Package: autostsm (via r-universe)

September 4, 2024

Type Package

Title Automatic Structural Time Series Models

Version 3.1.5

Date 2024-06-01

Description Automatic model selection for structural time series decomposition into trend, cycle, and seasonal components, plus optionality for structural interpolation, using the Kalman filter. Koopman, Siem Jan and Marius Ooms (2012) ``Forecasting Economic Time Series Using Unobserved Components Time Series Models" <doi:10.1093/oxfordhb/9780195398649.013.0006>. Kim, Chang-Jin and Charles R. Nelson (1999) ``State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications" <http://econ.korea.ac.kr/ cjkim/doi:10.7551/mitpress/6444.001.0001><http://econ.korea.ac.kr/~{}cjkim/>.

License GPL (>= 2)

Imports maxLik (>= 1.5-2), forecast (>= 8.15), lubridate (>= 1.7), ggplot2 (>= 3.3), gridExtra (>= 2.3), strucchange (>= 1.5), foreach (>= 1.5), doSNOW (>= 1.0.19), parallel (>= 4.1.1), lmtest (>= 0.9-38), ggrepel(>= 0.9), progress (>= 1.2), sandwich (>= 3.0), data.table (>= 1.15), kalmanfilter (>= 2.0.1)

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Alex Hubbard [aut, cre]

Maintainer Alex Hubbard <hubbard.alex@gmail.com>

Depends R (>= 3.5.0)

Date/Publication 2024-06-05 21:40:41 UTC

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

RemoteUrl https://github.com/cran/autostsm

RemoteRef HEAD

RemoteSha e875a3b77de4371968e9cf68fcbd859f784ab3ba

Contents

DGS5 *5 Year Treasury Yield*

Description

5 Year Treasury Yield

Usage

```
data(DGS5)
```

Format

data.table with columns DATE and DGS5, monthly frequency

Source

FRED

GDP *US GDP Seasonally Adjusted*

Description

US GDP Seasonally Adjusted

Usage

```
data(GDP)
```

Format

data.table with columns DATE and GDP, quarterly frequency

Source

FRED

NA000334Q *US GDP Not Seasonally Adjusted*

Description

US GDP Not Seasonally Adjusted

Usage

```
data(NA000334Q)
```

Format

data.table with columns DATE and NA000334Q, quarterly frequency

Source

FRED

SP500 *S&P 500*

Description

S&P 500

Usage

```
data(SP500)
```

Format

data.table with columns DATE and SP500, daily frequency

Source

FRED

stsm_bdiag *Build a block diagonal matrix from two matrices*

Description

Build a block diagonal matrix from two matrices

Usage

```
stsm_bdiag(A, B)
```

Arguments

A	The top left matrix
B	The bottom right matrix

Value

A block diagonal matrix

stsm_build_dates *Build the date sequence as a Date type*

Description

Build the date sequence as a Date type

Usage

```
stsm_build_dates(y)
```

Arguments

y	a list object created from stsm_detect_frequency

Value

a list with the univariate time series and corrected dates

stsm_check_exo *Data check for input exo*

Description

Checks for proper input of the table exo

Usage

```
stsm_check_exo(exo, y)
```

Arguments

exo	matrix of exogenous data
y	input data y

Value

none

stsm_check_exo_fc *Data check for input exo.fc*

Description

Checks for proper input of the table exo.fc

Usage

```
stsm_check_exo_fc(exo.fc, n.ahead)
```

Arguments

exo.fc	exogenous forecast data
n.ahead	forecast periods

Value

none

stsm_check_y *Data check for input y*

Description

Checks for proper input of the table y

Usage

```
stsm_check_y(y)
```

Arguments

y	input data y

Value

none

stsm_constraints *Set the inequality constraints for estimation*

Description

Inequality constraints: ineqA

Usage

```
stsm_constraints(
  prior,
  par,
  freq,
  unconstrained,
  det_trend,
  det_drift,
  det_cycle,
  det_seas,
  det_obs,
  saturating_growth
)
```

Arguments

prior	A data table created by stsm_prior
par	parameter values for the state space model
freq	Frequency of the data
unconstrained	Whether to remove inequality constraints on the trend during estimation
det_trend	Set the trend error variance to 0 (deterministic trend)
det_drift	Set the drift error variance to 0 (deterministic drift)
det_cycle	Set the cycle error variance to 0 (deterministic cycle)
det_seas	Set the seasonality error variances to 0 (deterministic seasonality)
det_obs	Set the observation equation error variance to 0 (deterministic observation equation)
saturating_growth	Force the growth rate to converge to 0 in the long term

Value

list containing the initial values for the Kalman filter

---

stsm_coxstuart *Cox-Stuart Test*

Description

Taken from the ‘tsutils‘ package. Performs the Cox-Stuart test for trend, deviation, or dispersion

Usage

```
stsm_coxstuart(
  y,
  type = c("trend", "deviation", "dispersion"),
  sig_level = 0.01
)
```

Arguments

y	input data
type	Type of test: "trend", "deviation", or "dispersion" If type = "trend", test for changes in trend If type = "deviation", test for changes in deviation If type = "dispersion", test for changes in dispersion (range)
sig_level	Significance level to determine statistically significant seasonal frequencies

Value

list describing the results

stsm_dates_to_interpolate *Create dates to interpolate*

Description

Create dates to interpolate

Usage

```
stsm_dates_to_interpolate(y, dates, exo = NULL, interpolate)
```

Arguments

y	Univariate time series of data values.
dates	Vector of date values for y
exo	Matrix of exogenous variables. Can be used to specify regression effects or other seasonal effects like holidays, etc.
interpolate	Character string of how to interpolate

Value

List of the data, dates, and exo

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
dates_interp = stsm_dates_to_interpolate(y = NA000334Q$y, dates = NA000334Q$date,
interpolate = "monthly")

## End(Not run)
```

`stsm_detect_anomalies` *Detect Anomalies*

Description

Detect anomalies using the estimated structural time series model

Usage

```
stsm_detect_anomalies(
  model,
  y = NULL,
  freq = NULL,
  exo_obs = NULL,
  exo_state = NULL,
  sig_level = 0.01,
  smooth = TRUE,
  plot = FALSE
)
```

Arguments

`model`	Structural time series model estimated using stsm_estimate.
`y`	Univariate time series of data values. May also be a 2 column data frame containing a date column.
`freq`	Frequency of the data (1 (yearly), 4 (quarterly), 12 (monthly), 365.25/7 (weekly), 365.25 (daily)), default is NULL and will be automatically detected
`exo_obs`	Matrix of exogenous variables to be used in the observation equation.
`exo_state`	Matrix of exogenous variables to be used in the state matrix.
`sig_level`	Significance level to determine statistically significant anomalies
`smooth`	Whether or not to use the Kalman smoother
`plot`	Whether to plot everything

Value

data table (or list of data tables) containing the dates of detected anomalies from the filtered and/or smoothed series

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
```

```
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
stsm = stsm_estimate(NA000334Q)
anomalies = stsm_detect_anomalies(model = stsm, y = NA000334Q, plot = TRUE)

## End(Not run)
```

`stsm_detect_breaks` *Detect Structural Breaks*

Description

Detect structural breaks using the estimated structural time series model

Usage

```
stsm_detect_breaks(
  model,
  y,
  components = c("trend", "cycle", "seasonal"),
  freq = NULL,
  exo_obs = NULL,
  exo_state = NULL,
  sig_level = 0.01,
  ci = 0.8,
  smooth = TRUE,
  plot = FALSE,
  cores = NULL,
  show_progress = FALSE
)
```

Arguments

`model`	Structural time series model estimated using stsm_estimate.
`y`	Univariate time series of data values. May also be a 2 column data frame containing a date column.
`components`	Vector of components to test for structural breaks
`freq`	Frequency of the data (1 (yearly), 4 (quarterly), 12 (monthly), 365.25/7 (weekly), 365.25 (daily)), default is NULL and will be automatically detected
`exo_obs`	Matrix of exogenous variables to be used in the observation equation.
`exo_state`	Matrix of exogenous variables to be used in the state matrix.
`sig_level`	Significance level to determine statistically significant anomalies
`ci`	Confidence interval, value between 0 and 1 exclusive.
`smooth`	Whether or not to use the Kalman smoother

plot	Whether to plot everything
cores	Number of cores to use for break detection
show_progress	Whether to show progress bar

Value

data table (or list of data tables) containing the dates of detected anomalies from the filtered and/or smoothed series

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
stsm = stsm_estimate(NA000334Q)
breaks = stsm_detect_breaks(model = stsm, y = NA000334Q, plot = TRUE, cores = 2)

## End(Not run)
```

---

stsm_detect_cycle *Detect cycle from the data*

Description

Detect cycle from the data

Usage

```
stsm_detect_cycle(
  y,
  freq,
  sig_level = 0.01,
  prior = NULL,
  interpolate = NA,
  cl = NULL,
  cores = NULL,
  show_progress = FALSE
)
```

Arguments

y	Univariate time series of data values.
freq	Frequency of the data (1 (yearly), 4 (quarterly), 12 (monthly), 365.25/7 (weekly), 365.25 (daily))
sig_level	Significance level to determine statistically significant seasonal frequencies
prior	A data table created by stsm_prior
interpolate	Character string giving frequency to interpolate to: i.e. "quarterly", "monthly", "weekly", "daily"
cl	a parallel cluster object
cores	Number of cores to use
show_progress	Whether to show progress bar

Value

Numeric value of cycle periodicity

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
cycle = stsm_detect_cycle(y = NA000334Q$y, freq = 4)

## End(Not run)
```

---

stsm_detect_frequency *Detect frequency and dates from the data*

Description

Detect frequency and dates from the data

Usage

```
stsm_detect_frequency(y, freq = NULL)
```

Arguments

y	Univariate time series of data values. May also be a 2 column data frame containing a date column.
freq	Initial setting for the frequency detection

Value

List giving the dates and frequency of the data

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
freq = stsm_detect_frequency(y = NA000334Q)

## End(Not run)
```

`stsm_detect_multiplicative`

Detect if log transformation is best

Description

Detect if log transformation is best

Usage

```
stsm_detect_multiplicative(y, freq, sig_level = 0.01, prior = NULL)
```

Arguments

`y`	an object created from stsm_detect_frequency
`freq`	Frequency of the data
`sig_level`	Significance level to determine statistically significant seasonal frequencies
`prior`	A data table created by stsm_prior

Value

a logical indicating if the model should be multiplicative or not

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
multiplicative = stsm_detect_multiplicative(y = NA000334Q$y, freq = 4)

## End(Not run)
```

`stsm_detect_seasonality` *Detect seasonality from the data*

Description

Detect seasonality from the data

Usage

```
stsm_detect_seasonality(
  y,
  freq,
  sig_level = 0.01,
  prior = NULL,
  interpolate = NA,
  cl = NULL,
  cores = NULL,
  show_progress = FALSE
)
```

Arguments

`y`	Univariate time series of data values.
`freq`	Frequency of the data (1 (yearly), 4 (quarterly), 12 (monthly), 365.25/7 (weekly), 365.25 (daily))
`sig_level`	Significance level to determine statistically significant seasonal frequencies
`prior`	A data table created from stsm_prior
`interpolate`	Character string giving frequency to interpolate to: i.e. "quarterly", "monthly", "weekly", "daily"
`cl`	a parallel cluster object
`cores`	Number of cores to use
`show_progress`	Whether to show progress bar

Value

Numeric vector of seasonal periodicities

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
seasonality = stsm_detect_seasonality(y = NA000334Q$y, freq = 4)

## End(Not run)
```

---

stsm_detect_trend *Detect trend type*

Description

Detect trend type

Usage

```
stsm_detect_trend(
  y,
  freq,
  decomp = "",
  sig_level = 0.01,
  prior = NULL,
  seasons = NULL,
  cycle = NULL,
  cl = NULL,
  cores = NULL,
  verbose = FALSE
)
```

Arguments

y	Univariate time series of data values. May also be a 2 column data frame containing a date column.
freq	Frequency of the data (1 (yearly), 4 (quarterly), 12 (monthly), 365.25/7 (weekly), 365.25 (daily))
decomp	Decomposition model ("tend-cycle-seasonal", "trend-seasonal", "trend-cycle", "trend-noise")

sig_level	Significance level to determine statistically significant seasonal frequencies
prior	A data table created by stsm_prior
seasons	The seasonal periods
cycle	The cycle period
cl	a parallel cluster object
cores	Number of cores to use
verbose	Logical whether to print messages or not

Value

list with trend type and logical flag for deterministic trend if the trend is determined to have 0 differencing

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
trend = stsm_detect_trend(y = NA000334Q$y, freq = 4)

## End(Not run)
```

---

stsm_estimate *Trend cycle seasonal decomposition using the Kalman filter.*

Description

Estimates a structural time series model using the Kalman filter and maximum likelihood. The seasonal and cycle components are assumed to be of a trigonometric form. The function checks three trend specifications to decompose a univariate time series into trend, cycle, and/or seasonal components plus noise. The function automatically detects the frequency and checks for a seasonal and cycle component if the user does not specify the frequency or decomposition model. This can be turned off by setting freq or specifying decomp. State space model for decomposition follows Yt = T_t + C_t + S_t + B*X_t + e_t, e_t ~ N(0, sig_e^2) Y is the data T is the trend component C is the cycle component S is the seasonal component X is the exogenous data with parameter vector B e is the observation error

Usage

```
stsm_estimate(
  y,
  exo_obs = NULL,
  exo_state = NULL,
  state_eqns = NULL,
  freq = NULL,
  decomp = NULL,
  trend = NULL,
  unconstrained = FALSE,
  saturating_growth = FALSE,
  multiplicative = NULL,
  par = NULL,
  seasons = NULL,
  cycle = NULL,
  arma = c(p = NA, q = NA),
  interpolate = NA,
  interpolate_method = NA,
  det_obs = FALSE,
  det_trend = NULL,
  det_seas = FALSE,
  det_drift = FALSE,
  det_cycle = FALSE,
  sig_level = NULL,
  sig_level_seas = NULL,
  sig_level_cycle = NULL,
  sig_level_trend = NULL,
  optim_methods = c("BFGS", "NM", "CG", "SANN"),
  maxit = 10000,
  verbose = FALSE,
  cores = NULL
)
```

Arguments

y	Univariate time series of data values. May also be a 2 column data frame containing a date column.
exo_obs	Matrix of exogenous variables to be used in the observation equation.
exo_state	Matrix of exogenous variables to be used in the state matrix.
state_eqns	Character vector of equations to apply exo_state to the unobserved components. If left as the default, then all variables in exo_state will be applied to all the unobserved components. The equations should look like: "trend ~ var - 1", "drift ~ var - 1", "cycle ~ var - 1", "seasonal ~ var - 1". If only some equations are specified, it will be assumed that the exogenous data will be applied to only those specified equations.
freq	Frequency of the data (1 (yearly), 4 (quarterly), 12 (monthly), 365.25/7 (weekly), 365.25 (daily)), default is NULL and will be automatically detected

`decomp` Decomposition model ("tend-cycle-seasonal", "trend-seasonal", "trend-cycle", "trend-noise")

`trend` Trend specification ("random-walk", "random-walk-drift", "double-random-walk", "random-walk2"). The default is NULL which will choose the best of all specifications based on the maximum likelihood. "random-walk" is the random walk trend. "random-walk-drift" is the random walk with constant drift trend. "double-random-walk" is the random walk with random walk drift trend. "random-walk2" is a 2nd order random walk trend as in the Hodrick-Prescott filter. If trend is "random-walk", the trend model is T_t = T_{t-1} + e_t, e_t ~ N(0, sig_t^2) If trend is "random-walk-drift", the trend model is T_t = T_{t-1} + D_{t-1} + e_t, e_t ~ N(0, sig_t^2) with D_t = d + phi_d*D_{t-1} + n_t, n_t ~ N(0, sig_d^2) If trend is "double-random-walk", the trend model is T_t = M_{t-1} + T_{t-1} + e_t, e_t ~ N(0, sig_t^2) with M_t = M_{t-1} + n_t, n_t ~ N(0, sig_d^2) If trend is "random-walk2", the trend model is T_t = 2T_{t-1} - T_{t-2} + e_t, e_t ~ N(0, sig_t^2)

`unconstrained` Logical whether to remove inequality constraints on the trend during estimation

`saturating_growth` Force the growth rate to converge to 0 in the long term

`multiplicative` If data should be logged to create a multiplicative model. If multiplicative = TRUE, then the data is logged and the original model becomes multiplicative (Y_t = T_t * C_t * S_t * BX_t * e_t)

`par` Initial parameters, default is NULL and will auto-select them

`seasons` The seasonal periods: i.e. c(365.25, 7 if yearly and weekly seasonality). Default is NULL and will be estimated via wavelet analysis. Can set to FALSE if want no seasonality

`cycle` The period for the longer-term cycle. Default is NULL and will be estimated via wavelet analysis. Can set to FALSE if want no cycle, "trig" for trigonometric specification only, or "arma" for ARMA(p,q) specification only.

`arma` Named vector with values for p and q corresponding to the ARMA(p,q) specification if cycle is set to 'arma'. If NA, then will auto-select the order.

`interpolate` Character string giving frequency to interpolate to: i.e. "quarterly", "monthly", "weekly", "daily"

`interpolate_method` Character string giving the interpolation method: i.e. "eop" for end of period, "avg" for period average, or "sum" for period sum.

`det_obs` Set the observation equation error variance to 0 (deterministic observation equation) If det_obs = TRUE then the error variance of the observation equation (sig_e) is set to 0

`det_trend` Set the trend error variance to 0 (deterministic trend) If det_trend = TRUE then the error variance of the trend equation (sig_t) is set to 0 and is referred to as a smooth trend

`det_seas` Set the seasonality error variances to 0 (deterministic seasonality) If det_seas = TRUE then the error variance all seasonality frequency j equations (sig_s) are set to 0 and is referred to as deterministic seasonality

det_drift	Set the drift error variance to 0 (deterministic drift) If det_drift = TRUE then the error variance of the drift equation (sig_d) is set to 0 and is refereed to as a deterministic drift
det_cycle	Set the cycle error variance to 0 (deterministic cycle) If det_cycle = TRUE then the error variance of the cycle equation (sig_c) is set to 0 and is referred to as a deterministic cycle
sig_level	Significance level to determine statistically significance for all tests. Default is 0.01
sig_level_seas	Significance level to determine statistically significant seasonal frequencies. Default is 0.01
sig_level_cycle	Significance level to determine a statistically significant cycle frequency. Default is 0.01
sig_level_trend	Significance level to determine statistically significant order of integration. Default is 0.01
optim_methods	Vector of 1 to 3 optimization methods in order of preference ("NR", "BFGS", "CG", "BHHH", or "SANN")
maxit	Maximum number of iterations for the optimization
verbose	Logical whether to print messages or not
cores	Number of cores to use for seasonality and cycle detection

Value

List of estimation values including a data table with coefficients, convergence code, frequency, decomposition, seasonality, cyclicality, and trend specification as well as the a data table with the original data with dates. Any exogenous data given is also returned.

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
stsm = stsm_estimate(NA000334Q)

## End(Not run)
```

`stsm_filter` *Kalman Filter*

Description

Kalman filter an estimated model from stsm_estimate output. This is a wrapper to stsm_forecast with n.ahead = 0.

Usage

```
stsm_filter(
  model,
  y,
  freq = NULL,
  exo_obs = NULL,
  exo_state = NULL,
  ci = 0.8,
  plot = FALSE,
  plot.decomp = FALSE,
  n.hist = NULL,
  smooth = TRUE,
  dampen_cycle = FALSE
)
```

Arguments

`model`	Structural time series model estimated using stsm_estimate.
`y`	Univariate time series of data values. May also be a 2 column data frame containing a date column.
`freq`	Frequency of the data (1 (yearly), 4 (quarterly), 12 (monthly), 365.25/7 (weekly), 365.25 (daily)), default is NULL and will be automatically detected
`exo_obs`	Matrix of exogenous variables to be used in the observation equation.
`exo_state`	Matrix of exogenous variables to be used in the state matrix.
`ci`	Confidence interval, value between 0 and 1 exclusive.
`plot`	Logical, whether to plot everything
`plot.decomp`	Logical, whether to plot the filtered historical data
`n.hist`	Number of historical periods to include in the forecast plot. If plot = TRUE and n.hist = NULL, defaults to 3 years.
`smooth`	Whether or not to use the Kalman smoother
`dampen_cycle`	Whether to remove oscillating cycle dynamics and smooth the cycle forecast into the trend using a sigmoid function that maintains the rate of convergence

Value

data table (or list of data tables) containing the filtered and/or smoothed series.

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
stsm = stsm_estimate(NA000334Q)
fc = stsm_filter(stsm, y = NA000334Q, plot = TRUE)

## End(Not run)
```

`stsm_fixed_pars` *Fixed parameter setting*

Description

Fixed parameter setting

Usage

```
stsm_fixed_pars(
  par,
  y,
  det_obs = FALSE,
  det_trend = FALSE,
  det_drift = FALSE,
  det_cycle = FALSE,
  det_seas = FALSE,
  saturating_growth = FALSE
)
```

Arguments

`par`	Initial parameters
`y`	Vector of univariate time series
`det_obs`	Set the observation equation error variance to 0 (deterministic observation equation) If det_obs = TRUE then the error variance of the observation equation (sig_e) is set to 0
`det_trend`	Set the trend error variance to 0 (deterministic trend) If det_trend = TRUE then the error variance of the trend equation (sig_t) is set to 0 and is referred to as a smooth trend

det_drift	Set the drift error variance to 0 (deterministic drift) If det_drift = TRUE then the error variance of the drift equation (sig_d) is set to 0 and is refereed to as a deterministic drift
det_cycle	Set the cycle error variance to 0 (deterministic cycle) If det_cycle = TRUE then the error variance of the cycle equation (sig_c) is set to 0 and is referred to as a deterministic cycle
det_seas	Set the seasonality error variances to 0 (deterministic seasonality) If det_seas = TRUE then the error variance all seasonality frequency j equations (sig_s) are set to 0 and is referred to as deterministic seasonality
saturating_growth	Force the growth rate to converge to 0 in the long term

---

`stsm_forecast` *Kalman Filter and Forecast*

Description

Kalman filter and forecast an estimated model from stsm_estimate output

Usage

```
stsm_forecast(
  model,
  y,
  n.ahead = 0,
  freq = NULL,
  exo_obs = NULL,
  exo_state = NULL,
  exo_obs.fc = NULL,
  exo_state.fc = NULL,
  ci = 0.8,
  plot = FALSE,
  plot.decomp = FALSE,
  plot.fc = FALSE,
  n.hist = NULL,
  smooth = TRUE,
  dampen_cycle = FALSE,
  envelope_ci = FALSE
)
```

Arguments

model	Structural time series model estimated using stsm_estimate.
y	Univariate time series of data values. May also be a 2 column data frame containing a date column.
n.ahead	Number of periods to forecast

freq	Frequency of the data (1 (yearly), 4 (quarterly), 12 (monthly), 365.25/7 (weekly), 365.25 (daily)), default is NULL and will be automatically detected
exo_obs	Matrix of exogenous variables to be used in the observation equation.
exo_state	Matrix of exogenous variables to be used in the state matrix.
exo_obs.fc	Matrix of exogenous variables in the observation matrix used for the forecast
exo_state.fc	Matrix of exogenous variables in the state matrix used for the forecast
ci	Confidence interval, value between 0 and 1 exclusive.
plot	Logical, whether to plot everything
plot.decomp	Logical, whether to plot the filtered historical data
plot.fc	Logical, whether to plot the forecast
n.hist	Number of historical periods to include in the forecast plot. If plot = TRUE and n.hist = NULL, defaults to 3 years.
smooth	Whether or not to use the Kalman smoother
dampen_cycle	Whether to remove oscillating cycle dynamics and smooth the cycle forecast into the trend using a sigmoid function that maintains the rate of convergence
envelope_ci	Whether to create a envelope for the confidence interval to smooth out seasonal fluctuations to the longest seasonal period

Value

data table (or list of data tables) containing the filtered and/or smoothed series.

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
stsm = stsm_estimate(NA000334Q)
fc = stsm_forecast(stsm, y = NA000334Q, n.ahead = floor(stsm$freq)*3, plot = TRUE)

## End(Not run)
```

stsm_format_exo *Format exo*

Description

Format the exo table

Usage

```
stsm_format_exo(exo_obs, exo_state, dates, range)
```

Arguments

exo_obs	exogenous observation data
exo_state	exogenous state data
dates	dates vector
range	range of data to include

Value

a data table

stsm_init_pars *Get initial parameter estimates for estimation*

Description

Get initial parameter estimates for estimation

Usage

```
stsm_init_pars(
  y,
  freq,
  trend,
  cycle,
  decomp = "",
  seasons = NULL,
  prior = NULL,
  sig_level = 0.01,
  arma = c(p = NA, q = NA),
  exo = NULL,
  state_eqns = NULL,
  interpolate = NA,
  interpolate_method = NA
)
```

Arguments

y	an object created from stsm_detect_frequency
freq	Frequency of the data
trend	Trend specification ("random-walk", "random-walk-drift", "double-random-walk", "random-walk2").
cycle	The period for the longer-term cycle
decomp	Decomposition model ("tend-cycle-seasonal", "trend-seasonal", "trend-cycle", "trend-noise")
seasons	The seasonal lengths to split the seasonality into
prior	A data table created by stsm_prior
sig_level	Significance level for statistical tests
arma	Named vector with values for p and q corresponding to the ARMA(p,q) specification if
exo	Matrix of exogenous variables. Can be used to specify regression effects or other seasonal effects like holidays, etc.
state_eqns	Character vector of equations to apply exo_state to the unobserved components. If left as the default, then all variables in exo_state will be applied to all the unobserved components. The equations should look like: "trend ~ var - 1", "drift ~ var - 1", "cycle ~ var - 1", "seasonal ~ var - 1". If only some equations are specified, it will be assumed that the exogenous data will be applied to only those specified equations.
interpolate	Character string giving frequency to interpolate to: i.e. "quarterly", "monthly", "weekly", "daily" cycle is set to 'arma'. If NA, then will auto-select the order.
interpolate_method	Character string giving the interpolation method:

Value

named vector containing the initial parameter estimates for estimation

---

stsm_na_kalman *Missing Value Imputation by Kalman Smoothing and State Space Models*

Description

Simplified version taken from the 'imputeTS' package. Uses Kalman Smoothing on structural time series models for imputation. It uses "StructTS" to build a "basic structural model" if the frequency of y is greater than 1. Otherwise, it uses a local trend model.

Usage

```
stsm_na_kalman(y)
```

Arguments

y	Univariate time series

`stsm_prior` *Return a naive model prior decomposition*

Description

Return a naive model prior decomposition

Usage

```
stsm_prior(y, freq, decomp = "", seasons = NULL, cycle = NULL)
```

Arguments

y	an object created from stsm_detect_frequency
freq	Frequency of the data
decomp	decomposition string
seasons	The seasonal periods to split the seasonality into
cycle	The cycle periods

Value

data table containing a naive decomposition using STL

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
prior = stsm_prior(y = NA000334Q$y, freq = 4)

## End(Not run)
```

`stsm_ssm` *State space model*

Description

Creates a state space model in list form yt = H*B + B^O X^O_t + e_t B = F*B_{t-1} + B^S X^S_t + u_t

Usage

```
stsm_ssm(
  par = NULL,
  yt = NULL,
  decomp = NULL,
  trend = NULL,
  init = NULL,
  model = NULL,
  prior = NULL,
  freq = NULL,
  seasons = NULL,
  cycle = NULL,
  interpolate = NULL,
  interpolate_method = NULL
)
```

Arguments

`par`	Vector of named parameter values, includes the harmonics
`yt`	Univariate time series of data values
`decomp`	Decomposition model ("tend-cycle-seasonal", "trend-seasonal", "trend-cycle", "trend-noise")
`trend`	Trend specification ("random-walk", "random-walk-drift", "double-random-walk", "random-walk2"). The default is NULL which will choose the best of all specifications based on the maximum likelihood. "random-walk" is the random walk trend. "random-walk-drift" is the random walk with constant drift trend. "double-random-walk" is the random walk with random walk drift trend. "random-walk2" is a 2nd order random walk trend as in the Hodrick-Prescott filter.
`init`	Initial state values for the Kalman filter
`model`	a stsm_estimate model object
`prior`	Model prior built from stsm_prior. Only needed if prior needs to be built for initial values
`freq`	Frequency of the data. Only needed if prior needs to be built for initial values and prior = NULL
`seasons`	Numeric vector of seasonal frequencies. Only needed if prior needs to be built for initial values and prior = NULL

cycle	Numeric value for the cycle frequency. Only needed if prior needs to be built for initial values and prior = NULL
interpolate	Character string of how to interpolate
interpolate_method	Character string for the method of interpolation

Value

List of space space matrices

Examples

```
## Not run:
#GDP Not seasonally adjusted
library(autostsm)
data("NA000334Q", package = "autostsm") #From FRED
NA000334Q = data.table(NA000334Q, keep.rownames = TRUE)
colnames(NA000334Q) = c("date", "y")
NA000334Q[, "date" := as.Date(date)]
NA000334Q[, "y" := as.numeric(y)]
NA000334Q = NA000334Q[date >= "1990-01-01", ]
stsm = stsm_estimate(NA000334Q)
ssm = stsm_ssm(model = stsm)

## End(Not run)
```

UNRATE *Unemployment Rate Seasonally Adjusted*

Description

Unemployment Rate Seasonally Adjusted

Usage

```
data(UNRATE)
```

Format

data.table with columns DATE and UNRATE, monthly frequency

Source

FRED

UNRATENSA *Unemployment Rate Not Seasonally Adjusted*

Description

Unemployment Rate Not Seasonally Adjusted

Usage

```
data(UNRATENSA)
```

Format

data.table with columns DATE and UNRATENSA, monthly frequency

Source

FRED

Index