# Package: tssim (via r-universe)

October 1, 2024

**Title** Simulation of Daily and Monthly Time Series

**Version** 0.1.7

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

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

**License** GPL-3

**Depends** R (>= 3.1.0)

**Encoding** UTF-8

**LazyData** false

**RoxygenNote** 7.1.1

**Imports** utils, xts, zoo, timeDate, stats, dsa, tsbox

**NeedsCompilation** no

**Author** Daniel Ollech [aut, cre]

**Repository** CRAN

**Date/Publication** 2021-06-25 11:20:02 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 = 2.5,
  change_sd = 0.05,
  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_1 = 0.9,
  beta_tau = 0,
  start = c(2020, 1),
  multiplicative = TRUE,
  extra_smooth = FALSE,
  calendar = list(which = "Easter", from = -2, to = 2),
  outlier = NULL,
  timewarping = TRUE,
  as_index = FALSE
)
```

## Arguments

`N` length in years

`sd` Standard deviation for all seasonal factors

`change_sd` Standard deviation of simulated change for all seasonal factors

`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 simulated change to seasonal factor for day-of-the-week

`month_change_sd` Standard deviation of simulated change to seasonal factor for month-of-the-week

`year_change_sd` Standard deviation of simulated change to seasonal factor for year-of-the-week

`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_1` Persistance wrt to previous period of the seasonal change

`beta_tau` Persistance wrt to one year/cycle before of the seasonal change

`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. If only change_sd is specified, the change parameters for the single seasonal factors are set individually as change_sd/365*(length of seasonal cycle) 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

**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, multiplicative=TRUE, outlier=list(k=5, type=c("AO", "LS"), effect_size=50))
ts.plot(x[,1])
```

## sim_monthly *Simulate a monthly seasonal series*

### Description

Simulate a monthly seasonal series

### Usage

```
sim_monthly(
  N,
  sd = 1,
  beta_1 = 0.9,
  change_sd = 0.025,
  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 |
| beta_1 | Persistance wrt to previous period of the seasonal change |
| change_sd | Standard deviation of simulated change for all seasonal factors |
| 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_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 = 0.02,
  beta_1 = 0.9,
  beta_tau = 0,
  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 = 3,
  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 simulation change to seasonal factor |
| `beta_1` | Persistance wrt to previous period of the seasonal change |
| `beta_tau` | Persistance wrt to one year/cycle before 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 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 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

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