Package 'BETS'

December 20, 2016

Type Package

Title Brazilian Economic Time Series

Version 0.0.98

Date 2016-12-19

Depends R (>= 3.2.0)

Imports testthat (>= 0.9.1), rootSolve, grnn, ggplot2, plotly, urca,
TTR, forecast, TSA, FinTS, fpp, stringi, sqldf, foreign,
lmtest, normtest, zoo, rugarch, colorspace, fracdiff, tseries,
timeDate, htmlwidgets, quadprog, Rcpp, gtable, scales, viridis,
rmarkdown, seasonal, stringr, mFilter, dygraphs

Author
Pedro Costa Ferreira <pedro.guilherme@fgv.br>, Jonatha Costa <jonatha.costa@fgv.br>,
Talitha Speranza <talitha.speranza@gmail.com>.

Maintainer Pedro Costa Ferreira <pedro.guilherme@fgv.br>

BugReports https://github.com/pedrocostaferreira/BETS/issues

Description
It provides easy access to the most important economic time series in Brazil, from the Getulio Vargas Foundation, the Central Bank of Brazil and the Brazilian Institute of Geography and Statistics. In addition to presenting tools for the management, analysis (generating dynamic documents automatically with the analyses) and export of these time series.

License GPL-3

LazyData true

Encoding UTF-8

URL https://github.com/pedrocostaferreira/BETS

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-20 01:31:49

R topics documented:

`add.notes` *Add notes*

Description

Add notes

Usage

```
add.notes(series.list, xlim, ylim, names = NULL)
```

Arguments

`series.list`	A `ts object`
`xlim`	A `vector`
`ylim`	A `vector`
`names`	A `character`

Author(s)

Talitha Speranza <`talitha.speranza@fgv.br`>

`BETS` *BETS: A package for obtaining and analysing thousands of Brazilian economic time series.*

Description

The Brazilian Economic Time Series (BETS) package provides access and information about the most important Brazilian economic time series.

These series are created by three influential centers: the Central Bank of Brazil (BCB), the Brazilian Institute of Geography and Statistics (IBGE) and the Brazilian Institute of Economics, from the Getulio Vargas Foundation (FVG-IBRE). Currently, there are more than 30.000 available time series, most of them free of charge. Besides providing access to this vast database, the package allows the user to interact with data in an easy and friendly way.

For instance, the user can search for a time series using keywords. More importantly, it installs several consecrated packages for time series analysis, giving the user the option to perform a complete analysis without having to worry about installing and loading other packages. In a near future, the authors will publish a series of R exercises to be solved with BETS and its statiscal/econometrical tools, therefore helping the user to understand the behavior of brazilian time series.

Note

The authors would like to thank the support given by the Getulio Vargas Foundation (FGV).

Author(s)

Pedro Costa Ferreira <`pedro.guilherme@fgv.br`>, Jonatha Costa <`jonatha.costa@fgv.br`>, Talitha Speranza <`talitha.speranza@fgv.br`>

BETS.chart — *Create a chart with a pre-defined BETS series*

Description

Create a chart with a pre-defined BETS series

Usage

```
BETS.chart(alias, lang = "en", out = "png", file = NULL, start = c(2006,
  1), ylim = NULL, open = TRUE)
```

Arguments

`alias` A `character`. The alias of the chart. A complete list of aliases for available charts is under the 'Details' section.

`lang` A `character`. The language. For now, only 'en' (english) is available.

`out` A `character`. The format of the output, that is, the image file. Can be either 'pdf' or 'png'. 'pdf' is a better choice if you need high resolution images.

`file` A `character`. The whole path, including a custom name, for the output (an image file). The default value is 'graphs//parameter_alias' (the 'graphs' directory is under the BETS installation directory).

`start` A `vector`.

`ylim` A `vector`.

`open` A `boolean`.

Details

VALUE	DESCRIPTION
'ipca_with_core'	National consumer price index (IPCA) - in 12 months and Broad national consumer price index - Core IP
'ulc'	Unit labor cost - ULC-US$ - June/1994=100
'eap'	Economically active population
'cdb'	Time deposits (CDB/RDB-preset) - Daily return (percentage)
'indprod'	Prodcution Indicators (2012=100) - General
'selic'	Interest rate - Selic accumulated in the month in annual terms (basis 252)
'unemp'	Open unemployment rate - by metropolitan region - Brasil (weekly)
'vargdp'	GDP - real percentage change in the year

Value

If the parameter `file` is not set by the user, the chart will be placed in the 'graphs' directory, under the BETS installation directory.

Author(s)

Talitha Speranza <talitha.speranza@fgv.br>

`BETS.corrgram` *Plot the ACF or the PACF of a time series*

Description

Plot correlograms using plot.ly and several other options that differ theses plots from forecasts ACF and PACF.

Usage

```
BETS.corrgram(ts, lag.max = 12, type = "correlation", mode = "simple",
  ci = 0.95, style = "plotly", knit = F)
```

Arguments

`ts`	An object of type `ts` or `xts`. The time series for which the plot must be constructed.
`lag.max`	A `numeric` value. The number of lags to be shown in the plot.
`type`	A `character`. Can be either 'correlation' (for the ACF) or 'partial' (for the PACF).
`mode`	A `character`. Set this parameter to 'bartlett' if you want the variance to be calculated according to Bartlett's formula. Otherwise, it is going to be simply equal to `1/sqrt(N)`.
`ci`	A `numeric` value. The confidence interval to be shown in the plot.
`style`	A `character`. Set this parameter to 'normal' if you want it made with ggplot2 or to 'plotly' if you want to be a plotly object.
`knit`	A `boolean`. If you're using this function to exhibit correlograms on a R dynamic report, set this parameter to true.

Value

A plot and a `vector` containing the correlations.

Author(s)

Talitha Speranza <talitha.speranza@fgv.br>

`BETS.dashboard` *Create a BETS custom dashboard*

Description

Generate thematic dashboards using a selection of BETS time series and charts. For now, themes and charts are pre-defined.

Usage

```
BETS.dashboard(type = "inflation", country = "BR", parameters = NULL,
  saveas = NA)
```

Arguments

`type` A `character`. The theme of the dashboard. The only option, for the time being, is ’inflation’.

`country` A `character`. The country from which data must drawn. For now, BETS only accepts ’BR’ (Brazil)

`parameters` A `list`. A list of parameters. See the ’Details’ section for a description of these parameters for each type of dashboard.

`saveas` A `character`. A path and a name for the dashboard file (a .pdf file). If this parameter is not provided, the dashboard will be saved inside the ’dashboards’ folder, under the BETS installation directory.

Details

Inflation Dashboard Parameters

- `text` - The text to be printed in the dashboard. Separate paragraphs with two backslashes ’n’ and pages with ’##’. There are no other syntax rules.
- `author` - The author’s name.
- `email` - The author’s email.
- `url` - The author’s webpage.
- `logo` - The author’s business logo.

Value

A .pdf file (the dashboard)

Author(s)

Talitha Speranza `<talitha.speranza@fgv.br>`

`BETS.deflate` *Deflate a time series*

Description

Deflate a time series using a deflator series. The deflator can be an index, a percentage or a point percentage series.

Usage

```
BETS.deflate(ts, deflator, type = "index")
```

Arguments

`ts`	A `ts` object. The time series to be deflated.
`deflator`	A `ts` object. The deflator series.
`type`	A `character`. Can be either `'index'`, `'point.perc'` (for point percentage) or `'perc'` (for percentage).

Value

The deflated series.

Author(s)

Talitha Speranza <`talitha.speranza@fgv.br`>

`BETS.dummy` *Create a monthly or quarterly dummy*

Description

Returns a monthly or quarterly dummy (a time series with only 0s and 1s).

Usage

```
BETS.dummy(start = NULL, end = NULL, frequency = 12, year = NULL,
  month = NULL, quarter = NULL, date = NULL, from = NULL, to = NULL)
```

Arguments

start — An integer vector. The period of the first observation. The first element of the vector specifies the year of the first observation, whereas the second, the month (for monthly dummies) or quarter (for quarterly dummies)

end — An integer vector. The period of the last observation. The first element of the vector specifies the year of the last observation, whereas the second, the month (for monthly dummies) or quarter (for quarterly dummies)

frequency — An integer. The frequency of the dummy, that is, the number of observations per unit of time. The default is 12 (a monthly dummy).

year — An integer, a seq or a vector. The years for which the dummy must be set to 1. All periods of these years will be set to 1.

month — An integer, a seq or a vector. The months for which the dummy must be set to 1. These months will be set to 1 for all years.

quarter — An integer, a seq or a vector. The quarters for which the dummy must be set to 1. The quarters will be set to 1 for all years.

date — a list. The periods for which the dummy must be set to one. Periods must be represented as integer vectors, as described for start and end.

from — An integer vector The starting period of a sequence of perids for which the dummy must be set to one. Periods must be represented as integer vectors, as described for start and end.

to — The ending period of a sequence of perids for which the dummy must be set to one. Periods must be represented as integer vectors, as described for start and end.

Value

A monthly or a quarterly ts object.

See Also

ts, BETS.dummy

Examples

```
#1 from a specific date to another specific date
BETS.dummy(start = c(2000,1),end = c(2012,5),frequency = 12,from = c(2005,1),to = c(2006,12))


#Other options that may be helpful:

#over a month equal to 1
BETS.dummy(start = c(2000,1), end = c(2012,5), frequency = 12, month = c(5,12))

#Months equal to 1 only for some year
BETS.dummy(start = c(2000,1), end = c(2012,5), frequency = 12, month = 5, year = 2010)
BETS.dummy(start = c(2000,1), end = c(2012,5), frequency = 12, month = 8, year = 2002)
```

```
#Months equal to 1 only for some years
BETS.dummy(start = c(2000,1), end = c(2012,5), frequency = 12, month = 5, year = 2005:2007)
BETS.dummy(start = c(2000,1), end = c(2012,5), frequency = 12, month = 3, year = c(2005,2007))
BETS.dummy(start = c(2000,1), end = c(2012,5), frequency = 12, month = 5:6, year = c(2005,2007))

#specific dates
BETS.dummy(start = c(2000,1), end = c(2012,5), frequency = 12, date = list(c(2010,1)))
BETS.dummy(start = c(2000,1), end = c(2012,5),
    freq = 12, date = list(c(2010,9), c(2011,1), c(2000,1)) )
```

BETS.get *Get a complete time series from a BETS database*

Description

Extracts a complete time series from either the Central Bank of Brazil (BCB), the Brazilian Institute of Geography and Statistics (IBGE) or the Brazilian Institute of Economics (FGV/IBRE).

Usage

```
BETS.get(code, data.frame = FALSE)
```

Arguments

code	A character. The unique code that references the time series. This code can be obtained by using the BETS.search function.
data.frame	A boolean. True if you want the output to be a data frame. True to ts output.

Value

A ts (time series) object containing the desired series.

Note

Due to the significant size of the databases, it could take a while to retrieve the values. However, it shouldn't take more than 90 seconds.

See Also

ts, BETS.search and seas

Examples

```
# Anual series: GDP at constant prices, in R$ (brazilian reais)
#BETS.get(1208)

# International reserves - Cash concept
#int.reserves <- BETS.get("3543")
#plot(int.reserves)

# Exchange rate - Free - United States dollar (purchase)
#us.brl <- BETS.get(3691)
#require(seasonal)
#us.brl.seasonally_adjusted <- seas(us.brl)
#plot(us.brl.seasonally_adjusted)
```

`BETS.grnn.test` *Test a set of General Regression Neural Networks*

Description

Given new values of the independent variables, tests a list of trained GRNNs and picks the best net, based on an accuracy measure between the forecasted and the actual values.

Usage

```
BETS.grnn.test(results, test.set)
```

Arguments

`results`	The object returned by BETS.grnn.train.
`test.set`	A `ts list`. The first element must be the actual values of the dependent variable. The others, the new values of the regressors.

Value

A `list` object representing the best network (according to forecasting MAPE). Its fields are:

- `mape`: The forecasting MAPE
- `model`: The network object (returned by grnn)
- `sigma`: The sigma parameter
- `id`: The id number of the network, as given by BETS.grnn.train
- `mean`: The predicted values
- `x`: The original series
- `fitted`: The fitted values
- `actual`: The actual values (to be compared with the predicted values)
- `residuals`: Difference between the fitted values and the series original values
- `regressors`: The regressors used to train the network

Author(s)

Talitha Speranza <talitha.speranza@fgv.br>

BETS.grnn.train *Train a General Regression Neural Network*

Description

Creates a set of probabilistic neural networks as proposed by Specht [1991]. The user provides a set of regressors and the function chooses which subset is the best, based on an accuracy measure (by default, the MAPE) between fited and actual values. These networks have only one parameter, the sigma, which is the standard deviation of each activation function (gaussian) of the pattern layer. Sigma can also be automatically chosen. This function builds on grnn-package.

Usage

```
BETS.grnn.train(train.set, sigma, step = 0.1, select = TRUE, names = NA)
```

Arguments

train.set	A ts list (a list of ts objects). The first element must be the dependent variable. The other elements, the regressors.
sigma	A numeric or a numeric vector. The sigma parameter, that is, the standard deviation of the activation functions (gaussians) of the pattern layer. Can be either a fixed value or a range (a vector containing the minimum and the maximum values).
step	A numeric value. If sigma is a range, the user must provide a step value to vary sigma. The function is going to select the best sigma based on MAPE.
select	A boolean. Must be set to FALSE if the regressors should not be chosen. The default is TRUE.
names	A character vector. Optional. The names of the regressors. If not provided, indexes will be used and reported.

Value

A list of result objects, each representing a network. These objects are ordered by MAPE (the 20 best MAPEs) and its fields are:

- accuracy: A numeric value. Accuracy measure between the fitted and the actual series values. By default, the MAPE. In future versions, it will be possible to change it.
- fitted: The fitted values, that is, one step ahead predicitions calculated by the trained net.
- net: An object returned by the grnn function. Represents a trained net.
- sigma: A numeric. The sigma that was chosen, either by the user or by the function itself (in case select was set to TRUE)

- `regressors`: A `character vector`. Regressors that were chosen, either by the user or by the fuction itself (in case `select` was set to `TRUE`)
- `sigma.accuracy`: A `data.frame`. Sigma versus accuracy value of the corresponding trained network. Those networks were trained using the best set of regressors.
- `residuals`: A `numeric vector`. Fitted values subtracted from the actual values.

BETS.grnn.train also returns a diagnostic of training rounds and a `sigma` versus `accuracy` plot.

Author(s)

Talitha Speranza `<talitha.speranza@fgv.br>`

`BETS.normalize` *Normalize a time series*

Description

Normalizes a time series, either by stardization or by mapping to values between 0 and 1.

Usage

```
BETS.normalize(series, mode)
```

Arguments

`series`	A `ts` object or a `ts list`. The series to be normalized.
`mode`	A `character`. The normalization method. Set this parameter to 'maxmin' to map series values to values between 0 and 1. Alternatively, set this parameter to 'scale' to standardize (substract the mean and divide by the standard deviation).

Value

A `ts` object or a `ts list`. The normalized series.

Author(s)

Talitha Speranza `<talitha.speranza@fgv.br>`

BETS.predict *Get the predicted values of a model and visualize it*

Description

This function is built upon forecast. Besides the model predictions, it returns an accuracy measure table (calculated by the accuracy function) and a graph showing the original series, the predicted values and the actual values.

Usage

```
BETS.predict(..., actual = NULL, main = "", ylab = "", xlim = NULL,
  style = "dygraphs", unnorm = NULL, knit = F)
```

Arguments

`...`	arguments passed on to forecast. If the model is a neural network, these arguments will be passed on to BETS.grnn.test.
`actual`	A `numeric vector`. The actual values (to be compared with predicted values).
`main`	A `character`. The name of the prediction plot.
`ylab`	A `character`. The Y axis label.
`xlim`	A `numeric vector`. The limits of the X axis.
`style`	A `character`. Can be either 'dygraphs' (the dygraph function will be use to make the plot, which is going to be HTML based) or 'normal' (standard R functions will be used to make the plot)
`unnorm`	A `numeric vector`. If predictions must be unnormalized, set the first element of this vector to the mean and the second, to the standard deviation.
`knit`	A `boolean`. Set this parameter to `TRUE` if

Value

Besides the prediction plot, this function returns an object whose fields are:

- `accuracy`: An object returned by accuracy. It is a table containing several accuracy measures
- `predictions`: A `numeric vector` containing the predicted values.

Author(s)

Talitha Speranza <`talitha.speranza@fgv.br`>

BETS.report *Create a dynamic report with a full analysis of a time series*

Description

Generate an automatic report with a complete analysis of a time series. For now, only a SARIMA analysis (Box & Jenkins approach) is possible. In a near future, a GRNN (General Regression Neural Network) analysis will be released. Soon after, Holt-Winters, Multilayer Perceptron, Fuzzy Logic and Box-Cox analysis will become available.

Usage

```
BETS.report(mode = "SARIMA", ts = 21864, parameters = NULL, saveas = NA)
```

Arguments

`mode`	A `character`. The type of the analysis. So far, only 'SARIMA' is available.
`ts`	A `integer` or a `ts` object. Either the ID of the series in the BETS database or a time series object (any series, not just BETS's)
`parameters`	A `list`. The parameters of the report. See the 'details' section for more information.
`saveas`	A `character`. A path and a name for the report file (an .html file). If this parameter is not provided, the report will be saved inside the 'reports' folder, under the BETS installation directory.

Details

SARIMA Report Parameters

- `lag.max`: Maximum number of lags to show on the ACFs e PACFs
- `n.ahead`: Prevision horizon (number of steps ahead)

Value

An .html file (the report)

Author(s)

Talitha Speranza `<talitha.speranza@fgv.br>`

BETS.save.sas *Export a time series to SAS*

Description

Writes a time series to a .sas (SAS) file.

Usage

```
BETS.save.sas(code, data = NULL, file.name = "series")
```

Arguments

code	An integer. The unique identifier of the series within the BETS database.
data	A data.frame or a ts. Contains the data to be written. If data is supplied, the BETS database will not be searched.
file.name	A character. The name of the output file. The default is 'series.sas'.

Value

None

Examples

```
 #Exchange rate - Free - United States dollar (purchase)
 #us.brl <- BETS.get(3691)
 #require(seasonal)
 #us.brl.seasonally_adjusted <- seas(us.brl)
#BETS.save.sas(data = us.brl.seasonally_adjusted,file.name="us.brl.seasonally_adjusted")
 # Or
 #BETS.save.sas(code=3691,file.name="us.brl")
```

BETS.save.spss *Export a time series to SPSS*

Description

Writes a time series to a .spss (SPSS) file.

Usage

```
BETS.save.spss(code, data = NULL, file.name = "series")
```

Arguments

code An `integer`. The unique identifier of the series within the BETS database.

data A `data.frame` or a `ts`. Contains the data to be written. If `data` is supplied, the BETS database will not be searched.

file.name A `character`. The name of the output file. The default is 'series.spss'.

Value

None

Examples

```
#Exchange rate - Free - United States dollar (purchase)
 #us.brl <- BETS.get(3691)
 #requires(seasonal)
 #us.brl.seasonally_adjusted <- seas(us.brl)
 #BETS.save.spss(data = us.brl.seasonally_adjusted,file.name="us.brl.seasonally_adjusted")
   # Or
 #BETS.save.spss(code=3691,file.name="us.brl")
```

BETS.save.stata *Export a time series to STATA*

Description

Writes a time series to a .dta (STATA) file.

Usage

```
BETS.save.stata(code, data = NULL, file.name = "series")
```

Arguments

code An `integer`. The unique identifier of the series within the BETS database.

data A `data.frame` or a `ts`. Contains the data to be written. If `data` is supplied, the BETS database will not be searched.

file.name A `character`. The name of the output file. The default is 'series.dta'.

Value

None

Examples

```
#Exchange rate - Free - United States dollar (purchase)
#us.brl <- BETS.get(3691)
#requires(seasonal)
#us.brl.seasonally_adjusted <- seas(us.brl)
#BETS.save.stata(data = us.brl.seasonally_adjusted,file.name="us.brl.seasonally_adjusted")
# Or
#BETS.save.stata(code=3691,file.name="us.brl")
```

BETS.search *Search for a Brazilian Economic Time Series*

Description

Searches the BETS databases for a time series by its description, source, periodicity, code, data, unit of measurement and database name.

Usage

```
BETS.search(description, src, periodicity, unit, code, start, view = TRUE,
  lang = "en")
```

Arguments

`description` A `character`. A search string to look for matching series descriptions. Check the syntax rules under the 'Details' section for better performance.

`src` A `character`. The source of the series. See the 'Details' section for a list of the available sources.

`periodicity` A `character`. The periodicity of the series. See the 'Details' section for a list of possible values.

`unit` A `character`. The unit of measurement of the data. See the 'Details' section for a list of possible values.

`code` An `integer`. The index of the series within the database.

`start` A `date`. Starting date of the series.

`view` A `boolean`. The default is `TRUE`. If set to `FALSE`, the results are NOT going to be shown.

`lang` A `character`. The search language. The default is "en" for english, but "pt" for portuguese is also possible.

Details

- Syntax rules for the parameter `description`, the search string to look for matching series descriptions:
  1. To search for alternative words, separate them by white spaces. Example: `description = "ipca core"` means that the series description must contain ’ipca’ AND’core’
  2. To search for whole expressions, surround them with `' '`. Example: `description = "'core ipca' index"` means that the series description must contain ’core ipca’ AND ’index’
  3. To exclude words from the search, insert a ~ before each of them. Example: `description = "ipca ~ core"` means that the series description must contain ’ipca’ AND must NOT contain ’core’
  4. To exclude whole expressions from the search, surround them with code’ ’ and insert a ~ before each of them. Example: `description = "~ 'ipca core' index"` means that the series description must contain ’index’ AND must NOT contain ’core ipca’
  5. It is possible to search for multiple words or expressions and to negate multiple words or expressions, as long as the preceeding rules are observed.
  6. The white space after the negation sign (~) is not required. But the white spaces AFTER expressions or words ARE required.
- Possible values for the parameter `src`:

IBGE	Brazilian Institute of Geography and Statistics
BCB	Central Bank of Brazil
FGV	Getulio Vargas Foundation
FGv-IBRE	Getulio Vargas Foundation - Brazilian Institute of Economics
BCB e FGV	Central Bank of Brazil and Getulio Vargas Foundation
BCB-Deban	Cetral Bank of Brazil - Department of Banking and Payments
BCB-Depin	Central Bank of Brazil - Department of International Reserves
BCB-Derin	Central Bank of Brazil - Department of International Affairs
BCB-Desig	Central Bank of Brazil - Department of Financial Monitoring
BCB-Secre	Central Bank of Brazil - Executive Secretariat
BCB-Demab	Central Bank of Brazil - Department of Open Market Operations
BCB-Denor	Central Bank of Brazil - Department of Financial System Regulation
BCB-Depec	Central Bank of Brazil - Department of Economics
Sisbacen	Central Bank of Brazil Information System
Abecip	Brazilian Association of Real Estate Loans and Savings Companies

- Possible values for the parameter `periodicity`:

A	anual data
M	monthly data
Q	quaterly data
W	weekly data
D	daily data

- Possible values for the parameter `unit`:

R$	brazilian reais
$	US dolars
%	percentage

Value

A `list` that can be interpreted as a `data.frame`. The fields are described below.

code	The code/index of the series within the database
description	The description of the series
periodicity	The periodicity of the series
start	Starting date of the series
source	The source of the series
unit	The unit of measurement of the data

Note

This function uses `sqldf` for optimization.

References

Central Bank of Brazil.

Examples

```
BETS.search(description="sales",view = FALSE)
# Output: BETS-package: Found 55 out of 12981 time series

BETS.search(src="Denor", view = FALSE)
# Output: BETS-package: Found 1 out of 12981 time series

BETS.search(periodicity="A", view = FALSE)
# Output: BETS-package: Found 2308 of 12981 time series
```

`BETS.std_resid` *Plot standardized residuals*

Description

Uses a model object to create a plot of standardized residuals. This model can be an Arima or an arima. In a near future, this function will also accept objects returned by BETS.grnn.train.

Usage

```
BETS.std_resid(model, alpha = 0.05)
```

Arguments

`model`	An Arima or an arima object. The model.
`alpha`	A `numeric` between 0 and 1. The significance level.

Value

Besides showing the plot, this functin returns a `numeric vector` containing the standardized residuals.

Author(s)

Talitha Speranza <`talitha.speranza@fgv.br`>

`BETS.t_test` *Test the significance of the parameters of an ARIMA model*

Description

Performs the t test on every parameter of an ARIMA model. This model can be an Arima or an arima.

Usage

```
BETS.t_test(model, nx = 0, alpha = 0.05)
```

Arguments

`model`	An Arima or an arima object. The model for which the parameters must be tested.
`nx`	An `integer`. The number of exogenous variables
`alpha`	A `numeric` value between 0 and 1. The significance level.

Value

A `data.frame` containing the standard erros, the t-statistic, the critical values and whether the null hypothesis should be rejected or not, for each model parameter.

Author(s)

Talitha Speranza <`talitha.speranza@fgv.br`>

Examples

```
require(forecast)
data("AirPassengers")
fit.air<- Arima(AirPassengers,order = c(1,1,1), seasonal = c(1,1,1), method ="ML",lambda=0)
summary(fit.air)

# Significance test for the model SARIMA(1,1,1)(1,1,1)[12]
# t.test(model = fit.air)
```

`BETS.ur_test` *Perform unit root tests*

Description

This function uses the package 'urca' to perform unit root tests on a pre-defined time series. Unlike urca functions, it returns a meaningful table summarizing the results.

Usage

```
BETS.ur_test(..., mode = "DF", level = "5pct")
```

Arguments

`...`	Arguments passed on to urca functions
`mode`	A `character`. The type of the test. For now, only the Augmented Dickey-Fuller test is available.
`level`	A `character`. The confidence level. Can be either '1pct','5pct' or '10pct'

Value

A `list` object. The first element is a `data.frame` with the test statistics, the critical values and the test results. The second, the model residuals.

Author(s)

Talitha Speranza <`talitha.speranza@fgv.br`>

`check.series` *Check series*

Description

Check series in BETS dataset

Usage

```
check.series(ts, message = NULL)
```

Arguments

`ts`	A `ts object`
`message`	A `character`

Author(s)

Talitha Speranza <`talitha.speranza@fgv.br`>

draw.cdb *Create a chart of the Time Deposits time series*

Description

Creates a plot of series 14

Usage

```
draw.cdb(file, start = NULL, ylim = NULL, open = TRUE)
```

Arguments

`file`	A `character`. The name of the file in which the plot must be printed. The extension can be either '.png' or '.pdf'. All charts are stored in the 'graphs' folder, under the BETS installation directory.
`start`	A `character`. The stating period of the series.
`ylim`	A `numeric vector`. Y axis limits.
`open`	A `boolean`. Indicates whether the plot must be opened after being created.

Value

An image file is saved in the 'graphs' folder, under the BETS installation directory.

draw.eap *Create a chart of the Economically Active Population time series*

Description

Creates a plot of series 10810

Usage

```
draw.eap(file, start = NULL, ylim = NULL, open = TRUE)
```

Arguments

`file`	A `character`. The name of the file in which the plot must be printed. The extension can be either '.png' or '.pdf'. All charts are stored in the 'graphs' folder, under the BETS installation directory.
`start`	A `character`. The stating period of the series.
`ylim`	A `numeric vector`. Y axis limits.
`open`	A `boolean`. Indicates whether the plot must be opened after being created.

Value

An image file is saved in the 'graphs' folder, under the BETS installation directory.

`draw.indprod` *Create a chart of the Production Indicators time series*

Description

Creates a plot of series 21859

Usage

```
draw.indprod(file, start = NULL, ylim = NULL, open = TRUE)
```

Arguments

`file`	A `character`. The name of the file in which the plot must be printed. The extension can be either '.png' or '.pdf'. All charts are stored in the 'graphs' folder, under the BETS installation directory.
`start`	A `character`. The stating period of the series.
`ylim`	A `numeric vector`. Y axis limits.
`open`	A `boolean`. Indicates whether the plot must be opened after being created.

Value

An image file is saved in the 'graphs' folder, under the BETS installation directory.

`draw.ipca` *Create a chart of the National Consumer Price Index time series*

Description

Creates a plot of series 13522 (NCPI), along with series 4466 (NCPI core)

Usage

```
draw.ipca(file, start = NULL, ylim = NULL, open = TRUE)
```

Arguments

`file`	A `character`. The name of the file in which the plot must be printed. The extension can be either '.png' or '.pdf'. All charts are stored in the 'graphs' folder, under the BETS installation directory.
`start`	A `character`. The stating period of the series.
`ylim`	A `numeric vector`. Y axis limits.
`open`	A `boolean`. Indicates whether the plot must be opened after being created.

Value

An image file is saved in the 'graphs' folder, under the BETS installation directory.

`draw.selic` *Create a chart of the Base Interest Rate (SELIC) time series*

Description

Creates a plot of series 4189

Usage

```
draw.selic(file, start = NULL, ylim = NULL, open = TRUE)
```

Arguments

`file` A `character`. The name of the file in which the plot must be printed. The extension can be either '.png' or '.pdf'. All charts are stored in the 'graphs' folder, under the BETS installation directory.

`start` A `character`. The stating period of the series.

`ylim` A `numeric vector`. Y axis limits.

`open` A `boolean`. Indicates whether the plot must be opened after being created.

Value

An image file is saved in the 'graphs' folder, under the BETS installation directory.

`draw.ulc` *Create a chart of the Unitary Labor Cost time series*

Description

Creates a plot of series 11777

Usage

```
draw.ulc(file, start = NULL, ylim = NULL, open = TRUE)
```

Arguments

`file` A `character`. The name of the file in which the plot must be printed. The extension can be either '.png' or '.pdf'. All charts are stored in the 'graphs' folder, under the BETS installation directory.

`start` A `character`. The stating period of the series.

`ylim` A `numeric vector`. Y axis limits.

`open` A `boolean`. Indicates whether the plot must be opened after being created.

Value

An image file is saved in the 'graphs' folder, under the BETS installation directory.

draw.unemp *Create a chart of the Open Unemployment Rate time series*

Description

Creates a plot of series 10777

Usage

```
draw.unemp(file, start = NULL, ylim = NULL, open = TRUE)
```

Arguments

`file`	A `character`. The name of the file in which the plot must be printed. The extension can be either '.png' or '.pdf'. All charts are stored in the 'graphs' folder, under the BETS installation directory.
`start`	A `character`. The stating period of the series.
`ylim`	A `numeric vector`. Y axis limits.
`open`	A `boolean`. Indicates whether the plot must be opened after being created.

Value

An image file is saved in the 'graphs' folder, under the BETS installation directory.

draw.vargdp *Create a chart of the Real Percentage Change of GDP in the Year time series*

Description

Creates a plot of series 7326

Usage

```
draw.vargdp(file, start = NULL, ylim = NULL, open = TRUE)
```

Arguments

`file`	A `character`. The name of the file in which the plot must be printed. The extension can be either '.png' or '.pdf'. All charts are stored in the 'graphs' folder, under the BETS installation directory.
`start`	A `character`. The stating period of the series.
`ylim`	A `numeric vector`. Y axis limits.
`open`	A `boolean`. Indicates whether the plot must be opened after being created.

Value

An image file is saved in the 'graphs' folder, under the BETS installation directory.

`get.data.frame` *Get a BETS series as a data.frame.*

Description

By default, `BETS.get` returns a `ts` object. However, there are many situations in which is more convenient to work with a data.frame. So, `get.data.frame` receives the code of a BETS series and returns a `data.frame` containing the data of the corresponding series. Alternatively, a `ts` can be supplied, in which case the BETS databases will not be searched.

Usage

```
get.data.frame(code, ts = NULL)
```

Arguments

`code`	An `integer`. The unique identifier of the series within the BETS database.
`ts`	An `ts` object. A time series to be formatted as a data.frame.

Value

A `data.frame`. The first column contains the dates. The second, its values.

`msg` *Format and show a console message.*

Description

Customizes a message and shows it in the console.

Usage

```
msg(..., skip_before = TRUE, skip_after = FALSE)
```

Arguments

`...`	Arguments to be passed to `message`
`skip_before`	A `boolean`. Indicates if a line should be skipped before the message.
`skip_after`	A `boolean`. Indicates if a line should be skipped after the message.

Value

None

Index