Modern data analytics

COVID-19 analysis

Sebastiaan Van den Broeck

KUL

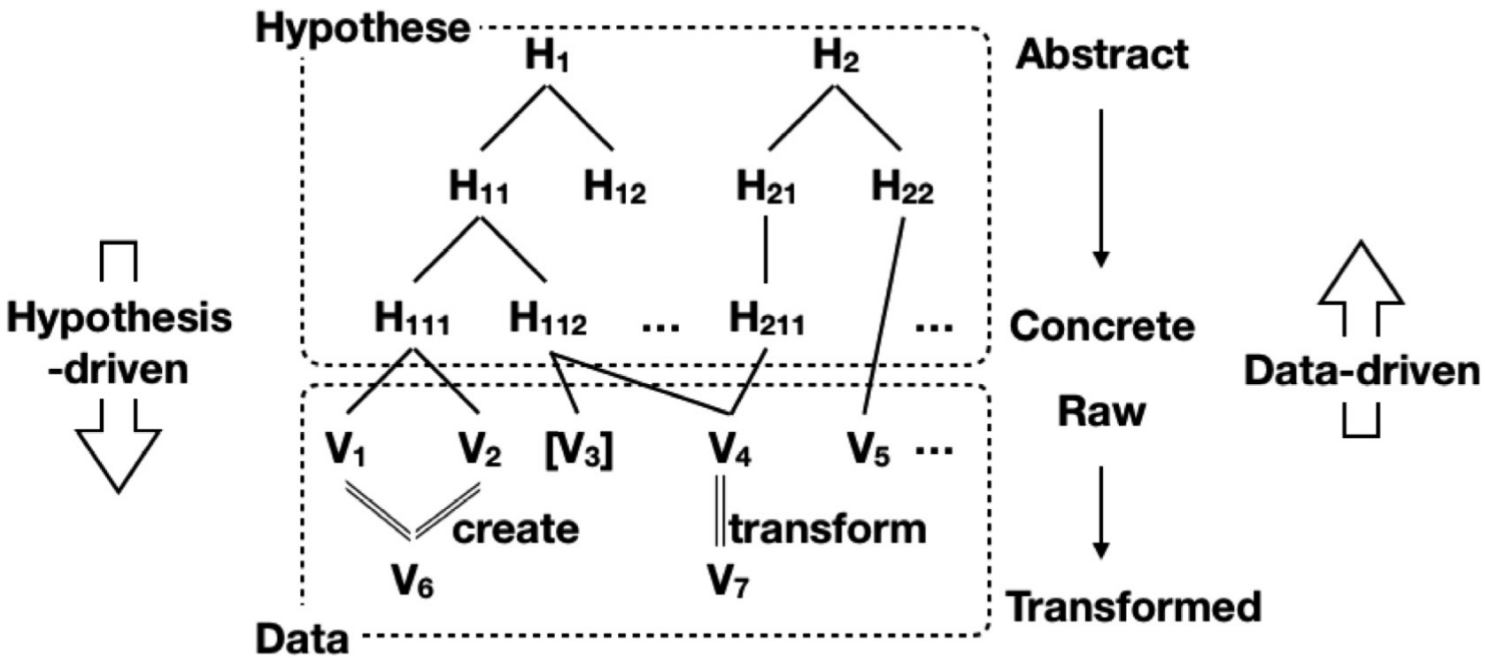
August 11, 2022

Outline

Introduction

This is the introduction

Methodology

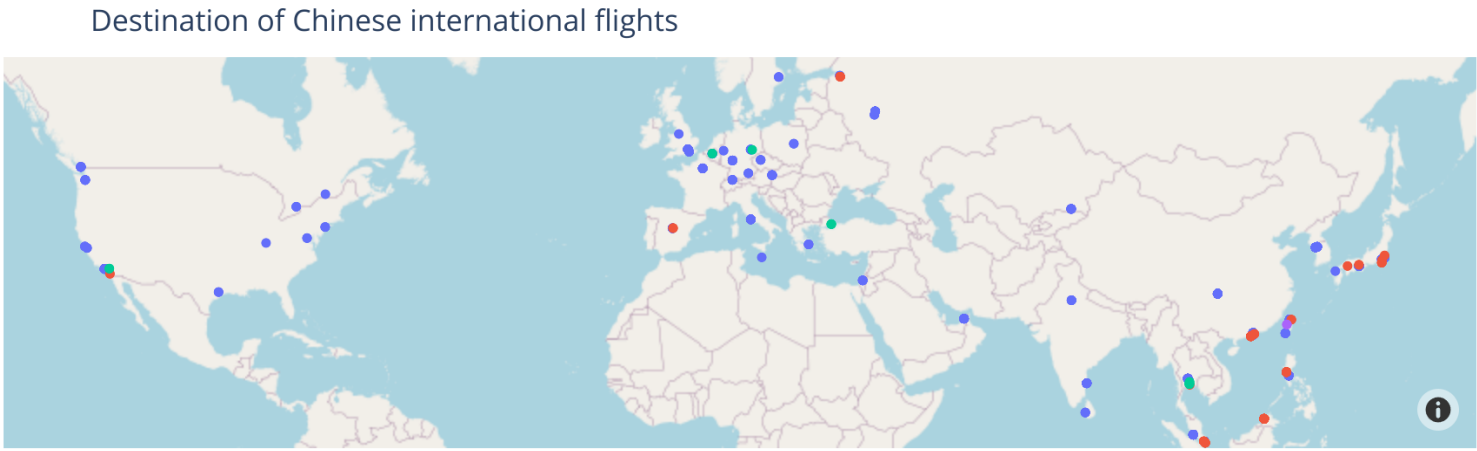


Data sources

1. The COVID-19 dataset
2. Economical information
3. The COVID-19 OpenSky dataset
4. The CORD-19 dataset

Exploratory data analysis

Graph mining



Graph mining

- Degree

  $c_D(i) = \sum_j^N x_{ij}$

  where i, j are nodes and x is the adjacency matrix.

- Betweenness

  $c_B(i) = \sum_{s,t \in i} \frac{\sigma(s,t|i)}{\sigma(s,t)}$

  where i is a node, s and t are source and target nodes, $\sigma(s,t)$ is the number of shortest paths between the source and target and $\sigma(s,t|v)$ is the number of shortest paths passing through the node v.

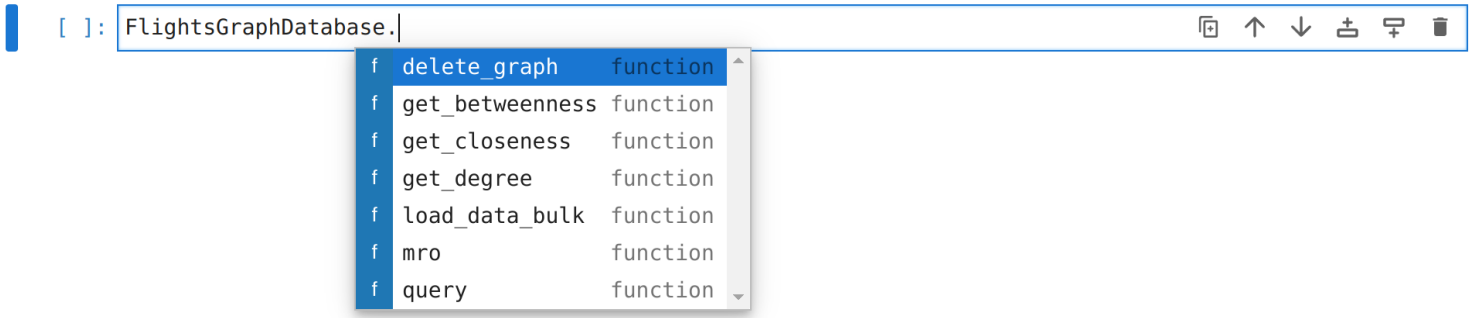
- Closeness

  $c_C(i) = \left[\sum_j^N d(i,j)\right]^{-1}$

  where d is a distance metric.

(Brandes, 2008; Opsahl et al., 2010)

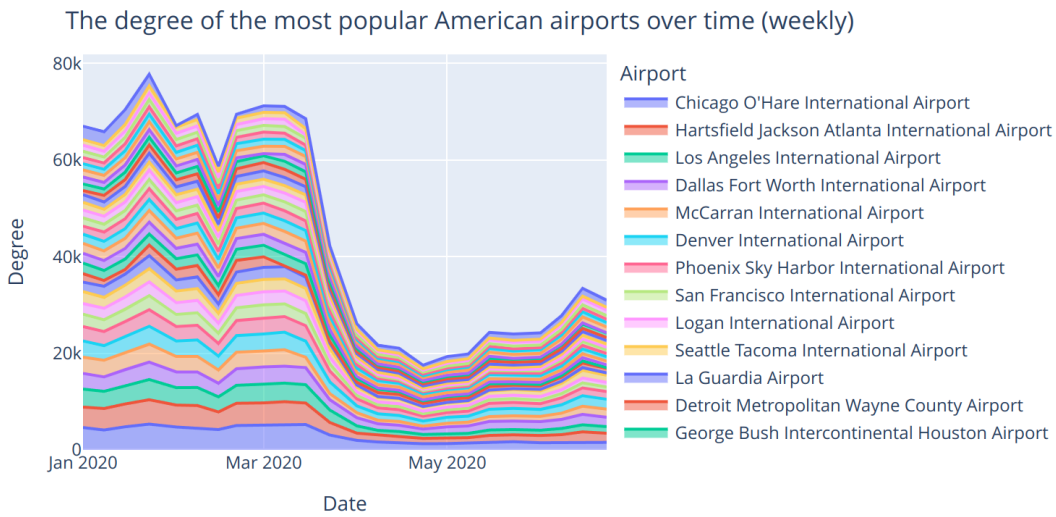
Graph mining - the code




```python
class FlightsGraphDatabase:
    def __init__(self, url, user, password):
        self.graph = Graph(url, auth=(user, password))


    def query(self, query) -> pd.DataFrame:
        """Runs a Cypher query on the graph and returns the results as a
        Pandas DataFrame."""

        return self.graph.run(query).to_data_frame()
```

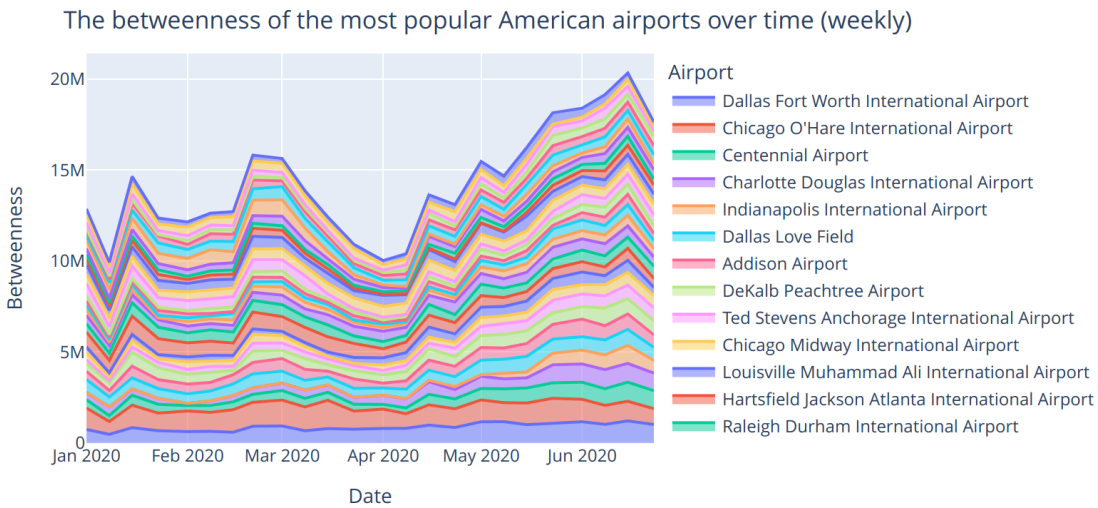
Graph mining - degree centrality

```python
def get_degree(self) -> pd.Dataframe:
    # First, create a projection
    self.query("CALL gds.graph.drop('flights', false)")
    self.query("CALL gds.graph.project('flights', 'Airport', 'FLIGHT') YIELD *")
    # Then, calculate the centrality for each node
    result = self.query("""CALL gds.degree.stream('flights')
                           YIELD nodeId, score MATCH (n:Airport)
                           WHERE ID(n) = nodeId RETURN n.name, n.code, score""")
    # Finally, sorting the results by degree
    result = result.sort_values("score", ascending=False)

    return result
```



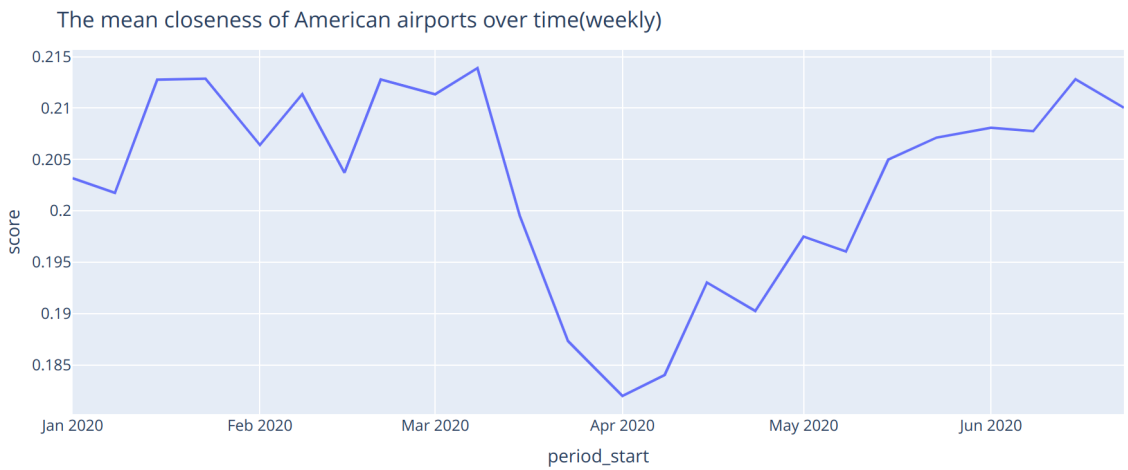
Graph mining - betweenness centrality

```python
def get_betweenness(self) -> pd.DataFrame:
    # First, create a projection
    self.query("CALL gds.graph.drop('flights', false)")
    self.query("CALL gds.graph.project('flights', 'Airport', 'FLIGHT') YIELD *")
    # Then, calculate the centrality for each node
    result = self.query("""CALL gds.betweenness.stream('flights')
                           YIELD nodeId, score MATCH (n:Airport)
                           WHERE ID(n) = nodeId RETURN n.name, n.code, score""")
    # Finally, sorting the results by degree
    result = result.sort_values("score", ascending=False)

    return result
```



Graph mining - closeness centrality

```python
def get_closeness(self) -> pd.DataFrame:
    # First, create a projection
    self.query("CALL gds.graph.drop('flights', false)")
    self.query("CALL gds.graph.project('flights', 'Airport', 'FLIGHT') YIELD *")
    # Then, calculate the centrality for each node
    result = self.query("""CALL gds.beta.closeness.stream('flights')
                           YIELD nodeId, score MATCH (n:Airport)
                           WHERE ID(n) = nodeId RETURN n.name, n.code, score""")
    # Finally, sorting the results by degree
    result = result.sort_values("score", ascending=False)

    return result
```



Text mining - preprocessing

```python
def preprocess_text(self, series: pd.Series) -> pd.Series:
    """Applies some preprocessing steps to a collection of documents."""
    # Remove stopwords
    series = [gensim.parsing.preprocessing.remove_stopwords(i)
              for i in series]
    # Stem
    series = gensim.parsing.porter.PorterStemmer().stem_documents(series)
    # Remove numeric
    series = [gensim.parsing.preprocessing.strip_numeric(i)
              for i in series]
    # Remove punctuation
    series = [gensim.parsing.preprocessing.strip_punctuation(i)
              for i in series]
    # Remove special characters
    series = [re.sub("\\W+", " ", i) for i in series]
    # Remove short words
    series = [gensim.parsing.preprocessing.strip_short(i) for i in series]

    return series
```

Text mining - latent dirichet allocation

This is a frame.

Conclusion

This is a frame.

Sources