Migration Data Visualization Dashboard

Neetha Reddy
IMT2018050
neetha.reddy@iiitb.ac.in

International Institute of Information Technology Bangalore

1 Introduction

According to the most recent UN Population Division estimates, the number of foreign migrants worldwide was 280.6 million (or 3.6 per cent of the world's population) as of mid-2020[1]. Although some people have the option to relocate, most are forced to do so owing to external forces (such as wars, lack of employment, poor economic conditions, or natural catastrophes). Understanding the causes and implications of migration, as well as forecasting future flows, is critical for the development of national public policy and urban resource planning. Visualization aids in the interpretation of vast volumes of migrant flow data and provides insights into how a country's policies influence migrant mobility. They are, in fact, necessary for extracting information from unstructured data. For example, the dataset I chose has almost 68,000 lines of raw data, and we cannot derive useful inferences from it without employing visualisation tools. I used the UNHCR migration datasets from 2000 to 2016 for the scope of visualisations in this tool [2].

2 Software Design Requirements

This visualisation tool was created to track international migration flows into and out of each country. The dashboard has two synchronised spatio-temporal visualisation displays that are juxtaposed with one other. The user should be able to determine the inflow/outflow of migrants from the chosen regions, evaluate the composition of migrants/immigrants, and examine interconnection across nations at any given snapshot.

3 Choice of Technology

This dashboard is built using Dash, a low-code framework used for rapidly building data apps, and Flask, in Python. This tech stack was chosen since Dash abstracts all of the technologies and protocols required to construct a full-stack web app with interactive data visualisations. Since Dash apps are rendered in the web browser, we may deploy them to Virtual Machines or Kubernetes clusters and then distribute them via URLs and are croos-platform and mobile ready to boot. Some of the other libraries used for the visualizations are Plotly and Networkx.

4 Implementation Details

This section covers the tool's implementation specifics, including data gathering methods, preprocessing of the collected data, and visualisation of the processed data. These processes are detailed below:

4.1 Dataset

The data used in this tool was collected from Refugee data published in the UNHCR Statistical Database[2]. The dataset comprised of a single .csv file containing ***7 fields***:

1. Country or territory of asylum or residence
2. Country or territory of origin

3. Year
4. Refugees
5. Refugees assisted by UNHCR
6. Total refugees and people in refugee-like situations
7. Total refugees and people in refugee-like situations assisted by UNHCR

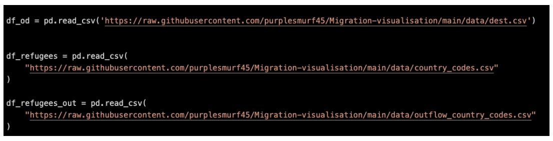
There are roughly 96,000 lines of raw data in the dataset, which spans the years 1975 to 2016.

4.2 Data Preprocessing

While reviewing the data, I discovered that the data collected from 1975 to 1999 was relatively sparse and limited to only a few countries. Furthermore, a large number of data points are missing from the data available for the number of refugees assisted by UNHCR and the total number of persons living in refugee-like conditions. For these reasons, I solely considered data from 2000 to 2016 and removed the aforementioned features from my visualisations.

I split the resultant dataset into 3 files for ease of parsing during visualisations. These files are:

1. ***dest.csv*** : The resultant dataset after performing the above mentioned operations.
2. ***country_codes.csv*** : In order to calculate the incoming migrants, I grouped the resultant dataset by the destination country and year. Then, in order to map a country to its proper geographical location in the map-based visualisation, I added a new column called "CODE" from 2014 world GDP dataset(from plotly sample dataset[4]) which plotly uses to plot the data.
3. ***outflow_country_codes.csv*** : In order to calculate the outgoing migrants, I grouped the resultant dataset by the origin country and year. Then, in order to map a country to its proper geographical location in the map-based visualisation, I added a new column called "CODE" from 2014 world GDP dataset(from plotly sample dataset[3]) which plotly uses to plot the data.



```
df_od = pd.read_csv('https://raw.githubusercontent.com/purplesmurf45/Migration-visualisation/main/data/dest.csv')


df_refugees = pd.read_csv(
    "https://raw.githubusercontent.com/purplesmurf45/Migration-visualisation/main/data/country_codes.csv"
)

df_refugees_out = pd.read_csv(
    "https://raw.githubusercontent.com/purplesmurf45/Migration-visualisation/main/data/outflow_country_codes.csv"
)
```

Figure 1: File paths

Owing to the large size of the dataset, data cleaning and pre-processing were done in a Kaggle notebook[4] to overcome computational limitations of my local system.

4.3 Data Visualization

I have visualised the preprocessed data using:

1. ***Choropleth*** : This visualization is is used to visualize logarithm of the number of migrants/immigrants(based on user selection) for all the countries in the world. Tealrose colormap was used for this visualisation (Fig. 2 and Fig. 3).
2. ***Bar Chart*** : This visualization is used to visualize the number of migrants/immigrants in the user selected regions (Fig. 4 and Fig. 5).
3. ***Sankey Diagram*** : This visualization is used to visualize the composition of migrants/immigrants in the user selected regions(Fig. 6 and Fig. 7).
4. ***Node-link Diagram*** : This visualization is used to visualize the interconnectivity between countries in the user selection regions. This can prove helpful while identifying migrant/immigrant hubs. Tealrose colormap was used for this visualisation(Fig. 8, Fig. 9 and Fig. 10).

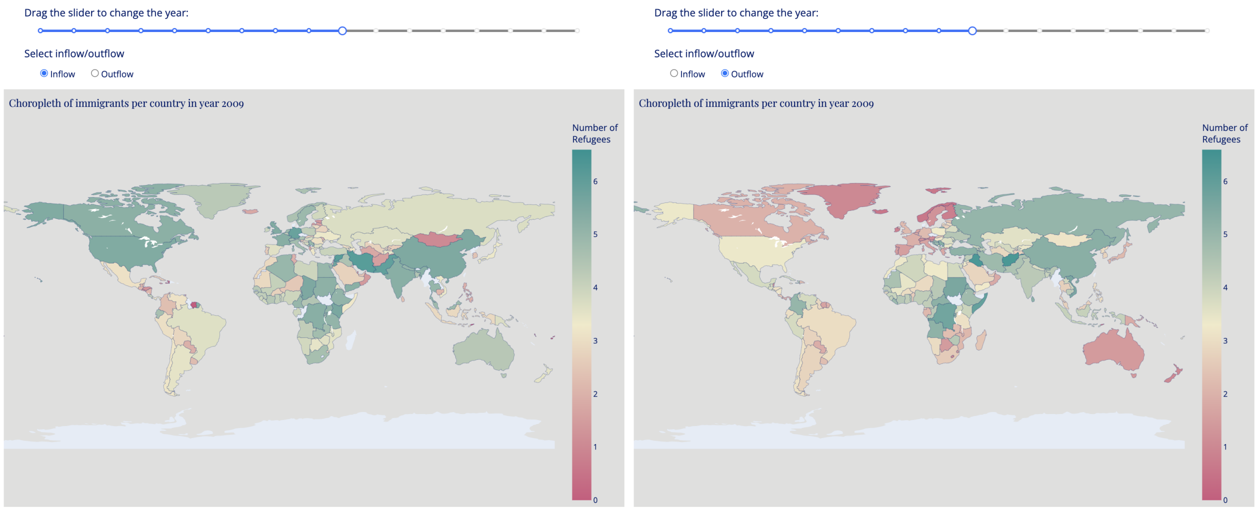
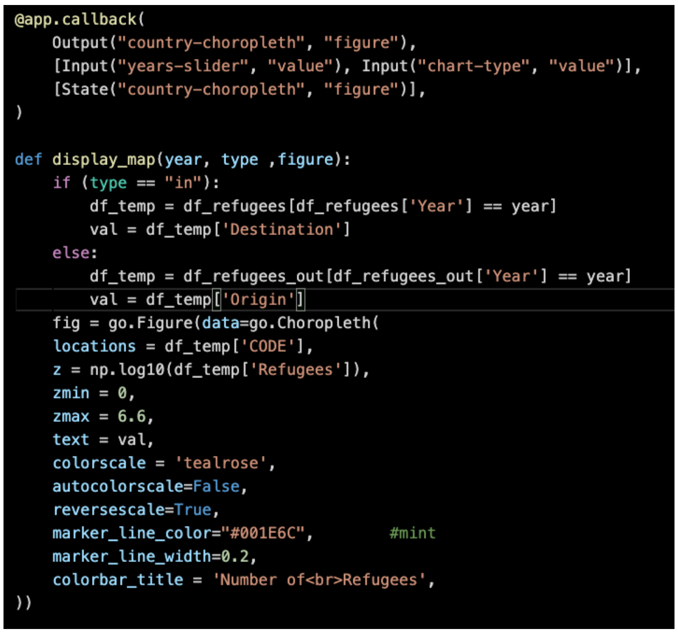

Drag the slider to change the year:
Select inflow/outflow
[✓ Inflow] Outflow
Choropleth of immigrants per country in year 2009
Number of Refugees
6
5
4
3
2
1
0
Drag the slider to change the year:
Select inflow/outflow
Inflow [✓ Outflow]
Choropleth of immigrants per country in year 2009
Number of Refugees
6
5
4
3
2
1
0

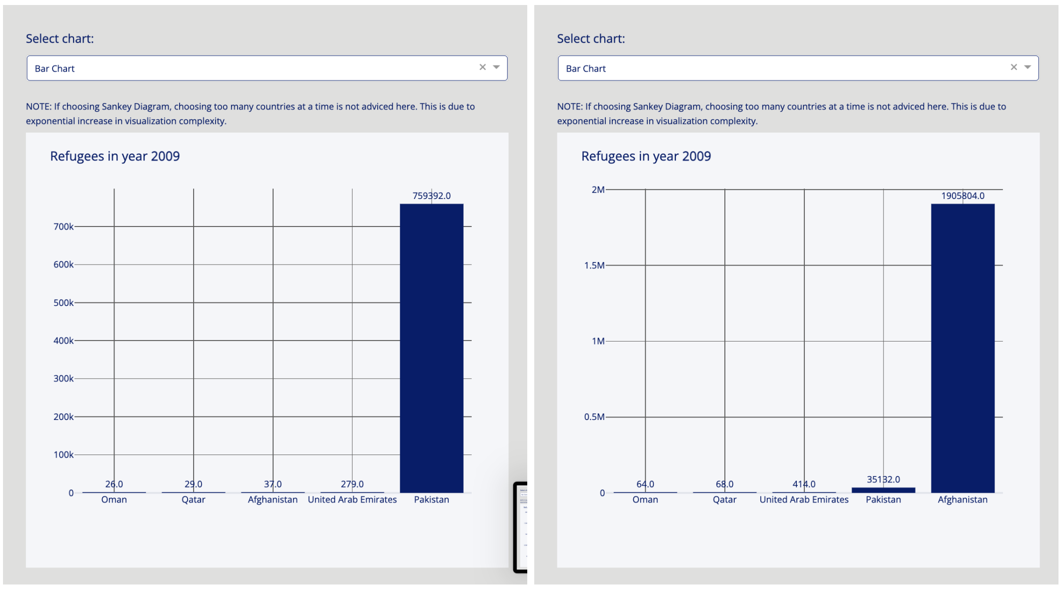
Figure 2: Inflow(Left) and outflow(Right) Choropleth Map for the year 2009



```python
@app.callback(
    Output("country-choropleth", "figure"),
    [Input("years-slider", "value"), Input("chart-type", "value")],
    [State("country-choropleth", "figure")],
)

def display_map(year, type ,figure):
    if (type == "in"):
        df_temp = df_refugees[df_refugees['Year'] == year]
        val = df_temp['Destination']
    else:
        df_temp = df_refugees_out[df_refugees_out['Year'] == year]
        val = df_temp['Origin']
    fig = go.Figure(data=go.Choropleth(
    locations = df_temp['CODE'],
    z = np.log10(df_temp['Refugees']),
    zmin = 0,
    zmax = 6.6,
    text = val,
    colorscale = 'tealrose',
    autocolorscale=False,
    reversescale=True,
    marker_line_color="#001E6C",        #mint
    marker_line_width=0.2,
    colorbar_title = 'Number of<br>Refugees',
))
```

Figure 3: Code snippet for Choropleth Map


Select chart:

Bar Chart

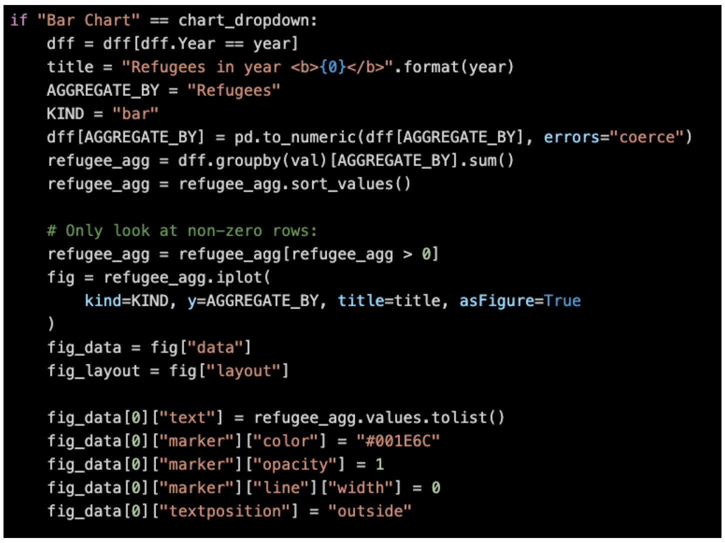
NOTE: If choosing Sankey Diagram, choosing too many countries at a time is not adviced here. This is due to exponential increase in visualization complexity.

Select chart:

Bar Chart

NOTE: If choosing Sankey Diagram, choosing too many countries at a time is not adviced here. This is due to exponential increase in visualization complexity.

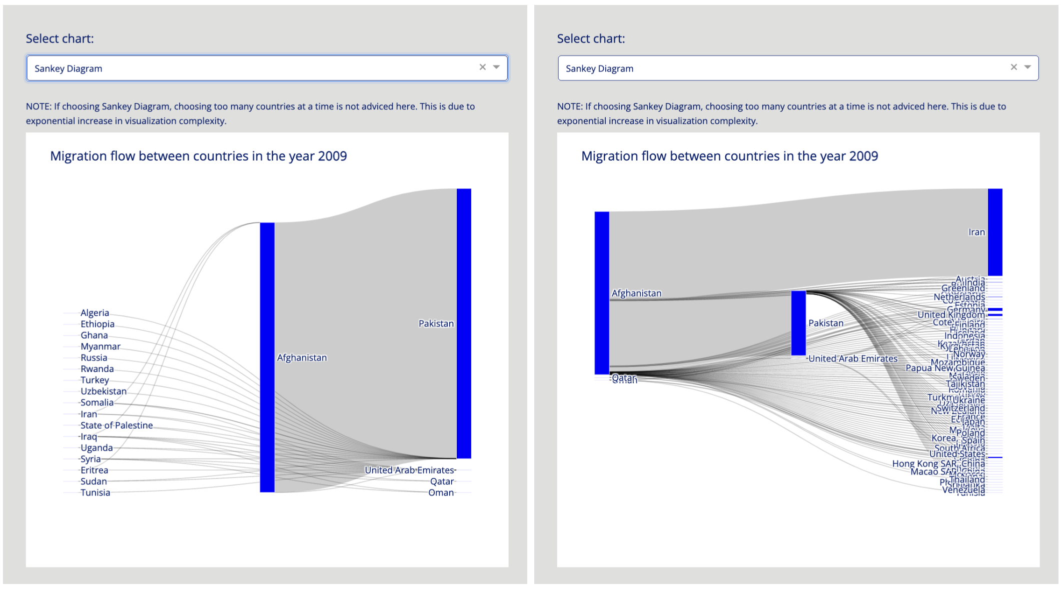
Figure 4: Inflow(Left) and outflow(Right) Bar Chart for the year 2009, user selected regions : (Oman, Qatar, Afghanistan, UAE, Pakistan)


```python
if "Bar Chart" == chart_dropdown:
    dff = dff[dff.Year == year]
    title = "Refugees in year <b>{0}</b>".format(year)
    AGGREGATE_BY = "Refugees"
    KIND = "bar"
    dff[AGGREGATE_BY] = pd.to_numeric(dff[AGGREGATE_BY], errors="coerce")
    refugee_agg = dff.groupby(val)[AGGREGATE_BY].sum()
    refugee_agg = refugee_agg.sort_values()

    # Only look at non-zero rows:
    refugee_agg = refugee_agg[refugee_agg > 0]
    fig = refugee_agg.iplot(
        kind=KIND, y=AGGREGATE_BY, title=title, asFigure=True
    )
    fig_data = fig["data"]
    fig_layout = fig["layout"]

    fig_data[0]["text"] = refugee_agg.values.tolist()
    fig_data[0]["marker"]["color"] = "#001E6C"
    fig_data[0]["marker"]["opacity"] = 1
    fig_data[0]["marker"]["line"]["width"] = 0
    fig_data[0]["textposition"] = "outside"
```

Figure 5: Code snippet for Bar Chart


Select chart:

Sankey Diagram

NOTE: If choosing Sankey Diagram, choosing too many countries at a time is not adviced here. This is due to exponential increase in visualization complexity.

Figure 6: Inflow(Left) and outflow(Right) Sankey Diagram for the year 2009, user selected regions : (Oman, Qatar, Afghanistan, UAE, Pakistan)

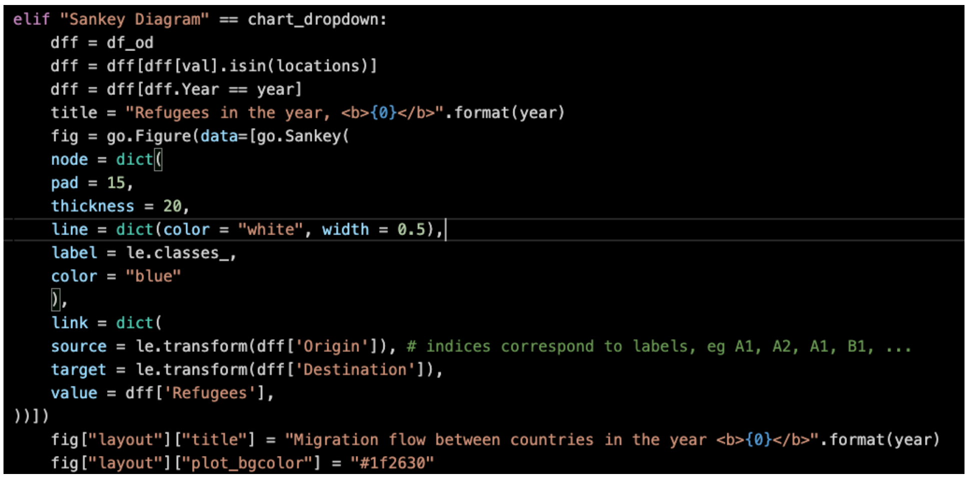
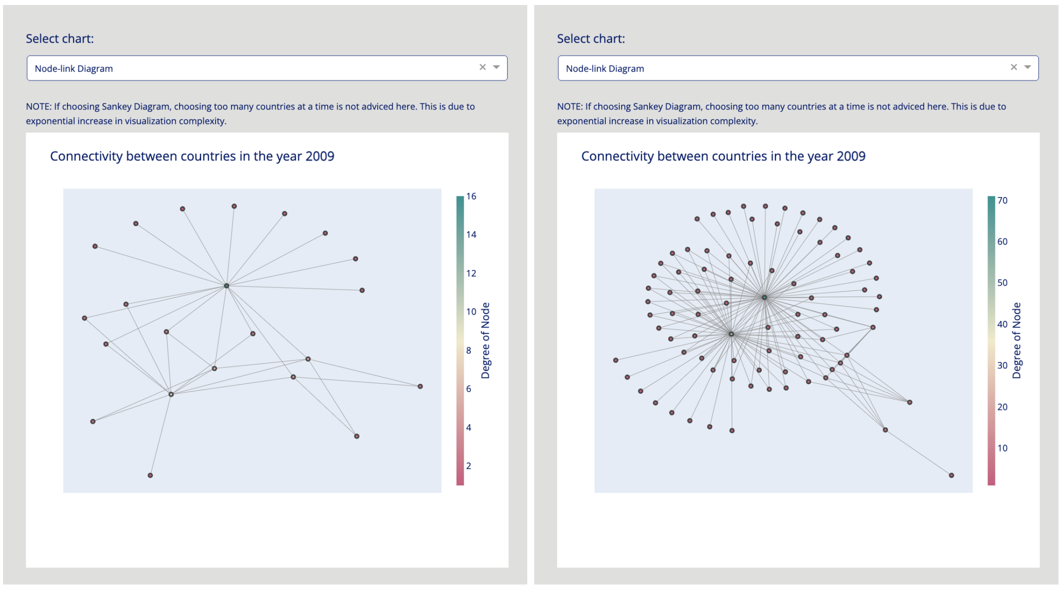

```
elif "Sankey Diagram" == chart_dropdown:
    dff = df_od
    dff = dff[dff[val].isin(locations)]
    dff = dff[dff.Year == year]
    title = "Refugees in the year, <b>{0}</b>".format(year)
    fig = go.Figure(data=[go.Sankey(
    node = dict(
    pad = 15,
    thickness = 20,
    line = dict(color = "white", width = 0.5),
    label = le.classes_,
    color = "blue"
    ),
    link = dict(
    source = le.transform(dff['Origin']), # indices correspond to labels, eg A1, A2, A1, B1, ...
    target = le.transform(dff['Destination']),
    value = dff['Refugees'],
))])
    fig["layout"]["title"] = "Migration flow between countries in the year <b>{0}</b>".format(year)
    fig["layout"]["plot_bgcolor"] = "#1f2630"
```

Figure 7: Code snippet for Sankey Diagram


Select chart:

Node-link Diagram

NOTE: If choosing Sankey Diagram, choosing too many countries at a time is not adviced here. This is due to exponential increase in visualization complexity.

Connectivity between countries in the year 2009

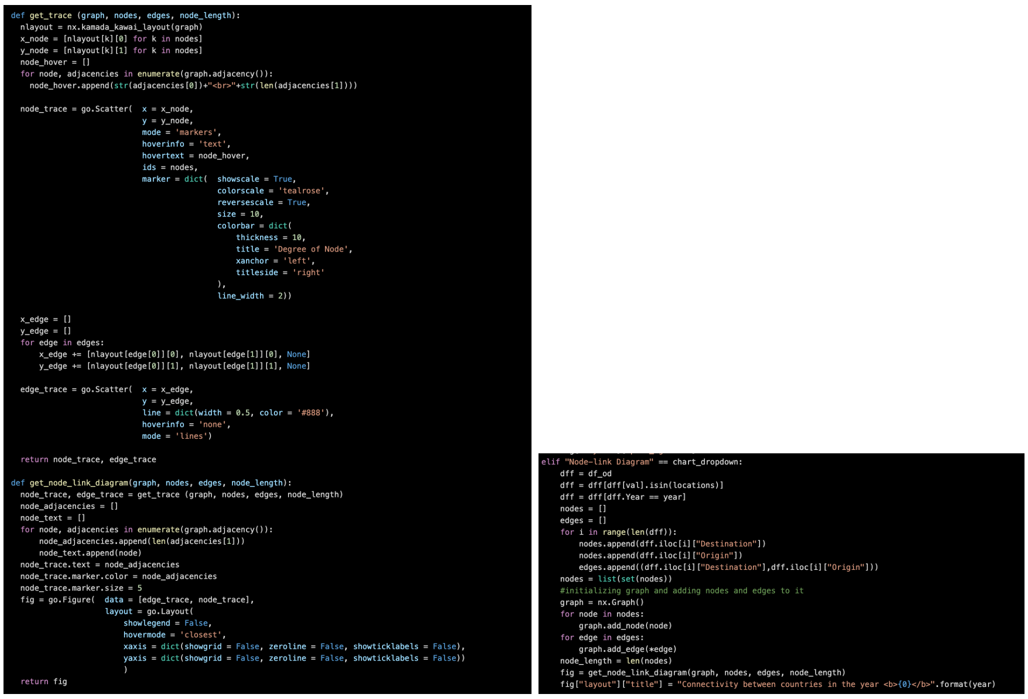
Select chart:

Node-link Diagram

NOTE: If choosing Sankey Diagram, choosing too many countries at a time is not adviced here. This is due to exponential increase in visualization complexity.

Connectivity between countries in the year 2009

Figure 8: Inflow(Left) and outflow(Right) Node-Link Diagram for the year 2009, user selected regions : (Oman, Qatar, Afghanistan, UAE, Pakistan)



```python
def get_trace (graph, nodes, edges, node_length):
  nlayout = nx.kamada_kawai_layout(graph)
  x_node = [nlayout[k][0] for k in nodes]
  y_node = [nlayout[k][1] for k in nodes]
  node_hover = []
  for node, adjacencies in enumerate(graph.adjacency()):
    node_hover.append(str(adjacencies[0])+"<br>"+str(len(adjacencies[1])))

  node_trace = go.Scatter(  x = x_node,
                            y = y_node,
                            mode = 'markers',
                            hoverinfo = 'text',
                            hovertext = node_hover,
                            ids = nodes,
                            marker = dict(  showscale = True,
                                            colorscale = 'tealrose',
                                            reversescale = True,
                                            size = 10,
                                            colorbar = dict(
                                                thickness = 10,
                                                title = 'Degree of Node',
                                                xanchor = 'left',
                                                titleside = 'right'
                                            ),
                                            line_width = 2))

  x_edge = []
  y_edge = []
  for edge in edges:
      x_edge += [nlayout[edge[0]][0], nlayout[edge[1]][0], None]
      y_edge += [nlayout[edge[0]][1], nlayout[edge[1]][1], None]

  edge_trace = go.Scatter(  x = x_edge,
                            y = y_edge,
                            line = dict(width = 0.5, color = '#888'),
                            hoverinfo = 'none',
                            mode = 'lines')

  return node_trace, edge_trace

def get_node_link_diagram(graph, nodes, edges, node_length):
  node_trace, edge_trace = get_trace (graph, nodes, edges, node_length)
  node_adjacencies = []
  node_text = []
  for node, adjacencies in enumerate(graph.adjacency()):
      node_adjacencies.append(len(adjacencies[1]))
      node_text.append(node)
  node_trace.text = node_adjacencies
  node_trace.marker.color = node_adjacencies
  node_trace.marker.size = 5
  fig = go.Figure(  data = [edge_trace, node_trace],
                    layout = go.Layout(
                        showlegend = False,
                        hovermode = 'closest',
                        xaxis = dict(showgrid = False, zeroline = False, showticklabels = False),
                        yaxis = dict(showgrid = False, zeroline = False, showticklabels = False))
                        )
  return fig
```

```python
elif "Node-link Diagram" == chart_dropdown:
    dff = df_od
    dff = dff[dff[val].isin(locations)]
    dff = dff[dff.Year == year]
    nodes = []
    edges = []
    for i in range(len(dff)):
        nodes.append(dff.iloc[i]["Destination"])
        nodes.append(dff.iloc[i]["Origin"])
        edges.append((dff.iloc[i]["Destination"],dff.iloc[i]["Origin"]))
    nodes = list(set(nodes))
    #initializing graph and adding nodes and edges to it
    graph = nx.Graph()
    for node in nodes:
        graph.add_node(node)
    for edge in edges:
        graph.add_edge(*edge)
    node_length = len(nodes)
    fig = get_node_link_diagram(graph, nodes, edges, node_length)
    fig["layout"]["title"] = "Connectivity between countries in the year <b>{0}</b>".format(year)
```

Figure 9: Code snippets for Node-link Diagram

5 GUI features

This dashboard allows 5 types of user interactions:

1. ***Slider for selecting year*** : This slider allows the user to select the year of his/her interest and the corresponding visualizations are updated accordingly(Fig. 10).

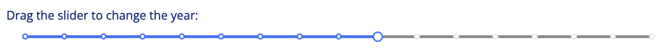


Figure 10: Select year slider

2. ***Select inflow/outflow*** : This radio button selection allows user to choose to study the inflow or outflow of people from countries (Fig. 11).

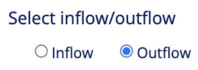


Figure 11: Select inflow or outflow radio button

3. ***Selecting regions of interest*** : This functionality allows the user to select specific regions of interest in the choropleth (box select or lasso select) and do further analysis using the other charts. Other visualizations get updated based on this selection (Fig. 12).

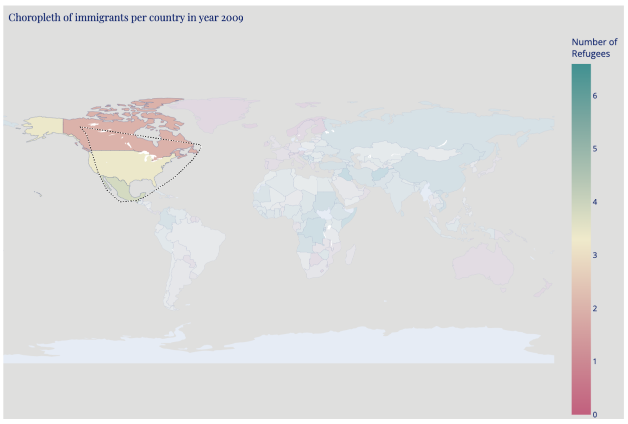


Figure 12: Selecting regions of interest using lasso select, year 2009, outflow ,user selected regions:(United States, Canada, Mexico)

4. ***Drop-down for selecting chart type*** : This drop-down allows users to select the visualisation of their choice (Bar, Sankey or Node-link) (Fig. 13).

Apart from above mentioned interactions, there is also a hover functionality for all the visualizations in the dashboard and zoom functionality for the choropleth to enable easy selection.

6 User Manual

6.1 Running locally

In order to run the web app locally, the user should clone the source code from here. Then:

1. User should install pip and then proceed to run the requirements.txt file using the command prompt **pip install -r requirements.txt**

2. After installation completes successfully, user should run the command **python -m flask run** to launch the web app locally.

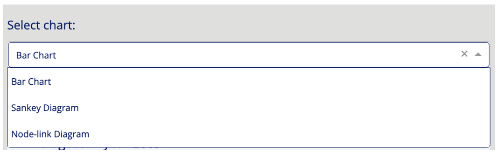


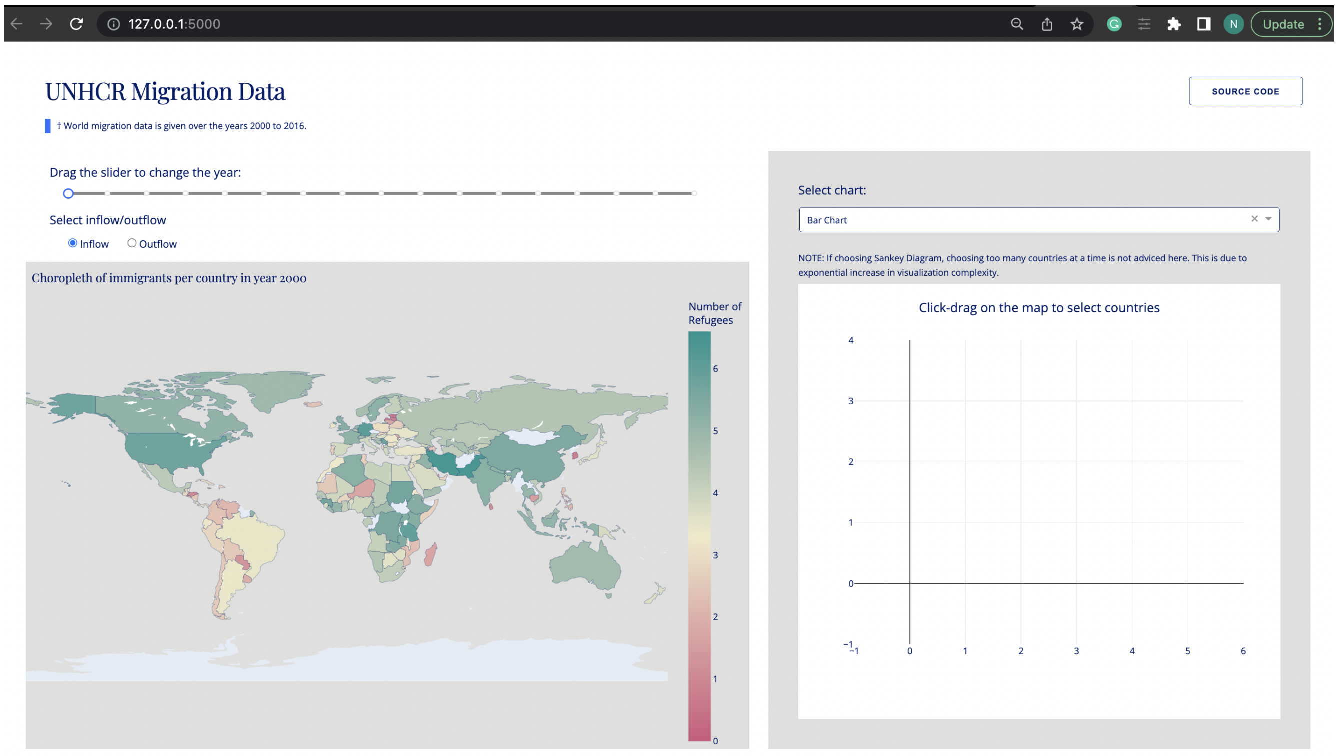
Figure 13: Select chart drop-down

6.2 Navigating the dashboard

To get familiar with the dashboard, the user can get started by following below-mentioned steps:

- First, the user can choose the year using the slider and select either inflow or outflow. The choropleth gets updated accordingly.
- Then, user can proceed to select their region of interest using either the box select/lasso select tool provided at the top of the visualization.
- After selecting this region, user can select their visualization chart from the select chart drop-down based on their exploration interests.
- User can repeat the above mentioned steps to get the visualisations of their choice.

7 Screenshots and detailed captions


UNHCR Migration Data

† World migration data is given over the years 2000 to 2016.

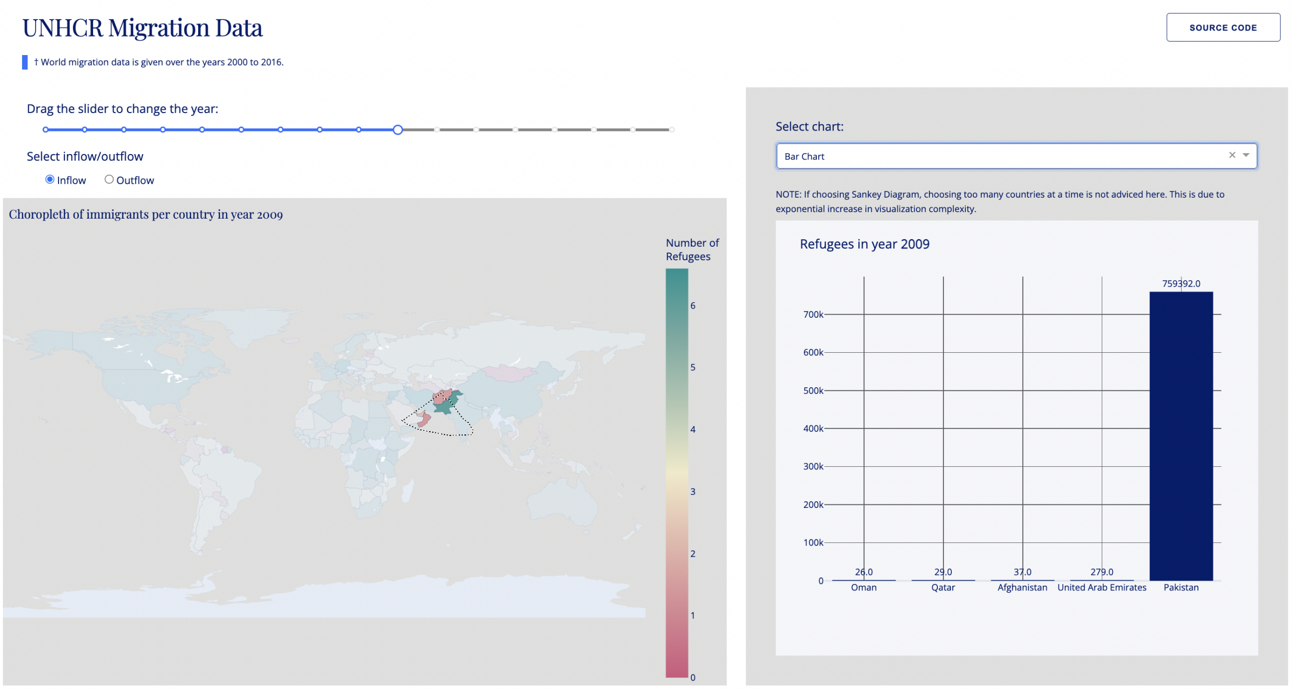
NOTE: If choosing Sankey Diagram, choosing too many countries at a time is not adviced here. This is due to exponential increase in visualization complexity.

127.0.0.1:5000
Update
SOURCE CODE
Drag the slider to change the year:
Select inflow/outflow
Inflow
Outflow
Choropleth of immigrants per country in year 2000
Number of Refugees
Select chart:
Bar Chart
Click-drag on the map to select countries

Figure 14: Dashboard starting page

8 References

1. International Migration 2020 Highlights


UNHCR Migration Data

SOURCE CODE

† World migration data is given over the years 2000 to 2016.

Drag the slider to change the year:

Select inflow/outflow

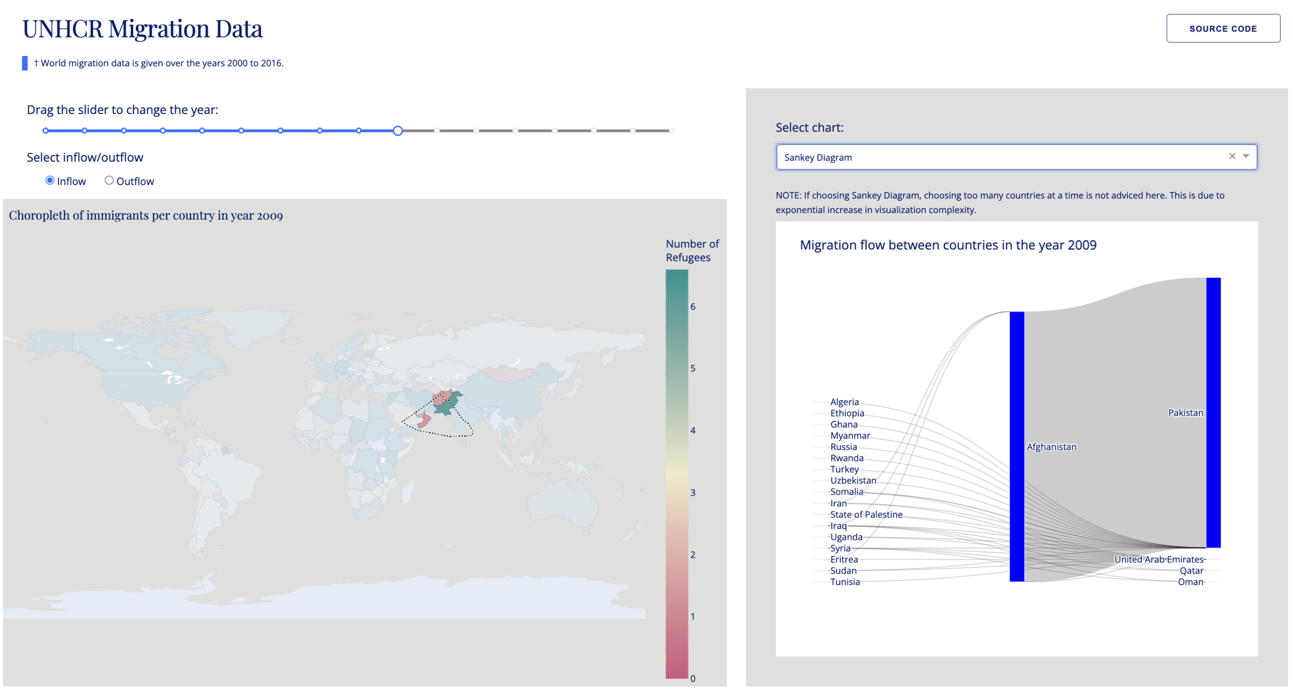
[✓ Inflow] Outflow

Choropleth of immigrants per country in year 2009

Select chart: Bar Chart

NOTE: If choosing Sankey Diagram, choosing too many countries at a time is not adviced here. This is due to exponential increase in visualization complexity.

Figure 15: Bar Chart for the year 2009, inflow, user selected regions : (Oman, Qatar, Afghanistan, UAE, Pakistan)


UNHCR Migration Data

SOURCE CODE

† World migration data is given over the years 2000 to 2016.

Drag the slider to change the year:

Select inflow/outflow

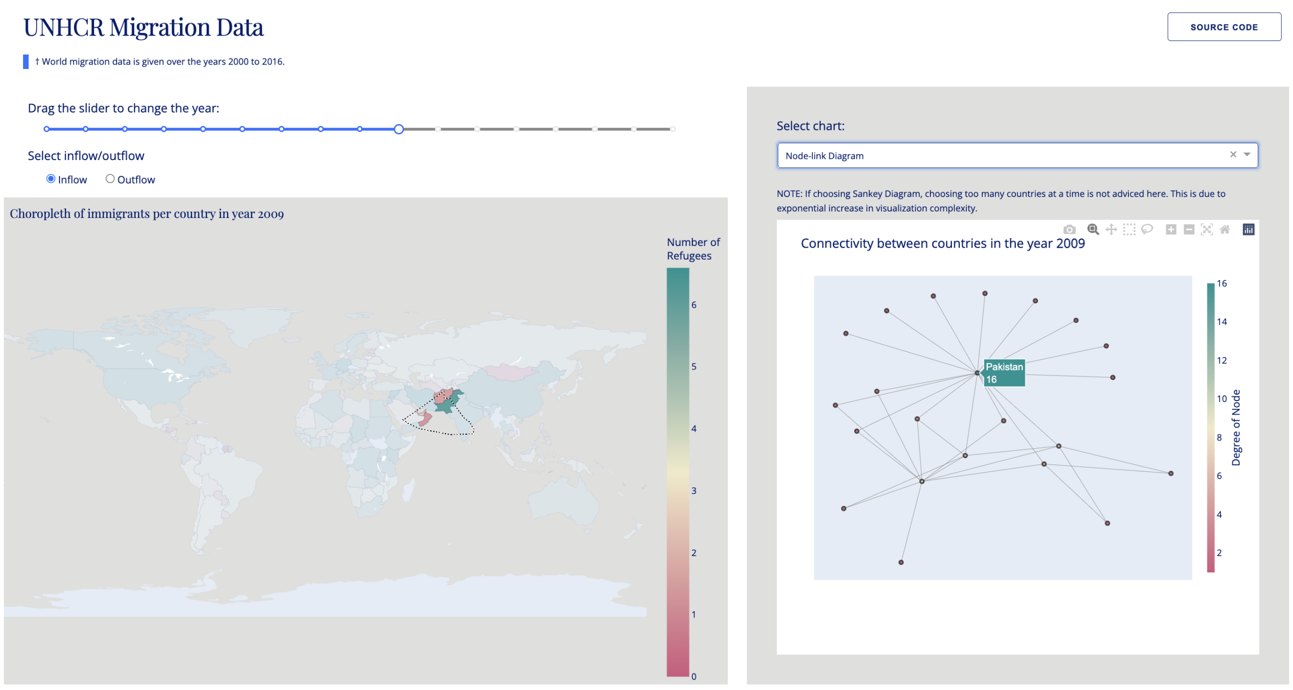
[✓ Inflow] Outflow

Choropleth of immigrants per country in year 2009

Select chart: Sankey Diagram

NOTE: If choosing Sankey Diagram, choosing too many countries at a time is not adviced here. This is due to exponential increase in visualization complexity.

Figure 16: Sankey Diagram for the year 2009, inflow, user selected regions : (Oman, Qatar, Afghanistan, UAE, Pakistan)


UNHCR Migration Data

† World migration data is given over the years 2000 to 2016.

Drag the slider to change the year:

Select inflow/outflow

[✓ Inflow] Outflow

Select chart: Node-link Diagram

NOTE: If choosing Sankey Diagram, choosing too many countries at a time is not adviced here. This is due to exponential increase in visualization complexity.

SOURCE CODE
Choropleth of immigrants per country in year 2009
Number of Refugees
0
1
2
3
4
5
6
Connectivity between countries in the year 2009
Pakistan
16
Degree of Node
2
4
6
8
10
12
14
16

Figure 17: Sankey Diagram for the year 2009, inflow, user selected regions : (Oman, Qatar, Afghanistan, UAE, Pakistan)

2. UNHCR Statistical Dataset
3. Plotly sample datasets
4. Preprocessing notebook (Kaggle)
5. Dash Web App