EASC2410 Special Topic 3

Geospatial Data Analysis for the Covid-19 Pandemic: GeoPandas

Dr. Binzheng Zhang
Department of Earth Sciences

Review of Lecture 17:

- Mathematical modeling of Epidemics
- The exponential (R0) model
- The S-I model
- The S-I-S model (no immunity)
- The S-I-R model (immunity)
- The S-E-I-R model (HW assignment)

In Lecture 18, you will learn:

- Geospatial Data files
- The GeoPandas module
- Work with GeoDataFrames
- Visualisation Geospatial Data with Python

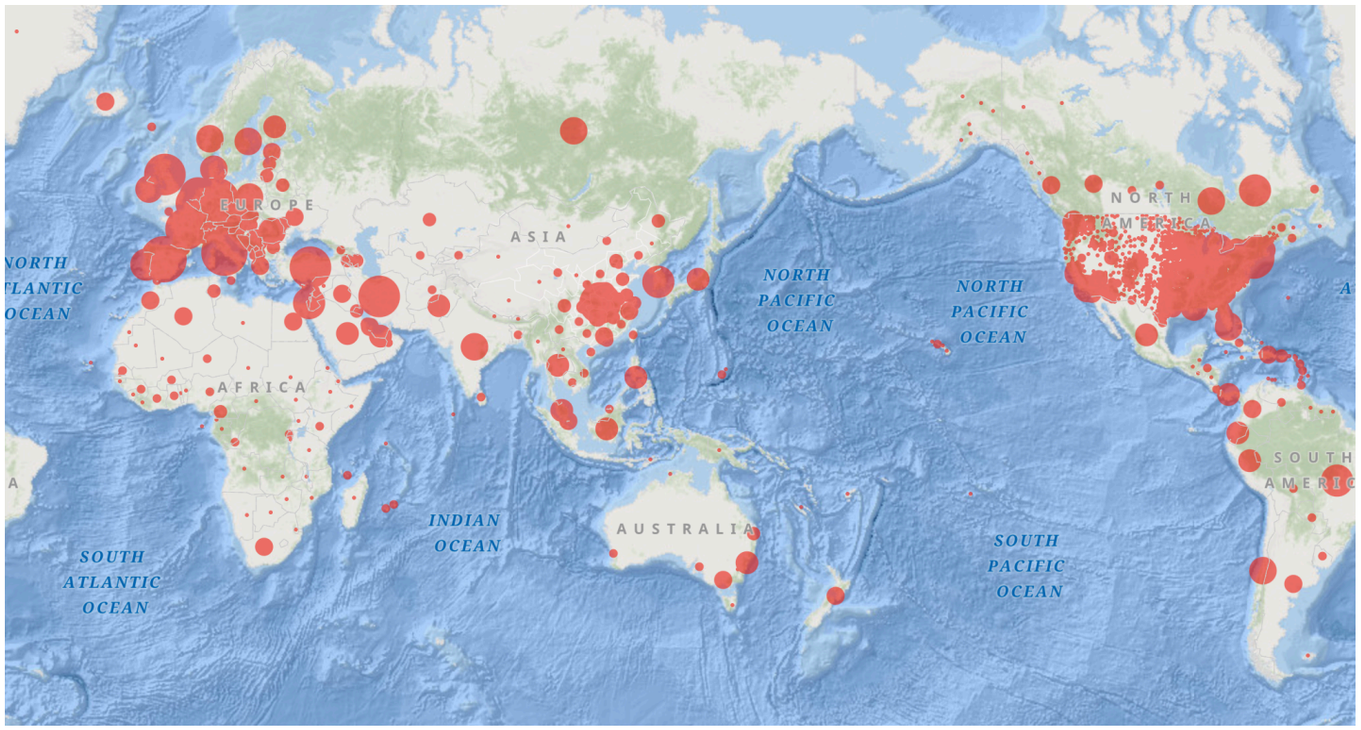
What is GeoSpatial Data?

Geospatial data comes in many forms and formats, and its structure is more complicated than tabular or even nongeographic geometric data. It is, in fact, a subset of spatial data, which is simply data that indicates where things are within a given *coordinate system*.

What sets geospatial data apart from other spatial data is that it is absolutely or relatively positioned on a planet, or *georeferenced*. That is, it has a *terrestrial coordinate system* that can be **shared** by other geospatial data. There are many ways to define a terrestrial coordinate system and also to transform it to any number of local coordinate systems, for example, to create a map projection.

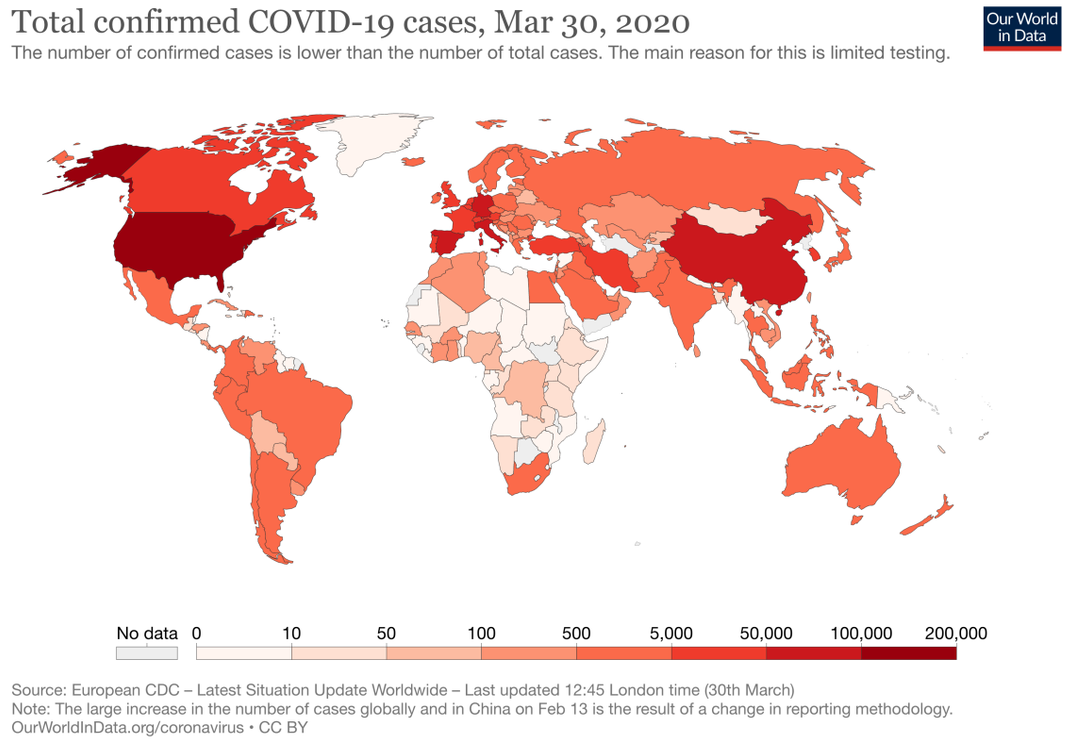
In Python, GeoSpatial data is basically DataFrame+Geometry

Example 1: JHU Coronavirus Map



Pandas + Basemap

Example 2: Map Distributions



GeoPandas

Why Python for Geospatial Data Analysis?

Python is extremely useful language to learn in terms of GIS since many (or most) of the different GIS Software packages (such as ArcGIS, QGIS, PostGIS etc.) provide an interface to do analysis using Python scripting. As an introduction, we will mostly focus on doing GIS without any third party softwares such as ArcGIS. **Why?** There are several reasons for doing GIS using Python without any additional software:

- **Everything is free**: you don't need to buy and expensive license for ArcGIS (for example)
- You will **learn and understand** much more deeply how different geoprocessing operations work
- Python is **highly efficient**: used for analysing Big Data
- Python is **highly flexible**: supports all data formats that you can imagine
- Using Python (or any other open-source programming language) **supports open source softwares/codes and open science** by making it possible for everyone to reproduce your work, free-of-charge.
- **Plug-in and chain different third-party softwares** to build e.g. a fancy web-GIS applications as you want (using e.g. GeoDjango with PostGIS as a back-end)

Map the cases of Covid-19 using a bubble plot

pandas + basemap

Step 1. the “total-and-daily-cases-covid-19.csv” file

```
fn = '../Datasets/total-and-daily-cases-covid-19.csv' # file name
df = pd.read_csv(fn) # load file into data frame
df.head() # show the first 10 rows

# first change the column names slightly
df.rename(columns={'Total confirmed cases (cases)':'TotalCases',\
                   'Daily new confirmed cases (cases)':'DailyCases'},\
                    inplace=True)

df_c = df[df.Date=='Apr 1, 2020']
df_c.head()
```

	Entity	Code	Date	TotalCases	DailyCases
82	Afghanistan	AFG	Apr 1, 2020	166	25
106	Albania	ALB	Apr 1, 2020	243	20
194	Algeria	DZA	Apr 1, 2020	584	73
213	Andorra	AND	Apr 1, 2020	376	6
224	Angola	AGO	Apr 1, 2020	7	0

Step 2. Create a world map using basemap

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap # import the basemap module using this syntax!

plt.figure(figsize=(20,15)) # create a new figure with size (16,15)

# make a map instance called 'm', pay attention to the boundaries of the map
m = Basemap(projection='merc',llcrnrlat=-65,urcrnrlat=65,llcrnrlon=-180,urcrnrlon=180)
m.drawcoastlines(); # put on the coastlines

m.drawparallels(np.arange(-90.,91.,30.), labels=[True,True,False,False]) # 
m.drawmeridians(np.arange(-180.,181.,60.),labels=[False,False,False,True]);

m.fillcontinents(color='white')
m.drawmapboundary(fill_color='lightblue');
m.drawcountries(color='black',linewidth=0.5)
plt.show()
```

How to map the number here?

Step 3. Get longitude, latitude for each number

Map the cases of Covid-19 using a bubble plot

Merge dataframes based on columns

Load the “geo.csv” file

```
fn = '../Datasets/geo.csv' # file name of the geospatial loc

geo = pd.read_csv(fn) # load data into a pandas datafram
geo.head()
```

	country/region\tlatitude\tlongitude\tname
0	AD\t42.546245\t1.601554\tAndorra
1	AE\t23.424076\t53.847818\tUnited Arab Emirates
2	AF\t33.93911\t67.709953\tAfghanistan
3	AG\t17.060816\t-61.796428\tAntigua and Barbuda
4	AI\t18.220554\t-63.068615\tAnguilla

\t here is the separator, which was not specified in the read_csv() function

Now try reload the “geo.csv” file with proper options:

```
fn = '../Datasets/geo.csv' # file name of the geos
geo = pd.read_csv(fn,sep='\t')
geo.head()
```

	country/region	latitude	longitude	name
0	AD	42.546245	1.601554	Andorra
1	AE	23.424076	53.847818	United Arab Emirates
2	AF	33.939110	67.709953	Afghanistan
3	AG	17.060816	-61.796428	Antigua and Barbuda
4	AI	18.220554	-63.068615	Anguilla

Hooray!

Now let’s merge the two data frames

Step 1: make the column names the same
Step 2: merge the data frames

```
# merge
geo.rename(columns={'name':'Entity'},inplace=True)
dmerge = df_c.merge(geo, on="Entity", how = 'inner')

dmerge.head()
```

Recall the covid-19 data frame df_c

	Entity	Code	Date	TotalCases	DailyCases
82	Afghanistan	AFG	Apr 1, 2020	166	25
106	Albania	ALB	Apr 1, 2020	243	20
194	Algeria	DZA	Apr 1, 2020	584	73
213	Andorra	AND	Apr 1, 2020	376	6
224	Angola	AGO	Apr 1, 2020	7	0

	Entity	Code	Date	TotalCases	DailyCases	DT	country/region	latitude	longitude
0	Afghanistan	AFG	Apr 1, 2020	166	25	2020-04-01	AF	33.939110	67.709953
1	Albania	ALB	Apr 1, 2020	243	20	2020-04-01	AL	41.153332	20.168331
2	Algeria	DZA	Apr 1, 2020	584	73	2020-04-01	DZ	28.033886	1.659626

Merged dataframe

Map the cases of Covid-19 using a bubble plot

Map scatter plots on map

Set index_column using “country/region”

```
dmerge['country/region']=dmerge['Entity']
dmerge.set_index('country/region',inplace=True)
dmerge.head()
#dfinal[dfinal.Entity=='China'].TotalCases
```

country/region	Entity	Code	Date	TotalCases	DailyCases	DT	latitude	longitude
Afghanistan	Afghanistan	AFG	Apr 1, 2020	166	25	2020-04-01	33.939110	67.709953
Albania	Albania	ALB	Apr 1, 2020	243	20	2020-04-01	41.153332	20.168331
Algeria	Algeria	DZA	Apr 1, 2020	584	73	2020-04-01	28.033886	1.659626
Andorra	Andorra	AND	Apr 1, 2020	376	6	2020-04-01	42.546245	1.601554
Angola	Angola	AGO	Apr 1, 2020	7	0	2020-04-01	-11.202692	17.873887

Now we can access the latitude, longitude using country names through pandas element access:

```
C = 'Afghanistan' # string variable for country
lat = dmerge['latitude'].loc[C]  # access the latitude, longitude
lon = dmerge['longitude'].loc[C]
print(lat,lon)
```

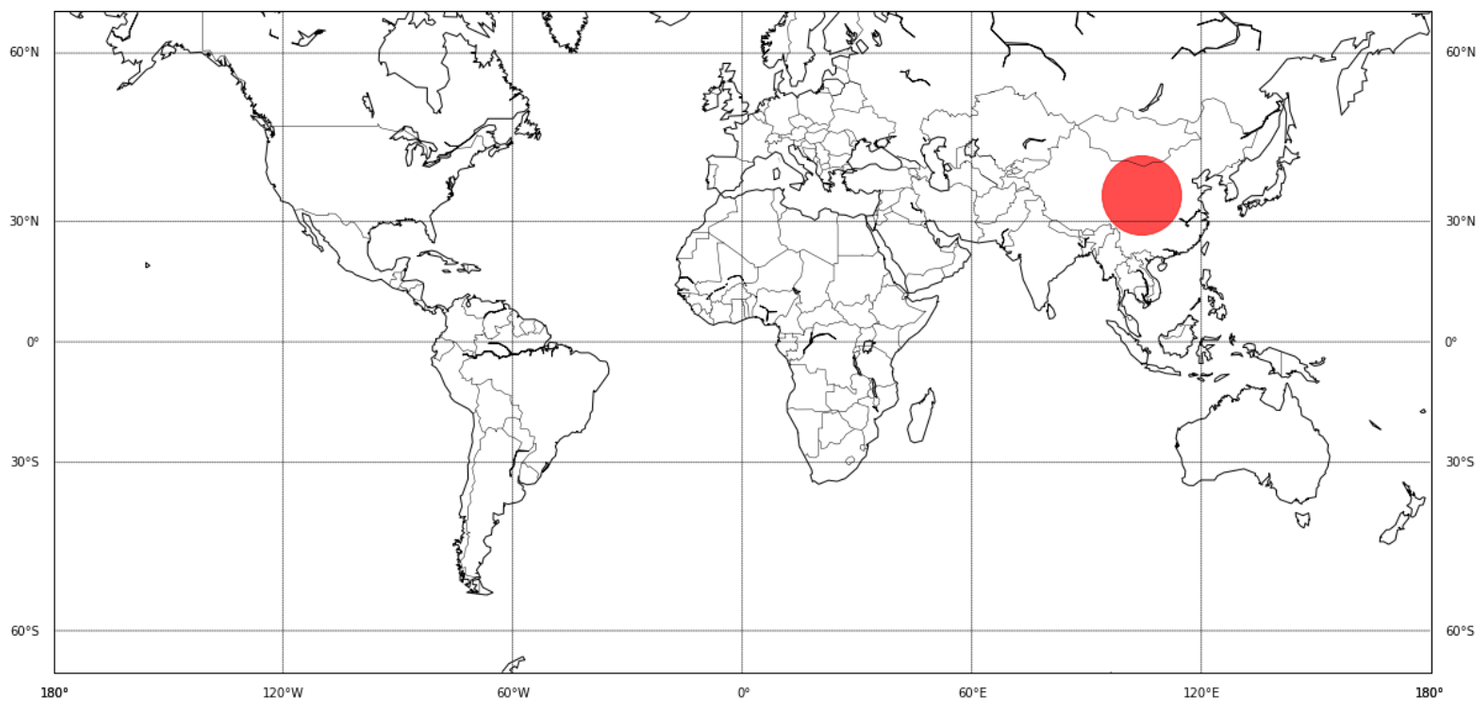
```
33.93911 67.709953
```

Let’s map the Lon, lat to the map coordinate and generate a bubble plot

Map the cases of Covid-19 using a bubble plot

Map scatter plots on map

```
# make a map instance called 'm', pay attention to the boundaries of the map
m = Basemap(projection='merc',llcrnrlat=-65,urcrnrlat=65,llcrnrlon=-180,urcrnrlon=180)
m.drawcoastlines(); # put on the coastlines

m.drawparallels(np.arange(-90.,91.,30.), labels=[True,True,False,False]) # plot the latitudes (parallels)
m.drawmeridians(np.arange(-180.,181.,60.),labels=[False,False,False,True]); # same for longitudes (meridians)

m.fillcontinents(color='none')
m.drawmapboundary(fill_color='none');
m.drawcountries(color='black',linewidth=0.5)

C = 'China' # string variable for country
lat = dmerge['latitude'].loc[C]  # access the latitude, longitude
lon = dmerge['longitude'].loc[C]
x,y = m(lon,lat) # map the location of China into the map coordinate m()

# now let's get the total number of cases on 2020-3-30
Case = dmerge['TotalCases'].loc[C] # total case of China on Mar 30, 2020

plt.scatter(x,y,Case/20,c='r',alpha=0.7)
plt.show()
```



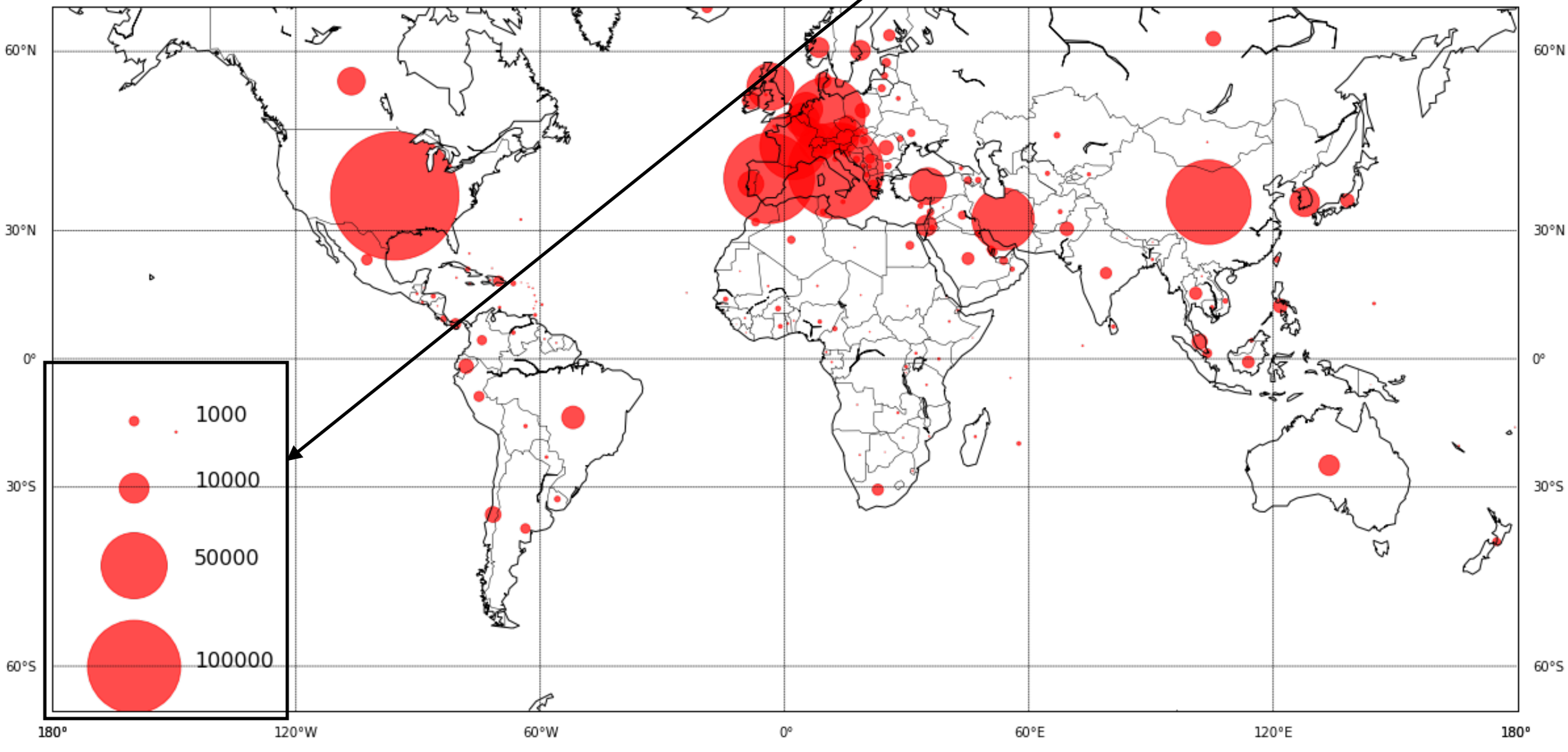
Step 1: create a world map (mercator)
Step 2: find the lat-lon of “China”
Step 3: map lat-lon to map coordinates
Step 4: bubble plot using plt.scatter( )

Map the more bubbles in a world map

Map scatter plots on map

```
lat = np.array(dmerge['latitude'])  # access the latitude, longitude
lon = np.array(dmerge['longitude'])
x,y = m(lon,lat) # map the location of China into the map coordinate m()

# now let's get the total number of cases on 2020-3-30
Case = np.array(dmerge['TotalCases']) # total case of China on Mar 30, 2020

plt.scatter(x,y,Case/20,c='r',alpha=0.7)
```

```
# now put on legend manually - can you do it automatically?
D = 'Mar 30, 2020'
lat=[-15,-30,-45,-60]
cas=[1000,10000,50000,100000]
for l,c in zip(lat,cas):
    x,y=m(-160,l)
    plt.scatter(x,y,s=c/20,c='r',alpha = 0.7)
    x,y=m(-145,l)
    plt.text(x,y,c,fontsize=16)
```



Map the cases with Geopandas

GeoPandas 0.7.0

GeoPandas is an open source project to make working with geospatial data in python easier. GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types. Geometric operations are performed by shapely. Geopandas further depends on fiona for file access and descartes and matplotlib for plotting.

Description

The goal of GeoPandas is to make working with geospatial data in python easier. It combines the capabilities of pandas and shapely, providing geospatial operations in pandas and a high-level interface to multiple geometries to shapely. GeoPandas enables you to easily do operations in python that would otherwise require a spatial database such as PostGIS.

Getting Started

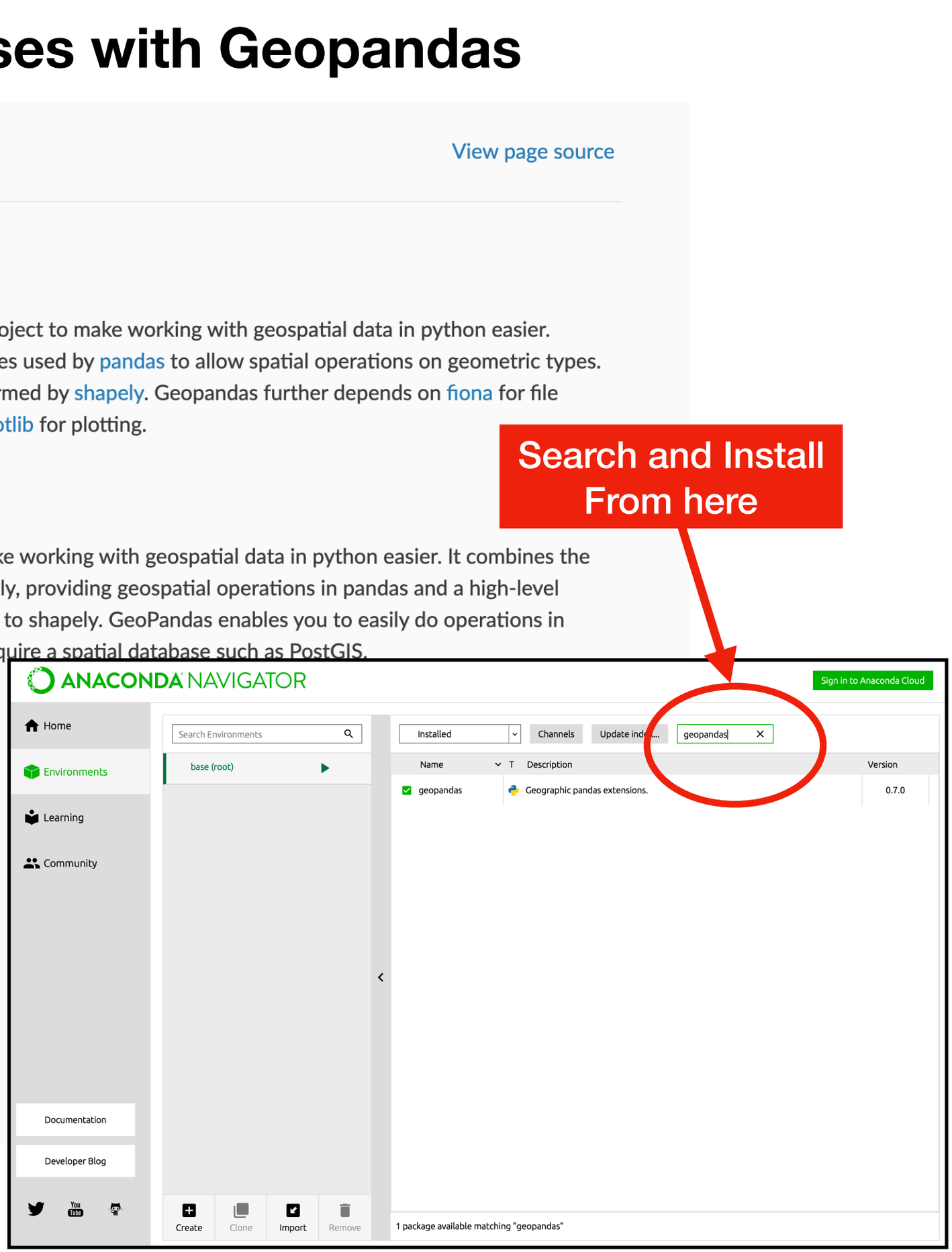
- Installation
- Examples Gallery

User Guide

- Data Structures
- Reading and Writing Files
- Indexing and Selecting Data
- Making Maps
- Managing Projections
- Geometric Manipulations
- Set Operations with overlay

http://geopandas.org

Search and Install From here



GeoDataFrames: the GeoPandas module

in short, "GeoPandas is a perfect marriage between geometric shape objects and dataframes".

GeoPandas is an open source project to make working with geospatial data in python easier. GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types. Geometric operations are performed by **shapely**. Geopandas further depends on **fiona** for file access and **descartes** and **matplotlib** for plotting.

The goal of GeoPandas is to make working with geospatial data in python easier. It combines the capabilities of **pandas** and **shapely**, providing geospatial operations in pandas and a high-level interface to multiple geometries to shapely. GeoPandas enables you to easily do operations in python that would otherwise require a spatial database such as PostGIS.

Here's the full documentation of geopandas: http://geopandas.org

```
import geopandas as gpd
#read country GeoDataFrame
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

# Examine country GeoDataFrame
# Note The special column "geometry"
world.head()
```

- **world is a GeoDataFrame**
- **world has a column “geometry”**
- **“geometry” contains shapes of each country**

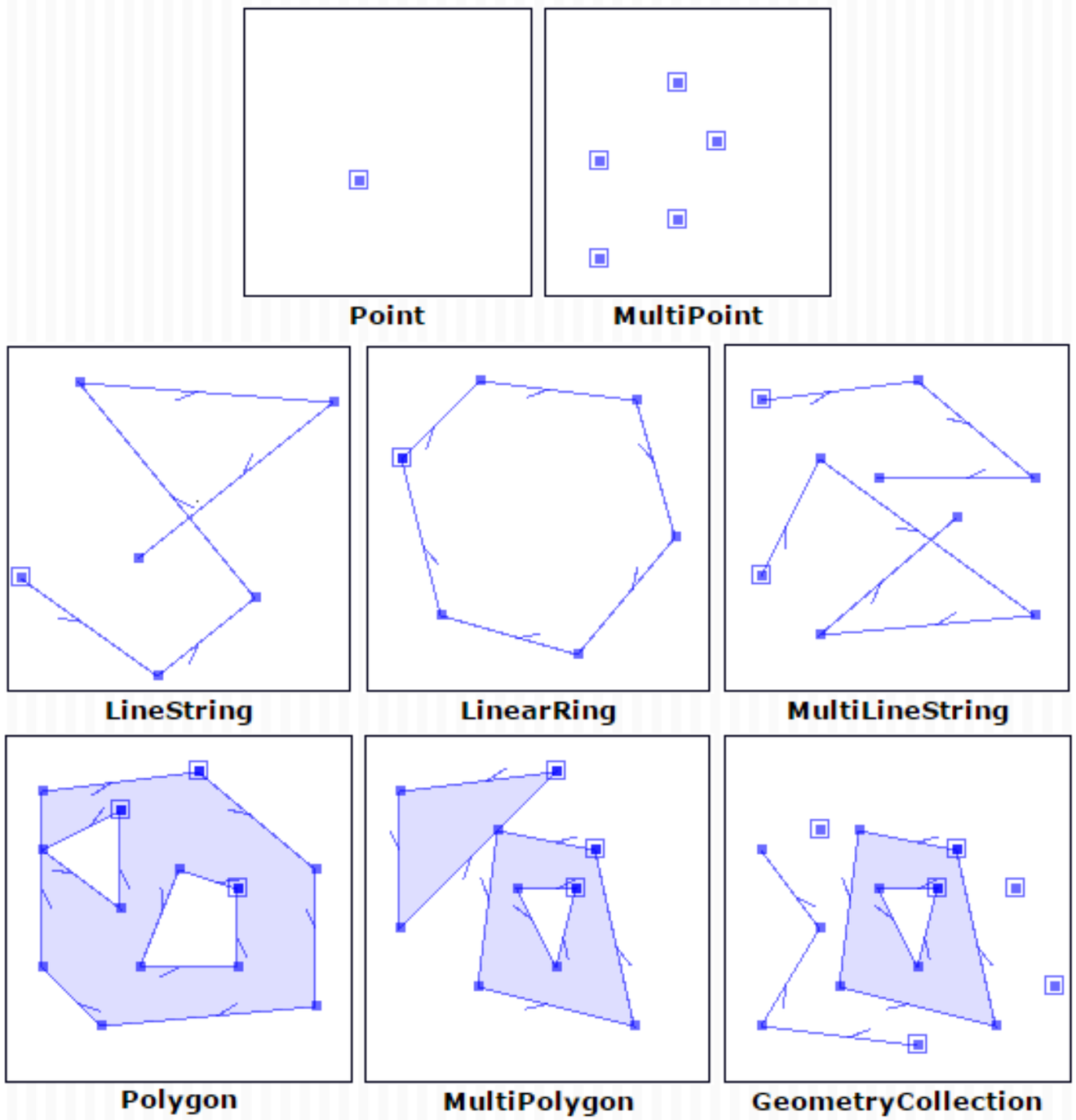
	pop_est	continent	name	iso_a3	gdp_md_est	geometry
0	920938	Oceania	Fiji	FJI	8374.0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...
1	53950935	Africa	Tanzania	TZA	150600.0	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...
2	603253	Africa	W. Sahara	ESH	906.5	POLYGON ((-8.66559 27.65643, -8.66512 27.58948...
3	35623680	North America	Canada	CAN	1674000.0	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...
4	326625791	North America	United States of America	USA	18560000.0	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...

An additional data column called “geometry” with **geometric objects** (shapes)

Advance topic

Geometric Objects: the Shapely module

Overview of geometric objects



The most fundamental geometric objects are **Points**, **Lines** and **Polygons** which are the basic ingredients when working with spatial data in vector format. Python has a specific module called Shapely that can be used to create and work with Geometric Objects.

Geometric Objects consist of coordinate tuples where:

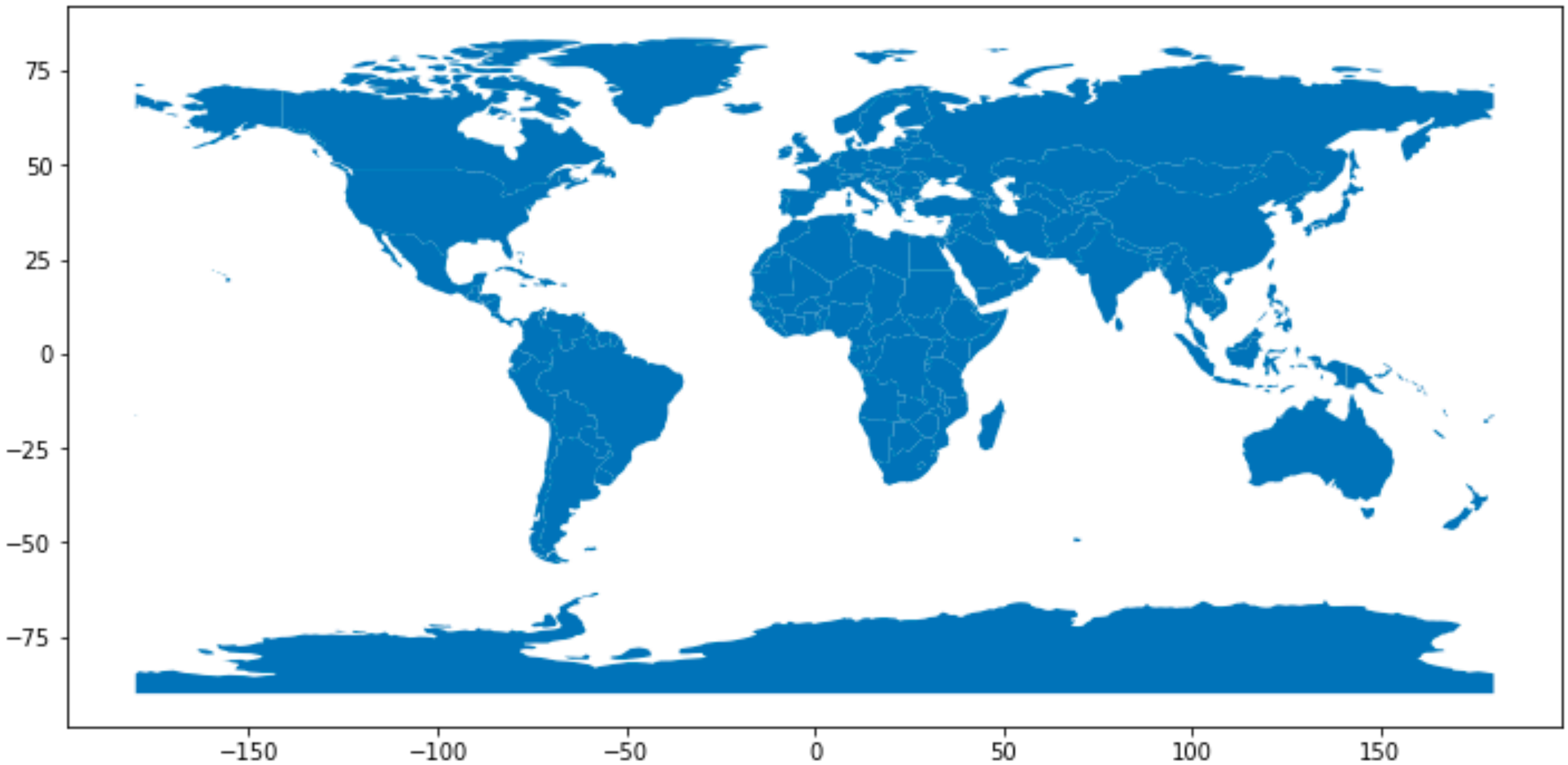
- **Point** -object represents a single point in space. Points can be either two-dimensional (x, y) or three dimensional (x, y, z).
- **LineString** -object (i.e. a line) represents a sequence of points joined together to form a line. Hence, a line consist of a list of at least two coordinate tuples
- **Polygon** -object represents a filled area that consists of a list of at least three coordinate tuples that forms the outerior ring and a (possible) list of hole polygons.

It is also possible to have a collection of geometric objects (e.g. Polygons with multiple parts):

- **MultiPoint** -object represents a collection of points and consists of a list of coordinate-tuples
- **MultiLineString** -object represents a collection of lines and consists of a list of line-like sequences
- **MultiPolygon** -object represents a collection of polygons that consists of a list of polygon-like sequences that construct from exterior ring and (possible) hole list tuples

GeoDataFrames: Visualize the geometry (shape) using .plot( )

```
1  world.plot(figsize=(12,8))
2  plt.show()
```



So the .plot( ) function for a geopandas data frame makes maps directly

GeoDataFrames: Data manipulations

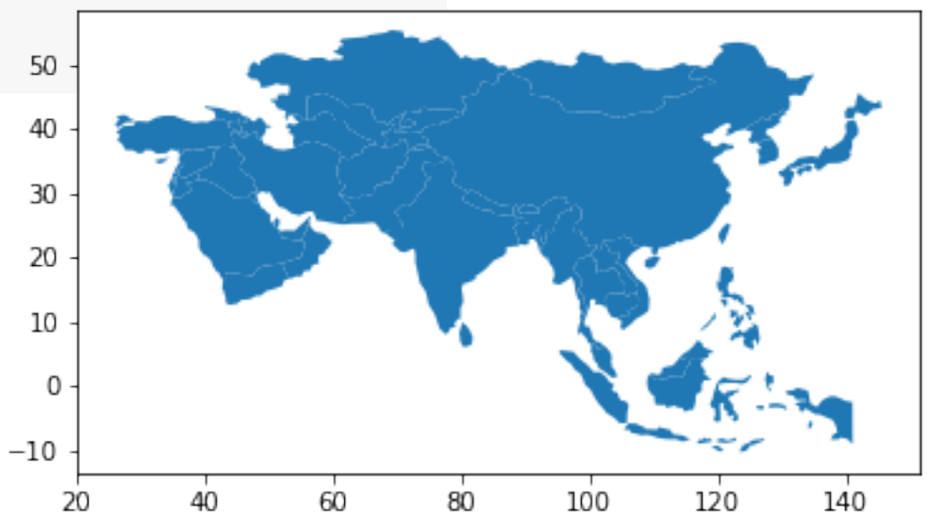
It's just like operating on a Pandas DataFrame for selection, create new variables etc.:

```
# Select only inhabited continents
world = world[(world.pop_est>0) & (world.name!="Antarctica")]

# Calculate GDP per capita for each country
world['gdp_per_cap'] = world.gdp_md_est / world.pop_est*1e6
world.head()
```

	pop_est	continent	name	iso_a3	gdp_md_est	geometry	gdp_per_cap
0	28400000.0	Asia	Afghanistan	AFG	22270.0	POLYGON ((61.21081709172574 35.65007233330923,...	784.154930
1	12799293.0	Africa	Angola	AGO	110300.0	(POLYGON ((16.32652835456705 -5.87747039146621...	8617.663491
2	3639453.0	Europe	Albania	ALB	21810.0	POLYGON ((20.59024743010491 41.85540416113361,...	5992.658787
3	4798491.0	Asia	United Arab Emirates	ARE	184300.0	POLYGON ((51.57951867046327 24.24549713795111,...	38407.907819
4	40913584.0	South America	Argentina	ARG	573900.0	(POLYGON ((-65.50000000000003 -55.199999999999...	14027.126052

For example, similar to the data filtering in Pandas, you can select a subset of the geopandas data frame for visualization. For example, we only want to show everythin in "Asia":

```
asia = world[world['continent']=='Asia'] # select all the countries in "Asia"
asia.plot() # visualize the new geometric object
plt.show()
```



Notes:

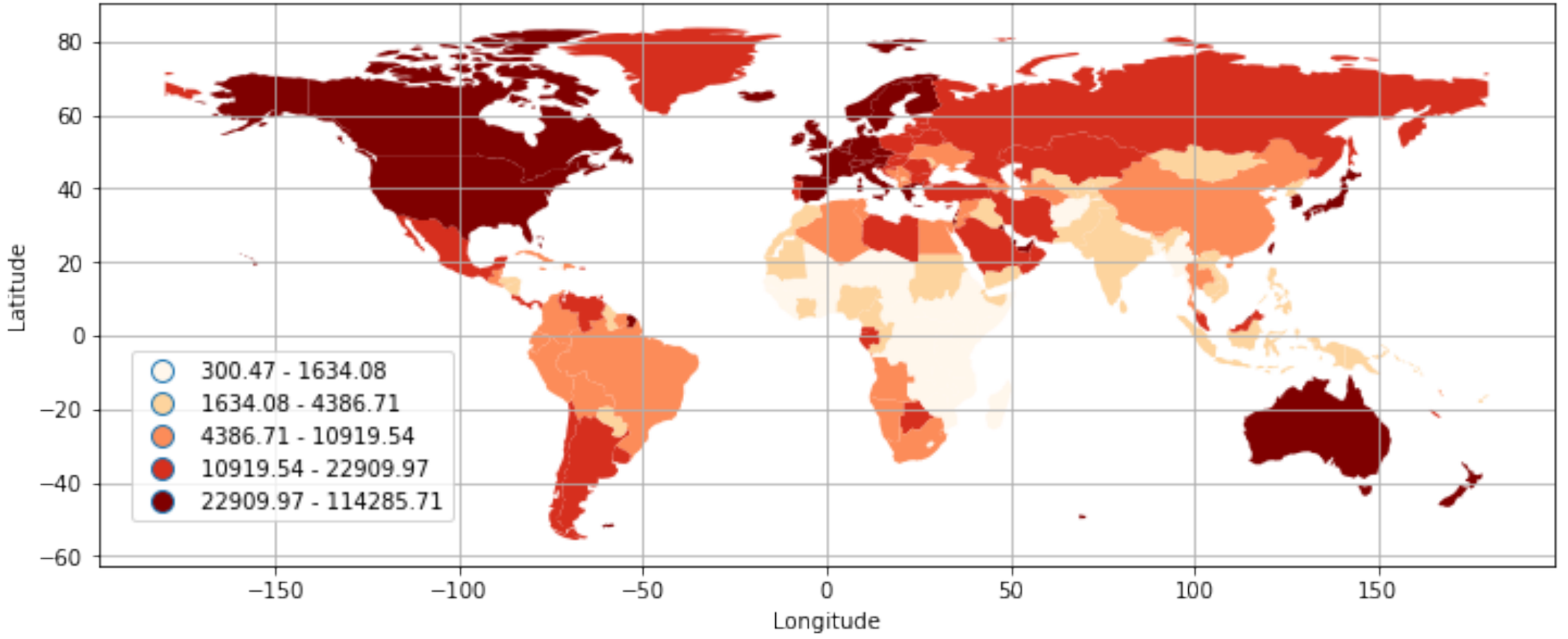
- Geodataframes are based on Pandas
- Can use all the data wrangling tricks in Pandas
- GeoDataFrames have an additional data column “geometry”

GeoDataFrames: Map data using Choropleth Maps

Maps where the color of each shape is based on the value of an associated variable:

```
# a simple map for gdp per capita
# column argument specifies the variable the color is based on
ax = world.plot(figsize=(12,6),column='gdp_per_cap',cmap='OrRd',scheme='quantiles',legend=True)
leg = ax.get_legend() # add a legend
leg.set_bbox_to_anchor((0., 0., 0.25, 0.4)) # specify the location of the legend (left corner)
ax.set_title('World GDP per Capita')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.grid(True) # add longitude-latitude grids
plt.show()
```

See more colormaps here:
https://matplotlib.org/tutorials/colors/colormaps.html



GeoDataFrames: Aggregation countries to continents

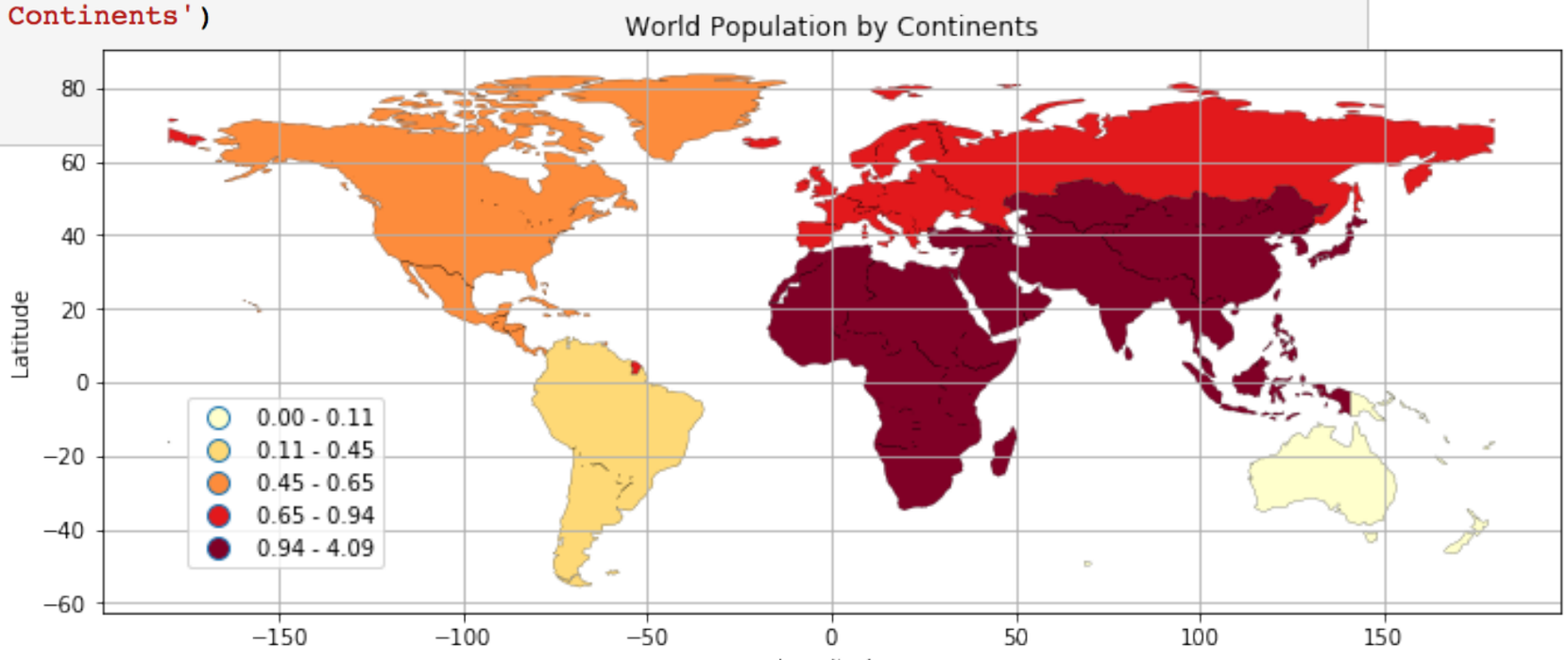
We can also dissolve the GeoDataFrame by continents, in this example:

```
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # load the geodata frame
world = world[(world.pop_est>0) & (world.name!="Antarctica")] # exclude Antartica

continents = world.dissolve(by='continent', aggfunc='sum') # aggregate countries to continents
continents['pop_est'] = continents['pop_est']/1e9 # convert population to billion

continents.head()
```

continent	geometry	pop_est	gdp_md_est
Africa	(POLYGON ((49.54351891459575 -12.4698328589405...	0.993282	2718368.20
Asia	(POLYGON ((120.7156087586305 -10.2395813940878...	4.085853	24826690.77
Europe	(POLYGON ((-52.55642473001839 2.50470530843705...	0.728131	18546159.00
North America	(POLYGON ((-61.68000000000001 10.76, -61.105 1...	0.539351	18537449.00
Oceania	(POLYGON ((173.0203747907408 -40.9190524228564...	0.033520	938913.50

```
ax = continents.plot(figsize=(12,6),edgecolor='k',linewidth=0.2,column = 'pop_est', scheme='quantiles',
                     cmap='YlOrRd', legend=True)
leg = ax.get_legend()
leg.set_bbox_to_anchor((0., 0., 0.2, 0.4))
ax.set_title('World Population by Continents')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.grid(True)
```

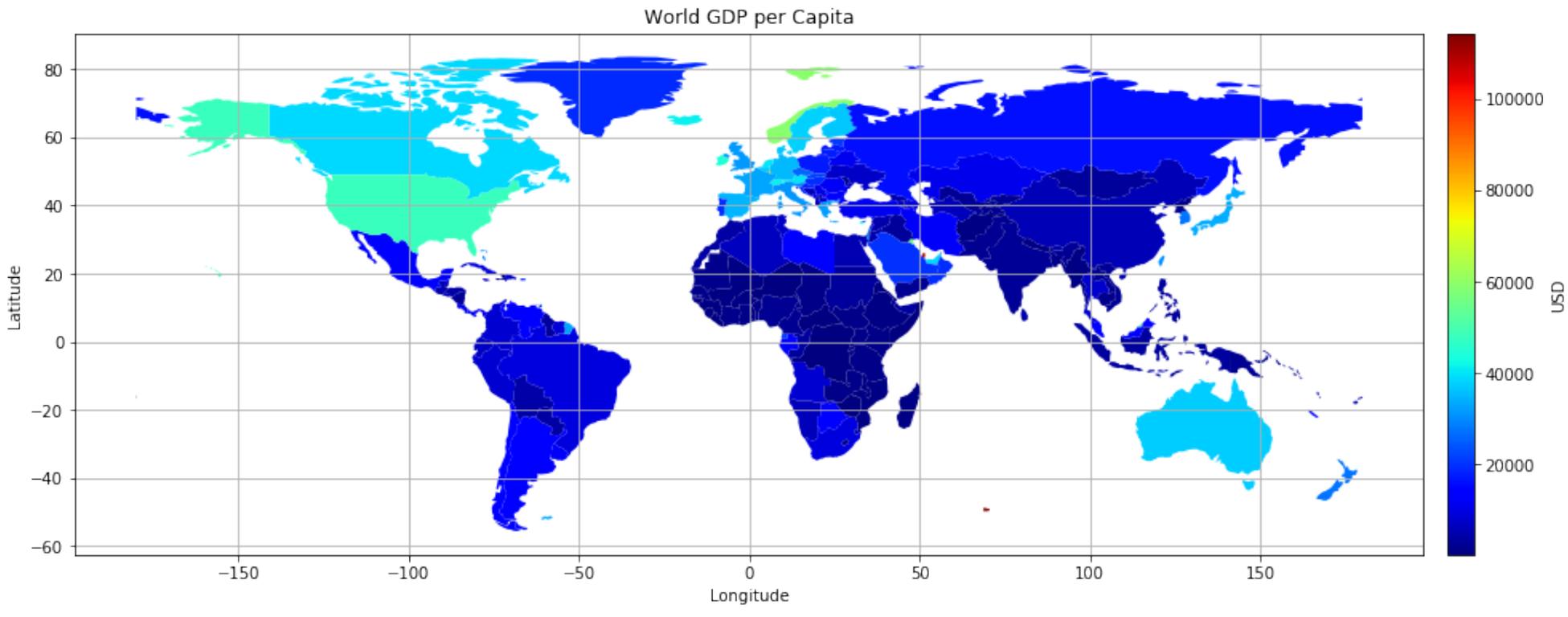
tada!



Advance topic

GeoDataFrames: Choropleth Maps with a colorbar

```
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib.colors import Normalize
from matplotlib import cm

world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres')) # load the geodata frame
world = world[(world.pop_est>0) & (world.name!="Antarctica")] # exclude Antartica

world['gdp_per_cap'] = world.gdp_md_est / world.pop_est*1e6

# a map for gdp per capita with a colorbar
# column argument specifies the variable the color is based on
ax = world.plot(figsize=(18,6),column='gdp_per_cap',cmap='jet',legend=False)
ax.set_title('World GDP per Capita')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.grid(True)

norm = Normalize(vmin = world['gdp_per_cap'].min(),vmax = world['gdp_per_cap'].max())
n_cmap = cm.ScalarMappable(norm = norm, cmap = 'jet')
n_cmap.set_array([])

divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="2%", pad=0.2)
bar = ax.get_figure().colorbar(n_cmap,ax=ax,cax=cax)
bar.set_label('USD');
```



Advance topic

GeoDataFrames: Maps with a multiple layers

Put Countries and Cities in the same map using the same coordinate reference system and projection¶

- One important attribute for geometries.
- Especially important when combining spatial data from different sources.

```
cities = gpd.read_file(gpd.datasets.get_path('naturalearth_cities')) # load geospatial data of capitals
cities.head()
```

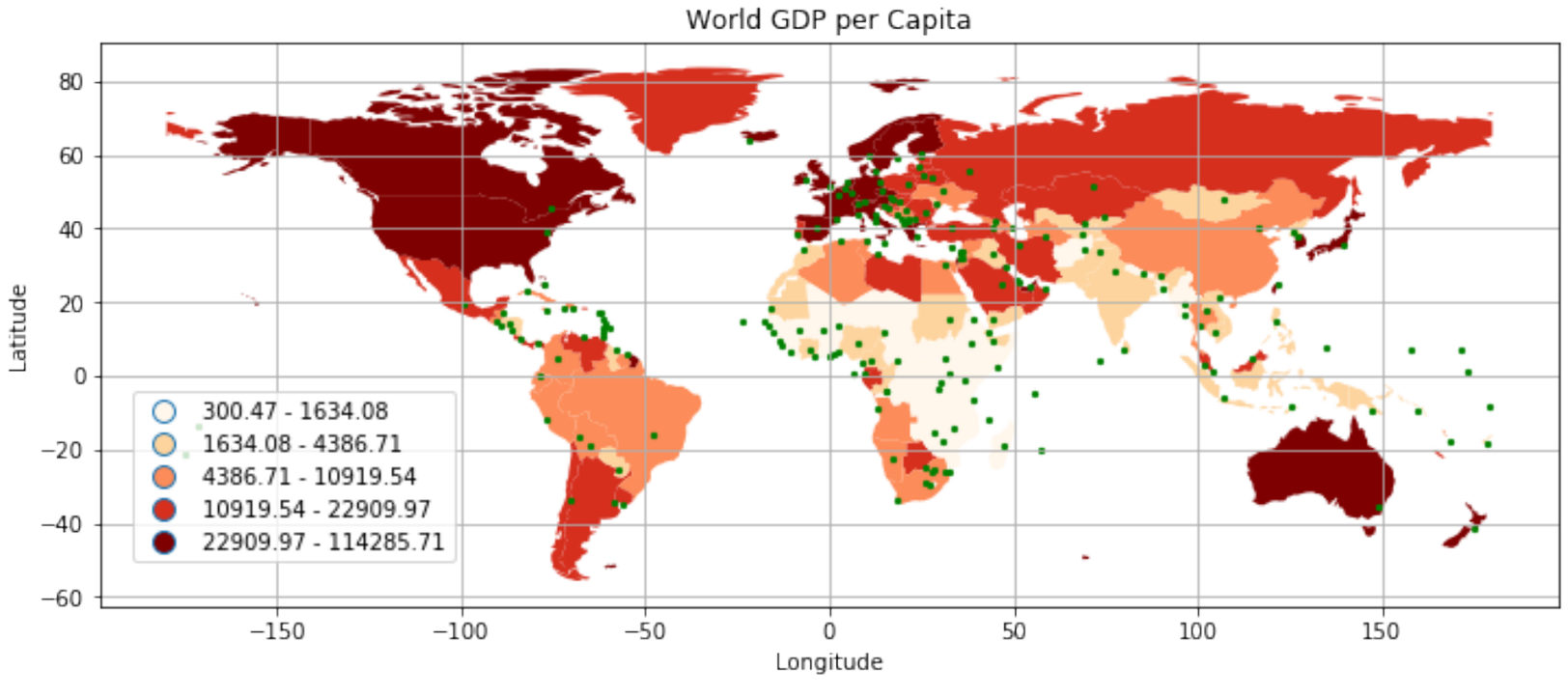
	name	geometry
0	Vatican City	POINT (12.45338654497177 41.90328217996012)
1	San Marino	POINT (12.44177015780014 43.936095834768)

"point" shape

```
ax = world.plot(figsize=(12,6),column='gdp_per_cap',cmap='OrRd',scheme='quantiles',legend=True)

leg = ax.get_legend()
leg.set_bbox_to_anchor((0., 0., 0.25, 0.4))
ax.set_title('World GDP per Capita')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.grid(True)

#base = world.plot(figsize=(12,6),color='white', edgecolor='black')
cities.plot(ax=ax, marker='o', color='g', markersize=5)
```

Layer 1

Layer 2



Map the situation of Covid-19 cases in a world map

Recall that we load the data file and filter by date:

```
1  fn = '../Datasets/total-and-daily-cases-covid-19.csv' # file name
2  df = pd.read_csv(fn) # load file into data frame
3  df.head() # show the first 10 rows
4
5  # first change the column names slightly
6  df.rename(columns={'Total confirmed cases (cases)':'TotalCases',\
7                     'Daily new confirmed cases (cases)':'DailyCases'},\
8                      inplace=True)
9
10 df_c = df[df.Date=='Apr 1, 2020']
11 df_c.head()
```

df_c

	Entity	Code	Date	TotalCases	DailyCases
82	Afghanistan	AFG	Apr 1, 2020	166	25
106	Albania	ALB	Apr 1, 2020	243	20
194	Algeria	DZA	Apr 1, 2020	584	73
213	Andorra	AND	Apr 1, 2020	376	6
224	Angola	AGO	Apr 1, 2020	7	0

The geopandas data frame “world”:

	pop_est	continent	name	iso_a3	gdp_md_est	geometry	gdp_per_cap
0	920938	Oceania	Fiji	FJI	8374.0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...	9092.903105
1	53950935	Africa	Tanzania	TZA	150600.0	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...	2791.425209
2	603253	Africa	W. Sahara	ESH	906.5	POLYGON ((-8.66559 27.65643, -8.66512 27.58948...	1502.686269
3	35623680	North America	Canada	CAN	1674000.0	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...	46991.214832
4	326625791	North America	United States of America	USA	18560000.0	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...	56823.436824

Thoughts:

- The geospatial map is generated using the “world” data frame (geopandas)
- The Covid-19 case information is in the “df_c” data frame (pandas)
- They have a common column with the country codes (merge)
- After merging the data frame “df_c” into “world”, use the .plot( ) function

Map the situation of Covid-19 cases in a world map

Change column name to “Code”

```
# merge the df_c dataframe into the world data frame
world.rename(columns={'iso_a3':'Code'},inplace=True)
covid = world.merge(df_c, on="Code", how = 'inner')

covid.head()
```

	pop_est	continent	name	iso_a3
0	920938	Oceania	Fiji	FJI
1	53950935	Africa	Tanzania	TZA
2	603253	Africa	W. Sahara	ESH
3	35623680	North America	Canada	CAN
4	326625791	North America	United States of America	USA

New data frame covid:

	pop_est	continent	name	Code	gdp_md_est	geometry	gdp_per_cap
0	920938	Oceania	Fiji	FJI	8374.0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...	9092.903105
1	53950935	Africa	Tanzania	TZA	150600.0	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...	2791.425209
2	35623680	North America	Canada	CAN	1674000.0	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...	46991.214832
3	326625791	North America	United States of America	USA	18560000.0	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...	56823.436824
4	18556698	Asia	Kazakhstan	KAZ	460700.0	POLYGON ((87.35997 49.21498, 86.59878 48.54918...	24826.615166

The “world” geopandas data frame

	Entity	Date	TotalCases	DailyCases
0	Fiji	Apr 1, 2020	5	0
1	Tanzania	Apr 1, 2020	19	0
2	Canada	Apr 1, 2020	8536	1112
3	United States	Apr 1, 2020	189618	24998
4	Kazakhstan	Apr 1, 2020	340	15

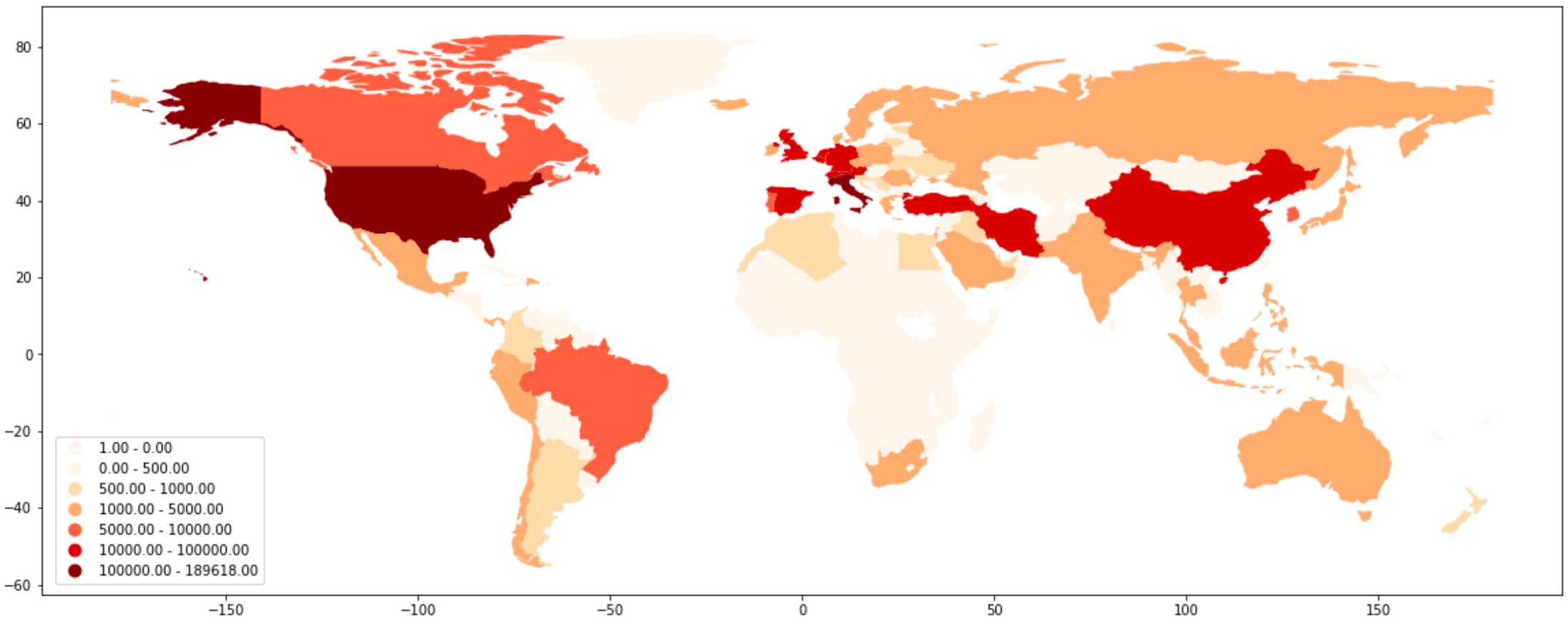
The “df_c” pandas data frame

Results:

- **After merge, the new geopandas data frame “covid” is created**
- **The “covid” geopandas data frame now has all the information for creating maps**
- **Now use the .plot( ) function to generate Choropleth Maps**

Map the situation of Covid-19 cases in a world map

```
# map the TotalCases into a word map, using an orange-red colormap
ax = covid.plot(figsize=(20,16),column='TotalCases',cmap='OrRd',scheme='userdefined',\
                classification_kwds=dict(bins=[0,500,1000,5000,10000,100000]),legend=True)

leg = ax.get_legend() # add a legend
leg.set_bbox_to_anchor((0., 0., 0.15, 0.25)) # specify the location of the legend (left corner)
```



	pop_est	continent	name	Code	gdp_md_est	geometry	gdp_per_cap	Entity	Date	TotalCases	DailyCases
0	920938	Oceania	Fiji	FJI	8374.0	MULTIPOLYGON (((180.00000 -16.06713, 180.00000...	9092.903105	Fiji	Apr 1, 2020	5	0
1	53950935	Africa	Tanzania	TZA	150600.0	POLYGON ((33.90371 -0.95000, 34.07262 -1.05982...	2791.425209	Tanzania	Apr 1, 2020	19	0
2	35623680	North America	Canada	CAN	1674000.0	MULTIPOLYGON (((-122.84000 49.00000, -122.9742...	46991.214832	Canada	Apr 1, 2020	8536	1112
3	326625791	North America	United States of America	USA	18560000.0	MULTIPOLYGON (((-122.84000 49.00000, -120.0000...	56823.436824	United States	Apr 1, 2020	189618	24998
4	18556698	Asia	Kazakhstan	KAZ	460700.0	POLYGON ((87.35997 49.21498, 86.59878 48.54918...	24826.615166	Kazakhstan	Apr 1, 2020	340	15

Now let's make some maps!