

Package: rjd3x13 (via r-universe)

September 18, 2024

Type Package

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

Version 3.2.3.9000

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
RegARIMA modelling (automatic ARIMA model with outlier
detection and trading days adjustment) and X-11 decomposition.

Depends R (>= 4.1.0)

Imports rJava (>= 1.0-6), rjd3toolkit (>= 3.2.2), RProtoBuf (>= 0.4.17)

Remotes github::rjdverse/rjd3toolkit

SystemRequirements Java (>= 17)

License EUPL

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

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

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

Encoding UTF-8

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

Repository <https://tanguybarthelemy.r-universe.dev>

RemoteUrl <https://github.com/rjdverse/rjd3x13>

RemoteRef HEAD

RemoteSha e0a52002f47d0f48bb07dd49b8a3dbebdb0b0542

Contents

deprecated-rjd3x13	2
jd3_utilities	3
refresh	3
regarima	6
regarima_outliers	7
set_x11	8
userdefined_variables_x13	10
x11	11
x13	12
x13_dictionary	13
x13_full_dictionary	13
x13_revisions	14
x13_spec	15
Index	17

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
)
```

Arguments

ts, spec, context, userdefined, name
Parameters.

jd3_utilities	<i>Java Utility Functions</i>
---------------	-------------------------------

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.

refresh	<i>Refresh a specification with constraints</i>
---------	---

Description

Function allowing to create a new specification by updating a specification used for a previous estimation. Some selected parameters will be kept fixed (previous estimation results) while others will be freed for re-estimation in a domain of constraints. See details and examples.

Usage

```

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

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

```

Arguments

spec	the current specification to be refreshed ("result_spec").
refspect	the reference specification used to define the domain considered for re-estimation ("domain_spec"). By default this is the "RG5c" or "RSA5" specification.
policy	the refresh policy to apply (see details).
period, start, end	additional parameters used to specify the span on which additive outliers (AO) are introduced when policy = "Current" or to specify the span on which outliers will be re-detected when policy = "Outliers" or policy = "Outliers_StochasticComponent", in this case end is unused. If start is not specified, outliers will be re-identified on the whole series. Span definition: period: numeric, number of observations in a year (12, 4...). start and end: defined as arrays of two elements: year and first period (for example, period = 12 and c(1980, 1) stands for January 1980) The dates corresponding start and end are included in the span definition.

Details

The selection of constraints to be kept fixed or re-estimated is called a revision policy. User-defined parameters are always copied to the new refreshed specifications. In X-13 only the reg-arma part can be refreshed. X-11 decomposition will be completely re-run, keeping all the user-defined parameters from the original specification.

Available refresh policies are:

Current: applying the current pre-adjustment reg-arma model and handling the new raw data points, or any sub-span of the series as Additive Outliers (defined as new intervention variables)

Fixed: applying the current pre-adjustment reg-arima model and replacing forecasts by new raw data points.

FixedParameters: pre-adjustment reg-arima model is partially modified: regression coefficients will be re-estimated but regression variables, Arima orders and coefficients are unchanged.

FixedAutoRegressiveParameters: same as FixedParameters but Arima Moving Average coefficients (MA) are also re-estimated, Auto-regressive (AR) coefficients are kept fixed.

FreeParameters: all regression and Arima model coefficients are re-estimated, regression variables and Arima orders are kept fixed.

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

Outliers_StochasticComponent: same as "Outliers" but Arima model orders (p,d,q)(P,D,Q) can also be re-identified.

Value

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

References

More information on revision policies in JDemetra+ online documentation: <https://jdemetra-new-documentation.netlify.app/t-rev-policies-production>

Examples

```
y<- rjd3toolkit::ABS$X0.2.08.10.M
# raw series for first estimation
y_raw <-window(y,end = c(2016,12))
# raw series for second (refreshed) estimation
y_new <-window(y,end = c(2017,6))
# specification for first estimation
spec_x13_1<-x13_spec("rsa5c")
# first estimation
sa_x13<- x13(y_raw, spec_x13_1)
# refreshing the specification
current_result_spec <- sa_x13$result_spec
current_domain_spec <- sa_x13$estimation_spec
# policy = "Fixed"
spec_x13_ref <- x13_refresh(current_result_spec, # point spec to be refreshed
  current_domain_spec, #domain spec (set of constraints)
  policy = "Fixed")
# 2nd estimation with refreshed specification
sa_x13_ref <- x13(y_new, spec_x13_ref)
# policy = "Outliers"
spec_x13_ref <- x13_refresh(current_result_spec,
  current_domain_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(current_result_spec,
  current_domain_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

```

y = rjd3toolkit::ABS$X0.2.09.10.M
sp = regarima_spec("rg5c")
sp = rjd3toolkit::add_outlier(sp,
                             type = c("AO"), c("2015-01-01", "2010-01-01"))
regarima_fast(y, spec = sp)
sp = rjd3toolkit::set_transform(
  rjd3toolkit::set_tradingdays(
    rjd3toolkit::set_easter(sp, enabled = FALSE),
    option = "workingdays"
  ),
  fun = "None"
)
regarima_fast(y, spec = sp)
sp = rjd3toolkit::set_outlier(sp, outliers.type = c("AO"))
regarima_fast(y, spec = sp)

```

regarima_outliers	<i>Outlier Detection with a RegARIMA Model</i>
-------------------	--

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

```
regarima_outliers(rjd3toolkit::ABS$X0.2.09.10.M)
```

 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")
)
```


Arguments

<code>x</code>	the specification to be modified, object of class "JD3_X11_SPEC", default X11 spec can be obtained as <code>'x=x11_spec()'</code>
<code>mode</code>	character: the decomposition mode. Determines the mode of the seasonal adjustment decomposition to be performed: "Undefined" - no assumption concerning the relationship between the time series components is made; "Additive" - assumes an additive relationship; "Multiplicative" - assumes a multiplicative relationship; "LogAdditive" - performs an additive decomposition of the logarithms of the series being adjusted; "PseudoAdditive" - assumes an pseudo-additive relationship. Could be changed by the program, if needed.
<code>seasonal.comp</code>	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).
<code>seasonal.filter</code>	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.: <code>seasonal.filter = "Msr"</code> or <code>seasonal.filter = "S3X3"</code>); or have a different value for each quarter (length 4) or each month (length 12) - (e.g. for quarterly series: <code>seasonal.filter = c("S3X3", "Msr", "S3X3", "Msr")</code>). Possible filters are: "Msr", "Stable", "X11Default", "S3X1", "S3X3", "S3X5", "S3X9", "S3X15". "Msr" - the program chooses the final seasonal filter automatically.
<code>henderson.filter</code>	numeric: the length of the Henderson filter (odd number between 3 and 101). If <code>henderson.filter = 0</code> an automatic selection of the Henderson filter's length for the trend estimation is enabled.
<code>lsigma</code>	numeric: the lower sigma boundary for the detection of extreme values, > 0.5, default=1.5.
<code>usigma</code>	numeric: the upper sigma boundary for the detection of extreme values, > lsigma, default=2.5.
<code>bcasts, fcasts</code>	numeric: the number of backcasts (bcasts) or forecasts (fcasts) generated by the RegARIMA model in periods (positive values) or years (negative values). Default values: <code>fcasts=-1</code> and <code>bcasts=0</code> .
<code>calendar.sigma</code>	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 <code>x11.sigmaVector</code> parameter ("Select", see parameter <code>sigma.vector</code>).
<code>sigma.vector</code>	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. Only used if <code>calendar.sigma = "Select"</code> . 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` TODO.

Value

a "JD3_X11_SPEC" object, containing all the parameters.

See Also

[x13_spec\(\)](#) and [x11_spec\(\)](#).

Examples

```
init_spec <- x11_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 = "LEGACY")
```

userdefined_variables_x13

Display a list of all the available output objects

Description

Function generating a comprehensive list of available output variables (series, parameters, diagnostics) from the estimation process by the `x13()`, `regarima()` and `x11()` functions. Some items are available in the default estimation output but the remainder can be added using the userdefined parameter. User-defined objects can be retrieved from the list of lists generated by the estimation process

Usage

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

Arguments

`x` a character to indicate the estimation function for which the output items list will be displayed.

Value

a vector containing the names of all the available output objects (series, diagnostics, parameters)

References

More information and examples related to 'JDemetra+' features in the online documentation: <https://jdemetra-new-documentation.netlify.app/>

Examples

```
userdefined_variables_x13("x13")
userdefined_variables_x13("regarima")
userdefined_variables_x13("x11")
```

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\(\)](#)).

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)
```

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

```

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 = rjd3toolkit::add_outlier(sp,
                             type = c("A0"), c("2015-01-01", "2010-01-01"))
sp = rjd3toolkit::set_transform(
  rjd3toolkit::set_tradingdays(
    rjd3toolkit::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)

```

x13_dictionary	<i>X-13 Dictionary</i>
----------------	------------------------

Description

X-13 Dictionary

Usage

```
x13_dictionary()
```

Value

A vector containing the names of all the available output objects (series, diagnostics, parameters).

x13_full_dictionary	<i>Title</i>
---------------------	--------------

Description

Title

Usage

```
x13_full_dictionary()
```

x13_revisions	<i>Revisions History</i>
---------------	--------------------------

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: <code>start</code> (first date to compute the history, in the format "YYYY-MM-DD") and <code>id</code> (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: <code>start</code> (first date to compute the history, in the format "YYYY-MM-DD"), <code>period</code> (the date of the studied) and <code>id</code> (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: <code>start</code> (first date to compute the history, in the format "YYYY-MM-DD"), <code>end</code> (last date to compute the history, in the format "YYYY-MM-DD") and <code>id</code> (the name of the component, see x13_dictionary()). As many series as periods between <code>start</code> and <code>end</code> will be exported. See example.
context	The context of the specification.

Examples

```
s <- rjd3toolkit::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)

```

x13_spec

*RegARIMA/X-13 Default Specifications***Description**

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

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 the name of a predefined specification.

Details

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

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

When setting a complete X13-Arima spec, x13_spec() without argument yields a RSA5c specification

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


Index

.jd2r_spec_regarima (jd3_utilities), 3
.jd2r_spec_x11 (jd3_utilities), 3
.jd2r_spec_x13 (jd3_utilities), 3
.jx13 (x13), 12
.r2jd_spec_regarima (jd3_utilities), 3
.r2jd_spec_x11 (jd3_utilities), 3
.r2jd_spec_x13 (jd3_utilities), 3
.x13_rs1ts (jd3_utilities), 3

deprecated-rjd3x13, 2

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

jd3_utilities, 3

refresh, 3
regarima, 6
regarima_fast (regarima), 6
regarima_outliers, 7
regarima_refresh (refresh), 3
regarima_spec (x13_spec), 15
rjd3toolkit::add_outlier(), 16
rjd3toolkit::add_ramp(), 16
rjd3toolkit::add_usrdefvar(), 16
rjd3toolkit::dictionary(), 12
rjd3toolkit::remove_outlier(), 16
rjd3toolkit::remove_ramp(), 16
rjd3toolkit::result(), 12
rjd3toolkit::set_arima(), 16
rjd3toolkit::set_automodel(), 16
rjd3toolkit::set_basic(), 16
rjd3toolkit::set_benchmarking(), 16
rjd3toolkit::set_easter(), 16
rjd3toolkit::set_estimate(), 16
rjd3toolkit::set_outlier(), 16
rjd3toolkit::set_tradingdays(), 16
rjd3toolkit::set_transform(), 16

set_x11, 8
set_x11(), 16

spec_regarima (deprecated-rjd3x13), 2
spec_x11 (deprecated-rjd3x13), 2
spec_x13 (deprecated-rjd3x13), 2

userdefined_variables_x13, 10

x11, 11
x11_spec (x13_spec), 15
x11_spec(), 10
x13, 12
x13_dictionary, 13
x13_dictionary(), 6, 11, 12, 14
x13_fast (x13), 12
x13_full_dictionary, 13
x13_refresh (refresh), 3
x13_revisions, 14
x13_spec, 15
x13_spec(), 10