Visualizing Statistical Data Using Seaborn

VISUALIZING RELATIONSHIPS AND DISTRIBUTIONS IN SEABORN

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Seaborn is a powerful visualization library

Built on top of Matplotlib

Tightly integrated with PyData stack

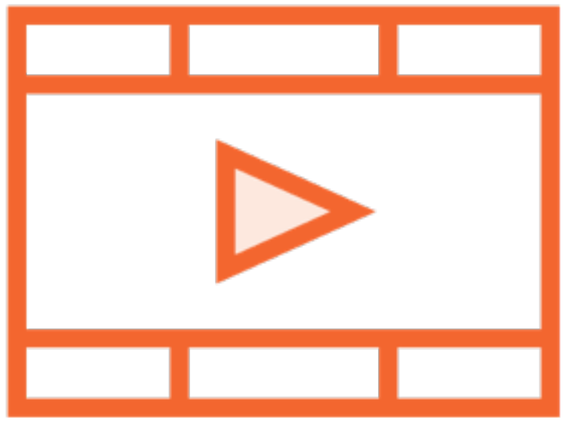
Matplotlib seeks to “make easy things easy and hard things possible”

Seaborn is a complement

Makes “production ready” plots

Prerequisites and Course Outline

Prerequisite Courses



Python: Getting Started

Python Fundamentals

Advanced Python

Software and Skills

Be very comfortable programming in Python (Python 3)

Be comfortable working with Jupyter notebooks

Understand basic high school level statistics

Course Outline

Visualizing relationships

- Univariate and bivariate relationships
- Histograms, KDE curves, scatter plots, box plots, violin plots

Building trellis plots

- Facet grid and pair grid

Plot aesthetics and style

- Themes, color palettes
- Fine grained control over figures and plots

Matplotlib

Tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.

matplotlib.org

Seaborn

Built on top of matplotlib and tightly integrated with the PyData stack, including support for numpy and pandas data structures and statistical routines from scipy and statsmodels.

seaborn.pydata.org

Seaborn For “Production Plots”

Matplotlib

Part of “Pydata” - open data science stack

Provides fine-grained control so that pretty much everything is possible

Two APIs - Matplotlib API (low-level) and Pyplot (higher level)

Production-level aesthetics possible, but need use of Matplotlib API

Seaborn

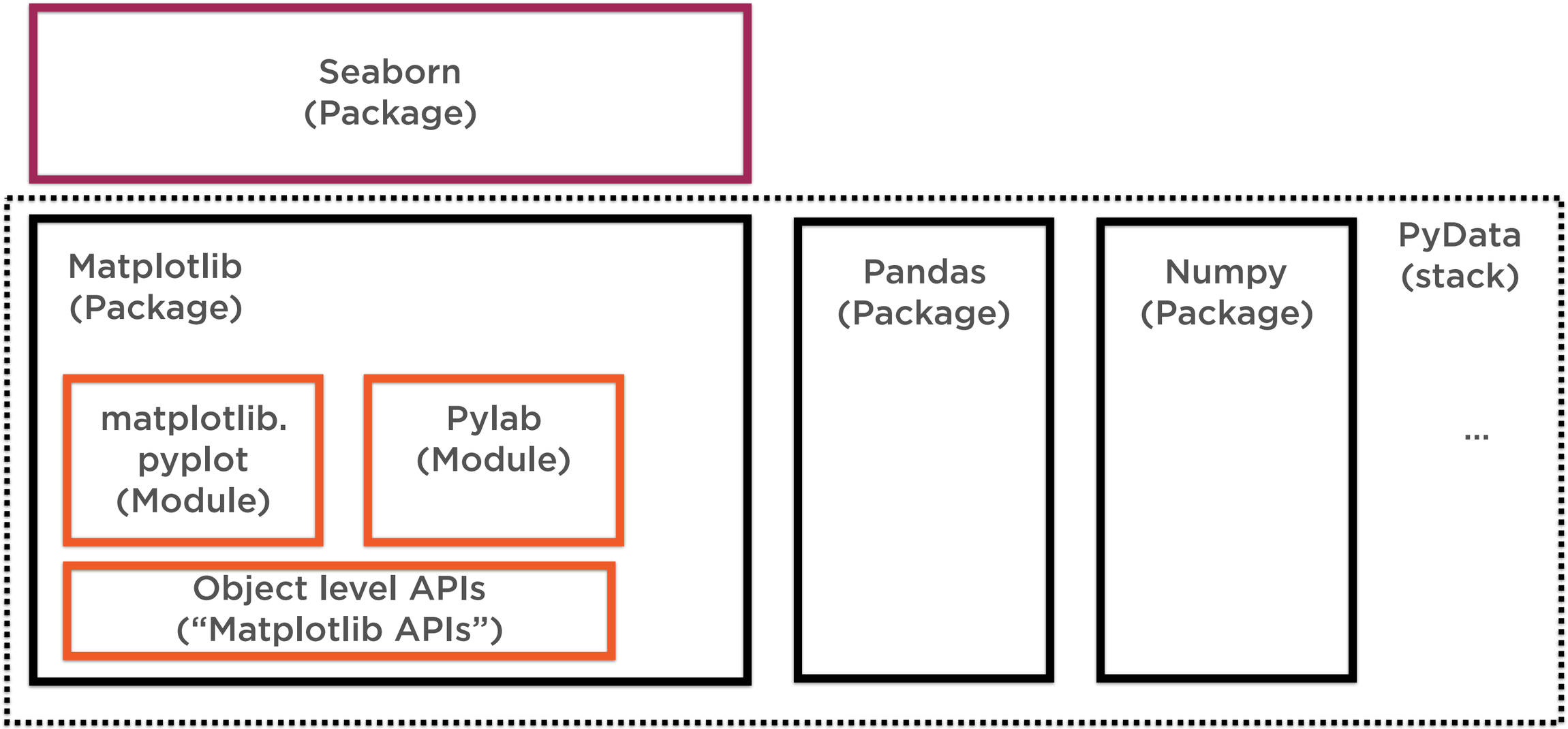
Built atop Matplotlib and tightly integrates with Pydata

High level, easy-to-use abstractions for common use cases

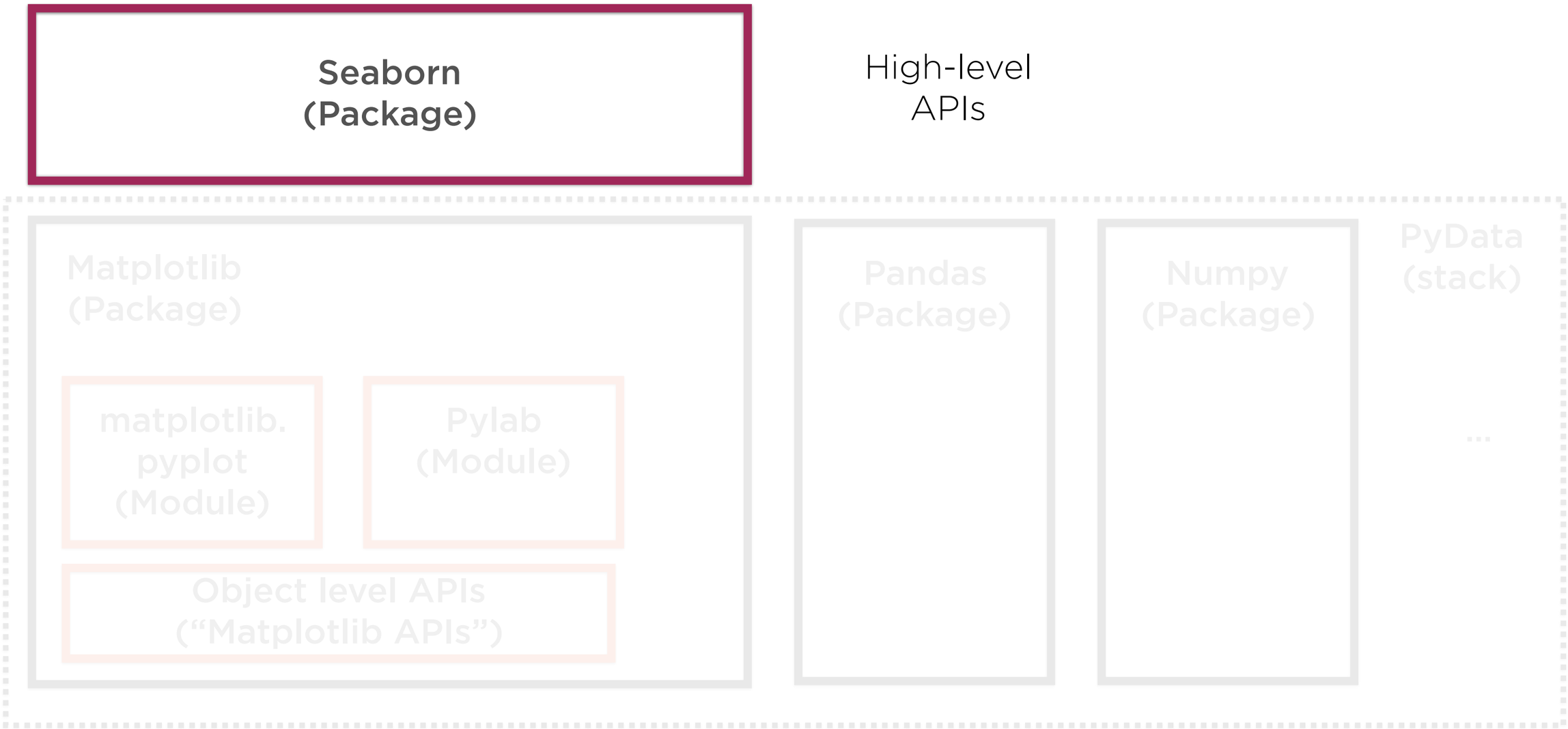
Even higher level than Pyplot (used alongside it)

Production-level aesthetics without need for low-level API

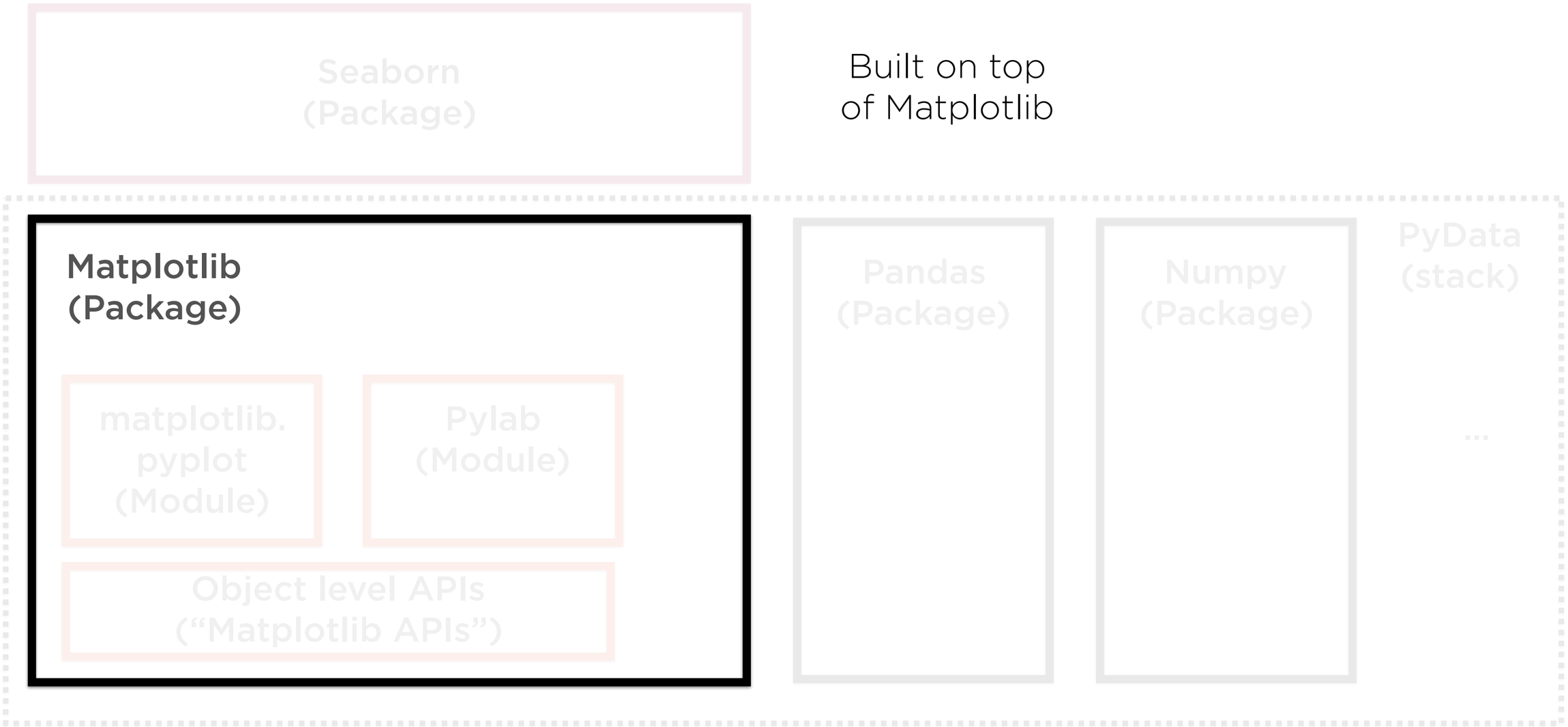
Matplotlib and Seaborn



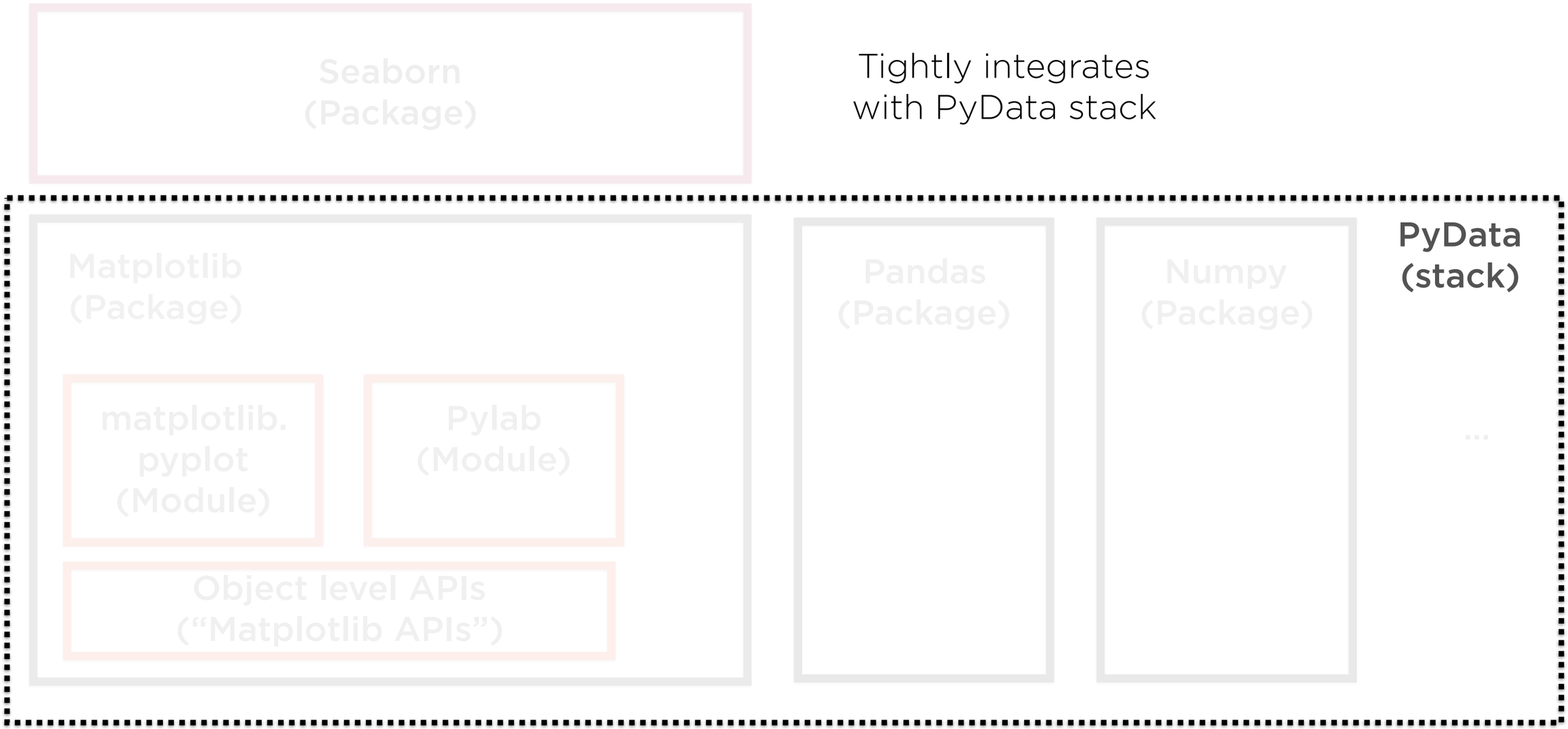
Matplotlib and Seaborn



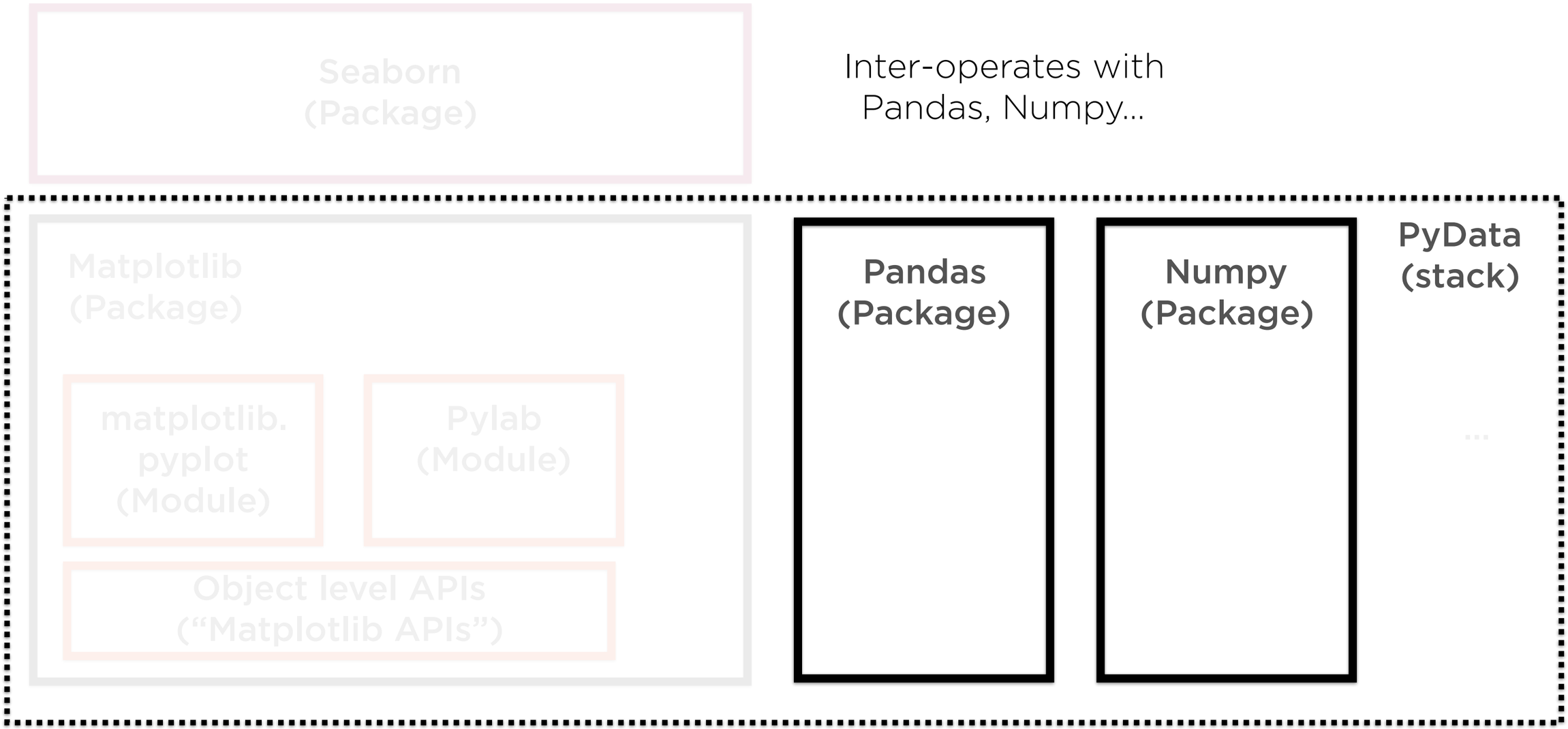
Matplotlib and Seaborn



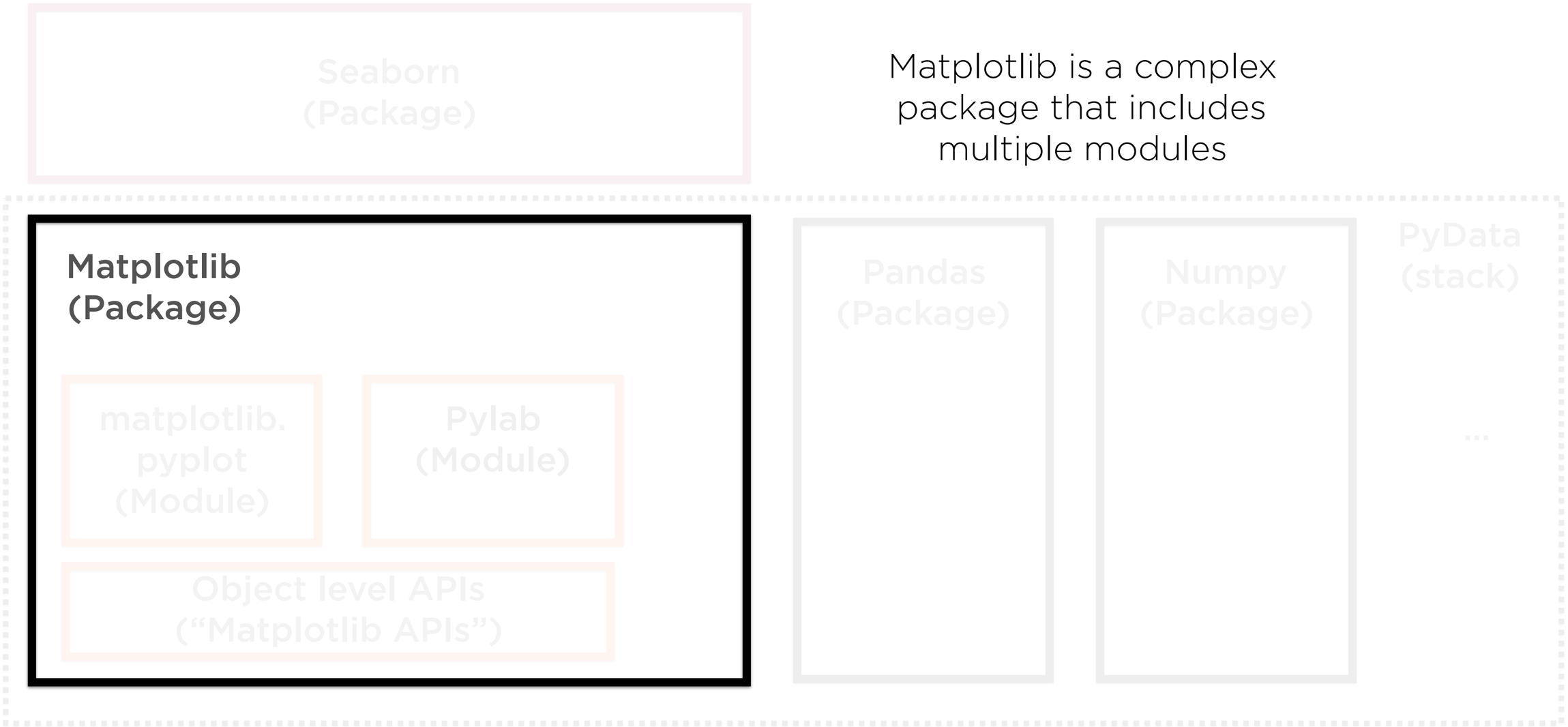
Matplotlib and Seaborn



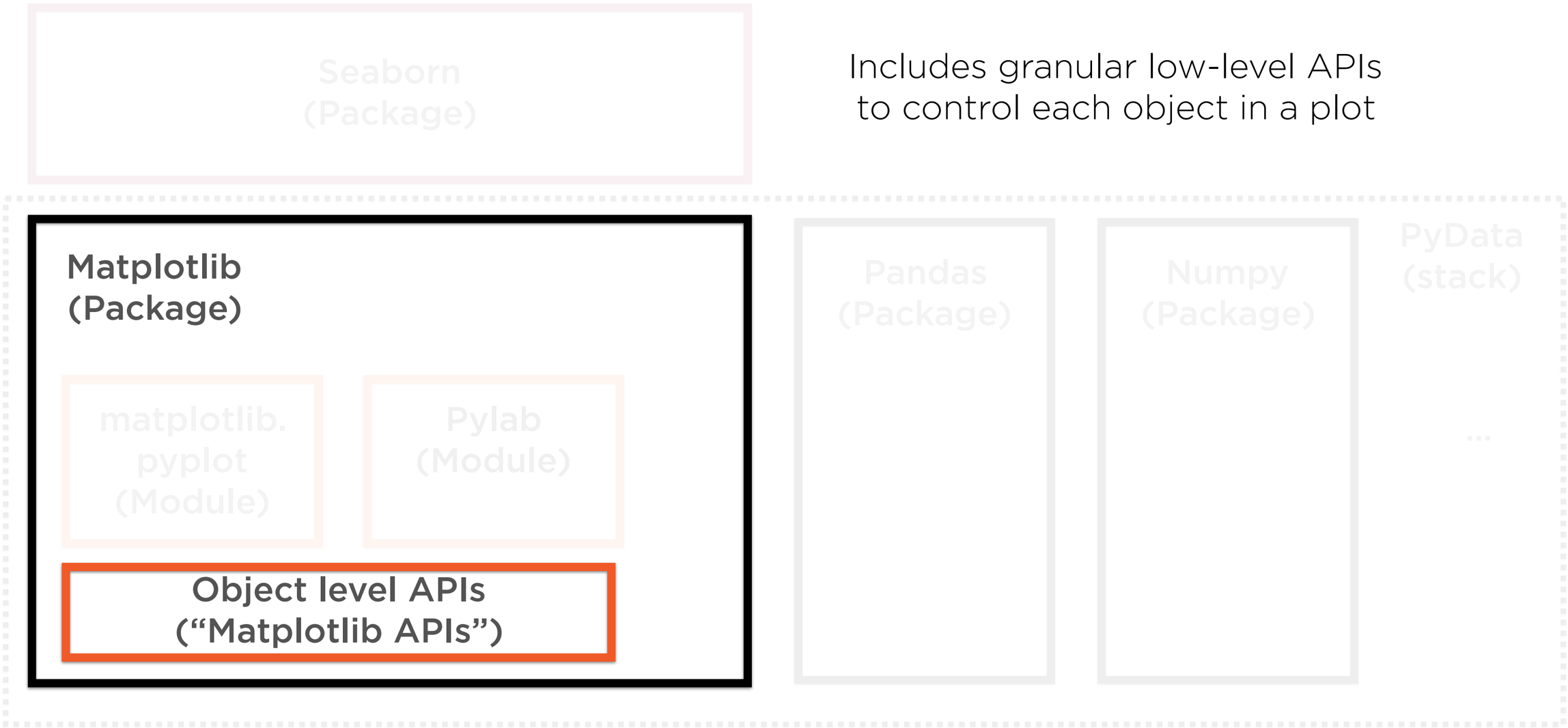
Matplotlib and Seaborn



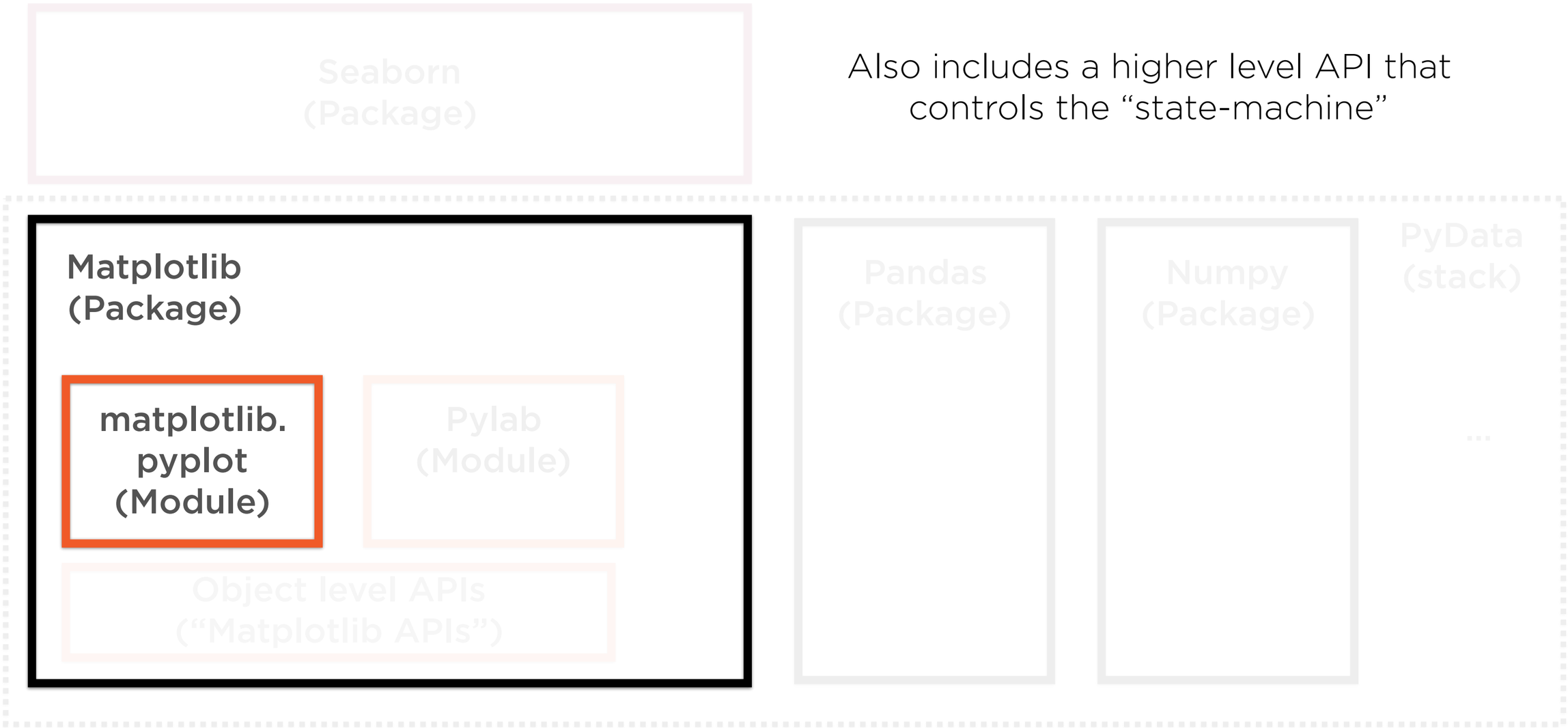
Matplotlib and Seaborn



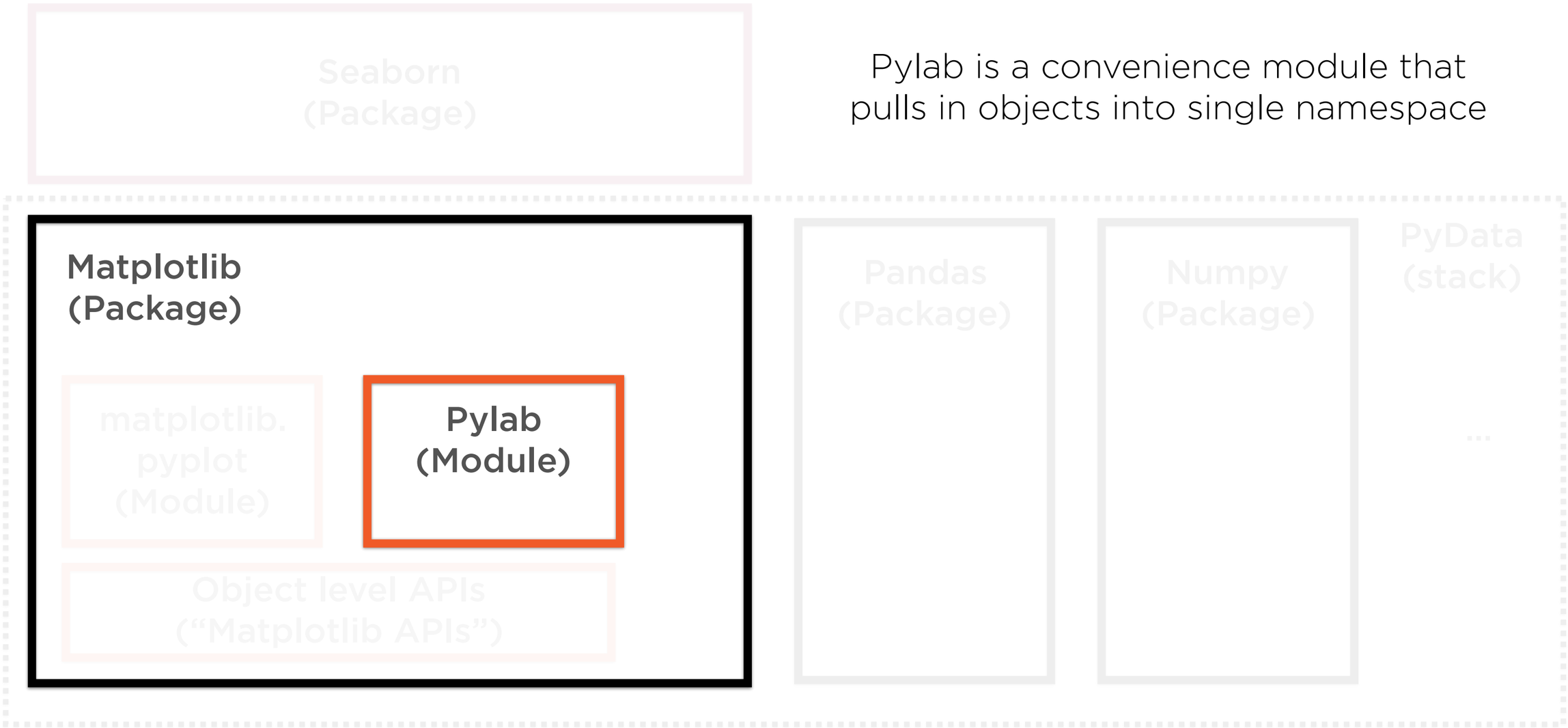
Matplotlib and Seaborn



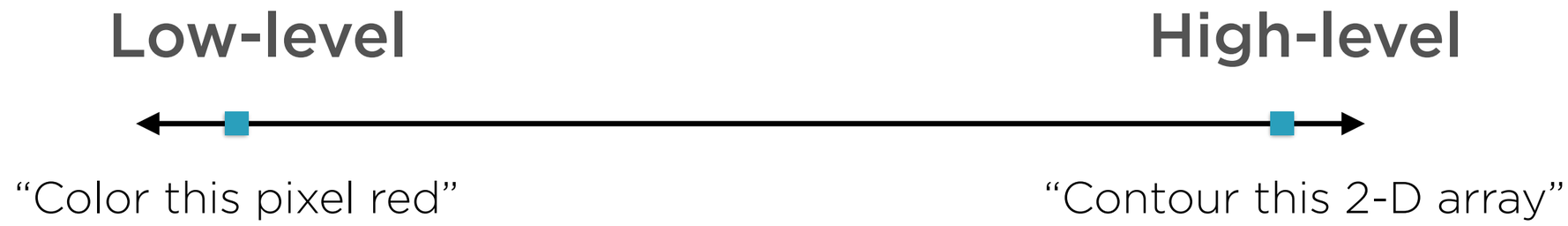
Matplotlib and Seaborn



Matplotlib and Seaborn



Hierarchy of Plotting Operations



Low-level operations act on specific plot elements, high-level operations act on plot as a whole

This hierarchy is formalized in the Matplotlib codebase

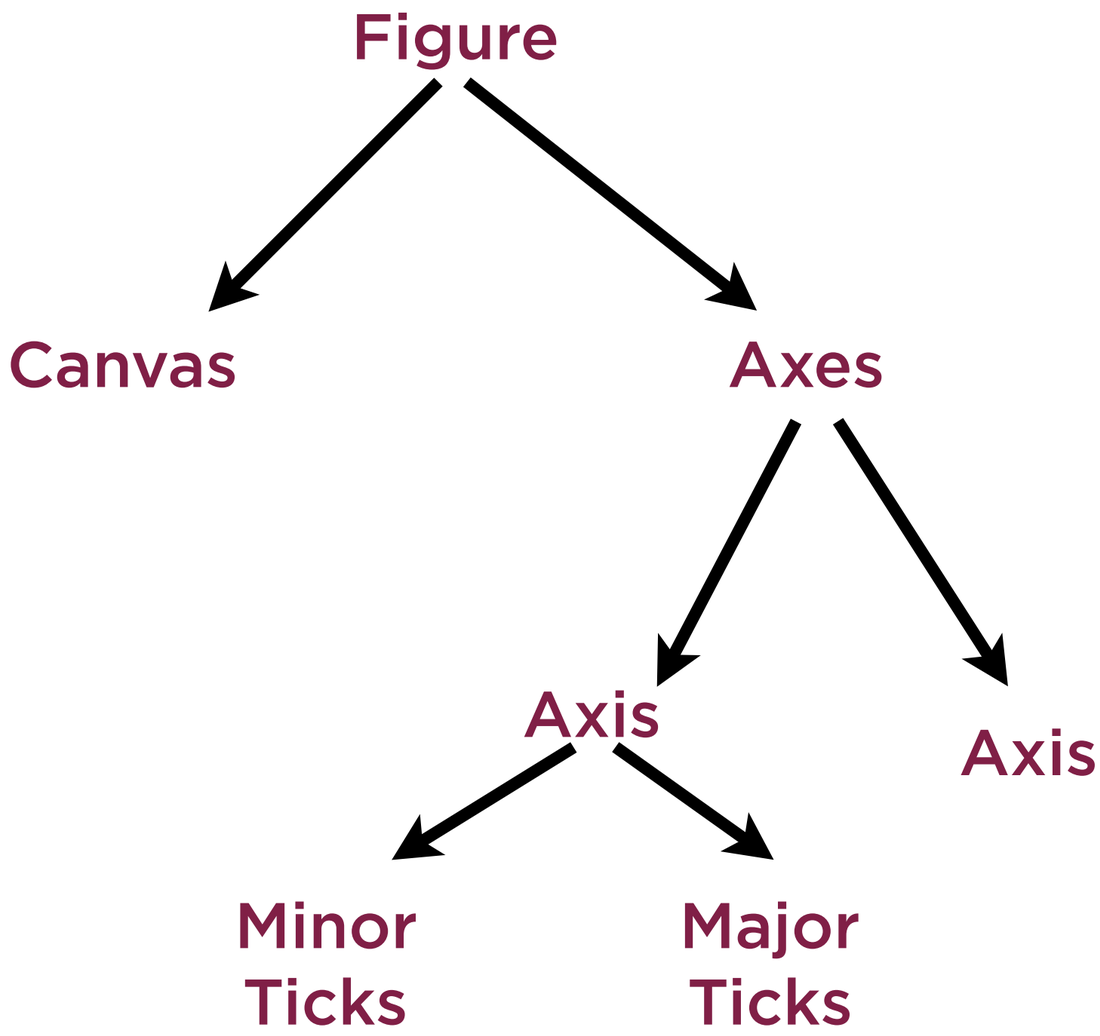
Hierarchy

Everything is an "Artist"

Artists are arranged in a hierarchy

Artist is an abstract base class

Figure is a container class

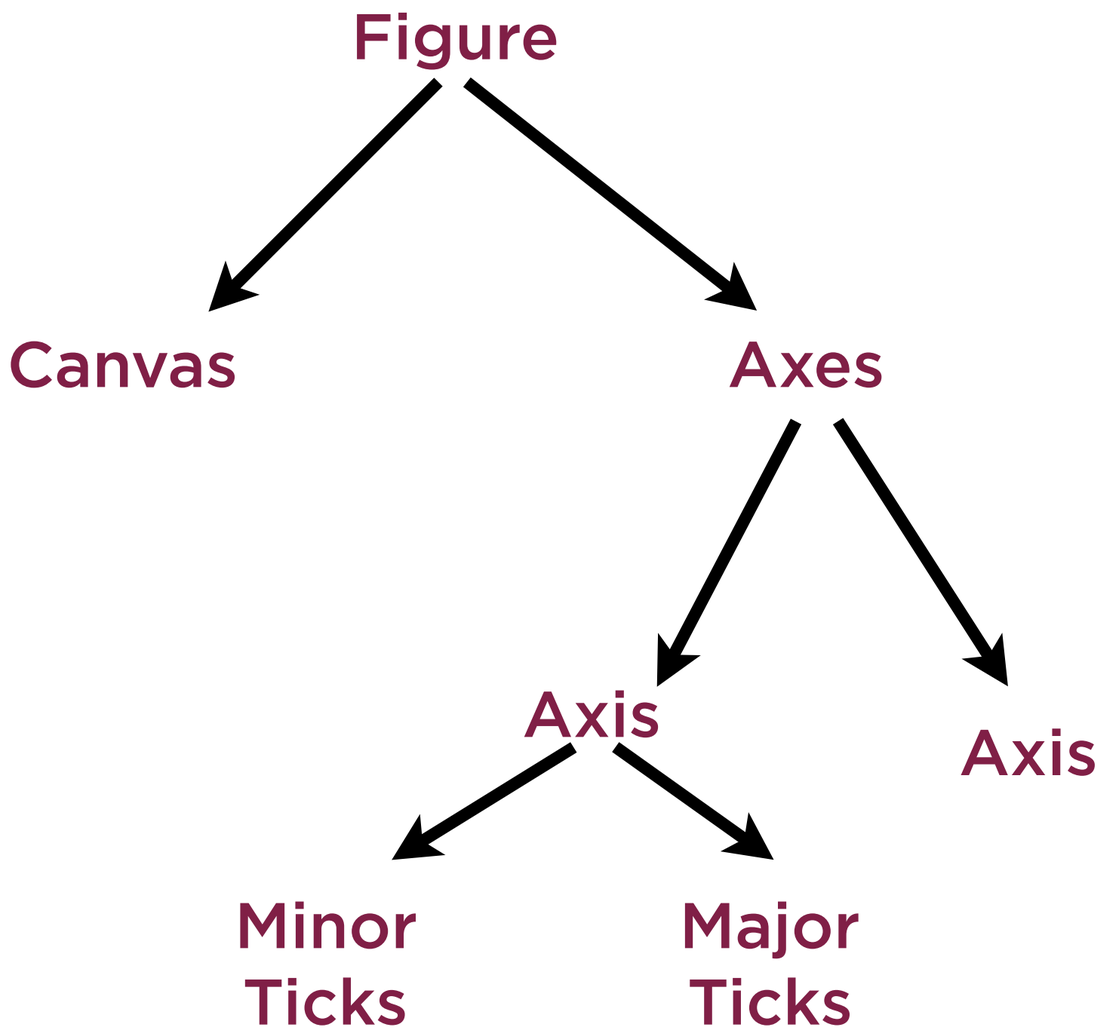


Hierarchy

Figure is a top-level container

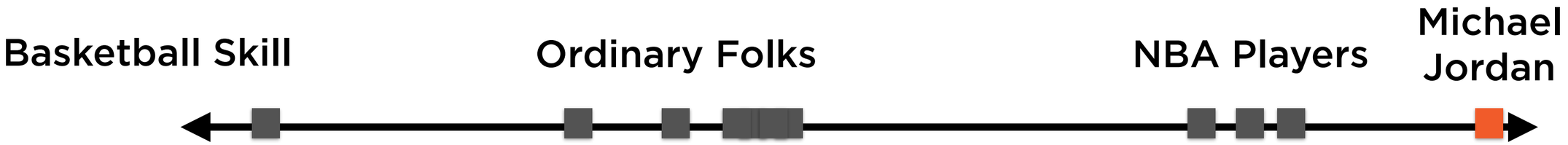
PyPlot APIs operate at higher levels

Matplotlib APIs at lower levels



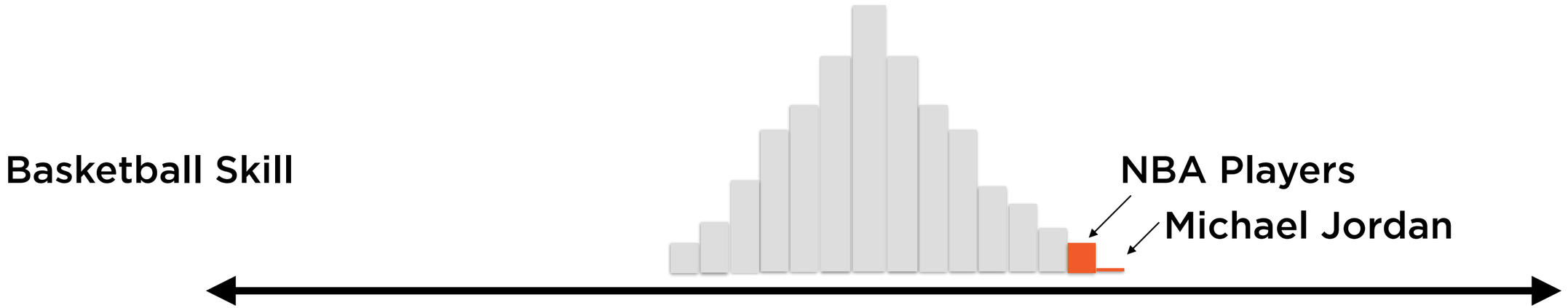
“Michael Jordan is a once-in-a-lifetime player”

Outliers



A once-in-a-lifetime player is an outlier, a point far from the pack

Outliers

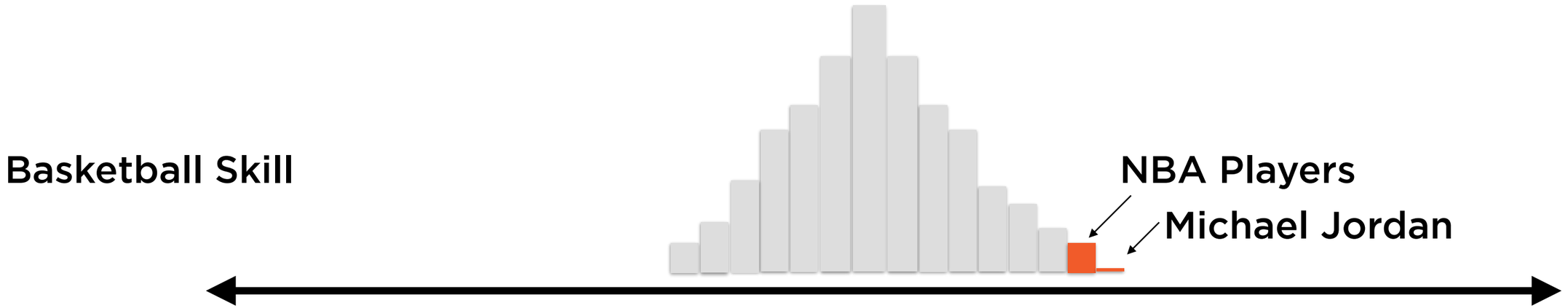


In reality, most ordinary folks would be clustered around an average level of skill

The NBA players would be outliers

Michael Jordan would be an even greater outlier

Outliers

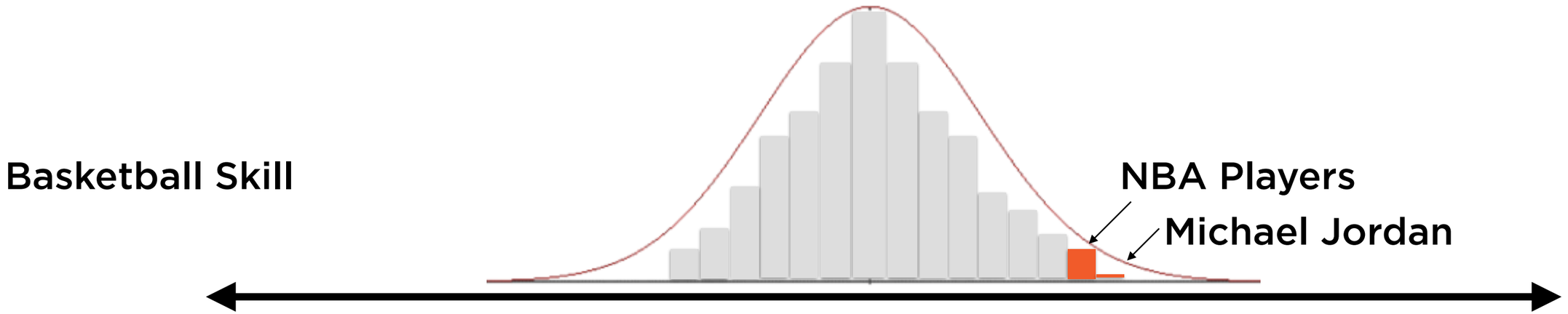


This chart above tells us how common a specific level of skill is

The shape of this chart resembles a bell

This is a Normal Probability Distribution

Outliers



This chart above tells us how common a specific level of skill is

The shape of this chart resembles a bell

This is a Normal Probability Distribution

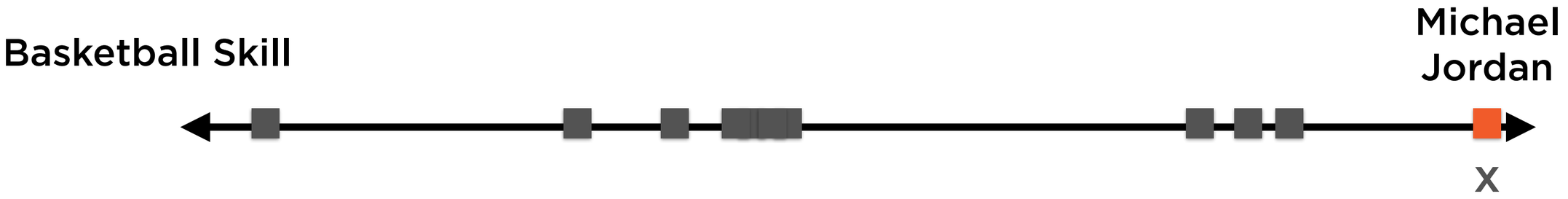
Outliers

Average is common

Very high and very low are both unusual

The bell curve occurs everywhere in nature

Outliers



What is the probability of any specific value x occurring in the data?

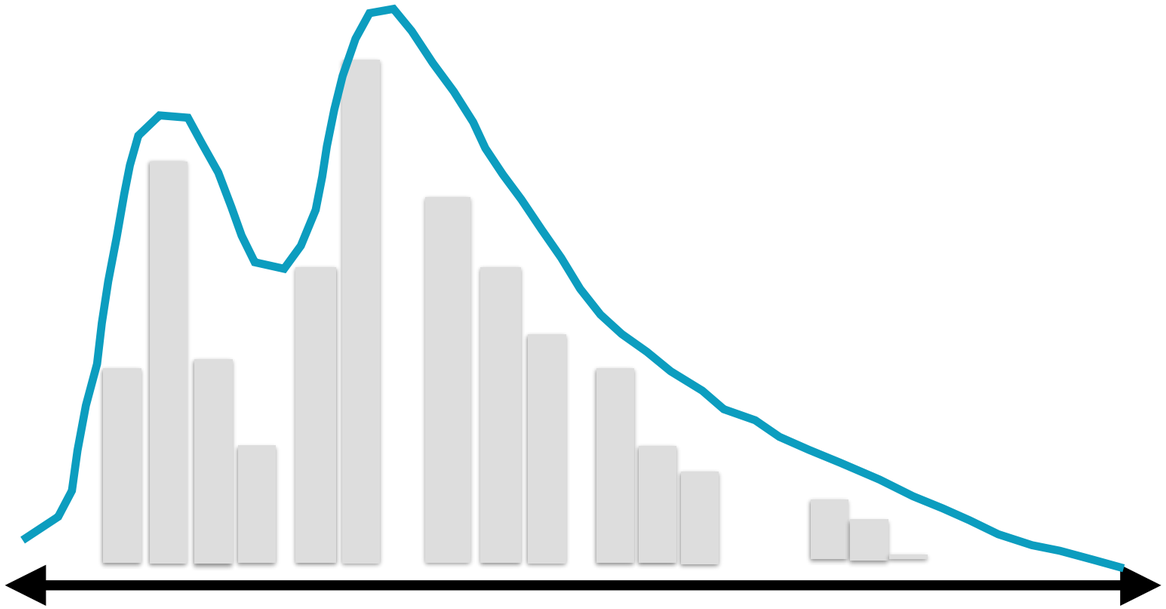
The answer lies in a probability distribution function

Kernel Density Estimation

Given a set of points

Figure out their probability distribution

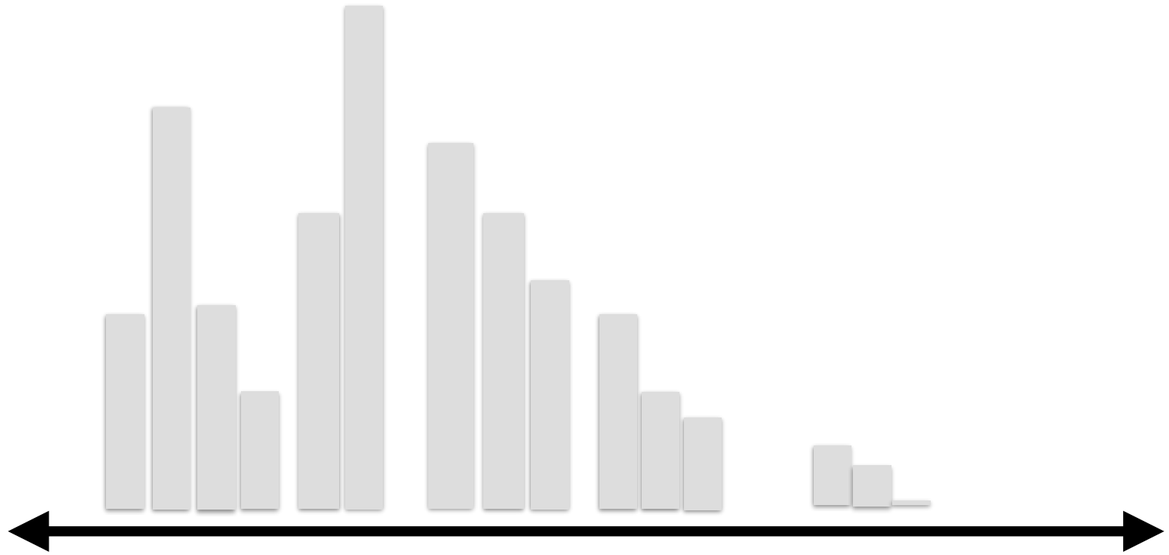
Area under curve must sum to 1



Kernel Density Estimation

KDE is a standard technique

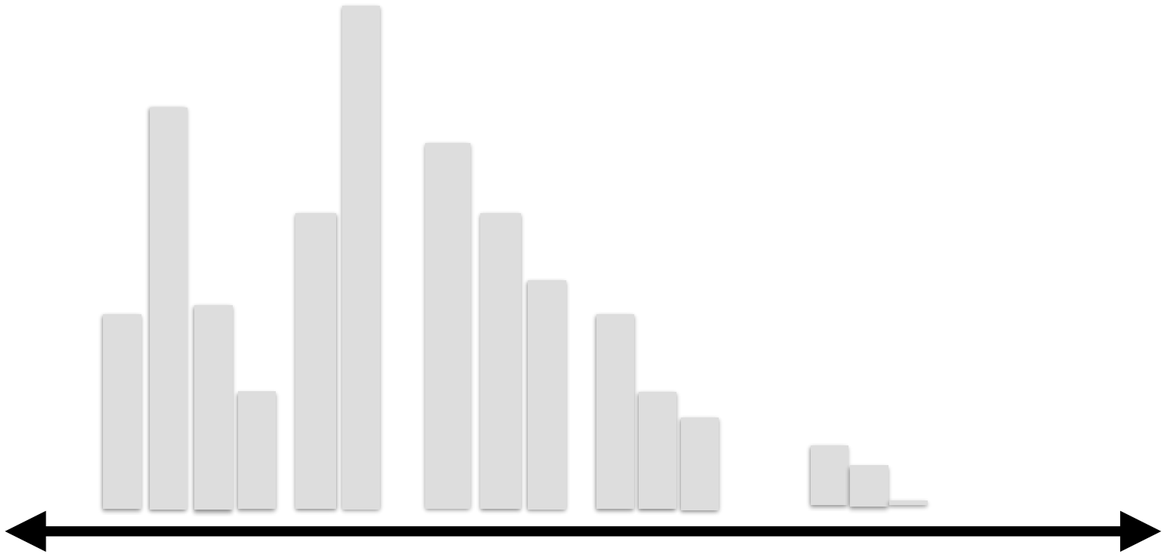
Non-parametric “Smoothing” technique



Kernel Density Estimation

Assume points have same distribution

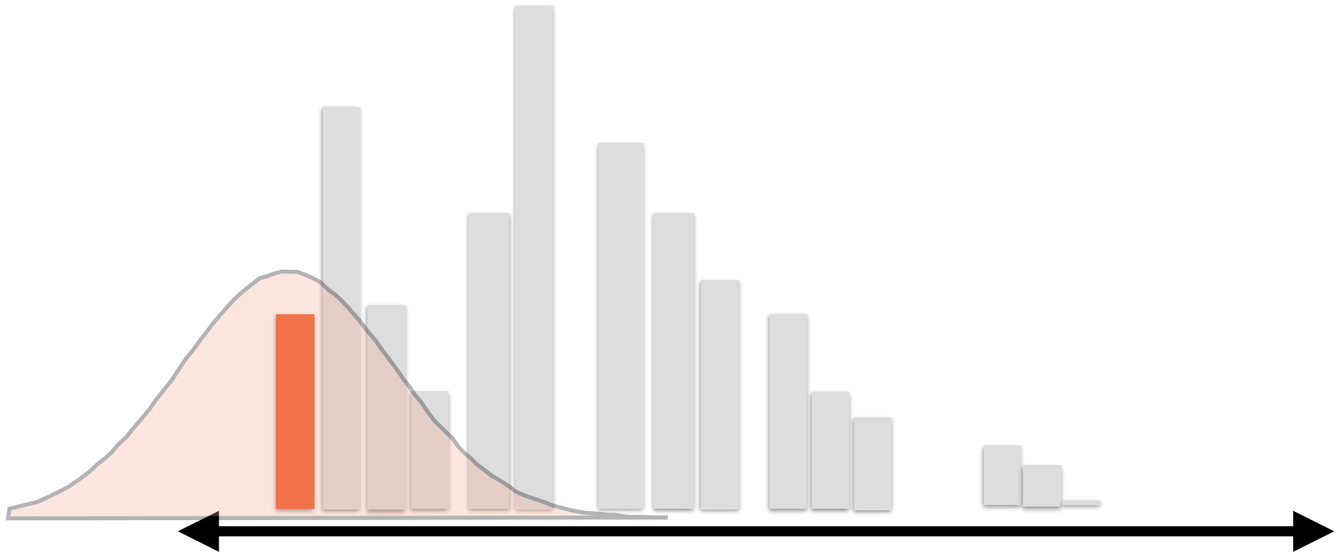
“Independent Identically Distributed”



Kernel Density Estimation

Assume points have same distribution

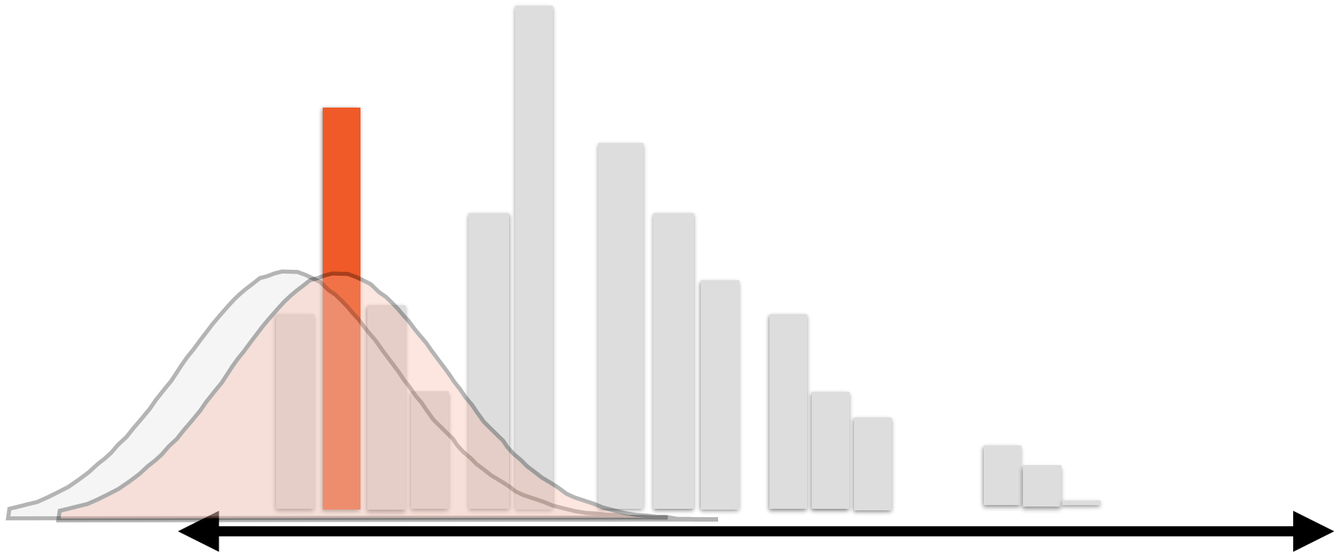
“Independent Identically Distributed”



Kernel Density Estimation

Assume points have same distribution

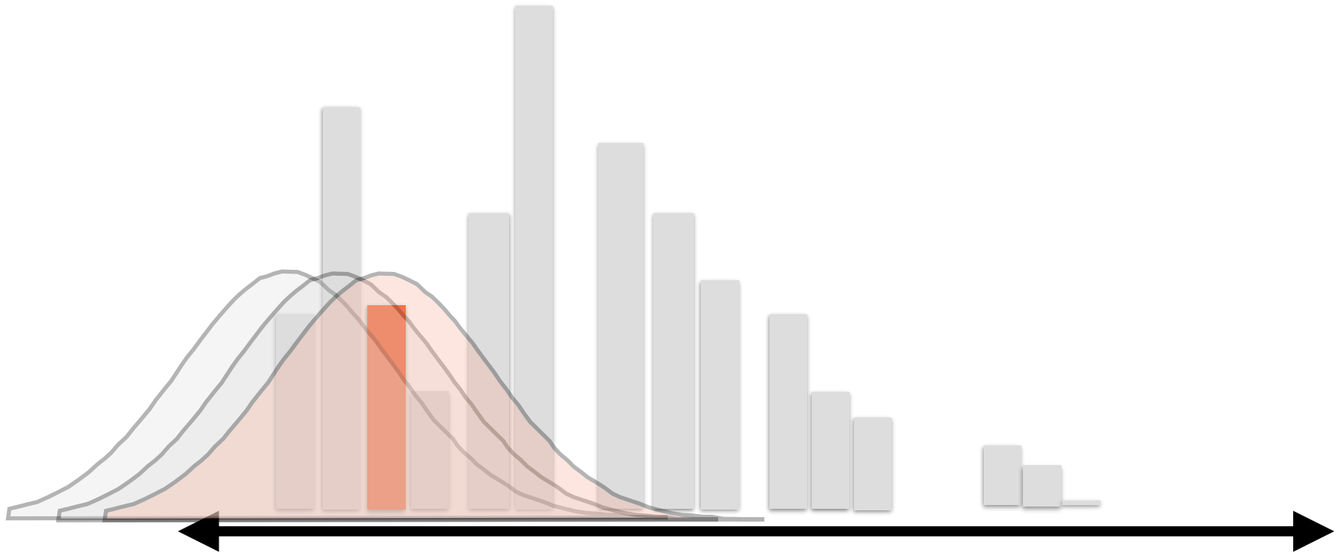
“Independent Identically Distributed”



Kernel Density Estimation

Assume points have same distribution

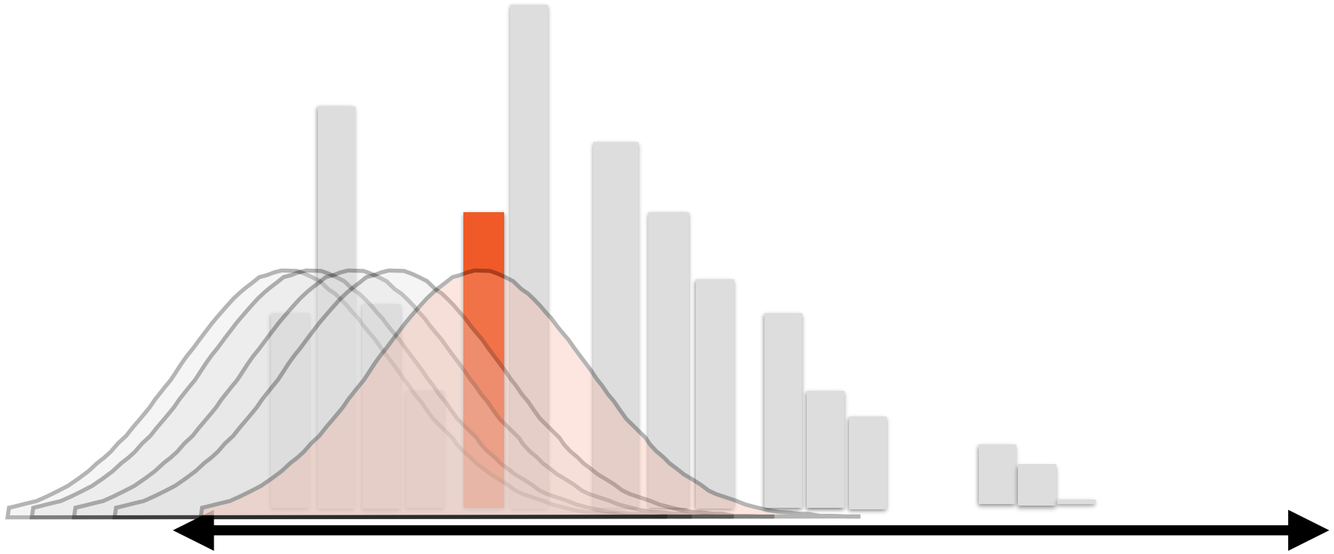
“Independent Identically Distributed”



Kernel Density Estimation

Assume points have same distribution

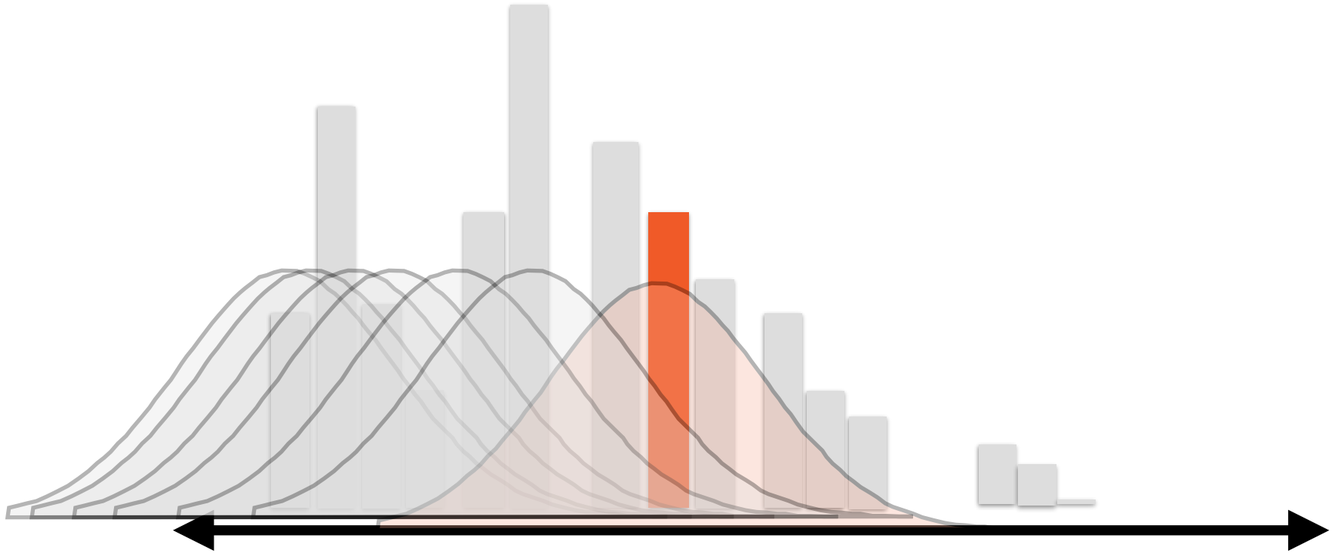
“Independent Identically Distributed”



Kernel Density Estimation

Assume points have same distribution

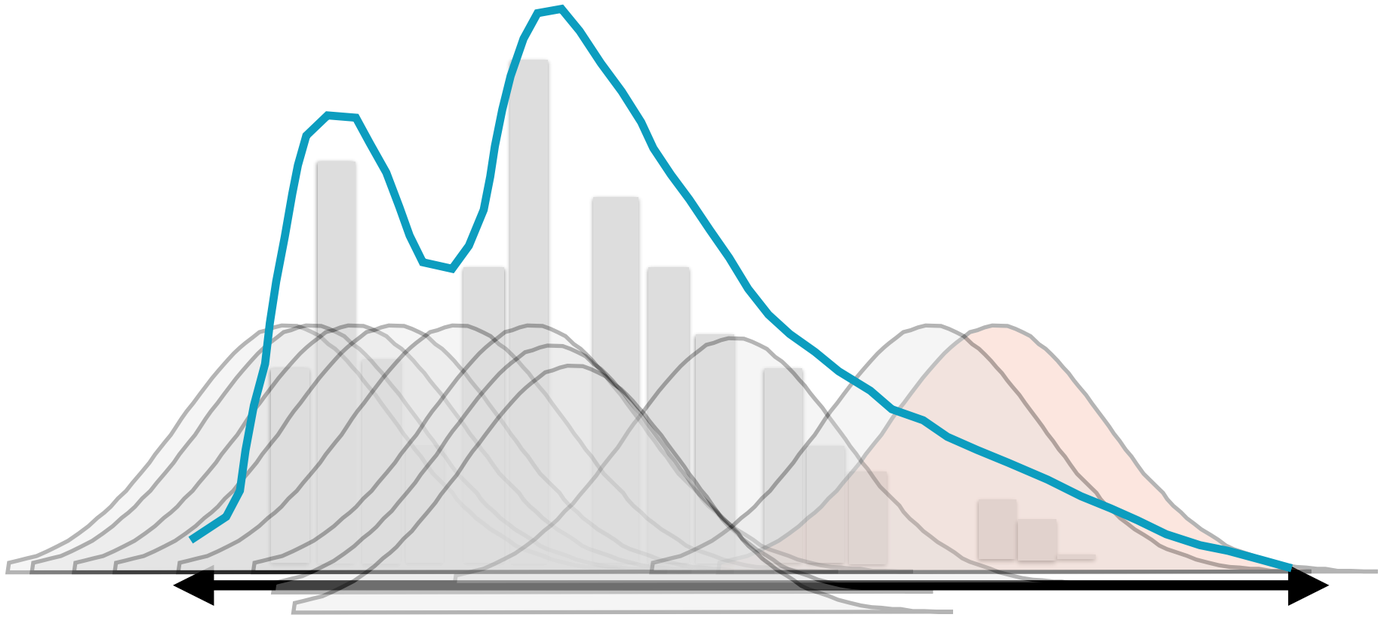
“Independent Identically Distributed”



Kernel Density Estimation

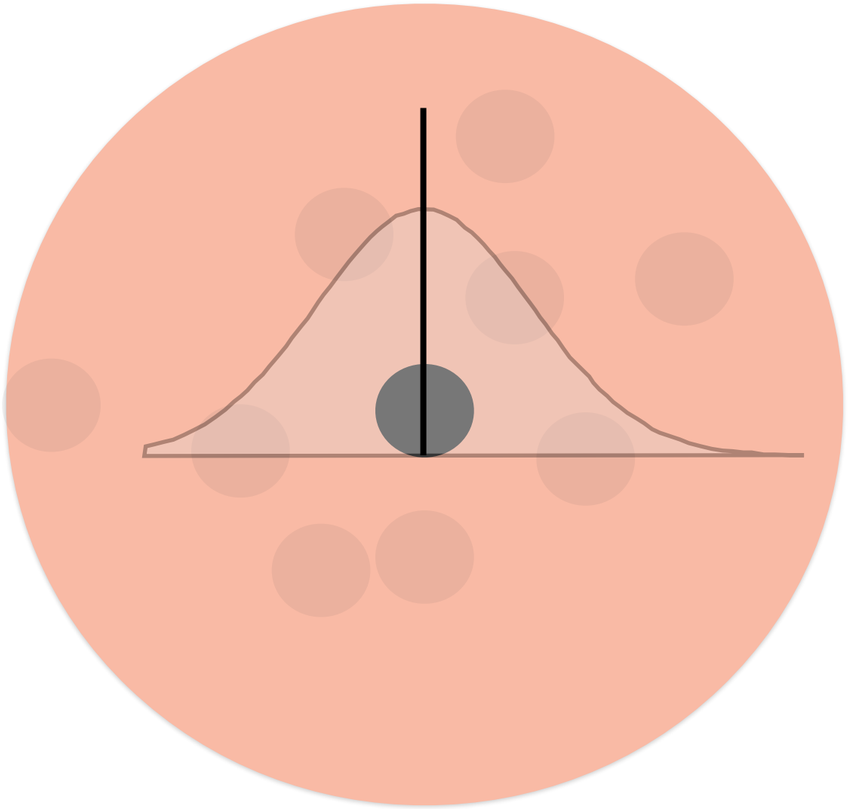
“Sum” them all up

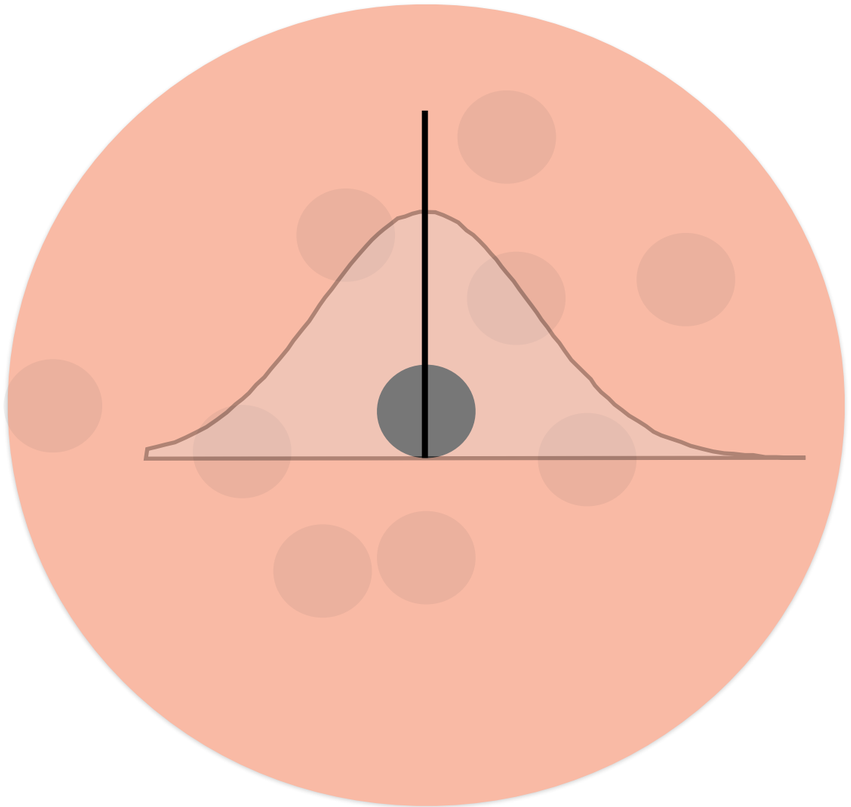
Get resulting PDF of data



Kernel Density Estimation

Fit distribution from histogram





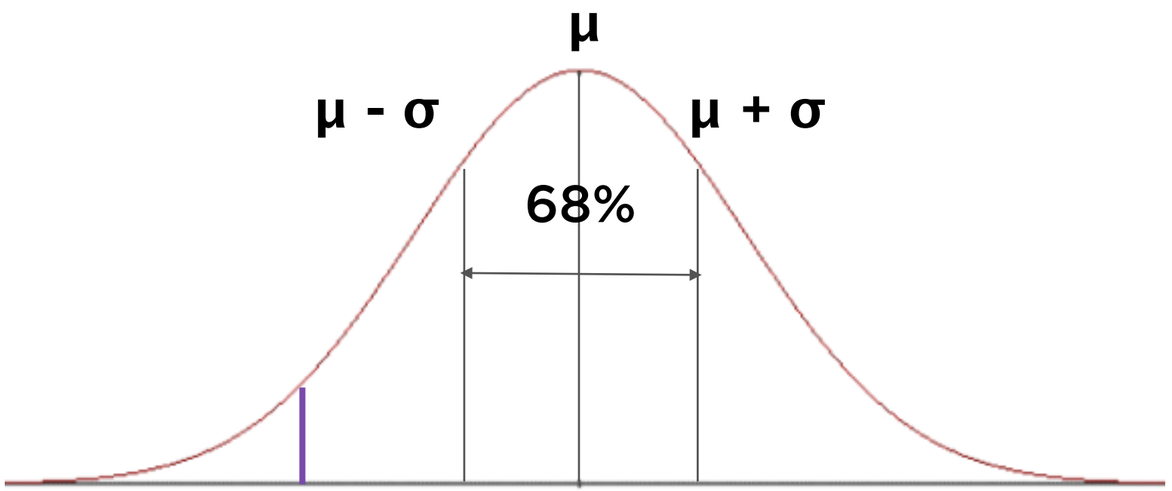
Gaussian Kernel

Gaussian probability distribution

Defined by

- mean μ
- standard deviation σ

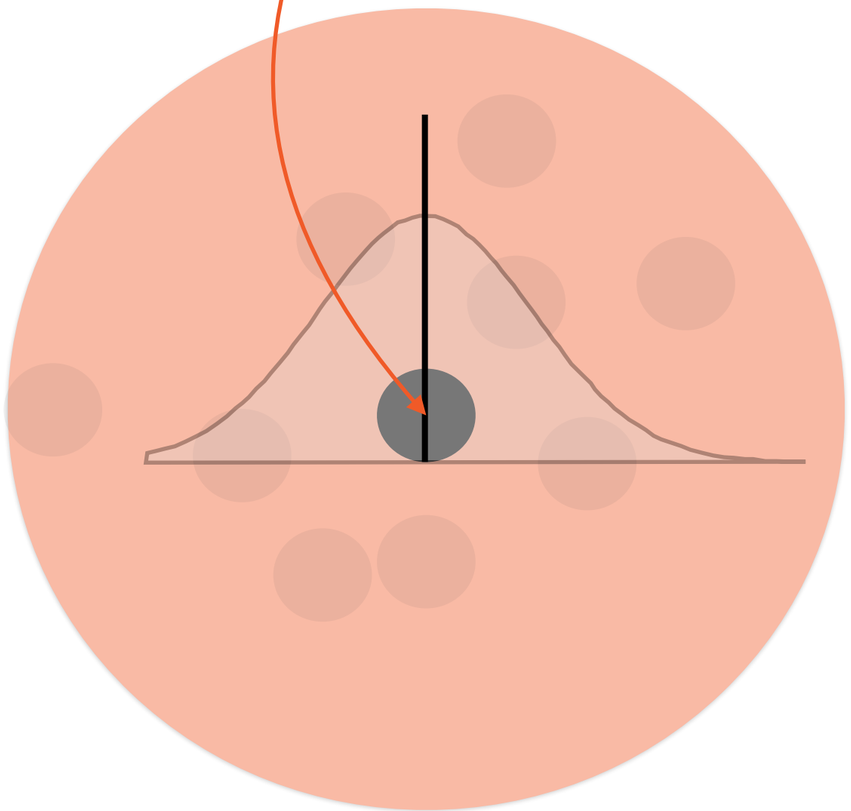
Gaussian Distribution



$N(\mu,\sigma)$

$$N(\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Mean = Center point



Gaussian Kernel

Mean μ = center point

Standard deviation σ ~ bandwidth

(Bandwidth is a hyperparameter)

Demo

Installing Seaborn

Exploring the wine dataset

Demo

Distplots

KDE plots

Joint plots

Pair plots

Heat maps

Demo

- Implots for linear relationships
- Regplots
- Controlling size and shape of plots
- Combination plots

Demo

Categorical plots

Box plots

Statistical estimation within categories

Wide form data

Factor plots

Summary

Seaborn is a powerful visualization library

Built on top of Matplotlib

Tightly integrated with PyData stack

Matplotlib seeks to “make easy things easy and hard things possible”

Seaborn is a complement

Makes “production ready” plots