Simulation and probability

October 30, 2017

Simulation and probability

We saw in the previous script the somewhat mysterious function `qt`. It actually belongs to a very rich family of functions that are associated with probability distributions with a common name structure. They are important especially when we try to do simulation, which may be important in many areas.

The idea is the following. For a given probability distribution there are a bunch of things that we may want to do. For instance, we would want to make a extraction from a normal distribution with a given μ or σ. Or given a normal distribution $N(\mu, \sigma)$ we may want to check what's the probability associated with a given value, or inversely, what is the value asociated with a given probability. All these operations are very common and they are well integrated in `R` using a simple convention: the first letter indicates the operation, while the rest of the name indicates the distribution. In particular, `r` indicate random generation, `d` density function, `q` quantile function, and `p` distribution function. Therefore, `rnorm` will make an extration out of a normal distribution, `pnorm` will give the probability associated with a quantile in a normal distribution and so on. R provides many distributions, Student's t, Snedecor's F, Gamma, Beta, ... and they all follow the same convention.

Thus, `qt` in the previous calculates the probability associated with a quantile in a t distribution with given dof. These functions are very useful. Let's see them in action:

```
rnorm(1)
qnorm(0.975, mean=1, sd=1)
pnorm(2.96, mean=1, sd=1)
```

We could of course use many other different distributions:

```
rbeta(1, 2, 1)
```

```
## [1] 0.869
```

```
pgamma(.3, 2, 2)
```

```
## [1] 0.122
```

Now that we know how to deal with probability distributions, we could use them to do simulations. To do non-trivial simulations we will need some of the tools that we will see later on, but we can start thinking about them, also to learn the limitations of the current approach and why we need more tools.

Let's start with the basics. For simulations it is probably a good idea to set up a seed for the RNG.

```
set.seed(20150211)
rnorm(1, 2, 1)
set.seed(20150211)
rnorm(1, 2, 1)
```

A very useful function for simulation is `replicate` that let's us repeat expressions a number of times. We will see later on another more general approach to this idea of *repeating* things through loops. Consider the sampling distribution of a statistic, for instance:

```
n <- 10
sd(replicate(999, mean(rnorm(n, 3, 2))))
```

```
## [1] 0.65
```

which is approximately $\sigma/\sqrt{n}$, as expected.

```
2/sqrt(n)
```

```
## [1] 0.632
```

We can also do bootstrap sampling using the same approach. The only thing that we need is something that a function that produces a sample with replacement from a vector:

```
x <- rnorm(25, 3.2, 1.7)
sd(replicate(999, mean(sample(x, length(x), replace=TRUE))))
```

```
## [1] 0.389
```

and that matches:

```
sqrt(vcov(lm(x ~ 1)))
```

```
##             (Intercept)
## (Intercept)       0.398
```

With these elements we can now think about, for instance, making extractions of the posterior distribution of the estimated coefficients in the section above to simulate confidence intervals. Or simulate the distribution of transformations of variables. But both tasks are probably easier with tools that we see when we talk about programming.