Taxonomy of Data Visualization Methods

Data visualization methods

The common definition for taxonomy comes from the biological sciences and refers to the organization into groups of members that **share similar characteristics**.

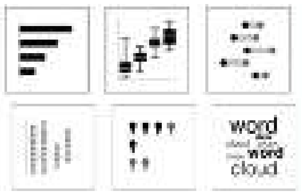
Andy Kirk . (2012). Data Visualization: a successful design process

Taxonomy in Visualization Methods

Method Classification

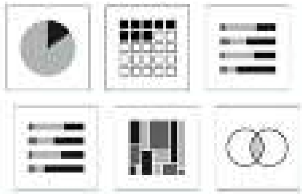
categorical

To facilitate comparisons between the relative and absolute sizes of categorical values.
Ex. bar charts, dot plots



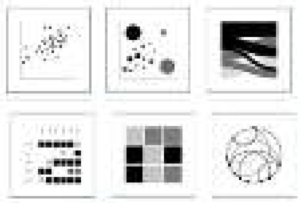
hierarchical

To provide a breakdown of categorical values in their relationship to a population of values or as constituent elements of hierarchical structures.
Ex. donut charts



relational

To assess the associations, distributions, & patterns that exists between multivariate datasets.
Ex. scatterplots, sankey diagrams

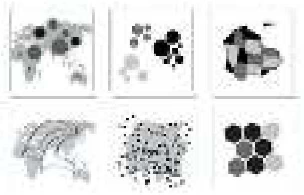


temporal

To exploit temporal data and show the changing trends and patterns of values over a continuous timeframe. Ex. line charts, stream graphs

spatial

To plot and present datasets with geo-spatial properties via the many different mapping frameworks. Ex. chloropleths, cartograms



Andy Kirk . (2012). Data Visualization: a successful design process

The Chartmaker Directory

- JS
- R
- Python (matplotlib, seabornlib)
- Excel
- Tableau
- Etc.

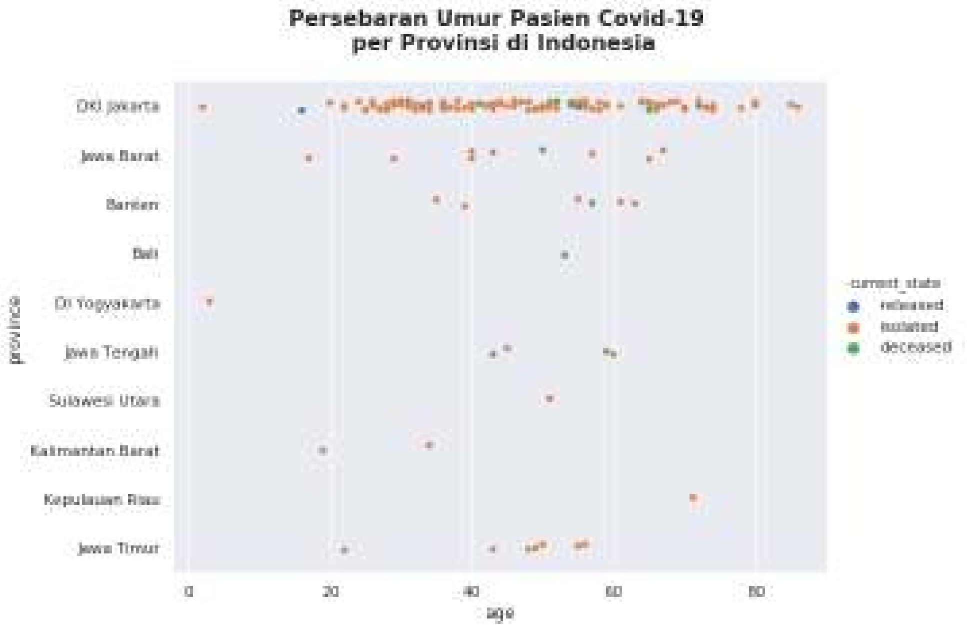
Example and Implementation

WITH PYTHON

METHOD CLASSIFICATION

1. CATEGORICAL - COMPARING CATEGORIES

Categorical – Dot Plot



Data variables :

2 x categorical (province, current_state),

1 x quantitative (age)

Visual Variables:

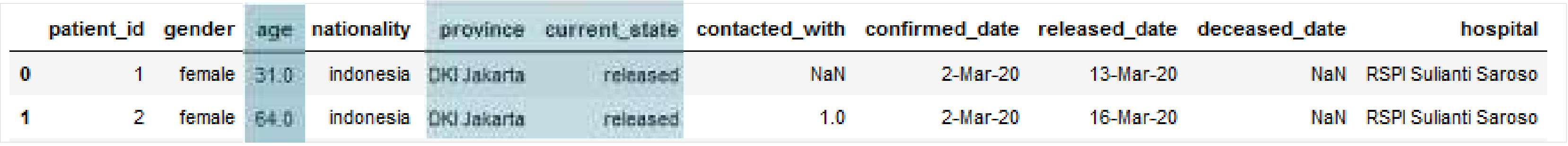
Position, color-hue

Data :

Indonesia Coronavirus Cases

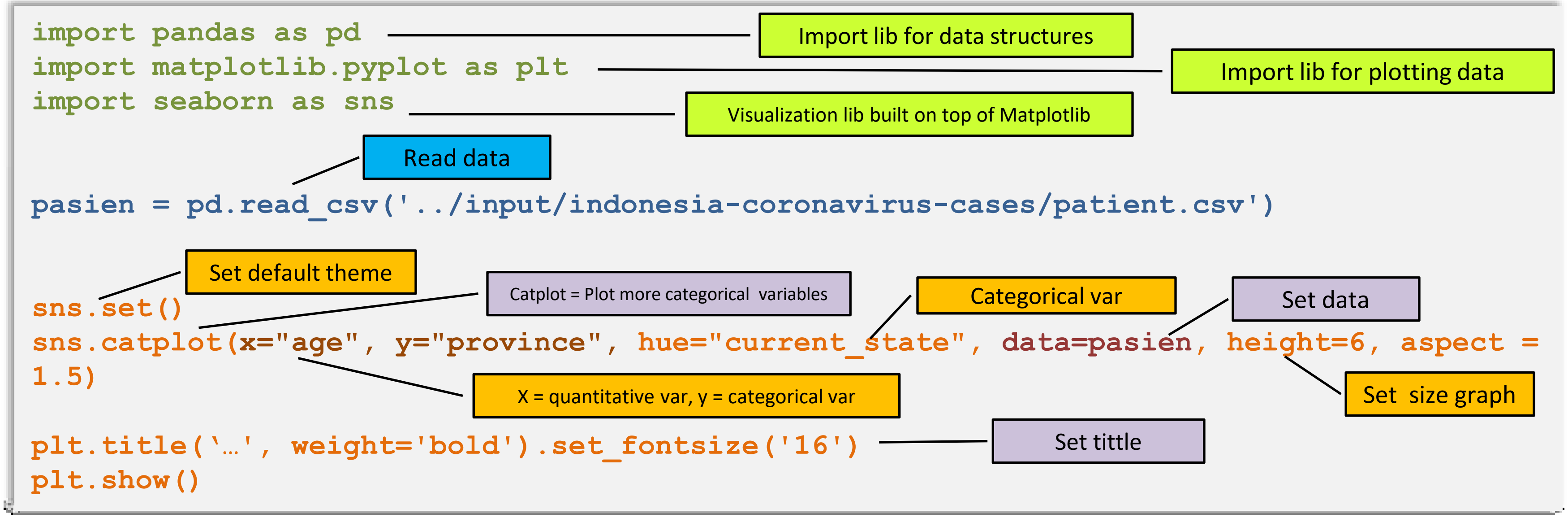
Source :

https://www.kaggle.com/ardisragen/indonesia-coronavirus-cases/



	patient_id	gender	age	nationality	province	current_state	contacted_with	confirmed_date	released_date	deceased_date	hospital
0	1	female	31.0	indonesia	DKI Jakarta	released	NaN	2-Mar-20	13-Mar-20	NaN	RSPI Sulianti Saroso
1	2	female	64.0	indonesia	DKI Jakarta	released	1.0	2-Mar-20	16-Mar-20	NaN	RSPI Sulianti Saroso

Source Code for Categorical – Dot Plot

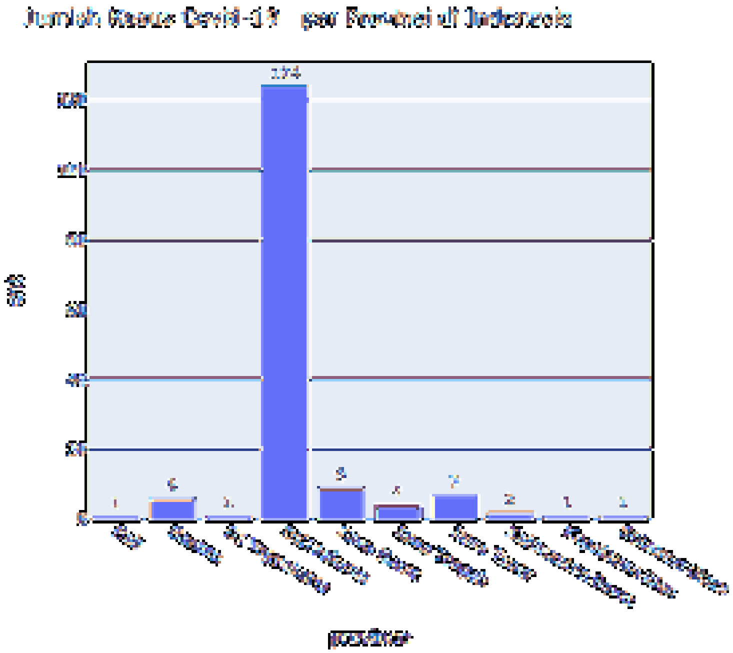


```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

pasien = pd.read_csv('../input/indonesia-coronavirus-cases/patient.csv')

sns.set()
sns.catplot(x="age", y="province", hue="current_state", data=pasien, height=6, aspect =
1.5)

plt.title(‘…', weight='bold').set_fontsize('16')
plt.show()
```

Import lib for data structures
Import lib for plotting data
Visualization lib built on top of Matplotlib
Read data
Set default theme
Catplot = Plot more categorical variables
Categorical var
Set data
X = quantitative var, y = categorical var
Set size graph
Set tittle

Note:
Seaborn is a *Python* data visualization library based on matplotlib.. Unlike when using matplotlib directly, seaborn lib isn’t necessary to translate the variables into parameters of the visualization (e.g., the specific color or marker to use for each category) (https://seaborn.pydata.org)

Categorical – Bar Chart



Data variables :

1 x categorical (province),

1 x quantitative (count)

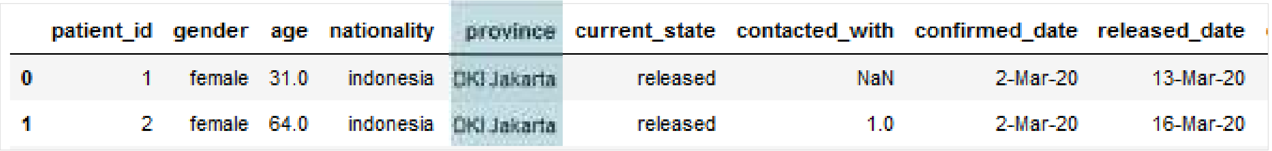
Visual Variables:

Position, color-hue

Data :

Indonesia Coronavirus Cases

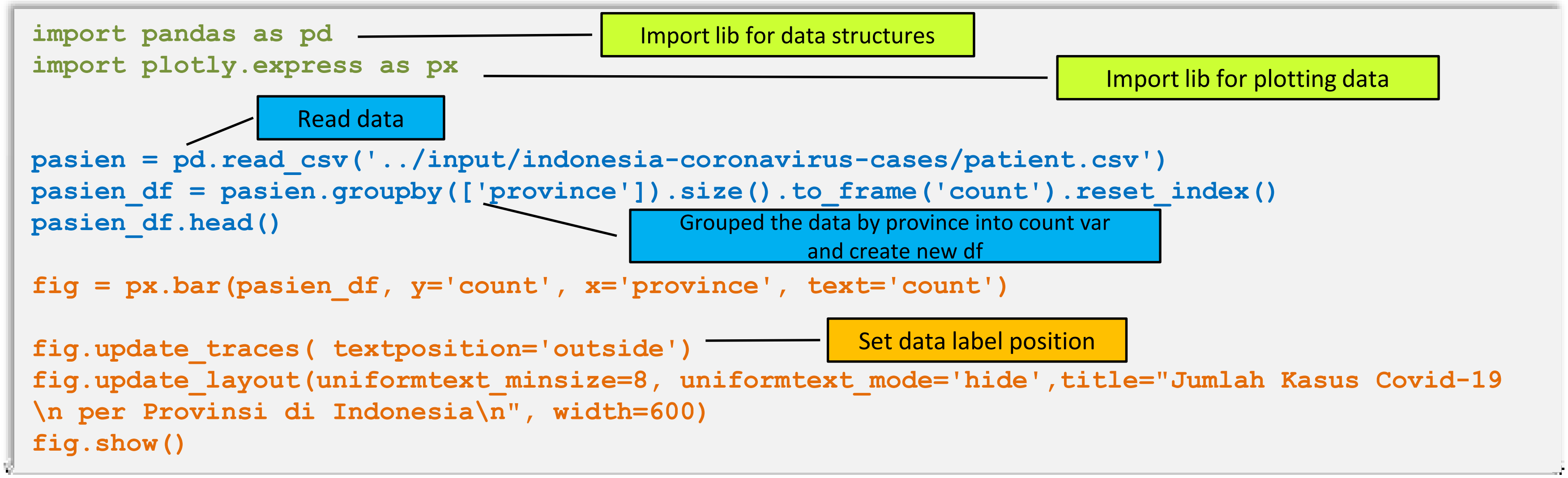
Source :
https://www.kaggle.com/ardisragen/indonesia-coronavirus-cases/



	patient_id	gender	age	nationality	province	current_state	contacted_with	confirmed_date	released_date
0	1	female	31.0	indonesia	DKI Jakarta	released	NaN	2-Mar-20	13-Mar-20
1	2	female	64.0	indonesia	DKI Jakarta	released	1.0	2-Mar-20	16-Mar-20

Count =
Group by province

Source Code for Categorical – Bar Chart



```
import pandas as pd
import plotly.express as px

pasien = pd.read_csv('../input/indonesia-coronavirus-cases/patient.csv')
pasien_df = pasien.groupby(['province']).size().to_frame('count').reset_index()
pasien_df.head()

fig = px.bar(pasien_df, y='count', x='province', text='count')

fig.update_traces( textposition='outside')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide',title="Jumlah Kasus Covid-19
\n per Provinsi di Indonesia\n", width=600)
fig.show()
```

Import lib for data structures

Import lib for plotting data

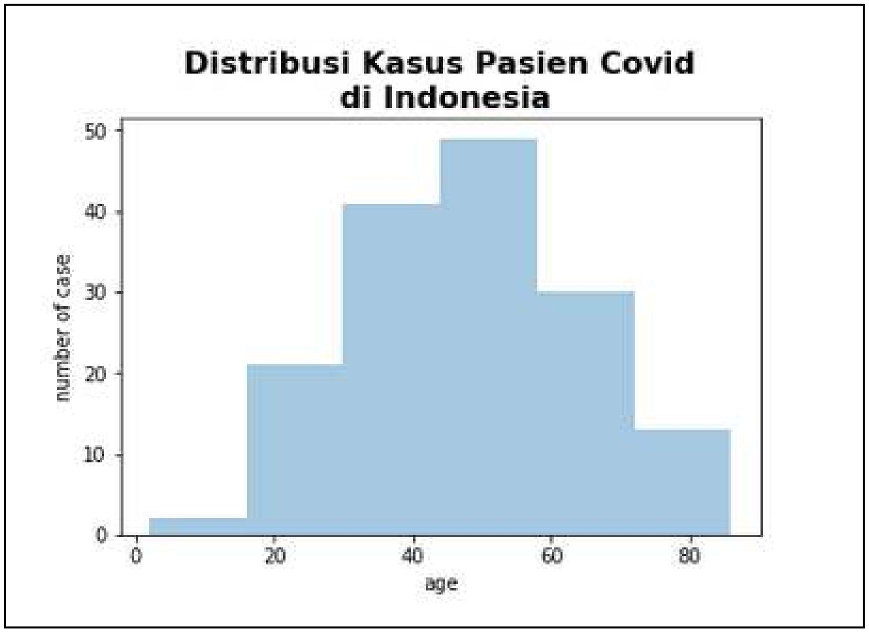
Read data

Grouped the data by province into count var and create new df

Set data label position

Note:
There are several ways to make bar chart, such as matplotlib bar function and seaborn barplot function.
Ex.
plt.bar(y_var, x_var, align='center', alpha=0.5)
sns.barplot(x=x_var', y=y_var', data=data2.head(10))

Categorical – Histogram



Data variables :

1 x quantitative-interval (age interval),

1 x quantitative ratio (number of case)

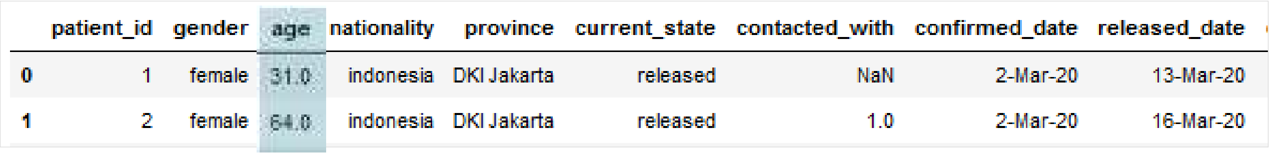
Visual Variables:

Height, Width

Data :

Indonesia Coronavirus Cases

Source :
https://www.kaggle.com/ardisragen/indonesia-coronavirus-cases/

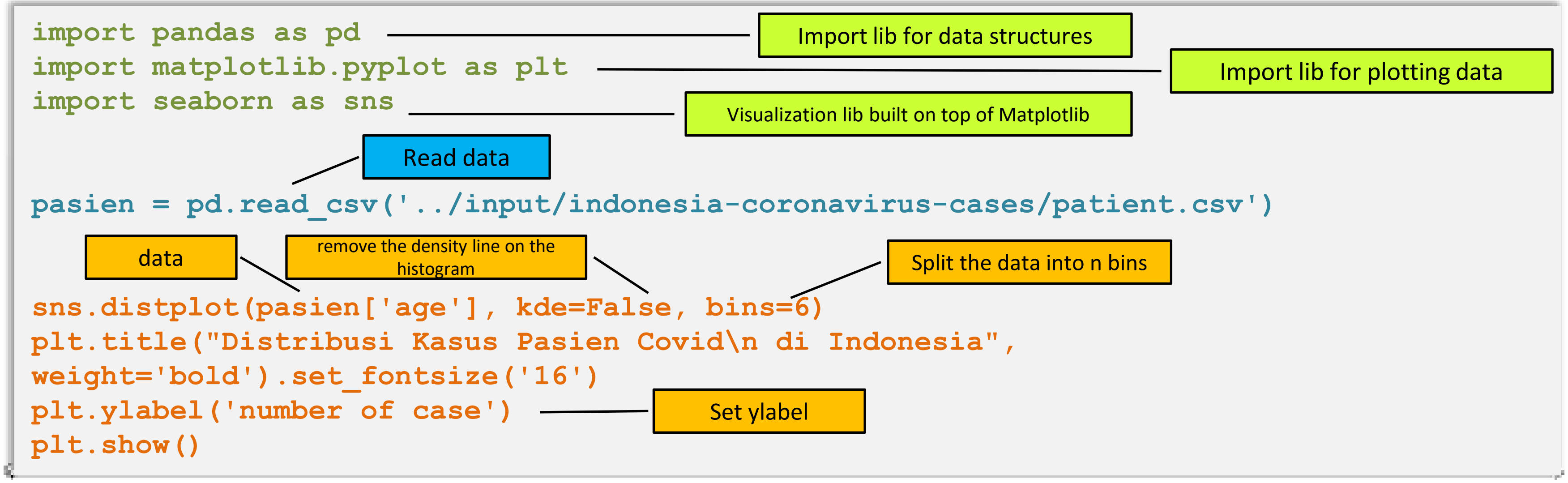


	patient_id	gender	age	nationality	province	current_state	contacted_with	confirmed_date	released_date
0	1	female	31.0	indonesia	DKI Jakarta	released	NaN	2-Mar-20	13-Mar-20
1	2	female	64.0	indonesia	DKI Jakarta	released	1.0	2-Mar-20	16-Mar-20

Age Interval =
(max-min)/bins

Number of case =
Group by Interval Age

Source Code for Categorical – Histogram



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

pasien = pd.read_csv('../input/indonesia-coronavirus-cases/patient.csv')

sns.distplot(pasien['age'], kde=False, bins=6)
plt.title("Distribusi Kasus Pasien Covid\n di Indonesia",
weight='bold').set_fontsize('16')
plt.ylabel('number of case')
plt.show()
```

Import lib for data structures
Import lib for plotting data
Visualization lib built on top of Matplotlib
Read data
data
remove the density line on the histogram
Split the data into n bins
Set ylabel

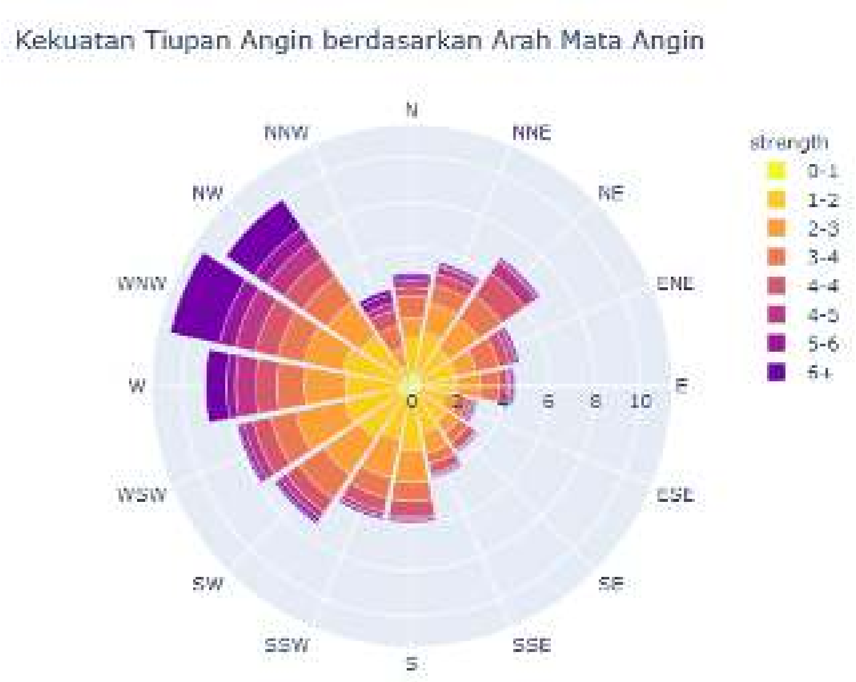
Note:

There are several ways to make histogram. The standard count-based histogram can be created with the *matplotlib* hist function

Ex.

plt.hist(x, bins=30, normed=True)

Categorical – Radial Chart



Data variables :

1 x categorical (direction),

2 x categorical ordinal (strength),

1 x quantitative (frequency)

Visual Variables:

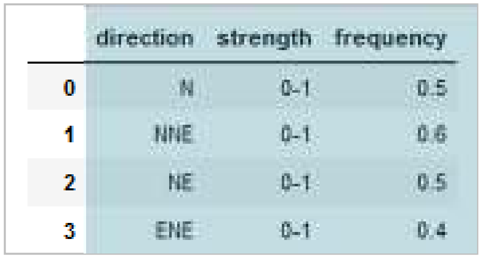
Position, color-saturation

Data :

Wind Rose.

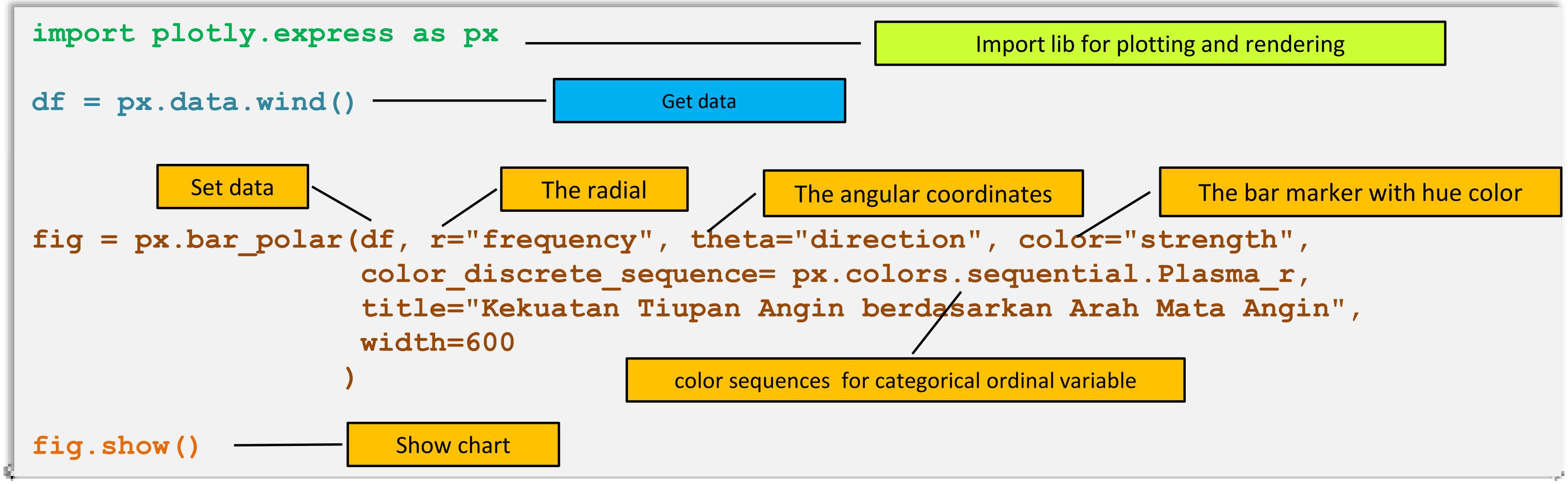
Source :

Dataset from plotly.express library



	direction	strength	frequency
0	N	0-1	0.5
1	NNE	0-1	0.6
2	NE	0-1	0.5
3	ENE	0-1	0.4

Source Code for Categorical – Radial



```
import plotly.express as px

df = px.data.wind()

fig = px.bar_polar(df, r="frequency", theta="direction", color="strength",
                   color_discrete_sequence= px.colors.sequential.Plasma_r,
                   title="Kekuatan Tiupan Angin berdasarkan Arah Mata Angin",
                   width=600
                  )

fig.show()
```

Import lib for plotting and rendering

Get data

Set data

The radial

The angular coordinates

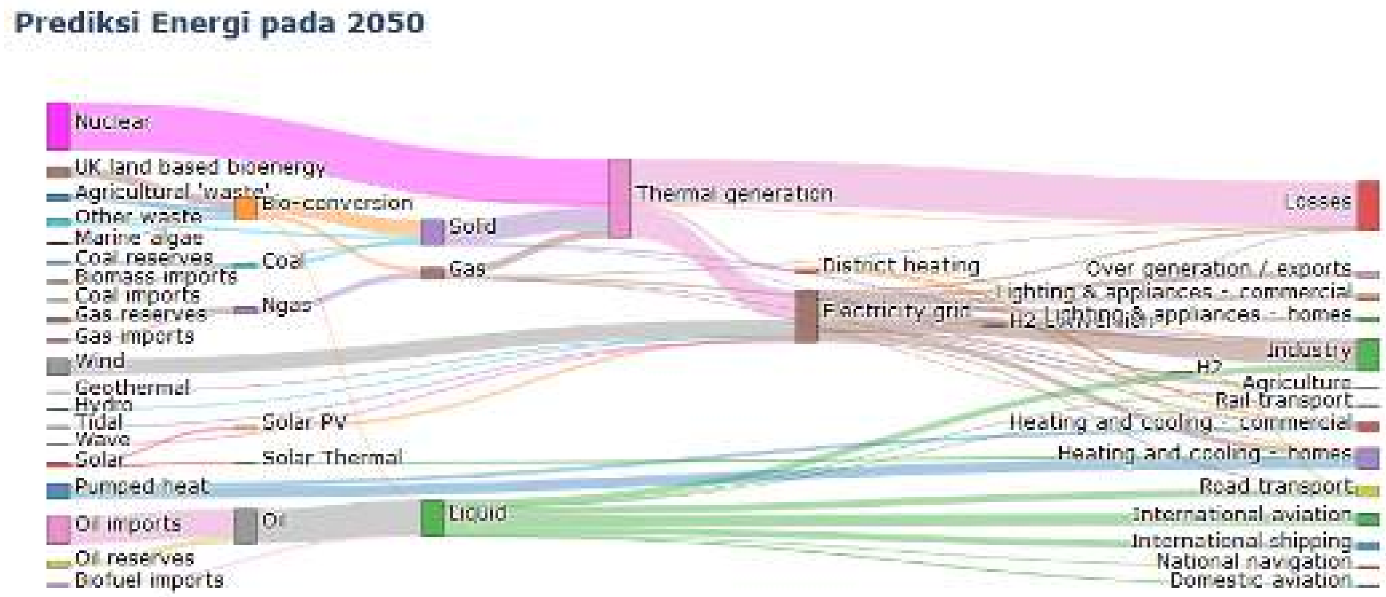
The bar marker with hue color

color sequences for categorical ordinal variable

Show chart

Note:
Polar charts,also known as radar charts. The regular chart of polar chart is a X-Y chart. In the polar chart, the X axis is wrapped around the perimeter while the Y axis extends from the center to the top. Read more https://www.highcharts.com/docs/chart-and-series-types/polar-chart

Categorical – Sankey Diagram



Data variables :

2 x categorical (source, target),

1 x quantitative (value)

Visual Variables:

Link, height, color-hue

Data :

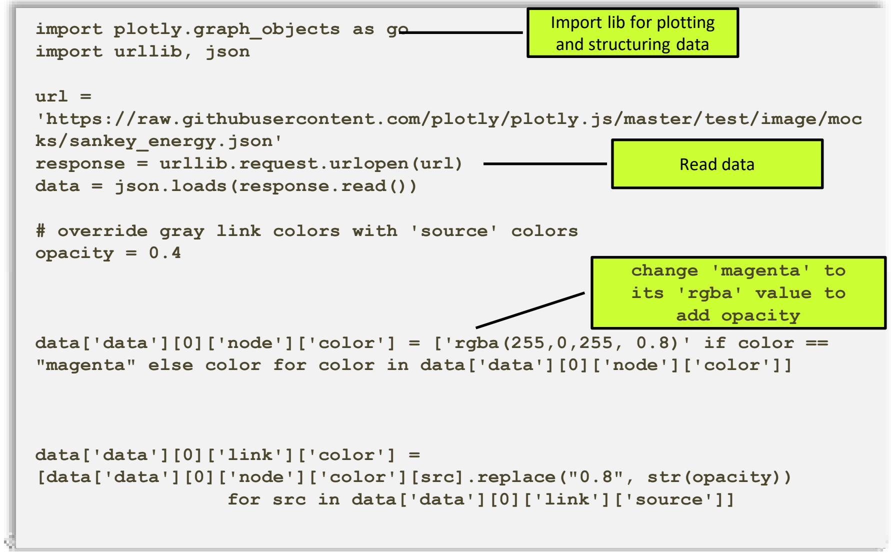
Energy Prediction in 2050.

Source :

https://raw.githubusercontent.com/plotly/plotly.js/master/test/image/mocks/sankey_energy.json

```
"link": {
     "source": [0, 1,..],
     "target": [4, 4,..],
     "value": [124.729,  0.597,..]
}, "node" : {
      "label": ["Agricultural 'waste'",..]}
```

Source Code for Categorical – Sankey Diagram

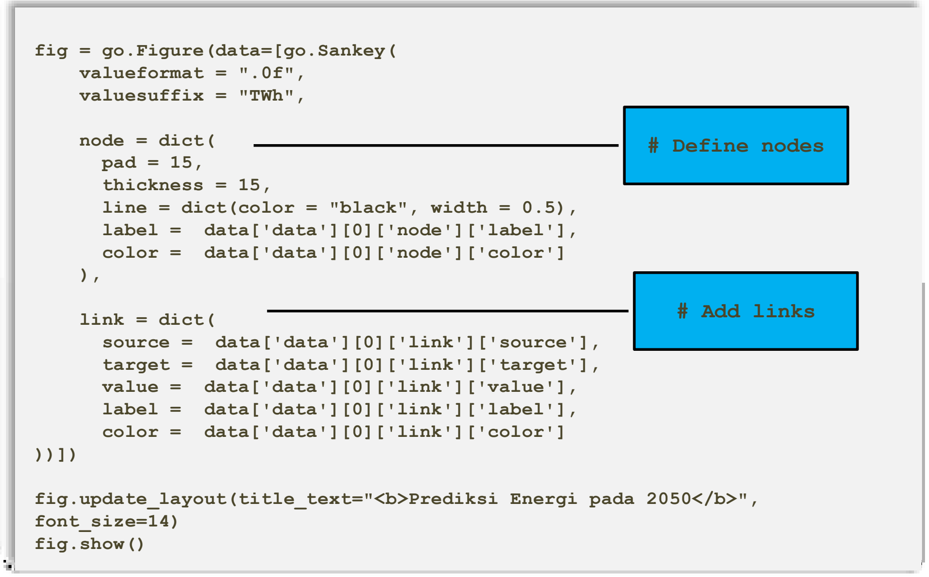


```
import plotly.graph_objects as go
import urllib, json

url =
'https://raw.githubusercontent.com/plotly/plotly.js/master/test/image/moc
ks/sankey_energy.json'
response = urllib.request.urlopen(url)
data = json.loads(response.read())

# override gray link colors with 'source' colors
opacity = 0.4



data['data'][0]['node']['color'] = ['rgba(255,0,255, 0.8)' if color ==
"magenta" else color for color in data['data'][0]['node']['color']]



data['data'][0]['link']['color'] =
[data['data'][0]['node']['color'][src].replace("0.8", str(opacity))
                 for src in data['data'][0]['link']['source']]
```

Import lib for plotting and structuring data

Read data

change 'magenta' to its 'rgba' value to add opacity



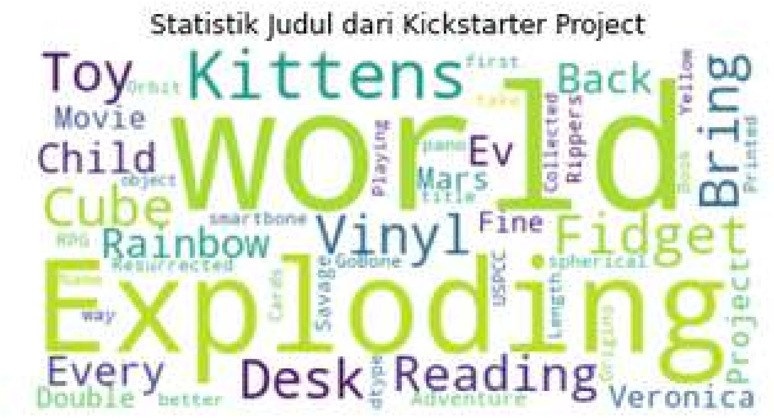
```
fig = go.Figure(data=[go.Sankey(
    valueformat = ".0f",
    valuesuffix = "TWh",

    node = dict(
      pad = 15,
      thickness = 15,
      line = dict(color = "black", width = 0.5),
      label =  data['data'][0]['node']['label'],
      color =  data['data'][0]['node']['color']
    ),

    link = dict(
      source =  data['data'][0]['link']['source'],
      target =  data['data'][0]['link']['target'],
      value =  data['data'][0]['link']['value'],
      label =  data['data'][0]['link']['label'],
      color =  data['data'][0]['link']['color']
))])

fig.update_layout(title_text="<b>Prediksi Energi pada 2050</b>",
font_size=14)
fig.show()
```

# Define nodes

# Add links

Note:
Sankey diagrams are used to convey the idea of flow.

Categorical – Word Cloud



Data variables :

1 x categorical (words),

1 x quantitative (count of word)

Visual Variables:

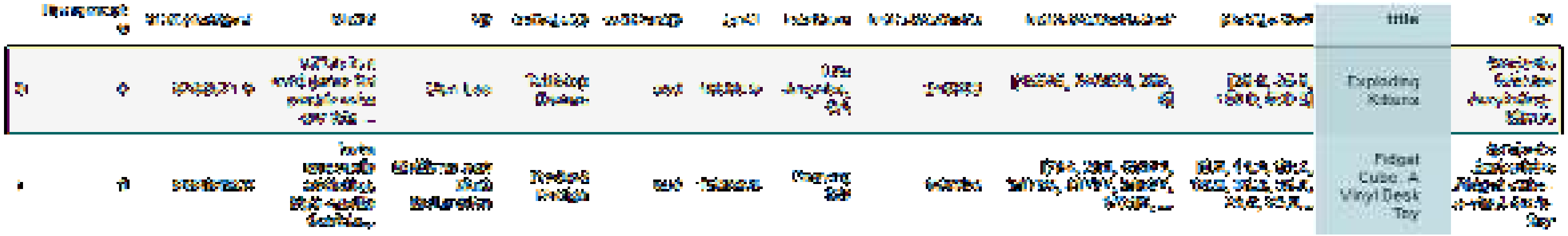
Size

Data :

Kickstarter (crowdfunding platform) Project Statistics.

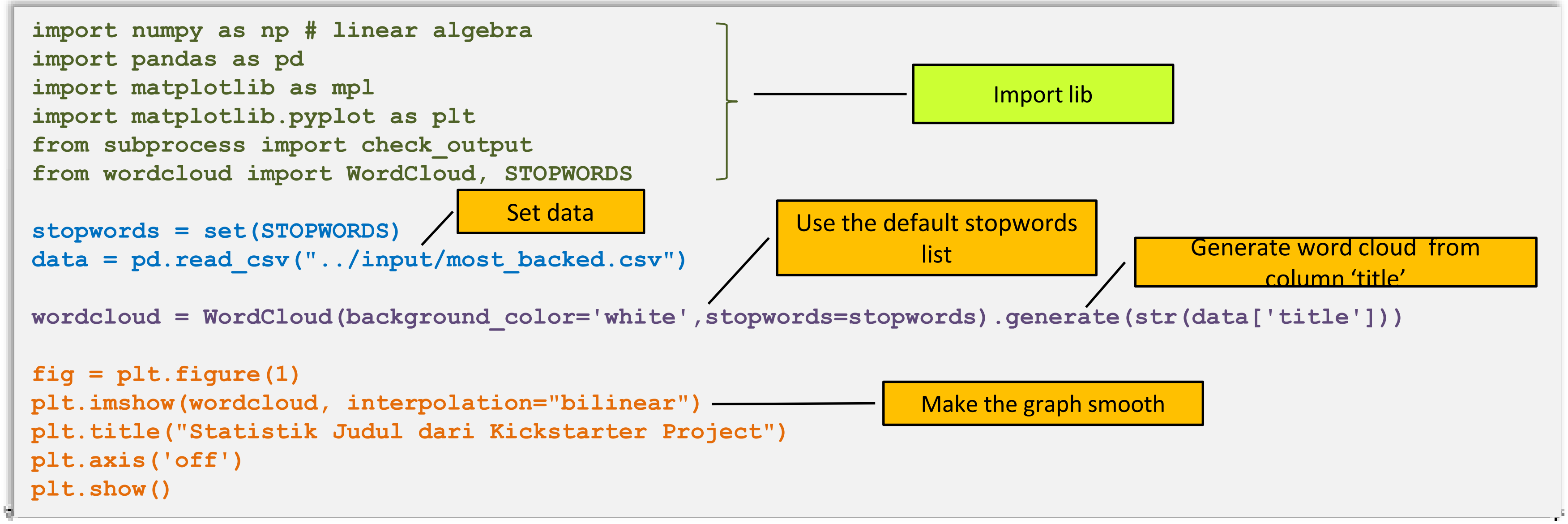
Source :

https://www.kaggle.com/adiljadoon/word-cloud-with-python/data?select=most_backed.c



[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	title	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Exploding Kittens	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Fidget Cube: A Vinyl Desk Toy	[illegible]

Source Code for Categorical – Word Cloud



```
import numpy as np # linear algebra
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from subprocess import check_output
from wordcloud import WordCloud, STOPWORDS

stopwords = set(STOPWORDS)
data = pd.read_csv("../input/most_backed.csv")

wordcloud = WordCloud(background_color='white',stopwords=stopwords).generate(str(data['title']))

fig = plt.figure(1)
plt.imshow(wordcloud, interpolation="bilinear")
plt.title("Statistik Judul dari Kickstarter Project")
plt.axis('off')
plt.show()
```

Import lib

Set data

Use the default stopwords list

Generate word cloud from column ‘title’

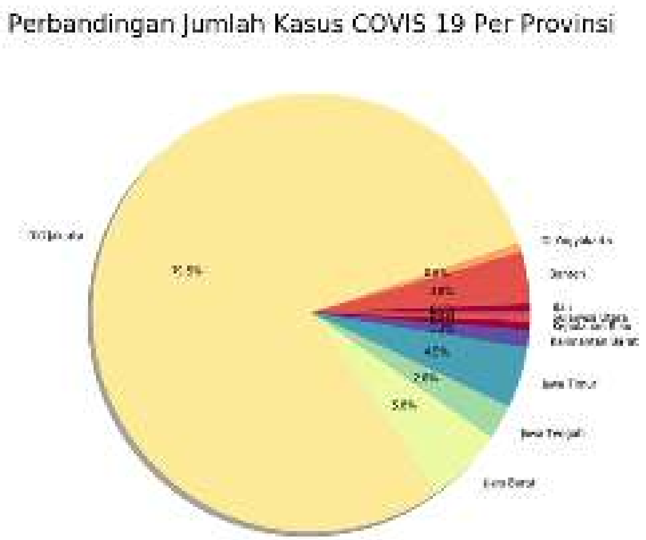
Make the graph smooth

Note:
Word clouds depict the frequency of words used in a given set of text. The font size indicates the quantity of each word's usage. Color is often just used as decoration (which you'll notice actually distorts the visual prominence) (Andy Kirk, 2012).

METHOD CLASSIFICATION

2. HIERARCHICAL

Hierarchical – Pie Chart



Data variables :

1 x categorical (province),

1 x quantitative (percentage of patient in each province)

Visual Variables:

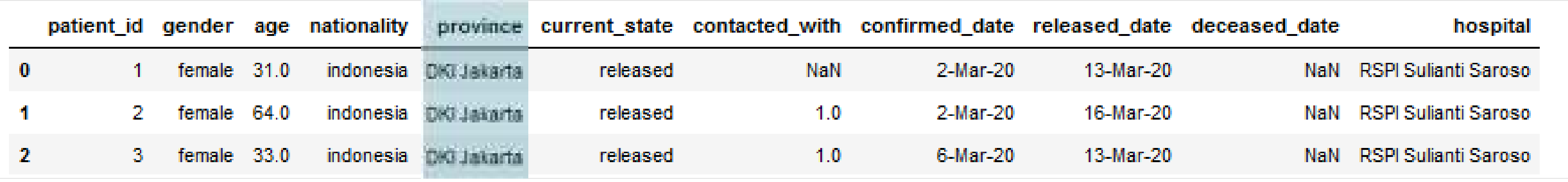
Angle, area, color-hue

Data :

Indonesia Coronavirus Cases

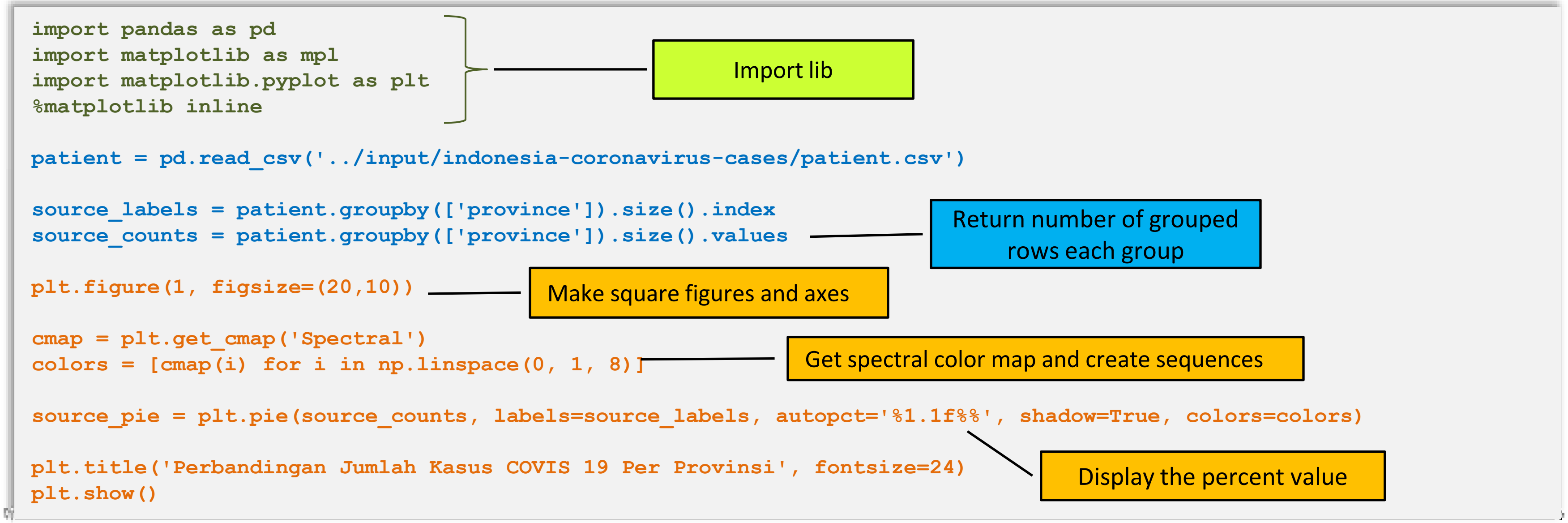
Source :

https://www.kaggle.com/ardisragen/indonesia-coronavirus-cases/



	patient_id	gender	age	nationality	province	current_state	contacted_with	confirmed_date	released_date	deceased_date	hospital
0	1	female	31.0	indonesia	DKI Jakarta	released	NaN	2-Mar-20	13-Mar-20	NaN	RSPI Sulianti Saroso
1	2	female	64.0	indonesia	DKI Jakarta	released	1.0	2-Mar-20	16-Mar-20	NaN	RSPI Sulianti Saroso
2	3	female	33.0	indonesia	DKI Jakarta	released	1.0	6-Mar-20	13-Mar-20	NaN	RSPI Sulianti Saroso

Source Code for Hierarchical – Pie Chart



```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline

patient = pd.read_csv('../input/indonesia-coronavirus-cases/patient.csv')

source_labels = patient.groupby(['province']).size().index
source_counts = patient.groupby(['province']).size().values

plt.figure(1, figsize=(20,10))

cmap = plt.get_cmap('Spectral')
colors = [cmap(i) for i in np.linspace(0, 1, 8)]

source_pie = plt.pie(source_counts, labels=source_labels, autopct='%1.1f%%', shadow=True, colors=colors)

plt.title('Perbandingan Jumlah Kasus COVIS 19 Per Provinsi', fontsize=24)
plt.show()
```

Import lib

Return number of grouped rows each group

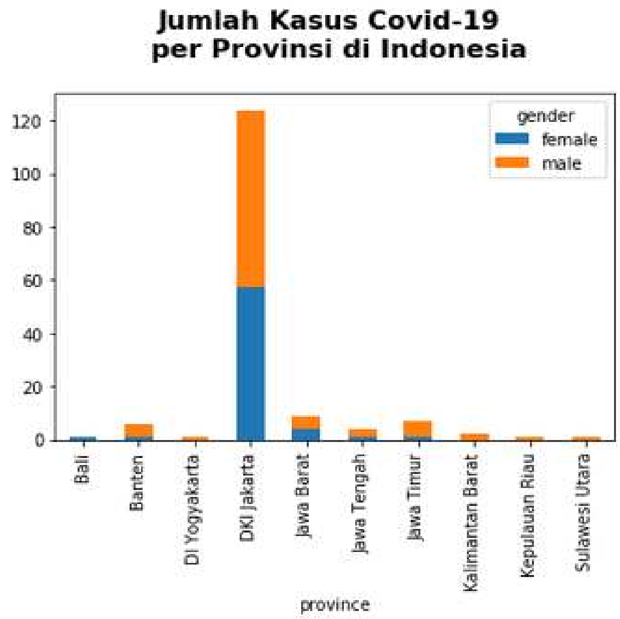
Make square figures and axes

Get spectral color map and create sequences

Display the percent value

Note:
A simple bar chart will suffice to demonstrate the part-to-whole relationship (Andy Kirk, 2012).

Hierarchical – Stacked Bar Chart



Data variables :

2 x categorical (province, gender)

1 x quantitative (count)

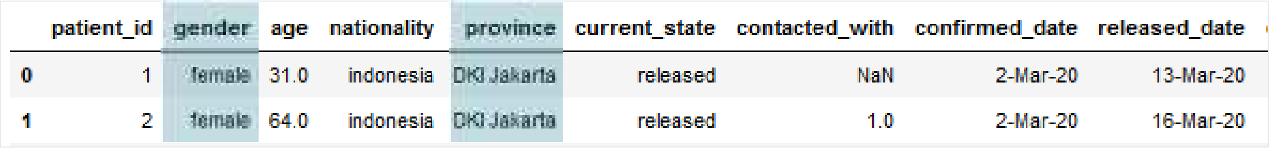
Visual Variables:

Length, position, color-hue

Data :

Indonesia Coronavirus Cases

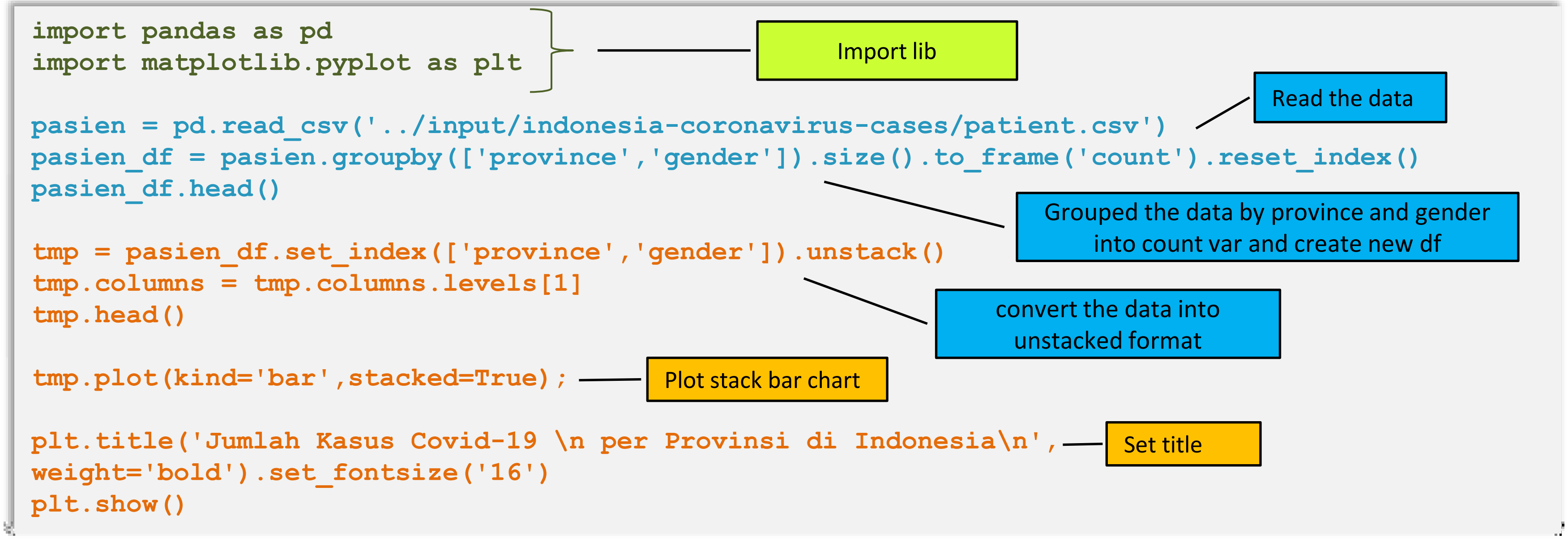
Source :
https://www.kaggle.com/ardisragen/indonesia-coronavirus-cases/



	patient_id	gender	age	nationality	province	current_state	contacted_with	confirmed_date	released_date
0	1	female	31.0	indonesia	DKI Jakarta	released	NaN	2-Mar-20	13-Mar-20
1	2	female	64.0	indonesia	DKI Jakarta	released	1.0	2-Mar-20	16-Mar-20

Count =
Group by province, gender

Source Code for Hierarchical – Stacked Bar Chart

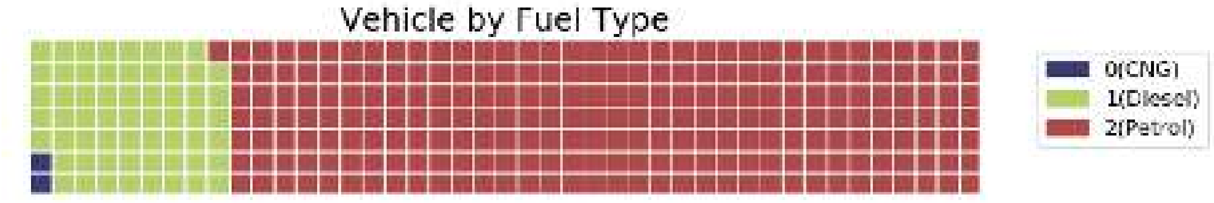


```
import pandas as pd
import matplotlib.pyplot as plt

pasien = pd.read_csv('../input/indonesia-coronavirus-cases/patient.csv')
pasien_df = pasien.groupby(['province','gender']).size().to_frame('count').reset_index()
pasien_df.head()

tmp = pasien_df.set_index(['province','gender']).unstack()
tmp.columns = tmp.columns.levels[1]
tmp.head()

tmp.plot(kind='bar',stacked=True);

plt.title('Jumlah Kasus Covid-19 \n per Provinsi di Indonesia\n',
weight='bold').set_fontsize('16')
plt.show()
```

Import lib
Read the data
Grouped the data by province and gender into count var and create new df
convert the data into unstacked format
Plot stack bar chart
Set title

Note:
Word clouds depict the frequency of words used in a given set of text. The font size indicates the quantity of each word's usage. Color is often just used as decoration (which you'll notice actually distorts the visual prominence) (Andy Kirk, 2012). Read more about stack and unstack dataframe https://www.datasciencemadesimple.com/reshape-using-stack-unstack-function-pandas-python/

Hierarchical – Square Pie Chart



Data variables :

1 x categorical (fuel type)

1 x quantitative (count)

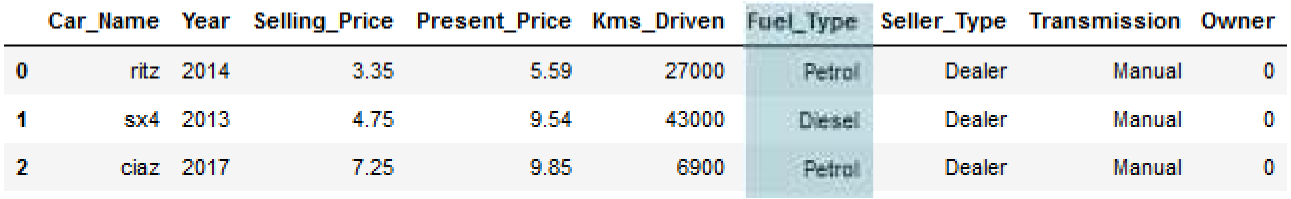
Visual Variables:

Length, position, color-hue

Data :

informasi mobil bekas yang terdaftar di www.cardekho.com.

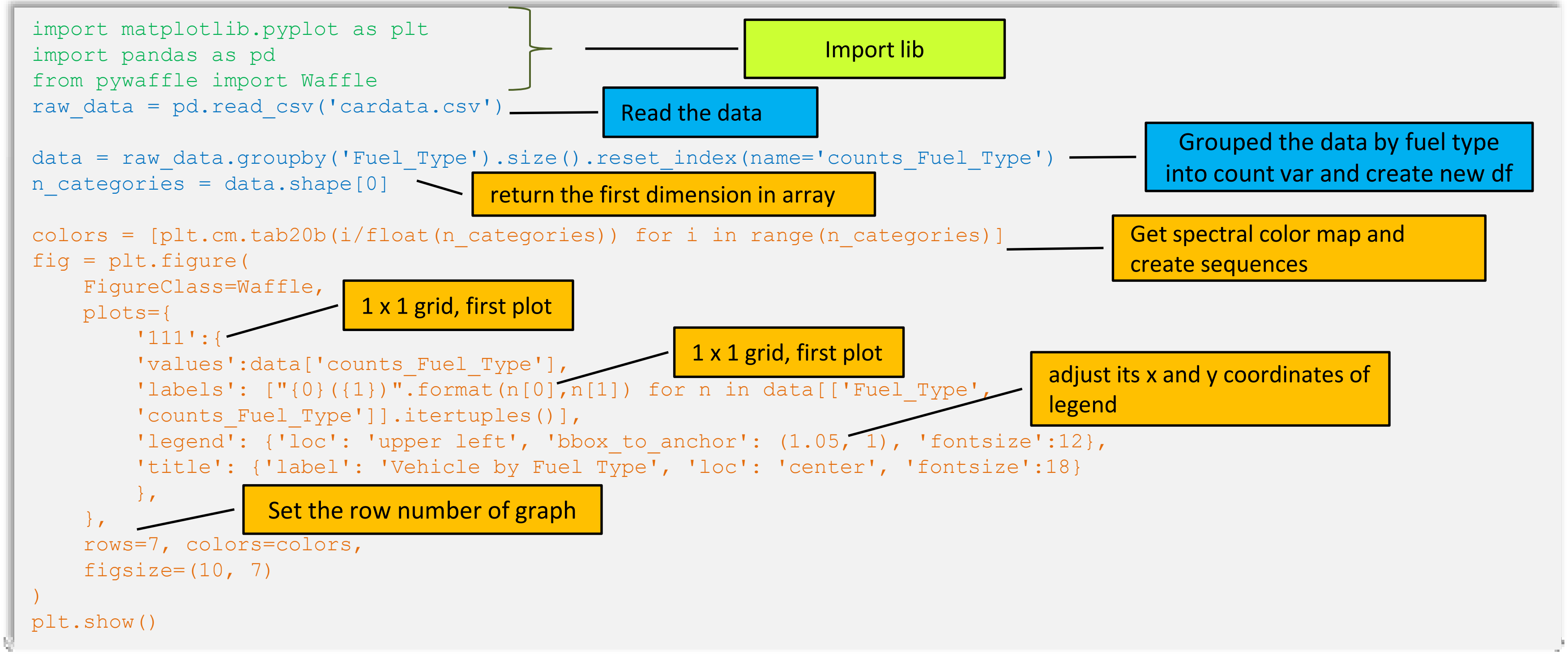
Source : https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho/version/1



	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0

Count =
Group by fuel type

Source Code for Hierarchical – Square Bar Chart



```
import matplotlib.pyplot as plt
import pandas as pd
from pywaffle import Waffle
raw_data = pd.read_csv('cardata.csv')

data = raw_data.groupby('Fuel_Type').size().reset_index(name='counts_Fuel_Type')
n_categories = data.shape[0]

colors = [plt.cm.tab20b(i/float(n_categories)) for i in range(n_categories)]
fig = plt.figure(
    FigureClass=Waffle,
    plots={
        '111':{
        'values':data['counts_Fuel_Type'],
        'labels': ["{0}({1})".format(n[0],n[1]) for n in data[['Fuel_Type',
        'counts_Fuel_Type']].itertuples()],
        'legend': {'loc': 'upper left', 'bbox_to_anchor': (1.05, 1), 'fontsize':12},
        'title': {'label': 'Vehicle by Fuel Type', 'loc': 'center', 'fontsize':18}
        },
    },
    rows=7, colors=colors,
    figsize=(10, 7)
)
plt.show()
```

Import lib

Read the data

Grouped the data by fuel type into count var and create new df

return the first dimension in array

Get spectral color map and create sequences

1 x 1 grid, first plot

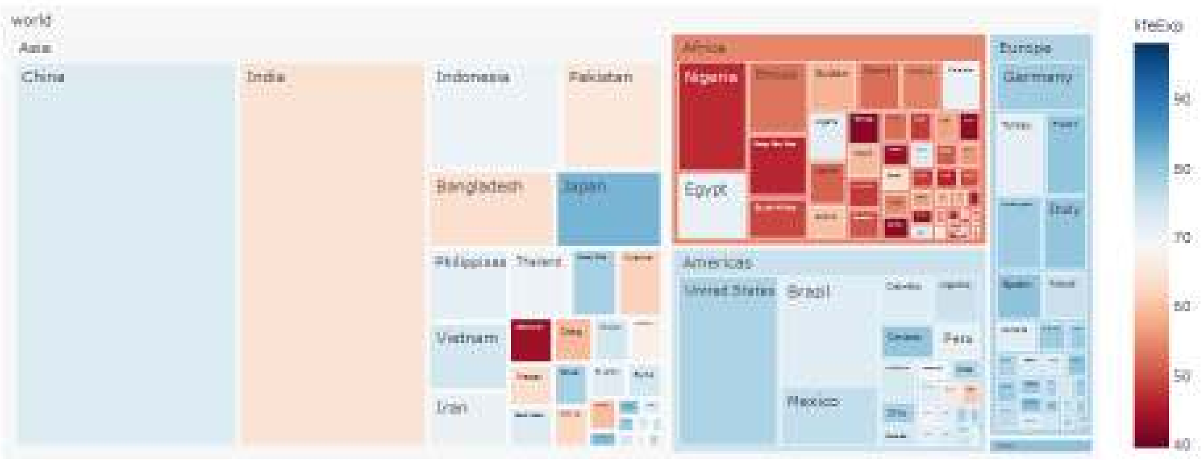
1 x 1 grid, first plot

adjust its x and y coordinates of legend

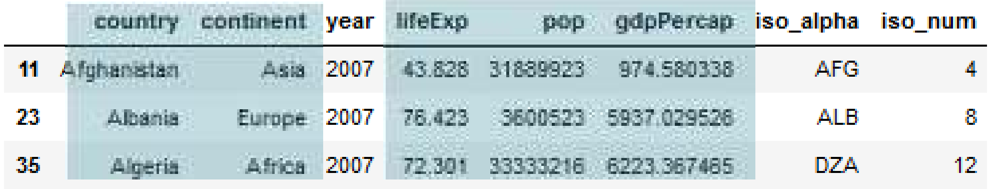
Set the row number of graph

Note:
There are several titles for this type of chart but the common technique involves a grid of units (may be squares or symbols) to represent parts of a whole.

Hierarchical – Tree Map



Population and Life Expectancy for Countries in the World



	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
11	Afghanistan	Asia	2007	43.828	31889923	974.580338	AFG	4
23	Albania	Europe	2007	76.423	3600523	5937.029526	ALB	8
35	Algeria	Africa	2007	72.301	33333216	6223.367465	DZA	12

Data variables :

1 x categorical (country)

1 x quantitative (population)

1 x quantitative (lifeExp)

Visual Variables:

Area, position, color-hue, color-saturation

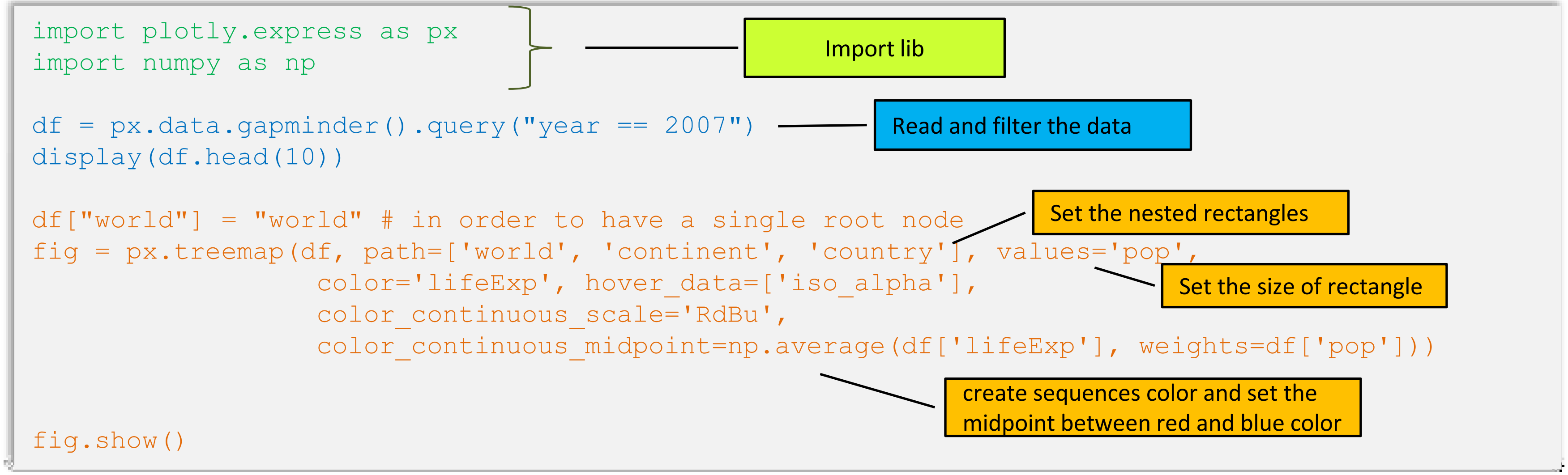
Data :

The *data* for pppulation and *life expectancy* for countries from *Gapminder*

Source :

Gapminder dataset from plotly library

Source Code for Hierarchical – Tree Map



```
import plotly.express as px
import numpy as np

df = px.data.gapminder().query("year == 2007")
display(df.head(10))

df["world"] = "world" # in order to have a single root node
fig = px.treemap(df, path=['world', 'continent', 'country'], values='pop',
                  color='lifeExp', hover_data=['iso_alpha'],
                  color_continuous_scale='RdBu',
                  color_continuous_midpoint=np.average(df['lifeExp'], weights=df['pop']))


fig.show()
```

Import lib

Read and filter the data

Set the nested rectangles

Set the size of rectangle

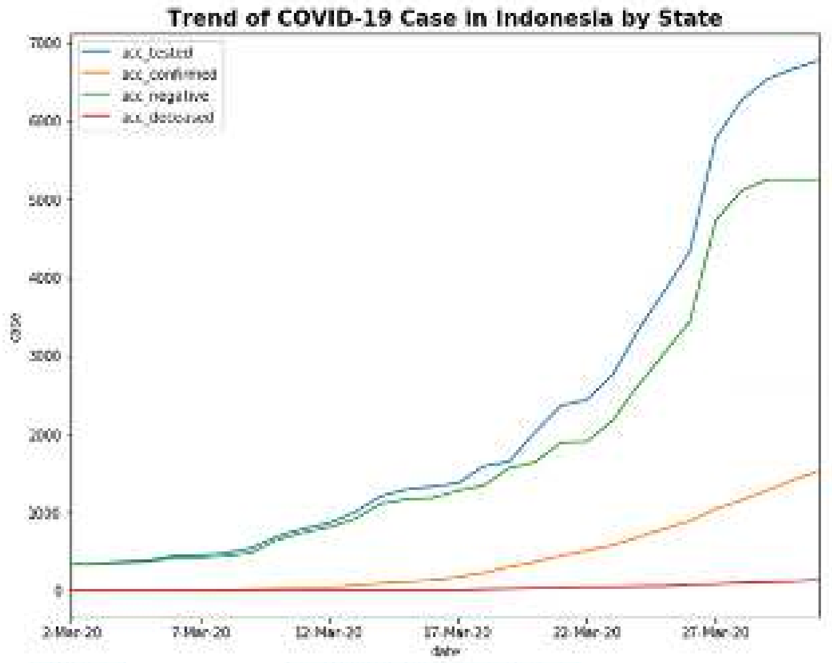
create sequences color and set the midpoint between red and blue color

Note.
Tree maps take the concept of a whole population and divide up portions of rectangular spaces within to represent organized, clustered constituent units sized according to their relative value. As well as arrangement, various properties of color are typically used to provide additional layers of quantitative or categorical insight (Andy Kirk, 2012)

METHOD CLASSIFICATION

3. TEMPORAL - SHOWING CHANGES OVER TIME

Temporal – Line Chart



Data variables :

1 x categorical (status)

4 x quantitative (acc_tested, acc_confirmed, acc_negative, acc_deceased)

1 x quantitative interval (date)

Visual Variables:

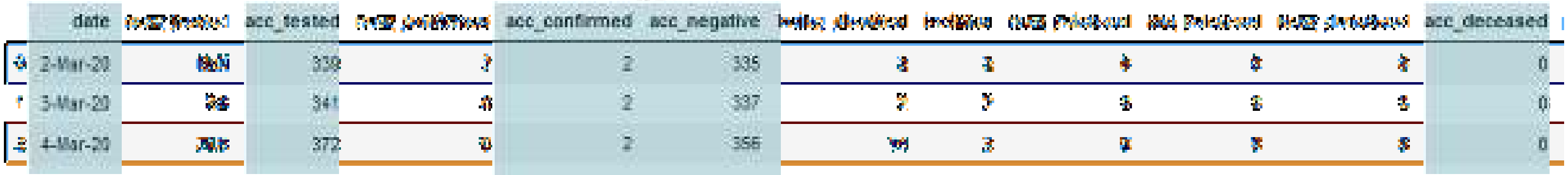
Position, slope, color-hue

Data :

Indonesia Coronavirus Cases

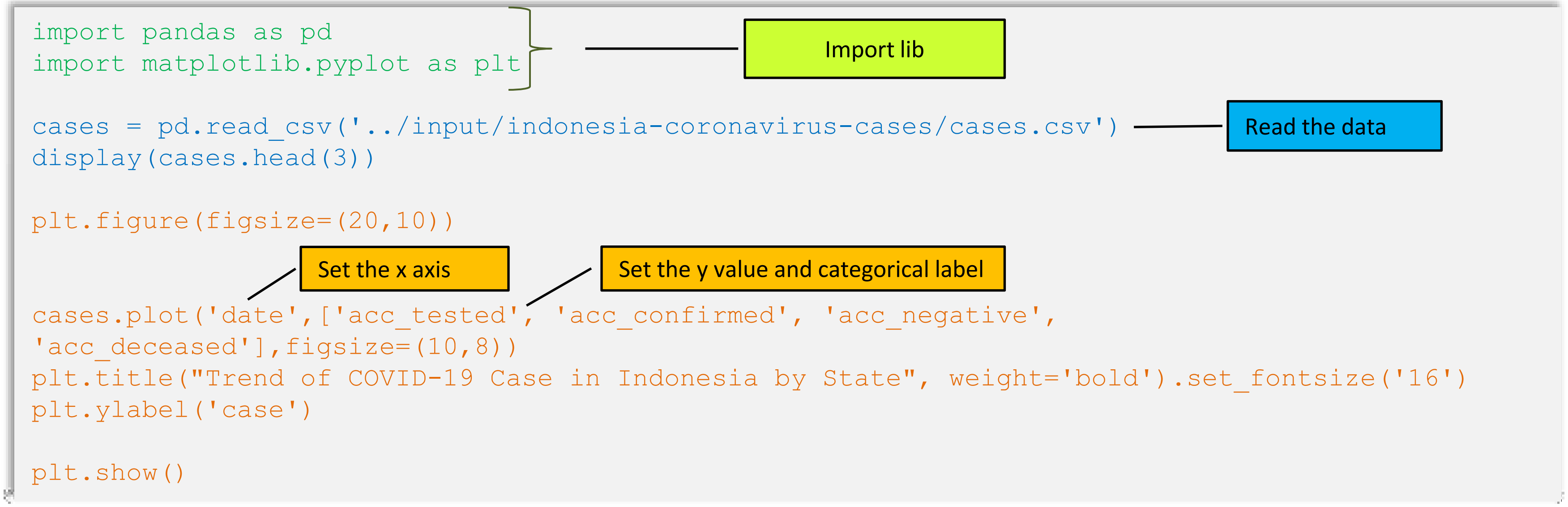
Source :

https://www.kaggle.com/ardisragen/indonesia-coronavirus-cases/



	date	[illegible]	acc_tested	[illegible]	acc_confirmed	acc_negative	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	acc_deceased
[illegible]	2-Mar-20	[illegible]	339	[illegible]	2	335	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0
[illegible]	3-Mar-20	[illegible]	341	[illegible]	2	337	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0
[illegible]	4-Mar-20	[illegible]	372	[illegible]	2	356	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0

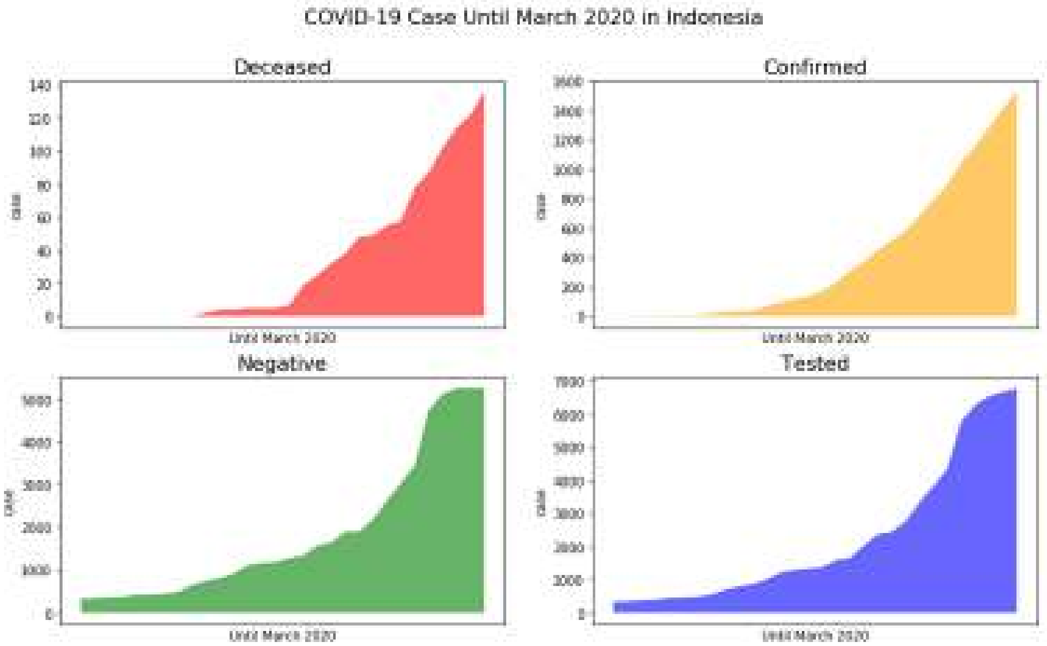
Source Code for Temporal – Line Chart



```
import pandas as pd
import matplotlib.pyplot as plt

cases = pd.read_csv('../input/indonesia-coronavirus-cases/cases.csv')
display(cases.head(3))

plt.figure(figsize=(20,10))

cases.plot('date',['acc_tested', 'acc_confirmed', 'acc_negative',
'acc_deceased'],figsize=(10,8))
plt.title("Trend of COVID-19 Case in Indonesia by State", weight='bold').set_fontsize('16')
plt.ylabel('case')

plt.show()
```

Note:
Line charts are used to compare a continuous quantitative variable on the x axis and the size of values on the y axis . Unlike bar charts, the y axis doesn't need to start from zero because we are looking at the relative pattern of the data journey (Andy Kirk, 2012).

Temporal – Area Chart



Data variables :

1 x categorical (status)

4 x quantitative (acc_tested, acc_confirmed, acc_negative, acc_deceased)

1 x quantitative interval (date)

Visual Variables:

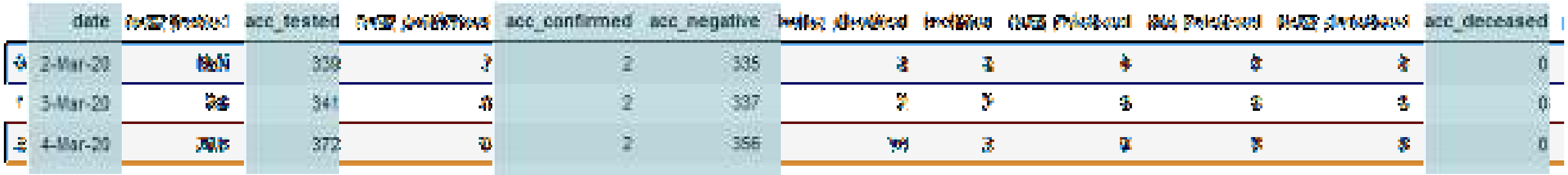
Height, slope, area, color-hue

Data :

Indonesia Coronavirus Cases

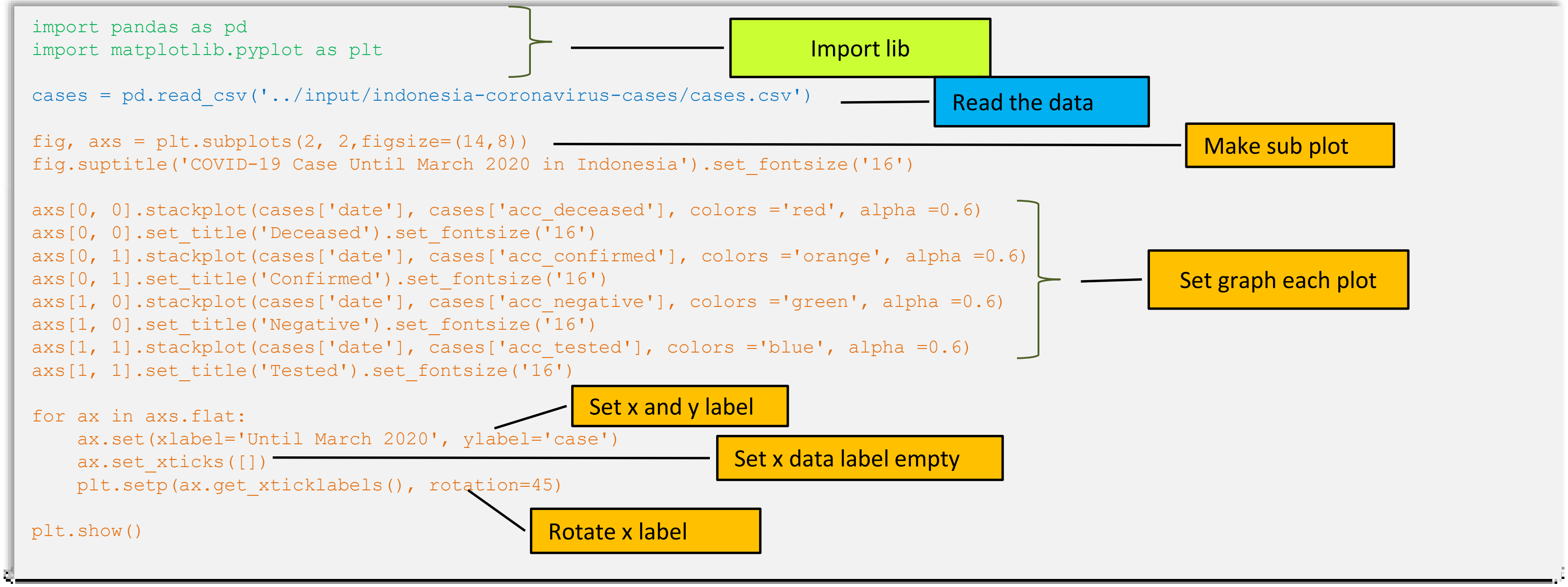
Source :

https://www.kaggle.com/ardisragen/indonesia-coronavirus-cases/



	date	[illegible]	acc_tested	[illegible]	acc_confirmed	acc_negative	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	acc_deceased
[illegible]	2-Mar-20	[illegible]	339	[illegible]	2	335	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0
[illegible]	3-Mar-20	[illegible]	341	[illegible]	2	337	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0
[illegible]	4-Mar-20	[illegible]	372	[illegible]	2	356	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0

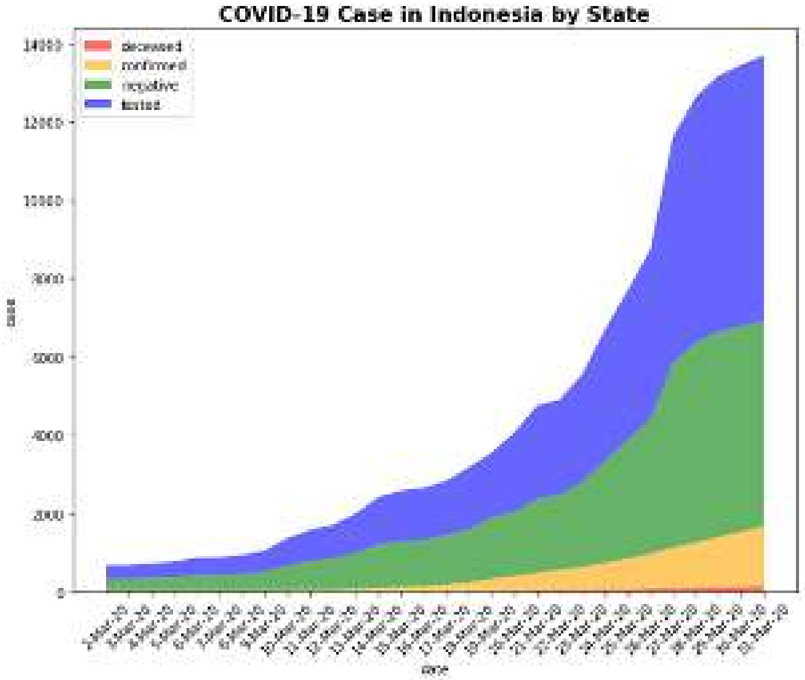
Source Code for Temporal – Area Chart



```
import pandas as pd
import matplotlib.pyplot as plt

cases = pd.read_csv('../input/indonesia-coronavirus-cases/cases.csv')

fig, axs = plt.subplots(2, 2,figsize=(14,8))
fig.suptitle('COVID-19 Case Until March 2020 in Indonesia').set_fontsize('16')

axs[0, 0].stackplot(cases['date'], cases['acc_deceased'], colors ='red', alpha =0.6)
axs[0, 0].set_title('Deceased').set_fontsize('16')
axs[0, 1].stackplot(cases['date'], cases['acc_confirmed'], colors ='orange', alpha =0.6)
axs[0, 1].set_title('Confirmed').set_fontsize('16')
axs[1, 0].stackplot(cases['date'], cases['acc_negative'], colors ='green', alpha =0.6)
axs[1, 0].set_title('Negative').set_fontsize('16')
axs[1, 1].stackplot(cases['date'], cases['acc_tested'], colors ='blue', alpha =0.6)
axs[1, 1].set_title('Tested').set_fontsize('16')

for ax in axs.flat:
    ax.set(xlabel='Until March 2020', ylabel='case')
    ax.set_xticks([])
    plt.setp(ax.get_xticklabels(), rotation=45)

plt.show()
```

Note:
Unlike a standard line chart, an area chart should have the y axis starting at zero to ensure the area judgment is being interpreted accurately (Andy Kirk, 2012).

Temporal – Stacked Area Chart



Data variables :

1 x categorical (status)

4 x quantitative (acc_tested, acc_confirmed, acc_negative, acc_deceased)

1 x quantitative interval (date)

Visual Variables:

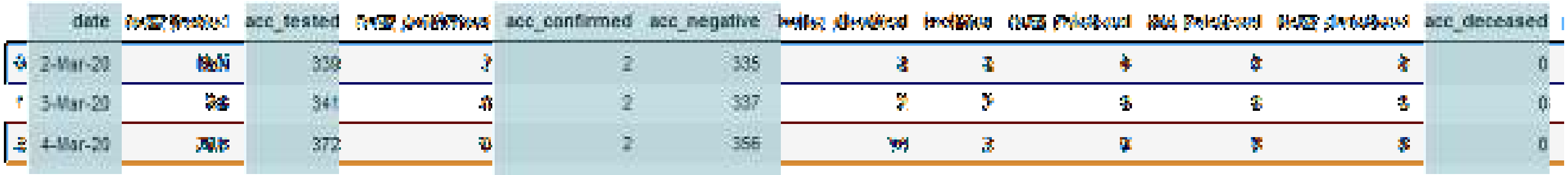
Height, slope, area, color-hue

Data :

Indonesia Coronavirus Cases

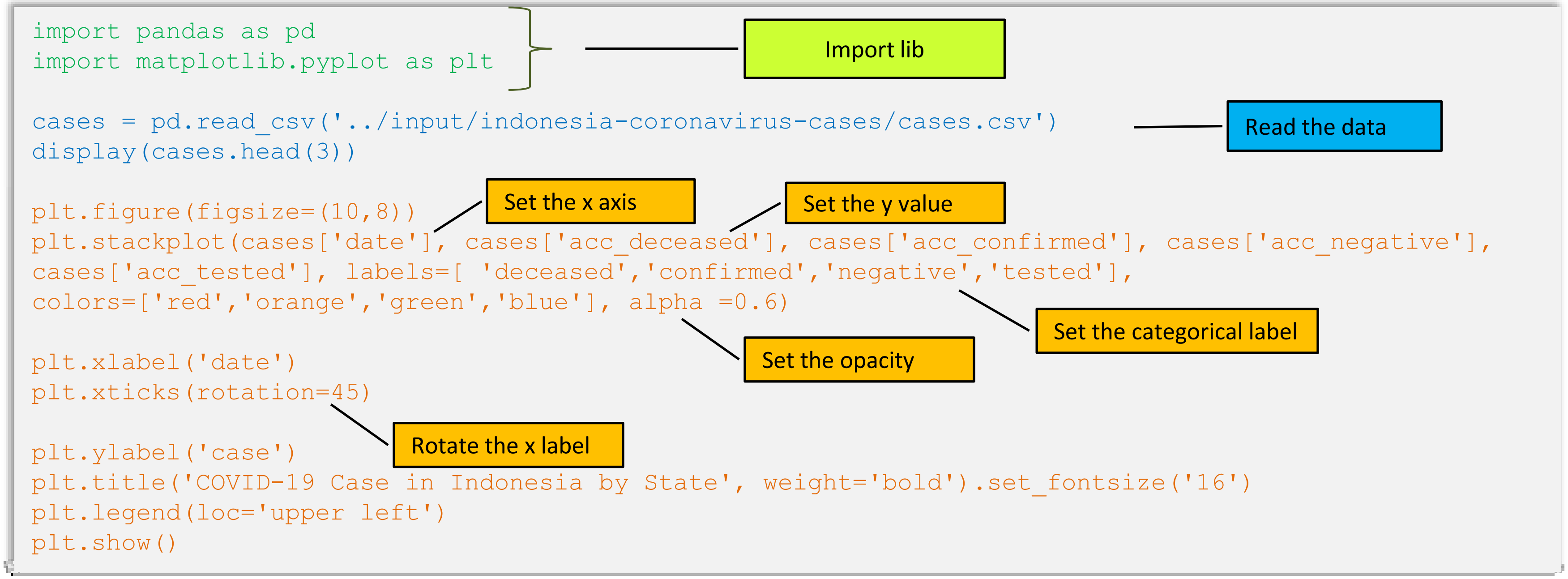
Source :

https://www.kaggle.com/ardisragen/indonesia-coronavirus-cases/



	date	[illegible]	acc_tested	[illegible]	acc_confirmed	acc_negative	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	acc_deceased
[illegible]	2-Mar-20	[illegible]	339	[illegible]	2	335	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0
1	3-Mar-20	[illegible]	341	[illegible]	2	337	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0
[illegible]	4-Mar-20	[illegible]	372	[illegible]	2	356	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0

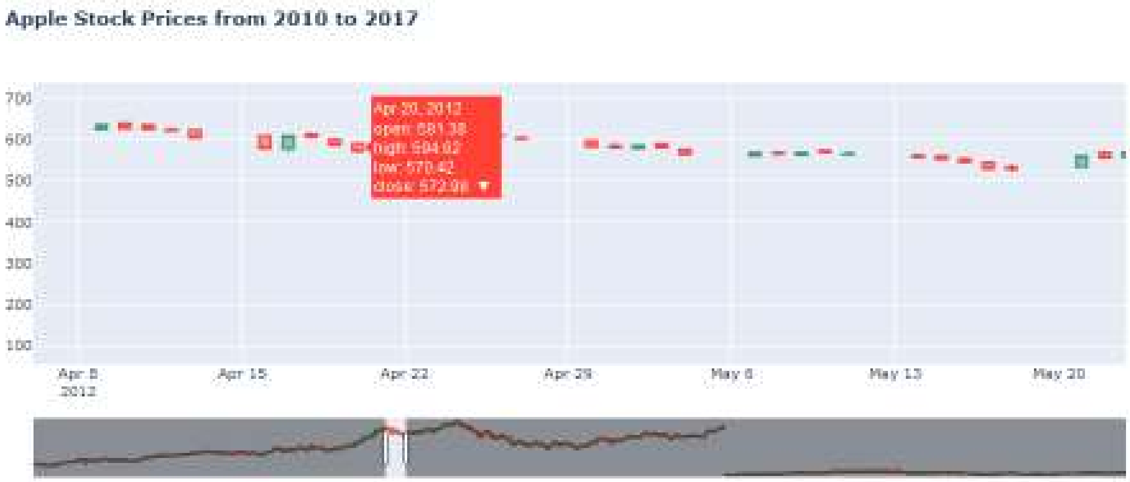
Source Code for Temporal – Stacked Area Chart



```
import pandas as pd
import matplotlib.pyplot as plt

cases = pd.read_csv('../input/indonesia-coronavirus-cases/cases.csv')
display(cases.head(3))

plt.figure(figsize=(10,8))
plt.stackplot(cases['date'], cases['acc_deceased'], cases['acc_confirmed'], cases['acc_negative'],
cases['acc_tested'], labels=[ 'deceased','confirmed','negative','tested'],
colors=['red','orange','green','blue'], alpha =0.6)

plt.xlabel('date')
plt.xticks(rotation=45)

plt.ylabel('case')
plt.title('COVID-19 Case in Indonesia by State', weight='bold').set_fontsize('16')
plt.legend(loc='upper left')
plt.show()
```

Note:
Unlike a standard line chart, an area chart should have the y axis starting at zero to ensure the area judgment is being interpreted accurately (Andy Kirk, 2012).

Temporal – Candlestick Chart



Data variables :

1 x quantitative-interval (date)

4 x quantitative-ratio(open, high, low, close)

Visual Variables:

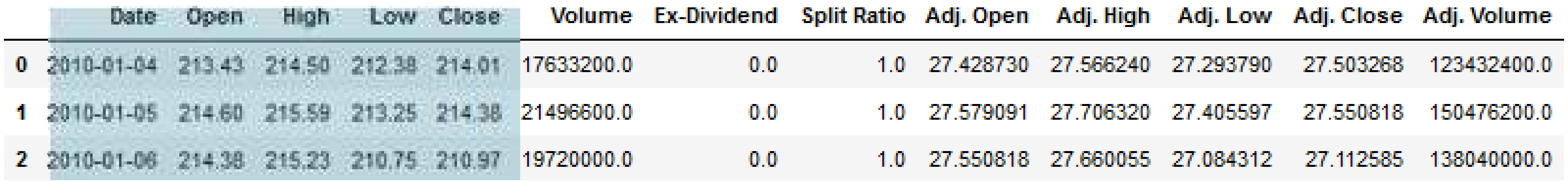
Position, height, color-hue

Data :

Apple Stock Prices from 2010 to 2017

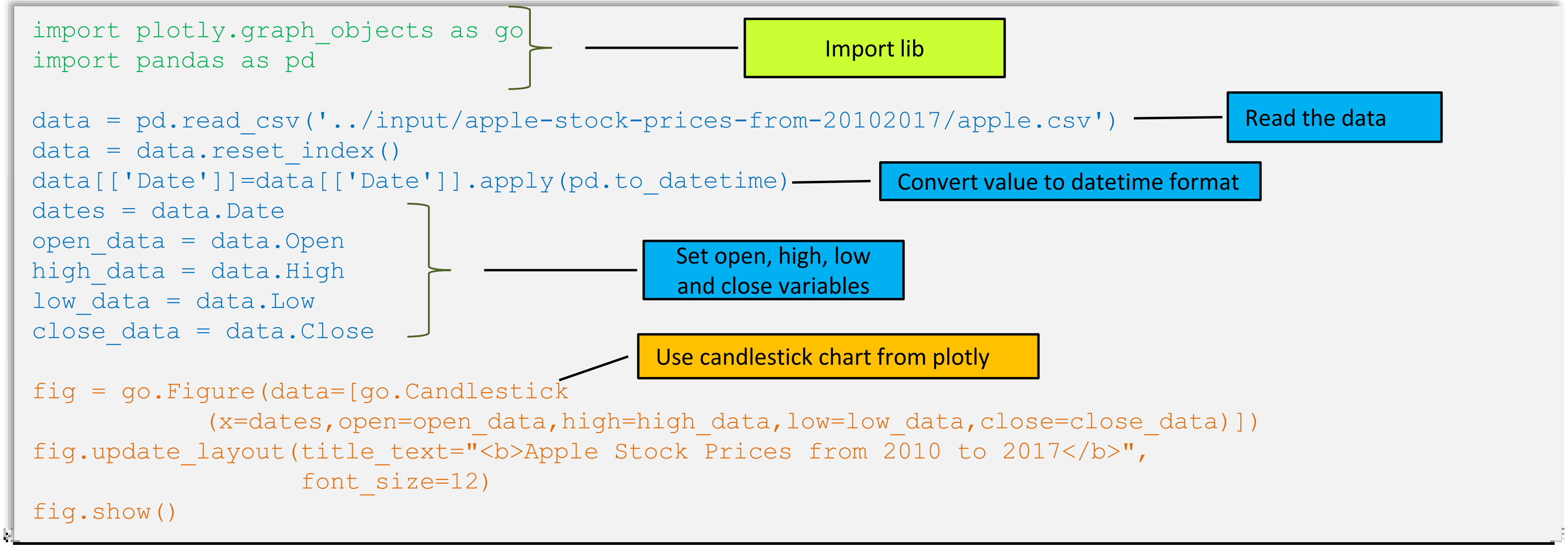
Source :

https://www.kaggle.com/fayomi/apple-stock-prices-from-20102017/notebooks



	Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
0	2010-01-04	213.43	214.50	212.38	214.01	17633200.0	0.0	1.0	27.428730	27.566240	27.293790	27.503268	123432400.0
1	2010-01-05	214.60	215.59	213.25	214.38	21496600.0	0.0	1.0	27.579091	27.706320	27.405597	27.550818	150476200.0
2	2010-01-06	214.38	215.23	210.75	210.97	19720000.0	0.0	1.0	27.550818	27.660055	27.084312	27.112585	138040000.0

Source Code for Temporal – Candlestick Chart



```
import plotly.graph_objects as go
import pandas as pd

data = pd.read_csv('../input/apple-stock-prices-from-20102017/apple.csv')
data = data.reset_index()
data[['Date']]=data[['Date']].apply(pd.to_datetime)
dates = data.Date
open_data = data.Open
high_data = data.High
low_data = data.Low
close_data = data.Close

fig = go.Figure(data=[go.Candlestick
            (x=dates,open=open_data,high=high_data,low=low_data,close=close_data)])
fig.update_layout(title_text="<b>Apple Stock Prices from 2010 to 2017</b>",
                  font_size=12)
fig.show()
```

Import lib

Read the data

Convert value to datetime format

Set open, high, low and close variables

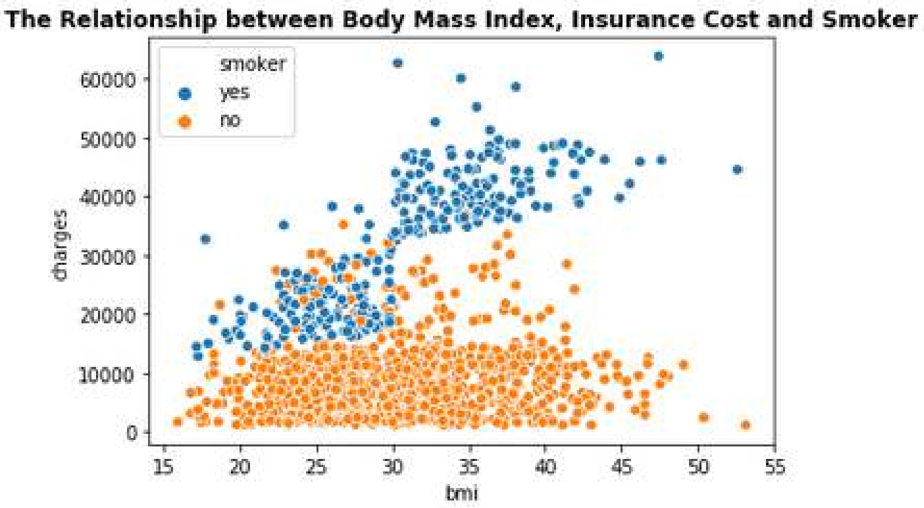
Use candlestick chart from plotly

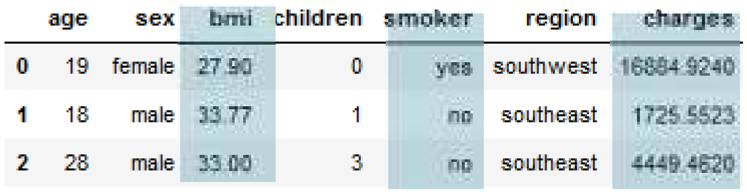
Note:
The candlestick chart is a style of financial chart describing open, high, low and close for a given x coordinate (most likely time). The boxes represent the spread between the open and close values and the lines represent the spread between the low and high values. Sample points where the close value is higher (lower) then the open value are called increasing (decreasing). By default, increasing candles are drawn in green whereas decreasing are drawn in red.(read more https://plotly.com/python/candlestick-charts/)

METHOD CLASSIFICATION

4. RELATIONAL - PLOTTING CONNECTION AND RELATIONSHIP

Temporal – Scatter Plot





	age	sex	bmi	children	smoker	region	charges
0	19	female	27.90	0	yes	southwest	16884.9240
1	18	male	33.77	1	no	southeast	1725.5523
2	28	male	33.00	3	no	southeast	4449.4620

Data variables :

2 x quantitative (bmi, charges)

1 x categorical (smoker)

Visual Variables:

Position, color-hue

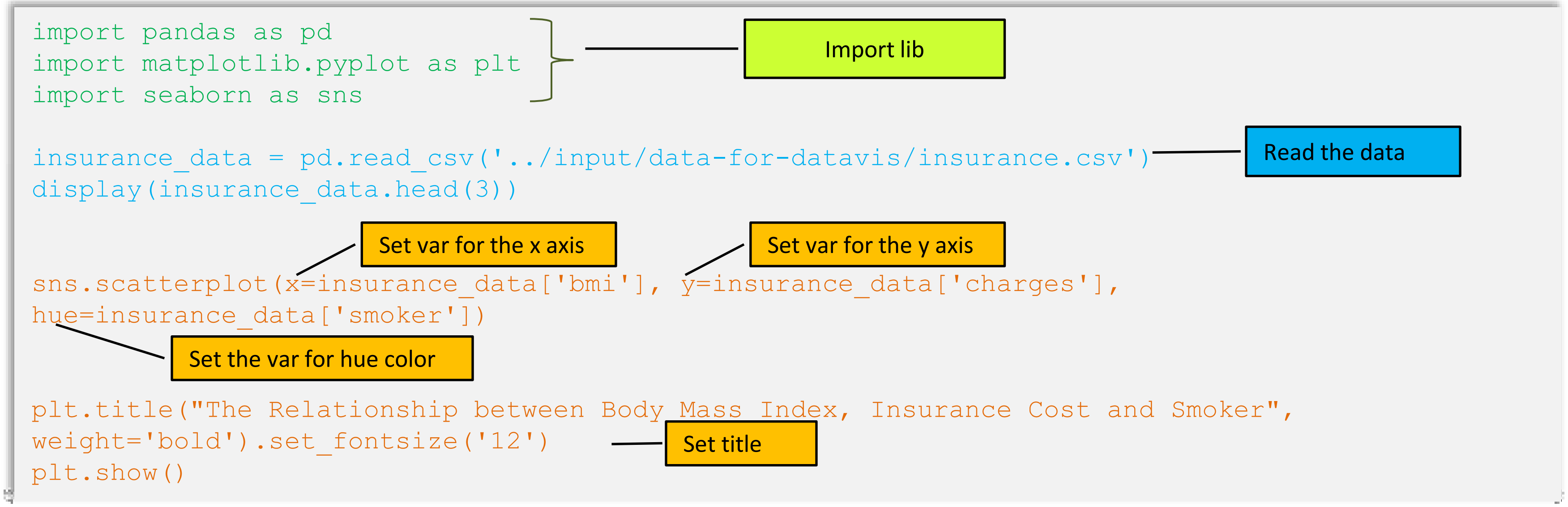
Data :

Insurance Dataset

Source :

https://www.kaggle.com/alexisbcook/data-for-datavis

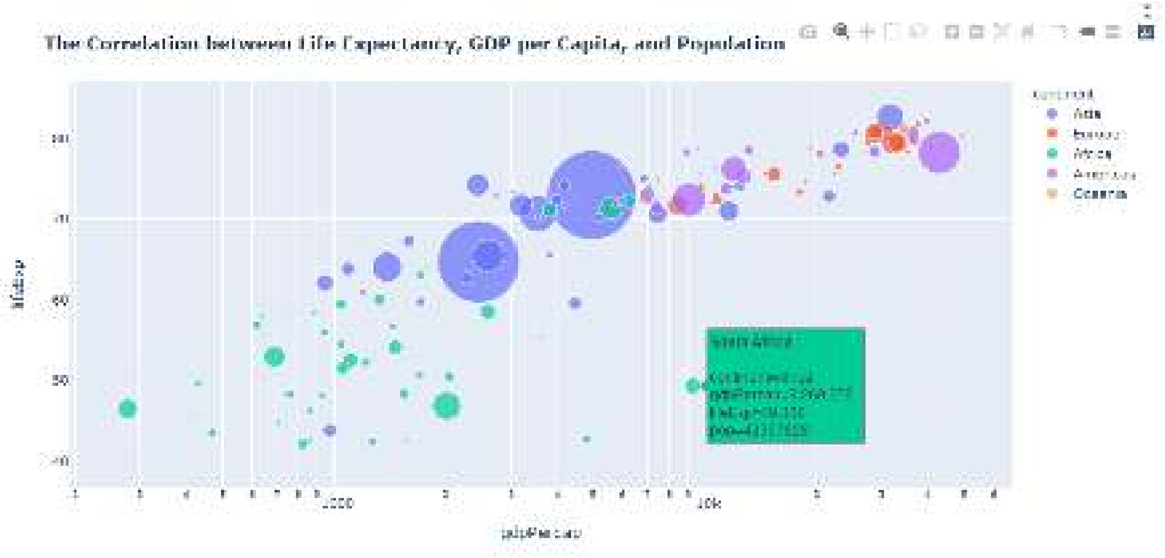
Source Code for Temporal – Scatter Plot

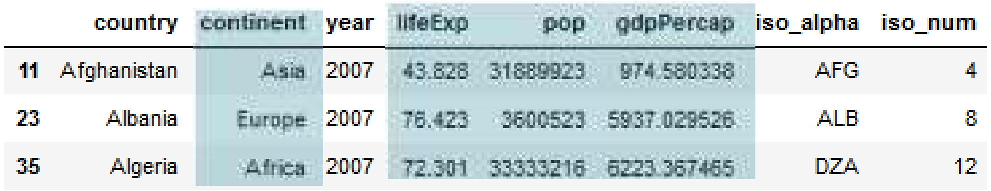


```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

insurance_data = pd.read_csv('../input/data-for-datavis/insurance.csv')
display(insurance_data.head(3))

sns.scatterplot(x=insurance_data['bmi'], y=insurance_data['charges'],
hue=insurance_data['smoker'])

plt.title("The Relationship between Body Mass Index, Insurance Cost and Smoker",
weight='bold').set_fontsize('12')
plt.show()
```

Note:
A scatter plot is a combination of two quantitative variables plotted on to the x and y axes in order to reveal patterns of correlations, clustering, and outliers. (Andy Kirk, 2012).

Correlation – Bubble Plot





	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
11	Afghanistan	Asia	2007	43.828	31889923	974.580338	AFG	4
23	Albania	Europe	2007	76.423	3600523	5937.029526	ALB	8
35	Algeria	Africa	2007	72.301	33333216	6223.367465	DZA	12

Data variables :

1 x categorical (continent)

1 x quantitative (population, lifeExp, gdpPercap)

Visual Variables:

Area, position, color-hue

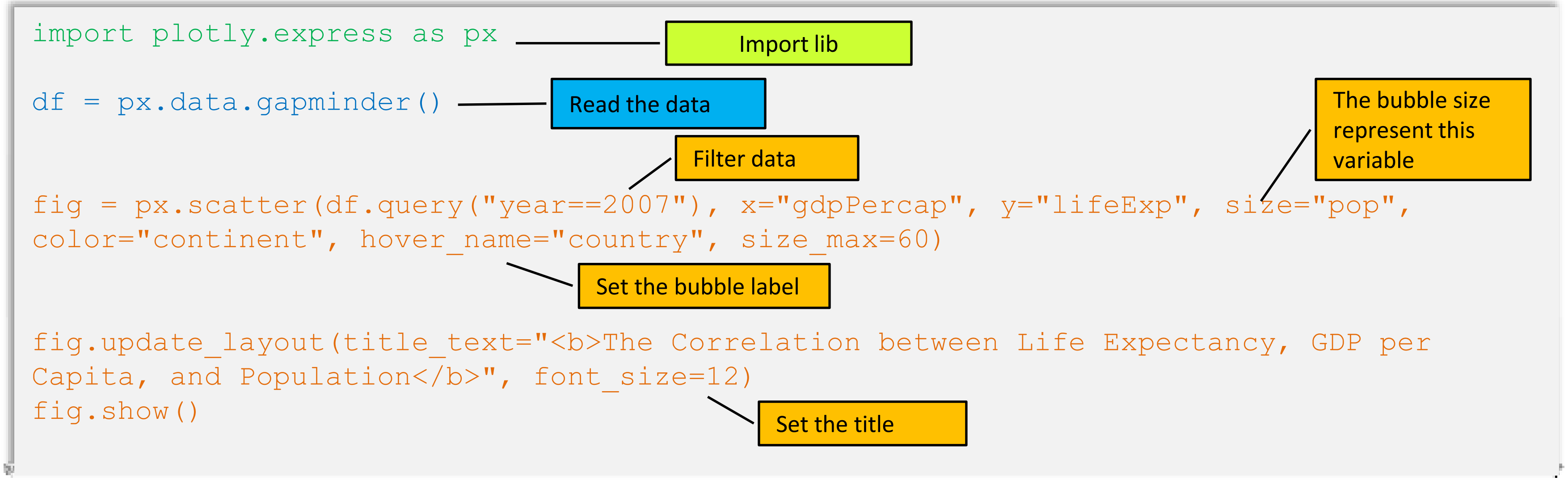
Data :

The *data* for population and *life expectancy* for countries from *Gapminder*

Source :

Gapminder dataset from plotly library

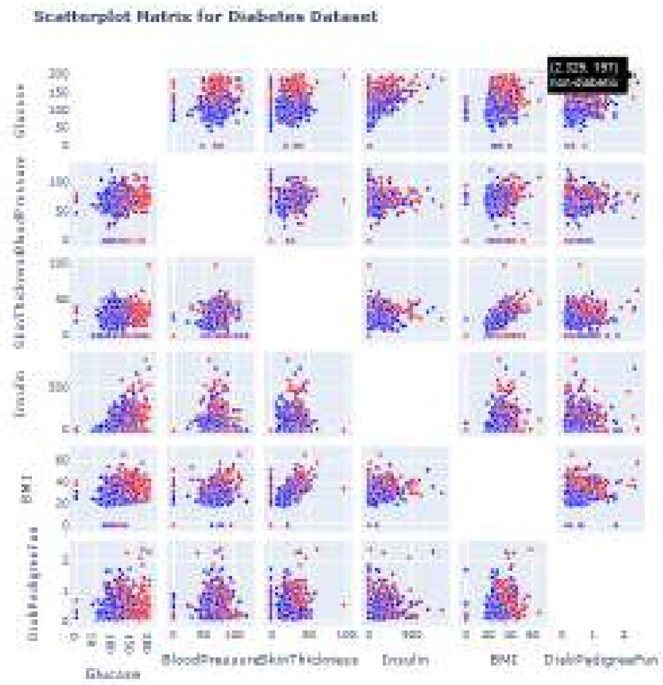
Source Code for Correlation – Bubble Plot



```
import plotly.express as px

df = px.data.gapminder()


fig = px.scatter(df.query("year==2007"), x="gdpPercap", y="lifeExp", size="pop",
color="continent", hover_name="country", size_max=60)


fig.update_layout(title_text="<b>The Correlation between Life Expectancy, GDP per
Capita, and Population</b>", font_size=12)
fig.show()
```

Note:
A bubble plot extends the potential of a scatter plot through multiple encoding of the data mark(Andy Kirk, 2012).

Correlation – Scatter Matrix Plot



Data variables :

6 x quantitative (glucose, bloodPressure, SkinThixckness, Insulin, BMI, DiabPedigreeFun)

1 x categorical (outcome)

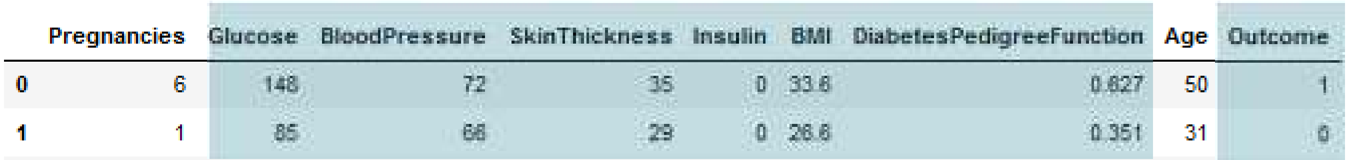
Visual Variables:

Position, color-hue

Data :

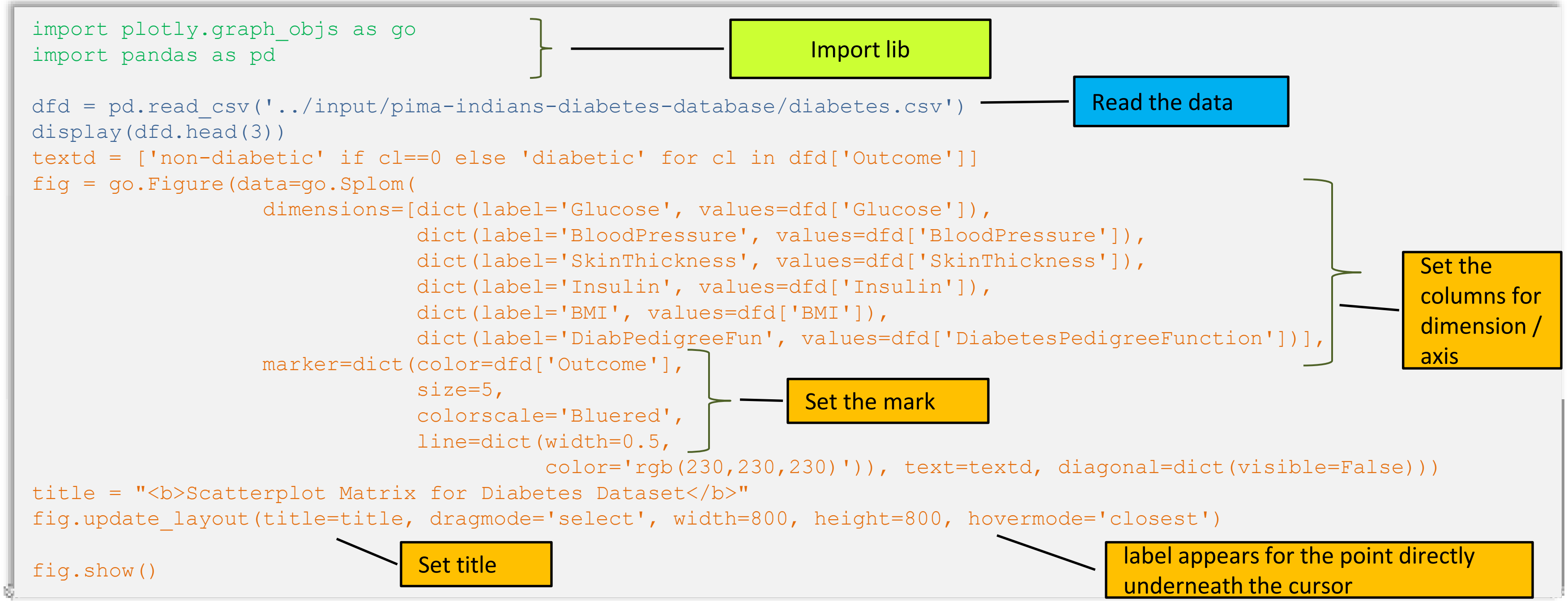
Pima Indian Diabetes Dataset

Source :
https://www.kaggle.com/uciml/pima-indians-diabetes-database



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0

Source Code for Correlation – Scatter Matrix Plot



```
import plotly.graph_objs as go
import pandas as pd

dfd = pd.read_csv('../input/pima-indians-diabetes-database/diabetes.csv')
display(dfd.head(3))
textd = ['non-diabetic' if cl==0 else 'diabetic' for cl in dfd['Outcome']]
fig = go.Figure(data=go.Splom(
                  dimensions=[dict(label='Glucose', values=dfd['Glucose']),
                              dict(label='BloodPressure', values=dfd['BloodPressure']),
                              dict(label='SkinThickness', values=dfd['SkinThickness']),
                              dict(label='Insulin', values=dfd['Insulin']),
                              dict(label='BMI', values=dfd['BMI']),
                              dict(label='DiabPedigreeFun', values=dfd['DiabetesPedigreeFunction'])],
                  marker=dict(color=dfd['Outcome'],
                              size=5,
                              colorscale='Bluered',
                              line=dict(width=0.5,
                                        color='rgb(230,230,230)')), text=textd, diagonal=dict(visible=False)))
title = "<b>Scatterplot Matrix for Diabetes Dataset</b>"
fig.update_layout(title=title, dragmode='select', width=800, height=800, hovermode='closest')

fig.show()
```

Import lib
Read the data
Set the columns for dimension / axis
Set the mark
Set title
label appears for the point directly underneath the cursor

Note:
Similar to the small multiples chart that we saw earlier, a scatter plot matrix takes advantage of the eye's rapid capability to spot patterns across multiple views of the same type of chart (Andy Kirk, 2012).