QMB 6358: Software Tools for Business Analytics
Executive Development Center
College of Business
University of Central Florida
Fall 2021

Assignment 7

Due Tuesday, November 16, 2020 at 11:59 PM
in your GitHub repo.

Instructions:

Complete this assignment within the space on your GitHub repo in a folder called `assignment_07`. In this folder, save a copy of the files called `regression_calculation.R` and `logit_calculation.py` that will contain all your R and Python code for Questions 1 and 2 in this assignment. Use the sample scripts in the `assignment_07` folder as a starting point.

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:

In this exercise, you will produce a script that calculates the OLS estimator for a linear regression model, using a number of numerical methods. Use the script `regression_calculation.R` and save it in your folder `assignment_07`.

The script provides estimates of the model $y = \beta_0 + \beta_1 x$, where y is aggregate income and x is the percentage of the labor force employed in agriculture.

a) Observe the value of the intercept and slope coefficient on the variable `agg_pct`, available from the `lm_model` object from the function `coef(lm_model)` and stored as `beta_hat_lm`. This is the objective of the calculations below.

b) Calculate the slope coefficient using the OLS estimator

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where $\bar{x}$ is the mean of x and $\bar{y}$ is the mean of y. Store this value as `beta_hat_calc`.

c) Obtain another estimate of the OLS estimator by solving a system of normal equations

$$(Ax =) \quad X^T X \hat{\beta} = X^T y \quad (= b),$$

which is, in matrix form,

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix}$$

i) Create the matrix A and vector b that represent the system of normal equations. There are a number of ways to do this but the simple way is to calculate the 5 numbers in the matrices above from the variables y and x and assemble them into two matrices with the `matrix()` command.

ii) Use the `solve` function to solve the system of normal equations and obtain another estimate for $\hat{\beta}$, the x argument in the system of equations $Ax = b$. Store this value as `beta_hat_norm`.

d) Estimate $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2)'$ by minimizing the sum of squared residuals, defined as

$$SSR(\beta) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x)^2$$

i) Create a function `ssr(beta, y, x)` that returns the value of the expression $SSR(\beta)$. You can test your function by reproducing the value of `Residual standard error` with the value $\sqrt{SSR(\beta)/8}$.

iii) Use the `optim` function to minimize $SSR(\beta)$ and obtain another estimate for $\hat{\beta}$. Store this value as `beta_hat_opt`. You will need to pass the data y and x as parameters and optimize over the main argument β.

Question 2:

The sample script `logit_calculation.py` uses the `statsmodels` module to estimate a model for the probability that borrowers will default on their loans. You will calculate the parameter estimates by applying numerical methods with root-solving and optimization methods to estimate these parameters. The dataset `credit_data.csv` in `demo_12_linear_models_in_python` includes the following variables.

default:	1 if borrower defaulted on a loan
bmaxrate:	the maximum rate of interest on any part of the loan
amount:	the amount funded on the loan
close:	1 if borrower takes the option of closing the listing until it is fully funded
AA:	1 if borrowers FICO score greater than 760
A:	1 if borrowers FICO score between 720 and 759
B:	1 if borrowers FICO score between 680 and 719
C:	1 if borrowers FICO score between 640 and 679
D:	1 if borrowers FICO score between 600 and 639

In the script `logit_calculation.py`, these data are loaded in and used to estimate a model using `statsmodels`, with only the variables relating to FICO scores to predict `default`. The function `logit_likelihood(beta, y, X)` is the (negative of the) log-likelihood function that is maximized to get the parameter estimates in `statsmodels`. The function `logit_gradient(beta, y, X)` is the (negative of the) first derivative of the log-likelihood function, which is zero at the maximal parameter values. These functions are already defined, since their calculation is the subject of a later course.

a) Run the script `logit_calculation.py` up to line 140 to see the results for the estimation with `statsmodels`. The goal is to match the parameter estimates in `logit_model_fit_sm.params` and achieve the maximum value of the log-likelihood function shown in the output from `logit_model_fit_sm.summary()`.

b) Calculate the parameter estimates by solving for the roots of `logit_gradient(beta, y, X)` using the function `optimize.root()` from the `scipy` module.

c) Calculate the parameter estimates by minimizing `logit_likelihood(beta, y, X)` using the function `minimize()` from the `scipy` module. Implement it several times using the following algorithms.

   i) Use the Nelder-Mead Simplex algorithm algorithm by passing the argument `method = 'nelder-mead'`.

   ii) Use the Davidon-Fletcher-Powell (DFP) algorithm by passing the argument `method = 'powell'`.

   iii) Use the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) algorithm by passing the argument `method = 'BFGS'`.

   iv) Use another version of the BFGS algorithm. This time, pass the additional argument `jac = logit_gradient` to use the first derivative to calculate the iterations within the algorithm.

d) Verify that your parameter estimates and the optimal values of the likelihood function are achieved with the methods in part (b), to match the results from `statsmodels`. You may need to pass additional arguments to the `options` argument, as in:

```
options = {'xtol':  1e-8, 'maxiter':  1000, 'disp':  True}
```

and to adjust the values as necessary. Compare the accuracy and number of iterations.

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_04_version_control` in the QMB6358F21 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.