FORECASTING CASE COUNTS USING REGRESSION MODELS AND AN ANALYSIS ON COVID-19 DATASET

Venkateshwaran Thamilselvan
University of Luxembourg
113-6, Avenue du swing
L-4367, Belvaux, Luxembourg
+352 661364956
venkateshwaran.thamilselvan.001@student.uni.lu

ABSTRACT

As the COVID-19 pandemic tends to grow in number worldwide, we record huge data related to COVID-19. With the huge computing power which we have today, we can try to analyze the data collected either to find some useful facts or create some machine learning models with which we can predict the growth of pandemic in different parts of the world. In this use-case study, I have taken the John Hopkins datasets [1] to perform data analysis and to create machine learning model (mainly based on Linear regression and Random forest regression) to predict and forecast the growth of the pandemic in a particular locality or a country. Here we mainly predict and forecast the confirmed case count and the recovered case count.

PROBLEM STATEMENT

In this case study, we are going to do a detailed analysis on the following intriguing concepts-

1. With this huge data in our hand we should be able to rank the countries based on -
   a. Current death count
   b. Current confirmed case count
   c. Current active case count
   d. Current recovered case count
   e. Case-Fatality-Ratio
   f. Incidence rate

   By performing the above analysis, we can understand different dynamics in different countries. For e.g. There may be countries who ranks higher in confirmed cases but lower fatality rate and death count. We can identify the nature of the pandemics in different countries.
2. Finding the correlation between the confirmed case counts, recovered case counts and death counts worldwide and for specific counties. With this analysis, we can understand how the different records correlate with each other.
3. Creating machine learning models to predict the confirmed case counts. These types of machine learning models will help us understand how pandemic is spreading in different regions around the world and help us forecast the confirmed case counts with some accuracy for some future date. This task includes-
   a. Creating random forest regression model with the whole data and predict the confirmed cases for a particular location (using latitude and longitude) at a particular date. We should measure the accuracy by some evaluation metrics such as RMSE or MSE.
   b. Creating the random forest regression model for the US specific data (using the US specific records in John Hopkins university data) and to predict the confirmed cases on a particular location in a state of the country. This makes sense because the US specific data has many features recorded such as hospitalization rate, testing rate etc.
4. Visualization of the trends-
   a. Plot the worldwide trend of confirmed, recovered and death counts as a line chart.
   b. Plot the trends of confirmed, recovered and death counts as a line chart separately for each country.
5. Forecasting the confirmed case count and recovery case count using the linear regression model for Luxembourg for the next 10 days and plotting it as a line chart.
6. Using the Mobility dataset [2] to find the correlation between the confirmed case counts and the following-
   a. Retail and recreation percent change from baseline
   b. Grocery and pharmacy percent change from baseline
   c. Parks percent change from baseline
   d. Transit stations percent change from baseline
   e. Workplaces percent change from baseline
   f. Residential percent change from baseline

APPROACH

Since we are handling huge data such as Mobility data (136MB – Single file), COVID-19 data provided by John Hopkins University (Totally 612MB), we should use some distributed systems to process and analyze the

data. Here in this case study, I have used the HPC cluster provided by University of Luxembourg [3] for distributed analysis.

Project Details-

i. In this project, I have used Scala as a frontend language for Spark.
ii. ML is used for creating regression models.
iii. MLlib is used for getting the statistics of the data.
iv. Scalaplot library [4] is used to plot the line charts.
v. Spark SQL is used to perform queries to filter our data.
vi. Spark SQL-Dataframe and RDD is used to store our data in the distributed manner.

Now we will see how to solve the above problem statement section-wise.

1. We have taken the csse_covid_19_data from JHU for this part as it contains all the required data for this section. In csse_covid_19_daily_reports, the date format is different for different period of time. Moreover, the date formats within a single file is same. So, we read each file separately using a loop and parse date with different date formats used and store it as timestamp. This while csv files are read as dataframes and the dataframe corresponding to each date are then merged with our original dataframe. This final dataframe contains all the data with proper timestamp. The unfilled entries in Confirmed, death and recovery are actually zeros, so we update nulls with 0 for those columns. The active is computed with Eq. 1, where act is active case count, conf is confirmed case count and deaths are the death count.

$$act = conf - recov - deaths \quad (1)$$

Since the all the data are accumulated, meaning that the count for today is computed by adding the previous count with the new cases today, the latest date has the total counts for all regions. So, to rank the countries, we take the maximum date in our data and select the rows with this maximum date. After that we group the data by countries and aggregate it to find the sum of confirmed, recovered, death and active case counts.

Since the Case-Fatality-Ratio (CFR) is computed with the Eq. 2, we cannot just sum up all the ratios. Since we have both the death and confirmed case counts for a country, we compute CFR with the Eq. 2.

$$CFR = deaths/\text{confirmed} \quad (2)$$

$$IR = cases \ / \ 100{,}000 \ persons \quad (3)$$

The Incidence Rate (IR) takes the population of the locality into account. As this is normalized, to compute IR for a country, we just take the average of IR of all the state/province of that country.

2. To find the correlation of a country, we first filter the data of that country from the whole data. Then, we group them by date and take aggregate to find the sum of confirmed, recovered, death and active case counts. In this aggregated count, we have the count of whole country for the corresponding date. Then we sort the date in descending order and compute the spearman correlation coefficient. To compute the correlation coefficient worldwide, we do exactly the above steps, except that at first, we do not select the data for a particular country.

3. a. To perform random forest regression, we first remove the string data types such as Combined_Key, Admin2 and some data which have lots of null rows such as Case-Fatality_Ratio and Incidence_Rate. Then, we group the data frame by date, province/state, country, latitude and longitude. After that we aggregate to find the sum of confirmed, recovered, deaths and active cases. Finally, we will get the aggregated data for all localities for all dates.

Some entries in our data do not have the latitude and longitude. There is a csv file named UID_ISO_FIPS_LookUp_Table.csv which has the latitude and longitude for each locality. We lookup that csv file with the province/state and country as key and take its latitude and longitude and put it in our data. This drastically reduces the null occurrence and helps us a lot for our regression.

We also have the population of each locality in the lookup table. So, we perform left join on those two data based on the latitude and longitude as the key. By doing that we get the population column for each locality in our dataframe. This may drastically increase the performance of our regression as the population plays a key role in the spread of pandemic.

Finally, we select the data with dates less than 2020-08-15 as train data and greater than that date as the test data. We also create param grid for cross validation. We use the the combination in table 1 for param grid creation. Then we compute the root mean square error for our test data.

Hyperparameter	values
Impurity	"variance"
Max Depth	10, 20
Max Bins	10, 100, 300

Table 1

b. Similarly, we perform the regression for US specific data. Before that, for preprocessing US data, we round the latitude and longitude to 5 decimals so as they match with the lookup table, and also update null occurrences of recovery with 0 and null occurrences of active with the newly computed value from Eq. 1

4. **NOTE:** For plotting the line charts, we should have opened the spark-shell as below (with the Scalaplot library's package) -

   spark-shell --packages = "org.sameersingh.scalaplot:scalaplot:0.0.4"

   For plotting the worldwide data, we group our dataframe by date and aggregate it to find the sum of Confirmed, recovered and deaths. Since we already have such a dataframe, we reuse it for this purpose. Thus, we will get the unique entries for each date.

   Since only the double data format is supported in the x axis of the line chart, we sort the dataframe in ascending order of date. Then the first date is encoded as 1 and incremented by 1 for the following dates. By this way we encode the date to doubles for the ease of plotting in Scalaplot library.

   To plot chart separately for each country, we first get the list of distinct country names from our dataframe. Then we iterate through each country name in the list and then select the data for that country and plot as above.

5. For forecasting, we use the timeseries data provided by JHU. The timeseries for confirmed and recovered case counts are given in separate files and each date is in separate columns. For our convenience we transpose the table such that the date columns come to row with its title as Date and other columns such as Province/State, Country/Region, Latitude and Longitude lies as it is. By doing so, we can easily lookup the count for a location on a specific date. Then we parse the date column to timestamp data type. Since the string datatypes are not supported, we drop the Country and province/state columns. We can then lookup the country and province/state with the help of the latitude and longitudes.
   For forecasting, we use the linear regression model. For cross validation we use the hyperparameter values from table 2.

Hyperparameter	values
Max Iteration	10, 20,30
Tolerance	0.00000001, 0.0000001, 0.000001
Optimization solver	"l-bfgs", "normal", "auto"
Elastic net param	0.5, 0.6, 0.8
Regularization param	0.1, 0.3, 0.5

Table 2

   The elastic net param, tolerance and regularization params should be chosen properly so that the overfitting of the data does not occur. The values in the above table are the perfect values for creating combination for training our model. We have filtered and taken the data only for Luxembourg for training our model. We put all of the data as the training data and we read the lux_test.csv. After creating the model for confirmed and recovered cases separately, we predict the confirmed and recovered cases for our test data. Finally we plot the actual confirmed and recovered case counts along with the forecasted confirmed and recovered case counts.

6. For mobility data analysis, we read the Global_Mobility_Report.csv file provided by google and then preprocess it by converting the date string to timestamp format. After that we group the dataframe by country and date so that we can analyze based on the country. Then we select only the Luxembourg's data and sort it based on the date. Then we take the required columns as rdd to compute the spearman correlation.

EVALUATION AND RESULT

1. Ranking

a. Ranking based on Death count

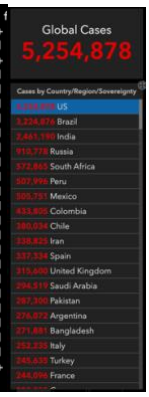
```
scala> deathList.sort(desc("Sum_Deaths")).show(20, false)
+--------------+----------+----------+----------+
|Country_Region|Min_Deaths|Max_Deaths|Sum_Deaths|
+--------------+----------+----------+----------+
|US            |0         |23602     |166023    |
|Brazil        |482       |25869     |104201    |
|Mexico        |228       |9598      |54666     |
|India         |0         |18650     |47033     |
|United Kingdom|0         |42072     |46791     |
|Italy         |23        |16833     |35225     |
|France        |0         |30247     |30375     |
|Spain         |0         |8464      |28579     |
|Peru          |0         |10183     |21713     |
|Iran          |18988     |18988     |18988     |
|Russia        |0         |4611      |15231     |
|Colombia      |0         |3998      |13837     |
|South Africa  |11010     |11010     |11010     |
|Chile         |1         |7992      |10205     |
|Belgium       |9900      |9900      |9900      |
|Germany       |0         |2627      |9213      |
|Canada        |0         |5709      |9052      |
|Netherlands   |0         |1542      |6182      |
|Pakistan      |58        |2297      |6139      |
|Ecuador       |5984      |5984      |5984      |
+--------------+----------+----------+----------+
only showing top 20 rows
```

The actual deaths which is obtained from the website [5] source provided by John Hopkins University is on the right (as of 15th Aug 2020). Our output is shown in the left image.

- By comparing these two data, we can see that we have computed correctly (Sum_Deaths is similar to the website's data) and so we can proceed to study the dataset further.
- The Min_Deaths is the minimum number of deaths marked by at least one of the province/states of the country.
- The Max_Deaths is the maximum number of deaths marked by at least one of the province/states of the country.

b. Ranking based on confirmed cases

```
scala> confirmedList.sort(desc("Sum_Confirmed")).show(20,
+--------------+-------------+-------------+-------------+
|Country_Region|Min_Confirmed|Max_Confirmed|Sum_Confirmed|
+--------------+-------------+-------------+-------------+
|US            |0            |229167       |5248702      |
|Brazil        |22242        |674455       |3224876      |
|India         |0            |560126       |2461190      |
|Russia        |162          |250303       |905762       |
|South Africa  |572865       |572865       |572865       |
|Mexico        |2524         |83683        |505751       |
|Peru          |0            |250708       |498555       |
|Colombia      |6            |149944       |433805       |
|Chile         |64           |264222       |380034       |
|Spain         |0            |90890        |337334       |
|Iran          |336324       |336324       |336324       |
|United Kingdom|0            |270971       |315600       |
|Saudi Arabia  |294519       |294519       |294519       |
|Pakistan      |2164         |124929       |286674       |
|Argentina     |276072       |276072       |276072       |
|Bangladesh    |269115       |269115       |269115       |
|Italy         |478          |97128        |252235       |
|Turkey        |245635       |245635       |245635       |
|France        |4            |230778       |244096       |
|Germany       |959          |53123        |222281       |
+--------------+-------------+-------------+-------------+
only showing top 20 rows
```



- The actual confirmed cases which obtained from the website [5] source provided by John Hopkins University is on right (as of 15th Aug 2020). The Left image is our output.
- By comparing our output and their website output, we can conclude that our result is correct. But it may not be precisely equal, and the websites data will be mostly slightly larger than our data. This is because website data is updated more frequently than the git repository.

c. Ranking based on Active cases

```
scala> activeList.sort(desc("Sum_Active")).show(20, false) //Top 20 rank list
+--------------+----------+----------+----------+
|Country_Region|Min_Active|Max_Active|Sum_Active|
+--------------+----------+----------+----------+
|US            |-1774648  |209312    |3306948   |
|India         |0         |150105    |661595    |
|Brazil        |-255      |190373    |598313    |
|United Kingdom|0         |228899    |267330    |
|Russia        |10        |52208     |175475    |
|Colombia      |4         |64991     |169166    |
|Spain         |0         |58978     |158353    |
|Peru          |-341938   |240525    |134904    |
|France        |1         |128785    |130092    |
|South Africa  |123978    |123978    |123978    |
|Bangladesh    |110687    |110687    |110687    |
|Argentina     |78276     |78276     |78276     |
|Sweden        |284       |21548     |78076     |
|Philippines   |74713     |74713     |74713     |
|Bolivia       |58855     |58855     |58855     |
|Netherlands   |1         |14698     |55966     |
|Belgium       |48362     |48362     |48362     |
```

Comparing active and confirmed case ranking, we can see that India is gaining in the active count compared to brazil and thus in near future the total deaths in India is likely to increase than Brazil. But we cannot say anything with certainty, because the death rate is linked to-

- The percentage of old aged people affected
- percentage of COVID-19 patients with other background illnesses
- the health facility availability and affordability.

d. Ranking based on Recovered cases

```
scala> recoveredList.sort(desc("Sum_Recovered")).show(20, false) //Top 20 rank list
+--------------+-------------+-------------+-------------+
|Country_Region|Min_Recovered|Max_Recovered|Sum_Recovered|
+--------------+-------------+-------------+-------------+
|Brazil        |9345         |457758       |2616981      |
|India         |0            |401442       |1808936      |
|US            |0            |1796326      |1796326      |
|Russia        |98           |194823       |721473       |
|South Africa  |461734       |461734       |461734       |
|Mexico        |1681         |68437        |410479       |
|Chile         |62           |251316       |355037       |
|Peru          |0            |354232       |354232       |
|Iran          |293811       |293811       |293811       |
|Pakistan      |1981         |118924       |265215       |
|Saudi Arabia  |262959       |262959       |262959       |
|Colombia      |2            |84153        |261293       |
|Turkey        |228980       |228980       |228980       |
|Italy         |390          |75012        |203326       |
|Germany       |0            |48400        |200440       |
|Argentina     |199005       |199005       |199005       |
|Bangladesh    |156623       |156623       |156623       |
|Spain         |0            |40736        |150376       |
|Iraq          |120129       |120129       |120129       |
|Qatar         |111258       |111258       |111258       |
+--------------+-------------+-------------+-------------+
only showing top 20 rows
```

e. Ranking based on CFR

```
scala> cfrList.sort(desc("CFR")).show(20, false) //Top 20 rank list
+--------------+-------------+----------+-------------------+
|Country_Region|Sum_Confirmed|Sum_Deaths|CFR                |
+--------------+-------------+----------+-------------------+
|Yemen         |1730         |494       |28.55491329479769  |
|MS Zaandam    |9            |2         |22.22222222222222  |
|United Kingdom|305572       |46278     |15.144712211851871 |
|Belgium       |69402        |9845      |14.185470159361401 |
|Italy         |247832       |35146     |14.181380935472417 |
|France        |225198       |30268     |13.440616701746908 |
|Hungary       |4526         |597       |13.190455148033584 |
|Netherlands   |55021        |6167      |11.208447683611713 |
|Mexico        |434193       |47472     |10.933386765793092 |
|Western Sahara|10           |1         |10.0               |
|Spain         |288522       |28445     |9.858866914827985  |
|Chad          |936          |75        |8.012820512820513  |
|Canada        |118510       |8986      |7.582482490929036  |
|Sweden        |80422        |5743      |7.141080798786402  |
|Ireland       |26109        |1763      |6.752460837259183  |
|Ecuador       |86232        |5736      |6.651822989145561  |
|Sudan         |11738        |752       |6.406542852274663  |
|Liberia       |1189         |75        |6.307821698906644  |
|Niger         |1136         |69        |6.073943661971831  |
|San Marino    |699          |42        |6.008583690987124  |
+--------------+-------------+----------+-------------------+
only showing top 20 rows
```

In above figure the aggregated CFR is calculated by summing up deaths and confirmed cases in that region and dividing the death by confirmed cases. It is then multiplied

with 100 to represent as percentage. There can be many reasons for a very large CFR. They are-

i. The percentage of people with underlying health conditions is larger.
ii. The preventive and precautionary measures are not properly undertaken.
iii. People of that country generally has bad hygiene.
iv. The health systems are not up to the par.
v. The testing is not properly done, therefore actual active cases in that region is very much higher than the reported value.

f. Ranking based on IR

```
scala> incidenceList.sort(desc("Avg_Incidence_Rate")).show(20, false) //Top 20 rank list
+--------------+------------------+
|Country_Region|Avg_Incidence_Rate|
+--------------+------------------+
|Qatar         |3975.342408696799 |
|Bahrain       |2706.421020896424 |
|Brazil        |2195.520780431633 7|
|San Marino    |2059.638163710295 4|
|Panama        |1840.237991938384 3|
|Kuwait        |1760.540706225385 |
|Oman          |1620.307906087429 3|
|Holy See      |1483.312731767614 4|
|Armenia       |1393.713760033801 |
|Chile         |1393.345096607095 2|
|Peru          |1374.528150499419 3|
|Andorra       |1280.010353976574 2|
|Luxembourg    |1182.952701062021 6|
|US            |1176.996252880352 1|
|Israel        |1052.273913323269 |
|Maldives      |1030.817216793514 7|
|South Africa  |976.4842217894208 |
|Singapore     |950.0297674854277 |
|Saudi Arabia  |849.954416152621  |
|Bolivia       |839.1138238412669 |
+--------------+------------------+
only showing top 20 rows
```

This considers the population of that country in account which is much reasonable.
Incidence rate is already there in the dataset, but it is there only for separate province/state. We can compute the incidence rate of a country by grouping and taking average of all its provinces/states. This way of computing is correct because the incidence rate for all region is in per 100,000 persons.

- The global incidence rate is around 335. This means that every 335 persons out of 100,000 in the world has COID-19. This count can be used to compare the various pandemics.
- In figure 8, we can see that Luxembourg has 3.5 times the global average incidence rate which is an alarming count.

2. Correlation of Confirmed-Recovered-Death

For Luxembourg:

```
scala> println(s"Confirmed-Recovered correlation is: $confirmed_recovered_correlation")
Confirmed-Recovered correlation is: 0.9993150450452986

scala> println(s"Confirmed-Death correlation is: $confirmed_death_correlation")
Confirmed-Death correlation is: 0.9867776276372286

scala> println(s"Death-Recovered correlation is: $death_recovered_correlation")
Death-Recovered correlation is: 0.9861071562346669
```

Worldwide:

```
scala> println(s"Confirmed-Recovered correlation is: $confirmed_recovered_correlation")
Confirmed-Recovered correlation is: 0.9988085537278034

scala> println(s"Confirmed-Death correlation is: $confirmed_death_correlation")
Confirmed-Death correlation is: 0.999923173126207

scala> println(s"Death-Recovered correlation is: $death_recovered_correlation")
Death-Recovered correlation is: 0.9986347486690708
```

- From the above results we can see that Confirmed case and Death count has the highest correlation.
- When we increase the sample count, the correlation increases. This means that all these data tend to correlate between each other.

3. Predicting confirmed cases

a. Worldwide data

```
scala> predictions.drop("featureVector").show(5)
+-----------+------------+-------------------+------+---------+---------+------+--------+-----+----------+------------------+
|   Latitude|   Longitude|        Last_Update|Deaths|Recovered|Confirmed|Active|     UID|code3|Population|        prediction|
+-----------+------------+-------------------+------+---------+---------+------+--------+-----+----------+------------------+
|    30.0572|     107.874|2020-08-16 04:27:42|     6|      577|      583|     0|   15603|  156|  31020000| 580.9532940788288|
|32.60154883|-85.35132246|2020-08-16 04:27:42|    47|        0|     2760|  2713|84001081|  840|    164542|2747.1719550319986|
|37.18049267|-89.32910838|2020-08-16 04:27:42|     0|        0|       39|    39|84017003|  840|      5761| 39.12968699892165 6|
|43.35671042|-92.31702714|2020-08-16 04:27:42|     0|        0|       56|    56|84019089|  840|      9158| 54.94668954763638|
|38.44188174|-84.32869741|2020-08-16 04:27:42|     0|        0|      123|   123|84021097|  840|     18886|124.11722826510236|
+-----------+------------+-------------------+------+---------+---------+------+--------+-----+----------+------------------+
only showing top 5 rows


scala> val rmse = evaluator.evaluate(predictions)
rmse: Double = 7399.476193416496

scala> println(s"Root Mean Squared Error (RMSE) on test data = $rmse")
Root Mean Squared Error (RMSE) on test data = 7399.476193416496
```

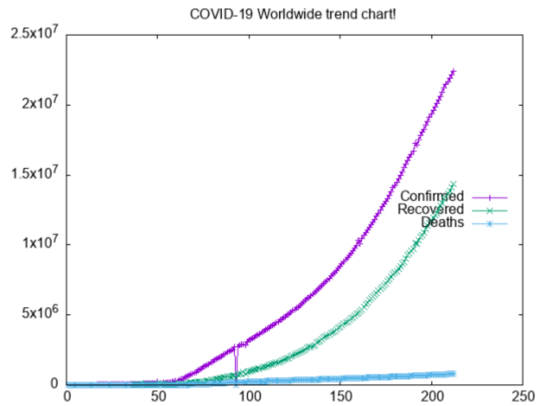
b. US specific data

```
scala> predictions.drop("featureVector").show(5)
+-----------+------------+-------------------+------+---------+---------+--------+-----+----------+------------------+
|   Latitude|   Longitude|        Last_Update|Deaths|Recovered|Confirmed|     UID|code3|Population|        prediction|
+-----------+------------+-------------------+------+---------+---------+--------+-----+----------+------------------+
|40.12570935|-122.2370171|2020-08-18 04:27:56|     1|        0|      318|84006103|  840|     65084|305.14701635617746|
|32.71391813|-83.98061238|2020-08-18 04:27:56|     0|        0|      120|84013079|  840|     12404|114.10500321600799|
|32.74976524|-81.60779441|2020-08-18 04:27:56|     9|        0|      240|84013251|  840|     13966|238.76949362065284|
|42.47104597| -91.3664492|2020-08-18 04:27:56|     1|        0|      139|84019055|  840|     17011| 128.4516432256428|
|37.19113093|-95.29849679|2020-08-18 04:27:56|     1|        0|      159|84020099|  840|     19618|146.76396296186698|
+-----------+------------+-------------------+------+---------+---------+--------+-----+----------+------------------+
only showing top 5 rows


scala> val rmse = evaluator.evaluate(predictions)
rmse: Double = 7780.5982667250855

scala> println(s"Root Mean Squared Error (RMSE) on test data = $rmse")
Root Mean Squared Error (RMSE) on test data = 7780.5982667250855
```

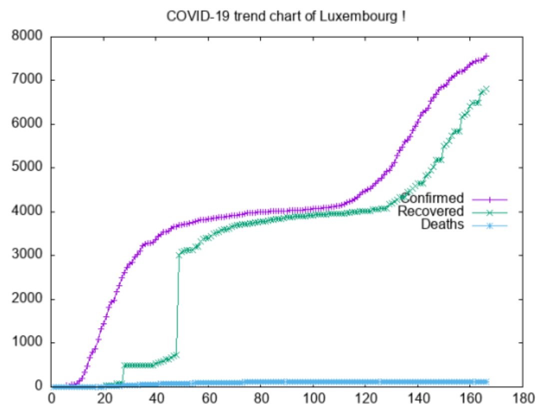
The above model predicts the confirmed cases for the past dates with some accuracy.

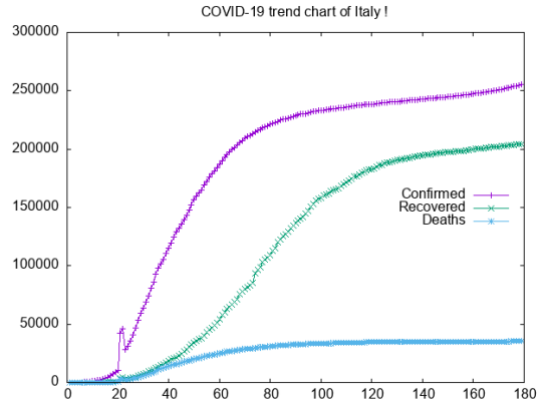
4. Trend charts

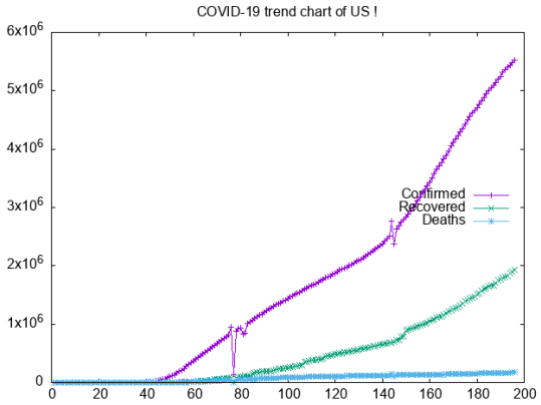
a. Worldwide trend chart

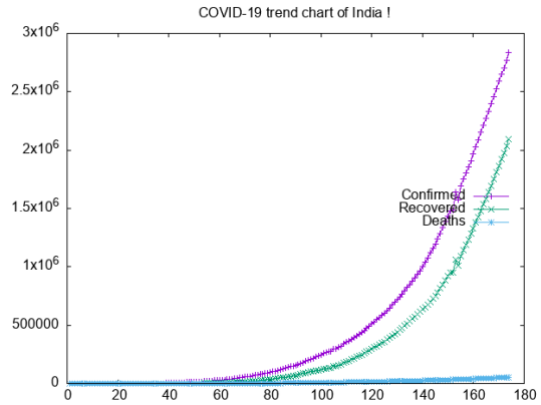


b. Country wise trend chart



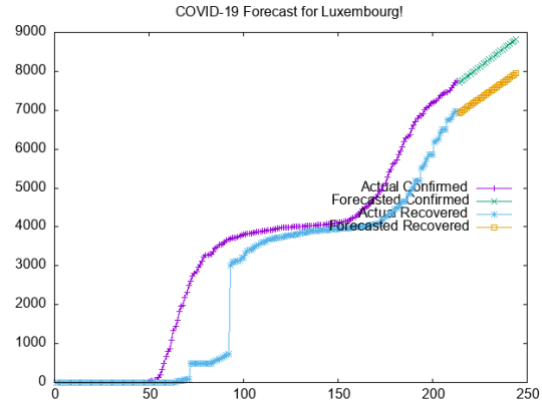






The trend chart for all the countries is in output/charts/countrywise.

5. Forecast for Luxembourg using linear regression



6. Correlation of Mobility and Confirmed

```
scala> println(s"Confirmed - Retail and Recreation correlation is: $confirmed_retail_correlation")
Confirmed - Retail and Recreation correlation is: 0.1289210374253674 2

scala> println(s"Confirmed - Grocery and Pharmacy correlation is: $confirmed_grocery_correlation")
Confirmed - Grocery and Pharmacy correlation is: 0.03291060989876557 6

scala> println(s"Confirmed - Park visit correlation is: $confirmed_parks_correlation")
Confirmed - Park visit correlation is: 0.7478884593099 2

scala> println(s"Confirmed - Transit station correlation is: $confirmed_transit_correlation")
Confirmed - Transit station correlation is: 0.1672224085827479

scala> println(s"Confirmed - Workplace visit correlation is: $confirmed_workplace_correlation")
Confirmed - Workplace visit correlation is: 0.12275871185003565

scala> println(s"Confirmed - Residential correlation is: $confirmed_residential_correlation")
Confirmed - Residential correlation is: -0.13104332252188106
```

Bibliography

[1] J. H. University, "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University," [Online]. Available: https://github.com/CSSEGISandData/COVID-19.

[2] Google, "COVID-19 Community Mobility Reports," [Online]. Available: https://www.google.com/covid19/mobility/ .

[3] U. o. Luxembourg, "High Performance Computing in Luxembourg," [Online]. Available: https://hpc.uni.lu/.

[4] Sameersingh. [Online]. Available: https://github.com/sameersingh/scalaplot .

[5] J. H. University, "John Hopkins university website for COVID-19 trends," [Online]. Available: https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6.