|  |  |
| --- | --- |
| **Lecture 1A: Data Basics** | **Lecture 2: Matplotlib Basics** |
|  | * Basic   + Histograms   + Bar Plot   + Line plot   + Scatter plot   + Boxplot   + Violin plot   + Heat map |
| **Lecture 3: NYC Rent** | **Lecture 4: Bulldozer** |
| * Data loading & cleaning * Data restriction * Histograms, scatter plots * Categorical features & bar plots * Numerical features & plots * Word clouds * Correlations & line plots | * Missing values * Data impute * Data cleaning * Categorical variables |
| **Lecture 6: Advanced Vis.** | **Lecture 7: Advanced Vis.** |
| * Advanced Scatter plots   + Add reference points & reference lines   + Annotating text   + Frame borders * Line Plots   + Aggregated data   + Multi-line plot   + Fill between lines   + Stacked Area chart | * Bar plot with 2 columns (min requirement)   + Basic aesthetic adjustments   + Adjust space & highlight a bar   + Color Outlines * Horizontal bar plot   + Order of bars/data * Side by side bar plot * Stacked bar plot * Percentage bar plot * Slope graph |
| **Lecture 9: Advanced Vis.** | **Lecture 5: Visualizations Principles** |
| * Advanced scatter plot with 4 variables (from HW3) * Point size, text annotation, colors, sectioned off * Legends * De-cluster a plot * Salary change over time, aesthetics, dotted lines * Stacked bar plots with subsections |  |
| **Lecture 1B: Date & Time** | **Lecture 8: Design Principles** |

# Import Data

#Packages to import:

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from scipy import stats

import plotly.express as px

import plotly.graph\_objects as go

%config InlineBackend.figure\_format = 'retina'

# Basic plot set up:

1. Establish Data
2. fig, ax = plt.subplots(figsize(width, height))
3. ax.plottype(arguments)
   1. bar(x, y)
   2. scatter(x, y)
   3. plot(x)
   4. hist(x, bins = #)
   5. barh(x, y)
   6. boxplot(x)
   7. violinplot(x)
4. plt.show()

# Types of Plots:

import matplotlib.pyplot as plt

a. bar(x, y, color= , edgecolor= , width= , label= )

# Creates a vertical bar plot.

# x: x-coordinates of the bars

# y: heights of the bars

# color: color of the bars (optional)

# edgecolor: color of the bar edges (optional)

# width: width of the bars (default is 0.8) (optional)

# label: label for the legend (optional)

b. scatter(x, y, s= , c= , alpha= , marker= , cmap= , edgecolor= )

# Creates a scatter plot.

# x: x-coordinates of the points

# y: y-coordinates of the points

# s: size of the dots (optional)

# c: color of the dots (optional), can also be an array for color mapping

# alpha: transparency level of the dots, between 0 (transparent) and 1 (opaque) (optional)

# marker: shape of the marker (default is 'o') (optional)

# cmap: colormap for mapping the colors (optional)

# edgecolor: color of the marker edges (optional)

c. plot(x, y= , color= , linestyle= , linewidth= , marker= , label= )

# Creates a line plot.

# x: x-coordinates of the data points

# y: y-coordinates of the data points (optional, if only x is provided, it assumes y = x)

# color: color of the line (optional)

# linestyle: style of the line (e.g., '-', '--', '-.', ':') (optional)

# linewidth: width of the line (optional)

# marker: marker style for data points (optional)

# label: label for the legend (optional)

d. hist(x, bins= , color= , edgecolor= , alpha= , density= , histtype= )

# Creates a histogram.

# x: data values

# bins: number of bins (or the bin edges) (optional)

# color: color of the bars (optional)

# edgecolor: color of the bar edges (optional)

# alpha: transparency level of the bars, between 0 (transparent) and 1 (opaque) (optional)

# density: if True, the result is the value of the probability density function at the bin (optional)

# histtype: type of histogram to draw (e.g., 'bar', 'barstacked', 'step', 'stepfilled') (optional)

e. barh(y, width= , color= , edgecolor= , height= , label= )

# Creates a horizontal bar plot.

# y: y-coordinates of the bars

# width: width of the bars

# color: color of the bars (optional)

# edgecolor: color of the bar edges (optional)

# height: height of the bars (default is 0.8) (optional)

# label: label for the legend (optional)

f. boxplot(x, notch= , vert= , patch\_artist= , widths= , labels= )

# Creates a box plot.

# x: data values

# notch: if True, make a notched box plot (optional)

# vert: if True, create vertical boxes (default is True) (optional)

# patch\_artist: if True, fill the box with color (optional)

# widths: width of the boxes (optional)

# labels: list of labels for each box (optional)

g. violinplot(x, showmeans= , showmedians= , showextrema= , bw= , widths= )

# Creates a violin plot.

# x: data values

# showmeans: if True, show the mean of the distribution (optional)

# showmedians: if True, show the median of the distribution (optional)

# showextrema: if True, show the extrema of the distribution (optional)

# bw: method for calculating the bandwidth (optional)

# widths: width of the violin (optional)

# Labeling:

# Create a figure and axis

fig, ax = plt.subplots()

# Set X-axis label

ax.set\_xlabel("Your X-axis label") # Label for the X-axis

# Set Y-axis label

ax.set\_ylabel("Your Y-axis label") # Label for the Y-axis

# Set title

ax.set\_title("Your title") # Title of the plot

# Set X-axis tick labels

ax.set\_xticks([1, 2, 3, 4]) # Positions of the X-axis ticks

ax.set\_xticklabels(["A", "B", "C", "D"]) # Labels for the X-axis ticks

# Set Y-axis tick labels

ax.set\_yticks([10, 20, 30, 40]) # Positions of the Y-axis ticks

ax.set\_yticklabels(["Low", "Medium", "High", "Very High"]) # Labels for the Y-axis ticks

# Add legend

ax.legend(["Series 1", "Series 2"], loc="best") # Adds a legend with labels for different series

# Add annotation

ax.annotate("Annotation text", xy=(2, 20), xytext=(3, 30),

arrowprops=dict(facecolor='black', shrink=0.05)) # Annotates a specific point with an arrow

# Add text

ax.text(2, 25, "Text", fontsize=12, color='red') # Adds text at a specific location with custom font size and color

# Show the plot

plt.show()

# Plot Additions:

 **Adding an average line:** a. ax.axvline(value, color= , linestyle= , linewidth= , label= )

* value: x-coordinate where the vertical line should be drawn.
* color: color of the line (optional).
* linestyle: style of the line (e.g., '-', '--', '-.', ':') (optional).
* linewidth: width of the line (optional).
* label: label for the line in the legend (optional).

 **Adding a horizontal line:** a. ax.hlines(y, xmin= , xmax= , color= , linestyle= , linewidth= , label= )

* y: y-coordinate where the horizontal line should be drawn.
* xmin: starting x-coordinate of the line (optional).
* xmax: ending x-coordinate of the line (optional).
* color: color of the line (optional).
* linestyle: style of the line (e.g., '-', '--', '-.', ':') (optional).
* linewidth: width of the line (optional).
* label: label for the line in the legend (optional).

 **Adding a vertical line:** a. ax.vlines(x, ymin= , ymax= , color= , linestyle= , linewidth= , label= )

* x: x-coordinate where the vertical line should be drawn.
* ymin: starting y-coordinate of the line (optional).
* ymax: ending y-coordinate of the line (optional).
* color: color of the line (optional).
* linestyle: style of the line (e.g., '-', '--', '-.', ':') (optional).
* linewidth: width of the line (optional).
* label: label for the line in the legend (optional).

 **Adding grid lines:** a. ax.grid(which= , axis= , color= , linestyle= , linewidth= )

* which: which grid lines to draw ('both', 'major', or 'minor') (optional).
* axis: axis to draw the grid lines on ('both', 'x', or 'y') (optional).
* color: color of the grid lines (optional).
* linestyle: style of the grid lines (e.g., '-', '--', '-.', ':') (optional).
* linewidth: width of the grid lines (optional).

 **Adding labels to axes:** a. ax.set\_xlabel('text', fontsize= , color= ) b. ax.set\_ylabel('text', fontsize= , color= )

* text: label text for the x-axis or y-axis.
* fontsize: font size of the label (optional).
* color: color of the label text (optional).

 **Adding a title:** a. ax.set\_title('text', fontsize= , color= , pad= )

* text: title text for the plot.
* fontsize: font size of the title (optional).
* color: color of the title text (optional).
* pad: spacing between the title and the plot (optional).

 **Adding a legend:** a. ax.legend(loc= , fontsize= , title= , title\_fontsize= )

* loc: location of the legend ('upper right', 'lower left', etc.) (optional).
* fontsize: font size of the legend text (optional).
* title: title for the legend (optional).
* title\_fontsize: font size of the legend title (optional).

 **Adding annotations:** a. ax.annotate('text', xy= , xytext= , arrowprops= , fontsize= , color= )

* text: annotation text.
* xy: position (x, y) to annotate.
* xytext: position (x, y) for the text (optional).
* arrowprops: dictionary of arrow properties (optional).
* fontsize: font size of the annotation text (optional).
* color: color of the annotation text (optional).

 **Adding ticks and tick labels:** a. ax.set\_xticks(ticks= ) b. ax.set\_xticklabels(labels= , rotation= , fontsize= ) c. ax.set\_yticks(ticks= ) d. ax.set\_yticklabels(labels= , rotation= , fontsize= )

* ticks: list of tick positions.
* labels: list of labels corresponding to the ticks.
* rotation: rotation angle for the tick labels (optional).
* fontsize: font size of the tick labels (optional).

 **Adding a colorbar:** a. cbar = plt.colorbar(mappable= , ax= , orientation= , fraction= , pad= )

* mappable: the image or plot to which the colorbar is associated.
* ax: the axes to which the colorbar is added (optional).
* orientation: orientation of the colorbar ('vertical' or 'horizontal') (optional).
* fraction: fraction of the original axes to use for the colorbar (optional).
* pad: padding between the colorbar and the plot (optional).

# Basic Plot Features:

1. **Setting Limits:**
   * **X-axis limits:** ax.set\_xlim(lower, upper)
   * **Y-axis limits:** ax.set\_ylim(lower, upper)
   * Sets the range of data to display on the x and y axes.
2. **Setting Aspect Ratio:**
   * **Equal aspect ratio:** ax.set\_aspect('equal')
   * **Auto aspect ratio:** ax.set\_aspect('auto')
   * Ensures that the scaling of the x and y axes is equal or adjusted automatically.
3. **Customizing Tick Marks:**
   * **Major ticks:** ax.xaxis.set\_major\_locator(plt.MultipleLocator(interval))
   * **Minor ticks:** ax.xaxis.set\_minor\_locator(plt.MultipleLocator(interval))
   * **Major tick labels:** ax.set\_xticklabels(labels)
   * **Minor tick labels:** ax.set\_yticklabels(labels)
   * Customize the intervals and labels of major and minor ticks.
4. **Customizing Plot Style:**
   * **Line style and color:** ax.plot(x, y, linestyle='--', color='r')
   * **Marker style and size:** ax.scatter(x, y, marker='o', s=50)
   * Choose different styles for lines and markers.
5. **Adding Text Annotations:**
   * **Text:** ax.text(x, y, 'text', fontsize= , color= , ha= , va= )
   * **ha (horizontal alignment):** 'left', 'center', 'right'
   * **va (vertical alignment):** 'top', 'center', 'bottom'
   * Add text annotations at specific locations with customizable alignment.
6. **Adding Legends:**
   * **Label for plots:** ax.plot(x, y, label='label')
   * **Legend display:** ax.legend()
   * Ensure each plot or line has a label, and display the legend to differentiate them.
7. **Adding Titles and Axis Labels:**
   * **Plot title:** ax.set\_title('Title', fontsize= , color= )
   * **X-axis label:** ax.set\_xlabel('X Label', fontsize= , color= )
   * **Y-axis label:** ax.set\_ylabel('Y Label', fontsize= , color= )
   * Add titles and labels to clarify what the plot represents.
8. **Customizing Figure Size:**
   * **Figure size:** plt.figure(figsize=(width, height))
   * Adjust the size of the entire figure for better presentation.
9. **Saving Plots:**
   * **Save plot:** plt.savefig('filename.png', dpi=300)
   * Save your plot to a file with specified resolution.
10. **Adding Gridlines:**
    * **Grid:** ax.grid(True, linestyle='--', color='gray')
    * Enable and style gridlines to make the plot easier to read.
11. **Using Different Plot Types:**
    * **Histograms:** ax.hist(data, bins= , color= )
    * **Box Plots:** ax.boxplot(data)
    * **Violin Plots:** ax.violinplot(data)
    * Be familiar with the different types of plots and their specific arguments.
12. **Setting Figure and Axes Parameters:**
    * **Figure size and layout:** plt.subplots(figsize=(width, height))
    * **Adjust layout:** plt.tight\_layout()
    * Ensure your plot is properly sized and laid out.
13. **Customizing Tick Marks and Labels:**
    * **Custom ticks:** ax.set\_xticks([1, 2, 3])
    * **Custom labels:** ax.set\_xticklabels(['A', 'B', 'C'])
    * Control the appearance and positioning of tick marks and labels.
14. **Plotting Multiple Lines:**
    * **Multiple lines:** ax.plot(x1, y1, label='line1'), ax.plot(x2, y2, label='line2')
    * Plot multiple lines or datasets on the same axes.
15. **Adding Error Bars:**
    * **Error bars:** ax.errorbar(x, y, yerr=error, fmt='o')
    * Display uncertainty or variability in the data.