Package ‘tssim’

December 13, 2024

Title Simulation of Daily and Monthly Time Series

Version 0.2.7

Maintainer Daniel Ollech <daniel.ollech@bundesbank.de>

Description Flexible simulation of time series using time series components, including seasonal, calendar and outlier effects. Main algorithm described in Ollech, D. (2021) <doi:10.1515/jtse-2020-0028>.

License GPL-3

Depends R (>= 3.1.0)

Encoding UTF-8

RoxygenNote 7.3.2

Imports dsa, forecast, mvtnorm, stats, timeDate, tsbox, utils, xts, zoo

NeedsCompilation no

Author Daniel Ollech [aut, cre]

Repository CRAN

Date/Publication 2024-12-13 13:20:02 UTC

Contents

`.stretch_re` *Use time warping to reduce the number of observations in a month*

Description

Reduce the number of observations in a month using time warping / stretching. Only relevant if a daily time series is simulated

Usage

```
.stretch_re(seas_component)
```

Arguments

`seas_component` Seasonal component for day-of-the-month

Details

Usually time warping would be used to stretch the number of observations of a time series in a given interval to more observations. Here it is used to reduce the number of observations (31) to the number of days in a given month while maintaining the underlying trajectory of the data. This is done by first creating a very long time series for each month, interpolating missing values by spline interpolation and then reducing the number of observations to the number suitable for a given month.

Value

Returns a `xts` time series containing the day-of-the-month effect.

Author(s)

Daniel Ollech

References

Ollech, D. (2021). Seasonal adjustment of daily time series. Journal of Time Series Econometrics. doi:10.1515/jtse20200028

sim_calendar *Simulate calendar effects*

Description

Simulate a time series containing specified calendar effects

Usage

```
sim_calendar(
  n,
  which = c("Easter", "Ascension"),
  from = 0,
  to = 0,
  freq = 12,
  effect_size = 3,
  start = "2020-01-01",
  multiplicative = TRUE,
  time_dynamic = 1,
  center = TRUE
)
```

Arguments

`n`	Time series length
`which`	Holidays to be used, functions from timeDate package used
`from`	days before the Holiday to include
`to`	days after the Holiday to include
`freq`	Frequency of the time series
`effect_size`	Mean size of calendar effect
`start`	Start Date of output time series
`multiplicative`	Boolean. Is multiplicative time series model assumed?
`time_dynamic`	Should the calendar effect change over time
`center`	Should calendar variable be center, i.e. mean=0

Details

If multiplicative is true, the effect size is measured in percentage. If is not true, the effect size is unit less and thus adopts the unit of the time series the calendars are added to. The time_dynamic parameter controls the change of the calendar effect. The effect of the previous year is multiplied by the time_dynamic factor.

Value

The function returns a time series of class `xts`

Author(s)

Daniel Ollech

References

Ollech, D. (2021). Seasonal adjustment of daily time series. Journal of Time Series Econometrics. doi:10.1515/jtse20200028

Examples

```
plot(sim_calendar(60, from=0, to=4, freq=12))
```

---

sim_daily *Simulate a daily seasonal series*

Description

Simulate a daily seasonal series as described in Ollech (2021).

Usage

```
sim_daily(
  N,
  sd = 5,
  moving = TRUE,
  week_sd = NA,
  month_sd = NA,
  year_sd = NA,
  week_change_sd = NA,
  month_change_sd = NA,
  year_change_sd = NA,
  innovations_sd = 1,
  sa_sd = NA,
  model = list(order = c(3, 1, 1), ma = 0.5, ar = c(0.2, -0.4, 0.1)),
  beta_tau7 = 0.01,
  beta_tau31 = 0,
  beta_tau365 = 0.2,
  start = c(2020, 1),
  multiplicative = TRUE,
  extra_smooth = FALSE,
  calendar = list(which = "Easter", from = -2, to = 2),
  outlier = NULL,
  timewarping = FALSE,
  as_index = FALSE
)
```

Arguments

`N`	length in years
`sd`	Standard deviation for all seasonal factors
`moving`	Is the seasonal pattern allowed to change over time
`week_sd`	Standard deviation of the seasonal factor for day-of-the-week
`month_sd`	Standard deviation of the seasonal factor for day-of-the-month
`year_sd`	Standard deviation of the seasonal factor for day-of-the-year
`week_change_sd`	Standard deviation of shock to seasonal factor
`month_change_sd`	Standard deviation of shock to seasonal factor
`year_change_sd`	Standard deviation of shock to seasonal factor
`innovations_sd`	Standard deviation of the innovations used in the non-seasonal regarima model
`sa_sd`	Standard deviation of the non-seasonal time series
`model`	Model for non-seasonal time series. A list.
`beta_tau7`	Persistance wrt to one year/cycle before of the seasonal change for day-of-the-week
`beta_tau31`	Persistance wrt to one year/cycle before of the seasonal change for day-of-the-month
`beta_tau365`	Persistance wrt to one year/cycle before of the seasonal change for day-of-the-year
`start`	Start date of output time series
`multiplicative`	Boolean. Should multiplicative seasonal factors be simulated
`extra_smooth`	Boolean. Should the seasonal factors be smooth on a period-by-period basis
`calendar`	Parameters for calendar effect, a list, see sim_calendar
`outlier`	Parameters for outlier effect, a list, see sim_outlier
`timewarping`	Should timewarping be used to obtain the day-of-the-month factors
`as_index`	Shall series be made to look like an index (i.e. shall values be relative to reference year = second year)

Details

Standard deviation of the seasonal factor is in percent if a multiplicative time series model is assumed. Otherwise it is in unitless. Using a non-seasonal ARIMA model for the initialization of the seasonal factor does not impact the seasonality of the time series. It can just make it easier for human eyes to grasp the seasonal nature of the series. The definition of the ar and ma parameter needs to be inline with the chosen model. The parameters that can be set for calendar and outlier are those defined in sim_outlier and sim_calendar.

Value

Multiple simulated daily time series of class xts including:

original The original series

seas_adj The original series without calendar and seasonal effects

sfac7 The day-of-the-week effect

sfac31 The day-of-the-month effect

sfac365 The day-of-the-year effect

cfac The calendar effects

outlier The outlier effects

Author(s)

Daniel Ollech

References

Ollech, D. (2021). Seasonal adjustment of daily time series. Journal of Time Series Econometrics. doi:10.1515/jtse20200028

Examples

```
x=sim_daily(5, sd=10, multiplicative=TRUE, outlier=list(k=5, type=c("AO", "LS")))
ts.plot(x[,1])
```

---

sim_daily_hs *Simulate a daily time series based on the HS model*

Description

This function simulates a daily time series with a Monte Carlo simulation based on an STS model based on Harvey and Shephard (1993) (HS model). The daily data consists of a trend, weekly seasonal, annual seasonal and irregular component. The components are each simulated by a transition process with daily random shocks. At the end of the simulation the components are combined and normalized to form the complete time series.

Usage

```
sim_daily_hs(
  N,
  multiplicative = TRUE,
  sizeWeeklySeas = 100,
  sizeAnnualSeas = 100,
  sizeTrend = 100,
  sizeDrift = 100,
  varIrregularity = 100,
```

```
  sizeWeeklySeasAux = 100,
  sizeAnnualSeasAux = 100,
  start = 2020,
  sizeBurnIn = 730,
  shockLevel = 1,
  shockDrift = 1,
  shockWeeklySeas = 1,
  shockAnnualSeas = 1,
  index = 100
)
```

Arguments

`N`	Length of the simulated time series in years.
`multiplicative`	If TRUE, a multiplicative model is simulated, an additive model if FALSE.
`sizeWeeklySeas`	Size and stability of the weekly seasonal factor.
`sizeAnnualSeas`	Size and stability of the annual seasonal factor.
`sizeTrend`	Size of the trend component.
`sizeDrift`	Size of the drift of the trend component.
`varIrregularity`	Variance of the random irregular component.
`sizeWeeklySeasAux`	Size of the auxiliary variable for the weekly seasonal factor.
`sizeAnnualSeasAux`	size of the auxiliary variable for the annual seasonal factor.
`start`	The initial date or year.
`sizeBurnIn`	Size of burn-in sample in days.
`shockLevel`	Variance of the shock to the level (trend).
`shockDrift`	Variance of the shock to the drift (trend).
`shockWeeklySeas`	Variance of the shock to the weekly seasonal.
`shockAnnualSeas`	Variance of the shock to the annual seasonal.
`index`	A value to which the mean of the base year (first effective year) of the time series is normalized.

Details

The size of the components and the variance of the irregular component are defaulted to 100 each and the variances of the shocks are defaulted to 1.

The first effective year serves as base year for the time series

The impact of a seasonal factor on the time series depends on its ratio to the other components. To increase (decrease) a factor's impact, the value of the size needs to be increased (decreased) while the other components need to be kept constant. Therefore, the stability of the seasonal factor also grows as the shocks on the given component have less impact. In order to increase the impact without increasing the stability, the variance of the shock also needs to be raised accordingly.

Value

Multiple simulated daily time series of class xts including:

original The original series

seas_adj The original series seasonal effects

sfac7 The day-of-the-week effect

sfac365 The day-of-the-year effect

Author(s)

Nikolas Fritz , Daniel Ollech, based on code provided by Ángel Cuevas and Enrique M Quilis

References

Cuevas, Ángel and Quilis, Enrique M., Seasonal Adjustment Methods for Daily Time Series. A Comparison by a Monte Carlo Experiment (December 20, 2023). Available at SSRN: https://ssrn.com/abstract=4670922 or http://dx.doi.org/10.2139/ssrn.4670922

Structural Time Series (STS) Monte Carlo simulation Z = trend + seasonal_weekly + seasonal_annual + irregular, according to Harvey and Shephard (1993): "Structural Time Series Models", in Maddala, G.S., Rao, C.R. and Vinod, H.D. (Eds.) Handbook of Statistics, vol. 11, Elsevier Science Publishers.

Examples

```
x <- sim_daily_hs(4)
ts.plot(x[,1])
```

`sim_daily_mstl` *Daily time series simulation for the MSTL-algorithm*

Description

This function simulates a daily time series according to the simulation model of Bandara, Hyndman and Bergmeir (2021) about the MSTL-algorithm for seasonal-trend decomposition. The simulated time series consists of a trend, weekly, annual and irregular component which are each simulated independently from each other. After the simulation process they are normalized and then combined to form the complete time series. As in the paper, this simulation function has the option to distinguish between a deterministic and a stochastic data generation process.

Usage

```
sim_daily_mstl(
  N,
  multiplicative = TRUE,
  start = 2020,
  sizeAnnualSeas = 100,
```

```
    sizeWeeklySeas = 100,
    sizeIrregularity = 100,
    shockAnnualSeas = 1,
    shockWeeklySeas = 1,
    deterministic = FALSE
  )
```

Arguments

N	length in years
multiplicative	If TRUE, a multiplicative model is simulated, if FALSE, the model is additive
start	Start year or start date of the simulation.
sizeAnnualSeas	Size of the annual seasonal factor, defaulted to 100.
sizeWeeklySeas	Size of the weekly seasonal factor, defaulted to 100.
sizeIrregularity	Size of the irregular component, defaulted to 100.
shockAnnualSeas	Shock to the annual seasonal coefficient, defaulted to 1.
shockWeeklySeas	Shock to the weekly seasonal coefficient, defaulted to 1.
deterministic	If TRUE, the seasonal coefficients are deterministic, meaning they do not change after a seasonal cycle. If FALSE, the coefficients are stochastic, meaning they change randomly after a seasonal cycle.

Value

Multiple simulated daily time series of class xts including:

original The original series

seas_adj The original series without seasonal effects

sfac7 The day-of-the-week effect

sfac365 The day-of-the-year effect

Author(s)

Nikolas Fritz, Daniel Ollech

References

Bandara, K., Hyndman, R. J., & Bergmeir, C. (2021). MSTL: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns. arXiv preprint arXiv:2107.13462.

Examples

```
x <- sim_daily_mstl(4)
ts.plot(x[,1])
```

sim_monthly *Simulate a monthly seasonal series*

Description

Simulate a monthly seasonal series

Usage

```
sim_monthly(
  N,
  sd = 5,
  change_sd = sd/10,
  beta_1 = 0.6,
  beta_tau = 0.4,
  moving = TRUE,
  model = list(order = c(3, 1, 1), ma = 0.5, ar = c(0.2, -0.4, 0.1)),
  start = c(2010, 1),
  multiplicative = TRUE,
  extra_smooth = FALSE
)
```

Arguments

N	Length in years
sd	Standard deviation for all seasonal factors
change_sd	Standard deviation of shock to seasonal factor
beta_1	Persistance wrt to previous period of the seasonal change
beta_tau	Persistence wrt to one year/cycle of the seasonal change
moving	Is the seasonal pattern allowed to change over time
model	Model for non-seasonal time series. A list.
start	Start date of output time series
multiplicative	Boolean. Should multiplicative seasonal factors be simulated
extra_smooth	Boolean. Should the seasonal factors be smooth on a period-by-period basis

Details

Standard deviation of the seasonal factor is in percent if a multiplicative time series model is assumed. Otherwise it is in unitless. Using a non-seasonal ARIMA model for the initialization of the seasonal factor does not impact the seasonality of the time series. It can just make it easier for human eyes to grasp the seasonal nature of the series. The definition of the ar and ma parameter needs to be inline with the chosen model.

Value

Multiple simulated monthly time series of class xts including:

original The original series

seas_adj The original series without seasonal effects

sfac The seasonal effect

Author(s)

Daniel Ollech

References

Ollech, D. (2021). Seasonal adjustment of daily time series. Journal of Time Series Econometrics. doi:10.1515/jtse20200028

Examples

```
x=sim_monthly(5, multiplicative=TRUE)
ts.plot(x[,1])
```

`sim_monthly_hs` *Simulate a monthly time series based on the HS model*

Description

This function simulates a monthly time series with a Monte Carlo simulation based on an STS model based on Harvey and Shephard (1993) (HS model). The monthly data consists of a trend, annual seasonal and irregular component. The components are each simulated by a transition process with monthly random shocks and then combined at the end of the simulation to form the complete time series.

Usage

```
sim_monthly_hs(
  N,
  multiplicative = TRUE,
  sizeSeasonality = 100,
  sizeTrend = 100,
  sizeDrift = 100,
  sizeSeasonalityAux = 100,
  varIrregularity = 1,
  start = 2020,
  sizeBurnIn = 24,
  shockLevel = 1,
  shockDrift = 1,
  shockSeasonality = 1,
  index = 100
)
```

Arguments

- `N` Length of the simulated time series in years.
- `multiplicative` If true, a multiplicative model is simulated, an additive model if FALSE.
- `sizeSeasonality` Size and stability of the annual seasonal factor.
- `sizeTrend` Size and stability of the trend component.
- `sizeDrift` Size and stability of the drift of the trend component.
- `sizeSeasonalityAux` Size of the auxiliary variable for the annual seasonal factor.
- `varIrregularity` Variance of the random irregular component.
- `start` The initial date or year.
- `sizeBurnIn` Size of burn-in sample in months.
- `shockLevel` Variance of the shock to the level (trend).
- `shockDrift` Variance of the shock to the drift (trend).
- `shockSeasonality` Variance of the shock to the annual seasonal.
- `index` A value to which the mean of the base year (first effective year) of the time series is normalized.

Details

The impact of a component on the time series depends on its ratio to the other components. To increase (decrease) a component's impact, the value of the size needs to be increased (decreased) while the other components need to be kept constant. Therefore, the stability of the component (e.g. the shape of a seasonal component) also grows as the shocks on the given component have less impact. In order to increase the impact without increasing the stability, the variance of the shock also needs to be raised accordingly. The size of the components are defaulted to 100 each and the variances of the shocks are defaulted to 1.

The first effective year serves as base year for the time series

Value

Multiple simulated monthly time series of class xts including:

original The original series

seas_adj The original series without seasonal effects

sfac The seasonal effect

Author(s)

Nikolas Fritz, Daniel Ollech, based on code provided by Ángel Cuevas and Enrique M Quilis

References

Cuevas, Ángel and Quilis, Enrique M., Seasonal Adjustment Methods for Daily Time Series. A Comparison by a Monte Carlo Experiment (December 20, 2023). Available at SSRN: https://ssrn.com/abstract=4670922 or http://dx.doi.org/10.2139/ssrn.4670922

Structural Time Series (STS) Monte Carlo simulation Z = trend + seasonal_weekly + seasonal_annual + irregular, according to Harvey and Shephard (1993): "Structural Time Series Models",in Maddala, G.S., Rao, C.R. and Vinod, H.D. (Eds.) Handbook of Statistics, vol. 11, Elsevier Science Publishers.

Examples

```
x <- sim_monthly_hs(4)
ts.plot(x[,1])
```

---

sim_monthly_mstl *Monthly time series simulation for the MSTL-algorithm*

Description

This function simulates a monthly time series according to the simulation model of Bandara, Hyndman and Bergmeir (2021) about the MSTL-algorithm for seasonal-trend decomposition. The simulated time series consists of a trend, annual seasonal and irregular component which are each simulated independently from each other. After the simulation process they are normalized and then combined to form the complete time series. As in the paper, this simulation function has the option to distinguish between a deterministic and a stochastic data generation process.

Usage

```
sim_monthly_mstl(
  N,
  multiplicative = TRUE,
  start = 2020,
  sizeSeasonality = 100,
  sizeIrregularity = 100,
  sizeTrend = 100,
  shockSeasonality = 1,
  deterministic = FALSE
)
```

Arguments

`N`	length in years
`multiplicative`	If TRUE, a multiplicative model is simulated, if FALSE, the model is additive
`start`	Start year or start date of the simulation.
`sizeSeasonality`	Size of the annual seasonal factor.

`sizeIrregularity`
: Size of the irregular component.

`sizeTrend`
: Size of trend component.

`shockSeasonality`
: Variance of the shock to the annual seasonal coefficient, defaulted to 1.

`deterministic`
: If TRUE, the seasonal coefficients are deterministic, meaning they do not change after a seasonal cycle. If FALSE, the coefficients are stochastic, meaning they change by random shocks after a seasonal cycle.

Value

Multiple simulated monthly time series of class xts including:

original The original series

seas_adj The original series without seasonal effects

sfac The seasonal effect

Author(s)

Nikolas Fritz, Daniel Ollech

References

Bandara, K., Hyndman, R. J., & Bergmeir, C. (2021). MSTL: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns. arXiv preprint arXiv:2107.13462.

Examples

```
x <- sim_monthly_mstl(4)
ts.plot(x[,1])
```

`sim_outlier` *Simulate an outlier*

Description

Simulate an outlier

Usage

```
sim_outlier(
  n,
  k,
  freq = 12,
  type = c("AO", "LS", "TC"),
  effect_size = 10,
  start = c(2020, 1),
  multiplicative = TRUE
)
```

Arguments

`n`	Time series length
`k`	Number of outliers
`freq`	Frequency of the time series
`type`	Type of outlier
`effect_size`	Mean size of outlier
`start`	Start date of output time series
`multiplicative`	Boolean. Is multiplicative time series model assumed?

Details

Three types of outliers are implemented: AO=Additive outlier, LS=Level shift, TC=Temporary Change. The effect size is stochastic as it is drawn from a normal distribution with mean equal to the specified effect_size and a standard deviation of 1/4*effect_size. This is multiplied randomly with -1 or 1 to get negative shocks as well. If multiplicative is true, the effect size is measured in percentage. If is not true, the effect size is unit less and thus adopts the unit of the time series the outliers are added to.

Value

The function returns k time series of class `xts` containing the k outlier effects

Author(s)

Daniel Ollech

References

Ollech, D. (2021). Seasonal adjustment of daily time series. Journal of Time Series Econometrics. doi:10.1515/jtse20200028

Examples

```
plot(sim_outlier(60, 4, type=c("AO", "LS")))
```

`sim_sfac` *Simulate a seasonal factor*

Description

Simulate a seasonal factor

Usage

```
sim_sfac(
  n,
  freq = 12,
  sd = 1,
  change_sd = sd/10,
  moving = TRUE,
  beta_1 = 0.6,
  beta_tau = 0.4,
  start = c(2020, 1),
  multiplicative = TRUE,
  ar = NULL,
  ma = NULL,
  model = c(1, 1, 1),
  sc_model = list(order = c(1, 1, 1), ar = 0.65, ma = 0.25),
  smooth = TRUE,
  burnin = 7,
  extra_smooth = FALSE
)
```

Arguments

`n`	Number of observations
`freq`	Frequency of the time series
`sd`	Standard deviation of the seasonal factor
`change_sd`	Standard deviation of shock to seasonal factor
`moving`	Is the seasonal pattern allowed to change over time
`beta_1`	Persistence wrt to previous period of the seasonal change
`beta_tau`	Persistence wrt to one year/cycle of the seasonal change
`start`	Start date of output time series
`multiplicative`	Boolean. Should multiplicative seasonal factors be simulated
`ar`	AR parameter
`ma`	MA parameter
`model`	Model for initial seasonal factor
`sc_model`	Model for the seasonal change
`smooth`	Boolean. Should initial seasonal factor be smoothed
`burnin`	(burnin*n-n) is the burn-in period
`extra_smooth`	Boolean. Should the seasonal factor be smoothed on a period-by-period basis

Details

Standard deviation of the seasonal factor is in percent if a multiplicative time series model is assumed. Otherwise it is in unitless. Using a non-seasonal ARIMA model does not impact the seasonality of the time series. It can just make it easier for human eyes to grasp the seasonal nature of the series. The definition of the ar and ma parameter needs to be in line with the chosen model.

Value

The function returns a time series of class `ts` containing a seasonal or periodic effect.

Author(s)

Daniel Ollech

References

Ollech, D. (2021). Seasonal adjustment of daily time series. Journal of Time Series Econometrics. doi:10.1515/jtse20200028

Examples

```
ts.plot(sim_sfac(60))
```

Index