

C++ Data Exploration

Output:

C++

Microsoft Visual Studio Debug Console

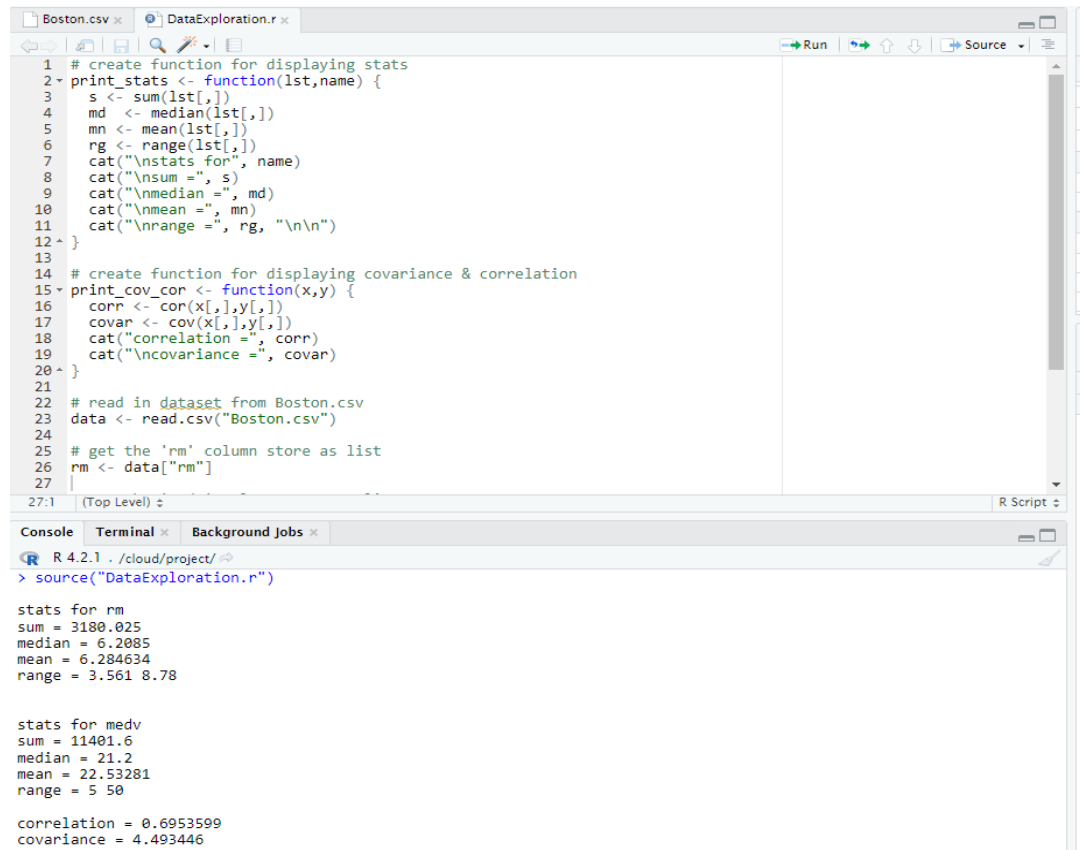
```
stats for rm
sum = 3180.03
mean = 6.28463
median = 6.2085
range = [3.561,8.78]
```

```
stats for medv
sum = 11401.6
mean = 22.5328
median = 21.2
range = [5,50]
```

covariance & correlation for both

```
covariance = 4.49345
correlation = 0.696737
```

R



```
Boston.csv x DataExploration.r x
1 # create function for displaying stats
2 print_stats <- function(lst,name) {
3   s <- sum(lst[,])
4   md <- median(lst[,])
5   mn <- mean(lst[,])
6   rg <- range(lst[,])
7   cat("\nstats for", name)
8   cat("\nsum =", s)
9   cat("\nmedian =", md)
10  cat("\nmean =", mn)
11  cat("\nrange =", rg, "\n\n")
12 }
13
14 # create function for displaying covariance & correlation
15 print_cov_cor <- function(x,y) {
16   corr <- cor(x[,],y[,])
17   covar <- cov(x[,],y[,])
18   cat("correlation =", corr)
19   cat("\ncovariance =", covar)
20 }
21
22 # read in dataset from Boston.csv
23 data <- read.csv("Boston.csv")
24
25 # get the 'rm' column store as list
26 rm <- data["rm"]
27
27:1 (Top Level)
R Script

Console Terminal Background Jobs
R 4.2.1 . /cloud/project/
> source("DataExploration.r")

stats for rm
sum = 3180.025
median = 6.2085
mean = 6.284634
range = 3.561 8.78

stats for medv
sum = 11401.6
median = 21.2
mean = 22.53281
range = 5 50

correlation = 0.6953599
covariance = 4.493446
\
```

Experience in C++ vs R

Doing this project in C++ was enlightening. More or less, all of the features explored within this exercise are built into R and readily available. I found myself looking back at the statistical definition of these functions at times. Even turning to other tools' documentation such as Excel or Python and comparing my output in C++ to that in R and as well as in those tools.

But in the end, the numbers I got in C++ were pretty close if not the same as what was got in R. One possible reason for the difference is that in C++, I kept casting from one type to another through various functions. Even on the initial read from a string to a float or double, the precision of the number will vary. Not to mention when computing things such as multiplication or subtraction even between integers and floating point numbers, the precision changes.

One issue I ran into in C++ was when I was implementing the median and range functions. When I first tried to compute the mean, I forgot to sort the input vector. While I did attain the middle value for both cases depending on the length, since it was out of order relative to how R was calculating it, the answer was wrong. It took me longer than I had hoped to catch it. In addition, I was worried in C++ about the mutability of vectors. While they were not passed by reference directly, I was applying the built-in sort function to get proper results for median and range, but this was done in-place. To refrain from having the ordering later come back to bite me in further calculations, I created a separate copy function to create a new vector object with all of the elements of the former inside. In doing so, I saw my calculations align more with what R had got in the end.

Descriptive Statistics: Mean, Median, Range

In descriptive statistics, the mean is the sum of all of the numerical data elements in a sequence divided by the number of elements there are. The median is the middle-value of an ordered sequence. The range is a pairing between the distinct minimum and maximum values within a dataset.

During exploratory data analysis (EDA), prior to machine learning, the goal is to get a feel for what a dataset contains and to understand it better. These descriptive statistics are very useful for EDA. For example, many times in the real world data is very messy and values could be missing. Preprocessing this data is a hard problem, there are many different techniques one could apply. But one of those techniques is to replace the missing values with the mean or the median. Generally speaking, the mean is an *average* observation - relative to the rest of the observations that are not empty - and the median is a *central* observation - that the other observations are approaching from either side. In addition, the mean is used as a base point to determine the variance of the data. The range can be helpful in this aspect as well, knowing the minimum and maximum values of a particular dataset, helps to form an idea of how the data is distributed overall.

Covariance and Correlation

To understand both of these, it's important to know about variance versus standard deviation. The variance describes the average relative distance between an observation and the mean for a given dataset. Whereas the standard deviation describes the spread of a given dataset, that is, how far apart the values are. Variance is computed as the average of the squared differences between an observation's value and the mean value for a column. Standard deviation on the other hand, is the square root of the variance.

Covariance works off of a similar concept to variance. It allows us to compare two features to each other to see how they relate *directionally* across their observations. For each feature, the difference between an observed value and the feature's mean is computed. The product of these two differences - one for each column - is made as well, and this is done relative to the ordering of the observations in the two datasets. These products are summed together and the average of their overall sum is then computed relative to one of their number of observations. In a nutshell, covariance tells us the *direction* that one feature's data is moving in relative to another's. For example, suppose the covariance between X and Y is a positive value, this signifies a positive relationship - meaning as X values increase, Y values tend to increase as well.

This is similar to Correlation in that, this tool also measures the directional flow of two features relative to each other. However, correlation also gives us insight into *how strong* a relationship is between two or more features. It is computed as the covariance of two features, divided by the product of the standard deviations of both features. An important concept with correlation is that of the correlation coefficient. This coefficient is a scalar value in the range $[-1, 1]$, and if the value is close to either end, then the relationship between the features is said to be strong. If the value tends towards 0, the relationship is weak.

Both covariance and correlation can be helpful in machine learning. For reasons discussed prior, in the case of EDA, both of these statistics allow us to have some quantifiable measurement of relationships within our data. For example, in the case of correlation if two features result in a correlation coefficient that tends towards 0, then we know they have a weak relationship. If this continues to happen as you test more features and you're able to see which features are not related to any of the others, or are very weakly so, then you could reduce the dimensionality of the data to include those features which are more correlated. However, it does introduce an issue of overfitting. Only keeping those features which are strongly correlated to one another, could have its own problems. Nonetheless, being aware of what features those are is better than not being aware of them.