**Week 1 - Lab Exercise: Setting up Python Environment and Creating and Evaluating Data Visualisations with Python**

**Setting Up Your Jupyter Environment for Data Visualization (COMP4010)**

Follow these steps to set up your Jupyter environment for the Data Visualization labs. This guide ensures you have everything ready to create and analyze visualizations using Python.

**Step 1: Install Python 3.13.2**

1. Download Python 3.13.2 from the official website: <https://www.python.org/downloads/>.
2. During installation, **check the box** that says **"Add Python to PATH"**.
3. Complete the installation.

**Step 2: Set Up a Virtual Environment**

A virtual environment keeps your project dependencies separate from other Python projects.

1. **Open your terminal/command prompt.**
2. **Create a project folder** to store your lab work:

bash

mkdir data\_visualisation\_lab

cd data\_visualisation\_lab

1. **Create a virtual environment:**

bash

python -m venv venv

1. **Activate the virtual environment:**
   * **On Windows:**

bash

venv\Scripts\activate

* + **On macOS/Linux:**

bash

source venv/bin/activate

**Step 3: Install Jupyter and Required Libraries**

1. **Upgrade pip:**

bash

python -m pip install --upgrade pip

1. **Install Jupyter Notebook:**

bash

pip install notebook

1. **Install Libraries for Data Visualisation:**

bash

pip install pandas matplotlib seaborn

**Step 4: Launch Jupyter Notebook**

1. **Start Jupyter Notebook:**

bash

jupyter notebook

1. A browser window will open. Navigate to your data\_visualisation\_lab folder.
2. **Create a new notebook:** Click on **"New"** > **"Python 3"** to start coding.

**Step 5: Verify Jupyter is Using the Correct Environment**

To ensure Jupyter is running from your virtual environment:

1. **Run this code in a new notebook cell:**

python

import sys

print(sys.executable)

1. **Check the output:** It should show a path ending with /venv/bin/python (macOS/Linux) or \venv\Scripts\python.exe (Windows).

If the path is correct, your setup is complete!

**Using Jupyter in Future Sessions**

You don’t need to reinstall everything each time. Follow these steps to continue working each week:

1. **Open Terminal/Command Prompt.**
2. **Navigate to your project folder:**

bash

cd path\_to\_project/data\_visualisation\_lab

1. **Activate the virtual environment:**
   * **On Windows:**

bash

venv\Scripts\activate

* + **On macOS/Linux:**

bash

source venv/bin/activate

1. **Launch Jupyter Notebook:**

bash

jupyter notebook

**Troubleshooting Tips**

* **If Jupyter doesn't open in the browser:**
  1. Look for a link in the terminal that looks like this:

ruby

http://localhost:8888/?token=some\_long\_token\_here

* 1. **Copy and paste** this link into your browser manually.
* **If packages are missing:** Ensure the virtual environment is activated before running Jupyter. If needed, reinstall missing packages using pip install package\_name.

**Lab Exercise 1: Creating and Evaluating Data Visualisations with Python**

**Objective:**

By the end of this lab, students will:

1. Create basic visualisations (bar charts, line charts, scatter plots) using Python libraries.
2. Recognise and correct common visualisation mistakes (e.g., truncated axes, overloaded charts).
3. Compare good and bad visualisations and explain their impact on data interpretation.

**Part 1: Setting Up**

1. **Import Required Libraries:**

python

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

1. **Load Dataset:** We'll use a simple dataset, like a CSV of monthly sales or temperature data.

python

# Example dataset

data = {

'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun'],

'Sales': [200, 220, 250, 270, 300, 320],

'Temperature': [30, 32, 35, 40, 42, 45]

}

df = pd.DataFrame(data)

df

**Part 2: Creating Basic Charts**

1. **Bar Chart: Monthly Sales**

python

plt.figure(figsize=(8, 5))

sns.barplot(x='Month', y='Sales', data=df)

plt.title('Monthly Sales')

plt.show()

1. **Line Chart: Temperature Over Time**

python

plt.figure(figsize=(8, 5))

sns.lineplot(x='Month', y='Temperature', data=df, marker='o')

plt.title('Monthly Temperature')

plt.show()

1. **Scatter Plot: Sales vs. Temperature**

python

plt.figure(figsize=(8, 5))

sns.scatterplot(x='Temperature', y='Sales', data=df)

plt.title('Sales vs. Temperature')

plt.show()

**Part 3: Identifying and Correcting Bad Visualisations**

1. **Truncated Y-Axis Example:**

python

plt.figure(figsize=(8, 5))

sns.barplot(x='Month', y='Sales', data=df)

plt.ylim(200, 350) # Truncated Y-axis

plt.title('Truncated Axis: Misleading Sales Growth')

plt.show()

**Task:** Identify how the truncated Y-axis exaggerates differences. Correct it by resetting the Y-axis to start at 0.

1. **Overloaded Chart Example:**

python

plt.figure(figsize=(8, 5))

sns.lineplot(x='Month', y='Sales', data=df, label='Sales', marker='o')

sns.lineplot(x='Month', y='Temperature', data=df, label='Temperature', marker='s')

plt.title('Overloaded Chart: Too Much Information')

plt.legend()

plt.show()

**Task:** Explain why this chart might be confusing. Suggest improvements, such as using separate charts or focusing on one variable.

1. **Misleading Pie Chart Example:**

python

pie\_data = [30, 40, 35]

labels = ['Product A', 'Product B', 'Product C']

plt.figure(figsize=(6, 6))

plt.pie(pie\_data, labels=labels, explode=(0, 0.1, 0), autopct='%1.1f%%', startangle=90)

plt.title('Misleading Pie Chart: Exaggerated Differences')

plt.show()

**Task:** Identify issues like the use of 3D effects, slice exaggeration, or unclear labels. Propose a clearer alternative.

**Lab exercise 2: Advanced Visualisation Tasks**

**Task 1: Adding Details and Annotations to Charts**

1. **Enhance the Bar Chart with Customisation:**

python

plt.figure(figsize=(8, 5))

sns.barplot(x='Month', y='Sales', data=df, palette='Blues\_d')

# Adding annotations

for index, row in df.iterrows():

plt.text(index, row['Sales'] + 5, row['Sales'], ha='center')

plt.title('Monthly Sales with Annotations')

plt.xlabel('Month')

plt.ylabel('Sales')

plt.grid(axis='y', linestyle='--')

plt.show()

1. **Customise the Line Chart with Styles:**

python

plt.figure(figsize=(8, 5))

sns.lineplot(x='Month', y='Temperature', data=df, marker='o', linestyle='--', color='red')

# Highlight the highest temperature

max\_temp = df['Temperature'].max()

max\_month = df[df['Temperature'] == max\_temp]['Month'].values[0]

plt.annotate(f'Highest Temp: {max\_temp}', xy=(max\_month, max\_temp), xytext=(max\_month, max\_temp + 2),

arrowprops=dict(facecolor='black', shrink=0.05))

plt.title('Monthly Temperature with Highlights')

plt.xlabel('Month')

plt.ylabel('Temperature (C)')

plt.show()

**Task 2: Combining Multiple Charts**

1. **Create Subplots to Compare Sales and Temperature:**

python

fig, ax = plt.subplots(2, 1, figsize=(10, 8))

sns.barplot(x='Month', y='Sales', data=df, ax=ax[0], palette='coolwarm')

ax[0].set\_title('Monthly Sales')

sns.lineplot(x='Month', y='Temperature', data=df, marker='o', ax=ax[1], color='green')

ax[1].set\_title('Monthly Temperature')

plt.tight\_layout()

plt.show()

1. **Overlay Line and Bar Charts:**

python

fig, ax1 = plt.subplots(figsize=(8, 5))

# Bar chart for Sales

sns.barplot(x='Month', y='Sales', data=df, ax=ax1, alpha=0.6, color='skyblue')

ax1.set\_ylabel('Sales')

# Line chart for Temperature

ax2 = ax1.twinx()

sns.lineplot(x='Month', y='Temperature', data=df, ax=ax2, marker='o', color='red')

ax2.set\_ylabel('Temperature (C)')

plt.title('Sales and Temperature Comparison')

plt.show()

**Part 4: Working with a Larger Dataset**

We'll now use a real-world dataset to create more complex visualisations.

**Dataset: COVID-19 Global Data**

1. **Download the dataset directly in Jupyter Notebook:**

python

url = 'https://covid.ourworldindata.org/data/owid-covid-data.csv'

covid\_df = pd.read\_csv(url)

covid\_df.head()

1. **Filter Data for a Specific Country (e.g., Vietnam):**

python

vietnam\_df = covid\_df[covid\_df['location'] == 'Vietnam']

vietnam\_df = vietnam\_df[['date', 'new\_cases', 'new\_deaths']].dropna()

vietnam\_df['date'] = pd.to\_datetime(vietnam\_df['date'])

vietnam\_df.head()

1. **Visualise New COVID-19 Cases Over Time:**

python

plt.figure(figsize=(12, 6))

sns.lineplot(data=vietnam\_df, x='date', y='new\_cases', color='blue')

plt.title('Daily New COVID-19 Cases in Vietnam')

plt.xlabel('Date')

plt.ylabel('New Cases')

plt.show()

1. **Compare New Cases and New Deaths with Dual Axes:**

python

fig, ax1 = plt.subplots(figsize=(12, 6))

sns.lineplot(data=vietnam\_df, x='date', y='new\_cases', ax=ax1, color='blue', label='New Cases')

ax1.set\_ylabel('New Cases', color='blue')

ax2 = ax1.twinx()

sns.lineplot(data=vietnam\_df, x='date', y='new\_deaths', ax=ax2, color='red', label='New Deaths')

ax2.set\_ylabel('New Deaths', color='red')

plt.title('COVID-19 New Cases and Deaths in Vietnam')

fig.tight\_layout()

plt.show()

**Part 5: Reflection Questions**

1. **Customisation Impact:**  
   How did adding annotations and custom styles improve the clarity of your visualisations?
2. **Combining Charts:**  
   What are the benefits and potential pitfalls of combining multiple data visualisations into one chart?
3. **Real-World Dataset Challenges:**  
   What difficulties did you face when working with the COVID-19 dataset? How did you address them?

✅ **Lab complete! If you haven’t changed the code by trying different examples, please do so. If you have done that, now move to the Lab Assignment.**