VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

Assignment-2

Data Visualization

Submitted By

NAMANU -1RN21CD029

Under the Guidance of

Ms Vinutha S

Asst. Professor

Department of CSE (Data Science)

RN SHETTY TRUST®

RNS INSTITUTE OF TECHNOLOGY

Autonomous Institution Affiliated to VTU, Recognized by GOK, Approved by AICTE
(NAAC 'A+ Grade' Accredited, NBA Accredited (UG - CSE, ECE, ISE, EIE and EEE)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098
Ph:(080)28611880,28611881 URL: www.rnsit.ac.in

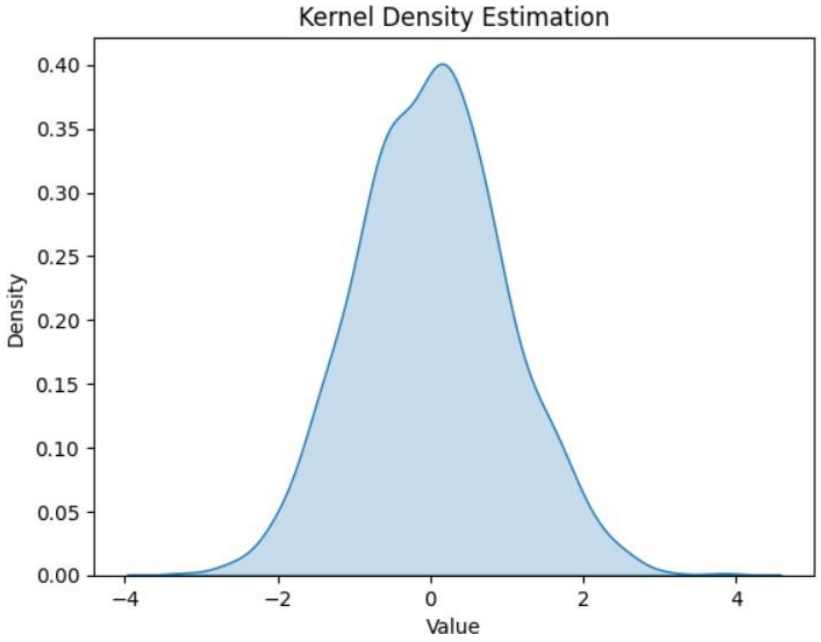
2024-2025

Question 1: Demonstrate Kernel Density Estimation

Code Snippet:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Generate some random data
np.random.seed(42)
data = np.random.randn(1000)

# Create a KDE plot using seaborn
sns.kdeplot(data, shade=True)

# Customize the plot
plt.title("Kernel Density Estimation")
plt.xlabel("Value")
plt.ylabel("Density")

plt.show()
```

Output:

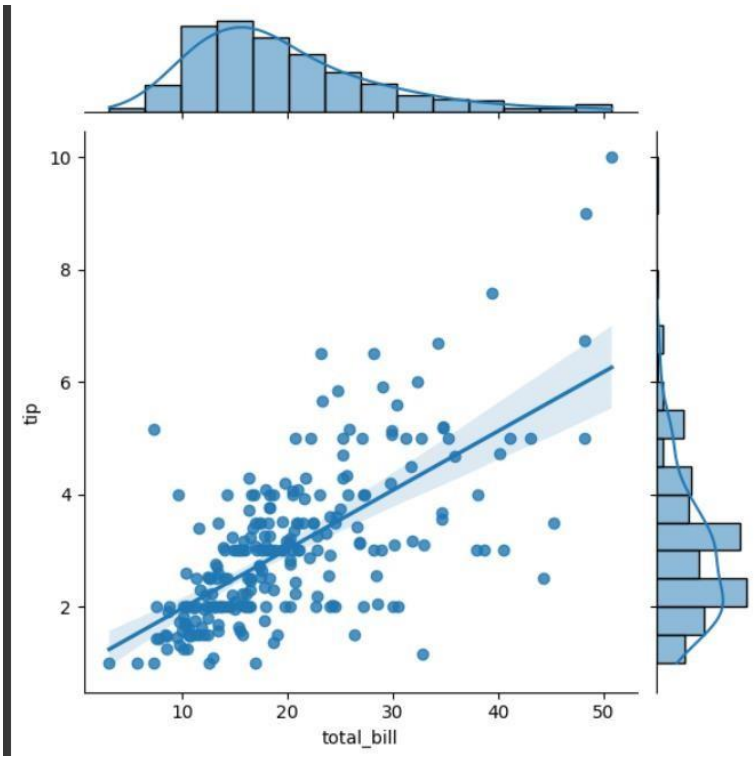


Question 2: Plot bivariate distribution

Code Snippet:

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

# Load the tips dataset
tips = sns.load_dataset("tips")

# Create a jointplot to visualize the relationship between total_bill and tip
sns.jointplot(x="total_bill", y="tip", data=tips, kind="reg")

plt.show()
```

Output :

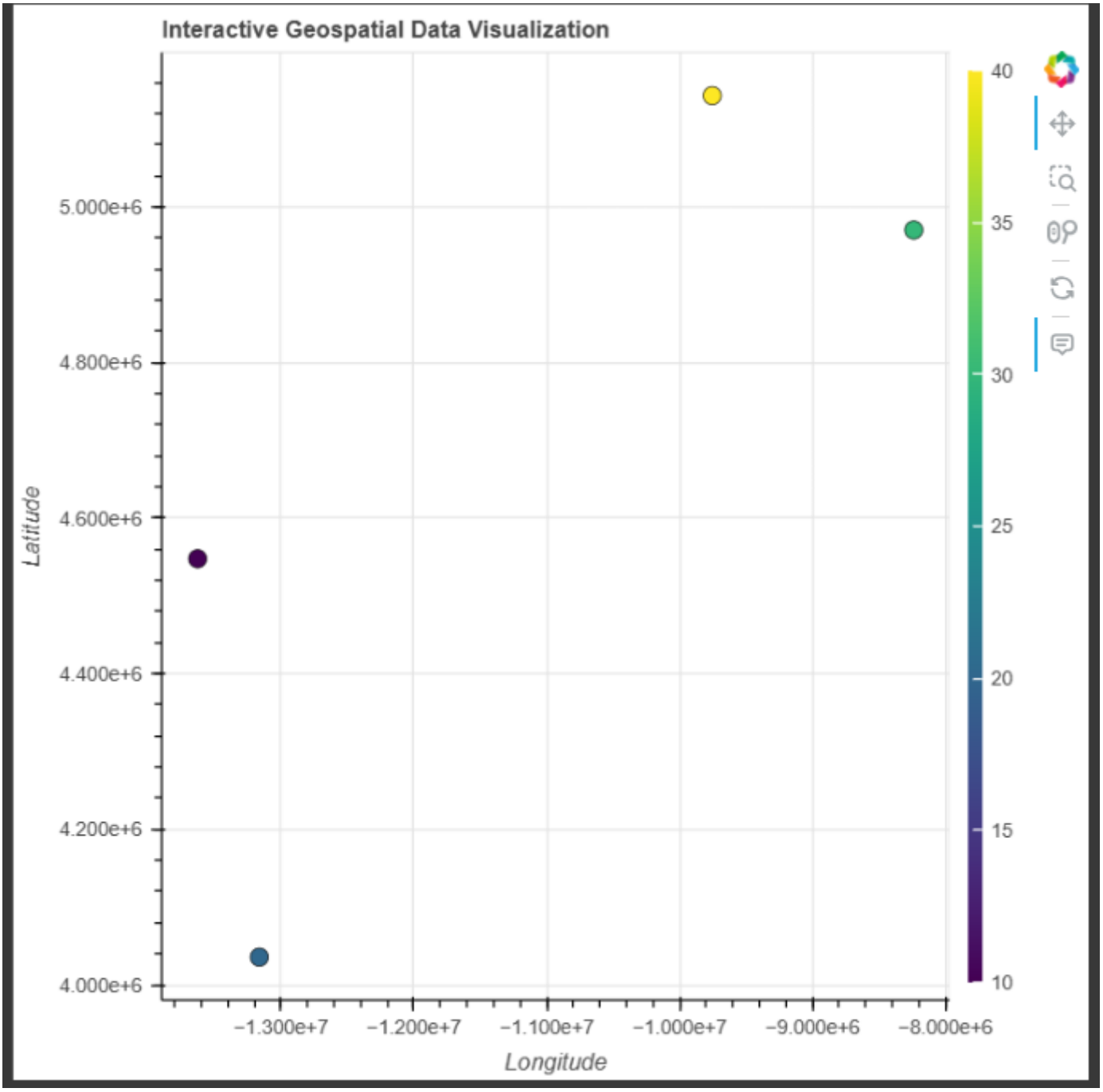


Question 3: Geospatial data and Bokeh

Code Snippet:

```
import pandas as pd
from bokeh.io import output_notebook, show
from bokeh.models import ColumnDataSource, HoverTool, ColorBar
from bokeh.plotting import figure
from bokeh.transform import linear_cmap
from bokeh.palettes import Viridis256
import numpy as np

# Enable inline plotting in Jupyter Notebook
output_notebook()

# Sample data: Create a simple DataFrame with geographical coordinates and values
data = {
    'name': ['Location A', 'Location B', 'Location C', 'Location D'],
    'latitude': [37.7749, 34.0522, 40.7128, 41.8781],
    'longitude': [-122.4194, -118.2437, -74.0060, -87.6298],
    'value': [10, 20, 30, 40]  # Example values for color mapping
}

# Create a DataFrame
df = pd.DataFrame(data)

# Convert latitude and longitude to Web Mercator coordinates
df['x'] = df['longitude'] * 20037508.34 / 180
df['y'] = np.log(np.tan((90 + df['latitude']) * np.pi / 360)) / (np.pi / 180)
df['y'] = df['y'] * 20037508.34 / 180

# Create a ColumnDataSource
source = ColumnDataSource(df)

# Create a figure
p = figure(title="Interactive Geospatial Data Visualization",
           tools="pan,wheel_zoom,box_zoom,reset,hover",
           x_axis_label='Longitude', y_axis_label='Latitude')

# Define color mapping
mapper = linear_cmap(field_name='value', palette=Viridis256, low=df['value'].min(), high=df['value'].max())

# Add scatter points to the figure
p.scatter(x='x', y='y', size=10, source=source, fill_color=mapper, line_color='black', line_width=0.5)

# Add hover tool
hover = p.select(dict(type=HoverTool))
hover.tooltips = [("Name", "@name"), ("Value", "@value")]

# Add color bar
color_bar = ColorBar(color_mapper=mapper['transform'], width=8, location=(0,0))
p.add_layout(color_bar, 'right')

# Show the plot
show(p)
```

Output:

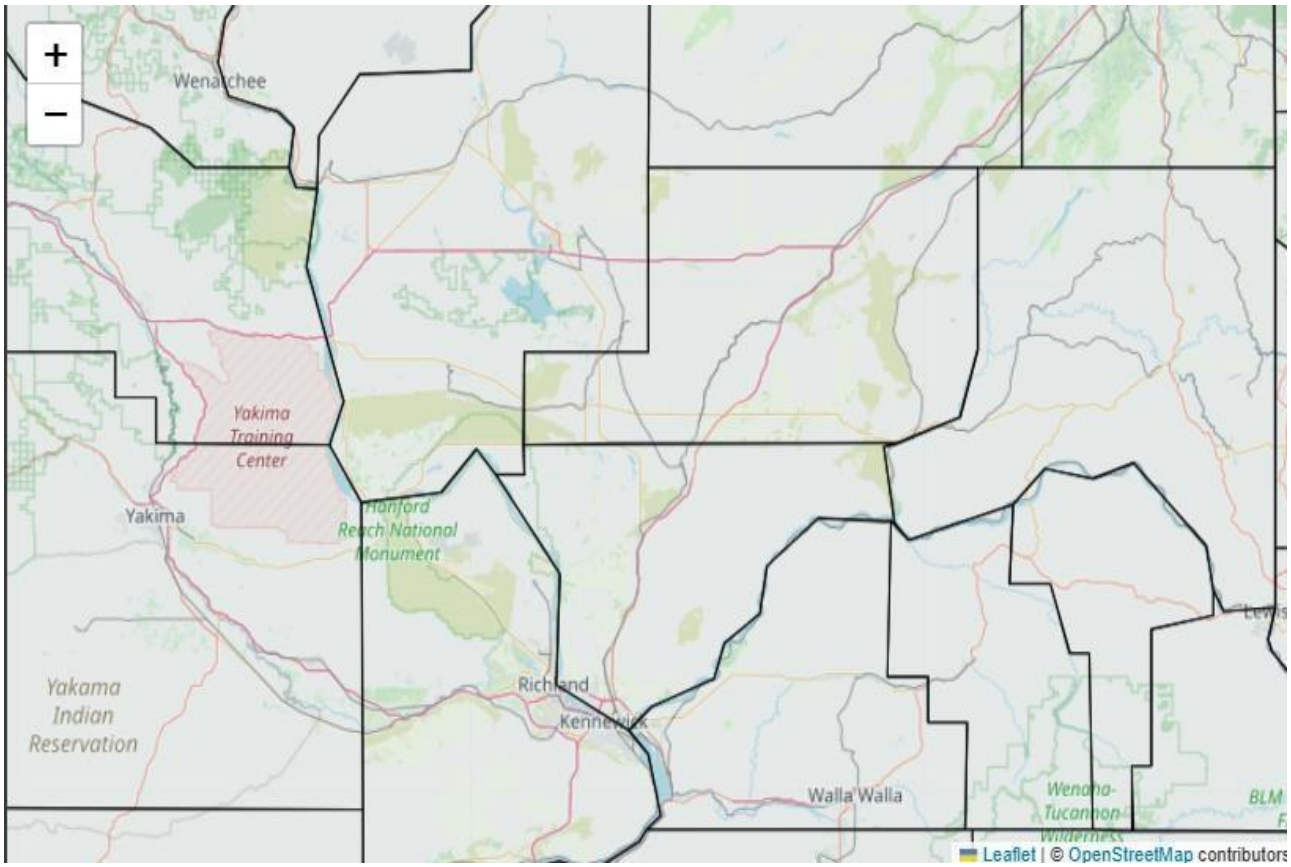

Interactive Geospatial Data Visualization
5.000e+6
4.800e+6
4.600e+6
4.400e+6
4.200e+6
4.000e+6
Latitude
−1.300e+7
−1.200e+7
−1.100e+7
−1.000e+7
−9.000e+6
−8.000e+6
Longitude
40
35
30
25
20
15
10

Question 4 : Plot network and interconnection using geospatial data

Code Snippet:

```
import geopandas as gpd
import folium

# Load the GeoJSON data for US States
gdf = gpd.read_file("https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json")

# Create a Folium map
m = folium.Map(location=[37.0902, -95.7129], zoom_start=5)

# Add the GeoJSON layer to the map
folium.GeoJson(
    gdf,
    style_function=lambda x: {'fillColor': 'lightblue', 'color': 'black', 'weight': 1}
).add_to(m)

import geopandas as gpd
import folium

# Load the GeoJSON data for US States
gdf = gpd.read_file("https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json")

# Create a Folium map
m = folium.Map(location=[37.0902, -95.7129], zoom_start=5)

# Add the GeoJSON layer to the map
folium.GeoJson(
    gdf,
    style_function=lambda x: {'fillColor': 'lightblue', 'color': 'black', 'weight': 1}
).add_to(m)

# Sample network data
network_data = [
    [(-95, 37), (-90, 40)],
    [(-90, 40), (-85, 45)],
    [(-95, 37), (-100, 35)]
]

# Add network lines to the map
for line in network_data:
    folium.PolyLine(line, color='red', weight=2).add_to(m)


network_data = [
    [(-95, 37), (-90, 40), {'color': 'red', 'weight': 3}],
    [(-90, 40), (-85, 45), {'color': 'blue', 'weight': 1}],
    [(-95, 37), (-100, 35), {'color': 'green', 'weight': 2}]
]

for line in network_data:
    folium.PolyLine(line[:2], color=line[2]['color'], weight=line[2]['weight']).add_to(m)

# Display the map
m
```

Output:


+
−
Wenatchee
Yakima Training Center
Yakima
Hanford Reach National Monument
Yakama Indian Reservation
Richland
Kennewick
Walla Walla
Wenaha-Tucannon Wilderness
Leaflet | © OpenStreetMap contributors

Question 5: Retrieving image over HTTP, parsing HTML and scraping web

```
import requests
from bs4 import BeautifulSoup
import urllib.request

def download_image(url, filename):
    response = requests.get(url, stream=True)
    with open("images.jpg", 'wb') as out_file:
        for chunk in response.iter_content(1024):
            if not chunk:
                break
            out_file.write(chunk)

def parse_html(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')
    # Find all image tags with 'src' attribute
    images = soup.find_all('img', src=True)
    for image in images:
        image_url = image['src']
        # Download the image
        download_image(image_url, f"image_{image_url.split('/')[-1]}")

def scrape_webpage(url):
    response = requests.get(url)
    html_content = response.text
    soup = BeautifulSoup(html_content, 'html.parser')

    # Find the desired text (adjust the selector as needed)
    text_elements = soup.find_all('p')  # Find all paragraph elements

    for text in text_elements:
        print(text.text.strip())

# Example usage:
url = "https://www.nasa.gov/"  # Replace with the desired URL
scrape_webpage(url)
```

Output:

```
Digital content creators are invited to register to attend the launch of NASA’s SpaceX Crew-10 mission that will carry astronau
ts to the International Space Station.
NASA Accelerates Space Exploration, Earth Science for All in 2024
NASA’s Hubble Celebrates Decade of Tracking Outer Planets
NASA Researchers Discover More Dark Comets
Hubble Spots a Spiral in the Celestial River
NASA Invites Media to Panama, Austria Artemis Accords Signings
NASA to Test Technology for X-59’s Unique Shock Wave Measurements
NASA’s PACE, US-European SWOT Satellites Offer Combined Look at Ocean
What’s Up: December 2024 Skywatching Tips from NASA
The Artemis II test flight will be NASA’s first mission with a crew under the Artemis campaign and will pave the way to land as
tronauts on the Moon on Artemis III and future missions.
For more than 50 years, NASA satellites have provided data on Earth's land, water, air, temperature, and climate. NASA's Earth
Information Center allows visitors to see how our planet is changing in six key areas: sea level rise and coastal impacts, heal
th and air quality, wildfires, greenhouse gases, sustainable energy, and agriculture.
Mary W. Jackson Portrait Revealed
A portrait of Mary W. Jackson is seen after it was unveiled, Friday, Dec. 6, 2024, at the NASA Headquarters Mary W. Jackson Bui
lding in Washington. Mary W. Jackson was a pioneering aerospace engineer and mathematician at NASA’s Langley Research Center.
Stay up-to-date on the latest news from NASA–from Earth to the Moon, the Solar System and beyond.
We will never share your email address.
Privacy Policy
NASA explores the unknown in air and space, innovates for the benefit of humanity, and inspires the world through discovery.
```

Question 6: Web services including eXtensible Markup Language

```python
import xml.etree.ElementTree as ET

# Simple XML-based "database" of book information
book_database = {
    "1": {"title": "Harry Potter", "author": "J.K. Rowling", "year": "1997"},
    "2": {"title": "Lord of the Rings", "author": "J.R.R. Tolkien", "year": "1954"},
    "3": {"title": "The Hobbit", "author": "J.R.R. Tolkien", "year": "1937"}
}

class BookWebService:
    def get_book_by_id(self, book_id):
        """
        Retrieve book information and return as XML
        """
        book = book_database.get(book_id)

        if not book:
            # Create XML for error response
            root = ET.Element('book-response')
            error = ET.SubElement(root, 'error')
            error.text = f"No book found with ID {book_id}"
            return ET.tostring(root, encoding='unicode')

        # Create XML response
        root = ET.Element('book-response')
        book_elem = ET.SubElement(root, 'book')

        # Add book details to XML
        for key, value in book.items():
            elem = ET.SubElement(book_elem, key)
            elem.text = value

        # Convert XML to string
        return ET.tostring(root, encoding='unicode')

def main():
    # Create web service instance
    book_service = BookWebService()

    # Demonstrate XML service calls
    print("Book 1 XML Response:")
    print(book_service.get_book_by_id("1"))

    print("\nBook 2 XML Response (Non-existent):")
    print(book_service.get_book_by_id("2"))

if __name__ == "__main__":
    main()
```

Output :

```
Book 1 XML Response:
<book-response><book><title>Harry Potter</title><author>J.K. Rowling</author><year>1997</year></book></book-response>

Book 2 XML Response (Non-existent):
<book-response><book><title>Lord of the Rings</title><author>J.R.R. Tolkien</author><year>1954</year></book></book-response>
```

GitHub link : https://github.com/Namanu08/data-visualization