Package 'boiwsa'

February 5, 2025

Type Package

Title Seasonal Adjustment of Weekly Data

Version 1.1.3

Maintainer Tim Ginker <tim.ginker@gmail.com>

Description Perform seasonal adjustment of weekly data. The package provides a user-friendly interface for computing seasonally adjusted estimates of weekly data and includes functions for the creation of country-specific prior adjustment variables, as well as diagnostic tools to assess the quality of the adjustments. The method is described in more detail in Ginker (2023) <doi:10.13140/RG.2.2.12221.44000>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, forecast, ggplot2, Hmisc, lubridate, stats, tidyr, rlang, gridExtra

LazyData true

Depends R (>= 2.10)

URL https://github.com/timginker/boiwsa

BugReports https://github.com/timginker/boiwsa/issues

NeedsCompilation no

Author Tim Ginker [aut, cre, cph] (<https://orcid.org/0000-0002-7138-5417>), Jon Lachman [ctb]

Repository CRAN

Date/Publication 2025-02-05 12:30:15 UTC

Contents

`boiwsa` *Seasonal adjustment of weekly data*

Description

Performs seasonal adjustment of weekly data. For more details on the usage of this function see the paper or the examples on Github.

Usage

```
boiwsa(
  x,
  dates,
  r = 0.8,
  auto.ao.search = TRUE,
  out.threshold = 3.8,
  ao.list = NULL,
  my.k_l = NULL,
  H = NULL,
  ic = "aicc",
  method = "additive"
)
```

Arguments

`x`	Input time series as a numeric vector
`dates`	a vector of class "Date", containing the data dates
`r`	Defines the rate of decay of the weights. Should be between zero and one. By default is set to 0.8.

auto.ao.search	Boolean. Search for additive outliers
out.threshold	t-stat threshold in outlier search. By default is 3.8
ao.list	Vector with user specified additive outliers in a date format
my.k_l	Numeric vector defining the number of yearly and monthly trigonometric variables. If NULL, is found automatically using the information criteria. The search range is 0:36 and 0:12 with the step size of 6 for the yearly and monthly variables, respectively.
H	Matrix with holiday- and trading day factors
ic	Information criterion used in the automatic search for the number of trigonometric regressors. There are thee options: aic, aicc and bic. By default uses aicc
method	Decomposition type: additive or multiplicative

Value

sa Seasonally adjusted series

my.k_l Number of trigonometric variables used to model the seasonal pattern

sf Estimated seasonal effects

hol.factors Estimated holiday effects

out.factors Estimated outlier effects

beta Regression coefficients for the last year

m lm object. Unweighted OLS regression on the full sample

Author(s)

Tim Ginker

Examples

```
# Not run
# Seasonal adjustment of weekly US gasoline production

data("gasoline.data")
res=boiwsa(x=gasoline.data$y,dates=gasoline.data$date)
```

---

dates_il *Israeli working dates*

Description

Israeli working dates

Usage

```
dates_il
```

Format

A data frame with 21550 rows and 4 variables:

`DATE_VALUE` Date

`ISR_WORKING_DAY_PART` 1: full working day, 0.5: half working day, 0: holiday

`JEWISH_FULL_DATE` Jewish date

`DATE_WEEK_NUMBER` Weekday

Source

Personal

`find_opt` *Find optimal number of fourier variables*

Description

Searches through the model space to identify the best number of trigonometric variables, with the lowest AIC, AICc or BIC value.

Usage

```
find_opt(
  x,
  dates,
  H = NULL,
  AO = NULL,
  method = "additive",
  l.max = 12,
  k.max = 42,
  by = 6
)
```

Arguments

`x`	Numeric vector. Time series to seasonally adjust
`dates`	a vector of class "Date", containing the data dates
`H`	(optional) Matrix with holiday and trading day variables
`AO`	(optional) Matrix with additive outlier variables
`method`	Decomposition method: "additive" or "multiplicative". By default uses the additive method
`l.max`	Maximal number of the monthly cycle variables to search for. By default is 12
`k.max`	Maximal number of the yearly cycle variables to search for. By default is 42
`by`	Step size in the search. By default is 6.

Value

list with the optimal number of (yearly and monthly) fourier variables according to AIC, AICc and BIC

Examples

```
data(gasoline.data)

res=find_opt(x=gasoline.data$y,dates=gasoline.data$date)
print(res)
```

`find_outliers` *Find additive outliers*

Description

Searches for additive outliers using the method described in Appendix C of Findley et al. (1998). If the number of trigonometric variables is not specified will search automatically through the model space to identify the best number of trigonometric variables, with the lowest AIC, AICc or BIC value.

Usage

```
find_outliers(
  x,
  dates,
  out.tolerance = 3.8,
  my.AO.list = NULL,
  H = NULL,
  my.k_l = NULL,
  method = "additive"
)
```

Arguments

`x`	Numeric vector. Time series to seasonally adjust
`dates`	a vector of class "Date", containing the data dates
`out.tolerance`	t-stat threshold for outliers (see Findley et al., 1998)
`my.AO.list`	(optional) Vector with user defined additive outlier variables
`H`	(optional) Matrix with holiday and trading day variables
`my.k_l`	(optional) Vector with the number of fourier terms to capture the yearly and monthly cycle. If NULL, would perform automatic search using AICc criterion
`method`	Decomposition method: "additive" or "multiplicative". By default uses the additive method

Value

my.k_l

ao list of AO dates

References

Findley, D.F., Monsell, B.C., Bell, W.R., Otto, M.C. and B.C Chen (1998). New capabilities and methods of the X-12-ARIMA seasonal-adjustment program. Journal of Business & Economic Statistics, 16(2), pp.127-152.

Examples

```
#Not run:
# Searching for additive outliers in Gasoline data
data(gasoline.data)
ao_list=find_outliers(x=gasoline.data$y,dates = gasoline.data$date)
```

---

fourier_vars *Create fourier predictors*

Description

Creates sine and cosine variables to capture intramonthly and intrayearly cycles.

Usage

```
fourier_vars(k = 1, l = 1, dates)
```

Arguments

k	Number of pairs of the yearly cycle trigonometric variables
l	Number of pairs of the monthly cycle trigonometric variables
dates	Vector of dates in a date format

Value

Matrix with fourier variables

Examples

```
# create a vector of dates
dates=seq.Date(from=as.Date("2023-01-02"),by="weeks",length.out = 100)
# Create a matrix with 20 yearly and 6 monthly pairs of sine and cosine variables
X=fourier_vars(k=20,l=6,dates=dates)
```

gasoline.data *US finished motor gasoline product supplied*

Description

Weekly data beginning 2 February 1991, ending 20 January 2017. Units are "million barrels per day".

Usage

```
gasoline.data
```

Format

Data.Frame:
A data frame with 1355 rows and 2 columns:

date date in a date format
y gasoline consumption

Source

Originally from the US Energy Information Administration. Copied from the fpp2 package.

genhol *Generate Holiday Regression Variables*

Description

Can be used to generate moving holiday regressors for the U. S. holidays of Easter, Labor Day, and Thanksgiving; or for Israeli Rosh Hashanah and Pesach. The variables are computed using the Easter formula in Table 2 of Findley et al. (1998). Uses calendar centring to avoid bias.

Usage

```
genhol(dates, holiday.dates, start = 7, end = 7)
```

Arguments

dates	a vector of class "Date", containing the data dates
holiday.dates	a vector of class "Date", containing the occurrences of the holiday. It can be generated with as.Date().
start	integer, shifts backwards the start point of the holiday. Use negative values if start is after the specified date.
end	integer, shifts end point of the holiday. Use negative values if end is before the specified date.

Value

a matrix with holiday variables that can be used as a user defined variable in boiwsa().

References

Findley, D.F., Monsell, B.C., Bell, W.R., Otto, M.C. and B.C Chen (1998). New capabilities and methods of the X-12-ARIMA seasonal-adjustment program. Journal of Business & Economic Statistics, 16(2), pp.127-152.

Examples

```
# Creating moving holiday variable for Israeli Rosh Hashanah
data(gasoline.data)
data(holiday_dates_il) # dates of Israeli Rosh Hashanah and Pesach
movehol=genhol(gasoline.data$date,holiday.dates = holiday_dates_il$rosh)
```

---

holiday_dates_il *Israeli moving holiday dates*

Description

Rosh Hashanah and Pesach dates

Usage

```
holiday_dates_il
```

Format

A data frame with 51 rows and 3 variables:

`year` Year

`rosh` Rosh Hashanah date

`pesah` Pesach date

Source

Personal

lbm — *Weekly number of initial registrations in Israeli Employment Services (adjusted for strikes)*

Description

Weekly data beginning 11 January 2014, ending 4 January 2020.

Usage

```
lbm
```

Format

Data.Frame:
A data frame with 313 rows and 2 columns:

date date in a date format
IES_IN_W_ADJ number of initial registrations

Source

Internal

my_ao — *Create additive outlier variables*

Description

Creates a matrix with additive outlier variables. Uses the original data dates and the user specified outlier dates.

Usage

```
my_ao(dates, out.list)
```

Arguments

dates	Vector of dates in a date format
out.list	Vector of outlier dates in a date format

Value

AO matrix with outlier variables

Examples

```
# create a sequence of dates
dates=seq.Date(from=as.Date("2023-01-02"),by="weeks",length.out = 100)
# create a vector of outlier dates
my_ao_dates=as.Date(c("2023-01-02","2023-01-03"))
# create a matrix of AO variables
my_ao(dates = dates,out.list = my_ao_dates)
# as you can see there is only one column corresponding to 2023-01-02,
# the second date is ignored because it is not present in the dates vector
```

---

my_rosh *Internal function for a specific application*

Description

Creates a dummy moving holiday variable for the weekly number of initial registrations at the Employment Service in Israel.

Usage

```
my_rosh(dates, holiday.dates, start = -11, end = 12)
```

Arguments

dates	a vector of class "Date", containing the data dates
holiday.dates	a vector of class "Date", containing the occurrences of the holiday. It can be generated with as.Date().
start	-11 for rosh, 3 for pesach
end	12 for rosh, -1 for pesach

Value

rosh holiday variable

Examples

```
# Creating moving holiday dummy variable for Israeli Rosh Hashanah
data(gasoline.data)
data(holiday_dates_il) # dates of Israeli Rosh Hashanah and Pesach
movehol=my_rosh(gasoline.data$date,holiday.dates = holiday_dates_il$rosh)
```

plot.boiwsa *Plot*

Description

S3 method for objects of class "boiwsa". Produces a ggplot object of seasonally decomposed time series.

Usage

```
## S3 method for class 'boiwsa'
plot(x, ...)
```

Arguments

x	Result of boiwsa
...	Additional arguments (currently not used).

plot_spec *Original and SA data AR spectrum*

Description

AR spectrum of the (detrended) original and seasonally adjusted data. Computed using `stats::spec.ar()` with order set to 60.

Usage

```
plot_spec(x)
```

Arguments

x	boiwsa results

Value

AR spectrum plot

Examples

```
# Not run
# Seasonal adjustment of weekly US gasoline production
res=boiwsa(x=gasoline.data$y,dates=gasoline.data$date)
plot_spec(res)
```

predict.boiwsa *Predict*

Description

S3 method for 'boiwsa' class. Returns forecasts and other information using a combination of nonseasonal auto.arima and estimates from boiwsa.

Usage

```
## S3 method for class 'boiwsa'
predict(object, ...)
```

Arguments

object	An object of class boiwsa.
...	Additional arguments: • n.ahead: Number of periods for forecasting (required). • level: Confidence level for prediction intervals. By default is set to c(80, 95). • new_H: Matrix with future holiday- and trading day factors. • arima.options: List of forecast::Arima arguments for custom modeling.

Value

A list containing the forecast values and ARIMA fit.

print *Generic print function*

Description

This is the generic print function.

Usage

```
print(x, ...)
```

Arguments

x	An object to print.
...	Additional arguments (currently not used).

print.boiwsa *Print method for boiwsa objects*

Description

S3 method for objects of class boiwsa. Prints a short model summary including the number of trigonometric terms and the position of outliers.

Usage

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

Arguments

x	Result of boiwsa.
...	Additional arguments (currently not used).

simple_td *Generate simple working day variable*

Description

Aggregates the count of full working days within a week and normalizes it.

Usage

```
simple_td(dates, df.td)
```

Arguments

dates	a vector of class "Date", containing the data dates
df.td	dataframe with working days. Its should consit of 2 columns named as "date" and "WORKING_DAY_PART". date column should be of class "Date". WORKING_DAY_PART should be similar to ISR_WORKING_DAY_PART in dates_il

Value

matrix with trading day variables

Examples

```
library(dplyr)
data(dates_il)
data(gasoline.data)

dates_il%>%
  dplyr::select(DATE_VALUE,ISR_WORKING_DAY_PART)%>%
  `colnames<-`(c("date","WORKING_DAY_PART"))%>%
  dplyr::mutate(date=as.Date(date))->df.td

td=simple_td(dates = gasoline.data$date,df.td = df.td)
```

`summary` *Generic summary function*

Description

This is the generic summary function.

Usage

```
summary(object, ...)
```

Arguments

`object`	An object to summarize.
`...`	Additional arguments (currently not used).

`summary.boiwsa` *Summary function*

Description

S3 method for objects of class "boiwsa". Prints the regression summary output.

Usage

```
## S3 method for class 'boiwsa'
summary(object, ...)
```

Arguments

`object`	An object of class `boiwsa`.
`...`	Additional arguments (currently not used).

Index