Power Simulations Comparing GVStat with LiMcLeod Portmanteau Tests

Esam Mahdi
University of Western Ontario

A. Ian McLeod
University of Western Ontario

Abstract

The results reported by Mahdi and McLeod (2010, Figure 1 and Tables 1-2) could be reproduced using the following simulation function, `simpower`. This function calls the function `onesim` which implements the parametric bootstrap test for one simulation. One can run these two functions on a PC with single CPU using the argument `RmpiQ = FALSE` or on a cluster computer with multiple CPU's using the argument `RmpiQ = TRUE` where the **Rmpi** package should be installed. Instructions to install and run `Rmpi` package under Windows is given in the link http://www.stats.uwo.ca/faculty/yu/Rmpi/windows.htm.

Keywords: Parametric Bootstrap Significance Test, Portmanteau Test, VARMA Models .

1. Power Simulations

After loading the packages **portes** and **Rmpi** in R session, users can copy the following two R scripts, `simpower` and `onesim`, then run them in R session.

```
R> library("portes")
R> library("Rmpi")
```

```
R> "simpower" <-
+  function(phi, theta, sigma, intercept = NA, n, lags = seq(5, 30, 5),
+   p = 1, NREP = 1000, NumSim = 10000, statistic = c("GVStat", "LiMcLeod"),
+   StableParameters = NA, RmpiQ = FALSE, SquaredQ = FALSE, Trunc.Series = NA,
+   alpha = c(0.01, 0.05, 0.1), Set.Seed=TRUE){
+      statistic <- match.arg(statistic)
+      if (RmpiQ==FALSE){
+           if (Set.Seed)
+               set.seed(321123789)
+        sim.stat <- replicate(NumSim, onesim(phi = phi, theta = theta,
+         sigma = sigma, intercept = intercept, n = n, lags = lags,
+         p = p, NREP = NREP, statistic = statistic, StableParameters =
+         StableParameters, SquaredQ = SquaredQ, Trunc.Series = Trunc.Series))
+       }
+       else {
+          mpi.spawn.Rslaves()
```

```
+          if (Set.Seed)
+             mpi.setup.rngstream(21597341)
+          mpi.bcast.Robj2slave(NumSim)
+          mpi.bcast.Robj2slave(NREP)
+          mpi.bcast.Robj2slave(phi)
+          mpi.bcast.Robj2slave(theta)
+          mpi.bcast.Robj2slave(sigma)
+          mpi.bcast.Robj2slave(intercept)
+          mpi.bcast.Robj2slave(n)
+          mpi.bcast.Robj2slave(lags)
+          mpi.bcast.Robj2slave(Trunc.Series)
+          mpi.bcast.Robj2slave(statistic)
+          mpi.bcast.Robj2slave(SquaredQ)
+          mpi.bcast.Robj2slave(blockToeplitz)
+          mpi.bcast.Robj2slave(GVStat)
+          mpi.bcast.Robj2slave(LiMcLeod)
+          mpi.bcast.Robj2slave(ImpulseVMA)
+          mpi.bcast.Robj2slave(InvertQ)
+          mpi.bcast.Robj2slave(simvarma)
+          mpi.bcast.Robj2slave(simvma)
+          mpi.bcast.Robj2slave(onesim)
+          if (all(!is.na(StableParameters))){
+                mpi.bcast.Robj2slave(StableParameters)
+                mpi.bcast.cmd(library(akima))
+                mpi.bcast.Robj2slave(interpp)
+                mpi.bcast.Robj2slave(interpp.old)
+                mpi.bcast.Robj2slave(MCTable3)
+                mpi.bcast.Robj2slave(MCTable4)
+                mpi.bcast.Robj2slave(MCTable5)
+                mpi.bcast.Robj2slave(MCTable7)
+                mpi.bcast.Robj2slave(FitStable)
+                mpi.bcast.Robj2slave(rstable)
+          }
+     if(statistic=="GVStat")
+      sim.stat <- mpi.parReplicate(NumSim, onesim(phi = phi, theta = theta,
+       sigma = sigma, intercept = intercept, n = n, lags = lags, p = p,
+       NREP = NREP, statistic = "GVStat", StableParameters = StableParameters,
+       SquaredQ = SquaredQ, Trunc.Series = Trunc.Series))
+     else
+      sim.stat <- mpi.parReplicate(NumSim, onesim(phi = phi, theta = theta,
+       sigma = sigma, intercept = intercept, n = n, lags = lags, p = p,
+       NREP = NREP, statistic = "LiMcLeod", StableParameters = StableParameters,
+       SquaredQ = SquaredQ, Trunc.Series = Trunc.Series))
+      mpi.close.Rslaves()
+     }
+    m <- length(lags)
+    out <- matrix(numeric(m*length(alpha)),ncol=m,nrow=length(alpha))
```

```
+    for (i in 1: length(alpha))
+      out[i,] <- rowMeans(sim.stat<=alpha[i])
+   return(out)
+ }

R> "onesim" <-
+   function(phi, theta, sigma, intercept = NA, n, lags = seq(5, 30, 5),
+       p = 1, NREP = 1000, statistic = c("GVStat", "LiMcLeod"),
+       StableParameters = NA, SquaredQ = FALSE, Trunc.Series = NA){
+         statistic <- match.arg(statistic)
+          if (is.na(Trunc.Series))
+            Trunc.Series <- min(100,ceiling(n/3))
+          sigma <- as.matrix(sigma)
+          k <- NCOL(sigma)
+       sim.data <- simvarma(phi = phi, theta = theta, sigma = sigma,
+        intercept = intercept, n = n, StableParameters = StableParameters,
+        Trunc.Series = Trunc.Series)
+  if (all(is.na(intercept)))
+   fitvar1 <- ar.ols(sim.data, aic = FALSE, intercept = FALSE, order.max = p)
+  else
+   fitvar1 <- ar.ols(sim.data, aic = FALSE, intercept = TRUE, order.max = p)
+   res <- ts(as.matrix(fitvar1$resid)[-(1:p),])
+     if(statistic=="GVStat")
+       obs.stat<-GVStat(res, lags,p,SquaredQ)[,2]
+     else
+       obs.stat<-LiMcLeod(res, lags,p,SquaredQ)[,2]
+     count<-rep(0, length(lags))
+     for (i in 1:NREP){
+         sigma <- as.matrix(fitvar1$var.pred)
+          if (is.array(fitvar1$ar)){
+           arrayphi <- array(numeric(k * k * p), dim = c(k^2, p))
+            for (i in 1:p) arrayphi[, i] <- c(fitvar1$ar[i, , ])
+             phi <- array(c(arrayphi), dim = c(k, k, p))
+          }
+          else
+             phi <- fitvar1$ar
+         theta <- NULL
+         if (!is.null(fitvar1$x.intercept))
+             intercept <- fitvar1$x.intercept
+         else
+             intercept <- rep(0,k)
+        bootdata <- simvarma(phi = phi, theta = theta, sigma = sigma,
+          intercept = intercept, n = n, StableParameters = StableParameters,
+          Trunc.Series = Trunc.Series)
+        FitSimModel <- ar.ols(bootdata, aic = FALSE, intercept = FALSE,
+                 order.max = p)
+        rboot <- ts(as.matrix(FitSimModel$resid)[-(1:p), ])
```

```
+     if(statistic=="GVStat")
+      sim.stat<-GVStat(rboot, lags,p,SquaredQ)[,2]
+     else
+      sim.stat<-LiMcLeod(rboot, lags,p,SquaredQ)[,2]
+     count <- count+(sim.stat>=obs.stat)
+   }
+   ans<-(count+1)/(NREP+1)
+   names(ans)<-lags
+   return(ans)
+ }
```

where

phi is a numeric or an array of `AR` or an array of `VAR` parameters with order p.

theta is a numeric or an array of `MA` or an array of `VMA` parameters with order q.

sigma is the variance of white noise series and must be entered as matrix in case of bivariate or multivariate time series.

intercept is the mean vector of the series.

n is the length of the series.

lags is the vector of lag values.

p is the order of the fitted `VAR` model in simulation procedures.

NREP is the number of bootstrap replications.

NumSim is the number of simulations.

statistic is the generalized test, `"GVStat"`, or the modified test, `"LiMcLeod"`.

StableParameters is the four stable parameters, `ALPHA, BETA, GAMMA,` and `DELTA`. If it is not `NA`, then the R package **akima** should be loaded.

```
R> library("akima")
```

SquaredQ when it is `TRUE` then apply test to squared residuals values in simulation procedures.

Trunc.Series is the truncation lag used to truncate the infinite `MA` or `VMA` Process.

1.1. Simulation Example Using Rmpi

The performance of the generalized variance portmanteau test, `GVStat`, and its competitor, `LiMcLeod`, were compared by the parametric bootstrap simulations. We consider Model 1 from Mahdi and McLeod (2010) with length series, $n = 100$, 10^3 simulations, and 10^2 replications to get some timings. The main simulations were run on a computer with double quad core CPU's using the **Rmpi** package (Yu 2002). The time was 25 minutes for the bootstrap generalized variance test as described in Mahdi and McLeod (2010) and 14 minutes for the bootstrap version of that one given by Li and McLeod (1981).

```
R> NREP <- 10^2
R> NumSim <- 10^3
R> k <- 2
R> sigma <- matrix(c(1, 0.71, 0.71, 1), k, k)
R> phi <- array(c(0.5, 0.4, 0.1, 0.5, 0, 0.3, 0, 0), dim = c(k,
+     k, 2))
R> theta <- NULL
R> lags <- seq(5, 30, 5)
R> Trunc.Series <- 50
R> intercept <- NA
R> n <- 100
R> alpha <- 0.05
R> Start1 <- proc.time()[3]
R> simpower(phi = phi, theta = theta, sigma = sigma, intercept = intercept,
+     n = n, lags = lags, p = 1, NREP = NREP, NumSim = NumSim,
+     statistic = "GVStat", Trunc.Series = Trunc.Series, alpha = alpha,
+     RmpiQ = TRUE, StableParameters = NA, SquaredQ = FALSE, Set.Seed = TRUE)

        8 slaves are spawned successfully. 0 failed.
master (rank 0, comm 1) of size 9 is running on: stats-c-emim
slave1 (rank 1, comm 1) of size 9 is running on: stats-c-emim
slave2 (rank 2, comm 1) of size 9 is running on: stats-c-emim
slave3 (rank 3, comm 1) of size 9 is running on: stats-c-emim
slave4 (rank 4, comm 1) of size 9 is running on: stats-c-emim
slave5 (rank 5, comm 1) of size 9 is running on: stats-c-emim
slave6 (rank 6, comm 1) of size 9 is running on: stats-c-emim
slave7 (rank 7, comm 1) of size 9 is running on: stats-c-emim
slave8 (rank 8, comm 1) of size 9 is running on: stats-c-emim
Loading required package: rlecuyer
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
[1,] 0.686 0.557 0.463 0.394 0.351 0.329
R> End1 <- proc.time()[3]
R> Total1 <- End1 - Start1
R> Total1
elapsed
1502.99

R> Start2 <- proc.time()[3]
R> simpower(phi = phi, theta = theta, sigma = sigma, intercept = intercept,
+     n = n, lags = lags, p = 1, NREP = NREP, NumSim = NumSim,
+     statistic = "LiMcLeod", Trunc.Series = Trunc.Series, alpha = alpha,
+     RmpiQ = TRUE, StableParameters = NA, SquaredQ = FALSE, Set.Seed = TRUE)

        8 slaves are spawned successfully. 0 failed.
master (rank 0, comm 1) of size 9 is running on: stats-c-emim
slave1 (rank 1, comm 1) of size 9 is running on: stats-c-emim
slave2 (rank 2, comm 1) of size 9 is running on: stats-c-emim
```

```
slave3 (rank 3, comm 1) of size 9 is running on: stats-c-emim
slave4 (rank 4, comm 1) of size 9 is running on: stats-c-emim
slave5 (rank 5, comm 1) of size 9 is running on: stats-c-emim
slave6 (rank 6, comm 1) of size 9 is running on: stats-c-emim
slave7 (rank 7, comm 1) of size 9 is running on: stats-c-emim
slave8 (rank 8, comm 1) of size 9 is running on: stats-c-emim
      [,1]  [,2]  [,3] [,4]  [,5]  [,6]
[1,] 0.519 0.347 0.287 0.25 0.222 0.197
R> End2 <- proc.time()[3]
R> Total2 <- End2 - Start2
R> Total2
elapsed
 839.54
```

References

Li WK, McLeod AI (1981). "Distribution of the Residual Autocorrelation in Multivariate ARMA Time Series Models." *Journal of the Royal Statistical Society, Series B*, **43**(2), 231–239.

Mahdi E, McLeod AI (2010). "Improved Multivariate Portmanteau Diagnostic Test." Submitted.

Yu H (2002). "Rmpi: Parallel Statistical Computing in R." *R News*, **2**(2), 10–14. URL http://CRAN.R-project.org/doc/Rnews/.

Affiliation:

A. Ian McLeod
Department of Statistical and Actuarial Sciences
University of Western Ontario
E-mail: aim@stats.uwo.ca
URL: http://www.stats.uwo.ca/mcleod
Esam Mahdi
Department of Statistical and Actuarial Sciences
University of Western Ontario
E-mail: emahdi@uwo.ca