Package: sTSD (via r-universe)

December 21, 2024

Type Package

Title Simulate Time Series Diagnostics

Version 0.1.0

Maintainer Steven Miller <steve@svmiller.com>

Description These are tools that allow users to do time series diagnostics, primarily tests of unit root, by way of simulation. While there is nothing necessarily wrong with the received wisdom of critical values generated decades ago, simulation provides its own perks. Not only is simulation broadly informative as to what these various test statistics do and what are their plausible values, simulation provides more flexibility for assessing unit root by way of different thresholds or different hypothesized distributions.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

RoxygenNote 7.3.2

NeedsCompilation no

Author Steven Miller [aut, cre]
(<https://orcid.org/0000-0003-4072-6263>)

Repository CRAN

Date/Publication 2024-12-20 10:40:02 UTC

Contents

spp_test *Simulate a Phillips-Perron Test to Assess Unit Root in a Time Series*

Description

spp_test() provides a simulation approach to assessing unit root in a time series by way of the Phillips-Perron test. It takes a vector and performs three Phillips-Perron tests (no drift, no trend; drift, no trend; drift and trend) and calculates both rho and tau statistics as one normally would. Rather than interpolate or approximate a *p*-value, it simulates some user-specified number of Phillips-Perron tests of a known, white-noise time series matching the length of the time series the user provides. This allows the user to make assessments of non-stationarity or stationarity by way of simulation rather than approximation from received critical values by way of books or tables some years out of date.

Usage

```
spp_test(x, lag_short = TRUE, n_sims = 1000, sim_hyp = "nonstationary")
```

Arguments

x	a vector
lag_short	logical, defaults to TRUE. If TRUE, the "short-term" lag is used for the Phillips-Perron test. If FALSE, the "long-term" lag is used.
n_sims	the number of simulations for calculating an interval or distribution of test statistics of a white-noise time series. Defaults to 1,000.
sim_hyp	can be either "stationary" or "nonstationary". If "stationary" (the default), the function runs Phillips-Perron tests on simulated stationary (pure white noise) data. This allows the user to assess compatibility/plausibility of the test statistic against a distribution of test statistics that are known to be pure white noise (in expectation). If "nonstationary", the function generates three different data sets of a pure random walk, a random walk with a drift, and a random walk with a drift and trend. It then runs Phillips-Perron tests on all those. This allows the user to assess the compatibility/plausibility of their test statistics with data that are known to be nonstationary in some form.

Details

Some knowledge of Augmented Dickey-Fuller and the Phillips-Perron procedure is assumed here. Generally, the Phillips-Perron test purports to build on the Augmented Dickey-Fuller procedure through two primary means. The first is relaxing the need to specify or assume lag structures ad hoc or ex ante. Only a short-term lag or long-term lag are necessary. The second is that its robust to various forms of heteroskedasticity in the error term.

The short-term and long-term lags follow the convention introduced in the Phillips-Perron test. The short-term lag uses the default number of Newey-West lags, defined as the floor of 4*(n/100)^.25 where ‘n‘ is the length of the time series. The long-term lag substitutes 4 for 12 in this equation.

This function specifies three different types of tests: 1) no drift, no trend, 2) drift, no trend, and 3) drift and trend. In the language of the ‘lm()‘ function, the first is ‘lm(y ~ ly - 1)‘ where ‘y‘ is the value of ‘y‘ and ‘ly‘ is its first-order lag. The second test is ‘lm(y ~ ly)‘, intuitively suggesting the *y*-intercept in this equation is the "drift". The third would be ‘lm(y ~ ly + t)‘ with ‘t‘ being a simple integer that increases by 1 for each observation (i.e. a time-trend).

There are two types of statistics in the Phillips-Perron test: rho and tau. Of the two, tau is the more intuitive statistic and compares favorably to its corollary statistic in the Augmented Dickey-Fuller test. It's why you'll typically see tau reported as the statistic of interest in other implementations. rho has its utility for more advanced diagnostics, though. Both are calculated in this function, though tau is the default statistic.

None of this is meant to discourage the use of Fuller (1976) or its various reproductions for the sake of diagnosing stationarity or non-stationary, and I will confess their expertise on these matters outpaces mine. Consider the justification for this function to be largely philosophical and/or experimental. Why not simulate it? It's not like time or computing power are huge issues anymore.

This is always awkwardly stated, but it's a good reminder that the classic Dickey-Fuller statistics are mostly intended to come back negative. That's not always the case, to be clear, but it is the intended case. You assess the statistic by "how negative" it is. Stationary time series will produce test statistics more negative ("smaller") than those produced by non-stationary time series. In a way, this makes the hypotheses implicitly one-tailed (to use that language).

This function removes missing values from the vector before calculating test statistics.

Value

`spp_test()` returns a list of length 3. The first element in the list is a matrix of rho statistics and tau statistics calculated by the Phillips-Perron test. The second element is a data frame of the simulated rho and tau statistics of either a known white-noise time series or three different non-stationary time series (pure random walk, random walk with drift, random walk with drift and trend). The third element is some attributes about the procedure for post-processing.

Author(s)

Steven V. Miller

Examples

```
a <- rnorm(25) # white noise
b <- cumsum(a) # random walk

spp_test(a, n_sims = 25)
spp_test(b, n_sims = 25)
```

ur_summary *Summarize Unit Root Test Simulations*

Description

`ur_summary()` provides a summary of the unit root tests included in this package.

Usage

```
ur_summary(obj, pp_stat = "tau", ...)
```

Arguments

`obj`	the object to be summarized, of class 'spp_test'
`pp_stat`	the statistic to be summarized: either "tau" or "rho"
`...`	additional argument, currently ignored

Details

This function makes ample use of the "attributes" element in the list produced by the unit root simulations.

Value

`ur_summary()` produces console output that offers a summary assessment about the presence of a unit root based on your simulations.

Author(s)

Steven V. Miller

USDSEK *The USD/SEK Exchange Rate*

Description

A data frame on the USD/SEK exchange rate (i.e. how many Swedish crowns does one dollar get you).

Usage

```
USDSEK
```

Format

A data frame with 3905 observations on the following 2 variables.

`date` a date

`close` the exchange rate at the close of trading

Details

Data come by way of **quantmod**.

Index