Histograms And Box Plots: Takeaways

Syntax

- Creating a frequency distribution:

```
norm_reviews['Fandango_RatingValue'].value_counts()
```

- Creating a histogram:

```
ax.hist(norm_reviews['Fandango_RatingValue'])
```

- Specifing the lower and upper range of bins within a histogram:

```
ax.hist(norm_reviews['Fandango_RatingValue'], range=(0,5))
```

- Setting y-axis limits:

```
ax.set_ylim(0,50)
```

- Setting number of bins for a histogram:

```
ax.hist(norm_reviews['Fandango_RatingValue'], bins = 20)
```

- Creating a box plot:

```
ax.boxplot(norm_reviews["RT_user_norm"])
```

- Creating a boxplot for multiple columns of data:

```
num_cols = ['RT_user_norm', 'Metacritic_user_nom', 'IMDB_norm', 'Fandango_Ratingvalue']
ax.boxplot(norm_reviews[num_cols].values)
```

Concepts

- Frequency distribution consists of unique values and corresponding frequencies.
- Bins are intervals of fixed length to cover all possible values.
- Histogram shows the distribution over numerical data.
- Quartiles divide the range of numerical values into four different regions.

- Box plot visually shows quartiles of a set of data as well as any outliers.
- Outliers are abnormal values that affect the overall observation of the data set due to their very high or low values.

Resources

- Documentation for histogram
- Documentation for boxplot
- Various ways to show dsistributions

Line Charts: Takeaways

Syntax

- Importing the pyplot module:

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

- Displaying the plot in a Jupyter Notebook cell:

```
%matplotlib inline
```

- Generating and displaying the plot:

```
plt.plot()

plt.show()
```

- Generating a line chart:

```
plt.plot(first_twelve['DATE'], first_twelve['VALUE'])
```

- To rotate axis ticks:

```
plt.xticks(rotation=90)
```

- To add axis labels:

```
plt.xlabel('Month')

plt.ylabel('Unemployment Rate')
```

- To add a plot label:

```
plt.title('Monthly Unemployment Trends, 1948')
```

Concepts

- To create line charts, we use the matplotlib library, which allows us to: quickly create common plots using high-level functions, extensively tweak plots, and create new kinds of plots from the ground up.
- By default, matplotlib displays a coordinate grid with: the x-axis and y-axis values ranging from -0.6 to 0.6, no grid lines, and no data.

- Visual reperesentations use visual objects like dots, shapes, and lines on a gird.
- Plots are a category of visual representation that allows us to easily understand the representation between variables.

Resources

- Documentation for pyplot
- Types of plots

Multiple plots: Takeaways

Syntax

- Creating a figure using the pyplot module:

```
fig = plt.figure()
```

- Adding a subplot to an existing figure with 2 plots and 1 column, one above the other:
    - Returns a new Axes object, which needs to be assigned to a variable:

```
ax1 = fig.add_subplot(2, 1, 1)

ax2 = fig.add_subplot(2, 1, 2)
```

- Generating a line chart within an Axes object:

```
ax1.plot(unrate['DATE'][:12], unrate['VALUE'][:12])

ax2.plot(unrate['DATE'][12:24], unrate['VALUE'][12:24])
```

- Changing the dimensions of the figure with the figsize parameter (width x height):

```
fig = plt.figure(figsize=(12, 5))
```

- Specifying the color for a certain line using the c parameter:

```
plt.plot(unrate[0:12]['MONTH'], unrate[0:12]['VALUE'], c='red')
```

- Creating a legend using the pyplot module and specifying its location:

```
plt.legend(loc="upper left")
```

- Setting the title for an Axes object:

```
ax.set_title('Unemployment Trend, 1948')
```

Concepts

- A figure acts as a container for all of our plots and has methods for customizing the appearance and behavior for the plots within that container.

- Pyplot uses the following when we create a single plot:
    - A container for all plots was created (returned as a Figure object.)
    - A container for the plot was positioned on a grid (the plot returned as an Axes object.)
    - Visual symbols were added to the plot (using the Axes methods.)
- With each subplot, matplotlib generates a coordinate grid that was similar to the one we generated using the plot() function:
    - The x-axis and y-axis values ranging from 0.0 to 1.0.
    - No gridlines.
    - No data.

Resources

- Methods to Specify Color in Matplotlib
- Lifecycle of a Plot

Bar Plots And Scatter Plots: Takeaways

Syntax

- Generating a vertical bar plot:

  ```
  pyplot.bar(bar_positions, bar_heights, width)
  ```

  OR

  ```
  Axes.bar(bar_positions, bar_heights, width)
  ```

- Using arange to return evenly seperated values:

  ```
  bar_positions = arange(5) + 0.75
  ```

- Using Axes.set_ticks(), which takes in a list of tick locations:

  ```
  ax.set_ticks([1, 2, 3, 4, 5])
  ```

- Using Axes.set_xticklabels(), which takes in a list of labels:

  ```
  ax_set_xticklabels(['RT_user_norm', 'Metacritic_user_nom', 'IMDB_norm',
  'Fandango_Ratingvalue', 'Fandango_Stars'])
  ```

- Rotating the labels:

  ```
  ax_set_xticklabels(['RT_user_norm', 'Metacritic_user_nom', 'IMDB_norm',
  'Fandango_Ratingvalue', 'Fandango_Stars'], rotation = 90)
  ```

- Using Axes.scatter() to create a scatter plot:

  ```
  ax.scatter(norm_reviews["Fandango_Ratingvalue"], norm_reviews["RT_user_norm"])
  ```

Concepts

- A bar plot uses rectangular bars whose lengths are proportional to the values they represent.
  - Bar plots help us to locate the category that corresponds to the smallest or largest value.
  - Bar plots can either be horizontal or vertical.
  - Horizontal bar plots are useful for spotting the largest value.

- A scatter plot helps us determine if 2 columns are weakly or strongly correlated.

Resources

- Documentation for Axes.scatter()
- Correlation Coefficient