QMB 3311: Python for Business Analytics
Department of Economics
College of Business
University of Central Florida
Spring 2022

Assignment 5

Due Tuesday, March 1, 2021 at 11:59 PM
in your GitHub repository

Instructions:

Complete this assignment within the space on your private GitHub repo in a folder called `assignment_05`. In this folder, save your answers to Questions 1 and 2 in files called `my_CES_A5.py` and `my_logit_A5.py` as described in Question 1. When you are finished, submit it by uploading your files to your GitHub repo using any one of the approaches outlined in Question 3. You are free to discuss your approach to each question with your classmates but you must upload your own work.

Question 1:

Module `my_CES_A5.py`:
Collect the following functions from your previous assignments into a module called `my_CES_A5.py`.

- `CES_utility()` from Assignment 2.
- `CES_utility_valid()` from Assignment 3.
- `CES_utility_in_budget()` from Assignment 3.

Module `my_logit_A5.py`:
Collect the following functions from your previous assignments into a module called `my_logit_A5.py`.

- `logit()` from Assignment 3.
- `logit_like()` from Assignment 3.
- `logit_like_sum()` from Assignment 4.

Testing:
For all of the Exercises in Question 2, and the functions listed above, use examples to test the functions you defined. Since the examples will all be contained within the docstrings of your functions, you can use the `doctest.testmod()` function within the `doctest` module to test your functions automatically at the bottom of the modules. Use an `if` statement to determine whether the tests will be run using `testmod`: if the script is executed as a script, i.e. as the `__main__` script, then run the tests; if it is imported, then skip the tests.

Question 2:

Follow the function design recipe to define functions for all of the following Exercises. For each function, create three examples to test your functions.

Exercise 1 Write a *helper* function `logit_di(x_i, k)` that helps you calculate the term d_i in the gradient vector. The formula for `logit_di()` is

$$d_i = \begin{cases} 1, & \text{if } k = 0, \\ x_i, & \text{if } k = 1, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

where x_i is a number. It will be a single observation of a list of covariates `x` when `logit_d_i()` is used in another function. Add this function to your module `my_logit_A5.py`.

Exercise 2 Write another helper function `logit_dLi_dbk(y_i, x_i, k, beta_0, beta_1)` that helps you calculate an individual term in the sum of the gradient vector. The formula for `logit_dLi_dbk()` is

$$\frac{\partial}{\partial \beta_k} L_i(y_i, x_i; \beta_0, \beta_1) = \begin{cases} d_i(1 - \ell(x_i; \beta_0, \beta_1)), & \text{if } y_i = 1, \\ d_i(-\ell(x_i; \beta_0, \beta_1)), & \text{if } y_i = 0, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where y_i and x_i is a single pair of observations from a list of observations in `y` and `x` that can be used in another function. Add this function to your module `my_logit_A5.py`.

Exercise 3 Write a function `CESdemand_calc(r, p_x, p_x, w)` that returns a list of two values `[x_star, y_star]` that achieve the maximum value of `CES_utility()`, subject to the budget constraint that the consumer's basket of goods should cost no more than their wealth w: $p_x x + p_y y \le w$. That is, given p_x and p_y, these values maximize the function `CESutility_in_budget`, without returning a value of `None`. A senior analyst used calculus to find the optimal values of x^* and y^*:

$$x^* = \frac{p_x^{\frac{1}{r-1}}}{p_x^{\frac{r}{r-1}} + p_y^{\frac{r}{r-1}}} \cdot w \text{ and } y^* = \frac{p_y^{\frac{1}{r-1}}}{p_x^{\frac{r}{r-1}} + p_y^{\frac{r}{r-1}}} \cdot w.$$

Add this function to your module `my_CES_A5.py`.

Exercise 4 Now write a function that finds values of `x` and `y` that maximize `CESutility_in_budget(x, y, r, p_x, p_y, w)` for given r, p_x, p_y, and w. Write a function definition `max_CES_xy(x_min, x_max, y_min, y_max, step, r, p_x, p_y, w)` as follows:

a) Find the values of `x` and `y` by evaluating `CESutility_in_budget(x, y, r, p_x, p_y, w)` over every combination of of (x, y) in two lists.

b) Create lists `x_list` and `y_list` from ranges $x = x^{min}, \ldots, x^{max}$ and $y = y^{min}, \ldots, y^{max}$, where the neighboring values of x or y are separated by distance `step`. The `np.arange()` function is useful for this.

c) Initialize the maximized value with `max_CES = float("-inf")`.

d) Loop over the index numbers `i` and `j`, corresponding to lists `x_list` and `y_list`.

   i) For each pair of `i` and `j`, extract the value `x_list[i]` and `y_list[j]`.

   ii) For each pair of `i` and `j`, evaluate `CESutility_in_budget(x, y, r, p_x, p_x, w)`.

   ii) If the value is higher than `max_CES`, record the new `i_min = i` and `j_min = j` and update the newest value of `max_CES`. If `CESutility_in_budget` returns `None`, make no changes and move on to the next values.

f) After the loops, return `[ x[i_min], y[j_min] ]`.

g) Verify that the result matches the values in Exercise 3 (up to accuracy `step`). You can use the same test cases as you did with Exercise 3 above. Add this function to your module `my_CES_A5.py`.

Question 3:

Push your completed files to your GitHub repository following one of these three methods.

Method 1: In a Browser

Upload your code to your GitHub repo using the interface in a browser.

1. Browse to your `assignment_0X` folder in your repository ("X" corresponds to Assignment X.).
2. Click on the "Add file" button and select "Upload files" from the drop-down menu.
3. Revise the generic message "Added files via upload" to leave a more specific message. You can also add a description of what you are uploading in the field marked "Add an optional extended description..."
4. Press "'Commit changes," leaving the button set to "Commit directly to the `main` branch."

Method 2: With GitHub Desktop

Upload your code to your GitHub repo using the interface in GitHub Desktop.

1. Save your file within the folder in your repository in GitHub Desktop.
2. When you see the changes in GitHub Desktop, add a description of the changes you are making in the bottom left panel.
3. Press the button "Commit to main" to commit those changes.
4. Press the button "Push origin" to push the changes to the online repository. After this step, the changes should be visible on a browser, after refreshing the page.

Method 3: At the Command Line

Push your code directly to the repository from the command line in a terminal window, such as GitBash on a Windows machine or Terminal on a Mac.

1. Open GitBash or Terminal and navigate to the folder inside your local copy of your git repo containing your assignments. Any easy way to do this is to right-click and open GitBash within the folder in Explorer. A better way is to navigate with UNIX commands, such as `cd`.
2. Enter `git add .` to stage all of your files to commit to your repo. You can enter `git add my_filename.ext` to add files one at a time, such as `my_functions.py` in this Assignment.
3. Enter `git commit -m "Describe your changes here"`, with an appropriate description, to commit the changes. This packages all the `add`ed changes into a single unit and stages them to `push` to your online repo.
4. Enter `git push origin main` to push the changes to the online repository. After this step, the changes should be visible on a browser, after refreshing the page.