BIOS 545 R - Homework #3

Due 11:59 PM on April 14, 2020

- Due by 11:59 PM on 04/14/20. Submit your answer sheet in a file named using the convention of `BIOS545_LastName_FirstName_HW3.Rmd` (or `BIOS545_LastName_FirstName_HW3.R`) to Canvas. A penalty of 15% per day late applies to submissions that arrive after the due date and time. After three days, late submissions will no longer be accepted and a score of 0 will be rendered.
- In submitting your homework you are indicating that all work is your own. Please do not develop your solutions collaboratively or share/accept code with other students (current or former).
- Putting comments in your code is helpful. At least we can understand what you were trying or intending to do which makes it easier for us to award partial credit even if the code isn't working as intended.
- Note that your figures should be very close to what you see here. Your code should generate a readable graphic that is easy to see.
- For purposes of this exercise you might find it helpful to work either on a large display or run R outside of R Studio since the graphics pane can sometimes behave strangely if it isn't sized correctly. You can always adjust the sizes of the panes within Rstudio to give the plot window more room.
- On Windows systems you can use the built in R Interface that provides more room when viewing graphics. On Apple systems you can use `quartz()` to see the graphics displayed in a larger window.

Problem 1 - Indometh Data - 25 Points

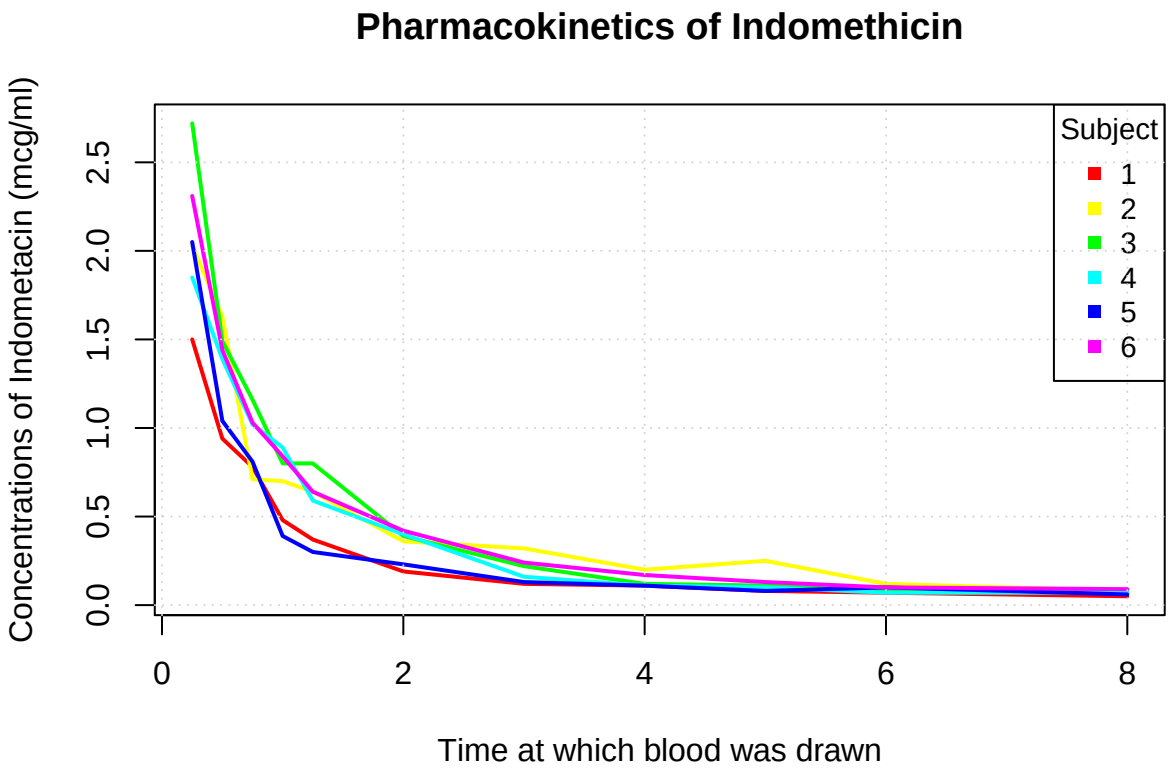
The purpose of this exercise is to plot some information relating to the Indometh data set which is provided to you as part of R.

1.1) You will be writing a function that does a line plot of the concentration of Indometacin observed in the blood over time. You must use BASE R GRAPHICS for this problem ! Each line will represent a different subject (and color) thus you will need a legend so users of your function can distinguish between the lines. This legend also needs to be in order of Subjects 1 through 6.

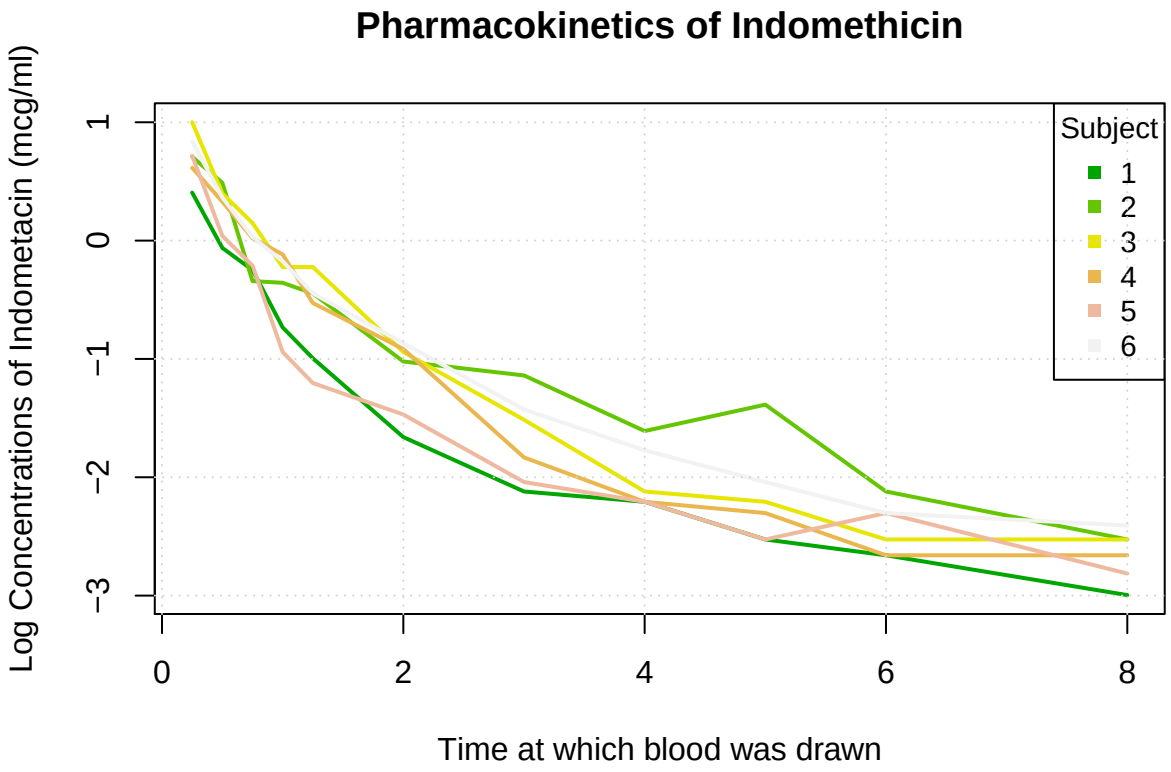
You will also allow users of your function to specify a "log"" argument that, if TRUE, would first take the natural logarithm of the concentration before making the plot. This helps make the decline in concentrations over time more apparent. Note that the log transformation will apply only to the concentration - NOT the time. Users can also supply a vector of colors. The default is to use 6 colors from the rainbow palette. A shell for the function, including default arguments, would look like:

```
plot.indometh <- function(mydf=Indometh,log=FALSE,colors=rainbow(6)) {

  # INPUT:      log=FALSE a T/F value. If TRUE then first take
  #             the log of the concentration before rendering the plot
  #
  #        colors - a vector of valid color values corresponding to
  #                 the colors to be used for each subject


       (your code goes here)
}
```

```
plot.indometh(mydf=Indometh)
```



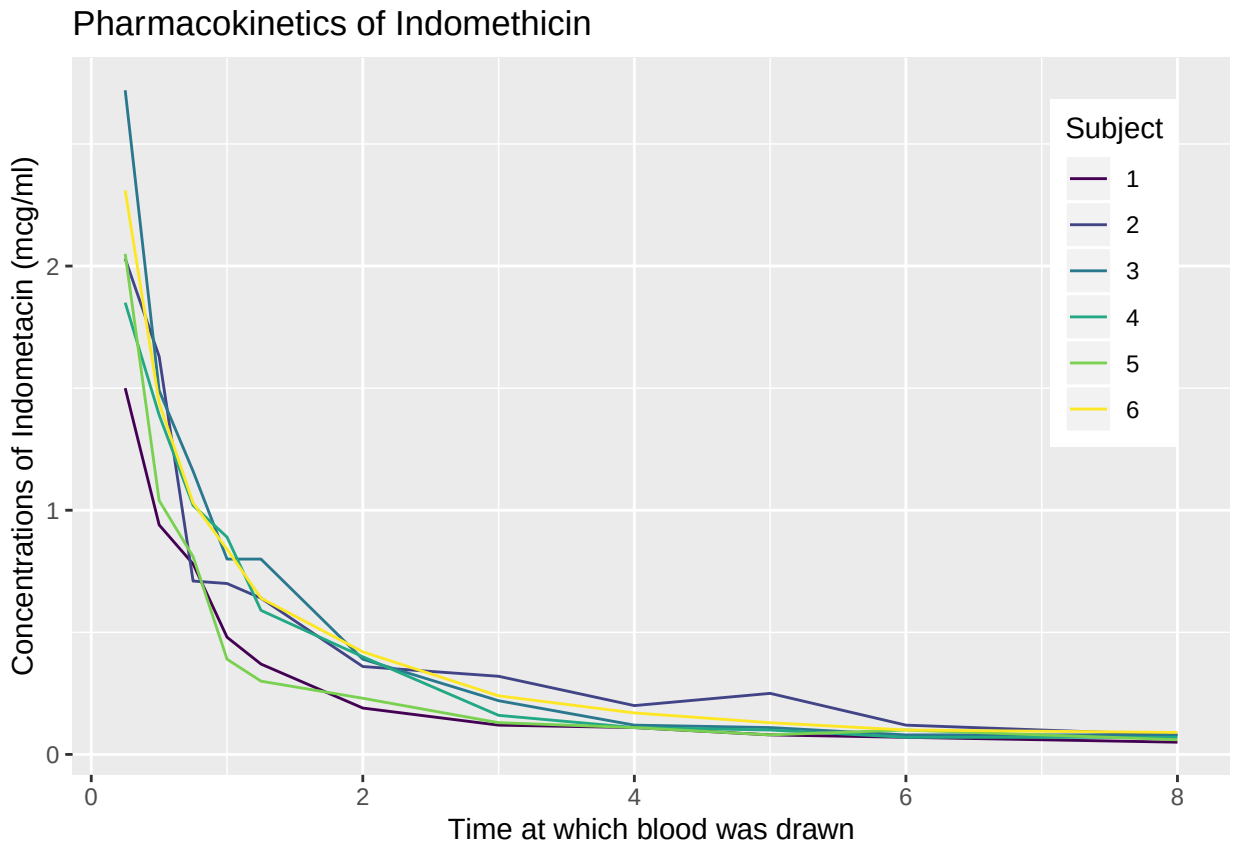

```
plot.indometh(mydf=Indometh, log=TRUE,colors=terrain.colors(6))
```



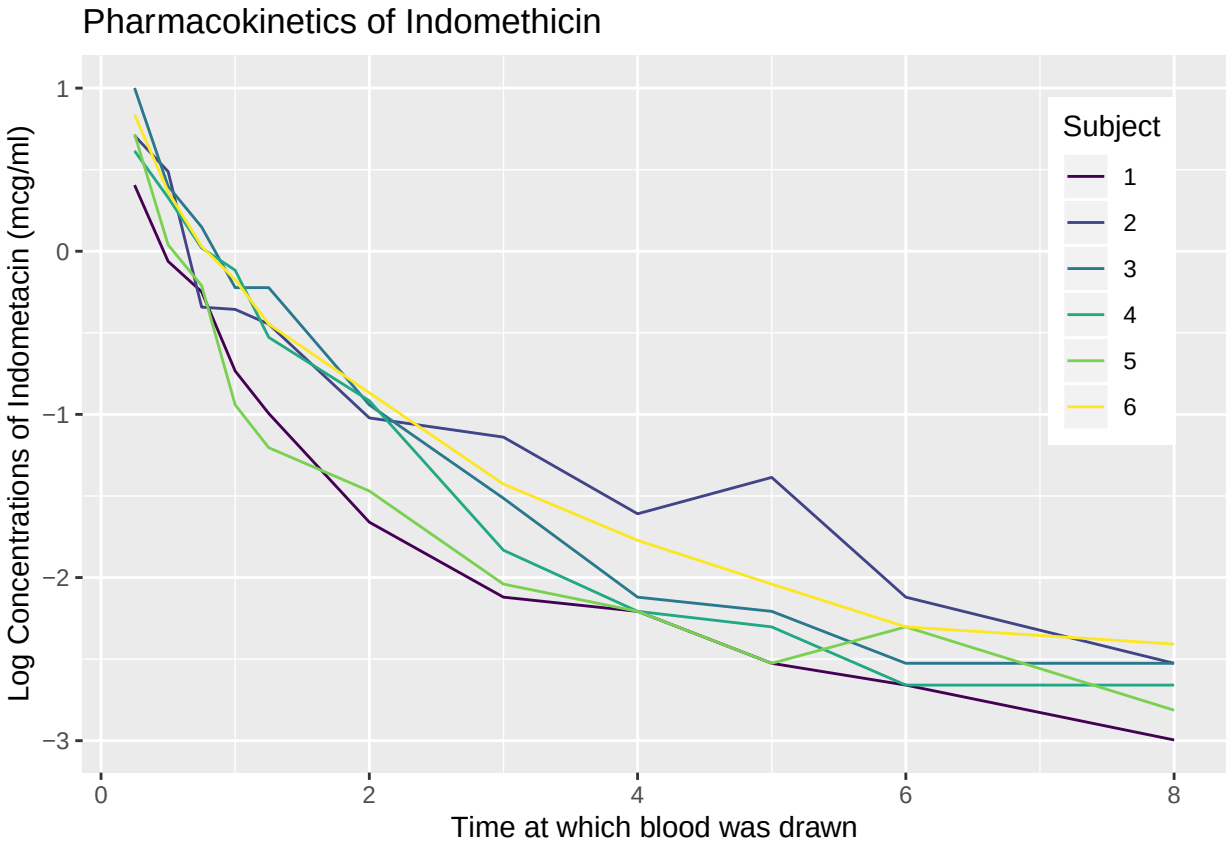
1.2) Now do it wth ggplot commands. The previously stated requirements apply:

```
plot.indometh.gg(mydf=Indometh)
```

```
## Loading required package: ggplot2
```




```
plot.indometh.gg(mydf=Indometh,log=TRUE)
```



Problem 2 - Unemployment Statistics - 25 Points

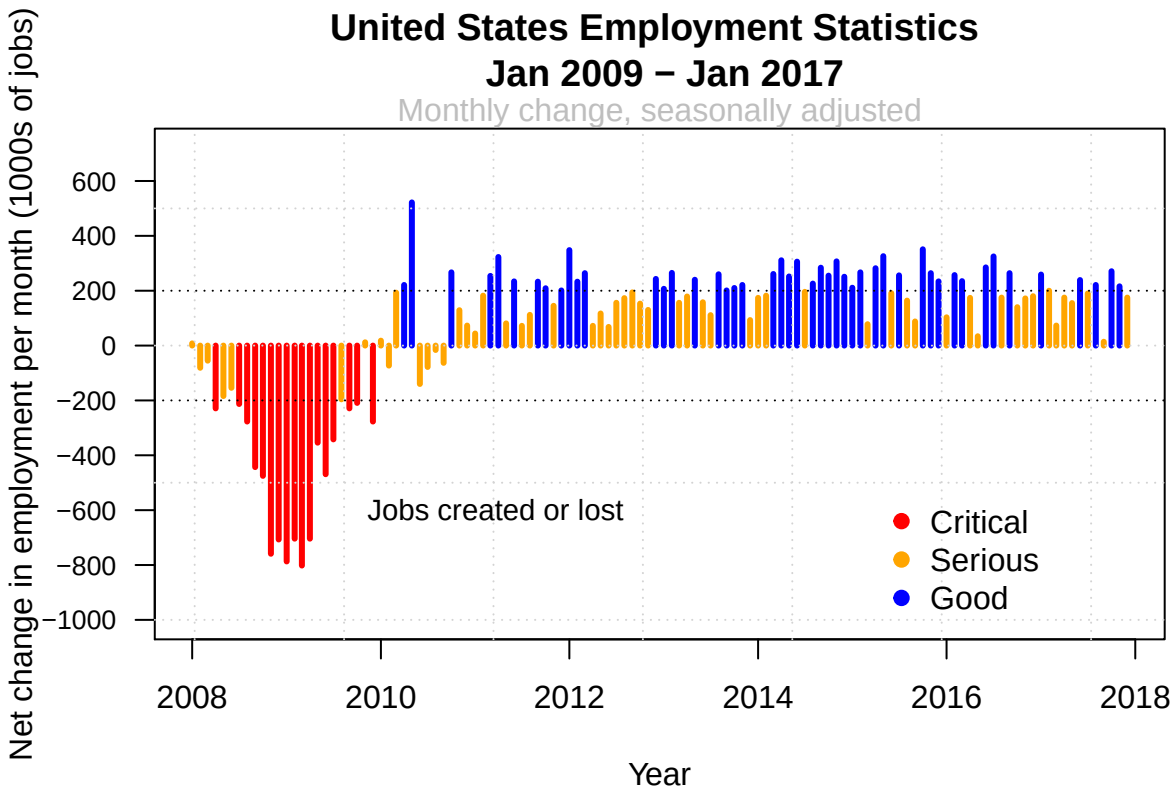
Here is some information relating to United States Employment Statistics during the period of January 2009 to January 2017. Downlaod the supporting file:

```
u <- "https://www.dropbox.com/s/np1penwhfm6r71y/unemployment.csv?dl=1"

download.file(u,"unemployment.csv")

unemp <- read.csv("unemployment.csv",stringsAsFactors = FALSE)
```

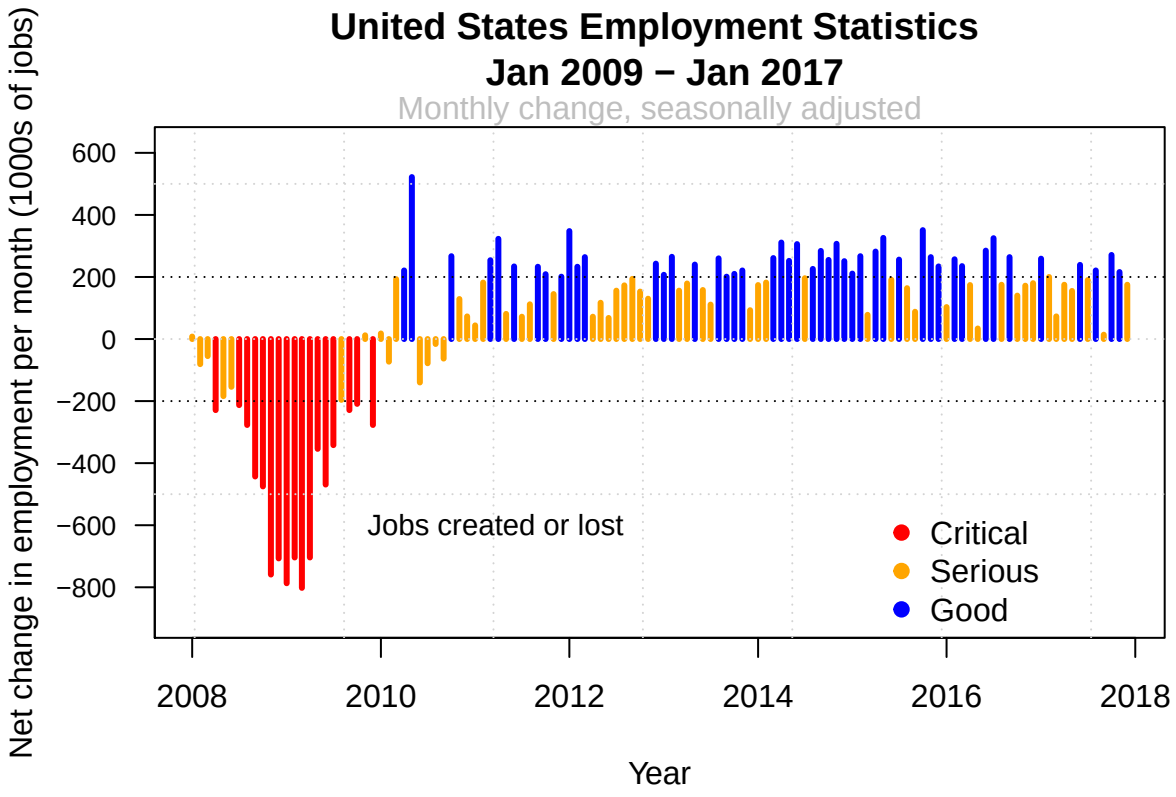
Your goal is to reproduce the following graph. Write a function called `myUnemp` that displays the following information. Notice that the plot type has what appear to be `skinny bars` which corresponds to one of the plot types we explored in the early part of the first graphics lecture.

- You will need to find a way to make the bars a little wider to match the output below.
- The values greater than the minimum Y value and less than or equal to -200 are to appear in red. The values greater than -200 and less than or equal to 200 are to be represented in orange. The values greater than 200 are to appear in blue.
- With respect to the legend, notice that it does not have a surrounding box.
- Notice also that the X-Axis are dates so when you attempt to display the legend and text on the graph you will have to take this into consideration.
- Notice that there are two dotted horizontal lines at -200 and 200 on the Y-Axis

```
myUnemp(unemp,200)
```




```
myUnemp(unemp,100)
```



```
myUnemp <- function(df=unemp, ylimit=100) {
#
# Function to plot the unemployment data
#
# INPUT: df - the unemployment data frame
#        ylimit - A factor by which to extend the y-axis
#                 limits
#

}
```

Problem 3 - Plotting MPG Data - 20 Points

Please refer to the plots below. Use the `ggpplot2` package to construct replicas of the graphics. You will be using the `mpg` dataset which is also part of the ggplot2 package. You are plotting the hwy (highway) mileage vs cty (city) mileage for each value of the trans (transmission) category. You should provide a function named `mympg()` that we can call to reproduce the plots. Here is a shell of the function with some default arguments. The arguments are largely to influence the plot character that will be used to display the points, the color of the points, and the position of the legend. Note too that the plot character requested must also be matched in the legend. We talked about an approach on how to do this in class.

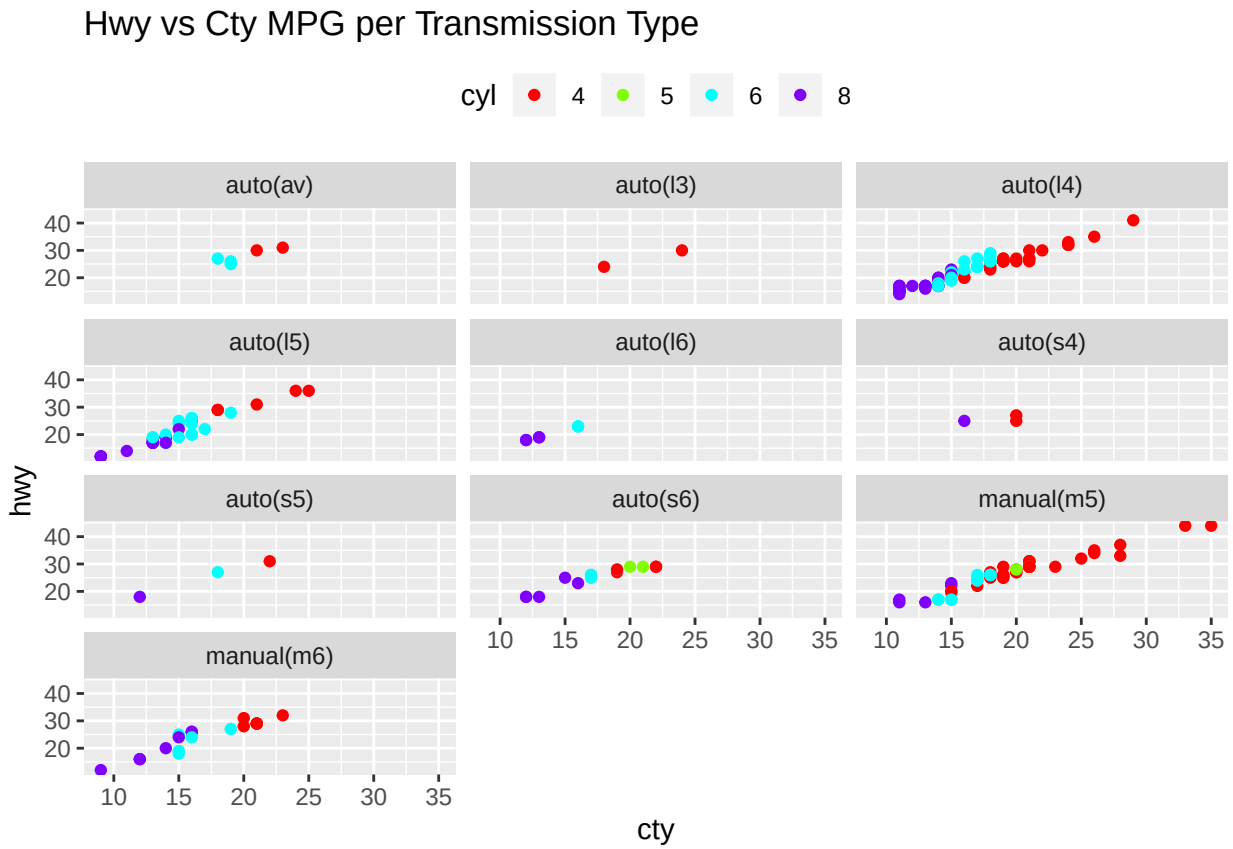
- Do reasonable error checking - Make sure the pch value is valid.
- Make sure that there are 4 colors
- It might help for you to turn trans into an actual factor since ggplot2 graphics likes for you to explicitly declare factors.
- Note the pos argument that tells you where the legend should be. Valid values are "left","right","top", or "bottom". Check for valid values.
- If the requested position is on the left or right then make sure the legend uses a single column whereas if the position is top or bottom then the number of columns should be 4.

```
mympg <- function(df=mpg,colors=rainbow(4),pos="top",pch=19,ncol=3) {


   # INPUT: df - input data frame, we use mpg here
   #        colors - a 4 element vector with valid color names
   #                 or hex values
   #        pos    - where to put the legend "left","right","top",
   #                 "bottom"
   #        pch    - a plot character value for the points and
   #                 legend
   #                 (See example(pch) )
   #        ncol   - number of columns, default is 3
   #
   # OUTPUT: a ggplot2 scatterplot

   # The following two statements are required to load in the mpg data

   stopifnot(require(ggplot2))
   data(mpg)

   Do error checking on arguments

   Read in mpg file
```

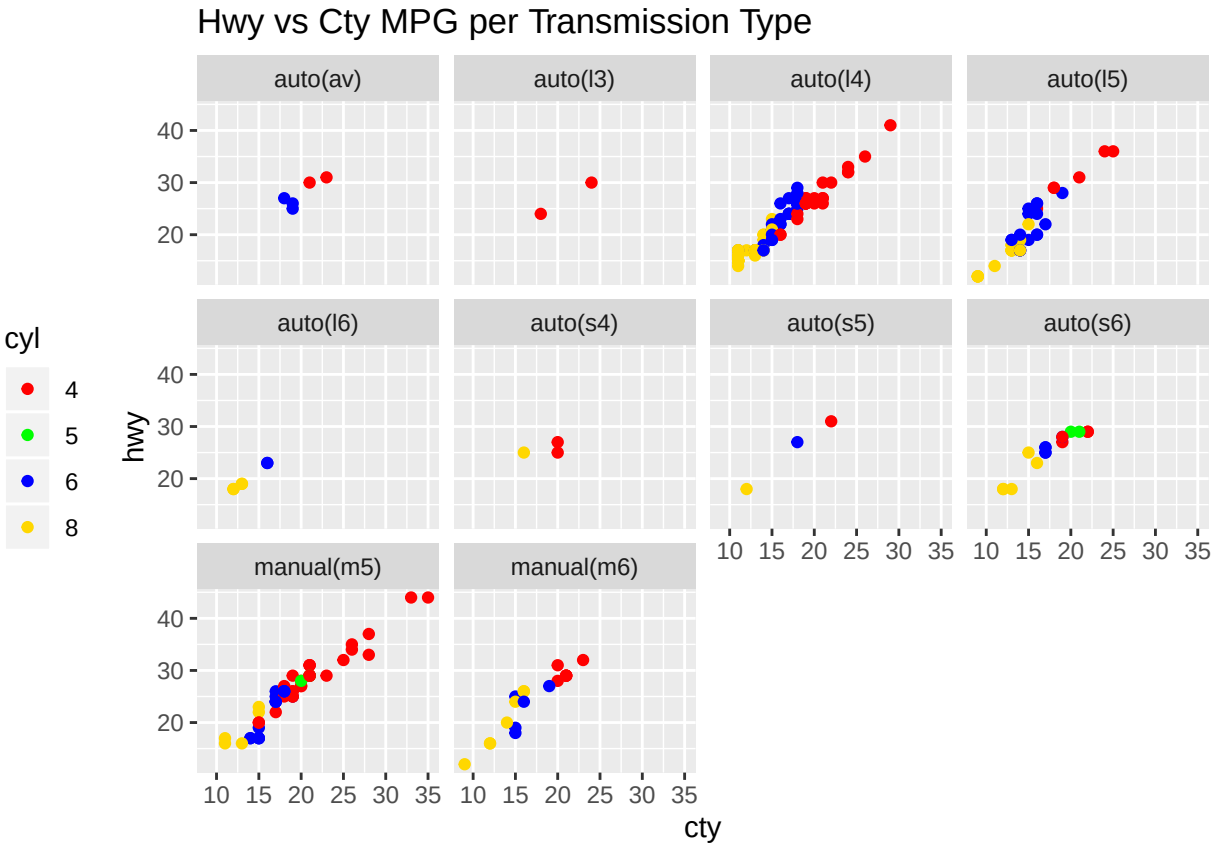
```
   Create factors if you want

   Display the data

}
```

```
mympg(pos="lleft")
Error in mympg(pos="lleft") :
  Sorry - valid positions are left, right, top, or bottom
```

```
mympg()
```




```
mympg(colors=c("red","green","blue","gold"),pos="left",ncol=4)
```



Problem 4) Plotting Confidence Intervals - 30 points

Write a function called plot.ci that plots confidence intervals as described herein.

We covered some of this during the lecture on vectors. Here we take a sample of a larger population, create a distribution of means, (or confidence intervals in this case), by sampling with replacement. This gives us, (in this case), 100 rows of confidence intervals. We have a suspected mean that might or might not be contained with in each of the 100 intervals. (Check below where I present the test data). You create a matix wherein each row contains the confidence intervals. So you will need to loop through these rows to determine if it containts the test/suspected mean under consideration.

If it does then we want to plot a blue line segment (hint: use segments() function. Type in ?segments to see the usage of this function). If it isn't then we want to plot a red line segment. The graphic will look like the one below. Hint: create a plot of type="n" and then create horizontal line segements corresponding to the length of the interval. You will have to set your xlimits to represent the extremes of all the intervals plus some "buffer" room (see the plot). If you don't then not all of the intervals will fit. Also with each successive interval, (row in the confidence interval matrix), you will have to move "up" vertically by some constant spacing value to prevent overplotting. I need to see some separation in your intervals. If the intervals are overplotted and its a big mess I can't give you full credit.

Note that, in addition to the confidence interval plot, it should also return a vector corresponding to the rows of sci that do not contain the suspected mean. In the graphic below the vertical line represents the suspected mean value.

This basically means that you have to set your ylim accordingly also. This will require experimentation. There are a 100 intervals to plot so that should help you start. You might find it useful, at least for purposes of testing, to add in an argument called "index" that sets the spacing factor. Use this to help compute a proper ylim, (even though you don't need to list y limits), and also to increment vertically in the loop.

```r
# Here is your test data. Don't put anything in your function
# that is hard coded to this data.

set.seed(199)
x <- rnorm(100,10)
smean <- mean(x)

sci <- replicate(100,t.test(sample(x,rep=T))$conf.int)
sci <- t(sci)

# Look at the first few rows of the 100 confidence intervals
head(sci)
```

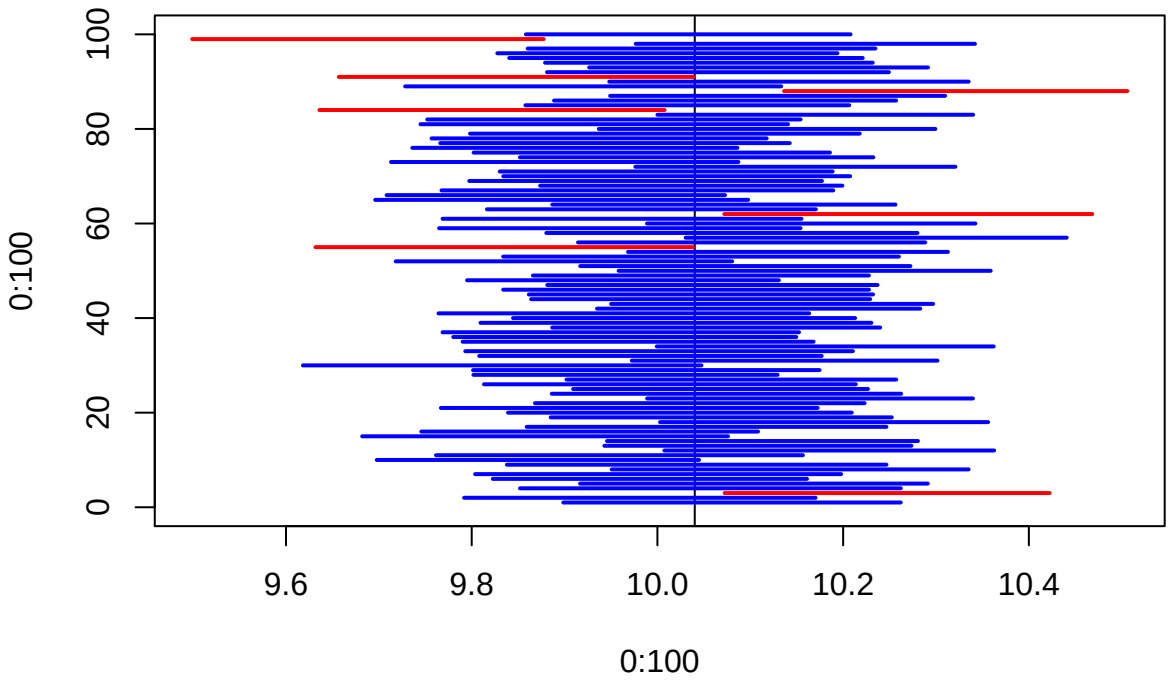
```
##           [,1]     [,2]
## [1,]  9.898567 10.26207
## [2,]  9.791784 10.17030
## [3,] 10.072362 10.42233
## [4,]  9.851935 10.26210
## [5,]  9.916648 10.29118
## [6,]  9.822679 10.16106
```

```r
# Look at the Mean

smean    # suspected mean
```

```
## [1] 10.04018
```

Now actually use the function

```r
# YOU WRITE THE FOLLOWING FUNCTION

outside <- plot.ci(sci, smean)
```



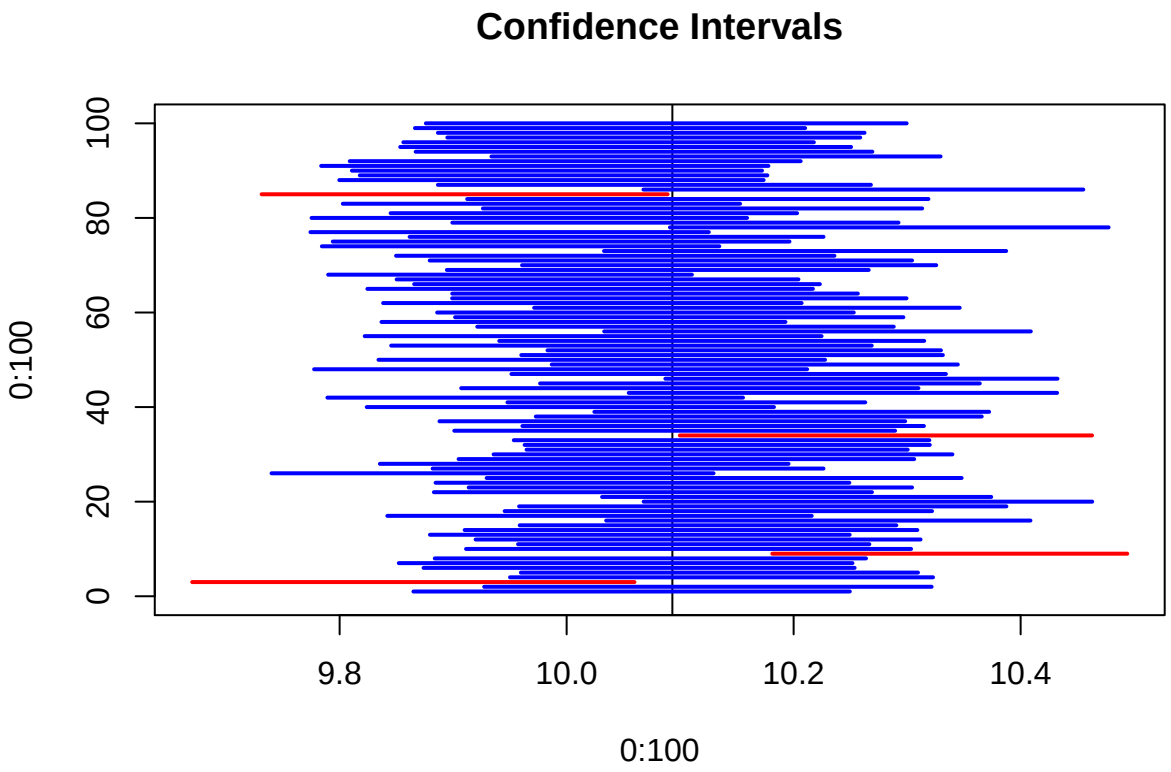

```
outside
```

```
## [1]  3 55 62 84 88 91 99
```

```
# Show the intervals that do not contain the mean

sci[outside,]
```

```
##           [,1]      [,2]
## [1,] 10.072362 10.422329
## [2,]  9.631683 10.038216
## [3,] 10.072018 10.468302
## [4,]  9.635984 10.007711
## [5,] 10.136403 10.506023
## [6,]  9.656866 10.038800
## [7,]  9.498947  9.877431
```

```
# I WILL ALSO TEST IT ON THIS DATA:

set.seed(430)
x <- rnorm(100,10)
smean <- mean(x)
sci <- replicate(100,t.test(sample(x,rep=T))$conf.int)
sci <- t(sci)

plot.ci(sci,smean)
```


Confidence Intervals
0:100
0
20
40
60
80
100
9.8
10.0
10.2
10.4
0:100

```
## [1]  3  9 34 85
```