ECP 3004: Python for Business Analytics
Department of Economics
College of Business
University of Central Florida
Spring 2021

Assignment 5

Due Sunday, February 28, 2021 at 11:59 PM
in your GitHub repository

Instructions:

Complete this assignment within the space on your *private* GitHub repo (not a fork of the course repo ECP3004S21!) in a folder called `assignment_05`. In this folder, save your answers to Questions 1 and 2 in a file called `my_A5_module.py`, following the sample script in the folder `assignment_05` in the course repository. 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:

Follow the function design recipe to define functions for all of the following Exercises. For each function, create three examples to test your functions. Record the definitions in the sample script `my_A5_module.py`.

Together, the function definitions will form a module called `my_A5_module` that you can read in and test using the script `my_A5_tests.py`. There are no particular requirements for `my_A5_tests.py` but you can use it to conduct any calculations or tests of your module. That way, you will not have any unnecessary commands in your module in `my_A5_module.py`, except for the function definitions.

Exercise 1 Write a function `variance(x)` that calculates the sample variance of the variable in the list x. The formula for the variance is

$$Var(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2,$$

where n is the number of items in the list x and $\bar{x}$ is the average of x.

Exercise 2 Write a function `covariance(y, x)` that calculates the sample covariance of the variables in the lists y and x. The formula for the covariance is

$$Cov(x) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}),$$

where the lists y and x both have length n.

Exercise 3 Now write a function that calculates the slope coefficients for the linear regression model. Using calculus, you can show that the following function minimizes $SSR(y, x, \beta_0, \beta_1)$ for β_1:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

which is called the ordinary least squares (OLS) estimator. Write a function that performs this calculation called `ols_slope(y, x)`.

Exercise 4 Now write a function `ols_intercept(y, x, beta_1_hat)` that calculates the intercept coefficient for the linear regression model. With the slope coefficient, the intercept can be calculated with

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}.$$

Exercise 5 Write a function `ssr(y, x, beta_0, beta_1)` that calculates the sum of squared residuals for the linear regression model.

$$SSR(y, x, \beta_0, \beta_1) = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2.$$

You can use the function `ssr_loops(y, x, beta_0, beta_1)` from Assignment 4 (including the solutions) as a template.

Exercise 6 Now find values of `beta_0` and `beta_1` that minimize `ssr(y, x, beta_0, beta_1)` for given `x` and `y`. Write a function `min_ssr(y, x, beta_0_min, beta_0_max, beta_1_min, beta_1_max, step)` as follows:

a) Find these values by evaluating `ssr(y, x, beta_0, beta_1)` over every combination of of (β_0, β_1) in two lists.

b) Create lists `beta_0_list` and `beta_1_list` from ranges $\beta_0 = \beta_0^{min}, \ldots, \beta_0^{max}$ and $\beta_1 = \beta_1^{min}, \ldots, \beta_1^{max}$, where the neighboring values of β_0 or β_1 are separated by distance `step`.

c) Start with `min_SSR = 999999`. Loop over the index numbers `i` and `j`, corresponding to lists `beta_0_list` and `beta_1_list`.

d) For each pair of `i` and `j`, extract the value `beta_0_list[i]` and `beta_1_list[j]`.

e) For each pair of `i` and `j`, evaluate $SSR(y, x, \beta_0, \beta_1)$. If it is lower than `min_SSR`, record the new `i_min = i` and `j_min = j` and update the newest value of `min_SSR`.

f) After the loops, return `[ beta_0[i_min], beta_1[j_min] ]`

g) Verify that the result matches the values in Exercises 4 and 5 (up to accuracy `step`).

Question 2:

For all of the Exercises in Question 1, use your examples to test the functions you defined. Since the examples are all contained within the docstrings of your functions, you can use the `doctest.testmod()` function within the `doctest` module to test your functions automatically. To conduct any tests, use the sample program `my_A5_tests.py`, run the lines of code, and make any corrections, as necessary.

Don't worry about false alarms: if there are some "failures" that are only different in the smaller decimal places, then your function is good enough. It is much more important that your function runs without throwing an error.

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.