Homework 2: Doing Excel work with Python

This second assignment is based on the material in the two weeks we did on using Python for Excel type tasks. As in HW1, make sure you use one of the cookiecutter templates (either cookiecutter-datascience-simple or cookiecutter-datascience-aap) to create a project folder structure for this assignment. You’ll just be zipping the entire folder to submit via Moodle. I highly recommend putting your project under version control (you can make it be a private repo in GitHub).

The first problem based on a homework problem I assign in my MIS 4460/5460 – Business Analytics class (the spreadsheet based class). I’ve eliminated some parts of the original assignment so that you are just focused on using Python for building the models and doing analysis with our Python equivalents of data tables, goal seek and Monte-Carlo simulation.

The object oriented model you’ll be building here will be similar in structure and complexity to the BookstoreModel class that we covered in the course module. You can use any combination of Jupyter notebooks and an IDE that you wish as you are developing your code, but you need to eventually create a Jupyter notebook that shows the various outputs such as data tables, plots and your summary comments in response to my questions.

You MUST use the whatif.py module provided with the assignment. It contains the various functions such as data\_table, goal\_seek, and simulate that you’ll need. You can simply place this module alongside your notebook (or script) so that you can do an import whatif. Or, you could pip install it as we learned in the class module. Here’s my import statements:



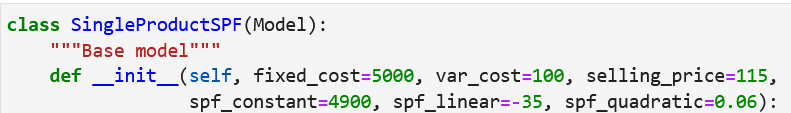
# Analysis 1 – Basic Break Even Analysis

The owner of a relatively small business is trying to determine the breakeven level of sales of the company’s single product. The fixed cost of manufacturing this product each month is $5,000. The variable cost of producing this product is $100 per unit. The current selling price is $115. Using historical data on selling price and monthly demand, the owner fit the following function relating demand and selling price:

where *D* is the monthly demand for the product and is the selling price. This is a 2nd order polynomial function. Let’s call this equation the Selling Price Function (or SPF, for short).

**Assume for now** that this business can meet the demand for its product each month. Your goal is to build a model that will allow you to do things like find the break even selling price (i.e. the value of selling price that gives a profit of zero) and do some sensitivity analysis to some of the key inputs.

**1a – Base Model**) Create a Python based model to that relates profit to the inputs. It must be an object oriented model. Since I’m going to ask you to create some data tables and to do some goal seeking, the structure of your model should be roughly similar to the BookstoreModel we did in the course notes. Obviously, the details will be different as it’s a different problem. Here’s a “check value” – for Selling Price=$115 you should get and Profit=$20028 Don’t worry about getting fractional demand values – it’s fine for this model. Here’s a screenshot of my model class declaration:



**1b – 1-way Data Table**) Once your base model is done, create a one-way data table using our data\_table function to show how profit and demand are related to selling price. Profit and demand are the outputs and selling price is the input. Let price range from $80 to $140 in steps of $10. In addition, create a graph based on your data table object so that it is easy to visualize the relationship between selling price and profit. Make sure your graph has all axes labeled, has titles, has axes properly formatted, and looks good. Discuss the shape of this relationship. Is it linear? Is it nonlinear? Why do you think the graph has the shape it does?

**1c – Break Even**) Use our goal\_seek function to find the break even selling price between $80 and $140.

**1d – 2-way Data Table**) You are also interested in the sensitivity of profit AND total cost to different combinations of selling price and the variable production cost. So, create a 2-way data table using data\_table for these two input variables. Let price range from $80 to $140 in steps of $10 and let the variable production cost vary from $85 to $110 in steps of $5. Figure out a way to create a plot based on the 2-way data table so that the user can visualize how price and variable production cost impact profit.

**1e – Digging deeper**) Now let’s explore this simple model a little bit further. Please do the following:

* Redo the Goal Seek but instead of using $80-$140 as your search range, use $80-$250. What happens? Why do you think this happens?
* Redo your 1-way data table and associated selling price vs profit plot but use $80-$250 as your input range. Now, why do you think that your Goal Seek failed?

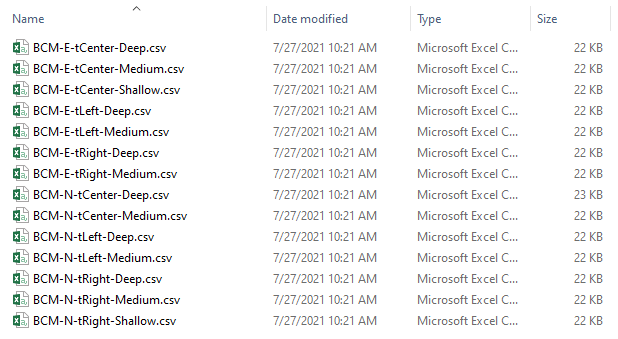
**1f – Simulation**) Let’s add some uncertainty to the variable cost. Specifically, let’s model it with a uniform distribution between (80, 120).

Use our simulate function to simulate this and answer the following questions:

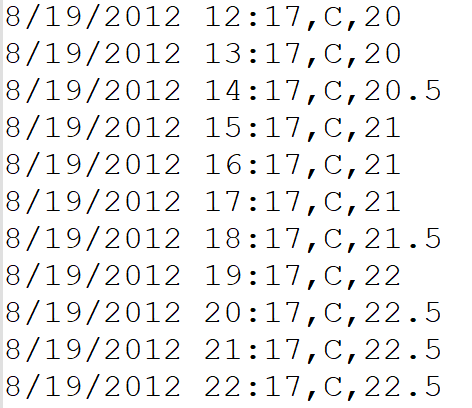
1. Create a histogram of profit
2. What is the probability that profit is negative based on our simulation?

# Analysis 2 – Wrangling csv files

Read through the whole problem before trying to do anything. You are working as a consultant for an environmental analytics firm. They’ve used temperature loggers to monitor the temperature in a number of local streams. The data from each logger has already been downloaded to a separate csv file. In reality, there are hundreds of files from several different streams. In the \logs folder, you’ll see that I’ve provided just a sample of 14 files, all of which are from one particular stream – BCM.



One good thing is that the firm used structured filenames so that it’s clear exactly where the data came from for each logger. Here’s what each of these files looks like:



Obviously, I’m not showing all the records – there are hundreds of rows of data. Notice also that there is NO header in the file. There are three data elements per row:

* Datetime
* Temperature scale (C=Centigrade)
* Temperature

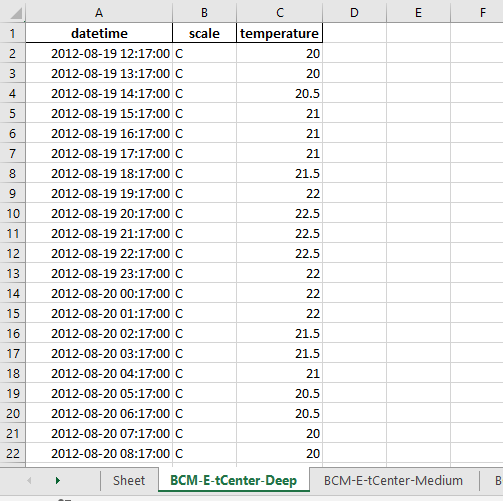
Unfortunately, the company does everything in Excel and have asked you to do the following.

## Step 1 – Consolidation

Combine all of the csv files in a single Excel workbook – one csv file per sheet. The sheet names should be the name of the csv file but without the csv extension. It’s ok if the first sheet is just a blank sheet followed by all of the data sheets for the csv files.

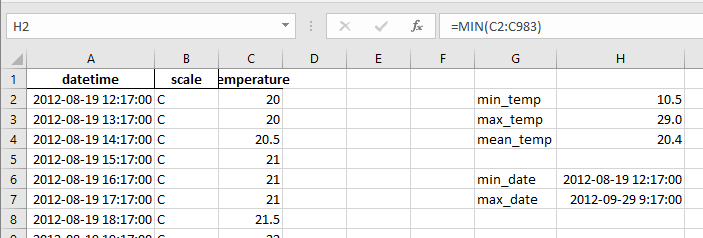
The Excel file should be named using just the characters in the filenames before the first hyphen. For the logs I’ve given you, the file will be called **BCM.xlsx**.

Each data sheet should have the following column headers in A1:C1 – datetime, scale, temperature. Here’s a little snippet of how the workbook looks after this consolidation phase:



**Step 2 – Summarization**

Your client now wants you to add some simple formulas to each sheet showing the minimum, maximum, and average of the temperature values. The labels should be in G2:G4 and the formulas in H2:H4. Notice, they want actual Excel formulas in H2:H4, not just computed values. Here’s a screenshot to make this clearer. Notice the nice cell formatting in column H. In addition, compute the minimum and maximum of the datetime field in rows 6 and 7.



Again, this needs to be done for every sheet. Of course, you don’t know in advance how many rows of data there are in each csv file.

## Your job

Well, this is a Python based course, so you’ve probably already guessed that you need to do this in Python. You cannot use Excel for any part of this other than for checking if what you are doing is working. 😊

There are a number of ways this could be done, but I will share a rough sketch of the general strategy I used and some general hints, suggestions and requirements.

At a high level, this was my approach:

1. Project setup
   1. Use one of our cookiecutters to set up an appropriate folder structure.
   2. Manually copy the \logs folder into an appropriate place in your project folder.
   3. Create a Jupyter Notebook or a .py file in which to develop your code.
   4. Put your project under version control with git.
2. Create blank Excel file named BCM.xlsx
   1. Use the openpyxl library to do this with Python. The openpyxl library is already installed in the aap conda virtual environment.
   2. You can simply hard code the filename BCM.xlsx.
   3. When you save the blank notebook using openpyxl, it will have one sheet in it (which is totally fine).
3. Insert the contents of each csv into a new sheet in BCM.xlsx
   1. For this I used pandas and pathlib. Just used file globbing and a loop.
   2. Read each csv into a pandas dataframe. HINT: Look at the pandas read\_csv documentation to see what useful things you can accomplish during the file reading process.
   3. Inserted each dataframe into the Excel file using the appropriate dataframe method.
4. Add the formulas
   1. Reopened the modified Excel file with openpyxl.
   2. Loop over the appropriate sheets and create the formulas and adjacent labels.
   3. The openpyxl Tutorial at <https://openpyxl.readthedocs.io/en/stable/tutorial.html> has some useful information.
   4. Save the Excel file.
5. Some general hints and suggestions
   1. The pathlib library is very useful for working with paths and filenames.
   2. Remember, cell addresses and even cell formulas are just strings. Python f-strings are quite handy for building strings based on variable values.
   3. There’s a very simple way to determine the row number of the last row of data in a file using openpyxl.
   4. In openpyxl, many collections (such as worksheets) are just Python lists.
   5. You must heavily comment your code to explain what you are doing.

**EXTRA CREDIT**: Create a version of your stream logger code that will work even if there are files from multiple streams and where each stream gets its own Excel file. For example, instead of just filenames starting with “BCM” (the stream identifier), there were other filenames starting with “JEF”, “MAE”, and more. Make sure that each csv gets put into the correct xlsx file.

# Deliverables

Each of the two problems has its own project folder. Just zip up each project folder and put any instructions for me in your readme files. Submit both of the zip files via Moodle