How to use the simulation framework

Hector Roux de Bézieux

26 April , 2019

This tutorial explains how to use the simulation framework.

Loading the data and the scripts.

We load the data that we want (i.e the binomial data that will serve to produce the simulation) and the scripts. The main function we care about is *simulate_outcome*.

```
sample_data <- read.csv(here("data", "sample_ga_data_binomial.csv"))
source(here("R", "sim_variables.R"))
```

Running a first simulation

We simulate $n = 1000$ observations.

```
sim1 <- simulate_outcome(sample_data = sample_data, n = 1000)
names(sim1)
```

```
## [1] "y"            "covariates"   "ranking_info"
```

The simulation returns the simulated outcome y as well as the *covariates* matrix used to do the simulation, and a $ranking\_info$ list that will use next.

Recovering the true ranking

There are two ranking functions implemented: $ranking_simple$ and $ranking_complex$. The first just return the value of the coefficients (taking into account non-linear and interaction terms). The other compute the actual effect based on the dataset at hand. Both return a vector of values. Variable should be ranked according to absolute values of this vector.

```
ranking_simple(sim1$ranking_info)
```

```
##  [1]  5.4733889  7.7299482 -0.9262701  1.7635074 -0.1198613  4.6146178
##  [7]  3.6312362  1.4840860 -0.7986878 -4.0094808 -3.3963666 -1.0922721
## [13]  2.2988378  2.0140501  1.6008102 -0.9634000  0.8007898 -0.4405895
## [19] -1.1562697 -5.2726323
```

```
ranking_complex(sim1$ranking_info)
```

```
##  [1] 602.0727818  44.4297138   7.4101610   0.0000000   1.4623082
##  [6]   9.2292356   7.2624724   0.0000000   0.7986878   4.0094808
## [11]   3.3963666   1.0922721   0.0000000   0.0000000   1.6008102
## [16] 710.9960677   8.0078983   5.7276637   1.1562697   0.0000000
```

Changing the complexity

We introduce a complexity argument that can be between 0 and 20. With $complexity = 0$, we simulate the log-odd of y as a linear combination of the covariate. Increasing the complexity by one adds one non-linear term (for example cubic or square root) and two interaction terms.

```
sim2 <- simulate_outcome(sample_data = sample_data, n = 1000, complexity = 10)
```

Reproducibility

The function also has a seed argument. It has a default value so that we can all have the same results, but you can change it if you want to iterate over multiple results.

More details

To better understand the details, note that + The covariate matrix is scaled and centered before being used for prediction. + The simulation will check if we have at least 5% of each outcome. Otherwise, it will re-run. This might, with very specific seeds, cause the code to run for very long. In that case, stop and change the seed before re-running.