Datasets HW

Jessica Smith
11602418
STATS 419
Monte J. Shaffer
9/5/2020

1. Create the "rotate matrix" functions as described in lectures. Apply to the example "myMatrix".

The original matrix:

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

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

Transposing the matrix:

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

transposeMatrix(myMatrix)
```

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

Rotation by 90 degrees:

```
rotateMatrix90 = function(mat)
{
  t(apply(mat, 2, rev))
}

rotateMatrix90(myMatrix)
```

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

Rotation by 180 degrees:

```
rotateMatrix180 = function(mat)
{
    t(apply(t(apply(mat, 2, rev)), 2, rev))
}

rotateMatrix180(myMatrix)
```

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

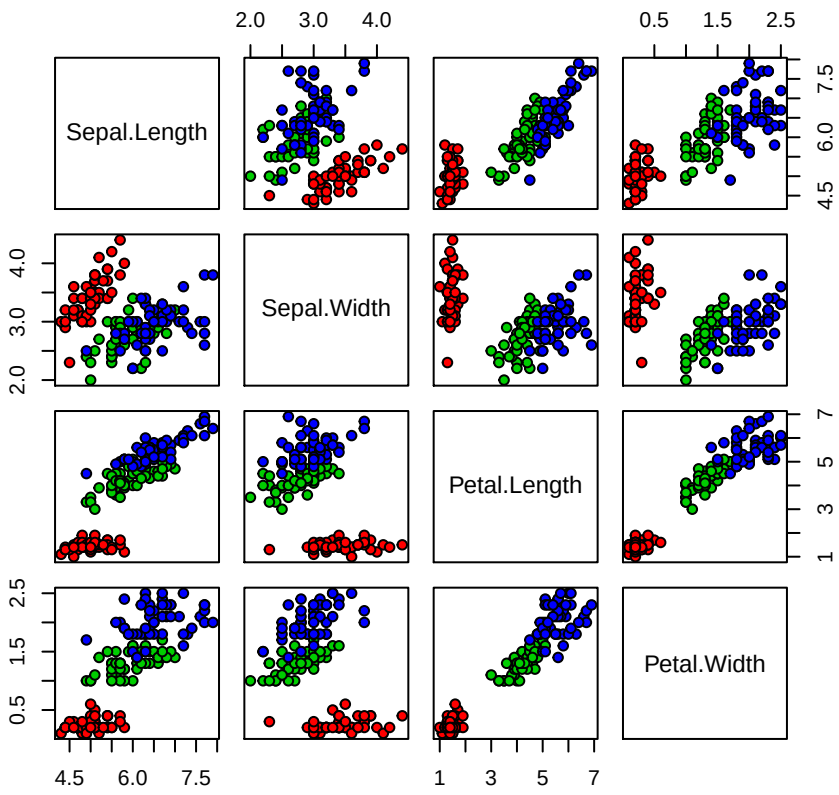
Rotation by 120 degrees:

```
rotateMatrix270= function(mat)
{
    apply(t(mat), 2, rev)
}

rotateMatrix270(myMatrix)
```

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

2. 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

```
data(iris)
pairs(iris[1:4], main = "Iris Data (red=setosa, green=versicolor,blue-virginica)",
      pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])
```



3. 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 well known iris data set contains 50 measurements, each from 3 species of iris flower. The metrics include petal length, petal width, sepal length, and sepal width. The dataset is commonly used to teach clustering analysis and to demonstrate basic programmatic functionality. (Kaggle, 2020)

4. Import "personality-raw.txt" into R. Remove the V00 column. Create two new columns from the current column "date_test": year and week. 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.

```
#read in the data
personality = read.table("personality/personality-raw.txt", header = TRUE, sep = "|", dec = ".")

#remove unwanted column
personality = subset(personality, select = -c(V00))
```

```
#create two columns from date_test, one year and one week
temp = strsplit(as.character(personality$date_test), " ")
personality$date = matrix(unlist(temp), ncol=2, byrow=TRUE)[,1]
personality$year = format(as.Date(personality$date, "%m/%d/%Y"), format="%Y")
personality$week = format(as.Date(personality$date, "%m/%d/%Y"), format= "%W")
personality = subset(personality, select = -c(date, date_test))
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

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

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

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

```
personality = personality  %>% select(md5_email, year, week, everything())

#Sort the new data frame by YEAR, WEEK so the newest tests are at the top of the df
personality = personality[order(-(as.numeric(personality$year)), -(as.numeric(personality$week))), ]

#remove duplicates using the unique function based on the column "md5_email"
unique = unique(personality$md5_email)
rows = match(unique, personality$md5_email)

unique_personalities = personality[rows,]

#Save the data frame in the "pipe-delimited format" as "personality-clean.txt"
write.table(unique_personalities,"personality-clean.txt",sep="|", row.names=FALSE)

cat("The raw dataset contains ",nrow(personality),
" records, and the cleaned dataset has ", nrow(unique_personalities),".", sep = "")
```

```
## The raw dataset contains 838 records, and the cleaned dataset has 678.
```

5. 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.

```
#get the email ids that appear more than once
candidates = setdiff(personality, unique_personalities)
```

```r
#isolate the most frequent
monteId = tail(names(sort(table(candidates$md5_email))), 1)

#get the Monte row from the clean dataset
monte = unique_personalities[unique_personalities$md5_email == monteId, ]
monte = monte[c(-1, -2, -3)]
```

```r
doSampleVariance = function(sample, method)
{
  n = length(sample)
  sumXs = sum(sample)
  myMean = sumXs / n

    if(method=="naive")
        {
      sumXSquared = 0
    for (x in sample)
      {
        sumXSquared = sumXSquared +  x^2
      }
    naiveVariance  = ((sumXSquared - ((sumXs * sumXs) / n))) / (n - 1)

    if (abs(sumXSquared-((sumXs * sumXs) / n))< 1e-16)
      {
      writeLines("Warning: The precision of this result may be very low. Use of another method
            is recommended.")
    }
     vals = data.frame(sumXs, sumXSquared, naiveVariance)
     colnames(vals) = c("sumXs", "sumXSquared", "variance")
     return(vals)
       }

    else
        {
        sumDiffSquared = 0
    for (x in sample)
      {
        sumDiffSquared = sumDiffSquared +  ((x - myMean) * (x - myMean))
      }
        #two-pass algorithm
    twoPassVariance = sumDiffSquared / (n - 1)

    vals = data.frame(sumXs, sumDiffSquared, twoPassVariance)
    colnames(vals) = c("sumXs", "sumDiffSquared", "variance")
    return(vals)
        }
}

#test
naive = doSampleVariance(monte, method = "naive")
cat("The naive sample variance is ", naive[1, "variance"], ".", sep="")
```

```
## The naive sample variance is 0.4865085.
```

```
twoPass = doSampleVariance(monte, method = "two-pass")
cat(" The two-pass sample variance is ", twoPass[1, "variance"], ".", sep = "")
```

```
##  The two-pass sample variance is 0.4865085.
```

```
doMode = function(x)
{
    #find frequencies
  df = as.data.frame(table(unlist(x)))

  #check if multiple
  if ("TRUE" %in%  duplicated(x))
  {
    #If bi-modal, store all of the ties
    modes <- unique(x[duplicated(x)])
    return(modes)
  }
  else
  {
    #if there is only one mode, return a vector of length one
    mode =  c(as.numeric(as.character(df[which.max(df$Freq),][1,1])))
    return(mode)
  }
}

#test
cat("The mode from the 'monte' sample is: ",doMode(monte), ".\n", sep = "" )
```

```
## The mode from the 'monte' sample is: 4.2.
```

```
testBiModal = c(1,1,2,2,3,4,5)
cat("A test vector to try the bimodal functionality: [",
    toString(paste0("",testBiModal,"") ), "]\n", sep = "")
```

```
## A test vector to try the bimodal functionality: [1, 1, 2, 2, 3, 4, 5]
```

```
cat("The modes from the 'bimodal' test sample are: "
    ,toString(paste0("", doMode(testBiModal),"") ), ".", sep = "" )
```

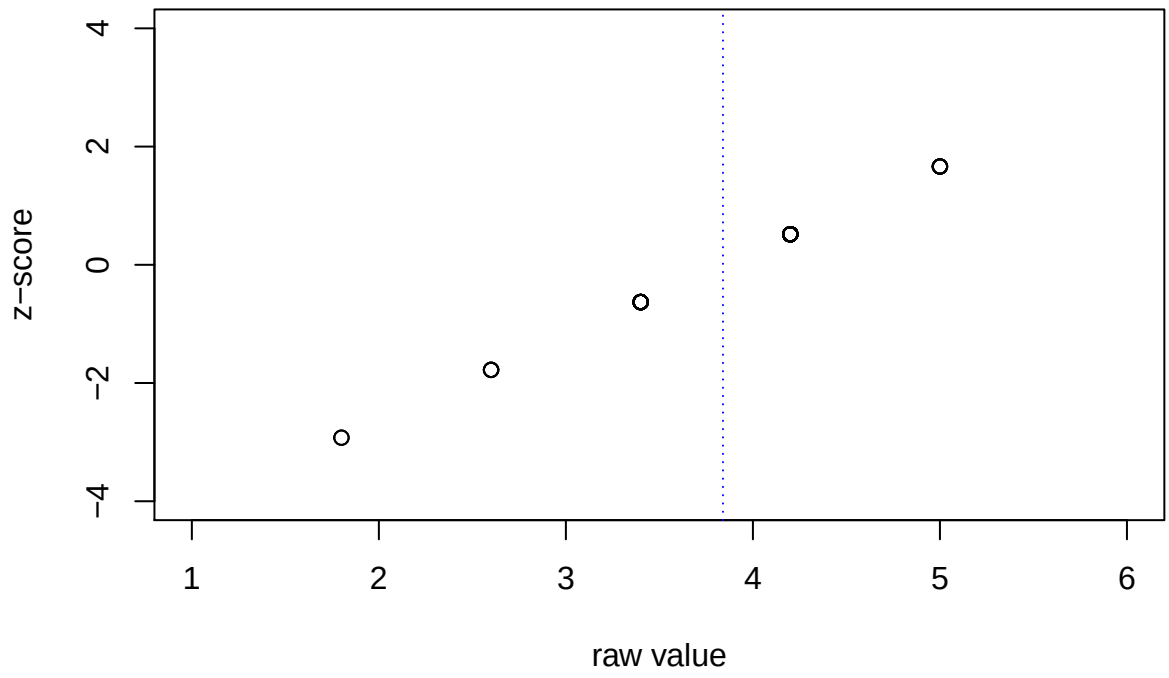
```
## The modes from the 'bimodal' test sample are: 1, 2.
```

```
doSummary = function(x)
    {
  x=as.vector(t(x))
  length_ = length(x)
    numNA = sum(is.na(x))
    mean_ = mean(x)
  median_ = median(x)
  mode_ = doMode(x)
  varianceN = doSampleVariance(x, "naive")$variance
```

```
  variance2 = doSampleVariance(x, "other")$variance
  sd_builtIn = sd(x)
  sd_customN = sqrt(varianceN)
  sd_custom2 = sqrt(variance2)

  results = data.frame("length" = length_, "NAs" = numNA, "mean" = mean_,
  "median" = median_, "mode" = mode_, "variance(naive)" = varianceN,
  "variance(2pass) " = variance2, "sd(built-in) " =
  sd_builtIn, "sd_naive" = sd_customN , "sd_2pass" = sd_custom2)
  results = results[1,]
  return(results)
}

#test
#doSummary(monte)
```

```
#create z-scores
zScores = function(x)
{
  rownames(x) <- c()
  zs = vector()
  i=0
  for (item in x)
  {
    z = (item - rowMeans(x))/sd(x)
    zs = append(zs,z, i)
    i = i + 1
  }
  results = rbind(x,(zs))
  rownames(results) <- c("value", "z-score")
  return(results)
}

#test
#zScores(monte)
```

```
#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.

data = zScores(monte)
plot(unlist(data["value", ]), unlist(data["z-score", ]), ylab = "z-score",
     xlab = "raw value", main = "monte.shaffer@gmail.com", xlim = (c(1,6)),
     ylim = (c(-4, 4)))
abline(v = rowMeans(data["value", ]), lty = 3, col = "blue")
```




```
writeLines("The zscore is a measure of how far from the mean a data point is, to phrase it
informally.
           \n The plot of the monte sample shows that as the raw value gets higher, the z-score
           gets lower. The \n mean of the sample is show by the blue dashed line at 3.84, so
           the z-scores closest to zero should \n be associated with raw values near the mean.")
```

```
## The zscore is a measure of how far from the mean a data point is, to phrase it
## informally.
##
##  The plot of the monte sample shows that as the raw value gets higher, the z-score
##            gets lower. The
##  mean of the sample is show by the blue dashed line at 3.84, so
##            the z-scores closest to zero should
##  be associated with raw values near the mean.
```

6. Compare Will Smith and Denzel Washington.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.

```
## Warning: package 'stringr' was built under R version 3.5.3
```

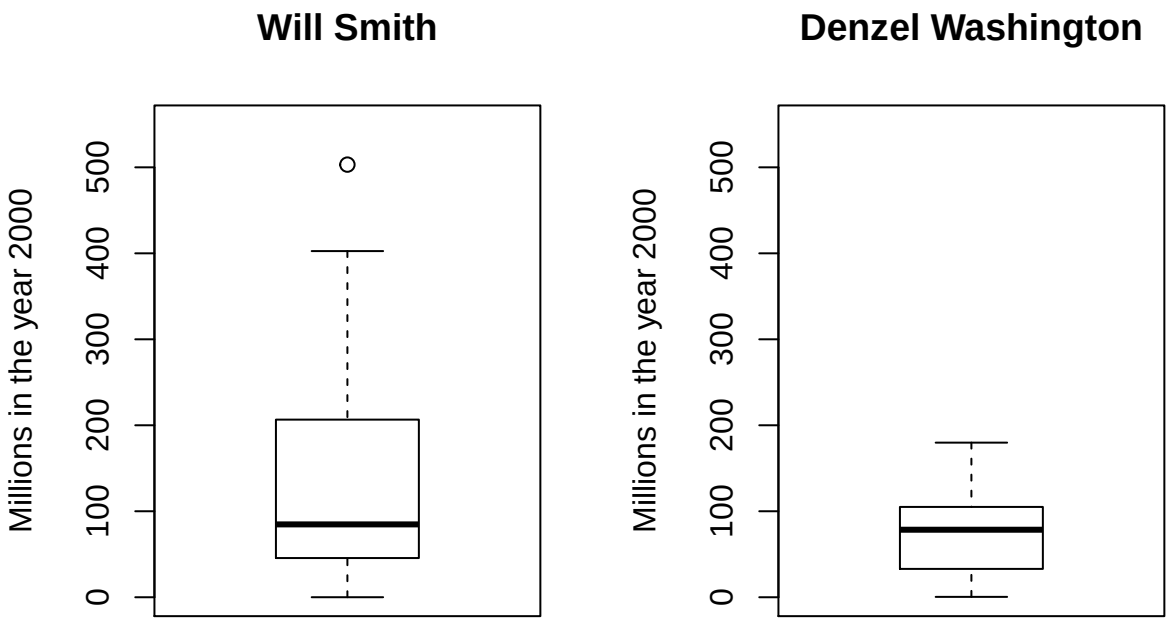
```
## Loading required package: xml2
```

```
## Warning: package 'xml2' was built under R version 3.5.3
```

```
nmid = "nm0000226";
will = grabFilmsForPerson(nmid);

nmid = "nm0000243";
denzel = grabFilmsForPerson(nmid);

#inflation: https://inflationdata.com/inflation/Inflation_Articles/CalculateInflation.asp
#CPI: https://inflationdata.com/Inflation/Consumer_Price_Index/HistoricalCPI.aspx?reloaded=true#Table
cpi = read.table("cpi.csv", header = TRUE, sep = ",")
mean = rowMeans(cpi[ , -1], na.rm= TRUE)
CPI = as.data.frame(cbind(year = cpi$AR ,cpi = mean))
cpi2000 = CPI[1,2]

library(dplyr)
D = denzel$movies.50 %>% inner_join(CPI, by = "year")
W = will$movies.50 %>% inner_join(CPI, by = "year")

D$millions_2000 = (D$millions*cpi2000)/D$cpi
W$millions_2000 = (W$millions*cpi2000)/W$cpi
```

7. 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.

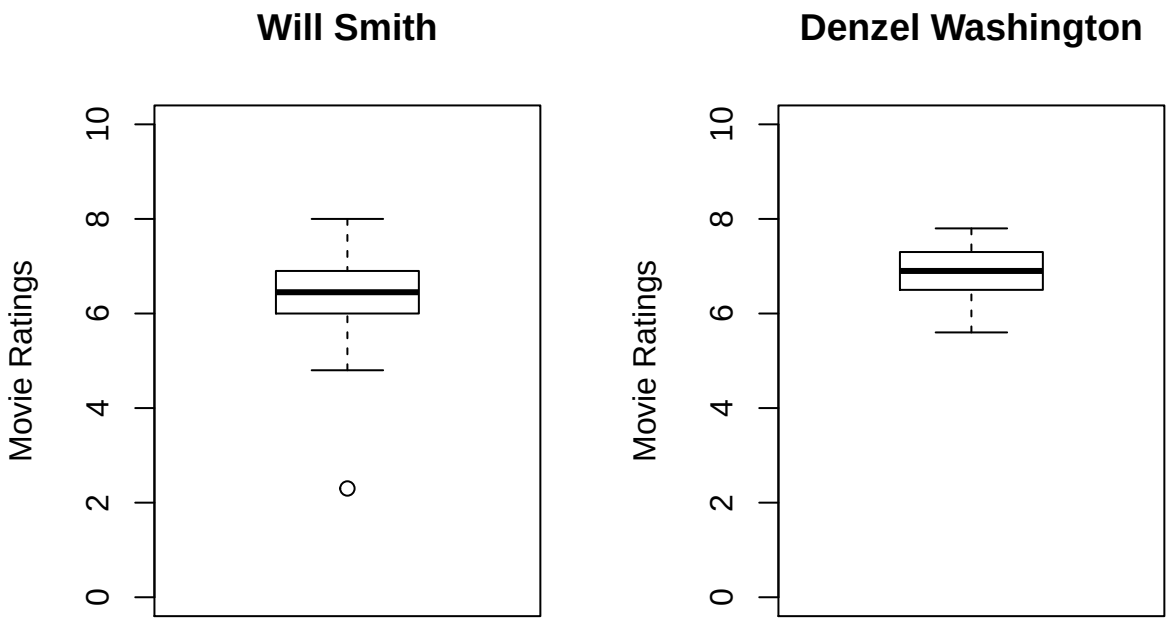
```
par(mfrow=c(1,2));
boxplot(W$millions_2000, main=will$name, ylim=c(0,550), ylab="Millions in the year 2000")
boxplot(D$millions_2000, main=denzel$name, ylim=c(0,550), ylab="Millions in the year 2000")
```




```
writeLines("The boxplots of box office sales show that the median for the two stars is about
the same.\n There is greater variation in the amount that Will Smith movies earn as shown by
the longer \n whiskers and larger interquartile range, while Denzel's movies appear to be
more consistent.")
```

```
## The boxplots of box office sales show that the median for the two stars is about
## the same.
##  There is greater variation in the amount that Will Smith movies earn as shown by
## the longer
##  whiskers and larger interquartile range, while Denzel's movies appear to be
## more consistent.
```

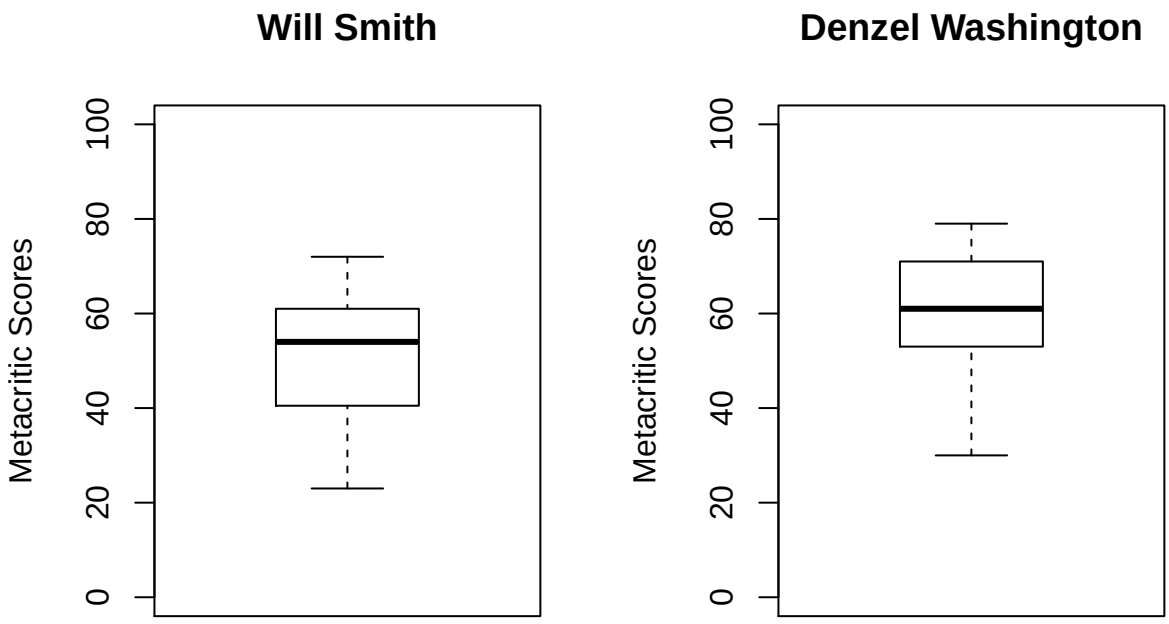
```
par(mfrow=c(1,2));
boxplot(W$ratings, main=will$name, ylim=c(0,10), ylab="Movie Ratings")
boxplot(D$ratings, main=denzel$name, ylim=c(0,10), ylab="Movie Ratings")
```




```
writeLines("The boxplots of movie ratings show that the median for the two stars is again
very close,\n but Denzel's score is a little higher.  There is a larger range of ratings
for Will Smith's movies,\n shown by the longer whiskers. The interquartile ranges have
similar sizes for both actors")
```

```
## The boxplots of movie ratings show that the median for the two stars is again
## very close,
##  but Denzel's score is a little higher.  There is a larger range of ratings
## for Will Smith's movies,
##  shown by the longer whiskers. The interquartile ranges have
## similar sizes for both actors
```

```
par(mfrow=c(1,2));
boxplot(W$metacritic, main=will$name, ylim=c(0,100), ylab="Metacritic Scores")
boxplot(D$metacritic, main=denzel$name, ylim=c(0,100), ylab="Metacritic Scores")
```




```
writeLines("The metacritic scores for the two actors show the mean score for Denzel to
be slightly higher.\n The whisker legth is similar, as is the size of the interquartile
range.")
```

```
## The metacritic scores for the two actors show the mean score for Denzel to
## be slightly higher.
##  The whisker legth is similar, as is the size of the interquartile
## range.
```