Topic No. 7
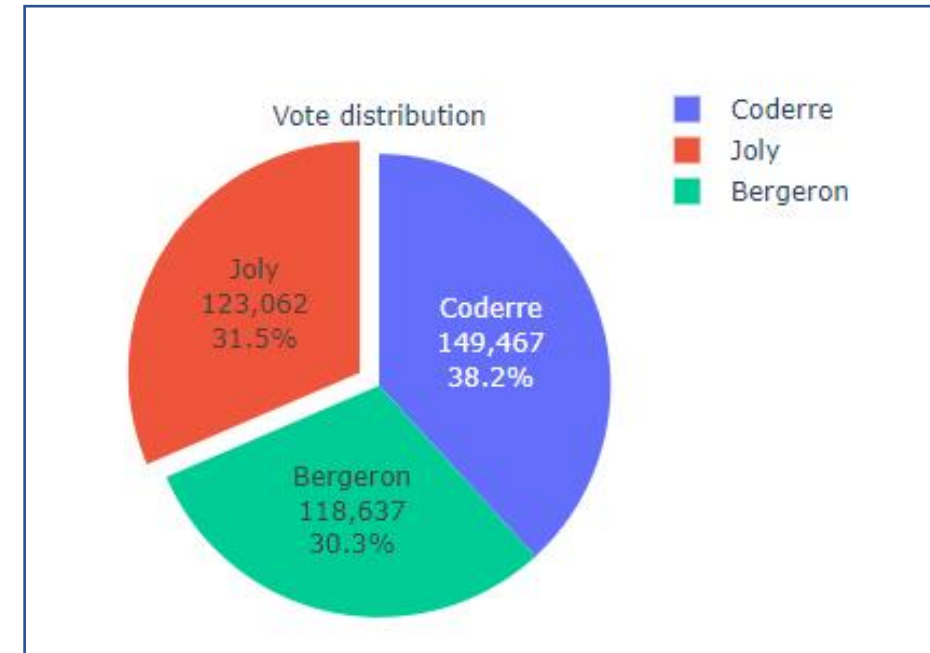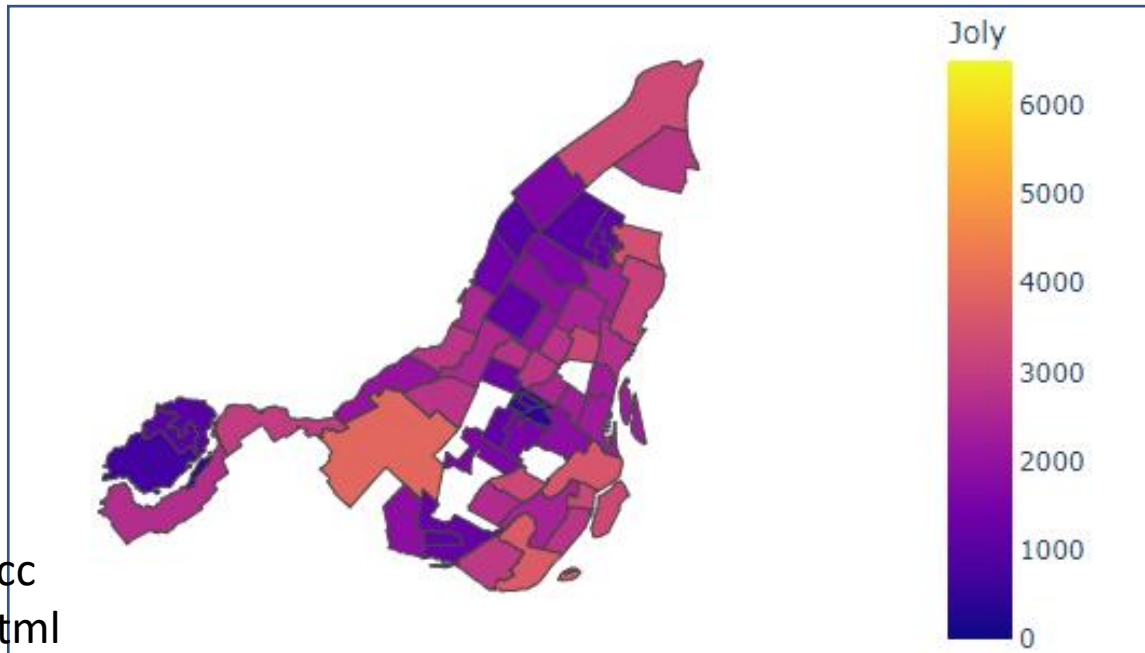
1. Web-based Dashboard Creation using Dash

2. Monte Carlo Method

Plotly Dash Components

An example dashboard built on Plotly Dash


```
import dash
from dash import dcc
from dash import html
from dash.dependencies import Input, Output
import plotly.express as px
import plotly.graph_objects as go
app = dash.Dash(__name__)
```

Data Structure

df = px.data.election()

geojson = px.data.election_geojson()

plotly.express.data. **election** ()

Each row represents voting results for an electoral district in the 2013 Montreal mayoral election.

Returns: ['district', 'Coderre', 'Bergeron', 'Joly', 'total', 'winner', 'result', 'district_id'].

Return type: A pandas.DataFrame with 58 rows and the following columns

district	Coderre	Bergeron	Joly	total	winner	result	district_id
101-Bois-de-Liesse	2481	1829	3024	7334	Joly	plurality	101
102-Cap-Saint-Jacques	2525	1163	2675	6363	Joly	plurality	102
11-Sault-au-RÃ©collet	3348	2770	2532	8650	Coderre	plurality	11
111-Mile-End	1734	4782	2514	9030	Bergeron	majority	111
112-DeLorimier	1770	5933	3044	10747	Bergeron	majority	112
113-Jeanne-Mance	1455	3599	2316	7370	Bergeron	plurality	113
12-Saint-Sulpice	3252	2521	2543	8316	Coderre	plurality	12
121-La Pointe-aux-Prairies	5456	1760	3330	10546	Coderre	majority	121
122-Pointe-aux-Trembles	4734	1879	2852	9465	Coderre	majority	122
123-RiviÃ¨re-des-Prairies	5737	958	1656	8351	Coderre	majority	123
13-Ahuntsic	2979	3430	2873	9282	Bergeron	plurality	13
131-Saint-Ã‰douard	1827	6408	2815	11050	Bergeron	majority	131
132-Ã‰tienne-Desmarteau	2331	5748	2788	10867	Bergeron	majority	132
133-Vieux-Rosemont	2670	4962	3234	10866	Bergeron	plurality	133
134-Marie-Victorin	3673	3155	2431	9259	Coderre	plurality	134
14-Bordeaux-Cartierville	3612	1554	2081	7247	Coderre	plurality	14
141-CÃ´te-de-Liesse	4308	1320	3959	9587	Coderre	plurality	141
142-Norman-McLaren	4104	1459	2822	8385	Coderre	plurality	142
151-Saint-LÃ©onard-Est	3931	882	1641	6454	Coderre	majority	151
152-Saint-LÃ©onard-Ouest	5387	1184	1908	8479	Coderre	majority	152

plotly.express.data. **election_geojson** ()

Each feature represents an electoral district in the 2013 Montreal mayoral election.

Returns: A GeoJSON-formatted dict with 58 polygon or multi-polygon features whose id is an electoral district numerical ID and whose district property is the ID and district name.

```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {
                "type": "MultiPolygon",
                "coordinates": [[[[-73.6363215300962, 45.5759177646435],
                    45.5647725775888], [-73.648867564748, 45.5586898267402],
                    45.5596724837829], [-73.6706609041685, 45.5610978251999],
                    45.5654269638586], [-73.663336397858, 45.5666288247853],
                    45.57083044563461], [-73.6620201425015, 45.5713925326191],
                    45.5748980132699], [-73.6639172423219, 45.5730041908097],
                    45.574385118162], [-73.6636711124334, 45.577018676761], [
                    45.5806752214364], [-73.659760079306, 45.5804007503503],
                    45.5780184952852], [-73.6617871566128, 45.5758213640561],
                    45.5793785624903], [-73.6570262958283, 45.5810509513563],
                    45.5826892716542], [-73.6363215300962, 45.5759177646435]]
                    45.585868003125], [-73.6552168328229, 45.5855392416017],
            },
            "properties": {
                "district": "11-Sault-au-R\u00e9collet"
            },
            "id": "11"
        }, {
            "type": "Feature",
            "geometry": {
                "type": "Polygon",
                "coordinates": [[[-73.6217484540132, 45.5544783077209], [
                    45.5503989164938], [-73.6325600663803, 45.5499795021129],
                    45.5441488267699], [-73.6407295659042, 45.5429686686779],
                    45.5277342274232], [-73.6698852350516, 45.5318835785726],
                    45.5321264506516], [-73.683124268159, 45.5305991300384],
                    45.5369404343695], [-73.6740837310465, 45.5378868187786],
                    45.5382590909533], [-73.6646235398403, 45.5394847807605],
                    45.5460790656787], [-73.6568659704249, 45.5450525873044],
                    45.5647725775888], [-73.6240413737876, 45.5555253903511],
            },
            "properties": {
                "district": "12-Saint-Sulpice"
            },
            "id": "12"
        }, {
            "type": "Feature", "geometry": {"type": "Polygon", "coordinat
```

Each component has unique Identifier (id)

id = 'candidate'

Candidate:

◉ Joly ○ Coderre ○ Bergeron
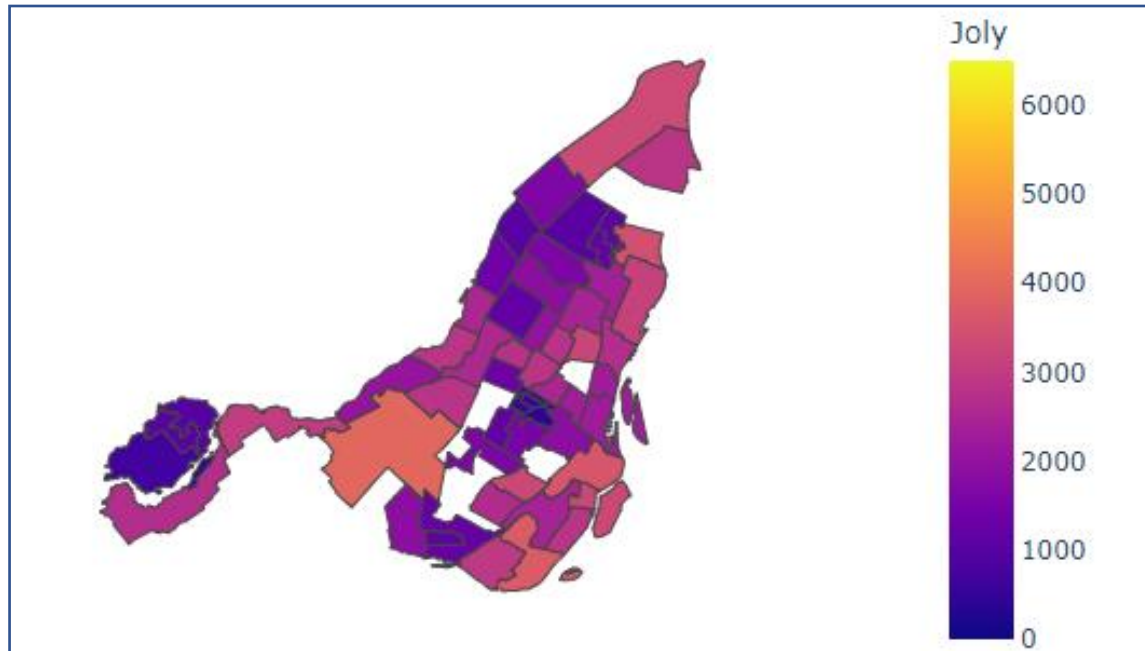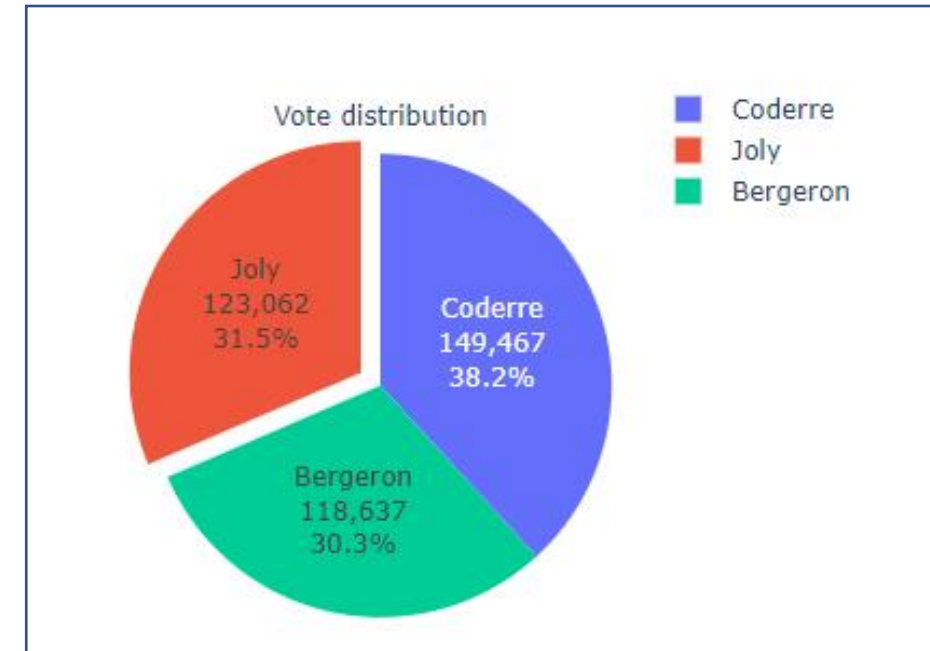
id = 'map'

id = 'piechart'

Each component has unique Identifier (id)

id = 'candidate'

Candidate:

◉ Joly ○ Coderre ○ Bergeron
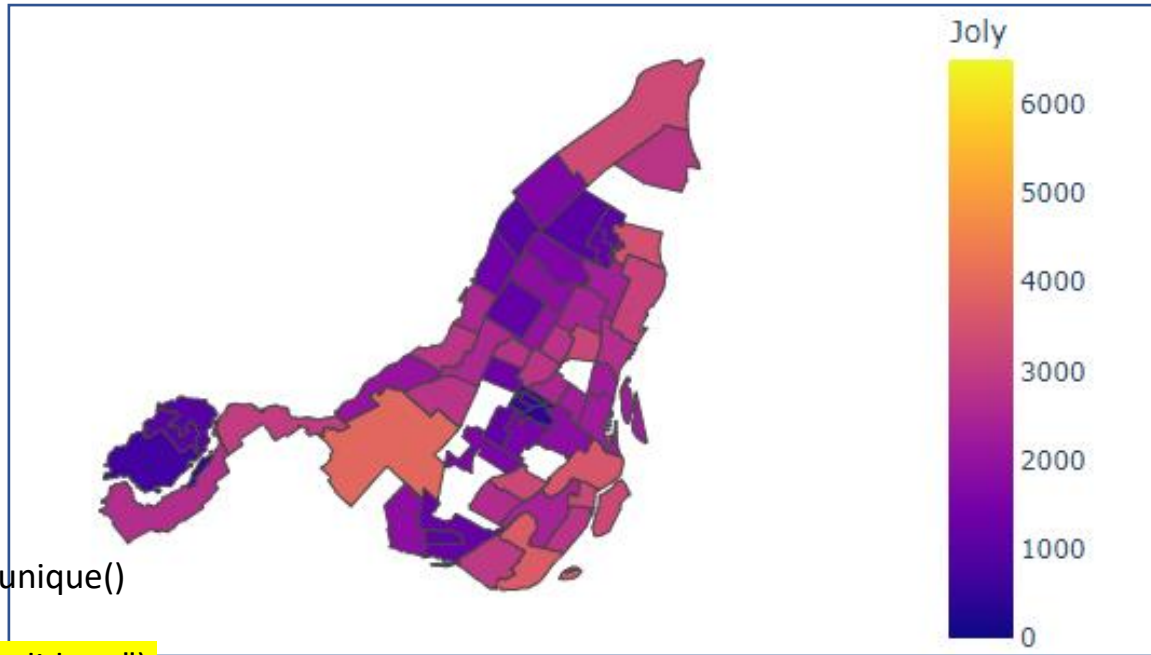
id = 'map'

id = 'piechart'
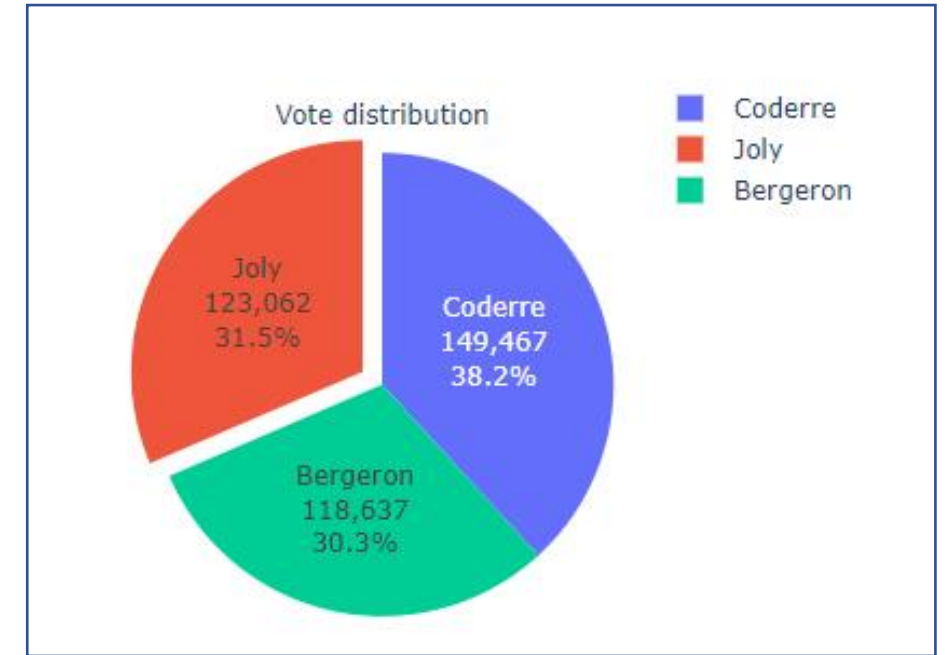

```
candidates = df.winner.unique()
app.layout = html.Div([
    html.Div([html.P("Candidate:"),
    dcc.RadioItems(
        id='candidate', options=[{'value': x, 'label': x} for x in candidates],
        value=candidates[0], labelStyle={'display': 'inline-block'}
    )]),
    dcc.Graph(id="map", style={'width': '60%', 'display': 'inline-block'}, clear_on_unhover=True),
    dcc.Graph(id="piechart", style={'width': '40%', 'display': 'inline-block'}),
], style={'width': '100%', 'display': 'inline-block'})
```

Plotly Dash callbacks

Callback architecture

We can add callback functions to user actions (e.g., mouseover, click, value change etc.) on each component

id = 'candidate'

Candidate:

◉ Joly ◯ Coderre ◯ Bergeron

candidate

value

1
63 ms

map

1
368 ms

figure   hoverData

piechart

figure

id = 'map'

Joly

6000

5000

4000

3000

2000

1000

0

id = 'piechart'

Vote distribution

■ Coderre
■ Joly
■ Bergeron

Joly
123,062
31.5%

Coderre
149,467
38.2%

Bergeron
118,637
30.3%
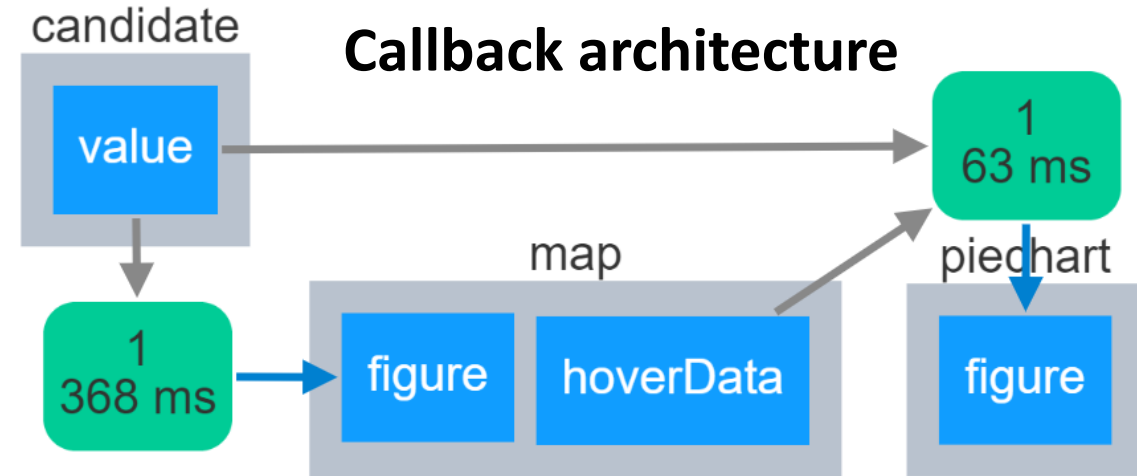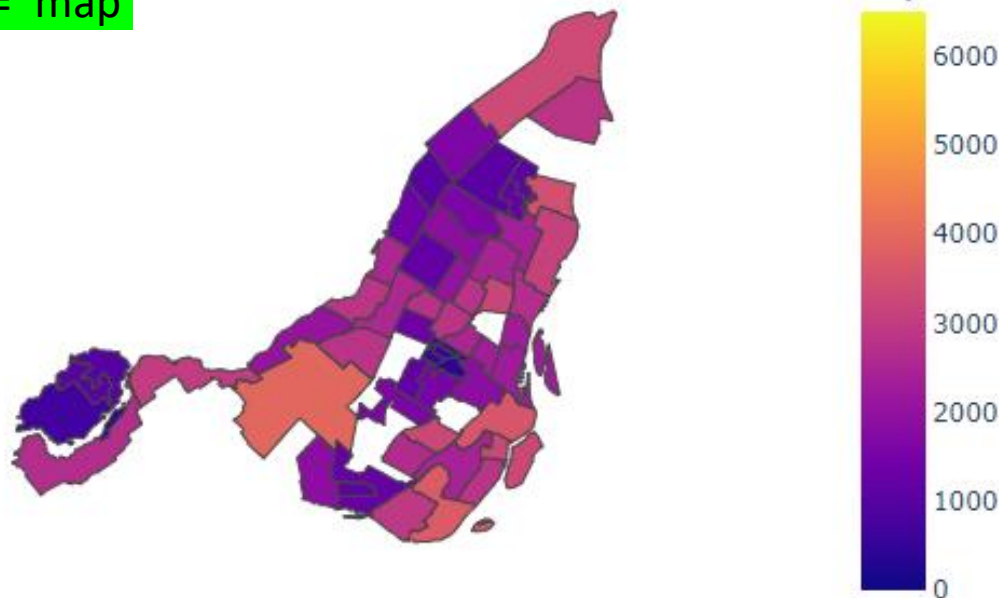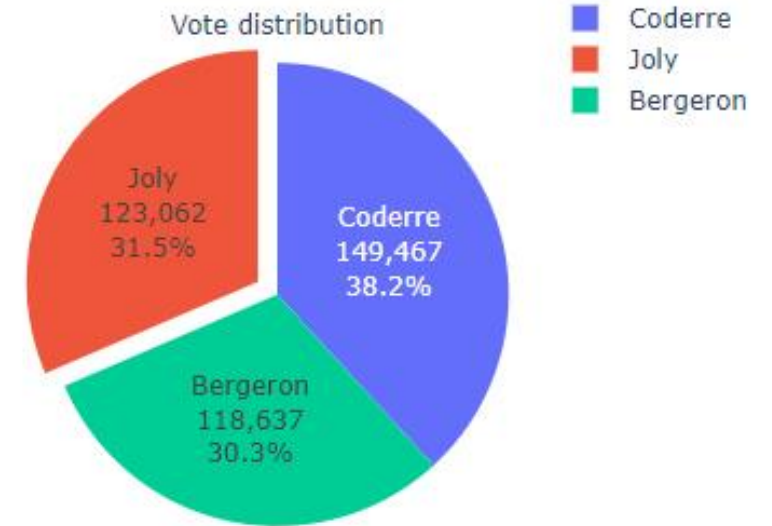
Output and Input components of Callback

```
@app.callback(
    Output("map", "figure"),
    [Input("candidate", "value")])
def display_choropleth(candidate):
    fig = px.choropleth(
        df, geojson=geojson, color=candidate,
        locations="district",
        featureidkey="properties.district",
        range_color=[0, 6500])
    return fig
```

id = 'candidate'

Candidate:

◉ Joly ○ Coderre ○ Bergeron

id = 'map'

id = 'piechart'

Joly

6000
5000
4000
3000
2000
1000
0

Vote distribution

■ Coderre
■ Joly
■ Bergeron

Joly
123,062
31.5%

Coderre
149,467
38.2%

Bergeron
118,637
30.3%

More on Plotly choropleth maps - https://plotly.github.io/plotly.py-docs/generated/plotly.express.choropleth.html

Output and Input components of Callback


```python
@app.callback(
    Output("piechart", "figure"),
    [Input("candidate", "value"),
     Input("map", "hoverData")])
def county_piechart(candidate, hoverData):
    if hoverData is None:
        votes_sum = df.sum(numeric_only=True)
        votes = [votes_sum[x] for x in candidates]
        title = "Vote distribution"
    else:
        district = hoverData['points'][0]['location']
        row = df.loc[df['district'] == district]
        votes = [row.iloc[0][x] for x in candidates]
        title = "Vote distribution in " + district
    pull = [0.1 if x == candidate else 0 for x in candidates]
    fig = go.Figure(data=[go.Pie(labels=candidates, values=votes,
                                 title=title,
                                 pull=pull,
                                 textinfo='label+value+percent')])
    return fig
```

id = 'candidate'

id = 'map'

id = 'piechart'

More on Ploty graph object pie charts - https://plotly.github.io/plotly.py-docs/generated/plotly.graph_objects.Pie.html

Complete Code

```python
import dash
from dash import dcc
from dash import html
from dash.dependencies import Input, Output
import plotly.express as px
import plotly.graph_objects as go
app = dash.Dash(__name__)

df = px.data.election()
geojson = px.data.election_geojson()
candidates = df.winner.unique()

app.layout = html.Div([
    html.Div([html.P("Candidate:"),
    dcc.RadioItems(
        id='candidate',
        options=[{'value': x, 'label': x} for x in candidates],
        value=candidates[0],
        labelStyle={'display': 'inline-block'}
    )]),
    dcc.Graph(id="map", style={'width': '60%', 'display': 'inline-block'},
clear_on_unhover=True),
    dcc.Graph(id="piechart", style={'width': '40%', 'display': 'inline-block'}),
], style={'width': '100%', 'display': 'inline-block'})

#callback for map listening from candidate ratio buttons
@app.callback(
    Output("map", "figure"),
    [Input("candidate", "value")])
def display_choropleth(candidate):
    fig = px.choropleth(
        df, geojson=geojson, color=candidate,
        locations="district", featureidkey="properties.district",
        range_color=[0, 6500])
    fig.update_geos(fitbounds="locations", visible=False)
    return fig
```
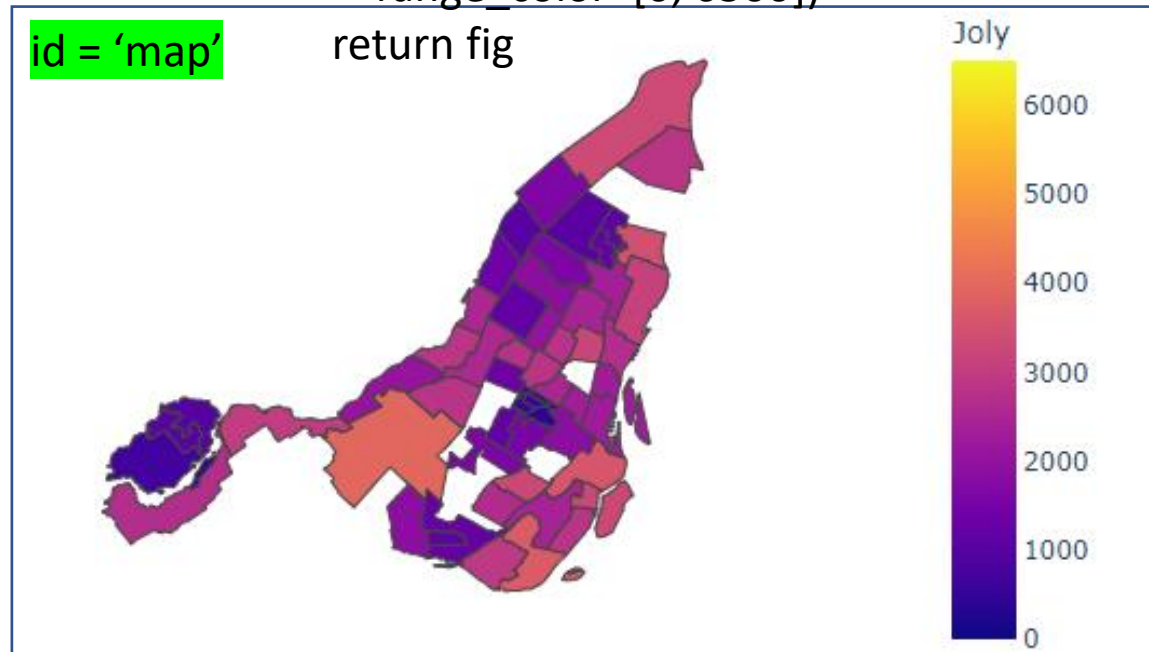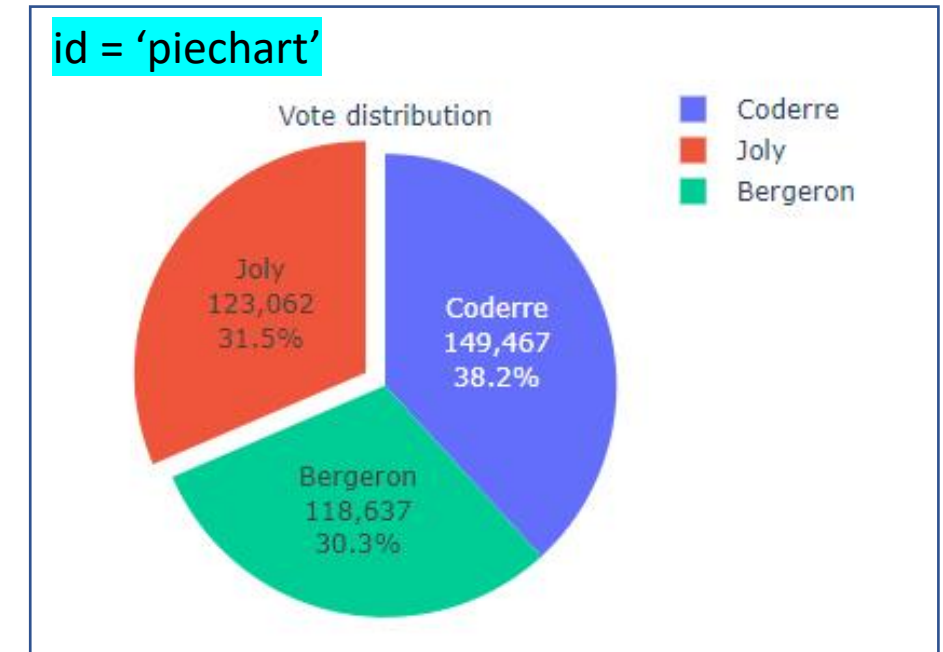
```python
#Call back for pie-chart listening from two inputs
@app.callback(
    Output("piechart", "figure"),
    [Input("candidate", "value"),
     Input("map", "hoverData")])
def county_piechart(candidate, hoverData):
    if hoverData is None:
        votes_sum = df.sum(numeric_only=True)
        votes = [votes_sum[x] for x in candidates]
        title = "Vote distribution"
    else:
        district = hoverData['points'][0]['location']
        row = df.loc[df['district'] == district]
        votes = [row.iloc[0][x] for x in candidates]
        title = "Vote distribution in " + district

    pull = [0.1 if x == candidate else 0 for x in candidates]
    fig = go.Figure(data=[go.Pie(labels=candidates, values=votes,
                    title=title,
                    pull=pull,
                    textinfo='label+value+percent')])
    return fig

app.run_server(debug=True,
        port=81,  #Specify port number here
        threaded=True)
```
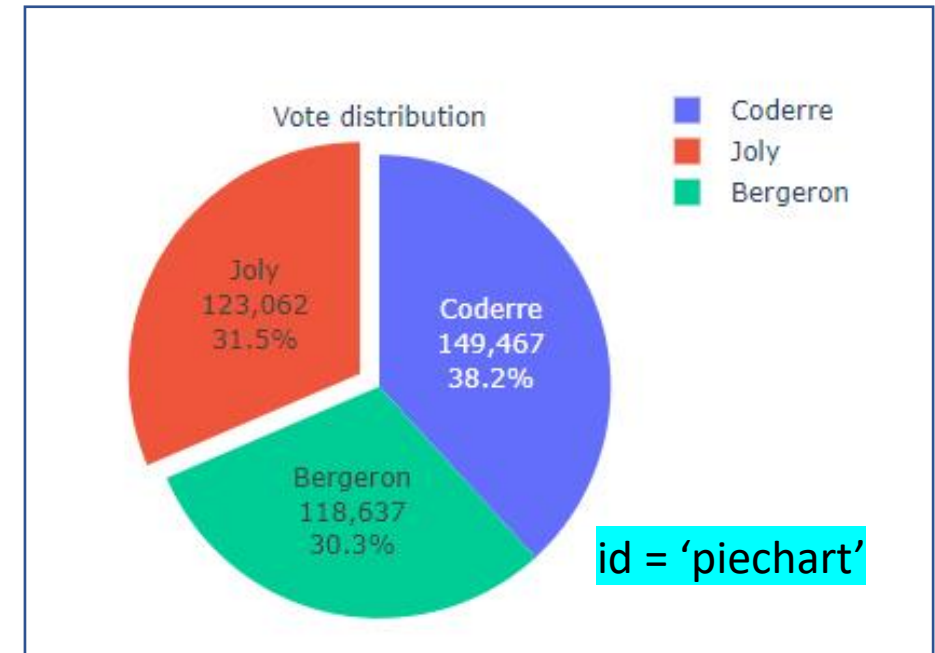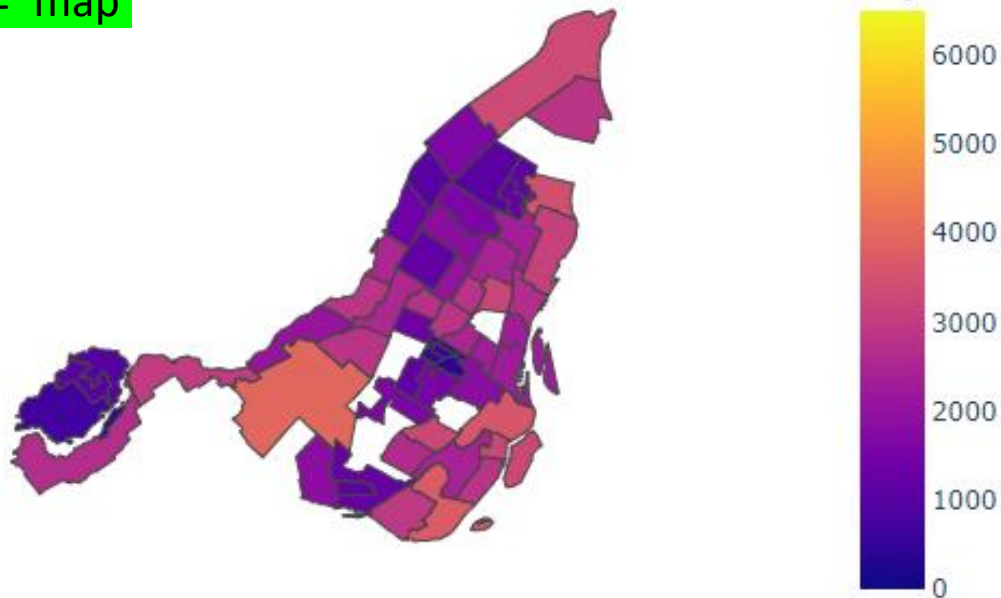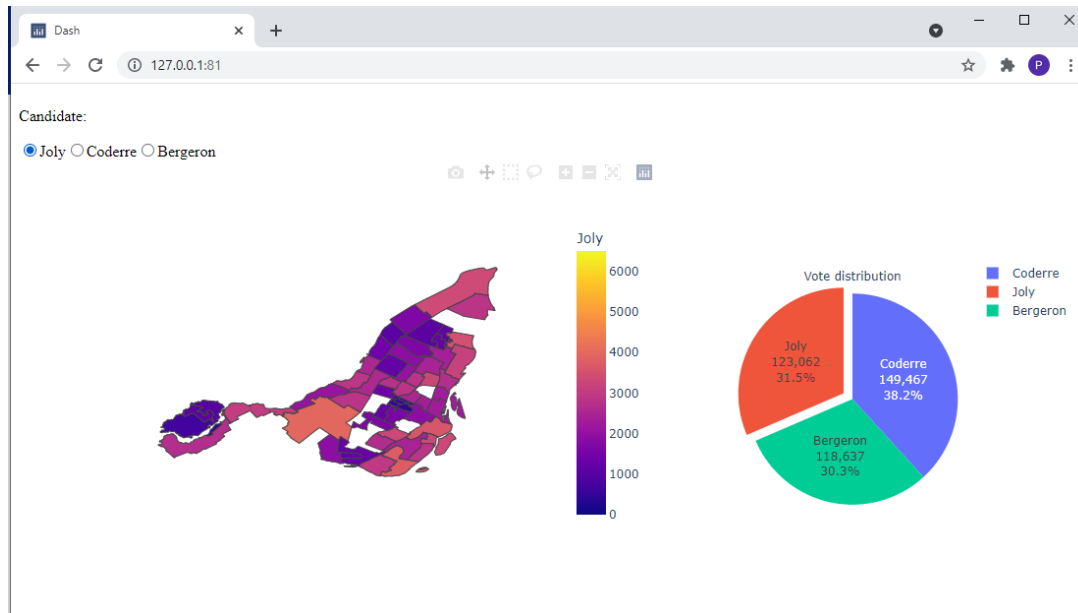
Steps to run Dash program

- Step 1: Save the code in previous slide to maps.py
- Step 2: Run python program maps.py (Install required libraries if necessary)
- Step 3: You will see the following output on the command terminal window. Go to web-browser and browser to the given URL http://127.0.0.1:81