Los Angeles, California

UCLA Anderson Math Bootcamp

Charles Rambo

UCLA Anderson

2024

Table of Contents I

About Me

About Me

If you have questions send them here

- charles.tutoring@gmail.com ← Send HW solutions here
- `github.com/charlesrambo` ← Course material is here
- `linkedin.com/in/charlesrambo`

Please email your questions to me at my gmail account. Also, please feel free to connect with me on LinkedIn!

Education

MFE from UCLA in 2020

BA in Mathematics from UC Berkeley in 2009

Experience

SSI Investment Management

Portfolio Research Analyst (3 years)

- Firm specializes in convertible securities
- I do valuation, credit, equity, and portfolio construction from a quantitative perspective.

Rambo Tutoring (Self-Employed)

Math tutor and author (10 years)

- Did tutoring for precalculus, calculus, linear algebra, differential equations, statistics, probability, and more!
- Wrote study material as well! Check out my stuff on Amazon.com!

How the Course Works

How the Course Works

- Videos emailed out ← maybe also link on Git Hub??
- Notes and homework posted at

  `github.com/charlesrambo/math_bootcamp_24`
- Grade of 70% or better to pass
- No exams

Homework

- Can work in teams of up to four people
- Make sure everybody's name is on the homework when you submit
- Email homework solutions to me at charles.tutoring@gmail.com
- Homework will be math problems which require some programming
- Please submit as an html or pdf file; no screenshots, ipynb, or py files please
- I look over the code but I don't run it so more important to make things readable than runable
- Due dates posted on homework
- Three assignments and one make-up assignment if you missed one or did badly

Tentative Course Outline

Unit	Description	Sessions
1	Calculus	July 9-18
2	Linear algebra and multivariable calculus	July 23-August 1
3	Combinatorics, probability, and statistics	August 6-15
4	Covariance matrices, PCA, and stochastic calculus	August 20-22

References

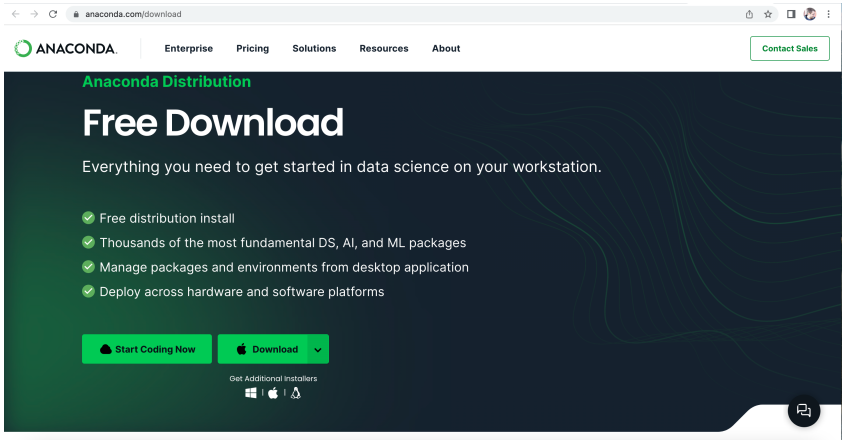
This is an incomplete list of references used to create the notes for this course. You do not need to purchase any of the books listed.

- James Stewart, *Calculus*, Brooke/Cole, 3rd ed., 1995.
- Walter Rudin, *Principles of Mathematical Analysis*, McGraw-Hill, 1976.
- Charles Pugh, *Real Mathematical Analysis*, Springer, 2002
- Serge Lang, *Linear Algebra*, Springer, 3rd ed., 1987.
- Steven Roman, *Advanced Linear Algebra*, Springer, 2nd ed., 2005.
- Morris DeGroot and Mark Schervish, *Probability and Statistics*, Pearson, 4th ed., 2013.
- Marcos Lopez de Prado, *Machine Learning for Asset Managers*, Cambridge University Press, 2020.
- Martin Haugh, *A Brief Introduction to Stochastic Calculus*, Access date June 2024, ⟨`https://www.columbia.edu/~mh2078/FoundationsFE/IntroStochCalc.pdf`⟩, Columbia University, 2016.

Python

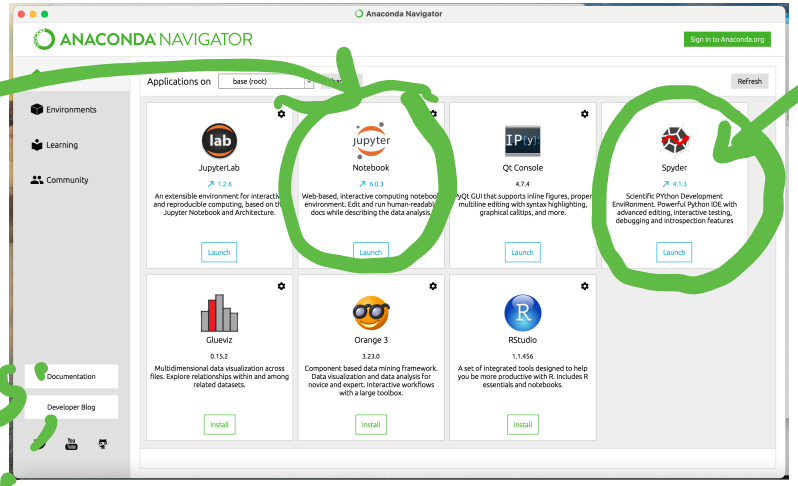
Python Installation

- We'll be using Python.
- If you haven't used Python before, I suggest downloading Anaconda at `https://www.anaconda.com/download`



Anaconda

After you've installed and opened Anaconda, a screen like this will appear. You have several choices for IDEs. Spyder is useful for data analysis. Jupyter Notebook is popular for explanatory work, so students and teachers tend to use it. You can use any IDE.



I use this for work

Good for explaining things, what students tend to use

Packages

Several popular modules (packages) are preinstalled in Anaconda.

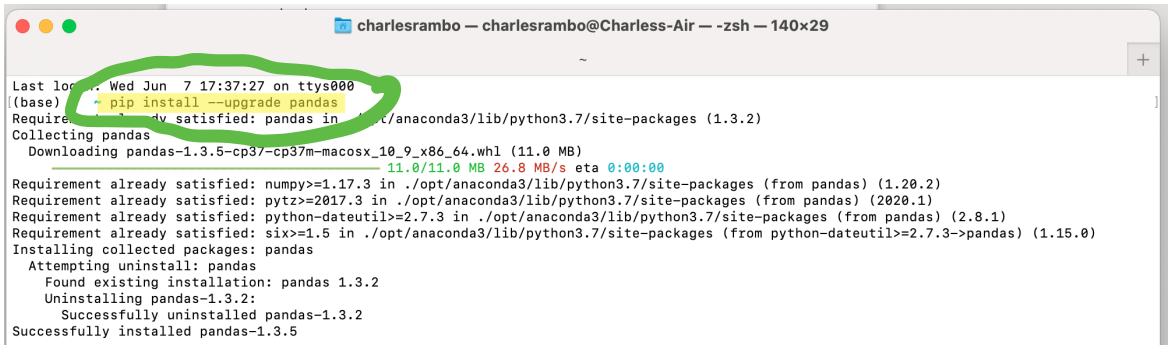
- *NumPy*: Useful math functions.
- *Matplotlib:* Graphing. Somewhat quirky syntax but very popular nonetheless.
- *SciPy*: Scientific computing functions.

Package Installation

To install a new module, go into Terminal, and type

`pip install ..`.

In this example, I'm upgrading pandas. Your Terminal will probably look differently.



Python Graphing Example

Example

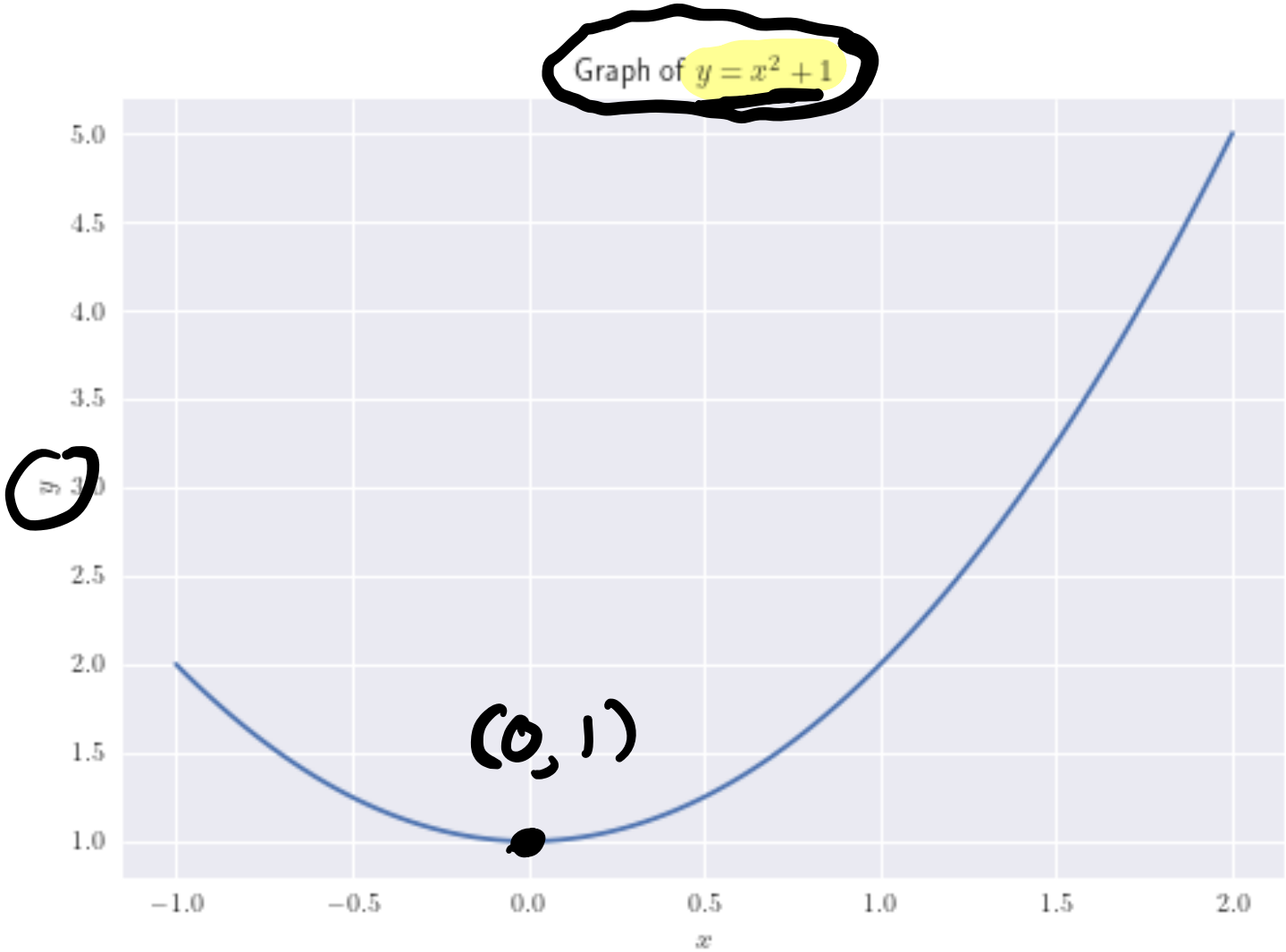
Let $f(x) = x^2 + 1$. Use Python to graph f on the domain $[-1, 2]$.

```python
# Import modules
import numpy as np
import matplotlib.pyplot as plt

# These steps add style; less important

# Use LaTeX
plt.rcParams['text.usetex'] = True

# Use Seaborn style
plt.style.use('seaborn')

# Define f
def f(x):
    return x**2 + 1

# Another option is to use a lambda function
# f = lambda x: x**2 + 1

# Get 100 x-values on [-1, 2]
x_vals = np.linspace(-1, 2, 100)

# Use list comprehension to get y-values
y_vals = [f(x) for x in x_vals]

# Automatically vectorized so this works too
# y_vals = f(x_vals)

# Generate the plot
plt.plot(x_vals, y_vals)

# Label the x-axis
plt.xlabel(r'$x$')

# Label the y-axis
plt.ylabel(r'$y$')

# Give the graph a title
plt.title(r'Graph of $y = x^2 + 1$')

# Save the figure
plt.savefig(path + r'ex0-1')

# Display the plot
plt.show()
```

np array

Python Graphing Result

y = x^2 + 1



Python Optimization Example

Example

Use Python to find the minimum of $f(x) = x^2 + 1$ on the interval $[-1, 2]$.

Solution. From the graph on the previous page, we know that the minimum is $y = 1$ which occurs when $x = 0$. But let's use Python to verify this. Suppose the code above is still in our local environment.

Python Optimization Example

```python
# Import minimize from scipy
from scipy.optimize import minimize

# Define f
def f(x):
    return x**2 + 1

# Minimize function; set bounds equal to the domain
minimize(f, x0 = [1], bounds = [(-1, 2)])
```

The output is shown below.

```
     fun: array([1.])
hess_inv: <1x1 LbfgsInvHessProduct with dtype=float64>
     jac: array([0.])
 message: b'CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL'
    nfev: 8
     nit: 2
  status: 0
 success: True
       x: array([-2.20890595e-10])
```

~ 0