Model Structure
ooooo

Project 1
ooooo

Build the Model
ooooooo

The Depth of a Financial Model, Continued
Extending a Simple Retirement Model in Python

Nick DeRobertis[1]

[1]University of Florida
Department of Finance, Insurance, and Real Estate

June 9, 2021

Model Structure
○●○○○

Project 1
○○○○○

Build the Model
○○○○○○○

Table of Contents

Model Structure
○●○○○

Project 1
○○○○○
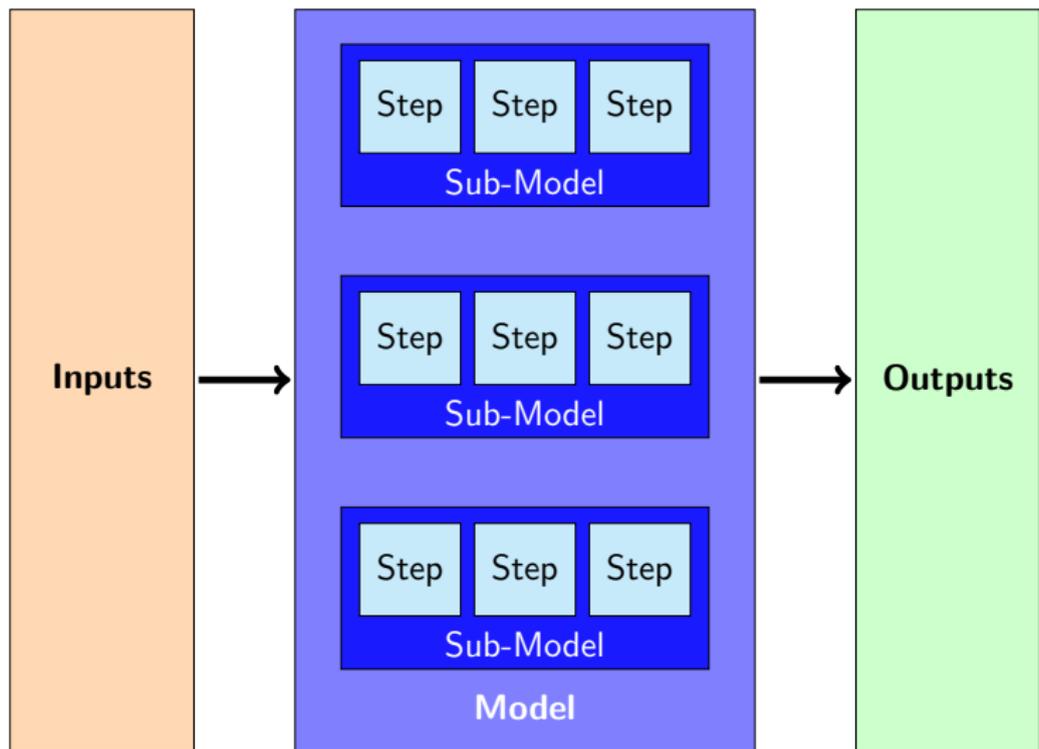
Build the Model
○○○○○○○

How to Structure a Python Financial Model

- We have seen how structure and organization can help the readability and maintainability of an Excel model. The same concept exists for our Python models.

- We already learned that we should use functions to organize logic and a `dataclass` for the model inputs

- Typically you'll have functions for each step, those may be wrapped up into other functions which perform larger steps, and ultimately you'll have one function which does everything by calling theother functions.

- Those are good ideas with any Python model, but working in Jupyter allows us some additional organization and presentation of the model

Using Jupyter for Structure of a Model

- In Jupyter, we can have code, nicely formatted text, equations, sections, hyperlinks, and graphics, all in one document

- For all you can do with these nicely formatted "Markdown" cells, see here and here.

- We can think of sections in Jupyter as analagous to Excel sheets/tabs. One section for each logical part of your model. Then you can have smaller headings for subsections of the model.

- Break your code up into small sections dealing with each step, with nicely formatted text explaining it. Add comments where anything is unclear in the code.

Model Structure
○○○●○

Project 1
○○○○○

Build the Model
○○○○○○○

Structuring a Python Model

Model Structure
○○○○●

Project 1
○○○○○

Build the Model
○○○○○○○

Workflow and Final Output

- When I develop in Jupyter, I have lots of cells going everywhere testing things out

- When I finish a project in Jupyter, I remove these testing cells and make sure it runs and logically flows from end to end (restart kernel and run all cells)

- Run your model with different inputs, and make sure the outputs change in the expected fashion. This is a good way to check your work.

- There may be outputs in each section, but the final output should be at the end of the notebook
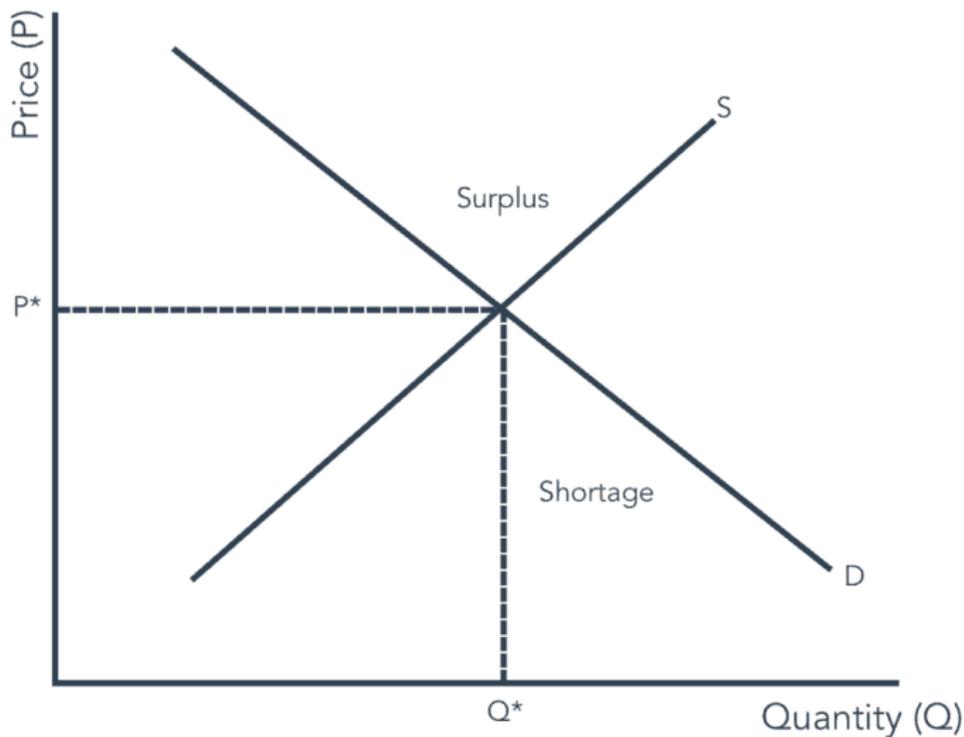
Model Structure
○○○○○

Project 1
●○○○○

Build the Model
○○○○○○○

Table of Contents

Model Structure
00000

Project 1
0●000

Build the Model
0000000

Introducing Project 1

- The first project is aimed at approaching a new time value of money and cash flow model

- It covers the same concepts as the retirement model, but in a capital budgeting setting

- We need to introduce some economic equations to handle this model. You should have covered these in microeconomics.

Model Structure
○○○○○

Project 1
○○●○○

Build the Model
○○○○○○○

A Quick Review of Supply and Demand

Model Structure
ooooo

Project 1
ooo●o

Build the Model
ooooooo

Equations for Project 1

New Required Equations

There are a couple of basic economic equations we haven't talked about that we'll need for this:

$$R = PQ \tag{1}$$

$$Q = min(D, S) \tag{2}$$

- R: Revenue
- Q: Quantity Purchased
- D: Quantity Demanded
- S: Quantity Supplied

Model Structure
○○○○○

Project 1
○○○○●

Build the Model
○○○○○○○

A Couple More Things on the Python Side

- We need to cover one more Python concept and one gotcha before you can complete the first project.
- On the next slide I'll introduce error handling, and show an example of how it's useful
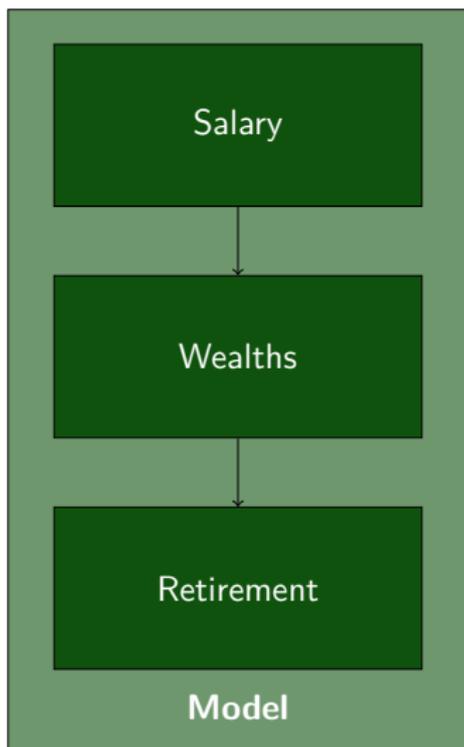
NPV Gotcha

- The NPV function in `numpy` works slightly differently than the NPV function in Excel.
- Excel treats the first cash flow as period 1, while `numpy` treats the first cash flow as period 0.
- If taking NPV where the first cash flow is period 1, pass directly to Excel, and for Python, pass 0 as the first cash flow, then the rest.
- If taking NPV where the first cash flow is period 0, pass from period 1 to end to Excel and add period 0 separately, pass directly to Python.

Model Structure
ooooo

Project 1
ooooo

Build the Model
●oooooo

Table of Contents

Model Structure
○○○○○

Project 1
○○○○○

Build the Model
○●○○○○○

The Structure of the Retirement Model

Model Structure
ooooo

Project 1
ooooo

Build the Model
oooooooo

Revisiting the Model Salary Equation

Salary with Promotions and Cost of Living Raises

$$S_t = S_0(1 + r_l)^t(1 + r_p)^p$$

- S_t: Salary at year t
- S_0: Starting wealth
- r_l: Return for cost of living
- r_p: Return for promotion
- t: Number of years
- p: Number of promotions

Model Structure
○○○○○

Project 1
○○○○○

Build the Model
○○○●○○○

Building the Wealth Model

- For wealths, we need to add the investment return and then the savings in each year

Calculating Wealth

$$W_t = W_{t-1}(1 + r_i) + S_t v$$

- S_t: Salary at year t
- W_t: Wealth at year t
- r_i: Investment return
- t: Number of years
- v: Savings rate

Model Structure
00000

Project 1
00000

Build the Model
0000●00

Creating a Full Model in Python

Dynamic Salary Retirement Model in Python

- I will now show the process I use to create a full model.
- I will be recreating the model "Dynamic Salary Retirement Model.ipynb"
- Go ahead and download that to follow along as you will also extend it in a lab exercise

Model Structure
ooooo

Project 1
ooooo

Build the Model
oooooo●o

Dynamic Desired Cash in Python

Determining Desired Cash in the Dynamic Salary Retirement Python Model

1. We want to relax the assumption that the amount needed in retirement is given by a fixed amount of desired cash

2. Start from the completed retirement model Jupyter notebook Dynamic Salary Retirement Model.ipynb

3. Add new inputs to the model, "Annual Cash Spend During Retirement" and "Years in Retirement"

4. Calculate desired cash based on interest, cash spend, and years in retirement

5. Use the calculated desired cash in the model to determine years to retirement

Answers: Slide 20   Resources: Slide 21

Model Structure
00000

Project 1
00000

Build the Model
0000000●

Extending the Simple Retirement Model in a Different Way

Practice Building A Model

1. Usually I would try to have smaller labs but it didn't fit the format of this lecture. Most will not be able to complete this during class.

2. For this lab, attempt the practice problem "P1 Python Retirement Savings Rate Problem.pdf"

3. This is similar to how the projects will be assigned, so it is good preparation

4. I would encourage you to try it from scratch. If you are totally stuck, try working off of the retirement model I completed today to have a lot of the structure already. If you still are having trouble with that, check the solution and see me in office hours.

5. Note: this is not an official lab exercise you need to submit, it is practice only, but I would highly encourage you to complete it.

Lecture Resources

Lecture Resources

1. Slides - The Depth of a Financial Model, Continued
2. Lecture Notes - The Depth of a Financial Model, Continued
3. Markdown Basic Syntax
4. Markdown Extended Syntax
5. LaTeX Equations in Jupyter
6. Dynamic Salary Retirement Model - Python

Dynamic Desired Cash in Python, Answers

Determining Desired Cash in the Dynamic Salary Retirement Python Model, Answers

1. If annual spend is 40k for 25 years in retirement, $563,757.78 should be the retirement cash and there should be 18 years to retirement.

Exercise: Slide <u>17</u>   Resources: Slide <u>21</u>

Dynamic Desired Cash in Python Resources

Determining Desired Cash in the Dynamic Salary Retirement Python Model Resources

1. Dynamic Salary Retirement Model - Python
2. Slides - The Depth of a Financial Model, Continued

Exercise: Slide   Answers: Slide