R-package “simctest”
A Short Introduction
Axel Gandy and Patrick Rubin-Delanchy
February 20, 2013

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].

The usage of the “mmctest” algorithm is explained in the vignette “simctest-mmctest-intro”.

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 Installation 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:

```r
   echo ".libPaths("'$HOME/R/library'")" >$HOME/.Rprofile
echo "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

```r
   > library(simctest)
```

This document can be accessed via

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

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

```r
   > ? 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: 0.07089737
Number of samples: 2017

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)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5%</td>
<td>97.5%</td>
</tr>
<tr>
<td>p.value</td>
<td>0.05782495</td>
</tr>
</tbody>
</table>

2.1.1 Behaviour at the Threshold

Next, consider an example where the true p-value is precisely equal to the threshold $\alpha$. 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 [ 0.04149886, 0.05952028 ]
Current estimate of the p.value: 0.0474
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 [ 0.04366204, 0.056915 ]
Current estimate of the p.value: 0.04855
Number of samples: 20000

2.1.2 A simple bootstrap test

An example from [Gandy 2009].

> dat <- matrix(nrow=5,ncol=7,byrow=TRUE,
+ c(1,2,2,1,1,0,1, 2,0,0,2,3,0,0, 0,1,1,2,7,3, 1,1,2,0,0,0,1, 0,1,1,1,1,0,0))
> loglikrat <- function(data){
+ cs <- colSums(data)
+ rs <- rowSums(data)
+ mu <- outer(rs,cs)/sum(rs)
+ 2*sum(ifelse(data<=0.5, 0,data*log(data/mu)))
}
```r
+ } > resample <- function(data){
+ cs <- colSums(data)
+ rs <- rowSums(data)
+ n <- sum(rs)
+ mu <- outer(rs,cs)/n/n
+ matrix(rmultinom(1,n,c(mu)),nrow=dim(data)[1],ncol=dim(data)[2])
+ }
> t <- loglikrat(dat);
> # function to generate samples
> gen <- function(){loglikrat(resample(dat))>=t}
> #using simctest
> simctest(gen,maxsteps=10000)

No decision reached.
Final estimate will be in [ 0.04111842 , 0.05949926 ]
Current estimate of the p.value: 0.0425
Number of samples: 10000

> #now trying simctest.cont
> res <- simctest(gen,maxsteps=500)
> res

No decision reached.
Final estimate will be in [ 0.0254717 , 0.09369025 ]
Current estimate of the p.value: 0.054
Number of samples: 500

> cont(res,20000)

p.value: 0.03950896
Number of samples: 7087

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]:

```r
> 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
+ }
+ }
```
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): [0.0419, 0.0619]
Estimate of power: 0.052

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): [0.027, 0.0769]
Estimate of power: 0.0482

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=0.02, maxdelta=1, llim=0, rlim=0.9),
+     epsilon=0.0001))

Est. progress  Conf. Int.
100%          [0.012, 0.131]

> res
Confidence interval (coverage prob: 0.99): [0.0123, 0.1314]
Estimate of power: 0.054

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

