BIOS 721 Practice Problems | Topic 5: Plotting in R Set 1 SOLUTION KEY

Introduction

The goal of this assignment is to familiarize you with basic plotting in R.

Concepts covered in this assignment include:

1. Basic plotting functions – hist(), boxplot(), plot(), lines()
2. Low vs. high level plotting commands
3. Customizing plots using options in plotting functions
4. Outputting/storing plots

Resources/Files needed for this assignment:

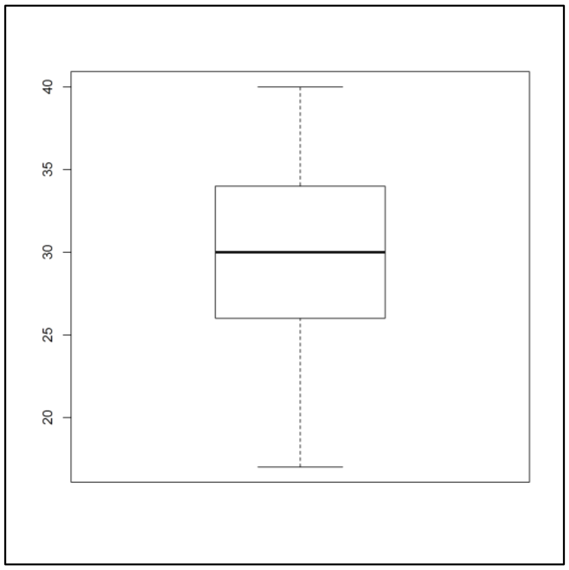
- Data file: MARKS2.DAT

---

1. The R code below sets the working directory (using the setwd() function; where the data file is stored), reads the MARK2.DAT file into R (stored as the object data), examines the contents of the data frame (using the str() function), and creates a boxplot of the data values for the Exam 1 variable using the boxplot() function in R. The resulting figure is shown below. Execute R code below and verify that you can create the same figure. Remember: You will need to change the working directory to the file path where you saved the data file!

```
setwd("M:\\Teaching\\Duke Courses\\BIOS 721\\")
data <- read.table("marks2.dat",header=TRUE)
str(data)
boxplot(data$exam1)
```

Figure:



The function call above uses the default options set in boxplot(). For example, there are no figure titles or axes labels. For most plotting functions in base R, there are multiple optional inputs available that can be used to create customized figures from the function.

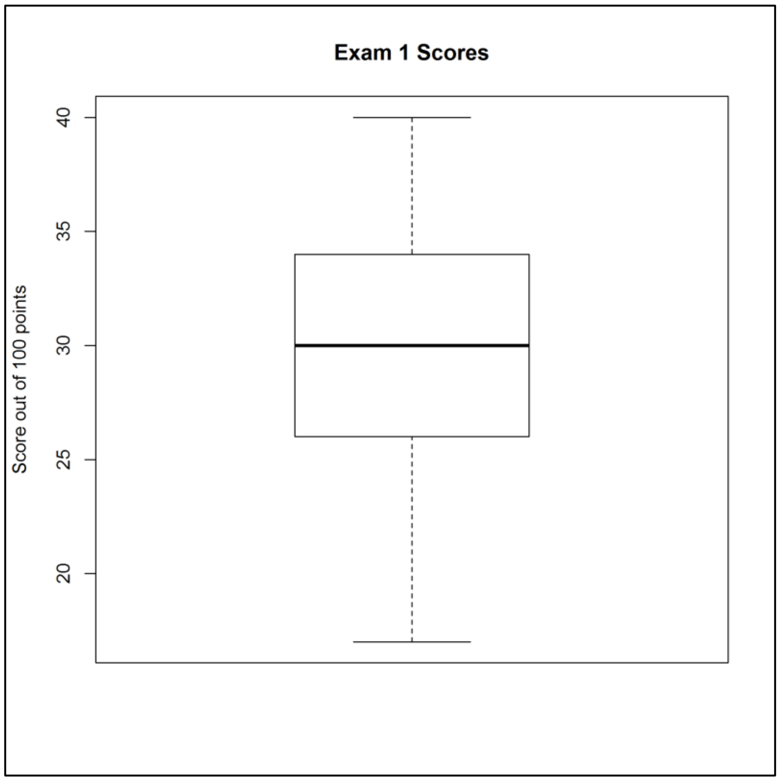
a. Using the R help page, find the optional inputs for adding a user-defined main title and y-axis label using the information listed below in (i) and (ii). Was the R help page helpful? Explain how it was or was not helpful. Were you able to create user-defined titles and labels? If so, provide the R code below.
   i. Title the figure ‘Exam 1 Scores’.
   ii. Label the y-axis as ‘Score out of 100 points’.

   The R help page does not list input variables for the boxplot() function for these features. However, it is always a good idea to try common plotting parameters/inputs (e.g. main, xlab/ylab, xlim/ylim, col, pch, etc.) in any base R plotting function, they usually work even if they are not listed on the help page.

   R Code:

```
boxplot(data$exam1,
        main='Exam 1 Scores',
        ylab='Score out of 100 points')
```

   R Output:

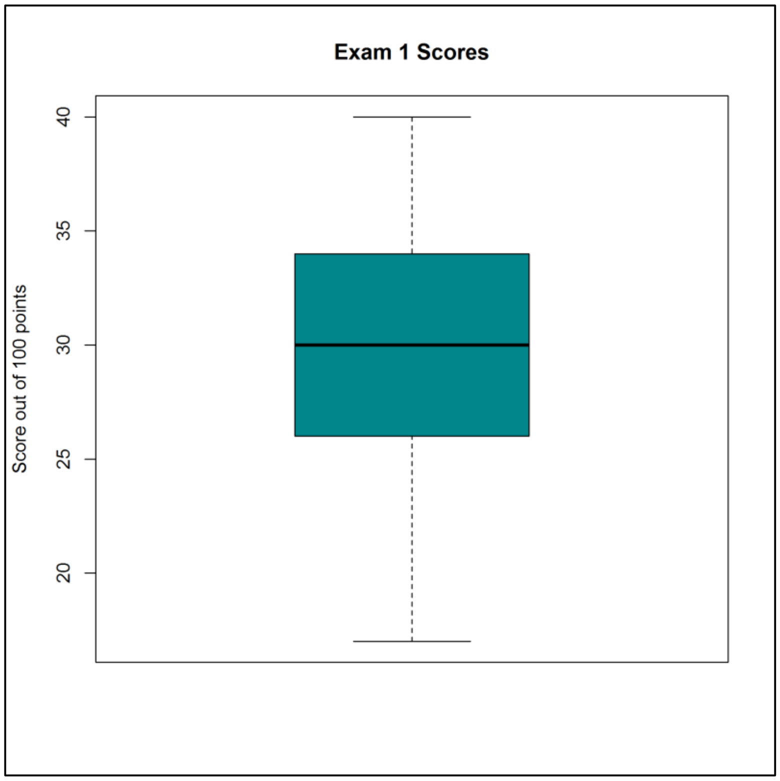


b. Shade the box of the boxplot with your favorite color. Provide the R code below.
   i. Hint: The link below contains a color chart for all plotting colors available in R. To select a color using its number id in the chart, use the command colors()[#] where # is the color id number.
   ii. http://research.stowers-institute.org/efg/R/Color/Chart/index.htm

Note: There are several ways to reference a color in R. You can use its numeric reference (for example, 1 – 8 for the standard colors or colors()[#] for the expanded color list in the colors() vector), you can use its character string handle (for example, 'turquoise4'), or you can use its hexadecimal reference (for example, "#00868B"; see the function below for finding this). The R code below shows all three options for the same color.

R Code:
```
# Multiple options that give the same figure

boxplot(data$exam1,
        main='Exam 1 Scores',
        ylab='Score out of 100 points',
        col=colors()[639])    # Color Number

     # OR

colors()[639]
boxplot(data$exam1,
        main='Exam 1 Scores',
        ylab='Score out of 100 points',
        col="turquoise4")   # Color Name

     # OR

GetColorHexAndDecimal <- function(color){
     c <- col2rgb(color)
     sprintf("#%02X%02X%02X %3d %3d %3d", c[1],c[2],c[3],
c[1], c[2], c[3])}
GetColorHexAndDecimal("turquoise4")
boxplot(data$exam1,
        main='Exam 1 Scores',
        ylab='Score out of 100 points',
        col="#00868B")   # Color Hexadecimal
```

R Output:

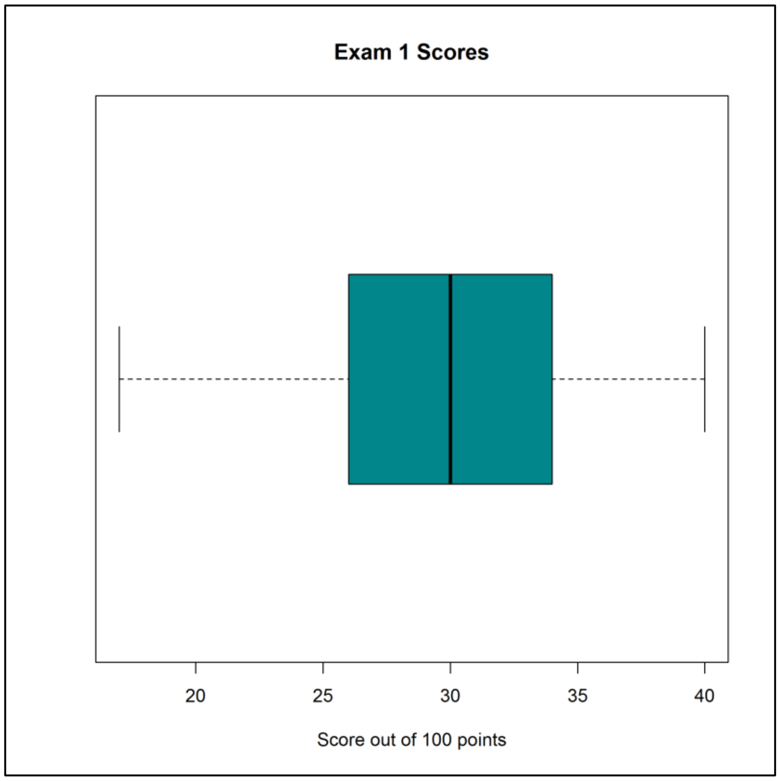


c. Suppose you prefer to have the score values listed on the x-axis instead of the y-axis. Using the R help page, find the optional input that will allow you to format the boxplot in this way. Provide the R code below.

R Code:

```
boxplot(data$exam1,
        main='Exam 1 Scores',
        xlab='Score out of 100 points',
        col=colors()[639],
        horizontal=TRUE)
```

R Output:



d. Once a figure has been created, you may want to export it so that it can be inserted into a report or presentation. There are many ways to achieve this. For example, you can use the Export button in the plotting window of R Studio save the figure as a png or pdf file manually. However, when creating several figures during a complex data analysis or simulation study, saving figures manually would become tedious. We can automate exporting figures using the pdf() or or png() or jpeg() functions in R. We will focus on png(), but the other functions work similarity.

The minimum input for these functions is "filename" which gives the directory, filename, and extension of the file that is being exported. By setting the working directory, you do NOT have to include the file path. For example, to export the boxplot made on page 1 as a figure file named Plot1 to the folder listed in the setwd() command as a png file, I would use the following R Code:

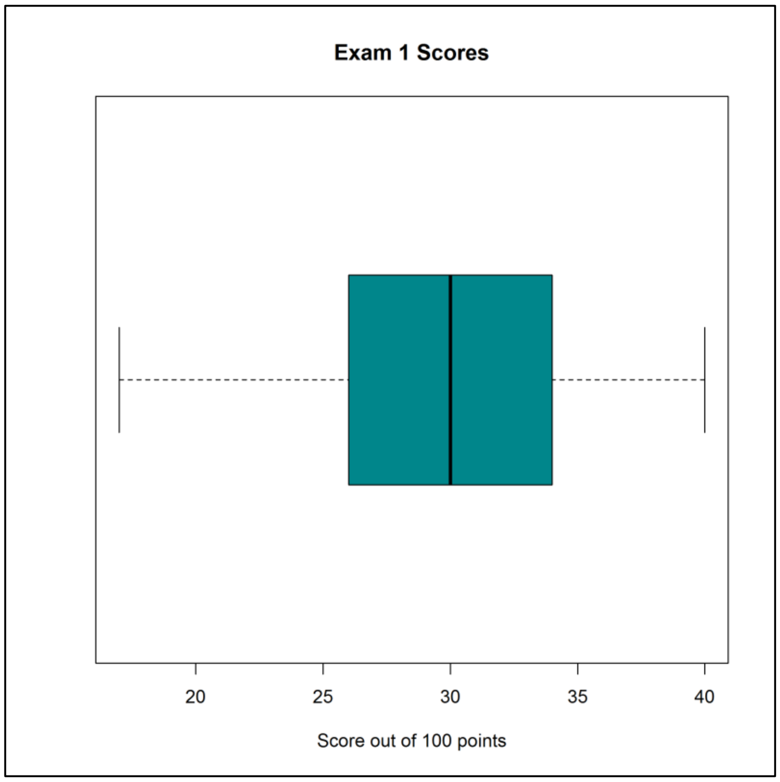
```
png(filename="Plot1.png")
# Insert plotting code
boxplot(data$exam1)
dev.off()
```

Using the code developed in part (c) and the example syntax above, export your figure as either png file or pdf file. Provide the R code below and exported figure below.

R Code:

```
# - Set working directory to where you want the figure
#   to be exported (could be different than where the
#   data file was stored)
setwd("M:\\BIOS 721")

png('Exam1Boxplot.png')
boxplot(data$exam1,
        main='Exam 1 Scores',
        xlab='Score out of 100 points',
        col=colors()[639],
        horizontal=TRUE)
dev.off()
```

R Output:

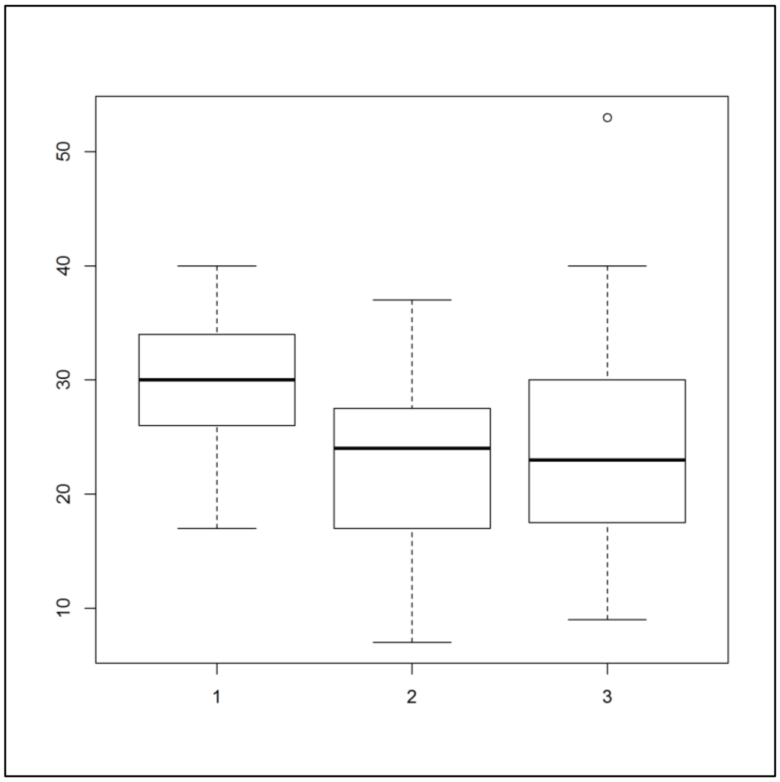


2. Suppose you now want to plot the scores for all three exams in a single figure as side-by-side vertical boxplots.
   a. Using the R help page, find the optional input that will allow you to format the boxplot in this way. Provide the R code below.
      i. Hint: You may have to do some data manipulation in order to create the figure.

R Code:

```
# - Multiple options give the same figure
exam <- c(data$exam1,data$exam2,data$exam3)
group <- c(rep(1,length(data$exam1)),
           rep(2,length(data$exam2)),
           rep(3,length(data$exam3)))
boxplot(exam~group)

     # OR

boxplot(data$exam1,data$exam2,data$exam3)
```

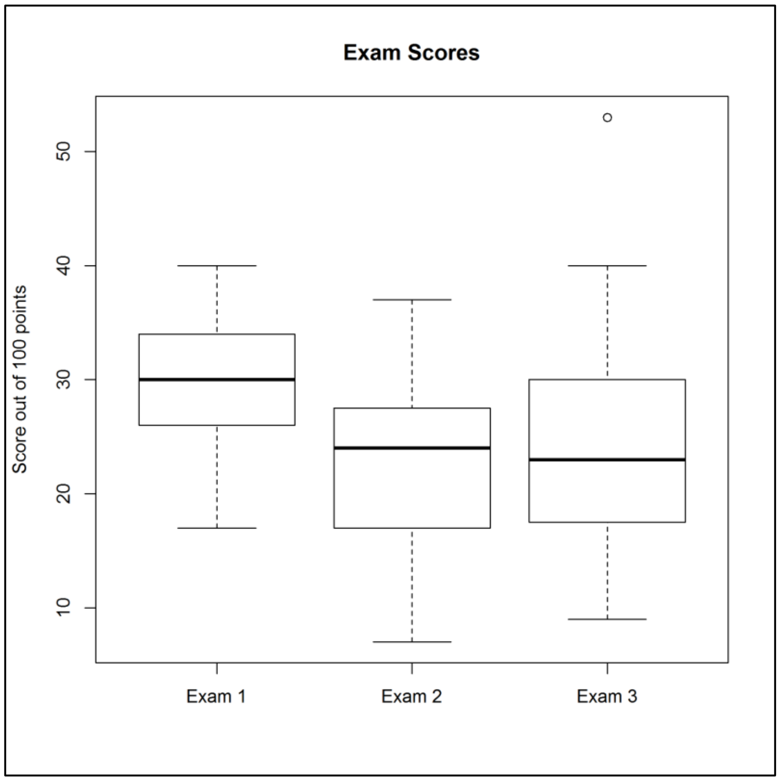
R Output:



b. Using the R help page, find the optional inputs that will allow you to add the main title and axes labels listed below in (i) – (iii). Provide the R code below.
   i. Title the figure 'Exam Scores'.
   ii. Label the y-axis as 'Score out of 100 points'.
   iii. Label the x-axis as 'Exam 1', 'Exam 2', and 'Exam 3'.

R Code:
```
boxplot(exam~group,
        main='Exam Scores',
        ylab='Score out of 100 points',
        names=c('Exam 1','Exam 2','Exam 3'))
```

R Output:



c. For Exam 3, there is an open circle plotted at the upper end of the range. By default, the boxplot() function in R will identify "potential outliers" using this convention.
   i. Using the R help page, determine how R identifies potential outliers. That is, what numerical algorithm is used?

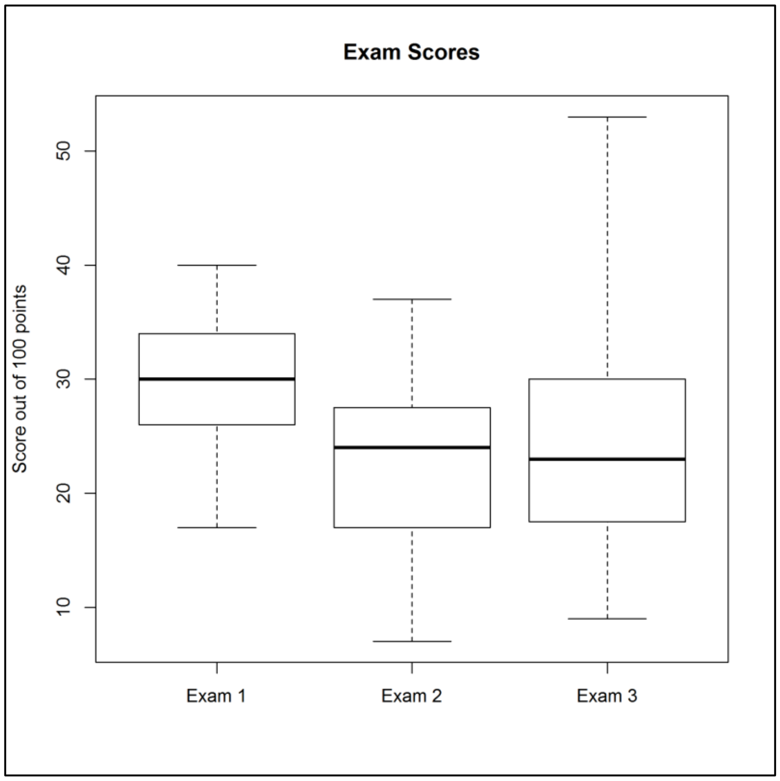
The ‘range’ input controls this feature of the figure. If the range > 0, then the whiskers extend to range*IQR of the observed data. The default value of the range is 1.5, so any data value is that more than 1.5*IQR below Q1 or above Q3 will be marked as a potential outlier by default.

ii. Using the R help page, find the optional inputs that will allow you to extend the “whiskers” of the boxplot (also known as a box-and-whisker plot) to the minimum and maximum data values. Provide the R code below.

R Code:

```
boxplot(exam~group,
        main='Exam Scores',
        ylab='Score out of 100 points',
        names=c('Exam 1','Exam 2','Exam 3'),
        range=0)
```

R Output:

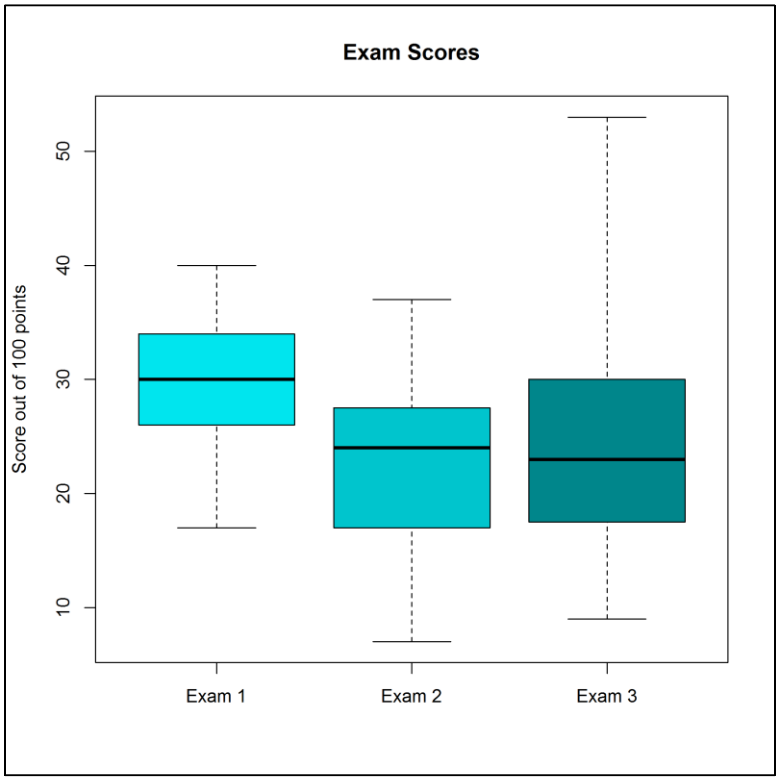


d. Using the R help page, find the optional input that will allow you to shade each box in the figure created in part (c) a different color. Pick your favorites! Provide the R code below.

R Code:

```
boxplot(exam~group,
        main='Exam Scores',
        ylab='Score out of 100 points',
        names=c('Exam 1','Exam 2','Exam 3'),
        range=0,
        col=colors()[637:639])
```

R Output:



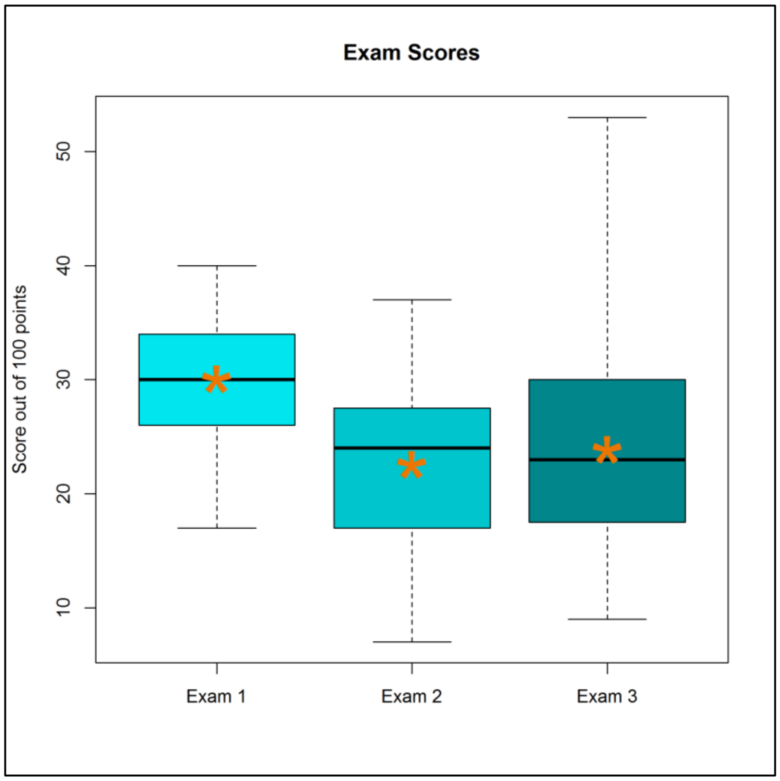
e. Using the R help page, examine the information for the 'at' input (last one listed). Using this information, add a large star (*) for each exam's mean score to the figure created in part (d). The star should be overlaid on top of each box plot, and the star should be a different color than the underlying boxplot. Make sure the code is automated – that is, do NOT hard code the value of the means into the figure code. The code to create the figure should work and report the correct means even if a new data set with updated scores was provided. Provide the R code below.

Note: Based on the R help page, the box plots are plotted at the values 1:n where n is the number of groups. Using the information, the approach for this problem should be to (1) calculate the exam means, (2) store them in a vector, and (3) then plot them as the 'y' with the 'x' being the vector 1:3.

R Code:

```
boxplot(exam~group,
        main='Exam Scores',
        ylab='Score out of 100 points',
        names=c('Exam 1','Exam 2','Exam 3'),
        range=0,
        col=colors()[637:639])

exam.means <- apply(data[,2:4],2,mean,na.rm=TRUE)

lines(1:3,exam.means,type='p',
      pch='*',cex=5,col='darkorange2')
```

R Output:

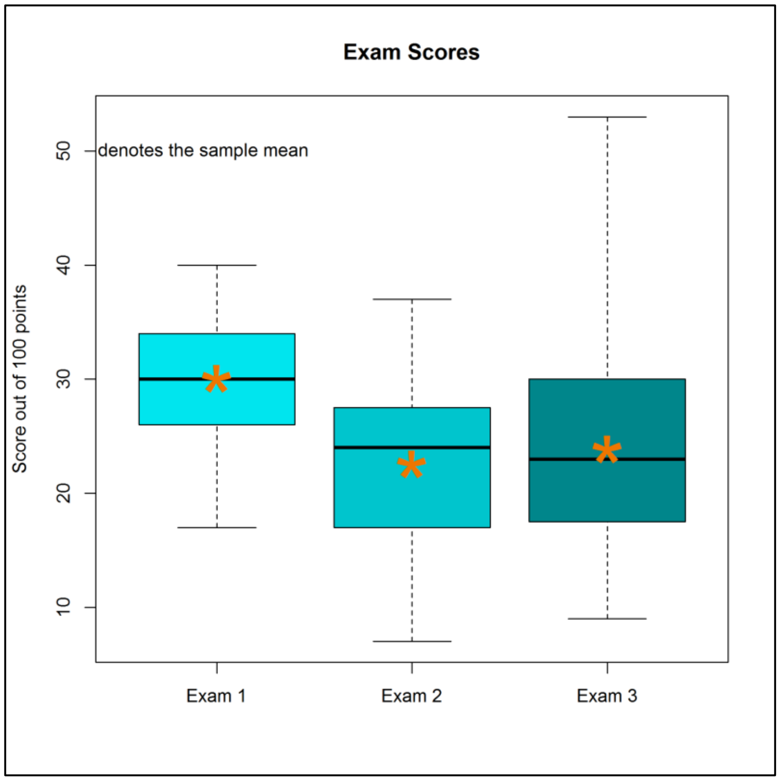


f. Add text to the figure created in part (e) that explains what the star represents. Please provide R code below.

R Code:

```
boxplot(exam~group,
        main='Exam Scores',
        ylab='Score out of 100 points',
        names=c('Exam 1','Exam 2','Exam 3'),
        range=0,
        col=colors()[637:639])

exam.means <- apply(data[,2:4],2,mean,na.rm=TRUE)

lines(1:3,exam.means,type='p',
      pch='*',cex=5,col='darkorange2')

text(0.9,50,labels='* denotes the sample mean')
```

R Output:

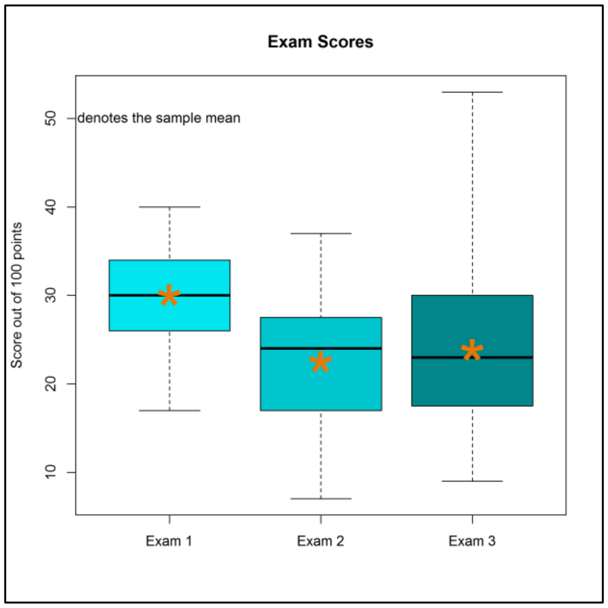


g. Export the figure created in part (f) as a png file. Provide the R code below and exported figure below.

R Code:

```
# - Set working directory to where you want the figure
#   to be exported (could be different than where the
#   data file was stored)
setwd("M:\\BIOS 721")

png('AllExamsBoxplot.png')
boxplot(exam~group,
        main='Exam Scores',
        ylab='Score out of 100 points',
        names=c('Exam 1','Exam 2','Exam 3'),
        range=0,
        col=colors()[637:639])

exam.means <- apply(data[,2:4],2,mean,na.rm=TRUE)

lines(1:3,exam.means,type='p',
      pch='*',cex=5,col='darkorange2')

text(0.9,50,labels='* denotes the sample mean')
dev.off()
```

R Output:



3. Suppose you wanted to compare normal distributions with different means and with different standard deviations by plotting the probability density function (pdf) of several different normal distributions.

   The function below creates a sequence of typical values for a normal random variable with mean μ and standard deviation σ and the corresponding values of the pdf using the dnorm() function. Using this function, respond to the following questions. That is, the function creates the 'plotting data' needs to create a figure of the pdf for any normal distribution.

```
norm.plot.data <- function(mu,sigma) {
     x <- seq(-3.5,3.5,by=0.01)*sigma + mu
     y <- dnorm(x,mean=mu,sd=sigma)
     return(list(x=x,y=y))              }
```

   a. Using the function given above, create the following sets of plotting data. Provide the R code below.
      i. Normal with mean 0 and standard deviation 1.
      ii. Normal with mean 2 and standard deviation 1.
      iii. Normal with mean 0 and standard deviation 2.
      iv. Normal with mean 2 and standard deviation 2.

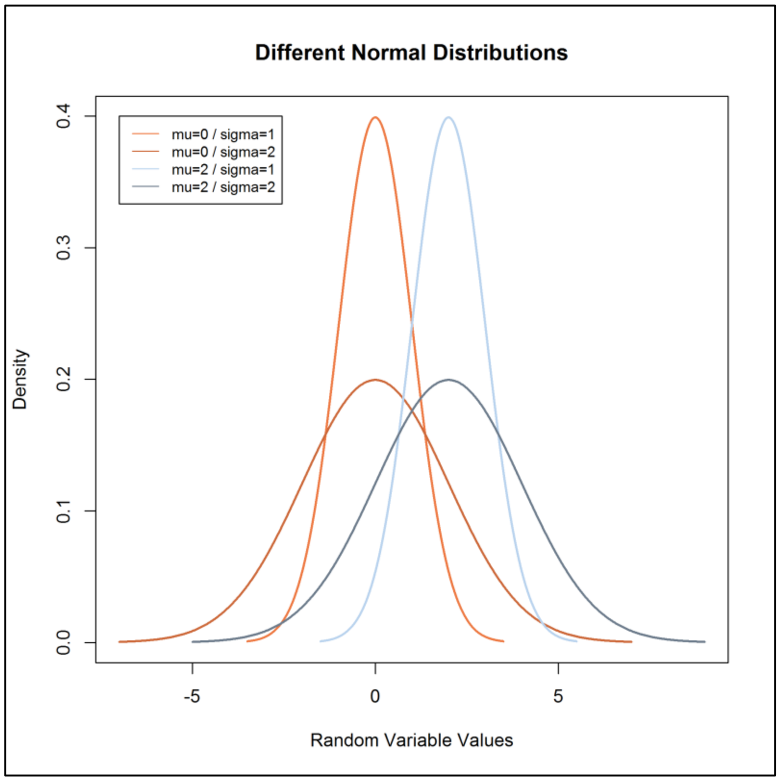
      R Code:
```
norm.01 <- norm.plot.data(mu=0,sigma=1)
norm.02 <- norm.plot.data(mu=0,sigma=2)
norm.21 <- norm.plot.data(mu=2,sigma=1)
norm.22 <- norm.plot.data(mu=2,sigma=2)
```

   b. Create a figure that plots all 4 densities in the same figure as different line. Plot each line with the same line style, but make each line a different color. Add an appropriate legend to the figure. Add appropriate main title and axes labels. Make sure the x-axis and y-axis limits are appropriate. Export the figure as a png file. Provide the R code below and exported figure below.

      R Code:
```
# - Set working directory to where you want the figure
#   to be exported (could be different than where the
#   data file was stored)
setwd("M:\\BIOS 721")
```

```
png(file='pdfsin1.png')
plot(norm.01$x,norm.01$y,
     type="l",col='sienna2',lwd=2,
     ylim=range(norm.01$y,norm.02$y,norm.21$y,norm.22$y),
     xlim=range(norm.01$x,norm.02$x,norm.21$x,norm.22$x),
     main='Different Normal Distributions',
     ylab='Density',
     xlab='Random Variable Values')
lines(norm.02$x,norm.02$y,col='sienna3',lwd=2)
lines(norm.21$x,norm.21$y,col='slategray2',lwd=2)
lines(norm.22$x,norm.22$y,col='slategray4',lwd=2)
legend(x=-7,y=0.4,cex=0.8,lty=rep(1,4),
       legend=c('mu=0 / sigma=1','mu=0 / sigma=2','mu=2 /
sigma=1','mu=2 / sigma=2'),

col=c('sienna2','sienna3','slategray2','slategray4'))
dev.off()
```

R Output:

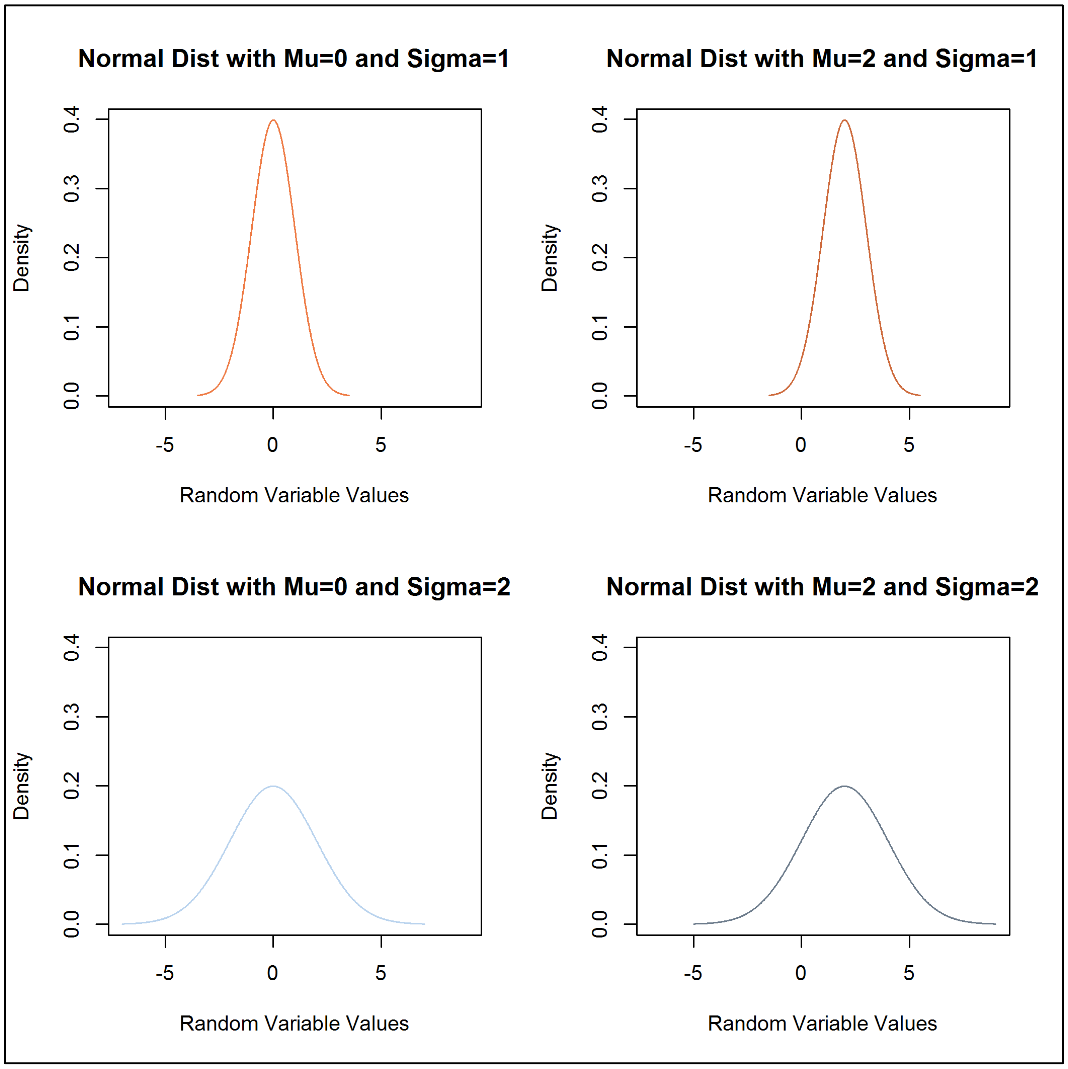


c. Crete a set of figures that plots each density in a separate plotting window such that you create one figure with 4 separate plots. Organize the plots so that the effect of changing the distribution's mean is compared across columns and the effect of changing the distribution's standard deviation is compared down rows. Instead of adding a legend, make sure to include an informative title with each plot in the figure. Make sure the x-axis and y-axis limits are the same for each plot in the figure and they capture all relevant information for all 4 plots. Provide the R code below and exported figure below.

R Code:

```
# - Set working directory to where you want the figure
#   to be exported (could be different than where the
#   data file was stored)
setwd("M:\\BIOS 721")

png(file='pdfs2by2.png')

y.lim=range(norm.01$y,norm.02$y,norm.21$y,norm.22$y)
x.lim= range(norm.01$x,norm.02$x,norm.21$x,norm.22$x)

par(mfrow=c(2,2))
plot(norm.01$x,norm.01$y,
     type="l",col='sienna2',
     ylim=y.lim, xlim=x.lim,
     main='Normal Dist with Mu=0 and Sigma=1',
     ylab='Density',
     xlab='Random Variable Values')

plot(norm.21$x,norm.21$y,col='sienna3',
     type="l",
     ylim=y.lim, xlim=x.lim,
     main='Normal Dist with Mu=2 and Sigma=1',
     ylab='Density',
     xlab='Random Variable Values')

plot(norm.02$x,norm.02$y,col='slategray2',
     type="l",
     ylim=y.lim, xlim=x.lim,
     main='Normal Dist with Mu=0 and Sigma=2',
     ylab='Density',
     xlab='Random Variable Values')

plot(norm.22$x,norm.22$y,col='slategray4',
     type="l",
     ylim=y.lim, xlim=x.lim,
     main='Normal Dist with Mu=2 and Sigma=2',
     ylab='Density',
     xlab='Random Variable Values')
dev.off()
```

R Output:

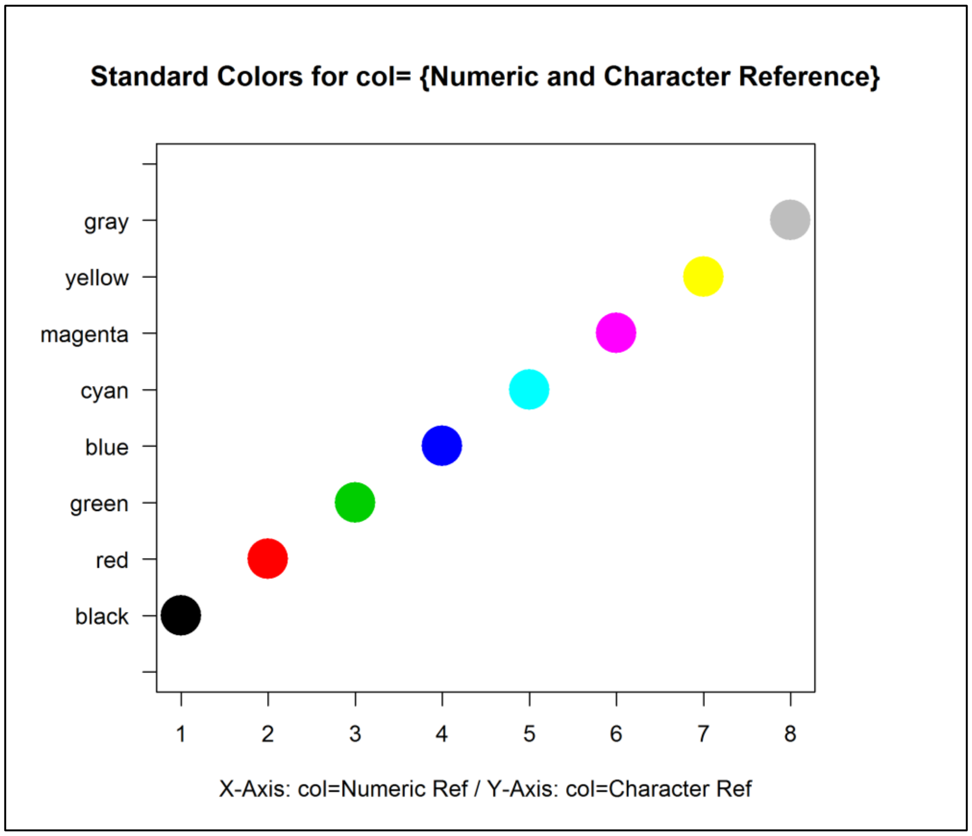


4. Suppose you are working with a data set containing 4 continuous variables. In order to examine the relationships between these variables, you would like to create a panel of bivariate scatter plots. That is, you would like to create a single figure that contains a matrix of scatter plots, one plot for each pair of continuous variables in your data set. This figure is often referred to as a Draftsman's plot. Using a search engine, find an R function that will create this type of figure. If you need to install a package in order to access the function, please note the name of the library.

   There are many correct answers …

   - pairs() function; no additional packages needed
   - scatterplot.matrix() function; 'car' package
   - cpairs() function; 'gclus' package
   - splom() function; 'lattice' package
   - plotmatrix() function; 'ggplot2' package

5. Recreate the following figure:



R Code:
```
# Increase plotting margins (especially the left side)
# to give extra room for text labels
# - Default is (bottom,left,top,right) = (5,4,4,2)
par(mar=c(5,5.5,5,5.5))

x <- 1:8
y <- 1:8

plot(x,y,
     col=1:8,
     cex=4,
     pch=16,
     ylim=c(0,9),
     axes=FALSE,
     main='Standard Colors for col= {Numeric and Character
Reference}',
     xlab='X-Axis: col=Numeric Ref / Y-Axis: col=Character
Ref',
     ylab='')
axis(1, at = 1:8)
axis(2, at = 0:9,
     labels = c(NA,'black','red','green','blue',
                'cyan','magenta','yellow','gray',NA),
     las=2)
box()
```