# Package 'timesboot'

February 20, 2015

**Type** Package

**Title** Bootstrap computations for time series objects

**Version** 1.0

**Date** 2013-07-31

**Author** Francisco Juretig

**Maintainer** Francisco Juretig <fjuretig@yahoo.com>

**Description** Computes bootstrap CI for the sample ACF and periodogram

**License** GPL-2

**Depends** boot

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2013-08-02 07:22:53

## R topics documented:



---

`boot_autocov` *A function that computes the bootstrapped autocovariances for a time series object. The computations are done via phase scrambling bootstrap*

---

### Description

The function resamples the time series object and returns the average , upper, and lower bounds for the autocovariances for each lag.

## Usage

```
boot_autocov(series, replic = 5000, plot = TRUE, alpha = 0.05)
```

## Arguments

| | |
|---|---|
| series | A time series object |
| replic | The amount of boostrap replicates |
| plot | TRUE,FALSE indicating whether the plot is desired |
| alpha | the alpha needed for the intervals |

## Value

| | |
|---|---|
| average | The average ACF for each lag |
| lower | The ACF lower quantile for each lag |
| upper | The ACF upper quantile for each lag |

## Author(s)

Francisco Juretig

## Examples

```
boot_autocov(AirPassengers,replic=1000,alpha=0.05)


function (series, replic = 5000, plot = TRUE, alpha = 0.05)
{
    if (is.ts(series) == TRUE) {
        library(boot)
        kas = tsboot(series, statistic, R = replic, sim = "scramble")
        quantiles = matrix(0, length(kas$t[1, ]), 3)
        for (i in 2:length(kas$t[1, ])) {
            cp = kas$t[, i]
            quantiles[i, 1] = quantile(cp, alpha)
            quantiles[i, 2] = quantile(cp, 1 - alpha/2)
            quantiles[i, 3] = mean(cp)
        }
        quantiles = quantiles[-1, ]
        if (plot == TRUE) {
            par(mfrow = c(1, 2))
            x = seq(1, length(quantiles[, 1]), 1)/frequency(series)
            plot(x, quantiles[, 1], type = "l", col = "blue",
                main = "Bootstraped Correlogram", ylab = "value",
                lwd = 1, xlab = "lag")
            polygon(c(x, rev(x)), c(quantiles[, 1], rev(quantiles[,
                2])), col = "skyblue")
            lines(x, quantiles[, 3], type = "o", col = "black",
                pch = 20)
            abline(a = 0, b = 0)
```

```
            plot(acf(series, plot = FALSE), main = "Asymptotic Correlogram",
                ylim = c(-1, 1))
        }
        lista = list(average = quantiles[, 1], upper = quantiles[,
            2], lower = quantiles[, 3])
        return(lista)
    }
    else {
        return("Object is not a time-series")
    }
}
```

---

`boot_spec` *Function that computes boostrapped confidence intervals for the sample periodogram.*

---

## Description

The function resamples a time series object using phase scramble bootstrap and returns the average,lower and upper confidence bounds.

## Usage

```
boot_spec(series, replic = 5000, spansa = c(11, 21),
plot = TRUE, de_trend = FALSE, alpha = 0.05)
```

## Arguments

| | |
|---|---|
| `series` | A time series object |
| `replic` | The amount of replications used in the bootstrap step |
| `spansa` | The spans for smoothing the spectrum |
| `plot` | TRUE,FALSE. Whether plotting is desired. Default is TRUE. |
| `de_trend` | TRUE,FALSE. Should de-trending be applied to the series.Default is FALSE |
| `alpha` | The alpha used in the construction of the CI. Default is 0.05 |

## Value

| | |
|---|---|
| `average` | The average value for each frequency |
| `upper` | The upper value for each frequency |
| `lower` | The lower value for each frequency |

## Author(s)

Francisco Juretig

## Examples

```
boot_spec(AirPassengers,replic=1000,alpha=0.05)

function (series, replic = 5000, spansa = c(11, 21), plot = TRUE,
    de_trend = FALSE, alpha = 0.05)
{
    if (is.ts(series) == TRUE) {
     library(boot)
        kas = tsboot(series, redraw, p = spansa, detrend = de_trend,
            replic, sim = "scramble")
        span1 = spansa[1]
        span2 = spansa[2]
        quantiles = matrix(0, length(kas$t[1, ]), 3)
        Xvalues = spec.pgram(series, spans = c(span1, span2),
            plot = FALSE)
        for (i in 1:length(kas$t[1, ])) {
            cp = kas$t[, i]
            quantiles[i, 1] = quantile(cp, alpha)
            quantiles[i, 2] = quantile(cp, 1 - alpha/2)
            quantiles[i, 3] = mean(cp)
        }
        if (plot == TRUE) {
            maximus = max(quantiles[, 3])
            plot(Xvalues$freq, quantiles[, 1], type = "l", col = "blue",
                ylim = c(0, maximus * 2), main = "Bootstraped Periodogram",
                ylab = "periodogram", lwd = 1, xlab = "frequency")
            polygon(c(Xvalues$freq, rev(Xvalues$freq)), c(quantiles[,
                1], rev(quantiles[, 2])), col = "skyblue")
            lines(Xvalues$freq, quantiles[, 3], type = "o", col = "black",
                pch = 20)
        }
        lix = list(freq = Xvalues$freq, upper = quantiles[, 2],
            lower = quantiles[, 1], mean = quantiles[, 3])
        return(lix)
    }
    else {
        return("Object is not a time-series")
    }
  }
```

---

redraw *Auxiliary function that computes the spectrum*

---

## Description

Auxiliary function that computes the spectrum

## Usage

```
redraw(series, ...)
```

### Arguments

| | |
|---|---|
| series | A time series object |
| ... | optional arguments |

### Value

| | |
|---|---|
| spec | Periodogram |

### Author(s)

Francisco Juretig

### Examples

```
function (series, ...)
{
    args <- list(...)
    ws = spec.pgram(series, spans = args$p, plot = FALSE, detrend = args$detrend)
    return(ws$spec)
  }
```

---

## statistic *Auxiliary function that returns the sample acf values*

---

### Description

Auxiliary function that returns the sample acf values

### Usage

```
statistic(ts)
```

### Arguments

| | |
|---|---|
| ts | A time series object |

### Value

| | |
|---|---|
| acf | acf values |

### Author(s)

Francisco Juretig

## Examples

```
function (ts)
{
    cm = acf(ts, plot = FALSE)
    return(cm$acf)
  }
```

# Index