Heatmaps

Estimated Time: 20 minutes

Objectives

In this reading, you will explore various aspects of text analysis, including:

- Describing heatmaps, their features and applications
- Creating Seaborn heatmaps
- Customizing Seaborn heatmaps

What are heatmaps

Heatmaps visually display data using color gradients on a two-dimensional grid. They are commonly used to visualize the magnitude of a phenomenon across different categories or variables. In a heatmap, each cell in the grid represents a value from the dataset, and its color intensity reflects the magnitude of that value.

Key features of heatmaps

Here are some key characteristics of heatmaps:

1. **Color encoding**: Heatmaps use color gradients to encode data values. Typically, a color scale is chosen where low values are represented by lighter colors (for example, shades of yellow or green), and high values are represented by darker colors (for example, shades of red or blue). The specific color scheme can vary depending on the data and the context of visualization.

2. **Matrix representation**: Heatmaps are often presented as matrices, with rows and columns corresponding to different categories, variables, or dimensions in the dataset. Each cell in the matrix represents a value, and its color indicates the magnitude of that value.

3. **Applications**: Heatmaps find diverse applications across multiple domains. They are commonly used for:

    - Visualizing correlation or relationships between variables.
    - Identifying patterns, clusters, or anomalies in data.
    - Displaying spatial distributions or density of events.
    - Monitoring changes over time in time-series data.
    - Highlighting areas of interest in geographic or spatial data.

4. **Interpretation**: Interpretation of a heatmap involves analyzing the color intensity of cells and understanding how it relates to the underlying data. Users can identify trends, outliers, or areas of interest by examining the patterns in the heatmap.

5. **Customization**: Heatmaps can be customized in terms of color schemes, labeling, scaling, and other visual elements as per the requirements of the analysis and enhance readability.

Overall, heatmaps provide an effective and intuitive way to explore and communicate patterns and trends in data, making them a popular tool in data visualization and analysis.

Applications of heatmaps

Some common applications of Heatmaps include:

1. **Web analytics**: Analyzing user behavior on websites, identifying popular areas and interaction patterns.

2. **Biological sciences**: Visualizing gene expression data, protein interactions, and DNA sequences to understand biological processes.

3. **Epidemiology**: Mapping disease outbreaks, identifying hotspots, and tracking the spread of infectious diseases.

4. **Finance**: Visualizing stock market performance, asset correlations, and sector-wise trends for informed investment decisions.

5. **GIS**: Mapping spatial data like population density, crime rates, and environmental factors for urban planning and decision-making.

6. **Sports analytics**: Analyzing player performance, movement patterns, and game strategies for improving team performance.

7. **Marketing and retail**: Analyzing consumer behavior, optimizing store layouts, and product placements to enhance sales.

8. **Healthcare**: Analyzing patient data, healthcare utilization patterns, and disease prevalence for resource allocation and public health interventions.

9. **Climate studies**: Visualizing temperature anomalies, precipitation patterns, and climate change indicators for understanding climate variations.

10. **Transportation planning**: Mapping traffic congestion, transportation demand, and transit accessibility for optimizing transportation infrastructure.

Creating Seaborn heatmaps

Let's now create a simple Seaborn heatmap.

Explanation of parameters:

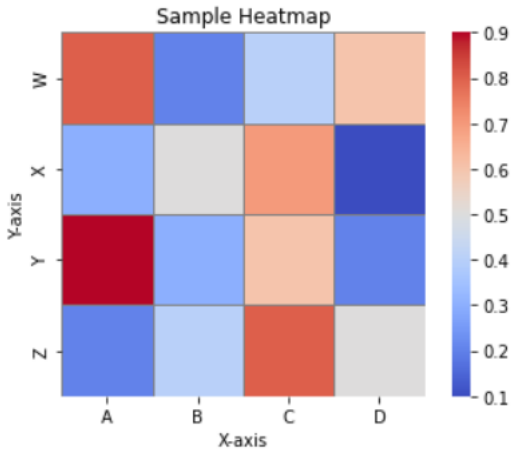
- `data`: The input data array for the heatmap.
- `annot`: If True, write the data value in each cell.
- `cmap`: The colormap to be used for coloring cells.
- `linewidths`: Width of the lines that will divide each cell.
- `linecolor`: Color of the lines that will divide each cell.
- `cbar`: If True, draw the color bar.
- `cbar_kws`: Additional keyword arguments to pass to the color bar creation function, here used to set the orientation of the color bar.
- `square`: If True, make the cells square-shaped.
- `fmt`: String formatting code to use when adding annotations. Here, '.2f' means two decimal places.

- `xticklabels` and `yticklabels`: Labels for the x-axis and y-axis ticks, respectively.

These parameters help customize the appearance and content of the heatmap according to your preferences and the characteristics of your data.

```
import seaborn as sns
import matplotlib.pyplot as plt
### Sample data for the heatmap
data = [
    [0.8, 0.2, 0.4, 0.6],
    [0.3, 0.5, 0.7, 0.1],
    [0.9, 0.3, 0.6, 0.2],
    [0.2, 0.4, 0.8, 0.5]
]
### Create the heatmap using Seaborn
sns.heatmap(data,
            annot=False,
            cmap='coolwarm',
            linewidths=0.5,
            linecolor='gray',
            cbar=True,
            cbar_kws={"orientation": "vertical"},
            square=True,
            fmt='.2f',
            xticklabels=['A', 'B', 'C', 'D'],
            yticklabels=['W', 'X', 'Y', 'Z'])
### Set title and labels
plt.title('Sample Heatmap')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
### Display the heatmap
plt.show()
```

Output:



Explanation:

The output of the Seaborn heatmap is a grid of colored squares, where each square represents a cell in the input data array. Each square is shaded with a color gradient based on the values in the corresponding cells of the input data array. In this example, the colormap 'coolwarm' is used, which ranges from cool (blue) to warm (red) colors. Higher values are represented by warmer colors, while lower values are represented by cooler colors.

Customization of Seaborn heatmaps

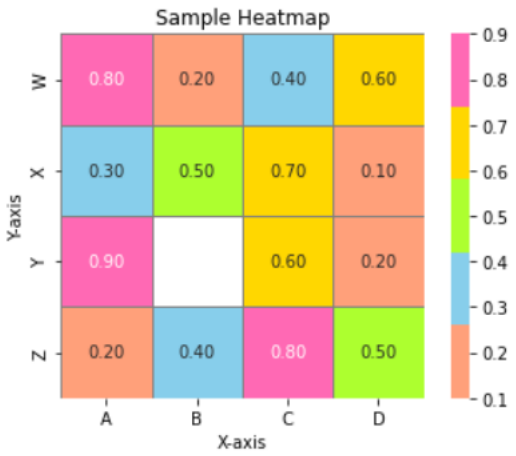
Colour customization, data annotation, data masking

Now, let's add color customization, annotation and masking to the heatmap. We will specify custom color values using the `colors` parameter. Here's the modified code:

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
# Sample data for the heatmap
data = [
    [0.8, 0.2, 0.4, 0.6],
    [0.3, 0.5, 0.7, 0.1],
    [0.9, 0.3, 0.6, 0.2],
    [0.2, 0.4, 0.8, 0.5]
]
# Define custom colors
colors = ["#FFA07A", "#87CEEB", "#ADFF2F", "#FFD700", "#FF69B4"]
# Create a mask array
mask = np.zeros_like(data)
mask[2, 1] = True  # Masking the value at row 2, column 1
# Create the heatmap using Seaborn with masking
sns.heatmap(data,
            annot=True,               # Add data annotations
            cmap=colors,              # Set custom colors
            linewidths=0.5,
            linecolor='gray',
            cbar=True,
            cbar_kws={"orientation": "vertical"},
            square=True,
            fmt='.2f',
            mask=mask,                # Apply the mask
            xticklabels=['A', 'B', 'C', 'D'],
```

```
                yticklabels=['W', 'X', 'Y', 'Z'])
    # Set title and labels
    plt.title('Sample Heatmap')
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    # Display the heatmap
    plt.show()
```

Output:



Explanation:

Now, let's have a look at the output of the customized seaborn heatmap is explained below:

1. **Customized color**: The heatmap uses custom colors specified in the `colors` list. Each color corresponds to a range of values in the data. For example, warmer colors like orange and red represent higher values, while cooler colors like green and blue represent lower values.
2. **Data annotation**: Each cell in the heatmap contains an annotation (data value) representing the corresponding value in the input data array. These annotations provide additional context and make it easier to interpret the heatmap.
3. **Masking**: In the heatmap, certain cells are masked based on the mask array. In this, the cell at row 2, column 1 is masked, meaning its value is not shown in the heatmap. Masking is useful for highlighting specific data points or areas of interest while excluding others.

Overall, the customized heatmap effectively visualizes the provided data with enhanced color representation, annotations for data clarity, and masking for focusing on specific areas of interest. It provides a clear and informative visualization of the underlying data patterns and relationships.

Conclusion

In conclusion, heatmaps serve as powerful tools for visualizing and analyzing data across various domains. Through color encoding, matrix representation, and customization options, heatmaps enable users to effectively explore patterns, trends, and relationships within their datasets.

This reading covered the fundamentals of heatmaps, including their definition, key features, and common applications. Additionally, it delved into creating and customizing heatmaps using the Seaborn library in Python, showcasing the process step by step with explanations of each parameter used.

By understanding how to create and customize heatmaps, users can effectively communicate insights from their data, whether it's in web analytics, biological sciences, finance, GIS, sports analytics, or any other field. Whether it's visualizing correlations, identifying outliers, or tracking changes over time, heatmaps offer a versatile and intuitive approach to data exploration and analysis.

Through the exploration of heatmaps and their applications, users can unlock valuable insights from their datasets, leading to informed decision-making and deeper understanding of complex phenomena. Therefore, becoming proficient in creating heatmap visualizations can be a valuable skill for individuals working in data analysis and visualization.

Author(s)

Geetika Pal