Package 'tsgc'

May 8, 2026

Title Time Series Methods Based on Growth Curves

Version 0.0

Description The 'tsgc' package provides comprehensive tools for the analysis and forecasting of epidemic trajectories.
It is designed to model the progression of an epidemic over time while accounting for the various uncertainties
inherent in real-
time data. Underpinned by a dynamic Gompertz model, the package adopts a state space approach, using the Kalman filter for flexible and robust estimation of the non-
linear growth pattern commonly observed in
epidemic data. The reinitialization feature enhances the model's ability to adapt to the emergence of new waves.
The forecasts generated by the package are of value to public health officials and researchers who need to
understand and predict the course of an epidemic to inform decision-
making. Beyond its application in public
health, the package is also a useful resource for researchers and practitioners in fields where the trajectories
of interest resemble those of epidemics, such as innovation diffusion. The package includes functionalities for
data preprocessing, model fitting, and forecast visualization, as well as tools for evaluating forecast accuracy.
The core methodologies implemented in 'tsgc' are based on well-
established statistical techniques as described in
Harvey and Kattuman (2020) <doi:10.1162/99608f92.828f40de>, Harvey and Kattuman (2021) <doi:10.1098/rsif.2021.0179>, and Ashby, Harvey, Kattuman, and Thamotheram (2024) <https:
//www.jbs.cam.ac.uk/wp-content/uploads/2024/03/cchle-tsgc-paper-2024.pdf>.

URL https://github.com/Craig-PT/tsgc

License GPL (>= 3)

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.2.3

Suggests ggfortify, knitr, RColorBrewer, rmarkdown, ggforce, gridExtra, latex2exp, here, timetk, testthat, purrr, kableExtra

Config/testthat/edition 3

Imports KFAS, xts, ggplot2, ggthemes, zoo, magrittr, scales, dplyr, tidyr, methods

BugReports https://github.com/Craig-PT/tsgc/issues

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author Craig Thamotheram [aut, cre]

Maintainer Craig Thamotheram <cpt@tacindex.com>

Repository CRAN

Date/Publication 2024-08-26 12:10:04 UTC

Contents

argmax *Return index and value of maximum*

Description

Something similar to Python's argmax.

Usage

```
argmax(x, decreasing = TRUE)
```

Arguments

x	Object to have its maximum found
decreasing	Logical value indicating whether x should be ordered in decreasing order. Default is TRUE. Setting this to FALSE would find the minimum.

Value

The maximum value and its index.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
argmax(gauteng)
```

df2ldl *Compute log growth rate of cumulated dataset*

Description

Helper method to compute the log growth rates of cumulated variables. It will compute the log cumulative growth rate for each column in the data frame.

Usage

```
df2ldl(dt)
```

Arguments

dt	Cumulated data series.

Value

A data frame of log growth rates of the cumulated variable which has been inputted via the parameter dt.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
df2ldl(gauteng)
```

`england` *Cumulative cases of Covid-19 in England.*

Description

Cumulative cases of Covid-19 in England.

Usage

```
data(england)
```

Format

An object of class `"xts"`;

Cases Cumulative cases of Covid-19

References

Downloaded from https://ukhsa-dashboard.data.gov.uk/topics/covid-19

Examples

```
data(england)
# plot daily cases
plot(diff(england))
```

`FilterResults-class` *FilterResults*

Description

Class for estimated Dynamic Gompertz Curve model and contains methods to extract smoothed/filtered estimates of the states, the level of the incidence variable y, and forecasts of y.

Methods

`get_growth_y(smoothed = FALSE, return.components = FALSE)` Returns the growth rate of the incidence (y) of the cumulated variable (Y). Computed as

$$g_t = \exp\{\delta_t\} + \gamma_t.$$

Parameters:

- `smoothed` Logical value indicating whether to use the smoothed estimates of δ and γ to compute the growth rate (`TRUE`), or the contemporaneous filtered estimates (`FALSE`). Default is `FALSE`.
- `return.components` Logical value indicating whether to return the estimates of δ and γ as well as the estimates of the growth rate, or just the growth rate. Default is `FALSE`.

Return Value: `xts` object containing smoothed/filtered growth rates and components (δ and γ), where applicable.

`get_gy_ci(smoothed = FALSE, confidence_level = 0.68)` Returns the growth rate of the incidence (y) of the cumulated variable (Y). Computed as

$$g_t = \exp\{\delta_t\} + \gamma_t.$$

Parameters:

- `smoothed` Logical value indicating whether to use the smoothed estimates of δ and γ to compute the growth rate (`TRUE`), or the contemporaneous filtered estimates (`FALSE`). Default is `FALSE`.
- `confidence_level` Confidence level for the confidence interval. Default is 0.68, which is one standard deviation for a normally distributed random variable.

Return Value: `xts` object containing smoothed/filtered growth rates and upper and lower bounds for the confidence intervals.

`predict_all(n.ahead, sea.on = FALSE, return.all = FALSE)` Returns forecasts of the incidence variable y, the state variables and the conditional covariance matrix for the states.

Parameters:

- `n.ahead` The number of forecasts you wish to create from the end of your sample period.
- `sea.on` Logical value indicating whether seasonal components should be included in the state-space model or not. Default is `TRUE`.
- `return.all` Logical value indicating whether to return all filtered estimates and forecasts (`TRUE`) or only the forecasts (`FALSE`). Default is `FALSE`.

Return Value: `xts` object containing the forecast (and filtered, where applicable) level of y (`y.hat`), δ (`level.t.t`), γ (`slope.t.t`), vector of states including the seasonals where applicable (`a.t.t`) and covariance matrix of all states including seasonals where applicable (`P.t.t`).

`predict_level( y.cum, n.ahead, confidence_level, sea.on = FALSE, return.diff = FALSE )` Forecast the cumulated variable or the incidence of it. This function returns the forecast of the cumulated variable Y, or the forecast of the incidence of the cumulated variable, y. For example, in the case of an epidemic, y might be daily new cases of the disease and Y the cumulative number of recorded infections.

Parameters:

- `y.cum` The cumulated variable.
- `n.ahead` The number of periods ahead you wish to forecast from the end of the estimation window.
- `confidence_level` The confidence level for the log growth rate that should be used to compute the forecast intervals of y.
- `return.diff` Logical value indicating whether to return the cumulated variable, Y, or the incidence of it, y (i.e., the first difference of the cumulated variable). Default is `FALSE`.

Return Value: `xts` object containing the point forecasts and upper and lower bounds of the forecast interval.

`print_estimation_results()` Prints a table of estimated parameters in a format ready to paste into LaTeX.

References

Harvey, A. C. and Kattuman, P. (2021). A Farewell to R: Time Series Models for Tracking and Forecasting Epidemics, Journal of the Royal Society Interface, vol 18(182): 20210179

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-20")

# Specify a model
model <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)
# Estimate a specified model
res <- model$estimate()
# Print estimation results
res$print_estimation_results()
# Forecast 7 days ahead from the end of the estimation window
res$predict_level(y.cum = gauteng[idx.est], n.ahead = 7,
  confidence_level = 0.68)
# Forecast 7 days ahead from the model and return filtered states
res$predict_all(n.ahead = 7, return.all = TRUE)
# Return the filtered growth rate and its components
res$get_growth_y(return.components = TRUE)
# Return smoothed growth rate of incidence variable and its confidence
# interval
res$get_gy_ci(smoothed = TRUE, confidence_level = 0.68)
```

`forecast.peak` *Returns forecast of number of periods until peak given estimated state variables δ and γ.*

Description

Since Harvey and Kattuman (2021) show that

$$g_{y,t+\ell|T} = \exp\{\delta_{T|T} + \ell\gamma_{T|T}\} + \gamma_{T|T},$$

we can compute the ℓ for which $g_{y,t} = 0$ and then will fall below zero. This ℓ is given by

$$\ell = \frac{\ln(-\gamma_{T|T}) - \delta_{T|T}}{\gamma_{T|T}}.$$

This is predicated on $\gamma_{T|T} < 0$, else there is super-exponential growth an no peak in sight. Of course, it only makes sense to investigate an upcoming peak for $g_{y,T|T} > 0$ (when cases are growing).

Usage

```
forecast.peak(delta, gamma)
```

Arguments

`delta`	The estimate of δ, the level of $\ln g$.
`gamma`	The estimate of γ, the slope of $\ln g$.

Value

Forecast of number of periods until peak.

Examples

```
# Forecasts the peak of an epidemic with gamma < 0 so that a peak is in
# sight.
forecast.peak(-2.87,-0.045)

# Does not return a result (returns an error as gamma > 0)
try(forecast.peak(-2.87,0.045), silent=TRUE)
```

`forecast_peak` *Returns forecast of number of periods until peak given* `KFAS::KFS` *output.*

Description

Since Harvey and Kattuman (2021) show that

$$g_{y,t+\ell|T} = \exp\{\delta_{T|T} + \ell\gamma_{T|T}\} + \gamma_{T|T},$$

we can compute the ℓ for which $g_{y,t} = 0$ and then will fall below zero. This ℓ is given by

$$\ell = \frac{\ln(-\gamma_{T|T}) - \delta_{T|T}}{\gamma_{T|T}}.$$

This is predicated on $\gamma_{T|T} < 0$, else there is super-exponential growth and no peak in sight. Of course, it only makes sense to investigate an upcoming peak for $g_{y,T|T} > 0$ (when cases are growing). The estimates of $\delta_{T|T}$ and $\gamma_{T|T}$ are extracted from the KFS object passed to the function.

Usage

```
forecast_peak(kfs_out)
```

Arguments

kfs_out	The KFAS::KFS object for which the forecast peak is to be calculated. This would be the output element of a model estimated in the SSModelDynamicGompertz or SSModelDynamic

Value

Forecast of number of periods until peak.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-06")

res <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)$estimate()

forecast_peak(res$output)
```

---

gauteng *Cumulative cases of Covid-19 in the South African province of Gauteng.*

Description

Cumulative cases of Covid-19 in the South African province of Gauteng.

Usage

```
data(gauteng)
```

Format

An object of class "xts";

Cases Cumulative cases of Covid-19 from 10th March 2020

References

Downloaded from https://sacoronavirus.co.za/

Examples

```
data(gauteng)
# plot daily cases
plot(diff(gauteng))
```

---

plot_forecast *Plots forecast and realised values of the log cumulative growth rate*

Description

Plots actual and filtered values of the log cumulative growth rate ($\ln(g_t)$) in the estimation sample and the forecast and realised log cumulative growth rate out of the estimation sample.

Usage

```
plot_forecast(
  res,
  y.eval,
  n.ahead = 14,
  plt.start.date = NULL,
  title = "",
  caption = ""
)
```

Arguments

res	Results object estimated using the estimate() method.
y.eval	The out-of-sample realisation of the log growth rate of the cumulated variable (i.e. the actual values to which the forecasts should be compared).
n.ahead	The number of time periods ahead from the end of the sample to be forecast. The default is 14.
plt.start.date	Plot start date. Default is NULL which is the start of the estimation sample.
title	Plot title. Enter as text string.
caption	Plot caption. Enter as text string.

Value

A ggplot2 plot.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-20")
idx.eval <- (zoo::index(gauteng) >= as.Date("2020-07-20")) &
     zoo::index(gauteng) <= as.Date("2020-07-27")

# Specify a model
model <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)
# Estimate a specified model
res <- model$estimate()

# Plot forecast and realised log growth rate of cumulative cases
plot_forecast(res, y.eval = df2ldl(gauteng[idx.eval]), n.ahead = 7,
  title = "Forecast ln(g)", plt.start.date = as.Date("2020-07-13"))
```

---

plot_gy_ci *Plots the growth rates and slope of the log cumulative growth rate*

Description

Plots the smoothed/filtered growth rate of the difference in the cumulated variable (g_y) and the associated confidence intervals.

Usage

```
plot_gy_ci(
  res,
  plt.start.date = NULL,
  smoothed = FALSE,
  title = NULL,
  series.name = NULL,
  pad.right = NULL
)
```

Arguments

res	Results object estimated using the estimate() method.
plt.start.date	Plot start date. Default is NULL which is the start of the estimation sample.
smoothed	Logical value indicating whether to used the smoothed estimates of δ and γ. Default is FALSE, in which case the filtered estimates are returned.
title	Title for plot. Enter as text string. NULL (i.e. no title) by default.

series.name	The name of the series the growth rate is being computed for. E.g. 'New cases'.
pad.right	Numerical value for the amount of time periods of blank space you wish to leave on the right of the graph. Extends the horizontal axis by the given number of time periods.

Value

A ggplot2 plot.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-20")

# Specify a model
model <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)
# Estimate a specified model
res <- model$estimate()

# Plot filtered gy, g and gamma
plot_gy_ci(res, plt.start.date = as.Date("2020-07-13"))
```

---

plot_gy_components *Plots the growth rates and slope of the log cumulative growth rate*

Description

Plots the smoothed/filtered growth rate of the difference in the cumulated variable (g_y), the smoothed/filtered growth rate of the the cumulated variable (g), and the smoothed/filtered slope of $\ln(g)$, γ. Following Harvey and Kattuman (2021), we compute $g_{y,t}$ as

$$g_{y,t} = \exp(\delta_t) + \gamma_t.$$

Usage

```
plot_gy_components(res, plt.start.date = NULL, smoothed = FALSE, title = NULL)
```

Arguments

res	Results object estimated using the estimate() method.
plt.start.date	Plot start date. Default is NULL which is the start of the estimation sample.
smoothed	Logical value indicating whether to used the smoothed estimates of δ and γ. Default is FALSE, in which case the filtered estimates are returned.
title	Title for plot. Enter as text string. NULL (i.e. no title) by default.

Value

A ggplot2 plot.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-20")

# Specify a model
model <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)
# Estimate a specified model
res <- model$estimate()

# Plot filtered gy, g and gamma
plot_gy_components(res, plt.start.date = as.Date("2020-07-06"))
```

---

plot_holdout *Plots the forecast of new cases (the difference of the cumulated variable) over a holdout sample.*

Description

Plots actual values of the difference in the cumulated variable, the forecasts of the cumulated variable (both including and excluding the seasonal component, where a seasonal is specified) and forecast intervals around the forecasts, plus the actual outcomes from the holdout sample. The forecast intervals are based on the prediction intervals for $\ln(g_t)$. Also reports the mean absolute percentage prediction error over the holdout sample.

Usage

```
plot_holdout(
  res,
  Y,
  Y.eval,
  confidence.level = 0.68,
  date_format = "%Y-%m-%d",
  series.name = NULL,
  title = NULL,
  caption = NULL
)
```

Arguments

res	Results object estimated using the estimate() method.
Y	Values of the cumulated variable to be used in the estimation window.

Y.eval	Values of the cumulated variable to be used in the holdout sample (i.e. to which the forecasts should be compared to).
confidence.level	Width of prediction interval for $\ln(g_t)$ to use in forecasts of $y_t = \Delta Y_t$. Default is 0.68, which is approximately one standard deviation for a Normal distribution.
date_format	Date format, e.g. '%Y-%m-%d', which is the default.
series.name	Name of the variable you are forecasting for the purposes of a y-axis label. E.g. if series.name = "Cases" the y-axis will show "New Cases".
title	Title for forecast plot. Enter as text string. NULL (i.e. no title) by default.
caption	Caption for forecast plot. Enter as text string. NULL (i.e. no caption) by default.

Value

A ggplot2 plot.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-20")
idx.eval <- (zoo::index(gauteng) >= as.Date("2020-07-20")) &
     zoo::index(gauteng) <= as.Date("2020-07-27")

# Specify a model
model <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)
# Estimate a specified model
res <- model$estimate()

# Plot forecasts and outcomes over evaluation period
plot_holdout(res = res, Y = gauteng[idx.est], Y.eval = gauteng[idx.eval])
```

---

plot_new_cases *Plots the forecast of new cases (the difference of the cumulated variable)*

Description

Plots actual values of the difference in the cumulated variable, the forecasts of the cumulated variable (both including and excluding the seasonal component, where a seasonal is specified) and forecast intervals around the forecasts. The forecast intervals are based on the prediction intervals for $\ln(g_t)$.

Usage

```
plot_new_cases(
  res,
  Y,
  n.ahead,
  confidence.level = 0.68,
  date_format = "%Y-%m-%d",
  title = NULL,
  plt.start.date = NULL
)
```

Arguments

res	Results object estimated using the estimate() method.
Y	Cumulated variable.
n.ahead	Number of forecasts (i.e. number of periods ahead to forecast from end of estimation window).
confidence.level	Width of prediction interval for $\ln g_t$ to use in forecasts of $y_t = \Delta Y_t$. Default is 0.68, which is approximately one standard deviation for a Normal distribution.
date_format	Date format. Default is '%Y-%m-%d'.
title	Title for forecast plot. Enter as text string. NULL (i.e. no title) by default.
plt.start.date	First date of actual data (from estimation sample) to plot on graph.NULL (i.e. plots all data in estimation window) by default.

Value

A ggplot2 plot.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-20")

# Specify a model
model <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)
# Estimate a specified model
res <- model$estimate()

# Plot forecast of new cases 7 days ahead
plot_new_cases(res, Y = gauteng[idx.est], n.ahead = 7,
confidence.level = 0.68, date_format = "%Y-%m-%d",
title = "Forecast new cases", plt.start.date = as.Date("2020-07-13"))
```

`reinitialise_dataframe` *Reinitialise a data frame by subtracting the* `reinit.date` *row from all columns*

Description

Reinitialise a data frame by subtracting the `reinit.date` row from all columns

Usage

```
reinitialise_dataframe(dt, reinit.date)
```

Arguments

`dt`	Cumulated data series.
`reinit.date`	Reinitialisation date. E.g. ‘`'2021-05-12'`’.

Value

The reinitialised data frame

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
reinitialise_dataframe(gauteng,as.Date("2021-01-01"))
```

`SSModelBase-class` *Base class for estimating time-series growth curve models. Classes* `SSModelDynamicGompertz` *and* `SSModelDynGompertzReinit` *refer back to this base class.*

Description

Base class for estimating time-series growth curve models. Classes `SSModelDynamicGompertz` and `SSModelDynGompertzReinit` refer back to this base class.

Methods

`estimate(sea.type = "trigonometric", sea.period = 7)` Estimates the dynamic Gompertz curve model when applied to an object of class `SSModelDynamicGompertz` or `SSModelDynGompertzReinit`.

Parameters:

- `sea.type` Seasonal type. Options are `'trigonometric'` and `'none'`. `'trigonometric'` will yield a model with a trigonometric seasonal component and `'none'` will yield a model with no seasonal component.
- `sea.period` The period of seasonality. For a day-of-the-week effect with daily data, this would be 7. Not required if `sea.type = 'none'`.

Return Value: An object of class `FilterResults` containing the result output for the estimated dynamic Gompertz curve model.

`get_dynamic_gompertz_model( y, q = NULL, sea.type = "trigonometric", sea.period = 7, a1 = NULL, P1 = NULL, Q` Returns dynamic Gompertz curve model.

Parameters:

- `y` The cumulated variable
- `q` The signal-to-noise ratio (ratio of slope to irregular variance). Defaults to `'NULL'`, in which case no signal-to-noise ratio will be imposed. Instead, it will be estimated.
- `sea.type` Seasonal type. Options are `'trigonometric'` and `'none'`. `'trigonometric'` will yield a model with a trigonometric seasonal component and `'none'` will yield a model with no seasonal component.
- `sea.period` The period of seasonality. For a day-of-the-week effect with daily data, this would be 7. Not required if `sea.type = 'none'`.
- `a1` Optional parameter specifying the prior mean of the states. Defaults to `'NULL'`. Leave as `'NULL'` for a diffuse prior (no prior information). If a proper prior is to be specified, both `a1` and `P1` must be given.
- `P1` Optional parameter specifying the prior mean of the states. Defaults to `'NULL'`. Leave as `'NULL'` for a diffuse prior (no prior information). If a proper prior is to be specified, both `a1` and `P1` must be given.
- `Q` Optional parameter specifying the state error variances where these are to be imposed rather than estimated. Defaults to `'NULL'` which will see the variances estimated.
- `H` Optional parameter specifying the irregular variance where this is to be imposed rather than estimated. Defaults to `'NULL'` which will see the variance estimated.

Description: The dynamic Gompertz with an integrated random walk (IRW) trend is

$$\ln g_t = \delta_t + \varepsilon_t, \quad \varepsilon_t \sim NID(0, \sigma_\varepsilon^2), \quad t = 2, ..., T,$$

where Y_t is the cumulated variable, $y_t = \Delta Y_t$, $\ln g_t = \ln y_t - \ln Y_{t-1}$ and

$$\delta_t = \delta_{t-1} + \gamma_{t-1},$$

$$\gamma_t = \gamma_{t-1} + \zeta_t, \quad \zeta_t \sim NID(0, \sigma_\zeta^2),$$

where the observation disturbances ε_t and slope disturbances ζ_t, are iid Normal and mutually independent. Note that, the larger the signal-to-noise ratio, $q_\zeta = \sigma_\zeta^2/\sigma_\varepsilon^2$, the faster the slope changes in response to new observations. Conversely, a lower signal-to-noise ratio induces smoothness.

For the model without seasonal terms (`sea.type = 'none'`) the are priors are

$$\begin{pmatrix}\delta_1 & \gamma_1\end{pmatrix} \sim N(a_1, P_1)$$

. The diffuse prior has $P_1 = \kappa I_{2\times 2}$ with $\kappa \to \infty$. Implementation of the diffuse prior is handled by the package `KFAS` (Helske, 2017). Where the model has a seasonal component (`sea.type = 'trigonometric'`), the vector of prior means a_1 and the prior covariance matrix P_1 are extended accordingly.
See the vignette for details of the variance matrix Q. $H = \sigma_\varepsilon^2$.

`update(pars, model, q, sea.type)` Update method for Kalman filter to implement the dynamic Gompertz curve model. A maximum of 3 parameters are used to set the observation noise (1 parameter), the transition equation slope and seasonal noise. If q (signal to noise ratio) is not null then the slope noise is set using this ratio.

Parameters:

- `pars` Vector of parameters.
- `model` `KFS` model object.
- `q` The signal-to-noise ratio (ratio of slope to irregular variance).
- `sea.type` Seasonal type. Options are `'trigonometric'` and `'none'`.

Return Value: `KFS` model object.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-06")

# Specify a model
model <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)
# Estimate a specified model
res <- model$estimate()
```

`SSModelDynamicGompertz-class`
Class for dynamic Gompertz curve state space model object.

Description

Class for dynamic Gompertz curve state space model object.

Methods: `get_model(y, q = NULL, sea.type = 'trigonometric', sea.period = 7)` Retrieves the model object.

Parameters:

- `y` The cumulated variable.
- `q` The signal-to-noise ratio (ratio of slope to irregular variance). Defaults to `'NULL'`, in which case no signal-to-noise ratio will be imposed. Instead, it will be estimated.

- sea.type Seasonal type. Options are 'trigonometric' and 'none'. 'trigonometric' will yield a model with a trigonometric seasonal component and 'none' will yield a model with no seasonal component.
- sea.period The period of seasonality. For a day-of-the-week effect with daily data, this would be 7. Not required if sea.type = 'none'.

Return Value: KFS model object.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-06")

# Specify a model
model <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)
# Estimate a specified model
res <- model$estimate()
```

SSModelDynGompertzReinit-class

Class for re-initialised dynamic Gompertz curve model

Description

This class allows the implementation of the reinitialisation procedure described in the vignette and summarised below. Let $t = r$ denote the re-initialization date and r_0 denote the date at which the cumulative series is set to 0. As the growth rate of cumulative cases is defined as $g_t \equiv \frac{y_t}{Y_{t-1}}$, we have:

$$\ln g_t = \ln y_t - \ln Y_{t-1} \quad t = 1, \ldots, r$$

$$\ln g_t^r = \ln y_t - \ln Y_{t-1}^r \quad t = r+1, \ldots, T$$

$$Y_t^r = Y_{t-1}^r + y_t \quad t = r, \ldots, T$$

where Y_t^r is the cumulative cases after re-initialization. We choose to set the cumulative cases to zero at $r_0 = r - 1, Y_{r-1}^r = 0$, such that the growth rate of cumulative cases is available from $t = r + 1$ onwards. We reinitialise the model by specifying the prior distribution for the initial states appropriately. See the vignette for details.

Methods:

- new(Y, q = NULL, reinit.date=NULL, original.results=NULL,use.presample.info=TRUE) Create an instance of the SSModelDynGompertzReinit class.

  Parameters:

  - Y The cumulated variable.
  - q The signal-to-noise ratio (ratio of slope to irregular variance). Defaults to 'NULL', in which case no signal-to-noise ratio will be imposed. Instead, it will be estimated.

  - reinit.date The reinitialisation date r. Should be specified as an object of class "Date". Must be specified.
  - original.results Rather than re-estimating the model up to the reinit.date, a FilterResults class object can be specified here and the parameters for the reinitialisation will be taken from this object. Default is NULL. This parameter is optional.
  - use.presample.info Logical value denoting whether or not to use information from before the reinitialisation date in the reinitialisation procedure. Default is TRUE. If FALSE, the model is estimated from scratch from the reinitialisation date and no attempt to use information from before the reinitialisation date is made.

- get_model(y, q=NULL, sea.type = NULL, sea.period) Retrieves the model object, which is a dynamic Gompertz curve model reinitialised at self$reinit.date.

  Parameters:

  - y The cumulated variable.
  - q The signal-to-noise ratio (ratio of slope to irregular variance). Defaults to 'NULL', in which case no signal-to-noise ratio will be imposed. Instead, it will be estimated.
  - sea.type Seasonal type. Options are 'trigonometric' and 'none'. 'trigonometric' will yield a model with a trigonometric seasonal component and 'none' will yield a model with no seasonal component.
  - sea.period The period of seasonality. For a day-of-the-week effect with daily data, this would be 7. Not required if sea.type = 'none'.

  Return Value: KFS model object.

Examples

```
library(tsgc)
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2021-05-20")

# Specify a model
model.reinit <- SSModelDynGompertzReinit$new(Y = gauteng[idx.est], q = 0.005,
  reinit.date = as.Date("2021-04-29"))
# Estimate a specified model
res.reinit <- model.reinit$estimate()

## Alternatively, we could feed in a prior results object rather than a
## reinitialisation date. The results are identical to the above.

# Specify initial model
idx.orig <- zoo::index(gauteng) <= as.Date("2021-04-29")
model.orig <- SSModelDynamicGompertz$new(Y = gauteng[idx.orig], q = 0.005)
res.orig <- model.orig$estimate()
# Estimate a specified model
model.reinit2 <- SSModelDynGompertzReinit$new(Y = gauteng[idx.est],
q = 0.005, reinit.date = as.Date("2021-04-29"), original.results = res.orig)
res.reinit2 <- model.reinit2$estimate()
```

`write_results` *Write a selection of relevant results to disc*

Description

Function writes the following results to csv files which get saved in the location specified in `res.dir`: forecast new cases or incidence variable, y; the filtered level and slope of $\ln g$, δ and γ; filtered estimates of g_y and the confidence intervals for these estimates.

Usage

```
write_results(res, res.dir, Y, n.ahead, confidence.level)
```

Arguments

`res`	Results object estimated using the '`estimate()`' method.
`res.dir`	File path to save the results to.
`Y`	Cumulated variable.
`n.ahead`	Number of periods ahead to forecast.
`confidence.level`	Confidence level to use for the confidence interval on the forecasts $\ln(g_t)$.

Value

A number of csv files saved in the directory specified in `res.dir`.

Examples

```
# Not run as do not wish to save to local disc when compiling documentation.
# Below will run if copied and pasted into console.
library(tsgc)
library(here)

res.dir <- tempdir()
data(gauteng,package="tsgc")
idx.est <- zoo::index(gauteng) <= as.Date("2020-07-06")
res <- SSModelDynamicGompertz$new(Y = gauteng[idx.est], q = 0.005)$estimate()

tsgc::write_results(
res=res, res.dir = res.dir, Y = gauteng[idx.est], n.ahead = 14,
confidence.level = 0.68
)
```

Index