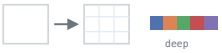
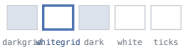
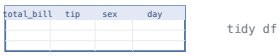
seaborn

An overview of seaborn's statistical plots, with sensible defaults for distributions, relationships, categories, and faceting.

Setup & Theming

Import seaborn, load tidy data, and set the global look in one call.

PURPOSE	COMMAND	RESULT
Standard imports.	`import seaborn as sns` `import seaborn.objects as so`	
Set the theme globally.	`sns.set_theme(style="whitegrid",` `context="notebook", palette="deep")`	
Pick a background style.	`sns.set_style("whitegrid")`	
Scale elements for the medium.	`sns.set_context("talk")`	
Load a built-in tidy dataset.	`tips =sns.load_dataset("tips")`	
Remove top/right spines.	`sns.despine()`	

seaborn assumes tidy data: each column a variable, each row an observation; set_theme controls the look

Relational plots

Scatter and line plots; relplot is the figure-level entry point.

PURPOSE	COMMAND	RESULT
Scatter onto one Axes.	`sns.scatterplot(` `data=tips, x=.., y=.., hue="time")`	
Encode 3 extra channels at once.	`sns.scatterplot(.., hue=, size=, style=)`	
Line plot with auto error band.	`sns.lineplot(data=fmri, .., errorbar="sd")`	
Figure-level relational + faceting.	`sns.relplot(data=tips, .., col="sex")`	
Switch a relplot to lines.	`sns.relplot(.., kind="line", col="region")`	

hue / size / style each map a column to a channel; relplot adds col / row faceting on the same grammar

Distributions

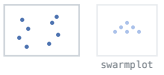
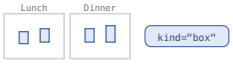
Histograms, KDEs, ECDFs; displot is the figure-level entry point.

PURPOSE	COMMAND	RESULT
Histogram, optionally with KDE.	`sns.histplot(data=penguins, x=.., kde=True)`	 bars + curve
Compare groups: stack the bars.	`sns.histplot(.., hue=, multiple="stack")`	 stack
Smooth density estimate.	`sns.kdeplot(.., hue="species", fill=True)`	 filled densities
Cumulative distribution.	`sns.ecdfplot(.., hue="species")`	 1.0 0 -> 1
Figure-level distribution + facets.	`sns.displot(.., col="island", kind="hist")`	 Biscoe 3 islands
Bivariate density / 2-D hist.	`sns.displot(.., x=, y=, kind="kde")`	 contours

histplot bins into bars (kde=True overlays a curve); kdeplot draws density alone; ecdfplot has nothing to tune

Categorical plots

A numeric value across categories; catplot is the figure-level entry point.

PURPOSE	COMMAND	RESULT
Box plot across categories.	`sns.boxplot(.., x="day", y=.., hue="smoker")`	 grouped
Violin (shape), split.	`sns.violinplot(.., hue="sex", split=True)`	 mirrored
Bar of an aggregate + error bar.	`sns.barplot(.., errorbar=("ci", 95))`	 mean by default
Count occurrences per category.	`sns.countplot(data=tips, x="day")`	 Σ x only, no y=
Show every point (jittered).	`sns.stripplot(.., x="day", y=..)`	 swarmplot
Figure-level categorical + facets.	`sns.catplot(.., col="time", kind="box")`	 Lunch Dinner kind="box"

distributions (box/violin), estimates (bar/point), and every-point (strip/swarm); catplot picks one with kind=

Regression & fits

Overlay a model fit on a scatter; lmplot is the figure-level entry point.

PURPOSE	COMMAND	RESULT
Scatter + linear fit + CI band.	`sns.regplot(data=tips, x=.., y=..)`	 fit + CI
Higher-order (polynomial) fit.	`sns.regplot(.., x=, y=, order=2)`	 order=2
Figure-level regression by group.	`sns.lmplot(.., hue="smoker", col="time")`	 smoker
Check the fit: residual plot.	`sns.residplot(.., x=, y=)`	 want no pattern
Pairwise scatter matrix.	`sns.pairplot(penguins, hue=, corner=True)`	 lower triangle
Joint plot: scatter + marginals.	`sns.jointplot(.., x=, y=, hue=)`	 + margins

regplot draws the fit + bootstrapped CI; lmplot fits a line per hue group and per facet

Matrix & correlation plots

Turn a rectangular table or correlation matrix into a colored grid.

PURPOSE	COMMAND	RESULT
Heatmap of a 2-D table.	`sns.heatmap(flights, cmap="mako")`	 + colorbar
Annotated correlation heatmap.	`sns.heatmap(corr, annot=True, cmap="vlag", center=0)`	 diverging, white at 0
Build the input matrix first.	`corr = penguins.select_dtypes("number").corr()`	 corr (1.0 diag)
Make cells square, add gridlines.	`sns.heatmap(corr, square=True, linewidths=.5)`	 white gutters
Cluster rows/cols + dendrograms.	`sns.clustermap(flights, standard_scale=1)`	 needs scipy

the common recipe: df.corr() into heatmap(annot=True, center=0) with a diverging palette

Faceting & grid control

Small multiples: col / row on figure-level functions, or drive the grids by hand.

PURPOSE	COMMAND	RESULT
Facet by one variable.	`sns.relplot(.., col="day", col_wrap=2)`	
Facet on a grid (rows x cols).	`sns.catplot(.., row="smoker", col="time")`	
Drive a FacetGrid by hand.	`g =sns.FacetGrid(tips, col=, row=)` `g.map_dataframe(sns.scatterplot, ..)`	
Relabel facets and axes.	`g.set_titles("{col_name}")` `g.set_axis_labels("Bill", "Tip")`	
Reposition the shared legend.	`sns.move_legend(g, "upper left", ..)`	
Save the whole figure.	`g.savefig("plot.png", dpi=300)`	

figure-level functions return a FacetGrid you tidy with set_titles, set_axis_labels, move_legend, savefig

The objects interface (so)

Compose a plot from marks and statistical transforms, layer by layer.

PURPOSE	COMMAND	RESULT
Start a Plot, declare mappings.	`so.Plot(penguins, x=, y=, color="species")`	
Add a mark (a layer).	`.add(so.Dot())`	
Add a transformed layer.	`.add(so.Line(),so.PolyFit())`	
Aggregate into bars.	`so.Plot(.., x, y).add(so.Bar(),so.Agg())`	
Facet + rescale + relabel (chain).	`.facet(col=).scale().label().theme()`	
Render or save the result.	`.plot() .save("out.png", dpi=200)`	

each layer is a mark (Dot, Line, Bar) plus an optional stat (Agg, PolyFit); everything chains and returns a Plot