# Package ‘QPBoot’

June 1, 2017

**Type** Package

**Title** Model Validation using Quantile Spectral Analysis and Parametric Bootstrap

**Version** 0.2

**Date** 2017-05-15

**Author** Stefan Birr [aut, cre]

**Maintainer** Stefan Birr <stefan.birr@ruhr-uni-bochum.de>

**Depends** R (>= 3.0.0), quantspec

**Imports** stats4, utils, methods, abind, rugarch

**Description** Provides functionality for model validation by computing a parametric bootstrap and comparing the Quantile Spectral Densities.

**License** GPL (>= 2)

**LazyData** TRUE

**Collate** 'aux-functions.R' 'generic-accessors.R' 'tsModel-class.R' 'Models.R' 'QPBoot-class.R' 'QPBoot-package.R' 'data.R'

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-06-01 12:18:50 UTC

## R topics documented:

## `QPBoot-package` *Quantile Spectral Analysis for Parametric Bootstrap*

### Description

Methods to validate model assumptions by using Quantile Spectral Analysis and Paramtric Bootstraps

### Author(s)

Stefan Birr

## `alphaq` *Returns a function to retrieve the $\alpha$-quantile from a vector.*

### Description

To be used in conjunction with apply.

### Usage

```
alphaq(a)
```

### Arguments

| | |
|---|---|
| `a` | the quantile level $\alpha$ |

### Value

Returns a function that takes an argument `x` and calculates the $\alpha$-quantile from the vector `x`.

### Examples

```
#####################################################################################
```

## `arg.names` *Returns the argument names of a function.*

### Description

To check if an estimate or simulate function is valid.

### Usage

```
arg.names(fun)
```

### Arguments

| | |
|---|---|
| `fun` | a function from which the arguments should be retrieved from |

### Value

a vector of characters containing the names of the arguments of the function `fun` ##############################################################################

## `compare.arg.names` *Compare arguments with character vector*

### Description

To check if an estimate or simulate function is valid.

### Usage

```
compare.arg.names(fun, names)
```

### Arguments

| | |
|---|---|
| `fun` | a function from which the arguments should be retrieved from |
| `names` | names the arguments should be compared to |

### Value

Logical if the names of the arguments are identical to `names` ##############################################################################

## `computeCIs-QPBoot` *Pointwise Confidence Intervalls*

### Description

Depending on `method` this calculates pointwise confidence intervalls for a smoothed periodgram that belongs to a time series defined by `model` and `param`. If `(method = "quantiles")` it computes the $\alpha/2$ and $1-\alpha/2$ quantiles from the Values of the simulated smoothed periodograms and returns those. If `(method = "norm")` it uses the asymptotic normality of the smoothed periodograms by estimating `mean` and `standard deviation` for each frequency and computing the $\alpha/2$ and $1-\alpha/2$ quantiles from a normal distribution with the estimated parameters.

### Usage

```
computeCIs(object, alpha = 0.05, method = c("quantiles", "norm"),
  levels = object@sPG@levels[[1]])
```

### Arguments

| | |
|---|---|
| `object` | the QPBoot object that will be plotted |
| `alpha` | the significiant level of the confidence intervalls, defaults to `0.05` |
| `method` | either "quantile" or "norm", determines how the confidence intervalls are calculated. see description for details |
| `levels` | numeric vector containing values between 0 and 1 for which the smoothedPG. Will be estimated. These are the quantiles levels that are used for the validation |

### Value

Returns a `list` with four elements

`q_up`

`q_low`

`mean`

`sd`

## `dax` *DAX: Deutscher Aktien Index 2000–2010*

### Description

Contains the closing values of the DAX stock index for the years 2000–2010.

### Format

A univariate time series with 2802 observations; a `numeric` vector

### Details

The data was downloaded from the YFinance section on the Quandl Website.

### References

Quandl `https://www.quandl.com/data/YAHOO/INDEX_GDAXI-DAX-Index-Germany`

### Examples

```
plot(dax,type = "l")
```

---

## `Estimate-tsModel` *Estimates the parameter of a tsModel-class*

---

### Description

The methods estimates the parameters from the time-series in `data`, using the method defined in the tsModel-class.

### Usage

```
## S4 method for signature 'tsModel'
Estimate(object, data)
```

### Arguments

| | |
|---|---|
| `object` | the tsModel-class for that the parameters shall be estimated |
| `data` | a univariate numeric vector containing the time-series data |

### Value

Returns the estimated parameters `par` and sets the `par` slot of the tsModel-class.

---

## `generics-accessors` *Generic functions for accessing attributes of objects These generic functions are needed to access or set the objects' attributes.*

---

### Description

Defines generic function names for the package QPBoot

### Usage

```
Estimate(object, ...)

Simulate(object, ...)

setEstimate(object, ...)

setSimulate(object, ...)

setParameter(object, ...)
```

### Arguments

`object` object from which to get the value

`...` optional parameters; for documentation see the documentation of the methods to each of the generic.

## `Models` *Predefined Time-Series Models*

### Description

Creates a tsModel-class object representing a time-series model

### Usage

```
getARMA(spec = list(ar.order = NA, ma.order = NA))

getAR(spec = list(ar.order = 1))

getMA(spec = list(ma.order = 1))

getNoise()

getGARCH(spec = list(alpha = 1, beta = 1))

getARCH(spec = list(alpha = 1))

getEGARCH(spec = list(alpha = 1, beta = 1))

getARMA_GARCH(spec = list(ar = 1, ma = 1, alpha = 1, beta = 1))
```

### Arguments

`spec` a list specifying the structure of the parameters of the model plot-QPBoot *Plot the values of a* QPBoot.

## Description

Creates a K x K plot depicting a smoothed quantile periodogram. Optionally pointwise confidence intervals from the parametric bootstrap can be displayed. In each of the subplots either the real part (on and below the diagonal; i. e., $\tau_1 \leq \tau_2$) or the imaginary parts (above the diagonal; i. e., $\tau_1 > \tau_2$) of

the smoothed quantile periodogram (blue line)

pointwise confidence intervals from the parametric bootstrap (light gray area)

for the combination of levels $\tau_1$ and $\tau_2$ denoted on the left and bottom margin of the plot are displayed. The method argument determines how the confidence intervalls are calculated.

**quantile** calculates the (1-$\alpha/2$) and $\alpha/2$ quantiles from the bootstrap

**norm** asymptotic normality of the smoothed Periodograms is used, mean and standard deviation are estimated from the bootstrap

## Usage

```
## S4 method for signature 'QPBoot,ANY'
plot(x, ptw.CIs = 0.1, method = "quantiles",
  ratio = 3/2, widthlab = lcm(1), xlab = expression(omega/2 * pi),
  ylab = NULL, type.scaling = c("individual", "real-imaginary", "all"),
  frequencies = x@sPG@frequencies, levels = intersect(x@sPG@levels[[1]],
  x@sPG@levels[[2]]))
```

## Arguments

| | |
|---|---|
| x | The SmoothedPG object to plot |
| ptw.CIs | the confidence level for the conspec = garchSpec(model = param)fidence intervals to be displayed; must be a number from [0,1]; if null, then no confidence intervals will be plotted. |
| method | either "quantile" or "norm", determines how the confidence intervalls are calculated. see description for details |
| ratio | quotient of width over height of the subplots; use this parameter to produce landscape or portrait shaped plots. |
| widthlab | width for the labels (left and bottom); default is lcm(1), cf. layout. |
| xlab | label that will be shown on the bottom of the plots; can be an expression (for formulas), characters or NULL to force omission (to save space). |
| ylab | label that will be shown on the left side of the plots; can be an expression (for formulas), characters or NULL to force omission (to save space). |

`type.scaling` a method for scaling of the subplots; currently there are three options: `"individual"` will scale each of the `K^2` subplots to minimum and maximum of the values in that plot, `"real-imaginary"` will scale each of the subplots displaying real parts and each of the subplots displaying imaginary parts to the minimum and maximum of the values display in these subportion of plots. The option `"all"` will scale the subplots to the minimum and maximum in all of the subplots.

`frequencies` a set of frequencies for which the values are to be plotted.

`levels` a set of levels for which the values are to be plotted.

### Value

Returns the plot described in the Description section.

## QPBoot-class *Class for a Parametric Bootstrap based on Quantile Spectral Analysis*

### Description

`QPBoot` is a class to compute und contain the results of a parametric bootstrap based on Quantile Spectral Analysis.

### Slots

`data` the original data where the parametric bootstrap is based upon

`sPG` a smoothed quantile Periodogram from the `quantspec`-package calculatet from `data`

`model` parametric model for the bootstrap from tsModel-class, some Examples can be found in Models

`param` parameter estimated for the `model` from the `data`

`sPGsim` smoothed quantile periodograms of the simulated time series

## QPBoot-constructor *qpBoot*

### Description

Create an instance of the `QPBoot` class by doing 3 things

1. Estimates a parametric `model` from a given set of `data`, this estimate can be overwritten by using the parameter `fix.param`
2. Simulates from that `model` and computes the smoothed Quantile Periodogram (smoothedPG) for each simulated time series and the given `data`
3. Returns an object of the class QPBoot with the calculated smoothed Periodograms

### Usage

```
qpBoot(data, model = getARMA(list(ar.order = 2, ma.order = 0)),
  levels = c(0.1, 0.5, 0.9), frequencies = 2 * pi/length(data) *
  0:(length(data) - 1), weight = kernelWeight(bw = 0.1), SimNum = 1000,
  fix.param = NULL)
```

### Arguments

`data` numeric vector, containing the time-series data

`model` an object from the class tsModel-class.

`levels` numeric vector containing values between 0 and 1 for which the smoothedPG. Will be estimated. These are the quantiles levels that are used for the validation

`frequencies` a vector containing frequencies at which to determine the smoothed periodogram.

`weight` an object of the class KernelWeight that is used to in the estimation of the smoothedPG.

`SimNum` number of bootstrap

`fix.param` defaults to `NULL`. In this case the parameters for the simulations are estimated via the methode defined in the argument `model`. If this is not `NULL`, it has to contain a list that can be used to set the parameters in the tsModel-class. All simulations are then done with these fixed parameters.

---

## `setEstimate-tsModel` *Sets the estimation Method of a tsModel-class*

---

### Description

Defines the method that should be used when calling Estimate on the tsModel-class. The `Estimate` function must have exactly the following two arguments: `data` (numeric vector) and `spec` (a list) and return a list containing the estimated parameters.

### Usage

```
## S4 method for signature 'tsModel'
setEstimate(object, estimate)
```

### Arguments

`object` the tsModel-class for that the estimation method shall be defined

`estimate` a function that has exactly two argumens: data (numeric vector) and spec (a list) and returns a list containing the estimated parameters.

### Value

Sets the `est_function` slot of the tsModel-class.

## setParameter-tsModel *Sets the Parameter of a tsModel-class manually*

### Description

This can be used to set the `par` slot of a tsModel-class by hand, in contrast to call Estimate on a given data set. The parameter should be a list with the name an the values of the parameters to be set. See the example below.

### Usage

```
## S4 method for signature 'tsModel'
setParameter(object, par)
```

### Arguments

| | |
|---|---|
| `object` | the tsModel-class for that the Parameter will be set |
| `par` | a list that contains the names and values of the parameters |

### Value

Nothing, it sets the `par` slot of the tsModel-class.

## setSimulate-tsModel *Sets the simulation Method of a tsModel-class*

### Description

Defines the method that will be used when calling Simulate on the tsModel-class. The passed `Simulate` function must have exactly the following three arguments: `n` (numeric), `spec` (a list) and `par` (another list). It returns a numeric vector with the simulated data.

### Usage

```
## S4 method for signature 'tsModel'
setSimulate(object, Simulate)
```

### Arguments

| | |
|---|---|
| `object` | the tsModel-class for that the estimation method shall be defined |
| `Simulate` | a function that has exactly three argumens: `n` (numeric), `spec` (a list) and `par` (another list). |

### Value

Nothing, it sets the `sim_function` slot of the tsModel-class.

## Simulate-tsModel *Simulates from a tsModel-class*

### Description

`Simulate` produces a numeric vector of length `n` with data simulated according to the `sim_function` defined in the tsModel-class. To use this function the `par` slot has to be set, either by calling Estimate or directly via setParameter.

### Usage

```
## S4 method for signature 'tsModel'
Simulate(object, n)
```

### Arguments

| | |
|---|---|
| `object` | the simulation will be based on this tsModel-class |
| `n` | length of the simulated data, will be rounded down if it is not a whole number |

### Value

Returns a numeric vector of length `n`, that contains data, simulated according to the slot `sim_function` and `par` in the tsModel-class object.

## tsModel-class *Class for a Parametric Time-Series Model*

### Description

`tsModel` is a class to contain parametric time-series models (like ARMA oder GARCH) so that they can be used as arguments for qpBoot. There are some premade Models

### Slots

`name` the name of the model (e.g. "GARCH")

`spec` a list containing additional specification of the model

`env` An environment to allow for slots which need to be accessable in a call-by-reference manner:

- `est_function` a function implementing an estimator for the parameters of the model. It has the argumens `object` and `data` and returns the estimated parameter. Also it sets `par` to the estimated value.
- `sim_function` a function implementing a way to simulate from the the model. It has the argumens `object` and `n`. Note that `par` has to be set in order to simulate.
- `par` a numeric vector that contains the parameters of the model. Can be empty at the beginning.

# Index