BDS test for independence

Stanisław Galus

27 October 2013

1 Name

bdstest — BDS test for independence

2 Synopsis

```
#include "shg/bdstest.h"
using namespace SHG;
class BDS_test {
public:
     struct Result {
          double stat;
          double pval;
     };
     BDS_test(const std::vector<double>& u,
              int maxm,
              const std::vector<double>& eps);
     BDS_test(const std::vector<double>& u);
     inline int maxm() const;
     inline const std::vector<double>& eps() const;
     inline const std::vector<std::vector<Result>>& res() const;
private:
     /* ... */
};

std::ostream& operator<<(std::ostream& stream, const BDS_test& b);
```

3 Description

The class performs the BDS test for independence [2].

Let $(u_t)_{t=1}^n$ be a time series. The BDS test statistic for an embedding dimension $m \geq 1$ and a threshold $\epsilon > 0$ is defined as

$$W(m, \epsilon) = \sqrt{n}\, \frac{C_{m,n}(\epsilon) - [C_{1,n}(\epsilon)]^m}{V_{m,n}(\epsilon)}, \tag{1}$$

where[1]

$$C_{m,n}(\epsilon) = \frac{2}{(n-m+1)(n-m)} \sum_{1\le s<t\le n-m+1} \chi_\epsilon\left(\max_{0\le i<m} |u_{s+i} - u_{t+i}|\right) \quad (2)$$

is the correlation integral[2] and

$$\begin{aligned} \frac{1}{4}V_{m,n}^2(\epsilon) = m(m-2)C^{2m-2}(K-C^2) + K^m - C^{2m} + \\ + 2\sum_{j=1}^{m-1}\left[C^{2j}(K^{m-j} - C^{2m-2j}) - mC^{2m-2}(K-C^2)\right], \end{aligned} \quad (3)$$

where

$$C = C_n(\epsilon) = \frac{1}{n^2}\sum_{s=1}^{n}\sum_{t=1}^{n}\chi_\epsilon(|u_s - u_t|), \quad (4)$$

$$K = K_n(\epsilon) = \frac{1}{n^3}\sum_{r=1}^{n}\sum_{s=1}^{n}\sum_{t=1}^{n}\chi_\epsilon(|u_r - u_s|)\chi_\epsilon(|u_s - u_t|). \quad (5)$$

If $(u_t)_{t=1}^n$ is a series of independent identically distributed random variables, then for $m \ge 2$ the statistic (1) converges in distribution to the standard normal distribution.

The first constructor requires the time series $(u_t)_{t=1}^n$, arranged in a vector with index running from 0 to $n-1$, the maximum embedding dimension *maxm* and the vector of thresholds *eps*. The second constructor requires only the time series and arbitrarily sets $maxm = 8$ and

$$eps = \begin{bmatrix} 0.5s & 0.75s & s & 1.25s & 1.5s & 1.75s & 2s \end{bmatrix},$$

where

$$s^2 = \frac{1}{n}\sum_{t=1}^{n}(u_t - \bar{u})^2, \quad \bar{u} = \frac{1}{n}\sum_{t=1}^{n} u_t. \quad (6)$$

After successful construction, the function `res()` returns an array of structures of type `BDS_test::Result`, whose member `res()[m][i].stat` reports the value of the statistic $W(m,\epsilon)$ defined by (1) for the embedding dimension $2 \le m \le maxm$ and the i-th threshold in *eps*. The member `res()[m][i].pval` reports the probability

$$\begin{cases} \Phi(W(m,\epsilon)) & \text{if } W(m,\epsilon) < 0, \\ 1 - \Phi(W(m,\epsilon)) & \text{if } W(m,\epsilon) \ge 0, \end{cases}$$

where Φ is the cumulative distribution function of the standard normal distribution. The values of `res()[m][i]` are undefined for $m = 0, 1$.

[1] χ_ϵ is the characteristic function of the interval $[0, \epsilon)$.

[2] Cf. [1, p. 120].

The functions `maxm()` and `eps()` return *maxm* and the vector *eps* used during construction, respectively.

The operator `<<` outputs the four-column plain table of results with ϵ, m, $W(m, \epsilon)$ and the p-value on each row.

4 Implementation

In the implementation, (3) is simplified to

$$\frac{1}{4}V_{m,n}^2(\epsilon) = K^m + (m-1)^2 C^{2m} - m^2 K C^{2m-2} + 2\sum_{j=1}^{m-1} C^{2j} K^{m-j} \tag{7}$$

and (5) is simplified to

$$K = \frac{1}{n^3}\sum_{s=1}^{n}\left[\sum_{t=1}^{n}\chi_\epsilon(|u_s - u_t|)\right]^2. \tag{8}$$

5 Errors

The constructors can throw `std::invalid_argument` if $n < 1$ or n is too big or $maxm < 2$ or $maxm \geq n$. They can also throw `std::range_error` if due to rounding errors $V_{m,n}^2(\epsilon)/n$, required to calculate (1), is negative or too small or the variance in (6) is negative or too small.

6 Example

The following program:

```
#include <iostream>
#include <iomanip>
#include "shg/bdstest.h"

/** Borosh-Niederreiter random number generator. See Donald E. Knuth,
    Sztuka programowania. Tom 2. Algorytmy seminumeryczne, WNT,
    Warszawa 2002, p. 113. */
double bn() {
     static unsigned long int x = 1ul;
     x = (1812433253ul * x) & 0xfffffffful;
     return x / 4294967296.0;
}

using namespace std;
using SHG::BDS_test;

int main() {
     vector<double> u(1000);
```

```
    cout << fixed << setprecision(5);

    for (vector<double>::size_type i = 0; i < u.size(); i++)
         u[i] = bn();
    /** eps is the standard deviation of U(0, 1). */
    cout << BDS_test(u, 8, {sqrt(1.0 / 12.0)}) << '\n';

    for (vector<double>::size_type i = 0; i < u.size(); i++)
         u[i] = i % 2 ? 2.0 * bn() : bn();
    /** eps is the standard deviation of the mixture of U(0, 1) and
        U(0, 2) with mixing weights 0.5. */
    cout << BDS_test(u, 8, {sqrt(13.0 / 48.0)});
}
```

produces:

```
0.28868 2 0.27392 0.39207
0.28868 3 0.26732 0.39461
0.28868 4 -0.33474 0.36891
0.28868 5 -0.97089 0.16580
0.28868 6 -1.83736 0.03308
0.28868 7 -2.35252 0.00932
0.28868 8 -2.16494 0.01520

0.52042 2 -3.96242 0.00004
0.52042 3 0.39043 0.34811
0.52042 4 -0.07102 0.47169
0.52042 5 1.30413 0.09609
0.52042 6 1.26937 0.10215
0.52042 7 2.17663 0.01475
0.52042 8 2.04631 0.02036
```

References

[1] Gregory L. Baker and Jerry P. Gollub. *Wstęp do dynamiki układów chaotycznych*. Wydawnictwo Naukowe PWN, Warszawa, 1998.

[2] W. A. Brock, W. D. Dechert, J. A. Scheinkman, and B. LeBaron. A test for independence based on the correlation dimension. *Econometric Reviews*, 15(3):197–235, 1996.