CFerrari_Assignment7

Charley Ferrari

Tuesday, March 10, 2015

Problem Set 1

Please write a function to compute the expected value and standard deviation of an array of values. Compare your results with that of R's mean and std functions. Please document your work in an R-Markdown file and ensure that you have good comments to help the reader follow your work.

```
vec <- c(2,4,4,4,5,5,7,9)

expectedValue <- function(vec){
  return(sum(vec)/length(vec))
}

mean(vec) == expectedValue(vec)
```

```
## [1] TRUE
```

```
standardDeviation <- function(vec){
  return(sqrt(sum((vec-expectedValue(vec))^2)/length(vec)))
}

standardDeviation(vec) == sd(vec)
```

```
## [1] FALSE
```

It looks like standard deviation isn't matching, but, if you look at the notes for the sd function:

Like var this uses denominator n - 1.

Lets test that out to see if using n - 1 as a denominator changes things:

```
standardDeviation <- function(vec){
  return(sqrt(sum((vec-expectedValue(vec))^2)/(length(vec)-1)))
}

sd(vec) == standardDeviation(vec)
```

```
## [1] TRUE
```

Now, consider that instead of being able to neatly fit the values in memory in an array, you have an infinite stream of numbers coming by. How would you estimate the mean and standard deviation of such a stream? Your function should be able to return the current estimate of the mean and standard deviation at any time it is asked. Your program should maintain these current estimates and return them back at any invocation of these functions. (Hint: You can maintain a rolling estimate of the mean and standard deviation and allow these to slowly change over time as you see more and more new values).

I decided to build a function that takes in two arguments: a list of stats and a vector. The vector is meant to be the updated flow of information, while the stats list is the cumulative stats received so far.

The list consists of a mean, standard deviation (sd), the number of observations (n), and the cumulative sum of the squares of all the observations (sumxsquared). The vector can be of any length. The variable sumxsquared is crucial to calculating the revised standard deviations.

If no statsList is entered (meaning you're at the beginning of the stream), the function assigns it to an empty list, and simply begins the calculation of the stats.

```
vec1 <- c(80,2,74,65,99,40,29,46,91)
vec2 <- c(5,7,18,93,66,59)

movingStats <- function(statsList = list(), vec){

  if(length(statsList) != 0){

    nnew <- statsList$n + length(vec)

    meannew <- (statsList$n*statsList$mean + sum(vec))/nnew

    sumxold <- statsList$n*statsList$mean
    sumxnew <- sumxold + sum(vec)

    sumxsquarednew <- statsList$sumxsquared + sum(vec^2)

    nsdsquarednew <- nnew*meannew^2 - 2*meannew*sumxnew + sumxsquarednew

    sdnew <- sqrt(nsdsquarednew/nnew)

  }else{

    nnew <- length(vec)

    meannew <- sum(vec) / nnew

    sumxnew <- sum(vec)

    sumxsquarednew <- sum(vec^2)

    nsdsquarednew <- nnew*meannew^2 - 2*meannew*sumxnew + sumxsquarednew
    sdnew <- sqrt(nsdsquarednew/nnew)
  }

  return(list(mean = meannew, sd = sdnew, n = nnew, sumxsquared = sumxsquarednew))

}

statsList1 <- movingStats(vec = vec1)

movingStats(statsList = statsList1, vec = vec2)
```

```
## $mean
## [1] 51.6
##
```

```
## $sd
## [1] 32.34151
##
## $n
## [1] 15
##
## $sumxsquared
## [1] 55628
```

Problem Set 2

For the auto-mpg data set that we looked at in Assignment 5, please compute the correlation and covariance matrices of the 5 variables in there.

```
setwd("E:/Downloads/Courses/CUNY/SPS/Git/IS 605 Fundamentals of Computational Mathematics/Assignment 7")

autodata <- scan("auto-mpg.data")

autodata <- t(matrix(autodata, nrow = 5))

means <- colMeans(autodata)

autodataCov <- matrix(ncol = ncol(autodata), nrow = nrow(autodata))

for(i in 1:ncol(autodata)){
  autodataCov[,i] <- autodata[,i] - means[i]
}

meansCov <- colMeans(autodataCov)

covMat <- matrix(nrow = 5, ncol = 5)

for(i in 1:nrow(covMat)){
  for(j in 1:ncol(covMat)){
    covMat[i,j] <- mean(autodataCov[,i] * autodataCov[,j])
  }
}

corrMat <- matrix(nrow = 5, ncol = 5)

for(i in 1:nrow(corrMat)){
  for(j in 1:ncol(corrMat)){
    corrMat[i,j] <- covMat[i,j] /
      (standardDeviation(autodata[,i])*standardDeviation(autodata[,j]))
  }
}

covMat
```

```
##             [,1]        [,2]        [,3]        [,4]         [,5]
## [1,] 10922.4329  3604.81427  82717.5463 -156.593939  -655.907693
## [2,]  3604.8143  1477.78988  28193.5141  -73.000266  -233.261349
```

```
## [3,] 82717.5463 28193.51406 719644.1868 -974.323377 -5503.365600
## [4,]  -156.5939   -73.00027   -974.3234    7.591915     9.092261
## [5,]  -655.9077  -233.26135  -5503.3656    9.092261    60.762738
```

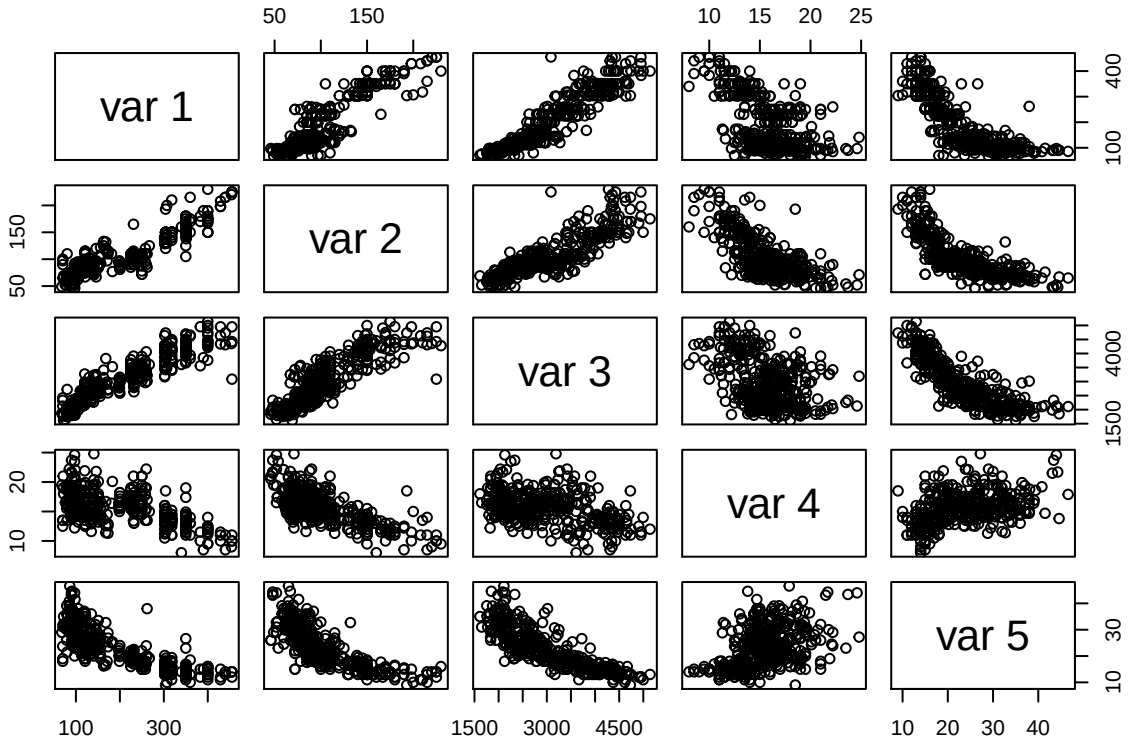
```
corrMat
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,]  0.9974490  0.8949681  0.9306143 -0.5424133 -0.8030731
## [2,]  0.8949681  0.9974490  0.8623323 -0.6874374 -0.7764410
## [3,]  0.9306143  0.8623323  0.9974490 -0.4157758 -0.8301211
## [4,] -0.5424133 -0.6874374 -0.4157758  0.9974490  0.4222486
## [5,] -0.8030731 -0.7764410 -0.8301211  0.4222486  0.9974490
```

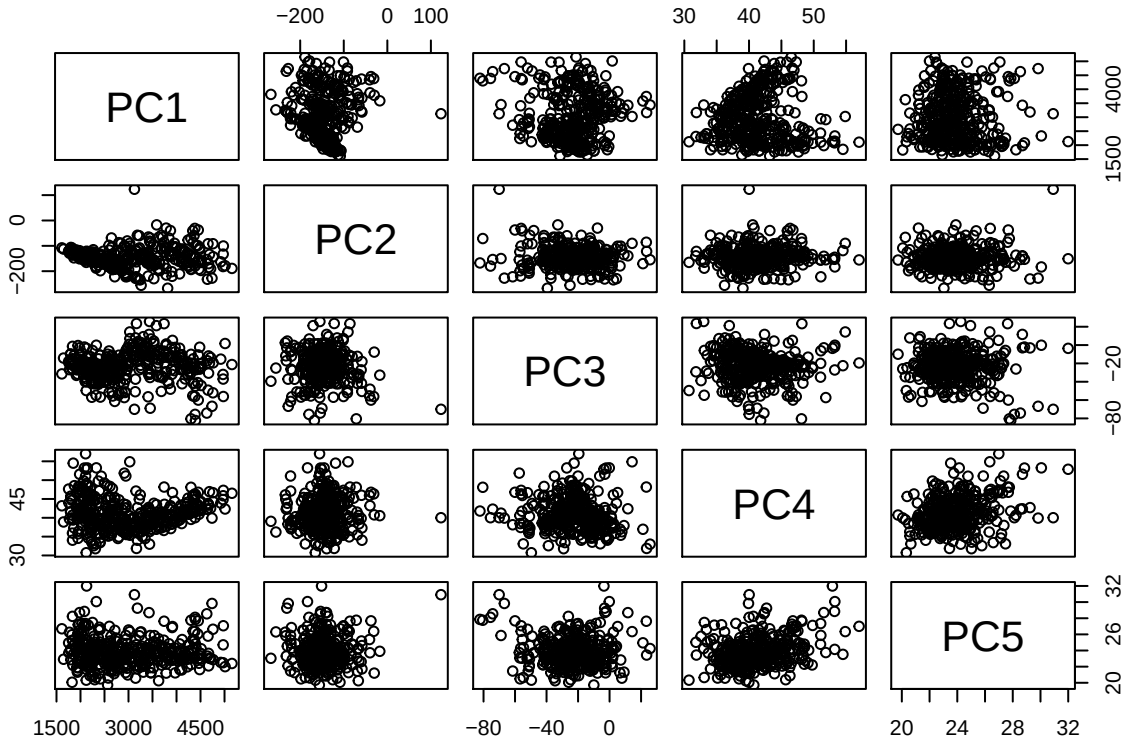
For bonus credit, please also perform Principal Components Analysis using prcomp command in R. This command has two options: centering and scaling. Make sure both are set to true. This option normalizes the random variables in order to make them comparable with each other. Please plot the output of your PCA and write a short paragraph on your observations. Please submit one R-markdown document containing the answers for both the problem sets.

Looking at the pairs graphs, it looks like scale should be set to FALSE in order to get truly uncorrelated values for autodata. Repeating the method outlined in the notes, this is what matches the eigenvector matrix calculated for

```
pca <- prcomp(covMat, center = TRUE, scale. = FALSE)

eigencov <- eigen(covMat)

proj <- autodata %*% pca$rotation

pairs(autodata)
```

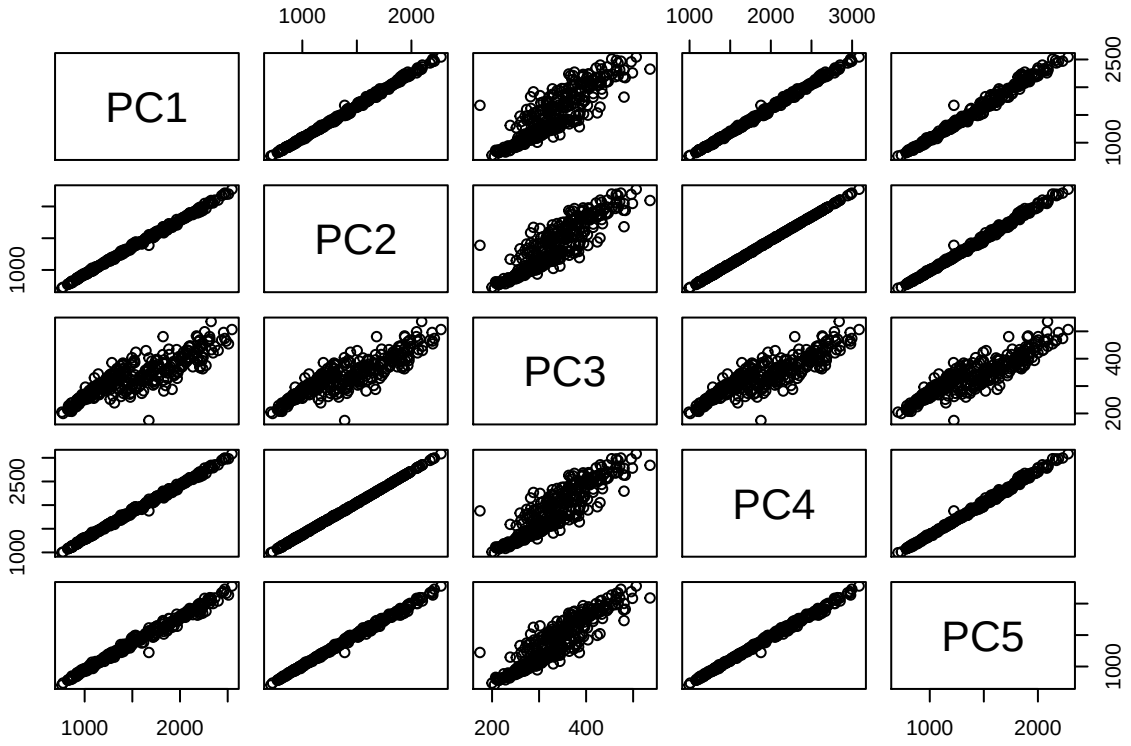

var 1
var 2
var 3
var 4
var 5
50
150
10
15
20
25
400
100
50
150
4000
1500
10
20
10
30
100
300
1500
3000
4500
10
20
30
40

```
pairs(proj)
```



I set scale to FALSE so the PCA would match the eigenvector vector matrix created from the covariance matrix. For some reason this was the only way to get these two methods to match, otherwise not only would the matrices not match, but the data was not uncorrelated. See below for an example:

```
pcatrue <- prcomp(covMat, center = TRUE, scale. = TRUE)

pairs(autodata %*% pcatrue$rotation)
```

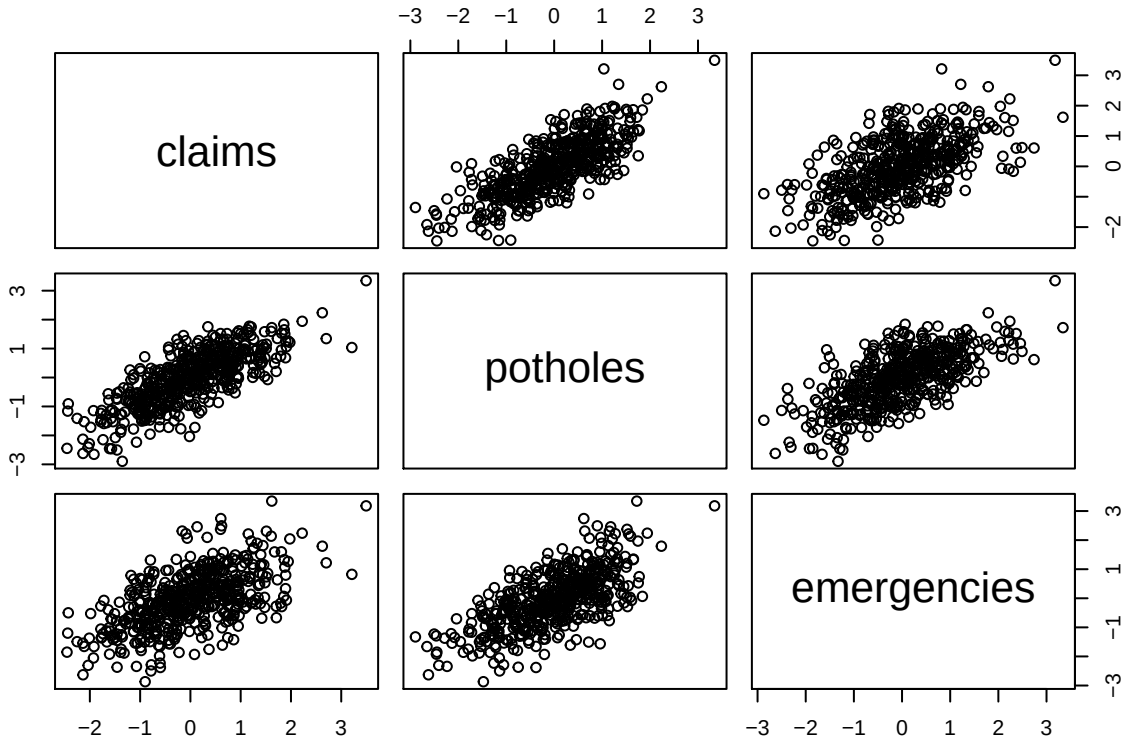


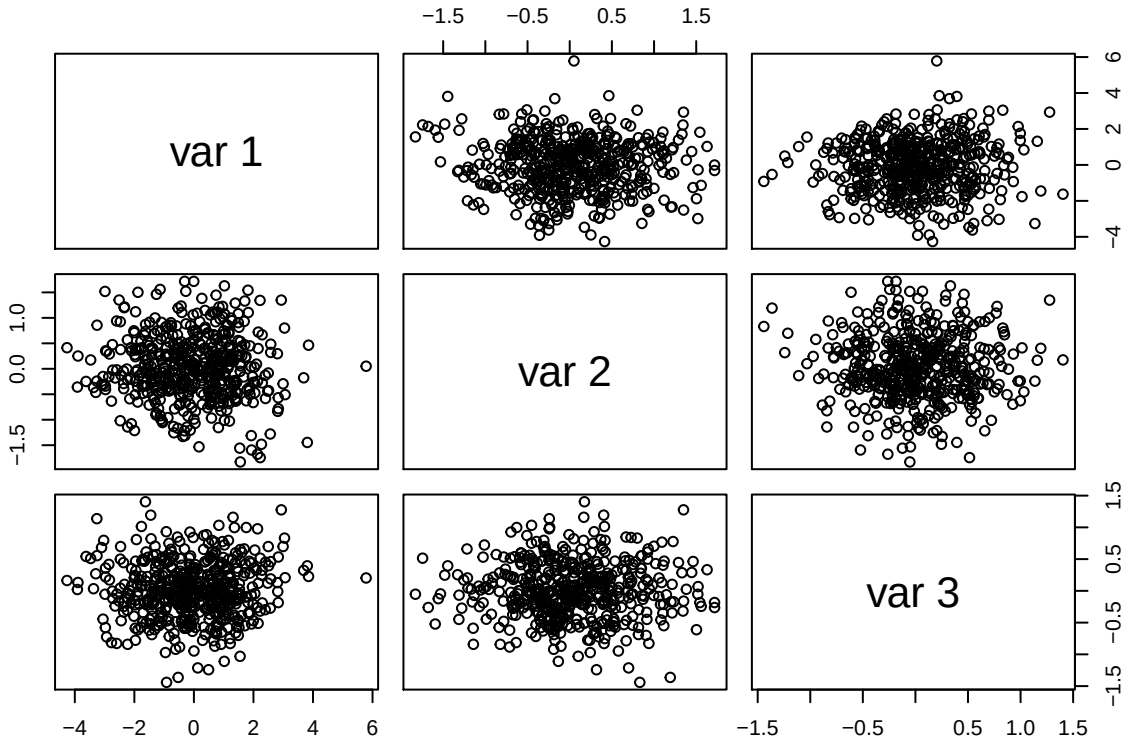
When scale is set to TRUE, the data appears even more correlated than before.

The block below is simply some of the practice I was running from the notes:

```
##                claims  potholes emergencies
## claims      1.0000000 0.7963905   0.5955484
## potholes    0.7963905 1.0000000   0.6964261
## emergencies 0.5955484 0.6964261   1.0000000
```

```
## $values
## [1] 2.3954661 0.4181581 0.1863758
##
## $vectors
##           [,1]       [,2]       [,3]
## [1,] 0.5785978  0.5653221  0.5879077
## [2,] 0.6038318  0.1876489 -0.7747097
## [3,] 0.5482807 -0.8032427  0.2327862
```


claims
potholes
emergencies


var 1
var 2
var 3