Author: Alexandre Ray

Date: September 01, 2021

Authenticate with google drive

First, you have to put the datasets inside your google drive folder and then authenticate it using the code below in order to access the datasets inside the folders and manipulate them in this notebook.

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

Install libraries

Some of the libraries are not default. So, you will need to install them manually using the code below:

```
!pip install fiona
!pip install geojson
!pip install osgeo
!pip install geopandas
!pip install geojsonio
!pip install geoplot
!pip install requests
```

```
Requirement already satisfied: uritemplate>=3.0.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: jwcrypto>=0.5.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: python-dateutil>=2.6.0 in /usr/local/lib/python3.7/
Requirement already satisfied: cryptography>=2.3 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: deprecated in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/loc
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist
Requirement already satisfied: wrapt<2,>=1.10 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: geoplot in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: geopandas in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: mapclassify>=2.1 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: contextily>=1.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: cartopy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: rasterio in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages (f
```

```
Requirement already satisfied: geopy in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: mercantile in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: numpy>=1.3 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: networkx in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pyshp>=2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: shapely>=1.5.6 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: fiona>=1.8 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: munch in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-package

Requirement already satisfied: geographiclib<2,>=1.49 in /usr/local/lib/python3.7/
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/lo
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: snuggs>=1.4.1 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: affine in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/loc
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/loc
```

Import libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly
import plotly.offline as plty
plt.rc('axes', labelsize=14)    # fontsize of the x and y labels
pd.set_option('display.max_column', 150)

import fiona
import geojson
import geopandas as gpd
import geojsonio
from IPython.display import HTML
import matplotlib.pyplot as plt
```

About the data sources

In this project, I will use two data sources: **IBGE** and **Geosampa**.

IBGE:

The Brazilian Institute of Geography and Statistics (Portuguese: Instituto Brasileiro de Geografia e Estatística; IBGE) is the agency responsible for official collection of statistical, geographic, cartographic, geodetic and environmental information in Brazil. IBGE performs a decennial national census; questionnaires account for information such as age, household income, literacy, education, occupation and hygiene levels.

References:

- What is IBGE?
- Official website

Geosampa:

The Geosampa Portal is a portal that follows the guidelines of the Strategic Master Plan, bringing together georeferenced data about the city of São Paulo, including about 12 thousand urban facilities, public transport network, geotechnical maps and important data about the population, such as density demographic and social vulnerability.

References:

- What is Geosampa?
- Official website

In order to understand the data, it is important to define what is **census sector**:

Census sector: According with IBGE, census sector is the smaller territorial portion used by IBGE to plan and carry out data surveys of the Census and Statistical Surveys, the Census Sector. This corresponds to a section of the national territory, considering the Political-Administrative Division and other territorial structures, which allows for the collection of statistical information within the period determined for collection.

The census sector will be used throughout this notebook.

Objetives

1. Study socioeconomic aspects versus accessibility to subway stations of the city of São Paulo.
2. Learn how to use and manipulate geolocation data using Python (geopandas and geojson).

obs: This is a simplification for this work. The data sources provide much more information which

Import datasets

```
# put your datasets inside this folder
path = 'drive/MyDrive/TAU/notebooks/data'

# ----------------------------------------------------
# IBGE data
# ----------------------------------------------------

# geographic data
sp_state = gpd.read_file(f"{path}/33SEE250GC_SIR.dbf")

# income data
df_income = pd.read_csv(f"{path}/PessoaRenda_SP1.csv", sep=';')

# ----------------------------------------------------
# Geosampa data
# ----------------------------------------------------

# census sector of sp city
sp_city = gpd.read_file(f"{path}/SAD6996_SETOR_CENSITARIO_2010.shp")

# metro station of sp city
sp_city_metro = gpd.read_file(f"{path}/SAD69-96_SHP_estacaometro_point.dbf")
```

State of São Paulo

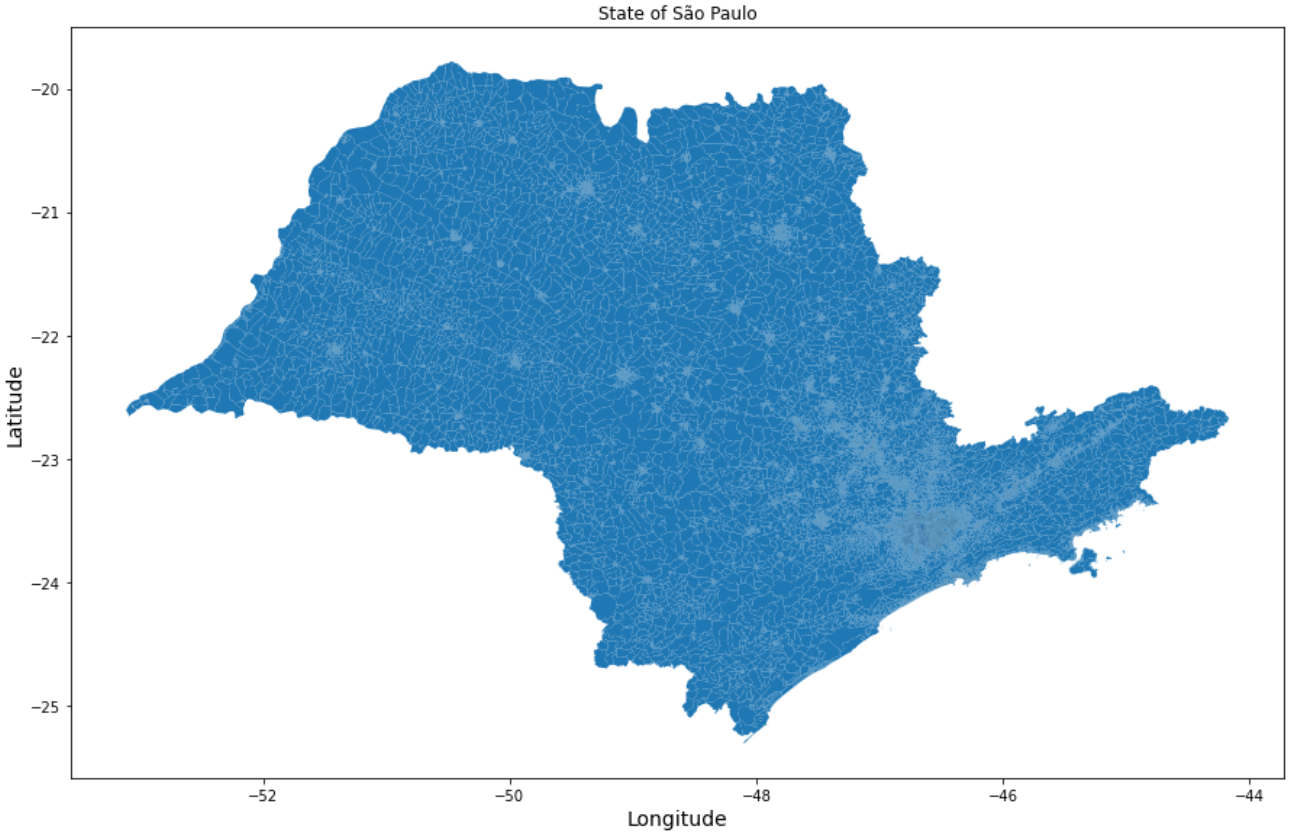
```
type(sp_state)
```

```
geopandas.geodataframe.GeoDataFrame
```

```
sp_state.head(3)
```

ID	CD_GEOCODI	TIPO	CD_GEOCODS	NM_SUBDIST	CD_GEOCODD	NM_DISTRI
						PRAI

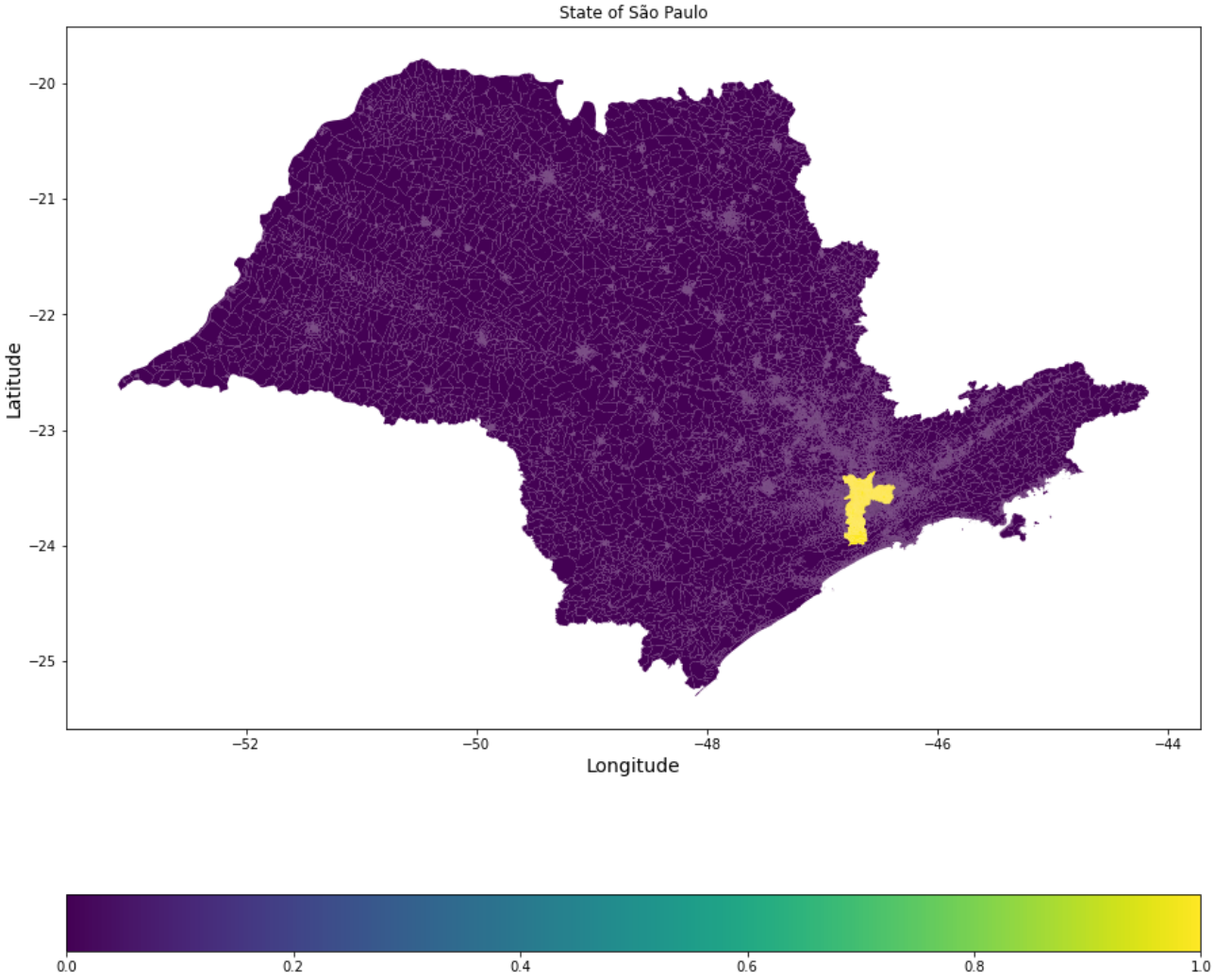
```
fig, ax = plt.subplots(1, 1, figsize=(15, 15))

sp_state.plot(ax=ax, legend=True)
plt.title("State of São Paulo")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```



City of São Paulo

```
sp_state['is_sp'] = sp_state['NM_MUNICIP'] == 'SÃO PAULO'
sp_state['is_sp'] = sp_state['is_sp'].astype(int)


fig, ax = plt.subplots(1, 1, figsize=(15, 15))
```

```
sp_state.plot(column='is_sp', ax=ax, legend=True,
        legend_kwds={'orientation': 'horizontal'})
plt.title("State of São Paulo")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```



São Paulo Metro

```
sp_city_metro.head()
```

	emt_empres	emt_situac	emt_linha	emt_nome	geometry
0	METRO	OPERANDO	VERMELHA	CORINTHIANS-ITAQUERA	POINT (349884.430 7395720.764)
1	METRO	OPERANDO	VERMELHA	ARTUR ALVIM	POINT (348502.644 7395929.882)
2	METRO	OPERANDO	VERMELHA	PATRIARCA	POINT (346777.766 7396920.334)

```
sp_city_metro['emt_nome'].nunique()
```

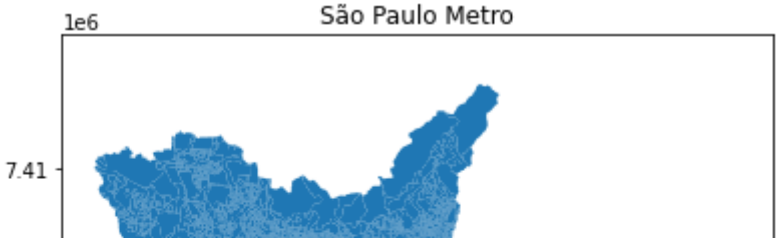
```
84
```

```
sp_city_metro['emt_situac'].value_counts()
```

```
OPERANDO    93
Name: emt_situac, dtype: int64
```

São Paulo has 84 operating metro stations. The following plot shows the stations:

```
fig, ax = plt.subplots(1, 1, figsize=(10, 10))

base = sp_city.plot(ax=ax, legend=True)
sp_city_metro.geometry.plot(ax=base, marker='o', color='red', markersize=10)
plt.title("São Paulo Metro")
plt.show()
```




```
print(f"how many census sector the city of são paulo has? {len(sp_city)}")
print(f"how many metro stations? {len(sp_city_metro)}")
print(f"# census sector X # metro stations? {len(sp_city) * len(sp_city_metro)}")
```

```
how many census sector the city of são paulo has? 18953
how many metro stations? 93
# census sector X # metro stations? 1762629
```

Distance between regions and metro stations

For each census sector, we calculate the distance between the **centroid** of the census sector and the **nearest metro station**. With this distance, we will investigate futher details about the accessibility of different regions by socieconomic behavior.

The regions are defined as a polygon and the centroid is been used as an approximation as the census sector tends to be a small area of the city (smaller then a district).


```
# calculate the centroid of each census sector
sp_city['centroid'] = sp_city.geometry.centroid
```

```
# the result is an object POINT. Now, we can calculate the distance between two POINTs
sp_city.centroid.head(1)
```

```
0    POINT (313956.205 7410298.593)
dtype: geometry
```

```
# metro station (it is also an object POINT)
sp_city_metro.geometry.head(1)
```

```
0    POINT (349884.430 7395720.764)
Name: geometry, dtype: geometry
```

```
sp_city.head()
```

	geometry	centroid

```
sp_city_metro.head()
```

	emt_empres	emt_situac	emt_linha	emt_nome	geometry
0	METRO	OPERANDO	VERMELHA	CORINTHIANS-ITAQUERA	POINT (349884.430 7395720.764)
1	METRO	OPERANDO	VERMELHA	ARTUR ALVIM	POINT (348502.644 7395929.882)
2	METRO	OPERANDO	VERMELHA	PATRIARCA	POINT (346777.766 7396920.334)
	METRO	OPERANDO	VERMELHA	GUILHERMINA-	POINT (345227.021

In order to accelerate calculations, I'll use just urban areas.

```
# The following lines can take time to run!

# def calculate_distance(centroid_census_sector):
#   dist = sp_city_metro.geometry.distance(centroid_census_sector)

#   return dist.min()

# for idx, row in sp_city_urban.iterrows():
#   sp_city_urban['nearest_metro'] = sp_city_urban['centroid'].apply(calculate_distance)

# this is calculated before because it takes time to run
#del sp_city_urban
sp_city = pd.read_csv(f"{path}/sf_setor.csv", index_col=0)

sp_city_urban = sp_city.loc[sp_city.TIPO == 'URBANO']
```

```
sp_city_urban.head()
```

	ID	AREA_M	CODSETOR	TIPO	POPULACAO	IPVS_V10	geomet
0	1	311095.19	355030803000062	URBANO	NaN	NaN	POLYGC ((313628.77097681 7410329.632962719
6	7	410829.23	355030861000118	URBANO	NaN	NaN	POLYGC ((319191.70997522 7412432.592812435
	9	1263124.28	355030861000083	URBANO	20.0	0.0	POLYGC ((320681.53169651

```
sp_city_urban.nearest_metro.describe(percentiles=[0.05, 0.25, 0.5, 0.75, 0.85, 0.90, 0.95]
```

```
count    18228.000000
mean      3885.375454
std       3643.817859
min         27.451046
```

```
5%         395.169257
25%       1209.194173
50%       2709.056134
75%       5483.815784
85%       7585.437206
90%       8809.370437
95%      11074.933671
max      28179.256245
Name: nearest_metro, dtype: float64
```

We see that the smallest distance between a metro station and a census sector is 27 meters and the greatest distance is 28 kilometers.

50% of the regions have access to a metro station within 2.7 kilometers which means about 30 minutes on foot.

Central vs extreme census sectors considering metro stations

We are going to measure the 5% further and the 5% closer census sector considering the distance between the centroids and the metro stations. After that, we will prepare the data to plot the result using geolocation data and geopandas library.

```
# 5% census sector more distant of a metro station
more_distant = sp_city_urban.loc[sp_city_urban.nearest_metro > 11074]
more_distant.shape
```

```
(912, 10)
```

```
# 5% census sector closer to a metro station
closer = sp_city_urban.loc[sp_city_urban.nearest_metro < 395]
closer.shape
```

```
(908, 10)
```

```
sp_city_urban['census_sector_away'] = sp_city_urban['CODSETOR'].isin(more_distant.CODSETOR
sp_city_urban['census_sector_away'] = sp_city_urban['census_sector_away'].astype(int)

sp_city_urban['census_sector_next'] = sp_city_urban['CODSETOR'].isin(closer.CODSETOR.tolis
sp_city_urban['census_sector_next'] = sp_city_urban['census_sector_next'].astype(int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarnir


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/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarnir


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

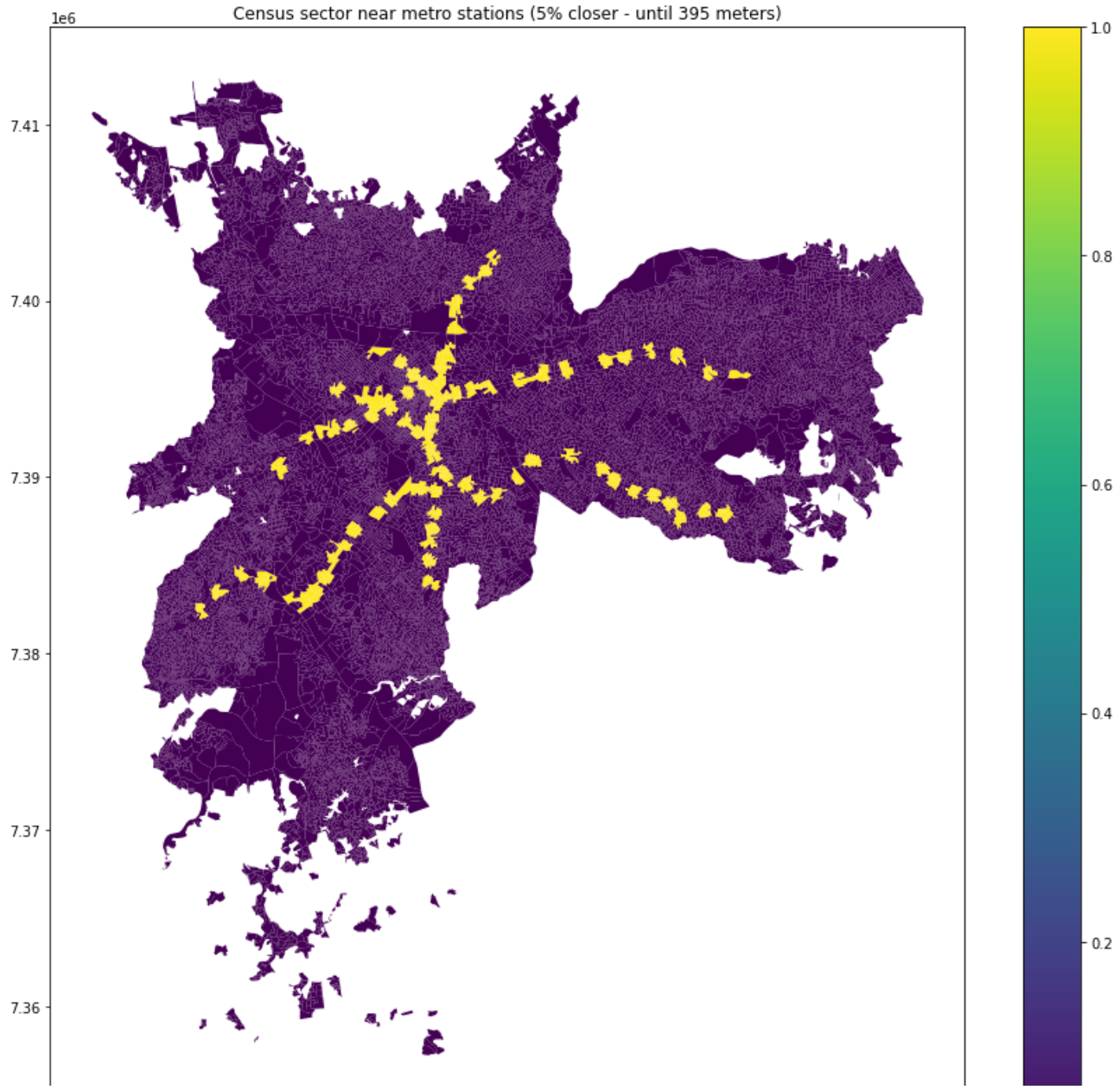
```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarnir


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/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarnir


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/us
```

```
sp_city_urban['geometry'] = gpd.GeoSeries.from_wkt(sp_city_urban['geometry'])
sp_city_urban = gpd.GeoDataFrame(sp_city_urban, geometry='geometry')
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarnir


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/us
```

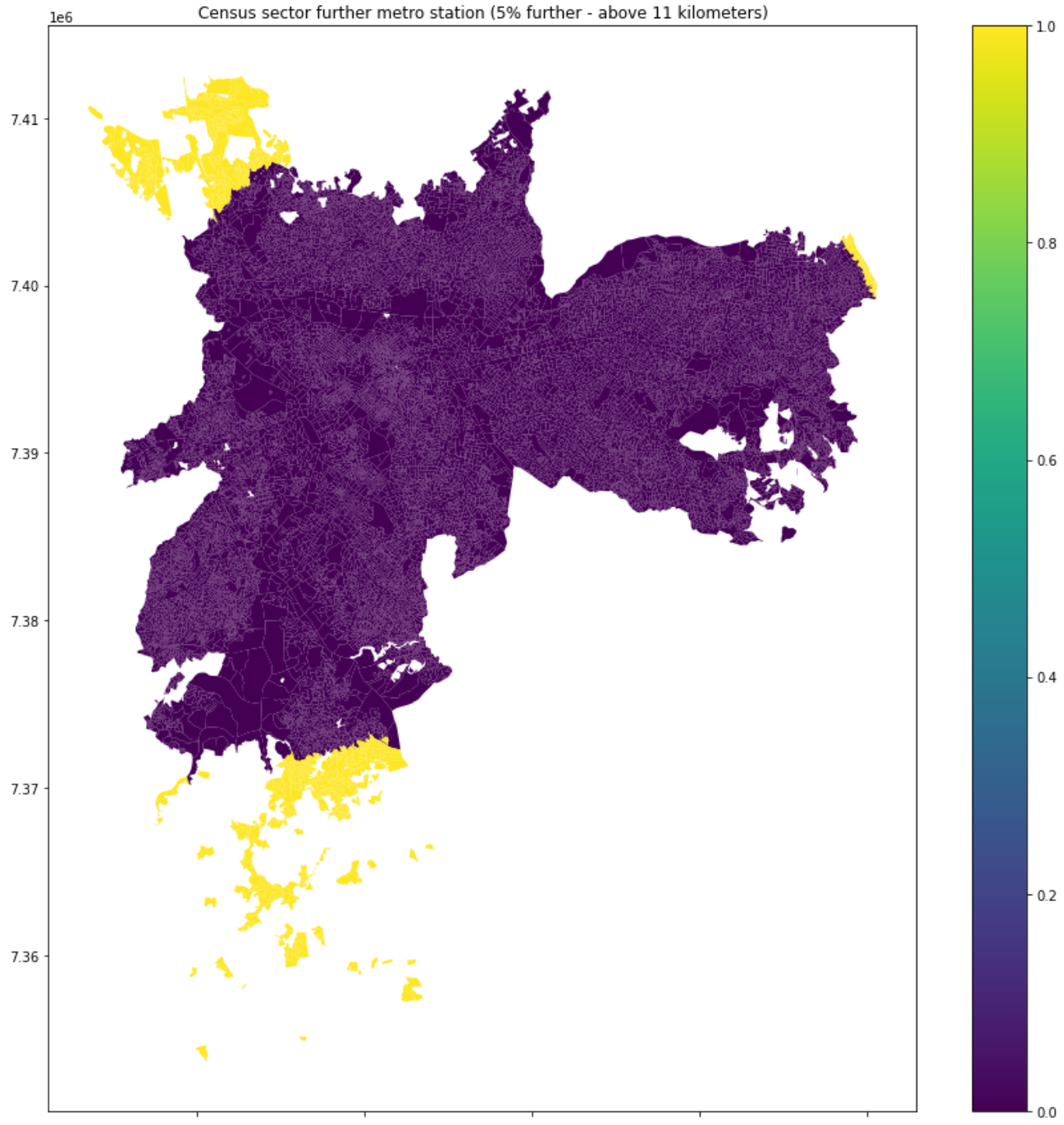
Census sector near metro station (5% closer - until 395 meters)

```
fig, ax = plt.subplots(1, 1, figsize=(15, 15))
sp_city_urban.plot(column='census_sector_next', ax=ax, legend=True)
plt.title("Census sector near metro stations (5% closer - until 395 meters)")
plt.show()
```



Census sector further metro station (5% further - above 11 kilometers)

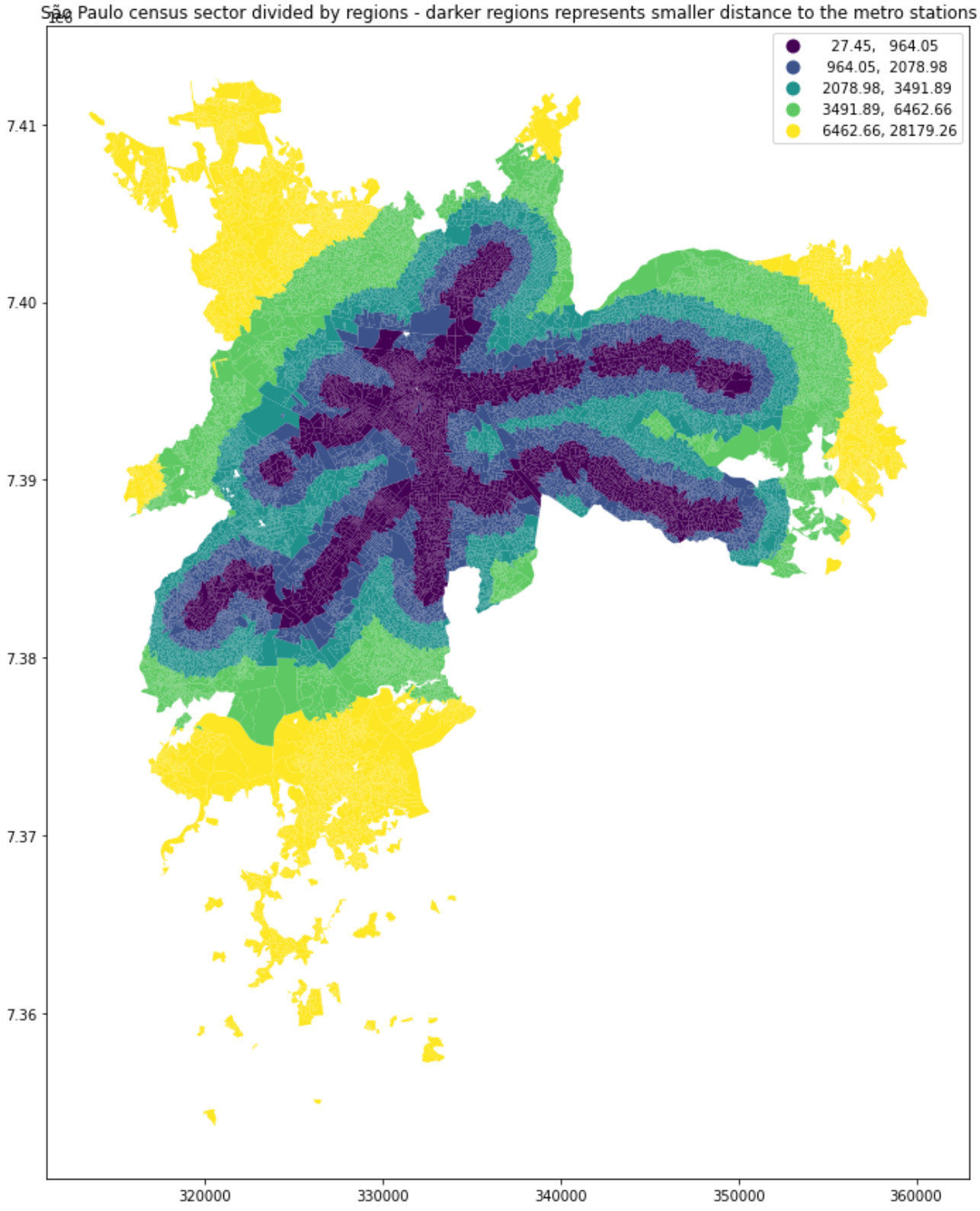
```
fig, ax = plt.subplots(1, 1, figsize=(15, 15))
sp_city_urban.plot(column='census_sector_away', ax=ax, legend=True)
plt.title("Census sector further metro station (5% further - above 11 kilometers)")
plt.show()
```



We clearly can see the regions further from the metro stations marked with yellow in the plot above. It corresponds to the 5% further census sector of the city of São Paulo. The areas involves extreme south, northwest and east of the city.

Quartile analysis - darkers regions demonstrates more accessibility to the metro station

```
fig, ax = plt.subplots(1, 1, figsize=(15, 15))
sp_city_urban.plot(column='nearest_metro', ax=ax, legend=True, scheme='quantiles')
plt.title("São Paulo census sector divided by regions - darker regions represents smaller
plt.show()
```



IBGE - income data

Now, we will analyse regions using socioeconomic data collected from IBGE.

The dataset below corresponds to income data for each census sector (key: **CODSETOR**)

The documentation and meaning of the variables can be found in the IBGE website.

```
df_income.head()
```

```
df_income.head()
```

	Cod_setor	Situacao_setor	V001	V002	V003	V004	V005	V006	V007	V008	V
0	355030801000001	1	2	93	180	92	82	63	9	3	
1	355030801000002	1	6	138	205	85	72	56	4	4	
2	355030801000003	1	1	32	102	46	67	19	5	9	
3	355030801000004	1	2	24	126	56	41	28	4	2	
4	355030801000005	1	3	99	168	81	59	35	5	3	

Merge Geosampa with IBGE (using census code key)

The first thing is to merge the IBGE data (socioeconomig data) with the Geosampa data. For this task, I will use the census sector code.

```
df_income.rename(columns={'Cod_setor': 'CODSETOR'}, inplace=True)

sp_city_urban.CODSETOR = sp_city_urban.CODSETOR.astype(int)

df = pd.merge(sp_city_urban, df_income, how='left', on='CODSETOR')

# choose some variables
income_features = ['CODSETOR', 'AREA_M', 'POPULACAO', 'nearest_metro', 'V011', 'V012', 'V0

df[income_features].head()
```

	CODSETOR	AREA_M	POPULACAO	nearest_metro	V011	V012	V013	V014	V01
0	355030803000062	311095.19	NaN	20384.719766	NaN	NaN	NaN	NaN	Na
1	355030861000118	410829.23	NaN	17278.520044	NaN	NaN	NaN	NaN	Na
2	355030861000083	1263124.28	20.0	16857.042761	X	X	X	X	
3	355030861000112	280118.32	NaN	16070.752726	NaN	NaN	NaN	NaN	Na
4	355030861000078	1513215.48	12.0	15550.383909	0	510	3408	0	20(

For simplification, some variables were chosen (V011 to V019). Follow the description of them:

Variable	Description
V011	Total nominal monthly income of persons aged 10 years and over with monthly nominal income of up to ½ minimum v
V012	Total monthly nominal income of persons aged 10 years and over with monthly nominal income of more than 1-2 to 1 r
V013	Total monthly nominal income of persons aged 10 years and over with monthly nominal income of more than 1 to 2 mi
V014	Total monthly nominal income of persons aged 10 years and over with monthly nominal income of more than 2 to 3 mi

Variable	Description
V015	Total monthly nominal income of persons aged 10 years and over with monthly nominal income of more than 3 to 5 mi
V016	Total monthly nominal income of persons aged 10 years and over with monthly nominal income of more than 5 to 10 m
V017	Total monthly nominal income of persons aged 10 years and over with monthly nominal income of more than 10 to 15
V018	Total monthly nominal income of persons aged 10 years or over with monthly nominal income of more than 15 to 20 m
V019	Total monthly nominal income of persons aged 10 years and over with monthly nominal income of more than 20 minim

Low income population

```
df.shape
```

```
(18228, 146)
```

```
vars = ['V011', 'V012', 'V013', 'V014', 'V015', 'V016', 'V017', 'V018', 'V019']
```

```
# Some rows are null
df[vars].isna().sum()
```

```
V011    317
V012    317
V013    317
V014    317
V015    317
V016    317
V017    317
V018    317
V019    317
dtype: int64
```

```
# Some rows are filled with X. Why?
(df[vars] == 'X').sum()
```

```
V011    71
V012    71
V013    71
V014    71
V015    71
V016    71
V017    71
V018    71
V019    71
dtype: int64
```

I will drop this rows with NA and X values below.

```
def handle_income_variables(df, vars):
  df = df.copy()
  for i in vars:
    df.drop(index=df.loc[(df[i].isna())].index.tolist(), inplace=True)
    df.drop(index=df.loc[(df[i] == 'X')].index.tolist(), inplace=True)
```

```
    df[i] = df[i].astype(int)

  return df

df = handle_income_variables(df, vars)

df.shape
```

```
(17840, 146)
```

Group income (Low, Medium and High)

In order to reduce the dimensionality of this data (9 variables). I will group them in Low, Medium and High income using the following formulas:

> $$low\_income = V011 + V012 + V013$$
> $$medium\_income = V014 + V015 + V016$$
> $$high\_income = V017 + V018 + V019$$

```
df['low_income'] = df['V011'] + df['V012'] + df['V013']
df['medium_income'] = df['V014'] + df['V015'] + df['V016']
df['high_income'] = df['V017'] + df['V018'] + df['V019']
```

Using this columns, I will do two data analysis using geolocation data:

1) People which can access subway 30 minutes on foot by income

2) People which can access subway above 5 km by income

---

1) People which can access subway 30 minutes on foot by income

obs: Assuming that a person can walk about 2.3 kilometers in 30 minutes.

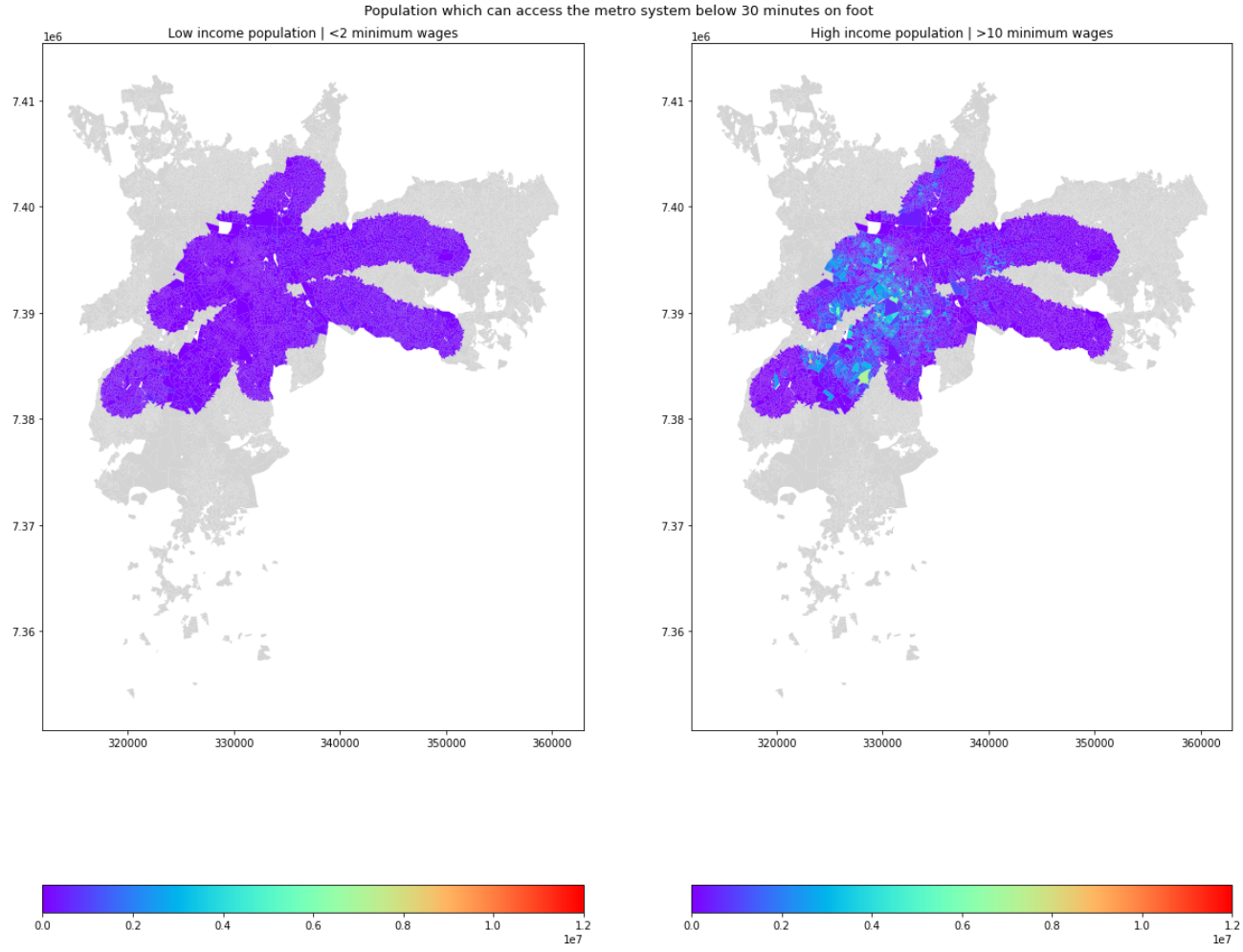
```
df['below_30_minutes_on_foot'] = df['nearest_metro'].apply(lambda x: True if x < 2300 else
df['below_30_minutes_on_foot'] = df['below_30_minutes_on_foot'].astype(int)

df_sample = df[['geometry', 'below_30_minutes_on_foot', 'low_income', 'medium_income', 'hi
print(df_sample.shape)
df_sample.head()
```

```
(17840, 5)
```

	geometry	below_30_minutes_on_foot	low_income	medium_income	high_income
4	POLYGON ((323845.814 7411697.682, 323855.562 7...	0	3918	5000	0
5	POLYGON ((324268.485	0	193204	153270	0

```
fig, ax = plt.subplots(1, 2, figsize=(20, 20))

vmin=0
vmax=12000000

base1 = df_sample.plot(column='low_income', ax=ax[0], color='lightgrey')
base2 = df_sample.plot(column='high_income', ax=ax[1], color='lightgrey')

fig.subplots_adjust(top=0.8)
fig.suptitle("Population which can access the metro system below 30 minutes on foot", font

df_sample.loc[df_sample['below_30_minutes_on_foot'] == 1].plot(column='low_income',
                                                               ax=base1,
                                                               legend=True,
                                                               legend_kwds={'orientation':
                                                               cmap = 'rainbow', vmin=vmin

df_sample.loc[df_sample['below_30_minutes_on_foot'] == 1].plot(column='high_income',
                                                               ax=base2,
                                                               legend=True,
                                                               legend_kwds={'orientation':
                                                               cmap = 'rainbow', vmin=vmin
ax[0].set_title("Low income population | <2 minimum wages")
ax[1].set_title("High income population | >10 minimum wages")


plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/geopandas/plotting.py:642: UserWarning:

Only specify one of 'column' or 'color'. Using 'color'.

/usr/local/lib/python3.7/dist-packages/geopandas/plotting.py:642: UserWarning:

Only specify one of 'column' or 'color'. Using 'color'.
```

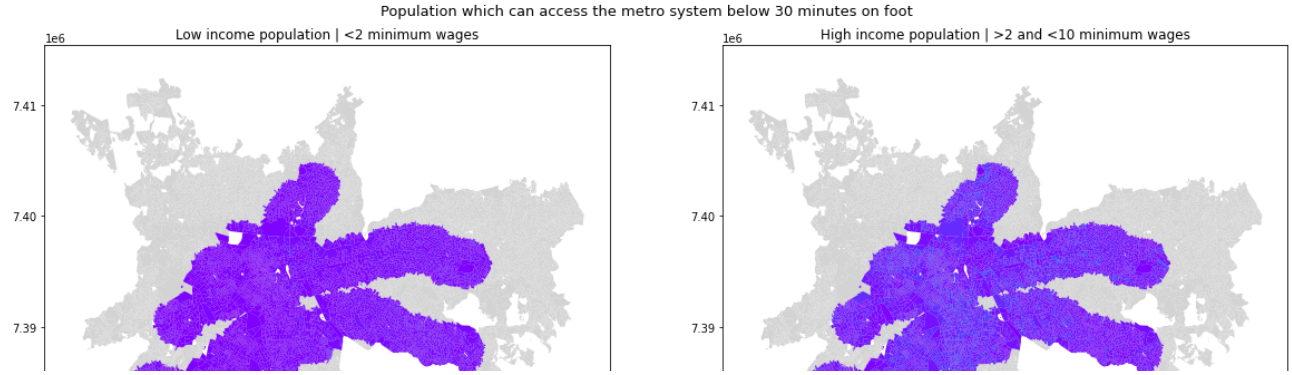


We can clearly evaluate that when considering people which can access metro stations easily (up to 30 minutes on foot), we observe higher concentration of high income (>10 minimum wages) in those census sectors.

```
fig, ax = plt.subplots(1, 2, figsize=(20, 20))

vmin=0
vmax=12000000

base1 = df_sample.plot(column='low_income', ax=ax[0], color='lightgrey')
base2 = df_sample.plot(column='medium_income', ax=ax[1], color='lightgrey')

fig.subplots_adjust(top=0.8)
fig.suptitle("Population which can access the metro system below 30 minutes on foot", font
```

```
df_sample.loc[df_sample['below_30_minutes_on_foot'] == 1].plot(column='low_income',
                                                               ax=base1,
                                                               legend=True,
                                                               legend_kwds={'orientation':
                                                               cmap = 'rainbow', vmin=vmin

df_sample.loc[df_sample['below_30_minutes_on_foot'] == 1].plot(column='medium_income',
                                                               ax=base2,
                                                               legend=True,
                                                               legend_kwds={'orientation':
                                                               cmap = 'rainbow', vmin=vmin
ax[0].set_title("Low income population | <2 minimum wages")
ax[1].set_title("Medium income population | >2 and <10 minimum wages")


plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/geopandas/plotting.py:642: UserWarning:

Only specify one of 'column' or 'color'. Using 'color'.

/usr/local/lib/python3.7/dist-packages/geopandas/plotting.py:642: UserWarning:

Only specify one of 'column' or 'color'. Using 'color'.
```



The behavior is similar to the previows graph. There are more people inside those census sectors classified as medium income than low income.

2) People which can access subway above 5 km by income

```
df['above_5km'] = df['nearest_metro'].apply(lambda x: True if x > 5000 else False)
df['above_5km'] = df['above_5km'].astype(int)


df_sample2 = df[['geometry', 'above_5km', 'low_income', 'medium_income', 'high_income']]
print(df_sample2.shape)
df_sample2.head()
```

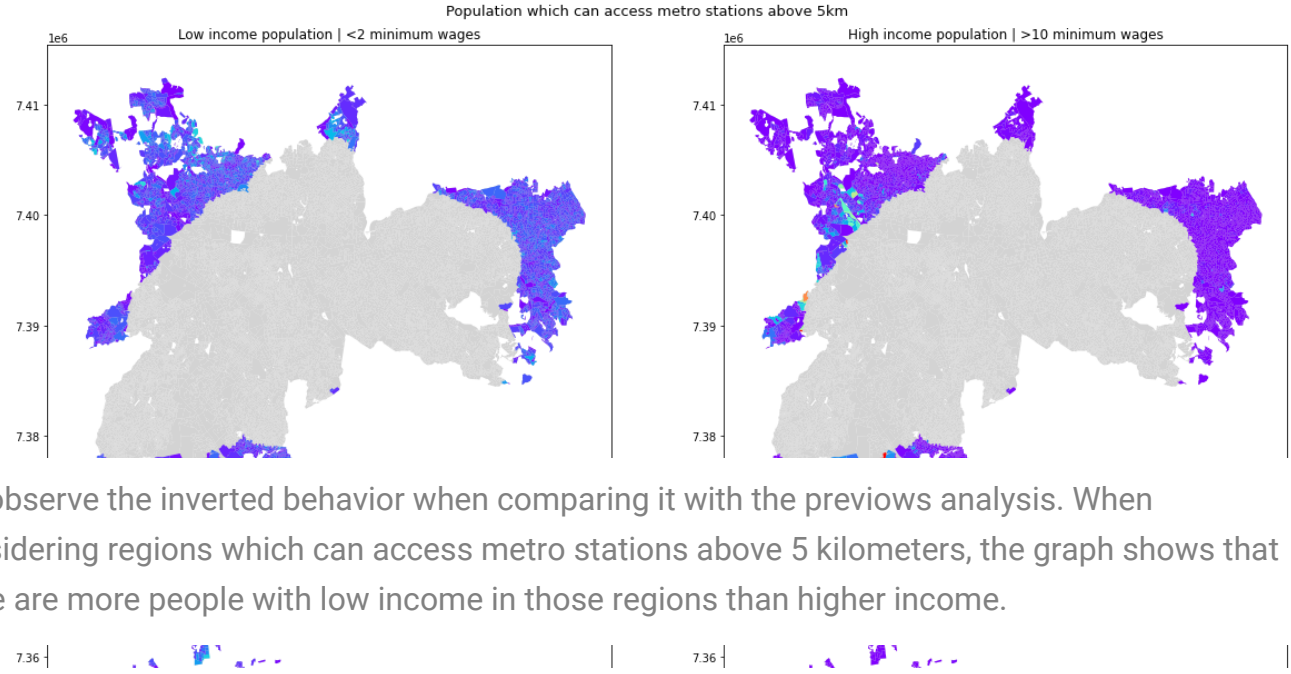
```
(17840, 5)
```

	geometry	above_5km	low_income	medium_income	high_income
4	POLYGON ((323845.814 7411697.682, 323855.562 7...	1	3918	5000	0
5	POLYGON ((324268.485 7411126.575, 324258.996 7...	1	193204	153270	0
6	POLYGON ((324252.505 7410983.620, 324253.811 7...	1	3150	6900	10000
	POLYGON ((340864.835				

```
fig, ax = plt.subplots(1, 2, figsize=(20, 20))

vmin=0
vmax=2000000

base1 = df_sample2.plot(column='low_income', ax=ax[0], color='lightgrey')
base2 = df_sample2.plot(column='high_income', ax=ax[1], color='lightgrey')

fig.subplots_adjust(top=0.8)
```

```
    fig.subplots_adjust(top=0.8)
    fig.suptitle("Population which can access metro stations above 5km", fontsize=13, y=0.80)

    df_sample2.loc[df_sample2['above_5km'] == 1].plot(column='low_income',
                                                      ax=base1,
                                                      legend=True,
                                                      legend_kwds={'orientation':
                                                      cmap = 'rainbow', vmin=vmin

    df_sample2.loc[df_sample2['above_5km'] == 1].plot(column='high_income',
                                                      ax=base2,
                                                      legend=True,
                                                      legend_kwds={'orientation':
                                                      cmap = 'rainbow', vmin=vmin
    ax[0].set_title("Low income population | <2 minimum wages")
    ax[1].set_title("High income population | >10 minimum wages")


    plt.show()
```

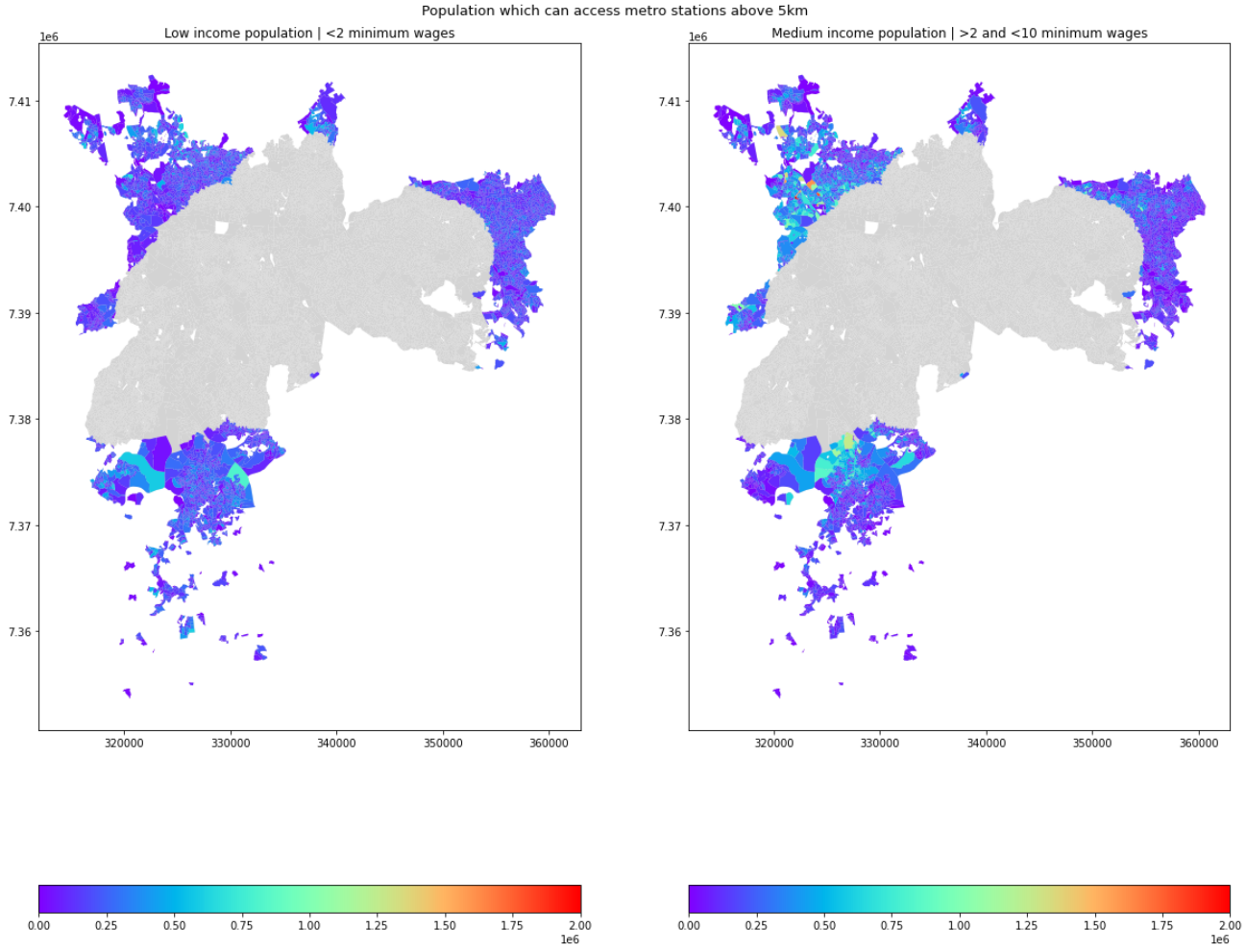
```
/usr/local/lib/python3.7/dist-packages/geopandas/plotting.py:642: UserWarning:

Only specify one of 'column' or 'color'. Using 'color'.

/usr/local/lib/python3.7/dist-packages/geopandas/plotting.py:642: UserWarning:

Only specify one of 'column' or 'color'. Using 'color'.
```



We observe the inverted behavior when comparing it with the previows analysis. When considering regions which can access metro stations above 5 kilometers, the graph shows that there are more people with low income in those regions than higher income.

```
fig, ax = plt.subplots(1, 2, figsize=(20, 20))

vmin=0
vmax=2000000

base1 = df_sample2.plot(column='low_income', ax=ax[0], color='lightgrey')
base2 = df_sample2.plot(column='medium_income', ax=ax[1], color='lightgrey')

fig.subplots_adjust(top=0.8)
fig.suptitle("Population which can access metro stations above 5km", fontsize=13, y=0.80)

df_sample2.loc[df_sample2['above_5km'] == 1].plot(column='low_income',
                                                  ax=base1,
                                                  legend=True,
                                                  legend_kwds={'orientation':
                                                  cmap = 'rainbow', vmin=vmin


df_sample2.loc[df_sample2['above_5km'] == 1].plot(column='medium_income',
                                                  ax=base2,
                                                  legend=True,
                                                  legend_kwds={'orientation':
                                                  cmap = 'rainbow', vmin=vmin
ax[0].set_title("Low income population | <2 minimum wages")
ax[1].set_title("Medium income population | >2 and <10 minimum wages")


plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/geopandas/plotting.py:642: UserWarning:

Only specify one of 'column' or 'color'. Using 'color'.

/usr/local/lib/python3.7/dist-packages/geopandas/plotting.py:642: UserWarning:

Only specify one of 'column' or 'color'. Using 'color'.
```



Some census sectors of south and northwest of São Paulo showed high concentration of people with medium income in those regions.

Conclusion

- In this notebook, we analysed data from the following data sources: IBGE and Geosampa. We used geolocation data and used the python library geopandas in order to analyse them.
- The main conclusion of the analysis is:

> **The accessibility of metro stations is directly related with income.** Other analysis could be done with train, bus, bicycle, etc which is available at Geosampa. Instead of income, it could be used age, sex, race and multiple other socioeconomical varibles avaliable at IBGE.

- Multiple data and variables were found in those data bases. Just a few of them were used in this work as a sample of what is possible to do with this data and tools.