STM4PSD – Workshop 6

Built-in probability distributions

Recall from last week that R has several built-in probability distributions, and to each is associated a function of the following form:

- `dxxxx`, the probability density function for the distribution.
- `pxxxx`, the probability distribution function for the distribution.
- `qxxxx`, the quantile function for the distribution.
- `rxxxx`, the random generation function for the distribution.

For the uniform, normal, exponential, and gamma distributions, the respective "`xxxx`" is `unif`, `norm`, `exp`, and `gamma`. For instance, the functions associated to the normal distribution are `dnorm`, `pnorm`, `qnorm` and `rnorm`.

1. Recall that the probability *distribution* function must be used to calculate probabilities for a continuous random variable. Complete the following using R, reading the documentation if necessary.

   (a) Let $X \sim \mathrm{Exp}(2)$. Compute each of the following:

   (i) $P(X \leqslant 1)$ (ii) $P(X < 1)$ (iii) $P(X \geqslant 1)$ (iv) $P(1 \leqslant X \leqslant 3)$

   (b) Let $X \sim \mathrm{Gamma}(4, 1)$. Compute each of the following:

   (i) $P(X \leqslant 2)$ (ii) $P(X < 2)$ (iii) $P(X \geqslant 2)$ (iv) $P(2 \leqslant X \leqslant 4)$

Integration in R

Like the `curve` function, the `integrate` function in R requires the input function to be vectorized (like `curve`, internally, R will evaluate the function on a large vector of x-values; this vector will then be used to approximate the integral).

2. If necessary, re-read the discussion on vectorized functions from Workshop 4 before continuing.

There is an alternative way to vectorize a function: using the `Vectorize` function. The `Vectorize` function takes as input the name of a function, and returns a function that is appropriately vectorized. You can run it like so:

```
f.before <- function(x) {
    if (x >= -1 & x <= 2) { (x-1)^2/3 }
    else { 0 }
}
f <- Vectorize(f.before)
```

Note that R treats functions just like any other variable—you can pass functions as arguments to other functions, and a function can be returned as the result of another function. Functions are objects in R.

You should be aware that, although the `Vectorize` function is convenient, it is not generally recommended. R is optimised for code which is written with vectorization in mind. When processing large data sets, using `Vectorize` can noticably worsen performance compared to writing code that exploits vectorization directly. In STM4PSD, you can opt for either option, but efficiency can definitely be relevant in real life applications.

The R function `integrate` is simple to use. For example, to calculate the integral $\int_{-10}^{10} f(x)\,dx$, simply write:

```
integrate(f, lower=-10, upper=10)
```

It is essential that the function is appropriately vectorized before using `integrate`, as otherwise it can lead to incorrect results. You can also perform integrals using ∞ and $-\infty$ by using the keyword `Inf`, like so:

- `integrate(f, lower=-Inf, upper=0)`
- `integrate(f, lower=5, upper=Inf)`
- `integrate(f, lower=-Inf, upper=Inf)`

These calculations are done numerically, which is why the output shows "with absolute error...". This indicates the error bound in the final approximation.

Remember that probabilities for continuous random variables are calculated using integrals:

$$P(X \leqslant b) = \int_{-\infty}^{b} f_X(x)\,dx.$$

$$P(a \leqslant X \leqslant b) = \int_{a}^{b} f_X(x)\,dx.$$

$$P(X \geqslant a) = \int_{a}^{\infty} f_X(x)\,dx.$$

3. Let $f(x)$ be the function from Question 1 of Workshop 5; that is, let $f(x)$ be given by:

$$f(x) = \begin{cases} 4x - 4 & \text{if } 1 \leqslant x \leqslant 1.5, \\ 8 - 4x & \text{if } 1.5 < x \leqslant 2, \\ 0 & \text{otherwise.} \end{cases}$$

   Define an R function `f` which implements this function. Ensure that the function is vectorized, using `ifelse` or `Vectorize`.

4. Use `integrate` with the function you wrote in Question 3 to verify the probabilities calculated in Question 1 of Workshop 5.

Expected value and variance in R

Recall that, for a continuous random variable X with mean μ, the expected value and variance are defined as follows:

$$E(X) = \int_{-\infty}^{\infty} x f_X(x)\,dx$$

$$\operatorname{Var}(X) = \int_{-\infty}^{\infty} (x-\mu)^2 f_X(x)\,dx$$

If we have already defined the density function in R, then this means we only need to define two more functions to integrate:

- Define a function that calculates $x f_X(x)$.
- Define a function that calculates $(x-\mu)^2 f_X(x)$.

Note that the second part will rely on the fact that the first part calculates μ.

First we consider the calculation of the mean. Assume that you have already defined a function named `f` which computes the density function for your random variable. Then you could calculate the expected value using the following:

```
used.for.ev <- function(x) { x * f(x) }
integrate(used.for.ev, lower=-Inf, upper=Inf)
```

If the function `f` is already vectorized, then you won't need to vectorize anything else, because multiplication is vectorized in R.

5. Use this to determine the expected value of the random variable considered in Question 4.

Next we want to determine the variance, which requires the expected value. So it would make sense to have the value of the mean stored in a variable. Your first guess might be to try this:

```
mean <- integrate(used.for.ev, lower=-Inf, upper=Inf)
```

Typing `mean` in the console will then show the result. This will cause a problem when trying to calculate the variance, though, because of the trailing "with absolute error...". Instead, put `$value` at the end, like this:

```
mean <- integrate(used.for.ev, lower=-Inf, upper=Inf)$value
```

If you now type `mean` in the console, it will show the numeric value only.

In many cases, such as when the distribution has parameters that can vary, it would be reasonable to define the expected value as a function. In this example, no input arguments are needed, but generally, they may be needed (see Question 8).

The approach below makes use of anonymous functions.

```
expected.value <- function() {
    integrate(function(x) { x*f(x) }, lower=-Inf, upper=Inf)$value
}
```

We can then compute the variance as follows:

```
used.for.variance <- function(x) {
    mean <- expected.value()
    (x - mean)^2 * f(x)
}
integrate(used.for.variance, lower=-Inf, upper=Inf)
```

As before, if you append `$value` to the end, it will return the numeric part only.

The following example shows how to use anonymous functions to define a `variance` function:

```
variance <- function() {
    mean <- expected.value()
    integrate(function(x) { (x-mean)^2*f(x) }, lower=-Inf, upper=Inf)$value
}
```

6. Use these methods to compute the variance for the random variable considered in Question 4.

7. Recall that the density function for a normal distribution with mean μ and standard deviation σ is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

   Define an R function `my.normal` which computes this probability density function. You will need to use three arguments to the function: `x`, `mu` and `sigma`. Note that the function e^x is computed in R by using `exp(x)`.

8. Use the `integrate` function with the `my.normal` function from Question 7 to write two functions, `my.ev` and `my.var`, with arguments `mu` and `sigma`, which, respectively, calculate the expected value and variance for the normal distribution.

   Since the expected value is `mu` and the variance is `sigma^2`, the correctness of your answer should be easy to verify.

9. **Extension material**. The density function given in Question 1 of Workshop 5 and used here is an example of a *triangular distribution*. Read about the triangular distribution on Wikipedia at this URL:

   https://en.wikipedia.org/wiki/Triangular_distribution

   Using suitable values of the parameters and formulas described in the Wikipedia article, compare the results from the known formulas with the results you obtained in the previous questions. Answers will not be provided for this question, but your tutor will be happy to discuss this question with you.