Description of project

Literature

- Pukthuanthong & Roll (“Global market integration: An alternative measure and its application,” 2008) and Cotter, Gabriel and Roll (“Can Housing Risk Be Diversified? A Cautionary Tale from the Housing Boom and Bust,” 2015) have suggested than real estate portfolios should be diversified based on a "cointegration factor".
- They argue that this is a more accurate measure than correlation due to the fact that correlation is backward-looking.
- Cotter, Gabriel and Roll define the integration factor for a US state as the R-squared value derived from regressing the real estate returns in each state against country level factors, which provides a measure of the level of integration between the state's returns and economic activity in the US overall.

- I apply Cotter et. al's methodology to Canadian provinces, using the following variables sourced from StatCan:

Dependent Variable:

- Housing Price Index for each province for residential property (indexed at 2017)
- 9 regression specifications are carried out for:

```
                                                Alberta
                                                British Columbia
                                                Saskatchewan
                                                Manitoba
                                                Quebec
                                                Ontario
                                                Newfoundland and L
abrador
                                                Nova Scotia
                                                Atlantic Region
                                                New Brunswick
                                                Prince Edward Isla
nd
```

Explanatory Variables:

- debt to disposable income ratio
- mortgage interest rates

- household consumption
- household expenditure
- % change in the S&P/TSX Composite in a given year

Visualization

- The R-squared factor for the p each regression level of integration is then plotted onto the Google Maps API as a chloropleth map, using shapefiles for provincial boundaries sourced from StatCan.

1. Cleaning data for regression

In [66]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import qeds
from sklearn import linear_model

import geopandas as gpd
from shapely.geometry import Point

import gmaps
import gmaps.datasets
import requests
import json
import random
import pygeoj

import colourmap
from colourmap import rgb2hex

import math
```

In [67]:
```
# - 1. load and clean housing data

# - read housing price data in
housing_data = pd.read_csv("project_data/housing-index-by-census-met-
#housing_data.info()

# - get date, geography and value only
housing_data.tail()
housing_data_small = housing_data[["REF_DATE", "GEO", "New housing p
# housing_data_small

```

```
# - drop to keep house only
housing_data_small_1 = housing_data_small.loc[housing_data_small["Ne
housing_data_small_1

# - drop to keep 2010 - 2018 only
# housing_data_small_1[["REF_DATE"]]
housing_data_small_1["Year"] = "20" + housing_data_small_1["REF_DATE
# housing_data_small_1
housing_data_small_2 = housing_data_small_1.loc[pd.to_numeric(housin
# housing_data_small_2

# - merge index based on GEO
housing_data_small_3 = housing_data_small_2.set_index(["GEO", "Year"
# housing_data_small_3


# - get only certain provinces
housing_data_small_4 = housing_data_small_3.loc[["Alberta",
                                                 "British Columbia",
                                                "Saskatchewan",
                                                "Manitoba",
                                                "Quebec",
                                                "Ontario",
                                                 "Newfoundland and La
                                                 "Nova Scotia",
                                                 "Atlantic Region",
                                                 "New Brunswick",
                                                 "Prince Edward Islan

                    ]]

housing_data_small_4 = housing_data_small_4.reset_index()
# get date
housing_data_small_4["Month"] = housing_data_small_4["REF_DATE"].str
housing_data_small_4["Date-str"] = "1 " + housing_data_small_4["Montl
housing_data_small_4["Date"] = pd.to_datetime(housing_data_small_4["D

# - put date on horizontal
housing_data_small_5 = housing_data_small_4.pivot_table(
    index= "Date",
    columns="GEO",
    values="VALUE")

housing_data_small_5.to_csv("cleaned_data/housing_index.csv")

housing_index_data_clean = housing_data_small_5
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:18: S
ettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

In [68]:
```
1 housing_index_data_clean.head()
```

Out[68]:

GEO	Alberta	Atlantic Region	British Columbia	Manitoba	New Brunswick	Newfoundland and Labrador	Nova Scotia	Ontario	Prince Edward Island
Date									
2010-01-01	92.7	93.6	94.5	80.2	97.3	92.9	90.6	75.7	101.2
2010-02-01	92.9	93.6	95.2	80.4	97.5	92.9	90.7	75.6	100.4
2010-03-01	93.3	93.6	96.2	80.9	97.5	92.9	90.8	76.1	99.8
2010-04-01	93.3	94.1	96.7	81.2	97.4	94.2	90.8	76.3	99.4
2010-05-01	93.7	94.4	97.0	81.5	97.9	94.5	90.9	76.9	99.8

In [69]:
```
1  # - 1a. load and clean data for factor group 1: Canadians' financial
2  income_data = pd.read_csv("project_data/household-financial-health-L
3  # income_data
4
5  # - only keep Canada level data
6  income_data_small = income_data.loc[income_data["GEO"] == "Canada"]
7
8  # - only keep financial health data
9  income_data_small_1 = income_data_small.loc[income_data_small["Charac
10
11 # - rename REF_DATE to year
12 income_data_small_1["Year"] = pd.to_numeric(income_data_small_1["REF_
13
14 # - get year between 2010 and 2017
15 income_data_small_2 = income_data_small_1.loc[(income_data_small_1["
16
17 income_data_small_2["Year"] = income_data_small_2["Year"].apply(str)
18
19 # - set year as index
```

```
# - set year as index
income_data_small_3 = income_data_small_2.pivot_table(
index = "Year",
columns = "Net worth indicators (wealth)",
values = "VALUE")

income_data_small_3 = income_data_small_3[["Debt to disposable income
income_data_small_3 = income_data_small_3.reset_index()

# - spread data to every year
income_data_small_3["Jan"] = "01 " + "Jan" + " " + income_data_small_
income_data_small_3["Feb"] = "01 " + "Feb" + " " + income_data_small_
income_data_small_3["Mar"] = "01 " + "Mar" + " " + income_data_small_
income_data_small_3["Apr"] = "01 " + "Apr" + " " + income_data_small_
income_data_small_3["May"] = "01 " + "May" + " " + income_data_small_
income_data_small_3["Jun"] = "01 " + "Jun" + " " + income_data_small_
income_data_small_3["Jul"] = "01 " + "Jul" + " " + income_data_small_
income_data_small_3["Aug"] = "01 " + "Aug" + " " + income_data_small_
income_data_small_3["Sep"] = "01 " + "Sep" + " " + income_data_small_
income_data_small_3["Oct"] = "01 " + "Oct" + " " + income_data_small_
income_data_small_3["Nov"] = "01 " + "Nov" + " " + income_data_small_
income_data_small_3["Dec"] = "01 " + "Dec" + " " + income_data_small_

# - melt
income_data_small_4 = income_data_small_3.melt(id_vars=["Year", "Debt
income_data_small_4["Date"] = pd.to_datetime(income_data_small_4["va

# get cols
income_data_small_5 = income_data_small_4[["Date",
                                           "Debt to disposable income

# - save as df
income_data_small_5.to_csv("cleaned_data/canadians_financial_health_(

financial_health_stats_clean = income_data_small_5
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:12: S
ettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:17: S
ettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
```

In [70]:
```
financial_health_stats_clean.head()
```

Out[70]:

	Date	Debt to disposable income ratio
0	2010-01-01	167.2
1	2011-01-01	169.7
2	2012-01-01	170.0
3	2013-01-01	169.9
4	2014-01-01	171.8

In [71]:
```
# - 1b. load and clean data for factor 2: mortgage interest rates
mortgage_data = pd.read_csv("project_data/canada-mortage-lending-rate
mortgage_data

# - keep only 2010 to 2018 data
mortgage_data["Year"] = mortgage_data["REF_DATE"].str[:4]
mortgage_data_small = mortgage_data.loc[pd.to_numeric(mortgage_data[


# - get average mortgage rate for each year
mortgage_data_small_1 = mortgage_data_small[["REF_DATE",
                                             "VALUE"
                                            ]]

mortgage_data_small_1["Date-str"] = mortgage_data_small_1["REF_DATE"
mortgage_data_small_1["Date"] = pd.to_datetime(mortgage_data_small_1

mortgage_data_small_1.rename(columns = {'VALUE':'Mortgage Rate'}, in

mortgage_data_small_2 = mortgage_data_small_1[["Date", "Mortgage Rate

# - write to csv
mortgage_data_small_2.to_csv("cleaned_data/mortgage_rate_clean.csv")

mortgage_rate_clean = mortgage_data_small_2
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15: S
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15: S
ettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:16: S
ettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:4133:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
```

In [72]:
```
1 mortgage_rate_clean.head()
```

Out[72]:

	Date	Mortgage Rate
708	2010-01-01	4.80
709	2010-02-01	4.73
710	2010-03-01	4.71
711	2010-04-01	5.15
712	2010-05-01	5.30

In [73]:
```
1 # - 1c. get factor group 3: household consumption/expenditure
```

```
household_spending = pd.read_csv("project_data/expenditure-consumptio

# - keep national level data only
household_spending_small = household_spending.loc[household_spending

# - keep 2010 to 2017 data only
household_spending_small_1 = household_spending_small.loc[pd.to_numer

# - keep expenditure and consumption only
household_spending_small_2 = household_spending_small_1.set_index("Ho
household_spending_small_3 = household_spending_small_2.loc[["Total e
                                                             "Total cu

# - reshape to set index and year
household_spending_small_3.rename(columns = {'REF_DATE':'Year'}, inpl

household_spending_small_3["Year"] = household_spending_small_3["Year


household_spending_small_4 = household_spending_small_3.pivot_table(
index = "Year",
columns = "Household expenditures, summary-level categories",
values = "VALUE")

household_spending_small_4 = household_spending_small_4.reset_index(
# - spread to monthly
# - spread data to every year
household_spending_small_4["Jan"] = "01 " + "Jan" + " " + household_s
household_spending_small_4["Feb"] = "01 " + "Feb" + " " + household_s
household_spending_small_4["Mar"] = "01 " + "Mar" + " " + household_s
household_spending_small_4["Apr"] = "01 " + "Apr" + " " + household_s
household_spending_small_4["May"] = "01 " + "May" + " " + household_s
household_spending_small_4["Jun"] = "01 " + "Jun" + " " + household_s
household_spending_small_4["Jul"] = "01 " + "Jul" + " " + household_s
household_spending_small_4["Aug"] = "01 " + "Aug" + " " + household_s
household_spending_small_4["Sep"] = "01 " + "Sep" + " " + household_s
household_spending_small_4["Oct"] = "01 " + "Oct" + " " + household_s
household_spending_small_4["Nov"] = "01 " + "Nov" + " " + household_s
household_spending_small_4["Dec"] = "01 " + "Dec" + " " + household_s

# - melt

household_spending_small_4 = household_spending_small_4.melt(id_vars=



household_spending_small_4["Date"] = pd.to_datetime(household_spendin

household_spending_small_5 = household_spending_small_4[["Date",
                                                         "Total expend
                                                         "Total curre
```

```
# - write to dataframe
household_spending_small_5.to_csv("cleaned_data/consumption_expenditu

household_spending_clean = household_spending_small_5
```

In [74]:
```
household_spending_clean.head()
```

Out[74]:

	Date	Total expenditure	Total current consumption
0	2010-01-01	72075.0	54013.0
1	2011-01-01	73646.0	55227.0
2	2012-01-01	75695.0	56330.0
3	2013-01-01	79098.0	58576.0
4	2014-01-01	80727.0	59055.0

In [75]:
```
# - 1d. get factor 4: tsx performance
tsx_data = pd.read_csv("project_data/tsx-monthly.csv")

# - keep only 2009 - 2019
tsx_data["Year"] = tsx_data["REF_DATE"].str[:4]
tsx_data_small = tsx_data.loc[pd.to_numeric(tsx_data["Year"]).betwee

# - index by tse stats
tsx_data_small_1 = tsx_data_small.set_index("Toronto Stock Exchange 
tsx_data_small_2 = tsx_data_small_1.loc[["Standard and Poor's/Toronto

# - get time series data
tsx_data_small_3 = tsx_data_small_2.set_index("Year")
tsx_data_small_4 = tsx_data_small_3[["VALUE",
                                     "REF_DATE"]]

# - get yearly change
tsx_data_small_4["Month"] = tsx_data_small_4["REF_DATE"].str[-2:]

tsx_data_small_4["Date"] = pd.to_datetime(tsx_data_small_4["REF_DATE
tsx_data_small_4 = tsx_data_small_4.reset_index()

tsx_data_small_5 = tsx_data_small_4[["Date",
                                     "VALUE",
                                     "Year"]]

tsx_data_small_6 = tsx_data_small_5.set_index(["Date", "Year"])

tsx_data_small_7 = tsx_data_small_6.pct_change()
tsx_data_small_7 = tsx_data_small_7.reset_index()
```

```
31
32 tsx_data_small_8 = tsx_data_small_7.loc[pd.to_numeric(tsx_data_small_
33
34 tsx_data_small_8["TSX Pct Change"] = tsx_data_small_8["VALUE"]*100
35 tsx_data_small_9 = tsx_data_small_8[["Date",
36                                      "TSX Pct Change"]]
37
38
39 # tsx_data_small_9.rename(columns = {'VALUE':'TSX pct change'}, inpl
40
41 # - write to csv
42 tsx_data_small_9.to_csv("cleaned_data/tsx_change_clean.csv")
43
44 tsx_change_clean = tsx_data_small_9
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:18: S
ettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:20: S
ettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

In [76]:
```
1 tsx_change_clean.head()
```

Out[76]:

	Date	TSX Pct Change
12	2010-01-01	-5.549071
13	2010-02-01	4.825176
14	2010-03-01	3.509140
15	2010-04-01	1.436899
16	2010-05-01	-3.666538

In [77]:

```
# - 2e. merge datasets into 1 df

# housing_index_data_clean
# mortgage_rate_clean
# tsx_change_clean
# household_spending_clean
# financial_health_stats_clean

# - merge housing index and mortgage rate
merge1 = pd.merge(mortgage_rate_clean, housing_index_data_clean, on =

# - merge in tsx change
merge2 = pd.merge(merge1, tsx_change_clean, on = "Date")

# - merge in household spending
merge3 = pd.merge(merge2, household_spending_clean, on = "Date")

# - merge in financial health stats
merge4 = pd.merge(merge3, financial_health_stats_clean, on = "Date")

regression_data = merge4.set_index("Date")

# - write to csv
regression_data.to_csv("cleaned_data/regression_data.csv")
```

In [78]:
```
1 regression_data
```

Out[78]:

Date	Mortgage Rate	Alberta	Atlantic Region	British Columbia	Manitoba	New Brunswick	Newfoundland and Labrador	Nova Scotia	Onta
2010-01-01	4.80	92.7	93.6	94.5	80.2	97.3	92.9	90.6	7
2010-02-01	4.73	92.9	93.6	95.2	80.4	97.5	92.9	90.7	7
2010-03-01	4.71	93.3	93.6	96.2	80.9	97.5	92.9	90.8	7
2010-04-01	5.15	93.3	94.1	96.7	81.2	97.4	94.2	90.8	7
2010-05-01	5.30	93.7	94.4	97.0	81.5	97.9	94.5	90.9	7
...	...	...	...	...	...	...	...	...	
2017-08-01	3.82	100.1	99.9	107.8	103.6	100.5	98.6	100.9	10
2017-09-01	3.89	99.9	100.0	108.2	104.1	100.5	98.6	101.0	10
2017-10-01	3.98	100.0	100.0	108.7	104.2	100.5	98.6	101.4	10
2017-11-01	4.04	100.2	100.2	108.7	104.4	100.9	98.8	101.4	10
2017-12-01	4.07	100.0	100.3	108.7	104.5	101.1	98.8	101.3	10

96 rows × 16 columns

In [79]:
```
1 regression_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 96 entries, 2010-01-01 to 2017-12-01
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Mortgage Rate                   96 non-null     float64
 1   Alberta                         96 non-null     float64
 2   Atlantic Region                 96 non-null     float64
 3   British Columbia                96 non-null     float64
 4   Manitoba                        96 non-null     float64
 5   New Brunswick                   96 non-null     float64
 6   Newfoundland and Labrador       96 non-null     float64
 7   Nova Scotia                     96 non-null     float64
 8   Ontario                         96 non-null     float64
 9   Prince Edward Island            96 non-null     float64
 10  Quebec                          96 non-null     float64
 11  Saskatchewan                    96 non-null     float64
 12  TSX Pct Change                  96 non-null     float64
 13  Total expenditure               96 non-null     float64
```

2. Data exploration

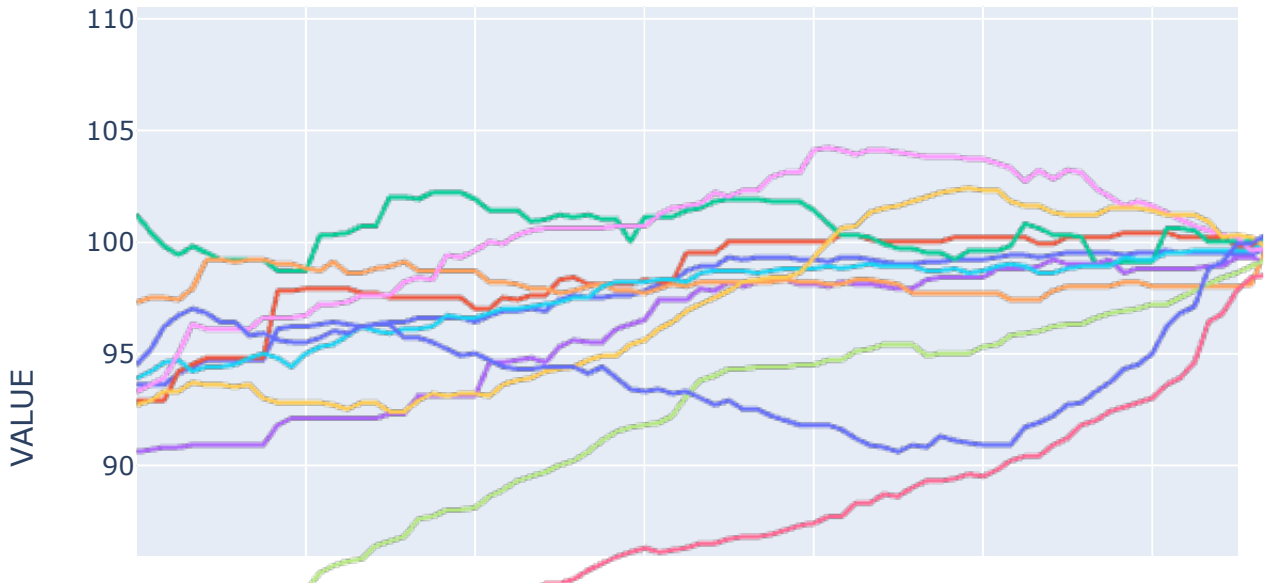
In [80]:
```
1  # - 2a. convert to date format
2  housing_data_small_plotting = housing_data_small_4.reset_index()
3  housing_data_small_plotting["Month"] = housing_data_small_plotting["
4  housing_data_small_plotting["Date-str"] = "1 " + housing_data_small_
5
6  housing_data_small_plotting["Date"] = pd.to_datetime(housing_data_sma
7
8  housing_data_small_plotting = housing_data_small_plotting[["Date",
9                                                             "VALUE",
10                                                            "GEO"]]
11 housing_data_small_plotting.head()
```

Out[80]:

	Date	VALUE	GEO
0	2010-01-01	93.6	Atlantic Region
1	2010-01-01	92.9	Newfoundland and Labrador
2	2010-01-01	101.2	Prince Edward Island
3	2010-01-01	90.6	Nova Scotia
4	2010-01-01	97.3	New Brunswick

In [81]:

```
# - 2b. plot time series by region
import plotly.express as px
import plotly.graph_objects as go


housing_index_plot = px.line(housing_data_small_plotting ,
                             x="Date",
                             y="VALUE",
                             color="GEO",
                             line_group="GEO",
                             hover_name="GEO")

# write to html
housing_index_plot.write_html("cleaned_data/housing_index_plot.html"

housing_index_plot.show()

# (sometimes the plot doesn't show)

# it usually works better when the zoom on the top right hand corner

# the data was standardized using 2017 as index, which is why the li
```



3. Multivariate Linear Regression

- Regress housing index for each province on all other factors
- Compute R squared value for each province, which gives measure of how integrated the province's housing returns are with the rest of the economy

In [82]:
```
# - 3a. write regression function to get R squared value for Alberta

# - get explanatory vars
X = regression_data.drop(["Alberta",
                          "British Columbia",
                          "Saskatchewan",
                          "Manitoba",
                          "Quebec",
                          "Ontario",
                          "Newfoundland and Labrador",
                          "Nova Scotia",
                          "Atlantic Region",
                          "New Brunswick",
                          "Prince Edward Island"], axis = 1).copy()

# - get y for Alberta
ab_true = regression_data["Alberta"]

# - set the model instance for Alberta
alberta_model = linear_model.LinearRegression()
alberta_model.fit(X, ab_true)

# - print coefficients to check
alberta_model.intercept_
# intercept : -101.25278654861009
alberta_coefs = pd.Series(dict(zip(list(X), alberta_model.coef_)))
alberta_coefs
```

Out[82]:
```
Mortgage Rate                    1.481234
TSX Pct Change                   0.015356
Total expenditure                0.003602
Total current consumption       -0.004274
Debt to disposable income ratio -0.019740
dtype: float64
```

In [83]:
```
# - get R-squared value for Alberta :
# - higher. R squared value means higher cointegration of Alberta's
ab_r2 = alberta_model.score(X, ab_true)
ab_r2
```

Out[83]: 0.9588117090190931

In [84]:
```
# - 3b. get r2 score for all other variables

def get_r2_score(province:str, df:pd.DataFrame, x:pd.DataFrame):
    """
    get r2 value for each province
    """
    # explanatory vars
    province_true = df[province]

    # set model instance
    province_model = linear_model.LinearRegression()
    province_model.fit(X, province_true)

    # get r2
    r2 = province_model.score(X, province_true)

    return r2
```

In [85]:
```
get_r2_score("Quebec", regression_data, X)
```

Out[85]: 0.9234561210118982

In [86]:

```
# - get dataframe of r2 scores

province  = ["Alberta",
                           "British Columbia",
                           "Saskatchewan",
                           "Manitoba",
                           "Quebec",
                           "Ontario",
                           "Newfoundland and Labrador",
                           "Nova Scotia",
                           "Atlantic Region",
                           "New Brunswick",
                           "Prince Edward Island"]

r2 = []
for p in province:
    score = get_r2_score(p, regression_data, X)
    r2.append(score)
print (r2)

r2_scores = pd.DataFrame({"province":province,
                          "r2score":r2})

# - write to csv
r2_scores.to_csv("cleaned_data/r2_data.csv")
```

```
[0.9588117090190931, 0.8363669273456951, 0.8070086015414639, 0.9536842
03647346, 0.9234561210118982, 0.9637419690886144, 0.7309201814771507,
0.9620597630269168, 0.9077251256117037, 0.6178935456376443, 0.22304545
44530599]
```

In [87]:
```
# - Alberta, Manitoba, Ontario, Nova Scotia and New Brunswick are hi
# - Saskatchewan, Newfoundland & Labrador, and PEI are the least int
r2_scores
```

Out[87]:

	province	r2score
0	Alberta	0.958812
1	British Columbia	0.836367
2	Saskatchewan	0.807009
3	Manitoba	0.953684
4	Quebec	0.923456
5	Ontario	0.963742
6	Newfoundland and Labrador	0.730920
7	Nova Scotia	0.962060
8	Atlantic Region	0.907725
9	New Brunswick	0.617894
10	Prince Edward Island	0.223045

5. Load results onto map

In [88]:
```
# - clean boundary file
boundary_data = gpd.read_file("project_data/boundary_clean.json")
boundary_data = boundary_data.set_index("PRENAME")

boundary_data_small = boundary_data.loc[["Alberta",
                            "British Columbia",
                            "Saskatchewan",
                            "Manitoba",
                            "Quebec",
                            "Ontario",
                            "Newfoundland and Labrador",
                            "Nova Scotia",
                            "New Brunswick",
                            "Prince Edward Island"]]

# - write to geojson file
boundary_data_small.to_file("cleaned_data/boundary_clean.json", drive
```

In [89]:

```
# - get colours by r2 value
r2_scores_plotting = r2_scores.set_index("province")
r2_scores_plotting = r2_scores_plotting.drop(["Atlantic Region"], ax
r2_scores_plotting["hue"] = np.rint((r2_scores_plotting["r2score"]-0
r2_scores_plotting["hue"] = r2_scores_plotting["hue"].astype(int)
r2_scores_plotting
```

Out[89]:

province	r2score	hue
Alberta	0.958812	152
British Columbia	0.836367	127
Saskatchewan	0.807009	121
Manitoba	0.953684	151
Quebec	0.923456	145
Ontario	0.963742	153
Newfoundland and Labrador	0.730920	106
Nova Scotia	0.962060	152
New Brunswick	0.617894	84
Prince Edward Island	0.223045	5

In [90]:

```
# - get rgb values
rgb = []
for c in r2_scores_plotting["hue"]:
    rgb_cols = '#%02x%02x%02x'% (255-c, c, 255)
    rgb = rgb + [rgb_cols]
print (rgb)


r2_scores_plotting["rgb"] = rgb
r2_scores_plotting
```

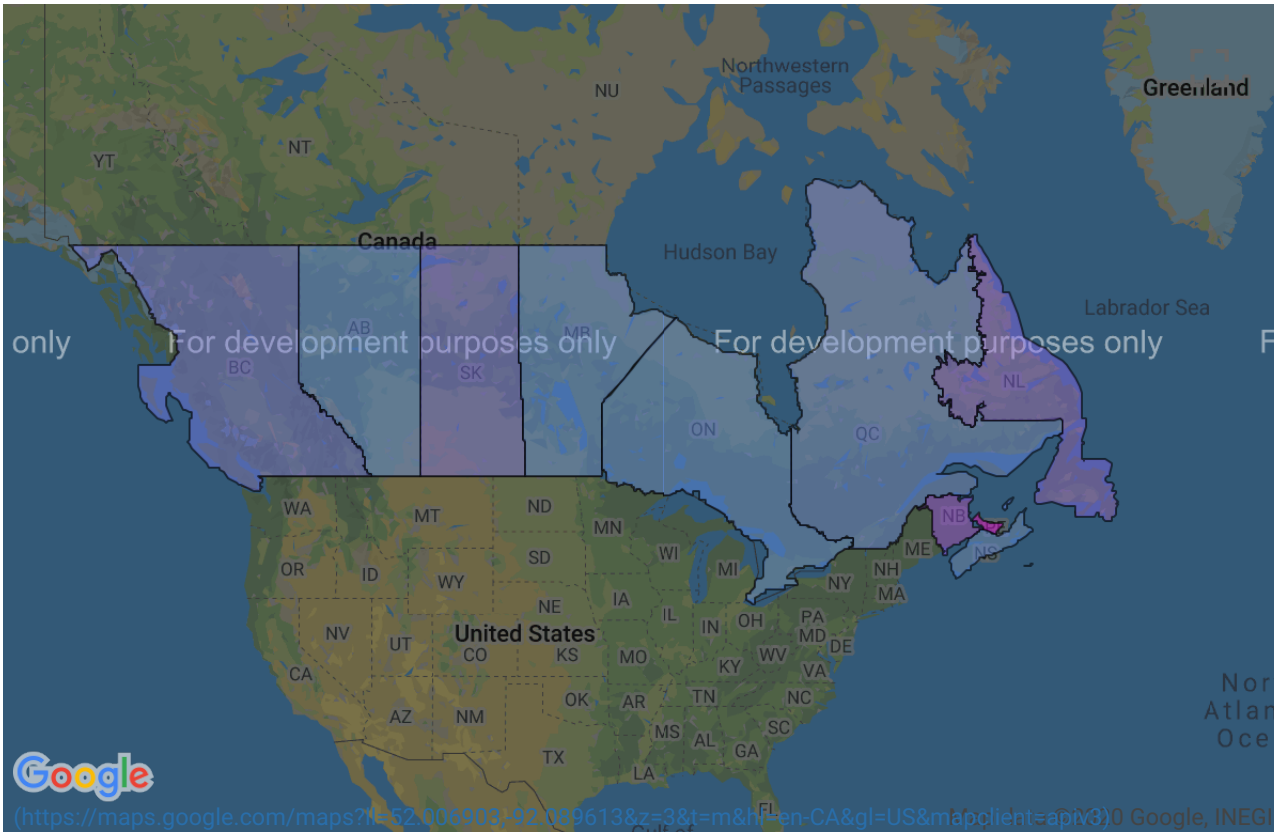
```
['#6798ff', '#807fff', '#8679ff', '#6897ff', '#6e91ff', '#6699ff', '#9
56aff', '#6798ff', '#ab54ff', '#fa05ff']
```

Out[90]:

province	r2score	hue	rgb
Alberta	0.958812	152	#6798ff
British Columbia	0.836367	127	#807fff
Saskatchewan	0.807009	121	#8679ff
Manitoba	0.953684	151	#6897ff
Quebec	0.923456	145	#6e91ff
Ontario	0.963742	153	#6699ff
Newfoundland and Labrador	0.730920	106	#956aff
Nova Scotia	0.962060	152	#6798ff
New Brunswick	0.617894	84	#ab54ff
Prince Edward Island	0.223045	5	#fa05ff

In [91]:

```
# - load onto google maps

# - get google maps api
apikey = "AIzaSyDzJhxiVPyQ4y0Ycb9f3hJ5vyzxMbjmdB4"
gmaps.configure(api_key = apikey)

# - get figure
fig = gmaps.figure()

# - get canada boundaries
canada_boundaries = open("cleaned_data/boundary_clean.json")
canada_boundaries = json.load(canada_boundaries)

colours = []

boundary_layer = gmaps.geojson_layer(canada_boundaries,
                                     fill_color = rgb)
fig.add_layer(boundary_layer)
# - we see that PEI, Newfoundland, Sask, and (surprisingly) British
#   less integrated with the Canadian eocnomy

fig
```



In [92]:

```
# - plot population onto map to see where people are moving to as he
```

```
2
3  # - read in data
4  population_data = pd.read_csv("project_data/employment.csv", delimite
5  population_data
6
7  # - clean data
8  population_data_small = population_data.loc[population_data["Statist
9  population_data_small["Year"] = population_data_small["REF_DATE"].sti
10
11 population_data_small = population_data_small.loc[population_data_sma
12
13 population_data_small = population_data_small.loc[population_data_sma
14
15 population_data_small = population_data_small[["REF_DATE",
16                        "GEO",
17                        "Labour force characteristics",
18                                              "VALUE"]]
19
20 # - get population change over last 5 years
21 relevant_2015 = (population_data_small["REF_DATE"] == "2015-03")
22 population_data_2015 = population_data_small.loc[relevant_2015]
23 population_data_2015.rename(columns = {'VALUE':'Population 2015'}, in
24
25 relevant_2020 = (population_data_small["REF_DATE"] == "2020-03")
26 population_data_2020 = population_data_small.loc[relevant_2020]
27 population_data_2020.rename(columns = {'VALUE':'Population 2020'}, in
28
29 population_clean = pd.merge(population_data_2020, population_data_20
30
31 population_clean["Population_change"] = (1.5 + 100*(population_clean
32
33 population_clean = population_clean.set_index("GEO")
34 population_clean = population_clean.drop(["Canada",
35                                           "Alberta",
36                             "British Columbia",
37                             "Saskatchewan",
38                             "Manitoba",
39                             "Quebec",
40                             "Ontario",
41                             "Newfoundland and Labrador",
42                             "Nova Scotia",
43                             "New Brunswick",
44                             "Prince Edward Island",
45                             "Ottawa-Gatineau, Ontario/Quebec",
46                                        "Ottawa-Gatineau, Quebec pa
47                                       "Ottawa-Gatineau, Ontario pa
```

```
/opt/anaconda3/lib/python3.7/site-packages/IPython/core/interactiveshe
ll.py:3063: DtypeWarning:

Columns (13) have mixed types. Specify dtype option on import or set lo
```

```
Columns (13) have mixed types.Specify dtype option on import or set lo
w_memory=False.

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: Se
ttingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:4133:
```

In [93]:

```
# convert to geodataframe
import geopandas as gpd
import geopy as gpy
from geopandas.tools import geocode
from geopy.geocoders import Nominatim

population_geo = gpd.GeoDataFrame(population_clean)
population_geo = population_geo.reset_index()

population_geo = population_geo.replace({"GEO": "St. Catharines-Niaga
population_geo = population_geo.replace({"GEO": "Kitchener-Cambridge-
population_geo = population_geo.replace({"GEO": "Abbotsford-Mission,
population_geo
```

Out[93]:

	GEO	REF_DATE_x	Labour force characteristics_x	Population 2020	REF_DATE_y	Labour force characteristics_y	Pop
0	St. John's, Newfoundland and Labrador	2020-03	Population	187.0	2015-03	Population	
1	Halifax, Nova Scotia	2020-03	Population	379.6	2015-03	Population	
2	Moncton, New Brunswick	2020-03	Population	133.0	2015-03	Population	
3	Saint John, New Brunswick	2020-03	Population	107.7	2015-03	Population	
4	Saguenay, Quebec	2020-03	Population	133.4	2015-03	Population	
5	Québec, Quebec	2020-03	Population	690.1	2015-03	Population	

In [94]:
```
# convert to geodataframe
import geopandas as gpd
import geopy as gpy
from geopandas.tools import geocode
from geopy.geocoders import Nominatim

population_geo = gpd.GeoDataFrame(population_clean)
population_geo = population_geo.reset_index()

population_geo.replace({"GEO": "St. Catharines-Niagara, Ontario"}, "
population_geo.replace({"GEO": "Kitchener-Cambridge-Waterloo, Ontari
population_geo.replace({"GEO": "Kitchener-Cambridge-Waterloo, Ontari
```

Out[94]:

	GEO	REF_DATE_x	Labour force characteristics_x	Population 2020	REF_DATE_y	Labour force characteristics_y	Pop
0	St. John's, Newfoundland and Labrador	2020-03	Population	187.0	2015-03	Population	
1	Halifax, Nova Scotia	2020-03	Population	379.6	2015-03	Population	
2	Moncton, New Brunswick	2020-03	Population	133.0	2015-03	Population	
3	Saint John, New Brunswick	2020-03	Population	107.7	2015-03	Population	
4	Saguenay, Quebec	2020-03	Population	133.4	2015-03	Population	
5	Québec, Quebec	2020-03	Population	690.1	2015-03	Population	

In [95]:
```
# convert to geodataframe
import geopandas as gpd
import geopy as gpy
from geopandas.tools import geocode
from geopy.geocoders import Nominatim

population_geo = gpd.GeoDataFrame(population_clean)
population_geo = population_geo.reset_index()

population_geo.replace({"GEO": "St. Catharines-Niagara, Ontario"}, "
population_geo.replace({"GEO": "Kitchener-Cambridge-Waterloo, Ontari
population_geo.replace({"GEO": "Kitchener-Cambridge-Waterloo, Ontari
```

Out[95]:

	GEO	REF_DATE_x	Labour force characteristics_x	Population 2020	REF_DATE_y	Labour force characteristics_y	Po

0	St. John's, Newfoundland and Labrador	2020-03	Population	187.0	2015-03	Population
1	Halifax, Nova Scotia	2020-03	Population	379.6	2015-03	Population
2	Moncton, New Brunswick	2020-03	Population	133.0	2015-03	Population
3	Saint John, New Brunswick	2020-03	Population	107.7	2015-03	Population
4	Saguenay, Quebec	2020-03	Population	133.4	2015-03	Population
5	Québec, Quebec	2020-03	Population	690.1	2015-03	Population
6	Sherbrooke, Quebec	2020-03	Population	189.4	2015-03	Population
7	Trois-Rivières, Quebec	2020-03	Population	135.6	2015-03	Population
8	Montréal, Quebec	2020-03	Population	3546.1	2015-03	Population
9	Kingston, Ontario	2020-03	Population	147.4	2015-03	Population
10	Peterborough, Ontario	2020-03	Population	108.4	2015-03	Population
11	Oshawa, Ontario	2020-03	Population	350.0	2015-03	Population
12	Toronto, Ontario	2020-03	Population	5649.2	2015-03	Population
13	Hamilton, Ontario	2020-03	Population	684.6	2015-03	Population
14	St. Catharines-Niagara, Ontario	2020-03	Population	360.2	2015-03	Population
15	Kitchener, Ontario	2020-03	Population	454.0	2015-03	Population
16	Brantford, Ontario	2020-03	Population	119.2	2015-03	Population
17	Guelph, Ontario	2020-03	Population	141.5	2015-03	Population
18	London, Ontario	2020-03	Population	452.6	2015-03	Population
	Windsor,					

19	Ontario	2020-03	Population	298.3	2015-03	Population
20	Barrie, Ontario	2020-03	Population	182.4	2015-03	Population
21	Greater Sudbury, Ontario	2020-03	Population	141.3	2015-03	Population
22	Thunder Bay, Ontario	2020-03	Population	104.6	2015-03	Population
23	Winnipeg, Manitoba	2020-03	Population	704.3	2015-03	Population
24	Regina, Saskatchewan	2020-03	Population	215.0	2015-03	Population
25	Saskatoon, Saskatchewan	2020-03	Population	279.9	2015-03	Population
26	Calgary, Alberta	2020-03	Population	1292.8	2015-03	Population
27	Edmonton, Alberta	2020-03	Population	1203.0	2015-03	Population
28	Kelowna, British Columbia	2020-03	Population	168.5	2015-03	Population
29	Abbotsford-Mission, British Columbia	2020-03	Population	160.1	2015-03	Population
30	Vancouver, British Columbia	2020-03	Population	2302.5	2015-03	Population
31	Victoria, British	2020-03	Population	324.8	2015-03	Population

In [96]:

```
# - the following code will time out, so the file was loaded in and
# geo = geocode(population_geo["GEO"], provider = "nominatim")
# geo
```

In [97]:
```
# - write to file
geo.to_file("cleaned_data/city_coords.json", driver="GeoJSON")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-97-d11604e34d95> in <module>
      1 # - write to file
----> 2 geo.to_file("cleaned_data/city_coords.json", driver="GeoJSON")

NameError: name 'geo' is not defined
```

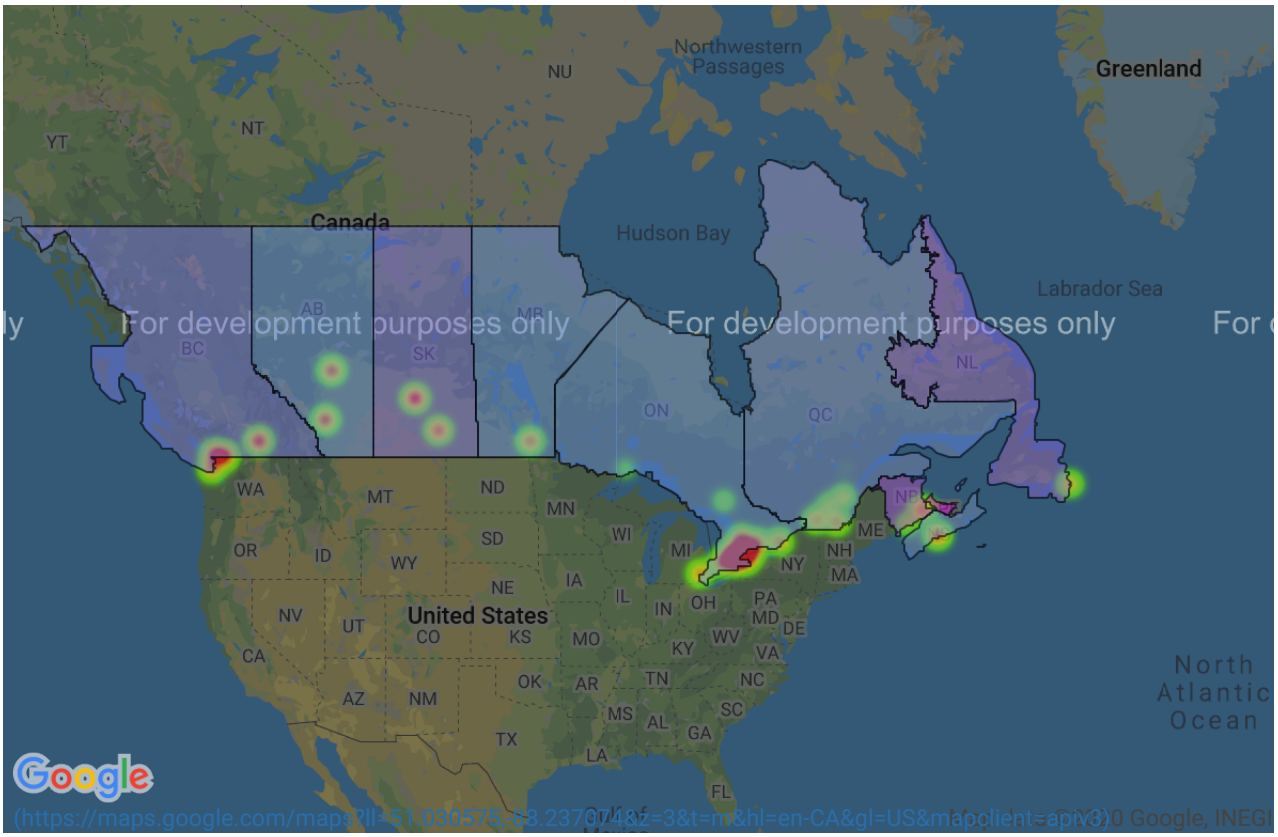
In [98]:
```
# - merge population change as weights for heatmap
population_locations = gpd.read_file("cleaned_data/city_coords.json"
population_locations["Population change"] = population_geo["Populati
population_locations["coordinates"] = (population_locations["geometry
# population_locations.to_file("cleaned_data/copied_data/city_coords_
population_locations.head()
```

Out[98]:

	address	geometry	Population change	coordinates
0	St. John's, Newfoundland, Newfoundland and Lab...	POINT (-52.71515 47.56170)	65.561798	POINT (-52.71515 47.56170)
1	Halifax, Halifax County, Nova Scotia, Canada	POINT (-63.58595 44.64862)	109.894722	POINT (-63.58595 44.64862)
2	Moncton, Moncton Parish, New Brunswick, Canada	POINT (-64.80011 46.09799)	94.545455	POINT (-64.80011 46.09799)
3	Saint John, City of Saint John, Saint John Cou...	POINT (-66.05804 45.27875)	34.886364	POINT (-66.05804 45.27875)
4	Saguenay, Saguenay–Lac-Saint-Jean, Québec, G7H...	POINT (-71.06918 48.40596)	2.416728	POINT (-71.06918 48.40596)

In [99]:

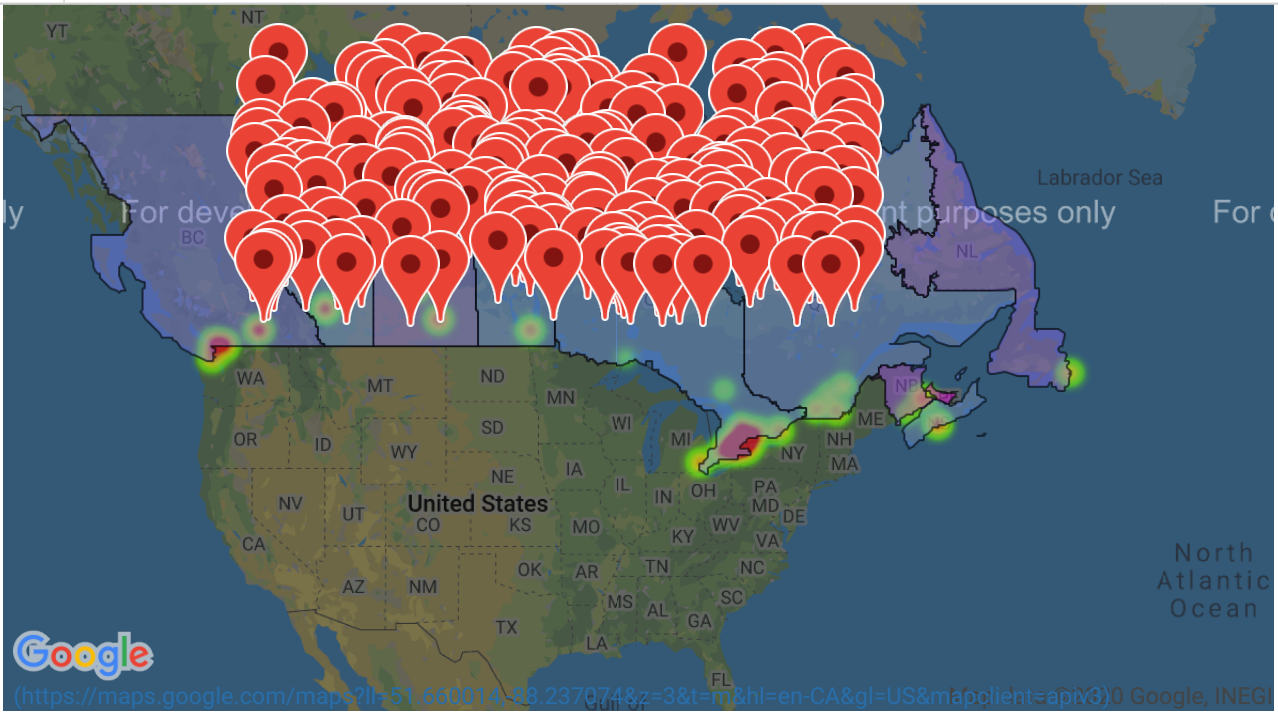
```
# add population flow layer
s = gpd.GeoSeries(population_locations["geometry"])
x_coords = s.apply(lambda p: p.x)

s = gpd.GeoSeries(population_locations["geometry"])
y_coords = s.apply(lambda p: p.y)

locations = pd.DataFrame(y_coords)
locations["longitude"] = x_coords
locations.rename(columns = {"geometry":'latitude'}, inplace = True)


weights = population_locations["Population change"]

fig.add_layer(gmaps.heatmap_layer(locations, weights = weights))

fig
# more intense colours show that more people have immigrating been t
```



In [100]:

```
# - plotting an arbitrary portfolio

# get random real estate portfolio

np.random.seed(123)
# get holding_type column
holding_type = ["Residential"]*50 + ["Industrial"]*50 + ["Office"]*5(
portfolio = {"holding_type": holding_type}
portfolio = pd.DataFrame.from_dict(portfolio)

# get market value column
portfolio['mkt_value'] = np.random.randint(100_000, 50_000_000, port:


#get random coordinates x
x_coords = np.random.randint(5000, 6000, 200)/100
portfolio['x_coords'] = x_coords


#get random coordinates y
y_coords = np.random.randint(-12000, -7000, 200)/100
portfolio['y_coords'] = y_coords


# get in point format
portfolio["coords"] = gpd.GeoSeries([Point(x, y) for x, y in zip(x_c(
portfolio["location"] = ([(x, y) for x, y in zip(x_coords, y_coords)

portfolio.to_csv("cleaned_data/portfolio")

locs = portfolio["location"]

mkt_vals = [f"${str(i)}" for i in list(portfolio["mkt_value"])]

holding_type = list(portfolio["holding_type"])

def join_lists(list1, list2):
    n = 0
    acc = []
    for i in list1:
        info = (f"Type = {list1[n]}, Price = {list2[n]}")
        acc = acc + [info]
        n = n + 1
    return acc

holding_info = join_lists(holding_type, mkt_vals)

fig.add_layer(gmaps.marker_layer(locations = locs, info_box_content =
```

In [101]:
```
1 # show figure
2 fig
```



In [ ]:
```
1
```