Homework 2 - Data Mining - Sapienza

Ivan Fardin 1747864

November 22^{th}, 2020

Contents

1 Problem 1

To download some products from `https://www.amazon.it/` I used the suggested **Requests** and **Beautiful Soup** packages. The first one performs a get request to `https://www.amazon.it/s?k=KEYWORD&page=X` where KEYWORD is the product we want to look for and X is the page number while the latter parses the retrieved page.

The *downloadProducts.py* program performs several requests using *computer* as keyword and an incremental page number for all available pages. Each page is then parsed to extract information of interest (product description, price, prime product or not, product URL and product rank) and these are written into the *products.tsv* file.

To avoid being blocked, the program performs requests using different user agents and adding a delay between different downloads of Amazon pages.
When a sufficient number of pages have been parsed, the user can close the application via Ctrl+C.

After that, the *invertedIndex.py* program preprocesses each product description stored in the *products.tsv* file to build and store an inverted index into the *invertedIndex.json* file.

The preprocessing part is implemented in the *PreprocessProducts* class in *preprocessProducts.py* and essentially tokenizes each product description in two phases: the first uses the *nltk word_tokenizer* and the second performs an additional custom tokenization.
Each resulting token is then normalized and filtered by a stopwords removal phase.
Finally, each token is stemmed and is stored in the inverted index map as key with value a map where each entry is the product description number (line in the *products.tsv* file) with value the number of occurrences in the product description.

In the end, the *queryProducts.py* application performs the query-processing part that consists of preprocessing the input query (like each product description above), finding the resulting tokens in the inverted index and for each token finding the document having the minimum number (line in the *products.tsv* file) and computing the cosine similarity between the query and tokens present in the document having the minimum number. The document having the maximum cosine similarity is then printed.

```
$python3 queryProducts.py
Enter your query: ram ssd

Processing the query...

The best related product is the following

Title: computer gaming Baieyu Mini pc Intel core i7-9750H processor,mini pc barebone,UHD Graphics 630+GeForce GTX 1650,64G RAM DDR4 +512G SSD+1TB HDD,HDMI+DP,Dual band WiFi,BT 4.2,Lan,4K HD

Price: 1.195,00 €

Prime product: False

Product url: /gp/slredirect/picassoRedirect.html/ref=pa_sp_atf_aps_sr_pg1_1?ie=UTF8&adId=A040632034J14GNI3P45F&url=%2Fcomputer-Baieyu-i7-9750H-processor-barebone%2Fdp%2FB08F9TXKXP%2Fref%3Dsr_1_1_sspa%3Fdchild%3D1%26keywords%3Dcomputer%26qid%3D1604773773%26sr%3D8-1-spons%26psc%3D1&qualifier=1604773773&id=7036191796464208&widgetName=sp_atf

Rank: 5,0
```

```
$python3 queryProducts.py
Enter your query: computer da gaming nvidia 16gb ram

Processing the query...

The best related product is the following

Title: Mini Gaming PC,Mini PC Windows 10 Pro,Intel Core i9 8950HK,NVIDIA GeForce GTX 1650 4GB, 16GB RAM DDR4, 512G NVME SSD, HDMI DP Type-C USB3.1 WIFI Bluetooth 5.0, Business Desktop Computer

Price: 1.170,00 €

Prime product: True

Product url: /Windows-GeForce-Bluetooth-Business-Computer/dp/B0896QC4T8/ref=sr_1_270?dchild=1&keywords=computer&qid=1604773834&sr=8-270

Rank: 4,2
```

```
$python3 queryProducts.py
Enter your query: notebook 512gb ssd i5

Processing the query...

The best related product is the following

Title: ASUS Zenbook 14 UX425JA-BM147T, Notebook in alluminio con Monitor 14" FHD Anti-Glare, Intel Core i5-1035G1, RAM 8GB LPDDR4X, 512GB SSD PCIE, Windows 10 Home, Argento

Price: 829,00 €

Prime product: True

Product url: /ASUS-UX425JA-BM147T-alluminio-Anti-Glare-i5-1035G1/dp/B086VPWHZ1/ref=sr_1_100?dchild=1&keywords=computer&qid=1604773791&sr=8-100

Rank: 4,5
```

```
$python3 queryProducts.py
Enter your query: ASUS Zenbook 14 UX425JA-BM147T, Notebook in alluminio con Monitor 14" FHD Anti-Glare, Intel Core i5-1035G1, RAM 8GB LPDDR4X, 512GB SSD PCIE, Windows 10 Home, Argento

Processing the query...

The best related product is the following

Title: ASUS Zenbook 14 UX425JA-BM147T, Notebook in alluminio con Monitor 14" FHD Anti-Glare, Intel Core i5-1035G1, RAM 8GB LPDDR4X, 512GB SSD PCIE, Windows 10 Home, Argento

Price: 829,00 €

Prime product: True

Product url: /ASUS-UX425JA-BM147T-alluminio-Anti-Glare-i5-1035G1/dp/B086VPWHZ1/ref=sr_1_100?dchild=1&keywords=computer&qid=1604773791&sr=8-100

Rank: 4,5
```

2 Problem 2

Starting from the previously downloaded products in **Problem 1**, I re-implemented the building of the inverted index and the query-processing part using the Spark framework.

The *invertedIndexSpark.py* program preprocesses each product description stored in the *products.tsv* file to build and store an inverted index into the *invertedIndexSpark.json* file.

This time, the preprocessing part is implemented in the *PreprocessProductsSpark* class in *preprocessProductsSpark.py* and simply preprocesses the product description as in **Problem 1** but returns a list of tuples (token, 1) instead of the inverted index.
This because the entire building of the inverted index is achieved using the map-reduce paradigm (each step is fully commented in the Python script).

In the end, the *queryProductsSpark.py* application performs the query-processing part that works similar to **Problem 1** but using the map-reduce paradigm (each step is fully commented in the Python script again).

```
Enter your query: ram ssd

Processing the query...

The best related product is the following

Title: computer gaming Baieyu Mini pc Intel core i7-9750H processor,mini pc barebone,UHD Graphics 630+GeForce GTX 1650,64G RAM DDR4 +512G SSD+1TB HDD,HDMI+DP,Dual band WiFi,BT 4.2,Lan,4K HD

Price: 1.195,00 €

Prime product: False

Product url: /gp/slredirect/picassoRedirect.html/ref=pa_sp_atf_aps_sr_pg1_1?ie=UTF8&adId=A040632034J14GNI3P45F&url=%2Fcomputer-Baieyu-i7-9750H-processor-barebone%2Fdp%2FB08F9TXKXP%2Fref%3Dsr_1_1_sspa%3Fdchild%3D1%26keywords%3Dcomputer%26qid%3D1604773773%26sr%3D8-1-spons%26psc%3D1&qualifier=1604773773&id=7036191796464208&widgetName=sp_atf

Rank: 5,0
```

```
Enter your query: computer da gaming nvidia 16gb ram

Processing the query...

The best related product is the following

Title: Mini Gaming PC,Mini PC Windows 10 Pro,Intel Core i9 8950HK,NVIDIA GeForce GTX 1650 4GB, 16GB RAM DDR4, 512G NVME SSD, HDMI DP Type-C USB3.1 WIFI Bluetooth 5.0, Business Desktop Computer

Price: 1.170,00 €

Prime product: True

Product url: /Windows-GeForce-Bluetooth-Business-Computer/dp/B0896QC4T8/ref=sr_1_270?dchild=1&keywords=computer&qid=1604773834&sr=8-270

Rank: 4,2
```

```
Enter your query: notebook 512gb ssd i5

Processing the query...

The best related product is the following

Title: ASUS Zenbook 14 UX425JA-BM147T, Notebook in alluminio con Monitor 14" FHD Anti-Glare, Intel Core i5-1035G1, RAM 8GB LPDDR4X, 512GB SSD PCIE, Windows 10 Home, Argento

Price: 829,00 €

Prime product: True

Product url: /ASUS-UX425JA-BM147T-alluminio-Anti-Glare-i5-1035G1/dp/B086VPWHZ1/ref=sr_1_100?dchild=1&keywords=computer&qid=1604773791&sr=8-100

Rank: 4,5
```

```
Enter your query: ASUS Zenbook 14 UX425JA-BM147T, Notebook in alluminio con Monitor 14" FHD Anti-Glare, Intel Core i5-1035G1, RAM 8GB LPDDR4X, 512GB SSD PCIE, Windows 10 Home, Argento

Processing the query...

The best related product is the following

Title: ASUS Zenbook 14 UX425JA-BM147T, Notebook in alluminio con Monitor 14" FHD Anti-Glare, Intel Core i5-1035G1, RAM 8GB LPDDR4X, 512GB SSD PCIE, Windows 10 Home, Argento

Price: 829,00 €

Prime product: True

Product url: /ASUS-UX425JA-BM147T-alluminio-Anti-Glare-i5-1035G1/dp/B086VPWHZ1/ref=sr_1_100?dchild=1&keywords=computer&qid=1604773791&sr=8-100

Rank: 4,5
```

3 Problem 3

To perform some forecasting using Covid-19 data, I used the **spark.ml** package as suggested.

Initially, I loaded the training data in a Spark DataFrame and selected a few states from it (all with the *County* and *Province_State* columns null).

Then, I started the **feature engineering** process:

- Transform features of the dataset by indexing the strings of the *Country_Region*, *Date* and *Target* columns and converting the *Population*, *TargetValue* and *Weight* columns to numbers;
- Extract features from the transformed dataset: *Country_Region*, *Population*, *Date* and *Target*.

After that, I began to **cross-validate** different regression algorithms (with label the *TargetValue* column) in order to find the best hyper-parameters among the ones I proposed. Clearly linear regression, isotonic regression and factorization machine regression were not suitable for the problem, but out of curiosity, I wanted to try.

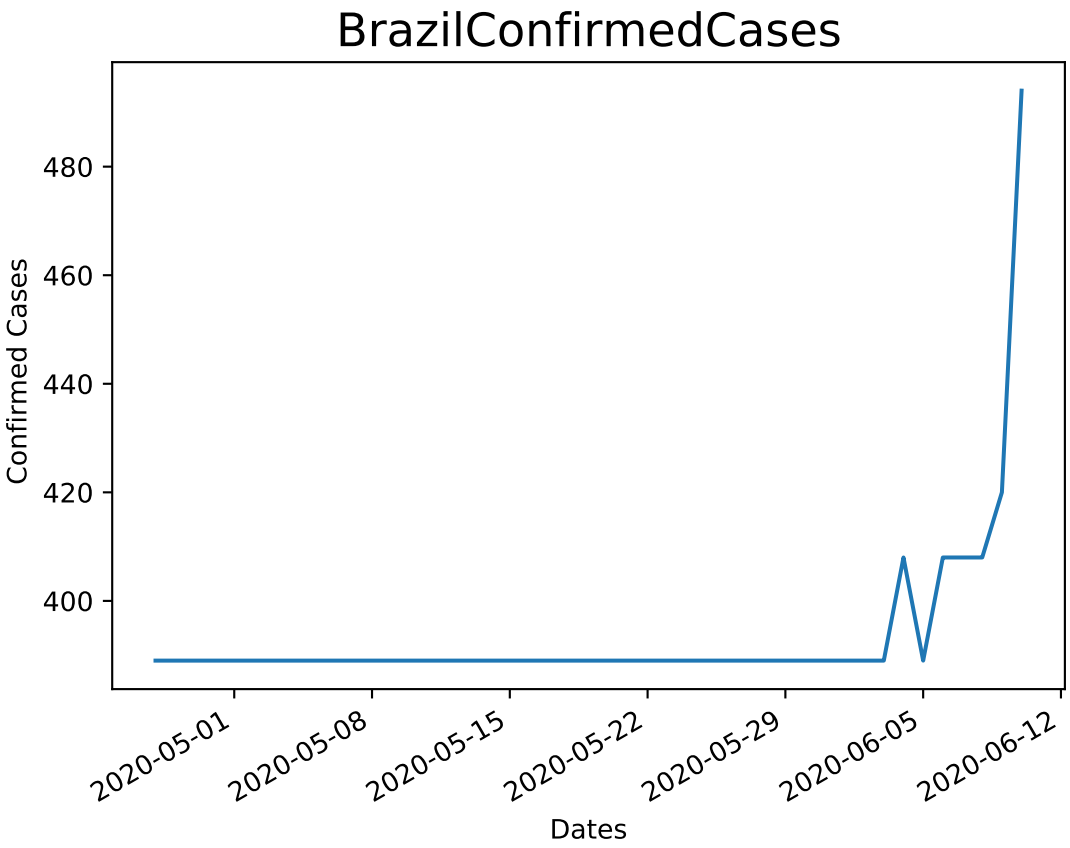
Once cross-validation was complete, I found the best model and used it to predict confirmed Covid-19 cases and fatalities using (during this phase) the same training set (after the cutting of nations and the feature engineering process) as test set to compute the **Root Mean Squared Error**.

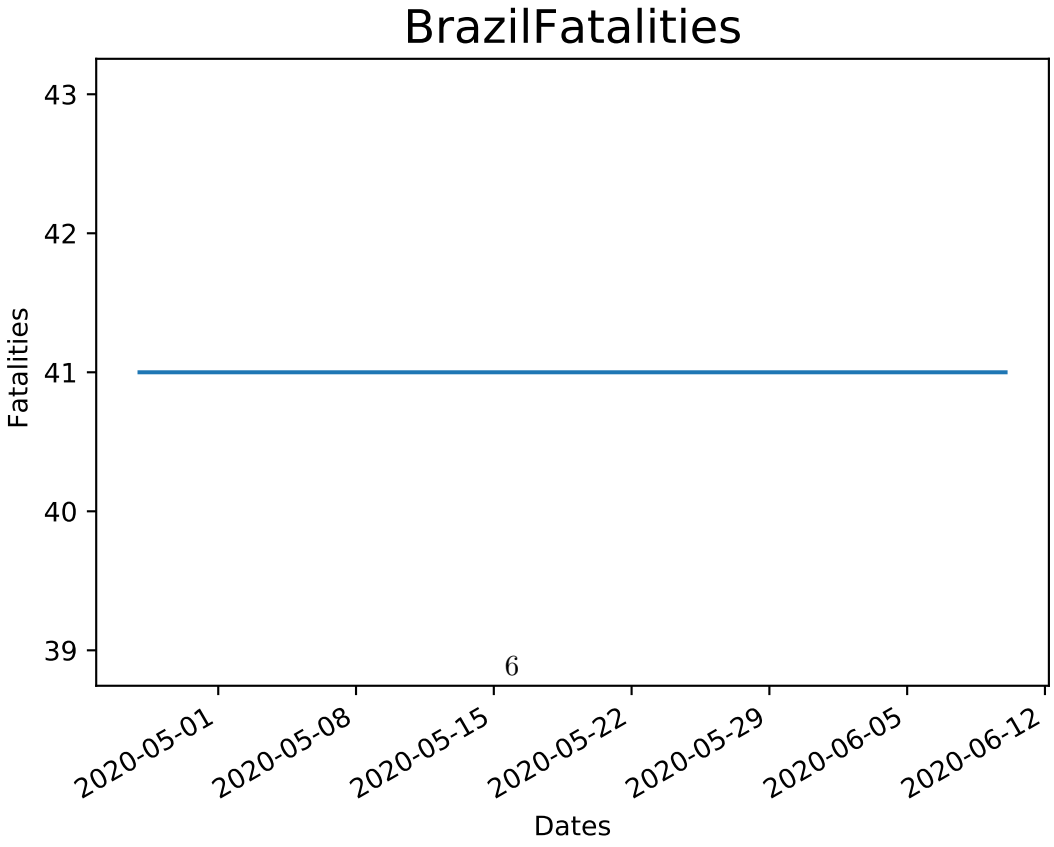
In the end, I plotted for each state two charts (one for the confirmed Covid-19 cases and one for the fatalities) using the **matplotlib** package. These are saved in the *./Problem3/images* directory that must be created before to run the application *covid19Forecasting.py*.

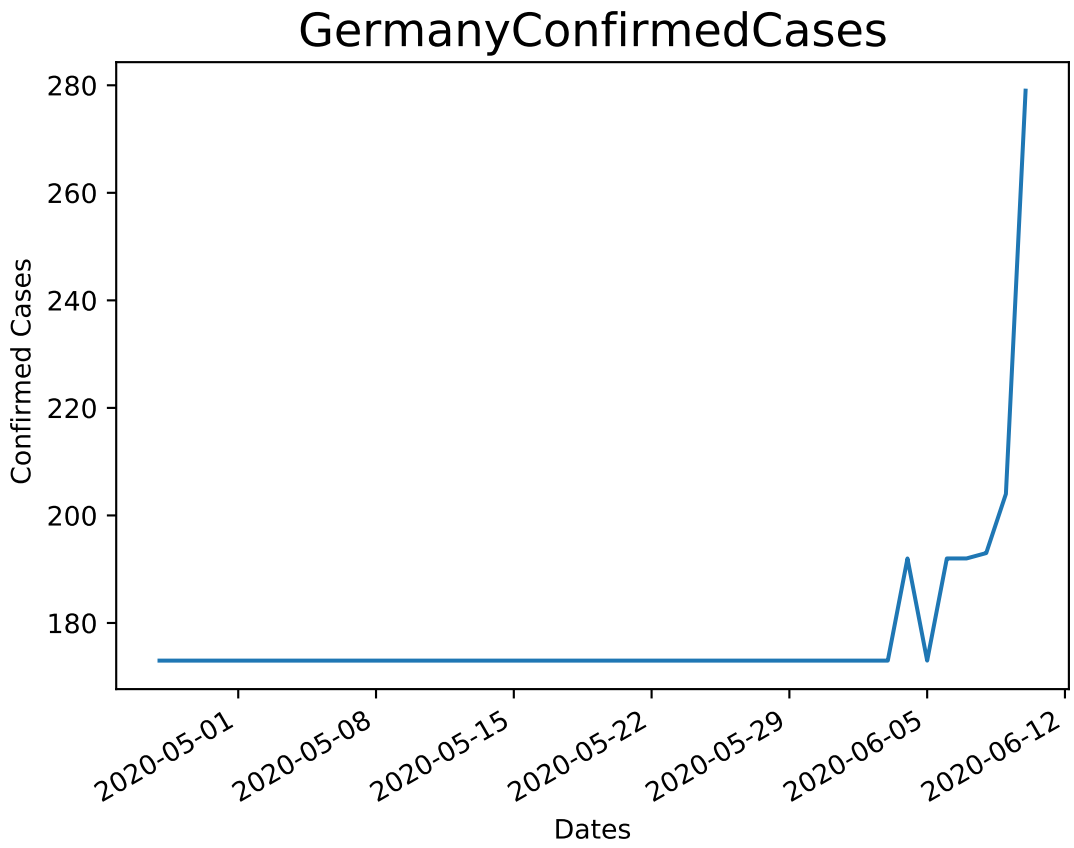
To evaluate each regression model I used the RMSE metric, the predictions values and the resulting plots and the least worst results are provided by the regression forest and the gradient-boost tree regression models.

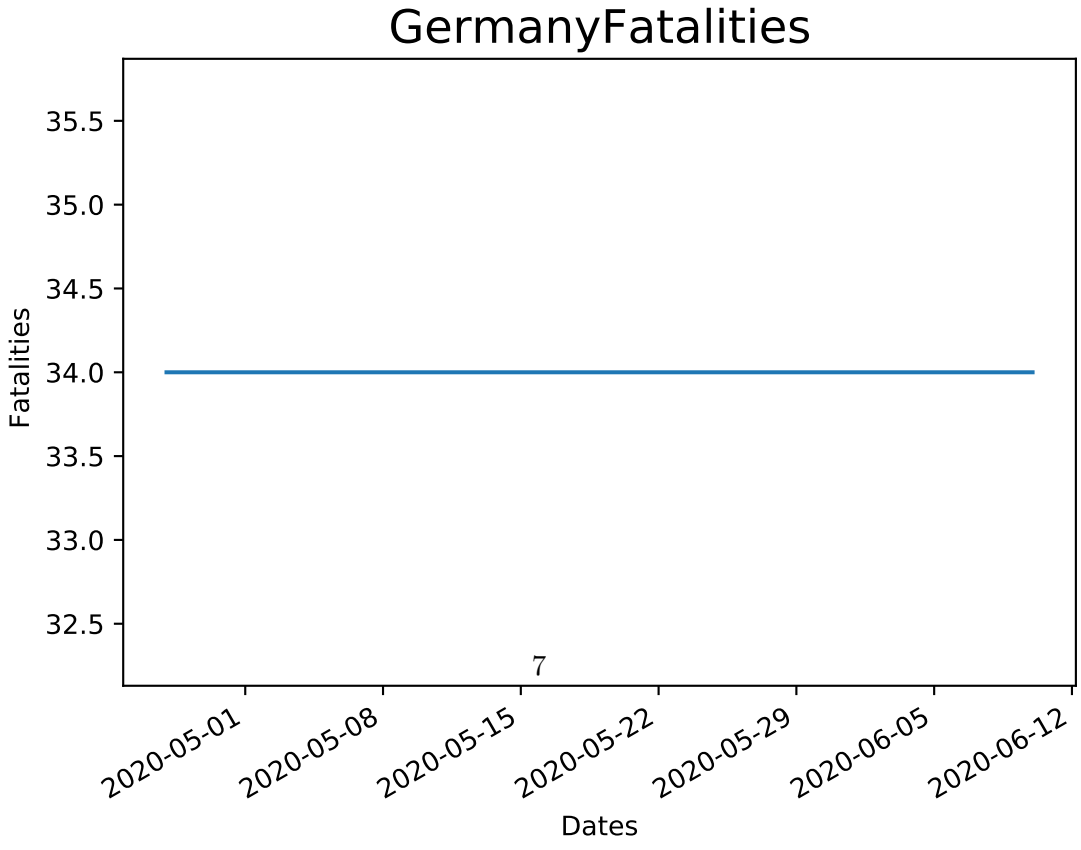
In my opinion, for having a better accurate prediction, I'd have had to build a deep learning model (Convolutional Neural Network, Recurrent Neural Network, ...) and use the data of previous days to forecast today's data.

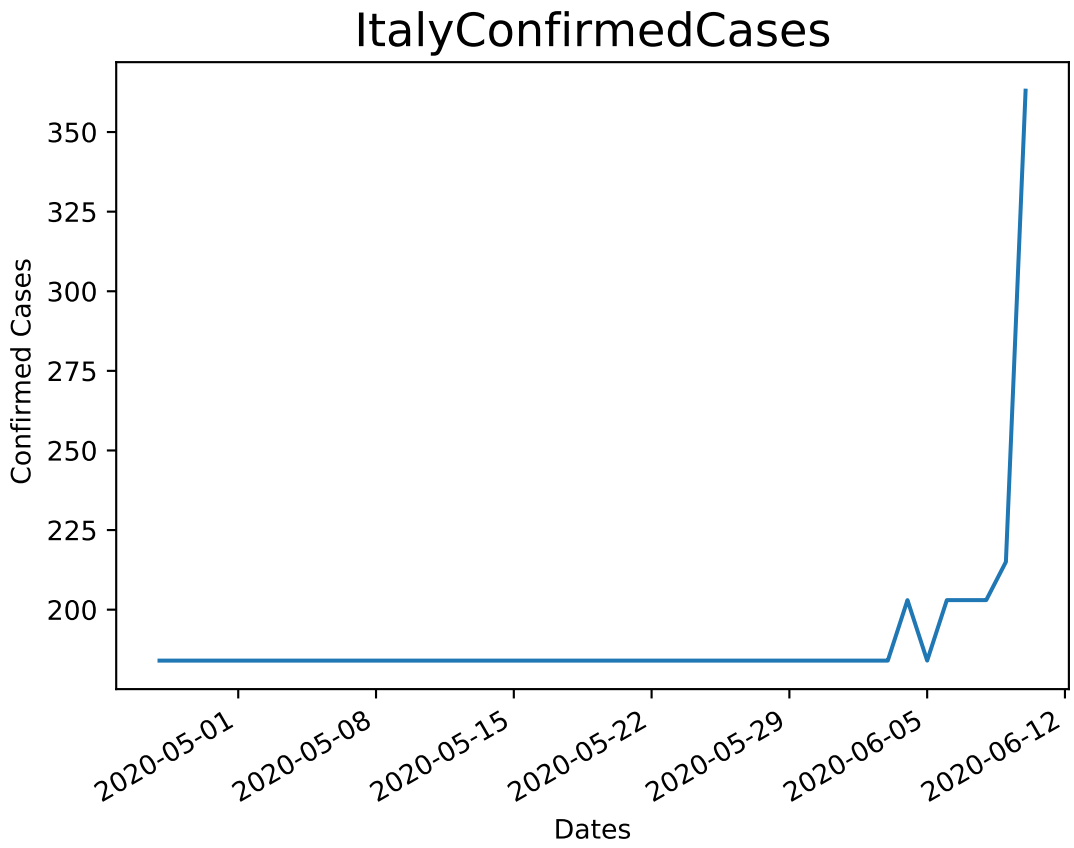
In the following pages, the results charts are shown (one country per page)

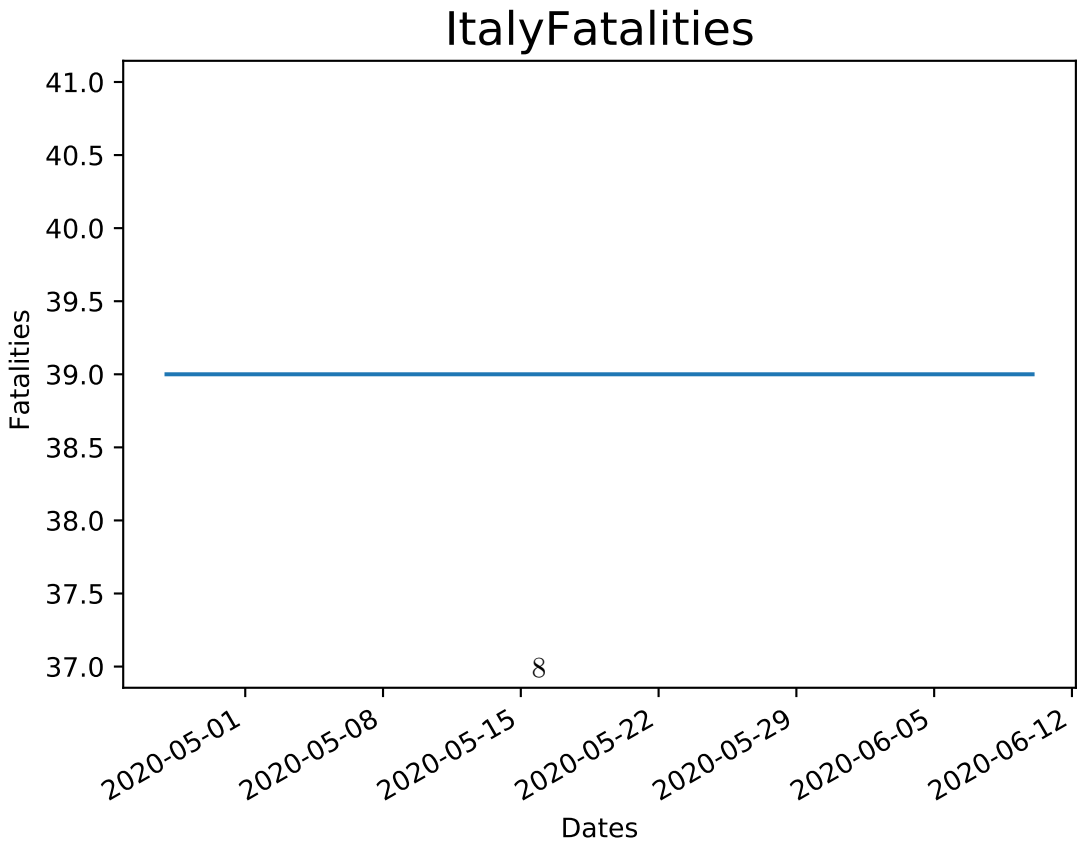

BrazilConfirmedCases
Confirmed Cases
480
460
440
420
400
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates

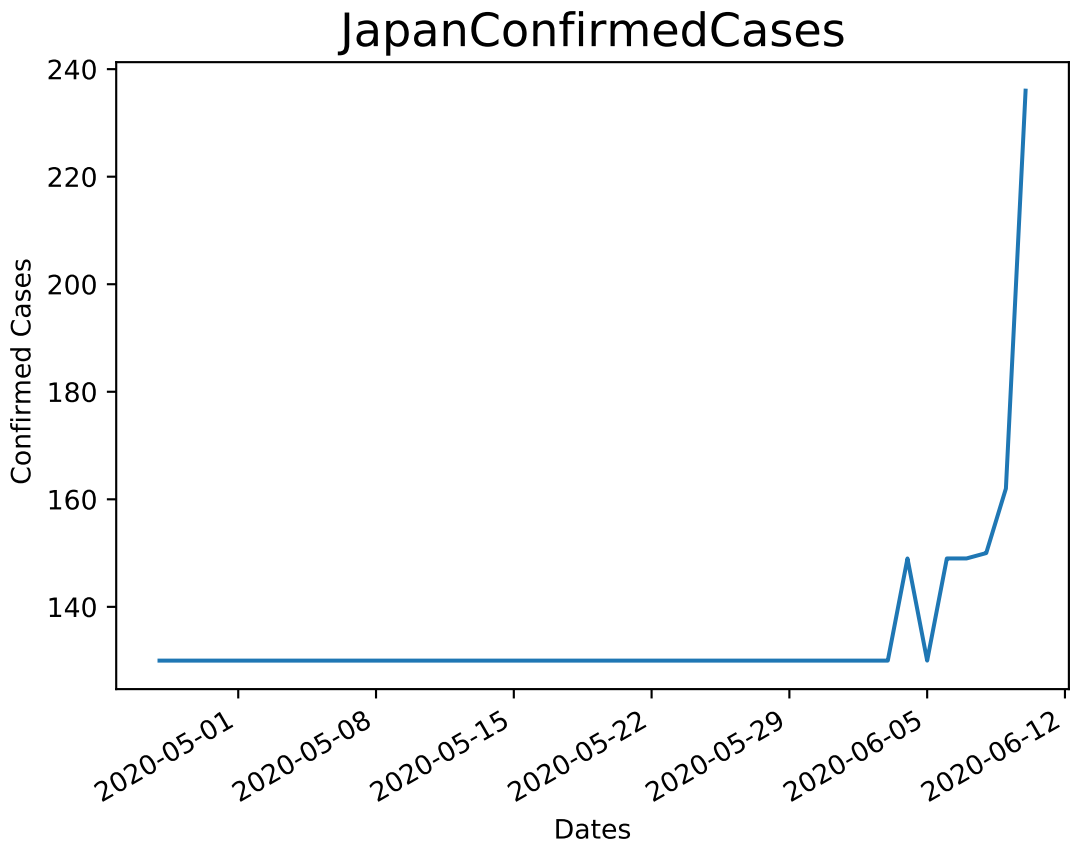

BrazilFatalities
Fatalities
43
42
41
40
39
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates

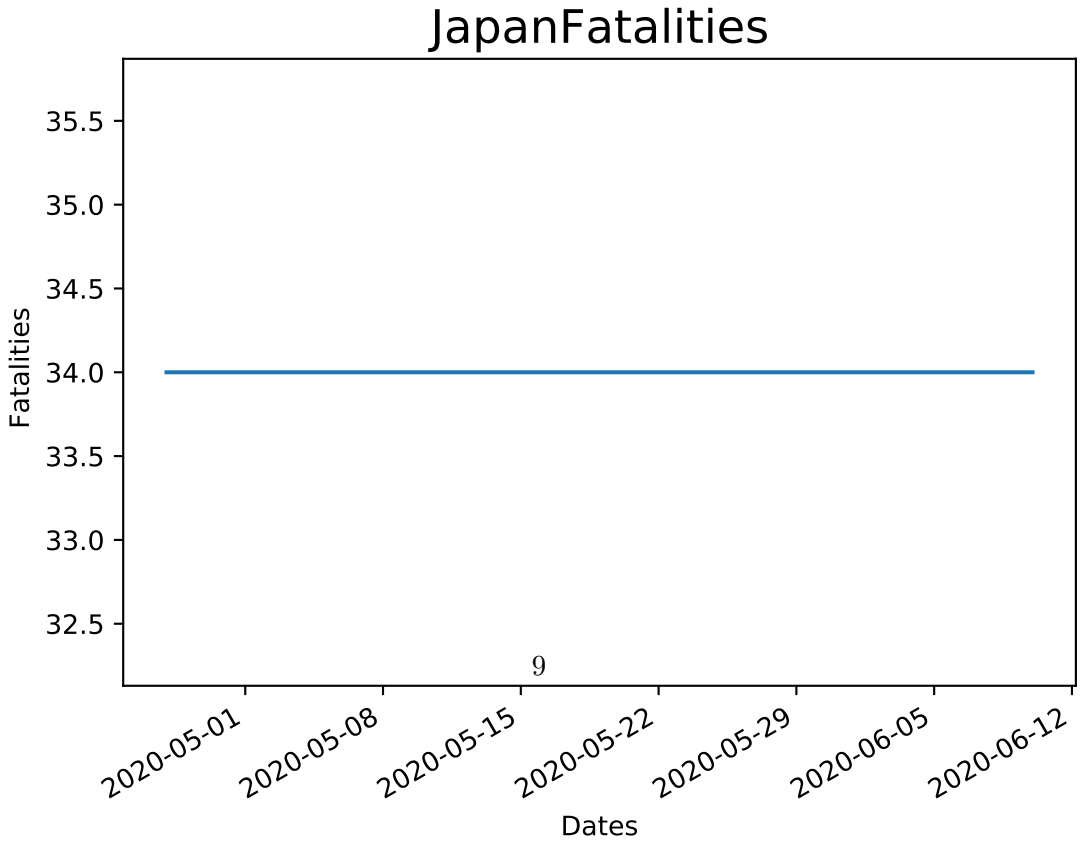

GermanyConfirmedCases
Confirmed Cases
280
260
240
220
200
180
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates

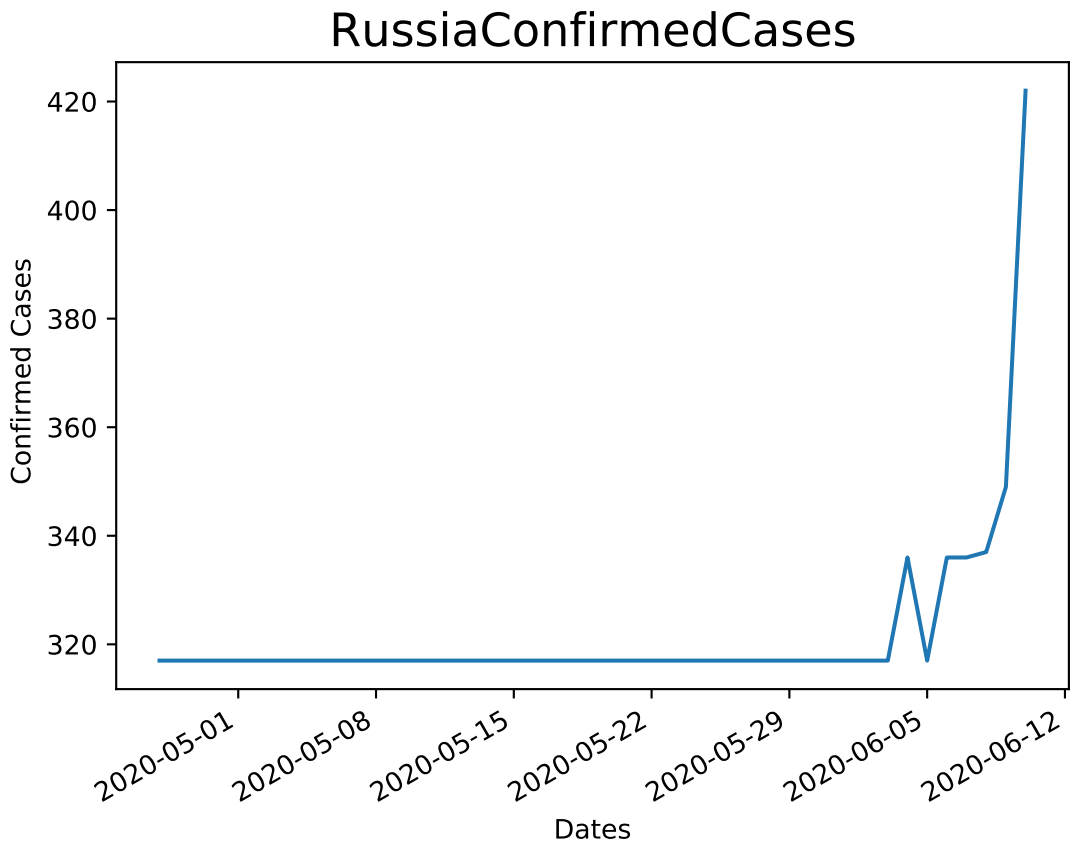

GermanyFatalities
Fatalities
35.5
35.0
34.5
34.0
33.5
33.0
32.5
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates

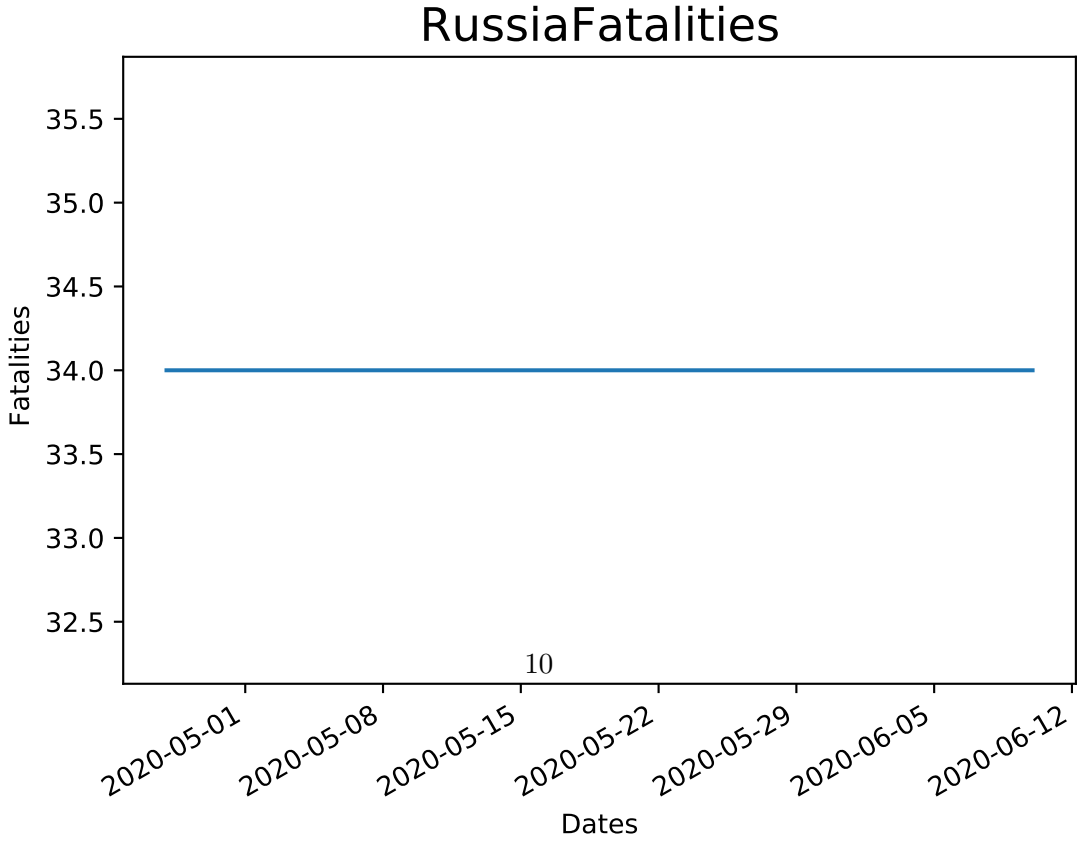

ItalyConfirmedCases
Confirmed Cases
350
325
300
275
250
225
200
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates

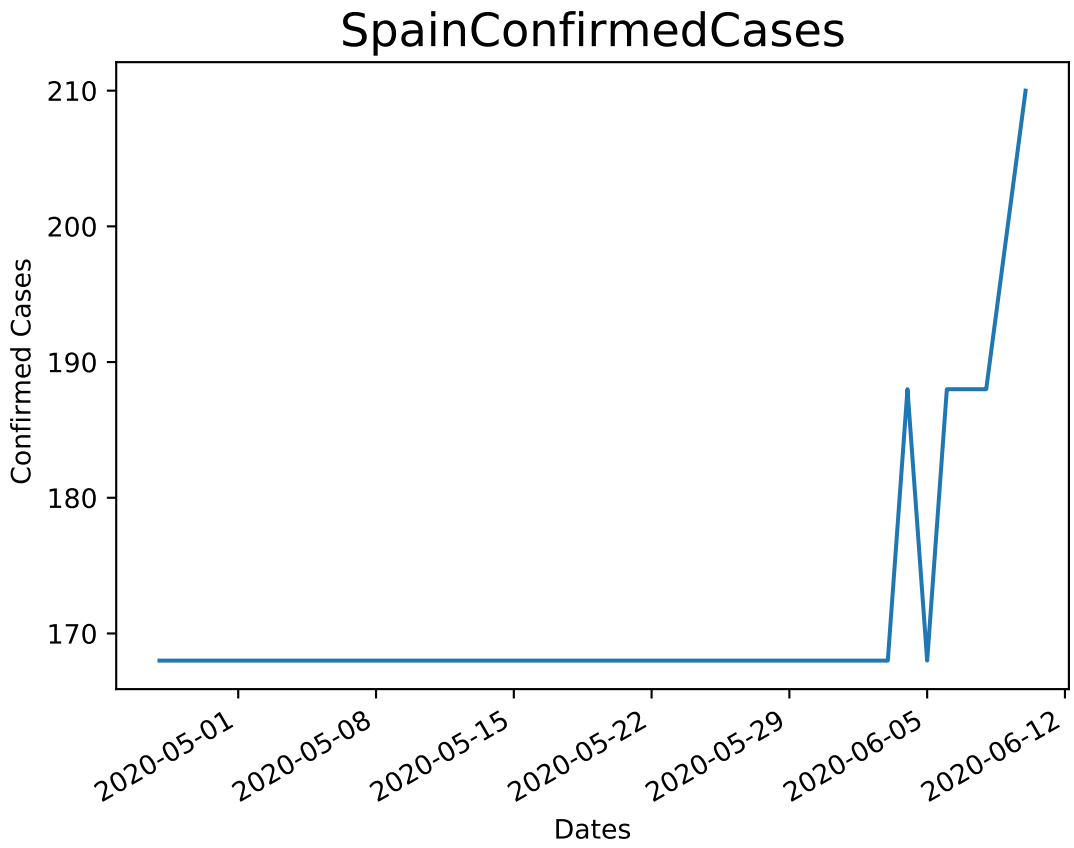

ItalyFatalities
Fatalities
41.0
40.5
40.0
39.5
39.0
38.5
38.0
37.5
37.0
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates

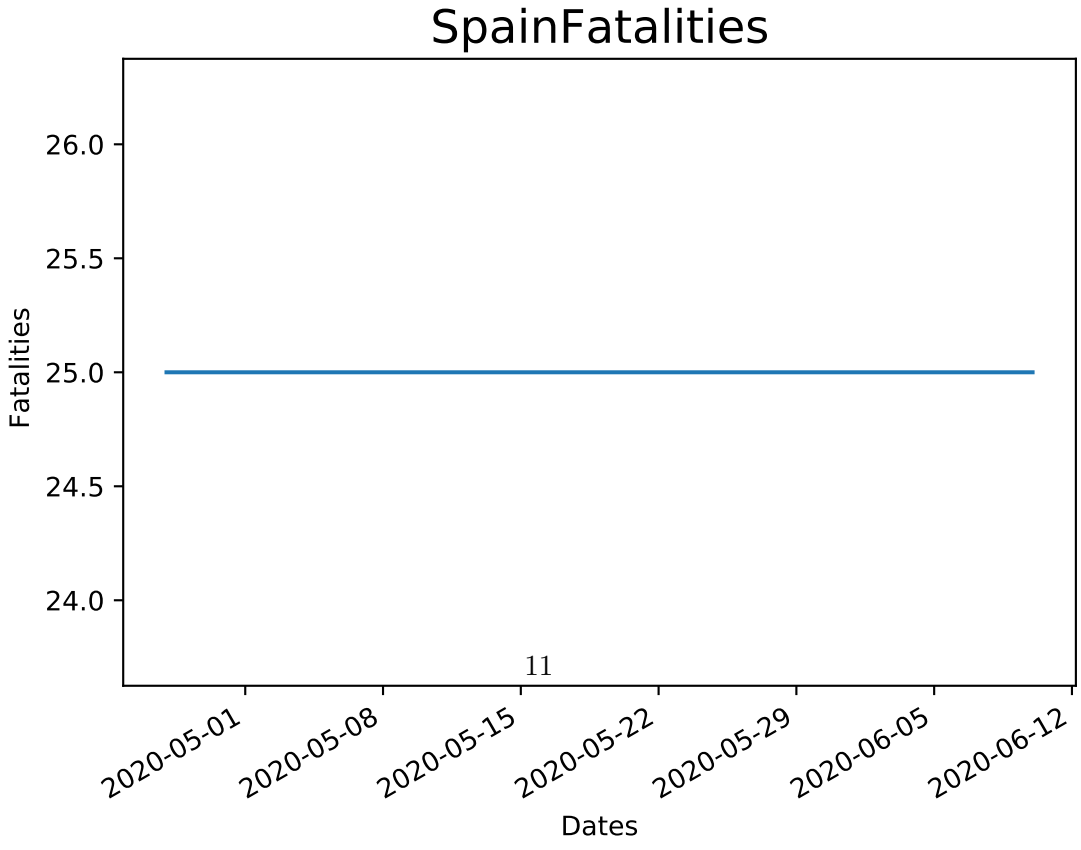

JapanConfirmedCases
Confirmed Cases
240
220
200
180
160
140
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates


JapanFatalities
Fatalities
35.5
35.0
34.5
34.0
33.5
33.0
32.5
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates


RussiaConfirmedCases
Confirmed Cases
420
400
380
360
340
320
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates


RussiaFatalities
Fatalities
35.5
35.0
34.5
34.0
33.5
33.0
32.5
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates


SpainConfirmedCases
210
200
190
180
170
Confirmed Cases
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates


SpainFatalities
26.0
25.5
25.0
24.5
24.0
Fatalities
2020-05-01
2020-05-08
2020-05-15
2020-05-22
2020-05-29
2020-06-05
2020-06-12
Dates