

# Understanding Complex Results

## An Introduction to Visualization and pandas

Nick DeRobertis<sup>1</sup>

<sup>1</sup>University of Florida  
Department of Finance, Insurance, and Real Estate

November 12, 2020

# Table of Contents

- 1 Visualization Introduction
- 2 Tables with Pandas DataFrames
- 3 Graphing using Pandas

# Why Visualize?

- So far we've had one main output from our model, number of years
- Salaries and wealth over time have also been outputs, but we haven't had a good way of understanding that output. It's a bunch of numbers.
- This is where visualization comes in. We have some complex result, and want to make it easily interpretable.

# What we Have so Far

## Retirement Info

Time	Salaries	Wealths
1	61,200	31,050
2	62,424	48,208
3	63,672	66,537
4	64,946	86,100
5	76,182	109,451
6	77,705	134,350
7	79,259	160,882
8	80,844	189,137
9	82,461	219,209
10	96,727	254,352
11	98,662	291,735
12	100,635	331,480

# Visualization in Excel



# An Overwhelming Number of Options in Python

## Python's Visualization Landscape



# Explaining Python Visualization in This Class

- Ultimately, we will be creating graphs using `matplotlib` but we won't use it directly.
- Instead, we will use `pandas`
- `pandas` is actually creating its graphs using `matplotlib` for us, but it is simpler to use.

# Visualization in Excel

## Adding Graphs to the Dynamic Salary Retirement Excel Model

- I will now go back to the "Dynamic Salary Retirement Model.xlsx" Excel model to add visualization
- I have also uploaded the completed workbook from this exercise as "Dynamic Salary Retirement Model Visualized.xlsx"
- Follow along as I go through the example.



# Table of Contents

- 1 Visualization Introduction
- 2 Tables with Pandas DataFrames
- 3 Graphing using Pandas

# Some Setup Before we can Visualize in Python

- pandas does **a lot** more than just graphing. We will use it throughout the rest of the class.
- Previously we've worked with lists, numbers, strings, and even our custom types (our model dataclasses)
- pandas provides the `DataFrame` as a new type that we can use.
- Before we can get to graphing, we must learn how to use the `DataFrame`.

# What is a DataFrame?

A DataFrame is essentially a table. It has rows and columns, just like in Excel.

## Some Features of the DataFrame

- Add or remove columns or rows
- Group by and aggregate
- Load in and output data from/to Excel and many other formats
- Merge and join data sets
- Reshape and pivot data
- Time-series functionality
- Slice and query your data
- Handle duplicates and missing data

# A Basic DataFrame Example

```
>>> import pandas as pd
>>> df = pd.DataFrame()
>>> df['Sales'] = [1052, 212, 346]
>>> df['Category'] = ['Aprons', 'Apples', 'Bowties']
df
```

	Sales	Category
0	1052	Aprons
1	212	Apples
2	346	Bowties

# Introduction to Pandas

## Creating and Using Pandas DataFrames

- I will now go through the notebook in "Intro to Pandas and Table Visualization.ipynb"
- Follow along as I go through the example.
- We will complete everything up until DataFrame Styling

# Intro Pandas Lab

## Getting Started with Pandas

- ① Work off of the Jupyter notebook Pandas and Visualization Labs.ipynb
- ② Complete the lab exercises in the first section entitled "Pandas"

---

Resources: Slide [26](#)

# Styling Pandas DataFrames

- It is possible to add styling to our displayed tabular data by styling the DataFrame
- The styling is very flexible and essentially allows you to do anything
- Out of the box, it is easy to change colors, size, and positioning of text, add a caption, do conditional formatting, and draw a bar graph over the cells.

# Introduction to Pandas

## Creating and Using Pandas DataFrames

- I will now go through the next section in "Intro to Pandas and Table Visualization.ipynb"
- Follow along as I go through the example.
- This time we are covering the remainder of the notebook starting from "DataFrame Styling"



# Pandas Styling Lab

## Styling Pandas DataFrames

- 1 Keep working with the same lab Jupyter Notebook
- 2 Complete the lab exercises in the second section entitled "Pandas Styling"

---

Resources: Slide [27](#)

# Table of Contents

- 1 Visualization Introduction
- 2 Tables with Pandas DataFrames
- 3 Graphing using Pandas

# A Minimal Plotting Example

## Line Graphs using pandas

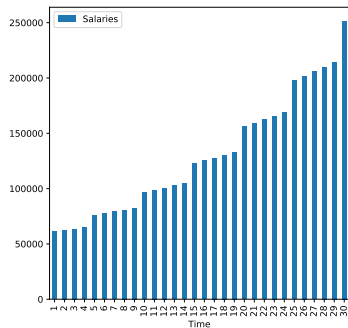
```
>>> %matplotlib inline  
>>> ret_df.plot.line(x='Time', y='Salaries')
```



# Basic Graph Types: Line Graphs



# Basic Graph Types: Bar Graphs



# Basic Graph Types: Box and Whisker Plots



# Introduction to Graphing

## Graphing Using Pandas

- I will now go through "Intro to Graphics.ipynb"
- Follow along as I go through the entire example notebook.

# Intro Visualization Lab

## Introduction to Graphing with Pandas

- 1 Keep working with the same lab Jupyter Notebook
- 2 Complete the lab exercises in the final section entitled "Graphics"

---

Resources: Slide [28](#)



# Lecture Resources

## Lecture Resources

- ① [Slides - Understanding Complex Results](#)
- ② [Lecture Notes - Understanding Complex Results](#)
- ③ [Dynamic Salary Retirement Model Visualized](#)
- ④ [Intro to Pandas and Table Visualization](#)
- ⑤ [10 Minutes to Pandas \(Official Intro\)](#)
- ⑥ [Pandas Official Styling Guide](#)
- ⑦ [Intro to Graphics](#)
- ⑧ [Pandas Official Visualization Guide](#)
- ⑨ [Dynamic Salary Retirement Model Visualized](#)
- ⑩ [Pandas and Visualization Labs](#)

# Intro Pandas Lab Resources

## Getting Started with Pandas Resources

- ① [Pandas and Visualization Labs](#)
- ② [Slides - Understanding Complex Results](#)
- ③ [10 Minutes to Pandas \(Official Intro\)](#)

---

Exercise: Slide [14](#)

# Pandas Styling Lab Resources

## Styling Pandas DataFrames Resources

- ① [Pandas and Visualization Labs](#)
- ② [Slides - Understanding Complex Results](#)
- ③ [Pandas Official Styling Guide](#)

---

Exercise: Slide [17](#)

# Intro Visualization Lab Resources

## Introduction to Graphing with Pandas Resources

- ① [Pandas and Visualization Labs](#)
- ② [Slides - Understanding Complex Results](#)
- ③ [Pandas Official Visualization Guide](#)

---

Exercise: Slide [24](#)