**BIOS545R Introduction to R Programming**

**2017 Spring Semester Final Exam**

**03/01/2017 3:00 – 4:50 PM**

**INSTRUCTIONS:**

Answer all three questions. All work and code must be your own. Save all code into a single text file called YOURNAME\_FINAL.R where YOURNAME should be replaced by your Last and First name. No email, text, chat (electronic or person-to-person) is allowed during the Final. You may not download packages to solve these problems. You may not Google for code. Before leaving the room please email your file to [dvandom@emory.edu](mailto:dvandom@emory.edu) and [wsp@emory.edu](mailto:wsp@emory.edu). Total is 100 points. Partial credit will be given.

**QUESTION #1) (40 points)**

In this question you will write functions to compute the variance, covariance and correlation of an input vector.

1. Do not use the built in **“var”** (variance function), **“sd”**, or **“cov”** (covariance) functions though you can use the **"mean"** function.
2. Try to vectorize your computations. Using loops is allowed but to receive full credit you should use vectorization where possible.

**a) Variance (10 points)**

In a [sample](http://stattrek.com/Help/Glossary.aspx?Target=Population), the **variance** is the average squared deviation from the sample mean, as defined by the following formula:

where x is a vector of length "n" and is the mean of the sample vector. Write a function called **sampvar** that implements this formula. As an example:

set.seed(123)

x <- rnorm(30,10)

sampvar(x)

[1] 0.962

**b) Covariance (15 points)**

The **covariance** of two variables, (*x* and *y*),in a data sample measures how the two are linearly related. A positive covariance would indicate a positive linear relationship between the variables, and a negative covariance would indicate the opposite. The **sample covariance** is defined in terms of the sample means as:

and represent the mean of x and y, respectively, and are vectors of length "n". Write a function called "**sampcov**" in R that, given vectors x and y, implements the above formula. Thus, it will return the covariance of two vectors x and y. As an example using the following vectors:

set.seed(123)

x <- rnorm(1000)

set.seed(456)

y <- rnorm(1000)

sampcov(x,y)

[1] 0.0134

**c) Pearson's Correlation Coefficient (15 points)**

We can compute the **Pearson's correlation coefficient** by using the formula below where x and y are vectors of length "n". and are the standard deviations of x and y respectively. Note that the standard deviation is the square root of the sample variance.

where and

Write a function called "**mypearson**" that implements the formula for ***r***. Take advantage of the functions you wrote in sections a and b to simplify your work. As an example:

set.seed(123)

x <- rnorm(1000)

set.seed(456)

y <- rnorm(1000)

mypearson(x,y)

[1] 0.0137

**QUESTION #2 (20 points)**

Read the following .csv file in

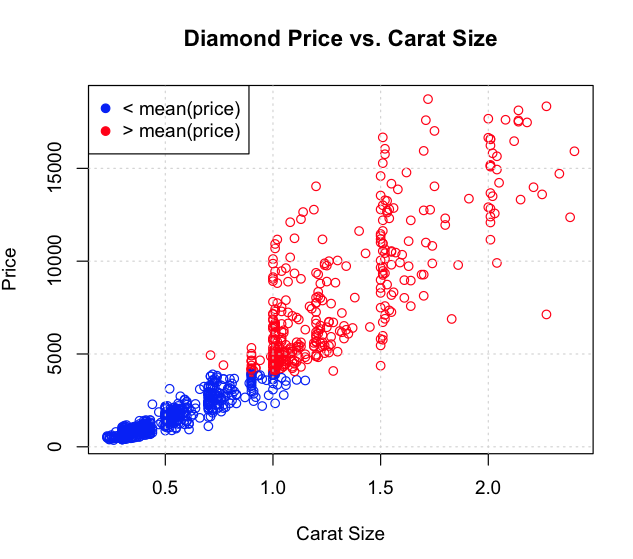
url = "http://steviep42.bitbucket.org/bios560rs2014/DATA.DIR/my.diamonds.csv"

myd = read.csv(url)

Create a function that presents a plot of price (y axis) vs. carat size (x axis) such that the points lying below the average price have the color blue and the points lying above the average price are in red. Use base graphics to create this function.

myplotter(mydf=myd)

To get full credit the figure generated from the function must have similar point type, colors, labels and titles as shown in the example.



**QUESTION #3) (40 points)**

Write a function called **window** that, given a numeric vector **xvec** and a numeric value **interval,** will compute a moving average for a specified interval of elements **(**e.g. 5**)** starting with the first element and so on. (See the example below for two specific demonstrations). You may **no**t use the **cumsum** or **filter** functions. You can do this easily with a for-loop structure. The function should return a vector containing the means for each "window".

window <- function(xvec, interval=5) {

# xvec: a numeric vector

# interval: a numeric value representing the size of the interval

# over which to take the mean loop

# Get the length of xvec

# Use a for loop to get the average of each "interval"

# (e.g. 5) number of elements starting with the first element

return(retvec)

}

Here are two specific examples: **nums** is a seven-element vector and the value of **interval** is 5. Get the average of every 5 elements starting with the first element. So begin with elements 1-5, then elements 2-6, then elements 3-7. We stop after that because there are only 4 remaining elements, which is less than the **interval** value of 5. So the value of **interval** has to be at least 1 but not more than the length of **nums**

set.seed(123)

(nums <- round(runif(7,1,30),0))

[1] 9 24 13 27 28 2 16

myw(nums,5)

[1] 20.2 18.8 17.2

9 24 13 27 28 2 16 # mean of nums[1:5] is 20.2

9 24 13 27 28 2 16 # mean of nums[2:6] is 18.8

9 24 13 27 28 2 16 # mean of nums[3:7] is 17.2

# Example 2 – Use a window size of 6

set.seed(123)

(nums <- round(runif(7,1,30),0))

[1] 9 24 13 27 28 2 16

myw(nums,6)

[1] 17.16667 18.33333

9 24 13 27 28 2 16 # mean of nums[1:5] is 17.16667

9 24 13 27 28 2 16 # mean of nums[2:6] is 18.33333