R Tutorial-05

T. Scofield

You may click here to access the .qmd file.

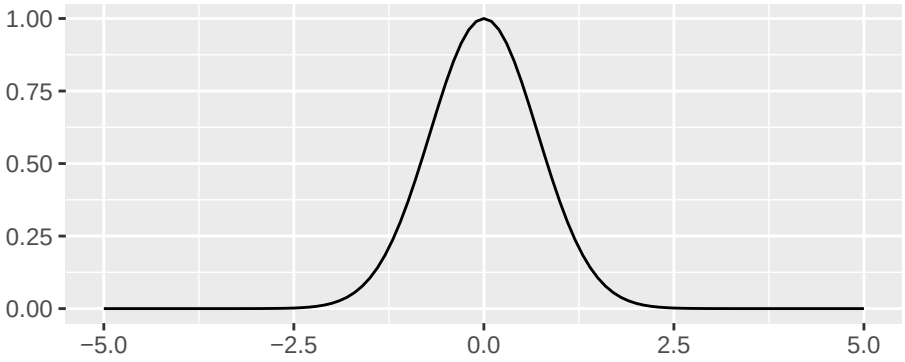
In this issue

- plotting functions using mosaic-loaded tools
- integrating functions in R

Creating/plotting functions

We want to be able to produce plots like those of graphing calculators and Desmos. Let's start with the function $f(x) = e^{-x^2}$.

```
f = makeFun(exp(-x^2) ~ x)
gf_fun(f(x) ~ x, xlim=c(-5,5)) |> gf_labs(y="")
```

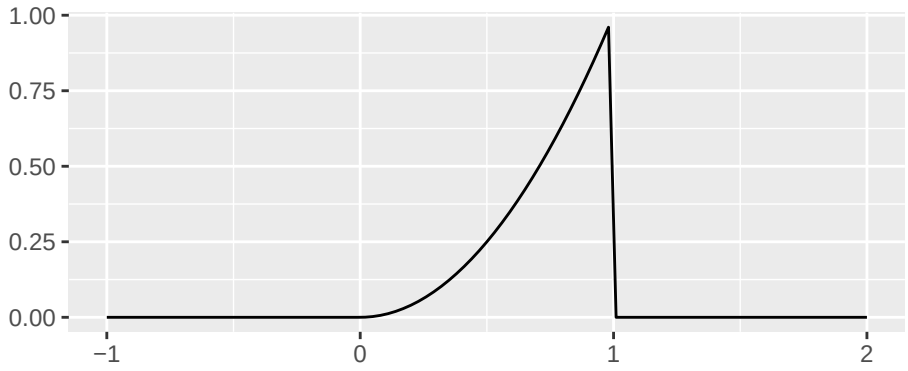


Or, perhaps we have a piecewise-defined function in mind, something like:

$$g(x) = \begin{cases} 0, & x < 0 \\ x^2, & 0 \leq x < 1 \\ 0, & x \geq 1 \end{cases}$$

Defining g before plotting it takes some care. Here, I use the boolean expression, `(x >= 0 & x < 1)` to switch on the active part of the expression:

```
g = makeFun(x^2 * (x>=0 & x<1) ~ x)
gf_fun(g(x)~x, xlim=c(-1,2)) |> gf_labs(y="")
```



Though quarto displays the main instructions in my code chunks, I have several instructions there which are not displayed, yet control the size of the figure. These are

#| fig-width: 5
#| fig-height: 2

which (at least roughly) make the figure 5 inches wide and 2 inches tall in the document.

Integrating functions in R

We can find the area under the curve $f(x) = e^{-x^2}$. In calculus, this is called **integration**. We can compute, numerically

$$\int_a^b f(x)dx,$$

even when $a = -\infty$ or $b = \infty$.

```
f    # unchanged from above
```

```
function (x)
exp(-x^2)
```

```
integrate(f, -1, 4)     # the definite integral from -1 to 4
```

```
1.633051 with absolute error < 7.3e-06
```

```
integrate(f, -Inf, Inf)     # the total area under f
```

```
1.772454 with absolute error < 4.3e-06
```

Notice that f, as a function, is

- never negative, and
- encloses a finite amount of area

We call such a function a **kernel function**. If the area it enclosed were *exactly* equal to 1, we would call f a **probability density function**, or **pdf**. To make f a pdf only requires a rescaling factor.

Take the new function

$$f(x) = \begin{cases} 0, & x < 0 \\ \frac{3}{4}x^2, & 0 \le x < 1 \\ \frac{9}{8} - \frac{3}{8}x, & 1 \le x < 3 \\ 0, & x \ge 3 \end{cases}$$

All of the enclosed area lies in the interval $0 \le x \le 3$, and that area is 1, making f a pdf.

```
f = makeFun(3*x^2*(x >= 0) * (x<=1) / 4 + (9 - 3*x) * (x>=1) * (x<=3) / 8 ~ x)
integrate(f, 0, 3)
```

```
1 with absolute error < 3.6e-05
```

For a random variable X associated with this pdf, the probability $\mathrm{P}(1 < X < 2)$ is found via integrating:

```
integrate(f, 1, 2)
```

```
0.5625 with absolute error < 6.2e-15
```

Important: Probability questions involving a continuous r.v. X correspond to areas under the pdf. Moreover,

$$\mathrm{P}(a \le X \le b), \qquad \mathrm{P}(a < X \le b), \qquad \mathrm{P}(a \le X < b), \qquad \mathrm{P}(a < X < b)$$

are all computed as the *same* area, $\int_a^b f(x)dx$, making them all equal. This means any question like $\mathrm{P}(X = a) = \int_a^a f(x)dx = 0$ for a continuous r.v.

If you load the **mosaicCalc** package it provides a function `antiD()`. As per the Fundamental Theorem of Calculus, you can obtain an antiderivative of f and evaluate it at the endpoints to obtain the probability $\mathrm{P}(1 \leq X \leq 2)$ produced by the previous integral:

```
require(mosaicCalc)
#F = antiD(f(x) ~ x)   # I believe this would usually work, but I had to use the next instead
F = antiD(as.numeric(f(x)) ~ x, lower.bound=-Inf)
F(2) - F(1)
```

```
[1] 0.5625
```

In fact, this antiderivative F is defined as

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

which represents $\mathrm{P}(X \leq x)$, the **cumulative distribution function** (**cdf**) of the continuous r.v. X. So, in the case of a continuous r.v., generally speaking,

- the pdf is the derivative of the cdf.
- the cdf is antiderivative of the pdf designed to give the "area under the pdf up to x", and
- every cdf increases from 0 to 1, just as it does for a discrete r.v.

Below are plots of f and F as they were most-recently defined:

```
p1 = gf_fun(f(x) ~ x, xlim=c(-0.2,3.2)) |> gf_labs(title="graph of pdf f", y="")
p2 = gf_fun(F(x) ~ x, xlim=c(-0.2,3.2)) |> gf_labs(title="graph of cdf F", y="")
grid.arrange(p1, p2, ncol=2)    # requires gridExtra package
```