Cheatography

Seaborn Cheat Sheet

by Arshdeep via cheatography.com/201979/cs/43039/

Introduction to Seaborn

Seaborn

Seaborn is a Python visualization library based on matplotlib that provides a high-level interface for drawing attractive statistical graphics. It is built on top of matplotlib and closely integrated with pandas data structures, making it an excellent tool for exploring and visualizing datasets.

Key Features

Simplified syntax for creating complex visualizations.

Built-in themes and color palettes to improve the aesthetics of plots.

Support for a wide range of statistical plots for exploring relationships in data.

Seamless integration with pandas DataFrames for easy data manipulation and visualization.

Capabilities for both univariate and multivariate visualizations.

Integration with matplotlib for fine-tuning and customization.

Getting Started

Install Seaborn using pip: pip install seaborn.

Import Seaborn in your Python script or Jupyter Notebook: import seaborn as sns.

Load your data into pandas DataFrame if not already in one.

Start exploring your data using Seaborn's high-level plotting functions.

Installing Seaborn

Using pip	`pip install seaborn`
Using conda	`conda install seaborn`
Verify Installation	`import seaborn as sns`

Loading Data

Using Pandas	`import pandas as pd`
	`df = pd.rea d_c sv( 'fi len ame.cs v')`
	`# Load CSV file`

Viewing Data	`df.head()  # View first few rows`

Understanding Data	`df.info()`
	`# Summary of DataFrame`
	`df.describe()`
	`# Descri ptive statistics`

Loading Data (cont)

Handling Missing Data	`df.dro pna()`
	`# Drop rows with missing values`
	`df.fillna(value)`
	`# Fill missing values`

Loading Built-in Datasets	`import seaborn as sns`
	`df = sns.lo ad_ dat ase t(' dat ase t _n ame')`

Basic Plotting Functions

`sns.sc att erp lot(x, y, da ta)`	Create a scatter plot to visualize the relationship between two variables.
`sns.li nep lot(x, y, data)`	Generate a line plot to show trends in data over continuous intervals.
`sns.ba rpl ot(x, y, data)`	Construct a bar plot to display the distribution of categorical data.
`sns.co unt plot(x, data)`	Plot the frequency of unique values in a categorical variable.
`sns.bo xpl ot(x, y, data)`	Draw a box plot to summarize the distribution of a continuous variable within different levels of a catego-rical variable.
`sns.vi oli npl ot(x, y, dat a)`	Create a violin plot to visualize the distribution of a continuous variable across different categories.
`sns.hi stp lot (data, x)`	Generate a histogram to display the distribution of a single variable.

Customizing Plots

Changing Colors	Use the color parameter to specify colors for elements such as lines, markers, and bars. Seaborn also provides color palettes (palette parameter) for different visualiza-tions.

Customizing Plots (cont)

Adjusting Line Styles and Markers	Control the style of lines with the linestyle parameter and markers with the marker parameter. Options include solid lines ('-'), dashed lines ('--'), and various marker shapes ('o', 's', 'D', etc.).
Setting Plot Size	Use the plt.figure(figsize=(width, height)) function to specify the size of your plot. Adjust the width and height values as needed to achieve the desired dimensions.
Adding Titles and Labels	Set the title of your plot with plt.title() and label the axes with plt.xlabel() and plt.ylabel(). Provide informative titles and labels to make your plots more understandable.
Changing Font Sizes	Customize font sizes for titles, labels, and ticks using parameters such as fontsize or by accessing individual text elements.
Adjusting Axis Limits	Control the range of values displayed on the x and y axes using plt.xlim() and plt.ylim() functions. Set appropriate limits to focus on specific regions of interest in your data.
Adding Grid Lines	Use plt.grid(True) to display grid lines on your plot, aiding in data interpretation.
Adding Legends	Include a legend to distinguish between multiple elements in your plot using the plt.legend() function. Customize the legend labels and placement for clarity.

Saving Plots

Syntax	`import seaborn as sns`
	`# Create your plot here`
	`sns.savefig("filename.extension")`

Example	`import seaborn as sns`
	`import matplo tli b.p yplot as plt`
	`# Create a scatter plot`
	`sns.scatterplot(x='x', y='y', data=data)`
	`# Save the plot as a PNG file`
	`plt.savefig("scatter_plot.png")`

Supported File Formats	PNG (Portable Network Graphics) JPG/JPEG (Joint Photographic Experts Group) PDF (Portable Document Format) SVG (Scalable Vector Graphics) and more

Categorical Plots

barplot()	Displays the central tendency and confidence interval of numeric variables across different categories. Useful for comparing the mean or aggregate statistic of numeric data for each category.
countplot()	Shows the count of observations in each category using bars. Suitable for exploring the distribution of categorical variables.
boxplot()	Visualizes the distribution of quantitative data across different levels of one or more categorical variables. Useful for identifying outliers and comparing distributions.

Categorical Plots (cont)

violinplot()	Combines the benefits of a box plot and a kernel density plot. Provides information about the distribution of data within each category.
stripplot() and swarmplot()	Scatterplots for categorical data. Show individual data points along with a categorical variable. Swarmplot avoids overlapping points by adjusting them along the categorical axis.
pointplot()	Represents the point estimates and confidence intervals using lines. Useful for visualizing the relationship between two categorical variables.
factorplot() (deprecated, use catplot() instead)	A versatile function that can create different types of categorical plots based on the kind parameter. Offers a convenient way to explore relationships between variables.
catplot()	Replaces factorplot and serves as a general plot function for categorical data. Supports various plot types such as stripplot, swarmplot, boxplot, etc., through the kind parameter.

Distribution Plots

Distribution Plots	Distribution plots in Seaborn allow you to visualize the distribution of a dataset. These plots help you understand the underlying distribution of your data, including its central tendency, spread, and skewness.

Distribution Plots (cont)

Histograms	sns.histplot(data, x='column'): Plot a histogram of the specified column in the dataset. Customize with parameters like bins, kde, color, and alpha.
Kernel Density Estimation (KDE) Plots	sns.kdeplot(data, x='column'): Generate a smooth estimate of the probability density function. Additional parameters include bw_method, fill, and common_norm.
Rug Plots	sns.rugplot(data, x='column'): Plot a line for each data point along the x-axis. Useful for visualizing individual data points in combination with other plots.
Cumulative Distribution Function (CDF)	sns.ecdfplot(data, x='column'): Plot the empirical cumulative distribution function. Helps to visualize the cumulative proportion of data points.
Joint Distribution Plots	sns.jointplot(data=data, x='x_column', y='y_column', kind='kind'): Plot the joint distribution of two variables along with their marginal distributions. kind parameter can be set to scatter, kde, hist, hex, or reg for different visualizations.
Pair Plots	sns.pairplot(data): Create pairwise plots for all numerical columns in the dataset. Offers a quick overview of relationships between multiple variables.

Distribution Plots (cont)

Violin Plots	sns.violinplot(data=data, x='x_column', y='y_column'): Visualize the distribution of a numeric variable for different categories. Provides insights into both the distribution and the probability density at different values.
Box Plots	sns.boxplot(data=data, x='x_column', y='y_column'): Summarize the distribution of a numeric variable for different categories using quartiles. Helps to identify outliers and compare distributions between categories.
Swarm Plots	sns.swarmplot(data=data, x='x_column', y='y_column'): Show each data point along with the distribution. Useful for small to moderate-sized datasets.
Violin-Swarm Combination	Combining violin and swarm plots can provide a comprehensive view of the distribution and individual data points.

Regression Plots

Regression Plots	Regression plots in Seaborn are useful for visualizing relationships between variables and fitting regression models to the data. Seaborn provides several functions for creating regression plots, allowing you to explore linear relationships, examine residuals, and detect outliers.

Regression Plots (cont)

lmplot()	Used for plotting linear models. Syntax: sns.lmplot(x, y, data, ...). Displays scatter plot with a linear regression line. Useful for visualizing the relationship between two variables and assessing the fit of a linear model.
regplot()	Similar to lmplot() but can be used in more general contexts. Syntax: sns.regplot(x, y, data, ...). Produces scatter plot with a regression line. Offers additional customization options compared to lmplot().
residplot()	Used for plotting the residuals of a linear regression. Syntax: sns.residplot(x, y, data, ...). Helps to diagnose the fit of the regression model by plotting the difference between observed and predicted values. Useful for identifying patterns or heteroscedasticity in residuals.
Additional Parameters	order: Specifies the order of the polynomial regression (default is 1 for linear). scatter_kws: Additional keyword arguments passed to the scatterplot function. line_kws: Additional keyword arguments passed to the line plot function. ci: Confidence interval size for the regression estimate. truncate: Truncates the regression line at the data limits.

Regression Plots (cont)

Example	`import seaborn as sns`
	`import matplo tli b.p yplot as plt`
	`# Load sample data`
	`tips = sns.lo ad_ dat ase t("t ips ")`
	`# Create a regression plot`
	`sns.lmplot(x="total_bill",`
	`y="tip", data=tips)`
	`# Show the plot`
	`plt.show()`

Matrix Plots

Matrix Plots	Matrix plots in Seaborn are useful for visualizing data in matrix form, typically with heatmap-style representations.
Heatmaps	Use sns.heatmap() to create a colored matrix plot, with each cell representing the value of a variable in the dataset. Ideal for displaying correlation matrices or any two-dimensional data.
Cluster Maps	sns.clustermap() creates a hierarchical clustering heatmap. It's handy for exploring relationships between variables by grouping similar ones together.
Pair Plots	Although not strictly matrix plots, sns.pairplot() generates a matrix of scatterplots and histograms for quick visualization of relationships between multiple variables in a dataset.
Customization	Seaborn allows extensive customization of matrix plots, including adjusting color schemes, annotating cells with values, and tweaking axes.

Time Series Plots

Time Series Plots	Time series plots in Seaborn are useful for visualizing data over time. Seaborn provides several functions to create informative time series plots.

Time Series Plots (cont)

`seabor n.l ine plot(x, y, data)`	Creates a line plot of y vs. x with optional data argument. Ideal for visualizing trends and patterns over time.
`seabor n.r elp lot(x, y, data, kind='line')`	Offers a high-level interface to create various plot types, including line plots for time series data. Use the kind parameter to specify the plot type (default is 'line').
`seabor n.s cat ter plot(x, y, d ata)`	Plots individual data points as scatter points. Suitable for visualizing relationships between variables over time.
`seabor n.t spl ot( data, time, unit, value)`	Deprecated since Seaborn version 0.9. Use other functions for time series visualization.
`seaborn.linearmodels.Tsplot( data, time, unit, value)`	Visualizes time series data with confidence intervals. Suitable for comparing multiple time series.

Style and Aesthetics

Seaborn Styles	seaborn.set_style(style=None): Set the aesthetic style of the plots. Styles include: 'darkgrid', 'whitegrid', 'dark', 'white', and 'ticks'.
Color Palettes	seaborn.color_palette(palette=None, n_colors=None, desat=None): Set the color palette for plots. Built-in palettes: 'deep', 'muted', 'bright', 'pastel', 'dark', 'colorblind', etc. Custom palettes can be created using seaborn.color_palette().

Style and Aesthetics (cont)	
Contexts	seaborn.set_context(context=None, font_scale=1, rc=None): Set the context parameters for the plot. Contexts control the scale of plot elements. Contexts include: 'paper', 'notebook', 'talk', and 'poster'.
Plot Aesthetics	seaborn.despine(fig=None, ax=None, top=True, right=-True, left=False, bottom=False, offset=None, trim=F-alse): Remove axes spines from the plot. seaborn.set_-palette(palette, n_colors=None, desat=None, color_-codes=False): Set the color palette for the current seaborn context. seaborn.set_context(context=None, font_scale=1, rc=None): Set the plotting context parameters.
Other Aesthetic Tweaks	seaborn.set(): Set aesthetic parameters in one step. seaborn.reset_defaults(): Restore default seaborn parameters. seaborn.set_theme(): Set the default seaborn theme.
Saving Aesthetic Settings	seaborn.axes_style(style=None, rc=None): Return a dictionary of parameters or use in a with statement to temporarily set the style. seaborn.plotting_context(co-ntext=None, font_scale=1, rc=None): Return a dictionary of parameters or use in a with statement to temporarily set the context.