Package ‘ggseas’

June 12, 2018

Title 'stats' for Seasonal Adjustment on the Fly with 'ggplot2'
Version 0.5.4
Maintainer Peter Ellis <peter.ellis2013nz@gmail.com>
Description Provides 'ggplot2' 'stats' that estimate seasonally adjusted series and rolling summaries such as rolling average on the fly for time series.
Depends R (>= 3.1.2), ggplot2 (>= 2.0.0)
Imports seasonal, rlang, stats, zoo
License GPL-3
LazyData true
RoxygenNote 6.0.1
BugReports https://github.com/ellisp/ggseas/issues
Suggests testthat, knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Author Peter Ellis [aut, cre], Christophe Sax [ctb]
Repository CRAN
Date/Publication 2018-06-12 13:33:33 UTC

R topics documented:

  ggseas-package .................................................. 2
gsdc .............................................................. 2
ldeaths_df .......................................................... 4
nzbop ............................................................... 5
stat_decomp ......................................................... 6
stat_index ........................................................... 8
stat_rollapplyr ...................................................... 9
stat_seas ............................................................ 10
stat_stl ............................................................. 12
tsd ................................................................. 14
ggseas-package  

Seasonal decomposition on the fly

Description

ggseas allows you to perform seasonal decomposition on the fly as part of a 'ggplot2' pipeline.

Details

Two main sets of functions are provided:

- `stat_seas()`, `stat_stl()` and friends do seasonal adjustment, indexing, rolling averages and by default render a line geom. You can consider these as taking the place of `geom_line()` in the ggplot2 pipeline
- `ggsdc()` goes where `ggplot()` normally does (ie at the beginning of the graphics part of the pipeline) and creates a graphic with four facets for the original data, trend, seasonal and random components

Author(s)

Maintainer: Peter Ellis <peter.ellis2013nz@gmail.com>

Other contributors:

- Christophe Sax [contributor]

See Also

Useful links:

- Report bugs at https://github.com/ellisp/ggseas/issues

---

ggsdc  

Visualise seasonal decomposition

Description

Creates a four-facet plot of seasonal decomposition showing observed, trend, seasonal and random components

Usage

```r
ggsdc(data, mapping, frequency = NULL, method = c("stl", "decompose", "seas"), start = NULL, s.window, type = c("additive", "multiplicative"), index.ref = NULL, index.basis = 100, facet.titles = c("observed", "trend", "seasonal", "irregular"))
```
Arguments

data  
dataset to use for plot.

mapping  
List of aesthetic mappings. Must include x and y, and optionally can include colour/color

frequency  
frequency of the period of the time series eg 12 = monthly

method  
function to use for performing the seasonal decomposition. stl and decompose are functions in the stats package; seas is access to the seats program from X-13-SEATS-ARIMA via the seasonal package

start  
starting time for the data; only needed if method = 'seas'.
s.window  
parameter to pass to stl()
type  
parameter to pass to decompose()
index.ref  
if not NULL, a vector of integers indicating which elements of the beginning of each series to use as a reference point for converting to an index. If NULL, no conversion takes place and the data are presented on the original scale.

index.basis  
if index.ref is not NULL, the basis point for converting to an index, most commonly 100 or 1000. See examples.

facet.titles  
a vector in the order of observed, trend, seasonal and irregular for the titles of the four facets of the decomposition. Make sure you get the order right...

Details

This function takes a data frame and performs seasonal decomposition on the variable mapped to the y aesthetic, grouped by the variable (if any) mapped to the colour or color aesthetic. This allows the user to perform the equivalent of plot(stats::decompose(x)) but in the ggplot2 environment for themes, polishing etc; and to overlay decompositions on the same graphic; and with the X13-SEATS-ARIMA seasonal decomposition (so far only with default settings).

The "seasonal" component can be either multiplicative (in which case it will in a small range of values around one) or additive (in which case it will be on the scale of the original data), depending on the settings.

Value

an object of class ggplot with four facets

See Also

decompose, stl, seas

Examples

# sample time series data in data frame
ap_df <- tsdf(AirPassengers)

Ggsdc(ap_df, aes(x = x, y = y), method = "decompose") +
geom_line()
Monthly Deaths from Lung Diseases in the UK

Description

A long form combination of fdeaths and mdeaths from the datasets package.

Usage

ldeaths_df

Format

A data frame with 141 rows and 3 variables.
Details

- YearMon. Approximate, regular decimal representation of the beginning of the period of measurement. January 1974 is 1974.000
- sex.
- deaths. Monthly deaths from bronchitis, emphysema and asthma. ...

Source


See Also

ldeaths

Description

New Zealand’s "BPM6 Quarterly, Balance of payments major components (Qrtly-Mar/Jun/Sep/Dec)".

Usage

nzbop

Format

An object of class data.frame with 3676 rows and 5 columns.

Details

"BPM6" refers to the sixth edition of the IMF’s Balance of Payments and International Investment Position Manual, which is the method used by Statistics New Zealand to prepare these data.

Note:

- ‘Value’ is in millions of New Zealand dollars and is not adjusted for inflation.
- ‘fob’ means ’free on board’.
- ‘inv.’ stands for investment
- TimePeriod is the last day of the quarterly reference period ie 1971-06-30 means the fourth, fifth and six months of 1971.

This dataset was downloaded from [http://www.stats.govt.nz/infoshare/](http://www.stats.govt.nz/infoshare/) and transformed in the following way:

- missing values were filtered out (ie of series that started later than the longest series)
- a ‘Balance’ indicator variable was added for easier manipulation and filtering
- the single variable categorisation was split into two (Account and Category) to make it tidier.
Source


stat_decomp

Classical seasonal adjustment Stat

Description

Conducts seasonal adjustment on the fly for ggplot2, from classical seasonal decomposition by moving averages

Usage

stat_decomp(mapping = NULL, data = NULL, geom = "line", position = "identity", show.legend = NA, inherit.aes = TRUE, frequency = NULL, type = c("additive", "multiplicative"), index.ref = NULL, index.basis = 100, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapping</td>
<td>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.</td>
</tr>
<tr>
<td>data</td>
<td>The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data.</td>
</tr>
<tr>
<td>geom</td>
<td>The geometric object to use display the data</td>
</tr>
<tr>
<td>position</td>
<td>Position adjustment, either as a string, or the result of a call to a position adjustment function.</td>
</tr>
<tr>
<td>show.legend</td>
<td>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.</td>
</tr>
<tr>
<td>inherit.aes</td>
<td>If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().</td>
</tr>
<tr>
<td>frequency</td>
<td>The frequency for the time series</td>
</tr>
<tr>
<td>type</td>
<td>The type of seasonal component</td>
</tr>
</tbody>
</table>
stat_decomp

index.ref if not NULL, a vector of integers indicating which elements of the beginning of each series to use as a reference point for converting to an index. If NULL, no conversion takes place and the data are presented on the original scale.

index.basis if index.ref is not NULL, the basis point for converting to an index, most commonly 100 or 1000. See examples.

... other arguments for the geom

Details

Classical decomposition is a very basic way of performing seasonal adjustment and is not recommended if you have access to X13-SEATS-ARIMA (stat_seas). stat_decomp cannot allow the seasonality to vary over time, or take outliers into account in calculating seasonality.

See Also

decompose

Other time series stats for ggplot2: stat_index, stat_rollapplyr, stat_seas, stat_stl

Examples

```r
ap_df <- tsdf(AirPassengers)

# Default additive decomposition (doesn't work well in this case!):
ggplot(ap_df, aes(x = x, y = y)) +
  stat_decomp()

# Multiplicative decomposition, more appropriate:
ggplot(ap_df, aes(x = x, y = y)) +
  stat_decomp(type = "multiplicative")

# Multiple time series example:
lldeaths_df, aes(x = YearMon, y = deaths, colour = sex)) +
geom_point() +
  facet_wrap(~sex) +
  stat_decomp() +
  ggtitle("Seasonally adjusted lung deaths")

# Example using index:
lldeaths_df, aes(x = YearMon, y = deaths, colour = sex)) +
  facet_wrap(~sex) +
  stat_decomp(index.ref = 1:12, index.basis = 1000) +
  ggtitle("Rolling annual median lung deaths, indexed (average month in 1974 = 1000)")
```
**stat_index**

**Index Stat**

**Description**

Convert a time series from the original scale to an index for ggplot2

**Usage**

```r
stat_index(mapping = NULL, data = NULL, geom = "line",
position = "identity", show.legend = NA, inherit.aes = TRUE,
index.ref = NULL, index.basis = 100, ...)
```

**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**
  The data to be displayed in this layer. There are three options:
  - If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data.

- **geom**
  The geometric object to use display the data

- **position**
  Position adjustment, either as a string, or the result of a call to a position adjustment function.

- **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. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

- **index.ref**
  if not `NULL`, a vector of integers indicating which elements of the beginning of each series to use as a reference point for converting to an index. If `NULL`, no conversion takes place and the data are presented on the original scale.

- **index.basis**
  if `index.ref` is not `NULL`, the basis point for converting to an index, most commonly 100 or 1000. See examples.

- **...**
  other arguments for the geom

**See Also**

Other time series stats for ggplot2: `stat_decomp`, `stat_rollapplyr`, `stat_seas`, `stat_stl`
Examples

```r
ap_df <- tsdf(AirPassengers)

ggplot(ldeaths_df, aes(x = YearMon, y = deaths, color = sex)) +
  stat_index(index.ref = 1:12, index.basis = 1000) +
  ylab("Deaths index\n(average of first 12 months = 100")
```

---

**stat_rollapplyr**  
**Rolling summary Stat**

**Description**

Calculates a rolling summary, usually rolling average, on the fly for ggplot2

**Usage**

```r
stat_rollapplyr(mapping = NULL, data = NULL, geom = "line",
  position = "identity", show.legend = NA, inherit.aes = TRUE, width,
  align = "right", FUN = mean, index.ref = NULL, index.basis = 100, ...)
```

**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`  
  The data to be displayed in this layer. There are three options:
  
  If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  
  A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  
  A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data.

- `geom`  
  The geometric object to use display the data

- `position`  
  Position adjustment, either as a string, or the result of a call to a position adjustment function.

- `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. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

- `width`  
  The width to which the rolling version of `FUN` is applied
align

specifies whether the transformed series should be left or right-aligned or centered compared to the rolling window of observations.

FUN

summary function, usually some kind of average, to apply on a rolling basis.

index.ref

if not NULL, a vector of integers indicating which elements of the beginning of each series to use as a reference point for converting to an index. If NULL, no conversion takes place and the data are presented on the original scale.

index.basis

if index.ref is not NULL, the basis point for converting to an index, most commonly 100 or 1000. See examples.

... other arguments for the geom

Details

Calculates a rolling summary (usually rolling average) on the fly for purposes of plotting with ggplot2.

See Also
decompose

Other time series stats for ggplot2: stat_decomp, stat_index, stat_seas, stat_stl

Examples

```r
ap_df <- tsdf(AirPassengers)

ggplot(ap_df, aes(x = x, y = y)) +
  stat_rollapplyr(width = 12)

# rolling average after converting to an index, 1000 = average value
# in the first 12 months.
  ggplot(ap_df, aes(x = x, y = y)) +
  stat_rollapplyr(width = 12, index.ref = 1:12, index.basis = 1000)

ggplot(ldeaths_df, aes(x = YearMon, y = deaths, colour = sex)) +
  geom_point() +
  facet_wrap(~sex) +
  stat_rollapplyr(width = 12, FUN = median) +
  ggtitle("Seasonally adjusted lung deaths")
```

Description

Conducts X13-SEATS-ARIMA seasonal adjustment on the fly for ggplot2.
Usage

```r
stat_seas(mapping = NULL, data = NULL, geom = "line",
position = "identity", show.legend = NA, inherit.aes = TRUE,
x13_params = NULL, index.ref = NULL, index.basis = 100,
frequency = NULL, start = NULL, ...)
```

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**
The data to be displayed in this layer. There are three options:
- If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
- A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
- A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data.

**geom**
The geometric object to use display the data

**position**
Position adjustment, either as a string, or the result of a call to a position adjustment function.

**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. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

**x13_params**
a list of other parameters for `seas`

**index.ref**
if not `NULL`, a vector of integers indicating which elements of the beginning of each series to use as a reference point for converting to an index. If `NULL`, no conversion takes place and the data are presented on the original scale.

**index.basis**
if `index.ref` is not `NULL`, the basis point for converting to an index, most commonly 100 or 1000. See examples.

**frequency**
The frequency for the time series

**start**
The starting point for the time series, in a format suitable for `ts()`

... other arguments for the geom

See Also

- `seas`

Other time series stats for ggplot2: `stat_decomp`, `stat_index`, `stat_rollapplyr`, `stat_stl`
Examples

```r
## Not run:
ap_df <- tsdf(AirPassengers)

# SEATS with defaults:
ggplot(ap_df, aes(x = x, y = y)) +
  stat_seas()

# X11 with no outlier treatment:
ggplot(ap_df, aes(x = x, y = y)) +
  stat_seas(x11_params = list(x11 = "", outlier = NULL))

# Multiple time series example:
ggplot(ldeaths_df, aes(x = YearMon, y = deaths, colour = sex)) +
  geom_point() +
  facet_wrap(~sex) +
  stat_seas() +
  ggtitle("Seasonally adjusted lung deaths")

# example use of index:
ggplot(ap_df, aes(x = x, y = y)) +
  stat_seas(x13_params = list(x11 = "", outlier = NULL),
            index_ref = 1, index.basis = 1000) +
  labs(y = "Seasonally adjusted index\n(first observation = 1000")

# if the x value is not a decimal eg not created with time(your_ts_object),
# you need to specify start and frequency by hand:
ggplot(subset(nzbop, Account == "Current account"),
  aes(x = TimePeriod, y = Value)) +
  stat_seas(start = c(1971, 2), frequency = 12) +
  facet_wrap(~Category, scales = "free_y")

## End(Not run)
```

---

**stat_stl**  
**LOESS seasonal adjustment Stat**

**Description**
Conducts seasonal adjustment on the fly for `ggplot2`, from LOESS seasonal decomposition

**Usage**
```
stat_stl(mapping = NULL, data = NULL, geom = "line",
         position = "identity", show.legend = NA, inherit.aes = TRUE,
         frequency = NULL, s.window, index.ref = NULL, index.basis = 100, ...)
```
Arguments

mapping Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes()}. If specified and \texttt{inherit.aes = TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.

data The data to be displayed in this layer. There are three options: If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}. A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a \texttt{data.frame}, and will be used as the layer data.

geom The geometric object to use display the data

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

show.legend logical. Should this layer be included in the legends? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If \texttt{FALSE}, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. \texttt{borders()}.

frequency The frequency for the time series

s.window either the character string “periodic” or the span (in lags) of the loess window for seasonal extraction, which should be odd and at least 7, according to Cleveland et al. This has no default and must be chosen.

index.ref if not \texttt{NULL}, a vector of integers indicating which elements of the beginning of each series to use as a reference point for converting to an index. If \texttt{NULL}, no conversion takes place and the data are presented on the original scale.

index.basis if \texttt{index.ref} is not \texttt{NULL}, the basis point for converting to an index, most commonly 100 or 1000. See examples.

... other arguments for the geom

See Also

Other time series stats for \texttt{ggplot2}: \texttt{stat_decomp}, \texttt{stat_index}, \texttt{stat_rollapplyr}, \texttt{stat Seas}

Examples

```r
ap_df <- tsdf(AirPassengers)
# periodic if fixed seasonality; doesn't work well:
ggplot(ap_df, aes(x = x, y = y)) +
  stat_stl(s.window = "periodic")

# seasonality varies a bit over time, works better:
```
```r
ggplot(ap_df, aes(x = x, y = y)) +
  stat_stl(s.window = 7)

# Multiple time series example:
ggplot(ldeaths_df, aes(x = YearMon, y = deaths, colour = sex)) +
  geom_point() +
  facet_wrap(~sex) +
  stat_stl(s.window = 7) +
  ggtitle("Seasonally adjusted lung deaths")

# Index so first value is 100:
ggplot(ap_df, aes(x = x, y = y)) +
  stat_stl(s.window = 7, index.ref = 1)
```

---

**tsdf**

*Time series to data frame*

**Description**

Convert a ts object to data.frame with columns for time period and the original data

**Usage**

`tsdf(timeseries, colname = "x")`

**Arguments**

- `timeseries`: an object of class ts or mts
- `colname`: Column name to give to the time period column

**Details**

A convenience function to create a data frame from a time series or multiple time series object. The motivation is to make it easy to pass time series data to functions that need data frames such as `ggplot2`.

**Value**

a data.frame with the same number of rows as the original time series

**Examples**

```r
head(tsdf(AirPassengers))

ld <- cbind(fdeaths, mdeaths)
head(tsdf(ld))
```
Index

*Topic datasets
  ldeaths_df, 4
  nzbop, 5

aes(), 6, 8, 9, 11, 13
aes_(), 6, 8, 9, 11, 13
borders(), 6, 8, 9, 11, 13
decompose, 3, 7, 10
fortify(), 6, 8, 9, 11, 13
ggplot(), 6, 8, 9, 11, 13
ggsdc, 2
ggseas (ggseas-package), 2
  ggseas-package, 2
ldeaths, 5
  ldeaths_df, 4

nzbop, 5

seas, 3, 11
stat_decomp, 6, 8, 10, 11, 13
stat_index, 7, 8, 10, 11, 13
stat_rollapplyr, 7, 8, 9, 11, 13
stat_seas, 7, 8, 10, 10, 13
stat_stl, 7, 8, 10, 11, 12
stl, 3

tsdf, 14