

Data Visualization in Python



QB Bootcamp, Day 3
Friday, 7 September 2023
11:30am - 12:00pm

Matplotlib is the foundation for most graphics in Python



Version 3.1.1

[home](#) | [examples](#) | [tutorials](#) | [API](#) | [contents](#) »

[modules](#) | [index](#)

Fork me on GitHub

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.



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. For examples, see the [sample plots](#) and [thumbnail gallery](#).

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Installation

Visit the [Matplotlib installation instructions](#).

Documentation

This is the documentation for Matplotlib version 3.1.1.

To get started, read the [User's Guide](#).

Other versions are available:

- [3.1.1](#) Stable version.
- [2.2.4 LTS](#) LTS version.
- [3.2.x](#) Latest git master (unstable)
- [3.0.3](#) Older stable version.

Trying to learn how to do a particular kind of plot? Check out the [examples gallery](#) or the [list of plotting commands](#).

Other learning resources

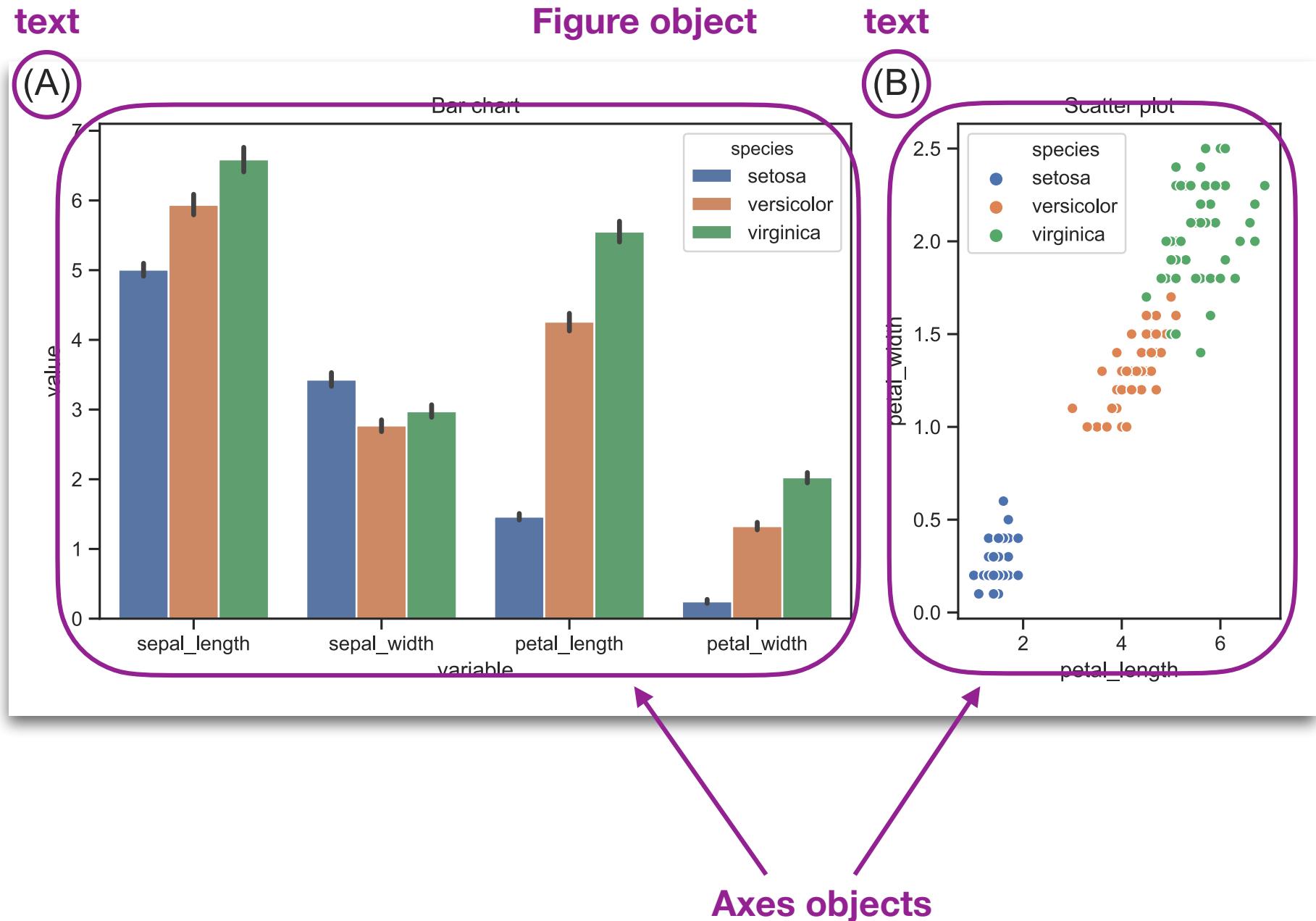
There are many [external learning resources](#) available including printed material, videos and tutorials.

Quick search

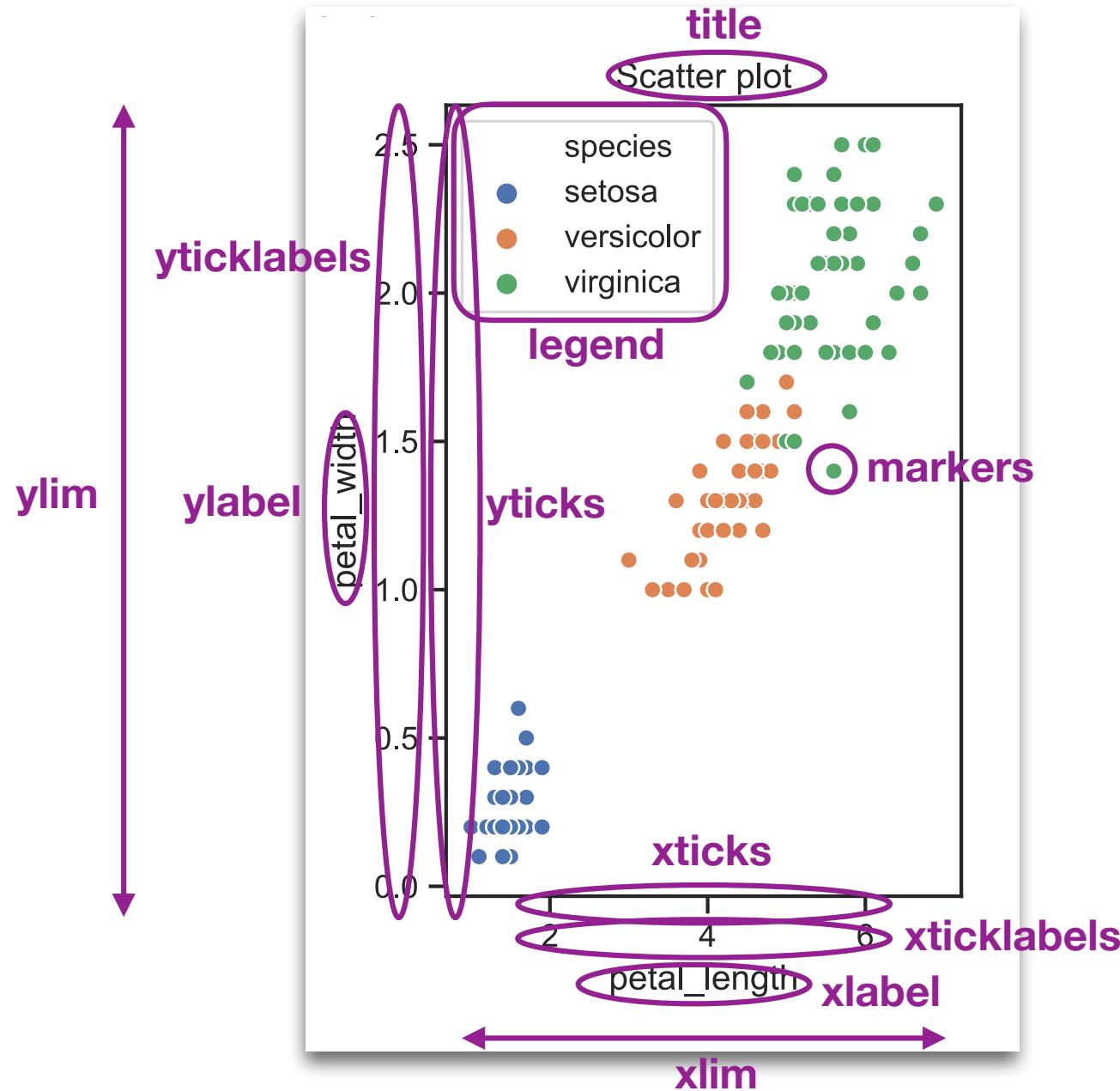
Matplotlib 3.0 is Python 3 only.
For Python 2 support,
Matplotlib 2.2.x will be
continued as a LTS release
and updated with bugfixes
until January 1, 2020.

[Support Matplotlib](#)

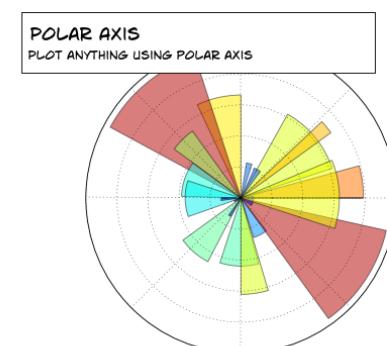
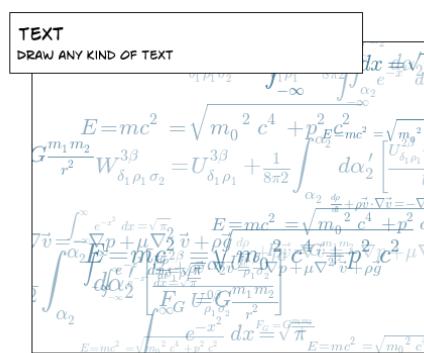
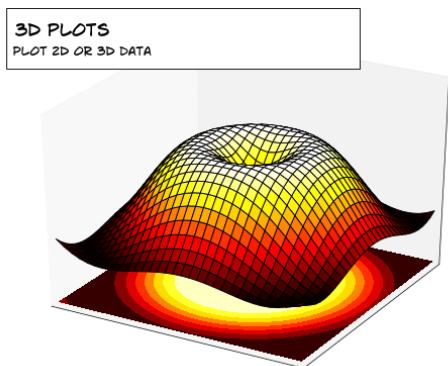
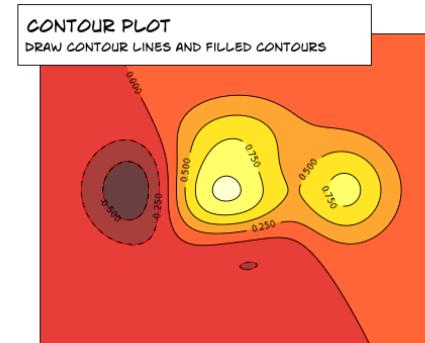
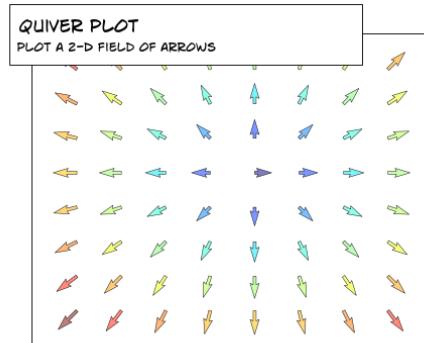
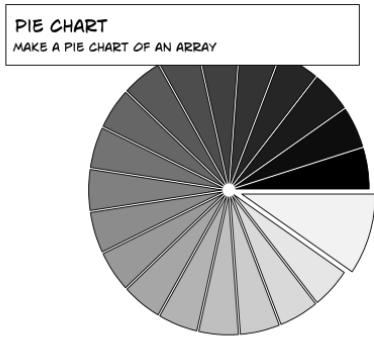
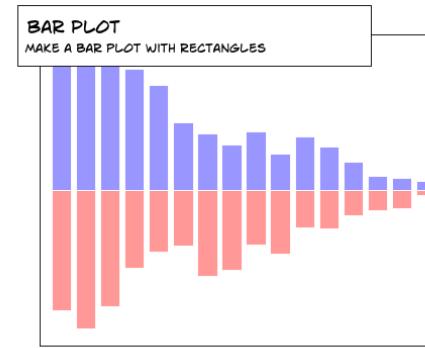
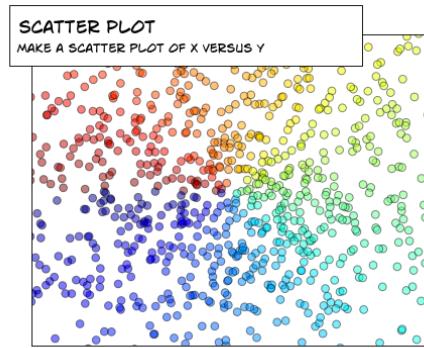
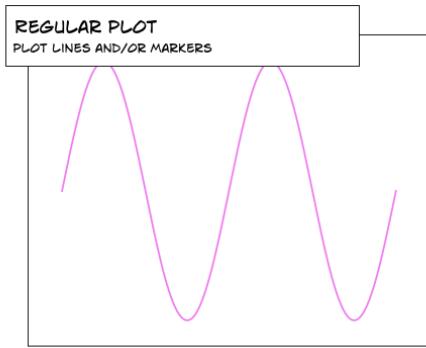
Matplotlib figures contain one or more Axes objs



Each Axes object contains many individual elements that can be adjusted



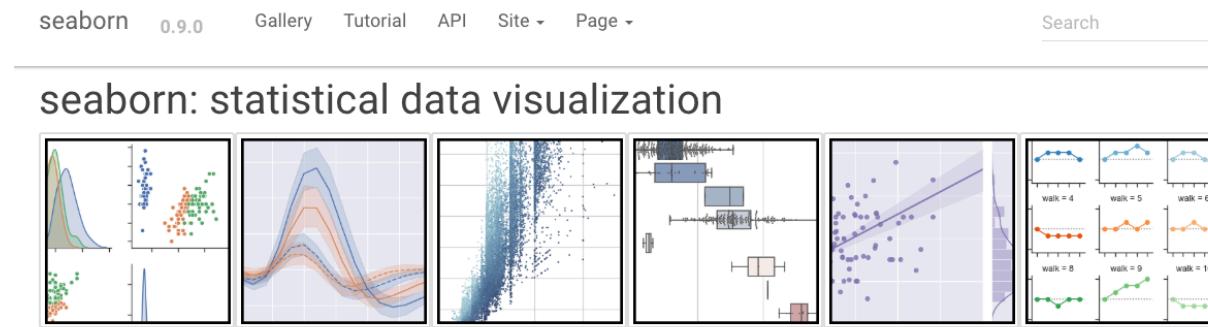
There are many, many types of plots that one can make in Matplotlib



Seaborn facilitates rapid data visualization

Seaborn is a *wrapper* for Matplotlib.

It provides methods for rapidly generating a wide range of decent-looking plots from data stored in Pandas dataframes.



seaborn 0.9.0 [Gallery](#) [Tutorial](#) [API](#) Site ▾ Page ▾ Search

seaborn: statistical data visualization

Seaborn is a Python data visualization library based on [matplotlib](#). It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the [introductory notes](#). Visit the [installation page](#) to see how you can download the package. You can browse the [example gallery](#) to see what you can do with seaborn, and then check out the [tutorial](#) and [API reference](#) to find out how.

To see the code or report a bug, please visit the [github repository](#). General support issues are most at home on [stackoverflow](#), where there is a seaborn tag.

Contents

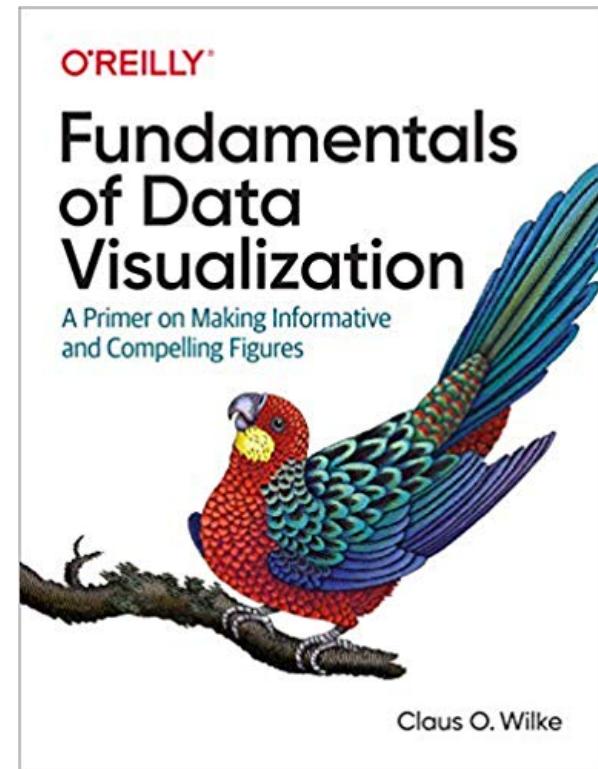
- [Introduction](#)
- [Release notes](#)
- [Installing](#)
- [Example gallery](#)
- [Tutorial](#)
- [API reference](#)

Features

- Relational: [API](#) | [Tutorial](#)
- Categorical: [API](#) | [Tutorial](#)
- Distributions: [API](#) | [Tutorial](#)
- Regressions: [API](#) | [Tutorial](#)
- Multiples: [API](#) | [Tutorial](#)
- Style: [API](#) | [Tutorial](#)
- Color: [API](#) | [Tutorial](#)

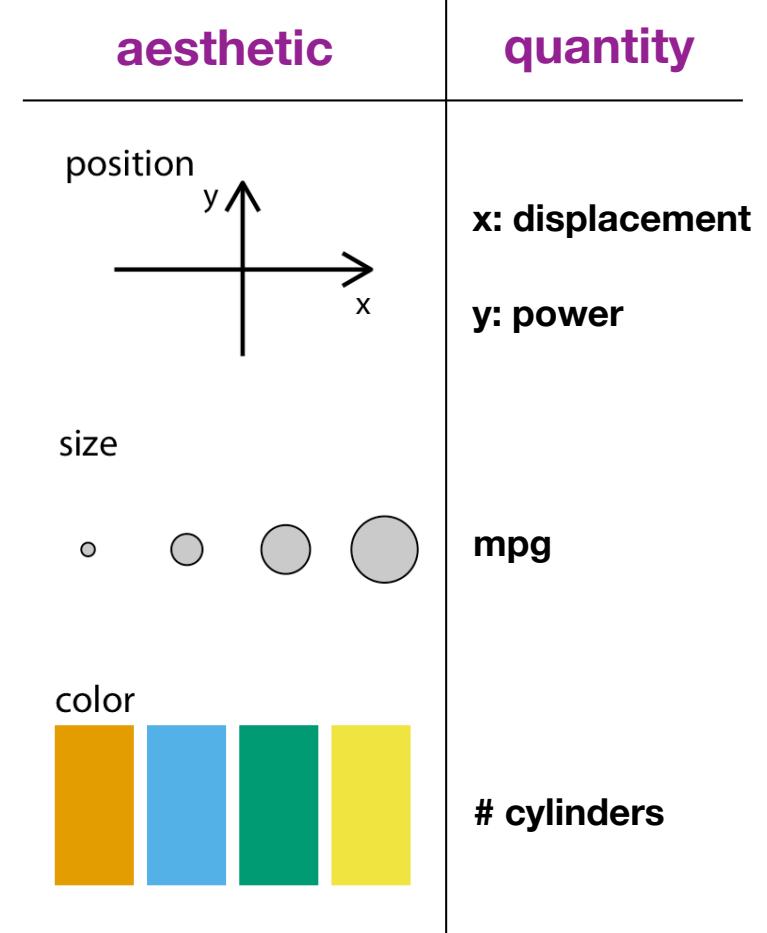
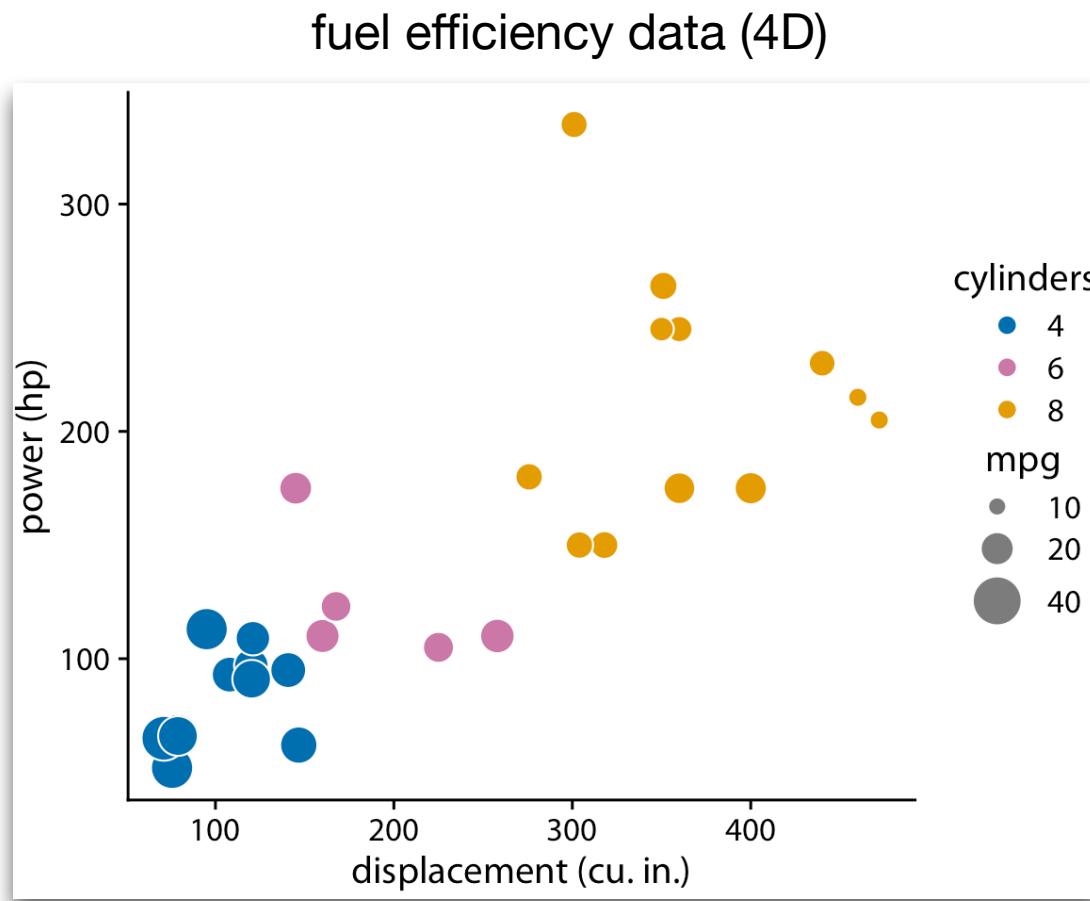
Data visualization resources

Claus Wilke gives excellent guidance
on data visualization do's and don'ts



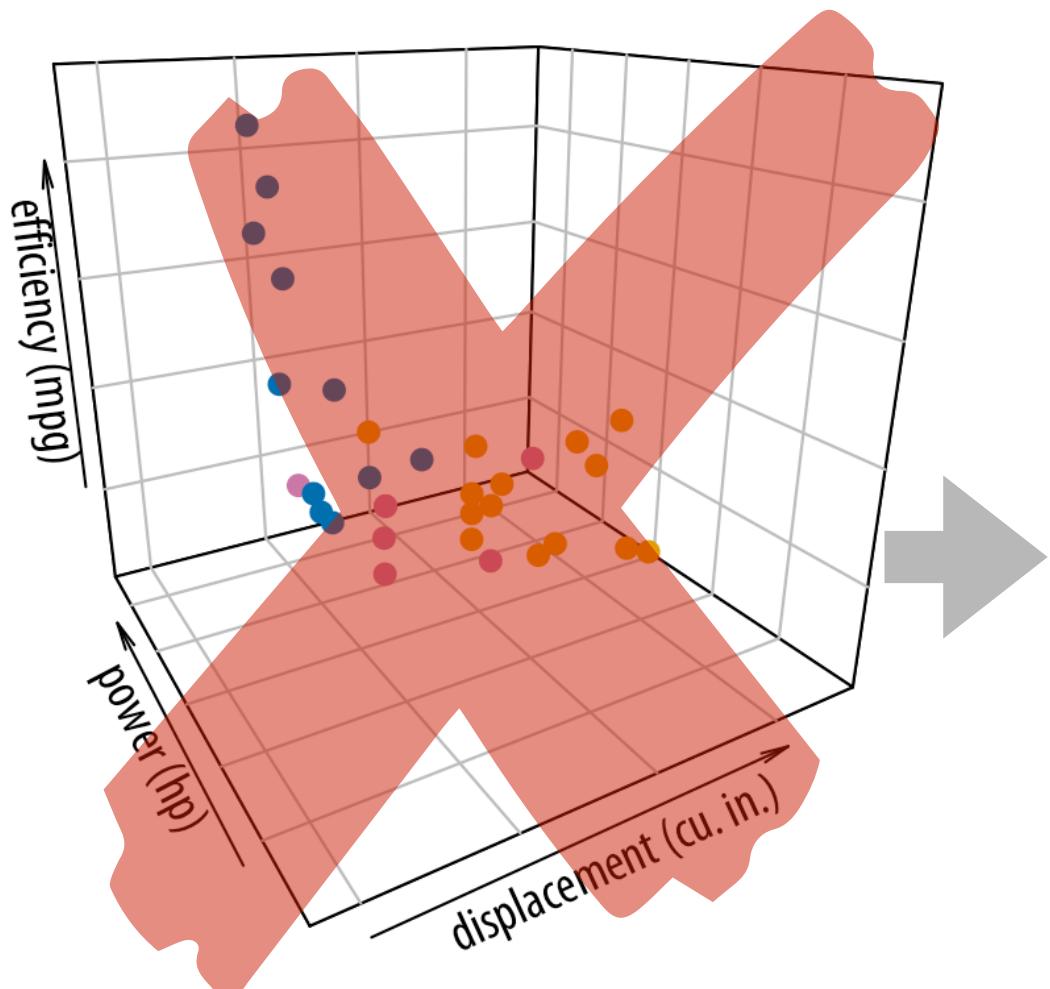
**Fundamentals of
Data Visualization**
Wilke, 2019

All data visualization uses “aesthetics” to encode quantitative information

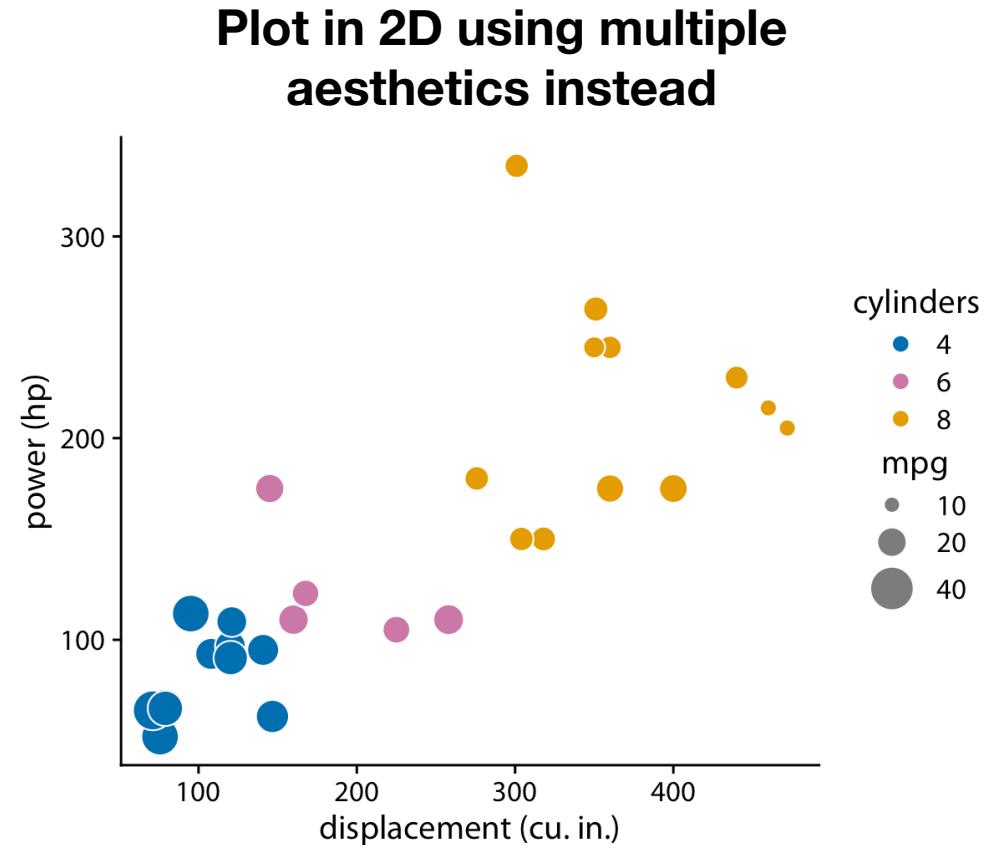


Please don't plot data in 3D if you can help it

In a 3D figure, it's virtually impossible to figure out the coordinates of each point by eye.

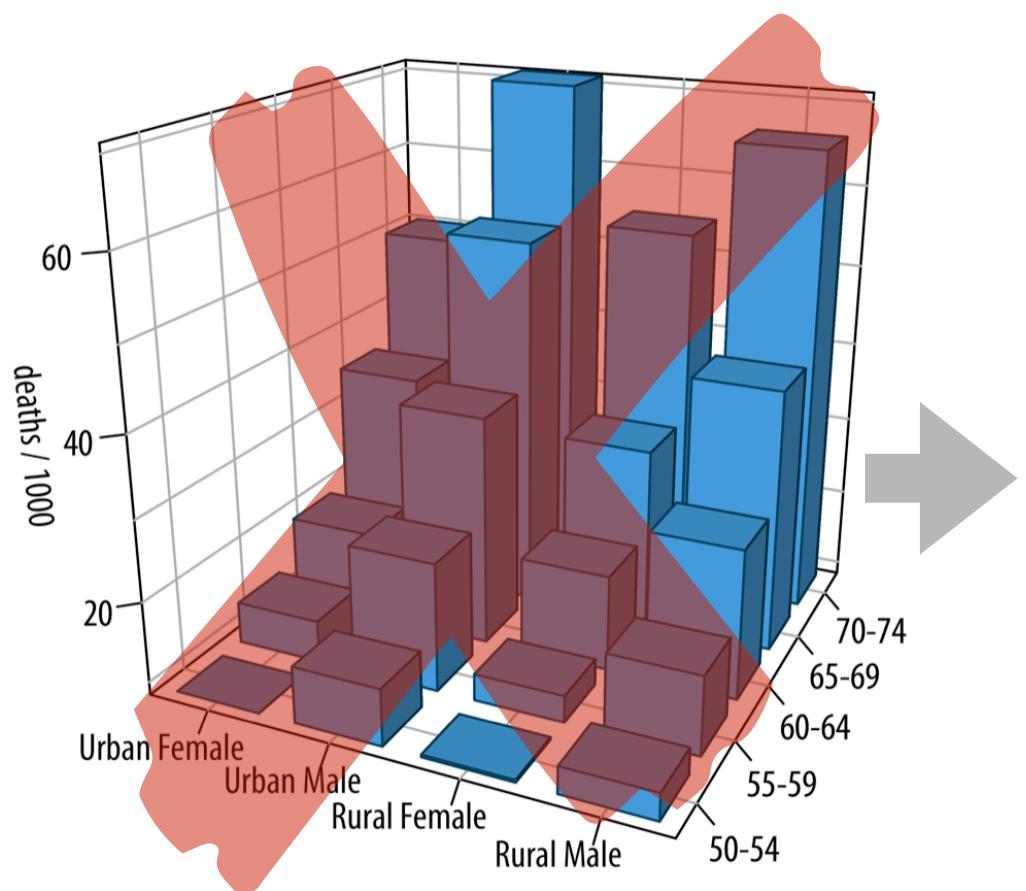


cylinders • 4 • 6 • 8

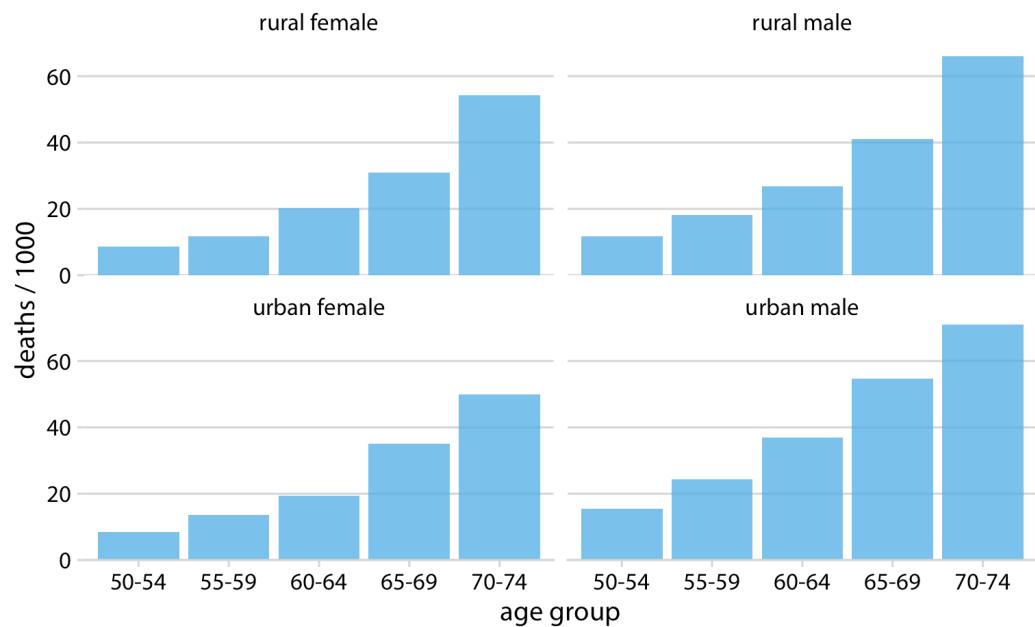


Please don't plot data in 3D if you can help it

In a 3D figure, it's virtually impossible to figure out the coordinates of each point by eye.

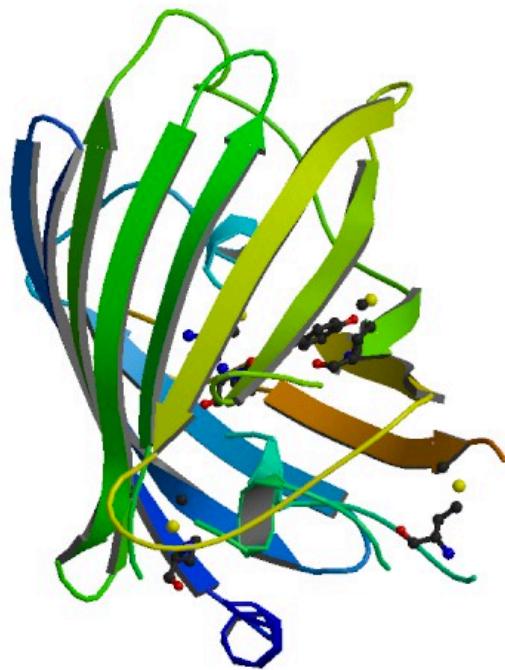


Use “small multiples” of 2D plots

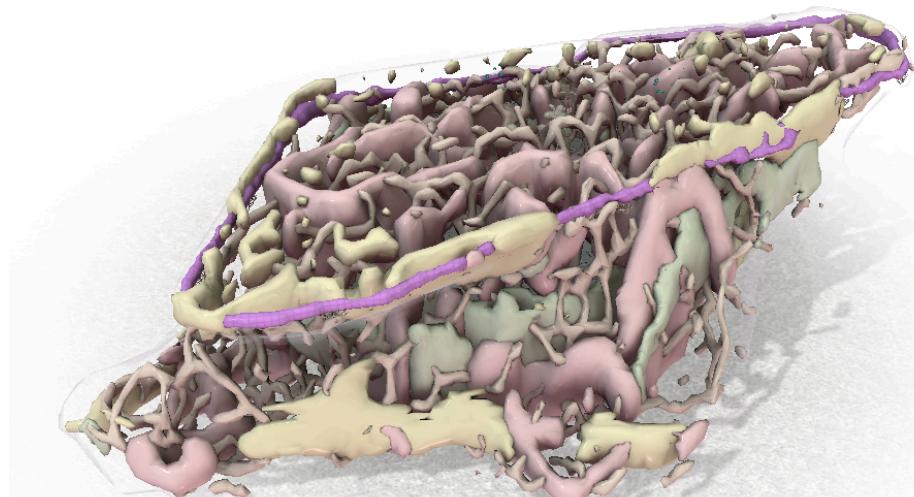


Only use 3D if you're illustrating some inherently 3D object

Structure of GFP
(PDB:1EMA)



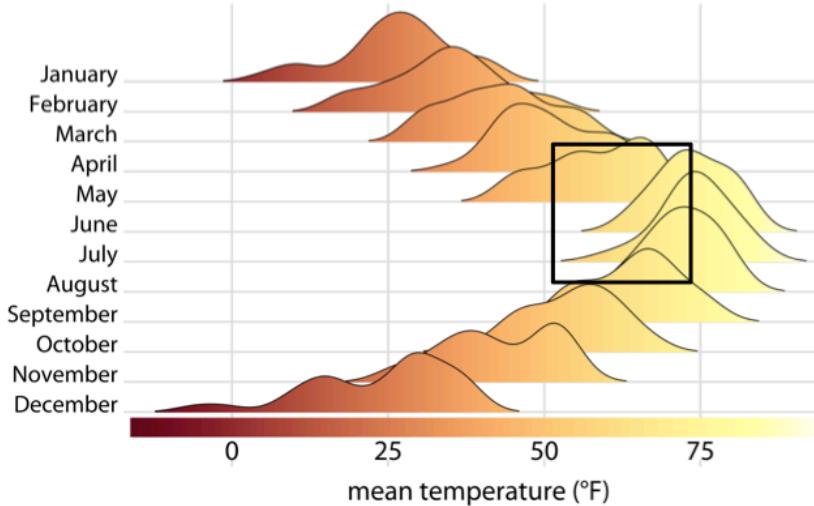
cellular structures
(Allen Cell Explorer)



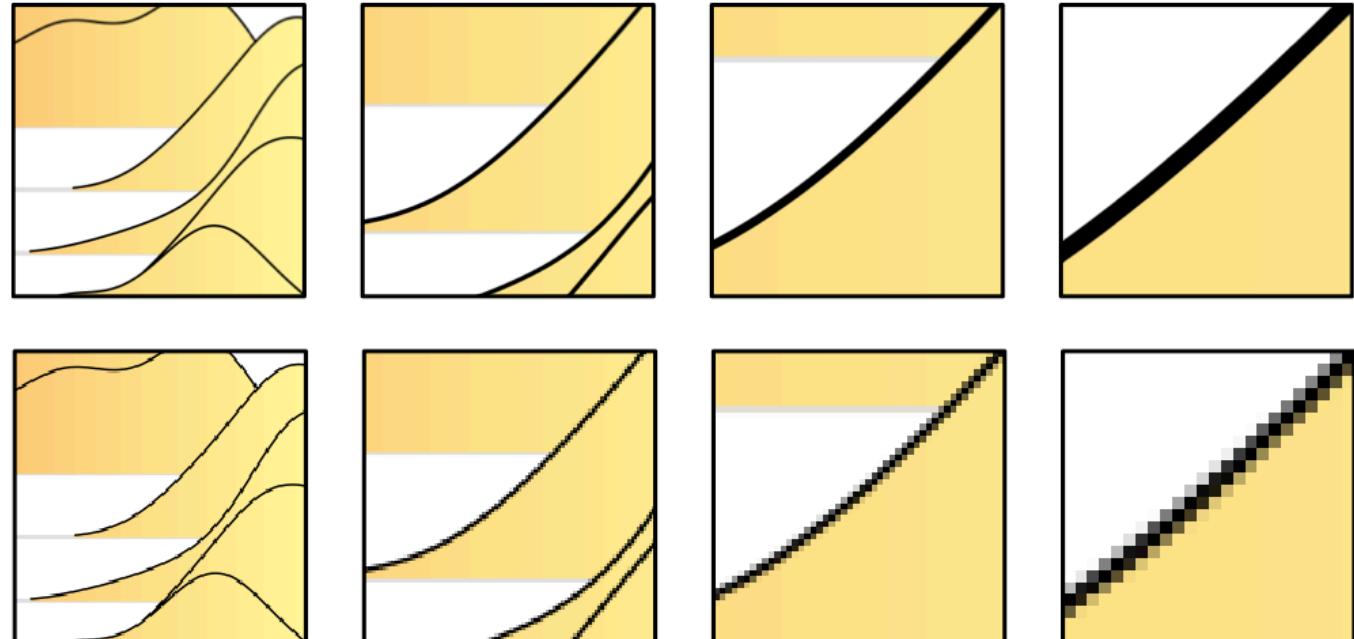
<https://www.rcsb.org/structure/1ema>

<https://www.allencell.org/visual-guide-to-human-cells.html>

Keep in mind the difference between vector graphics and bitmaps



vector format
(PDF, SVG)



bitmap format
(TIFF, PNG, JPEG)

JPEG is often good for compressing images, but can produce strange artifacts



Quick guide to graphics file formats

PDF

- Standard vector graphics format
- Easily imported and edited in Illustrator
- Bad for plotting many datapoints

PNG

- Lossless bitmap compression
- Standard bitmap format for plots
- Doesn't compress natural images well

JPEG

- Lossy image compression
- Standard bitmap format for images
- Can produce strange artifacts in plots

Tips on making figures

- Be a perfectionist.
- Draw your figures so that they look like figures in your target journal.
- Use Adobe Illustrator.
- Make all figures the correct size to begin with (use size guidelines for target journal).
- Make all figures in a single .ai file.
- Computationally generate plots so that they can be “placed” within the .ai file behind any artwork, and without requiring any modification.
- Write scripts to automate the generating of publication-ready plots.
- Separate your computationally expensive analysis scripts from your figure-generating scripts.

