Homework 5 1,3,5,6

Andrew Marshall

6/28/2018

# Data

As with Homework 4, all the numeric values you need, other than 0.05, 0, 1, 2 and 3 are defined below:

Year=c(1936, 1946, 1951, 1963, 1975, 1997, 2006)  
CaloriesPerRecipeMean <- c(2123.8, 2122.3, 2089.9, 2250.0, 2234.2, 2249.6, 3051.9)  
CaloriesPerRecipeSD <- c(1050.0, 1002.3, 1009.6, 1078.6, 1089.2, 1094.8, 1496.2)  
CaloriesPerServingMean <- c(268.1, 271.1, 280.9, 294.7, 285.6, 288.6, 384.4)  
CaloriesPerServingSD <- c(124.8, 124.2, 116.2, 117.7, 118.3, 122.0, 168.3)  
ServingsPerRecipeMean <- c(12.9, 12.9, 13.0, 12.7, 12.4, 12.4, 12.7)  
ServingsPerRecipeSD <- c(13.3, 13.3, 14.5, 14.6, 14.3, 14.3, 13.0)  
  
CookingTooMuch.dat <- data.frame(  
 Year=Year,  
 CaloriesPerRecipeMean = CaloriesPerRecipeMean,  
 CaloriesPerRecipeSD = CaloriesPerRecipeSD,  
 CaloriesPerServingMean = CaloriesPerServingMean,  
 CaloriesPerServingSD = CaloriesPerServingSD,  
 ServingsPerRecipeMean = ServingsPerRecipeMean,  
 ServingsPerRecipeSD = ServingsPerRecipeSD  
)  
  
sample.size <- 18  
tenth.increment <- 0.10  
hundredth.increment <- 0.100  
idx.1936 <- 1  
idx.2006 <- length(CaloriesPerRecipeMean)  
idxs36\_07 <- c(idx.1936,idx.2006)  
alpha=0.05

Nearly all the same restrictions apply. Specifically

* There are 6 exercises. Choose 4 to be graded.
* One of the exercises must be completed in both SAS and R. Make sure you document this in the output.
* The other 3 exercises are to be complete in either R or SAS. Make sure you document this in the output.
* You may choose to work the other exercises. If you do, put these *after* the exercises you want graded. Otherwise, we’ll grade the first four exercises and stop grading there. Time permitting, we’ll provide feedback on the additional exercises.
* You are not required to write additional (other than previous Homework) functions for this exercise, but you may. You will be expected to clearly document additional functions - identify the expected inputs and outputs.
* There are no unit tests for this exercise. Where applicable, you should compare your results to comparable results in previous homework.
* This is an exercise in working with data tables. For three exercises, you will create data tables from sequences; for the last two you are expected to read data from files. One requires you to convert a data table to matrices, without calling data.frame directly.
* If you choose SAS, some of the exercises will require you to transfer data between PROC IML and the DATA step. You may need to redefine macros from previous assignments.

# Exercise 1

Repeat the analysis from Exercise 1, Homework 4. This time, the results wil be in a data table with 49 rows. There will be 7 columns in the final table, Year1, Year2, Mean1, SD1, Mean2, SD2 and CohenD. This table will have the same duplications as your matrix in Homework 4 (don’t worry, we’ll remove those in later exercises).

## Part a

Create a data table where each row represents a different combination between years. It should look something like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Year1 | Year2 | Mean1 | Mean2 | SD1 | SD2 |
| 1936 | 1936 | 2123.8 | 2123.8 | 1050.0 | 1050.0 |
| 1946 | 1936 | 2122.3 | 2123.8 | 1002.3 | 1050.0 |
| 1951 | 1936 | 2089.9 | 2123.8 | 1009.6 | 1050.0 |
| 1963 | 1936 | 2250.0 | 2123.8 | 1078.6 | 1050.0 |
| … | … | … | … | … | … |
| 1936 | 1946 | 2123.8 | 2122.3 | 1050.0 | 1002.3 |
| 1946 | 1946 | 2122.3 | 2122.3 | 1002.3 | 1002.3 |
| … | … | … | … | … | … |

Start with the vectors defined in Data. You can reuse the matrices from the last homework, if you wish, or you can create new sequences while constructing the data table

If you do this exercise in SAS, use IML to create the vectors, the use CREATE to create a data table.

#Initial Data frame setup  
Year1.df <- data.frame(  
 Year1 = rep(CookingTooMuch.dat$Year[1:7],idx.2006)  
)  
Year2.df <- data.frame(  
 Year2 = rep(CookingTooMuch.dat$Year[1:7],each=idx.2006)  
)  
Mean1.df <- data.frame(  
 Mean1 = rep(CookingTooMuch.dat$CaloriesPerRecipeMean[1:7],idx.2006)  
)  
Mean2.df <- data.frame(  
 Mean2 = rep(CookingTooMuch.dat$CaloriesPerRecipeMean[1:7],each=idx.2006)  
)  
SD1.df <- data.frame(  
 SD1 = rep(CookingTooMuch.dat$CaloriesPerRecipeSD[1:7],idx.2006)  
)  
SD2.df <- data.frame(  
 SD2 = rep(CookingTooMuch.dat$CaloriesPerRecipeSD[1:7],each=idx.2006)  
  
)  
  
#Final data frame setup  
CPR.df <- data.frame(  
 Year1.df,  
 Year2.df,  
 Mean1.df,  
 Mean2.df,  
 SD1.df,  
 SD2.df  
 )  
  
CPR.df

## Year1 Year2 Mean1 Mean2 SD1 SD2  
## 1 1936 1936 2123.8 2123.8 1050.0 1050.0  
## 2 1946 1936 2122.3 2123.8 1002.3 1050.0  
## 3 1951 1936 2089.9 2123.8 1009.6 1050.0  
## 4 1963 1936 2250.0 2123.8 1078.6 1050.0  
## 5 1975 1936 2234.2 2123.8 1089.2 1050.0  
## 6 1997 1936 2249.6 2123.8 1094.8 1050.0  
## 7 2006 1936 3051.9 2123.8 1496.2 1050.0  
## 8 1936 1946 2123.8 2122.3 1050.0 1002.3  
## 9 1946 1946 2122.3 2122.3 1002.3 1002.3  
## 10 1951 1946 2089.9 2122.3 1009.6 1002.3  
## 11 1963 1946 2250.0 2122.3 1078.6 1002.3  
## 12 1975 1946 2234.2 2122.3 1089.2 1002.3  
## 13 1997 1946 2249.6 2122.3 1094.8 1002.3  
## 14 2006 1946 3051.9 2122.3 1496.2 1002.3  
## 15 1936 1951 2123.8 2089.9 1050.0 1009.6  
## 16 1946 1951 2122.3 2089.9 1002.3 1009.6  
## 17 1951 1951 2089.9 2089.9 1009.6 1009.6  
## 18 1963 1951 2250.0 2089.9 1078.6 1009.6  
## 19 1975 1951 2234.2 2089.9 1089.2 1009.6  
## 20 1997 1951 2249.6 2089.9 1094.8 1009.6  
## 21 2006 1951 3051.9 2089.9 1496.2 1009.6  
## 22 1936 1963 2123.8 2250.0 1050.0 1078.6  
## 23 1946 1963 2122.3 2250.0 1002.3 1078.6  
## 24 1951 1963 2089.9 2250.0 1009.6 1078.6  
## 25 1963 1963 2250.0 2250.0 1078.6 1078.6  
## 26 1975 1963 2234.2 2250.0 1089.2 1078.6  
## 27 1997 1963 2249.6 2250.0 1094.8 1078.6  
## 28 2006 1963 3051.9 2250.0 1496.2 1078.6  
## 29 1936 1975 2123.8 2234.2 1050.0 1089.2  
## 30 1946 1975 2122.3 2234.2 1002.3 1089.2  
## 31 1951 1975 2089.9 2234.2 1009.6 1089.2  
## 32 1963 1975 2250.0 2234.2 1078.6 1089.2  
## 33 1975 1975 2234.2 2234.2 1089.2 1089.2  
## 34 1997 1975 2249.6 2234.2 1094.8 1089.2  
## 35 2006 1975 3051.9 2234.2 1496.2 1089.2  
## 36 1936 1997 2123.8 2249.6 1050.0 1094.8  
## 37 1946 1997 2122.3 2249.6 1002.3 1094.8  
## 38 1951 1997 2089.9 2249.6 1009.6 1094.8  
## 39 1963 1997 2250.0 2249.6 1078.6 1094.8  
## 40 1975 1997 2234.2 2249.6 1089.2 1094.8  
## 41 1997 1997 2249.6 2249.6 1094.8 1094.8  
## 42 2006 1997 3051.9 2249.6 1496.2 1094.8  
## 43 1936 2006 2123.8 3051.9 1050.0 1496.2  
## 44 1946 2006 2122.3 3051.9 1002.3 1496.2  
## 45 1951 2006 2089.9 3051.9 1009.6 1496.2  
## 46 1963 2006 2250.0 3051.9 1078.6 1496.2  
## 47 1975 2006 2234.2 3051.9 1089.2 1496.2  
## 48 1997 2006 2249.6 3051.9 1094.8 1496.2  
## 49 2006 2006 3051.9 3051.9 1496.2 1496.2

## Part b.

Below is a wrapper function that accepts a vector as a parameter and returns Cohen’s for values in that vector. Modify this function so that it selects appropriate elements from the vector and calls \*\*your\* Cohen’s function from the previous homework. Assume that table.row is a row from the data table you created in Part a.

# cohen.wrapper <- function(table.row) {  
# return(cohen.d(table.row[3],table.row[5],table.row[4],table.row[6]))  
  
#Cohen's d function for use with Cohen Wrapper  
d\_12 <- function(m\_1,m\_2,s\_1,s\_2) {  
d\_12\_var <- (abs(m\_1-m\_2))/(sqrt((s\_1^2 + s\_2^2)/2))  
return(d\_12\_var)  
}  
  
  
cohen.wrapper <- function(table.row) {  
 return(d\_12(table.row[3],table.row[4],table.row[5],table.row[6]))  
}  
  
CohenD.df <- data.frame(  
 cohen.wrapper(CPR.df)   
)  
colnames(CohenD.df) <- c("CohenD")  
  
CohenD.df

## CohenD  
## 1 0.0000000000  
## 2 0.0014613799  
## 3 0.0329126822  
## 4 0.1185648879  
## 5 0.1031988304  
## 6 0.1172813930  
## 7 0.7180655556  
## 8 0.0014613799  
## 9 0.0000000000  
## 10 0.0322081482  
## 11 0.1226529318  
## 12 0.1069122990  
## 13 0.1212878216  
## 14 0.7300001111  
## 15 0.0329126822  
## 16 0.0322081482  
## 17 0.0000000000  
## 18 0.1532541622  
## 19 0.1374083574  
## 20 0.1516529874  
## 21 0.7537385973  
## 22 0.1185648879  
## 23 0.1226529318  
## 24 0.1532541622  
## 25 0.0000000000  
## 26 0.0145768162  
## 27 0.0003680766  
## 28 0.6148491002  
## 29 0.1031988304  
## 30 0.1069122990  
## 31 0.1374083574  
## 32 0.0145768162  
## 33 0.0000000000  
## 34 0.0141025177  
## 35 0.6248568588  
## 36 0.1172813930  
## 37 0.1212878216  
## 38 0.1516529874  
## 39 0.0003680766  
## 40 0.0141025177  
## 41 0.0000000000  
## 42 0.6119970024  
## 43 0.7180655556  
## 44 0.7300001111  
## 45 0.7537385973  
## 46 0.6148491002  
## 47 0.6248568588  
## 48 0.6119970024  
## 49 0.0000000000

If you choose SAS for this exercise, define a macro to implement Cohen’s formula, using syntax compatible with IML. This macro should have four parameters appropriate for Cohen’s .

## Part c

Compute CohenD and add this to your table using apply and the wrapper function in Part b. Print this table and compare to the matrix you produced in Homework 4.

If you choose SAS, create a second data table, starting with the data table in part a (use SET in the data statement). Insert your macro in the body of this data step. Print your table. Use the names of your data table as parameters to this macro. Your macro will be replaced with the formula; assign the macro invocation a data variable CohenD.

# using Apply function to compute CohenD column  
CohenD <- apply(CPR.df,1,cohen.wrapper)  
  
#Adding new column to data frame  
cbind(CPR.df,CohenD)

## Year1 Year2 Mean1 Mean2 SD1 SD2 CohenD  
## 1 1936 1936 2123.8 2123.8 1050.0 1050.0 0.0000000000  
## 2 1946 1936 2122.3 2123.8 1002.3 1050.0 0.0014613799  
## 3 1951 1936 2089.9 2123.8 1009.6 1050.0 0.0329126822  
## 4 1963 1936 2250.0 2123.8 1078.6 1050.0 0.1185648879  
## 5 1975 1936 2234.2 2123.8 1089.2 1050.0 0.1031988304  
## 6 1997 1936 2249.6 2123.8 1094.8 1050.0 0.1172813930  
## 7 2006 1936 3051.9 2123.8 1496.2 1050.0 0.7180655556  
## 8 1936 1946 2123.8 2122.3 1050.0 1002.3 0.0014613799  
## 9 1946 1946 2122.3 2122.3 1002.3 1002.3 0.0000000000  
## 10 1951 1946 2089.9 2122.3 1009.6 1002.3 0.0322081482  
## 11 1963 1946 2250.0 2122.3 1078.6 1002.3 0.1226529318  
## 12 1975 1946 2234.2 2122.3 1089.2 1002.3 0.1069122990  
## 13 1997 1946 2249.6 2122.3 1094.8 1002.3 0.1212878216  
## 14 2006 1946 3051.9 2122.3 1496.2 1002.3 0.7300001111  
## 15 1936 1951 2123.8 2089.9 1050.0 1009.6 0.0329126822  
## 16 1946 1951 2122.3 2089.9 1002.3 1009.6 0.0322081482  
## 17 1951 1951 2089.9 2089.9 1009.6 1009.6 0.0000000000  
## 18 1963 1951 2250.0 2089.9 1078.6 1009.6 0.1532541622  
## 19 1975 1951 2234.2 2089.9 1089.2 1009.6 0.1374083574  
## 20 1997 1951 2249.6 2089.9 1094.8 1009.6 0.1516529874  
## 21 2006 1951 3051.9 2089.9 1496.2 1009.6 0.7537385973  
## 22 1936 1963 2123.8 2250.0 1050.0 1078.6 0.1185648879  
## 23 1946 1963 2122.3 2250.0 1002.3 1078.6 0.1226529318  
## 24 1951 1963 2089.9 2250.0 1009.6 1078.6 0.1532541622  
## 25 1963 1963 2250.0 2250.0 1078.6 1078.6 0.0000000000  
## 26 1975 1963 2234.2 2250.0 1089.2 1078.6 0.0145768162  
## 27 1997 1963 2249.6 2250.0 1094.8 1078.6 0.0003680766  
## 28 2006 1963 3051.9 2250.0 1496.2 1078.6 0.6148491002  
## 29 1936 1975 2123.8 2234.2 1050.0 1089.2 0.1031988304  
## 30 1946 1975 2122.3 2234.2 1002.3 1089.2 0.1069122990  
## 31 1951 1975 2089.9 2234.2 1009.6 1089.2 0.1374083574  
## 32 1963 1975 2250.0 2234.2 1078.6 1089.2 0.0145768162  
## 33 1975 1975 2234.2 2234.2 1089.2 1089.2 0.0000000000  
## 34 1997 1975 2249.6 2234.2 1094.8 1089.2 0.0141025177  
## 35 2006 1975 3051.9 2234.2 1496.2 1089.2 0.6248568588  
## 36 1936 1997 2123.8 2249.6 1050.0 1094.8 0.1172813930  
## 37 1946 1997 2122.3 2249.6 1002.3 1094.8 0.1212878216  
## 38 1951 1997 2089.9 2249.6 1009.6 1094.8 0.1516529874  
## 39 1963 1997 2250.0 2249.6 1078.6 1094.8 0.0003680766  
## 40 1975 1997 2234.2 2249.6 1089.2 1094.8 0.0141025177  
## 41 1997 1997 2249.6 2249.6 1094.8 1094.8 0.0000000000  
## 42 2006 1997 3051.9 2249.6 1496.2 1094.8 0.6119970024  
## 43 1936 2006 2123.8 3051.9 1050.0 1496.2 0.7180655556  
## 44 1946 2006 2122.3 3051.9 1002.3 1496.2 0.7300001111  
## 45 1951 2006 2089.9 3051.9 1009.6 1496.2 0.7537385973  
## 46 1963 2006 2250.0 3051.9 1078.6 1496.2 0.6148491002  
## 47 1975 2006 2234.2 3051.9 1089.2 1496.2 0.6248568588  
## 48 1997 2006 2249.6 3051.9 1094.8 1496.2 0.6119970024  
## 49 2006 2006 3051.9 3051.9 1496.2 1496.2 0.0000000000

# Exercise 2

In this exercise, we reproduce and extend the plot from Exercise 2, Homework 4.

## Part a

Create a data table with a sequence from x3 = using increments defined by tenth.increment, as before. Name this column X.

mu <- 0  
sigma <- 1  
  
 # Generating sequences for x3.df data frames  
 x3 <- seq(from=(mu-3\*sigma),to=(mu+3\*sigma),by=tenth.increment)  
  
x3.df <- data.frame(  
 x = x3  
)  
x3.df

## x  
## 1 -3.0  
## 2 -2.9  
## 3 -2.8  
## 4 -2.7  
## 5 -2.6  
## 6 -2.5  
## 7 -2.4  
## 8 -2.3  
## 9 -2.2  
## 10 -2.1  
## 11 -2.0  
## 12 -1.9  
## 13 -1.8  
## 14 -1.7  
## 15 -1.6  
## 16 -1.5  
## 17 -1.4  
## 18 -1.3  
## 19 -1.2  
## 20 -1.1  
## 21 -1.0  
## 22 -0.9  
## 23 -0.8  
## 24 -0.7  
## 25 -0.6  
## 26 -0.5  
## 27 -0.4  
## 28 -0.3  
## 29 -0.2  
## 30 -0.1  
## 31 0.0  
## 32 0.1  
## 33 0.2  
## 34 0.3  
## 35 0.4  
## 36 0.5  
## 37 0.6  
## 38 0.7  
## 39 0.8  
## 40 0.9  
## 41 1.0  
## 42 1.1  
## 43 1.2  
## 44 1.3  
## 45 1.4  
## 46 1.5  
## 47 1.6  
## 48 1.7  
## 49 1.8  
## 50 1.9  
## 51 2.0  
## 52 2.1  
## 53 2.2  
## 54 2.3  
## 55 2.4  
## 56 2.5  
## 57 2.6  
## 58 2.7  
## 59 2.8  
## 60 2.9  
## 61 3.0

If you choose SAS, do this step in IML.

## Part b

Compute the likelihood and assign this to three columns, L1, L2, L3. These columns will correspond to the values computed for sequences x1, x2 and x3. To make columns L1 and L2 fit in this data table, pad the columns with NA values.

|  |  |  |  |
| --- | --- | --- | --- |
| X | L1 | L2 | L3 |
|  | - | - |  |
|  | - | - |  |
|  | - | - |  |
| … | … | … | … |
|  | - |  |  |
|  | - |  |  |
|  | - |  |  |
| … | … | … | … |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| … | … | … | … |
|  | - | - |  |
|  | - | - |  |

A couple different approaches you might try:

* Use L3 values for all columns, but change values to NA outside the appropriate range (you can use boolean indexes in R).
* Reuse the sequences from Homework 4 to build the data frame, concatenating with sequences of NA as needed.

If you choose SAS, costruct this data in IML, then use CREATE to create a data table.

# Generating sequences for x1.df,x2.df data frames  
 x1 <- seq(from=(mu-sigma),to=(mu+sigma),by=tenth.increment)  
 x2 <- seq(from=(mu-2\*sigma),to=(mu+2\*sigma),by=tenth.increment)  
  
  
  
  
# assigning values for use with data frame setup  
l1 <- (1/(sigma \* sqrt(2 \* pi))) \* (exp(-((x1-mu)^2)/(2\*sigma^2)))  
l2 <- (1/(sigma \* sqrt(2 \* pi))) \* (exp(-((x2-mu)^2)/(2\*sigma^2)))   
l3 <- (1/(sigma \* sqrt(2 \* pi))) \* (exp(-((x3-mu)^2)/(2\*sigma^2)))  
  
  
# Defining L2 and L2 data frames  
L1.df <- data.frame(  
 L1 = l1  
)  
  
L2.df <- data.frame(  
 L2 = l2  
)  
L2.df

## L2  
## 1 0.05399097  
## 2 0.06561581  
## 3 0.07895016  
## 4 0.09404908  
## 5 0.11092083  
## 6 0.12951760  
## 7 0.14972747  
## 8 0.17136859  
## 9 0.19418605  
## 10 0.21785218  
## 11 0.24197072  
## 12 0.26608525  
## 13 0.28969155  
## 14 0.31225393  
## 15 0.33322460  
## 16 0.35206533  
## 17 0.36827014  
## 18 0.38138782  
## 19 0.39104269  
## 20 0.39695255  
## 21 0.39894228  
## 22 0.39695255  
## 23 0.39104269  
## 24 0.38138782  
## 25 0.36827014  
## 26 0.35206533  
## 27 0.33322460  
## 28 0.31225393  
## 29 0.28969155  
## 30 0.26608525  
## 31 0.24197072  
## 32 0.21785218  
## 33 0.19418605  
## 34 0.17136859  
## 35 0.14972747  
## 36 0.12951760  
## 37 0.11092083  
## 38 0.09404908  
## 39 0.07895016  
## 40 0.06561581  
## 41 0.05399097

L3.df <- data.frame (  
 L3 = l3  
)  
  
 #Creating matrix for transition to data frame  
 L.mat <-matrix(nrow = 61,ncol = 4)  
 L.mat[1:61,1] <- x3.df[1:61,1]  
 L.mat[1:61,4] <- L3.df[1:61,1]  
 L.mat[1:61,3] <- L2.df[1:61,1]  
 L.mat[1:61,2] <- L1.df[1:61,1]  
   
   
 #Assigning matrix to data frame  
 L.df <- data.frame(  
 L.mat  
 )  
   
 L.df

## X1 X2 X3 X4  
## 1 -3.0 0.2419707 0.05399097 0.004431848  
## 2 -2.9 0.2660852 0.06561581 0.005952532  
## 3 -2.8 0.2896916 0.07895016 0.007915452  
## 4 -2.7 0.3122539 0.09404908 0.010420935  
## 5 -2.6 0.3332246 0.11092083 0.013582969  
## 6 -2.5 0.3520653 0.12951760 0.017528300  
## 7 -2.4 0.3682701 0.14972747 0.022394530  
## 8 -2.3 0.3813878 0.17136859 0.028327038  
## 9 -2.2 0.3910427 0.19418605 0.035474593  
## 10 -2.1 0.3969525 0.21785218 0.043983596  
## 11 -2.0 0.3989423 0.24197072 0.053990967  
## 12 -1.9 0.3969525 0.26608525 0.065615815  
## 13 -1.8 0.3910427 0.28969155 0.078950158  
## 14 -1.7 0.3813878 0.31225393 0.094049077  
## 15 -1.6 0.3682701 0.33322460 0.110920835  
## 16 -1.5 0.3520653 0.35206533 0.129517596  
## 17 -1.4 0.3332246 0.36827014 0.149727466  
## 18 -1.3 0.3122539 0.38138782 0.171368592  
## 19 -1.2 0.2896916 0.39104269 0.194186055  
## 20 -1.1 0.2660852 0.39695255 0.217852177  
## 21 -1.0 0.2419707 0.39894228 0.241970725  
## 22 -0.9 NA 0.39695255 0.266085250  
## 23 -0.8 NA 0.39104269 0.289691553  
## 24 -0.7 NA 0.38138782 0.312253933  
## 25 -0.6 NA 0.36827014 0.333224603  
## 26 -0.5 NA 0.35206533 0.352065327  
## 27 -0.4 NA 0.33322460 0.368270140  
## 28 -0.3 NA 0.31225393 0.381387815  
## 29 -0.2 NA 0.28969155 0.391042694  
## 30 -0.1 NA 0.26608525 0.396952547  
## 31 0.0 NA 0.24197072 0.398942280  
## 32 0.1 NA 0.21785218 0.396952547  
## 33 0.2 NA 0.19418605 0.391042694  
## 34 0.3 NA 0.17136859 0.381387815  
## 35 0.4 NA 0.14972747 0.368270140  
## 36 0.5 NA 0.12951760 0.352065327  
## 37 0.6 NA 0.11092083 0.333224603  
## 38 0.7 NA 0.09404908 0.312253933  
## 39 0.8 NA 0.07895016 0.289691553  
## 40 0.9 NA 0.06561581 0.266085250  
## 41 1.0 NA 0.05399097 0.241970725  
## 42 1.1 NA NA 0.217852177  
## 43 1.2 NA NA 0.194186055  
## 44 1.3 NA NA 0.171368592  
## 45 1.4 NA NA 0.149727466  
## 46 1.5 NA NA 0.129517596  
## 47 1.6 NA NA 0.110920835  
## 48 1.7 NA NA 0.094049077  
## 49 1.8 NA NA 0.078950158  
## 50 1.9 NA NA 0.065615815  
## 51 2.0 NA NA 0.053990967  
## 52 2.1 NA NA 0.043983596  
## 53 2.2 NA NA 0.035474593  
## 54 2.3 NA NA 0.028327038  
## 55 2.4 NA NA 0.022394530  
## 56 2.5 NA NA 0.017528300  
## 57 2.6 NA NA 0.013582969  
## 58 2.7 NA NA 0.010420935  
## 59 2.8 NA NA 0.007915452  
## 60 2.9 NA NA 0.005952532  
## 61 3.0 NA NA 0.004431848

## Part c

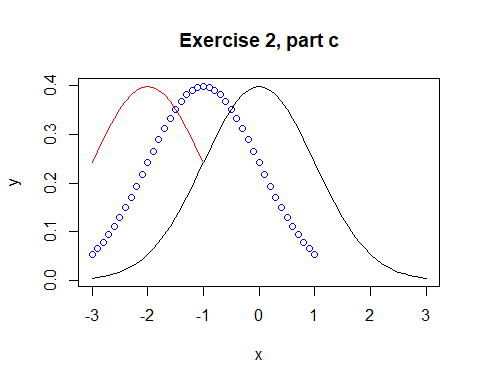
Plot L3 vs X, using formula syntax, with this data table as a parameter to plot, using points as the symbol.

Add points for L2, then L1, using different colors or symbols. Complete the plot by adding vertical lines at and , using colors matching L1 and L2, respectively.

If you use SAS, you will only need to call one SGPLOT block, with multiple series or scatter statements.

If you follow the instructions, you should have a graph of a normal probability distribution with different colors for the parts of the curve representing 1, 2 and 3 standard deviations.

#Assignments for L3 and x3 plots  
x <- c(L.df$X1)  
y <- c(L.df$X4)  
  
#Assignments for L2 and L1 plots  
y2 <- c(L.df$X3)  
y1 <- c(L.df$X2)  
  
# L1, L2, and L3 plots vs x  
plot(x,y,main = "Exercise 2, part c",type="l")  
points(x,y2,col="blue",type="p")  
points(x,y1,col="red",type="l")



## Part d

Use apply (or similar function), compute the sum of columns L1, L2 and L3, multiplied by tenth.increment. Compare these values with the sums calculated for the previous exercise.

sum(L1.df,L2.df,L3.df) \* tenth.increment

## [1] 2.663923

If you use SAS, you can use PROC SUMMARY for this step.

# Exercise 3

Starting with ‘CookingToMuch.dat’, repeat the analysis from Homework 4, Exercise 5.

Append an appropriate column (named Intercept), with all values of 1, to CookingToMuch.dat, then use column indexes to extract appropriate and variables from CookingToMuch.dat. Do not create new matrices or frames for and . You may to coerce or to matrices.

If you use SAS for this exercise, use the data table CookingToMuch, and use IML functions (USE/READ) to read from this data table into matrices.

Compute and print beta.hat as before, and compare to

lm(CaloriesPerRecipeMean ~ CaloriesPerServingMean,data=CookingTooMuch.dat)

##   
## Call:  
## lm(formula = CaloriesPerRecipeMean ~ CaloriesPerServingMean,   
## data = CookingTooMuch.dat)  
##   
## Coefficients:  
## (Intercept) CaloriesPerServingMean   
## -166.923 8.339

Change the eval flag in this for an alternative model.

lm(CaloriesPerRecipeMean ~ 0 + Intercept + CaloriesPerServingMean,data=CookingTooMuch.dat)

# Exercise 4

This exercise will be similar to Exercise 4 in Homework 4.

## Part a

First, find the minimum and maximum values for ServingsPerRecipeMean.

## Part b

Using your Poisson confidence interval function from Homework 3, Exercise 4, calculate the lower and upper bounds to the minimum and maximum means found in Part a. Set LB as the smallest (single value) of these bounds, set UB as the largest (single value) of these bounds.

## Part c

Create a data frame with a column ‘Y’ as a sequence from floor(LB) to ceiling(UB). Add to this data frame two columns, one the Poisson probability from Homework 4 using the maximum servings per recipe mean, and the other using the minimum servings per recipe mean.

## Part d

Plot both probability series against Y, using different colors. Use formula notation for the plots.

Add to this plot two vertical lines, located at the minimum and maximum mean values. Use the same color as the corresponding probablity curves.

Finally, add two pairs of lines corresonding to the upper and lower CI of the minimum and maximum means, calculated in Part b. Use different line types for these lines.

If you do this exercise in SAS, you can do parts a-c in IML, saving the matrices to a data table and use SGPLOT for part d.

# Exercise 5

I was shopping for a motorcycle this spring, and in researching models, found a list of the fastest production motorcycles (<https://en.wikipedia.org/wiki/List_of_fastest_production_motorcycles>) . I edited this page to create a data table in CSV format.

## Part a

Download the file fastest.csv from D2L and read the file into a data frame or table. Print a summary of the table.

PathToFastest = "C:/Users/drewm/Documents/GitHub/code-stat700/fastest.csv"  
Fastest.dat <- read.csv(PathToFastest,header=TRUE)  
  
summary(Fastest.dat)

## Make Model Initial.Model.Year  
## Kawasaki : 7 V-twin : 3 Min. :1894   
## Brough Superior: 3 650SS : 1 1st Qu.:1920   
## BSA : 2 900SS : 1 Median :1965   
## Ducati : 2 Black Lightning : 1 Mean :1958   
## Honda : 2 CBR1100XX Super Blackbird: 1 3rd Qu.:1990   
## Bimota : 1 F Knucklehead : 1 Max. :2015   
## (Other) :17 (Other) :26   
## Engine CC Horsepower MPH   
## Flat twin : 1 Min. : 216.0 Min. : 2.50 Min. : 22.00   
## Inline four :13 1st Qu.: 904.2 1st Qu.: 41.50 1st Qu.: 88.75   
## Inline three : 2 Median : 996.5 Median : 77.00 Median :128.50   
## Parallel twin: 3 Mean : 940.2 Mean : 94.65 Mean :126.35   
## Single : 3 3rd Qu.:1001.2 3rd Qu.:150.75 3rd Qu.:174.50   
## V-four : 1 Max. :1500.0 Max. :310.00 Max. :249.00   
## V-twin :11 NA's :2

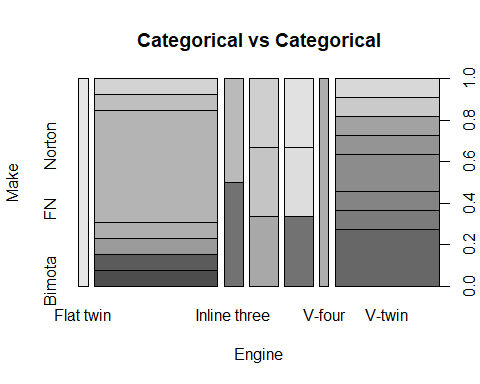
## Part b

To show that the data was read correctly, create three plots. Plot

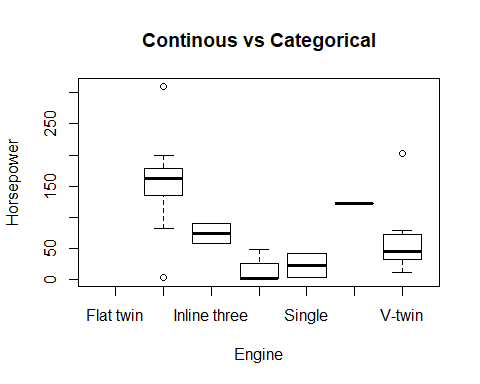
1. Make vs Engine
2. Horsepower vs Engine
3. MPH vs Horsepower

These three plots should reproduce the three types of plots shown in the RegressionEtcPlots video, **Categorical vs Categorical**, **Continuous vs Continuous** and **Continuous vs Categorical**. Add these as titles to your plots, as appropriate.

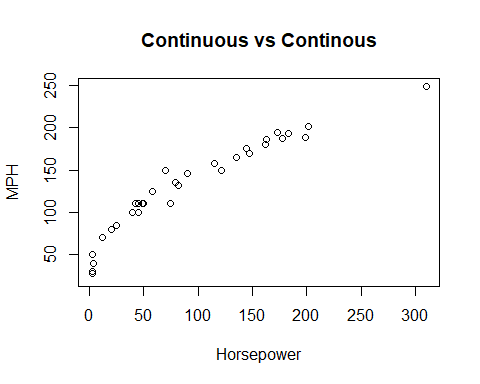
#Categorical vs Categorical plot  
plot(main="Categorical vs Categorical",Make~Engine,data = Fastest.dat)



#Continous vs Categorical plot  
plot(main = "Continous vs Categorical", Horsepower~Engine,data = Fastest.dat)



#Continuous vs Continous plot  
plot(main = "Continuous vs Continous", MPH~Horsepower,data = Fastest.dat)



# Exercise 6

## Part a

Go to <http://www.itl.nist.gov/div898/strd/anova/AtmWtAg.html> and download the file listed under Data File in Table Format (ttps://www.itl.nist.gov/div898/strd/anova/AtmWtAgt.dat)

#File has been downloaded as requested

## Part b

Edit this into a file that can be read into R or SAS, or find an appropriate function that can read the file as-is. You will need to upload this file to D2L along with your Rmd/SAS files. Provide a brief comment on changes you make, or assumptions about the file needed for you file to be read into R/SAS. Read the file into a data frame or data table.

#Comment: No file editing was done. File was downloaded and the code provided access to the necessary file and data.  
  
#Path information for reading file from local machine  
PathToAtmWtAgt = "C:/Users/drewm/Documents/GitHub/code-stat700/AtmWtAgt.dat"  
  
#Assigning data from file to data frame  
AtmWtAgt.df <- read.delim(PathToAtmWtAgt,header=TRUE,skip= 57,sep = "",as.is=TRUE)  
  
#Displaying data in dataframe  
AtmWtAgt.df

## X1 X2  
## 1 107.8682 107.8681  
## 2 107.8681 107.8681  
## 3 107.8682 107.8682  
## 4 107.8682 107.8681  
## 5 107.8681 107.8682  
## 6 107.8682 107.8681  
## 7 107.8682 107.8682  
## 8 107.8681 107.8681  
## 9 107.8682 107.8681  
## 10 107.8682 107.8681  
## 11 107.8682 107.8682  
## 12 107.8681 107.8681  
## 13 107.8681 107.8681  
## 14 107.8682 107.8681  
## 15 107.8682 107.8681  
## 16 107.8682 107.8682  
## 17 107.8681 107.8681  
## 18 107.8682 107.8682  
## 19 107.8682 107.8681  
## 20 107.8681 107.8681  
## 21 107.8681 107.8681  
## 22 107.8681 107.8681  
## 23 107.8682 107.8681  
## 24 107.8681 107.8681

## Part c

Calculate mean, sd and sample size for the two columns in this data; printing the results. You should store the values in variables. Use function(s) from Homework 3 to answer these two questions:

1. Is the difference between the two columns a small, medium or large effect size?
2. Is the difference between the two columns statistically significant?

Do this by printing function call(s) and results.

#Cohen D function  
d\_12 <- function(m\_1,m\_2,s\_1,s\_2) {  
 d\_12\_var <- (abs(m\_1-m\_2))/(sqrt((s\_1^2 + s\_2^2)/2))  
 return(d\_12\_var)  
}  
  
#Fisher LSD funtion  
fisher.lsd <- function(s\_i, n\_i, s\_j, n\_j, alpha=0.05) {  
 s2 <- ((n\_i-1)\*s\_i^2 + (n\_j-1)\*s\_j^2) / ((n\_i-1)+(n\_j-1))  
 critical.t <- qt(1 - alpha/2,n\_i+n\_j-2)  
 return(critical.t\*sqrt(s2\*(1/n\_i + 1/n\_j)))  
}  
  
# Function for chi^2 calcuations  
  
chisq.ci <- function(x,alpha=0.05) {  
p\_lower <- alpha/2  
p\_upper <- (1-alpha)/2  
df\_lower <- 2\*x  
df\_upper <- 2\*(x+1)  
local.chisqlower <- qchisq(p\_lower, df\_lower, ncp = 0, lower.tail = TRUE, log.p = FALSE)/2  
local.chisqupper <- qchisq(p\_upper, df\_upper, ncp = 0, lower.tail = TRUE, log.p = FALSE)  
return(list(chisqlower = local.chisqlower,chisqupper = local.chisqupper))  
}  
  
# pwr.t.test(n = , d = , sig.level =0.05 , power =NULL , type = c("two.sample", "one.sample", "paired"))  
  
#Mean, SD, and Sample Size for column 1  
AtmWtAgt1.mean <- mean(AtmWtAgt.df$X1,na.rm = TRUE)  
m\_1<-AtmWtAgt1.mean  
m\_1

## [1] 107.8682

AtmWtAgt1.SD <- sd(AtmWtAgt.df$X1)  
s\_1<-AtmWtAgt1.SD  
s\_1

## [1] 1.306311e-05

AtmWtAgt1.SampleSize <- chisq.ci(18,alpha = 0.05)  
AtmWtAgt1.SampleSize

## $chisqlower  
## [1] 10.66794  
##   
## $chisqupper  
## [1] 36.79701

#Mean, SD and Sample Size for Column 2  
AtmWtAgt2.mean <- mean(AtmWtAgt.df$X2,na.rm = TRUE)  
m\_2 <- AtmWtAgt2.mean  
m\_2

## [1] 107.8681

AtmWtAgt2.SD <- sd(AtmWtAgt.df$X2)  
s\_2<-AtmWtAgt2.SD  
s\_2

## [1] 1.690168e-05

AtmWtAgt2.SampleSize <- chisq.ci(18,alpha = 0.05)  
AtmWtAgt2.SampleSize

## $chisqlower  
## [1] 10.66794  
##   
## $chisqupper  
## [1] 36.79701

d\_12(m\_1,m\_2,s\_1,s\_2)

## [1] 1.152777

fisher.lsd(s\_1, sample.size , s\_2, sample.size , alpha=0.05)

## [1] 1.023224e-05

#Q1 Answer: small  
#Q2 Answer: No