MECH60017/MECH96014/MECH96038 STATISTICS COMPUTING TUTORIAL SHEET IV

Instructions and relevant commands are written in **black** for MATLAB, in **blue** for Python and in **red** for R (if no Python or R commands are provided, that means they are not needed). The resulting figures presented are the ones obtained using MATLAB - these can differ slightly when using Python or R.

The following exercises are to develop your skills in MATLAB/Python/R and some useful packages/toolboxes for statistical data analysis. You should attempt all these exercises in order to ensure that you are familiar with the statistical functions in MATLAB/Python/R for the coursework.

1 Simulating from the Multivariate Normal distribution

In this exercise, we will simulate random vectors from a 3-dimensional multivariate Normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ given by:

$$\boldsymbol{\mu} = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix}$$

and

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.5 & 0 \\ 0.5 & 2 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}$$

Tasks

1. Generate a 3×1 vector of standard normal random variables, lets call this `z`.
2. Define a mean vector `mu` and covariance matrix `Sigma` as above.
3. Calculate `x = mu + chol(Sigma)*z` (Cholesky decomposition in Python/R is achieved using `cholesky` from the `linalg` module of the `NumPy` library / `chol`). This is a random vector from the desired multivariate Normal distribution.
4. Repeat steps (1-3) and generate 1,000 random vectors, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{1000}$.
5. Reproduce Figure 1 shown on the next page using your samples. Can you explain how the features of the plot relate to the parameters of this particular multivariate Normal distribution?

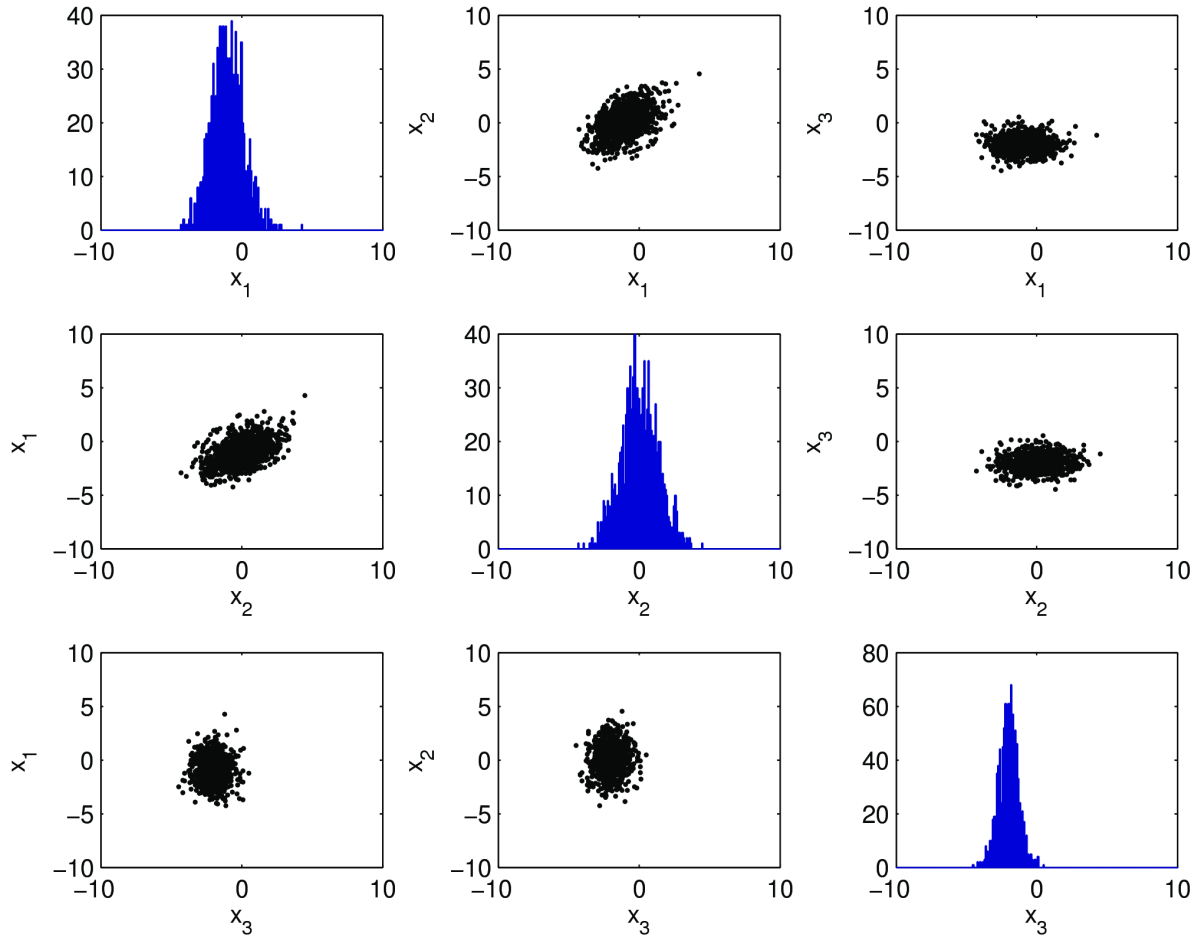

Figure 1: Simulating MVN random vectors

2 Introduction to bootstrapping

In statistics, bootstrapping involves *random sampling with replacement* from an empirical or theoretical distribution. Bootstrapping can be useful for understanding the properties of statistics, such as the mean, the trimmed mean and the median. In this exercise, you will use bootstrapping to look at the distribution of the mean of a normal sample, and the distributions of the mean, the trimmed mean and the median of the Old Faithful data.

Tasks

1. Draw a random sample of size $n = 100$ from a normal distribution with mean 1 and variance 2, using MATLAB/Python/R `normrnd`/`random.normal` from the `NumPy` library/`rnorm`. Save this sample (so it is fixed). Starting with this sample, draw from it a random sample with replacement, of the same size of the original sample. The MATLAB/Python/R command is `randsample` / `random.choice` from the `NumPy` library/`sample`. Calculate the mean of the resample. Plot a histogram of the resample, and compare with the original sample. Repeat this 3 times (3 resamples and calculate mean of each).

2. Repeat the resampling with replacement procedure k times, and each time calculate and store the mean of the resample (you choose a value of k). Plot a kernel density smooth of the resample means. Recall that the distribution of the mean of n i.i.d. normal random variables with mean μ and variance σ^2 is normal with mean μ and variance σ^2/n. Overlay the theoretical density on your histogram - how well do they agree? What is the effect of changing the number of resamples k? What is the effect of changing the size of the original sample, n?

3. Repeat 1 and 2 using the waiting times from the Old Faithful dataset on Blackboard as your starting sample.

4. Now, using the Old Faithful waiting times, calculate (i) the median and (ii) the 5% trimmed mean of each of your k resamples, and plot the resulting distributions of medians and trimmed means. Note we do not have a theoretical result like the CLT for the median or the trimmed mean. (Trimmed means can be useful when we want our sample statistic to be robust to extreme values or outliers. To determine the 5% trimmed mean, trim the top 5% and bottom 5% of observations and calculate the mean of the remaining observations - MATLAB/Python/R `trimmean` / `trim_mean` from the `stats` module of the `SciPy` library / `mean(..., trim=...)`.)

   MATLAB has a command `bootstrp` that does the resampling with replacement and calculates the sample statistic for you - try using this command for question 4.

3 Simulating from bivariate distributions

In lecture notes (see exercise JOINT PDF 1, p58), we looked at the following bivariate distribution:

$$f(x,y) = \frac{6}{5}(x+y^2),\ 0 \le x \le 1, 0 \le y \le 1.$$

We also found that the marginal distribution $f_Y(y)$ (see exercise MARGINAL PDF, p60) is given by:

$$f_Y(y) = \frac{6}{5}\left(\frac{1}{2}+y^2\right), 0 \le y \le 1$$

and the conditional distribution $f_{X|Y}(x|y)$ (see exercise CONDITIONAL DISTRIBUTIONS, p65) is given by:

$$f_{X|Y}(x|y) = \frac{x+y^2}{\frac{1}{2}+y^2}.$$

Tasks

1. Verify that the cumulative distribution functions $F_Y(y) = \Pr(Y \le y)$ and $F_{X|Y}(x|y) = \Pr(X \le x|Y = y)$ are given by:

   $$F_Y(y) = \frac{3}{5}y + \frac{2}{5}y^3$$

   and

   $$F_{X|Y}(x|y) = \frac{x^2+2xy^2}{1+2y^2}$$

2. Use the above results and the inverse transform sampling method to sample random variates from the joint probability density $f(x,y)$:

   (a) Generate a random sample, u_1 from a $U(0,1)$ distribution.

   (b) Now, solve the following equation to find y:

   $$u_1 = \frac{3}{5}y + \frac{2}{5}y^3$$

   Note: This is a cubic equation - use the `roots` / `roots` from the `NumPy` library / polyroot function in MATLAB/Python/R and select the appropriate root.

   (c) Generate a random sample, u_2 from a $U(0,1)$ distribution.

   (d) Now, solve the following equation to find x using the value of y you obtained in Step (b):

   $$u_2 = \frac{x^2+2xy^2}{1+2y^2}$$

   The pair (x,y) will be a random sample from the bivariate distribution $f(x,y)$.

3. Generate a number of random samples (pairs). Plot these pairs, and to overlay the theoretical density use the `contour` / `contour` from the `PyPlot` module of the `Matplotlib` library / `contour` function, with `meshgrid` / `meshgrid` from the `NumPy` library / `meshgrid` from the `pracma` package to specify the points at which the density is evaluated. Can you reproduce Figure 32 in lecture notes (p79)?