Package 'boiwsa'

December 14, 2025

Type Package

Title Seasonal Adjustment of Weekly Data

Version 1.1.4

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

Description Perform seasonal adjustment and forecasting of weekly data. The package provides a user-friendly interface for computing seasonally adjusted estimates and forecasts of weekly time series and includes functions for the construction of country-specific prior adjustment variables, as well as diagnostic tools to assess the quality of the adjustments. The methodology is described in more detail in Ginker (2024) <doi:10.13140/RG.2.2.12221.44000>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

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] (ORCID: <https://orcid.org/0000-0002-7138-5417>), Jon Lachman [ctb]

Repository CRAN

Date/Publication 2025-12-14 10:30:01 UTC

Contents

`boiwsa` *Seasonal adjustment of weekly data*

Description

Performs seasonal adjustment and forecasting of weekly time series using a regression-based decomposition framework estimated by discounted least squares. Seasonality is modeled using trigonometric regressors, while holiday, trading-day, and calendar effects are incorporated through additional covariates. Additive outliers can be detected automatically using an optional outlier search procedure. The function supports both additive and multiplicative decompositions and produces seasonally adjusted series together with the associated model components. Worked examples and additional usage illustrations are available in the package repository on GitHub. See Ginker (2024), *boiwsa: An R Package for Seasonal Adjustment of Weekly Data*, *The R Journal*, 16(3), 186–197.

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	Numeric vector containing the observed weekly time series.
dates	A vector of class "Date" corresponding to the observation dates.
r	Numeric scalar in (0, 1] defining the rate of decay of the observation weights. Defaults to 0.8.
auto.ao.search	Logical. If TRUE, additive outliers are detected automatically.
out.threshold	Numeric. t-statistic threshold used in the additive outlier search. Defaults to 3.8.
ao.list	Optional vector of class "Date" specifying user-defined additive outlier dates.
my.k_l	Optional numeric vector of length two specifying the number of yearly and monthly trigonometric variables. If NULL, these are selected 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	Optional matrix of holiday and trading-day regressors with the same number of rows as x.
ic	Character string specifying the information criterion used in the automatic selection of trigonometric regressors. One of "aic", "aicc", or "bic". Defaults to "aicc".
method	Character string specifying the decomposition type. Either "additive" or "multiplicative".

Details

The methodological framework implemented in this function is described in Ginker (2024), *boiwsa: An R Package for Seasonal Adjustment of Weekly Data*, *The R Journal*, 16(3), 186–197.

Value

A list with the following components:

sa Seasonally adjusted series.

my.k_l Number of trigonometric regressors used to model seasonality.

sf Estimated seasonal component.

hol.factors Estimated holiday and trading-day effects.

out.factors Estimated additive outlier effects.

beta Regression coefficients estimated for the last year of data.

m Unweighted lm object estimated 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

Performs a grid search over combinations of yearly and monthly Fourier (trigonometric) regressors and selects the number of terms that minimizes AIC, AICc, or BIC. Candidate models are fitted by OLS to a detrended series, where the trend is estimated using `supsmu`. Optional holiday/trading-day regressors (`H`) and additive-outlier regressors (`AO`) are included in every candidate specification if provided.

Usage

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

Arguments

x	Numeric vector containing the observed weekly time series.
dates	A vector of class "Date" corresponding to the observation dates.
H	Optional matrix of holiday and trading-day regressors with nrow(H) = length(x).
AO	Optional matrix of additive-outlier regressors with nrow(AO) = length(x).
method	Character string specifying the decomposition type. Either "additive" or "multiplicative". If "multiplicative", the series is log-transformed prior to detrending. Defaults to "additive".
l.max	Integer. Maximum number of monthly-cycle Fourier harmonics to consider. Defaults to 12.
k.max	Integer. Maximum number of yearly-cycle Fourier harmonics to consider. Defaults to 42.
by	Integer. Step size for the grid search over k and l. Defaults to 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` *Detect additive outliers in weekly time series*

Description

Detects additive outliers (AOs) using a regression-based t-statistic search procedure following Findley et al. (1998). The function operates on a detrended series, where the trend is estimated using `supsmu`. Optional holiday and trading-day regressors can be included. If the number of Fourier (trigonometric) terms is not supplied via `my.k_l`, it is selected automatically by minimizing AICc over a grid of yearly and monthly Fourier terms.

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 containing the observed weekly time series.
`dates`	A vector of class `"Date"` corresponding to the observation dates.
`out.tolerance`	Numeric. Absolute t-statistic threshold used for AO inclusion. Defaults to `3.8`.
`my.AO.list`	Optional vector of class `"Date"` specifying pre-defined AO dates. These dates are included in the regression and excluded from the forward search.
`H`	Optional matrix of holiday and trading-day regressors with `nrow(H) = length(x)`.
`my.k_l`	Optional numeric vector of length two specifying the number of yearly and monthly Fourier harmonics `c(k, l)`. If `NULL`, `my.k_l` is selected automatically by AICc.
`method`	Character string specifying the decomposition type. Either `"additive"` or `"multiplicative"`. If `"multiplicative"`, the series is log-transformed prior to detrending. Defaults to `"additive"`.

Value

A list with the following components:

ao Vector of class `"Date"` containing detected additive outlier dates, or `NULL` if none are detected.

my.k_l Numeric vector `c(k, l)` giving the number of yearly and monthly Fourier terms used in the regression.

References

Findley, D.F., Monsell, B.C., Bell, W.R., Otto, M.C. and Chen, B.C. (1998). New capabilities and methods of the X-12-ARIMA seasonal-adjustment program. *Journal of Business and Economic Statistics*, 16(2), 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 (trigonometric) regressors for weekly data*

Description

Constructs sine and cosine regressors to capture seasonal variation at intrayearly and intramonthly frequencies in weekly time series. The Fourier terms are defined using the day-of-year and day-of-month corresponding to each observation date, allowing the seasonal frequencies to adapt to varying month and year lengths.

Usage

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

Arguments

k	Integer. Number of yearly-cycle Fourier harmonics (pairs of sine and cosine terms) to include.
l	Integer. Number of monthly-cycle Fourier harmonics (pairs of sine and cosine terms) to include.
dates	A vector of class "Date" corresponding to the observation dates.

Value

A numeric matrix with `length(dates)` rows and `2 * (k + l)` columns containing the Fourier regressors. Columns are ordered with yearly terms first, followed by monthly terms. If both `k = 0` and `l = 0`, `NULL` is returned.

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 a moving-holiday regressor for weekly data*

Description

Generates moving-holiday regressors for weekly data based on supplied holiday occurrence dates using the Easter formula described in Table 2 of Findley et al. (1998). The function can be used to construct regressors for U.S. holidays such as Easter, Labor Day, and Thanksgiving, as well as for Israeli holidays such as Rosh Hashanah and Pesach. The resulting weekly holiday regressors are calendar-centered to avoid bias.

Usage

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

Arguments

dates	A vector of class "Date" corresponding to the weekly observation dates.
holiday.dates	A vector of class "Date" giving the holiday occurrence dates (e.g., Easter, Labor Day, Thanksgiving, Rosh Hashanah, Pesach). Dates outside the range of dates are ignored.
start	Integer. Number of days before each holiday date to include in the moving-holiday window. Negative values may be used to shift the start of the window to dates after the holiday.
end	Integer. Number of days after each holiday date to include in the moving-holiday window. Negative values may be used to shift the end of the window to dates before the holiday.

Value

A data frame with two columns:

date Weekly dates corresponding to dates.

moving_holiday Calendar-centered moving-holiday regressor at weekly frequency.

The returned object can be merged into a matrix of holiday or trading-day regressors supplied to boiwsa() via the H argument.

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

```
# Moving-holiday regressor 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 indicator variables*

Description

Constructs a matrix of additive outlier (AO) indicator variables based on a set of user-specified outlier dates. For each outlier date that coincides with an observation date, a binary indicator equal to one is created at the corresponding position and zero elsewhere. Outlier dates not present in the observation dates are silently ignored.

Usage

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

Arguments

dates	A vector of class "Date" corresponding to the observation dates.
out.list	A vector of class "Date" specifying candidate additive outlier dates.

Value

A numeric matrix with length(dates) rows and one column per outlier date present in dates. Column names are of the form "AO <date>". If none of the supplied outlier dates coincide with dates, NULL is returned.

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 *Compare AR spectra of original and seasonally adjusted series*

Description

Computes and plots autoregressive (AR) spectral density estimates for the detrended original series and its seasonally adjusted counterpart. Spectra are estimated using `spec.ar` with AR order set to 60. The plot highlights frequencies corresponding to intramonthly and intrayearly cycles.

Usage

```
plot_spec(x)
```

Arguments

`x`	A `boiwsa()` result object. Must contain components `x`, `sa`, and `trend`.

Value

A `ggplot2` object showing the AR spectral density estimates for the detrended original and seasonally adjusted series.

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 a simple working-day trading-day regressor*

Description

Constructs a weekly trading-day regressor by counting the number of full working days within each weekly period and centering the resulting series by subtracting its sample mean. Daily working-day information is supplied via `df.td` and mapped to the weekly dates provided in `dates`.

Usage

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

Arguments

`dates`	A vector of class `"Date"` corresponding to the weekly observation dates.
`df.td`	A data frame containing daily working-day information with two columns: `date` (class `"Date"`) and `WORKING_DAY_PART`. Full working days should be coded as `1`; all other values are treated as non-working days.

Value

A data frame with two columns:

date Weekly dates corresponding to `dates`.

td Centered weekly count of full working days.

The returned object can be merged into a matrix of holiday/trading-day regressors supplied to `boiwsa()` via the `H` argument.

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