Package ‘ggdemetra’

September 12, 2019

Type Package

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

Version 0.2.1

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)

Imports RJDemetra (>= 0.1.2), ggrepel, gridExtra

Suggests knitr, rmarkdown

SystemRequirements Java SE 8 or higher

License EUPL

URL https://github.com/AQLT/ggdemetra

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

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Alain Quartier-la-Tente [aut, cre]

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

Repository CRAN

Date/Publication 2019-09-12 11:10:02 UTC
Description

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

Usage

```r
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()` or `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.
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**

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

Adds a table of diagnostics to the plot

**Usage**

```r
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()` or `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 `theme_default()`).

- **inherit.aes**: If `FALSE`, overrides the default aesthetics, rather than combining with them.

Examples

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

<table>
<thead>
<tr>
<th>geom_outlier</th>
<th>Outliers texts</th>
</tr>
</thead>
</table>

**Description**

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

**Usage**

```r
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()` or `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**

A boolean indicating if the estimates coefficients are printed. By default coefficients = FALSE.

**digits**

An 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 peleled away from each other and away from the data points.

**Examples**

```r
p_sa_ipi_fr <- ggplot(data = ipi_c_eu_df, mapping = aes(x = date, y = FR)) + geom_line() + labs(title = "Seasonal adjustment of the French industrial production index", x = "time", y = NULL) + geom_sa(color = "red", 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,
      vjust = 4,
      ylim = c(NA, 65), force = 10,
      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,
      vjust = 4,
      ylim = c(NA, 65), force = 10,
      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. Aids the eye in seeing patterns in the presence of overplotting. `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**

```r
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)
```

```r
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()` or `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

```r
p_ipi_fr <- ggplot(data = ipi_c_eu_df, mapping = aes(x = date, y = FR)) +
  geom_line() +
  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 = "red")

# 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) +
  geom_sa(component = "sa", color = "red", message = FALSE) +
  geom_sa(component = "sa_f", color = "red", linetype = 2, message = FALSE)
p_sa
```

---

### 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 37 variables for ipi_c_eu and a data.frame for ipi_c_eu_df.

Details

The dataset contains 37 time series corresponding to the following geographical area:

- EU28: European Union (current composition)
- EU27_2019: European Union (without United Kingdom)
- EA19: Euro area (19 countries)
- 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
- ME: Montenegro
- MK: Former Yugoslav Republic of Macedonia, the
- RS: Serbia
TR  Turkey
BA  Bosnia and Herzegovina

Source

Index

*Topic datasets
  ipi_c_eu, 8

aes(), 2, 4, 5, 7
aes_(), 2, 4, 5, 7

geom_arima, 2
geom_diagnostics, 3
geom_label(), 2, 5, 6
geom_label_repel(), 5, 6
geom_outlier, 5
geom_sa, 7
geom_text(), 2, 5, 6
geom_text_repel(), 5, 6

ipi_c_eu, 8
ipi_c_eu_df (ipi_c_eu), 8

layer(), 2, 4, 6, 8

stat_sa (geom_sa), 7

tramoseats(), 2, 4, 6, 8

theme_default(), 4

user_defined_variables(), 4, 8

x13(), 2, 4, 6, 8