MECH60017/MECH96014/MECH96038 STATISTICS COMPUTING TUTORIAL SHEET III

Instructions and relevant commands are written in **blue** for Python and in **red** for R. The resulting figures may differ slightly when using Python or R.

The following exercises are to develop your skills in Python/R and the Statistics Toolbox for statistical data analysis. You should attempt all these exercises in order to ensure that you are familiar with the statistical functions in Python/R for the coursework that will be issued in Spring Term.

1 Building your own random number generator

It is possible to build your own versions of `random.binomial`, `random.normal`, `random.exponential`, / `rnorm`, `rbinom`, `rexp` etc using just the basic `random.uniform` and `random.randomint` / `runif` and `sample.int` commands in Python/R. As you saw in Exercises II, in Python, we have the command `uniform` from the `random` module of the `NumPy` library, with `uniform(0, 1, n)` generating `n` random numbers between 0 and 1 and `randint` from the `random` module of the `NumPy` library, with `randint(low=..., high=..., size=n)` generating `n` random integers between the value of the `low` argument and the `high` argument minus 1 (due to the Python indexing being slightly different). In R, one can use `runif(n, 0, 1)` to generate `n` random numbers between 0 and 1 and `sample.int(n, size=..., replace=...)` to generate as many random integers as specified by the `size` argument, which will be between 1 and `n`, while the `replace` argument takes either `TRUE` or `FALSE` as inputs, to indicate if sampling is done with replacement.

Suppose we want to draw random numbers according to a random distribution with cumulative distribution function $F(x)$ but only have access to a uniform random number generator. How do we proceed?

1.1 Inverse Transform Sampling

If the inverse $F^{-1}(x)$ of the c.d.f. is available then the **inverse transform sampling** approach can be used:

1. Generate a random number between 0 and 1, u - ie, u is from a Uniform$(0, 1)$ distribution.
2. Calculate $x = F^{-1}(u)$ - x is taken to be a random sample from the distribution with c.d.f. $F(x)$.

This approach is usually fine for discrete probability distributions where it is easy to compute the c.d.f. $F(x)$ and continuous distributions where the integrals required for the c.d.f. $F(x)$ are availiable.

Tasks

Use the inverse transform sampling method to build your own functions to generate random samples from some standard distributions.

1. Exponential distribution, $X \sim \text{Exp}(\lambda)$.

   You choose the value of the parameter, and you decide the number n of random samples to generate - try the following for a few different values for n.

   (a) Generate u, a random number between 0 and 1, and calculate $x = F^{-1}(u)$. (Note you should explicitly derive F^{-1} for the exponential distribution.) Repeat for n random samples from an $\text{Exp}(\lambda)$ distribution.

   (b) Use a Q-Q plot (`qqplot(np.array(x), PD, line='45')`, with `qqplot` being imported by the `graphics.gofplots` module of the `statsmodels` library / `qqPlot(x, distribution=PD, param.list = ..., add.line=TRUE)`, with `qqPlot` being imported by the `EnvStats` package) to check your method by plotting the quantiles of your random numbers against the theoretical exponential quantiles. In Python, you can define the theoretical distribution by importing it from the `stats` module of the `SciPy` library. In R, you can define the theoretical distribution against which you want to compare your obtained quantiles within the `qqPlot` function (see arguments `distribution` and `param.list`).

   (c) How does sample size affect the Q-Q plots?

2. Poisson distribution, $X \sim \text{Poisson}(\lambda)$.

   Python/R actually has inverse functions, e.g. `poisson.ppf` (with `poisson` being imported from the `stats` module of the `SciPy` library) / `qpois`. Repeat the tasks above for the Poisson distribution, using `poisson.ppf` / `qpois` instead of deriving an expression for F^{-1}.

   Is the Q-Q plot appropriate to check both the Poisson and exponential distributional assuptions?

1.2 Box-Muller Method

The inverse transform sampling method cannot be used to generate normally distributed random numbers as the c.d.f. for the normal distribution is not available analytically.

The **Box-Muller** method provides a method for generating normally distributed random variables:

1. Generate two uniformly distributed random numbers between 0 and 1, u_1 and u_2.
2. Calculate $z = \sqrt{-2 \ln u_1} \cos(2\pi u_2)$.

z is then a sample from a normal distribution with mean 0 and variance 1. The transformation $x = \sigma z + \mu$ will give an observation from a $\text{N}(\mu, \sigma^2)$.

Tasks

1. Use the Box-Muller method to build your own function to generate random samples from the normal distribution with mean μ and variance σ^2 (you choose numerical values for μ and σ^2).

2. Use a Q-Q plot (`qqplot(X)` / `qqPlot(X)` to check that your method by plotting the quantiles of your random samples against the theoretical normal quantiles.