Package: ggdemetra (via r-universe)

October 21, 2026

Type Package

Title 'ggplot2' Extension for Seasonal and Trading Day Adjustment with 'RJDemetra'

Version 0.2.9

Description Provides 'ggplot2' functions to return the results of seasonal and trading day adjustment made by 'RJDemetra'. 'RJDemetra' is an 'R' interface around 'JDemetra+' (<https://github.com/jdemetra/jdemetra-app>), the seasonal adjustment software officially recommended to the members of the European Statistical System and the European System of Central Banks.

Depends R (>= 3.1.2), ggplot2 (>= 2.0.0), RJDemetra (>= 0.1.2),

Imports ggrepel, gridExtra

Suggests knitr, rmarkdown

SystemRequirements Java (>= 8)

License EUPL

URL https://aqlt.github.io/ggdemetra/, https://github.com/AQLT/ggdemetra

BugReports https://github.com/AQLT/ggdemetra/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Alain Quartier-la-Tente [aut, cre] (ORCID: <https://orcid.org/0000-0001-7890-3857>)

Maintainer Alain Quartier-la-Tente <alain.quartier@yahoo.fr>

Config/pak/sysreqs default-jdk

Repository https://cran.r-universe.dev

Date/Publication 2025-06-25 08:45:16 UTC

RemoteUrl https://github.com/cran/ggdemetra

RemoteRef HEAD

RemoteSha 14793b59c605bf716c83d2fa297a3f6ffd044ccd

Contents

autoplot.SA 2
components 3
geom_arima 3
geom_diagnostics 5
geom_outlier 7
geom_sa 9
init_ggplot 11
ipi_c_eu 12
siratio 13
ts2df 15

Index **16**

`autoplot.SA` *Plot 'RJDemetra' model*

Description

Plot 'RJDemetra' model

Usage

```
## S3 method for class 'SA'
autoplot(
  object,
  components = c("y", "sa", trend = "t", seasonal = "s", irregular = "i"),
  forecast = FALSE,
  ...
)
```

Arguments

`object`	a `"SA"` or `"jSA"` model.
`components`	components to print, can be `"y"` (input time series), `"sa"` (seasonal adjusted), `"t"` (trend-cycle), `"y_cal"` (calendar adjusted), `"s"` (seasonal), `"i"` (irregular), `"cal"` (calendar). The vector can be named to change the label
`forecast`	boolean indicating if the forecast series should be printed.
`...`	unused arguments.

Examples

```
x = RJDemetra::jx13(ipi_c_eu[,"FR"])
ggplot2::autoplot(x)
```

`components` *Extract Component from 'RJDemetra' model*

Description

Extract Component from 'RJDemetra' model

Usage

```
seasonal(x, forecast = FALSE)

trendcycle(x, forecast = FALSE)

irregular(x, forecast = FALSE)

seasonaladj(x, forecast = FALSE)

calendaradj(x, forecast = FALSE)

calendar(x, forecast = FALSE)

raw(x, forecast = FALSE)
```

Arguments

`x`	a `"SA"` or `"jSA"` model.
`forecast`	boolean indicating if the forecast series should be returned.

`geom_arima` *ARIMA model*

Description

Function to add directly to the plot the ARIMA model used in the pre-adjustment process of the seasonal adjustment.

Usage

```
geom_arima(
  mapping = NULL,
  data = NULL,
  stat = "arima",
  geom = c("text", "label"),
  position = "identity",
  ...,
  method = c("x13", "tramoseats"),
  spec = NULL,
  frequency = NULL,
  message = TRUE,
  x_arima = NULL,
  y_arima = NULL,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by aes(). If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	A `data.frame` that contains the data used for the seasonal adjustment.
`stat`	The statistical transformation to use on the data for this layer, as a string.
`geom`	character. The geometric to use to display the data: `GeomText` (`geom = "text"`, the default, see geom_text()) or `GeomLabel` (`geom = "label"`, see geom_label()).
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`...`	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`.
`method`	the method used for the seasonal adjustment. `"x13"` (by default) for the X-13ARIMA method and `"tramoseats"` for TRAMO-SEATS.
`spec`	the specification used for the seasonal adjustment. See x13() or tramoseats().
`frequency`	the frequency of the time series. By default (`frequency = NULL`), the frequency is computed automatically.
`message`	a `boolean` indicating if a message is printed with the frequency used.
`x_arima, y_arima`	position of the text of the ARIMA model. By default, the first position of the `data` is used.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them.

Details

With the parameter geom = "text", the ARIMA model used in the pre-adjustment process of the seasonal adjustment are directly added to the plot. With geom = "label" a rectangle is drawn behind the ARIMA model, making it easier to read.

Examples

```
p_sa_ipi_fr <- ggplot(data = ipi_c_eu_df, mapping = aes(x = date, y = FR)) +
    geom_line(color =  "#F0B400") +
    labs(title = "Seasonal adjustment of the French industrial production index",
         x = "time", y = NULL) +
    geom_sa(color = "#155692", message = FALSE)

# To add the ARIMA model
p_sa_ipi_fr +
    geom_arima(geom = "label",
               x_arima = - Inf, y_arima = -Inf,
               vjust = -1, hjust = -0.1,
               message = FALSE)
```

---

geom_diagnostics *Table of diagnostics*

Description

Adds a table of diagnostics to the plot

Usage

```
geom_diagnostics(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  method = c("x13", "tramoseats"),
  spec = NULL,
  frequency = NULL,
  message = TRUE,
  diagnostics = NULL,
  digits = 2,
  xmin = -Inf,
  xmax = Inf,
  ymin = -Inf,
  ymax = Inf,
  table_theme = ttheme_default(),
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	A data.frame that contains the data used for the seasonal adjustment.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3.
method	the method used for the seasonal adjustment. "x13" (by default) for the X-13ARIMA method and "tramoseats" for TRAMO-SEATS.
spec	the specification used for the seasonal adjustment. See x13() or tramoseats().
frequency	the frequency of the time series. By default (frequency = NULL), the frequency is computed automatically.
message	a boolean indicating if a message is printed with the frequency used.
diagnostics	vector of character containing the name of the diagnostics to plot. See user_defined_variables() for the available parameters.
digits	integer indicating the number of decimal places to be used for numeric diagnostics. By default digits = 2.
xmin, xmax	x location (in data coordinates) giving horizontal location of raster.
ymin, ymax	y location (in data coordinates) giving vertical location of raster.
table_theme	list of theme parameters for the table of diagnostics (see ttheme_default()).
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them.

Examples

```
p_sa_ipi_fr <- ggplot(data = ipi_c_eu_df, mapping = aes(x = date, y = FR)) +
    geom_line(color =  "#F0B400") +
    labs(title = "Seasonal adjustment of the French industrial production index",
         x = "time", y = NULL) +
    geom_sa(color = "#155692", message = FALSE)

# To add of diagnostics with result of the X-11 combined test and the p-values
# of the residual seasonality qs and f tests:
diagnostics <- c("diagnostics.combined.all.summary", "diagnostics.qs", "diagnostics.ftest")
p_sa_ipi_fr +
    geom_diagnostics(diagnostics = diagnostics,
                     ymin = 58, ymax = 72, xmin = 2010,
                     table_theme = gridExtra::ttheme_default(base_size = 8),
                     message = FALSE)

# To customize the names of the diagnostics in the plot:

diagnostics <- c(`Combined test` = "diagnostics.combined.all.summary",
                 `Residual qs-test (p-value)` = "diagnostics.qs",
                 `Residual f-test (p-value)` = "diagnostics.ftest")
```

```
p_sa_ipi_fr +
    geom_diagnostics(diagnostics = diagnostics,
                     ymin = 58, ymax = 72, xmin = 2010,
                     table_theme = gridExtra::ttheme_default(base_size = 8),
                     message = FALSE)

# To add the table below the plot:

p_diag <- ggplot(data = ipi_c_eu_df, mapping = aes(x = date, y = FR)) +
    geom_diagnostics(diagnostics = diagnostics,
                     table_theme = gridExtra::ttheme_default(base_size = 8),
                     message = FALSE) +
    theme_void()

gridExtra::grid.arrange(p_sa_ipi_fr, p_diag,
                        nrow = 2, heights  = c(4, 1))
```

`geom_outlier` *Outliers texts*

Description

Function to add directly to the plot the outliers used in the pre-adjustment process of the seasonal adjustment.

Usage

```
geom_outlier(
  mapping = NULL,
  data = NULL,
  stat = "outlier",
  geom = c("text", "label", "text_repel", "label_repel"),
  position = "identity",
  ...,
  method = c("x13", "tramoseats"),
  spec = NULL,
  frequency = NULL,
  message = TRUE,
  first_date = NULL,
  last_date = NULL,
  coefficients = FALSE,
  digits = 1,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	A data.frame that contains the data used for the seasonal adjustment.
stat	The statistical transformation to use on the data for this layer, as a string.
geom	character. The geometric to use to display the data: GeomText (geom = "text", the default, see geom_text()); GeomLabel (geom = "label", see geom_label()); GeomTextRepel (geom = "text_repel", the default, see geom_text_repel()); GeomLabelRepel (geom = "label_repel", the default, see geom_label_repel()).
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). They may be parameters of geom_text() (if geom = "text"), geom_label() (if geom = "label"), geom_text_repel() (if geom = "text_repel") or geom_label_repel() (if geom = "label_repel").
method	the method used for the seasonal adjustment. "x13" (by default) for the X-13ARIMA method and "tramoseats" for TRAMO-SEATS.
spec	the specification used for the seasonal adjustment. See x13() or tramoseats().
frequency	the frequency of the time series. By default (frequency = NULL), the frequency is computed automatically.
message	a boolean indicating if a message is printed with the frequency used.
first_date	A numeric specifying the first date from which the outliers are plotted. By default (first_date = NULL) the outliers are plotted from the beginning of the time series.
last_date	A numeric specifying the first date from which the outliers are plotted. By default (first_date = NULL) the outliers are plotted until the end of the time series.
coefficients	boolean indicating if the estimates coefficients are printed. By default coefficients = FALSE.
digits	integer indicating the number of decimal places to be used for numeric diagnostics. By default digits = 1.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them.

Details

With the parameter geom = "text", the outliers used in the pre-adjustment process of the seasonal adjustment are directly added to the plot. With geom = "label" a rectangle is drawn behind the names of the outliers, making them easier to read. The same with geom = "text_repel" or geom = "label_repel" but text labels are also repeled away from each other and away from the data points (see geom_label_repel()).

Examples

```
p_sa_ipi_fr <- ggplot(data = ipi_c_eu_df, mapping = aes(x = date, y = FR)) +
    geom_line(color =  "#F0B400") +
    labs(title = "Seasonal adjustment of the French industrial production index",
         x = "time", y = NULL)  +
    geom_sa(color = "#155692", message = FALSE)

# To add the outliers:
p_sa_ipi_fr + geom_outlier(geom = "label",
                    message = FALSE)


# To have a more readable plot with outliers names that repeled away from each other
# and from the data points:
p_sa_ipi_fr +
    geom_outlier(geom = "label_repel",
                 message = FALSE,
                 ylim = c(NA, 65),
                 arrow = arrow(length = unit(0.03, "npc"),
                               type = "closed", ends = "last"))

# To only plot the outliers from a specific date (2009):
p_sa_ipi_fr +
    geom_outlier(geom = "label_repel",
                 message = FALSE,
                 first_date = 2009,
                 ylim = c(NA, 65),
                 arrow = arrow(length = unit(0.03, "npc"),
                               type = "closed", ends = "last"))
```

---

geom_sa *Seasonal adjustment time series*

Description

Performs a seasonal adjustment and plots a time series. `geom_sa()` and `stat_sa()` are aliases: they both use the same arguments. Use `stat_sa()` if you want to display the results with a non-standard geom.

Usage

```
geom_sa(
  mapping = NULL,
  data = NULL,
  stat = "sa",
  position = "identity",
  ...,
  method = c("x13", "tramoseats"),
  spec = NULL,
```

```
  frequency = NULL,
  message = TRUE,
  component = "sa",
  show.legend = NA,
  inherit.aes = TRUE
)

stat_sa(
  mapping = NULL,
  data = NULL,
  geom = "line",
  position = "identity",
  ...,
  method = c("x13", "tramoseats"),
  spec = NULL,
  frequency = NULL,
  message = TRUE,
  component = "sa",
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	A data.frame that contains the data used for the seasonal adjustment.
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3.
method	the method used for the seasonal adjustment. "x13" (by default) for the X-13ARIMA method and "tramoseats" for TRAMO-SEATS.
spec	the specification used for the seasonal adjustment. See x13() or tramoseats().
frequency	the frequency of the time series. By default (frequency = NULL), the frequency is computed automatically.
message	a boolean indicating if a message is printed with the frequency used.
component	a character equals to the component to plot. The result must be a time series. See user_defined_variables() for the available parameters. By default (component = 'sa') the seasonal adjusted component is plotted.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them.
geom	The geometric object to use to display the data

Examples

```
p_ipi_fr <- ggplot(data = ipi_c_eu_df, mapping = aes(x = date, y = FR)) +
    geom_line(color =  "#F0B400") +
    labs(title = "Seasonal adjustment of the French industrial production index",
         x = "time", y = NULL)

# To add the seasonal adjusted series:
p_ipi_fr +
    geom_sa(color = "#155692")

# To add the forecasts of the input data and the seasonal adjusted series:
p_sa <- p_ipi_fr +
    geom_sa(component = "y_f", linetype = 2, message = FALSE, color =  "#F0B400") +
    geom_sa(component = "sa", color = "#155692", message = FALSE) +
    geom_sa(component = "sa_f", color = "#155692", linetype = 2, message = FALSE)
p_sa
```

---

init_ggplot *Initialise 'ggplot2' with 'SA' model*

Description

Initialise 'ggplot2' with 'SA' model

Usage

```
init_ggplot(x, ...)
```

Arguments

x	A "SA" or "jsA" model created with 'RJDemetra'.
...	Other parameters passes to ggplot2::ggplot()

Examples

```
mod <- RJDemetra::x13(ipi_c_eu[, "FR"])
init_ggplot(mod) +
    geom_line(color =  "#F0B400") +
    geom_sa(component = "sa", color = "#155692")
```

ipi_c_eu *Industrial Production Indices in manufacturing in the European Union*

Description

A dataset containing on monthly industrial production indices in manufacturing in the European Union (from sts_inpr_m dataset of Eurostat). Data are based 100 in 2015 and are unadjusted, i.e. neither seasonally adjusted nor calendar adjusted.

Usage

```
ipi_c_eu

ipi_c_eu_df
```

Format

A monthly `ts` object from january 1990 to december 2017 with 34 variables for `ipi_c_eu` and a `data.frame` for `ipi_c_eu_df`.

An object of class `data.frame` with 360 rows and 35 columns.

Details

The dataset contains 34 time series corresponding to the following geographical area

BE	Belgium
BG	Bulgaria
CZ	Czechia
DK	Denmark
DE	Germany (until 1990 former territory of the FRG)
EE	Estonia
IE	Ireland
EL	Greece
ES	Spain
FR	France
HR	Croatia
IT	Italy
CY	Cyprus
LV	Latvia
LT	Lithuania
LU	Luxembourg
HU	Hungary
MT	Malta
NL	Netherlands
AT	Austria
PL	Poland
PT	Portugal

RO	Romania
SI	Slovenia
SK	Slovakia
FI	Finland
SE	Sweden
UK	United Kingdom
NO	Norway
CH	Switzerland
ME	Montenegro
MK	Former Yugoslav Republic of Macedonia, the
RS	Serbia
TR	Turkey
BA	Bosnia and Herzegovina

Source

Eurostat, 'sts_inpr_m' database.

`siratio` *SI-ratio*

Description

SI-ratio

Usage

```
siratio(x, ...)

siratioplot(
  x,
  labels = NULL,
  add = FALSE,
  box = TRUE,
  col.s = "darkblue",
  col.i = "gray",
  col.mean = "red",
  cex.i = 0.1,
  lwd.s = par("lwd"),
  lwd.mean = lwd.s,
  main = "SI ratio",
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  start = NULL,
  end = NULL,
```

```
  ...
)

ggsiratioplot(
  x,
  labels = NULL,
  col.s = "darkblue",
  col.i = "gray",
  col.mean = "red",
  cex.i = 0.5,
  lwd.s = 1,
  lwd.mean = lwd.s,
  main = "SI ratio",
  xlab = NULL,
  ylab = NULL,
  start = NULL,
  end = NULL,
  ...
)
```

Arguments

x	input model or data.
...	unused parameters.
labels	labels.
add	boolean indicating whether a new plot should be drawn.
box	boolean indicating a box around the current plot should be drawn.
col.s, col.i, col.mean	colors of the different components.
cex.i, lwd.s, lwd.mean	graphical parameters.
main, xlab, ylab	title, X and Y axis label.
xlim, ylim	X and Y axis limits.
start, end	first and last dates plotted.

Examples

```
x <- RJDemetra::x13(ipi_c_eu[,"FR"])
siratioplot(x)
ggsiratioplot(x)
```

ts2df *Convert 'ts' object to 'data.frame'*

Description

Function to a ts or mts object to a data.frame that can be directly used in the plot functions.

Usage

```
ts2df(x)
```

Arguments

x	a ts or mts object.

Value

a data.frame object.

Examples

```
# To get the ipi_c_eu_df object:
ts2df(ipi_c_eu)
```

Index

∗ **datasets**
    ipi_c_eu, 12

aes(), *4*, *6*, *8*, *10*
autoplot.SA, 2

calendar *(*components*)*, 3
calendaradj *(*components*)*, 3
components, 3

geom_arima, 3
geom_diagnostics, 5
geom_label(), *4*, *8*
geom_label_repel(), *8*
geom_outlier, 7
geom_sa, 9
geom_text(), *4*, *8*
geom_text_repel(), *8*
ggplot2::ggplot(), *11*
ggsiratioplot *(*siratio*)*, 13

init_ggplot, 11
ipi_c_eu, 12
ipi_c_eu_df *(*ipi_c_eu*)*, 12
irregular *(*components*)*, 3

layer(), *4*, *6*, *8*, *10*

raw *(*components*)*, 3

seasonal *(*components*)*, 3
seasonaladj *(*components*)*, 3
siratio, 13
siratioplot *(*siratio*)*, 13
stat_sa *(*geom_sa*)*, 9

tramoseats(), *4*, *6*, *8*, *10*
trendcycle *(*components*)*, 3
ts2df, 15
ttheme_default(), *6*

user_defined_variables(), *6*, *10*

x13(), *4*, *6*, *8*, *10*