R-package “simctest”
A Short Introduction

Axel Gandy and Patrick Rubin-Delanchy

November 7, 2011

This document describes briefly how to use the R-package which implements the methods from “Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk” and “An algorithm to compute the power of Monte Carlo tests with guaranteed precision”, based on Gandy [2009] and Gandy and Rubin-Delanchy [2011].

1 Installation

The installation is as for most R-packages that do not reside in CRAN. The general procedure is described in the Section 6 on “Add-on packages” in the R Manual on Istallation and Administration:
`http://cran.r-project.org/doc/manuals/R-admin.html`.

The following is merely an adaptation of those procedures to our package.

1.1 Linux/Unix

If you do not have write access to the package repository:

1. Download the package “simctest_1.0-3.tar.gz” and place it into your home directory.
2. Issue the following commands:

   ```
   echo ".libPaths(\"$HOME/Rlibrary\")" >$HOME/.Rprofile
   R CMD INSTALL -L $HOME/Rlibrary simctest_1.0-0.tar.gz
   ```

3. You may now delete the file “simctest_1.0-3.tar.gz”.

2 Usage

Obviously, the package is loaded by typing

```
> library(simctest)
```

This document can be accessed via

```
> vignette("simctest-intro")
```

Documentation of the most useful commands can be obtained as follows:

```
> ? simctest
> ? mcp
```

2.1 Implementing a Monte Carlo test

The following is an artificial example. By default the algorithm will report back after at most 10000 steps, work with a threshold of $\alpha = 0.05$ and use the spending sequence

$$\epsilon_n = 0.001 \frac{n}{1000 + n}.$$

A simple example of a test with true p-value 0.07.

```
> res <- simctest(function() runif(1) < 0.07)
> res

p.value: 0.07303371
Number of samples: 1602
```

One can also obtain a confidence interval (wrt the resampling procedure) of the computed p-value. By default a 95% confidence interval is computed.

```
> confint(res)

             2.5 %     97.5 %
p.value 0.05859433 0.08477432
```

2.1.1 Behaviour at the Threshold

Next, consider an example where the true p-value is precisely equal to the threshold α. Here, we will expect that the algorithm stops only with probability $2\epsilon = 0.002$. If the algorithm has not stopped after 10000 steps the algorithm will return.

```
> res <- simctest(function() runif(1) < 0.05)
> res

No decision reached.
Final estimate will be in [ 0.04164563 , 0.05953323 ]
Current estimate of the p.value: 0.0495
Number of samples: 10000
```

Note that a part of the output it the interval in which the final estimator will lie.

One can always take a few more steps

```
> res <- cont(res, 10000)
> res

No decision reached.
Final estimate will be in [ 0.04360971 , 0.05691097 ]
Current estimate of the p.value: 0.0476
Number of samples: 20000
```

2.1.2 A simple bootstrap test

An example from [Davison and Hinkley, 1997, section 11.4, p. 534]:

```
> data(fir, package = "boot")
> fir.mle <- c(sum(fir$count), nrow(fir))
> fir.gen <- function(data, mle) {
+     d <- data
+     y <- sample(x = mle[2], size = mle[1], replace = TRUE)
+     d$count <- tabulate(y, mle[2])
+     d
+ }
> fir.fun <- function(data) (nrow(data) - 1) * var(data$count)/mean(data$count)
> resampl <- function() {
+     obs < fir.fun(fir.gen(data = fir, mle = fir.mle))
+ }
> obs <- fir.fun(fir)
> simctest(resampl)

p.value: 0.3103448
Number of samples: 29
```

2.2 Computing the power of a test

2.2.1 Setup

The user will have to create a function that represents the power computation problem. Specifically, it is a function that returns *binary stream* representing replicates of $T \geq t$ for one dataset (if the test rejects for large values of the observed test), where t is the observed test statistic for that dataset, and T is its replicate under the null hypothesis. Example: permutation test from Boos and Zhang [2000]:

```
> genstream <- function() {
+     D <- list(group1 = rnorm(8, mean = 0), group2 = rnorm(4,
+         mean = 0.5))
+     t <- mean(D$group2) - mean(D$group1)
+     stream <- function() {
+         group <- (c(D$group1, D$group2))[sample.int(12)]
+         T <- mean(group[9:12]) - mean(group[1:8])
+         T >= t
+     }
+ }
```

2.2.2 Making things go faster

Users are best advised to devise a procedure by which streams can be generated in batches.

```
genstream <- function(){
     ...
     function(N){
        ...
        T >= t ##T a vector of test replicates
     }
}
```

For testing purposes, in the following we will use an example with power 0.05:

```
> genstream <- function() {
+     p <- runif(1)
+     function(N) {
+         runif(N) <= p
+     }
+ }
```

2.2.3 Default settings

In the default settings reports is TRUE, and on-screen reports on the progress are shown. However, for this demonstration we explicitly set reports to FALSE. (The on-screen reports print backspaces in order to reprint over the same line, which can cause issues on some platforms.)

In the default settings, the confidence interval returned will have length 0.02 if it contains at least one value smaller than 0.05 or at least one value larger 0.95, and a confidence interval not greater than 0.1 otherwise.

```
> res <- mcp(genstream, options = list(reports = FALSE))
> res
```

```
Confidence interval (coverage prob: 0.99): [0.0442,0.0642]
Estimate of power: 0.0543
```

2.2.4 Fixed delta

If we want a fixed CI length no matter what, just set delta, e.g.

```
> res <- mcp(genstream, delta = 0.05, options = list(reports = FALSE))
> res

Confidence interval (coverage prob: 0.99): [0.0334,0.0832]
Estimate of power: 0.0555
```

2.2.5 Make your own adaptive delta

If the default adaptive delta is not your taste, we suggest using the provided template:

```
> res <- mcp(genstream, options = list(reports = FALSE, deltamid = mkdeltamid(mindelta = 0.0
+     maxdelta = 1, llim = 0, rlim = 0.9), epsilon = 1e-04))

Est. progress      Conf. Int.
100%               [0.011,0.107]

> res

Confidence interval (coverage prob: 0.99): [0.011,0.1074]
Estimate of power: 0.057
```

Here the confidence interval returned will have length 0.02 if it contains at least one value smaller than 0 (impossible) or at least one value larger than 0.9, and a confidence interval not greater than 1 otherwise. Basically, this means we only care about the power if it is higher than 0.9. The above example should finished quite early, since it does not take many samples before it can be established that the power is not above 0.9 (it is 0.05).

2.2.6 Interrupting

In the main loop, the algorithm can be interrupted manually. If everything goes well, the method should still return the confidence interval computed so far.

References

D.D. Boos and J.~Zhang. Monte Carlo evaluation of resampling-based hypothesis tests. *Journal of the American Statistical Association*, 95(450):486–492, 2000.

A.C. Davison and D.V. Hinkley. *Bootstrap methods and their application.* Cambridge University Press, 1997.

A.~Gandy. Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk. *Journal of the American Statistical Association*, 104(488):1504–1510, 2009.

A.~Gandy and P.~Rubin-Delanchy. An algorithm to compute the power of monte carlo tests with guaranteed precision. arXiv:1110.1248v1, 2011.