

Section 1

MANE 3351

Subsection 1

Lecture 3, August 30

Classroom Management

Agenda

- Attendance
- Textbook

Subsection 2

Resources

Handouts

- [Lecture 3 Slides](#)
- [Lecture 3 Marked Slides](#)

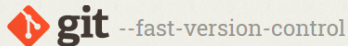
Calendar

Lecture	Date	Content
1	August 23	Welcome to Class, Syllabus
2	August 25	Error Analysis
3	August 30	Introduction to Jupyter Notebook

Assignments

- Set up PC to run Jupyter Notebook

Git



Git is a [free and open source](#) distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is [easy to learn](#) and has a [tiny footprint with lightning fast performance](#). It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like [cheap local branching](#), convenient [staging areas](#), and [multiple workflows](#).



Source

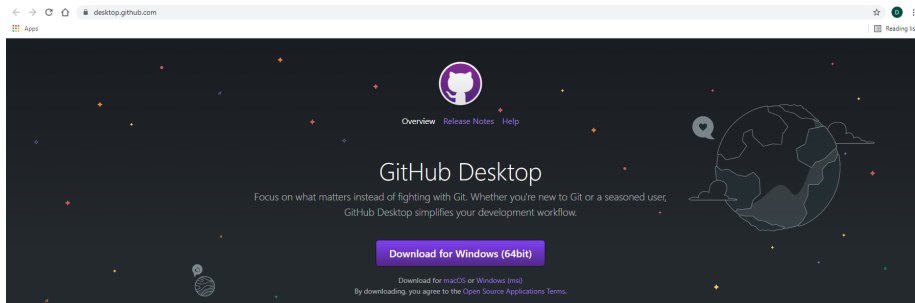
GitHub



The screenshot shows the Wikipedia article for GitHub. On the left is the Wikipedia logo and a sidebar with links like 'Main page', 'Contents', 'Current events', 'Random article', 'About Wikipedia', 'Contact us', and 'Donate'. The main content area has tabs for 'Article' and 'Talk'. The title 'GitHub' is prominently displayed, followed by the text 'From Wikipedia, the free encyclopedia'. A note states 'Not to be confused with Git.' The article text begins with 'GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project.^[4] Headquartered in California, it has been a subsidiary of Microsoft since 2018.^[5]' It continues with 'It is commonly used to host open-source projects.^[6] As of January 2020, GitHub reports having over 40 million users^[7] and more than 190 million repositories^[8] (including at least 28 million public repositories).^[9] It is the largest source code host as of April 2020.^[10]' On the right, there is a user profile icon labeled 'Not logged in' and buttons for 'Read', 'Edit', 'View history', and 'Search'.

- Source
- Classroom lecture materials will be available at http://github.com/douglastimmer/MANE3351_Fall2021
- Easiest approach is download on PC or Mac using gitHub Desktop
- For Raspberry Pi, git must be used (command line)
- Laboratories will be handled slightly differently (more later)

GitHub Desktop



- Source
- Graphical interface that makes using GitHub very easy to use
- Downloading lecture materials
 - First use: clone repository
 - Subsequent uses: pull repository (will update local material)
- Please do not push

Python with Jupyter Notebook

← → ↻ 🏠 anaconda.com/products/individual

Apps



Products ▾

Pricing

Solutions ▾

Resources ▾

Blog

Company ▾

Get Started



Individual Edition

Your data science toolkit

With over 25 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

Anaconda Individual Edition

Download 

For Windows

Python 3.8 • 64-Bit Graphical Installer • 477 MB

Get Additional Installers



Open Source

Anaconda Individual Edition is the world's most popular Python distribution platform with over 25



Conda Packages

Search our cloud-based repository to find and install over 7,500 data science and machine learning packages. With



Manage Environments

Individual Edition is an open source, flexible solution that provides the utilities to build, distribute, install,

GitHub Demonstration

Standard Normal, Case 1

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as sct
import math
```

```
a=0.5
```

```
x=np.linspace(-4,4,500)
y=sct.norm.pdf(x,0,1)
y2=0.0*x
maske =(x<a)
```

```
plt.plot(x,y,'b')
plt.fill_between(x,y,color='#666666',where=maske)
plt.plot(x,y2,'b')
plt.show()
```

```
answer=sct.norm.cdf(a,0,1)
```

First 4 Lines

- Imports allow external packages to be used
- Most standard packages are included in the Anaconda installation
 - **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”^[1]
 - **NumPy** “is the fundamental package for scientific computing with Python. It contains among other things: 1). a powerful N-dimensional array object, 2). sophisticated (broadcasting) functions, 3). tools for integrating C/C++ and Fortran code, and 4). useful linear algebra, Fourier transform, and random number capabilities.”^[2]
 - **SciPy** “is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages: NumPy, SciPy library, Matplotlib, IPython, SymPy, and pandas.”^[3]
 - **Math** “provides access to the mathematical functions defined by the C standard.

Python Libraries

Numpy Linspace

numpy.linspace

`numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)`

Return evenly spaced numbers over a specified interval.

[\[source\]](#)

Returns *num* evenly spaced samples, calculated over the interval *[start, stop]*.

The endpoint of the interval can optionally be excluded.

Changed in version 1.16.0: Non-scalar *start* and *stop* are now supported.

Changed in version 1.20.0: Values are rounded towards `-inf` instead of `0` when an integer `dtype` is specified. The old behavior can still be obtained with `np.linspace(start, stop, num).astype(int)`

- Source
- Experiment with endpoint

scipy.stats.norm

scipy.stats.norm

`scipy.stats.norm = <scipy.stats._continuous_distns.norm_gen object>`

[\[source\]](#)

A normal continuous random variable.

The location (**loc**) keyword specifies the mean. The scale (**scale**) keyword specifies the standard deviation.

As an instance of the **rv_continuous** class, **norm** object inherits from it a collection of generic methods (see below for the full list), and completes them with details specific for this particular distribution.

Notes

The probability density function for **norm** is:

$$f(x) = \frac{\exp(-x^2/2)}{\sqrt{2\pi}}$$

for a real number x .

The probability density above is defined in the “standardized” form. To shift and/or scale the distribution use the **loc** and **scale** parameters. Specifically, `norm.pdf(x, loc, scale)` is identically equivalent to `norm.pdf(y) / scale` with $y = (x - \text{loc}) / \text{scale}$. Note that shifting the location of a distribution does not make it a “noncentral” distribution; noncentral generalizations of some

Matplotlib

[Fork me on GitHub](#)[Installation](#) [Documentation](#) [Examples](#) [Tutorials](#) [Contributing](#)[home](#) | [contents](#) » [User's Guide](#) » [Tutorials](#) » [Pyplot tutorial](#)[previous](#) | [next](#) | [modules](#) | [index](#)

Pyplot tutorial

An introduction to the pyplot interface.

Intro to pyplot

`matplotlib.pyplot` is a collection of functions that make matplotlib work like MATLAB. Each `pyplot` function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

In `matplotlib.pyplot` various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that "axes" here and in most places in the documentation refers to the axes part of a figure and not the strict mathematical term for more than one axis).

Note

the pyplot API is generally less-flexible than the object-oriented API. Most of the function calls you see here can also be called as methods from an `Axes` object. We recommend browsing the tutorials and examples to see how this works.

Generating visualizations with pyplot is very quick:

Source

Table of Contents

Pyplot tutorial

- [Intro to pyplot](#)
 - [Formatting the style of your plot](#)
- [Plotting with keyword strings](#)
- [Plotting with categorical variables](#)
- [Controlling line properties](#)
- [Working with multiple figures and axes](#)
- [Working with text](#)
 - [Using mathematical expressions in text](#)
 - [Annotating text](#)
- [Logarithmic and other nonlinear axes](#)

[Show Page Source](#)