STATS 419 Analysis of Multivariate Analysis

Week 3 Assignment

Michaela Bayerlova
(michaela.bayerlova@wsu.edu)
[]

Instructor: Monte J. Shaffer

23 September 2020

```
library(devtools); # required for function source_url to work
github.path = "https://raw.githubusercontent.com/MichaelaB1/WSU_STATS419_FALL2020/";
source_url(paste0(github.path, "master/functions/libraries.R"));

source_url(paste0(github.path, "master/functions/functions-imdb.R"));
```

1 Matrix

Create the "rotate matrix" functions described in lecture. Apply to the example "myMatrix".

```
source_url(paste0(github.path, "master/functions/functions-matrix.R"));

myMatrix = matrix( c (
                              1, 0, 2,
                              0, 3, 0,
                              4, 0, 5
                                 ), nrow=3, byrow=T);

transposeMatrix = function(mat)
{
  t(mat);
          }
```

```
# clockwise
rotateMatrix90 = function(mat)
{
  t(mat[nrow(mat):1,,drop=FALSE]);
}

rotateMatrix180 = function(mat)
{
  rotateMatrix90(rotateMatrix90(mat));
```

```
}

rotateMatrix270 = function(mat)
{
  rotateMatrix90(rotateMatrix90(rotateMatrix90(mat)));
}


# counter clockwise
rotateMatrix90_cc = function(mat)
{
  apply(t(mat), 2, rev);
}

rotateMatrix180_cc = function(mat)
{
  rotateMatrix90_cc(rotateMatrix90_cc(mat));
}

rotateMatrix270_cc = function(mat)
{
  rotateMatrix90_cc(rotateMatrix90_cc(rotateMatrix90_cc(mat)));
}
```

```
#source_url(paste0(github.path, "master/WEEK-03/functions/HW2_functions.R"));
# Rotate clockwise
rotateMatrix90(myMatrix);
```

```
##      [,1] [,2] [,3]
## [1,]    4    0    1
## [2,]    0    3    0
## [3,]    5    0    2
```

```
rotateMatrix180(myMatrix);
```

```
##      [,1] [,2] [,3]
## [1,]    5    0    4
## [2,]    0    3    0
## [3,]    2    0    1
```

```
rotateMatrix270(myMatrix);
```

```
##      [,1] [,2] [,3]
## [1,]    2    0    5
## [2,]    0    3    0
## [3,]    1    0    4
```

```
# Rotate counter clockwise
rotateMatrix90_cc(myMatrix);
```

```
##      [,1] [,2] [,3]
## [1,]    2    0    5
## [2,]    0    3    0
## [3,]    1    0    4
```

```
rotateMatrix180_cc(myMatrix);
```

```
##      [,1] [,2] [,3]
## [1,]    5    0    4
## [2,]    0    3    0
## [3,]    2    0    1
```

```
rotateMatrix270_cc(myMatrix);
```

```
##      [,1] [,2] [,3]
## [1,]    4    0    1
## [2,]    0    3    0
## [3,]    5    0    2
```

2 IRIS Scatterplot

Recreate the graphic for the IRIS Data Set using R. Same titles, same scales, same colors. See: https://en.wikipedia.org/wiki/Iris_flower_data_set#/media/File:Iris_dataset_scatterplot.svg

```
# Iris Data Set
IRIS_Data = iris
class(iris)
```

```
## [1] "data.frame"
```

```
# Scatterplot
pairs(IRIS_Data[1:4], main = "Iris Data (red=setosa,green=versicolor,blue=virginica)",
      pch = 21, bg = c("red", "green", "blue")[unclass(IRIS_Data$Species)])
```

3 IRIS Question

Right 2-3 sentences concisely defining the IRIS Data Set. Maybe search KAGGLE for a nice template. Be certain the final writeup are your own sentences (make certain you modify what you find, make it your own, but also cite where you got your ideas from). NOTE: Watch the video, Figure 8 has a +5 EASTER EGG.

The Iris flower data set is a multivariate data set. The data set contains four measurements (sepals length and width, petals length and width) for 150 records of flowers. Each is represented in the three species of iris: Iris setosa, Iris versicolor and Iris virginica. The Iris setosa is from a wide range across the Arctic sea. The Iris versicolor is found in North America, like Eastern United States and Eastern Canada. The Iris virginica is native to eastern North America.

4 Personality

Import "personality-raw.txt" into R. Remove the V00 column. Create two new columns from the current column "date_test": year and week. Stack Overflow may help: https://stackoverflow.com/questions/22439540/how-to-get-week-numbers-from-dates ... Sort the new data frame by YEAR, WEEK so the newest tests are first ... The newest tests (e.g., 2020 or 2019) are at the top of the data frame. Then remove duplicates using the unique function based on the column "md5_email". Save the data frame in the same "pipe-delimited format" (| is a pipe) with the headers. You will keep the new data frame as "personality-clean.txt" for future work (you will not upload it at this time). In the homework, for this tasks, report how many records your raw dataset had and how many records your clean dataset has.

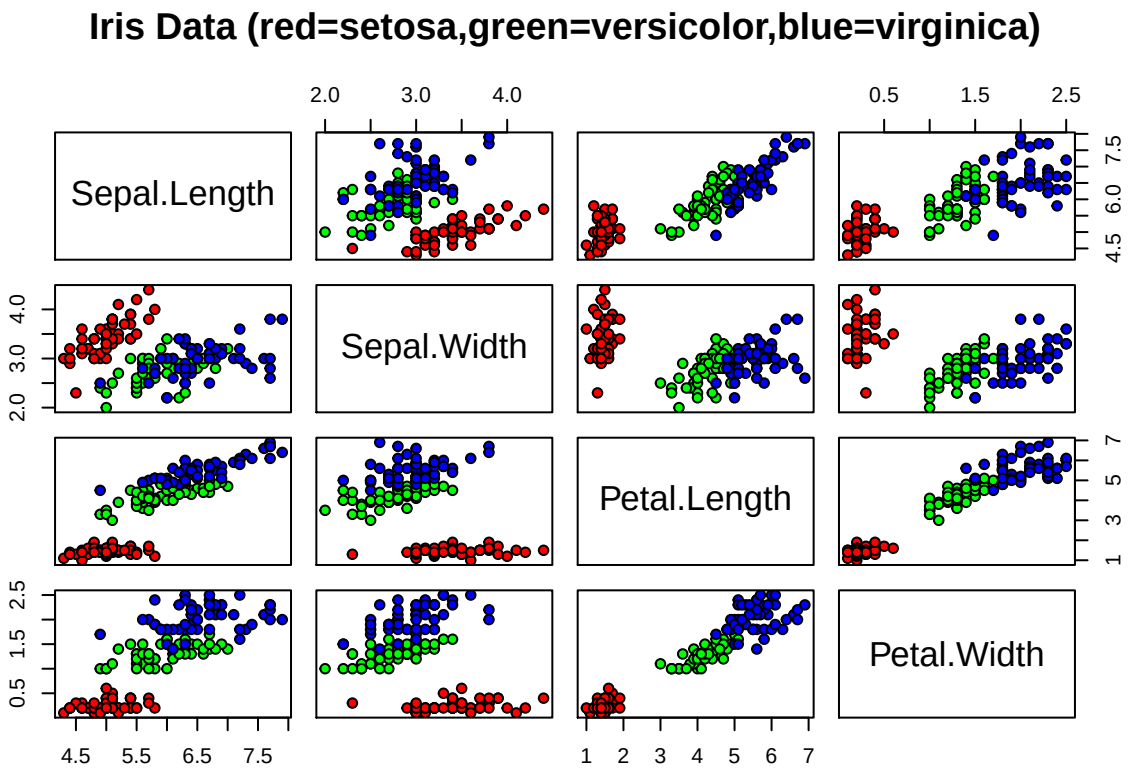


Figure 1: Iris Data scatterplot

```
personality.folder = "C:/Users/michaela.bayerlova/Documents/STATS 419/reading assignments/";
raw.file = paste0(personality.folder, "personality-raw.txt");

#my_data = read.delim("C:\\Users\\michaela.bayerlova\\Documents\\STATS 419\\reading assignments\\persona

my_data = utils::read.csv(raw.file, header=TRUE, sep="|");

#head(my_data);
nrow_raw_data <- nrow(my_data);
dim(my_data);
```

```
## [1] 838  63
```

```
# remove column VOO
my_data$VO0 <- NULL;

# Create two new columns from the current column "date_test":  year and week
d = strptime(my_data$date_test, format="%m/%d/%Y %H:%M");
d.year = as.numeric(strftime(d, format='%Y'));
d.week = as.numeric(strftime(d, format='%W'));
my_data$year = d.year;
my_data$week = d.week;
date_index = which(names(my_data)=="date_test");

# Remove date_test
my_data$date_test <- NULL;

# Sort the new data frame by YEAR, WEEK so the newest tests are first
# my_data <- arrange(my_data, c("year","week"),);
df.dim = dim(my_data);
df.pos = df.dim[2];
which( names(my_data)=="year" );
```

```
## [1] 62
```

```
which( names(my_data)=="week" );
```

```
## [1] 63
```

```
my_data = my_data[, c(1,62,63, 2:61)];

if(date_index == 1)
{
  reorder = c( (df.pos-1):df.pos, 2:(df.pos-2) );
}else{
     reorder = c( 1:(date_index-1), (df.pos-1):df.pos,  2:(df.pos-2) );
     }

my_data = my_data[, reorder];
my_data_sorted = my_data[ order(-my_data[,date_index],-my_data[,(1+date_index)]), ];
#head(my_data_sorted[,1:5]);

# Remove duplicates using the unique function based on the column "md5_email", only leave most recent
```

```
#my_data <- my_data[!duplicated(my_data$md5_email),];
u = unique( my_data_sorted["md5_email"] );
my_data_clean = my_data_sorted[!duplicated(my_data_sorted["md5_email"]), ];

# In the homework, for this tasks, report how many records your raw dataset had and
# how many records your clean dataset has
nrow_clean_data <- nrow(my_data_clean)
dim(my_data_clean);
```

```
## [1] 678  63
```

```
#install.packages("data.table")
#library(data.table);
#fwrite(my_data_clean,"personality-clean.txt", sep="|",col.names = TRUE, row.names = FALSE);

clean.file = paste0(personality.folder, "personality-clean.txt");


utils::write.table(my_data_clean, file=clean.file, quote=FALSE, col.names=TRUE, row.names=FALSE, sep="|"


#utils::read.csv(mycache, header=TRUE, sep="|");
```

5 Variance and Z-scores

Write functions for doSummary and sampleVariance and doMode . . . test these functions in your homework on the "monte.shaffer@gmail.com" record from the clean dataset. Report your findings. For this "monte.shaffer@gmail.com" record, also create z-scores. Plot(x,y) where x is the raw scores for "monte.shaffer@gmail.com" and y is the z-scores from those raw scores. Include the plot in your assignment, and write 2 sentences describing what pattern you are seeing and why this pattern is present.

```
library(humanVerseWSU);
```

```
##
## Attaching package: 'humanVerseWSU'
```

```
## The following objects are masked _by_ '.GlobalEnv':
##
##     rotateMatrix, rotateMatrix180, rotateMatrix270, rotateMatrix90,
##     rotateMatrixAngle, transposeMatrix
```

```
x.norm = rnorm(100,0,1);
s.norm = doStatsSummary ( x.norm );
str(s.norm);  # mode is pretty meaningless on this data
```

```
## List of 32
##  $ length           : int 100
##  $ length.na        : int 0
##  $ length.good      : int 100
##  $ mean             : num -0.00974
##  $ mean.trim.05     : num -0.0041
##  $ mean.trim.20     : num 0.0288
```

```
##  $ median           : num 0.125
##  $ MAD              : num 1.02
##  $ IQR              : num 1.42
##  $ quartiles        : Named num [1:3] -0.809 0.125 0.614
##   ..- attr(*, "names")= chr [1:3] "25%" "50%" "75%"
##  $ deciles          : Named num [1:9] -1.29 -0.868 -0.545 -0.164 0.125 ...
##   ..- attr(*, "names")= chr [1:9] "10%" "20%" "30%" "40%" ...
##  $ centiles         : Named num [1:99] -2.12 -1.65 -1.49 -1.46 -1.45 ...
##   ..- attr(*, "names")= chr [1:99] "1%" "2%" "3%" "4%" ...
##  $ median.weighted  : num 1.02
##  $ MAD.weighted     : num 0.125
##  $ max              : num 2.02
##  $ min              : num -2.32
##  $ range            : num 4.33
##  $ xlim             : num [1:2] -2.32 2.02
##  $ max.idx          : num 32
##  $ min.idx          : num 53
##  $ freq.max         : num [1:100] -2.32 -2.12 -1.64 -1.49 -1.46 ...
##  $ mode             : num [1:100] -2.32 -2.12 -1.64 -1.49 -1.46 ...
##  $ which.min.freq   : num [1:100] -2.32 -2.12 -1.64 -1.49 -1.46 ...
##  $ ylim             : int [1:2] 1 1
##  $ sd               : num 0.936
##  $ var              : num 0.876
##  $ var.naive        :List of 3
##   ..$ x.bar: num -0.00974
##   ..$ s.var: num 0.876
##   ..$ s.sd : num 0.936
##  $ var.2step        :List of 3
##   ..$ x.bar: num -0.00974
##   ..$ s.var: num 0.876
##   ..$ s.sd : num 0.936
##  $ shapiro          :List of 4
##   ..$ statistic: Named num 0.983
##   .. ..- attr(*, "names")= chr "W"
##   ..$ p.value  : num 0.215
##   ..$ method   : chr "Shapiro-Wilk normality test"
##   ..$ data.name: chr "xx"
##   ..- attr(*, "class")= chr "htest"
##  $ shapiro.is.normal:List of 3
##   ..$ 0.10: logi TRUE
##   ..$ 0.05: logi TRUE
##   ..$ 0.01: logi TRUE
##  $ outliers.z       :'data.frame':   0 obs. of  2 variables:
##   ..$ value    : Factor w/ 0 levels:
##   ..$ direction: Factor w/ 0 levels:
##  $ outliers.IQR     :'data.frame':   0 obs. of  3 variables:
##   ..$ value    : Factor w/ 0 levels:
##   ..$ fence    : Factor w/ 0 levels:
##   ..$ direction: Factor w/ 0 levels:
```

```
x.unif = runif(100,0,1);
s.unif = doStatsSummary ( x.unif );
str(s.unif);  # mode is pretty meaningless on this data
```

```
## List of 32
##  $ length           : int 100
##  $ length.na        : int 0
##  $ length.good      : int 100
##  $ mean             : num 0.423
##  $ mean.trim.05     : num 0.417
##  $ mean.trim.20     : num 0.409
##  $ median           : num 0.43
##  $ MAD              : num 0.385
##  $ IQR              : num 0.512
##  $ quartiles        : Named num [1:3] 0.155 0.43 0.668
##   ..- attr(*, "names")= chr [1:3] "25%" "50%" "75%"
##  $ deciles          : Named num [1:9] 0.052 0.121 0.186 0.275 0.43 ...
##   ..- attr(*, "names")= chr [1:9] "10%" "20%" "30%" "40%" ...
##  $ centiles         : Named num [1:99] 0.00266 0.00756 0.01458 0.01613 0.0321 ...
##   ..- attr(*, "names")= chr [1:99] "1%" "2%" "3%" "4%" ...
##  $ median.weighted  : num 0.385
##  $ MAD.weighted     : num 0.43
##  $ max              : num 0.99
##  $ min              : num 0.00179
##  $ range            : num 0.988
##  $ xlim             : num [1:2] 0.00179 0.98982
##  $ max.idx          : num 35
##  $ min.idx          : num 43
##  $ freq.max         : num [1:100] 0.00179 0.00267 0.00766 0.0148 0.01619 ...
##  $ mode             : num [1:100] 0.00179 0.00267 0.00766 0.0148 0.01619 ...
##  $ which.min.freq   : num [1:100] 0.00179 0.00267 0.00766 0.0148 0.01619 ...
##  $ ylim             : int [1:2] 1 1
##  $ sd               : num 0.29
##  $ var              : num 0.0843
##  $ var.naive        :List of 3
##   ..$ x.bar: num 0.423
##   ..$ s.var: num 0.0843
##   ..$ s.sd : num 0.29
##  $ var.2step        :List of 3
##   ..$ x.bar: num 0.423
##   ..$ s.var: num 0.0843
##   ..$ s.sd : num 0.29
##  $ shapiro          :List of 4
##   ..$ statistic: Named num 0.94
##   .. ..- attr(*, "names")= chr "W"
##   ..$ p.value  : num 0.00019
##   ..$ method   : chr "Shapiro-Wilk normality test"
##   ..$ data.name: chr "xx"
##   ..- attr(*, "class")= chr "htest"
##  $ shapiro.is.normal:List of 3
##   ..$ 0.10: logi FALSE
##   ..$ 0.05: logi FALSE
##   ..$ 0.01: logi FALSE
##  $ outliers.z       :'data.frame':   0 obs. of  2 variables:
##   ..$ value    : Factor w/ 0 levels:
##   ..$ direction: Factor w/ 0 levels:
##  $ outliers.IQR     :'data.frame':   0 obs. of  3 variables:
##   ..$ value    : Factor w/ 0 levels:
```

```
##   ..$ fence    : Factor w/ 0 levels:
##   ..$ direction: Factor w/ 0 levels:
```

```
#####################################################################

doStatsSummary = function(x)
    {
    result = list();
        result$length = length(x);
    xx = stats::na.omit(x);
        result$length.na = length(x) - length(xx);
        result$length.good = length(xx);
    result$mean = mean(xx);
    result$mean.trim.05 = mean(xx, trim=0.05);
    result$mean.trim.20 = mean(xx, trim=0.20);

    result$median = stats::median(xx);
    result$MAD = stats::mad(xx);
    result$IQR = stats::IQR(xx);
    result$quartiles = stats::quantile(xx, prob=c(.25,.5,.75));
    result$deciles = stats::quantile(xx, prob=seq(0.1,0.9,by=0.1) );
    result$centiles = stats::quantile(xx, prob=seq(0.01,0.99,by=0.01) );

    result$median.weighted = matrixStats::weightedMad(xx);
    result$MAD.weighted = matrixStats::weightedMedian(xx);

    result$max = max(xx);
    result$min = min(xx);
    result$range = result$max - result$min;
    result$xlim = range(xx);

    result$max.idx = whichMax(x);
    result$min.idx = whichMin(x);

    result$mode = result$freq.max = doMode(x);  # elements with highest frequency
    result$which.min.freq = doModeOpposite(x);

    result$ylim = c( freqMin(xx), freqMax(xx) );

    # you could later get indexes of each mode(freq.max)/freq.min
    #using findAllIndexesWithValueInVector

    result$sd = stats::sd(xx);
    result$var = stats::var(xx);

    result$var.naive = doSampleVariance(x,"naive");
    result$var.2step = doSampleVariance(x,"2step");


    ## normality
    result$shapiro = stats::shapiro.test(xx);
    result$shapiro.is.normal = list("0.10" = isTRUE(result$shapiro$p.value > 0.10), "0.05" = isTRUE(res

    result$outliers.z = findOutliersUsingZscores(x);
```

```
    result$outliers.IQR = findOutliersUsingIQR(x);

    #result£z = calculateZscores(x);

    result;
  }

#####################################################################

doSampleVariance = function(x, method="two-pass")
    {
    x = stats::na.omit(x);
    if(method=="naive")
        {
        n = 0;
        sum = 0;
        sum2 = 0;

        for(i in 1:length(x))  ## stats::na.omit(x)
            {
            n = n + 1;
            sum = sum + x[i];
            sum2 = sum2 + x[i]*x[i];
            }

        if(n < 2) { return(NULL);} #
            x.bar = sum/n;
            s.var = (sum2 - (sum*sum)/n)/(n-1);

        } else  {
                # two-pass algorithm # testing
                n = sum = sum2 = 0;
                ## first pass
                for(i in 1:length(x))  ## stats::na.omit(x)
                    {
                    n = n + 1;
                    sum = sum + x[i];
                    }
        if(n < 2) { return(NULL);} #
                x.bar = sum/n;
                ## second pass
                for(i in 1:length(x))  ## stats::na.omit(x)
                    {
                    deviation = x[i] - x.bar;
                    sum2 = sum2 + deviation * deviation;
                    }
                s.var = sum2/(n-1);
                }

        s.sd = sqrt(s.var);
    list("x.bar"=x.bar,"s.var"=s.var,"s.sd"=s.sd);
    }
```

```
#####################################################################

doMode = function(x) # alias ?
    {
    whichMaxFreq(x);
  }


whichMaxFreq = function(x)  # doMode
    {
    x.table = as.data.frame( table(x) );
        freq.max = max( x.table$Freq );
    x.list = x.table[x.table$Freq==freq.max,];
    xs = as.numeric( as.vector (x.list$x) );
    xs;
    }

# R does not have a "mode" function built in that will capture ties.
#   This function will.
# In the process, I wrote other functions that are also not robust in R.
# For example ?which.max versus my function ?whichMax

which.max( c(87, presidents[1:30], 87) );
```

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

```
whichMax( c(87, presidents[1:30], 87) );
```

```
## [1]  1  3 32
```

```
## a function can also be referenced using class::method notation

base::which.max( c(87, presidents[1:30], 87) );
```

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

```
#humanVerseWUS::whichMax( c(87, presidents[1:30], 87) );

# typos ...
humanVerseWSU::whichMax( c(87, presidents[1:30], 87) );
```

```
## [1]  1  3 32
```

```
## this will prevent confusion if functions have the same name (in different packages)

#####################################################################

# not a requirement for your homework, but here is a function that will do it.
calculateZscores = function(x, x.bar=NULL, s.hat=NULL)
    {
  if(is.numeric(x.bar) && is.numeric(s.hat)) { return ((x - x.bar) / s.hat);}
  # maybe throw a warning if one is null, but not the other
```

```
  if( (is.null(x.bar) + is.null(s.hat)) == 1)
      {
      warning("Only one value was entered for x.bar / s.hat ... Computing these values instead.")
      }


    dsv = doSampleVariance(x);

    x.bar = dsv$x.bar;
    s.hat = dsv$s.sd;

    if(is.null(s.hat)) { return (NULL); }  # we take care of
    #division by zero in our custom sampleVarianceFunction

    (x - x.bar) / s.hat;
    }
```

5.1 Variance

5.1.1 Naive

```
# https://raw.githubusercontent.com/MonteShaffer/humanVerseWSU/
#master/humanVerseWSU/R/functions-standardize.R

v.norm = doSampleVariance(x.norm, "naive");


# if x is really small
vsmall.df = as.data.frame( t(unlist(v.norm)) );

x.small = x.norm;
for(i in 1:20)
  {
  # every loop make it 1000 times smaller
  # notice I am looping over "i" but not using it.
  x.small = standardizeToFactor(x.small, 1/1000);
  v.small = doSampleVariance(x.small, "naive");
    v.row = t(unlist(v.small));
  vsmall.df = rbind(vsmall.df, v.row);
  }

vsmall.df;
```

```
##            x.bar         s.var         s.sd
## 1  -9.744576e-03  8.761886e-01 9.360495e-01
## 2  -9.744576e-06  8.761886e-07 9.360495e-04
## 3  -9.744576e-09  8.761886e-13 9.360495e-07
## 4  -9.744576e-12  8.761886e-19 9.360495e-10
## 5  -9.744576e-15  8.761886e-25 9.360495e-13
## 6  -9.744576e-18  8.761886e-31 9.360495e-16
## 7  -9.744576e-21  8.761886e-37 9.360495e-19
## 8  -9.744576e-24  8.761886e-43 9.360495e-22
## 9  -9.744576e-27  8.761886e-49 9.360495e-25
```

```
## 10 -9.744576e-30  8.761886e-55 9.360495e-28
## 11 -9.744576e-33  8.761886e-61 9.360495e-31
## 12 -9.744576e-36  8.761886e-67 9.360495e-34
## 13 -9.744576e-39  8.761886e-73 9.360495e-37
## 14 -9.744576e-42  8.761886e-79 9.360495e-40
## 15 -9.744576e-45  8.761886e-85 9.360495e-43
## 16 -9.744576e-48  8.761886e-91 9.360495e-46
## 17 -9.744576e-51  8.761886e-97 9.360495e-49
## 18 -9.744576e-54 8.761886e-103 9.360495e-52
## 19 -9.744576e-57 8.761886e-109 9.360495e-55
## 20 -9.744576e-60 8.761886e-115 9.360495e-58
## 21 -9.744576e-63 8.761886e-121 9.360495e-61
```

```
# if x is really big
vlarge.df = as.data.frame( t(unlist(v.norm)) );

x.large = x.norm;
for(i in 1:20)
  {
  # every loop make it 1000 times larger
  # notice I am looping over "i" but not using it.
  x.large = standardizeToFactor(x.large, 1000);
  v.large = doSampleVariance(x.large, "naive");
    v.row = t(unlist(v.large));
  vlarge.df = rbind(vlarge.df, v.row);
  }

vlarge.df;
```

```
##            x.bar         s.var         s.sd
## 1  -9.744576e-03  8.761886e-01 9.360495e-01
## 2  -9.744576e+00  8.761886e+05 9.360495e+02
## 3  -9.744576e+03  8.761886e+11 9.360495e+05
## 4  -9.744576e+06  8.761886e+17 9.360495e+08
## 5  -9.744576e+09  8.761886e+23 9.360495e+11
## 6  -9.744576e+12  8.761886e+29 9.360495e+14
## 7  -9.744576e+15  8.761886e+35 9.360495e+17
## 8  -9.744576e+18  8.761886e+41 9.360495e+20
## 9  -9.744576e+21  8.761886e+47 9.360495e+23
## 10 -9.744576e+24  8.761886e+53 9.360495e+26
## 11 -9.744576e+27  8.761886e+59 9.360495e+29
## 12 -9.744576e+30  8.761886e+65 9.360495e+32
## 13 -9.744576e+33  8.761886e+71 9.360495e+35
## 14 -9.744576e+36  8.761886e+77 9.360495e+38
## 15 -9.744576e+39  8.761886e+83 9.360495e+41
## 16 -9.744576e+42  8.761886e+89 9.360495e+44
## 17 -9.744576e+45  8.761886e+95 9.360495e+47
## 18 -9.744576e+48 8.761886e+101 9.360495e+50
## 19 -9.744576e+51 8.761886e+107 9.360495e+53
## 20 -9.744576e+54 8.761886e+113 9.360495e+56
## 21 -9.744576e+57 8.761886e+119 9.360495e+59
```

```
## from these two experiments it looks okay!

## CS purists say it will fail eventually
## maybe I have to use a smaller n to demo failure?

## examine the function , the failure point is:    sum2 = sum2 + x[i]*x[i];
## [+5 Easter to first x-vec of 100 numbers that causes "naive" to fail!]
## by fail, I mean the "two-pass" approach and built ?sd or ?var function
## shows something entirely different ...
```

5.1.2 Traditional Two Pass

```
v2.norm = doSampleVariance(x.norm, "two-pass");
v2b.norm = doSampleVariance(x.norm);  # default value is "two-pass" in the function
v2c.norm = doSampleVariance(x.norm, "garblideljd=-gook"); # if logic defaults to "two-pass"

unlist(v2.norm);
```

```
##        x.bar        s.var         s.sd
## -0.009744576  0.876188625  0.936049478
```

```
unlist(v2b.norm);
```

```
##        x.bar        s.var         s.sd
## -0.009744576  0.876188625  0.936049478
```

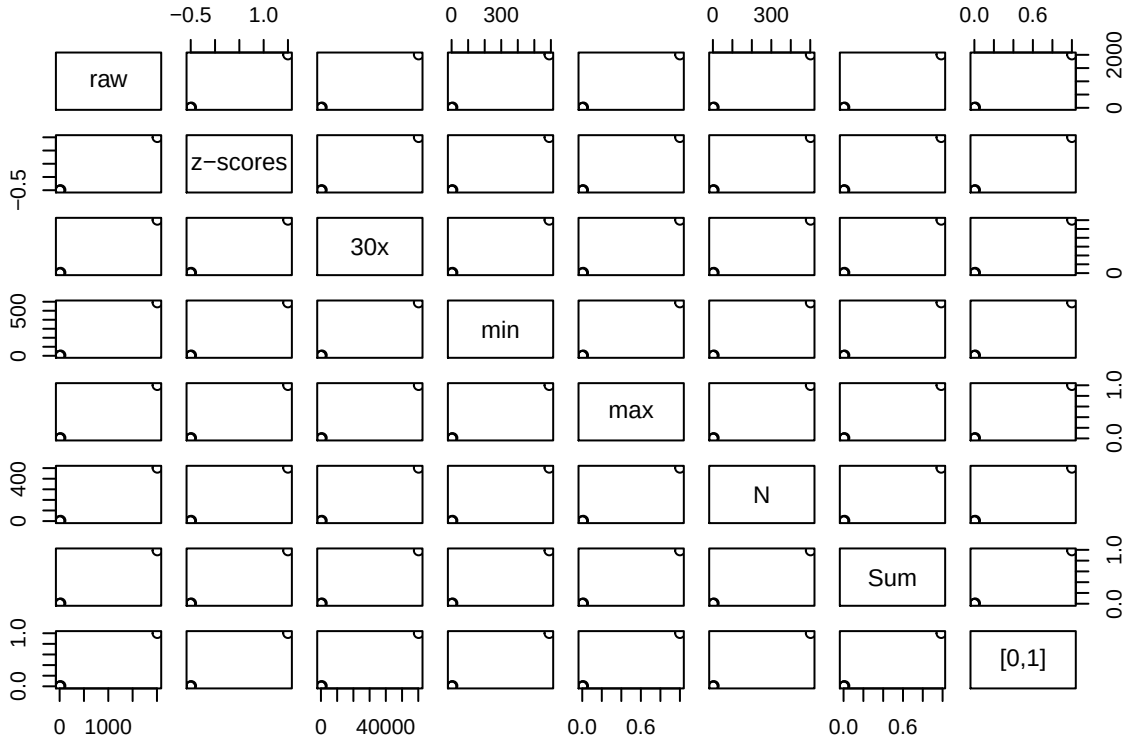
```
unlist(v2c.norm);
```

```
##        x.bar        s.var         s.sd
## -0.009744576  0.876188625  0.936049478
```

5.2 Z-scores

```
# the built in function ?scale you should find useful.
# a z-score is taken on a vector of data requires x.bar and s.hat
# generally, we assume x.bar and s.hat comes from the vector of data.
# It doesn't have to.

library(digest);
md5_monte = digest("monte.shaffer@gmail.com", algo="md5");  # no workee???
md5_monte = "b62c73cdaf59e0a13de495b84030734e";

# jQuery [b62c73cdaf59e0a13de495b84030734e]     https://www.jqueryscript.net/demo/MD5-Hash-String/
# Javascript [b62c73cdaf59e0a13de495b84030734e] http://md5.mshaffer.com/
# PHP     [b62c73cdaf59e0a13de495b84030734e]    https://onlinegdb.com/rJUGCTkrw
# Python enthusiasts:  I recommend WingIDE  ... https://wingware.com/
# [+5 investigate the issue... write a Python function that passes in a string
#   and returns a md5 string, write an onlinegdb.com example for C,
#   and write an onlinegdb.com example for C++ ... summarize your findings.]

row = my_data_clean[my_data_clean$md5_email == md5_monte, ];
```

```
vec.start = getIndexOfDataFrameColumns(row,"V01");  # 5
vec.end = getIndexOfDataFrameColumns(row,"V60"); # 64

vec = as.numeric( row[vec.start:vec.end] );  # vector functions require "vector form"
# recall the concept of a vector basis? (e.g., basis of vector space)
# linear combinations of this basis?


vdf = as.data.frame( t(vec) );  # dim(vdf) tells me to transpose it.
  myRows = c("raw");

z.vec = calculateZscores(vec);
  vdf = rbind(vdf,z.vec);     myRows=c(myRows,"z-scores");

z.vec30 = standardizeToFactor(vec, 30);
  vdf = rbind(vdf,z.vec30);  myRows=c(myRows,"30x");

z.vecmin = standardizeToMin(vec);
# like z-scores, should rewrite to allow it be passed in, by default it computes
  vdf = rbind(vdf,z.vecmin);  myRows=c(myRows,"min");

z.vecmax = standardizeToMax(vec);
# like z-scores, should rewrite to allow it be passed in, by default it computes
  vdf = rbind(vdf,z.vecmax);  myRows=c(myRows,"max");

z.vecN = standardizeToN(vec);
# like z-scores, should rewrite to allow it be passed in, by default it computes
  vdf = rbind(vdf,z.vecN);  myRows=c(myRows,"N");

z.vecSum = standardizeToSum (vec);
# like z-scores, should rewrite to allow it be passed in, by default it computes
  vdf = rbind(vdf,z.vecSum);  myRows=c(myRows,"Sum");

z.vecBound = standardizeFromOneRangeToAnother(vec, c(0,1) );
# like z-scores, should rewrite to allow it be passed in, by default it computes
  vdf = rbind(vdf,z.vecBound);  myRows=c(myRows,"[0,1]");

rownames(vdf) = myRows;

tvdf = as.data.frame( t(vdf) ); # why transpose it?

graphics::plot( tvdf );
```




```
# linear transformations are "linear" ... should not be surprising
# how does perfect linearity relate to "correlation"?

# Multiplying by a negative number (not shown) is also a vector-basis manipulation.
# As is rotating by an angle.
# What is an example of a nonlinear combination?
# Hint look at your will/denzel problem.
# plot(will$movies.50[,c(1,6,7:10)]); ... one relationship is strong, nonlinear
```

6 Will vs. Denzel

Compare Will Smith and Denzel Washington. [See 03_n greater 1-v2.txt for the necessary functions and will-vs-denzel.txt for some sample code and in DROPBOX: __student_access__\unit_01_exploratory_data_analysis\week_02\
You will have to create a new variable $millions.2000 that converts each movie's $millions based on the $year of the movie, so all dollars are in the same time frame. You will need inflation data from about 1980-2020 to make this work. The dataset of Will Smith and Denzel Washington and the infrmation about movies they have participated in comes from the IMDB website, which can get updated and therefore is used very up to date.\citep{IMDB:2020}.

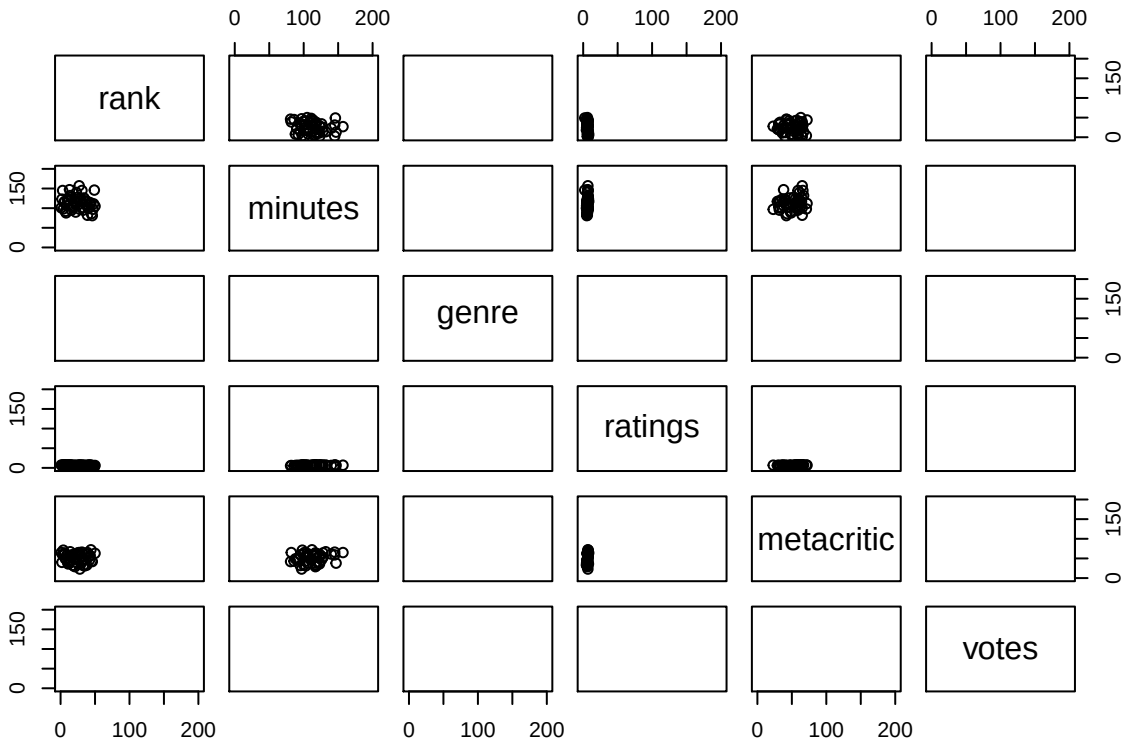


Figure 2: Will Smith scatterplot: IMDB(2020)

```
source_url(paste0(github.path,"master/functions/functions-imdb.R"));
```

6.1 Will Smith

```
nmid = "nm0000226";
    will = grabFilmsForPerson(nmid);
    plot(will$movies.50[,c(1,6,7:10)], ylim=c(0,200), xlim=c(0,200));
```

```
    boxplot(will$movies.50$millions);
```

```
        widx =  which.max(will$movies.50$millions);
    will$movies.50[widx,];
```

```
##    rank   title      ttid year rated minutes                       genre ratings
## 15   15 Aladdin tt6139732 2019     6     128 Adventure, Family, Fantasy       7
##    metacritic  votes millions
## 15         53 217004   355.56
```

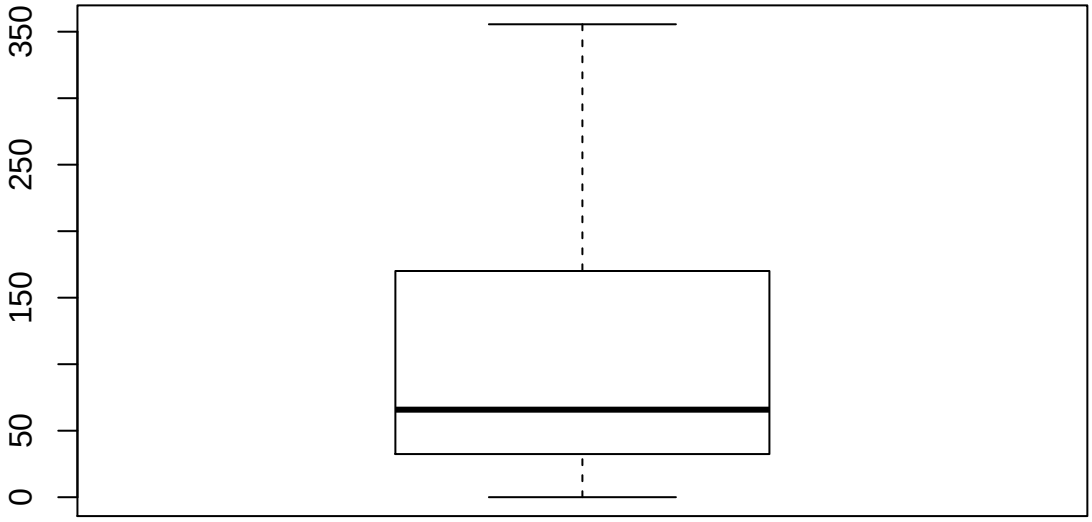


Figure 3: Will Smith boxplot raw millions: IMDB(2020)

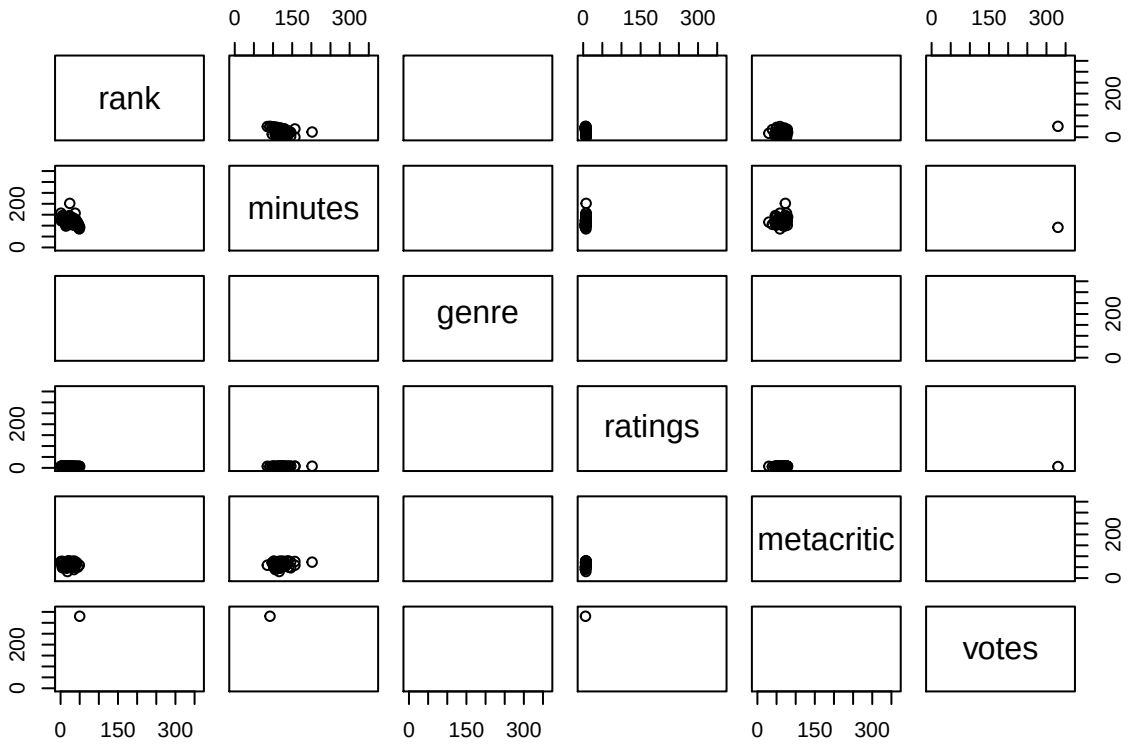


Figure 4: Denzel Washington scatterplot: IMDB(2020)

```
        summary(will$movies.50$year);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1993    2001    2006    2007    2014    2020
```

6.2 Denzel Washington

```
nmid = "nm0000243";
    denzel = grabFilmsForPerson(nmid);
    plot(denzel$movies.50[,c(1,6,7:10)], ylim=c(0,360), xlim=c(0,360));
```

```
    boxplot(denzel$movies.50$millions);
```

```
        didx =  which.max(denzel$movies.50$millions);
    denzel$movies.50[didx,];
```

```
##   rank             title      ttid year rated minutes                   genre
## 1    1 American Gangster tt0765429 2007    16     157 Biography, Crime, Drama
##   ratings metacritic  votes millions
## 1     7.8         76 384322   130.16
```

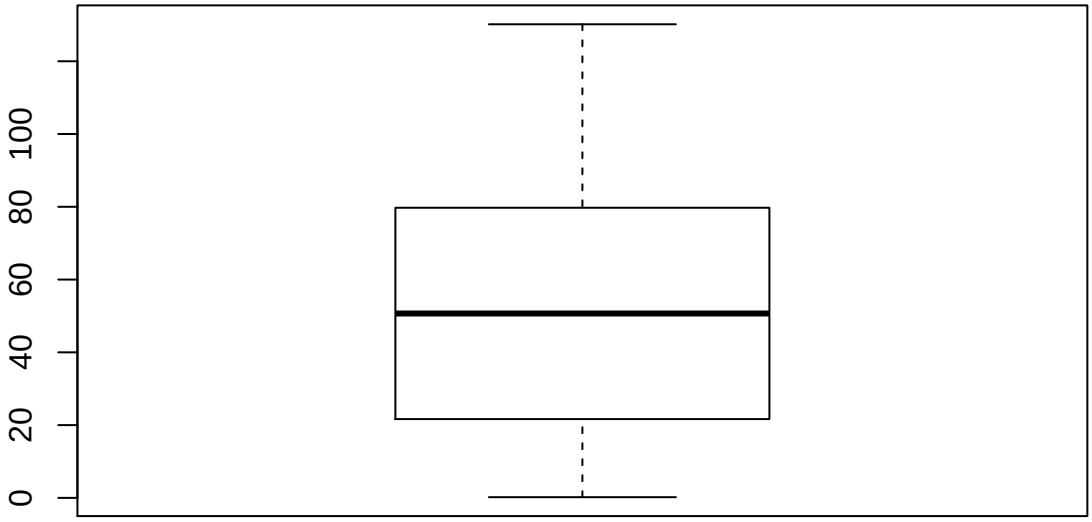


Figure 5: Denzel Washington boxplot raw millions: IMDB(2020)

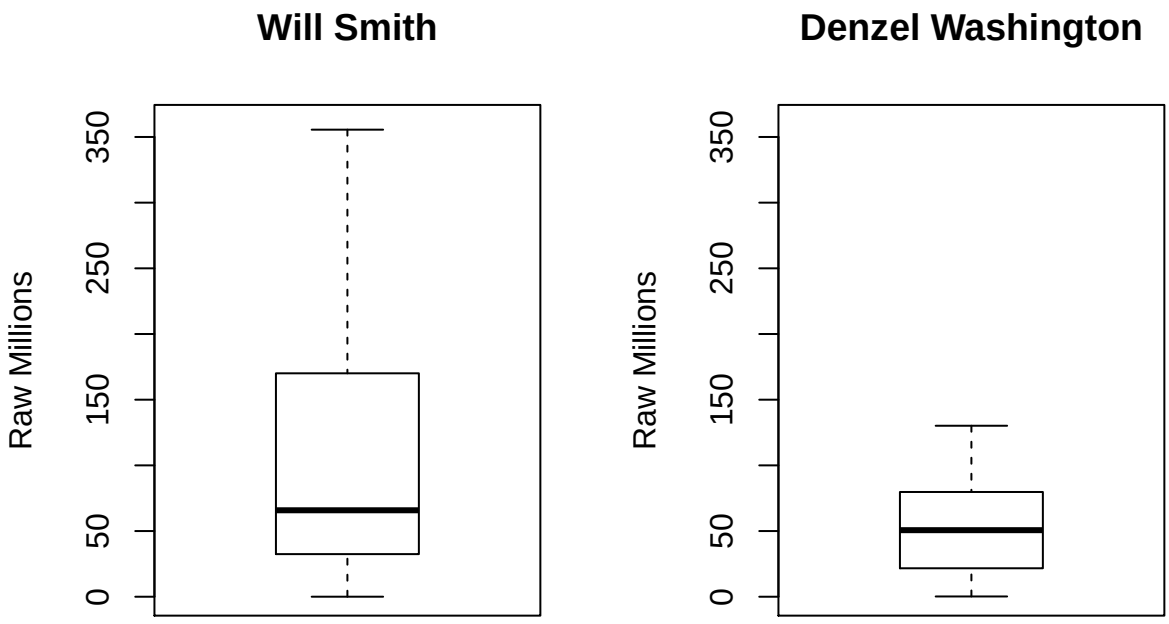


Figure 6: Will Smith vs Denzel Washington boxplot of Top-50 movies using Raw Millions: IMDB(2020)

```
        summary(denzel$movies.50$year);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1981    1993    1999    2000    2008    2018
```

6.3 Boxplot of Top-50 movies using Raw Dollars

```
par(mfrow=c(1,2));
    boxplot(will$movies.50$millions, main=will$name, ylim=c(0,360), ylab="Raw Millions" );
    boxplot(denzel$movies.50$millions, main=denzel$name, ylim=c(0,360), ylab="Raw Millions" );
```

```
    par(mfrow=c(1,1));
```

6.4 Film Count Of Will Smith

```
new.will = will$movies.50;
new.will$nmid = will$nmid;
new.will$name = will$name;
new.will$countfilms = will$countfilms$totalcount;
new.will = new.will[, c(12,13,14, 1:11)];
```

6.5 Film Count Of Denzel Washington

```
new.denzel = denzel$movies.50;
new.denzel$nmid = denzel$nmid;
new.denzel$name = denzel$name;
new.denzel$countfilms = denzel$countfilms$totalcount;
new.denzel = new.denzel[, c(12,13,14, 1:11)];
```

6.6 Combined Dataframe of Will Smith and Denzel Washington

```
df.will.denzel = rbind(new.will, new.denzel);
```

7 Side by side comparison

Build side-by-side box plots on several of the variables (including #6) to compare the two movie stars. After each box plot, write 2+ sentence describing what you are seeing, and what conclusions you can logically make. You will need to review what the box plot is showing with the box portion, the divider in the box, and the whiskers.

7.1 Adjusted Dollars (2000)

7.1.1 Will Smith Standardize Dollars

```
# https://raw.githubusercontent.com/MonteShaffer/humanVerseWSU
#master/humanVerseWSU/R/functions-inflation.R

humanVerseWSU::loadInflationData();
str(will$movies.50);
```

```
## 'data.frame':    50 obs. of  11 variables:
##  $ rank      : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ title     : chr  "I Am Legend" "Suicide Squad" "Independence Day" "Men in Black" ...
##  $ ttid      : chr  "tt0480249" "tt1386697" "tt0116629" "tt0119654" ...
##  $ year      : num  2007 2016 1996 1997 2004 ...
##  $ rated     : chr  "16" "16" "12" "12" ...
##  $ minutes   : num  101 123 145 98 115 117 92 88 106 118 ...
##  $ genre     : chr  "Action, Adventure, Drama" "Action, Adventure, Fantasy" "Action, Adventure, Sci-
##  $ ratings   : num  7.2 6 7 7.3 7.1 8 6.4 6.2 6.8 6.6 ...
##  $ metacritic: num  65 40 59 71 59 64 49 49 58 58 ...
##  $ votes     : num  675802 588726 520979 508031 491899 ...
##  $ millions  : num  256 325 306 251 145 ...
```

```
will$movies.50 = standardizeDollarsInDataFrame(will$movies.50, 2000, "millions", "year", "millionsAdj")
str(will$movies.50);
```

```
## 'data.frame':    50 obs. of  12 variables:
##  $ rank       : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ title      : chr  "I Am Legend" "Suicide Squad" "Independence Day" "Men in Black" ...
##  $ ttid       : chr  "tt0480249" "tt1386697" "tt0116629" "tt0119654" ...
##  $ year       : num  2007 2016 1996 1997 2004 ...
##  $ rated      : chr  "16" "16" "12" "12" ...
```

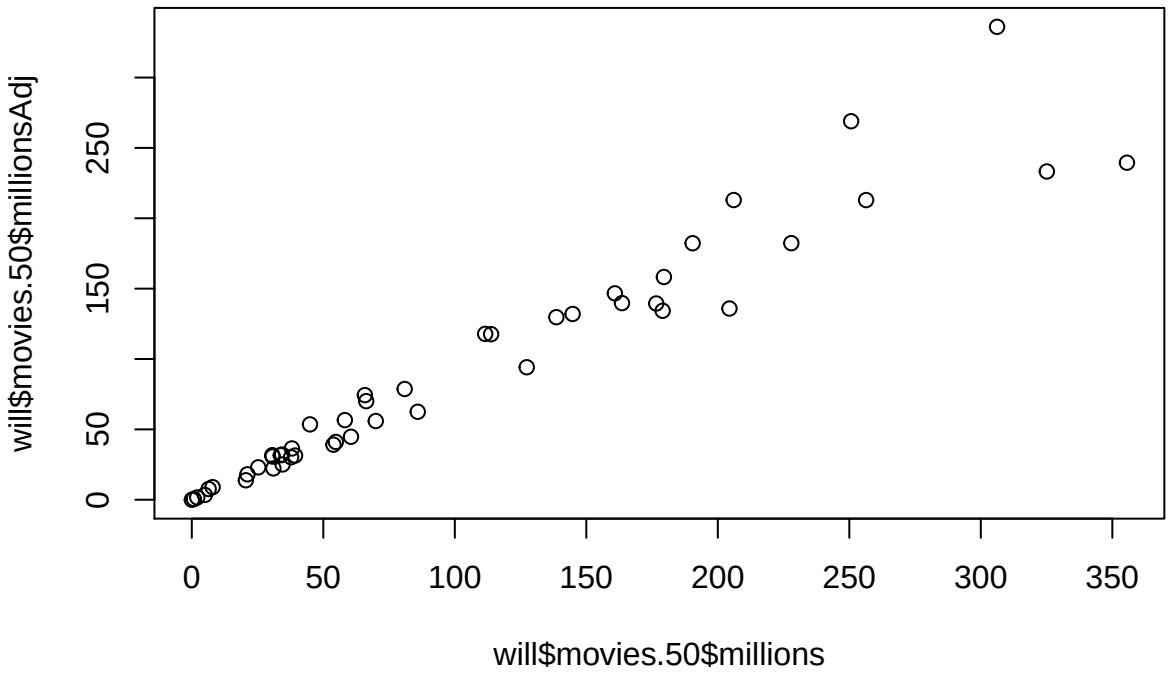


Figure 7: Will Smith scatterplot of Adjusted Millions: IMDB(2020)

```
##  $ minutes    : num  101 123 145 98 115 117 92 88 106 118 ...
##  $ genre      : chr  "Action, Adventure, Drama" "Action, Adventure, Fantasy" "Action, Adventure, Sci
##  $ ratings    : num  7.2 6 7 7.3 7.1 8 6.4 6.2 6.8 6.6 ...
##  $ metacritic : num  65 40 59 71 59 64 49 49 58 58 ...
##  $ votes      : num  675802 588726 520979 508031 491899 ...
##  $ millions   : num  256 325 306 251 145 ...
##  $ millionsAdj: num  213 233 336 269 132 ...
```

```
plot(will$movies.50$millions,will$movies.50$millionsAdj);
```

7.1.2 Denzel Washington Standardize Dollars

```
# https://raw.githubusercontent.com/MonteShaffer/humanVerseWSU
#master/humanVerseWSU/R/functions-inflation.R

humanVerseWSU::loadInflationData();
str(denzel$movies.50);
```

```
## 'data.frame':    50 obs. of  11 variables:
##  $ rank      : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ title     : chr  "American Gangster" "Training Day" "Inside Man" "The Equalizer" ...
```

```
##  $ ttid      : chr  "tt0765429" "tt0139654" "tt0454848" "tt0455944" ...
##  $ year      : num  2007 2001 2006 2014 2004 ...
##  $ rated     : chr  "16" "16" "12" "16" ...
##  $ minutes   : num  157 122 129 132 146 138 126 118 125 115 ...
##  $ genre     : chr  "Biography, Crime, Drama" "Crime, Drama, Thriller" "Crime, Drama, Mystery" "Acti
##  $ ratings   : num  7.8 7.7 7.6 7.2 7.7 7.3 7 6.9 7.7 6.7 ...
##  $ metacritic: num  76 69 76 57 47 76 59 53 66 52 ...
##  $ votes     : num  384322 382457 332327 326531 324445 ...
##  $ millions  : num  130.2 76.6 88.5 101.5 77.9 ...
```

```
denzel$movies.50 = standardizeDollarsInDataFrame(denzel$movies.50, 2000, "millions", "year", "millionsA
str(denzel$movies.50);
```

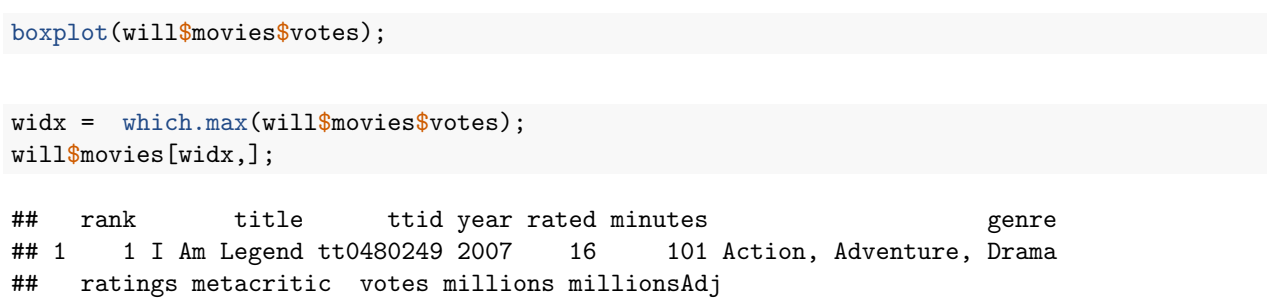
```
## 'data.frame':    50 obs. of  12 variables:
##  $ rank       : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ title      : chr  "American Gangster" "Training Day" "Inside Man" "The Equalizer" ...
##  $ ttid       : chr  "tt0765429" "tt0139654" "tt0454848" "tt0455944" ...
##  $ year       : num  2007 2001 2006 2014 2004 ...
##  $ rated      : chr  "16" "16" "12" "16" ...
##  $ minutes    : num  157 122 129 132 146 138 126 118 125 115 ...
##  $ genre      : chr  "Biography, Crime, Drama" "Crime, Drama, Thriller" "Crime, Drama, Mystery" "Acti
##  $ ratings    : num  7.8 7.7 7.6 7.2 7.7 7.3 7 6.9 7.7 6.7 ...
##  $ metacritic : num  76 69 76 57 47 76 59 53 66 52 ...
##  $ votes      : num  384322 382457 332327 326531 324445 ...
##  $ millions   : num  130.2 76.6 88.5 101.5 77.9 ...
##  $ millionsAdj: num  108.1 74.5 75.6 73.9 71 ...
```

```
plot(denzel$movies.50$millions,denzel$movies.50$millionsAdj);
```

7.2 Total Votes (Divide by 1,000,000)

7.2.1 Votes for Will Smith

```
boxplot(will$movies$votes);
```



```
widx =  which.max(will$movies$votes);
will$movies[widx,];
```

```
##   rank       title      ttid year rated minutes                    genre
## 1    1 I Am Legend tt0480249 2007    16     101 Action, Adventure, Drama
##   ratings metacritic  votes millions millionsAdj
## 1     7.2         65 675802   256.39    212.9349
```

```
total.votes.w = will$movies$votes
will.votes = 0;
for (vote in total.votes.w){
  will.votes = will.votes + total.votes.w[i];
}
will.votes;
```

```
## [1] 8303450
```

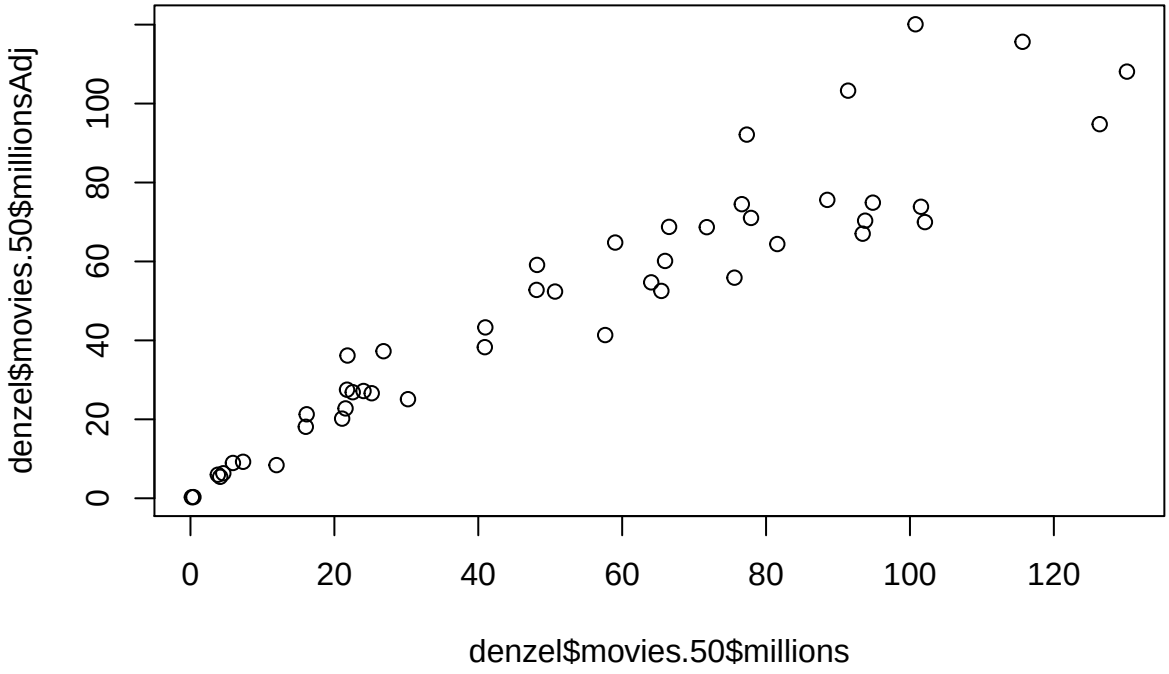


Figure 8: Denzel Washington scatterplot of Adjusted Millions: IMDB(2020)

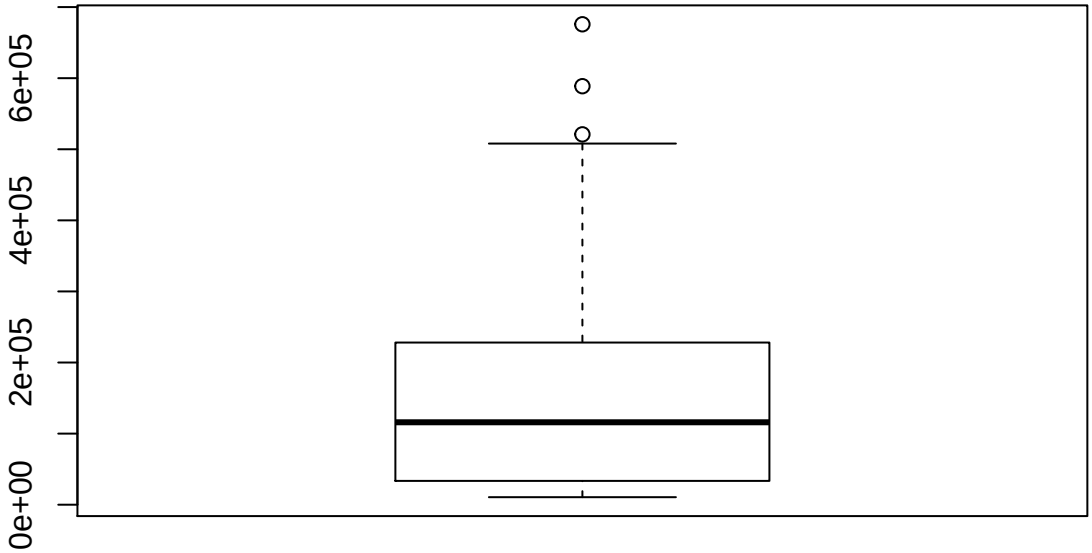


Figure 9: Will Smith boxplot of Votes: IMDB(2020)

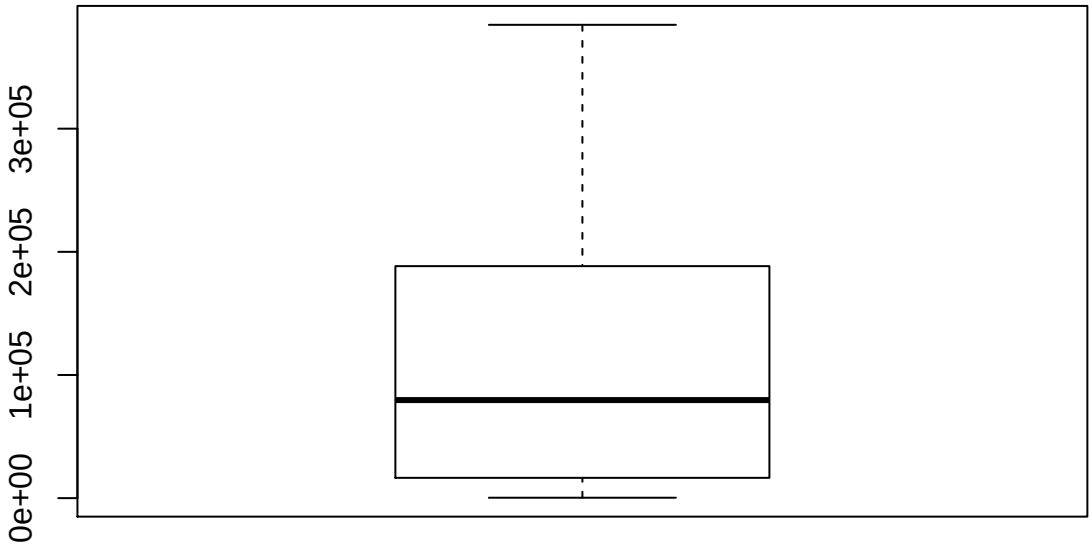


Figure 10: Denzel Washington boxplot of Votes: IMDB(2020)

```
will.votel.p = will.votes/1000000;
```

7.2.2 Votes for Denzel Washington

```
boxplot(denzel$movies$votes);
```

```
widx =  which.max(denzel$movies$votes);
denzel$movies[widx,];
```

```
##   rank             title      ttid year rated minutes                    genre
## 1    1 American Gangster tt0765429 2007    16     157 Biography, Crime, Drama
##   ratings metacritic  votes millions millionsAdj
## 1     7.8         76 384322   130.16    108.0994
```

```
total.votes.d = denzel$movies$votes
denzel.votes = 0;
for (vote in total.votes.d){
  denzel.votes = denzel.votes + total.votes.d[i];
}
denzel.votes;
```

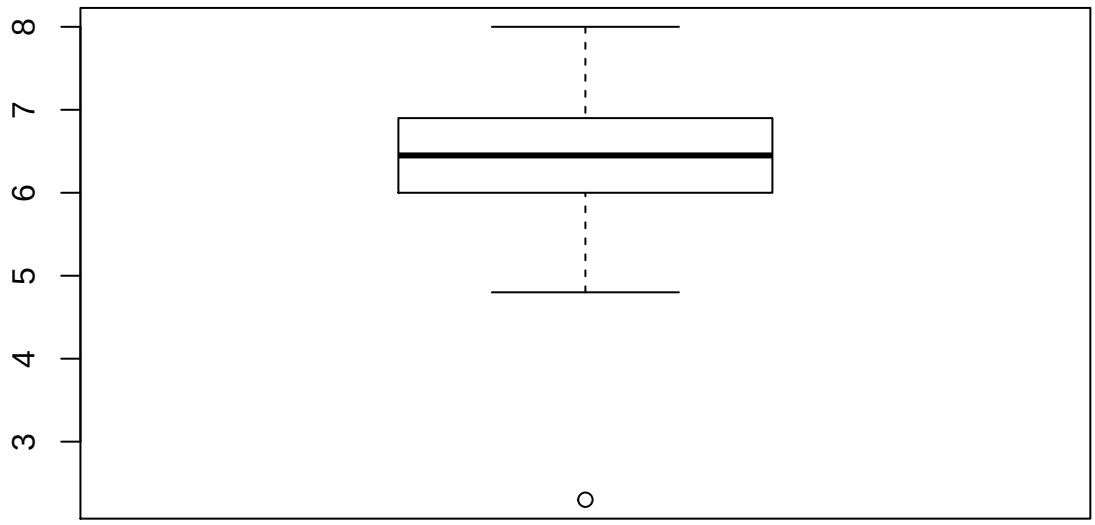


Figure 11: Will Smith boxplot of Ratings: IMDB(2020)

```
## [1] 5118550
```

```
denzel.votes.p = denzel.votes/1000000;
```

7.3 Average Ratings

7.3.1 Average Ratings Will Smith

```
boxplot(will$movies$ratings);
```

```
widx =  which.max(will$movies$ratings);
will$movies[widx,];
```

```
##   rank                  title      ttid year rated minutes            genre
## 6    6 Das Streben nach Glück tt0454921 2006     0     117 Biography, Drama
##   ratings metacritic  votes millions millionsAdj
## 6       8         64 438612   163.57     139.716
```

```
total.ratings.w = will$movies$ratings
will.ratings = 0;
for (rating in total.ratings.w){
```

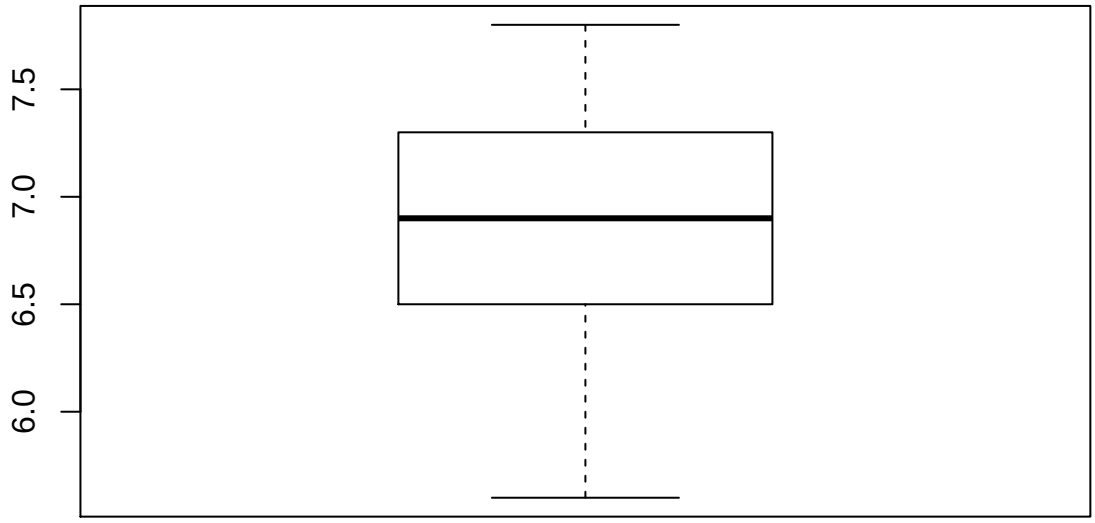


Figure 12: Denzel Washington boxplot of Ratings: IMDB(2020)

```
  will.ratings = will.ratings + total.ratings.w[i];
}
will.ratings;
```

```
## [1] 315
```

```
summary(will$movies.$ratings);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.300   6.000   6.450   6.328   6.875   8.000
```

7.3.2 Average Ratings Denzel Washington

```
boxplot(denzel$movies$ratings);
```

```
widx =  which.max(denzel$movies$ratings);
denzel$movies[widx,];
```

```
##   rank             title      ttid year rated minutes                    genre
## 1    1 American Gangster tt0765429 2007    16     157 Biography, Crime, Drama
##   ratings metacritic  votes millions millionsAdj
## 1     7.8         76 384322   130.16    108.0994
```

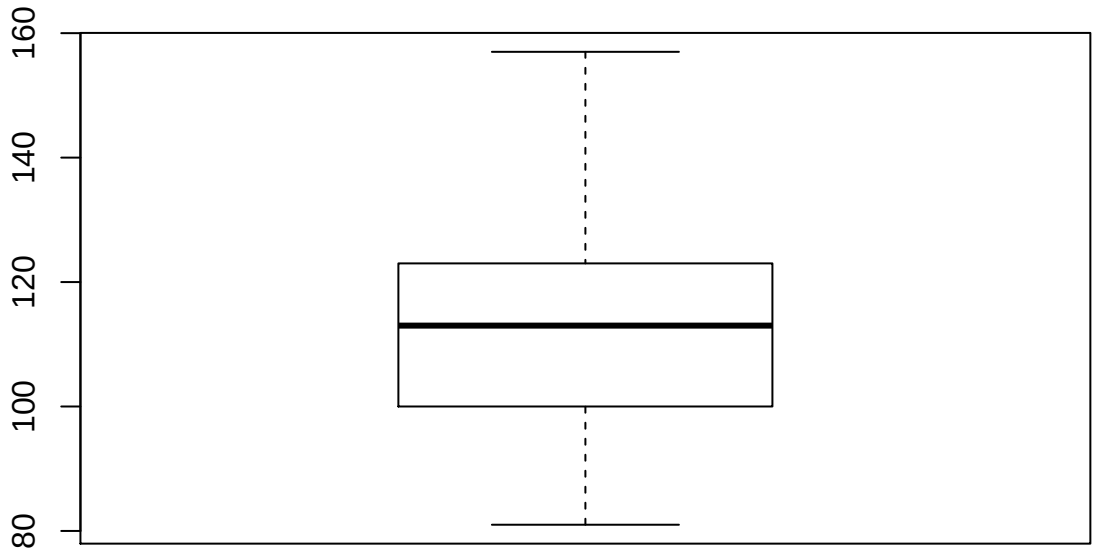


Figure 13: Will Smith boxplot of Minutes: IMDB(2020)

```
total.ratings.d = denzel$movies$ratings
denzel.ratings = 0;
for (rating in total.ratings.d){
  denzel.ratings = denzel.ratings + total.ratings.d[i];
}
denzel.ratings;
```

```
## [1] 330
```

```
summary(denzel$movies.$ratings);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.600   6.525   6.900   6.852   7.300   7.800
```

7.4 Who has the longest total and average Movie length in Minutes?

7.4.1 Lenght of Movies Will Smith

```
boxplot(will$movies$minutes);
```

```
widx =  which.max(will$movies$minutes);
will$movies[widx,];
```

```
##    rank title      ttid year rated minutes                   genre ratings
## 27   27   Ali tt0248667 2001    12     157 Biography, Drama, Sport     6.8
##    metacritic votes millions millionsAdj
## 27         65 92568     58.2    56.58972
```

```
total.minutes.w = will$movies$minutes
will.minutes = 0;
for (minute in total.minutes.w){
  will.minutes = will.minutes + total.minutes.w[i];
}
will.minutes;
```

```
## [1] 5950
```

```
summary(will$movies.$minutes);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    81.0   100.2   113.0   113.0   123.0   157.0
```

7.4.2 Lenght of Movies Denzel Washington

```
boxplot(denzel$movies$minutes);
```



```
widx =  which.max(denzel$movies$minutes);
denzel$movies[widx,];
```

```
##    rank     title      ttid year rated minutes                     genre
## 24   24 Malcolm X tt0104797 1992    12     202 Biography, Drama, History
##    ratings metacritic votes millions millionsAdj
## 24     7.7         73 83724    48.17    59.12241
```

```
total.minutes.d = denzel$movies$minutes
denzel.minutes = 0;
for (minute in total.minutes.d){
  denzel.minutes = denzel.minutes + total.minutes.d[i];
}
denzel.minutes;
```

```
## [1] 6450
```

```
summary(denzel$movies.$minutes);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    85.0   106.0   118.0   120.2   129.0   202.0
```

7.5 Metacritic (NA values)

7.5.1 Metacritic of Will Smith Movies

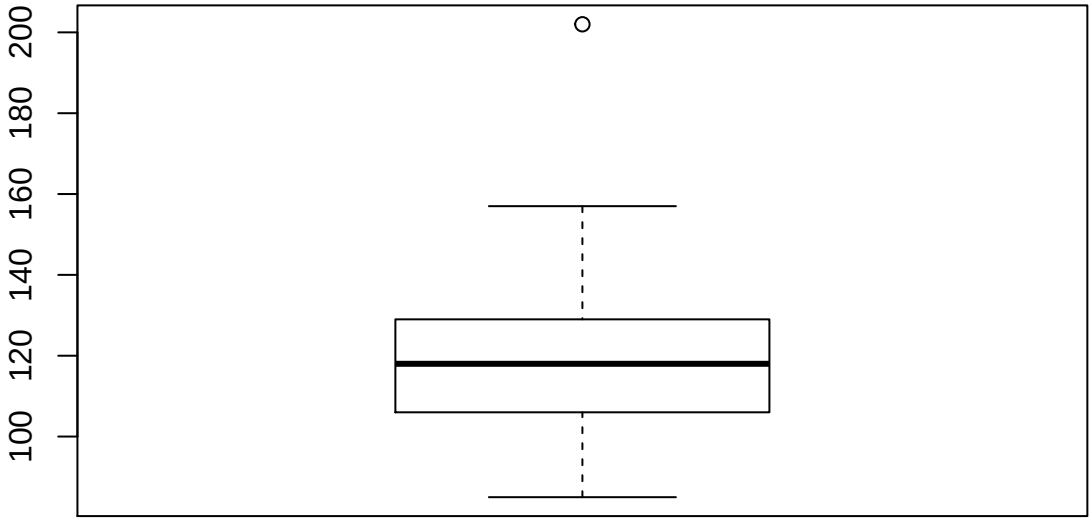


Figure 14: Denzel Washington boxplot of Minutes: IMDB(2020)

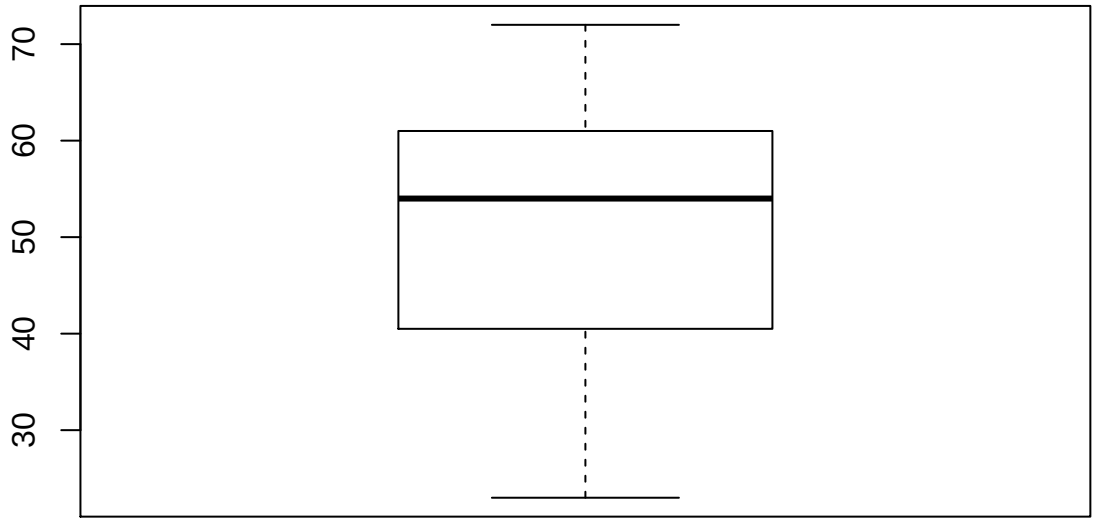


Figure 15: Will Smith boxplot of Metacritic: IMDB(2020)

```
boxplot(will$movies$metacritic);
```

```
widx =  which.max(will$movies$metacritic);
will$movies[widx,];
```

```
##    rank                       title      ttid year rated minutes
## 44   44 Das Leben - Ein Sechserpack tt0108149 1993     6     112
##                     genre ratings metacritic votes millions millionsAdj
## 44 Comedy, Drama, Mystery     6.8         72 19443     6.41    7.638768
```

```
total.metacritic.w = will$movies$metacritic
will.metacritic = 0;
for (meta in total.metacritic.w){
  will.metacritic = will.metacritic + total.metacritic.w[i];
}
will.metacritic;
```

```
## [1] 3050
```

```
summary(will$movies.$metacritic);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   23.00   40.50   54.00   50.87   61.00   72.00       3
```

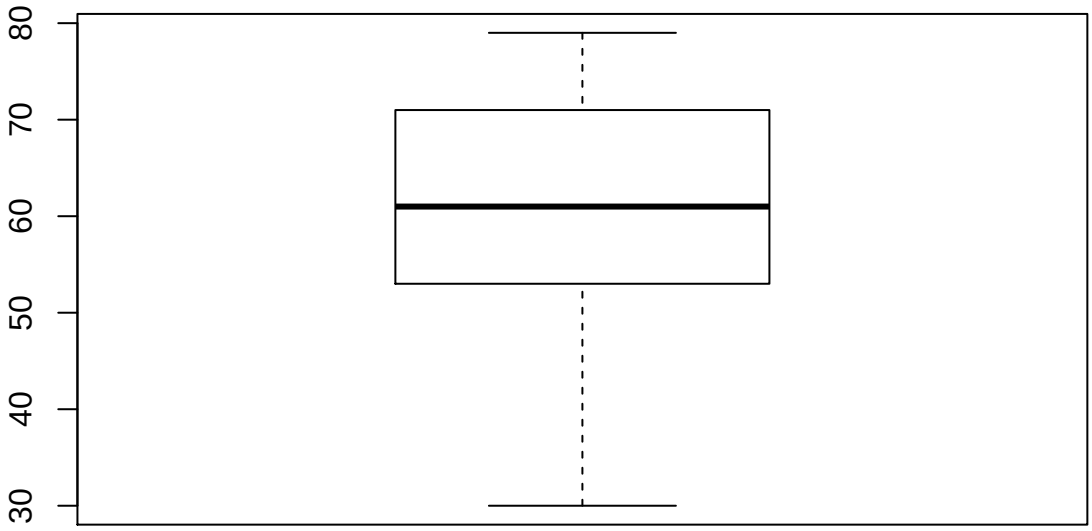


Figure 16: Denzel Washington boxplot of Metacritic: IMDB(2020)

7.5.2 Metacritic of Denzel Washington Movies

```
boxplot(denzel$movies$metacritic);
```

```
widx =  which.max(denzel$movies$metacritic);
denzel$movies[widx,];
```

```
##    rank  title      ttid year rated minutes genre ratings metacritic votes
## 22   22 Fences tt2671706 2016     6     139 Drama     7.2         79 95617
##    millions millionsAdj
## 22    57.64    41.35549
```

```
total.metacritic.d = denzel$movies$metacritic
denzel.metacritic = 0;
for (meta in total.metacritic.d){
  denzel.metacritic = denzel.metacritic + total.metacritic.d[i];
}
denzel.metacritic;
```

```
## [1] 3800
```

```
summary(denzel$movies.$metacritic);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   30.00   53.00   61.00   61.15   71.00   79.00       9
```