Package: rjd3x13 (via r-universe)

October 22, 2026

Type Package

Title Seasonal Adjustment with 'X-13' in 'JDemetra+' 3.x

Version 3.9.0

Description R Interface to 'JDemetra+' 3.x
(<https://github.com/jdemetra>) time series analysis software.
It offers full access to options and outputs of 'X-13',
including Reg-ARIMA modelling (automatic AutoRegressive
Integrated Moving Average (ARIMA) model with outlier detection
and trading days adjustment) and X-11 decomposition.

License EUPL

URL https://github.com/rjdverse/rjd3x13,
https://rjdverse.github.io/rjd3x13/

BugReports https://github.com/rjdverse/rjd3x13/issues

Depends R (>= 4.1.0)

Imports rJava (>= 1.0-6), rjd3jars (>= 0.0.6), rjd3toolkit (>= 3.9.0),
RProtoBuf (>= 0.4.25)

Config/roxygen2/version 8.1.0

Encoding UTF-8

SystemRequirements Java (>= 21)

Collate 'deprecated.R' 'print.R' 'regarima_generic.R' 'utils.R'
'regarima_outliers.R' 'regarima_spec.R' 'x13_rslts.R'
'x13_spec.R' 'revisions.R' 'rjd3x13-package.R' 'set_x11_spec.R'
'x13.R' 'zzz.R'

NeedsCompilation no

Author Jean Palate [aut], Alain Quartier-la-Tente [aut] (ORCID:
<https://orcid.org/0000-0001-7890-3857>), Tanguy Barthelemy
[aut, cre, art, cph], Eulalie Delaune [aut], Anna Smyk [aut]

Maintainer Tanguy Barthelemy <timeserieswithjdemetraandr@gmail.com>

Config/pak/sysreqs default-jdk

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

Date/Publication 2026-09-22 12:30:02 UTC

RemoteUrl https://github.com/cran/rjd3x13

RemoteRef HEAD

RemoteSha 3369aa6b29e477744946665eb79914854715201c

Contents

deprecated-rjd3x13 . . . 2
get_x13_option . . . 3
jd3_utilities . . . 4
refresh . . . 5
regarima . . . 9
regarima_outliers . . . 10
set_x11 . . . 11
x11 . . . 14
x13 . . . 14
x13_dictionary . . . 16
x13_option . . . 17
x13_revisions . . . 18
x13_spec . . . 19

Index **21**

`deprecated-rjd3x13` *Deprecated functions*

Description

Deprecated functions

Usage

```
spec_x13(name = c("rsa4", "rsa0", "rsa1", "rsa2c", "rsa3", "rsa5c"))

spec_regarima(name = c("rg4", "rg0", "rg1", "rg2c", "rg3", "rg5c"))

spec_x11()

fast_x13(
  ts,
  spec = c("rsa4", "rsa0", "rsa1", "rsa2c", "rsa3", "rsa5c"),
  context = NULL,
  userdefined = NULL
)

fast_regarima(
```

```
  ts,
  spec = c("rg4", "rg0", "rg1", "rg2c", "rg3", "rg5c"),
  context = NULL,
  userdefined = NULL
)

.jx13(
  ts,
  spec = c("rg4", "rg0", "rg1", "rg2c", "rg3", "rg5c"),
  context = NULL,
  userdefined = NULL
)

userdefined_variables_x13(x = c("X-13", "RegArima", "X-11"))
```

Arguments

name	name of a predefined specification.
ts	an univariate time series.
spec	the model specification. Can be either the name of a predefined specification or a user-defined specification.
context	list of external regressors (calendar or other) to be used for estimation
userdefined	a vector containing additional output variables (see x13_dictionary()).
x	useless parameter

Value

All these functions are deprecated and return the same value as the function that replaces them:

- spec_x13() returns the same value as x13_spec()
- spec_regarima() returns the same value as regarima_spec()
- spec_x11() returns the same value as x11_spec()
- fast_x13() returns the same value as x13_fast()
- fast_regarima() returns the same value as regarima_fast()
- .jx13() returns the same value as jx13()
- userdefined_variables_x13() returns the same value as x13_dictionary()

---

get_x13_option *Set an option for x13*

Description

Set an option for x13

Usage

```
get_x13_option(name)
```

Arguments

name	Name of the option

Value

The requested option or NULL if it doesn't exist

Examples

```
x13_option("test", "DUMMY")
get_x13_option("test")
```

---

jd3_utilities *Java Utility Functions*

Description

These functions are used in all JDemetra+ 3.0 packages to easily interact between R and Java objects.

Usage

```
.x13_rslts(jrslts)

.jd2r_spec_x11(jspec)

.r2jd_spec_x11(spec)

.r2jd_spec_regarima(spec)

.jd2r_spec_regarima(jspec)

.r2jd_spec_x13(spec)

.jd2r_spec_x13(jspec)
```

Arguments

spec, jspec, jrslts	parameters.

Value

These functions return specification in Java, proto or R.

refresh *Refresh a specification with constraints*

Description

Functions x13_refresh and regarima_refresh allow to create a new specification by updating an existing one. Some selected parameters will be kept fixed while others will be freed within the boundaries of a reference specification. In practice each freed parameter of the specification to be updated (spec) is replaced by the corresponding parameter of the reference specification (refspec). See details and examples.

Usage

```
regarima_refresh(
  spec,
  refspec = NULL,
 policy = c("FreeParameters", "Complete", "Outliers_StochasticComponent", "Outliers",
    "FixedParameters", "FixedAutoRegressiveParameters", "Fixed", "Current"),
  period = 0,
  start = NULL,
  end = NULL
)

x13_refresh(
  spec,
  refspec = NULL,
 policy = c("FreeParameters", "Complete", "Outliers_StochasticComponent", "Outliers",
    "FixedParameters", "FixedAutoRegressiveParameters", "Fixed", "Current"),
  period = 0,
  start = NULL,
  end = NULL
)
```

Arguments

spec	specification to be refreshed Object of class "JD3_X13_SPEC" or "JD3_REGARIMA_SPEC", can be obtained as an output of x13_spec or regarima_spec and customised with set_ functions, see x13_spec documentation
refspec	reference specification By default "RG4c" or "rsa4" specification. Object of class "JD3_X13_SPEC" or "JD3_REGARIMA_SPEC", can be obtained as an output of x13_spec or regarima_spec and customised with set_ functions, see x13_spec documentation
policy	refresh policy to apply (see details)
period, start, end	additional parameters used to specify the span when policy = "Current" or policy = "Outliers" or policy = "Outliers_StochasticComponent" period:

numeric, number of observations in a year (12, 4...), compulsory, if false or missing, re-estimation with refreshed specification won't work. When `policy = "Outliers"` or `policy = "Outliers_StochasticComponent"` `start` has to be specified as the date from which outliers will be re-identified `end` is not used, if specified it will be ignored. When `policy = "Current"` `start` and `end` have to be both specified and indicate the span on which additive outliers (AO) will be added. Span definition: `start` and `end`: defined as arrays of two elements: year and first period (for example, `period = 12` and `start=c(1980, 1)` stands for January 1980)

Details

A particular selection of parameters to be kept fixed or re-estimated is called a revision policy.

Available refresh policies are:

1. **Current**: applying the current pre-adjustment reg-arima model from and handling the new raw data points, or any sub-span of the series as Additive Outliers (defined as new intervention variables); X11 and Benchmarking part parameters are untouched.
2. **Fixed**: applying the current pre-adjustment reg-arima model and replacing forecasts by new raw data points; X11 and Benchmarking part parameters are untouched.
3. **FixedParameters**: pre-adjustment reg-arima model is partially modified: regression coefficients will be re-estimated but regression variables, Arima orders and coefficients are unchanged;
4. **FixedAutoRegressiveParameters**: same as FixedParameters but Arima Moving Average coefficients (MA) are also re-estimated, Auto-regressive (AR) coefficients are kept fixed; X11 and Benchmarking part parameters are untouched.
5. **FreeParameters**: all regression and Arima model coefficients are re-estimated, regression variables and Arima orders are kept fixed; X11 and Benchmarking part parameters are untouched.
6. **Outliers**: regression variables and Arima orders are kept fixed, but outliers will be re-detected on the defined span, thus all regression and Arima model coefficients are re-estimated; X11 and Benchmarking part parameters are untouched.
7. **Outliers_StochasticComponent**: same as "Outliers" but Arima model orders (p,d,q)(P,D,Q) can also be re-identified; X11 and Benchmarking part parameters are untouched.
8. **Complete**: All the parameters are re-identified and re-estimated, unless constrained in the reference spec. X11 and Benchmarking part parameters are entirely reset to values in the reference specification.

Value

a new specification, an object of class `"JD3_X13_SPEC"` or `"JD3_REGARIMA_SPEC"`.

References

More information on revision policies in JDemetra+ documentation: https://doc.jdemetra.org/a-rev-policies

Examples

```
library("rjd3toolkit")
y <- rjd3toolkit::ABS$X0.2.08.10.M
# raw series for first estimation
y_raw <- window(y, end = c(2016, 12))
# raw series for second (refreshed) estimation: new data points
y_new <- window(y, end = c(2017, 6))

# Example 1 : refresh mechanism
# Create reference spec, here the default "rsa3"
rsa3<- x13_spec("rsa3")
# Customize this spec
## Reg-Arima part
### For example, disable automatic arima modelling
user_spec <- set_automodel(rsa3, enabled = FALSE)
### set a user-defined arima model
user_spec <- set_arima(
   user_spec,
   mean = 0.2,
   mean.type = "Fixed",
   p = 1,
   d = 2,
   q = 0,
   bp = 1,
   bd = 1,
   bq = 0,
   coef = c(0.6, 0.7),
   coef.type = c("Initial", "Fixed")
)
#print(user_spec)

## Customize the x11 part
user_spec<-set_x11(user_spec,
                   lsigma = 2,
                   usigma = 3,
                   fcasts = -2,
                   bcasts = -1)
#print(user_spec)
user_spec<- set_benchmarking(
   user_spec,
   enabled = TRUE,
   target = "Original",
   rho = 0.7,
   lambda = 0.5,
   forecast = TRUE,
   bias = "Multiplicative")
#print(user_spec)
# Use policy: "Outliers_StochasticComponent"
x13_spec_ref <- x13_refresh(spec= user_spec,
                            refspec= rsa3,
                            policy = "Outliers_StochasticComponent"
)
```

```
# print(x13_spec_ref)
# user defined reg-arima model is reset and outliers will be re-identified
# on the whole series as no start and end specified, X11 and Benchmarking parameters
# are left unchanged
# Use policy: "Complete"
x13_spec_ref <- x13_refresh(spec= user_spec,
                            refspec= rsa3,
                            policy = "Complete"
)
# print(x13_spec_ref)
# all user defined parameters are reset and replaced with "rsa3" parameters,
# including for X11 and Benchmarking parameters

# Example 2 : practical re-estimation use-case
sa_x13 <- x13(y_raw, user_spec)
# refreshing the specification resulting from the first estimation
# to partially adapt it to new data
spec_to_refresh <- sa_x13$result_spec
reference_spec <- sa_x13$estimation_spec
# policy = "Fixed"
spec_x13_ref <- x13_refresh(spec_to_refresh,
                            reference_spec,
                            policy = "Fixed"
)
# 2nd estimation with refreshed specification
sa_x13_ref <- x13(y_new, spec_x13_ref)
# policy = "Outliers"
spec_x13_ref <- x13_refresh(spec_to_refresh,
                            reference_spec,
                            policy = "Outliers",
                            period = 12,
                            start = c(2017, 1)
)
# outliers will be re-detected from January 2017 included
# 2nd estimation with refreshed specification
sa_x13_ref <- x13(y_new, spec_x13_ref)

# policy = "Current"

spec_x13_ref <- x13_refresh(spec_to_refresh,
                            reference_spec,
                            policy = "Current",
                            period = 12,
                            start = c(2017, 1),
                            end = end(y_new)
)
# Points from January 2017 (included) until the end of the series will be
# treated as Additive Outliers, the previous reg-Arima model being otherwise
# kept fixed 2nd estimation with refreshed specification
sa_x13_ref <- x13(y_new, spec_x13_ref)
```

regarima *RegARIMA model, pre-adjustment in X13*

Description

RegARIMA model, pre-adjustment in X13

Usage

```
regarima(
  ts,
  spec = c("rg4", "rg0", "rg1", "rg2c", "rg3", "rg5c"),
  context = NULL,
  userdefined = NULL
)

regarima_fast(
  ts,
  spec = c("rg4", "rg0", "rg1", "rg2c", "rg3", "rg5c"),
  context = NULL,
  userdefined = NULL
)
```

Arguments

ts	an univariate time series.
spec	the model specification. Can be either the name of a predefined specification or a user-defined specification.
context	list of external regressors (calendar or other) to be used for estimation
userdefined	a vector containing additional output variables (see x13_dictionary()).

Value

the regarima() function returns a list with the results ("JD3_REGARIMA_RSLTS" object), the estimation specification and the result specification, while regarima_fast() is a faster function that only returns the results.

Examples

```
library("rjd3toolkit")

y <- ABS$X0.2.09.10.M
sp <- regarima_spec("rg5c")
sp <- add_outlier(sp,
    type = c("AO"), c("2015-01-01", "2010-01-01")
)
```

```
regarima_fast(y, spec = sp)
sp <- set_transform(
    set_tradingdays(
        set_easter(sp, enabled = FALSE),
        option = "workingdays"
    ),
    fun = "None"
)
regarima_fast(y, spec = sp)
sp <- set_outlier(sp, outliers.type = c("AO"))
regarima_fast(y, spec = sp)
```

`regarima_outliers` *Outlier Detection with a RegARIMA Model*

Description

Outlier Detection with a RegARIMA Model

Usage

```
regarima_outliers(
  y,
  order = c(0L, 1L, 1L),
  seasonal = c(0L, 1L, 1L),
  mean = FALSE,
  X = NULL,
  X.td = NULL,
  ao = TRUE,
  ls = TRUE,
  tc = FALSE,
  so = FALSE,
  cv = 0,
  clean = FALSE
)
```

Arguments

`y`	the dependent variable (a `ts` object).
`order`, `seasonal`	the orders of the ARIMA model.
`mean`	Boolean to include or not the mean.
`X`	user defined regressors (other than calendar).
`X.td`	calendar regressors.
`ao`, `ls`, `so`, `tc`	Boolean to indicate which type of outliers should be detected.

cv	numeric. The entered critical value for the outlier detection procedure. If equal to 0 the critical value for the outlier detection procedure is automatically determined by the number of observations.
clean	Clean missing values at the beginning/end of the series. Regression variables are automatically resized, if need be.

Value

a "JD3_REGARIMA_OUTLIERS" object, containing input variables and results

Examples

```
# estimate model
model <- regarima_outliers(rjd3toolkit::ABS$X0.2.09.10.M)
# print outliers
model$model$variables
```

---

set_x11 *Set X-11 Specification*

Description

Set X-11 Specification

Usage

```
set_x11(
  x,
  mode = c(NA, "Undefined", "Additive", "Multiplicative", "LogAdditive",
    "PseudoAdditive"),
  seasonal.comp = NA,
  seasonal.filter = NA,
  henderson.filter = NA,
  lsigma = NA,
  usigma = NA,
  fcasts = NA,
  bcasts = NA,
  calendar.sigma = c(NA, "None", "Signif", "All", "Select"),
  sigma.vector = NA,
  exclude.forecast = NA,
  bias = c(NA, "LEGACY", "SMOOTH", "RATIO")
)
```

Arguments

x	the specification to be modified, object of class "JD3_X11_SPEC", default X11 spec can be obtained as 'x=x11_spec()'
mode	character: the decomposition mode in X11. Determines the mode of the seasonal adjustment decomposition to be performed: If a pre-adjustment is performed mode is inherited: • if log transformation in pre-adjustment, X11 mode is "Multiplicative" by default (or can also be "LogAdditive" if set by the user) • if no log transformation, X11 mode is "Additive" by default (or can also be "PseudoAdditive" if set by the user) • If NO pre-adjustment is performed, just X-11 decomposition using x = x11_spec() spec, mode is "Multiplicative" by default, it can be set to "Additive", "LogAdditive", "PseudoAdditive" (could be changed by the program, if needed). If set to "Undefined" an additive decomposition is performed, see details.
seasonal.comp	logical: if TRUE, the program computes a seasonal component. Otherwise, the seasonal component is not estimated and its values are all set to 0 (additive decomposition) or 1 (multiplicative decomposition).
seasonal.filter	a vector of character(s) specifying which seasonal moving average (i.e. seasonal filter) will be used to estimate the seasonal factors for the entire series. The vector can be of length: 1 - the same seasonal filter is used for all periods (e.g.: seasonal.filter = "Msr" or seasonal.filter = "S3X3"); or have a different value for each quarter (length 4) or each month (length 12) - (e.g. for quarterly series: seasonal.filter = c("S3X3", "Msr", "S3X3", "Msr")). Possible filters are: "Msr", "Stable", "X11Default", "S3X1", "S3X3", "S3X5", "S3X9", "S3X15". "Msr" - the program chooses the final seasonal filter automatically.
henderson.filter	numeric: the length of the Henderson filter (odd number between 3 and 101). If henderson.filter = 0 an automatic selection of the Henderson filter's length for the trend estimation is enabled.
lsigma	numeric: the lower sigma boundary for the detection of extreme values, > 0.5, default=1.5.
usigma	numeric: the upper sigma boundary for the detection of extreme values, > lsigma, default=2.5.
bcasts, fcasts	numeric: the number of backcasts (bcasts) or forecasts (fcasts) generated by the RegARIMA model in periods (positive values) or years (negative values).Default values: fcasts=-1 and bcasts=0.
calendar.sigma	character to specify if the standard errors used for extreme values detection and adjustment are computed: from 5 year spans of irregulars ("None", default value); separately for each calendar period ("All"); separately for each period only if Cochran's hypothesis test determines that the irregular component is heteroskedastic by calendar month/quarter ("Signif"); separately for two complementary sets of calendar months/quarters specified by the x11.sigmaVector

parameter ("Select", see parameter sigma.vector).If sigma.vector is specified then calendar.sigma is set to "Select"‘.

sigma.vector — a vector to specify one of the two groups of periods for which standard errors used for extreme values detection and adjustment will be computed separately. Possible values are: 1 or 2.

exclude.forecast — Boolean to exclude forecasts and backcasts. If TRUE, the RegARIMA model forecasts and backcasts are not used during the detection of extreme values in the seasonal adjustment routines. Default = FALSE.

bias — If mode = "LogAdditive" (only in this case) a correction is applied when computing final components (S, T, I) in level from components estimated in log. Then bias= "RATIO" (default) Average of S (on complete years) and I (on the whole series) is set to 1, the correction is allocated to T.). "LEGACY" or "SMOOTH" corrections are based on trend filters.

Details

Decomposition mode formulas (Y: raw series, T: trend, S: seasonal, I: Irregular, SA: seasonally adjusted series)

- Additive: Y=T+S+I, SA =Y-S=T+I
- Multiplicative Y=T*S*I, SA =Y/S=T*I
- LogAdditive Log(Y) = T + S + I, SA=exp(T+I)=Y/exp(S)
- PseudoAdditive Y=T*(S+I-1), SA=T*I

Value

a "JD3_X11_SPEC" object, containing all the parameters.

See Also

x13_spec() and x11_spec().

Examples

```
init_spec <- x13_spec()
new_spec <- set_x11(init_spec,
    mode = "LogAdditive",
    seasonal.comp = 1,
    seasonal.filter = "S3X9",
    henderson.filter = 7,
    lsigma = 1.7,
    usigma = 2.7,
    fcasts = -1,
    bcasts = -1,
    calendar.sigma = "All",
    sigma.vector = NA,
    exclude.forecast = FALSE,
    bias = "RATIO"
```

```
)
```

---

x11 *X-11 Decomposition Algorithm*

Description

X-11 Decomposition Algorithm

Usage

```
x11(ts, spec = x11_spec(), userdefined = NULL)
```

Arguments

ts	an univariate time series.
spec	the specification.
userdefined	a vector containing additional output variables (see x13_dictionary()).

Value

the x11() function returns a list with the results (series) and final parameters

Examples

```
y <- rjd3toolkit::ABS$X0.2.09.10.M
x11_spec <- x11_spec()
x11(y, x11_spec)
x11_spec <- set_x11(x11_spec, henderson.filter = 13)
x11(y, x11_spec)
```

---

x13 *Seasonal Adjustment with X13-ARIMA*

Description

Seasonal Adjustment with X13-ARIMA

Usage

```
x13(
  ts,
  spec = c("rsa4", "rsa0", "rsa1", "rsa2c", "rsa3", "rsa5c"),
  context = NULL,
  userdefined = NULL
)

x13_fast(
  ts,
  spec = c("rsa4", "rsa0", "rsa1", "rsa2c", "rsa3", "rsa5c"),
  context = NULL,
  userdefined = NULL
)

jx13(
  ts,
  spec = c("rsa4", "rsa0", "rsa1", "rsa2c", "rsa3", "rsa5c"),
  context = NULL,
  userdefined = NULL
)
```

Arguments

ts	an univariate time series.
spec	the model specification. Can be either the name of a predefined specification or a user-defined specification.
context	list of external regressors (calendar or other) to be used for estimation
userdefined	a vector containing additional output variables (see x13_dictionary()).

Value

the x13() function returns a list with the results, the estimation specification and the result specification, while x13_fast() is a faster function that only returns the results. The jx13() functions only returns results in a java object which will allow to customize outputs in other packages (use rjd3toolkit::dictionary() to get the list of variables and rjd3toolkit::result() to get a specific variable). In the estimation functions x13() and x13_fast() you can directly use a specification name (string). If you want to customize a specification you have to create a specification object first.

Examples

```
library("rjd3toolkit")

y <- rjd3toolkit::ABS$X0.2.09.10.M
x13_fast(y, "rsa3")
x13(y, "rsa5c")
regarima_fast(y, "rg0")
```

```
regarima(y, "rg3")

sp <- x13_spec("rsa5c")
sp <- add_outlier(sp,
    type = c("AO"), c("2015-01-01", "2010-01-01")
)
sp <- set_transform(
    set_tradingdays(
        set_easter(sp, enabled = FALSE),
        option = "workingdays"
    ),
    fun = "None"
)
x13(y, spec = sp)
sp <- set_x11(sp,
    henderson.filter = 13
)
x13_fast(y, spec = sp)
j <- jx13(y, spec = sp)
class(j)
```

`x13_dictionary` *X-13 Dictionary*

Description

Functions to provide information for all output objects (series, diagnostics, parameters) available with `x13()` function.

Usage

```
x13_dictionary()

x13_full_dictionary()
```

Details

These functions provide lists of output names (series, diagnostics, parameters) available with the `x13()` function. These names can be used to generate customized outputs with the userdefined option of the `x13()` function (see examples). The `x13_full_dictionary` function provides additional information on object format and description.

Value

`x13_dictionary()` returns a character vector containing the names of all output objects (series, diagnostics, parameters) available with the `x13()` function, whereas `x13_full_dictionary()` returns a `data.frame` with format and description, for all the output objects.

Examples

```
library("rjd3toolkit")

# Visualize the dictionary
print(x13_dictionary())
summary(x13_dictionary())

# first 10 lines
head(x13_full_dictionary(), n = 10)
# For more structured information call `View(x13_full_dictionary())`

# Extract names of output of interest
user_defined_output <- x13_dictionary()[c(65, 95, 135)]
user_defined_output

# Generate the corresponding output in an estimation
y <- ABS$X0.2.09.10.M
m <- x13(y,"rsa3", userdefined=user_defined_output)

# Retrieve user defined output
tail(m$user_defined$ylin)
m$user_defined$residuals.kurtosis
m$user_defined$sa_f
```

`x13_option` *Set an option for x13*

Description

Set an option for x13

Usage

```
x13_option(name, obj)
```

Arguments

`name`	Name of the option
`obj`	Option

Value

Invisibly returns `NULL`

Examples

```
x13_option("test", "DUMMY")
```

x13_revisions *Revisions History*

Description

Compute revisions history

Usage

```
x13_revisions(
  ts,
  spec,
  data_ids = NULL,
  ts_ids = NULL,
  cmp_ids = NULL,
  context = NULL
)
```

Arguments

ts	The time series used for the estimation.
spec	The specification used.
data_ids	A list of list to specify the statistics to export. Each sub-list must contain two elements: start (first date to compute the history, in the format "YYYY-MM-DD") and id (the name of the statistics, see x13_dictionary()). See example.
ts_ids	A list of list to specify the specific date of a component whose history is to be studied. Each sub-list must contain three elements: start (first date to compute the history, in the format "YYYY-MM-DD"), period (the date of the studied) and id (the name of the component, see x13_dictionary()). See example.
cmp_ids	A list of list to specify the component whose history is to be studied. Each sub-list must contain three elements: start (first date to compute the history, in the format "YYYY-MM-DD"), end (last date to compute the history, in the format "YYYY-MM-DD") and id (the name of the component, see x13_dictionary()). As many series as periods between start and end will be exported. See example.
context	The context of the specification.

Value

returns a list

Examples

```
library("rjd3toolkit")

s <- ABS$X0.2.09.10.M
```

```
sa_mod <- x13(s)
data_ids <- list(
    # Get the coefficient of the trading-day coefficient from 2005-jan
    list(start = "2005-01-01", id = "regression.td(1)"),
    # Get the ljung-box statistics on residuals from 2010-jan
    list(start = "2010-01-01", id = "residuals.lb")
)
ts_ids <- list(
    # Get the SA component estimates of 2010-jan from 2010-jan
    list(period = "2010-01-01", start = "2010-01-01", id = "sa"),
    # Get the irregular component estimates of 2010-jan from 2015-jan
    list(period = "2010-01-01", start = "2015-01-01", id = "i")
)
cmp_ids <- list(
    # Get the SA component estimates (full time series) 2010-jan to 2020-jan
    list(start = "2010-01-01", end = "2020-01-01", id = "sa"),
    # Get the trend component estimates (full time series)  2010-jan to 2020-jan
    list(start = "2010-01-01", end = "2020-01-01", id = "t")
)
rh <- x13_revisions(s, sa_mod$result_spec, data_ids, ts_ids, cmp_ids)
rh$data
rh$series
rh$components
```

---

x13_spec *RegARIMA/X-13 Default Specifications*

Description

Set of functions to create default specification objects associated with X-13ARIMA seasonal adjustment method.

Specification setting of sheer X-11 decomposition method (without reg-arima pre-adjustment) is supported by `x11_spec()` function only and doesn't appear among possible X13-Arima default specifications.

Specification setting can be restricted to the reg-arima part with `regarima_spec()` function, without argument `regarima_spec()` yields a RG5c specification.

Setting a complete X13-Arima spec, `x13_spec()` without argument yields a RSA5c specification.

Usage

```
regarima_spec(name = c("rg4", "rg0", "rg1", "rg2c", "rg3", "rg5c"))

x13_spec(name = c("rsa4", "rsa0", "rsa1", "rsa2c", "rsa3", "rsa5c"))

x11_spec()
```

Arguments

name	name of a predefined specification.

Details

Available predefined 'JDemetra+' model specifications are described in the table below:

Identifier	Log/level detection	Outliers detection	Calendar effects	ARIMA
RSA0/RG0	*NA*	*NA*	*NA*	Airline(+mean)
RSA1/RG1	automatic	AO/LS/TC	*NA*	Airline(+mean)
RSA2c/RG2c	automatic	AO/LS/TC	2 td vars + Easter	Airline(+mean)
RSA3/RG3	automatic	AO/LS/TC	*NA*	automatic
RSA4c/RG4c	automatic	AO/LS/TC	2 td vars + Easter	automatic
RSA5c/RG5c	automatic	AO/LS/TC	7 td vars + Easter	automatic

Value

an object of class `"JD3_X13_SPEC"` (`x13_spec()`), `"JD3_REGARIMA_SPEC"` (`regarima_spec()`) or `"JD3_X11_SPEC"` (`x11_spec()`).

See Also

- To set the pre-processing parameters: `rjd3toolkit::set_arima()`, `rjd3toolkit::set_automodel()`, `rjd3toolkit::set_basic()`, `rjd3toolkit::set_easter()`, `rjd3toolkit::set_estimate()`, `rjd3toolkit::set_outlier()`, `rjd3toolkit::set_tradingdays()`, `rjd3toolkit::set_transform()`, `rjd3toolkit::add_outlier()`, `rjd3toolkit::remove_outlier()`, `rjd3toolkit::add_ramp()`, `rjd3toolkit::remove_ramp()`, `rjd3toolkit::add_usrdefvar()`.
- To set the decomposition parameters: `set_x11()`.
- To set the benchmarking parameters: `rjd3toolkit::set_benchmarking()`.

Examples

```
init_spec <- x11_spec()
init_spec
init_spec <- regarima_spec("rg4")
init_spec
init_spec <- x13_spec("rsa5c")
init_spec
```

Index

.jd2r_spec_regarima *(jd3_utilities)*, 4
.jd2r_spec_x11 *(jd3_utilities)*, 4
.jd2r_spec_x13 *(jd3_utilities)*, 4
.jx13 *(deprecated-rjd3x13)*, 2
.r2jd_spec_regarima *(jd3_utilities)*, 4
.r2jd_spec_x11 *(jd3_utilities)*, 4
.r2jd_spec_x13 *(jd3_utilities)*, 4
.x13_rslts *(jd3_utilities)*, 4

deprecated-rjd3x13, 2

fast_regarima *(deprecated-rjd3x13)*, 2
fast_x13 *(deprecated-rjd3x13)*, 2

get_x13_option, 3

jd3_utilities, 4
jx13 *(x13)*, 14

refresh, 5
regarima, 9
regarima_fast *(regarima)*, 9
regarima_outliers, 10
regarima_refresh *(refresh)*, 5
regarima_spec *(x13_spec)*, 19
rjd3toolkit::add_outlier(), *20*
rjd3toolkit::add_ramp(), *20*
rjd3toolkit::add_usrdefvar(), *20*
rjd3toolkit::dictionary(), *15*
rjd3toolkit::remove_outlier(), *20*
rjd3toolkit::remove_ramp(), *20*
rjd3toolkit::result(), *15*
rjd3toolkit::set_arima(), *20*
rjd3toolkit::set_automodel(), *20*
rjd3toolkit::set_basic(), *20*
rjd3toolkit::set_benchmarking(), *20*
rjd3toolkit::set_easter(), *20*
rjd3toolkit::set_estimate(), *20*
rjd3toolkit::set_outlier(), *20*
rjd3toolkit::set_tradingdays(), *20*
rjd3toolkit::set_transform(), *20*

set_x11, 11
set_x11(), *20*
spec_regarima *(deprecated-rjd3x13)*, 2
spec_x11 *(deprecated-rjd3x13)*, 2
spec_x13 *(deprecated-rjd3x13)*, 2

userdefined_variables_x13
(deprecated-rjd3x13), 2

x11, 14
x11_spec *(x13_spec)*, 19
x11_spec(), *13*
x13, 14
x13_dictionary, 16
x13_dictionary(), *3*, *9*, *14, 15*, *18*
x13_fast *(x13)*, 14
x13_full_dictionary *(x13_dictionary)*, 16
x13_option, 17
x13_refresh *(refresh)*, 5
x13_revisions, 18
x13_spec, 19
x13_spec(), *13*