Simulation: Change in Mean

Kai Wenger

2019-11-11

The following simulations demonstrate that the implemented tests for a change in mean in a long-memory time series do what they are supposed. For this purpose, we replicate results of papers that published the implemented tests. We use 500 Monte Carlo replications throuhout all simulations.

Wilcoxon LM and self-normalized Wilcoxon test

First, we replicate a comparision of the Wilcoxon LM test by Dehling, Rooch, and Taqqu (2013) and the self-normalized Wilcoxon test by Betken (2016), which is published in Tables III to VI of Betken (2016) pp. 793-794. For the power simulations (Tables V and VI) we just consider the case that the break is in the middle of the sample ($\tau = 0.5$ following their notation) and the break magnitude is $\Delta = 2$. Note that $d = H - 0.5$.

```r
set.seed(410)
# Parameter setting considered
T_grid              <- c(10,50,100,500)
d_grid              <- c(0.1,0.2,0.3,0.4)
N                   <- 500

# Generate array to save the results
resultmat           <- array(NA, dim=c(length(T_grid),length(d_grid),4))
dimnames(resultmat) <- list(paste("T=",T_grid,sep=""),paste("d=",d_grid,sep=""),
                            paste(rep(c("WilcoxonLM","snWilcoxon"),2),c("size","size",
                "power, Delta=1, tau=0.5","power, Delta=1,tau=0.5"),sep=" "))

# Monte Carlo simulation
for(TTT in 1:length(T_grid))
{
  T <- T_grid[TTT]
  for(ddd in 1:length(d_grid))
  {
    d                 <- d_grid[ddd]
    result_vec        <- 0
    for(i in 1:N)
    {
    # Simulate a fractionally integrated (long-memory) time series of
    # length T with memory d that is not subject to a shift.
    tseries     <- fracdiff::fracdiff.sim(n=T,d=d)$series

    # Simulate a fractionally integrated (long-memory) time series of
    # length T with memory d that is subject to a shift in the middle of
    # the sample of magnitude 2.
    changep     <- c(rep(0,T/2),rep(2,T/2))
    tseries2    <- tseries+changep

    # They suppose that d is known and not estimated.
    # Apply both functions on both time series.
    testIsize    <- wilcoxonLM(tseries,d=d)
    testIIsize   <- snwilcoxon(tseries,d=d)
    testIpower   <- wilcoxonLM(tseries2,d=d)
    testIIpower  <- snwilcoxon(tseries2,d=d)

    # Save if the tests reject at the 5% significance level.
    testIsize    <- testIsize["Teststatistic"]   > testIsize["95%"]
    testIIsize   <- testIIsize["Teststatistic"]  > testIIsize["95%"]
    testIpower   <- testIpower["Teststatistic"]  > testIpower["95%"]
    testIIpower  <- testIIpower["Teststatistic"] > testIIpower["95%"]

    result_vec   <- result_vec+c(testIsize,testIIsize,testIpower,testIIpower)
    }
  resultmat[TTT,ddd,] <- result_vec/N
  print(c(TTT,ddd))
  }
}
# Results
resultmat
```

This yields the following results

```
> resultmat
, , WilcoxonLM size

      d=0.1 d=0.2 d=0.3 d=0.4
T=10  0.014 0.012 0.090 0.264
T=50  0.022 0.028 0.054 0.140
T=100 0.044 0.032 0.034 0.112
T=500 0.038 0.028 0.042 0.082

, , snWilcoxon size

      d=0.1 d=0.2 d=0.3 d=0.4
T=10  0.072 0.026 0.032 0.016
T=50  0.038 0.038 0.058 0.056
T=100 0.052 0.056 0.038 0.080
T=500 0.042 0.060 0.050 0.050

, , WilcoxonLM power, Delta=1, tau=0.5

      d=0.1 d=0.2 d=0.3 d=0.4
T=10  0.426 0.452 0.558 0.726
T=50  1.000 0.976 0.852 0.788
T=100 1.000 1.000 0.952 0.824
T=500 1.000 1.000 1.000 0.912

, , snWilcoxon power, Delta=1, tau=0.5

      d=0.1 d=0.2 d=0.3 d=0.4
T=10  0.522 0.426 0.214 0.130
T=50  0.980 0.908 0.648 0.464
T=100 0.994 0.948 0.792 0.508
```

```
T=500 1.000 1.000 0.944 0.628
```

Apart from some small Monte Carlo differences, our results are equivalent to those of Betken (2016) and show the same pattern.

Self-normalized sup-Wald test

Here, we replicate results for the self-normalized sup-Wald test proposed by Shao (2011). More precisely, we replicate part of the results of Tables 2 and 3 from pp. 602-603 (Section 3) of Shao (2011). For the power simulations (Tables III) we just consider the case that the break is in the middle of the sample ($\tau = 0.5$ following their notation) and the break magnitude is $\Delta = 1$. The significance level is $\alpha \in [10\%, 5\%]$.

```
set.seed(410)
# Parameter setting considered
T_grid              <- c(100,400)
d_grid              <- c(0,0.2,0.4)
N                   <- 500

# Generate array to save the results
resultmat           <- array(NA, dim=c(length(T_grid)*2,length(d_grid),2))
dimnames(resultmat) <- list(paste(rep(paste("T=",T_grid,sep=""),each=2),rep(c("10%","5%"),2)),
                paste("d=",d_grid,sep=""),paste(rep(c("snsupwald"),2),
                c("size","power, Delta=1, tau=0.5"),sep=" "))


# Monte Carlo simulation
for(TTT in 1:length(T_grid))
{
  T <- T_grid[TTT]
  for(ddd in 1:length(d_grid))
  {
    d                 <- d_grid[ddd]
    result_vec        <- 0
    for(i in 1:N)
    {
      # Simulate a fractionally integrated (long-memory) time series of
      # length T with memory d that is not subject to a shift.
      tseries       <- fracdiff::fracdiff.sim(n=T,d=d)$series

      # Simulate a fractionally integrated (long-memory) time series of
      # length T with memory d that is subject to a shift in the middle of
      # the sample of magnitude Delta=1.
      changep       <- c(rep(0,T/2),rep(1,T/2))
      tseries2      <- tseries+changep

      # Estimate the long-memory parameter of both series with the local Whittle
      # estimator using the suggested bandwidth T^0.65.
      d_est         <- LongMemoryTS::local.W(tseries, m=floor(1+T^0.65))$d
      d_est2        <- LongMemoryTS::local.W(tseries2, m=floor(1+T^0.65))$d

      # Apply the function on both time series.
      testIsize     <- snsupwald(tseries,d=d_est)
      testIpower    <- snsupwald(tseries2,d=d_est2)
      # Save if the tests reject at the 10% and 5% significance level.
      testIsize10   <- testIsize["Teststatistic"]  > testIsize["90%"]
      testIsize5    <- testIsize["Teststatistic"]  > testIsize["95%"]
      testIpower10  <- testIpower["Teststatistic"] > testIpower["90%"]
      testIpower5   <- testIpower["Teststatistic"] > testIpower["95%"]

      result_vec    <- result_vec+c(testIsize10,testIsize5,testIpower10,testIpower5)
    }
    resultmat[((2*TTT-1):2*TTT),ddd,] <- result_vec/N
    print(c(TTT,ddd))
  }
}
# Results
resultmat
```

This yields the following results

```
> resultmat
, , snsupwald size

            d=0 d=0.2 d=0.4
T=100 10% 0.084 0.118 0.098
T=100 5%  0.050 0.056 0.048
T=400 10% 0.104 0.106 0.090
T=400 5%  0.056 0.048 0.046

, , snsupwald power, Delta=1, tau=0.5

            d=0 d=0.2 d=0.4
T=100 10% 0.712 0.382 0.208
T=100 5%  0.540 0.240 0.104
T=400 10% 0.996 0.702 0.234
T=400 5%  0.984 0.562 0.128
```

We observe that the results coincide with the results obtained in Shao (2011).

Sup-Wald fixed-b test

The first column of Table II (Section 5.1, p. 47) of Iacone, Leybourne, and Taylor (2014) is replicated in the following.

```
set.seed(410)
# Parameter setting considered
T_grid              <- c(128,256,512)
d_grid              <- c(0,-0.2,0.2,-0.4,0.4)
N                   <- 500

# Generate array to save the results
resultmat           <- array(NA, dim=c(length(T_grid),length(d_grid),1))
dimnames(resultmat) <- list(paste("T=",T_grid,sep=""),paste("d=",d_grid,sep=""),
                            paste(rep(c("fixbsupw"),1),c("size"),sep=" "))

# Monte Carlo simulation
for(TTT in 1:length(T_grid))
```

```
{
  T <- T_grid[TTT]
  for(ddd in 1:length(d_grid))
  {
    d                  <- d_grid[ddd]
    result_vec         <- 0
    for(i in 1:N)
    {
    # Simulate a fractionally integrated (long-memory) time series of
    # length T with memory d that is not subject to a shift.
    tseries     <- fracdiff::fracdiff.sim(n=T,d=d)$series

    # Estimate the long-memory parameter of both series with the local Whittle
    # estimator using the suggested bandwidth T^0.65.
    d_est       <- LongMemoryTS::local.W(tseries, m=floor(1+T^0.65))$d

    # Apply the function on the time series.
    testIsize    <- fixbsupw(tseries,d=d_est)

    # Save if the tests reject at the 5% significance level.
    testIsize    <- testIsize["Teststatistic"] > testIsize["95%"]

    result_vec   <- result_vec+c(testIsize)
    }
  resultmat[TTT,ddd,] <- result_vec/N
  print(c(TTT,ddd))
  }
}
# Results
resultmat
```

This yields the following results

```
> resultmat
, , fixbsupw size

        d=0 d=-0.2 d=0.2 d=-0.4 d=0.4
T=128 0.064  0.034 0.054  0.044 0.076
T=256 0.076  0.050 0.056  0.038 0.076
T=512 0.064  0.068 0.046  0.030 0.050
```

We observe that the results obtained are slightly different to those of Iacone, Leybourne, and Taylor (2014). If we had chosen another seed, they would have been closer to the previously simulated values. However, also in our simulation the test shows the correct pattern.

CUSUM LM and simple CUSUM test

Table I (Section 4, p. 93) of Wenger, Leschinski, and Sibbertsen (2018) is replicated in the following.

```
set.seed(410)
# Parameter setting considered
T_grid              <- c(250,500,1000)
d_grid              <- c(0,0.2,0.4)
m_grid              <- c(0.7,0.75,0.8)
```

```r
N                    <- 500

# Generate array to save the results
resultmat            <- array(NA, dim=c(length(T_grid),length(d_grid)*length(m_grid),4))
dimnames(resultmat) <- list(paste("T=",T_grid,sep=""),paste(rep(paste("d=",d_grid,sep=""),
            length(m_grid)),",m=",rep(m_grid,each=3),sep=""),
            paste(rep(c("CUSUM_simple","CUSUM-LM"),2),
            c("size","size","power","power"),sep=" "))


# Monte Carlo simulation
for(TTT in 1:length(T_grid))
{
  T <- T_grid[TTT]
  for(mmm in 1:length(m_grid))
  {
  m                  <- m_grid[mmm]
    for(ddd in 1:length(d_grid))
    {
    d                  <- d_grid[ddd]
    result_vec         <- 0
      for(i in 1:N)
      {
        # Simulate a fractionally integrated (long-memory) time series of
        # length T with memory d that is not subject to a shift.
        tseries     <- fracdiff::fracdiff.sim(n=T,d=d)$series

        # Simulate a fractionally integrated (long-memory) time series of
        # length T with memory d that is subject to a shift in the middle of
        # the sample of magnitude Delta=1.
        changep     <- c(rep(0,T/2),rep(1,T/2))
        tseries2    <- tseries+changep

        # Estimate the long-memory parameter of both series with the local Whittle
        # estimator using the suggested bandwidth T^m.
        d_est       <- LongMemoryTS::local.W(tseries, m=floor(1+T^m))$d
        d_est2      <- LongMemoryTS::local.W(tseries2, m=floor(1+T^m))$d

        # Apply both functions on both time series.
        testIsize    <- CUSUM_simple(tseries,d=d)
        testIIsize   <- CUSUMLM(tseries,d=d,delta=m)
        testIpower   <- CUSUM_simple(tseries2,d=d)
        testIIpower  <- CUSUMLM(tseries2,d=d,delta=m)

        # Save if the tests reject at the 5% significance level.
        testIsize    <- testIsize["p-value"]          < 0.05
        testIIsize   <- testIIsize["Teststatistic"]  > testIIsize["95%"]
        testIpower   <- testIpower["p-value"]         < 0.05
        testIIpower  <- testIIpower["Teststatistic"] > testIIpower["95%"]

        result_vec   <- result_vec+c(testIsize,testIIsize,testIpower,testIIpower)
        }
  resultmat[TTT,(ddd+(mmm-1)*3),] <- result_vec/N
  print(c(TTT,ddd))
  }
  }
}
# Results
resultmat
```

This yields the following results

```
> resultmat
, , CUSUM_simple size

       d=0,m=0.7 d=0.2,m=0.7 d=0.4,m=0.7 d=0,m=0.75 d=0.2,m=0.75 d=0.4,m=0.75 d=0,m=0.8
T=250      0.038       0.036       0.026      0.046        0.044        0.032     0.040
T=500      0.062       0.048       0.038      0.054        0.042        0.056     0.048
T=1000     0.042       0.038       0.050      0.042        0.054        0.048     0.054

    d=0.2,m=0.8 d=0.4,m=0.8
T=250   0.044       0.044
T=500   0.068       0.054
T=1000  0.038       0.038

, , CUSUM-LM size

       d=0,m=0.7 d=0.2,m=0.7 d=0.4,m=0.7 d=0,m=0.75 d=0.2,m=0.75 d=0.4,m=0.75 d=0,m=0.8
T=250      0.038       0.044       0.022      0.048        0.056        0.030     0.040
T=500      0.062       0.050       0.036      0.062        0.032        0.064     0.058
T=1000     0.046       0.040       0.058      0.052        0.050        0.040     0.052

    d=0.2,m=0.8 d=0.4,m=0.8
T=250   0.052       0.044
T=500   0.064       0.042
T=1000  0.058       0.032

, , CUSUM_simple power

       d=0,m=0.7 d=0.2,m=0.7 d=0.4,m=0.7 d=0,m=0.75 d=0.2,m=0.75 d=0.4,m=0.75 d=0,m=0.8
T=250          1       0.890       0.216          1        0.866        0.196         1
T=500          1       0.988       0.276          1        0.990        0.292         1
T=1000         1       1.000       0.338          1        1.000        0.328         1

    d=0.2,m=0.8 d=0.4,m=0.8
T=250   0.852       0.228
T=500   0.982       0.238
T=1000  0.998       0.308

, , CUSUM-LM power

       d=0,m=0.7 d=0.2,m=0.7 d=0.4,m=0.7 d=0,m=0.75 d=0.2,m=0.75 d=0.4,m=0.75 d=0,m=0.8
T=250          1       0.910       0.254          1        0.896         0.24         1
T=500          1       0.984       0.328          1        0.994         0.32         1
T=1000         1       1.000       0.390          1        1.000         0.35         1

    d=0.2,m=0.8 d=0.4,m=0.8
```

```
T=250   0.886       0.282
T=500   0.984       0.278
T=1000  1.000       0.356
```

We observe that the results coincide with Table I of Wenger, Leschinski, and Sibbertsen (2018).

Fixed bandwidth CUSUM tests

Parts of Tables I to IV (pp. 5-8) of Wenger and Leschinski (2019) is replicated in the following. We just simulate the tests for the bandwidth that are recommended by Wenger and Leschinski (2019) (i.e. $b = 0.1$, $M = 10$).

```
set.seed(410)
# Parameter setting considered
T_grid               <- c(250,500)
d_grid               <- c(-0.2,0,0.2,0.4)
N                    <- 500

# Generate array to save the results
resultmat            <- array(NA, dim=c(length(T_grid),length(d_grid),8))
dimnames(resultmat) <- list(paste("T=",T_grid,sep=""),paste("d=",d_grid,sep=""),
                              paste(rep(c("fixed-b type-A","fixed-b type-B","fixed-m type-A",
                "fixed-m type-B"),2),c(rep("size",4),rep("power",4)),sep=" "))

# Monte Carlo simulation
for(TTT in 1:length(T_grid))
{
  T <- T_grid[TTT]
  for(ddd in 1:length(d_grid))
  {
    d                 <- d_grid[ddd]
    result_vec        <- 0
    for(i in 1:N)
    {
      # Simulate a fractionally integrated (long-memory) time series of
      # length T with memory d that is not subject to a shift.
      tseries     <- fracdiff::fracdiff.sim(n=T,d=d)$series

      # Simulate a fractionally integrated (long-memory) time series of
      # length T with memory d that is subject to a shift in the middle of
      # the sample of magnitude 2.
      changep     <- c(rep(0,T/2),rep(sd(tseries),T/2))
      tseries2    <- tseries+changep

      # Estimate the long-memory parameter of both series with the local Whittle
      # estimator using the suggested bandwidth T^0.8.
      d_est       <- LongMemoryTS::local.W(tseries, m=floor(1+T^0.8))$d
      d_est2      <- LongMemoryTS::local.W(tseries2, m=floor(1+T^0.8))$d

      # Apply both functions on both time series.
      testIsize    <- CUSUMfixed(tseries, d=d_est, procedure="CUSUMfixedb_typeA", bandw=0.1)
      testIIsize   <- CUSUMfixed(tseries, d=d_est, procedure="CUSUMfixedb_typeB", bandw=0.1)
      testIIIsize  <- CUSUMfixed(tseries, d=d_est, procedure="CUSUMfixedm_typeA", bandw=10)
      testIVsize   <- CUSUMfixed(tseries, d=d_est, procedure="CUSUMfixedm_typeB", bandw=10)
      testIpower   <- CUSUMfixed(tseries2, d=d_est2, procedure="CUSUMfixedb_typeA", bandw=0.1)
      testIIpower  <- CUSUMfixed(tseries2, d=d_est2, procedure="CUSUMfixedb_typeB", bandw=0.1)
      testIIIpower <- CUSUMfixed(tseries2, d=d_est2, procedure="CUSUMfixedm_typeA", bandw=10)
      testIVpower  <- CUSUMfixed(tseries2, d=d_est2, procedure="CUSUMfixedm_typeB", bandw=10)

      # Save if the tests reject at the 5% significance level.
      testIsize    <- testIsize["Teststatistic"]   > testIsize["95%"]
      testIIsize   <- testIIsize["Teststatistic"]  > testIIsize["95%"]
      testIIIsize  <- testIIIsize["Teststatistic"] > testIIIsize["95%"]
      testIVsize   <- testIVsize["Teststatistic"]  > testIVsize["95%"]
      testIpower   <- testIpower["Teststatistic"]  > testIpower["95%"]
      testIIpower  <- testIIpower["Teststatistic"] > testIIpower["95%"]
      testIIIpower <- testIIIpower["Teststatistic"]> testIIIpower["95%"]
      testIVpower  <- testIVpower["Teststatistic"] > testIVpower["95%"]

      result_vec   <- result_vec+c(testIsize,testIIsize,testIIIsize,testIVsize,testIpower.
            testIIpower,testIIIpower,testIVpower)
    }
    resultmat[TTT,ddd,] <- result_vec/N
    print(c(TTT,ddd))
  }
}
# Results
resultmat
```

This yields the following results

```
> resultmat
, , fixed-b type-A size

      d=-0.2   d=0 d=0.2 d=0.4
T=250  0.010 0.046 0.064 0.094
T=500  0.012 0.054 0.106 0.072

, , fixed-b type-B size

      d=-0.2   d=0 d=0.2 d=0.4
T=250  0.044 0.052 0.042 0.064
T=500  0.034 0.070 0.066 0.052

, , fixed-m type-A size

      d=-0.2   d=0 d=0.2 d=0.4
T=250  0.030 0.042 0.034 0.070
T=500  0.044 0.056 0.074 0.056

, , fixed-m type-B size

      d=-0.2   d=0 d=0.2 d=0.4
T=250  0.032 0.042 0.034 0.066
T=500  0.044 0.056 0.082 0.050

, , fixed-b type-A power
```

```
      d=-0.2 d=0 d=0.2 d=0.4
T=250      1   1 0.870 0.458
T=500      1   1 0.966 0.450

, , fixed-b type-B power

      d=-0.2 d=0 d=0.2 d=0.4
T=250      1   1 0.732 0.282
T=500      1   1 0.890 0.322

, , fixed-m type-A power

      d=-0.2 d=0 d=0.2 d=0.4
T=250      1   1 0.816 0.364
T=500      1   1 0.964 0.378

, , fixed-m type-B power

      d=-0.2 d=0 d=0.2 d=0.4
T=250      1   1 0.786 0.322
T=500      1   1 0.942 0.346
```

Again, we observe that the results coincide with Tables I to IV of Wenger and Leschinski (2019).

References

Betken, Annika. 2016. "Testing for Change-Points in Long-Range Dependent Time Series by Means of a Self-Normalized Wilcoxon Test." *Journal of Time Series Analysis* 37 (6). Wiley Online Library: 785–809. doi:10.1111/jtsa.12187.

Dehling, Herold, Aeneas Rooch, and Murad S Taqqu. 2013. "Non-Parametric Change-Point Tests for Long-Range Dependent Data." *Scandinavian Journal of Statistics* 40 (1). Wiley Online Library: 153–73. doi:10.1111/j.1467-9469.2012.00799.x.

Iacone, Fabrizio, Stephen Leybourne, and Robert AM Taylor. 2014. "A Fixed-B Test for a Break in Level at an Unknown Time Under Fractional Integration." *Journal of Time Series Analysis* 35 (1). Wiley Online Library: 40–54. doi:10.1111/jtsa.12049.

Shao, Xiaofeng. 2011. "A Simple Test of Changes in Mean in the Possible Presence of Long-Range Dependence." *Journal of Time Series Analysis* 32 (6). Wiley Online Library: 598–606. doi:10.1111/j.1467-9892.2010.00717.x.

Wenger, Kai, and Christian Leschinski. 2019. "Fixed-Bandwidth Cusum Tests Under Long Memory." *Econometrics and Statistics*. Elsevier. doi:10.1016/j.ecosta.2019.08.001.

Wenger, Kai, Christian Leschinski, and Philipp Sibbertsen. 2018. "A Simple Test on Structural Change in Long-Memory Time Series." *Economics Letters* 163. Elsevier: 90–94. doi:10.1016/j.econlet.2017.12.007.