aide

COVID-19 ETL Data Pipeline with Dagster, Spark, and Plotly

A Data Engineering Project in "Fundamental Data Enginnering" Course - AIDE Institute

Thang Bui Q.

[AIDE] FDE02 - Student
Faculty of Mathematics and Computer Science
University of Science (VNUHCM-US)

April 15, 2023

Overview

aide

1. Overview of Datapipeline
2. Collecting data and project goal
3. Setup data pipeline with Spark
4. Data mining on data warehouse and hosting Plotly

Before the report

aide

With sincere thanks...

Table of Contents

aide

Data Flow Diagram

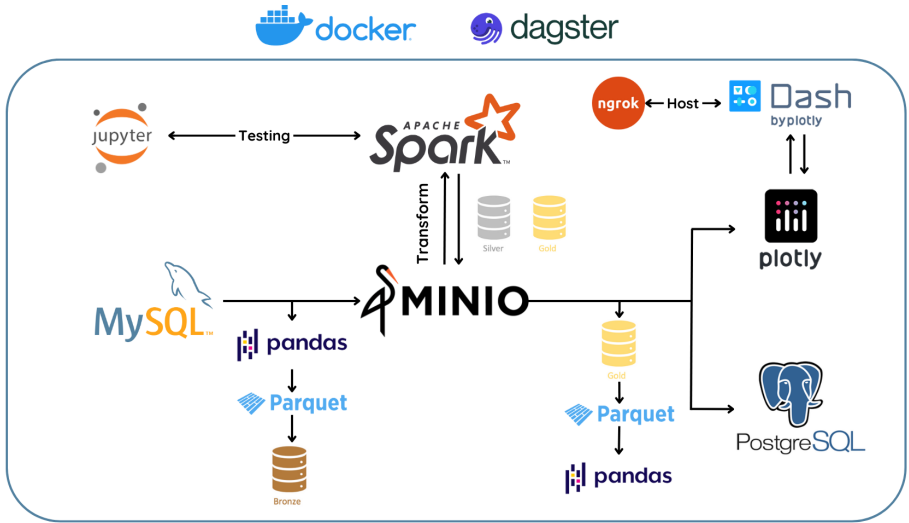


Table of Contents

aide

Collecting data

aide

- Time-series data (date-by-date), clean above 80%.
- Git-repo: https://github.com/CSSEGISandData/COVID-19

Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered	Active	WHO Region
Alberta	Canada	53.93	-116.58	2020-07-14	8912	163	0	8749	Americas
Alberta	Canada	53.93	-116.58	2020-07-15	8994	163	0	8831	Americas
Alberta	Canada	53.93	-116.58	2020-07-16	9114	165	0	8949	Americas
Alberta	Canada	53.93	-116.58	2020-07-17	9219	167	0	9052	Americas
Alberta	Canada	53.93	-116.58	2020-07-18	9219	167	0	9052	Americas
Alberta	Canada	53.93	-116.58	2020-07-19	9219	167	0	9052	Americas
Alberta	Canada	53.93	-116.58	2020-07-20	9587	170	0	9417	Americas
Alberta	Canada	53.93	-116.58	2020-07-21	9728	172	0	9556	Americas
Alberta	Canada	53.93	-116.58	2020-07-22	9728	172	0	9556	Americas
Alberta	Canada	53.93	-116.58	2020-07-23	9975	176	0	9799	Americas
Alberta	Canada	53.93	-116.58	2020-07-24	10086	178	0	9908	Americas
Alberta	Canada	53.93	-116.58	2020-07-25	10086	178	0	9908	Americas
Alberta	Canada	53.93	-116.58	2020-07-26	10086	178	0	9908	Americas
Alberta	Canada	53.93	-116.58	2020-07-27	10390	186	0	10204	Americas
	Afghanistan	33.94	67.71	2020-01-22	0	0	0	0	Eastern Mediterranean
	Albania	41.15	20.17	2020-01-22	0	0	0	0	Europe
	Algeria	28.03	1.66	2020-01-22	0	0	0	0	Africa
	Andorra	42.51	1.52	2020-01-22	0	0	0	0	Europe
	Angola	-11.2	17.87	2020-01-22	0	0	0	0	Africa
	Antigua and Barbuda	17.06	-61.8	2020-01-22	0	0	0	0	Americas
	Argentina	-38.42	-63.62	2020-01-22	0	0	0	0	Americas
	Armenia	40.07	45.04	2020-01-22	0	0	0	0	Europe
	Austria	47.52	14.55	2020-01-22	0	0	0	0	Europe
	Azerbaijan	40.14	47.58	2020-01-22	0	0	0	0	Europe

Explore Dataset

aide

covid19_cases_position.csv

RAW Lat/Long wise no. of cases

covid19_country_wise.csv

RAW country level no. of cases

covid19_time_series.csv

RAW Date wise no. of cases

covid19_worldometer.csv

RAW Worldometers data (has Continent info)

Project Goal (WHO Spreadmap)

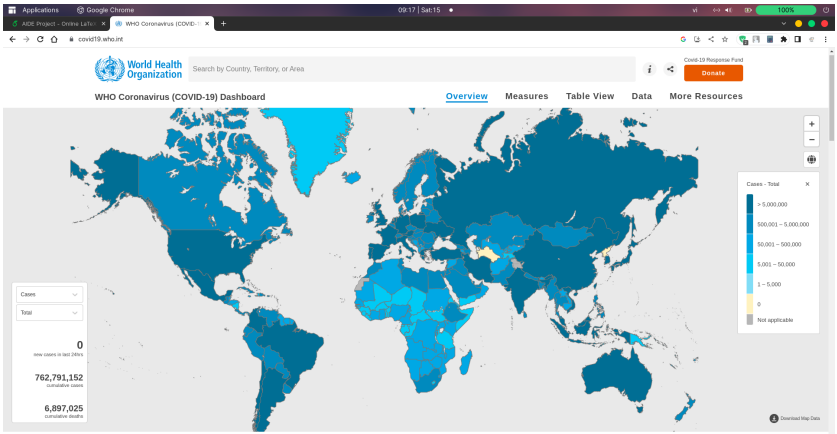


Table of Contents

Design database for Covid19

aide

We need to initialize the schemas for the RAW data in MySQL with the 4 tables (as 4 csv files). Then, we need to create clean tables with aggregated and filtered data for silver layer from 4 assets. And create two tables at last in gold layer:

1. **covid19_daily_stats**
2. **covid19_continent_stats**

Sample SQL Script For MySQL Schemas

aide

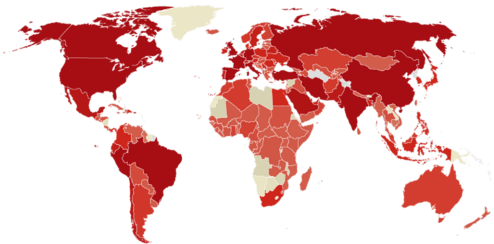
```
-- Doing the same things with 3 others
DROP TABLE IF EXISTS covid19_time_series;
CREATE TABLE covid19_time_series (
    `date`           date,
    country_region   varchar(64),
    confirmed        int4,
    deaths           int4,
    recovered        int4,
    active           int4,
    new_cases        int4,
    new_deaths       int4,
    new_recovered    int4,
    who_region       varchar(64),
    CONSTRAINT PK_covid19_timeseries PRIMARY KEY (`date`, `country_region`)
);
LOAD DATA LOCAL INFILE '/tmp/covid19-clean/covid19_time_series.csv'
INTO TABLE covid19_time_series FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Concepts of covid19_daily_stats

This asset will provide **daily** statistics for each country, including confirmed cases, deaths, and recoveries (**with latitude and longtitude**). It will join data from the covid19_cases_position and covid19_time_series tables.

Concepts of covid19_continent_stats

aide

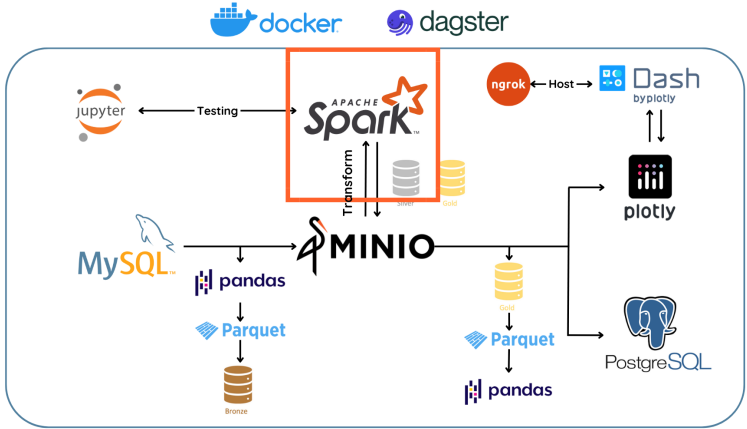


This asset will provide statistics **by continent**, including **total** confirmed cases, deaths, and recoveries. It will join data from the covid19_country_wise and covid19_worldometer tables.

Review Data Flow Diagram

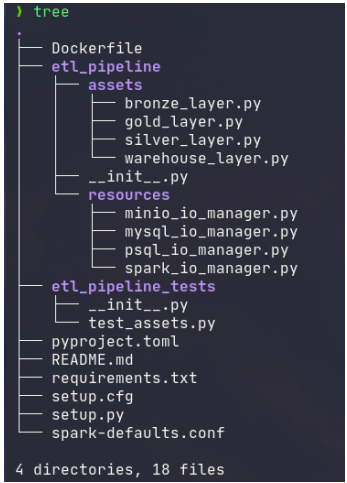
aide

- What's handle the output of Spark DataFrame and Spark (with spark-master and 2 spark-workers with 2 CORES CPU 4GB RAM)?



Design IO Manager for Spark

aide



Connecting MinIO to it and handling the output of a Pandas DataFrame, **spark_io_manager** use the Hadoop S3A connector to read and write data to **MinIO**. The IO Manager use the Spark DataFrame APIs in Spark to manipulate and transform the data as needed, and then convert it to a Pandas DataFrame using the **spark.sql(sql_stm).toPandas()** method.

Install right Python and Java version for etl-pipeline

aide

Spark need `Python` <= **3.9.16** and `Java` **17** to work right. Don't forget to add `dagster-spark==0.18.6` and `pyspark==3.3.2` in your **requirements.txt**.

```
FROM python:3.9.16-slim
SHELL ["/bin/bash", "-o", "pipefail", "-c"]
USER root
RUN apt-get update --yes && \
    apt-get install --yes --no-install-recommends \
    "openjdk-17-jre-headless" \
    ca-certificates-java && \
    apt-get clean && rm -rf /var/lib/apt/lists/* \
    pip install --upgrade pip && pip install -r requirements.txt
```

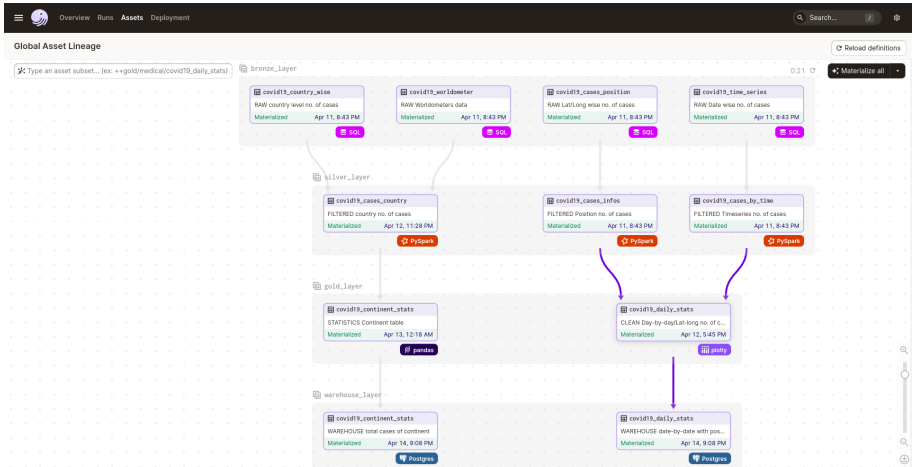
Sample Asset Code Using Spark IO Manager

aide

```python
@asset(
    ins = {"table"  : AssetIn(key_prefix = ["bronze"])},
    key_prefix=["silver"],
    io_manager_key="spark_io_manager",
    compute_kind="PySpark"
)
def sample_asset(context, pandas_data: pd.DataFrame) -> Output[pd.DataFrame]:
    spark = connect()
    spark_data = spark.createDataFrame(pandas_data)
    spark_data.createOrReplaceTempView("data")
    sql_stm = "SELECT * FROM data;"
    sparkDF = spark.sql(sql_stm)
    pd_data = sparkDF.toPandas()
    return Output(pd_data)
```

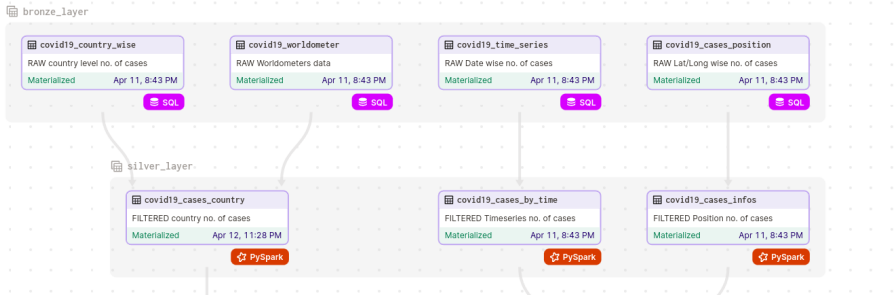
Dagster Dagit - ETL

- Full view of COVID-19 Data Lineage (EXTRACT - TRANSFORM - LOAD)



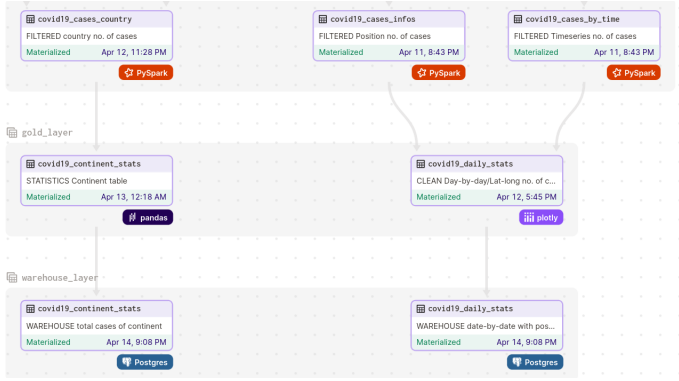
Extract - Transform

- Extract data from MYSQL Database.
- Transform by SPARK and load them into SILVER layer.



Transform - Load

- Transform by SPARK and load them into GOLD layer.
- Host Plotly by Dash, also load data into PostgreSQL Database.



Benchmark sparkSQL on gold layer

aide

GOLD: Covid19_daily_stats

CLEAN Day-by-day/Lat-long no. of cases

```
%%timeit -r 4
sql_stm1 = """
-- 1
SELECT
    t.`date`,
    t.country_region,
    t.confirmed,
    t.deaths,
    t.recovered
FROM covid19_time_series AS t
JOIN covid19_cases_position AS c
ON t.country_region = c.country_region
JOIN covid19_country_wise AS w
ON t.country_region = w.country_region
WHERE t.confirmed > 0 OR t.deaths > 0 OR t.recovered > 0;
"""
spark.sql(sql_stm1).toPandas()
```

[7]

```
1.07 s ± 252 ms per loop (mean ± std. dev. of 4 runs, 1 loop each)
```

Benchmark sparkSQL on gold layer

aide

GOLD: Covid19_continent_stats

STATISTICS Continent table

```
%%timeit -r 4
sql_stm2 = """
-- 1
SELECT
    w.continent AS Continent,
    SUM(cw.confirmed) AS TotalCases,
    SUM(cw.deaths) AS TotalDeaths,
    SUM(cw.recovered) AS TotalRecovered
FROM covid19_worldometer AS w
JOIN covid19_cases_position AS c
ON w.country_region = c.country_region
JOIN covid19_country_wise AS cw
ON w.country_region = cw.country_region
GROUP BY
    w.continent;
"""
spark.sql(sql_stm2).toPandas()
```

[12]

```
735 ms ± 240 ms per loop (mean ± std. dev. of 4 runs, 1 loop each)
```

Test Spark SQL Result

```
[13]
covid19_daily_stats = spark.sql(sql_stm1)
covid19_daily_stats.show()
```

```
+----------+--------------+---------+------+---------+
|      date|country_region|confirmed|deaths|recovered|
+----------+--------------+---------+------+---------+
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
|2020-01-22|         China|      548|    17|       28|
+----------+--------------+---------+------+---------+
only showing top 20 rows
```

```
[7]
covid19_continent_stats = spark.sql(sql_stm2)
covid19_continent_stats.show()
```

```
+-----------------+----------+-----------+--------------+
|        Continent|TotalCases|TotalDeaths|TotalRecovered|
+-----------------+----------+-----------+--------------+
|           Europe|   4860471|     476988|       2430932|
|           Africa|    828239|      17759|        487793|
|Australia/Oceania|    124070|       1358|         76031|
|    North America|   2049543|     157599|        428613|
|    South America|   3780484|     135506|       2714173|
|             Asia|   3799878|      85740|       2722251|
+-----------------+----------+-----------+--------------+
```

Mining before doing the final step

aide

It seems like you're planning to plot some data using Plotly and host it with Dash. But you're not ready yet? Why not? What do you even have to plot? First, you should focus on getting into the nitty-gritty of data mining and really enjoy it before trying to plot something. So, take a deep dive into the covid19_daily_stats, get mining and dig up some golden insights before trying to plot anything!

Table of Contents

aide

First look at gold_layer (using J-Lab as testing)

aide

```
spark_covid19_daily_stats = spark.read.parquet("s3a://warehouse/gold/medical/covid19_daily_stats.pq")
spark_covid19_continent_stats = spark.read.parquet("s3a://warehouse/gold/medical/covid19_continent_stats.pq")
df1 = pd.DataFrame(spark_covid19_daily_stats.toPandas())
df2 = pd.DataFrame(spark_covid19_continent_stats.toPandas())
```

[3]

	date	country_region	confirmed	deaths	recovered	active	latitude	longtitude	who_region
0	2020-07-27	Russia	816680	13334	602249	201097	61.5240	105.3190	Europe
1	2020-07-26	Russia	811073	13249	599172	198652	61.5240	105.3190	Europe
2	2020-07-25	Russia	805332	13172	596064	196096	61.5240	105.3190	Europe
3	2020-07-24	Russia	799499	13026	587728	198745	61.5240	105.3190	Europe
4	2020-07-23	Russia	793720	12873	579295	201552	61.5240	105.3190	Europe
...	...	...	...	...	...	...	...	...	...
49063	2020-01-26	Netherlands	0	0	0	0	18.0425	-63.0548	Europe
49064	2020-01-25	Netherlands	0	0	0	0	18.0425	-63.0548	Europe
49065	2020-01-24	Netherlands	0	0	0	0	18.0425	-63.0548	Europe
49066	2020-01-23	Netherlands	0	0	0	0	18.0425	-63.0548	Europe
49067	2020-01-22	Netherlands	0	0	0	0	18.0425	-63.0548	Europe

49068 rows × 9 columns

	Continent	TotalCases	TotalDeaths	TotalRecovered
0	Europe	2482951	155775	1605640
1	Africa	828239	17759	487793
2	Australia/Oceania	16949	189	10854
3	North America	768505	59215	428613
4	South America	3780484	135506	2714173
5	Asia	3799878	85740	2722251

Show the continent gradient table of COVID-19

aide

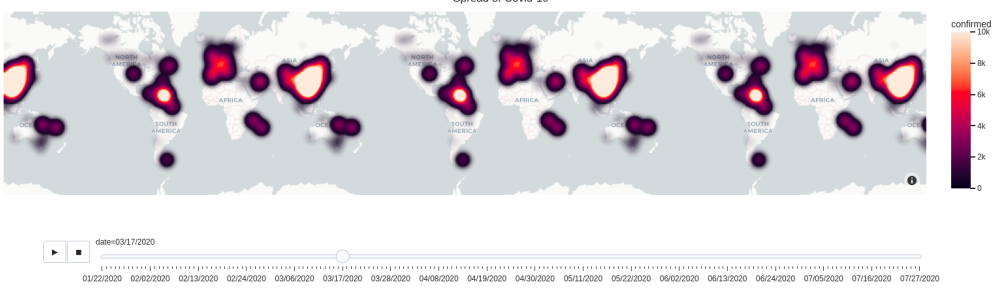
```
[10]
covid19_continent_stats.style.background_gradient(cmap="Reds")
```

	Continent	TotalCases	TotalDeaths	TotalRecovered
0	Europe	2482951	155775	1605640
1	Africa	828239	17759	487793
2	Australia/Oceania	16949	189	10854
3	North America	768505	59215	428613
4	South America	3780484	135506	2714173
5	Asia	3799878	85740	2722251

Plot the spread map of COVID-19

```
import plotly.io as pio
pio.templates.default = "seaborn"
df1['date'] = df1['date'].dt.strftime('%m/%d/%Y')
fig_map = px.density_mapbox(df1, lat='latitude', lon='longtitude', z='confirmed', radius=20,zoom=1, hover_data=["country_region","who_region","confirmed"],
                            mapbox_style="carto-positron", animation_frame = 'date', range_color= [0, 10000],title='Spread of Covid-19')
fig_map.update_layout(margin={"r":0,"t":30,"l":0,"b":0})
fig_map.show()
```

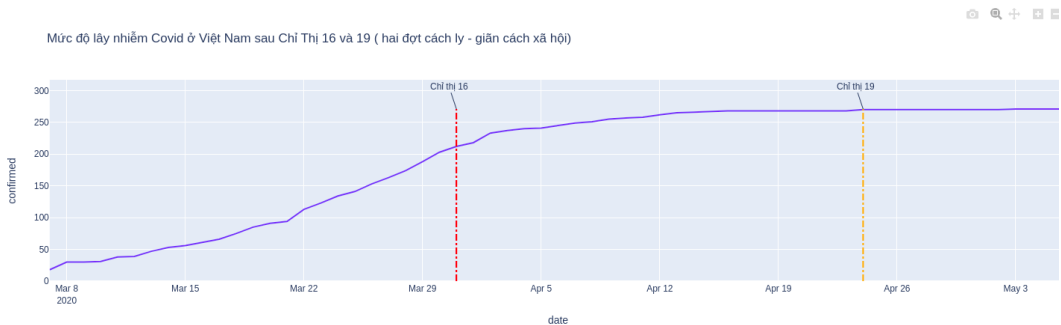
MagicPython



It's done! But wait...it's timeseries!

aide

- How does DS do forecasting? Why you must design gold layer like that?
- Example: (Covid19 Confirmed Cases Rate At Vietnam) - data doesn't lie
- 16/CT-TTg about quarantine (src: vncdc.gov.vn): 31-03-2020

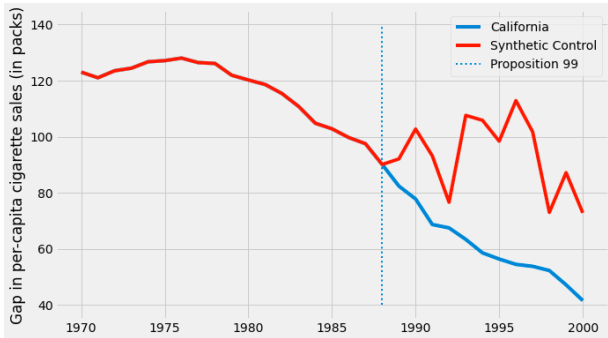


Concepts of Synthetic Control

aide

- Given the weights $\boldsymbol{W} = (w_2, ..., w_{J+1})$, the synthetic control estimate of Y_{jt}^N is:

$$\hat{Y}_{jt}^N = \sum_{j=2}^{J+1} w_j Y_{jt}$$



Train models by Linear Regression

aide

```
Get data_synth = Filter (date < 2020/4/20, continent == Southest Asia)
#With sincere thanks: Minh Man Le (HCMUS)
from sklearn.linear_model import LinearRegression
Y = data_synth["Vietnam"].values # Vietnam
X = data_synth.drop(columns="Vietnam").values # other SA countries
weights_lr = (
    LinearRegression(fit_intercept = False)
    .fit(X,Y)
    .coef_
)
```

Multiply matrices to make covid_synth_lr

aide

Fit Weights bằng Linear Regression:

$$||X_1 - X_0 W|| = \left(\sum_{h=1}^{k} v_h \left(X_{h1} - \sum_{j=2}^{J+1} w_j X_{hj} \right)^2 \right)^{\frac{1}{2}}$$

```
[18]  weights_lr = LinearRegression(fit_intercept = False).fit(X,Y).coef_
```

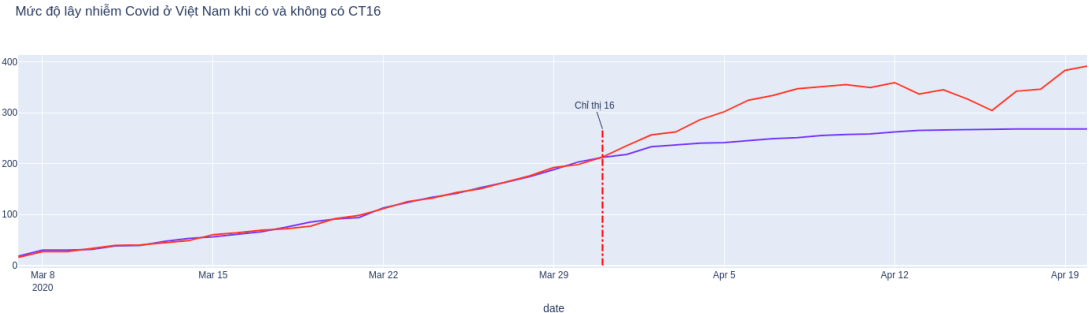
```
array([-0.10893782,  8.73269379, -1.80399272,  0.02270781, -0.00903101,
        2.86670085,  3.11041653,  0.20735996,  0.05780642,  0.26259073])
```

Đường synthetic control của Covid19 (các dữ liệu ước lượng $\hat{Y}$)

$$\hat{Y}_{jt}^{N} = \sum_{j=2}^{J+1} w_j Y_{jt}$$

```
    covid_synth_lr = data_for_synth_line.values.dot(weights_lr)
```

Plot the synthetic control line of Vietnam



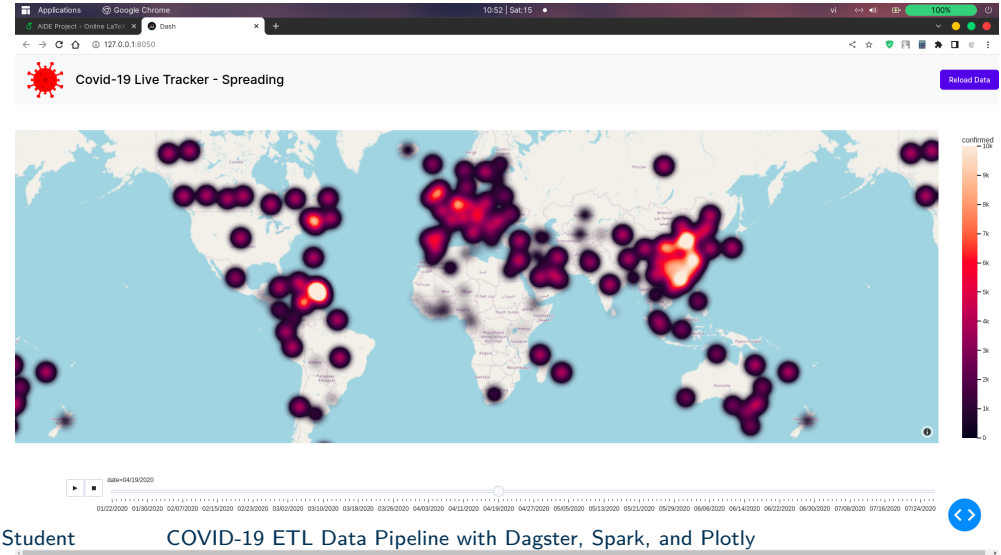
Build Dash App with Plotly

aide

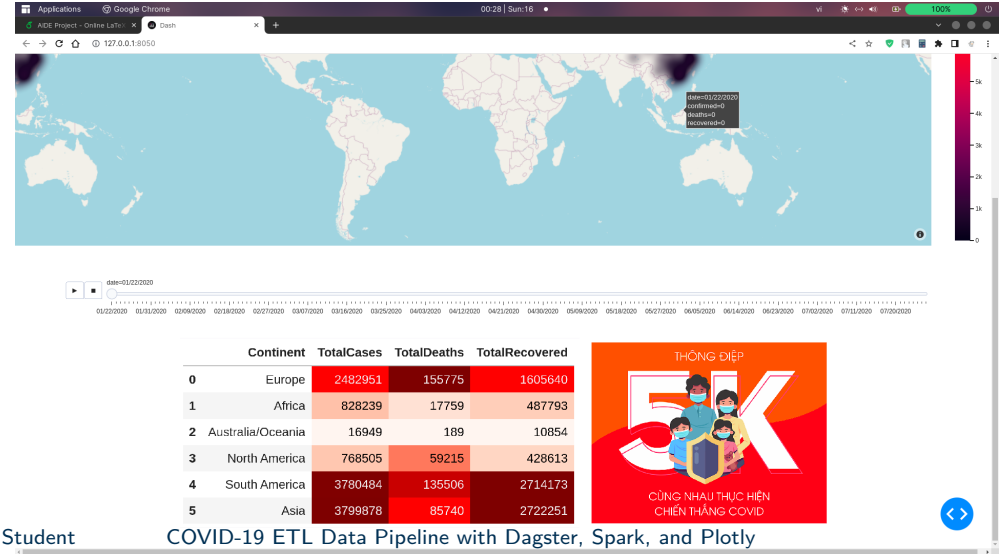
```
%%writefile my_dash_app.py
import dash
import dash_core_components as dcc
import dash_html_components as html
app = dash.Dash(__name__)
#fig_map define
app.layout = html.Div(children=[
    html.Div([
        dcc.Graph(
            id='graph1',
            figure=fig_map
        ),
    ])])
if __name__ == '__main__':
    app.run_server(debug=True, use_reloader=True)
```

!python3 ./my_dash_app.py && ngkrok http 8050

Final Result and Demo



Final Result and Demo



	Continent	TotalCases	TotalDeaths	TotalRecovered
0	Europe	2482951	155775	1605640
1	Africa	828239	17759	487793
2	Australia/Oceania	16949	189	10854
3	North America	768505	59215	428613
4	South America	3780484	135506	2714173
5	Asia	3799878	85740	2722251

Full source code

aide

- See full source code on my Github Repository at thangbuiq/covid19-etl-pipeline: **https://github.com/thangbuiq/covid19-etl-pipeline**
- Or scan this red-thing:

References


AIDE Institute (2023)
Fundamental of Data Engineering
Slides and Courseworks


Ong Xuan Hong (2023)
Medium.com
DataOps 03: Trino + DBT + Spark — Everything Everywhere All at Once


Eduardo Sarmento (2020)
Medium.com
How to Create a Simple Dashboard With Plotly

Thanks for listening!