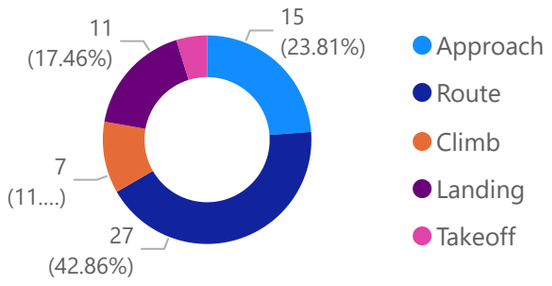
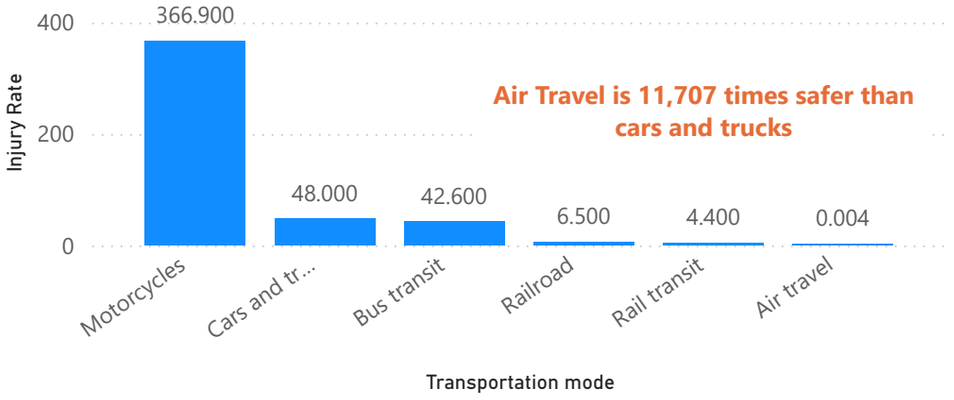
Overview of Airline and Motor Vehicle Fatalities: Trends, Comparison, and Distributions

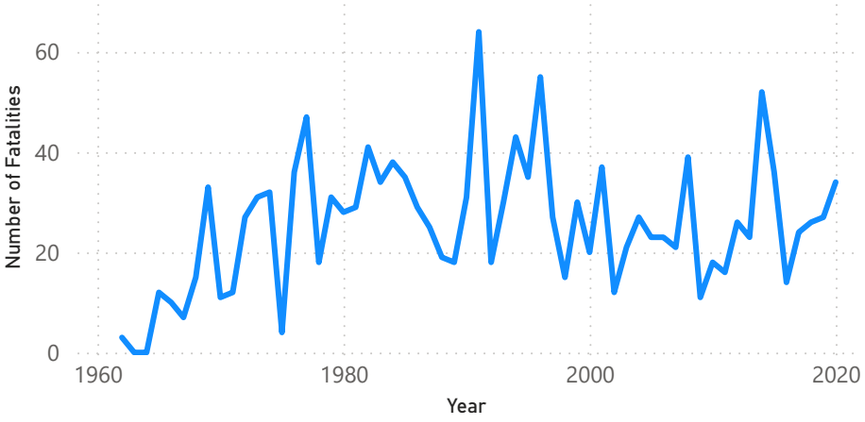
Distribution of Airline Accidents by Phase



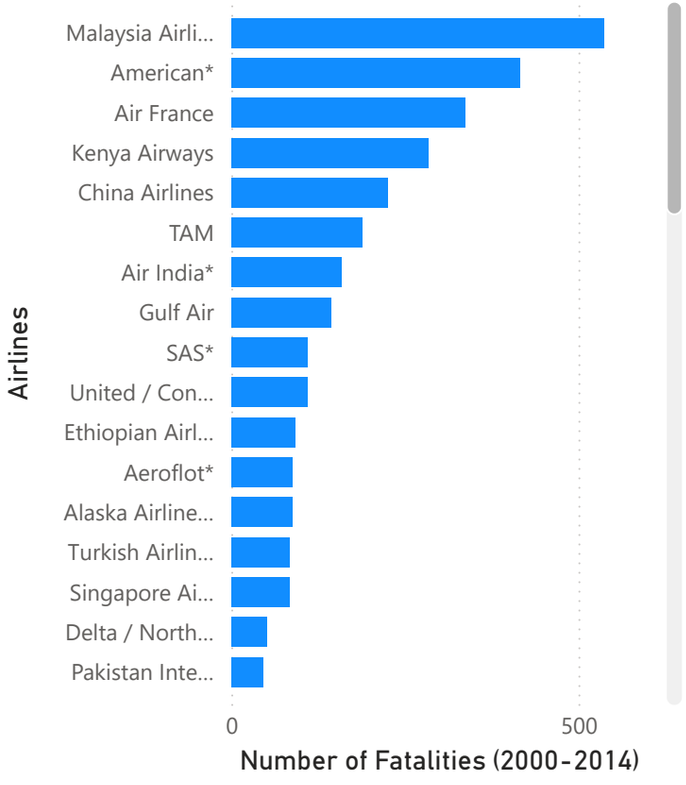
Injury Rate by Transportation mode



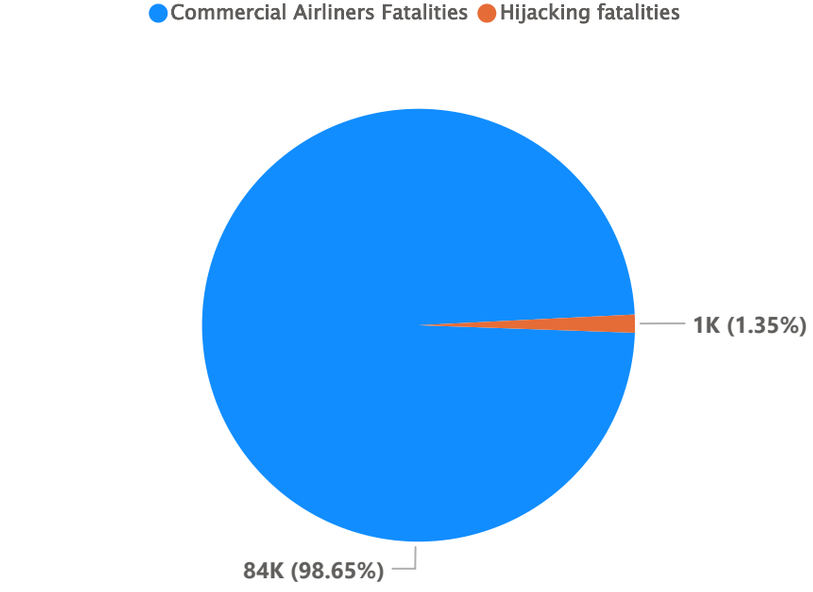
Historical Fatalities Trend (1960-2020)



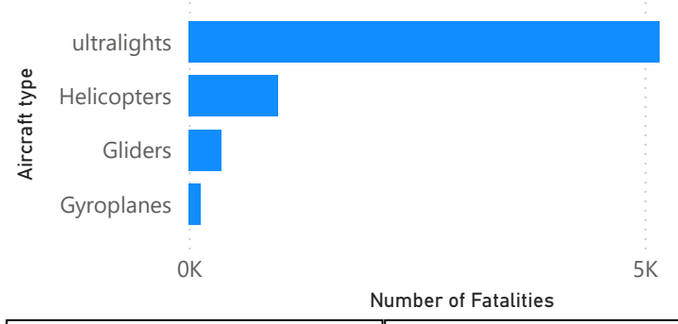
Airline Fatalities (2000-2014)



Commercial Airliners Fatalities and Hijacking fatalities



Trends in Aircraft Fatalities by Type (2011-2021)



1M

Total Motor Vehicle Death (Last 30 Years)

25K

Total Airplane Fatalities (Last 30 Years)

Said Moussadeq

Project Task 1

DSC640

Visualization Plan for Airline Safety Data Analysis

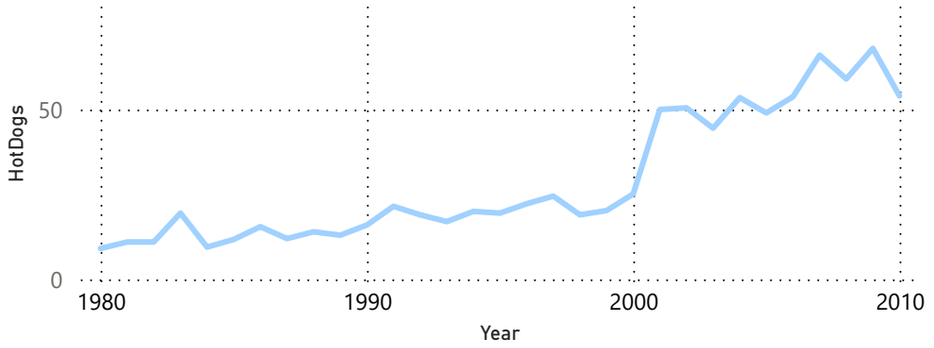
In analyzing our dataset, I chose visualizations that effectively convey trends and comparisons in airline safety data. A line chart displays the historical trend of fatalities from 1940-2020, highlighting fluctuations over time, despite increased population and air traffic. The donut chart for airline accident phases shows the breakdown of incidents by phase. A bar chart compares injury rates by transportation mode, illustrating the relative safety of air travel. Another bar chart compares airline fatalities from 2000-2014, highlighting differences among airlines. A pie chart illustrates that 1.35% of fatalities are due to hijacking. Finally, summary cards provide quick insights into motor vehicle deaths and airplane fatalities over 30 years.

These visualizations were chosen for their clarity in presenting data trends and comparisons. Line and bar charts effectively show changes over time and comparisons among categories, while pie charts are ideal for displaying part-to-whole relationships. Summary cards offer a quick, impactful overview of key statistics.

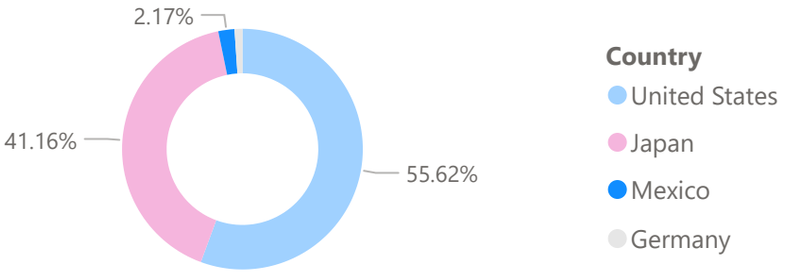
For presenting to the internal team, I plan to use Power BI's interactive features, allowing team members to explore the data themselves. Findings indicate that while air travel is significantly safer compared to other transportation modes, there are notable differences in safety records among airlines and flight phases, including incidents due to hijacking. All these factors should be considered to enhance passenger safety.

Working for an airline company, the biggest ethical challenge is to avoid bias and present data accurately and transparently. It's crucial to represent the data as it is, without manipulation, to maintain credibility and trust. Ensuring objectivity and acknowledging any data limitations to support informed decision-making is an integral part of professional integrity.

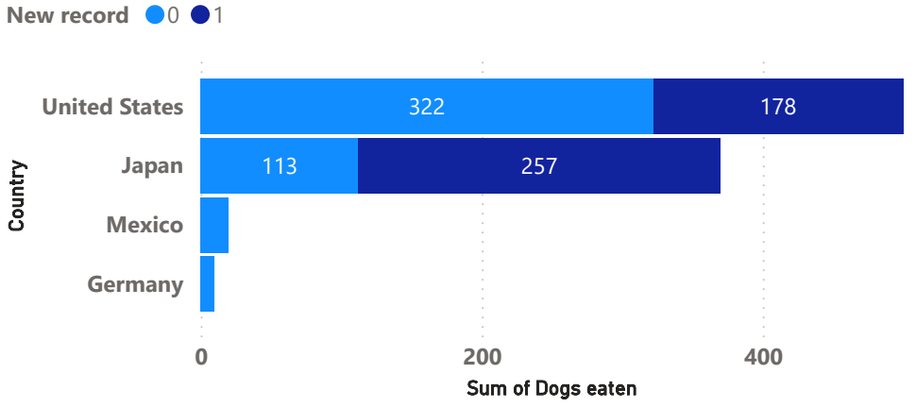
Line Chart: Number of Hotdogs Eaten Over the Years



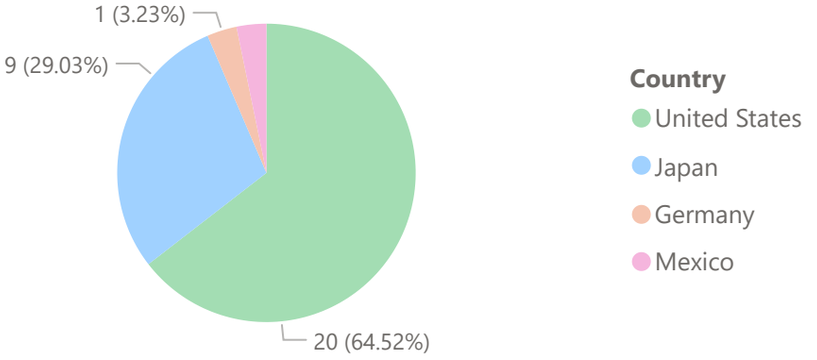
Donut Chart: HotDogs eaten by Country



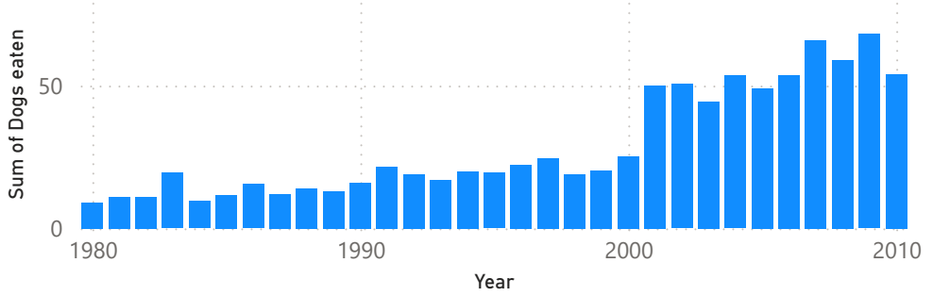
Stacked Bar Chart: HotDogs eaten by Country and New record



Pie Chart: Count of Winner by Country



Bar Chart: Number of HotDogs eaten Over the Year



PowerBi Charts

DSC640_R_Visualization

Said Moussadeq

2024-06-18

R Visualization

```
# Load the necessary libraries
library(readxl)
library(ggplot2)
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Obama dataframe
Obama_df <- read_excel("obama-approval-ratings.xls")
head(Obama_df)
```

```
## # A tibble: 6 x 4
##   Issue           Approve Disapprove  None
##   <chr>             <dbl>      <dbl> <dbl>
## 1 Race Relations       52         38    10
## 2 Education            49         40    11
## 3 Terrorism            48         45     7
## 4 Energy Policy        47         42    11
## 5 Foreign Affairs      44         48     8
## 6 Environment          43         51     6
```

```
## HotDog dataframe
HotDog_df <- read_excel("hotdog-contest-winners.xlsm")
head(HotDog_df)
```

```
## # A tibble: 6 x 5
##    Year Winner                       `Dogs eaten` Country       `New record`
##   <dbl> <chr>                               <dbl> <chr>                <dbl>
## 1  1980 Paul Siederman & Joe Baldini          9.1 United States            0
## 2  1981 Thomas DeBerry                         11   United States            0
## 3  1982 Steven Abrams                          11   United States            0
## 4  1983 Luis Llamas                            19.5 Mexico                   0
## 5  1984 Birgit Felden                          9.5 Germany                  0
```

```
## 6  1985 Oscar Rodriguez                      11.8 United States            0
```

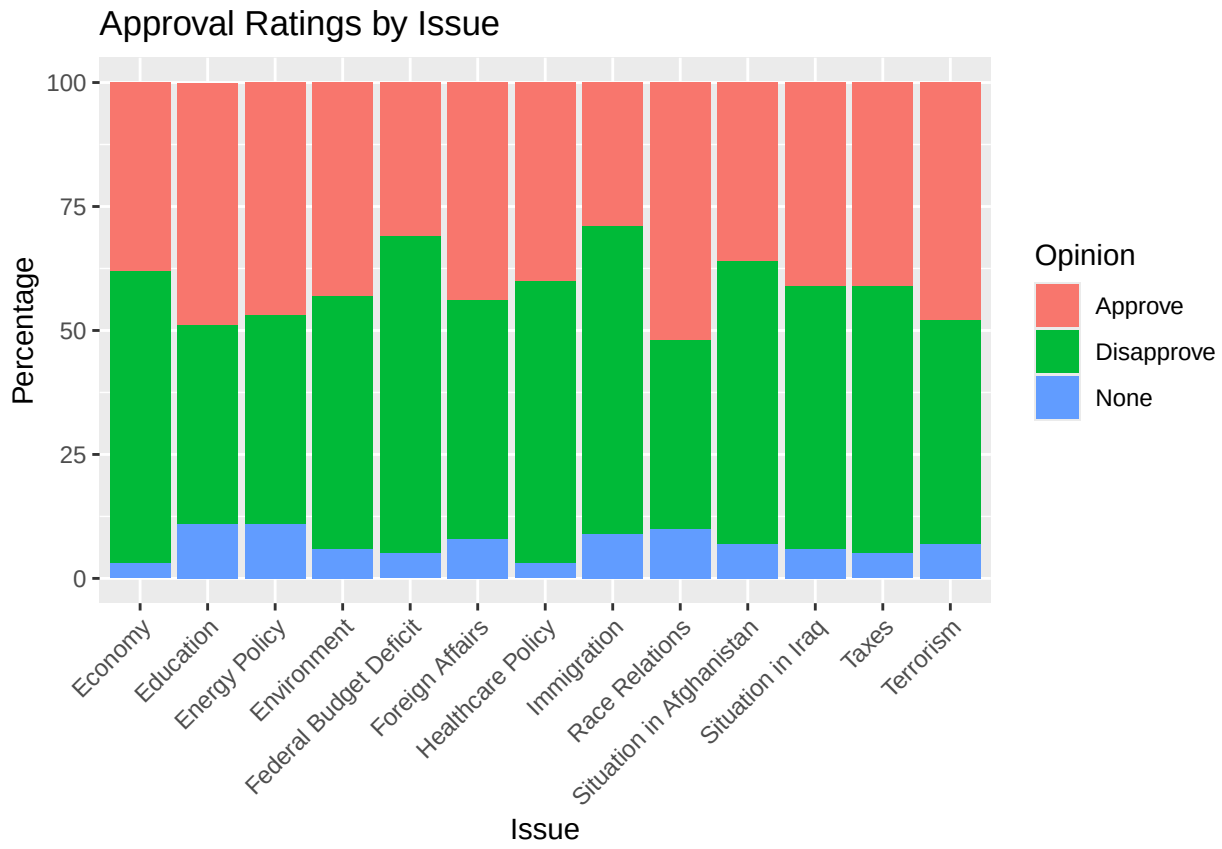
```
## HotDog dataframe

hotdog_places <- read_excel("hotdog-places.xlsm")
head(hotdog_places)
```

```
## # A tibble: 3 x 11
##   `2000` `2001` `2002` `2003` `2004` `2005` `2006` `2007` `2008` `2009` `2010`
##    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1     25   50     50.5   44.5   53.5     49     54     66     59   68       54
## 2     24   31     26     30.5   38       37     52     63     59   64.5     43
## 3     22   23.5   25.5   29.5   32       32     37     49     42   55       37
```

```
# Reshape the data to long format
Obama_df_long <- gather(Obama_df, key = "Opinion", value = "Percentage", -Issue)

# Display the reshaped data
head(Obama_df_long)
```

```
## # A tibble: 6 x 3
##   Issue           Opinion Percentage
##   <chr>           <chr>        <dbl>
## 1 Race Relations  Approve         52
## 2 Education       Approve         49
## 3 Terrorism       Approve         48
## 4 Energy Policy   Approve         47
## 5 Foreign Affairs Approve         44
## 6 Environment     Approve         43
```

```
# Create the stacked bar chart
ggplot(Obama_df_long, aes(x = Issue, y = Percentage, fill = Opinion)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Approval Ratings by Issue", x = "Issue", y = "Percentage", fill = "Opinion")
```




```
# Display the first few rows of the dataframe
head(Obama_df)
```

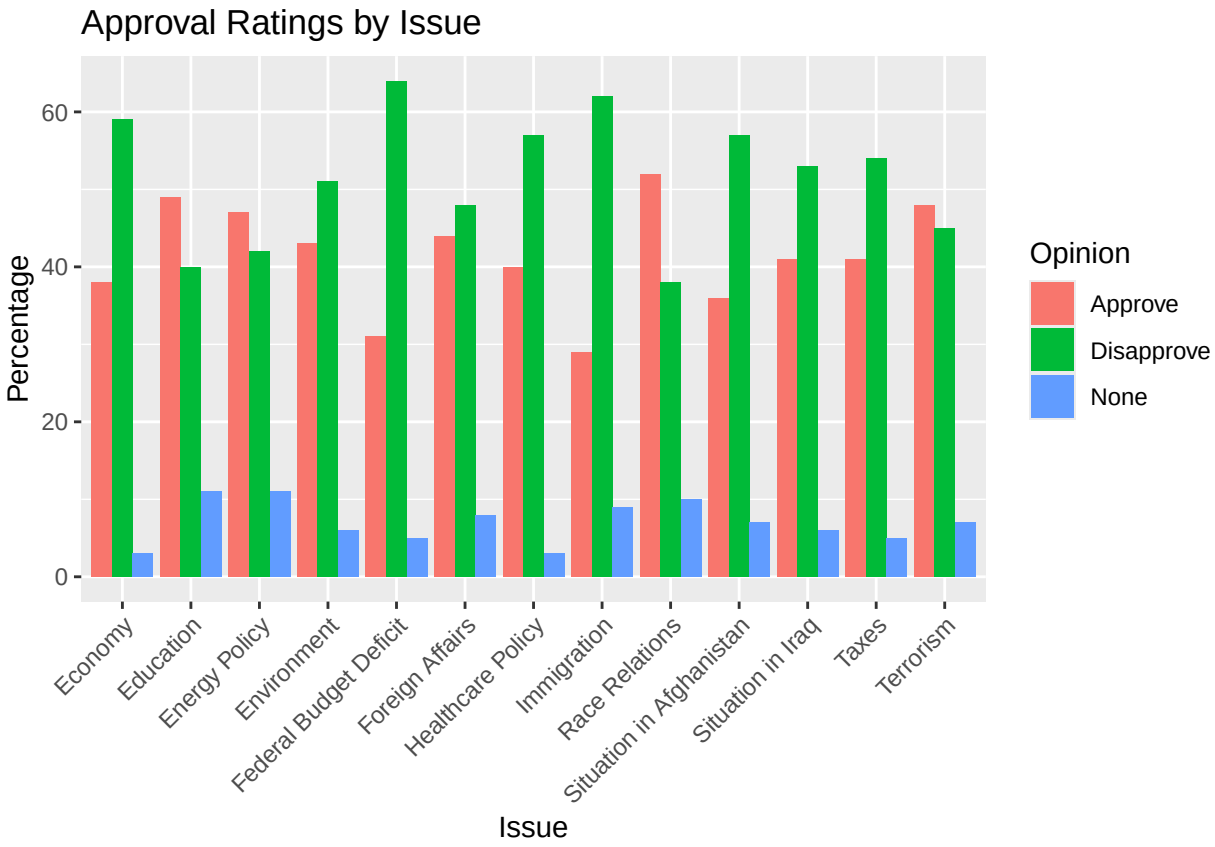
```
## # A tibble: 6 x 4
##   Issue           Approve Disapprove  None
##   <chr>             <dbl>      <dbl> <dbl>
## 1 Race Relations       52         38    10
## 2 Education            49         40    11
## 3 Terrorism            48         45     7
## 4 Energy Policy        47         42    11
## 5 Foreign Affairs      44         48     8
## 6 Environment          43         51     6
```

```
# Reshape the data to long format
Obama_df_long <- gather(Obama_df, key = "Opinion", value = "Percentage", -Issue)

# Display the reshaped data
head(Obama_df_long)
```

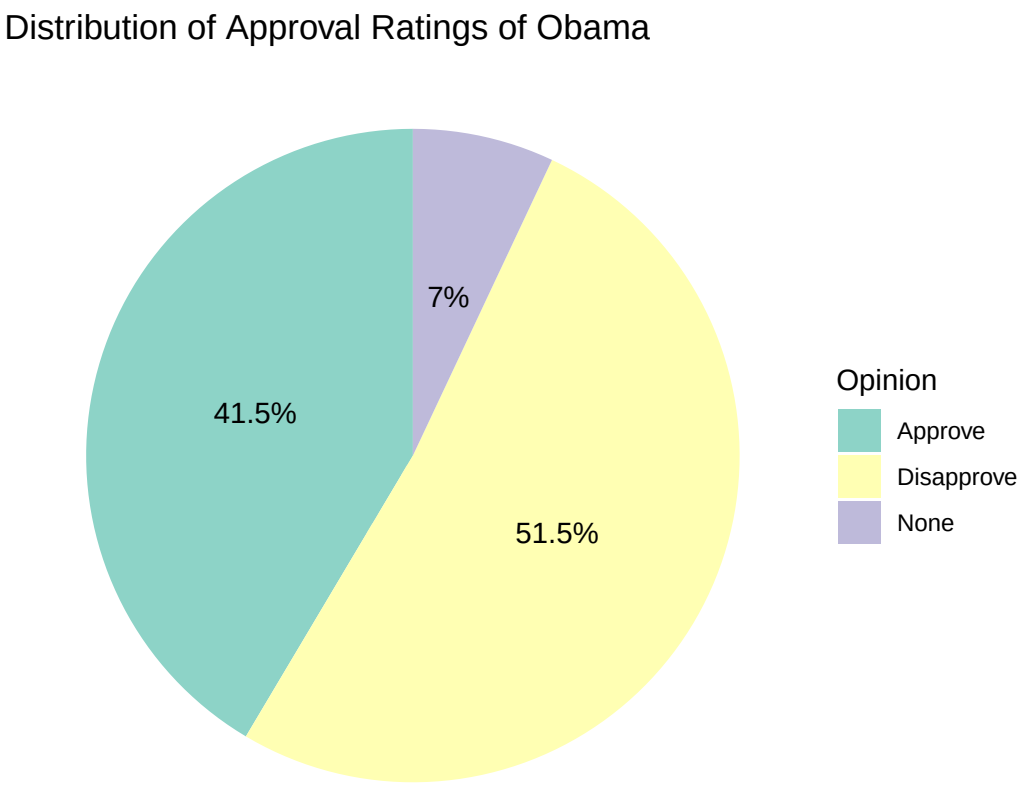
```
## # A tibble: 6 x 3
##   Issue           Opinion Percentage
##   <chr>           <chr>        <dbl>
## 1 Race Relations  Approve         52
## 2 Education       Approve         49
## 3 Terrorism       Approve         48
## 4 Energy Policy   Approve         47
## 5 Foreign Affairs Approve         44
```

```
## 6 Environment     Approve          43
```

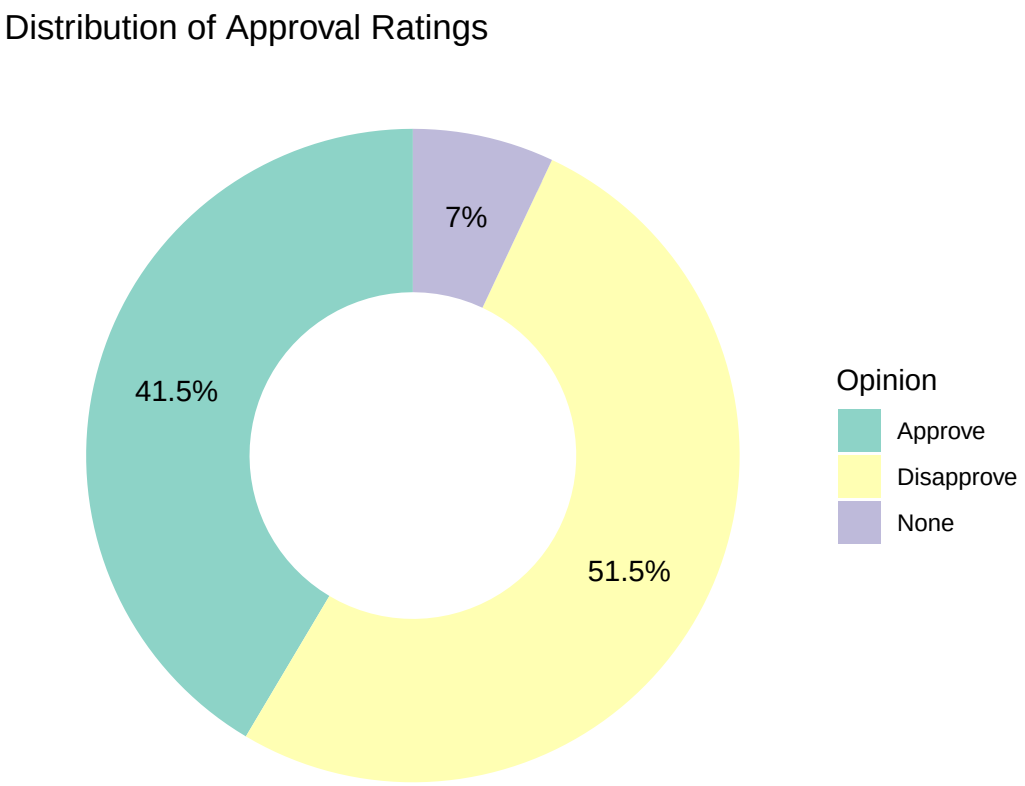
```
# Create the bar chart
ggplot(Obama_df_long, aes(x = Issue, y = Percentage, fill = Opinion)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Approval Ratings by Issue", x = "Issue", y = "Percentage")
```



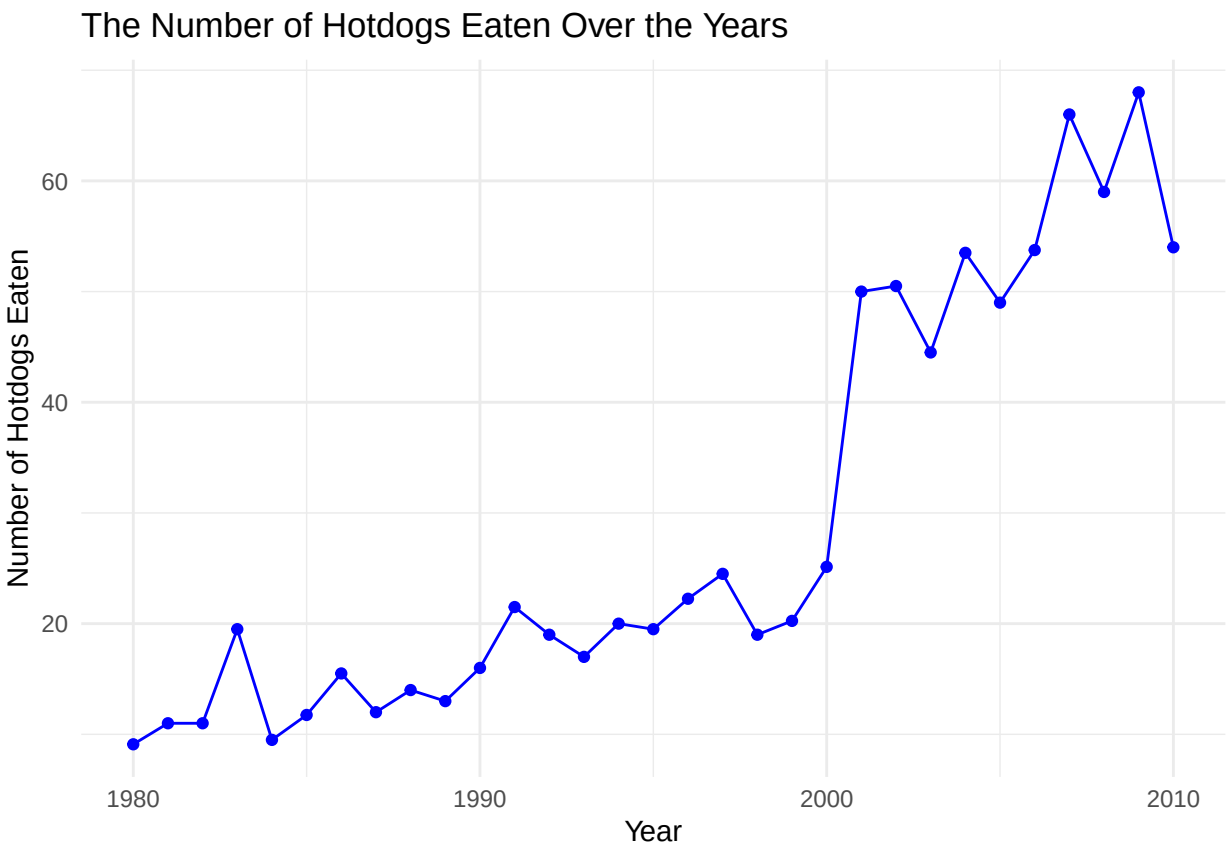

```
# Sum the totals for each opinion category to create a pie chart
data_totals <- Obama_df_long %>%
  group_by(Opinion) %>%
  summarise(Count = sum(Percentage))

# Create the pie chart
ggplot(data_totals, aes(x = "", y = Count, fill = Opinion)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar(theta = "y") +
  theme_void() +
  labs(title = "Distribution of Approval Ratings of Obama") +
  scale_fill_brewer(palette = "Set3") +
  geom_text(aes(label = paste0(round(Count / sum(Count) * 100, 1), "%")),
            position = position_stack(vjust = 0.5))
```




```
# Create the donut chart
ggplot(data_totals, aes(x = 2, y = Count, fill = Opinion)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  theme_void() +
  labs(title = "Distribution of Approval Ratings") +
  scale_fill_brewer(palette = "Set3") +
  geom_text(aes(label = paste0(round(Count / sum(Count) * 100, 1), "%")),
            position = position_stack(vjust = 0.5)) +
  xlim(0.5, 2.5)
```




```
# Create the line chart
ggplot(HotDog_df, aes(x = Year, y = `Dogs eaten`)) +
  geom_line(color = "blue") +
  geom_point(color = "blue") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "The Number of Hotdogs Eaten Over the Years", x = "Year", y = "Number of Hotdogs Eaten")
  theme_minimal()
```

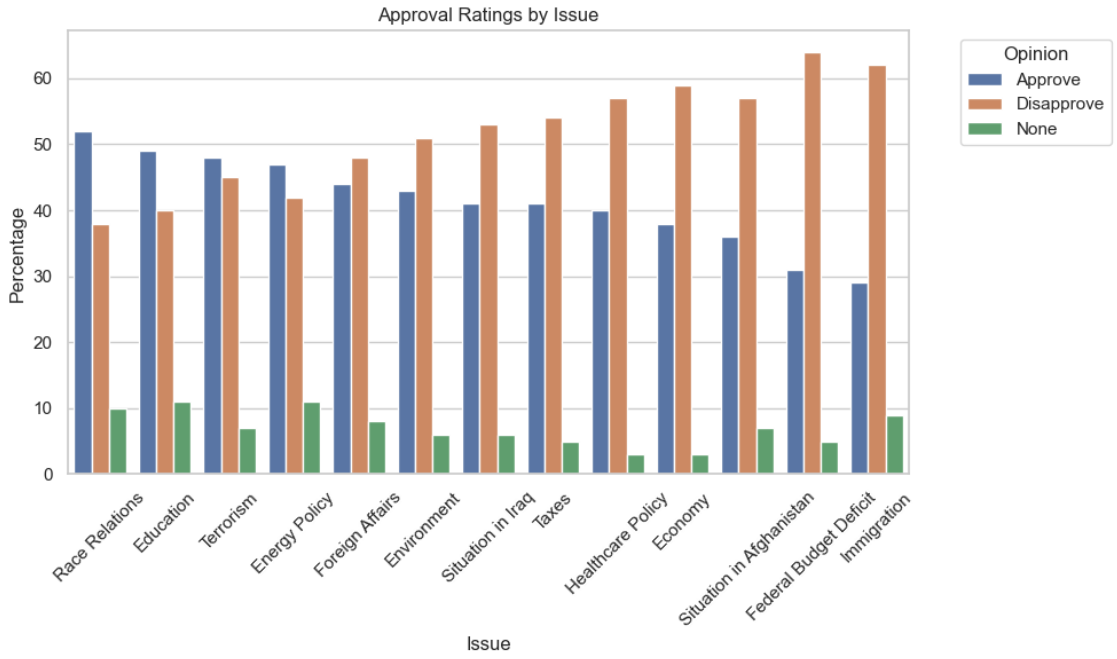

The Number of Hotdogs Eaten Over the Years
Number of Hotdogs Eaten
60
40
20
1980
1990
2000
2010
Year

Python Visualtizations

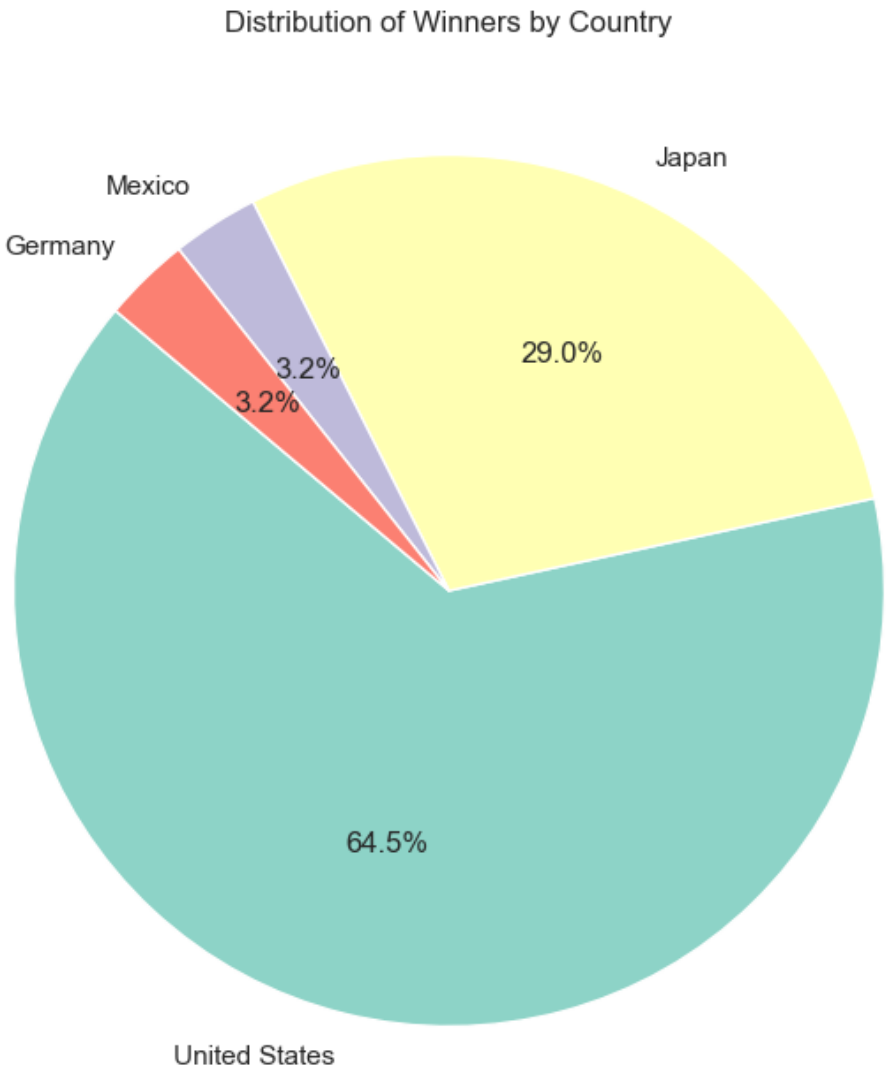
June 20, 2024

[50]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

# Load the hotdog contest data
hotdog_data = pd.read_excel('hotdog-contest-winners.xlsm')

# Load the Obama approval ratings data
obama_data = pd.read_excel('obama-approval-ratings.xls')

# Set up the plotting environment
sns.set(style="whitegrid")
```

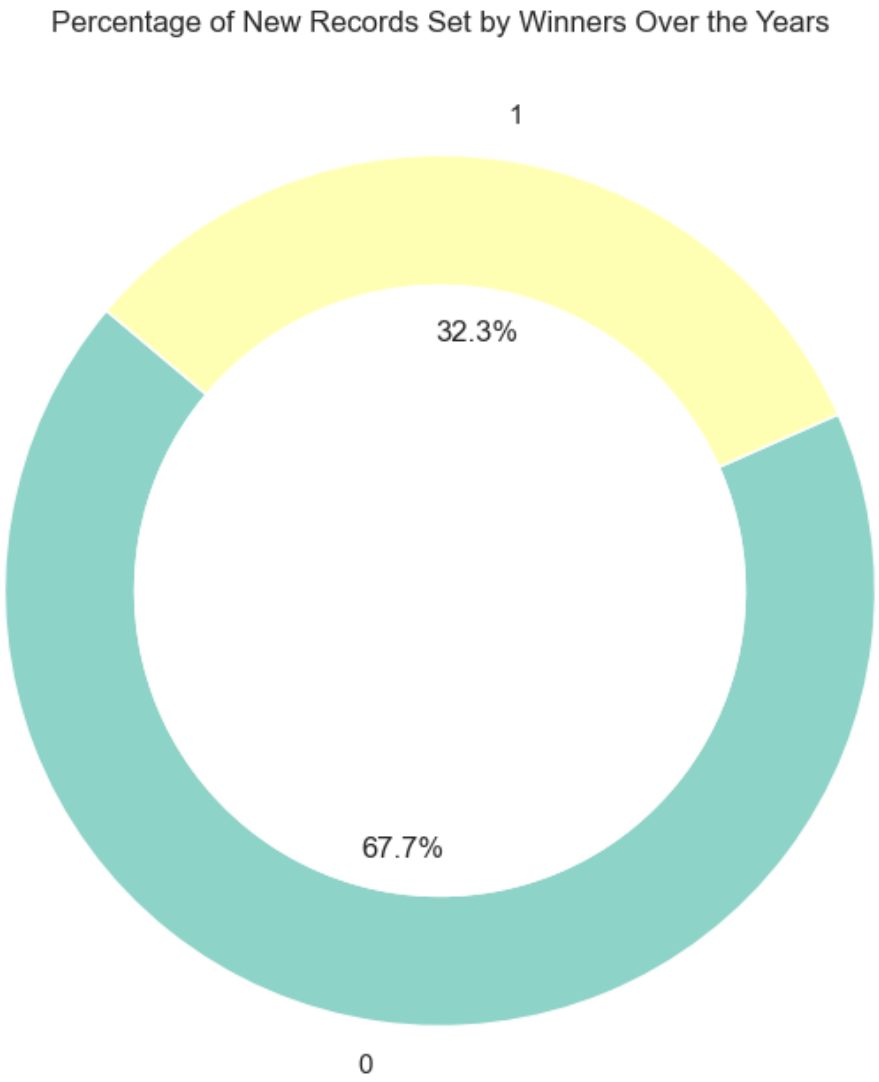
[51]:
```
# Create the stacked bar chart
plt.figure(figsize=(10, 6))
sns.barplot(data=obama_data_long, x='Issue', y='Percentage', hue='Opinion')
plt.title('Approval Ratings by Issue')
plt.xticks(rotation=45)
plt.xlabel('Issue')
plt.ylabel('Percentage')
plt.legend(title='Opinion', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



[52]:
```
# Pie Chart: Distribution of winners by country
winner_counts = hotdog_data['Country'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(winner_counts, labels=winner_counts.index, autopct='%1.1f%%',␣
 ↪startangle=140, colors=sns.color_palette("Set3"))
plt.title('Distribution of Winners by Country')
plt.show()
```

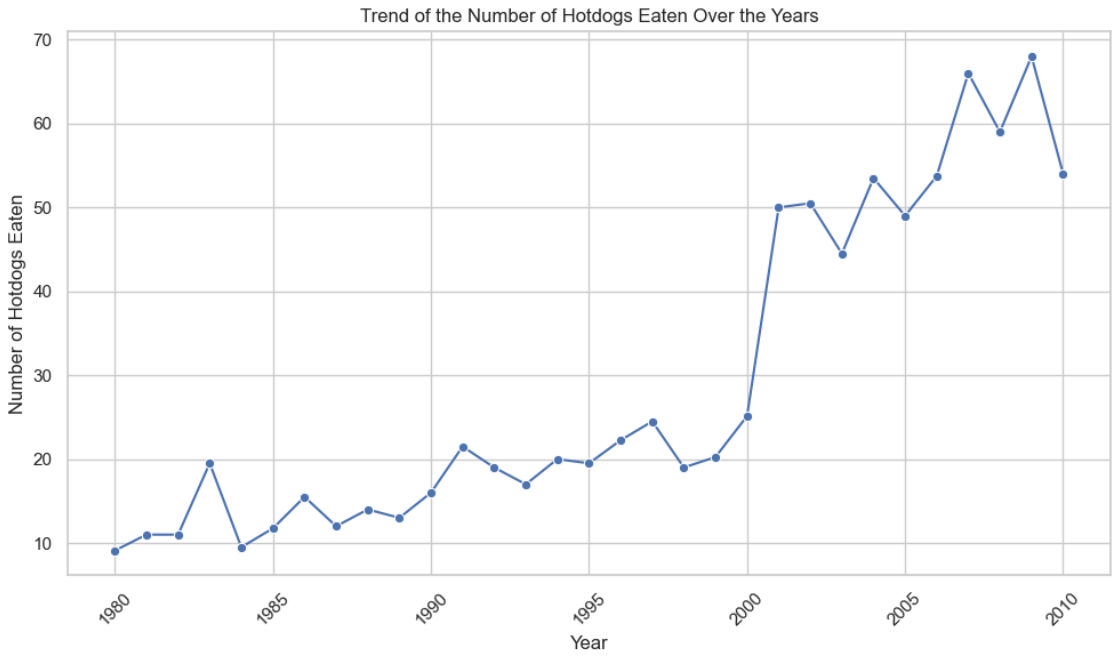


[53]:
```
# Donut Chart: Percentage of new records set by winners over the years
new_record_counts = hotdog_data['New record'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(new_record_counts, labels=new_record_counts.index, autopct='%1.1f%%',␣
  ↪startangle=140, colors=sns.color_palette("Set3"), wedgeprops=dict(width=0.3))
plt.title('Percentage of New Records Set by Winners Over the Years')
plt.gca().add_artist(plt.Circle((0, 0), 0.70, color='white'))
plt.show()
```



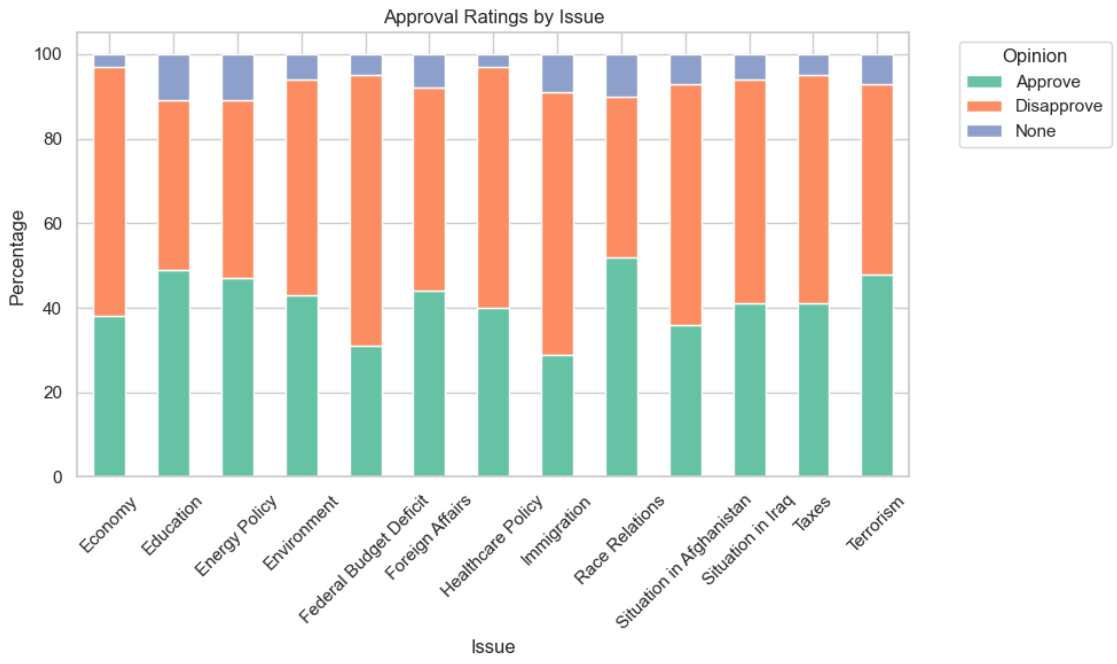
[58]:
```
# Suppress the specific FutureWarning related to use_inf_as_na
warnings.filterwarnings("ignore", category=FutureWarning, module="seaborn.
 ↪_oldcore")

# Replace infinite values with NaNs (if any)
hotdog_data.replace([float('inf'), -float('inf')], float('nan'), inplace=True)

# Drop any rows with NaN values to ensure clean data
hotdog_data.dropna(subset=['Year', 'Dogs eaten'], inplace=True)

# Set up the plotting environment
sns.set(style="whitegrid")
```

```
# Line Chart: Trend of the number of hotdogs eaten over the years
plt.figure(figsize=(10, 6))
sns.lineplot(data=hotdog_data, x='Year', y='Dogs eaten', marker='o')
plt.title('Trend of the Number of Hotdogs Eaten Over the Years')
plt.xticks(rotation=45)
plt.xlabel('Year')
plt.ylabel('Number of Hotdogs Eaten')
plt.tight_layout()
plt.show()
```



[57]:
```
# Reshape the data to long format
obama_data_long = obama_data.melt(id_vars=['Issue'], var_name='Opinion',␣
 ↪value_name='Percentage')

# Set up the plotting environment
sns.set(style="whitegrid")

# Create a pivot table for the stacked bar plot
pivot_data = obama_data_long.pivot(index='Issue', columns='Opinion',␣
 ↪values='Percentage')

# Plot the stacked bar chart
pivot_data.plot(kind='bar', stacked=True, figsize=(10, 6), color=sns.
 ↪color_palette("Set2"))
```

```
plt.title('Approval Ratings by Issue')
plt.xticks(rotation=45)
plt.xlabel('Issue')
plt.ylabel('Percentage')
plt.legend(title='Opinion', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



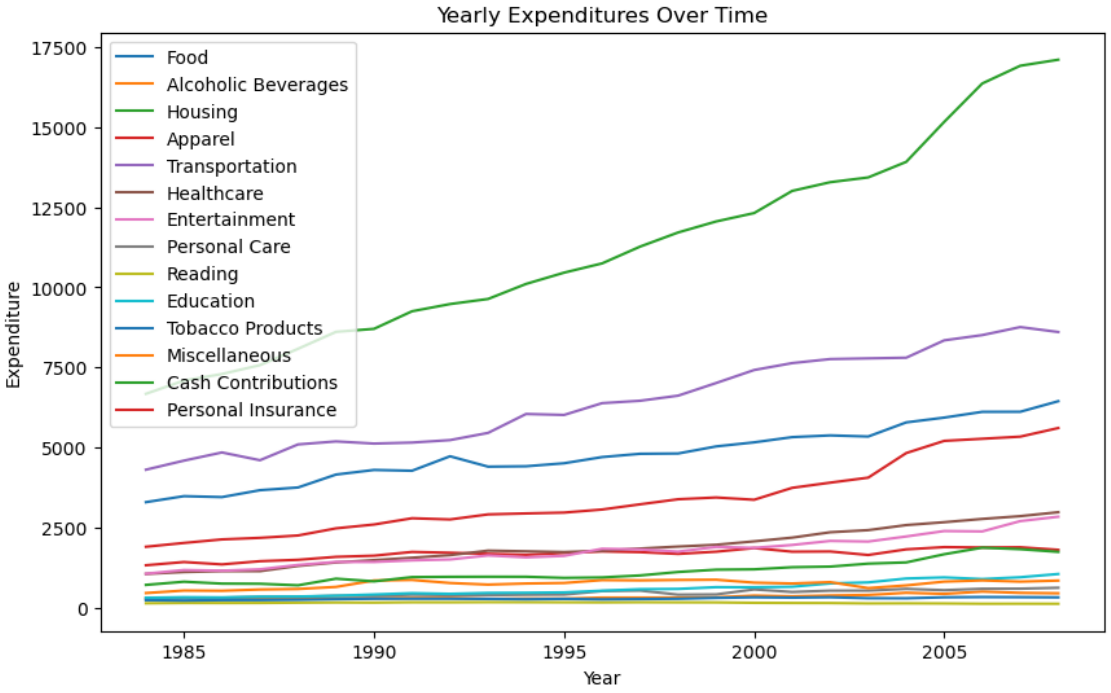
[ ]:

DSC640_Visualizations_week3-4(Python)

July 7, 2024

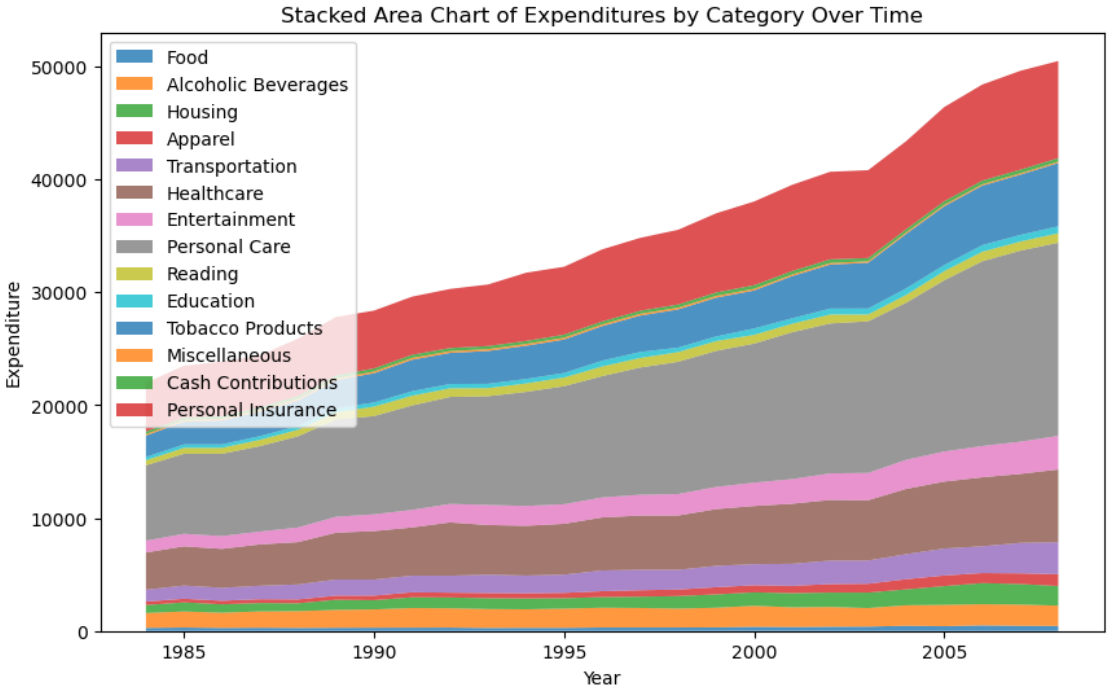
Area Chart

[19]:
```
import pandas as pd
import squarify
import matplotlib.pyplot as plt

# Load the expenditures data
expenditures_df = pd.read_csv('expenditures.txt', sep='\t')

# Plot Area Chart
plt.figure(figsize=(10, 6))
for category in expenditures_df['category'].unique():
    category_data = expenditures_df[expenditures_df['category'] == category].
 ↪groupby('year')['expenditure'].sum()
    plt.plot(category_data.index, category_data.values, label=category)
plt.title('Yearly Expenditures Over Time')
plt.xlabel('Year')
plt.ylabel('Expenditure')
plt.legend()
plt.show()
```



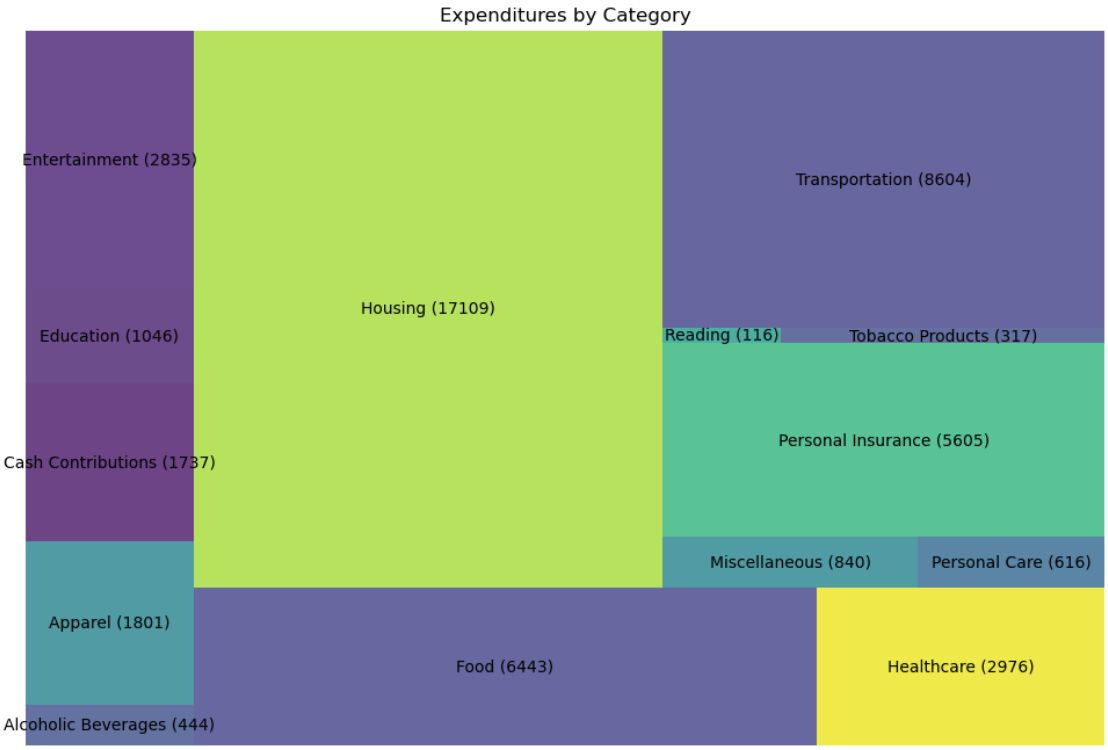
Stacked Area Chart

[20]:
```
# Stacked Area Chart
plt.figure(figsize=(10, 6))
categories = expenditures_df['category'].unique()
stacked_data = expenditures_df.pivot_table(values='expenditure', index='year',␣
 ↪columns='category', aggfunc='sum').fillna(0)
plt.stackplot(stacked_data.index, stacked_data.T, labels=categories, alpha=0.8)
plt.title('Stacked Area Chart of Expenditures by Category Over Time')
plt.xlabel('Year')
plt.ylabel('Expenditure')
plt.legend(loc='upper left')
plt.show()
```



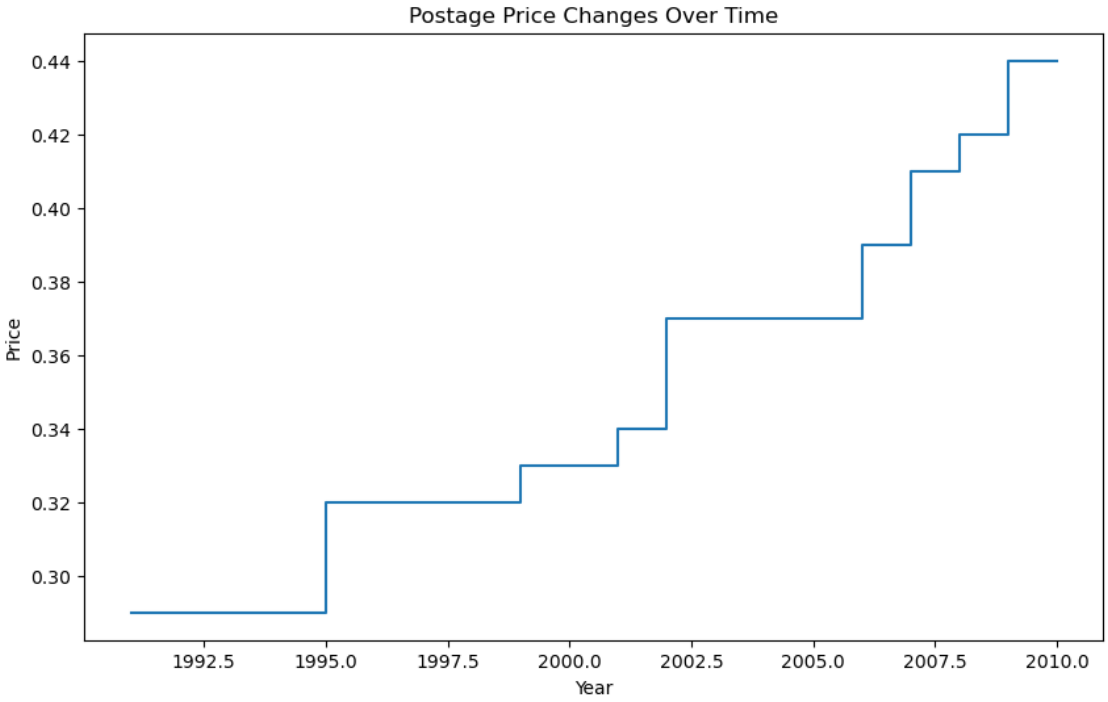
Treemap

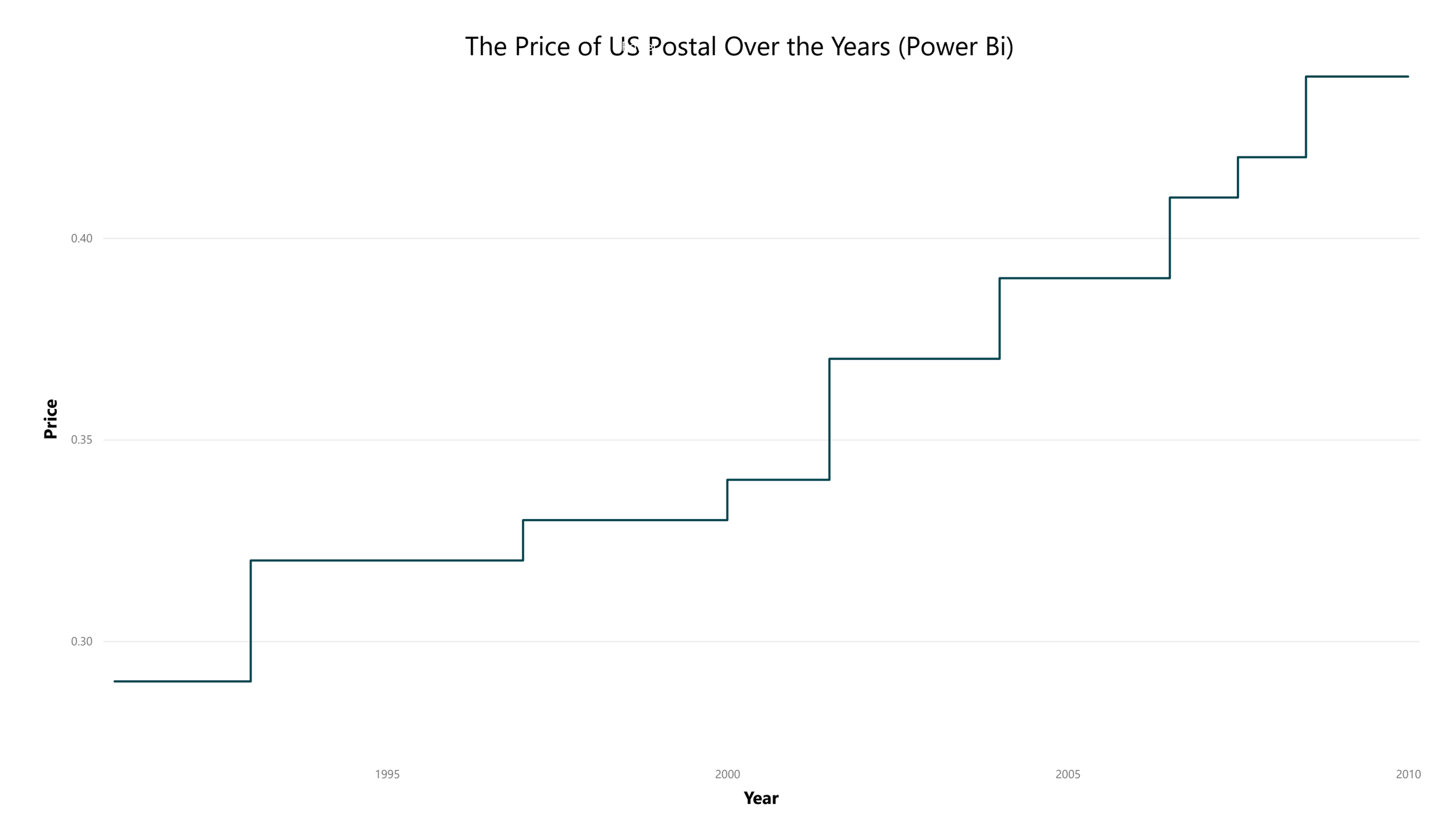
[21]:
```
import squarify

# Prepare data for Treemap
latest_year = expenditures_df['year'].max()
latest_data = expenditures_df[expenditures_df['year'] == latest_year].
 ↪groupby('category')['expenditure'].sum().reset_index()
sizes = latest_data['expenditure']
labels = latest_data['category'] + ' (' + latest_data['expenditure'].
 ↪astype(str) + ')'

plt.figure(figsize=(12, 8))
squarify.plot(sizes=sizes, label=labels, alpha=0.8)
plt.title('Expenditures by Category')
plt.axis('off')
plt.show()
```



Step Chart

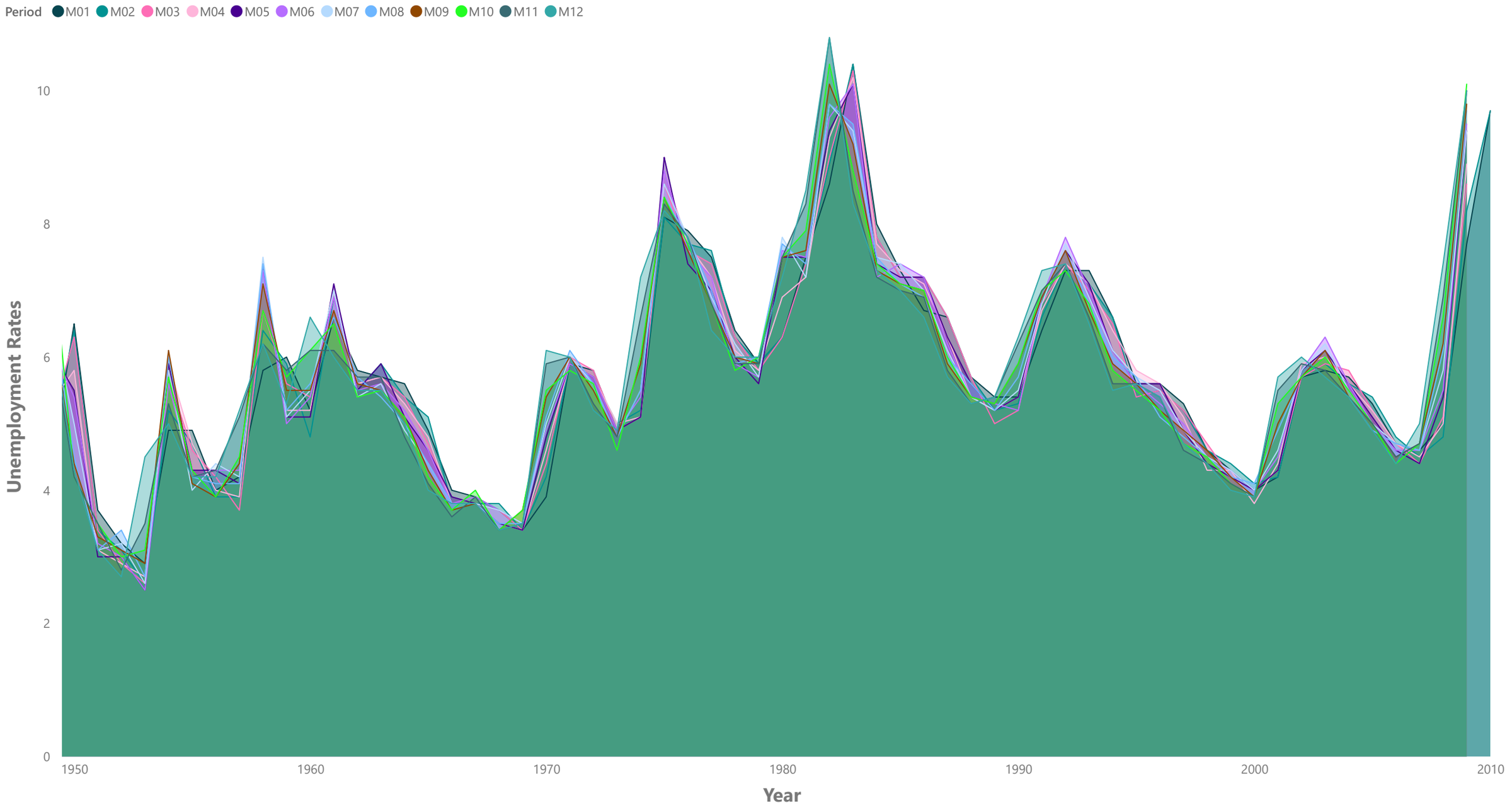
[22]:
```
# Load the us-postage data
us_postage_df = pd.read_excel('us-postage.xlsm', sheet_name='us-postage')

# Step Chart
plt.figure(figsize=(10, 6))
years = us_postage_df['Year']
prices = us_postage_df['Price']
step_years = [years.iloc[0]]
step_prices = [prices.iloc[0]]
for i in range(1, len(years)):
    step_years.extend([years.iloc[i-1], years.iloc[i]])
    step_prices.extend([prices.iloc[i-1], prices.iloc[i]])

plt.step(step_years, step_prices, where='post')
plt.title('Postage Price Changes Over Time')
plt.xlabel('Year')
plt.ylabel('Price')
plt.show()
```

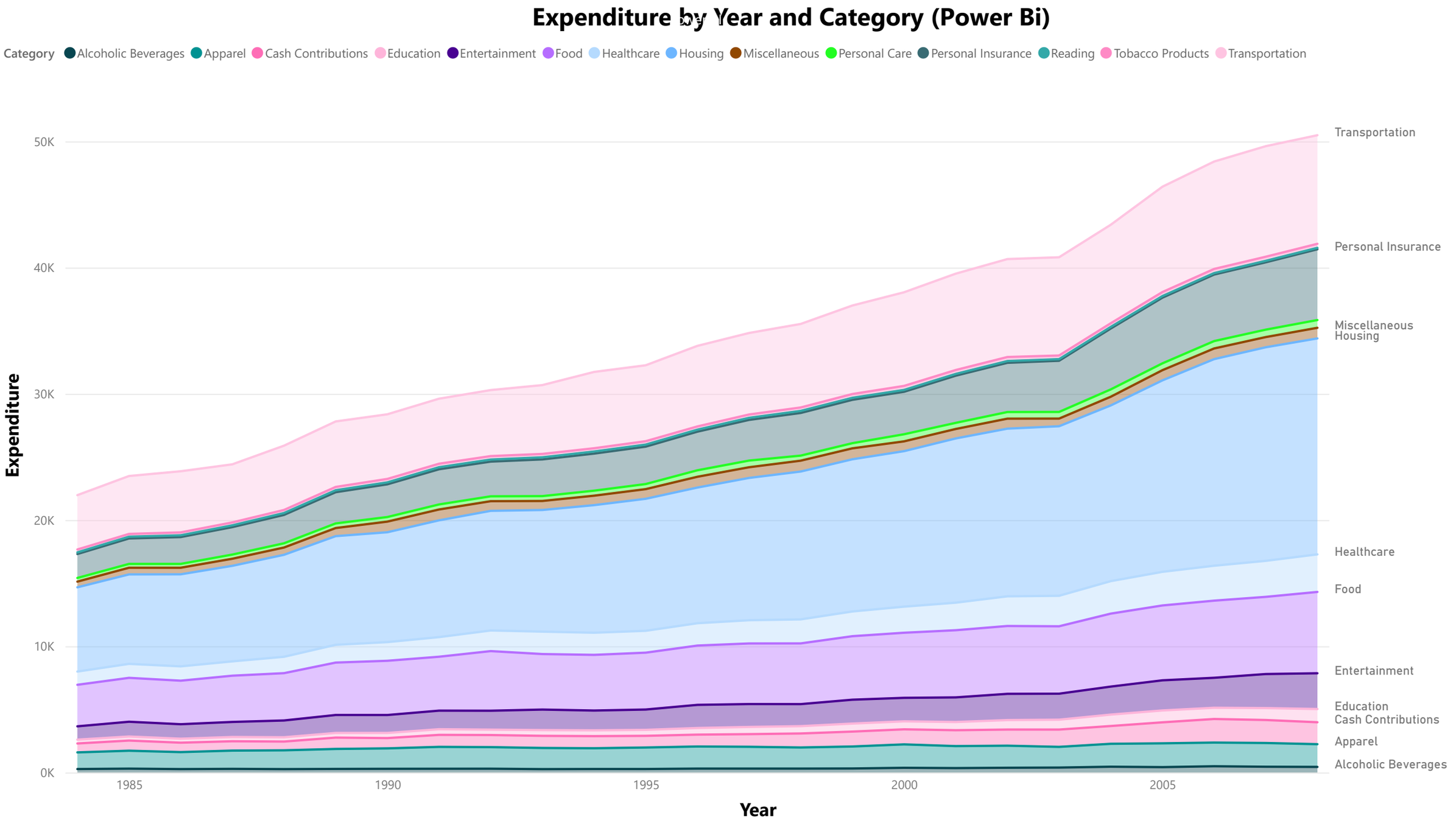

Postage Price Changes Over Time
Price
0.44
0.42
0.40
0.38
0.36
0.34
0.32
0.30
1992.5
1995.0
1997.5
2000.0
2002.5
2005.0
2007.5
2010.0
Year


The Price of US Postal Over the Years (Power Bi)
0.40
0.35
0.30
Price
1995
2000
2005
2010
Year

Expenditure by Category (Power Bi)

Unemployment by Years and Months (Power Bi)




Expenditure by Year and Category (Power Bi)
Category
Alcoholic Beverages
Apparel
Cash Contributions
Education
Entertainment
Food
Healthcare
Housing
Miscellaneous
Personal Care
Personal Insurance
Reading
Tobacco Products
Transportation
50K
40K
30K
20K
10K
0K
Expenditure
1985
1990
1995
2000
2005
Year
Transportation
Personal Insurance
Miscellaneous
Housing
Healthcare
Food
Entertainment
Education
Cash Contributions
Apparel
Alcoholic Beverages

Week3-4 (R)

Said Moussadeq

2024-07-03

R Visualization

Area Chart

```
library(ggplot2)
library(dplyr)
```

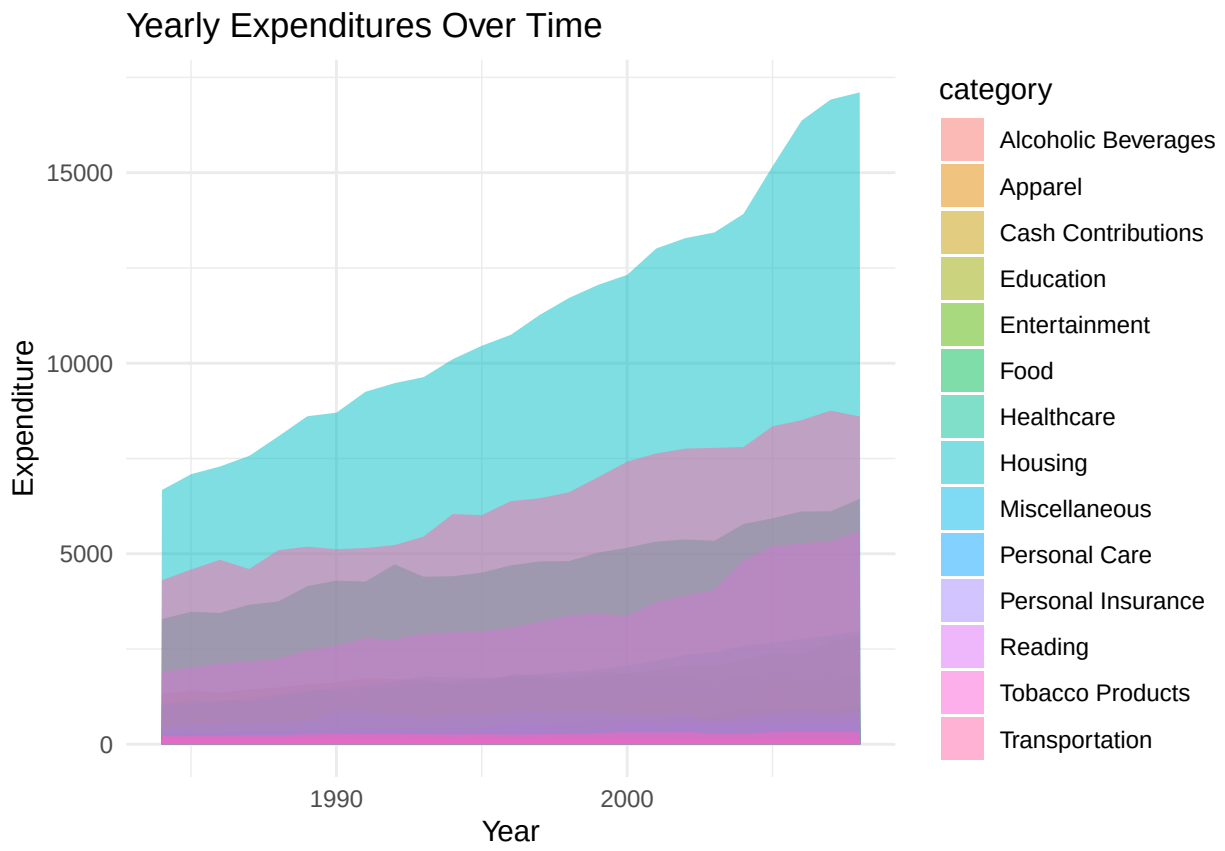
```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

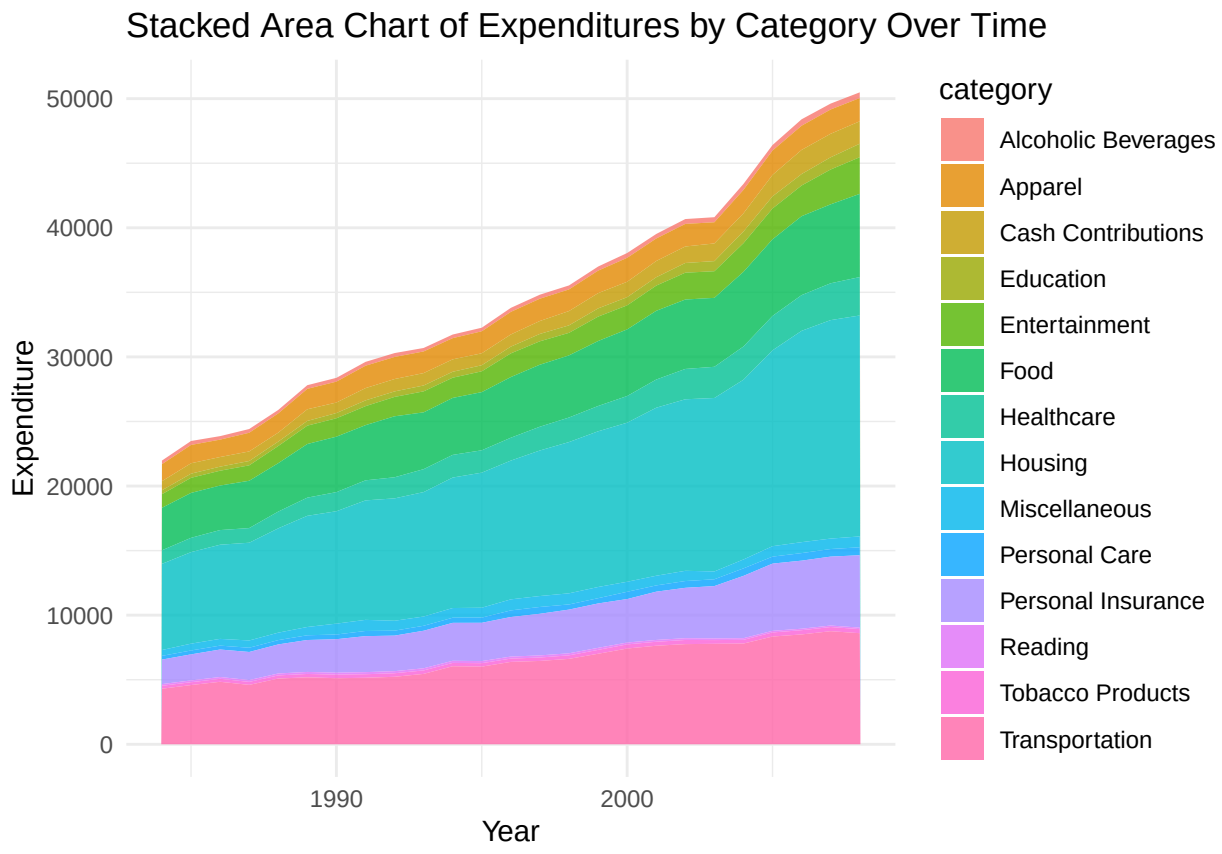
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Set the working directory
setwd("C:/Users/TheArchitect/Downloads")


# Load the data
expenditures <- read.table("expenditures.txt", header=TRUE, sep="\t")

# Area Chart
ggplot(expenditures, aes(x=year, y=expenditure, fill=category)) +
  geom_area(position="identity", alpha=0.5) +
  labs(title="Yearly Expenditures Over Time", x="Year", y="Expenditure") +
  theme_minimal()
```



Stacked Area Chart

```
# Stacked Area Chart
ggplot(expenditures, aes(x=year, y=expenditure, fill=category)) +
  geom_area(position="stack", alpha=0.8) +
  labs(title="Stacked Area Chart of Expenditures by Category Over Time", x="Year", y="Expenditure") +
  theme_minimal()
```

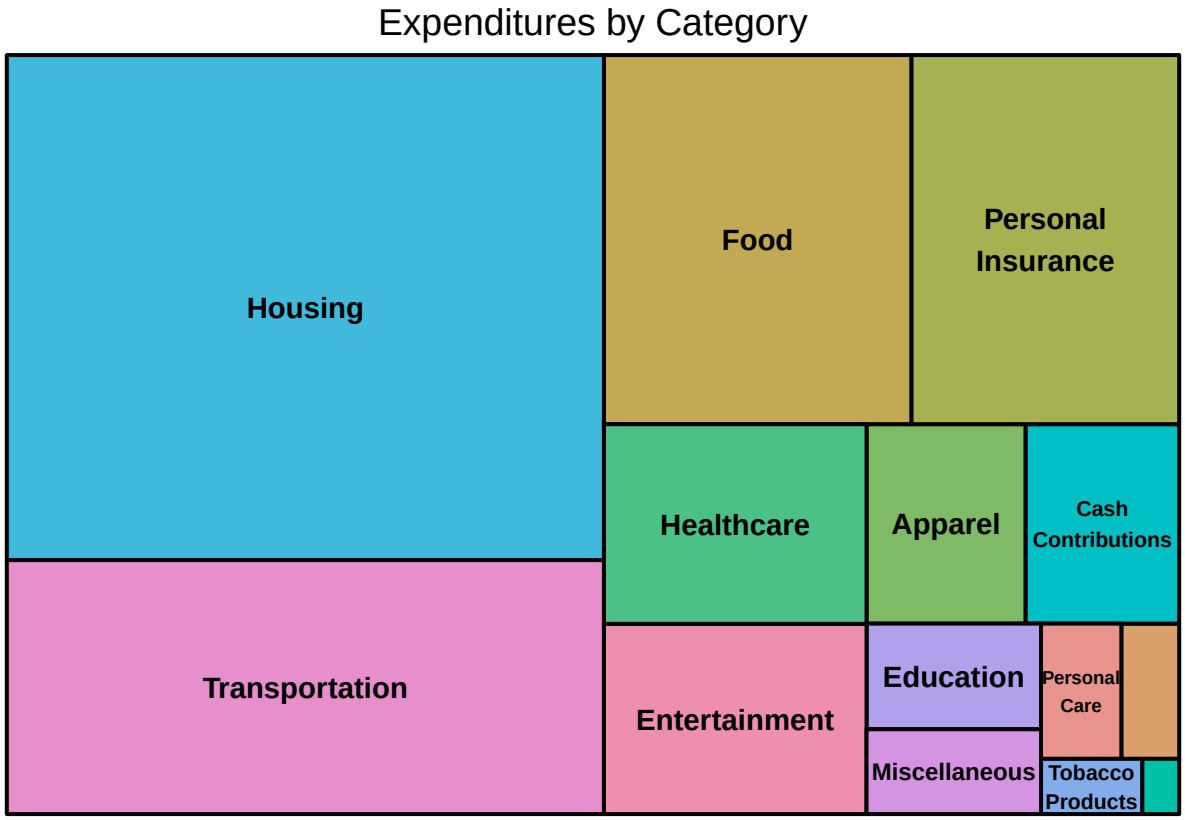


Treemap

```
# Install necessary packages if not already installed
if (!requireNamespace("treemap", quietly = TRUE)) install.packages("treemap", dependencies = TRUE)
if (!requireNamespace("dplyr", quietly = TRUE)) install.packages("dplyr", dependencies = TRUE)
if (!requireNamespace("readxl", quietly = TRUE)) install.packages("readxl", dependencies = TRUE)

# Load the libraries
library(treemap)
```

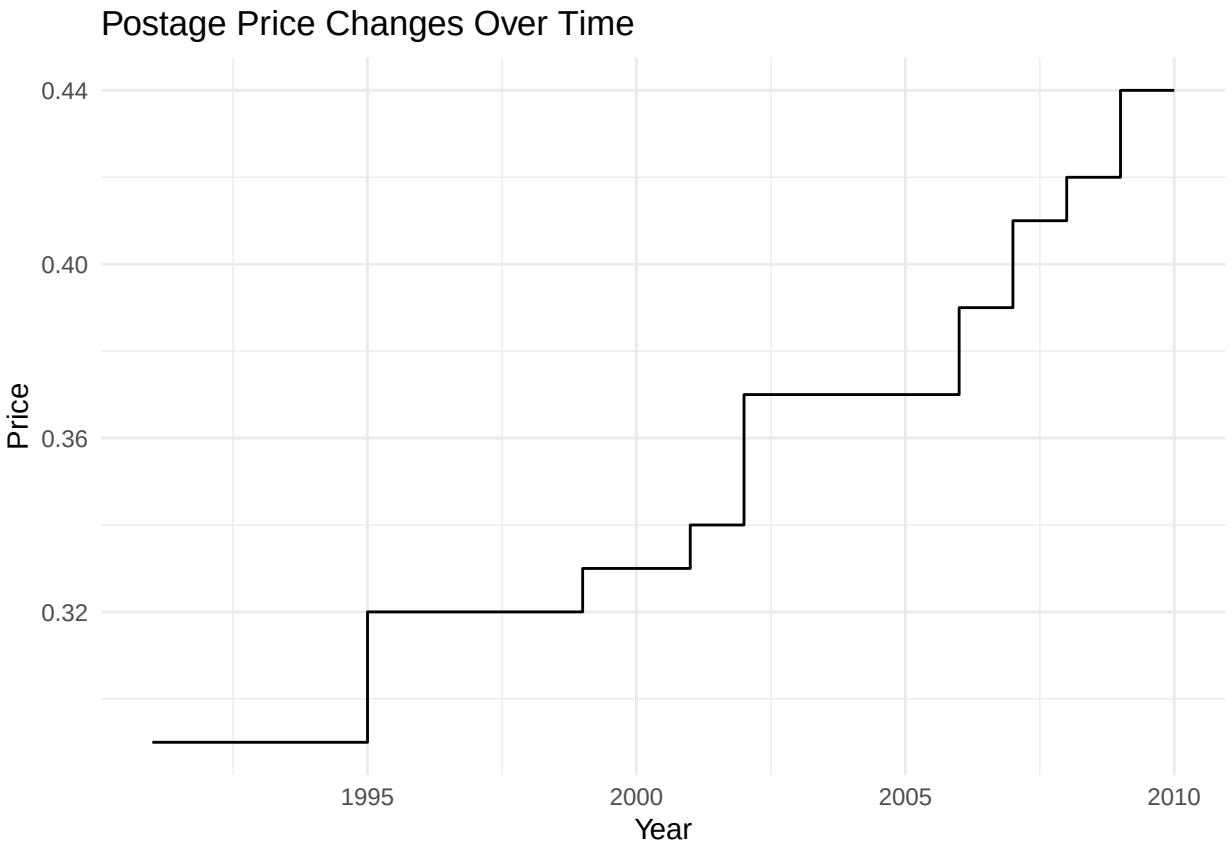
```
## Warning: package 'treemap' was built under R version 4.3.3
```

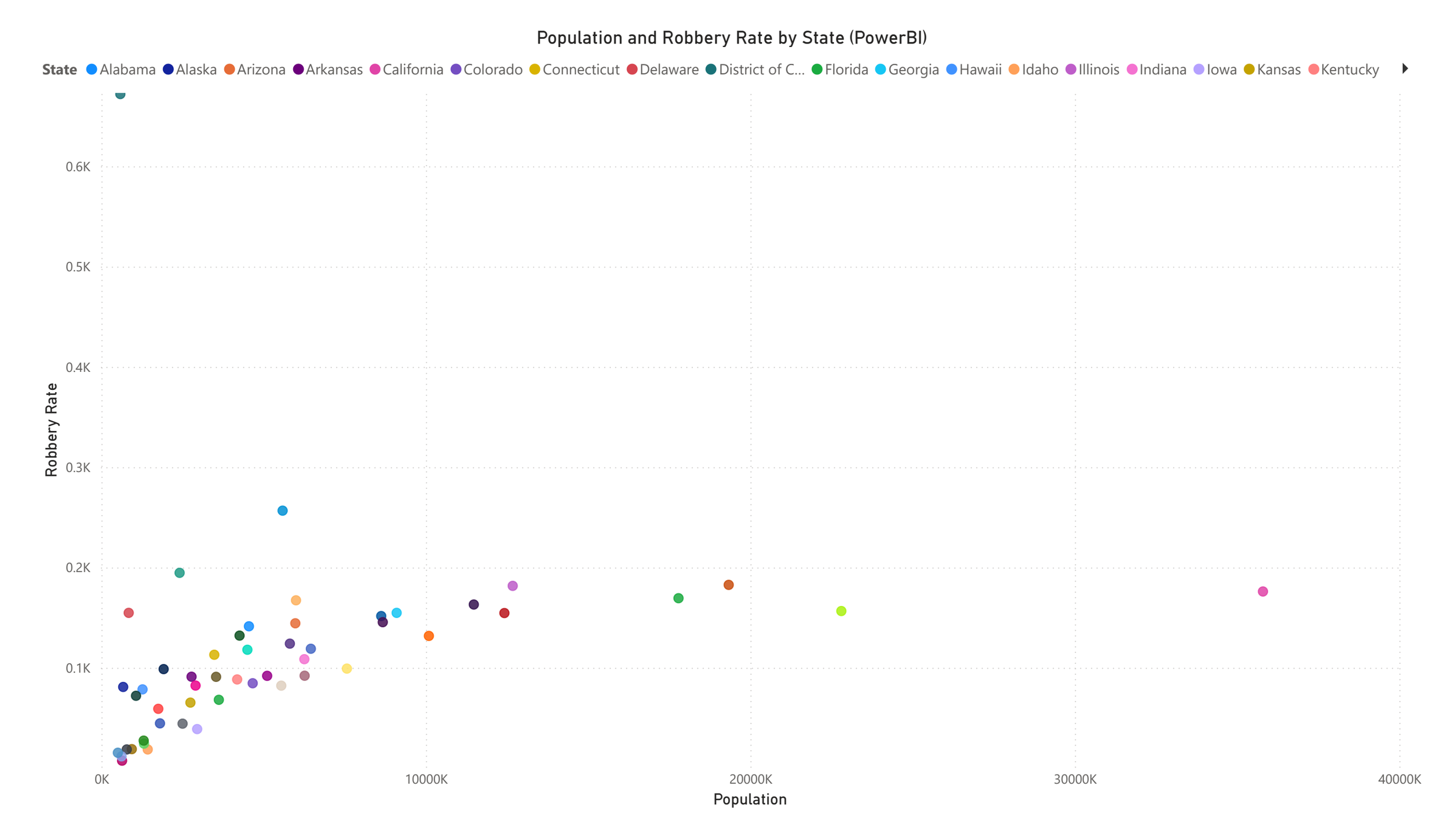
```
library(dplyr)
library(readxl)


# Prepare data for Treemap
latest_year <- max(expenditures$year)
latest_data <- expenditures %>% filter(year == latest_year) %>% group_by(category) %>% summarise(expendi

# Treemap
treemap(latest_data, index="category", vSize="expenditure", title="Expenditures by Category")
```



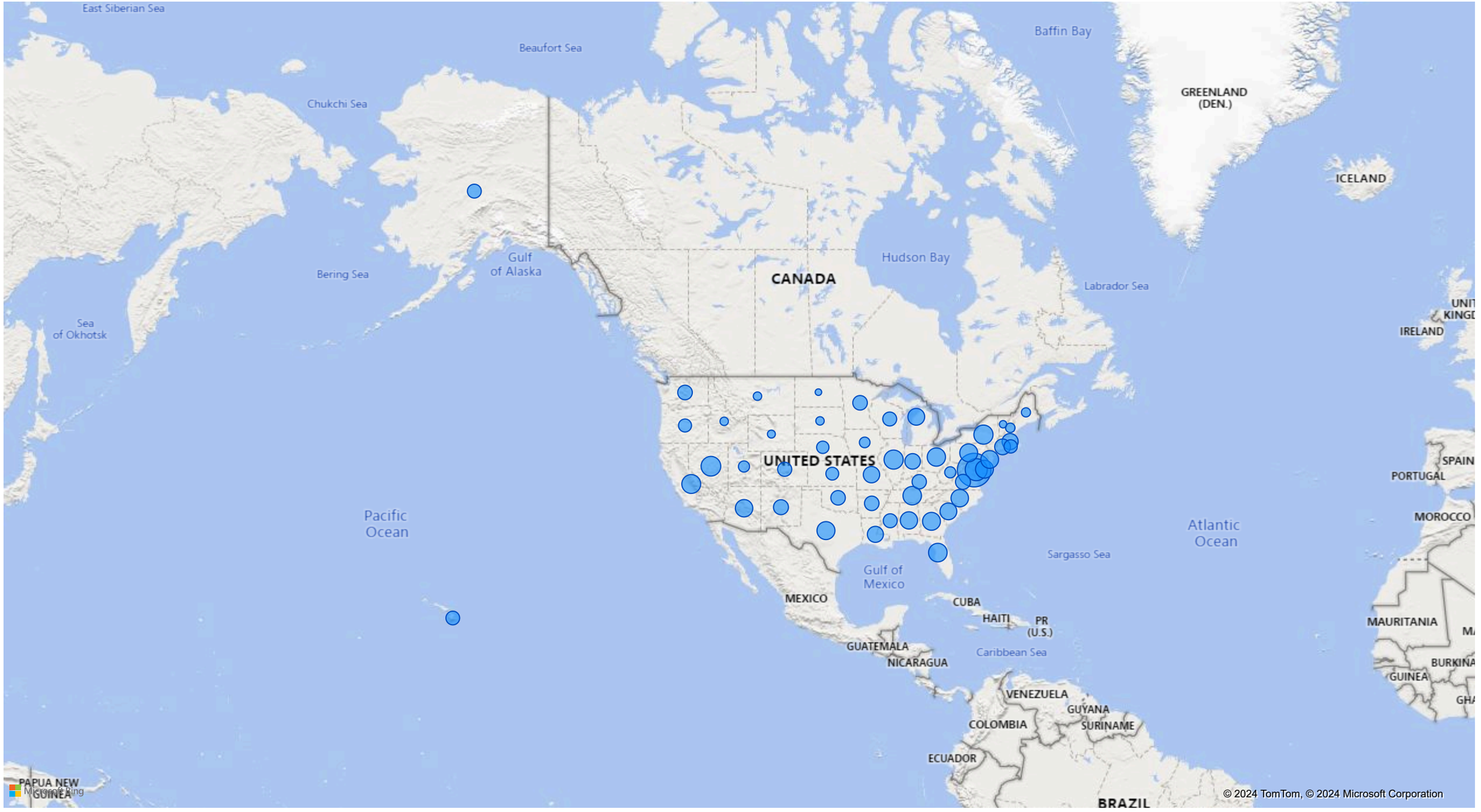
Step Chart

```
library(ggplot2)
# Set the working directory
setwd("C:/Users/TheArchitect/Downloads")


# Load the data
us_postage <- readxl::read_excel("us-postage.xlsm", sheet="us-postage")

# Step Chart
ggplot(us_postage, aes(x=Year, y=Price)) +
  geom_step() +
  labs(title="Postage Price Changes Over Time", x="Year", y="Price") +
  theme_minimal()
```

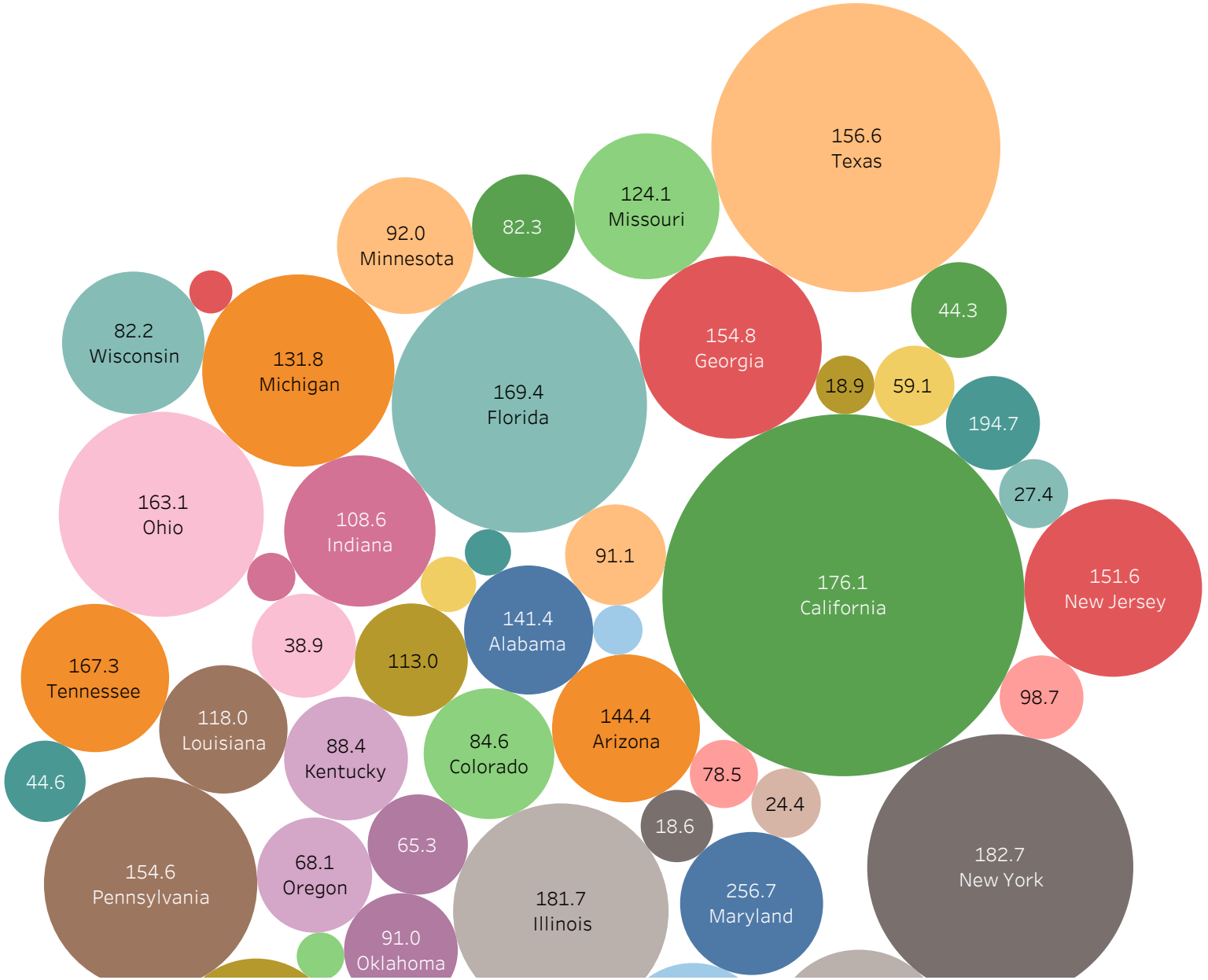

Postage Price Changes Over Time
0.44
0.40
0.36
0.32
Price
1995
2000
2005
2010
Year


Population and Robbery Rate by State (PowerBI)
State
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of C...
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Robbery Rate
0.6K
0.5K
0.4K
0.3K
0.2K
0.1K
0K
10000K
20000K
30000K
40000K
Population

Robbery Rate by state (PowerBI)



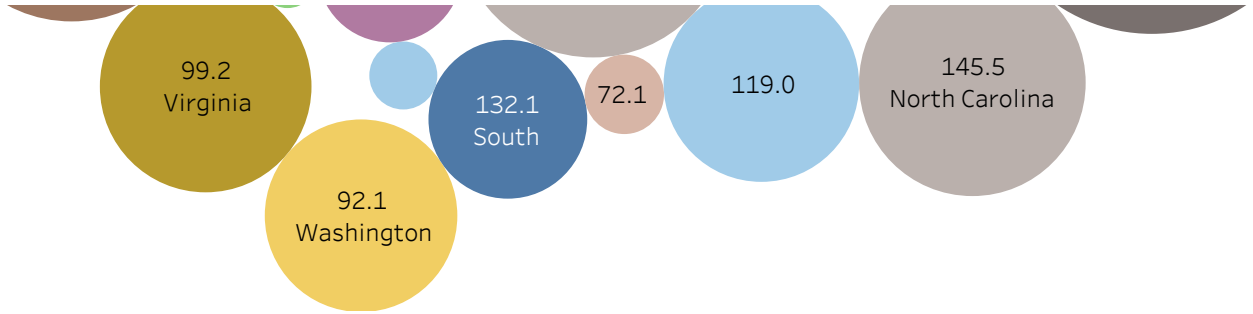
Robbery Rate by State (Tableau)



Sum of Robbery and State. Color shows details about State. Size shows sum of Population. The marks are labeled by sum of Robbery and State.

State
- Alabama
- Alaska
- Arizona
- Arkansas
- California
- Colorado
- Connecticut
- Delaware
- District of Columbia
- Florida
- Georgia
- Hawaii
- Idaho
- Illinois
- Indiana
- Iowa
- Kansas
- Kentucky
- Louisiana
- Maine
- Maryland
- Massachusetts
- Michigan
- Minnesota
- Mississippi
- Missouri
- Montana
- Nebraska
- Nevada
- New Hampshire
- New Jersey
- New Mexico
- New York

Robbery Rate by State (Tableau)



Sum of Robbery and State. Color shows details about State. Size shows sum of Population. The marks are labeled by sum of Robbery and State.

State

- Alabama
- Alaska
- Arizona
- Arkansas
- California
- Colorado
- Connecticut
- Delaware
- District of Columbia
- Florida
- Georgia
- Hawaii
- Idaho
- Illinois
- Indiana
- Iowa
- Kansas
- Kentucky
- Louisiana
- Maine
- Maryland
- Massachusetts
- Michigan
- Minnesota
- Mississippi
- Missouri
- Montana
- Nebraska
- Nevada
- New Hampshire
- New Jersey
- New Mexico
- New York

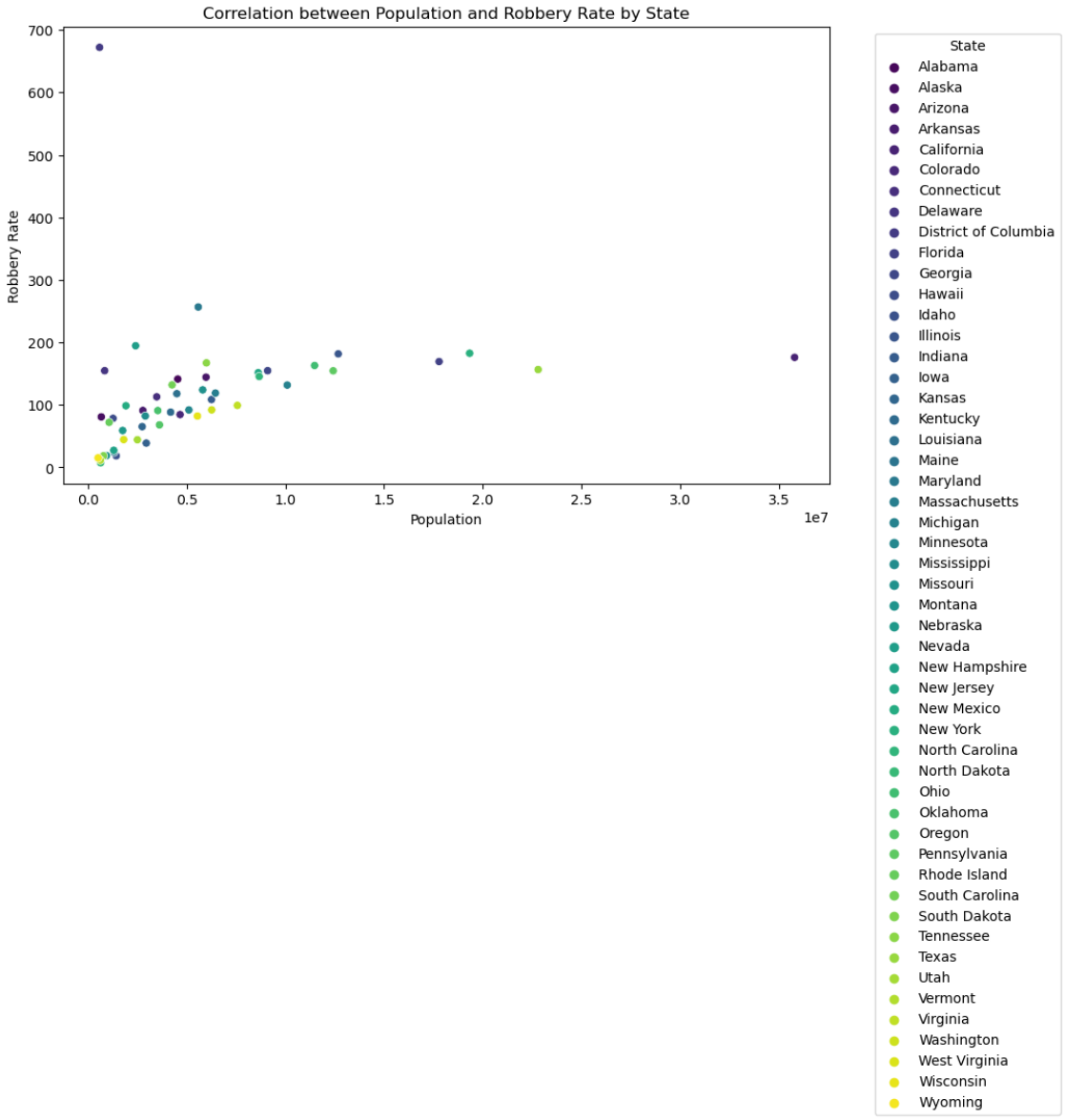
DSC640_Week5-6_Moussadeq

July 9, 2024

Python Visualizations

0.0.1 Scatter Plot

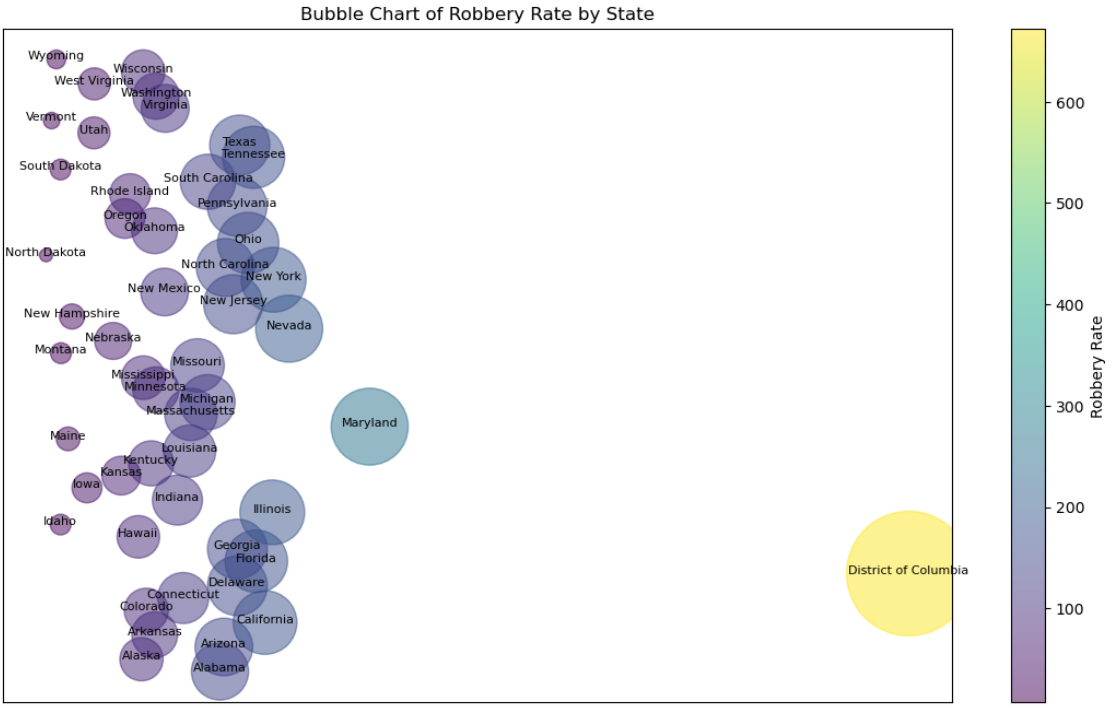
[39]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
data = pd.read_csv('crimerates-by-state-2005.csv')

# Scatter Plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='population', y='robbery', hue='state', data=data,␣
 ↪palette='viridis')
plt.title('Correlation between Population and Robbery Rate by State')
plt.xlabel('Population')
plt.ylabel('Robbery Rate')
plt.legend(title='State', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



0.0.2 Bubble Chart

[40]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv('C:/Users/TheArchitect/Downloads/crimerates-by-state-2005.
  ↪csv')

# Create the bubble chart
plt.figure(figsize=(14, 8))
```

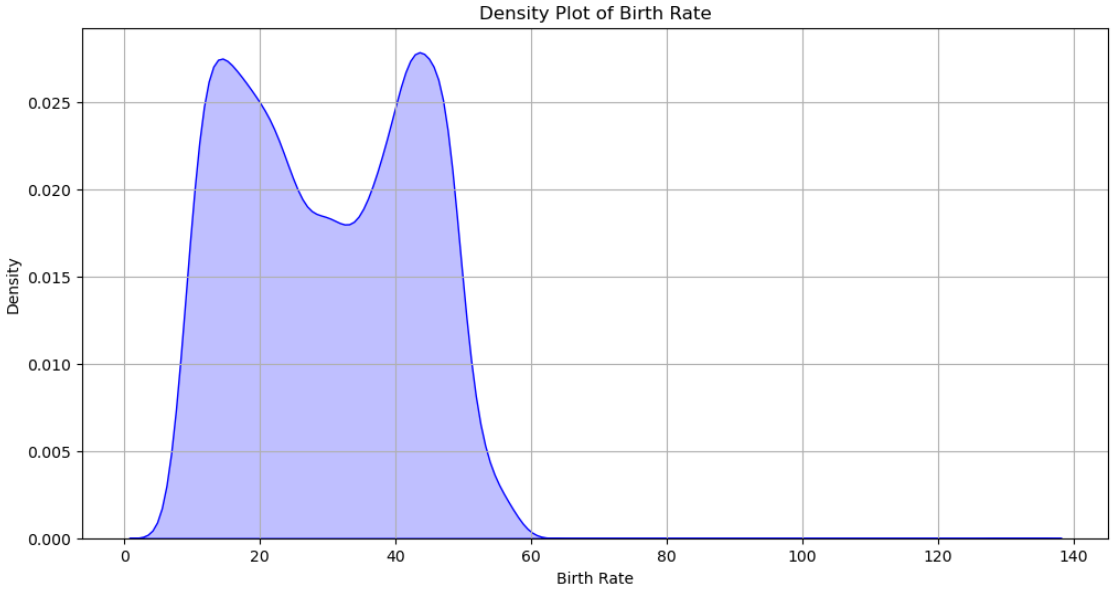
```
scatter = plt.scatter(data['robbery'], range(len(data)), s=data['robbery']*10,␣
 ↪alpha=0.5, c=data['robbery'], cmap='viridis')
plt.title('Bubble Chart of Robbery Rate by State')

# Add colorbar
cbar = plt.colorbar(scatter)
cbar.set_label('Robbery Rate')

# Annotate state names
for i in range(len(data)):
    plt.text(data['robbery'][i], i, data['state'][i], fontsize=8, ha='center')

# Remove x-axis and y-axis labels and ticks
plt.xticks([])
plt.yticks([])

plt.xlabel('')
plt.ylabel('')

plt.show()
```



0.0.3 Density Plot

[41]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Load the data
data = pd.read_csv('C:/Users/TheArchitect/Downloads/birth-rates-yearly.csv')

# Convert 'rate' column to numeric, forcing non-numeric values to NaN
data['rate'] = pd.to_numeric(data['rate'], errors='coerce')

# Replace infinite values with NaN
data['rate'].replace([np.inf, -np.inf], np.nan, inplace=True)

# Drop rows with NaN values in 'rate'
data.dropna(subset=['rate'], inplace=True)

# Create the density plot
plt.figure(figsize=(12, 6))
sns.kdeplot(data=data, x='rate', fill=True, color='blue')
plt.title('Density Plot of Birth Rate')
plt.xlabel('Birth Rate')
plt.ylabel('Density')
plt.grid(True)
plt.show()
```

```
C:\Users\TheArchitect\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```


Density Plot of Birth Rate
Density
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
120
140
Birth Rate

[ ]:

Week5-6_Moussadeq

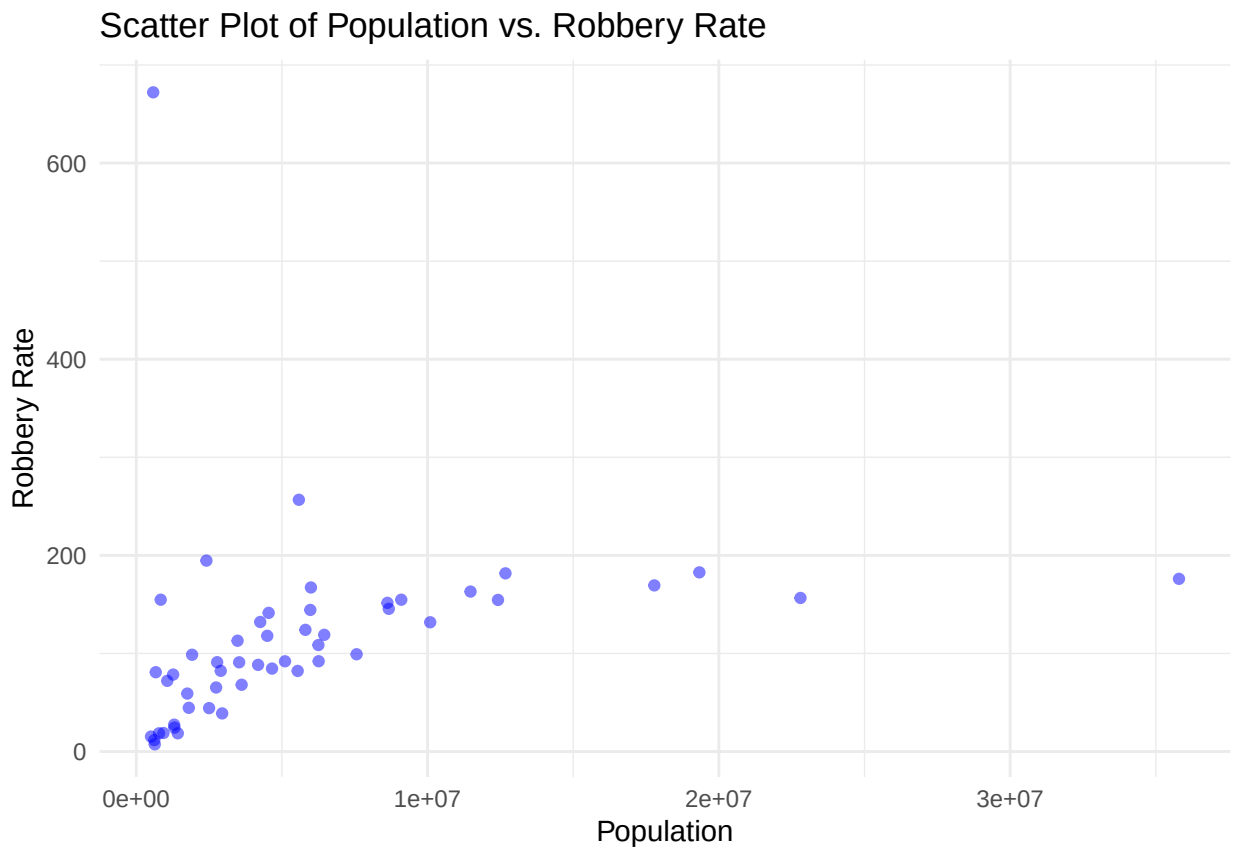
Said Moussadeq

2024-07-09

R Visualization

Scatter Plot of Population vs. Robbery Rate

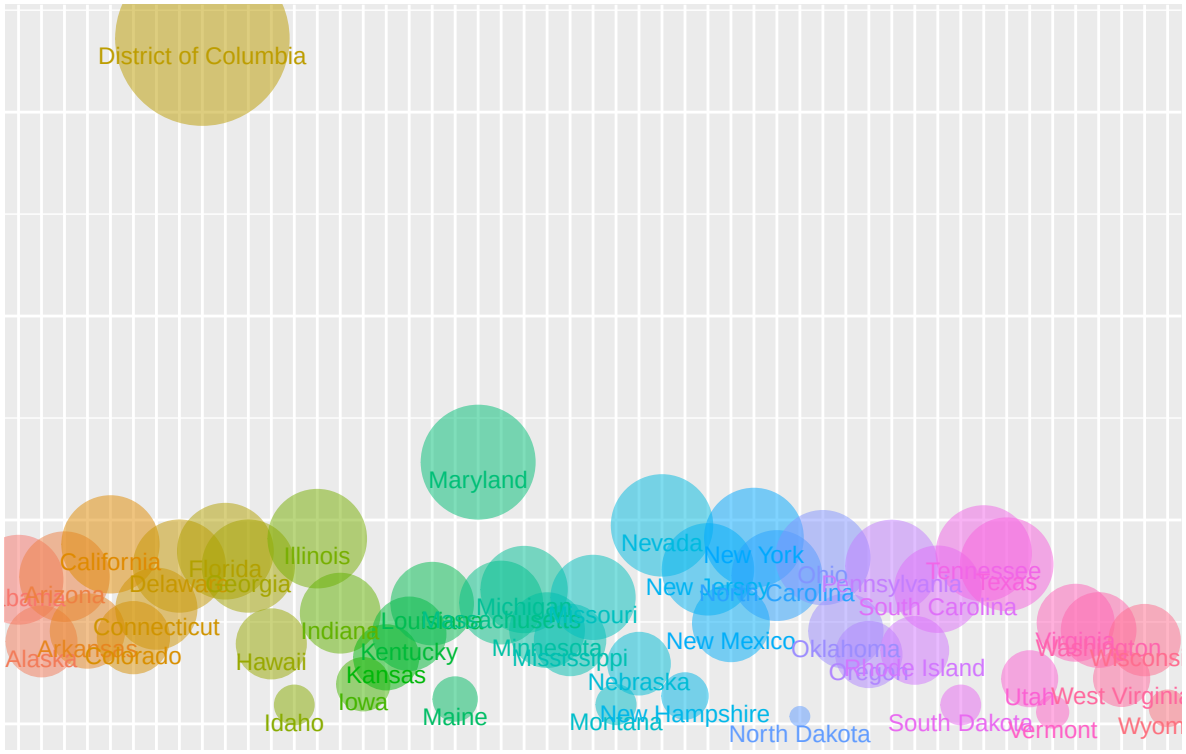
```
library(ggplot2)

# Set the working directory
setwd("C:/Users/TheArchitect/Downloads")

# Load the data
data <- read.csv('crimerates-by-state-2005.csv')

# Create the scatter plot
library(ggplot2)

ggplot(data, aes(x = population, y = robbery)) +
  geom_point(alpha=0.5, color='blue') +
  labs(title = "Scatter Plot of Population vs. Robbery Rate", x = "Population", y = "Robbery Rate") +
  theme_minimal()
```



Bubble chart of Robbery Rate by State

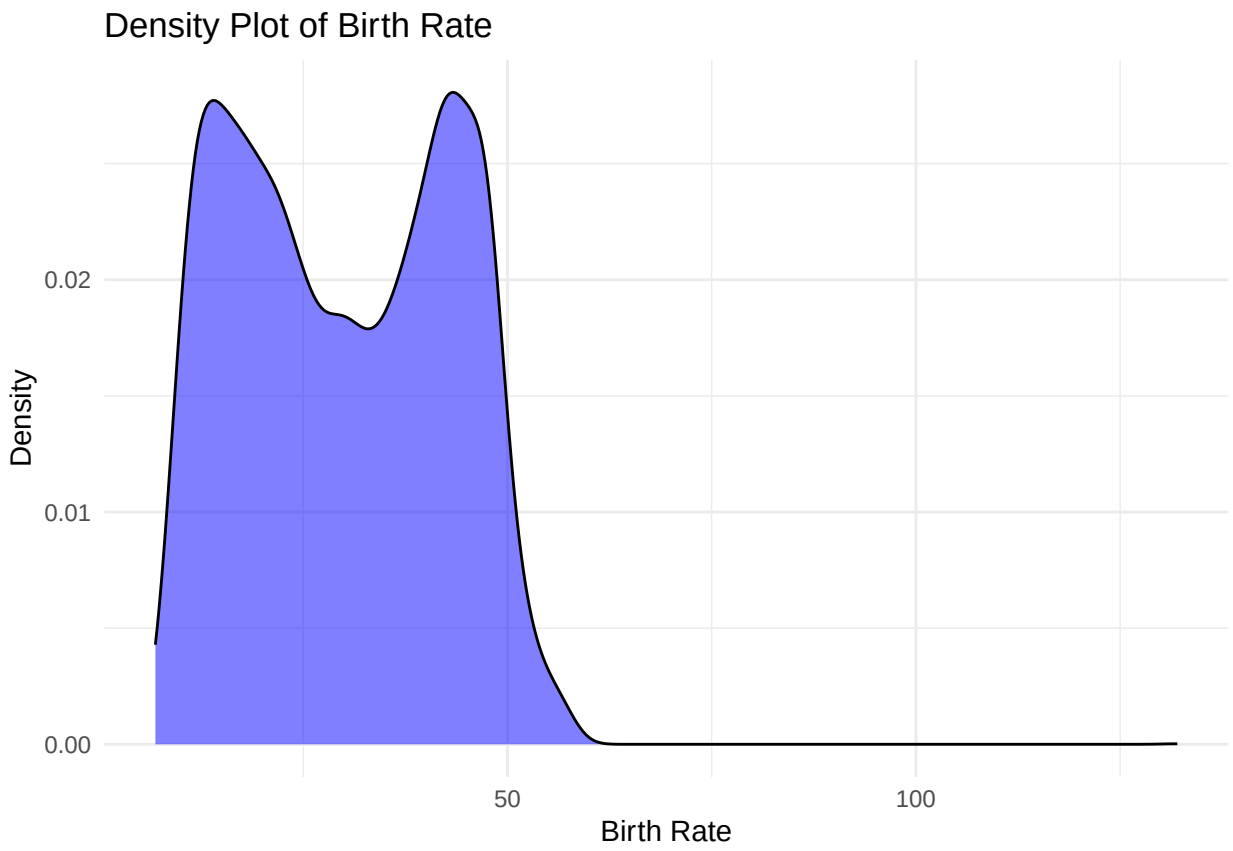
```
# Create the bubble chart
library(ggplot2)

ggplot(data, aes(x = state, y = robbery, size = robbery, color = state)) +
  geom_point(alpha=0.5) +
  scale_size_continuous(range = c(3, 30)) +
  labs(title = "Bubble Chart of Robbery Rate by State", x = "", y = "") +
  theme(axis.text.x = element_blank(), axis.text.y = element_blank(),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank(),
        legend.position = "none") +
  geom_text(aes(label = state), size = 3, vjust = 1.5)
```

Bubble Chart of Robbery Rate by State

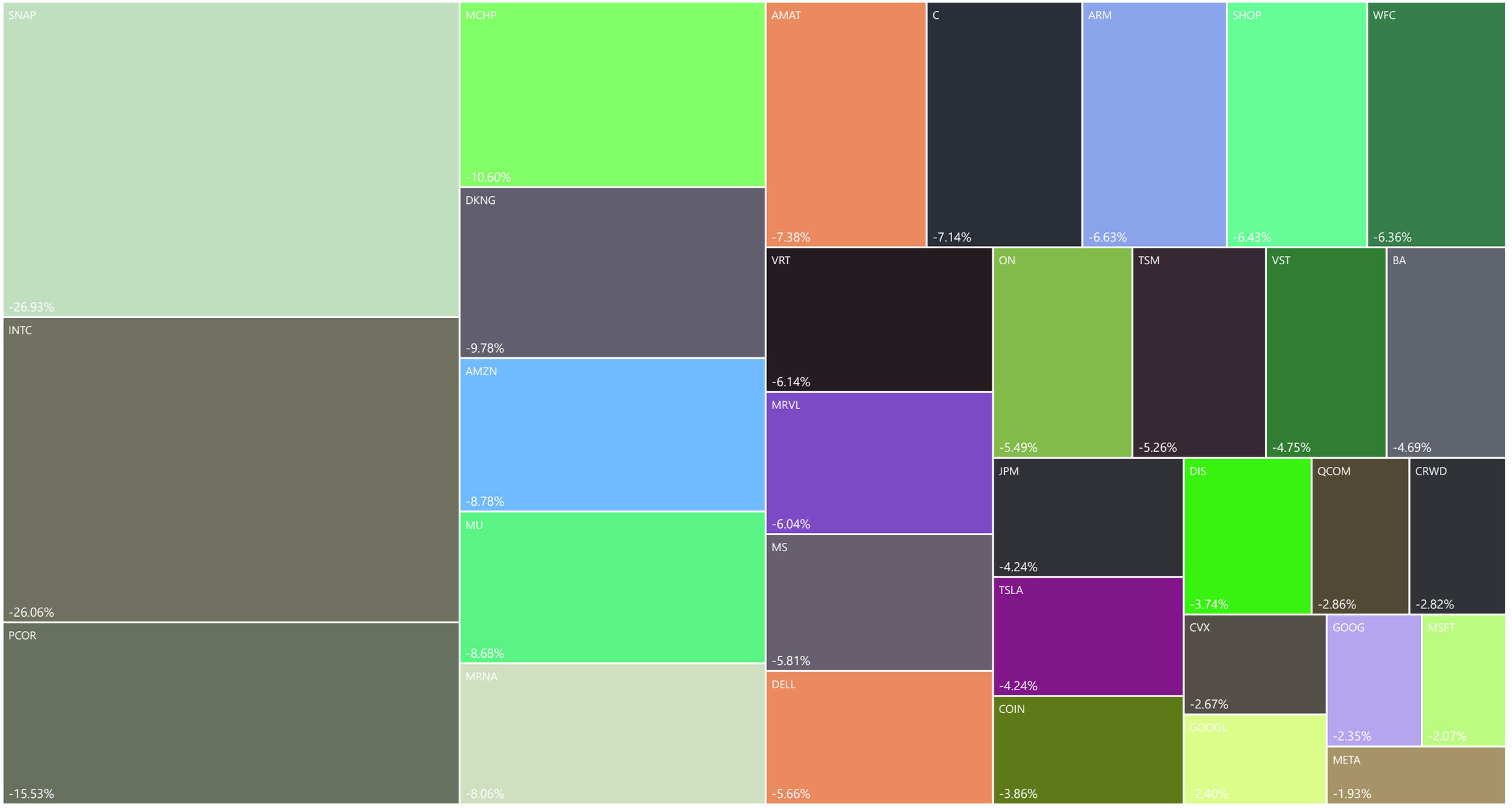


Density Plot of Birth Rate

```r
# Set the working directory
setwd("C:/Users/TheArchitect/Downloads")

# Load the data
data <- read.csv('birth-rates-yearly.csv')

# Create the density plot
library(ggplot2)

ggplot(data, aes(x = rate)) +
  geom_density(fill="blue", alpha=0.5) +
  labs(title = "Density Plot of Birth Rate", x = "Birth Rate", y = "Density") +
  theme_minimal()
```

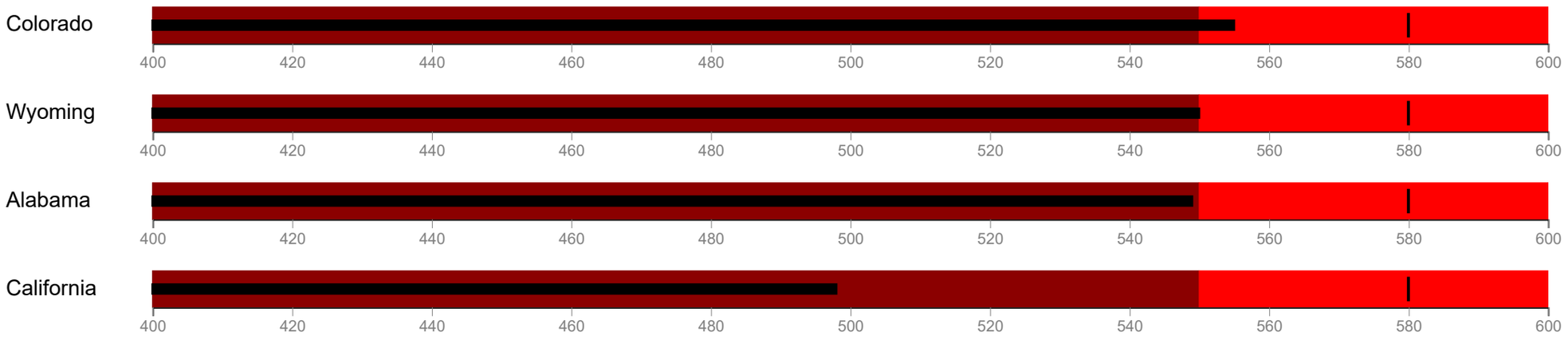

Density Plot of Birth Rate
Density
0.02
0.01
0.00
50
100
Birth Rate

Top Stock Losses - 08-02-2024 (PowerBi)

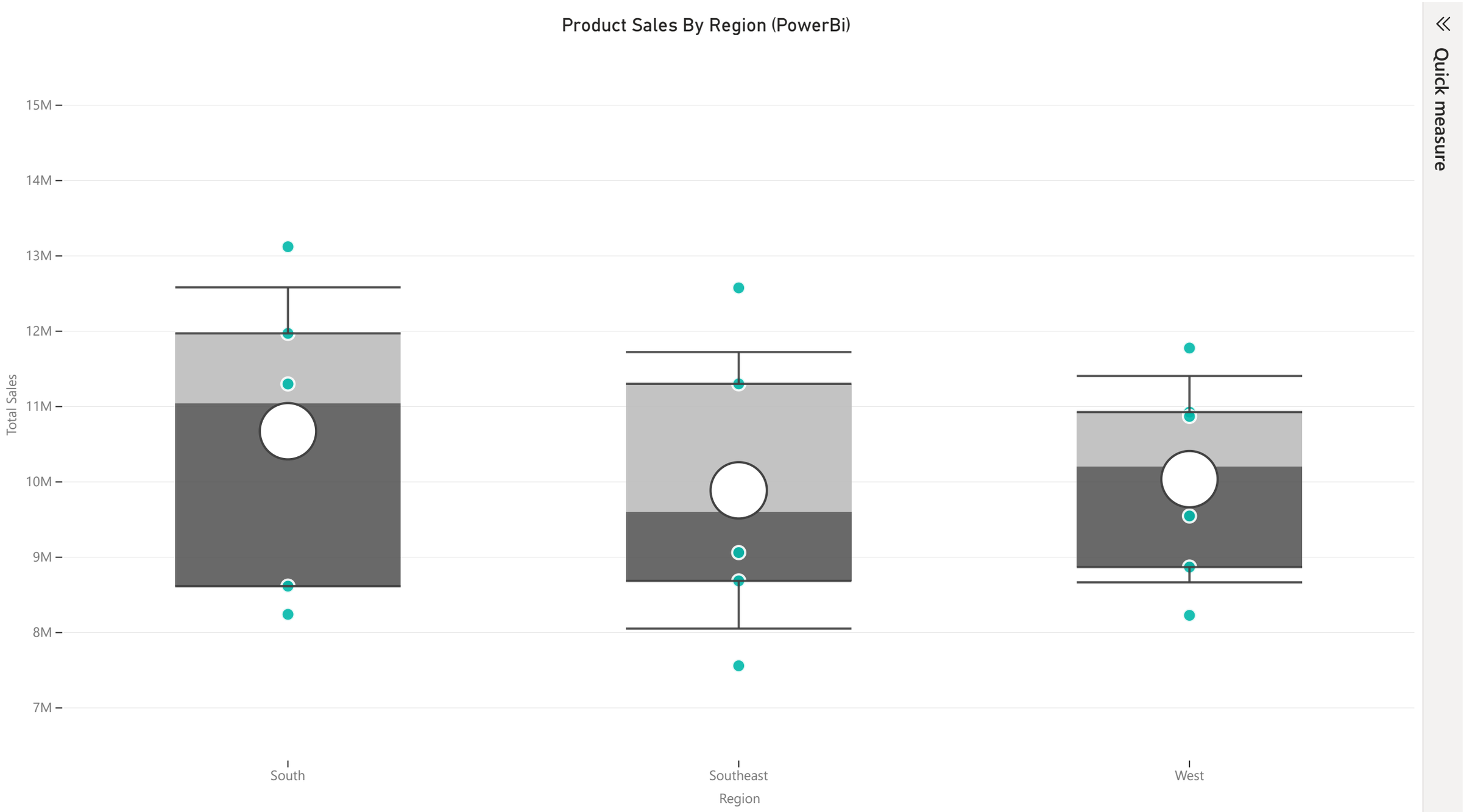


Quick measure

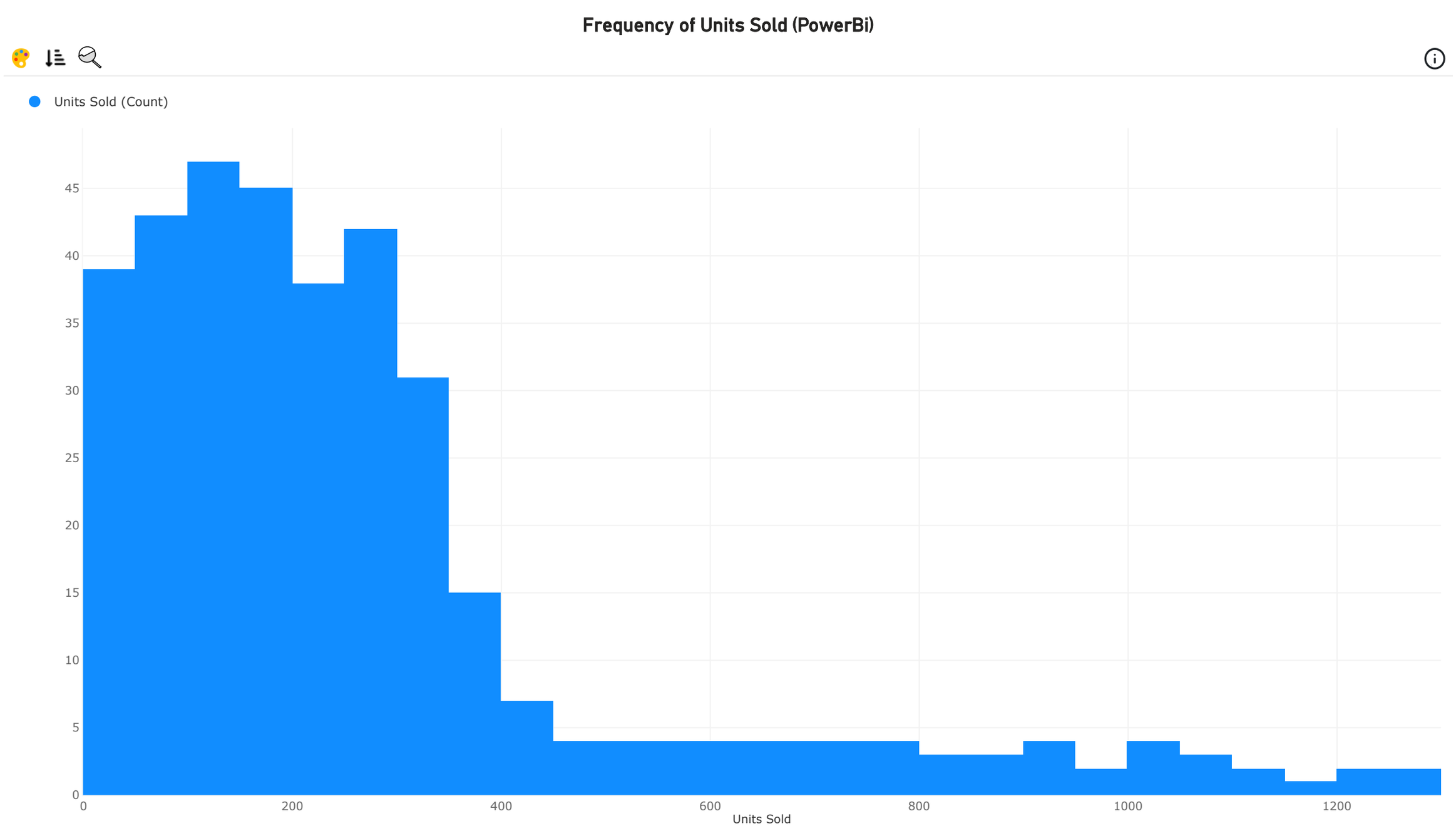
Writing Scores by State (PowerBi)



Quick measure



Quick measure


Frequency of Units Sold (PowerBi)
Units Sold (Count)
45
40
35
30
25
20
15
10
5
0
0
200
400
600
800
1000
1200
Units Sold
Quick measure

R Visualizations

Said Moussadeq

2024-08-03

R Visualization (Week9-10)

Tree Map of Top Stock Losses

```
# Ensure necessary libraries are installed
if (!require(treemap)) install.packages("treemap", dependencies=TRUE)
```

```
## Loading required package: treemap
```

```
## Warning: package 'treemap' was built under R version 4.3.3
```

```
if (!require(dplyr)) install.packages("dplyr", dependencies=TRUE)
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
if (!require(stringr)) install.packages("stringr", dependencies=TRUE)
```

```
## Loading required package: stringr
```

```
# Load the libraries
library(treemap)
library(dplyr)
library(stringr)

# Load the data
data <- read.csv("C:/Users/TheArchitect/Downloads/all-us-exchanges-volume-leaders-08-02-2024.csv")

# Inspect the data structure to verify column names and data
print(head(data))
```

```
##   Symbol                      Name   Last Change   X.Chg   High    Low    Volume
## 1   AMZN           Amazon.com Inc 167.90 -16.17  -8.78% 168.77 160.55 141240766
## 2   AMAT        Applied Materials 181.81 -14.49  -7.38% 189.57 179.63  12987746
## 3   PCOR Procore Technologies Inc  56.78 -10.44 -15.53%  56.95  49.46  15378653
```

```
## 4   META       Meta Platforms Inc 488.14  -9.60  -1.93% 501.14 476.15  23844764
## 5   TSLA                Tesla Inc 207.67  -9.19  -4.24% 216.13 205.78  81852469
## 6   MCHP     Microchip Technology  75.43  -8.94 -10.60%  79.64  74.60  17482164
##       Time
## 1 8/2/2024
## 2 8/2/2024
## 3 8/2/2024
## 4 8/2/2024
## 5 8/2/2024
## 6 8/2/2024
```

```
print(str(data))
```

```
## 'data.frame':    200 obs. of  9 variables:
##  $ Symbol: chr  "AMZN" "AMAT" "PCOR" "META" ...
##  $ Name  : chr  "Amazon.com Inc" "Applied Materials" "Procore Technologies Inc" "Meta Platforms Inc"
##  $ Last  : num  167.9 181.8 56.8 488.1 207.7 ...
##  $ Change: num  -16.17 -14.49 -10.44 -9.6 -9.19 ...
##  $ X.Chg : chr  "-8.78%" "-7.38%" "-15.53%" "-1.93%" ...
##  $ High  : num  169 190 57 501 216 ...
##  $ Low   : num  160.6 179.6 49.5 476.1 205.8 ...
##  $ Volume: int  141240766 12987746 15378653 23844764 81852469 17482164 18038021 35983648 29394145 11
##  $ Time  : chr  "8/2/2024" "8/2/2024" "8/2/2024" "8/2/2024" ...
## NULL
```

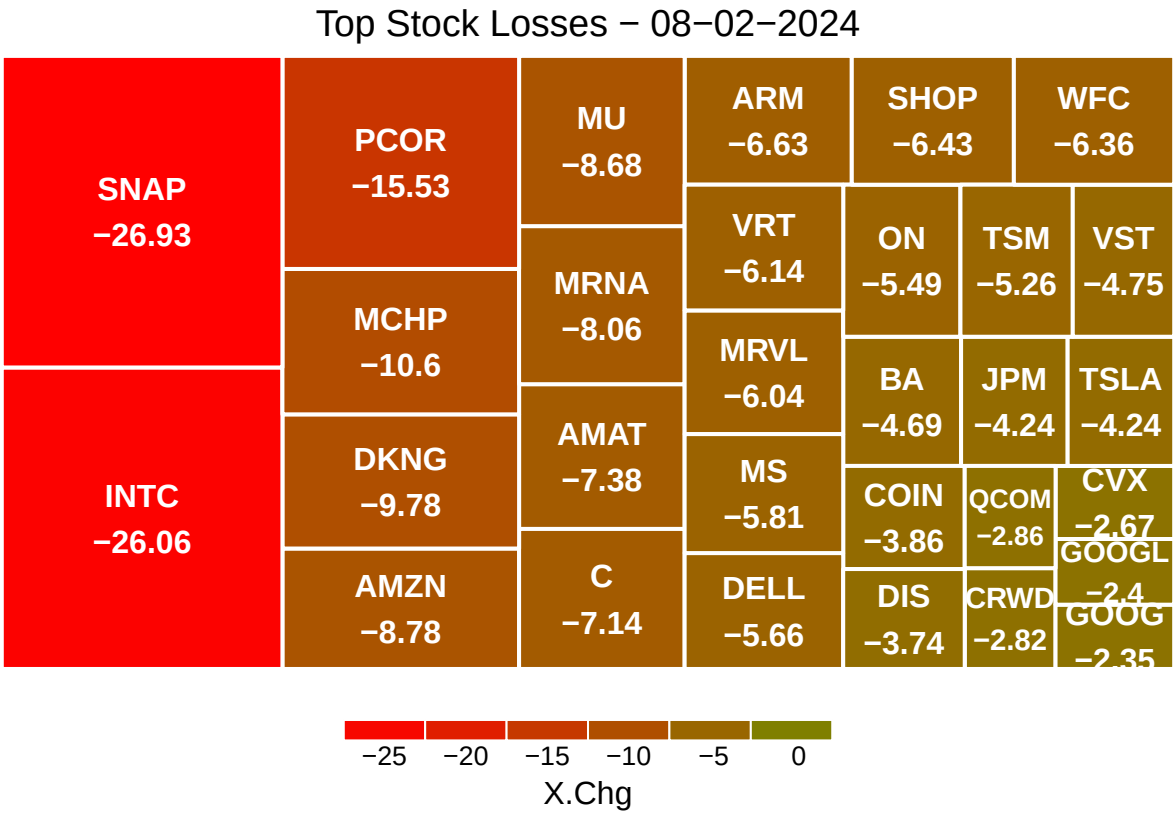
```
print(names(data))
```

```
## [1] "Symbol" "Name"   "Last"   "Change" "X.Chg"  "High"   "Low"    "Volume"
## [9] "Time"
```

```
# Ensure the X.Chg column exists and is correctly named
if("X.Chg" %in% names(data)) {
  # Convert X.Chg to numeric and remove the percentage sign
  data$`X.Chg` <- as.numeric(str_remove(data$`X.Chg`, "%"))

  # Check for NA values after conversion
  print(summary(data$`X.Chg`))

  # Filter out NA values if any
  data <- data %>% filter(!is.na(`X.Chg`))

  # Sort and filter the top 30 stock losses by X.Chg
  top_losses <- data %>% arrange(`X.Chg`) %>% head(30)

  # Use absolute values for sizes to avoid negative size error
  top_losses$AbsChg <- abs(top_losses$`X.Chg`)

  # Create labels combining Symbol and X.Chg
  top_losses$Label <- paste(top_losses$Symbol, round(top_losses$`X.Chg`, 2), sep = "\n")

  # Create the treemap
  treemap(
    top_losses,
    index = "Label",
    vSize = "AbsChg",
    vColor = "X.Chg",
```

```
    type = "value",
    title = "Top Stock Losses - 08-02-2024",
    palette = c("red", "green"),
    fontsize.labels = 12,
    fontcolor.labels = "white",
    border.col = "white",
    draw.legend = FALSE  # Hide the legend
  )
} else {
  print("Error: The column 'X.Chg' does not exist in the dataset.")
}
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -26.930  -7.550  -5.735  -7.043  -3.830  -1.930     168
```



Box Plot of Distribution of Price per Unit across Product Categories

```
# Install necessary packages if not already installed
if (!require("ggplot2")) install.packages("ggplot2", dependencies = TRUE)
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
if (!require("readxl")) install.packages("readxl", dependencies = TRUE)
```

```
## Loading required package: readxl
```

```r
if (!require("dplyr")) install.packages("dplyr", dependencies = TRUE)
if (!require("tidyr")) install.packages("tidyr", dependencies = TRUE)
```

```
## Loading required package: tidyr
```

```r
# Load necessary libraries
library(ggplot2)
library(readxl)
library(dplyr)
library(tidyr)

# Load the data
file_path <- "C:/Users/TheArchitect/Downloads/Adidas US Sales Datasets.xlsx"
data <- read_excel(file_path, skip = 5)  # Skip the first 5 rows if necessary
```

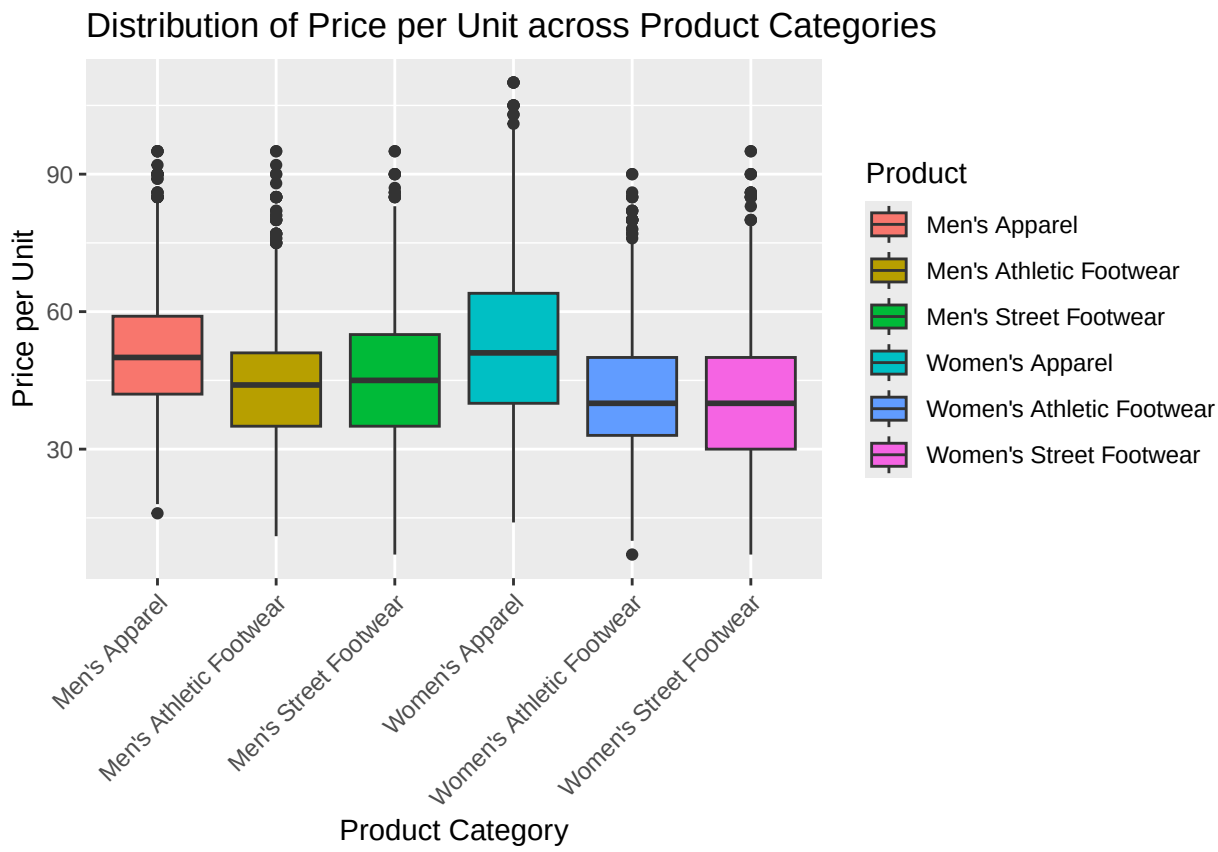
```
## New names:
## * `New York` -> `New York...5`
## * `New York` -> `New York...6`
```

```r
# Rename columns
colnames(data) <- c("Retailer", "Retailer_ID", "Invoice_Date", "Region", "State", "City", "Product",
                    "Price_Per_Unit", "Units_Sold", "Total_Sales", "Operating_Profit", "Operating_Margi
                    "Sales_Method")

# Convert columns to appropriate data types
data <- data %>%
  mutate(
    Price_Per_Unit = as.numeric(Price_Per_Unit),
    Units_Sold = as.numeric(Units_Sold),
    Total_Sales = as.numeric(gsub(",", "", Total_Sales)),
    Operating_Profit = as.numeric(gsub(",", "", Operating_Profit)),
    Operating_Margin = as.numeric(gsub("%", "", Operating_Margin)) / 100,
    Product = as.factor(Product)
  )

# Filter out rows where Price_Per_Unit is not a valid number
data <- data %>%
  filter(!is.na(Price_Per_Unit))

# Ensure 'Product' column has valid values and remove missing or invalid 'Product' entries
data <- data %>%
  filter(!is.na(Product) & Product != "")

# Group by product and calculate summary statistics
data_summary <- data %>%
  group_by(Product) %>%
  summarise(Price_Per_Unit = list(Price_Per_Unit)) %>%
  unnest(Price_Per_Unit)

# Check for unique values in 'Product' column
print(unique(data_summary$Product))
```

```
## [1] Men's Apparel             Men's Athletic Footwear
## [3] Men's Street Footwear     Women's Apparel
## [5] Women's Athletic Footwear Women's Street Footwear
```

```
## 6 Levels: Men's Apparel Men's Athletic Footwear ... Women's Street Footwear
```

```
# Create the box plot
ggplot(data_summary, aes(x = Product, y = Price_Per_Unit)) +
  geom_boxplot(aes(fill = Product)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Distribution of Price per Unit across Product Categories",
       x = "Product Category",
       y = "Price per Unit")
```



Bullet Chart of Writing Scores by State with Reading Targets and Math Benchmarks

```
# Install necessary packages if not already installed
if (!require("ggplot2")) install.packages("ggplot2", dependencies = TRUE)
if (!require("dplyr")) install.packages("dplyr", dependencies = TRUE)
if (!require("readr")) install.packages("readr", dependencies = TRUE)
```

```
## Loading required package: readr
```

```
# Load the libraries
library(ggplot2)
library(dplyr)
library(readr)

# Load the data
file_path <- "C:/Users/TheArchitect/Downloads/Education.csv"
data <- read_csv(file_path)
```

```
## Rows: 52 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): state
## dbl (6): reading, math, writing, percent_graduates_sat, pupil_staff_ratio, d...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

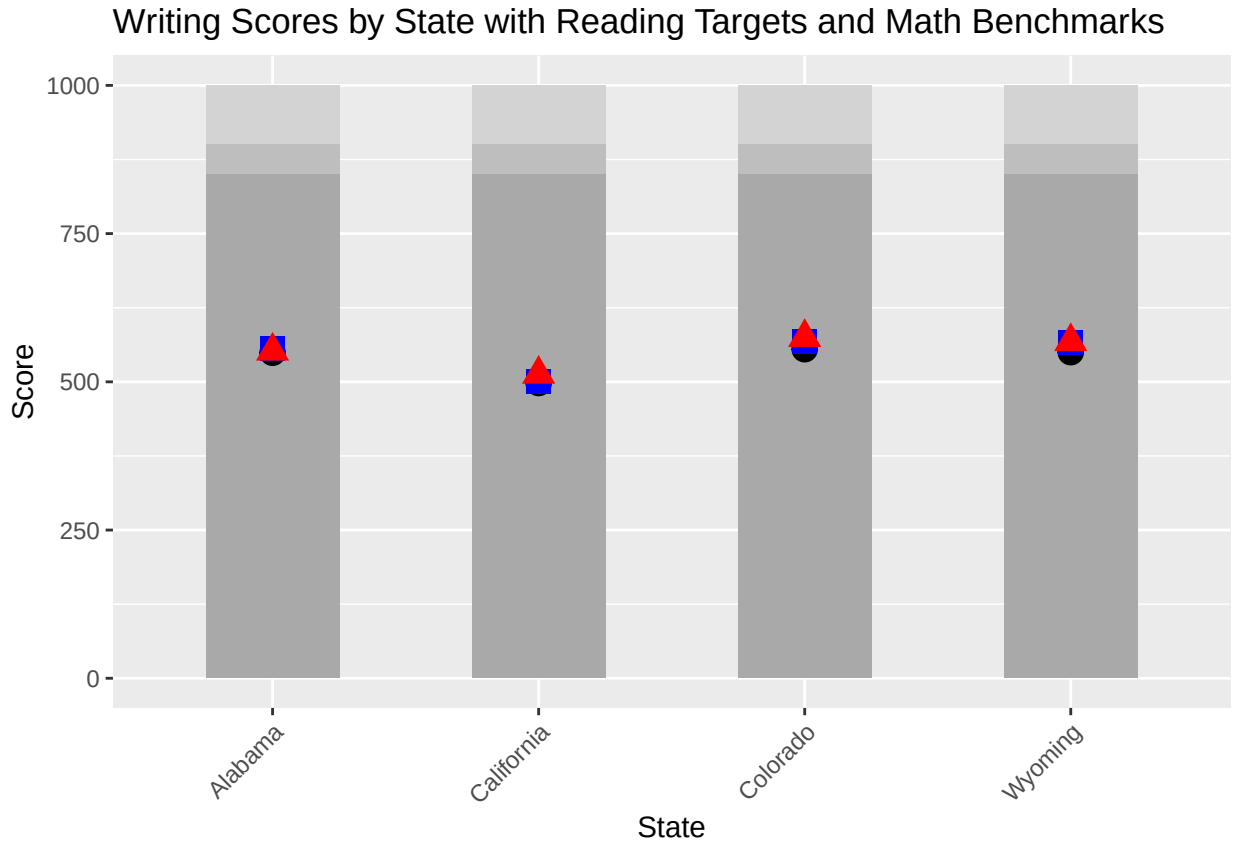
```
# Print the first few rows of the data to verify it's loaded correctly
print(head(data))
```

```
## # A tibble: 6 x 7
##   state         reading  math writing percent_graduates_sat pupil_staff_ratio
##   <chr>           <dbl> <dbl>   <dbl>                 <dbl>              <dbl>
## 1 United States     501   515     493                    46                7.9
## 2 Alabama           557   552     549                     7                6.7
## 3 Alaska            520   516     492                    46                7.9
## 4 Arizona           516   521     497                    26               10.4
## 5 Arkansas          572   572     556                     5                6.8
## 6 California        500   513     498                    49               10.9
## # i 1 more variable: dropout_rate <dbl>
```

```
# Filter the data for specific states
filtered_data <- data %>%
  filter(state %in% c("Colorado", "Wyoming", "Alabama", "California"))

# Print the filtered data to verify the filter is applied correctly
print(filtered_data)
```

```
## # A tibble: 4 x 7
##   state      reading  math writing percent_graduates_sat pupil_staff_ratio
##   <chr>        <dbl> <dbl>   <dbl>                 <dbl>              <dbl>
## 1 Alabama        557   552     549                     7                6.7
## 2 California     500   513     498                    49               10.9
## 3 Colorado       568   575     555                    20                8.1
## 4 Wyoming        567   568     550                     5                5.6
## # i 1 more variable: dropout_rate <dbl>
```

```
# Create the bullet chart
ggplot(filtered_data) +
  geom_bar(aes(x = state, y = 1000), stat = "identity", fill = "lightgrey", width = 0.5) +
  geom_bar(aes(x = state, y = 900), stat = "identity", fill = "grey", width = 0.5) +
  geom_bar(aes(x = state, y = 850), stat = "identity", fill = "darkgrey", width = 0.5) +
  geom_point(aes(x = state, y = writing), color = "black", size = 4) +
  geom_point(aes(x = state, y = reading), color = "blue", size = 4, shape = 15) +
  geom_point(aes(x = state, y = math), color = "red", size = 4, shape = 17) +
  labs(title = "Writing Scores by State with Reading Targets and Math Benchmarks",
       x = "State", y = "Score") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Histogram of Frequency of Units SOld

```
# Load necessary libraries
library(readxl)
library(ggplot2)
library(dplyr)

# Load the data
file_path <- "C:/Users/TheArchitect/Downloads/Adidas US Sales Datasets.xlsx"
data <- read_excel(file_path, col_names = FALSE)  # Load without assuming headers
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
## * `` -> `...9`
## * `` -> `...10`
## * `` -> `...11`
## * `` -> `...12`
## * `` -> `...13`
```

```
# Manually set the column names based on the known structure of your data
colnames(data) <- c("Index", "Retailer", "Retailer_ID", "Invoice_Date", "Region", "State",
```

```
                    "City", "Product", "Price_Per_Unit", "Units_Sold", "Total_Sales",
                    "Operating_Profit", "Operating_Margin", "Sales_Method")

# Ensure 'Units_Sold' is numeric
data <- data %>%
  mutate(Units_Sold = as.numeric(Units_Sold))
```

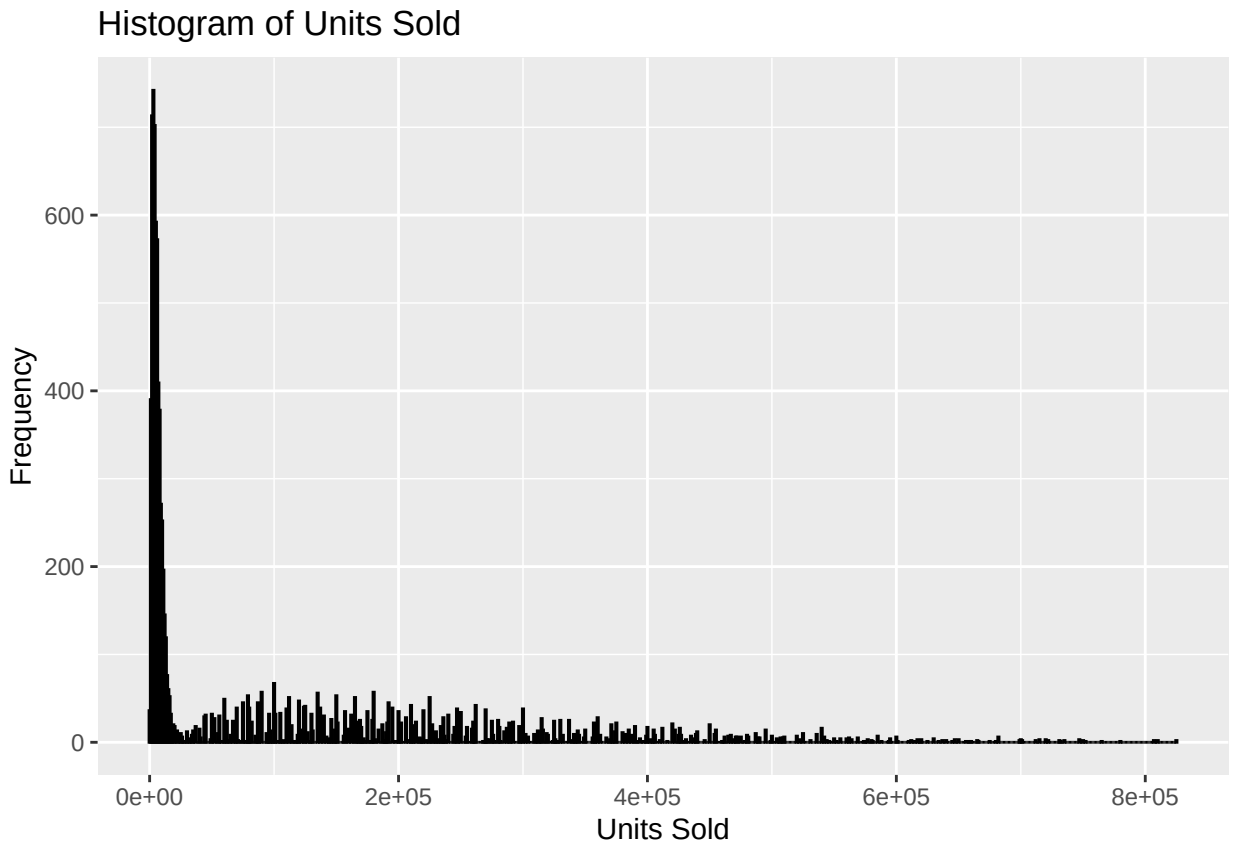
```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Units_Sold = as.numeric(Units_Sold)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
# Check the first few rows to verify the data
print(head(data))
```

```
## # A tibble: 6 x 13
##   Index       Retailer       Retailer_ID Invoice_Date Region State City  Product
##   <chr>       <chr>          <chr>       <chr>        <chr>  <chr> <chr> <chr>
## 1 <NA>        Adidas Sales ~ <NA>        <NA>         <NA>   <NA>  <NA>  <NA>
## 2 <NA>        <NA>           <NA>        <NA>         <NA>   <NA>  <NA>  <NA>
## 3 <NA>        <NA>           <NA>        <NA>         <NA>   <NA>  <NA>  <NA>
## 4 Retailer    Retailer ID    Invoice Da~ Region       State  City  Prod~ Price ~
## 5 Foot Locker 1185732        43831       Northeast    New Y~ New ~ Men'~ 50
## 6 Foot Locker 1185732        43832       Northeast    New Y~ New ~ Men'~ 50
## # i 5 more variables: Price_Per_Unit <chr>, Units_Sold <dbl>,
## #   Total_Sales <chr>, Operating_Profit <chr>, Operating_Margin <chr>
```

```
# Create a histogram for Units Sold
ggplot(data, aes(x = Units_Sold)) +
  geom_histogram(binwidth = 1000, fill = "blue", color = "black") +
  labs(title = "Histogram of Units Sold", x = "Units Sold", y = "Frequency")
```

```
## Warning: Removed 4 rows containing non-finite outside the scale range
## (`stat_bin()`).
```


Histogram of Units Sold
Frequency
600
400
200
0
0e+00
2e+05
4e+05
6e+05
8e+05
Units Sold

DSC640_Week9-10_Moussadeq

August 8, 2024

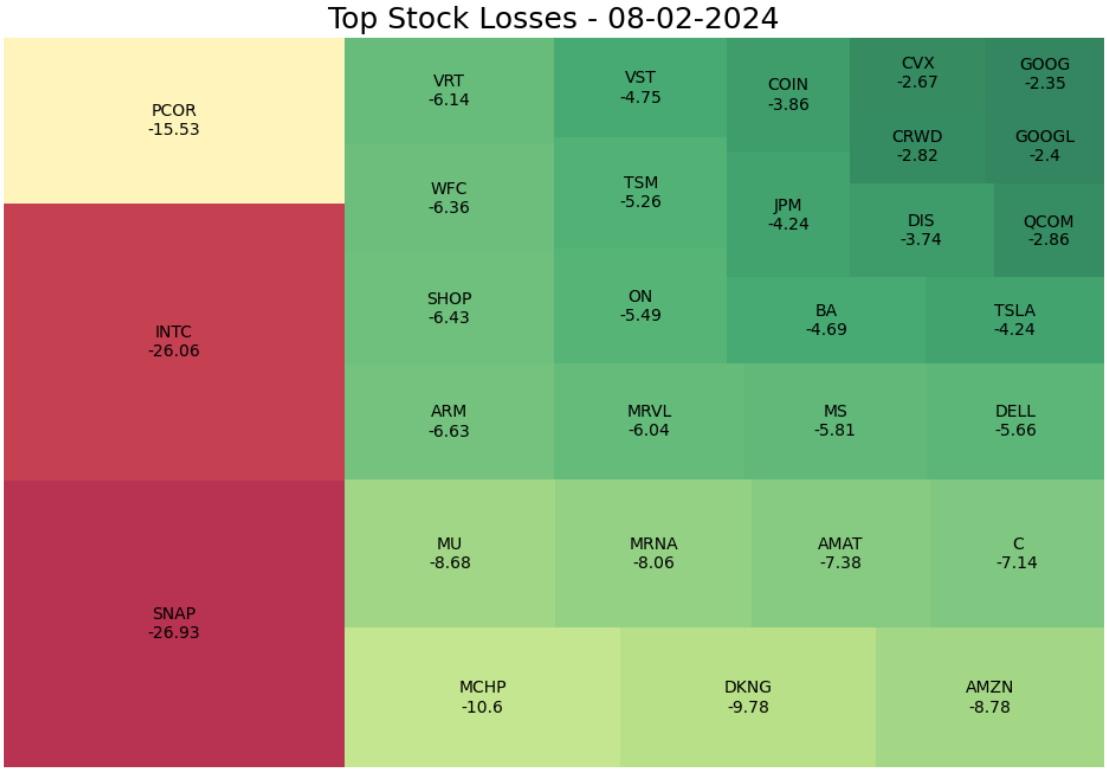
Python Visualizations

Treemap of Top Stock Losses

[10]:
```
import pandas as pd
import matplotlib.pyplot as plt
import squarify
import numpy as np


# Load the data
data = pd.read_csv("all-us-exchanges-volume-leaders-08-02-2024.csv")

# Convert %Chg to numeric and remove the percentage sign
data['%Chg'] = data['%Chg'].str.rstrip('%').astype('float')

# Check for NA values after conversion
print(data['%Chg'].describe())

# Filter out NA values if any
data = data.dropna(subset=['%Chg'])

# Sort and filter the top 30 stock losses by %Chg
top_losses = data.nsmallest(30, '%Chg')

# Use absolute values for sizes to avoid negative size error
sizes = top_losses['%Chg'].abs()  # Use absolute values for sizes

# Create labels combining Symbol and %Chg
labels = top_losses['Symbol'] + '\n' + (top_losses['%Chg']).round(2).astype(str)

# Create a color map
cmap = plt.cm.RdYlGn  # Red to Green
norm = plt.Normalize(vmin=min(top_losses['%Chg']), vmax=max(top_losses['%Chg']))
colors = [cmap(norm(value)) for value in top_losses['%Chg']]

# Plotting the treemap
plt.figure(figsize=(12, 8))
```

```
squarify.plot(sizes=sizes, label=labels, alpha=.8, color=colors)
plt.axis('off')
plt.title('Top Stock Losses - 08-02-2024', fontsize=18)
plt.show()
```

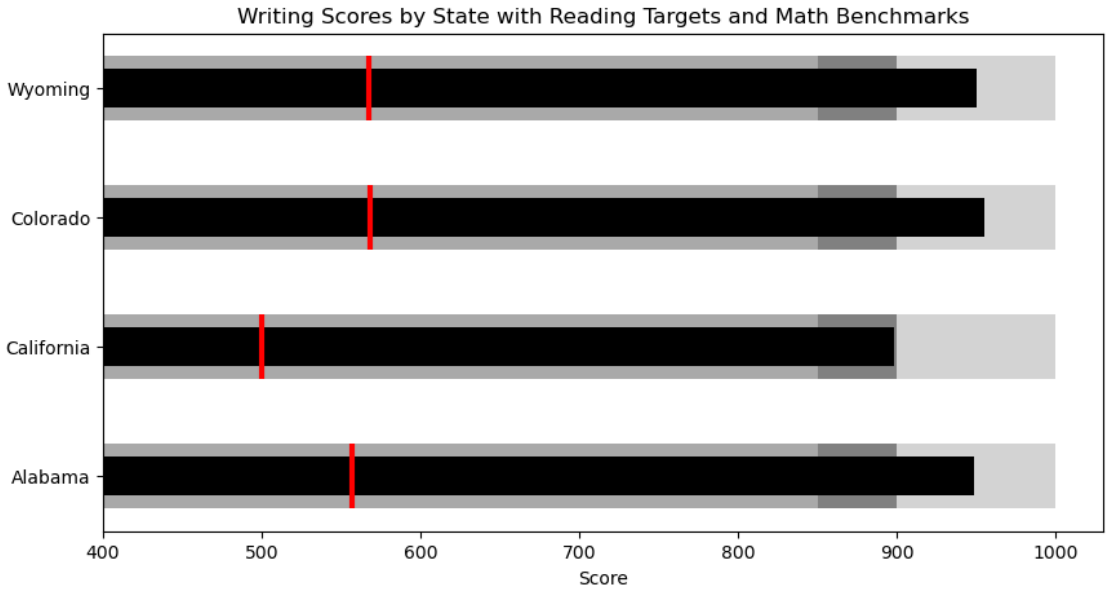
```
count    32.000000
mean     -7.043125
std       5.859877
min     -26.930000
25%      -7.550000
50%      -5.735000
75%      -3.830000
max      -1.930000
Name: %Chg, dtype: float64
```



Bullet Chart of Writing Scores by State with Reading Targets and Math Benchmarks

[41]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = "C:/Users/TheArchitect/Downloads/Education.csv"
```

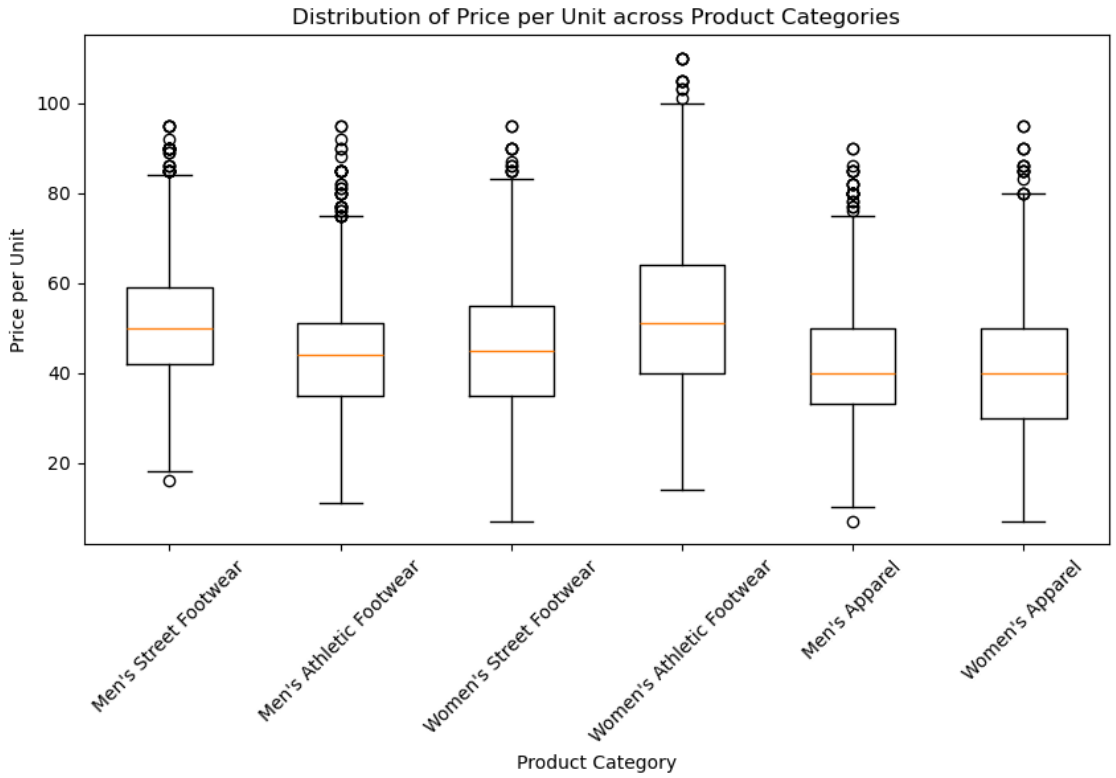
```
data = pd.read_csv(file_path)

# Filter the data for specific states
filtered_data = data[data['state'].isin(['Colorado', 'Wyoming', 'Alabama',␣
 ↪'California'])]

# Extract columns
states = filtered_data['state']
writing_scores = filtered_data['writing']
reading_scores = filtered_data['reading']
math_scores = filtered_data['math']

# Create the bullet chart
fig, ax = plt.subplots(figsize=(10, 5))

for i, state in enumerate(states):
    # Plot the benchmark range (Math scores)
    ax.barh(i, 600, left=400, color='lightgrey', edgecolor='none', height=0.5)
    ax.barh(i, 500, left=400, color='grey', edgecolor='none', height=0.5)
    ax.barh(i, 450, left=400, color='darkgrey', edgecolor='none', height=0.5)

    # Plot the actual score (Writing scores)
    ax.barh(i, writing_scores.iloc[i], left=400, color='black', height=0.3)

    # Plot the target (Reading scores)
    ax.vlines(reading_scores.iloc[i], i - 0.25, i + 0.25, color='red',␣
 ↪linewidth=3)

# Set labels
ax.set_yticks(range(len(states)))
ax.set_yticklabels(states)
ax.set_xlabel('Score')
ax.set_title('Writing Scores by State with Reading Targets and Math Benchmarks')

plt.show()
```



Box Plot of Distribution of Price per Unit across Product Categories

[37]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data from the Excel file
file_path = "C:/Users/TheArchitect/Downloads/Adidas US Sales Datasets.xlsx"
df = pd.read_excel(file_path)

# Correct the column names (using row 4 in this case)
df.columns = ["Index", "Retailer", "Retailer_ID", "Invoice_Date", "Region",␣
 ↪"State", "City", "Product", "Price_Per_Unit",
              "Units_Sold", "Total_Sales", "Operating_Profit",␣
 ↪"Operating_Margin", "Sales_Method"]

# Convert 'Price_Per_Unit' to numeric, coercing errors to NaN
df['Price_Per_Unit'] = pd.to_numeric(df['Price_Per_Unit'], errors='coerce')

# Drop rows with NaN values in 'Price_Per_Unit' to clean the data
df = df.dropna(subset=['Price_Per_Unit'])

# Extracting the price per unit data grouped by product category
price_per_unit_data = df.groupby('Product')['Price_Per_Unit'].apply(list)

# Box plot
plt.figure(figsize=(10, 5))
plt.boxplot(price_per_unit_data)
```

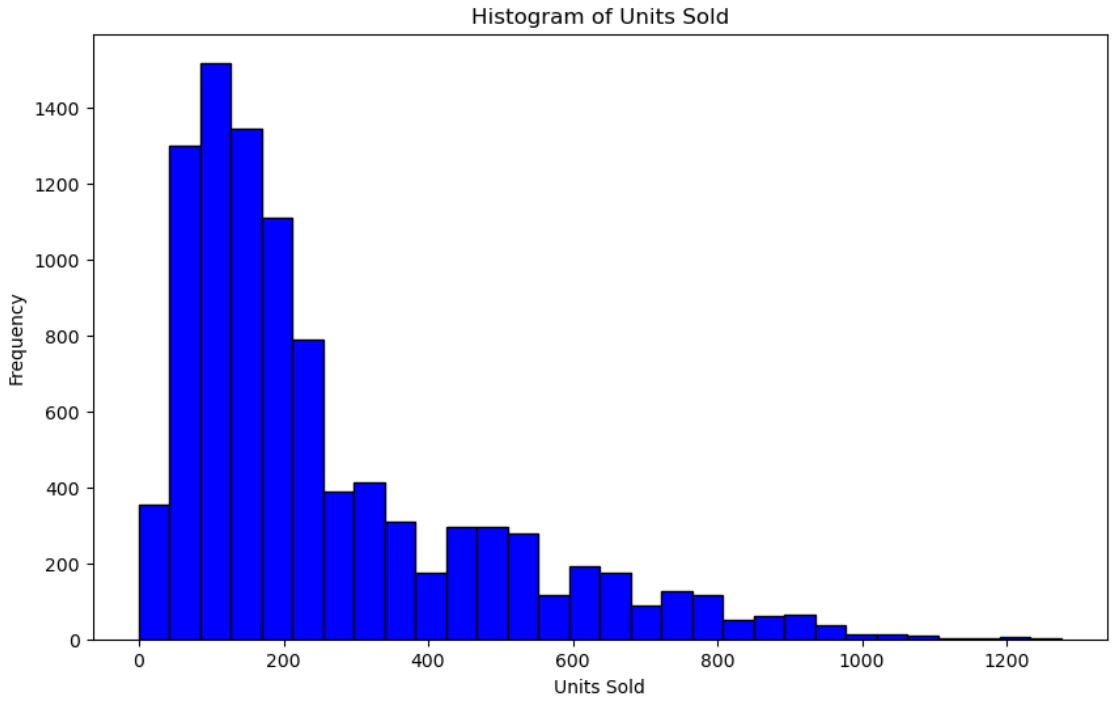
```
plt.xlabel('Product Category')
plt.ylabel('Price per Unit')
plt.title('Distribution of Price per Unit across Product Categories')
plt.xticks(rotation=45)

product_categories = df['Product'].unique()
plt.xticks(range(1, len(product_categories) + 1), product_categories)

plt.show()
```



Histogram of Units Sold

[44]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = "C:/Users/TheArchitect/Downloads/Adidas US Sales Datasets.xlsx"
data = pd.read_excel(file_path, header=None)  # Load the data without assuming␣
 ↪headers

# Rename the columns
```

```
data.columns = ["Index", "Retailer", "Retailer_ID", "Invoice_Date", "Region",␣
 ↪"State",
                "City", "Product", "Price_Per_Unit", "Units_Sold",␣
 ↪"Total_Sales",
                "Operating_Profit", "Operating_Margin", "Sales_Method"]

# Convert the 'Units_Sold' column to numeric, coercing any errors to NaN
data['Units_Sold'] = pd.to_numeric(data['Units_Sold'], errors='coerce')

# Drop rows where 'Units_Sold' is NaN (this happens when conversion fails)
data = data.dropna(subset=['Units_Sold'])

# Create a histogram for Units Sold
plt.figure(figsize=(10, 6))
plt.hist(data['Units_Sold'], bins=30, color='blue', edgecolor='black')
plt.title('Histogram of Units Sold')
plt.xlabel('Units Sold')
plt.ylabel('Frequency')
plt.show()
```



[ ]:

Airline Safety Facts

1M
Total Motor Vehicle Death (Last 30 Years)

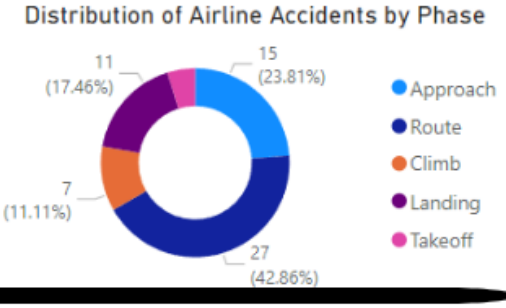
25K
Total Airplane Fatalities (Last 30 Years)

1970 Fatalities → 4.77 Per Million Passengers

2021 Fatalities → 0.05 Per Million Passengers

Significant Increase in Air passengers, Yet Safer Skies

Understanding When Accidents Happen can Help Predict and Prevent Accidents



2023 Airline Safety

2023 has set a new record for airline safety with one accident every 1.26 million flights, 72 fatalities, and 30 total accidents with only one fatal accident

New Safety Record Achieved!

Airplane Accidents Causes

- Pilott Error
- Mechanical Failure
- Weather Conditions
- Air Traffic Control
- Terrorism

Most Airplane Accidents can be Prevented with Better Training, Predictive Modeling of Weather, and Enhaced Security

The Future Looks Bright with Detachable Cabin that can Land Safely in Water or Land in Case of an An emergency

Resources:

- www.iata/en/pressroom/2024-releases/2024-02-28-01/
- www.boredpanda.com/detachable-cabin-plane-craswwwh-aircraft-safety-vladimir-tatarenko
- www.airlinesafetydsc640sm.blogpost.com/2024/07/how-aviation-safety-is-always-getting.html

About the Infographic

The infographic "Airline Safety Facts" was designed to effectively communicate the progress in aviation safety to a public audience. The visualizations were carefully selected to provide a clear, concise, simple, and impactful presentation of key data points. The comparison between 1970 and 2021 fatalities was highlighted using a line graph to emphasize the significant reduction in fatalities, demonstrating the effectiveness of safety measures over time. A pie chart was used to show the distribution of airline accidents by phase, as this visualization makes it easy to see where most accidents occur, thus highlighting areas for improvement. The safety of air travel compared to motor vehicles card was used to show the huge difference in fatalities, making it easy for the public to read, avoiding charts that may take longer to process.

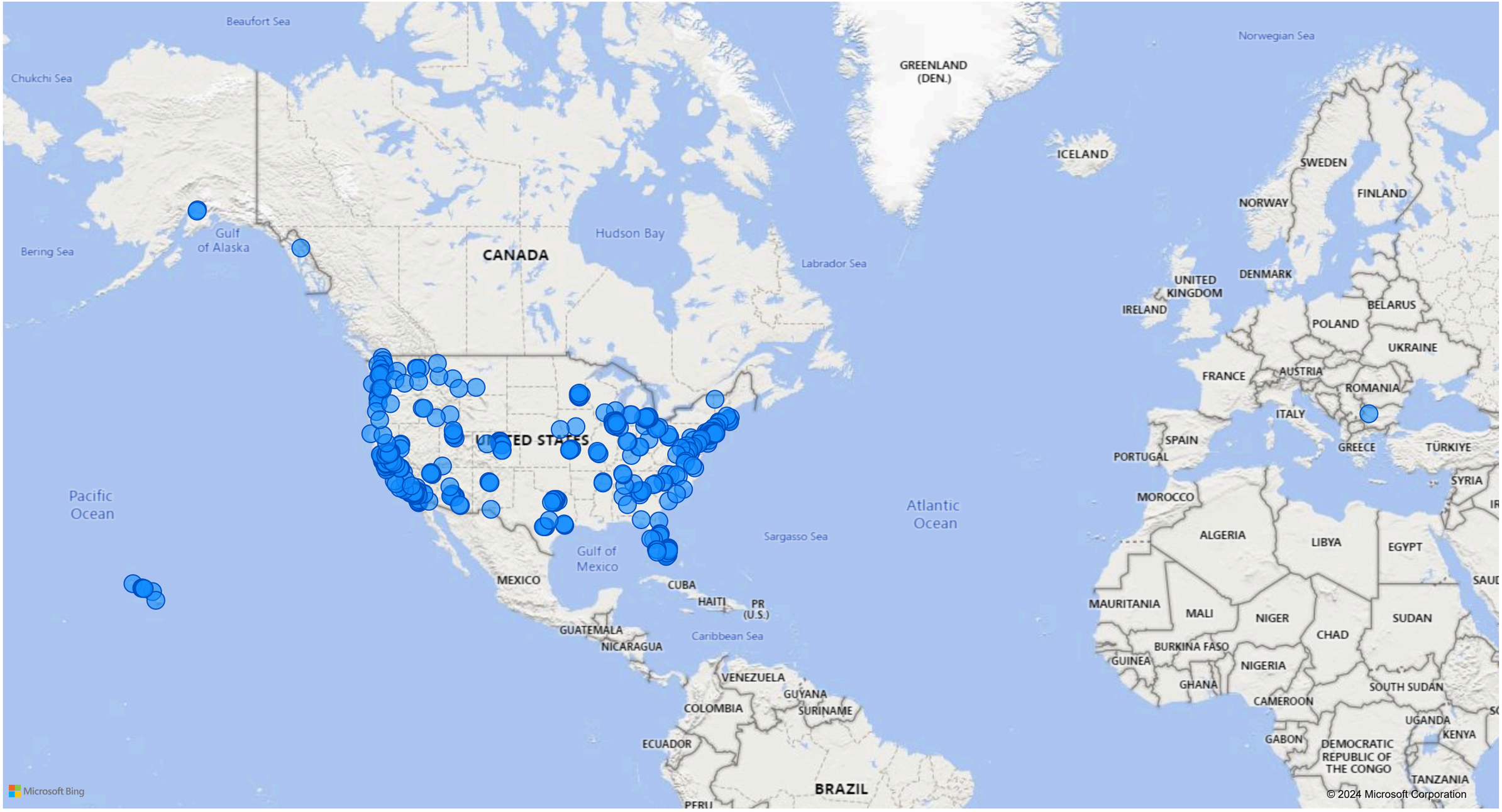
The inclusion of a card highlighting the 2023 airline safety record serves to emphasize the remarkable progress in aviation safety, showcasing the industry's achievements and demonstrating how far safety improvements have come. The top causes of airplane accidents section were chosen to explain that most of the accidents can be avoided with better training, predictive modeling of weather, and enhanced security. Finally, the futuristic detachable cabin concept was included to showcase innovative solutions that promise even greater safety in the future.

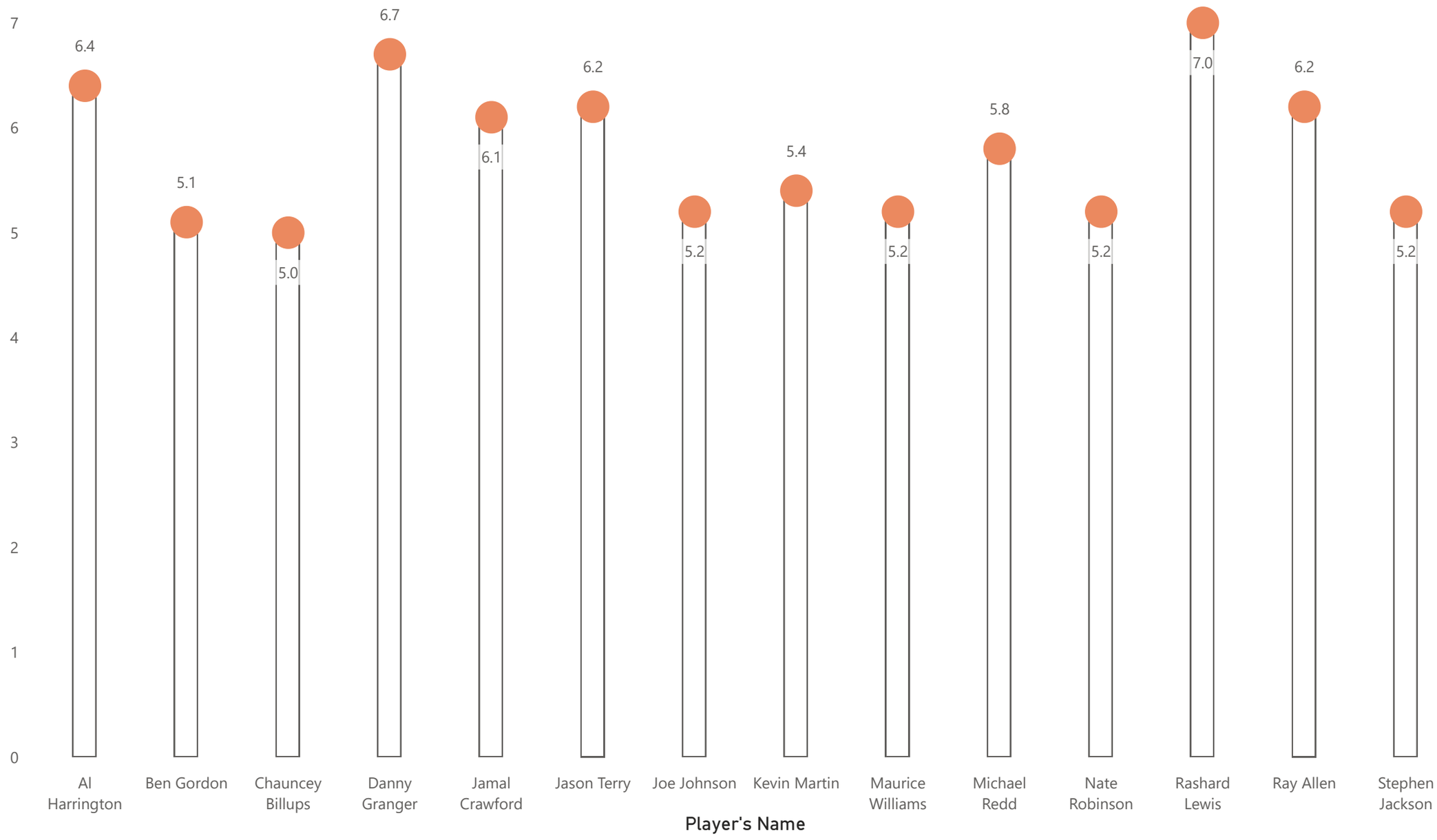
This public-facing infographic differs from internal presentations, as it focuses on reassuring and informing the public without delving into technical and long details. Ethical considerations included ensuring accuracy, avoiding sensationalism, and presenting data transparently to build trust and hope for an ever-evolving future of airplane aviation.

NBA Points Per Game of 2008(PowerBI)

Name	3PA	TRB	TO	STL	PTS	PF	ORB	MIN	G	FTP	FTM	FTA	FGP	FGM	FGA	DRB	BLK	AST	3PP	3PM
Al Harrington	6.40	6.20	2.20	1.20	20.10	3.10	1.40	34.90	73	0.79	3.20	4.00	0.44	7.30	16.60	4.90	0.30	1.40	0.36	2.30
Al Jefferson	0.10	11.00	1.80	0.80	23.10	2.80	3.40	36.60	50	0.74	3.70	5.00	0.50	9.70	19.50	7.50	1.70	1.60	0.00	0.00
Allen Iverson	1.70	3.00	2.60	1.50	17.50	1.50	0.50	36.70	57	0.78	4.80	6.10	0.42	6.10	14.60	2.50	0.10	5.00	0.28	0.50
Amare Stoudemire	0.10	8.10	2.80	0.90	21.40	3.10	2.20	36.80	53	0.84	6.10	7.30	0.54	7.60	14.10	5.90	1.10	2.00	0.43	0.10
Andre Iguodala	3.20	5.70	2.70	1.60	18.80	1.90	1.10	39.80	82	0.72	4.60	6.40	0.47	6.60	14.00	4.60	0.40	5.30	0.31	1.00
Antawn Jamison	3.90	8.90	1.50	1.20	22.20	2.70	2.40	38.20	81	0.75	4.20	5.60	0.47	8.30	17.80	6.50	0.30	1.90	0.35	1.40
Ben Gordon	5.10	3.50	2.40	0.90	20.70	2.20	0.60	36.60	82	0.86	4.00	4.70	0.46	7.30	16.00	2.80	0.30	3.40	0.41	2.10
Brandon Roy	2.80	4.70	1.90	1.10	22.60	1.60	1.30	37.20	78	0.82	5.30	6.50	0.48	8.10	16.90	3.40	0.30	5.10	0.38	1.10
Carmelo Anthony	2.60	6.80	3.00	1.10	22.80	3.00	1.60	34.50	66	0.79	5.60	7.10	0.44	8.10	18.30	5.20	0.40	3.40	0.37	1.00
Caron Butler	3.10	6.20	3.10	1.60	20.80	2.50	1.80	38.60	67	0.86	5.10	6.00	0.45	7.30	16.20	4.40	0.30	4.30	0.31	1.00
Chauncey Billups	5.00	3.00	2.20	1.20	17.70	2.00	0.40	35.30	79	0.91	5.30	5.80	0.42	5.20	12.40	2.60	0.20	6.40	0.41	2.10
Chris Bosh	0.60	10.00	2.30	0.90	22.70	2.50	2.80	38.10	77	0.82	6.50	8.00	0.49	8.00	16.40	7.20	1.00	2.50	0.25	0.20
Chris Paul	2.30	5.50	3.00	2.80	22.80	2.70	0.90	38.50	78	0.87	5.80	6.70	0.50	8.10	16.10	4.70	0.10	11.00	0.36	0.80
Corey Maggette	1.90	5.50	2.40	0.90	18.60	3.80	1.00	31.10	51	0.82	6.70	8.10	0.46	5.70	12.40	4.60	0.20	1.80	0.25	0.50
Danny Granger	6.70	5.10	2.50	1.00	25.80	3.10	0.70	36.20	67	0.88	6.00	6.90	0.45	8.50	19.10	4.40	1.40	2.70	0.40	2.70
David West	0.30	8.50	2.10	0.60	21.00	2.70	2.10	39.30	76	0.88	4.80	5.50	0.47	8.00	17.00	6.40	0.90	2.30	0.24	0.10
Deron Williams	3.30	2.90	3.40	1.10	19.40	2.00	0.40	36.90	68	0.85	4.80	5.60	0.47	6.80	14.50	2.50	0.30	10.70	0.31	1.00
Devin Harris	3.20	3.30	3.10	1.70	21.30	2.40	0.40	36.10	69	0.82	7.20	8.80	0.44	6.60	15.10	2.90	0.20	6.90	0.29	0.90
Dirk Nowitzki	2.10	8.40	1.90	0.80	25.90	2.20	1.10	37.70	81	0.89	6.00	6.70	0.48	9.60	20.00	7.30	0.80	2.40	0.36	0.80
Dwight Howard	0.00	13.80	3.00	1.00	20.60	3.40	4.30	35.70	79	0.59	6.40	10.70	0.57	7.10	12.40	9.60	2.90	1.40	0.00	0.00
Dwyane Wade	3.50	5.00	3.40	2.20	30.20	2.30	1.10	38.60	79	0.77	7.50	9.80	0.49	10.80	22.00	3.90	1.30	7.50	0.32	1.10
Jamal Crawford	6.10	3.00	2.30	0.90	19.70	1.40	0.40	38.10	65	0.87	4.60	5.30	0.41	6.40	15.70	2.60	0.20	4.40	0.36	2.20
Jason Terry	6.20	2.40	1.60	1.30	19.60	1.90	0.50	33.60	74	0.88	2.70	3.00	0.46	7.30	15.80	1.90	0.30	3.40	0.37	2.30
Joe Johnson	5.20	4.40	2.50	1.10	21.40	2.20	0.80	39.50	79	0.83	3.80	4.60	0.44	7.80	18.00	3.60	0.20	5.80	0.36	1.90
John Salmons	3.80	4.20	2.10	1.10	18.30	2.30	0.70	37.50	79	0.83	3.60	4.40	0.47	6.50	13.80	3.50	0.30	3.20	0.42	1.60
Josh Howard	3.20	5.10	1.70	1.10	18.00	2.60	1.10	31.90	52	0.78	3.30	4.20	0.45	6.80	15.10	3.90	0.60	1.60	0.35	1.10
Kevin Durant	3.10	6.50	3.00	1.30	25.30	1.80	1.00	39.00	74	0.86	6.10	7.10	0.48	8.90	18.80	5.50	0.70	2.80	0.42	1.30
Kevin Martin	5.40	3.60	2.90	1.20	24.60	2.30	0.60	38.20	51	0.87	9.00	10.30	0.42	6.70	15.90	3.00	0.20	2.70	0.42	2.30
Kobe Bryant	4.10	5.20	2.60	1.50	26.80	2.30	1.10	36.20	82	0.86	5.90	6.90	0.47	9.80	20.90	4.10	0.50	4.90	0.35	1.40
LaMarcus Aldridge	0.30	7.50	1.50	1.00	18.10	2.60	2.90	37.10	81	0.78	3.20	4.10	0.48	7.40	15.30	4.60	1.00	1.90	0.25	0.10
LeBron James	4.70	7.60	3.00	1.70	28.40	1.70	1.30	37.70	81	0.78	7.30	9.40	0.49	9.70	19.90	6.30	1.10	7.20	0.34	1.60
Maurice Williams	5.20	3.40	2.20	0.90	17.80	2.70	0.60	35.00	81	0.91	2.60	2.80	0.47	6.50	13.90	2.90	0.10	4.10	0.44	2.30
Michael Redd	5.80	3.20	1.60	1.10	21.20	1.40	0.70	36.40	33	0.81	4.00	4.90	0.46	7.50	16.60	2.50	0.10	2.70	0.37	2.10
Monta Ellis	1.00	4.30	2.70	1.60	19.00	2.70	0.60	35.60	25	0.83	3.10	3.80	0.45	7.80	17.20	3.80	0.30	3.70	0.31	0.30
Nate Robinson	5.20	3.90	1.90	1.30	17.20	2.80	1.30	29.90	74	0.84	3.40	4.00	0.44	6.10	13.90	2.60	0.10	4.10	0.33	1.70
O.J. Mayo	4.60	3.80	2.80	1.10	18.50	2.50	0.70	38.00	82	0.88	3.00	3.40	0.44	6.90	15.60	3.10	0.20	3.20	0.38	1.80
Pau Gasol	0.00	9.60	1.90	0.60	18.90	2.10	3.20	37.10	81	0.78	4.20	5.40	0.57	7.30	12.90	6.40	1.00	3.50	0.50	0.00
Paul Pierce	3.80	5.60	2.80	1.00	20.50	2.70	0.70	37.40	81	0.83	5.70	6.80	0.46	6.70	14.60	5.00	0.30	3.60	0.39	1.50
Rashard Lewis	7.00	5.70	2.00	1.00	17.70	2.50	1.20	36.20	79	0.84	2.80	3.40	0.44	6.10	13.80	4.60	0.60	2.60	0.40	2.80

Costco Stores in The USA (PowerBI)




Top Three-Point Attempts Scorers in The NBA (PowerBi)
7
6
5
4
3
2
1
0
6.4
5.1
5.0
6.7
6.1
6.2
5.2
5.4
5.2
5.8
5.2
7.0
6.2
5.2
Al Harrington
Ben Gordon
Chauncey Billups
Danny Granger
Jamal Crawford
Jason Terry
Joe Johnson
Kevin Martin
Maurice Williams
Michael Redd
Nate Robinson
Rashard Lewis
Ray Allen
Stephen Jackson
Player's Name

Week7&8 R Visualizations

Said Moussadeq

2024-07-25

R Visualization

Lollilop Chart of Top 10 Players by Points per Game (PTS) in 2008

```
# Set the working directory
setwd("C:/Users/TheArchitect/Downloads")

# Load the data
data <- read.csv('ppg2008.csv')

# Clean column names
colnames(data) <- trimws(colnames(data))

# Inspect the PTS column to ensure it's convertible to numeric
print(data$PTS)
```

```
##  [1] 30.2 28.4 26.8 25.9 25.8 25.3 24.6 23.1 22.8 22.8 22.7 22.6 22.2 22.0 21.4
## [16] 21.4 21.3 21.2 21.0 20.8 20.8 20.8 20.7 20.7 20.6 20.5 20.1 19.7 19.7 19.6
## [31] 19.6 19.4 19.3 19.0 18.9 18.9 18.8 18.6 18.5 18.3 18.3 18.2 18.1 18.0 17.8
## [46] 17.8 17.7 17.7 17.5 17.2
```

```
# Convert PTS column to numeric, handle non-numeric values
data$PTS <- as.numeric(gsub("[^0-9.]", "", data$PTS))

# Print the PTS column after conversion
print(data$PTS)
```

```
##  [1] 30.2 28.4 26.8 25.9 25.8 25.3 24.6 23.1 22.8 22.8 22.7 22.6 22.2 22.0 21.4
## [16] 21.4 21.3 21.2 21.0 20.8 20.8 20.8 20.7 20.7 20.6 20.5 20.1 19.7 19.7 19.6
## [31] 19.6 19.4 19.3 19.0 18.9 18.9 18.8 18.6 18.5 18.3 18.3 18.2 18.1 18.0 17.8
## [46] 17.8 17.7 17.7 17.5 17.2
```

```
# Select top 10 players by PTS
top_10_pts <- data[order(-data$PTS), ][1:10, ]

# Print the selected top 10 players
print(top_10_pts)
```

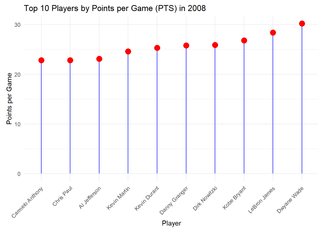
```
##                Name  G  MIN  PTS  FGM  FGA   FGP FTM  FTA   FTP X3PM X3PA  X3PP
## 1       Dwyane Wade  79 38.6 30.2 10.8 22.0 0.491 7.5  9.8 0.765  1.1  3.5 0.317
## 2     LeBron James  81 37.7 28.4  9.7 19.9 0.489 7.3  9.4 0.780  1.6  4.7 0.344
## 3      Kobe Bryant  82 36.2 26.8  9.8 20.9 0.467 5.9  6.9 0.856  1.4  4.1 0.351
## 4    Dirk Nowitzki  81 37.7 25.9  9.6 20.0 0.479 6.0  6.7 0.890  0.8  2.1 0.359
## 5    Danny Granger  67 36.2 25.8  8.5 19.1 0.447 6.0  6.9 0.878  2.7  6.7 0.404
## 6     Kevin Durant  74 39.0 25.3  8.9 18.8 0.476 6.1  7.1 0.862  1.3  3.1 0.422
## 7    Kevin Martin  51 38.2 24.6  6.7 15.9 0.420 9.0 10.3 0.867  2.3  5.4 0.415
## 8     Al Jefferson  50 36.6 23.1  9.7 19.5 0.497 3.7  5.0 0.738  0.0  0.1 0.000
## 9       Chris Paul  78 38.5 22.8  8.1 16.1 0.503 5.8  6.7 0.868  0.8  2.3 0.364
## 10 Carmelo Anthony  66 34.5 22.8  8.1 18.3 0.443 5.6  7.1 0.793  1.0  2.6 0.371
##    ORB DRB  TRB  AST STL BLK  TO  PF
## 1  1.1 3.9  5.0  7.5 2.2 1.3 3.4 2.3
## 2  1.3 6.3  7.6  7.2 1.7 1.1 3.0 1.7
## 3  1.1 4.1  5.2  4.9 1.5 0.5 2.6 2.3
## 4  1.1 7.3  8.4  2.4 0.8 0.8 1.9 2.2
## 5  0.7 4.4  5.1  2.7 1.0 1.4 2.5 3.1
## 6  1.0 5.5  6.5  2.8 1.3 0.7 3.0 1.8
## 7  0.6 3.0  3.6  2.7 1.2 0.2 2.9 2.3
## 8  3.4 7.5 11.0  1.6 0.8 1.7 1.8 2.8
## 9  0.9 4.7  5.5 11.0 2.8 0.1 3.0 2.7
## 10 1.6 5.2  6.8  3.4 1.1 0.4 3.0 3.0
```

```
# Install ggplot2 if not already installed
if (!require("ggplot2")) install.packages("ggplot2", dependencies = TRUE)
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(ggplot2)

# Create the lollipop chart
ggplot(top_10_pts, aes(x = reorder(Name, PTS), y = PTS)) +
  geom_segment(aes(x = Name, xend = Name, y = 0, yend = PTS), color = "blue") +
  geom_point(color = "red", size = 4) +
  theme_minimal() +
  labs(title = "Top 10 Players by Points per Game (PTS) in 2008", x = "Player", y = "Points per Game") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Heat Map

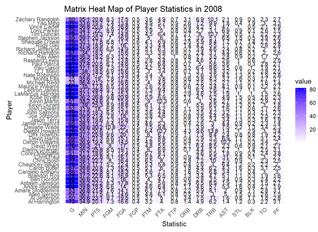
```
# Set the working directory
setwd("C:/Users/TheArchitect/Downloads")

# Load the data
data <- read.csv('ppg2008.csv')

# Clean column names
colnames(data) <- trimws(colnames(data))

# Print column names to check their exact names
print(colnames(data))
```

```
##  [1] "Name" "G"    "MIN"  "PTS"  "FGM"  "FGA"  "FGP"  "FTM"  "FTA"  "FTP"
## [11] "X3PM" "X3PA" "X3PP" "ORB"  "DRB"  "TRB"  "AST"  "STL"  "BLK"  "TO"
## [21] "PF"
```

```
# Load necessary libraries
if (!require("ggplot2")) install.packages("ggplot2", dependencies = TRUE)
if (!require("reshape2")) install.packages("reshape2", dependencies = TRUE)
```

```
## Loading required package: reshape2
```

```
library(ggplot2)
library(reshape2)

# Select relevant columns for the heatmap
selected_columns <- c("Name", "G", "MIN", "PTS", "FGM", "FGA", "FGP", "FTM", "FTA", "FTP", "3PM", "3PA", "3PP",
"ORB", "DRB", "TRB", "AST", "STL", "BLK", "TO", "PF")

# Ensure all selected columns exist
selected_columns <- selected_columns[selected_columns %in% colnames(data)]

# Subset the data with selected columns
heatmap_data <- data[, selected_columns]

# Melt the data for ggplot
melted_data <- melt(heatmap_data, id.vars = "Name")
# Create the heatmap with labels
ggplot(melted_data, aes(x = variable, y = Name)) +
  geom_tile(aes(fill = value), color = "white") +
  geom_text(aes(label = round(value, 1)), size = 3) +
  scale_fill_gradient(low = "white", high = "blue") +
  theme_minimal() +
  labs(title = "Matrix Heat Map of Player Statistics in 2008", x = "Statistic", y = "Player") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Spatial Chart of Costco Locations in the US

```
# Set the working directory
setwd("C:/Users/TheArchitect/Downloads")

# Set CRAN mirror
options(repos = c(CRAN = "https://cran.rstudio.com"))

# Install and load necessary libraries
if (!require("leaflet")) install.packages("leaflet")
```

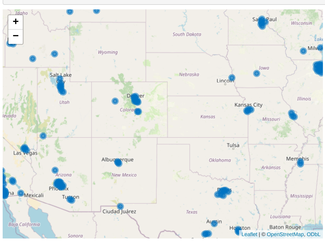
```
## Loading required package: leaflet
```

```
## Warning: package 'leaflet' was built under R version 4.3.3
```

```
if (!require("htmltools")) install.packages("htmltools")
```

```
## Loading required package: htmltools
```

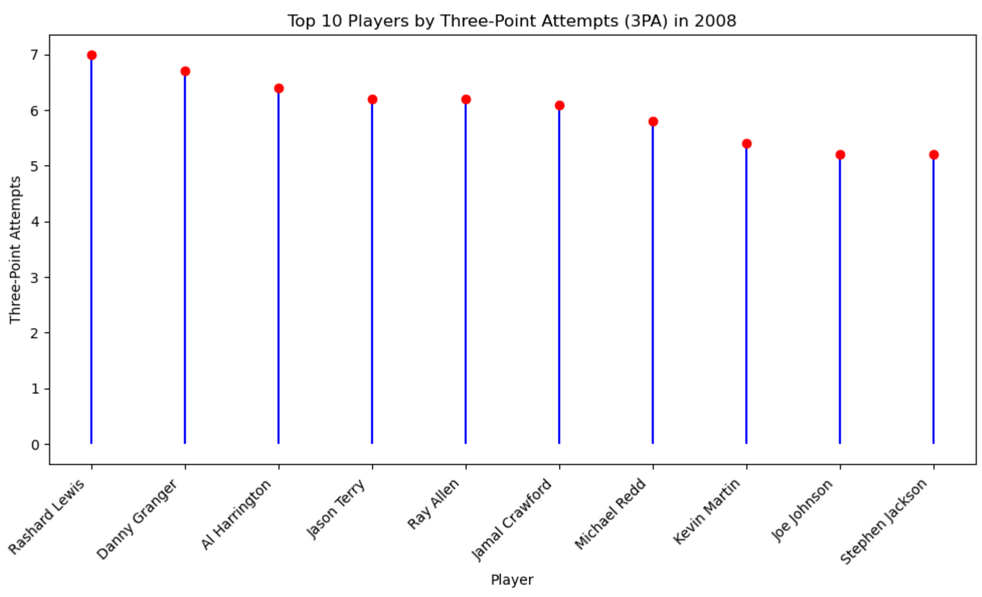
```
library(leaflet)
library(htmltools)


# Load the data
data <- read.csv('costcos-geocoded.csv')

# Initialize a leaflet map centered around the average coordinates
m <- leaflet(data) %>%
  addTiles() %>%
  setView(lng = mean(data$Longitude), lat = mean(data$Latitude), zoom = 5)

# Add points to the map
m <- m %>%
  addCircleMarkers(lng = ~Longitude, lat = ~Latitude, popup = ~Address, radius = 5, fill = TRUE, fillOpacity = 0.
7, color = "#0073C2")

# Add a title using the htmltools package
library(htmltools)
title <- tags$div(
  tags$style(HTML("
    .leaflet-control.map-title {
      position: fixed;
      left: 50%;
      text-align: center;
      padding-left: 10px;
      padding-right: 10px;
      background: white;
      color: black;
      font-weight: bold;
      font-size: 20px;
    }
  ")),
  HTML("<div class='leaflet-control map-title'>Costco Locations in the US</div>")
)

# Add the title to the map
m <- m %>% addControl(title, position = "topright", className = "map-title")

# Print the map
m
```



Python Visualizations

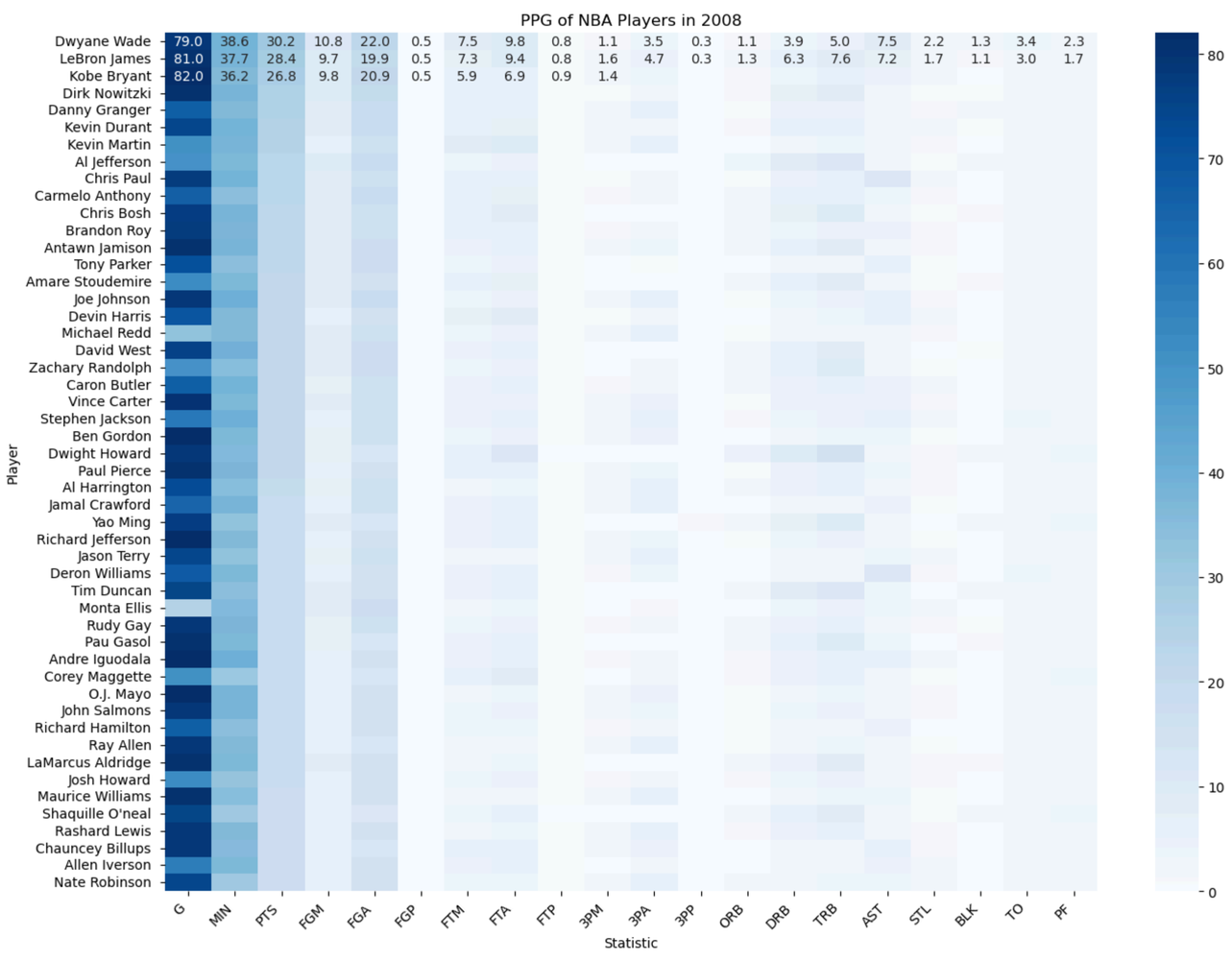
Lollilop Chart of Top 10 Players by Three-Point Attempts (3PA) in 2008 {-}

In [11]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
file_path = 'ppg2008.csv'
data = pd.read_csv(file_path)

# Remove trailing spaces from column names
data.columns = data.columns.str.strip()

# Select top 10 players by 3PA
top_10_3pa = data.nlargest(10, '3PA')

# Create the lollipop chart with a vertical orientation
plt.figure(figsize=(10, 6))
plt.vlines(x=top_10_3pa['Name'], ymin=0, ymax=top_10_3pa['3PA'], color='blue')
plt.plot(top_10_3pa['Name'], top_10_3pa['3PA'], 'o', color='red')
plt.title('Top 10 Players by Three-Point Attempts (3PA) in 2008')
plt.ylabel('Three-Point Attempts')
plt.xlabel('Player')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Matrix Heat Map of Player Statistics in 2008 {-}

In [19]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the data
file_path = 'ppg2008.csv'
data = pd.read_csv(file_path)

# Remove trailing spaces from column names
data.columns = data.columns.str.strip()

# Select relevant columns for the heatmap
heatmap_data = data[["Name", "G", "MIN", "PTS", "FGM", "FGA", "FGP", "FTM", "FTA", "FTP", "3PM", "3PA", "3PP", "ORB", "DRB", "TRB", "AST", "STL", "BLK", "TO", "PF"]]

# Set the player names as the index
heatmap_data.set_index("Name", inplace=True)

# Create the heatmap with labels
plt.figure(figsize=(14, 10))
sns.heatmap(heatmap_data, cmap="Blues", annot=True, fmt=".1f")
plt.title("PPG of NBA Players in 2008")
plt.xlabel("Statistic")
plt.ylabel("Player")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



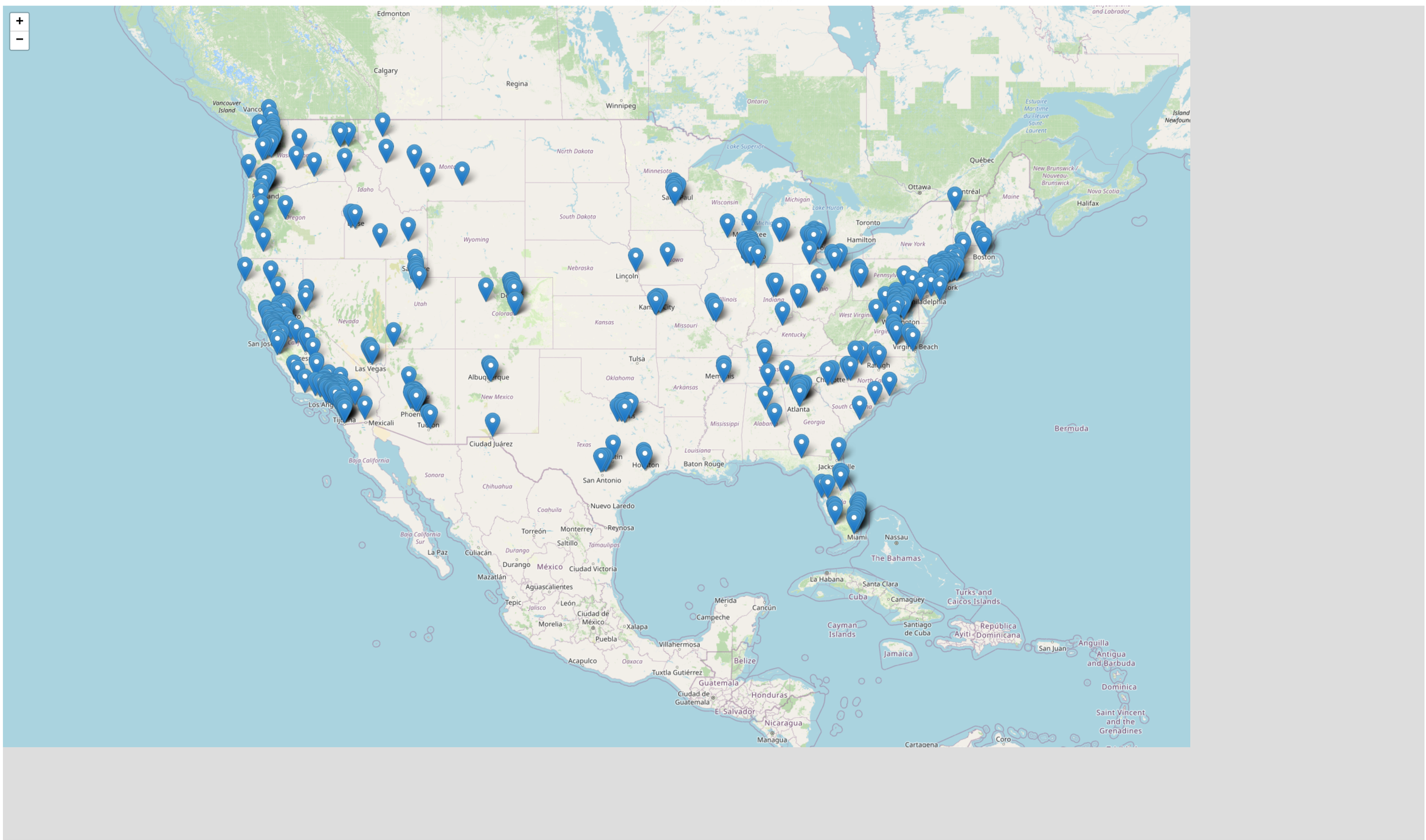
Spatial Chart of Costco Locations in the US {-}

In [21]:
```
import pandas as pd
import geopandas as gpd
import folium

# Load the data
file_path = 'costcos-geocoded.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the data
print(data.head())
```

```
                    Address        City    State    Zip Code   Latitude  \
0   1205 N. Memorial Parkway  Huntsville  Alabama  35801-5930  34.743095
1       3650 Galleria Circle      Hoover  Alabama  35244-2346  33.377649
2     8251 Eastchase Parkway  Montgomery  Alabama       36117  32.363889
3  5225 Commercial Boulevard      Juneau   Alaska  99801-7210  58.359200
4       330 West Dimond Blvd   Anchorage   Alaska  99515-1950  61.143266

    Longitude
0  -86.600955
1  -86.812420
2  -86.150884
3 -134.483000
4 -149.884217
```

In [25]:
```
# Initialize a map centered around the average coordinates
m = folium.Map(location=[data['Latitude'].mean(), data['Longitude'].mean()], zoom_start=5)

# Add points to the map
for idx, row in data.iterrows():
    folium.Marker([row['Latitude'], row['Longitude']], popup=row['Address']).add_to(m)

# Add a title to the map using HTML
title_html = '''
     <h3 align="center" style="font-size:20px"><b>Costco Locations in the US</b></h3>
     '''
m.get_root().html.add_child(folium.Element(title_html))

# Save the map to an HTML file
m.save('costco_locations_map.html')
m
```

Out[25]:


Costco Locations in the US
Edmonton
Calgary
Regina
Winnipeg
Vancouver Island
Ontario
Québec
Ottawa
Toronto
Hamilton
Boston
Halifax
Lincoln
Las Vegas
Albuquerque
Tulsa
Atlanta
Ciudad Juárez
San Antonio
Baton Rouge
Bermuda
The Bahamas
Miami
Nassau
La Habana
Cuba
Cayman Islands
Jamaica
Turks and Caicos Islands
Santiago de Cuba
San Juan
Anguilla
Antigua and Barbuda
Dominica
Saint Vincent and the Grenadines
Mérida
Cancún
Campeche
Villahermosa
Belize
Guatemala
Honduras
El Salvador
Nicaragua
Managua
Ciudad de México
Monterrey
Reynosa
Nuevo Laredo
Torreón
Saltillo
Durango
Mazatlán
Culiacán
La Paz
Guadalajara
Puebla
Acapulco
Oaxaca
Tuxtla Gutiérrez
Ciudad Victoria
Aguascalientes
León
Morelia
Xalapa
Tepic
Cartagena
Coro

In [ ]: