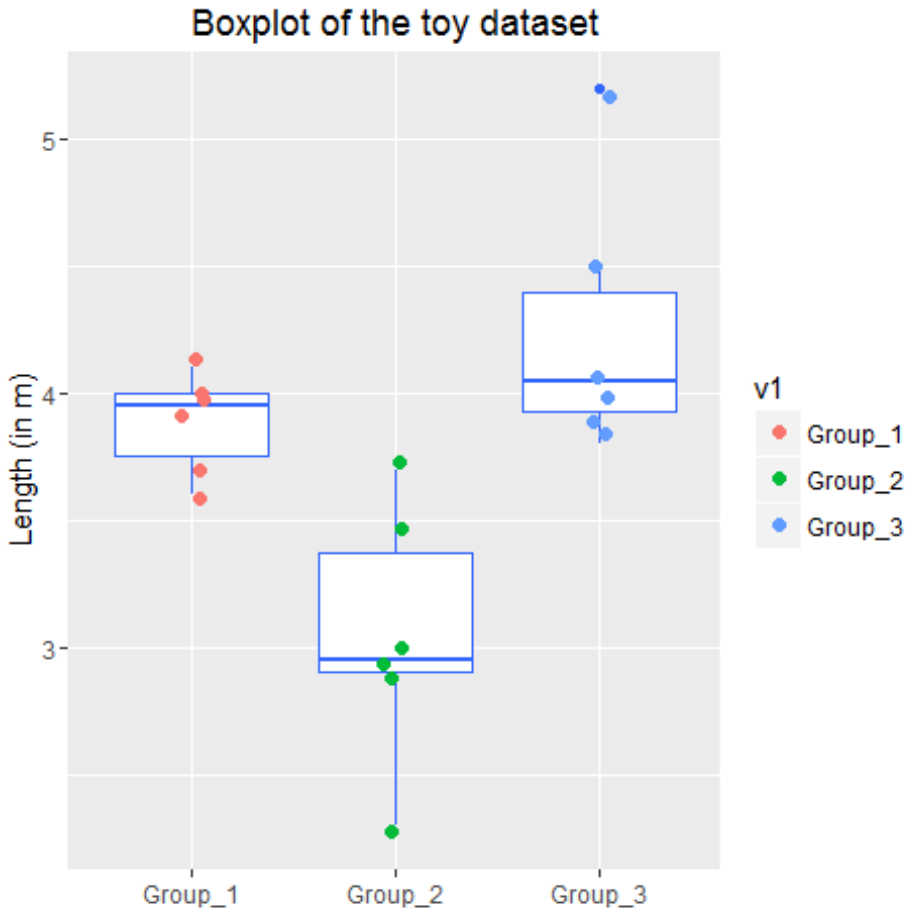
```
#ADD POINTS IF YOU WISH
p <- p + geom_jitter(width = 0.2, aes(colour = v1), size=2)   #add points
p   # display plot
```

Output



```r
# beautify plot
p <- p + theme(
         legend.position = c(1.01, 0.19), legend.justification=c(1,1),      #p
osition the legend
        legend.position = "bottom", legend.justification=c(1,1),            #p
osition the legend
          legend.text = element_text(size=6.5))                            #fo
rmat legend text

#plot
windows()
p
```

Output



Descriptive Statistics

```
#--
# Get N
N.dim = dim( toy.df )
N <- ( N.dim[1] * N.dim[2] ) - N.dim[1]        # could have simply said N.dim
[1], but setting program up for big_wine.df
N

Output [1] 18
```

R-CODE

```
# group counts
n <- toy.df %>% group_by(v1) %>% summarise_each(funs(length))
n                                                  # data frame [3 x 14]

Output

# Source: local data frame [3 x 2]

        v1    v2
    (fctr) (int)
 1 Group_1     6
 2 Group_2     6
 3 Group_3     6

#-
```

R-CODE

```
K = 3                                              # K = number of groups
#--
#  Note
# mean(big_wine.df) returns column means
# mean( as.matrix(big_wine.df ) ) # one value.

x_2bar <-  mean(as.matrix(toy.df[, -c(1)] ))
x_2bar



Output

[1] 3.727778

#-
```

R-CODE

```
toy.table <- toy.df %>% group_by(v1) %>% summarise_each(funs(length, mean, sd
, min, max))
toy.table$stderr     <- toy.table$sd / sqrt(toy.table$length)
toy.table$E          <- qnorm(.975)*toy.table$stderr        # margin of error
95% CI
toy.table$lowerbound <- toy.table$mean - toy.table$E
toy.table$upperbound <- toy.table$mean + toy.table$E

descriptive.stats <- data.frame(
           Groups = c(1:3),
                N = c(toy.table$length),
             Mean = c(toy.table$mean),
  "Std Deviation" = c(toy.table$sd),
      "Std Error" = c(toy.table$stderr),
 "CI_Lower Bound" = c(toy.table$lowerbound),
 "CI_Upper Bound" = c(toy.table$upperbound),
        "Minimum" = c(toy.table$min),
       "Maximum" = c(toy.table$max)
 )


#--

#show table

Descriptive.stats
```

Output

```
#  Groups  N     Mean Std.Deviation  Std.Error CI_Lower.Bound CI_Upper.Bound Minimum Maximum
# 1      1  6 3.883333     0.1940790 0.07923243       3.728041       4.038626     3.6     4.1
# 2      2  6 3.050000     0.4969909 0.20289570       2.652332       3.447668     2.3     3.7
# 3      3  6 4.250000     0.5244044 0.21408721       3.830397       4.669603     3.8     5.2
```

***** We will extend this to out wine dataset in out next lecture *****