# Lab 02 Welcome to Plotly

*COMP7507 Visualization and Visual Analytics*

*June 21, 2024*

## Goal

The goal of this lab session is to get familiar with Plotly.py and to write a simple Plotly.py program for basic charts.

## Brief Introduction to Plotly

The Plotly Python Library (Plotly.py) is an interactive, open-source plotting library that built on top of the Plotly JavaScript library (Plotly.js). It enables Python users to create over 40 unique chart types that can be displayed in Jupyter notebooks, saved to standalone HTML files, or served as part of pure Python-built web applications using Dash. Plotly also provides great support for non-web contexts including desktop editors (e.g. Spyder, PyCharm) and static document publishing (e.g. exporting notebooks to PDF with high-quality vector images).

## Getting Started

If you have finished the environment setting up steps, you are ready to create your first data visualization with Plotly.

Download the Plotly folder of Lab 2 from moodle, then open your **Anaconda Prompt** (Windows) or **Terminal** (MacOS), activate the environment by typing:

```
conda activate Plotly
```

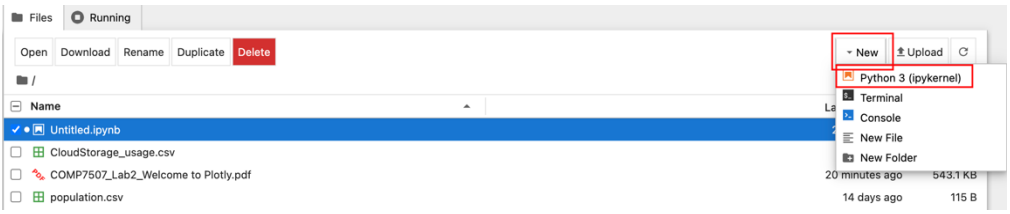
Navigate to the Plotly folder, then open **Jupyter Notebook**:

```
jupyter notebook
```

## Draw a Bar Chart by Plotly.Express

By learning how to draw a bar chart, we will know Plotly Express. Plotly Express is the easy-to-use, high-level interface to Plotly, which operates on a variety of types of data and produces easy-to-style figures.

Open a blank Python3 notebook by clicking the “New” button.




In the first cell, enter the following code to import required packages. In this step, we import the plotly.express module as px.

```
[1]: import pandas as pd
     import plotly
     import plotly.express as px
```

Hit Shift+Enter key, run the first cell and insert the next cell. Enter the following code to read the dataset you uploaded.

```
[2]: data = pd.read_csv("population.csv")
     print(data)
```

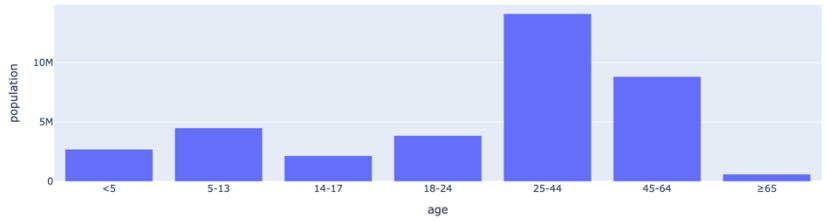
Print the data and you should see something like this:

```
      age  population
0      <5     2704659
1    5-13     4499890
2   14-17     2159981
3   18-24     3853788
4   25-44    14106543
5   45-64     8819342
6     ≥65      612463
```

To draw a very basic bar chart for your data, use the px.bar() function to link the dataset to the figure and set the x and y axis. Then we use the show() function to display the bar chart on your browser.

```
[3]: fig = px.bar(data, x = 'age', y = 'population')
     fig.show()
```

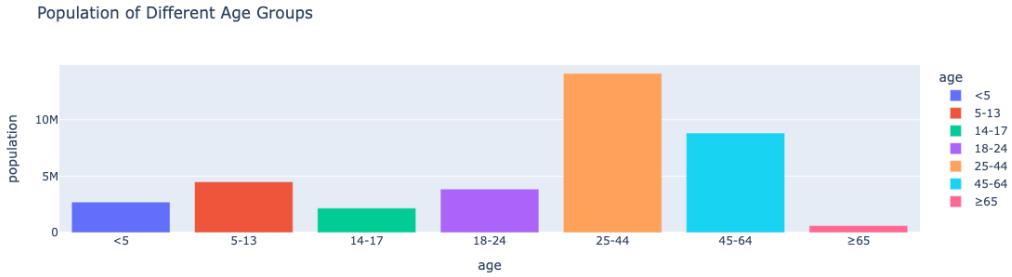
Then you will get a basic bar chart like this:




Now we can try to do some modification. Similarly, we draw the bar chart by the px.bar() function. But this time, we firstly set the color of each bar changed by the “age” column, and we set the graph’s title.

```
[4]: fig = px.bar(data, x = 'age', y = 'population', color = 'age', title = 'Population of Different Age Groups')
     fig.show()
```

And you will get a different bar chart like this:




To summarize, The plotly.express module contains functions that can create entire figures at once. All modifications on the figure (e.g. color, title) can be done by a single function-call. As the high-level interface to Plotly, plotly.express helps us to draw fully-populated figures in an easy and fast way.

## Draw a Pie Chart by Plotly.Graph_Object

At this part, we will not only draw a pie chart, but also learn some advanced concepts in Plotly, including the plotly.graph_object module, the update_layout() function and the update_trace() function.

To import required packages, this time we use plotly.graph_objects (go) module rather than plotly.express module. Read the uploaded dataset and print it out:

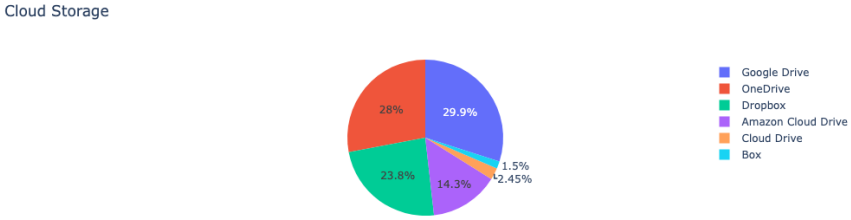
[5]:
```
import plotly.graph_objects as go
pie_data = pd.read_csv('cloudStorage_usage.csv')
print(pie_data)
```

```
     CloudStorageName  numOfpeople
0        Google Drive       120000
1             Dropbox        95400
2            OneDrive       112130
3                 Box         6000
4  Amazon Cloud Drive        57323
5         Cloud Drive         9833
```

Now we draw a basic pie chart by passing a graph object trace (an instance of go.Pie) to the data argument of the go.Figure() constructor, and set the title of the figure by passing a dictionary to the layout argument.

[6]:
```
fig = go.Figure(data = [go.Pie(labels = pie_data.CloudStorageName,
                               values = pie_data.numOfpeople)],
                layout = dict(title = dict (text = 'Cloud Storage')))
fig.show()
```

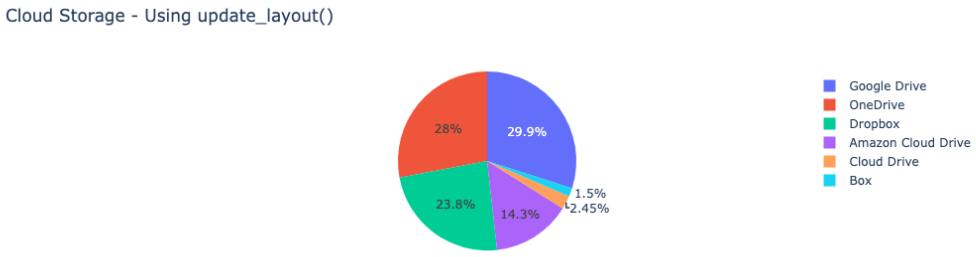
Using the show() function, you will see a result like this:



The update_layout() method may be used to update multiple nested properties of a figure's layout. Now we use the update_layout() function to change the title's text and font size.

```
[7]: fig.update_layout(title_text = 'Cloud Storage - Using update_layout()')
     fig.show()
```

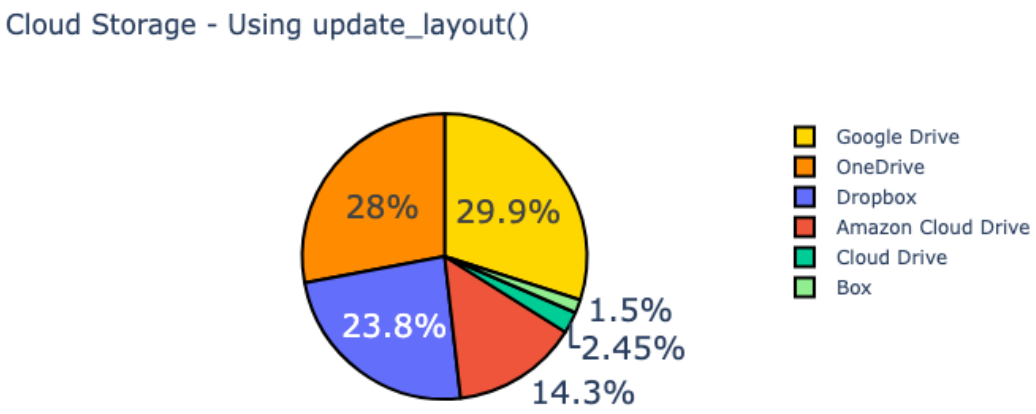
As a result, the title is changed:




Graph object figures support an update_traces() method that may be used to update multiple nested properties of one or more of a figure's traces. Here we update the font size of text inside the pie chart and set the color to a predefined color set. We also set the color of the outline to be black and the width to be 2.

```
[8]: my_cols = ['gold', 'mediumquoise', 'darkorange', 'lightgreen']
     fig.update_traces(textfont_size = 20,
                       marker = dict(colors = my_cols, line = dict(color = '#000', width = 2)))
     fig.show()
```

Result is shown as below:




## Export Your Plots

1. Export static images

Plotly allows you to save static images of your plots in many formats like PNG, JPEG,

SVG or PDF. To do this, enter the following code after the cell where you draw the graph.

```
[9]: import os
     if not os.path.exists('PATH_TO_OUTPUT'):
         os.mkdir('PATH_TO_OUTPUT')
     fig.write_image('PATH_TO_OUTPUT/FILE_NAME.png')
```

2. Export interactive HTML

Any figure can be saved an HTML file using the `write_html` method. These HTML files can be opened in any web browser to access the fully interactive figure. Similar to the way you export images, enter the following code at the end of your notebook.

```
[10]: fig.write_html('PATH_TO_OUTPUT/FILE_NAME.html')
```

## Troubleshooting

1. **Where can I find my exported HTML and PNG files?**

   If you never change the default directory of Jupyter before, you should be able to find your files at "/Users/username" (Mac OS) or "C:\Users\username" (Windows).

2. **(Mac OS X) After installation, typing `jupyter notebook` into Terminal returns `jupyter: Command not found`.**
   a. Typing `echo $PATH` into Terminal, check if `path/to/anaconda/bin` (this path depends on where you install the anaconda) is on the list
   b. If the path is not on the list, typing `sudo nano /etc/paths`
   c. Enter your login password when prompted
   d. Add the path to the bottom of the file
   e. Hit control-x to quit
   f. Hit Y to save, then hit Enter to leave
   g. Type `jupyter notebook` to check again

3. **What is the main difference between the plotly.express (px) module and the plotly.graph_objects (go) module?**
   In fact, every figure produced with the plotly library uses graph objects under the hood, unless manually constructed out of dictionaries. Thus the recommended way to create figures is using the functions in the plotly.express (px) module, because the figures produced by px in a single function-call are easy to customize at creation-time, and to manipulate after creation using the update_* and add_* methods. But getting familiar with the plotly.graph_object module will provide us more possibility and space on creating complex graphs and accurately describing datasets.

## Task for You

This week's task is to submit the exported images of bar chart and pie chart to Moodle before **July 5, 2024**.

You are free to change styles for the figures, but remember to add your student number

at the end of each figure's title. Examples are shown below: