Data Exploration

The Premise

> *"In class, we covered how to do data exploration with statistical functions in R. In this assignment, you recreate that functionality in C++ code. This will prepare us to write algorithms in C++ in future assignments"*

For me this is both a review of C++, but also a review of what correlation is.

Notes

- I deliberate whether returning range as the min and max, or as the difference between the two. I eventually chose just returning a min and max.
- I can't get relative links to work at the moment, hope it's fine that it is linking to the file hosted on the main site
-

Conclusion

The Code

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <algorithm>
#include <iterator>
#include <cmath>

using namespace std;

// TODO: Convert double vectors to taking in (explicitly) any numeric value
// Reference: - Iterators: https://www.geeksforgeeks.org/iterators-c-stl/
//            - What get's passed into sort: https://cplusplus.com/reference/iterator/RandomAccessIterator/

class Explore
{
public:
        // Calculate the sum of the vector
        double sum_vector(vector<double> vect)
        {
                double sum = 0;
                for (int i = 0; i < vect.size(); i++)
                {
                        sum += vect[i];
                }
                return sum;
        }

        // Calculate the mean of a vector
        double mean_vector(vector<double> vect)
        {
                double mean = sum_vector(vect) / vect.size();
                return mean;
        }

        // Calculate the median of a vector
        double median_vector(vector<double> vect)
        {
                double median;
                // Use an iterator because it is probably better -internet
                vector<double>::iterator it;
                // Find the center if it is even or odd
                sort(vect.begin(), vect.end());
                if (vect.size() % 2 == 0) // If there is an even number of elements
                {
                        it = vect.begin() + vect.size() / 2 - 1;
                        median = (*it + *(it + 1)) / 2;
                }
                else // if there is an odd number of elements
```

```
		{
			it = vect.begin() + vect.size() / 2;
			median = *it;
		}
		return median;
	}

	// Calculate the range of a vector
	vector<double> range_vector(vector<double> vect)
	{
		vector<double> range = {max_vector(vect), min_vector(vect)};
		return range;
	}

	// Calculate the max of a vector (Just for range)
	double max_vector(vector<double> vect)
	{
		double max;
		vector<double>::iterator it;
		sort(vect.begin(), vect.end());
		it = vect.end() - 1;
		max = *it;
		return max;
	}

	// Calculate the min of a vector (Just for range) just with a loop
	double min_vector(vector<double> vect)
	{
		double min;
		vector<double>::iterator it;
		sort(vect.begin(), vect.end());
		it = vect.begin();
		min = *it;
		return min;
	}

	// Calculate the covariance of two vectors
	// Cov(x,y) = E((x-x_mean)(y-y_mean)))/n-1
	double covar_vector(vector<double> x, vector<double> y)
	{
		double sum = 0;
		double mean_x = mean_vector(x);
		double mean_y = mean_vector(y);
		for (int i = 0; i < x.size(); i++)
		{
			float x_i_diff = x[i] - mean_x;
			float y_i_diff = y[i] - mean_y;
			float y_times_x_diff = x_i_diff * y_i_diff;
			// cout << x_i_diff << " * " << y_i_diff << " = " << y_times_x_diff << endl;
			sum = sum + y_times_x_diff;
		}
		return sum / (x.size() - 1);
	}

	// Calculate the correlation of two vectors
	// Cor(x,y) = Cov(x,y)/(standard_devation(x)*standard_devation(y))
	// Using the hint from the assignment:
	// "sigma of a vector can be calculated as the square root of variance(v,v)"
	double cor_vector(vector<double> x, vector<double> y)
	{
		double covar = covar_vector(x, y);
		double sigma_x = sqrt(covar_vector(x, x));
		double sigma_y = sqrt(covar_vector(y, y));
		return covar / (sigma_x * sigma_y);
	}

	// Run suite of statistcal functions on a vector
	void print_stats(vector<double> vect)
	{
		cout << "Sum:    " << sum_vector(vect) << endl;
		cout << "Mean:   " << mean_vector(vect) << endl;
		cout << "Median: " << median_vector(vect) << endl;
		vector<double> range = range_vector(vect);
		cout << "Range:  " << range[1] << ", " << range[0] << endl;
	}
};

int main(int argc, char **argv)
```

```
{
        ifstream inFS;
        string line;
        string rm_in, medv_in;
        const int MAX_LEN = 1000;
        vector<double> rm(MAX_LEN), medv(MAX_LEN);

        cout << "Opening file Boston.csv." << endl;

        inFS.open("Boston.csv");
        if (!inFS.is_open())
        {
                cout << "Error opening file Boston.csv." << endl;
                return 1;
        }

        cout << "Reading line 1 of Boston.csv." << endl;
        getline(inFS, line);

        // echo heading
        cout << "Headings: " << line << endl;

        // read data
        int numObservations = 0;
        while (inFS.good())
        {
                getline(inFS, rm_in, ',');
                getline(inFS, medv_in, '\n');
                rm.at(numObservations) = stof(rm_in);
                medv.at(numObservations) = stof(medv_in);

                numObservations++;
        }

        rm.resize(numObservations);
        medv.resize(numObservations);

        cout << "New Length: " << rm.size() << endl;

        cout << "Closing file Boston.csv." << endl;
        inFS.close(); // Done

        cout << "Number of records: " << numObservations << endl;

        // Create an Explore object to use stats functions
        Explore explore;

        cout << "\nStats for rm" << endl;
        explore.print_stats(rm);

        cout << "\nStats for medv" << endl;
        explore.print_stats(medv);

        cout << "\n Covariance = " << explore.covar_vector(rm, medv) << endl;

        cout << "\n Correlation = " << explore.cor_vector(rm, medv) << endl;

        cout << "\nProgram terminated." << endl;
}
```

Returns

```
Opening file Boston.csv.
Reading line 1 of Boston.csv.
Headings: rm,medv
New Length: 506
Closing file Boston.csv.
Number of records: 506

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 = 4.49345

 Correlation = 0.69536

Program terminated.
```

Built into R or C++

I believe that it was clearly easier to use R functions vs going through and making these functions in C++. This indicates the value of use using R into the future for machine learning. A more straight forward way to analyze data will allow us to understand our data models.

I will note that someone fluent in C++ would do better than I did, considering I was just stumbling around for a bit trying to remember how import a library for a second there.

Statistical Value

What statistical measures did I evaluate:

- ***Mean***: A mean is an average of the data set, and represents the typical or most likely value from a dataset. Knowing what values in a dataset tend to be is important in understanding what general trend of all the data in your dataset is. You can compare values to this to find outliers and such.
- ***Median***: The center value of the data as it is in sorted order. This tells you a central tendency independent of skewed data or large outliers
- ***Range***: is the minimum and maximum values the values of the dataset might take. It is useful to understand how a dataset is bounded, both to see how far outliers might be from the center of a dataset, and just to get an idea for what the data looks like in scale.

Whenever we are organizing data for a machine learning algorithm, we must understand our data ourselves to have a hope of predicting a trend given our data. Looking at values like mean or median tell us easy to understand generalizations about data so that we may get a general understanding of the meaning of our data without having to analyze every data point. Given more and more powerful generalization tools, or methods of analyzation, we can grow more and more confident in the understanding of our data.

If we can see a general trend using our descriptive statistical measurements, then we can assure our model will be able to eventually get the specific trend data we hope to predict from that dataset.

Covariance and Correlation

Given two attributes that may or may not be related, we may find the *covariance* and *correlation* between those two bits of data. The covariance tells how one attributes data might relate to another. If x's covariance to y is a high positive number, we know that as x goes up y goes down, and vice versa. Correlation is just a version of that number, scaled down to a range of (-1, 1) in order to make the factor much more uniform

The values are different from those above, considering they are not just measurements of an attribute of data but an extrapolation from that data about relationships or patterns. This is very useful when working in ml, as our end goal is figuring out how data tends to relate to certain results. Using how data correlates then directly supports our end goal of predicting outcomes, or just understanding complex relationships.