QMB 6358: Software Tools for Business Analytics
Department of Economics
College of Business
University of Central Florida
Fall 2023

Assignment 5

Due Monday, October 23, 2023 at 11:59 PM
in your GitHub repo.

Instructions:

Complete this assignment within the space on your GitHub repo in a folder called `assignment_05`. In this folder, save a copy of the sample file called `A5_functions.py` that will contain all your Python code for Questions 1 and 2 in this assignment. Use the sample script `my_functions.py` as an example, which is located in in the `demo_11_python_functions` folder within the code repository QMB6358F23.

When you are finished, submit your code by pushing your changes to your GitHub repo, following the instructions in Question 3. You are free to discuss your approach to each question with your classmates but you must `git push` your own work.

Question 1:

Write functions that perform the following operations. Enter your function definitions in your script `A5_functions.py` above the `main()` function.

a) Write a python function `utility()` that will calculate the value of the Cobb-Douglass utility function $u(x, y; \alpha) = x^{\alpha}y^{1-\alpha}$. The first two arguments are x and y, respectively, and the third is α, written out in text as `alpha`.

b) Write a python function `logit_prob(beta, x_i)` that will calculate the inverse of the logit link function,

$$p(\boldsymbol{\beta}; \mathbf{x}_i) = Prob\{y = 1|\mathbf{x}_i\} = \frac{e^{\mathbf{x}_i'\boldsymbol{\beta}}}{1 + e^{\mathbf{x}_i'\boldsymbol{\beta}}} = \frac{e^{\boldsymbol{\beta}_0 + \sum_{k=1}^{K} \mathbf{x}_{ik}\boldsymbol{\beta}_k}}{1 + e^{\boldsymbol{\beta}_0 + \sum_{k=1}^{K} \mathbf{x}_{ik}\boldsymbol{\beta}_k}}.$$

The first argument is a $1 \times (K + 1)$ vector $\boldsymbol{\beta}$ and the second is a $1 \times K$ vector $\mathbf{x}_i$.

c) Write a python function `logit_like_i(beta, y_i, x_i)` that will calculate the contribution to the logit likelihood function for observation $(y_i, \mathbf{x}_i)$,

$$L_i(\boldsymbol{\beta}; y_i, \mathbf{x}_i) = p(\boldsymbol{\beta}; \mathbf{x}_i)^{y_i}(1 - p(\boldsymbol{\beta}; \mathbf{x}_i))^{1-y_i},$$

where y_i is a binary indicator, equal to either one or zero, and the arguments $\boldsymbol{\beta}$ and $\mathbf{x}_i$ are defined as in `logit_prob(beta, x_i)`. You could call your function `logit_prob(beta, x_i)` within this function but it will be more computationally efficient to simplify the above expression and create a separate function.

d) Write a python function `logit_log_like_i(beta, y_i, x_i)` that will calculate the contribution to the log-likelihood function of the logit model for observation $(y_i, \mathbf{x}_i)$,

$$\ell_i(\boldsymbol{\beta}; y_i, \mathbf{x}_i) = \log\{p(\boldsymbol{\beta}; \mathbf{x}_i)^{y_i}(1 - p(\boldsymbol{\beta}; \mathbf{x}_i))^{1-y_i}\},$$

where y_i is a binary indicator, equal to either one or zero, and the arguments x_i and β are defined as in `logit_prob(beta, x_i)`. As with `logit_like_i()`, you could call your function `logit_like_i()` within this function but it will be more computationally efficient to simplify the above expression and create a separate function.

e) Write a python function `logit_likelihood(beta, y, X)` that will calculate the logit log-likelihood function for `y`, an $n \times 1$ array of n observations of a binary indicator y_i, and `X`, an $n \times K$ array of n observations $\mathbf{x}_i$,

$$\ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{n} \ell_i(\boldsymbol{\beta}; y_i, \mathbf{x}_i).$$

f) Write a python function `logit_gradient(beta, y, X)` that will calculate the gradient of the logit likelihood function for `X` an $n \times K$ array of n observations $\mathbf{x}_i$

$$g(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) = \frac{\partial}{\partial \boldsymbol{\beta}} \ell(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) = \frac{\partial}{\partial \boldsymbol{\beta}} \sum_{i=1}^{n} \ell_i(\boldsymbol{\beta}; y_i, \mathbf{x}_i) = \sum_{i=1}^{n} \frac{\partial}{\partial \beta} \ell(\boldsymbol{\beta}; y_i, \mathbf{x}_i),$$

which it can be shown is equal to

$$\sum_{i=1}^{n} (y_i - p(\boldsymbol{\beta}; \mathbf{x}_i))\mathbf{x}_i.$$

Note that this output, in contrast to the other functions, is a vector or list of length K, the same length as $\boldsymbol{\beta}$, which is clear from the last multiplication by $\mathbf{x}_i$ in $(y_i - p(\boldsymbol{\beta}; \mathbf{x}_i))\mathbf{x}_i$.

Question 2:

As you create the functions in Question 1, you should think of some examples to test whether the functions operate correctly. Enter 4 examples per function into the `main()` function of the script `A5_functions.py`. Test your library of functions by running the entire script from beginning to end. The following workflow can guide you through the process of designing and refining your functions.

1. Enter the function definitions in the top portion of the script called `A5_functions.py`.

2. Define the functions one-by-one, by running the blocks of code in `A5_functions.py` that define each function.

3. Test the functions one-by-one, by running the blocks of code in the `main()` function of the script `A5_functions.py`.

4. Check whether the results are correct. If there are any errors or incorrect calculations, repeat the process, making adjustments to the function definitions in the top part of `A5_functions.py` and run the tests in the `main()` function again.

5. As you test these examples, modify the examples in the docstrings to each function. When you run the `else` block of code at the bottom, python should show the results of all your comparisons using the `doctest` module. It will also run these tests whenever a user runs the command `import A5_functions`.

Question 3:

Push your completed files to your GitHub repository following these steps. See the `README.md` and the `GitHub_Quick_Reference.md` in the folder `demo_02_version_control` in the QMB6358F23 course repository for more instructions.

1. Open GitBash 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.

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_filename.ext`. in this example.

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.