About The Project

This repository provides C++ codes used to conduct Monte Carlo simulations and empirical applications in the paper: `A Distance-Based Test of Independence Between two Multivariate Time Series`

Pre-requisites

- GNU C++ compiler (GNU GCC 7.5)
- GSL - GNU Scientific Library 2.7
- Boost C++ library 1.78.0
- SHOGUN machine learning toolbox 6.1.4
- Plotcpp
- Ubuntu 20.04 LTS

Build

On an Ubuntu machine, a binary executable can be built from a `C++ main file` by running following shell script from the terminal:

```
g++ -Wno-deprecated -O3 -Wfloat-equal -Wfatal-errors -m64 -std=gnu++17 -fopenmp -lshogun -lspdlog -lboost_thread -Wunknown-pragmas -Wall -Waggressive-loop-optimizations -mavx2 -march=native -I/<path-to-the-folder-containing-source-codes>/ -I/usr/include -I/usr/lib/gcc/x86_64-linux-gnu/4.9.3/include -I/<path-to-plotcpp-library> -I/<path-to-gsl-2.7>/include -I/<path-to-shogun-library>/include -I/usr/include/eigen3 -I/usr/local/include -c/<path-to-the-folder-containing-source-codes>/<main file with extension *.cpp> -o .objs/main.o

g++ -L/<path-to-gsl-2.7>/lib -L/<path-to-shogun-library>/lib -L/usr/lib/x86_64-linux-gnu -L/usr/lib -o <name-of-the-binary-to-be-built> .objs/main.o  -fopenmp -O3 -m64 -lshogun -lspdlog  -lgsl -lgslcblas -lm
```

List of C++ main files

C++ main file	Description
`main_VAR_CC_MGARCH_MGARCH_data.cpp`	to simulate the proposed test with the true DGP CC_MGARCH(1) to generate and fit data
`main_VAR_CC_MGARCH_VAR_data.cpp`	to simulate the proposed test with the true DGP VAR(1) to generate and fit data
`main_VAR1_updated.cpp`	to simulate the proposed test using VAR(1) to generate data and Random Forest (RF) to fit data
`main_CC_MGARCH.cpp`	to simulate the proposed test using CC-MGARCH(1, 1) to generate data and RF to fit data
`main_others_VAR1.cpp`	to simulate the other tests using VAR(1) to generate and fit data
`main_others_CC_MGARCH.cpp`	to simulate the other tests using CC-MGARCH(1, 1) to generate and fit data
`main_others_VAR1_misspec.cpp`	to simulate the other tests using VAR(1) to generate data and CC-MGARCH(1, 1) to fit data
`main_others_CC_MGARCH_misspec.cpp`	to simulate the other tests using CC-MGARCH(1, 1) to generate data and VAR(1) to fit data
`main_bootstrp_using_MGARCH_Stock_app.cpp`	to implement the proposed bootstrap test using RF in the empirical application (stocks vs. bonds)
`main_timing_hsic_dcorr_tests.cpp`	to time the proposed test and the HSIC-based test
`main_Beta_bivariate.cpp`	to simulate the proposed test for the bivariate case using the Beta errors
`main_Beta_univariate.cpp`	to simulate the proposed test for the univariate case using the Beta errors

C++ main file	Description
`main_Exp_bivariate.cpp`	to simulate the proposed test for the bivariate case using the exponential errors
`main_Exp_univariate.cpp`	to simulate the proposed test for the univariate case using the exponential errors
`main_MN_bivariate.cpp`	to simulate the proposed test for the bivariate case using the mixtures of standard normal errors
`main_MN_univariate.cpp`	to simulate the proposed test for the univariate case using the mixtures of standard normal errors
`main_SN_bivariate.cpp`	to simulate the proposed test for the bivariate case using the skew-normal errors
`main_SN_univariate.cpp`	to simulate the proposed test for the univariate case using the skew-normal errors
`main_Beta_others.cpp`	to simulate all the other tests using the Beta errors
`main_Exp_others.cpp`	to simulate all the other tests using the exponential errors
`main_MN_others.cpp`	to simulate all the other tests using the mixtures of standard normal errors
`main_SN_others.cpp`	to simulate all the other tests using the skew-normal errors

Note: You may obtain some numbers that are slightly different from the numbers reported in the paper, but these differences will not change the main findings reported in the Monte-Carlo study section. The reason is that we use a GSL random number generator algorithm to generate random samples from known probability distributions, and this algorithm employs hardware configurations and interrupts so that random numbers are actually hardware-dependent. Also, to ensure that the generated random samples are distinct and have maximum randomness, we use random seeds (gsl_rng_get) in Monte-Carlo loops.

License

Contact

Ba Chu - ba.chu@carleton.ca

Project Link: https://github.com/wave1122/DcorrTest