Package ‘x12’

December 5, 2017

Version 1.9.0
Date 2017-12-05
Title Interface to ‘X12-ARIMA’/’X13-ARIMA-SEATS’ and Structure for Batch Processing of Seasonal Adjustment
Author Alexander Kowarik <alexander.kowarik@statistik.gv.at>, Angelika Meraner
Maintainer Alexander Kowarik <alexander.kowarik@statistik.gv.at>
Depends R (>= 2.14.0),stats,utils,grDevices,x13binary
Imports stringr,methods
Suggests parallel, testthat
Description The ‘X13-ARIMA-SEATS’ <https://www.census.gov/srd/www/x13as/> methodology and software is a widely used software and developed by the US Census Bureau. It can be accessed from ‘R’ with this package and ‘X13-ARIMA-SEATS’ binaries are provided by the ‘R’ package ‘x13binary’.
License GPL (>= 2)
LazyData TRUE
ByteCompile TRUE
URL https://github.com/statistikat/x12
Collate 'class.R' 'crossVal-methods.R' 'env_functions.R' 'ggplot2Fn.R'
   'parameter-methods.R' 'plot-methods.R' 'plotFunctions.R'
   'readSpec.R' 'summary-methods.R' 'summary.output.x12.R'
   'summary.x12.R' 'times.R' 'x12-methods.R' 'x12work.R'
   'readx12Out.R' 'zzz.R'
NeedsCompilation no
Repository CRAN
Date/Publication 2017-12-05 17:52:47 UTC

R topics documented:

AirPassengersX12 ................................................................. 2
AirPassengersX12Batch ......................................................... 3
airpassengersX12

Description

x12 Single object with the AirPassengers time series

Usage

data(AirPassengersX12)

Examples

data(AirPassengersX12)
summary(AirPassengersX12)
summary(AirPassengersX12, oldOutput=10)
Description

x12Batch object of four AirPassengers series with parameters and output objects.

Usage

data(AirPassengersX12Batch)

Examples

data(AirPassengersX12Batch)
summary(AirPassengersX12Batch)

crossVal

Methods for Function crossVal in Package x12

Description

Cross Validation with function crossVal in package x12.

Usage

## S4 method for signature 'ts'
crossVal(object, x12Parameter, x12BaseInfo, 
showCI=FALSE, main="Cross Validation",
col_original="black", col_fc="#2020ff", col_bc="#2020ff",
col_ci="#d1d1ff", col_cishade="#d1d1ff",
lty_original=1, lty_fc=2, lty_bc=2, lty_ci=1,
lwd_original=1, lwd_fc=1, lwd_bc=1, lwd_ci=1, ytop=1,
points_bc=FALSE, points_fc=FALSE, points_original=FALSE,
showLine=TRUE, col_line="grey", lty_line=3,
ylab="Value", xlab="Date", ylim=NULL, span=NULL)

## S4 method for signature 'x12Single'
crossVal(object, x12BaseInfo=new("x12BaseInfo"),
showCI=FALSE, main="Cross Validation",
col_original="black", col_fc="#2020ff", col_bc="#2020ff",
col_ci="#d1d1ff", col_cishade="#d1d1ff",
lty_original=1, lty_fc=2, lty_bc=2, lty_ci=1,
lwd_original=1, lwd_fc=1, lwd_bc=1, lwd_ci=1, ytop=1,
points_bc=FALSE, points_fc=FALSE, points_original=FALSE,
showLine=TRUE, col_line="grey", lty_line=3,
ylab="Value", xlab="Date", ylim=NULL, span=NULL)
Arguments

- **object**: object of class `ts` or `x12Single-class`.
- **x12Parameter**: object of class `x12Parameter`.
- **x12BaseInfo**: object of class `x12BaseInfo`.
- **showCI**: logical specifying if the prediction interval should be plotted.
- **main**: plot title.
- **col_original**: color of the original time series.
- **col_fc**: color of the forecasts.
- **col_bc**: color of the backcasts.
- **col_ci**: color of the prediction interval.
- **col_cishade**: color of the shading of the prediction interval.
- **lty_original**: line type of the original time series.
- **lty_fc**: line type of the forecasts.
- **lty_bc**: line type of the backcasts.
- **lty_ci**: line type of the prediction interval.
- **lwd_original**: line width of the original time series.
- **lwd_fc**: line width of the forecasts.
- **lwd_bc**: line width of the backcasts.
- **lwd_ci**: line width of the prediction interval.
- **ytop**: multiplication factor for `ylim`.
- **points_bc**: logical specifying if backcasts should additionally be indicated with points.
- **points_fc**: logical specifying if forecasts should additionally be indicated with points.
- **points_original**: logical specifying if the original time series should additionally be indicated with points.
- **showLine**: logical indicating if a boundary line should be drawn before/after fore-/backcasts.
- **col_line**: color of `showLine`.
- **lty_line**: line type of `showLine`.
- **ylab**: label of y-axis.
- **xlab**: label of x-axis.
- **ylim**: range of the y-axis.
- **span**: vector of length 4, limiting the data used for the plot.
  Start and end date of said time interval can be specified by 4 integers in the format `c(start year, start seasonal period, end year, end seasonal period)`

Value

An S4 object of class `crossValidation-class`.
crossValidation-class

Methods

signature(object = "ts")
signature(object = "x12Single")

Author(s)

Alexander Kowarik, Angelika Meraner

See Also

x12, plot, plotSpec, plotSeasFac, plotRsdAcf

Examples

## Not run:
s <- new("x12Single", ts=AirPassengers, tsName="air")
s <- setP(s, list(estimate=TRUE, regression.variables="AO1950.1", outlier.types="all", outlier.critical=list(LS=3.5, TC=2.5), backcast_years=1/2, forecast_years=1))

cv <- crossVal(s, showLine=TRUE)
cv

## End(Not run)

crossValidation-class   Class "crossValidation"

Description

Standardized object for saving the output of crossVal in R.

Objects from the Class

Objects can be created by calls of the form new("crossValidation", ...).

Slots

backcast: Object of class "dfOrNULL" ~-
forecast: Object of class "dfOrNULL" ~-

Author(s)

Alexander Kowarik, Angelika Meraner

Examples

showClass("crossValidation")
diagnostics-class  

Description

The x12 binaries produce a file with the suffix .udg. This class is a list of a selection of its content.

Objects from the Class

Objects can be created by calls of the form `new("diagnostics", ...)`. It is used internally by the methods for x12Batch and x12Single objects.

Slots

`.Data`: Object of class "list" ~-

Extends

Class "list", from data part.

Author(s)

Alexander Kowarik

Examples

`showClass("diagnostics")`

fbcast-class  

Description

Objects to save estimate, lowerci and upperci of fore- and/or backcasts in one standardized list. Used by the functions in this package.

Objects from the Class

Objects can be created by calls of the form `new("fbcast", ...)`. 

Slots

`estimate`: Object of class "ts" ~-

`lowerci`: Object of class "ts" ~-

`upperci`: Object of class "ts" ~-
getP-methods

Author(s)
Alexander Kowarik

Examples

```r
showClass("fbcast")
```

---

**Description**

getP and setP for retrieving and setting parameters from a `x12Single-class`, `x12Batch-class` or `x12Parameter-class` object.

**Usage**

```r
## S4 method for signature 'x12Single'
getP(object, whichP)
## S4 method for signature 'x12Batch'
getP(object, whichP, index=NULL)
## S4 method for signature 'x12Parameter'
getP(object, whichP)
## S4 method for signature 'x12Single'
setP(object, listP)
## S4 method for signature 'x12Batch'
setP(object, listP, index=NULL)
## S4 method for signature 'x12Parameter'
setP(object, listP)
```

**Arguments**

- `object` object of class `x12Single-class`, `x12Batch-class` or `x12Parameter-class`
- `whichP` character vector with the names of the parameters to extract
- `listP` named list of parameters to change
- `index` index of the series in `x12Batch-class` to change or extract (NULL=all)

**Methods**

```r
signature(object = "x12Batch")
signature(object = "x12Parameter")
signature(object = "x12Single")
```

**See Also**

`x12`, `x12Single`, `x12Batch`
Examples

```r
## Not run:
# Create new batch object with 4 time series
xb <- new("x12Batch",list(AirPassengers,AirPassengers,AirPassengers,AirPassengers))
# change the automdl to FALSE in all 4 elements
xb <- setP(xb,list(automdl=FALSE))
# change the arima.model and arima.smodel settings for the first ts object
xb <- setP(xb,list(arima.model=c(1,1,0),arima.smodel=c(1,1,0)),1)
# change the arima.model and arima.smodel settings for the second ts object
xb <- setP(xb,list(arima.model=c(0,1,1),arima.smodel=c(0,1,1)),2)
# change the arima.model and arima.smodel settings for the third ts object
xb <- setP(xb,list(arima.model=c(0,1,1),arima.smodel=c(0,1,1)),3)
# change the arima.model and arima.smodel settings for the fourth ts object
xb <- setP(xb,list(arima.model=c(1,1,1),arima.smodel=c(1,1,1)),4)
# run x12 on all series
xb <- x12(xb)
summary(xb)
# Set automdl=TRUE for the first ts
xb <- setP(xb,list(automdl=TRUE),1)
getP(xb,"automdl")
# rerun x12 on all series (the binaries will only run on the first one)
xb <- x12(xb)
# summary with oldOutput
summary(xb,oldOutput=10)
# Change the parameter and output of the first series back to the first run
xb <- prev(xb,index=1,n=1)
# summary with oldOutput (--- No valid previous runs. ---)
summary(xb,oldOutput=10)

## End(Not run)
```

### Description

Functions `loadP` and `saveP` load and save parameter settings.

### Usage

```r
## S4 method for signature 'x12Single'
loadP(object, file)
## S4 method for signature 'x12Batch'
loadP(object, file)
## S4 method for signature 'x12Parameter'
loadP(object, file)
## S4 method for signature 'x12Single'
saveP(object, file)
```
loadP

## S4 method for signature 'x12Batch'
saveP(object, file)
## S4 method for signature 'x12Parameter'
saveP(object, file)

**Arguments**

- **object**: object of class `x12Single-class`, `x12Batch-class` or `x12Parameter-class`.
- **file**: filepath

**Methods**

- `signature(object = "x12Batch")`
- `signature(object = "x12Parameter")`
- `signature(object = "x12Single")`

**See Also**

- `x12`, `x12Batch`

**Examples**

```r
## Not run:
#Create new batch object with 4 time series and change some parameters
xb <- new("x12Batch", list(AirPassengers, AirPassengers, AirPassengers, AirPassengers))
xb <- setP(xb, list(automdl=FALSE))
xb <- setP(xb, list(arima.model=c(1,1,0), arima.model=c(1,1,0), 1))
xb <- setP(xb, list(arima.model=c(0,1,1), arima.model=c(0,1,1)), 2)
xb <- setP(xb, list(arima.model=c(0,1,1), arima.model=c(1,1,1)), 3)
xb <- setP(xb, list(arima.model=c(1,1,1), arima.model=c(1,1,1)), 4)

#save all parameters
saveP(xb, file="xyz.Rdata")
xbl <- new("x12Batch", list(AirPassengers, AirPassengers, AirPassengers, AirPassengers))

#load all parameters and save it to the corresponding series inside a x12Batch-object
xbl <- loadP(xbl, file="xyz.Rdata")

xs <- new("x12Single", ts=AirPassengers)
x <- setP(xs, list(arima.model=c(2,1,1), arima.model=c(2,1,1)))
#Save the parameters
saveP(xs, file="xyz1.RData")

#Load a saved parameter set to a x12Single object
xs <- new("x12Single", ts=AirPassengers)
x <- loadP(xs, file="xyz1.RData")

#Replace all parameters in a x12Batch object with one parameter set
xb <- new("x12Batch", list(AirPassengers, AirPassengers, AirPassengers, AirPassengers))
xb <- loadP(xb, file="xyz1.RData")
```
Description

Plot function for \texttt{x12} output in package \texttt{x12}.

Usage

```r
## S4 method for signature 'x12Single'
plot(x, original=TRUE, sa=FALSE, trend=FALSE, log_transform=FALSE,
     ylab="Value", xlab="Date", main="TS", col_original="black", col_sa="blue",
     col_trend="green", lwd_original=1, lwd_sa=1, lwd_trend=1, lty_sa=1, lty_trend=1, ytop=1,
     showAllOut=FALSE, showAllOutlines=FALSE, showOut=NULL, annComp=TRUE, annCompTrend=TRUE,
     col_ao="red", col_ls="red", col_tc="red", col_annComp="grey", lwd_out=1, cex_out=1.5,
     pch_ao=4, pch_ls=2, pch_tc=23, plot_legend=TRUE, legend_horiz=TRUE, legend_bty="o",
     forecast=FALSE, backcast=FALSE,
     showCI=TRUE, col_bc="#2020ff", col_ci="#d1d1ff",
     col_cishade="#d1d1ff", lty_original=1, lty_fc=2, lty_bc=2, lty_ci=1, lwd_fc=1, lwd_bc=1,
     lwd_ci=1, points_bc=FALSE, points_fc=FALSE, points_original=FALSE, showLine=FALSE,
     col_line="grey", lty_line=3, ylim=NULL, span=NULL, ...)  
## S4 method for signature 'x12Batch'
plot(x, what="ask", original=TRUE, sa=FALSE, trend=FALSE, log_transform=FALSE,
     ylab="Value", xlab="Date", main="TS", col_original="black", col_sa="blue",
     col_trend="green", lwd_original=1, lwd_sa=1, lwd_trend=1, lty_sa=1, lty_trend=1, ytop=1,
     showAllOut=FALSE, showAllOutlines=FALSE, showOut=NULL, annComp=TRUE, annCompTrend=TRUE,
     col_ao="red", col_ls="red", col_tc="red", col_annComp="grey", lwd_out=1, cex_out=1.5,
     pch_ao=4, pch_ls=2, pch_tc=23, plot_legend=TRUE, legend_horiz=TRUE, legend_bty="o",
     forecast=FALSE, backcast=FALSE,
     showCI=TRUE, col_bc="#2020ff", col_ci="#d1d1ff",
     col_cishade="#d1d1ff", lty_original=1, lty_fc=2, lty_bc=2, lty_ci=1, lwd_fc=1, lwd_bc=1,
     lwd_ci=1, points_bc=FALSE, points_fc=FALSE, points_original=FALSE, showLine=FALSE,
     col_line="grey", lty_line=3, ylim=NULL, span=NULL, ...)  
## S4 method for signature 'x12Output'
plot(x, original=TRUE, sa=FALSE, trend=FALSE, log_transform=FALSE,
     ylab="Value", xlab="Date", main="TS", col_original="black", col_sa="blue",
     col_trend="green", lwd_original=1, lwd_sa=1, lwd_trend=1, lty_sa=1, lty_trend=1, ytop=1,
     showAllOut=FALSE, showAllOutlines=FALSE, showOut=NULL, annComp=TRUE, annCompTrend=TRUE,
     col_ao="red", col_ls="red", col_tc="red", col_annComp="grey", lwd_out=1, cex_out=1.5,
     pch_ao=4, pch_ls=2, pch_tc=23, plot_legend=TRUE, legend_horiz=TRUE, legend_bty="o",
     forecast=FALSE, backcast=FALSE, showCI=TRUE,
     col_bc="#2020ff", col_ci="#d1d1ff", col_cishade="#d1d1ff",
     lty_original=1, lty_fc=2, lty_bc=2, lty_ci=1, lwd_fc=1, lwd_bc=1, lwd_ci=1,
     points_bc=FALSE, points_fc=FALSE, points_original=FALSE,
```

Arguments

- **x**: object of class `x12Output-class` or `x12Single-class`.
- **original**: logical defining whether the original time series should be plotted.
- **sa**: logical defining whether the seasonally adjusted time series should be plotted.
- **trend**: logical defining whether the trend should be plotted.
- **log_transform**: logical defining whether the log transform should be plotted.
- **showAllout**: logical defining whether all outliers should be plotted.
- **showOut**: character in the format “TypeYear.Seasonalperiod” defining a specific outlier to be plotted.
- **annComp**: logical defining whether an annual comparison should be performed for the outlier defined in `showOut`.
- **forecast**: logical defining whether the forecasts should be plotted.
- **backcast**: logical defining whether the backcasts should be plotted.
- **showCI**: logical defining whether the prediction intervals should be plotted.
- **ylab**: label of y-axis.
- **xlab**: label of x-axis.
- **main**: plot title.
- **col_original**: color of the original time series.
- **col_sa**: color of the seasonally adjusted time series.
- **col_trend**: color of the trend.
- **lwd_original**: line width of the original time series.
- **lwd_sa**: line width of the seasonally adjusted time series.
- **lwd_trend**: line width of the trend.
- **lty_original**: line type of the original time series.
- **lty_sa**: line type of the seasonally adjusted time series.
- **lty_trend**: line type of the trend.
- **ytop**: multiplication factor for `ylim`.
- **showAllOutLines**: logical specifying if vertical lines should be plotted with the outliers.
- **annCompTrend**: logical specifying if the trend of the annual comparison should be plotted.
- **col_ao**: color of additive outliers.
- **col_ls**: color of level shifts.
- **col_tc**: color of transitory changes.
- **col_annComp**: color of annual comparison.
- **lwd_out**: line width of outliers.
cex_out magnification factor for size of symbols used for plotting outliers.
pch_ao symbols used for additive outliers.
pch_ls symbols used for level shifts.
pch_tc symbols used for transitory changes.
plot_legend logical specifying if a legend should be plotted.
legend_horiz Orientation of the legend
legend_bty the type of box to be drawn around the legend. The allowed values are "o" (the default) and "n".
col_fc color of forecasts.
col_bc color of backcasts.
col_ci color of prediction interval.
col_cishade color of prediction interval shading.
lty_fc line type of forecasts.
lty_bc line type of backcasts.
lty_ci line type of prediction interval.
lwd_fc line width of forecasts.
lwd_bc line width of backcasts.
lwd_ci line width of prediction interval.
points_bc logical specifying if backcasts should additionally be indicated with points.
points_fc logical specifying if forecasts should additionally be indicated with points.
points_original logical specifying if the original time series should additionally be indicated with points.
showLine logical indicating if a boundary line should be drawn before/after fore-/backcasts.
col_line color of showLine.
lty_line line type of showLine.
ylim range of the y-axis.
span vector of length 4, limiting the data used for the plot.
Start and end date of said time interval can be specified by 4 integers in the format `cHstart yearL start seasonal periodL end yearL end seasonal periodI`
what How multiple plots should be treated. "ask" is the only option at the moment.
... ignored.

Methods

signature(x = "x12Output")
signature(x = "x12Single")

Author(s)

Alexander Kowarik, Angelika Meraner
See Also

plotSpec, plotSeasFac, plotRsdAcf

Examples

## Not run:

```r
# new("x12Single",ts=AirPassengers,tsName="air")
s <- setP(s,list(estimate=TRUE,regression.variables="AO1958.1",outlier.types="all",
    outlier.critical=list(LO=3.5,TC=2.5),backcast_years=1/2))
s <- x12(s)

# w/o outliers
plot(s@x12Output,sa=TRUE,trend=TRUE,original=FALSE)

# with (all) outliers
plot(s,showAllout=TRUE,sa=TRUE,trend=TRUE,log_transform=TRUE,lwd_out=1,pch_ao=4)

# with showOut
plot(s,showOut="AO1960 .Jun",sa=FALSE,trend=FALSE,annComp=TRUE,log_transform=TRUE)

# w/o legend
plot(s,showAllout=TRUE,plot_legend=FALSE)

# Forecasts & Backcasts
plot(s,forecast=TRUE)

# Seasonal Factors and SI Ratios
plotSeasFac(s)
```

## End(Not run)
plot.x12work  

**Plot method for objects of class x12work**

**Description**

Plot method for objects of class "x12work".

**Usage**

```r
## S3 method for class 'x12work'
plot(x, plots=c(1:9), ...)
```

**Arguments**

- `x` an object of class "x12work".
- `plots` a vector containing numbers between 1 and 9.
- `...` further arguments (currently ignored).

**Details**

Plots:
1: Original
2: Original Trend Adjusted
3: Log Original
4: Seasonal Factors
5: Seasonal Factors with SI Ratios
6: Spectrum Adjusted Original
7: Spectrum Seasonal Adjusted
8: Spectrum Irregular
9: Spectrum Residuals

**Author(s)**

Alexander Kowarik

**See Also**

- `x12work`

**Examples**

```r
data(AirPassengersX12)
#plot(AirPassengersX12)
```
Description

Plot of the (partial) autocorrelations of the (squared) residuals with function `plotRsdAcf` in package `x12`.

Usage

```r
## S4 method for signature 'x12output'
plotRsdAcf(x, which="acf",
xlab="Lag", ylab="ACF",
main="default", col_acf="darkgrey", lwd_acf=4,
col_ci="blue", lt_ci=2, ylim="default", ...)
## S4 method for signature 'x12single'
plotRsdAcf(x, which="acf",
xlab="Lag", ylab="ACF",
main="default", col_acf="darkgrey", lwd_acf=4,
col_ci="blue", lt_ci=2, ylim="default", ...)
```

Arguments

- `x` object of class `x12output-class` or `x12Single-class`.
- `which` character specifying the type of autocorrelation of the residuals that should be plotted, i.e. the autocorrelations or partial autocorrelations of the residuals or the autocorrelations of the squared residuals ("acf", "pacf", "acf2").
- `xlab` label of the x-axis.
- `ylab` label of the y-axis.
- `main` plot title.
- `col_acf` color of the autocorrelations.
- `lwd_acf` line width of the autocorrelations.
- `col_ci` color of the +- 2 standard error limits.
- `lt_ci` line type of the +- 2 standard error limits.
- `ylim` range of the y-axis.
- `...` ignored.

Methods

```r
signature(x = "x12output")
signature(x = "x12Single")
```
Author(s)

Alexander Kowarik, Angelika Meraner

See Also

x12.plot, plotSpec, plotSeasFac

Examples

```r
## Not run:
s <- new("x12Single", ts=AirPassengers, tsName="air")
s <- setP(s, list(estimate=TRUE, regression.variables="AO1950.1", outlier.types="all",
    outlier.critical=list(LS=3.5, TC=2.5), backcast_years=1/2))
s <- x12(s)
# w/o outliers
plot(s@x12Output, sa=TRUE, trend=TRUE, original=FALSE)
plot(s)
# w/o (all) outliers
plot(s, showAllOut=TRUE, sa=TRUE, trend=TRUE, log_transform=TRUE, lwd_out=1, pch_ao=4)
plot(s, showAllOut=TRUE, sa=TRUE, trend=TRUE, original=FALSE, showAllOutLines=TRUE,
    col_tc="purple")#, log_transform=TRUE)#, lwd_out=3)
# with showOut
plot(s, showOut="AO1960.Jun", sa=FALSE, trend=FALSE, annComp=TRUE, log_transform=TRUE)
plot(s, showOut="AO1958.Mar", sa=TRUE, trend=TRUE, annComp=TRUE, annCompTrend=FALSE)
plot(s, showOut="AO1950.Jun", annComp=FALSE, cex_out=3, pch_ao=19, col_ao="orange")
plot(s, showOut="TC1954.Feb")
plot(s, showOut="TC1954.Feb", col_tc="green3")

# w/o legend
plot(s, showAllOut=TRUE, plot_legend=FALSE)
plot(s, plot_legend=FALSE)
plot(s, showOut="AO1950.1", plot_legend=FALSE, lwd_out=2, col_ao="purple")
plot(s, showOut="TC1954.Feb", col_tc="orange", col_ao="magenta", plot_legend=FALSE)
plot(s, showOut="AO1950.1", col_tc="orange", col_ao="magenta", plot_legend=FALSE)

# Forecasts & Backcasts
plot(s, forecast=TRUE)
plot(s, backcast=TRUE, showLine=TRUE)
plot(s, backcast=TRUE, forecast=TRUE, showCI=FALSE)
plot(s, forecast=TRUE, points_fc=TRUE, col_fc="purple", lty_fc=2, lty_original=3, lwd_fc=0.9,
    lwd_ci=2)
plot(s, sa=TRUE, plot_legend=FALSE)

# Seasonal Factors and SI Ratios
plotSeasFac(s)
# Spectra
plotSpec(s)
plotSpec(s, highlight=FALSE)
# Autocorrelations of the Residuals
plotRsdAcf(s)
plotRsdAcf(s, col_acf="black", lwd_acf=1)
```
Methods for Function `plotSeasFac` in Package `x12`

Description

Seasonal factor plots with function `plotSeasFac` in package `x12`.

Usage

```r
## S4 method for signature 'x12Output'
plotSeasFac(x, SI_Ratios=TRUE, ylab="Value", xlab="", lwd_seasonal=1, col_seasonal="black", lwd_mean=1, col_mean="blue", col_siratio="darkgreen", col_replaced="red", cex_siratio=.9, cex_replaced=.9, SI_Ratios_replaced=TRUE, plot_legend=TRUE, legend_horiz=FALSE, legend_bty="o", ...)
## S4 method for signature 'x12Single'
plotSeasFac(x, SI_Ratios=TRUE, ylab="Value", xlab="", lwd_seasonal=1, col_seasonal="black", lwd_mean=1, col_mean="blue", col_siratio="darkgreen", col_replaced="red", cex_siratio=.9, cex_replaced=.9, SI_Ratios_replaced=TRUE, plot_legend=TRUE, legend_horiz=FALSE, legend_bty="o", ...)
```

Arguments

- `x` object of class `x12Output-class` or `x12Single-class`.
- `SI_Ratios` logical specifying if the SI ratios should be plotted.
- `ylab` label of the y-axis.
- `xlab` label of the x-axis.
- `lwd_seasonal` line width of the seasonal factors.
- `col_seasonal` color of the seasonal factors.
- `lwd_mean` line width of the mean.
- `col_mean` color of the mean.
- `col_siratio` color of the SI ratios.
- `col_replaced` color of the replaced SI ratios.
- `cex_siratio` magnification factor for the size of the symbols used for plotting the SI ratios.
- `cex_replaced` magnification factor for the size of the symbols used for plotting the replaced SI ratios.
- `SI_Ratios_replaced` logical specifying if the replaced SI ratios should be plotted.
plotlegend logical specifying if a legend should be plotted.
legend_horiz Orientation of the legend
legend_bty the type of box to be drawn around the legend. The allowed values are "o" (the default) and "n".
... ignored.

Methods

signature(x = "x12Output")
signature(x = "x12Single")

Author(s)
Alexander Kowarik, Angelika Meraner

See Also
x12, plot, plotSpec, plotRsdAcf

Examples

```r
## Not run:
s <- new("x12Single", ts=AirPassengers, tsName="air")
s <- setP(s, list(estimate=TRUE, regression.variables="AO1950.1", outlier.types="all",
    outlier.critical=list(LS=3.5, TC=2.5), backcast_years=1/2))
s <- x12(s)
#w/o outliers
plot(s@x12Output, sa=TRUE, trend=TRUE, original=FALSE)
plot(s)
#with (all) outliers
plot(s, showAllOut=TRUE, sa=TRUE, trend=TRUE, log_transform=TRUE, lwd_out=1, pch_ao=4)
plot(s, showAllOut=TRUE, sa=TRUE, trend=TRUE, original=FALSE, showAllOutLines=TRUE,
    col_tc="purple", log_transform=TRUE, lwd_out=3)
# with showOut
plot(s, showOut="AO1950. Jun", sa=FALSE, trend=FALSE, annComp=TRUE, log_transform=TRUE)
plot(s, showOut="AO1958. Mar", sa=TRUE, trend=TRUE, annComp=TRUE, annCompTrend=FALSE)
plot(s, showOut="AO1950. Jun", annComp=FALSE, cex_out=3, pch_ao=19, col_ao="orange")
plot(s, showOut="TC1954. Feb")
plot(s, showOut="TC1954. Feb", col_tc="green3")

#w/o legend
plot(s, showAllOut=TRUE, plot_legend=FALSE)
plot(s, plot_legend=FALSE)
plot(s, showOut="AO1950.1", plot_legend=FALSE, lwd_out=2, col_ao="purple")
plot(s, showOut="TC1954. Feb", col_tc="orange", col_ao="magenta", plot_legend=FALSE)
plot(s, showOut="AO1950.1", col_tc="orange", col_ao="magenta", plot_legend=FALSE)

#Forecasts & Backcasts
plot(s, forecast=TRUE)
plot(s, backcast=TRUE, showLine=TRUE)
```
plot(s, backcast=TRUE, forecast=TRUE, showCI=FALSE)
plot(s, forecast=TRUE, points_fc=TRUE, col_fc="purple", lty_fc=2, lty_original=3,
lwd_fc=0.9, lwd_ci=2)
plot(s, sa=TRUE, plot_legend=FALSE)

# Seasonal Factors and SI Ratios
plotSeasFac(s)
# Spectra
plotSpec(s)
plotSpec(s, highlight=FALSE)

# Autocorrelations of the Residuals
plotRsdAcf(s)
plotRsdAcf(s, col_acf="black", lwd_acf=1)

## End(Not run)

---

**plotSpec**

~~ Methods for Function `plotSpec` in Package `x12` ~~

### Description

Spectral plots with function `plotSpec` in package `x12`.

### Arguments

- **x**: an object of class `x12Output-class`, `x12Single-class` or `spectrum-class`.
- **which**: a string defining the executable of the editor to use ("sa" for the Spectrum of the Seasonally Adjusted Series, "original" for the Spectrum of the Original Series, "irregular" for the Spectrum of the Irregular Series and "residuals" for the Spectrum of the RegARIMA Residuals).
- **frequency**: frequency of the time series (has to be specified for objects of class "spectrum" only).
- **xlab**: label of the x-axis.
- **ylab**: label of the y-axis.
- **main**: plot title.
- **col_bar**: color of bars.
- **col_seasonal**: color of seasonal frequencies.
- **col_td**: color of trading day frequencies.
- **lwd_bar**: line width of bars.
- **lwd_seasonal**: line width of seasonal frequencies.
- **lwd_td**: line width of trading day frequencies.
- **plot_legend**: logical specifying if a legend should be plotted.
Methods

signature(x = "x12output", which = "sa", xlab = "Frequency", ylab = "Decibels", main = "Spectrum", col_bar = "darkgrey", col_seasonal = "red", col_td = "blue", lwd_bar = 1, lwd_seasonal = 1, lwd_td = 1, plot_legend = TRUE)

signature(x = "x12single", which = "sa", xlab = "Frequency", ylab = "Decibels", main = "Spectrum", col_bar = "darkgrey", col_seasonal = "red", col_td = "blue", lwd_bar = 1, lwd_seasonal = 1, lwd_td = 1, plot_legend = TRUE)

signature(x = "spectrum", frequency, xlab = "Frequency", ylab = "Decibels", main = "Spectrum", col_bar = "darkgrey", col_seasonal = "red", col_td = "blue", lwd_bar = 1, lwd_seasonal = 1, lwd_td = 1, plot_legend = TRUE)

Author(s)
Alexander Kowarik, Angelika Meraner

See Also
x12, plot, plotSeasFac, plotRsdAcf

Examples

```r
### Not run:
s <- new("x12single", ts=AirPassengers, tsName="air")
s <- setP(s, list(estimate=TRUE, regression.variables="AO1950.1", outlier.types="all",
    outlier.critical=list(LS=3.5,TC=2.5), backcast_years=1/2))
s <- x12(s)
#w/o outliers
plot(s@x12Output, sa=TRUE, trend=TRUE, original=FALSE)
plot(s)
#with (all) outliers
plot(s, showAllout=TRUE, sa=TRUE, trend=TRUE, log_transform=TRUE, lwd_out=1, pch_ao=4)
plot(s, showAllout=TRUE, sa=TRUE, trend=TRUE, original=FALSE, showAlloutLines=TRUE, col_tc="purple")#, log_transform=TRUE)#, lwd_out=3)
#with showOut
plot(s, showOut=TRUE, sa=FALSE, trend=FALSE, annComp=TRUE, log_transform=TRUE)
plot(s, showOut=TRUE, sa=TRUE, trend=TRUE, annComp=TRUE, annCompTrend=FALSE)
plot(s, showOut=TRUE, annComp=FALSE, cex_out=3, pch_ao=19, col_ao="orange")
plot(s, showOut=TRUE, col_ao="orange")
plot(s, showOut=TRUE, col_ao="orange")
#w/o legend
plot(s, showAllout=TRUE, plot_legend=FALSE)
plot(s, plot_legend=FALSE)
plot(s, showOut=TRUE, plot_legend=FALSE)
plot(s, showOut=TRUE, col_tc="purple")
plot(s, showOut=TRUE, col_ao="magenta")
plot(s, showOut=TRUE, col_ao="magenta")
#Forecasts & Backcasts
plot(s, forecast=TRUE)
plot(s, backcast=TRUE, showLine=TRUE)
plot(s, backcast=TRUE, forecast=TRUE, showCI=FALSE)
plot(s, forecast=TRUE, points_fc=TRUE, col_fc="purple", lty_fc=2, lty_original=3,
    lwd_fc=0.9, lwd_ci=2)
```

plotSpec
Methods for Function `prev` and `cleanArchive` in Package `x12`

Description

Function `prev` in package `x12` reverts to previous parameter settings and output. Function `cleanHistory` resets `x12` parameter and output.

Usage

```r
## S4 method for signature 'x12Single'
prev(object, n=NULL)
## S4 method for signature 'x12Batch'
prev(object, index=NULL, n=NULL)
## S4 method for signature 'x12Single'
cleanHistory(object)
## S4 method for signature 'x12Batch'
cleanHistory(object, index=NULL)
```

Arguments

- `object` object of class `x12Single-class` or `x12Batch-class`.
- `n` index corresponding to a previous run.
- `index` index corresponding to (an) object(s) of class "x12Single".

Methods

```r
signature(object = "x12Single")
signature(object = "x12Batch")
```

Note

`cleanHistory` is deprecated and `cleanArchive` should be used instead.
readSpc

Author(s)
Alexander Kowarik

See Also
x12

Examples

data(AirPassengersX12)
summary(AirPassengersX12)
# a maximum of 10 previous x12 runs are added to the summary
summary(AirPassengersX12,oldOutput=10)
#the x12Parameter and x12Output of the x12Single is set to the previous run of x12
Ap=prev(AirPassengersX12)
summary(AirPassengersX12,oldOutput=10)

readSpc

Function to read X12-spc Files in x12Parameter R objects

Description
Still an early beta, so it will not work in specific situations

Usage
readSpc(file, filename=TRUE)

Arguments
file character vector containing filenames of spc files
filename if TRUE the filename (without) ".spc" will be used as name for the serie

Details
Not all arguments of an X12 spc file are supported, but the parameters described in x12 should be covered.

Value
The function returns an object of class "x12Single" if the file argument has length 1, otherwise it returns an "x12Batch" object.

Author(s)
Alexander Kowarik
spectrum-class

See Also

x12

Examples

```r
## Not run:
x12SingleObject1 <- readSpc("D:/aaa.spc")
x12SingleObject2 <- readSpc("D:/ak_b.SPC")
x12BatchObject1 <- readSpc(c("D:/ak_b.SPC","D:/aaa.spc"))
setwd("M:/kowarik/Test/x12test")
lf <- list.files()
lf <- lf[unlist(lapply(lf,function(x)substr(x,nchar(x)-2,nchar(x)))]
lf <- lf[-c(grep("ind",lf))]
allSPC <- readSpc(lf)
a <- x12(allSPC)
plot(a@x12List[[1]])
summary(a@x12List[[1]])

## End(Not run)
```

---

spectrum-class  

Class "spectrum"

Description

Standardized object for saving the spectrum output of the x12 binaries in R. Used by functions in this package.

Objects from the Class

Objects can be created by calls of the form `new("spectrum", ...)`. 

Slots

- `frequency`: Object of class "numeric"
- `spectrum`: Object of class "numeric"

Author(s)

Alexander Kowarik

Examples

`showClass("spectrum")`
Summary Methods

Methods for Function summary in Package x12

Description

Delivers a diagnostics summary for x12 output.

Usage

```r
## S4 method for signature 'x12Output'
summary(object, fullSummary=FALSE, spectra.detail=FALSE, 
  almostout=FALSE, rsd.autocorr=NULL, 
  quality.stat=FALSE, likelihood.stat=FALSE, aape=FALSE, id.rsdsseas=FALSE, 
  slidingspans=FALSE, 
  history=FALSE, identify=FALSE, print=TRUE)
## S4 method for signature 'x12Single'
summary(object, fullSummary=FALSE, spectra.detail=FALSE, 
  almostout=FALSE, rsd.autocorr=NULL, 
  quality.stat=FALSE, likelihood.stat=FALSE, aape=FALSE, id.rsdsseas=FALSE, 
  slidingspans=FALSE, 
  history=FALSE, identify=FALSE, oldOutput=NULL, print=TRUE)
## S4 method for signature 'x12Batch'
summary(object, fullSummary=FALSE, spectra.detail=FALSE, 
  almostout=FALSE, rsd.autocorr=NULL, 
  quality.stat=FALSE, likelihood.stat=FALSE, aape=FALSE, id.rsdsseas=FALSE, 
  slidingspans=FALSE, 
  history=FALSE, identify=FALSE, oldOutput=NULL, print=TRUE)
```

Arguments

- **object**: object of class `x12Output-class`, `x12Single-class` or `x12Batch-class`.
- **fullSummary**: logical defining whether all available optional diagnostics below should be included in the summary.
- **spectra.detail**: logical defining whether more detail on the spectra should be returned.
- **almostout**: logical defining whether "almost" outliers should be returned.
- **rsd.autocorr**: character or character vector specifying the type of autocorrelation of the residuals that should be returned, i.e. the autocorrelations and/or partial autocorrelations of the residuals and/or the autocorrelations of the squared residuals ("acf", "pacf", "paccf2").
- **quality.stat**: logical defining whether the second Q statistic, i.e. the Q Statistic computed w/o the M2 Quality Control Statistic, and the M statistics for monitoring and quality assessment should be returned as well.
- **likelihood.stat**: if TRUE, the likelihood statistics AIC, AICC, BIC and HQ are returned as well as the estimated maximum value of the log likelihood function of the model for the untransformed data.
aape logical defining whether the average absolute percentage error for forecasts should be returned.
id.rsds seas logical defining whether the presence/absence of residual seasonality should be indicated.
slidingspans logical defining whether the diagnostics output of the slidingspans analysis should be returned.
history logical defining whether the diagnostics output of the (revision) history analysis should be returned.
identify logical defining whether the (partial) autocorrelations of the residuals generated by the "identify" specification should be returned.
oldoutput integer specifying the number of previous x12 runs stored in the x12oldoutput slot of an x12Single-class or an x12Batch-class object that should be included in the summary.
print TRUE/FALSE if the summary should be printed.

Methods
signature(x = "x12Output")
signature(x = "x12Single")
signature(x = "x12Batch")

Author(s)
Alexander Kowarik, Angelika Meraner

See Also
prev, cleanArchive

Examples

## Not run:
# Summary of an "x12Single" object
x12path("../x12a.exe")
s <- new("x12Single",ts=AirPassengers,tSName="air")
s <- setP(s,list(estimate=TRUE,regression.variables="A01950.1",outlier.types="all",
    outlier.critical=list(LS=3.5,TC=2.5),backcast_years=1/2))
s <- x12(s)
summary.output<-summary(s)
s <- x12(setP(s,list(arima.model=c(0,1,1),arima.smodel=c(0,2,1))))
summary.output<-summary(s,oldOutput=1)
s <- x12(setP(s,list(arima.model=c(0,1,1),arima.smodel=c(1,0,1))))
summary.output<-summary(s,fullSummary=TRUE,oldOutput=2)

# Summary of an "x12Batch" object
xb <- new("x12Batch",list(AirPassengers,AirPassengers,
    AirPassengers),tsName=c("air1","air2","air3"))
Diagnostics summary for objects of class "x12work".

Usage

## S3 method for class 'x12work'
summary(object, fullSummary=FALSE, spectra.detail=FALSE, almostout=FALSE, rsd.autocorr=NULL, quality.stat=FALSE, likelihood.stat=FALSE, aape=FALSE, id.rsdseas=FALSE, slidingspans=FALSE, history=FALSE, identify=FALSE,...)

Arguments

object      an object of class "x12work".
fullSummary logical defining whether all available optional diagnostics below should be included in the summary.
spectra.detail logical defining whether more detail on the spectra should be returned.
almostout    logical defining whether "almost" outliers should be returned.
rsd.autocorr character or character vector specifying the type of autocorrelation of the residuals that should be returned, i.e. the autocorrelations and/or partial autocorrelations of the residuals and/or the autocorrelations of the squared residuals ("acf", "pacf", "acf2").
quality.stat logical defining whether the second Q statistic, i.e. the Q Statistic computed w/o the M2 Quality Control Statistic, and the M statistics for monitoring and quality assessment should be returned as well.
likelihood.stat if TRUE, the likelihood statistics AIC, AICC, BIC and HQ are returned as well as the estimated maximum value of the log likelihood function of the model for the untransformed data.
times

aape  logical defining whether the average absolute percentage error for forecasts should be returned.

id.rsdseas  logical defining whether the presence/absence of residual seasonality should be indicated.

slidingspans  logical defining whether the diagnostics output of the slidingspans analysis should be returned.

history  logical defining whether the diagnostics output of the (revision) history analysis should be returned.

identify  logical defining whether the (partial) autocorrelations of the residuals generated by the "identify" specification should be returned.

...  ignored at the moment

Details

Delivers a diagnostics summary.

Author(s)

Alexander Kowarik, Angelika Meraner

See Also

x12work, diagnostics-class, x12-methods

Examples

data(AirPassengers)
## Not run:
summary(x12work(AirPassengers,...),quality.stat=TRUE,res.autocorr="acf")
## End(Not run)

Description

Combination of start() and end() for ts objects-

Usage

times(x)
## S4 method for signature 'x12Output'
times(x)
## S4 method for signature 'x12Single'
times(x)
**Arguments**

- **x**  
  a x12Single or x12Output object

**Value**

Returns a list with start and end for original, backcast and forecast timeseries

**Methods**

signature(x = "x12output")
signature(x = "x12Single")

**Author(s)**

Alexander Kowarik

**See Also**

x12, x12Single, x12Batch, x12Parameter, x12List, x12Output, x12BaseInfo,
summary.x12work, x12work

---

**Description**

~~ Methods for Function x12 in Package x12 ~~

**Usage**

x12(object, x12Parameter=new("x12Parameter"), x12BaseInfo=new("x12BaseInfo"),...)

**Arguments**

- **object**  
  object of class ts, x12Single-class or x12Batch-class
- **x12Parameter**  
  object of class x12Parameter
- **x12BaseInfo**  
  object of class x12BaseInfo
- **...**  
  at the moment only forceRun=FALSE

**Methods**

signature(object = "ts")
signature(object = "x12Single")
signature(object = "x12Batch")
Value
An S4 object of class \texttt{x12Output-class} if object is of class \texttt{ts}
An S4 object of class \texttt{x12Single-class} if object is of class \texttt{x12Single-class}
An S4 object of class \texttt{x12Batch-class} if object is of class \texttt{x12Batch-class}

Note
Parallelization is implemented for \texttt{x12Batch} objects with help of the package ‘parallel’. To process
in parallel set the option ‘\texttt{x12.parallel}’ to an integer value representing the number of cores to use (options(x12.parallel=2)). Afterwards all calls to the function ‘\texttt{x12}’ on an object of class ‘\texttt{x12Batch}’
will be parallelized (For resetting use options(x12.parallel=NULL)).
cleanHistory is deprecated and cleanArchive should be used instead.

Author(s)
Alexander Kowarik, Angelika Meraner

Source
https://www.census.gov/srd/www/x13as/

References
Alexander Kowarik, Angelika Meraner, Matthias Templ, Daniel Schopfhauser (2014). Seasonal
Adjustment with the R Packages x12 and x12GUI. Journal of Statistical Software, 62(2), 1-21.
URL http://www.jstatsoft.org/v62/i02/.

See Also
\texttt{summary}, \texttt{plot}, \texttt{x12env}, \texttt{setP}, \texttt{getP}, \texttt{loadP}, \texttt{saveP}, \texttt{prev}, \texttt{cleanArchive}, \texttt{crossVal}

Examples
\begin{verbatim}
xts <- x12(AirPassengers)
summary(xts)
xs <- x12(new("x12Single",ts=AirPassengers))
summary(xs)

## Not run:
xb<-x12(new("x12Batch",list(AirPassengers,AirPassengers,AirPassengers)))
summary(xb)

#Create new batch object with 4 time series
xb <- new("x12Batch",list(AirPassengers,AirPassengers,AirPassengers,AirPassengers))

# change the automdl to FALSE in all 4 elements
xb <- setP(xb,list(automdl=FALSE))
# change the arima.model and arima.smodel setting for the first ts object
xb <- setP(xb,list(arima.model=c(1,1,0),arima.smodel=c(1,1,0)),1)
# change the arima.model and arima.smodel setting for the second ts object
\end{verbatim}
xb <- setP(xb,list(arima.model=c(0,1,1),arima.smodel=c(0,1,1)),2)
#change the arima.model and arima.smodel setting for the third ts object
xb <- setP(xb,list(arima.model=c(0,1,1),arima.smodel=c(1,1,1)),3)
#change the arima.model and arima.smodel setting for the fourth ts object
xb <- setP(xb,list(arima.model=c(1,1,1),arima.smodel=c(1,1,1)),4)
#run x12 on all series
xb <- x12(xb)
summary(xb)

#Set automdl=TRUE for the first ts
xb <- setP(xb,list(automdl=TRUE),1)

#rerun x12 on all series (the binaries will only run on the first one)
xb <- x12(xb)

#summary with oldOutput
summary(xb,oldOutput=10)

#Change the parameter and output of the first series back to the first run
xb <- prev(xb,index=1,n=1)
#summary with oldOutput (--- No valid previous runs. ---)
summary(xb,oldOutput=10)

## End(Not run)

---

x12BaseInfo-class  Class "x12BaseInfo"

Description

Baseinfo for use with the x12 function and classes.

Objects from the Class

Objects can be created by calls of the form new("x12BaseInfo", x12path, use, showWarnings).

Slots

x12path: Object of class "character0rNULL" ~
use: Object of class "character" ~
showWarnings: Object of class "logical" ~

Methods

No methods defined with class "x12BaseInfo" in the signature.

Author(s)

Alexander Kowarik
See Also
$x12, x12Single, x12Batch, x12Parameter, x12List, x12Output$

Examples

```r
showClass("x12BaseInfo")
```

---

**Description**

Concatenation of multiple `x12Single-class` objects.

**Objects from the Class**

Objects can be created by calls of the form `new("x12Batch", tsList, tsName, x12BaseInfo)`.

**Slots**

- `x12List`: Object of class "x12List"
- `x12BaseInfo`: Object of class "x12BaseInfo"

**Methods**

- `setP` signature(object = "x12Batch"): ...
- `getP` signature(object = "x12Batch"): ...
- `prev` signature(object = "x12Batch"): ...
- `cleanArchive` signature(object = "x12Batch"): ...
- `loadP` signature(object = "x12Batch"): ...
- `saveP` signature(object = "x12Batch"): ...
- `summary` signature(object = "x12Batch"): ...
- `x12` signature(object = "x12Batch"): ...
- `dim` signature(x = "x12Batch"): ...
- `length` signature(x = "x12Batch"): ...
- `cleanHistory` signature(object = "x12Batch"): ...

**Note**

`cleanHistory` is deprecated and `cleanArchive` should be used instead.

**Author(s)**

Alexander Kowarik


References


See Also

x12, x12Single,

x12Parameter, x12List, x12Output, x12BaseInfo, summary, getP,

x12work

Examples

```r
# Not run:
# object containing 4 time series and the corresponding parameters and output
data(AirPassengersX12Batch)
summary(AirPassengersX12Batch)
# summary with oldOutput
summary(AirPassengersX12Batch, oldOutput=10)
# Change the parameter and output of the first series back to the first run
AirPassengersX12Batch <- prev(AirPassengersX12Batch, index=1, n=1)
# summary with oldOutput (--- No valid previous runs. ---)
summary(AirPassengersX12Batch, oldOutput=10)

# Create new batch object with 4 time series
xb <- new("x12Batch", list(AirPassengers, Ideath, nottem, UKgas),
  c("Air","Ideath","nottem","UKgas"))
# change outlier.types to "all" in all 4 elements
xb <- setP(xb, list(outlier.types="all"))
# change the arima.model and arima.smodel setting for the first ts object
xb <- setP(xb, list(arima.model=c(0,1,1), arima.smodel=c(0,1,1)), 1)
# change the arima.model and arima.smodel setting for the second ts object
xb <- setP(xb, list(arima.model=c(0,1,1), arima.smodel=c(0,1,1)), 2)
# change the arima.model and arima.smodel setting for the third ts object
xb <- setP(xb, list(arima.model=c(0,1,1), arima.smodel=c(0,1,1)), 3)
# change the arima.model and arima.smodel setting for the fourth ts object
xb <- setP(xb, list(arima.model=c(0,1,1), arima.smodel=c(0,1,1)), 4)
# run x12 on all series
xb <- x12(xb)
summary(xb)
# Set automdl=TRUE for the first ts
xb <- setP(xb, list(automdl=TRUE), 1)
# rerun x12 on all series (the binaries will only run on the first one)
xb <- x12(xb)
# summary with oldOutput
summary(xb, oldOutput=10)
# Change the parameter and output of the first series back to the first run
xb <- prev(xb, index=1, n=1)
# summary with oldOutput (--- No valid previous runs. ---)
summary(xb, oldOutput=10)
```
#Create new batch object by combining objects of class x12Single
s1 <- new("x12Single", ts=AirPassengers, tsName="air")
s1 <- setP(s1, list(estimate=TRUE, regression.variables="AO1950.1", outlier.types="all", outlier.critical=list(LS=3.5, TC=2.5)))
s2 <- new("x12Single", ts=UKgas, tsName="UKgas")
s2 <- setP(s2, list(slidingspans=TRUE, history=TRUE,
    history.estimates=c("adj", "adjchng", "trend", "trendchng", "seasonal", "aic"),
    history.sadjlags=c(1,2), automdl=TRUE))
b <- new("x12Batch", list(s1, s2))
b <- x12(b)

## End(Not run)

---

**x12List-class**

*Class* "x12List"

---

**Description**

Support class for **x12Batch-class** containing multiple **x12Single-class**.

**Objects from the Class**

Objects can be created by calls of the form `new("x12List", ...)`.

**Slots**

/data: Object of class "list" ~~

**Extends**

Class "list", from data part. Class "vector", by class "list", distance 2.

**Methods**

No methods defined with class "x12List" in the signature.

**Author(s)**

Alexander Kowarik

**See Also**

x12, x12Single, x12Batch, x12Parameter,
x12Output, x12BaseInfo

**Examples**

`showClass("x12List")`
Description

Output class for \texttt{x12}.

Objects from the Class

Objects can be created by calls of the form \texttt{new("x12output", ...)}. 

Slots

\begin{itemize}
\item \texttt{a1}: Object of class "ts" - the original time series.
\item \texttt{d10}: Object of class "ts" - the final seasonal factors.
\item \texttt{d11}: Object of class "ts" - the final seasonally adjusted data.
\item \texttt{d12}: Object of class "ts" - the final trend cycle.
\item \texttt{d13}: Object of class "ts" - the final irregular components.
\item \texttt{d16}: Object of class "ts" - the combined adjustment factors.
\item \texttt{c17}: Object of class "ts" - the final weights for the irregular component.
\item \texttt{d9}: Object of class "ts" - the final replacements for the SI ratios.
\item \texttt{e2}: Object of class "ts" - the differenced, transformed, seasonally adjusted data.
\item \texttt{d8}: Object of class "ts" - the final unmodified SI ratios.
\item \texttt{b1}: Object of class "ts" - the prior adjusted original series.
\item \texttt{td}: Object of class "tsOrNULL" - the trading day component
\item \texttt{ot1}: Object of class "tsOrNULL" - the outlier regression series
\item \texttt{sp0}: Object of class "spectrum" - the spectrum of the original series.
\item \texttt{sp1}: Object of class "spectrum" - the spectrum of the differenced seasonally adjusted series.
\item \texttt{sp2}: Object of class "spectrum" - the spectrum of modified irregular series.
\item \texttt{spr}: Object of class "spectrum" - the spectrum of the regARIMA model residuals.
\item \texttt{forecast}: Object of class "fcast" - the point forecasts with prediction intervals
\item \texttt{backcast}: Object of class "fcast" - the point backcasts with prediction intervals
\item \texttt{dg}: Object of class "list", containing several seasonal adjustment and regARIMA modeling diagnostics, i.e.:
  \begin{itemize}
  \item x11regress, transform, samode, seasonalma, trendma, arimamdl, automdl, regmdl, nout, nautoout, nalmostout, ...
  \end{itemize}
\item \texttt{file}: Object of class "character" - path to the output directory and filename
\item \texttt{tblnames}: Object of class "character" - tables read into R
\item \texttt{Rtblnames}: Object of class "character" - names of tables read into R
\end{itemize}
**Parameter-class**

Methods

- **summary** signature(object = "x12output"): ...
- **plot** signature(object = "x12output"): ...
- **plotSpec** signature(object = "x12output"): ...
- **plotSeasFac** signature(object = "x12output"): ...
- **plotRsdAcf** signature(object = "x12output"): ...

Author(s)

Alexander Kowarik, Angelika Meraner

See Also

- x12, x12Single, x12Batch, x12Parameter, x12List, x12output, x12BaseInfo,
- summary.x12work, x12work

Examples

```r
data(AirPassengersX12)
summary(AirPassengersX12)
showClass("x12output")
```

---

**x12Parameter-class**

*Class "x12Parameter"*

**Description**

Parameter class for x12.

**Objects from the Class**

Objects can be created by calls of the form new("x12Parameter", ...).

**Slots**

- `series.span`: Object of class "numericOrNULLOrcharacter" - vector of length 4, limiting the data used for the calculations and analysis to a certain time interval.
  
  Start and end date of said time interval can be specified by 4 integers in the format c(start year, start seasonal period, end year, end seasonal period). If the start or end date of the time series object should be used, the respective year and seasonal period are to be set to NA.

- `series.modelspan`: Object of class "numericOrNULLOrcharacter" - vector of length 4, defining the start and end date of the time interval of the data that should be used to determine all regARIMA model coefficients. Specified in the same way as span.

- `transform.function`: Object of class "character" - transform parameter for x12 ("auto", "log", "none").
Parameter-class

transform.power: Object of class "numericOrNULL" - numeric value specifying the power of the
Box Cox power transformation.

transform.adjust: Object of class "characterOrNULL" - determines the type of adjustment to be
performed, i.e. transform.adjust="lon" for length-of-month adjustment on monthly data,
transform.adjust="log" for length-of-quarter adjustment on quarterly data or transform.adjust="lpyear"
for leap year adjustment of monthly or quarterly data (which is only allowed when either
transform.power=0 or transform.function="log").

regression.variables: Object of class "characterOrNULL" - character or character vector rep-
resenting the names of the regression variables.

regression.user: Object of class "characterOrNULL" - character or character vector defining
the user parameters in the regression argument.

regression.file: Object of class "characterOrNULL" - path to the file containing the data val-
ues of all regression.user variables.

regression.usertype: Object of class "characterOrNULL" - character or character vector ass-
signing a type of model-estimated regression effect on each user parameter in the regression
argument ("seasonal", "td", "lpyear", "user", ...).
By specifying a character vector of length greater one each variable can be given its own type.
Otherwise the same type will be used for all user parameters.

regression.centeruser: Object of class "characterOrNULL" - character specifying the removal
of the (sample) mean or the seasonal means from the user parameters in the regression argu-
ment ("mean", "seasonal").
Default is no modification of the respective user-defined regressors.

regression.start: Object of class "numericOrNULLOrcharacter" - start date for the values of
the regression.user variables, specified as a vector of two integers in the format c(year, seasonal period).

regression.aictest: Object of class "characterOrNULL" - character vector defining the regres-
sion variables for which an AIC test is to be performed.

outlier.types: Object of class "characterOrNULL" - to enable the "outlier" specification in the
spc file, this parameter has to be defined by a character or character vector determining the
method(s) used for outlier detection ("AO", "LS", "TC", "all").

outlier.critical: Object of class "listOrNULLOrnumeric" - number specifying the critical
value used for outlier detection (same value used for all types of outliers) or named list (pos-
sible names of list elements being A0,LS and TC) where each list element specifies the respective
critical value used for detecting the corresponding type of outlier.
If not specified, the default critical value is used.

outlier.span: Object of class "numericOrNULLOrcharacter" - vector of length 4, defining the
span for outlier detection. Specified in the same way as span.

outlier.method: Object of class "characterOrNULL" - character determining how detected out-
liers should be added to the model ("addone", "addall"). If not specified,"addone" is used
by default.

identify: Object of class "logical" - if TRUE, the "identify" specification will be enabled in the
spc file.

identify.diff: Object of class "numericOrNULL" - number or vector representing the orders of
nonseasonal differences specified, default is 0.

identify.sdiff: Object of class "numericOrNULL" - number or vector representing the orders of
seasonal differences specified, default is 0.
identify.maxlag: Object of class "numericOrNULL" - number of lags specified for the ACFs and PACFs, default is 36 for monthly series and 12 for quarterly series.

arima.model: Object of class "numericOrNULL" - vector of length 3, defining the arima parameters.

arima.smodel: Object of class "numericOrNULL" - vector of length 3, defining the sarima parameters.

arima.ar: Object of class "numericOrNULLOrcharacter" - numeric or character vector specifying the initial values for nonseasonal and seasonal autoregressive parameters in the order that they appear in the arima.model argument. Empty positions are created with NA.

arima.ma: Object of class "numericOrNULLOrcharacter" - numeric or character vector specifying the initial values for all moving average parameters in the order that they appear in the arima.model argument. Empty positions are created with NA.

automdl: Object of class "logical" - TRUE/FALSE for activating auto modeling.

automdl.acceptdefault: Object of class "logical" - logical for automdl defining whether the default model should be chosen if the Ljung-Box Q statistic for its model residuals is acceptable.

automdl.balanced: Object of class "logical" - logical for automdl defining whether the automatic model procedure will tend towards balanced models. TRUE yields the same preference as the TRAMO program.

automdl.maxorder: Object of class "numeric" - vector of length 2, specifying the maximum order for automdl. Empty positions are created with NA.

automdl.maxdiff: Object of class "numeric" - vector of length 2, specifying the maximum diff. order for automdl. Empty positions are created with NA.

forecast.years: Object of class "numericOrNULL" - number of years to forecast, default is 1 year.

backcast.years: Object of class "numericOrNULL" - number of years to backcast, default is no backcasts.

forecast.conf: Object of class "numeric" - probability for the confidence interval of forecasts.

estimate: Object of class "logical" - if TRUE, the term "estimate" will be added to the spc file.

estimate.outofsample: Object of class "logical" - logical defining whether "out of sample" or "within sample" forecast errors should be used in calculating the average magnitude of forecast errors over the last three years.

check: Object of class "logical" - TRUE/FALSE for activating the "check" specification in the spc file.

check.maxlag: Object of class "numericOrNULL" - the number of lags requested for the residual sample ACF and PACF, default is 24 for monthly series and 8 for quarterly series.

slidingspans: Object of class "logical" - if TRUE, "slidingspans" specification will be enabled in the spc file.

slidingspans.fixmdl: Object of class "characterOrNULL" - ("yes" (default), "no", "clear").

slidingspans.fixreg: Object of class "characterOrNULL" - character or character vector specifying the trading day, holiday, outlier or other user-defined regression effects to be fixed ("td", "holiday", "outlier", "user"). All other regression coefficients will be re-estimated for each sliding span.
slidingspans.length: Object of class "numericOrNULL" - numeric value specifying the length of each span in months or quarters (>3 years, <17 years).

slidingspans.numspans: Object of class "numericOrNULL" - numeric value specifying the number of sliding spans used to generate output for comparisons (must be between 2 and 4, inclusive).

slidingspans.outlier: Object of class "characterOrNULL" - "keep" (default), "remove", "yes".

slidingspans.additivesa: Object of class "characterOrNULL" - "difference" (default), "percent".

slidingspans.start: Object of class "numericOrNULLOrcharacter" - specified as a vector of two integers in the format c(start year, start seasonal period).

history: if TRUE, the history specification will be enabled.

history.estimates: Object of class "characterOrNULL" - character or character vector determining which estimates from the regARIMA modeling and/or the x11 seasonal adjustment will be analyzed in the history analysis ("sadj" (default), "sadjchng", "trend", "trendchng", "seasonal", "aic", "fcst").

history.fixmdl: Object of class "logical" - logical determining whether the regARIMA model will be re-estimated during the history analysis.

history.fixreg: Object of class "characterOrNULL" - character or character vector specifying the trading day, holiday, outlier or other user-defined regression effects to be fixed ("td", "holiday", "outlier", "user"). All other coefficients will be re-estimated for each history span.

history.outlier: Object of class "characterOrNULL" - "keep" (default), "remove", "auto"

history.sadjlags: Object of class "numericOrNULL" - integer or vector specifying up to 5 revision lags (each >0) that will be analyzed in the revisions analysis of lagged seasonal adjustments.

history.trendlags: Object of class "numericOrNULL" - integer or vector specifying up to 5 revision lags (each >0) that will be used in the revision history of the lagged trend components.

history.start: Object of class "numericOrNULLOrcharacter" - specified as a vector of two integers in the format c(start year, start seasonal period).

history.target: Object of class "characterOrNULL" - character determining whether the revisions of the seasonal adjustments and trends calculated at the lags specified in history.sadjlags and history.trendlags should be defined by the deviation from the concurrent estimate or the deviation from the final estimate ("final" (default), "concurrent").

x11.signalim: Object of class "numericOrNULL" - vector of length 2, defining the limits for sigma in the x11 methodology, used to downweight extreme irregular values in the internal seasonal adjustment iterations.

x11.type: Object of class "characterOrNULL" - character, i.e. "summary", "trend" or "sa". If x11.type="trend", x11 will only be used to estimate the final trend-cycle as well as the irregular components and to adjust according to trading days. The default setting is type="sa" where a seasonal decomposition of the series is calculated.

x11.sfshort: Object of class "logical" - logical controlling the seasonal filter to be used if the series is at most 5 years long. If TRUE, the arguments of the seasonalma filter will be used wherever possible. If FALSE, a stable seasonal filter will be used irrespective of seasonalma.

x11.samode: Object of class "characterOrNULL" - character defining the type of seasonal adjustment decomposition calculated ("mult", "add", "pseudoadd", "logadd").
x11.seasonalma: Object of class "characterOrNULL" - character or character vector of the format c("snxm", "snxm", ...) defining which seasonal nxm moving average(s) should be used for which calendar months or quarters to estimate the seasonal factors. If only one ma is specified, the same ma will be used for all months or quarters. If not specified, the program will invoke an automatic choice.

x11.trendma: Object of class "numericOrNULL" - integer defining the type of Henderson moving average used for estimating the final trend cycle. If not specified, the program will invoke an automatic choice.

x11.appendfcst: Object of class "logical" - logical defining whether forecasts should be included in certain x11 tables.

x11.appendbcst: Object of class "logical" - logical defining whether forecasts should be included in certain x11 tables.

x11.calendarsigma: Object of class "characterOrNULL" - regulates the way the standard errors used for the detection and adjustment of extreme values should be computed ("all", "signif", "select" or no specification).

x11.excludfcst: Object of class "logical" - logical defining if forecasts and backcasts from the regARIMA model should not be used in the generation of extreme values in the seasonal adjustment routines.

x11.final: Object of class "character" - character or character vector specifying which type(s) of prior adjustment factors should be removed from the final seasonally adjusted series ("AO", "LS", "TC", "user", "none").

x11.regression: Object of class "logical" - if TRUE, x11Regression will be performed using the respective regression and outlier commands above, i.e. regression.variables, regression.user, regression.file, regression.usertype, regression.centeruser and regression.start as well as outlier.critical, outlier.span and outlier.method.

Methods

getP signature(object = "x12Parameter"): ...

setP signature(object = "x12Parameter"): ...

Author(s)

Alexander Kowarik, Angelika Meraner

Examples

showClass("x12Parameter")
**Description**

"x12env" is used to store the x12path and x13path (and more for the GUI).

**Usage**

```r
x12env
gx12path(path=NULL)

putd(x, value)
getd(x, mode="any")
rmd(x)
existd(x, mode="any")
```

**Arguments**

- `path` The path to the X12 or X13 binaries.
- `x` a character for the name
- `value` value that should be assigned to the object with name `x`.
- `mode` the mode or type of object sought

**Author(s)**

Alexander Kowarik

**See Also**

`get`, `assign`, `exists`, `x12`

**Examples**

```r
## Not run:
x12path()
x12path("d:/x12/x12a.exe")
x12path()
getd("x12path")

## End(Not run)
```
x12Single-class  Class "x12Single"

Description

Class consisting of all information for x12.

Objects from the Class

Objects can be created by calls of the form new("x12Single", ...).

Slots

ts: Object of class ts
x12Parameter: Object of class x12Parameter-class
x120output: Object of class x12Output-class
x120ldParameter: Object of class list
x120ld0output: Object of class list
tsName: Object of class character0rNULL
firstRun: Object of class logical

Methods

setP signature(object = "x12Single")
getP signature(object = "x12Single")
prev signature(object = "x12Single")
cleanArchive signature(object = "x12Single")
loadP signature(object = "x12Single")
saveP signature(object = "x12Single")
summary signature(object = "x12Single")
x12 signature(object = "x12Single")
plot signature(object = "x12Single")
crossVal signature(object = "x12Single")
plotSpec signature(object = "x12Single")
plotSeasFac signature(object = "x12Single")
plotRsdAcf signature(object = "x12Single")
cleanHistory signature(object = "x12Single")

Note

cleanHistory is deprecated and cleanArchive should be used instead.
Author(s)

Alexander Kowarik

See Also

x12, x12Batch, x12Parameter, x12List, x12Output, x12BaseInfo, summary, getP, x12work

Examples

```r
## Not run:
s <- new("x12Single", ts=AirPassengers, tsName="air")
s <- setP(s, list(estimate=TRUE, regression.variables="AO1950.1", outlier.types="all",
                  outlier.critical=list(LS=3.5, TC=2.5)))
s <- x12(s)

## End(Not run)
```

---

Run x12 on an R TS-object

Description

A wrapper function for the x12 binaries. It creates a specification file for an R time series and runs x12, afterwards the output is read into R.

Usage

```r
x12work(tso, period=frequency(tso), file="Rout",
         series.span=.NULL, series.modelspan=NULL,
         transform.function="auto", transform.power=NULL, transform.adjust=NULL,
         regression.variables= NULL, regression.user=NULL, regression.file=NULL,
         regression.usertype=NULL, regression.centeruser=NULL, regression.start=NULL,
         regression.aictest= NULL,
         outlier.types=NULL, outlier.critical=NULL, outlier.span=NULL, outlier.method=NULL,
         identify=FALSE, identify.diff=NULL, identify.sdiff=NULL, identify.maxlag=NULL,
         arima.model= NULL, arima.smodel=NULL, arima.ar=NULL, arima.ma=NULL,
         automdl=FALSE, automdl.acceptdefault=FALSE, automdl.balanced=TRUE,
         automdl.maxorder= c(3,2), automdl.maxdiff= c(1,1),
         forecast_years=NULL, backcast_years=NULL, forecast_conf=.95,
         estimate=FALSE, estimate.outofsample=TRUE,
         check=TRUE, check.maxlag=NULL,
         slidingspans=FALSE,
         slidingspans.fixmdl=NULL, slidingspans.fixreg=NULL,
         slidingspans.length=NULL, slidingspans.numspans=NULL,
         slidingspans.outlier=NULL,
```

---
slidingspans.additivesa= NULL, slidingspans.start= NULL, history= FALSE, history. estimates= NULL, history. fixmdl= FALSE, history. fixreg= NULL, history. outlier= NULL, history. sadjlags= NULL, history. trendlags= NULL, history. start= NULL, history. target= NULL, x11. sigmalim= c(1.5, 2.5), x11. type= NULL, x11. sfshort= FALSE, x11. samode= NULL, x11. seasonalma= NULL, x11. trendma= NULL, x11. appendfcst= TRUE, x11. appendbcst= FALSE, x11. calendarsigma= NULL, x11. excludefcst= TRUE, x11. final= "user", x11. regression= FALSE, tblnames= NULL, Rtblnames= NULL, x12path= NULL, use= "x12", keep_x12out= TRUE, showWarnings= TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tso</td>
<td>a time series object.</td>
</tr>
<tr>
<td>period</td>
<td>frequency of the time series.</td>
</tr>
<tr>
<td>file</td>
<td>path to the output directory and filename, default is the working directory and Rout.*.</td>
</tr>
<tr>
<td>series.span</td>
<td>vector of length 4, limiting the data used for the calculations and analysis to a certain time interval. Start and end date of said time interval can be specified by 4 integers in the format c(start year, start seasonal period, end year, end seasonal period) If the start or end date of the time series object should be used, the respective year and seasonal period are to be set to NA.</td>
</tr>
<tr>
<td>series.modelspan</td>
<td>vector of length 4, defining the start and end date of the time interval of the data that should be used to determine all regARIMA model coefficients. Specified in the same way as span.</td>
</tr>
<tr>
<td>transform.function</td>
<td>transform parameter for x12 (&quot;auto&quot;, &quot;log&quot;, &quot;none&quot;).</td>
</tr>
<tr>
<td>transform.power</td>
<td>numeric value specifying the power of the Box Cox power transformation.</td>
</tr>
<tr>
<td>transform.adjust</td>
<td>determines the type of adjustment to be performed, i.e. transform.adjust=&quot;lom&quot; for length-of-month adjustment on monthly data, transform.adjust=&quot;log&quot; for length-of-quarter adjustment on quarterly data or transform.adjust=&quot;lpyear&quot; for leap year adjustment of monthly or quarterly data (which is only allowed when either transform.power=0 or transform.function=&quot;log&quot;).</td>
</tr>
<tr>
<td>regression.variables</td>
<td>character or character vector representing the names of the regression variables.</td>
</tr>
<tr>
<td>regression.user</td>
<td>character or character vector defining the user parameters in the regression argument.</td>
</tr>
</tbody>
</table>
regression.file
path to the file containing the data values of all regression.user variables.

regression.usertype
character or character vector assigning a type of model-estimated regression
effect on each user parameter in the regression argument ("seasonal", "td", "lpyear", "user", ...).
By specifying a character vector of length greater one each variable can be given
its own type. Otherwise the same type will be used for all user parameters.

regression.centeruser
character specifying the removal of the (sample) mean or the seasonal means
from the user parameters in the regression argument ("mean", "seasonal").
Default is no modification of the respective user-defined regressors.

regression.start
start date for the values of the regression.user variables, specified as a vector
of two integers in the format c(year, seasonal period).

regression.aictest
character vector defining the regression variables for which an AIC test is to be
performed.

outlier.types
to enable the "outlier" specification in the spc file, this parameter has to be de-
dined by a character or character vector determining the method(s) used for out-
lier detection ("AO", "LS", "TC", "all").

outlier.critical
number specifying the critical value used for outlier detection (same value used
for all types of outliers) or named list (possible names of list elements being
AO, LS and TC) where each list element specifies the respective critical value used
for detecting the corresponding type of outlier.
If not specified, the default critical value is used.

outlier.span
vector of length 2, defining the span for outlier detection.

outlier.method
character determining how detected outliers should be added to the model ("addone",
"addall"). If not specified, "addone" is used by default.

identify
Object of class "logical" - if TRUE, the "identify" specification will be enabled
in the spc file.

identify.diff
number or vector representing the orders of nonseasonal differences specified,
default is 0.

identify.sdff
number or vector representing the orders of seasonal differences specified, de-
fault is 0.

identify.maxlag
number of lags specified for the ACFs and PACFs, default is 36 for monthly
series and 12 for quarterly series.

arima.model
vector of length 3, defining the arima parameters.

arima.smodel
vector of length 3, defining the sarima parameters.

arima.ar
numeric or character vector specifying the initial values for nonseasonal and sea-
sonal autoregressive parameters in the order that they appear in the arima.model
argument. Empty positions are created with NA.
arima.ma numeric or character vector specifying the initial values for all moving average parameters in the order that they appear in the arima.model argument. Empty positions are created with NA.

automdl TRUE/FALSE for activating auto modeling.

automdl.acceptdefault
logical for automdl defining whether the default model should be chosen if the Ljung-Box Q statistic for its model residuals is acceptable.

automdl.balanced
logical for automdl defining whether the automatic model procedure will tend towards balanced models. TRUE yields the same preference as the TRAMO program.

automdl.maxorder
vector of length 2, maximum order for automdl. Empty positions are created with NA.

automdl.maxdiff
vector of length 2, maximum diff. order for automdl. Empty positions are created with NA.

forecast_years number of years to forecast, default is 1 year.

backcast_years number of years to backcast, default is no backcasts.

forecast_conf probability for the confidence interval of forecasts

estimate if TRUE, the term "estimate" will be added to the spc file.

estimate.outofsample
logical defining whether "out of sample" or "within sample" forecast errors should be used in calculating the average magnitude of forecast errors over the last three years.

check TRUE/FALSE for activating the "check" specification in the spc file.

check.maxlag the number of lags requested for the residual sample ACF and PACF, default is 24 for monthly series and 8 for quarterly series.

slidingspans if TRUE, "slidingspans" specification will be enabled in the spc file.

slidingspans.fixmdl
("
yes" (default), "no", "clear").

slidingspans.fixreg
character or character vector specifying the trading day, holiday, outlier or other user-defined regression effects to be fixed ("td", "holiday", "outlier", "user"). All other regression coefficients will be re-estimated for each sliding span.

slidingspans.length
numeric value specifying the length of each span in months or quarters (>3 years, <17 years).

slidingspans.numspans
numeric value specifying the number of sliding spans used to generate output for comparisons (must be between 2 and 4, inclusive).

slidingspans.outlier
("
keep" (default), "remove", "yes").
slidingspans.additivesa
("difference" (default), "percent").

slidingspans.start
specified as a vector of two integers in the format (start year, start seasonal period).

history
if TRUE, the history specification will be enabled.

history.estimates
character or character vector determining which estimates from the regARIMA modeling and/or the x11 seasonal adjustment will be analyzed in the history analysis ("sadj" (default), "sadjchng", "trend", "trendchng", "seasonal", "aic", "fcst").

history.fixmdl
logical determining whether the regARIMA model will be re-estimated during the history analysis.

history.fixreg
character or character vector specifying the trading day, holiday, outlier or other user-defined regression effects to be fixed ("td", "holiday", "outlier", "user"). All other coefficients will be re-estimated for each history span.

history.outlier
("keep" (default), "remove", "auto")

history.sadjlags
integer or vector specifying up to 5 revision lags (each >0) that will be analyzed in the revisions analysis of lagged seasonal adjustments.

history.trendlags
integer or vector specifying up to 5 revision lags (each >0) that will be used in the revision history of the lagged trend components.

history.start
specified as a vector of two integers in the format (start year, start seasonal period).

history.target
character determining whether the revisions of the seasonal adjustments and trends calculated at the lags specified in history.sadjlags and history.trendlags should be defined by the deviation from the concurrent estimate or the deviation from the final estimate ("final" (default), "concurrent").

x11.signalim
vector of length 2, defining the limits for sigma in the x11 methodology, used to downweight extreme irregular values in the internal seasonal adjustment iterations.

x11.type
character, i.e. "summary", "trend" or "sa". If x11.type="trend", x11 will only be used to estimate the final trend-cycle as well as the irregular components and to adjust according to trading days. The default setting is type="sa" where a seasonal decomposition of the series is calculated.

x11.sfshort
logical controlling the seasonal filter to be used if the series is at most 5 years long. If TRUE, the arguments of the seasonalma filter will be used wherever possible. If FALSE, a stable seasonal filter will be used irrespective of seasonalma.

x11.samode
character defining the type of seasonal adjustment decomposition calculated ("mult", "add", "pseudoadd", "logadd").

x11.seasonalma
character or character vector of the format c("snxm","snxm", ...) defining which seasonal nxm moving average(s) should be used for which calendar months or quarters to estimate the seasonal factors. If only one ma is specified, the same ma will be used for all months or quarters. If not specified, the program will invoke an automatic choice.
x11.trendma integer defining the type of Henderson moving average used for estimating the final trend cycle. If not specified, the program will invoke an automatic choice.

x11.appendfcst logical defining whether forecasts should be included in certain x11 tables.

x11.appendbcst logical defining whether forecasts should be included in certain x11 tables.

x11.calendarsigma regulates the way the standard errors used for the detection and adjustment of extreme values should be computed ("all", "signif", "select" or no specification).

x11.excludefcst logical defining if forecasts and backcasts from the regARIMA model should not be used in the generation of extreme values in the seasonal adjustment routines.

x11.final character or character vector specifying which type(s) of prior adjustment factors should be removed from the final seasonally adjusted series ("AO", "LS", "TC", "user", "none").

x11.regression if TRUE, x11Regression will be performed using the respective regression and outlier commands above, i.e. regression.variables, regression.user, regression.file, regression.usertype, regression.centeruser and regression.start as well as outlier.critical, outlier.span and outlier.method.

tblnames character vector of additional tables to be read into R.

Rtblnames character vector naming the additional tables.

x12path path to the x12 binaries, for example d:\x12\x12a.exe.

use "x12" or "x13", at the moment only "x12" is tested properly.

keep_x12out if TRUE, the output files generated by x12 are stored in the folder "gra" in the output directory and are not deleted at the end of a successful run.

showWarnings logical defining whether warnings and notes generated by x12 should be returned. Errors will be displayed in any case.

Details

Generates an x12 specification file, runs x12 and reads the output files.

Value

x12work returns an object of class "x12".

The function summary is used to print a summary of the diagnostics results.

An object of class "x12" is a list containing at least the following components:

a1 original time series
d10 final seasonal factors
d11 final seasonally adjusted data
d12 final trend cycle
d13 final irregular components
d16 combined adjustment factors
Note

Only working with available x12 binaries.

Author(s)

Alexander Kowarik, Angelika Meraner

Source

https://www.census.gov/srd/www/x13as/

References


See Also

x12, ts, summary.x12work, plot.x12work, x12-methods

Examples

```r
### Examples
data(AirPassengers)
## Not run:
x12out <- x12work(AirPassengers, x12path=".../x12a.exe", transform.function="auto",
arima.model=c(0,1,1), arima.model=c(0,1,1), regression.variables="lpyear",
x11.sigma1im=c(2.0,3.0), outlier.types="all", outlier.critical=list(LS=3.5,TC=3),
x11.seasonalma="s3x3")
summary(x12out)
## End(Not run)
```
Index

*Topic \textasciitilde\textasciitilde other possible keyword(s) \textasciitilde\textasciitilde
plotRsdAcf, 15
plotSeasFac, 17
plotSpec, 19
prev-methods, 21
*Topic \texttt{aplot}
plot\_x12work, 14
*Topic \texttt{asummary}
summary\_x12work, 26
x12Output-class, 34
*Topic \texttt{classes}
crossValidation-class, 5
diagnostics-class, 6
fbcast-class, 6
spectrum-class, 23
x12BaseInfo-class, 30
x12Batch-class, 31
x12List-class, 33
x12Parameter-class, 35
x12Single-class, 41
*Topic \texttt{datasets}
AirPassengers\_x12, 2
AirPassengers\_x12Batch, 3
*Topic \texttt{manip}
x12path, 40
*Topic \texttt{methods}
crossVal, 3
getP-methods, 7
loadP, 8
plotRsdAcf, 15
plotSeasFac, 17
plotSpec, 19
prev-methods, 21
x12, 28
AirPassengers\_x12, 2
AirPassengers\_x12Batch, 3
assign, 40
class, 47
cleanArchive, 25, 29, 31, 41
cleanArchive (prev-methods), 21
cleanArchive, x12Batch-method (prev-methods), 21
cleanArchive, x12Single-method (prev-methods), 21
cleanArchive-methods (prev-methods), 21
cleanHistory, 31, 41
cleanHistory (prev-methods), 21
cleanHistory, x12Batch-method (prev-methods), 21
cleanHistory, x12Single-method (prev-methods), 21
cleanHistory-methods (prev-methods), 21
crossVal, 3, 29, 41
crossVal, ts-method (crossVal), 3
crossVal, x12Single-method (crossVal), 3
crossVal-methods (crossVal), 3
crossValidation-class, 5
diagnostics-class, 6
dim, x12Batch-method (x12Batch-class), 31
existsd (x12path), 40
exists, 40
fbcast-class, 6
get, 40
getd (x12path), 40
getP, 29, 31, 32, 39, 41, 42
getP (getP-methods), 7
getP, x12Batch-method (getP-methods), 7
getP, x12Parameter-method (getP-methods), 7
getP, x12Single-method (getP-methods), 7
getP-methods, 7
length, x12Batch-method (x12Batch-class), 31
x12path, 40
x12Single, 7, 28, 31–33, 35
x12Single (x12Single-class), 41
x12Single-class, 41
x12work, 14, 27, 28, 32, 35, 42, 42