

## C++ Data Exploration

### Introduction

This program is a C++ implementation of statistical analysis of data stored in a CSV file named "Boston.csv". It computes the sum, mean, median, range, covariance, and correlation of two sets of data, "rm" and "medv".

### Libraries

- `iostream`
- `fstream`
- `algorithm`
- `vector`
- `string`
- `math.h`
- `numeric`

### Main Function

- Opens the "Boston.csv" file
- Reads the data from the file and stores it in two vectors "rm" and "medv"
- Computes the number of observations in the data
- Calls the **print\_stats** function to display the statistical information for the two sets of data
- Computes the covariance and correlation of the two sets of data
- Closes the file and terminates the program.

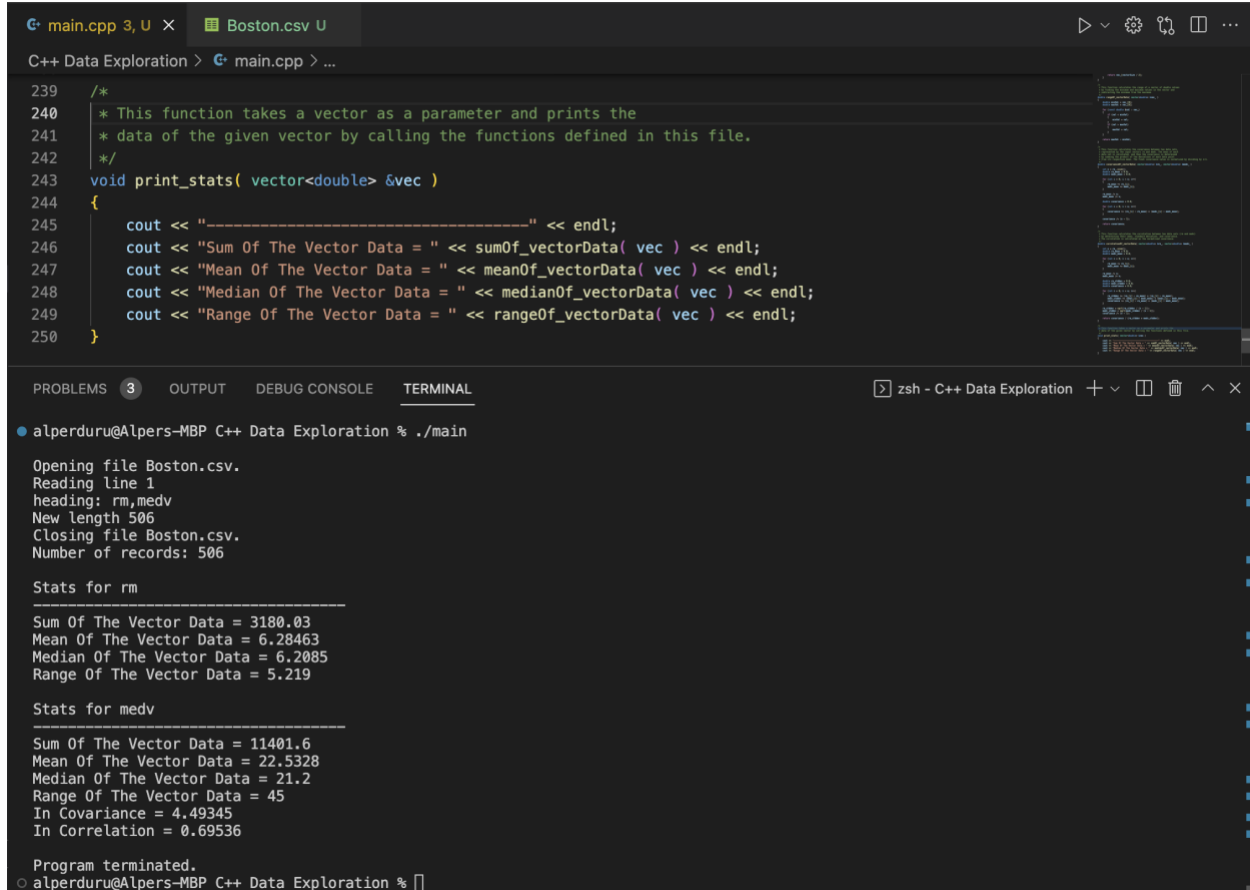
### Functions

1. **sumOf\_vectorData**- computes the sum of a given vector of data
2. **meanOf\_vectorData**- computes the mean of a given vector of data
3. **medianOf\_vectorData** - computes the median of a given vector of data
4. **rangeOf\_vectorData** - computes the range of a given vector of data
5. **covarianceOf\_vectorData** - computes the covariance of two given vectors of data
6. **correlationOf\_vectorData** - computes the correlation of two given vectors of data
7. **print\_stats** - displays the sum, mean, median, and range of a given vector of data

### Conclusion

The program reads data from a CSV file, performs statistical analysis on the data, and displays the results on the console. The program can be used to perform statistical analysis on different sets of data by changing the input file and the data contained within it.

### Output of the written code:



The screenshot shows a C++ IDE with two tabs: 'main.cpp 3, U' and 'Boston.csv U'. The main.cpp file contains a function `print_stats` that takes a `vector<double>` and prints statistical data. The terminal output shows the execution of `./main`, which reads data from 'Boston.csv' and prints statistics for 'rm' and 'medv'.

```
239  /*
240  * This function takes a vector as a parameter and prints the
241  * data of the given vector by calling the functions defined in this file.
242  */
243  void print_stats( vector<double> &vec )
244  {
245      cout << "-----" << endl;
246      cout << "Sum Of The Vector Data = " << sumOf_vectorData( vec ) << endl;
247      cout << "Mean Of The Vector Data = " << meanOf_vectorData( vec ) << endl;
248      cout << "Median Of The Vector Data = " << medianOf_vectorData( vec ) << endl;
249      cout << "Range Of The Vector Data = " << rangeOf_vectorData( vec ) << endl;
250  }
```

```
alperduru@Alpers-MBP C++ Data Exploration % ./main
Opening file Boston.csv.
Reading line 1
heading: rm,medv
New length 506
Closing file Boston.csv.
Number of records: 506

Stats for rm
-----
Sum Of The Vector Data = 3180.03
Mean Of The Vector Data = 6.28463
Median Of The Vector Data = 6.2085
Range Of The Vector Data = 5.219

Stats for medv
-----
Sum Of The Vector Data = 11401.6
Mean Of The Vector Data = 22.5328
Median Of The Vector Data = 21.2
Range Of The Vector Data = 45
In Covariance = 4.49345
In Correlation = 0.69536

Program terminated.
alperduru@Alpers-MBP C++ Data Exploration %
```

### My experience using built-in functions in R versus coding my own functions in C++:

When it comes to comparing the use of built-in functions in R and coding your own functions in C++, there are pros and cons to both approaches. Using built-in functions in R can be more convenient and quicker as it provides a wide range of pre-written and tested functions for various statistical and data manipulation tasks. This can save time and effort for the user. However, coding your own functions in C++ allows for greater control and customization of the process, which can be beneficial for specific tasks or needs. Additionally, custom functions in C++ can result in faster execution times, making it ideal for performance-critical applications. Ultimately, the decision between using built-in functions in R or coding your own functions in C++ will depend on the task at hand, the level of control and performance desired, and the user's comfort level with programming.

### Descriptive statistical measures mean, median, and range, and how these values might be useful in data exploration prior to machine learning:

The mean, median, and range are key descriptive statistical measures used for understanding the distribution of data. The mean is the average value of a dataset, calculated by adding up all the

values and dividing by the number of values. The median is the middle value of a dataset, found by arranging the values in order and selecting the value that is in the middle. The range is the difference between the largest and smallest value in a dataset.

These measures can be extremely useful in data exploration prior to machine learning as they provide quick insights into the overall distribution of the data. The mean and median can give a sense of the central tendency of the data, while the range provides information about the spread of the data. This information can be used to identify any outliers or skewness in the data, which can impact the results of machine learning algorithms. For example, if the range of the data is very large, it may indicate the presence of outliers that could affect the performance of the algorithm. Understanding the descriptive statistics of the data can help to inform data preparation and feature selection, leading to better performance of machine learning models.

*Covariance and correlation statistics, and what information they give about two attributes. How might this information be useful in machine learning?*

Covariance and correlation are important statistical measures that describe the relationship between two attributes. Covariance measures the degree to which two variables change together, and can be calculated as the expected value of the product of the deviations of each variable from their respective means. Correlation, on the other hand, measures the strength of the linear relationship between two variables and is expressed as a value between -1 and 1. A value of 1 indicates a perfect positive linear relationship, a value of -1 indicates a perfect negative linear relationship, and a value of 0 indicates no linear relationship.

In machine learning, these statistics can be used to gain insights into the relationship between different attributes and to help inform feature selection. For example, if two variables are highly correlated, it may be redundant to include both variables in a machine learning model as they are providing similar information. Additionally, covariance and correlation can be used to help identify non-linear relationships between variables, which can be important in building accurate machine learning models. By understanding the relationships between different attributes, practitioners can make informed decisions about which features to include in their models and how to engineer new features that may improve performance.