Exercise set 4

Introduction to Python Programming for Economics & Finance

Richard Foltyn
University of Glasgow

June 1, 2023

Contents

1 Draws from uniform distribution **1**

2 Linear regressions with `lstsq()` **2**
2.1 Create a sample 2
2.2 Estimating coefficients using OLS 2

1 Draws from uniform distribution

In this exercise, we plot histograms against the actual PDF of a uniformly distributed random variable for increasing sample sizes.

Consider the random variable X distributed as

$$X \sim \text{Unif}(a, b)$$

where $a = 1$ and $b = 3$ are the lower and upper bound of the support.

Perform the following tasks:

1. Draw random samples from the uniform distribution for a sequence of increasing sample sizes of 50, 100, 500, 1000, 5000 and 10000.

   Hint: To draw samples from this distribution, use NumPy's `uniform()` method. Note that this method accepts arguments

   ```
   a = 1.0
   b = 3.0
   uniform(low=a, high=b, size=...)
   ```

2. Create a single figure with 6 panels in which you plot a histogram of the samples you have drawn. Use matplotlib's `hist()` function to do this, and pass the argument `bins = 50` so that each panel uses the same number of bins.

3. Add the actual PDF of the uniform distribution to each panel. To evaluate the PDF, use the `pdf()` method of the `uniform` distribution you imported from `scipy.stats`.

   Hint: To plot the PDF of this distribution, use the `uniform` distribution from `scipy.stats`. Note that the methods of SciPy's `uniform` use a different naming convention for their arguments and need to be called as follows:

   ```
   uniform.pdf(..., loc=a, scale=(b-a))
   ```

2 Linear regressions with `lstsq()`

Consider the following quadratic relationship,

$$y_i^* = a(x_i - b)^2 + c$$

with parameters a, b and c. Assume that we only observe it with additive measurement error,

$$y_i = y_i^* + \epsilon_i = a(x_i - b)^2 + c + \epsilon_i, \qquad \epsilon_i \overset{\text{iid}}{\sim} N\left(0, \sigma^2\right)$$

Assume that the model is parametrised by $a = 0.5$, $b = 1$, $c = 1$ and $\sigma = 1.0$.

For a given sample, we can estimate the parameters by OLS by running the regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$$

Note: This does not estimate a, b and c directly, these are instead going to be functions of $(\beta_0, \beta_1, \beta_2)$.

2.1 Create a sample

1. Create a uniformly-spaced grid of x_i values with 30 points on the interval $[-1, 5]$.
2. Create the true response variable y_i^*.
3. Draw 30 values for ϵ_i using the seed `123`.
4. Create the "observed" response variable y_i.

2.2 Estimating coefficients using OLS

Recall that the OLS estimator $\boldsymbol{\beta}$ is implicitly defined by the linear equation system

$$\mathbf{X}'\mathbf{X}\boldsymbol{\beta} = \mathbf{X}'\mathbf{y}$$

While we tend to write the estimator as

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{y}$$

we would usually not manually invert $\mathbf{X}'\mathbf{X}$ to obtain this solution directly. We instead use linear algebra routines to find the solution $\mathbf{x}$ for the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where $\mathbf{A}$ is a coefficient matrix and $\mathbf{b}$ is vector. Clearly, in the case of OLS we have

$$\begin{aligned}\mathbf{A} &= \mathbf{X}'\mathbf{X} \\ \mathbf{b} &= \mathbf{X}'\mathbf{y} \\ \mathbf{x} &= \boldsymbol{\beta}\end{aligned}$$

1. Create the matrix $\mathbf{X}'\mathbf{X}$ and the vector $\mathbf{X}'\mathbf{y}$.
2. Estimate the coefficient vector $\boldsymbol{\beta}$ by running NumPy's `lstsq()`:

   ```
   beta, *rest = lstsq(XX, Xy, rcond=None)
   ```

3. Compute the predicted values $\widehat{y}_i = \mathbf{x}_i'\boldsymbol{\beta}$
4. Scatter-plot the raw data, and add lines for the true and the estimated relationships.