NetMigrationMap

June 28, 2022

0.1 Net Migration Map of Georgia by Citizenship

Tip: Hover the bars or different countries in the given choropleth map with your mouse in order to get a tooltip showing the relevant details.

[1]:
```
from IPython.display import display, HTML, IFrame

map_html = HTML(
    '<div style="position:relative;width:100%;height:0;padding-bottom:60%;"'
    '  id="net-migration-map-container" >')
map_html.data += IFrame(src="map.html", width='100%', height='100%', extras=[
    'style="position:absolute;width:100%;height:100%;left:0;top:0;'
    'border:none !important;"', 'allowfullscreen',
]).._repr_html_() + '</div>'
display(map_html)
```

```
<IPython.core.display.HTML object>
```

* Four bars (like a sandwitch) are placed in the Black Sea near the Georgia on the map above in order to reflect information about the data that is not associated with a citizenship of any particular country, such as: - **Stateless** bar: Shows net migration of stateless persons who do not have citizenship of any country. - **Not stated** bar: Shows net migration of persons whos citizenship is not stated in the data. - **Other** bar: Shows net migration of persons who were identified as citizens of some particular country other than the ones listed by the Geostat in the published data. - **Total** bar: Shows total net migration of Georgia over the given period of time which include all the migrants regardless of citizenship.

1 Net Migration of Georgia

Purpose of this notebook is to create a choropleth world map reflecting a net migration of Georgia by citizenship over long period of time (period of time depends on the data available). The resulting map is displayed above. Two different color scales are used, red and green, the red color scale is used for highlighting countries if the net migration in Georgia is a negative number for their respective citizens and the green color scale is used in a same way but for positive numbers. Color (red/green) and the color intensity are assigned automatically based on the relevant statistics.

Definitions: - **Immigrant** (as defined by the Geostat for the data used here) – "a person recorded when crossing the National border i) who entered the country and has cumulated a minimum of 183 days of residence in the country during the twelve following months; and ii) who was not usual resident of the country when entering the country which means that he spent at least a cumulate

duration of 183 days of residence outside the country during the twelve months before entering the country." - **Emigrant** (as defined by the Geostat for the data used here) – "a person recorded when crossing the National border and i) who crossed the border and left the country and has cumulated a minimum of 183 days of residence outside the country during the twelve following months; and ii) who was usual resident of the country when leaving the country which means that he spent at least a cumulate duration of 183 days of residence inside the country during the twelve months before leaving the country." - **Net migration** (as used in this notebook) – a difference between the number of immigrants and the number of emigrants during the given interval of time.

Data sources: - Statistical data of the Number of immigrants and emigrants of Georgia by sex and citizenship (in Excel format) is retrieved on Jun 9, 2022 from the website of the National Statistics Office of Georgia (Geostat): geostat.ge

- Statistical data of the Number of immigrants and emigrants of Georgia by sex and citizenship (in CSV format) is retrieved on Jun 11, 2022 from the "Statistics Database" website of the National Statistics Office of Georgia (Geostat): pc-axis.geostat.ge
- Metadata (in PDF format) for the given statistical data is retrieved on Jun 14, 2022 from the website of the National Statistics Office of Georgia (Geostat): geostat.ge
- Map data of the borders of the countries *(1:10m Cultural Vectors - Admin 0 – Countries - (latest) version 5.1.1)* is retrieved on Jun 11, 2022 from the "Natural Earth" website: naturalearthdata.com

[2]:
```
from datetime import datetime, timedelta
nb_st = datetime.utcnow()
print(f"\nNotebook START time: {nb_st} UTC\n")
```

```
Notebook START time: 2022-06-28 10:41:12.015740 UTC
```

[3]:
```
%%HTML
<script>
  function code_toggle() {
    if (code_shown){
      $('div.input').hide('500');
      $('#toggleButton').val('Show Python Code')
    } else {
      $('div.input').show('500');
      $('#toggleButton').val('Hide Python Code')
    }
    code_shown = !code_shown
  }

  $( document ).ready(function(){
    code_shown=false;
    $('div.input').hide();
    $('div.input:contains("%%HTML")').removeClass( "input")
    $('div.input:contains("%%capture")').removeClass("input")
  });
```

```
</script>
<form action="javascript:code_toggle()">
  <input type="submit" id="toggleButton" value="Show Python Code">
</form>
```

```
<IPython.core.display.HTML object>
```

[4]:
```
import numpy as np
import pandas as pd
import geopandas
import warnings
import io
import json
import folium
import branca.colormap as cmp
from folium.plugins import Fullscreen
from folium.utilities import normalize
```

[5]:
```
VERBOSE = False
```

[6]:
```
migration_df = pd.read_csv(
    "data/geostat/EN/CSV/Migration.csv",
    skiprows=2,
    index_col="citizenship"
).head(-2).astype(int)

migration_df = migration_df.T.assign(
    Year=lambda df: pd.Series(
        [
            str(x).strip().split()[0]
            for x in df.index
        ],
        index=df.index
    ).astype(int),
    MigrantType=lambda df: pd.Series(
        [
            str(x).strip().split()[1].strip()
            for x in df.index
        ],
        index=df.index
    ).replace({
        'Immigrants': 'Immigrant',
        'Emigrants': 'Emigrant'
    }).astype('category'),
    Sex=lambda df: pd.Series(
        [
            str(x).strip().split()[-1].strip()
            for x in df.index
```

```
        ],
        index=df.index
    ).replace({
        'Males': 'Male',
        'Females': 'Female',
        'sexes': 'All'
    }).astype('category')
)

start_year, end_year = migration_df['Year'].min(), migration_df['Year'].max()

print("\nGiven migration data covers the interval of time"
      f" from {start_year} to {end_year} (inclusive).\n")

if VERBOSE:
    display(migration_df)
```

```
Given migration data covers the interval of time from 2012 to 2021 (inclusive).
```

[7]:
```
NOT_COUNTRY_NAMES = ('Stateless', 'Not stated', 'Other', 'Total',)

def get_net_migration_by_citizenship_df(df: pd.DataFrame,
                                        Sex: str = 'All') -> pd.DataFrame:
    assert Sex in df['Sex'].cat.categories, (
        'Data not found for the Sex="{Sex}" filter, available categories are: '
        f'{", ".join(list(df["Sex"].cat.categories))}.')
    df = pd.DataFrame({
        'Immigrant': df.loc[
            (df['Sex'] == Sex) & (df['MigrantType'] == 'Immigrant'),
            df.columns.difference(['Year'])
        ].sum(numeric_only=True),
        'Emigrant': df.loc[
            (df['Sex'] == Sex) & (df['MigrantType'] == 'Emigrant'),
            df.columns.difference(['Year'])
        ].sum(numeric_only=True)
    })
    df = (df['Immigrant'] - df['Emigrant']).reset_index(name='NetMigration')
    df = pd.concat([
        df.loc[
            ~df['citizenship'].isin(NOT_COUNTRY_NAMES)
        ].sort_values(by='NetMigration', ascending=False),
        df.loc[
            df['citizenship'].isin(NOT_COUNTRY_NAMES)
        ].sort_values(by='NetMigration', ascending=False)
    ]).reset_index(drop=True)
```

```
    return df
```

[8]:
```
net_migration = {}

for Sex in ("Female", "Male", "All"):
    net_migration[Sex] = get_net_migration_by_citizenship_df(migration_df,
                                                             Sex=Sex)

    if VERBOSE:
        print(f"\n{start_year}-{end_year} Net Migration of Georgia "
              f"by citizenship for sex=\"{Sex}\":\n")
        display(net_migration[Sex])
```

[9]:
```
if VERBOSE:

    for Sex in ("Female", "Male", "All"):
        print('\nDescriptive statistics of the '
              f'full {start_year}-{end_year} '
              f'Net Migration data for Sex="{Sex}":\n')
        display(
            net_migration[Sex].loc[
                net_migration[Sex]['citizenship'] != 'Total'
            ].describe()
        )
        print('\nDescriptive statistics of the positive values '
              'having identifiable citizenship '
              f'from the {start_year}-{end_year} Net Migration data '
              f'for Sex="{Sex}":\n')
        display(
            net_migration[Sex].loc[
                (net_migration[Sex]['NetMigration'] > 0) &
                (~net_migration[Sex]['citizenship'].isin(NOT_COUNTRY_NAMES)),
                'NetMigration'
            ].describe()
        )
        print('\nDescriptive statistics of the negative values '
              'having identifiable citizenship '
              f'from the {start_year}-{end_year} Net Migration data '
              f'for Sex="{Sex}":\n')
        display(
            net_migration[Sex].loc[
                (net_migration[Sex]['NetMigration'] < 0) &
                (~net_migration[Sex]['citizenship'].isin(NOT_COUNTRY_NAMES)),
                'NetMigration'
            ].describe()
        )
        print('\n\n\n')
```

[10]:
```
countries_geodf = geopandas.read_file(
    'data/naturalearth/ne_10m_admin_0_countries/ne_10m_admin_0_countries.shp'
)
```

[11]:
```
def number_of_mismatched_names(
        df: pd.DataFrame,
        geodf: geopandas.GeoDataFrame = countries_geodf) -> int:
    mismatch = df.loc[
        (~df['citizenship'].isin(geodf['NAME'])) &
        (~df['citizenship'].isin(NOT_COUNTRY_NAMES)),
        'citizenship'
    ]
    N = len(mismatch)

    if VERBOSE:
        print('Number of mismatched names between map data and '
              f'statistic data is: {N}')

        if N > 0:
            print(f'Mismatched names: {[name for name in mismatch]}')
    return N
```

[12]:
```
number_of_mismatched_names(net_migration["All"])
pass
```

[13]:
```
def search_in_countries_geodf(substring: str) -> pd.Series:
    return countries_geodf.loc[countries_geodf['NAME'].str.contains(substring),
                               'NAME']

if VERBOSE:
    print("Searching for equivalents used by the map data "
          "for the mismatched names detected above: ")
    print(search_in_countries_geodf('Russ'))
    print(search_in_countries_geodf('Iran'))
```

[14]:
```
def fix_mismatched_names(df: pd.DataFrame) -> pd.DataFrame:
    return df.replace({
        'Russian Federation': 'Russia',
        'Iran, Islamic Republic of': 'Iran',
    })


for Sex in ("Female", "Male", "All"):
    net_migration[Sex] = fix_mismatched_names(net_migration[Sex])

    if VERBOSE:
        print('Replaced mismatched names in the data '
```

```
            f'for Sex="{Sex}":\n - ', end="")
    assert number_of_mismatched_names(net_migration[Sex]) == 0, "ERROR:\
 Please resolve name mismatch first..."
```

[15]:
```
with io.BytesIO() as buffer:
    with warnings.catch_warnings():
        warnings.filterwarnings("ignore", category=FutureWarning)
        countries_geodf.to_file(buffer, driver='GeoJSON')
    countries_geojson = json.loads(buffer.getvalue().decode("utf-8"))

with open("black_sea_sandwitch.json") as f:
    sandwitch_geojson = json.load(f)

countries_geojson['features'] += sandwitch_geojson['features']
```

[16]:
```
min_value = min([net_migration[Sex]['NetMigration'].min()
                 for Sex in ("Female", "Male", "All")])

max_value = max([net_migration[Sex]['NetMigration'].max()
                 for Sex in ("Female", "Male", "All")])

color_scale = cmp.LinearColormap(
    ['red', '#ffcccd', '#ccffcd', 'green'],
    index=[min_value, 0, 0, max_value],
    vmin=min_value, vmax=max_value,
    # caption=f'{start_year}-{end_year} Net Migration of Georgia by Citizenship'
)


def get_map_color(NAME: str, Sex: str) -> str:
    x = net_migration[Sex].loc[net_migration[Sex]['citizenship']==NAME,
                               'NetMigration']
    return color_scale(x.item()) if len(x) else '#000000'


if VERBOSE:
    print("Color scale:")
    display(color_scale)
```

[17]:
```
def get_net_migration_tooltip_by_citizenship(citizenship: str) -> str:
    tooltip = (f'&amp;quot;{start_year}-{end_year} '
               'Net Migration of Georgia&amp;quot;<br>')

    n_female, n_male,  n_all = [
        (lambda df: df.loc[df['citizenship']==citizenship,
                           'NetMigration'])(
            net_migration[Sex])
```

```
        for Sex in ("Female", "Male", "All")
    ]

    if citizenship in NOT_COUNTRY_NAMES:
        tooltip += {
            'Stateless': '<strong>Stateless persons</strong>',
            'Not stated': 'Citizenship <strong>not stated</strong>',
            'Other': '<strong>Other</strong> (citizenship not given)',
            'Total': '<strong>Total Net Migration of Georgia</strong>',
        }[citizenship]
    else:
        tooltip += ("Migrated citizens of "
                    f"<strong>{citizenship}</strong> in Georgia")
    tooltip += ': <br>'

    if len(n_all) == 0:
        tooltip += "Not given (included in \"Other\")"
    else:
        tooltip += f"<strong>{n_all.item():+}</strong> "
        n_male = n_male.item() if len(n_male) else 0
        n_female = n_female.item() if len(n_female) else 0

        if abs(n_male) > abs(n_female):
            tooltip += f"(Male: {n_male:+}, Female: {n_female:+})"
        else:
            tooltip += f"(Female: {n_female:+}, Male: {n_male:+})"
    return tooltip


if VERBOSE:
    print("\nExamples of tooltips:\n")
    display(HTML("<p>{}</p><p>{}</p><br>".format(
        get_net_migration_tooltip_by_citizenship("Georgia"),
        get_net_migration_tooltip_by_citizenship("Antarctica"))))
```

[18]:
```
tbilisi_coordinate = [41.69339329182433, 44.80151746492941]
m = folium.Map(location=tbilisi_coordinate, zoom_start=6)

color_scale.add_to(m)

Sex = (
    "Female",  # 0
    "Male",  # 1
    "All"  # 2
)[2]

geojson = countries_geojson.copy()
```

```
m_layer = folium.GeoJson(
    geojson,
    style_function=lambda feature: {
        'fillColor': get_map_color(feature['properties']['NAME'], Sex),
        'color': 'black',  # border color
        'weight': 1,  # border thikness
        # 'dashArray': '5, 3',  # dashed 'line length,space length'
        'fillOpacity': 0.7,
        'nanFillOpacity': 0.4,
        'lineOpacity': 0.2,
    },
    name='Net Migration of Georgia by citizenship' + (
        f' (Sex="{Sex}")' if Sex!="All" else ''
    ),
    # zoom_on_click=True,
    # show=Sex=="All",
)

for feature in geojson['features']:
    feature['properties'][
        'tooltip_msg'
    ] = get_net_migration_tooltip_by_citizenship(feature['properties']['NAME'])
folium.GeoJsonTooltip(
    fields=["tooltip_msg"],
    labels=False
).add_to(m_layer)

for feature in sandwitch_geojson['features']:
    coord = feature['geometry']['coordinates'][0][3]
    folium.map.Marker(
        [coord[1], coord[0]],
        icon=folium.DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html=('<div style="font-size: inherit; color:#333333; '
                  'white-space:nowrap;"><b>{:s}</b></div>').format(
                      feature['properties']['NAME']),
            class_name="div-icon-text"
        )
    ).add_to(m_layer)
m_layer.add_to(m)



m.get_root().html.add_child(folium.Element("""
<style>
@media (max-width: 500px) {
```

```
    .leaflet-right .legend {
        visibility: hidden;
    }
}
@media (max-width: 288px) {
    .leaflet-right {
        display: none;
    }
}
</style>
"""))
m.get_root().html.add_child(folium.Element(f"""
<script type="text/javascript">
window.onload = () => {{
    let zoomText = () => {{
        $(".div-icon-text")
            .css("font-size", (0.02 * {m.get_name()}.getZoom()**2) + "em");
    }};
    zoomText();
    {m.get_name()}.on("zoomend", () => {{zoomText();}});
}};
</script>
"""))
folium.LayerControl().add_to(m)
Fullscreen().add_to(m)


# display(m)
pass
```

[19]:
```
##
# Instead of displaying here save the map as static `map.html` file
# in order to display in the beggining of this notebook.
# FIXME: This approach may require 2x run of notebook to update the map
##
m.save("map.html")
```

[20]:
```
print(f"\n ** Total Elapsed time: {datetime.utcnow() - nb_st} ** \n")
print(f"Notebook END time: {datetime.utcnow()} UTC\n")
```

```
 ** Total Elapsed time: 0:00:08.673397 **

Notebook END time: 2022-06-28 10:41:20.689200 UTC
```

[21]:
```
%%capture
%mkdir OGP_classic
```

[22]:
```
%%capture
%%file "OGP_classic/conf.json"
{
  "base_template": "classic",
  "preprocessors": {
    "500-metadata": {
      "type": "nbconvert.preprocessors.ClearMetadataPreprocessor",
      "enabled": true,
      "clear_notebook_metadata": true,
      "clear_cell_metadata": true
    },
    "900-files": {
      "type": "nbconvert.preprocessors.ExtractOutputPreprocessor",
      "enabled": true
    }
  }
}
```

[23]:
```
%%capture
%%file "OGP_classic/index.html.j2"
{%- extends 'classic/index.html.j2' -%}
{%- block html_head -%}

{#  OGP attributes for shareability #}
<meta property="og:url"           content="https://sentinel-1.github.io/
 ↪net_migration_map_Georgia/" />
<meta property="og:type"          content="article" />
<meta property="og:title"         content="Net Migration Map of Georgia" />
<meta property="og:description"   content="Choropleth Map of the Net Migration␣
 ↪of Georgia by Citizenship" />
<meta property="og:image"         content="https://raw.githubusercontent.com/
 ↪sentinel-1/net_migration_map_Georgia/master/screenshots/
 ↪2022-06-28_(1200x628).png" />
<meta property="og:image:alt"     content="Screen Shot of the resulting map" />
<meta property="og:image:type"    content="image/png" />
<meta property="og:image:width"   content="1200" />
<meta property="og:image:height"  content="628" />

<meta property="article:published_time" content="2022-06-14T10:55:04+00:00" />
<meta property="article:modified_time"  content="{{ resources.
 ↪iso8610_datetime_now }}" />
<meta property="article:publisher"      content="https://sentinel-1.github.io" /
 ↪>
<meta property="article:author"         content="https://github.com/sentinel-1"␣
 ↪/>
<meta property="article:section"        content="datascience" />
<meta property="article:tag"            content="datascience" />
```

```
<meta property="article:tag"            content="geospatialdata" />
<meta property="article:tag"            content="Python" />
<meta property="article:tag"            content="data" />
<meta property="article:tag"            content="analytics" />
<meta property="article:tag"            content="datavisualization" />
<meta property="article:tag"            content="bigdataunit" />
<meta property="article:tag"            content="visualization" />
<meta property="article:tag"            content="migration" />
<meta property="article:tag"            content="Georgia" />


{{ super() }}

{%- endblock html_head -%}


{% block body_header %}
<body>

<div class="container">
  <nav class="navbar navbar-default">
    <div class="container-fluid">
      <ul class="nav nav-pills  navbar-left">
        <li role="presentation">
          <a href="/">
            <svg xmlns="http://www.w3.org/2000/svg"
                 viewBox="0 0 576 512" width="1em">
              <path
                fill="#999999"
d="M 288,0 574,288 511,288 511,511 352,511 352,352 223,352 223,511 62,511␣
 ↪64,288 0,288 Z"
              />
            </svg> Home
          </a>
        </li>
      </ul>
      <ul class="nav nav-pills  navbar-right">
        <li role="presentation" class="active">
          <a href="/net_migration_map_Georgia/">[] English </a>
        </li>
        <li role="presentation">
          <a href="/net_migration_map_Georgia/ka/">[]     </a>
        </li>
      </ul>
    </div>
  </nav>
</div>
```

```
  <div tabindex="-1" id="notebook" class="border-box-sizing">
    <div class="container" id="notebook-container">
{% endblock body_header %}

{% block body_footer %}
    </div>
  </div>
  <footer>
    <div class="container"
         style="display:flex; flex-direction: row; justify-content: center;␣
 ↪align-items: center;">
      <p style="margin: 3.7em auto;"> © 2022
        <a href="https://github.com/sentinel-1" target="_blank">Sentinel-1</a>
      </p>
      <!-- TOP.GE ASYNC COUNTER CODE -->
      <div id="top-ge-counter-container" data-site-id="116052"
           style="margin-right: 3.7em;float: right;"></div>
      <script async src="//counter.top.ge/counter.js"></script>
      <!-- / END OF TOP.GE COUNTER CODE -->
     </div>
  </footer>
</body>
{% endblock body_footer %}
```

This notebook is originally published under the Apache License (Version 2.0) at the following GitHub repository: sentinel-1/net_migration_map_Georgia

For the issues, feedback or suggestions regarding the original notebook (if any) feel free to open an issue at the corresponding Issues page of the repository