Package: rjd3sts (via r-universe)

September 11, 2024

Type Package

Title State Space Framework and Structural Time Series with 'JDemetra+ 3.x'

Version 2.1.1

Description R Interface to 'JDemetra+ 3.x' (<https://github.com/jdemetra>) time series analysis software. It offers access to several functions on state space models and structural time series.

Depends R (>= 4.1.0)

Imports rJava (>= 1.0-6), RProtoBuf (>= 0.4.17), rjd3toolkit (>= 3.2.2), methods

Remotes github::rjdverse/rjd3toolkit@*release

SystemRequirements Java (>= 17)

License EUPL

URL https://github.com/rjdverse/rjd3sts, https://rjdverse.github.io/rjd3sts

LazyData TRUE

Suggests knitr, rmarkdown

RoxygenNote 7.3.1

BugReports https://github.com/rjdverse/rjd3sts

Roxygen list(markdown = TRUE)

Encoding UTF-8

Collate 'utils.R' 'jd3_seasonalbreaks.R' 'jd3_ssf.R' 'jd3_sts.R' 'jd3_stsoutliers.R' 'protobuf.R' 'zzz.R'

VignetteBuilder knitr

Repository https://rjdverse.r-universe.dev

RemoteUrl https://github.com/rjdverse/rjd3sts

RemoteRef v2.1.1

RemoteSha 45ac3a3ff27f8558c07b2ac87d5551e1f40f69b1

Contents

`add` *Title*

Description

Title

Usage

```
add(model, item)
```

Arguments

`item`

`add_equation` *Add a building block to the considered equation*

Description

Add a building block to the considered equation

Usage

```
add_equation(equation, item, coeff = 1, fixed = TRUE, loading = NULL)
```

Arguments

`equation` the equation

`item` the block of the state array that will be linked to the observation corresponding to this equation through the specified loading and coefficient

`coeff` the value of the coefficient associated to the block of latent variables defined by `item`.

`fixed` logical that triggers estimation of coeff (FALSE) or fixes it (TRUE) to a pre-specified value

`loading` the loading that links the block to the observation

aggregation *Title*

Description

Title

Usage

```
aggregation(name, components)
```

Arguments

components

ar *Autoregressive model*

Description

Functions to create an autoregressive model (ar) or a modified autoregressive model (ar2)

Usage

```
ar(
  name,
  ar,
  fixedar = FALSE,
  variance = 0.01,
  fixedvariance = FALSE,
  nlags = 0,
  zeroinit = FALSE
)

ar2(
  name,
  ar,
  fixedar = FALSE,
  variance = 0.01,
  fixedvariance = FALSE,
  nlags = 0,
  nfcasts = 0
)
```

Arguments

`ar` vector of the AR coefficients $(\varphi_1, \dots, \varphi_p)$.

`fixedar` boolean that triggers the estimation of the AR coefficients (`FALSE`) or fixed it (`TRUE`) to a pre-specified value set by the parameter `ar`.

`variance` the variance (σ^2_{ar}).

`fixedvariance` boolean that triggers the estimation of the variance (`FALSE`) or fixed it (`TRUE`) to a pre-specified value set by the parameter `variance`.

`nlags` integer specifying how many lags of the state variable are needed

`zeroinit` boolean determining the initial condition for the state variable, which is equal to zero if `zeroinit` = `TRUE`. The default (`zeroinit` = `FAKSE`) triggers the an initialization based on the unconditional mean and variance of the AR(p) process.

`nfcasts` integer specifying how many forecasts of the state variable are needed

Details

The AR process is defined by

$$\Phi\left(B\right) y_t = \epsilon_t$$

where

$$\Phi\left(B\right) = 1 + \varphi_1 B + \cdots + \varphi_p B^p$$

is an auto-regressive polynomial.

`arima` *Autoregressive Integrated Moving Average (ARIMA) Model*

Description

Autoregressive Integrated Moving Average (ARIMA) Model

Usage

```
arima(name, ar, diff, ma, var = 1, fixed = FALSE)
```

Arguments

`fixed`

arma *Autoregressive Moving Average (ARMA) Model*

Description

Autoregressive Moving Average (ARMA) Model

Usage

```
arma(name, ar, fixedar = FALSE, ma, fixedma = FALSE, var = 1, fixedvar = FALSE)
```

Arguments

`fixedvar`

cumul *Title*

Description

Title

Usage

```
cumul(name, core, period, start = 0)
```

Arguments

`start`

cycle *Title*

Description

Title

Usage

```
cycle(
  name,
  factor = 0.9,
  period = 60,
  fixed = FALSE,
  variance = 0.01,
  fixedvariance = FALSE
)
```

Arguments

fixedvariance

---

equation *Create equation*

Description

Create equation

Usage

```
equation(name, variance = 0, fixed = TRUE)
```

Arguments

fixed

---

estimate *Estimate a SSF Model*

Description

Estimate a SSF Model

Usage

```
estimate(
  model,
  data,
  marginal = FALSE,
  concentrated = TRUE,
  initialization = c("Augmented_Robust", "Diffuse", "SqrtDiffuse", "Augmented",
    "Augmented_NoCollapsing"),
  optimizer = c("LevenbergMarquardt", "MinPack", "BFGS", "LBFGS"),
  precision = 1e-15,
  initialParameters = NULL
)
```

Arguments

`model` the model

`data` a matrix containing the data (one time series per column, time series dimension on the rows)

`marginal` logical value used to specify whether the marginal likelihood definition is used (TRUE) or not (FALSE) during the optimization. The marginal likelihood is recommended when there is at least one variable that loads on a non-stationary latent variable and the loading coefficient needs to be estimated.

`concentrated` logical value used to specify whether the likelihood is concentrated (TRUE) or not (FALSE) during the optimization

`initialization` initialization method.

`precision` indicating the largest likelihood deviations that make the algorithm stop.

`initialParameters`

`filtered_states` *Title*

Description

Title

Usage

```
filtered_states(model)
```

Arguments

`model`

`filtered_states_stdev` *Title*

Description

Title

Usage

```
filtered_states_stdev(model)
```

Arguments

`model`

`filtering_states` *Title*

Description

Title

Usage

```
filtering_states(model)
```

Arguments

`model`

`filtering_states_stdev` *Title*

Description

Title

Usage

```
filtering_states_stdev(model)
```

Arguments

`model`

`loading` *Title*

Description

Title

Title

Usage

```
loading(pos = NULL, weights = NULL)
```

```
loading(pos = NULL, weights = NULL)
```

Arguments

`pos` defines the position of each one of the elements of the block of states defined. `NULL` indicates by default the first state included in the block (pos=0)

`weights` defines the weights associated to each one of the state variables included in the block.

`obs`

`loading_cyclical` *Title*

Description

Title

Usage

```
loading_cyclical(period, startpos)
```

Arguments

`startpos`

`loading_periodic` *Title*

Description

Title

Usage

```
loading_periodic(period, startpos)
```

Arguments

`startpos`

`loading_sum` *Title*

Description

Title

Usage

```
loading_sum(length = 0)
```

Arguments

`length`

`locallevel` *Local Level*

Description

Local Level

Usage

```
locallevel(name, variance = 0.01, fixed = FALSE, initial = NaN)
```

Arguments

`name`	name of the component.
`variance`	the value of the variance (σ_l^2).
`fixed`	boolean that triggers estimation of σ_l^2 (`FALSE`) or fixes it (`TRUE`) to a pre-specified value set by the parameter `variance`.
`initial`	initial value of the level (l_0).

Details

$$\begin{cases} l_{t+1} = l_t + \mu_t \\ \mu_t \sim N(0, \sigma^2 \sigma_l^2) \end{cases}$$

`locallineartrend` *Local Linear Trend*

Description

Local Linear Trend

Usage

```
locallineartrend(
  name,
  levelVariance = 0.01,
  slopevariance = 0.01,
  fixedLevelVariance = FALSE,
  fixedSlopeVariance = FALSE
)
```

Arguments

`name` name of the component.

`levelVariance` variance of the level (σ_l^2)

`fixedLevelVariance`, `fixedSlopeVariance`
boolean that triggers the estimation of the variances σ_l^2 and σ_n^2 (`FALSE`) or fixes it (`TRUE`) to a pre-specified value set by the parameters `levelVariance` and `slopevariance`.

Details

$$\begin{cases} l_{t+1} = l_t + n_t + \xi_t \\ n_{t+1} = n_t + \mu_t \\ \xi_t \sim N(0, \sigma^2 \sigma_l^2) \\ \mu_t \sim N(0, \sigma^2 \sigma_n^2) \end{cases}$$

`model` *Create Composite Model*

Description

Create Composite Model

Usage

```
model()
```

msae *Modeling errors in surveys with overlapping panels*

Description

Modeling errors in surveys with overlapping panels

Usage

```
msae(name, nwaves, ar, fixedar = TRUE, lag = 1)

msae2(name, vars, fixedvars = FALSE, ar, fixedar = TRUE, lag = 1)

msae3(name, vars, fixedvars = FALSE, ar, fixedar = TRUE, k, lag = 1)
```

Arguments

`name`	name of the component.
`nwaves`	integer representing the number of waves
`ar`	matrix representing the covariance structure of the wave specific survey error.
`fixedar`	logical that triggers the estimation of the correlation patterns (`TRUE`) or fixes them to the values given by the entries `ar` (`FALSE`)
`lag`	integer specifying the number of time periods (in the base frequency) that compose the survey period. This coincides with the number of time periods an individual has to wait between two different waves. Note that if the survey period is one quarter, all of them have already responded in the previous wave exactly 3 months ago (because individuals are always interviewed at the same stint during each survey period).

msignal *Title*

Description

Title

Usage

```
msignal(object, m, pos = NULL, stdev = FALSE)
```

Arguments

`stdev`

noise *Noise component*

Description

Noise component

Usage

```
noise(name, variance = 0.01, fixed = FALSE)
```

Arguments

`fixed`

parameters *Get Parameters of SSF Model*

Description

Get Parameters of SSF Model

Usage

```
parameters(model)
```

Arguments

`model`

periodic *Title*

Description

Title

Usage

```
periodic(name, period, harmonics, variance = 0.01, fixedvariance = FALSE)
```

Arguments

`fixedvariance`

`print.JD3STS` *Title*

Description

Title

Usage

```
## S3 method for class 'JD3STS'
print(x, ...)
```

Arguments

`x`	
`...`	

`reg` *Time Varying Regressors*

Description

Time Varying Regressors

Usage

```
reg(name, x, var = NULL, fixed = FALSE)
```

Arguments

`x`	matrix containing the regressors
`fixed`	

reg_td *Title*

Description

Title

Usage

```
reg_td(
  name,
  period,
  start,
  length,
  groups = c(1, 2, 3, 4, 5, 6, 0),
  contrast = TRUE,
  variance = 1,
  fixed = FALSE
)
```

Arguments

`fixed`

sae *Title*

Description

Title

Usage

```
sae(name, ar, fixedar = FALSE, lag = 1, zeroinit = FALSE)
```

Arguments

`zeroinit`

sarima *Title*

Description

Title

Usage

```
sarima(
  name,
  period,
  orders,
  seasonal,
  parameters = NULL,
  fixedparameters = FALSE,
  var = 1,
  fixedvariance = FALSE
)
```

Arguments

`fixedvariance`

seasonal *Title*

Description

Title

Usage

```
seasonal(
  name,
  period,
  type = c("Trigonometric", "Crude", "HarrisonStevens", "Dummy"),
  variance = 0.01,
  fixed = FALSE
)
```

Arguments

`fixed`

seasonalbreaks *Title*

Description

Title

Usage

```
seasonalbreaks(
  y,
  period = NA,
  level = 1,
  slope = 1,
  noise = 1,
 seasonal = c("HarrisonStevens", "Trigonometric", "Dummy", "Crude", "Fixed", "Unused"),
  X = NULL,
  X.td = NULL
)
```

Arguments

`y` input time series.

`period` annual frequency.

`level` -1 = no level, 0 = fixed level, 1 = sotchastic level

`slope`

`noise`

`seasonal` Seasonal model

`X` Regression variables (same length as y) or NULL

`X.td` Groups of days for trading days regressors. The length of the array must be 7. It indicates to what group each week day belongs. The first item corresponds to Mondays and the last one to Sundays. The group used for contrasts (usually Sundays) is identified by 0. The other groups are identified by 1, 2,... n (<= 6). For instance, usual trading days are defined by `c(1,2,3,4,5,6,0)`, week days by `c(1,1,1,1,1,0,0)`, etc...

Examples

```
x<-rjd3toolkit::retail$BookStores
seasonalbreaks(x)
```

signal *Title*

Description

Title

Usage

```
signal(object, obs = 1, pos = NULL, loading = NULL, stdev = FALSE)
```

Arguments

`stdev`

smoothed_components *Retrieves the components of the model (univariate case) or the components corresponding to a given equation (multivariate case)*

Description

Retrieves the components of the model (univariate case) or the components corresponding to a given equation (multivariate case)

Usage

```
smoothed_components(model, equation = 1, fast = TRUE)
```

Arguments

`model`	Estimated state space model
`equation`	Equation containing the components
`fast`	if true, only the components are computed. Otherwise, their stdev are also computed (not returned but available for future use).

Value

A matrix with the components

`smoothed_components_stdev` *Retrieves the stdev of the components of the model (univariate case) or of the components corresponding to a given equation (multivariate case)*

Description

Retrieves the stdev of the components of the model (univariate case) or of the components corresponding to a given equation (multivariate case)

Usage

```
smoothed_components_stdev(model, equation = 1)
```

Arguments

`model`	Estimated state space model
`equation`	Equation containing the components

Value

A matrix with the stdev of the components

`smoothed_states` *Title*

Description

Title

Usage

```
smoothed_states(model)
```

Arguments

`model`

smoothed_states_stdev *Title*

Description

Title

Usage

```
smoothed_states_stdev(model)
```

Arguments

model

splines_daily *Title*

Description

Title

Usage

```
splines_daily(name, startYear, nodes, start = 1, variance = 1, fixed = FALSE)
```

Arguments

fixed

splines_regular *Title*

Description

Title

Usage

```
splines_regular(
  name,
  period,
  nnodes = 0,
  nodes = NULL,
  start = 1,
  variance = 1,
  fixed = FALSE
)
```

Arguments

`fixed`

`ssf` *Title*

Description

Title

Usage

```
ssf(initialization, dynamics, measurement)
```

Arguments

`measurement`

`sts` *Title*

Description

Title

Usage

```
sts(
  y,
  X = NULL,
  X.td = NULL,
  level = 1,
  slope = 1,
  cycle = -1,
  noise = 1,
 seasonal = c("Trigonometric", "Dummy", "Crude", "HarrisonStevens", "Fixed", "Unused"),
  diffuse.regs = TRUE,
  tol = 1e-09
)
```

Arguments

`y` input time series.

`X` Regression variables (same length as y) or NULL

`X.td` Groups of days for trading days regressors. The length of the array must be 7. It indicates to what group each week day belongs. The first item corresponds to Mondays and the last one to Sundays. The group used for contrasts (usually Sundays) is identified by 0. The other groups are identified by 1, 2,... n (<= 6). For instance, usual trading days are defined by `c(1,2,3,4,5,6,0)`, week days by `c(1,1,1,1,1,0,0)`, etc...

`level` -1 = no level, 0 = fixed level, 1 = sotchastic level

`slope`

`cycle`

`noise`

`seasonal` Seasonal model

`diffuse.regs`

`tol`

Examples

```
x<-rjd3toolkit::retail$BookStores
sts(x)
```

`sts_forecast` *Forecast with STS model*

Description

Forecast with STS model

Usage

```
sts_forecast(y, model = c("none", "td2", "td3", "td7", "full"), nf = 12)
```

Arguments

`y` Series

`model` Model for calendar effects

- td2: leap year + week days (week-end derived)
- td3: leap year + week days + saturdays (sundays derived)
- td7: leap year + all days (sundays derived)
- full: td3 + easter effect
- none: no calendar effect

`nf` number of forecasts

Examples

```
fcasts<-sts_forecast(rjd3toolkit::ABS$X0.2.09.10.M)
```

---

sts_outliers *Title*

Description

Title

Usage

```
sts_outliers(
  y,
  period = NA,
  X = NULL,
  X.td = NULL,
  level = 1,
  slope = 1,
  noise = 1,
 seasonal = c("Trigonometric", "Dummy", "Crude", "HarrisonStevens", "Fixed", "Unused"),
  ao = TRUE,
  ls = TRUE,
  so = FALSE,
  cv = 0,
  tcv = 0,
  estimation.forward = c("Score", "Point", "Full"),
  estimation.backward = c("Point", "Score", "Full")
)
```

Arguments

`y` input time series.

`period` annual frequency.

`X` Regression variables (same length as y) or NULL

`X.td` Groups of days for trading days regressors. The length of the array must be 7. It indicates to what group each week day belongs. The first item corresponds to Mondays and the last one to Sundays. The group used for contrasts (usually Sundays) is identified by 0. The other groups are identified by 1, 2,... n (<= 6). For instance, usual trading days are defined by `c(1,2,3,4,5,6,0)`, week days by `c(1,1,1,1,1,0,0)`, etc...

`level` -1 = no level, 0 = fixed level, 1 = sotchastic level

`slope`

`noise`

`seasonal` Seasonal model

`ao, ls, so` boolean indicating if additive outliers (`ao`), level shift (`ls`) and seasonal outliers (`so`) should be detected.

`cv`

`tcv`

`estimation.forward`

`estimation.backward`

Examples

```
x<-rjd3toolkit::retail$BookStores
sts_outliers(x)
```

`sts_raw` *Title*

Description

Title

Usage

```
sts_raw(
  y,
  period = NA,
  X = NULL,
  X.td = NULL,
  level = 1,
  slope = 1,
  cycle = -1,
  noise = 1,
 seasonal = c("Trigonometric", "Dummy", "Crude", "HarrisonStevens", "Fixed", "Unused"),
  diffuse.regs = TRUE,
  tol = 1e-09
)
```

Arguments

y

period

X

X.td

level

slope

cycle

noise

seasonal

diffuse.regs

tol

---

var_loading *Title*

Description

Title

Usage

```
var_loading(pos, weights)
```

Arguments

weights

`var_locallevel` *Title*

Description

Title

Usage

```
var_locallevel(name, std, scale = 1, fixed = FALSE, initial = NaN)
```

Arguments

`initial`

`var_locallineartrend` *Title*

Description

Title

Usage

```
var_locallineartrend(
  name,
  lstd,
  sstd = NULL,
  levelScale = 1,
  slopeScale = 1,
  fixedLevelScale = FALSE,
  fixedSlopeScale = FALSE
)
```

Arguments

`fixedSlopeScale`

var_noise *Title*

Description

Title

Usage

```
var_noise(name, std, scale = 1, fixed = FALSE)
```

Arguments

fixed

var_reg *Time Varying Regressor*

Description

Time Varying Regressor

Usage

```
var_reg(name, x, stderr, scale = 1, fixed = FALSE)
```

Arguments

x	Regression variable. Numerics
stderr	Standard error of the innovations of the coefficient (1 in extrapolation)
scale	Scaling factor
fixed	Fixed scaling factor

Examples

```
x<-rjd3toolkit::retail$BookStores
std<-rep(1, length(x))
std[c(20, 50, 150)]<-5
v<-var_reg(”vx”, x, std, 0.1)
```

`var_seasonal` *Title*

Description

Title

Usage

```
var_seasonal(
  name,
  period,
  type = c("Trigonometric", "Crude", "HarrisonStevens", "Dummy"),
  std,
  scale = 1,
  fixed = FALSE
)
```

Arguments

`fixed`

Index