02 Other Plots

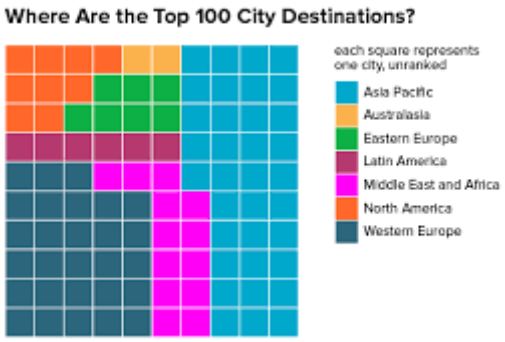
```
import matplotlib.pyplot as plt
```

Waffle Charts

- A waffle chart is a type of visualization that uses a grid of squares to represent data in a percentage format
- Each square in the grid represents a fraction of the total, making it easy to visualize parts-to-whole relationships
- Waffle charts are particularly useful for displaying progress towards a goal, comparing different categories, or illustrating proportions

Structure of a Waffle Chart

- **Grid Layout** : Typically a 10x10 grid (100 squares) is used, where each square represents 1% of the total. This can be adjusted based on the desired precision and visual preference.
- **Colored Squares** : Different colors are used to represent different categories or segments.
- **Labels** : Categories and percentages are often labeled for clarity



Steps to create Waffle Charts

- The first step into creating a waffle chart is determing the proportion of each category with respect to the total
- The second step is defining the overall size of the `waffle` chart
- The third step is using the proportion of each category to determe it respective number of tiles

- The fourth step is creating a matrix that resembles the `waffle` chart and populating it
- Map the `waffle` chart matrix into a visual
- Prettify the chart
- Create a legend and add it to chart

```
# Define the data
data = {
    'Asia Pacific': 32,
    'Australasia': 6,
    'Eastern Europe': 6,
    'Latin America': 8,
    'Middle East and Africa': 9,
    'North America': 10,
    'Western Europe': 29
}

# Define the color for each category
colors = {
    'Asia Pacific': '#17becf',
    'Australasia': '#ff7f0e',
    'Eastern Europe': '#2ca02c',
    'Latin America': '#d62728',
    'Middle East and Africa': '#9467bd',
    'North America': '#8c564b',
    'Western Europe': '#1f77b4'
}

# Create the waffle chart
fig, ax = plt.subplots(figsize=(10, 6))

# Number of rows and columns
n_rows = 10
n_cols = 10

# Plot each category
category_index = 0
current_pos = 0
for category, count in data.items():
    for i in range(count):
        row = current_pos // n_cols
        col = current_pos % n_cols
        rect = patches.Rectangle((col, n_rows - 1 - row), 1, 1,
facecolor=colors[category], edgecolor='black', linewidth=1)
        ax.add_patch(rect)
        current_pos += 1
    category_index += 1

# Create legend
```

```
legend_handles = [patches.Patch(color=color, label=category) for category,
color in colors.items()]
ax.legend(handles=legend_handles, loc='upper right', bbox_to_anchor=(1.3,
1))

# Set limits and aspect
ax.set_xlim(0, n_cols)
ax.set_ylim(0, n_rows)
ax.set_aspect('equal')

# Remove axes
ax.set_xticks([])
ax.set_yticks([])

plt.title('Where Are the Top 100 City Destinations?')
plt.show()
```

WordClouds

- A word cloud (or tag cloud) is a visual representation of text data, where the size of each word indicates its frequency or importance within a given dataset
- Words that appear more frequently in the source text are displayed in larger fonts, while less common words are shown in smaller sizes
- Word clouds are often used to highlight prominent themes, topics, or keywords in various types of content
- **Social media posts :** To visualize popular topics or sentiments
- **Survey responses :** To summarize open-ended feedback
- **Articles or essays :** To highlight main ideas or topics
- Creating a word cloud typically involves text processing steps like tokenization (splitting text into words), removing common stop words (like "the," "and," etc.), and counting the occurrences of each word

```
from wordcloud import WordCloud
```

```
# Sample text
text = """
Artificial intelligence is transforming industries. Machine learning is a
key component of AI.
Many companies are adopting AI technologies to enhance efficiency and
innovation.
"""

# Create a word cloud
wordcloud = WordCloud(width=800, height=400,
```

```
background_color='white').generate(text)

# Display the word cloud using matplotlib
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')  # Hide the axes
plt.show()
```

Regression plots

- Graphical representations that illustrate the relationship between a dependent variable and one or more independent variables using regression analysis
- These plots help visualize how the dependent variable changes as the independent variable(s) vary, allowing for better understanding and interpretation of the underlying patterns in the data
- Types of Regression plots
  - Scatter Plot with Regression Line
  - Multiple Regression Plots
  - Residual Plots