Package: rjd3sts (via r-universe)

September 8, 2026

Type Package

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

Version 2.4.1.9000

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 (>= 0.4.20), rjd3toolkit (>= 3.7.1), rjd3xjars (>= 0.1.0), methods, stats

Remotes github::rjdverse/rjd3toolkit, github::rjdverse/rjd3xjars,

SystemRequirements Java (>= 21)

License file LICENSE

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

LazyData TRUE

Suggests knitr, rmarkdown

RoxygenNote 7.3.3

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

Roxygen list(markdown = TRUE)

Encoding UTF-8

Collate 'utils.R' 'jd3_rawssf.R' 'jd3_rawssf_algorithms.R' 'jd3_rawssf_loadings.R' 'jd3_rawssf_models.R' 'jd3_rawssf_states.R' 'jd3_seasonalbreaks.R' 'jd3_ssf.R' 'jd3_sts.R' 'jd3_stsoutliers.R' 'protobuf.R' 'tdarima.R' 'zzz.R'

VignetteBuilder knitr

Config/pak/sysreqs make default-jdk libprotobuf-dev protobuf-compiler libprotoc-dev

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

Date/Publication 2026-05-08 07:48:56 UTC

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

RemoteRef HEAD

RemoteSha d2096a8704bf273499a083409a91f4905be17c3f

Contents

.airline . . . 4
.akf_likelihood . . . 5
.arima . . . 5
.arima2 . . . 6
.bsm2spec . . . 7
.bsm2ucm . . . 7
.circular_loading . . . 8
.ckms_likelihood . . . 8
.composite . . . 8
.cyclical_loading . . . 9
.dk_likelihood . . . 9
.loading . . . 10
.loading_composite . . . 10
.local_level . . . 11
.local_linear_trend . . . 11
.mssf . . . 12
.mssf_measurements . . . 12
.noise . . . 13
.proc_diffuselikelihood . . . 13
.r2jd_bsm . . . 14
.sarima . . . 14
.sarma . . . 15
.sarma2 . . . 15
.seasonal . . . 16
.ssf . . . 17
.ssf_as_time_invariant . . . 17
.ssf_B . . . 18
.ssf_P0 . . . 18
.ssf_S . . . 18
.ssf_smooth . . . 19
.ssf_T . . . 19
.ssf_V . . . 20
.ssf_Z . . . 20
.state_diffuse_dim . . . 21
.state_dim . . . 21
.state_of . . . 22
add . . . 22
add_equation . . . 23

aggregation . . . 23
ar . . . 24
arima . . . 25
arma . . . 25
block_d0 . . . 26
block_p0 . . . 26
block_t . . . 26
block_v . . . 27
bsm_model . . . 27
bsm_to_ucm . . . 28
components_pos . . . 28
cumul . . . 29
cycle . . . 29
equation . . . 30
estimate . . . 30
filtered_states . . . 31
filtered_states_stdev . . . 31
filtering_states . . . 32
filtering_states_stdev . . . 32
loading . . . 32
loading_cyclical . . . 33
loading_periodic . . . 33
loading_sum . . . 34
loadings . . . 34
locallevel . . . 35
locallineartrend . . . 36
ltd_airline . . . 37
model . . . 37
msae . . . 38
msignal . . . 38
noise . . . 39
parameters . . . 39
periodic . . . 40
print.JD3STS . . . 40
reg . . . 40
reg_td . . . 41
sae . . . 41
sarima . . . 42
seasonal . . . 42
seasonalbreaks . . . 43
signal . . . 44
smoothed_components . . . 44
smoothed_components_stdev . . . 45
smoothed_states . . . 46
smoothed_states_stdev . . . 46
splines_daily . . . 47
splines_generic . . . 47
splines_regular . . . 48

ssf 48
sts 49
sts_forecast 50
sts_outliers 50
sts_raw 51
tdairline_estimation 52
var_ar 52
var_loading 53
var_locallevel 53
var_locallineartrend 54
var_noise 54
var_reg 55
var_seasonal 55

Index **56**

`.airline` *Creates the state space form of an airline model;*

Description

Creates the state space form of an airline model;

Usage

```
.airline(period = 12, theta = -0.6, btheta = -0.8)
```

Arguments

`period`	Period of the model.
`theta`	Regular moving average parameter.
`btheta`	Seasonal moving average parameter.

Value

The raw state space form of the Airline model.

Examples

```
ssf<-.airline(24)
.dk_likelihood(ssf, rjd3toolkit::ABS$X0.2.09.10.M)
```

.akf_likelihood — *Computes the diffuse likelihood by means of the augmented Kalman filter.*

Description

Computes the diffuse likelihood by means of the augmented Kalman filter.

Usage

```
.akf_likelihood(
  ssf,
  data,
 qtype = c("NORMAL", "PARTIAL_TRIANGULARIZATION", "FULL_TRIANGULARIZATION", "QR"),
  collapsing = TRUE,
  rescalingFactor = TRUE
)
```

Arguments

ssf	A state space form.
data	Arrays of data.
qtype	Type of the initialization in the augmented Kalman filter
rescalingFactor	True if the innovation covariance matrix is defined up to a scaling factor.

Value

The diffuse likelihood

Examples

```
sarima<-.sarima(12, NULL, 1,  -.6, NULL, 1, -.5)
ssf<-.ssf(sarima, .loading(0))
.akf_likelihood(ssf, rjd3toolkit::ABS$X0.2.09.10.M)
```

.arima — *Creates an ARIMA state block (representation I)*

Description

Creates an ARIMA state block (representation I)

Usage

```
.arima(ar, delta, ma, var = 1)
```

Arguments

ar	Stationary auto-regressive polynomial, including the constant (=1). True signs.
delta	Non-stationary auto-regressive polynomial, including the constant (=1). True signs.
ma	Moving average polynomial, including the constant (=1). True signs.
var	Variance of the innovations

Value

A raw java state block.

Examples

```
sb<-.arima(c(1, -.5), NULL, c(1,-.8))
.ssf_P0(sb)
```

.arima2 *Creates an ARIMA state block (representation II)*

Description

Creates an ARIMA state block (representation II)

Usage

```
.arima2(ar, delta, ma, var = 1)
```

Arguments

ar	Stationary auto-regressive polynomial, including the constant (=1). True signs.
delta	Non-stationary auto-regressive polynomial, including the constant (=1). True signs.
ma	Moving average polynomial, including the constant (=1). True signs.
var	Variance of the innovations

Value

A raw java state block

Examples

```
sb<-.arima2(c(1, -.5), NULL, c(1,-.8))
.ssf_P0(sb)
```

.bsm2spec *Title*

Description

Title

Usage

```
.bsm2spec(jbsm, fixed = FALSE, fixedCycle = TRUE)
```

Arguments

fixedCycle

Examples

```
bsm<-bsm_model(12)
jbsm<-.r2jd_bsm(bsm)
jspec<-.bsm2spec(jbsm)
```

.bsm2ucm *Title*

Description

Title

Usage

```
.bsm2ucm(jbsm, normalized = TRUE)
```

Arguments

normalized

Examples

```
bsm<-bsm_model(12)
jbsm<-.r2jd_bsm(bsm)
jucm<-.bsm2ucm(jbsm)
```

.circular_loading *Title*

Description

Title

Usage

```
.circular_loading(period, start = 0)
```

Arguments

start

.ckms_likelihood *Title*

Description

Title

Usage

```
.ckms_likelihood(ssf, data, rescalingFactor = TRUE)
```

Arguments

rescalingFactor

.composite *Title*

Description

Title

Usage

```
.composite(cmps)
```

Arguments

cmps

.cyclical_loading *Title*

Description

Title

Usage

```
.cyclical_loading(period, start = 0)
```

Arguments

start

.dk_likelihood *Computes the diffuse likelihood by means of the diffuse Kalman filter (Durbin-Koopman).*

Description

Computes the diffuse likelihood by means of the diffuse Kalman filter (Durbin-Koopman).

Usage

```
.dk_likelihood(ssf, data, sqr = TRUE, rescalingFactor = TRUE)
```

Arguments

ssf	A state space form.
data	Arrays of data.
sqr	Square root diffuse initialization, if any
rescalingFactor	True if the innovation covariance matrix is defined up to a scaling factor.

Value

The diffuse likelihood

Examples

```
sarima<-.sarima(12, NULL, 1,  -.6, NULL, 1, -.5)
ssf<-.ssf(sarima, .loading(0))
.dk_likelihood(ssf, rjd3toolkit::ABS$X0.2.09.10.M)
```

.loading *Title*

Description

Title

Usage

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

Arguments

weights

.loading_composite *Creates a composite loading*

Description

Creates a composite loading

Usage

```
.loading_composite(dims, cmps)
```

Arguments

dims	The dimensions of the corresponding state blocks
cmps	The loadings

Examples

```
l<-.loading_composite(c(1,2,3), list(.loading(0), .loading(0), .loading(0)))
.ssf_Z(l, 6, 0)
```

.local_level — *Creates a local level state block*

Description

Creates a local level state block

Usage

```
.local_level(var = 1, start = NaN)
```

Arguments

var	Innovation variance of local level
start	Initial value of the state block. Should be NaN for a diffuse initialization

Value

A wrapper around the java object (class JD3_RawStateBlock).

Examples

```
sb<-.local_level(1.5, 0)
.ssf_T(sb, 0)
```

.local_linear_trend — *Creates a local linear trend state block.*

Description

Creates a local linear trend state block.

Usage

```
.local_linear_trend(lvar, svar = 0)
```

Arguments

lvar	Innovation variance of the level equation.
svar	Innovation variance of the slope equation.

Value

A wrapper around the java object (class JD3_RawStateBlock).

Examples

```
sb<-.local_linear_trend(1.5, 0.5)
.ssf_T(sb, 0)
```

.mssf *Title*

Description

Title

Usage

```
.mssf(component, measurements)
```

Arguments

measurements

.mssf_measurements *Title*

Description

Title

Usage

```
.mssf_measurements(Z, H)
```

Arguments

H

.noise *Creates a white noise.*

Description

Creates a white noise.

Usage

```
.noise(var = 1)
```

Arguments

var	Variance of the noise.

Value

A wrapper around the java object (class JD3_RawStateBlock).

Examples

```
sb<-.noise(.01)
.ssf_T(sb, 0)
```

.proc_diffuselikelihood *Title*

Description

Title

Usage

```
.proc_diffuselikelihood(jrslt, prefix)
```

Arguments

prefix

.r2jd_bsm *Title*

Description

Title

Usage

```
.r2jd_bsm(bsm)
```

Arguments

bsm

.sarima *Creates an ARMA state block*

Description

Creates an ARMA state block

Usage

```
.sarima(period, phi, d = 1, theta, bphi, bd = 1, btheta)
```

Arguments

period	Period of the seasonality
phi	Regular stationary auto-regressive polynomial. True signs, without the constant
d	Regular differencing order
theta	Regular moving average polynomial. True signs, without the constant
bphi	Seasonal stationary auto-regressive polynomial. True signs, without the constant
bd	Seasonal differencing order
btheta	Seasonal moving average polynomial. True signs, without the constant

Value

A raw java state block

Examples

```
sb<-.sarima(12, c(0.5, -.2, .1), 1, -.8, NULL, 1, -.6)
.ssf_P0(sb)
```

.sarma — *Creates an ARMA state block (representation I)*

Description

Creates an ARMA state block (representation I)

Usage

```
.sarma(period, phi, theta, bphi, btheta)
```

Arguments

period	Period of the seasonality
phi	Regular stationary auto-regressive polynomial. True signs, without the constant
theta	Regular moving average polynomial. True signs, without the constant
bphi	Seasonal stationary auto-regressive polynomial. True signs, without the constant
btheta	Seasonal moving average polynomial. True signs, without the constant

Value

A raw java state block

Examples

```
sb<-.sarma(12, c(0.5, -.2, .1), -.8, NULL, -.6)
.ssf_P0(sb)
```

.sarma2 — *Creates an ARMA state block (representation II)*

Description

Creates an ARMA state block (representation II)

Usage

```
.sarma2(period, phi, theta, bphi, btheta)
```

Arguments

period	Period of the seasonality
phi	Regular stationary auto-regressive polynomial. True signs, without the constant
theta	Regular moving average polynomial. True signs, without the constant
bphi	Seasonal stationary auto-regressive polynomial. True signs, without the constant
btheta	Seasonal moving average polynomial. True signs, without the constant

Value

A raw java state block

Examples

```
sb<-.sarma2(12, c(0.5, -.2, .1), -.8, NULL, -.6)
.ssf_P0(sb)
```

.seasonal — *Creates a seasonal component, corresponding to a multivariate random walk, with an aggregation constraint to 0 and various covariances for the innovations of the transition equation.*

Description

Creates a seasonal component, corresponding to a multivariate random walk, with an aggregation constraint to 0 and various covariances for the innovations of the transition equation.

Usage

```
.seasonal(
  period,
  type = c("Trigonometric", "Crude", "HarrisonStevens", "Dummy"),
  var = 1
)
```

Arguments

period	Period of the seasonality
type	Type of the innovations of the transition equation
var	Variance of the innovations

Value

A wrapper around the java object (class JD3_RawStateBlock)

Examples

```
sb<-.seasonal(4, "HarrisonStevens", .01)
.ssf_V(sb, 0)
```

.ssf *Title*

Description

Title

Usage

```
.ssf(component, loading, evar = 0)
```

Arguments

evar

.ssf_as_time_invariant *Transforms a time invariant state space form based on functions into a state space models represented by matrices.*

Description

Transforms a time invariant state space form based on functions into a state space models represented by matrices.

Usage

```
.ssf_as_time_invariant(jssf)
```

Arguments

jssf	The object oriented (java) state space form, which should be time invariant

Value

A new Java object based on matrices

Examples

```
ll<-.local_linear_trend(0.1, 0.1)
s<-.seasonal(12, var=.5)
m<-.composite(list(ll, s))
ssf1<-.ssf(m, .loading(c(0,2)), 1)
ssf2<-.ssf_as_time_invariant(ssf1)
ll1<-.akf_likelihood(ssf1, rjd3toolkit::ABS$X0.2.09.10.M)
ll2<-.akf_likelihood(ssf2, rjd3toolkit::ABS$X0.2.09.10.M)
print(ll1$ll-ll2$ll)
```

.ssf_B *Title*

Description

Title

Usage

```
.ssf_B(x)
```

Arguments

x

.ssf_P0 *Title*

Description

Title

Usage

```
.ssf_P0(x)
```

Arguments

x

.ssf_S *Title*

Description

Title

Usage

```
.ssf_S(x, pos = 0)
```

Arguments

pos

.ssf_smooth — *Computes smoothed states by means of the augmented Kalman filter in the case of diffuse initialization*

Description

Computes smoothed states by means of the augmented Kalman filter in the case of diffuse initialization

Usage

```
.ssf_smooth(
  ssf,
  data,
  all = TRUE,
 qtype = c("NORMAL", "PARTIAL_TRIANGULARIZATION", "FULL_TRIANGULARIZATION", "QR")
)
```

Arguments

ssf	A state space form.
data	Arrays of data.
all	True if the covariances of the smoothed states are compted.
qtype	Type of the initialization in the augmented Kalman filter

Value

A matrix with the smoothed states and - if requested - their standard deviations.

.ssf_T — *Gets the transition matrix.*

Description

Gets the transition matrix.

Usage

```
.ssf_T(x, pos = 0)
```

Arguments

x	A state block or a state space form.
pos	The 0-based position (use 0 for time invariant models)

Value

The requested matrix.

Examples

```
sarima<-.sarma(12, .9, -.6, .9, -.5)
.ssf_T(sarima)
```

.ssf_V — *Gets the covariance of the innovations in the transition equation.*

Description

Gets the covariance of the innovations in the transition equation.

Usage

```
.ssf_V(x, pos = 0)
```

Arguments

x	A state block or a state space form.
pos	The 0-based position (use 0 for time invariant models)

Value

The requested matrix.

Examples

```
sarima<-.sarma(12, .9, -.6, .9, -.5)
.ssf_V(sarima)
```

.ssf_Z — *Gets the loading vector. It should have the same length as the corresponding state block.*

Description

Gets the loading vector. It should have the same length as the corresponding state block.

Usage

```
.ssf_Z(x, n, pos)
```

Arguments

x	Loading
n	the Length of the corresponding state block (on which the loading will be applied)
pos	0-based position of the filter

Value

the loading vector

.state_diffuse_dim *Title*

Description

Title

Usage

```
.state_diffuse_dim(x)
```

Arguments

x

.state_dim *Retrieves the dimension of a state block*

Description

Retrieves the dimension of a state block

Usage

```
.state_dim(x)
```

Arguments

x	A state block

Value

The length of the state block

Examples

```
s<-.seasonal(12)
.state_dim(s)
```

.state_of — *Gets the state of the state space form*

Description

Gets the state of the state space form

Usage

```
.state_of(ssf)
```

Arguments

ssf	A state space form

Value

A state block

add — *Adds a state block or a measurement equation to a given state space model*

Description

Adds a state block or a measurement equation to a given state space model

Usage

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

Arguments

model	A state space model
item	A state block or a measurement equation

Examples

```
model<-model()
llt<-locallineartrend("llt")
add(model,llt)
```

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 observations

Examples

```
eq<- equation('eq1')
ll<-locallevel('ll')
n<-noise("n", variance = 1, fixed = TRUE)
add_equation(eq, ll)
add_equation(eq, n, coeff=0.1, fixed=FALSE)
```

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 (See @details Additional details...).
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

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

The stability of the auto-regressive polynomial is not checked.

Value

An element of type "JD3_SsfStateBlock" (wrapper around the corresponding Java object))

Examples

```
b_ar<-ar("my_ar", c(.8,-.3,.2), variance=1)
block_p0(b_ar)
```

---

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`

block_d0 *Title*

Description

Title

Usage

```
block_d0(block, pos = 0)
```

Arguments

pos

block_p0 *Title*

Description

Title

Usage

```
block_p0(block)
```

Arguments

block

block_t *Title*

Description

Title

Usage

```
block_t(block, pos = 0)
```

Arguments

pos

block_v *Title*

Description

Title

Usage

```
block_v(block, pos = 0)
```

Arguments

block

bsm_model *Title*

Description

Title

Usage

```
bsm_model(
  period,
  noise.var = 1,
  level.var = 1,
  slope.var = NULL,
  seasonal.var = 1,
 seasonal.model = c("HarrisonStevens", "Trigonometric", "Dummy", "Crude", "Fixed"),
  cycle.var = NULL,
  cycle.dumpingfactor = 0.9,
  cycle.length = 6
)
```

Arguments

cycle.length

bsm_to_ucm *Title*

Description

Title

Usage

```
bsm_to_ucm(bsm, normalized = TRUE)
```

Arguments

normalized

components_pos *Position of the components*

Description

Position of the components

Usage

```
components_pos(model)
```

Arguments

model	Estimated model

Value

The first position of the different blocks in the state array

Examples

```
model<-model()
ll<-locallevel("ll")
seas<-seasonal("seas", 12, "Crude")
n<-noise("n")
add(model,ll)
add(model,seas)
add(model,n)
emodel<-estimate(model, rjd3toolkit::Retail$BookStores)
print(components_pos(emodel))
```

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

name	Name of the block
variance	Variance of the innovations
fixedvariance	Indicates if the variance if fixed

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

Usage

```
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.

`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

loadings *Give all the loadings for a given variable*

Description

Give all the loadings for a given variable

Usage

```
loadings(object, var = 1)
```

Arguments

object	Estimated model
var	loadings for the given variable (or equation). Useless in the case of univariate models

Value

A matrix with the requested loadings. The number of rows corresponds to the number of observations.

Examples

```
model<-model()
llt<-locallineartrend("llt")
seas<-seasonal("seas", 12, "Crude")
n<-noise("n")
add(model,llt)
add(model,seas)
add(model,n)
y<-rjd3toolkit::Retail$BookStores
emodel<-estimate(model, y)
print(loadings(emodel))
```

`locallevel` *Local Level*

Description

Local Level

Usage

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

Arguments

`name`	Name of the block
`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

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

Examples

```
ll<-locallevel('ll', variance=1)
print(block_t(ll))
```

`locallineartrend` *Local linear trend state block*

Description

Local linear trend state block

Usage

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

Arguments

`name`	Name of the block
`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

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

Examples

```
llt<-locallineartrend('llt', levelVariance=1, slopeVariance=.25)
print(block_t(llt))
```

ltd_airline *Title*

Description

Title

Usage

```
ltd_airline(
  name,
  length,
  period,
  th0 = -0.6,
  th1 = -0.6,
  bth0 = -0.6,
  bth1 = -0.6,
  fixedth = FALSE,
  variance = 0.01,
  fixedvariance = FALSE
)
```

Arguments

fixedvariance

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 block
`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

`object`	Estimated model
`stdev`	

noise — *Noise state block*

Description

Noise state block

Usage

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

Arguments

name	Name of the block
variance	Variance of the noise
fixed	Indicates if the variance is fixed

Examples

```
n<-noise("n", 1)
block_t(n)
```

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

name	Name of the block
fixedvariance	

print.JD3STS *Title*

Description

Title

Usage

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

Arguments

...	

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

`name`	Name of the block
`zeroinit`	

sarima *Title*

Description

Title

Usage

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

Arguments

fixedvariance

seasonal *Seasonal state block*

Description

Seasonal state block

Usage

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

Arguments

name	Name of the block
period	Period of the seasonality
type	Type of the seasonal component
variance	Innovation variance
fixed	Indicates if the variance is fixed

Examples

```
seas1<-seasonal("seas1", 12, "HarrisonStevens", variance=1)
print(block_v(seas1))
seas2<-seasonal("seas2", 12, "Trigonometric", variance=1)
print(block_v(seas2))
```

---

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
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

Examples

```
model<-model()
llt<-locallineartrend("llt")
seas<-seasonal("seas", 12, "Crude")
n<-noise("n")
add(model,llt)
add(model,seas)
add(model,n)
y<-rjd3toolkit::Retail$BookStores
emodel<-estimate(model, y)
scmp<-smoothed_components(emodel)
high<-cbind(scmp[,1], y-scmp[,2])
low<-scmp[,c(2,3)]
matplot(high, type='l')
matplot(low, type='l')
```

`smoothed_components_stdev`

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

Description

Retrieves the standard deviations 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 *Standard deviations of the smoothed states*

Description

Standard deviations of the smoothed states

Usage

```
smoothed_states_stdev(model)
```

Arguments

model	Estimated model

Value

A matrix with the standard deviations of the states (a row corresponding to one time point)

Examples

```
model<-model()
ll<-locallevel("ll")
seas<-seasonal("seas", 12, "Crude")
n<-noise("n")
add(model,ll)
add(model,seas)
add(model,n)
emodel<-estimate(model, rjd3toolkit::Retail$BookStores)
ess<-smoothed_states_stdev(emodel)
cmps<-ess[,c(1,2,13)]
matplot(cmps, type='l')
```

splines_daily *Title*

Description

Title

Usage

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

Arguments

fixed

splines_generic *Title*

Description

Title

Usage

```
splines_generic(
  name,
  period,
  knots,
  order = 4,
  start = 1,
  variance = 1,
  fixed = FALSE
)
```

Arguments

fixed

splines_regular *Title*

Description

Title

Usage

```
splines_regular(
  name,
  period,
  nknots = 0,
  knots = 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
`seasonal`	Seasonal model
`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
seasonal	Seasonal model
ao, ls, so	boolean indicating if additive outliers (`ao`), level shift (`ls`) and seasonal outliers (`so`) should be detected.
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

tol

---

tdairline_estimation *Title*

Description

Title

Usage

```
tdairline_estimation(s, td = NULL, vartd = FALSE, precision = 1e-09)
```

Arguments

precision

---

var_ar *Title*

Description

Title

Usage

```
var_ar(
  name,
  ar,
  fixedar = FALSE,
  stderr,
  scale = 1,
  fixed = FALSE,
  zeroinit = FALSE
)
```

Arguments

`name` Name of the block

`zeroinit`

---

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

.airline, 4
.akf_likelihood, 5
.arima, 5
.arima2, 6
.bsm2spec, 7
.bsm2ucm, 7
.circular_loading, 8
.ckms_likelihood, 8
.composite, 8
.cyclical_loading, 9
.dk_likelihood, 9
.loading, 10
.loading_composite, 10
.local_level, 11
.local_linear_trend, 11
.mssf, 12
.mssf_measurements, 12
.noise, 13
.proc_diffuselikelihood, 13
.r2jd_bsm, 14
.sarima, 14
.sarma, 15
.sarma2, 15
.seasonal, 16
.ssf, 17
.ssf_B, 18
.ssf_P0, 18
.ssf_S, 18
.ssf_T, 19
.ssf_V, 20
.ssf_Z, 20
.ssf_as_time_invariant, 17
.ssf_smooth, 19
.state_diffuse_dim, 21
.state_dim, 21
.state_of, 22

add, 22
add_equation, 23
aggregation, 23
ar, 24
ar2 *(ar)*, 24
arima, 25
arma, 25

block_d0, 26
block_p0, 26
block_t, 26
block_v, 27
bsm_model, 27
bsm_to_ucm, 28

components_pos, 28
cumul, 29
cycle, 29

equation, 30
estimate, 30

filtered_states, 31
filtered_states_stdev, 31
filtering_states, 32
filtering_states_stdev, 32

loading, 32
loading_cyclical, 33
loading_periodic, 33
loading_sum, 34
loadings, 34
locallevel, 35
locallineartrend, 36
ltd_airline, 37

model, 37
msae, 38
msae2 *(msae)*, 38
msae3 *(msae)*, 38
msignal, 38

noise, 39

parameters, 39
periodic, 40
print.JD3STS, 40

reg, 40
reg_td, 41

sae, 41
sarima, 42
seasonal, 42
seasonalbreaks, 43
signal, 44
smoothed_components, 44
smoothed_components_stdev, 45
smoothed_states, 46
smoothed_states_stdev, 46
splines_daily, 47
splines_generic, 47
splines_regular, 48
ssf, 48
sts, 49
sts_forecast, 50
sts_outliers, 50
sts_raw, 51

tdairline_estimation, 52

var_ar, 52
var_loading, 53
var_locallevel, 53
var_locallineartrend, 54
var_noise, 54
var_reg, 55
var_seasonal, 55