Package 'rjd3production'

May 9, 2026

Title Prepare for Production of Seasonal Adjustment with 'JDemetra+'

Version 1.1.0

Description A comprehensive tool for setting up seasonal data pipelines using 'JDemetra+' (version 3) and 'rjdverse'. This includes setting up a new working environment, creating and selecting calendar regressors, managing specifications (trading-days regressors and outliers) at the workspace level, making a workspace usable by the 'cruncher', removing insignificant outliers, and comparing workspaces.

Depends R (>= 4.2.0)

Imports rjd3toolkit, rjd3x13, rjd3workspace, rjd3providers, utils, dygraphs, flextable, stats, tidyr, tools, yaml, zoo, shiny, constructive, usethis, TBox, date4ts, lintr

Suggests testthat (>= 3.0.0), knitr, quarto, rmarkdown, waldo

License EUPL

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr, quarto

Config/testthat/edition 3

URL https://inseefr.github.io/rjd3production/

Config/Needs/build moodymudskipper/devtag

NeedsCompilation no

Author Tanguy Barthelemy [aut, cre, art]

Maintainer Tanguy Barthelemy <tanguy.barthelemy@insee.fr>

Repository CRAN

Date/Publication 2026-04-21 09:02:16 UTC

Contents

add_raw_data_path 2
assign_outliers 3

compare . . . 6
create_specs_set . . . 7
create_ws_from_data . . . 8
get_jsai_by_name . . . 9
get_named_variables . . . 9
get_series . . . 10
init_env . . . 11
insee_modelling . . . 12
make_ws_crunchable . . . 14
random-spec . . . 15
remove_non_significative_outliers . . . 15
run_app . . . 17
select_td . . . 18
set_minimum_span . . . 19
translate-spec . . . 20

Index **22**

`add_raw_data_path` *Add raw data from a file to a JWS workspace*

Description

This function completes the ts metadata (moniker) to make the workspace refreshable and crunchable.

Usage

```
add_raw_data_path(jws, path, ...)
```

Arguments

`jws`	A Java Workspace object, as returned by `rjd3workspace::jws_open()` or `rjd3workspace::jws_new()`
`path`	A character string. Path to the input data file. Must be a `.csv` file (support for `.xlsx` is not yet implemented).
`...`	Addional arguments passed to `rjd3providers::txt_data()` (e.g., delimiter, date format, clean missing argument...).

Details

Currently, only CSV files are supported. Each column of the input file is interpreted as a time series and matched against the series names in the workspace.

The difference with the function `make_ws_crunchable` is that `add_raw_data_path()` will associate the workspace with a non temporary data path.

Value

The modified `jws` object invisibly.

Examples

```
library("rjd3workspace")
library("rjd3x13")
library("rjd3toolkit")

my_data <- ABS
path_ABS <- system.file("extdata", "ABS.csv", package = "rjd3providers")

jws <- create_ws_from_data(my_data)
add_raw_data_path(jws, path_ABS, delimiter = "COMMA")
```

---

assign_outliers *Manage regression components in JDemetra+ workspaces*

Description

These functions allow extracting, exporting, importing and assigning regression components used in JDemetra+ workspaces.

Usage

```
assign_outliers(jws, outliers, verbose = TRUE)

assign_td(jws, td, verbose = TRUE)

export_outliers(outliers, path = NULL, verbose = TRUE)

import_outliers(path, verbose = TRUE)

export_td(td, path = NULL, verbose = TRUE)

import_td(path, verbose = TRUE)

retrieve_outliers(
  jws,
  domain = TRUE,
  estimation = FALSE,
  point = FALSE,
  verbose = TRUE
)

retrieve_td(
  jws,
  domain = TRUE,
```

```
  estimation = FALSE,
  point = FALSE,
  verbose = TRUE
)
```

Arguments

jws	A Java Workspace object, as returned by rjd3workspace::jws_open() or rjd3workspace::jws_new()
outliers	[data.frame] A data.frame created with retrieve_outliers or import_outliers. See Format section for more information about the format of this argument.
verbose	Boolean. Print additional informations. Default is TRUE.
td	[data.frame] A data.frame created by retrieve_td or import_td. See Format section for more information about the format of this argument.
path	character Path to a YAML file to read or write a table.
domain	Boolean indicating if outliers should be extracted from the domain specification.
estimation	Boolean indicating if outliers should be extracted from the estimation specification.
point	Boolean indicating if outliers should be extracted from the point specification.

Details

Two types of regression components are currently supported:

- **Outliers**
- **Trading-day regressors (TD)**

Value

- retrieve_outliers() and import_outliers() returns data.frame representing the outliers.
- retrieve_td() and import_td() returns data.frame representing the trading days variables
- export_outliers() and export_td() functions invisibly return the path of the YAML file written.
- assign_XXX() functions invisibly return the updated workspace jws.

Format

Outliers table:

Outliers are represented by a data.frame with **three columns**:

- series : name of the series in the workspace.
- type : type of outlier (AO, LS, TC or SO).
- date : date of the outlier in YYYY-MM-DD format.

These tables are typically created with retrieve_outliers() or import_outliers().

Trading-day table:

Trading-day specifications are represented by a data.frame with **two columns**:

- `series` : name of the series in the workspace.
- `regs` : name of the trading-day regressor set to apply (e.g. `REG1`, `REG2`, ..., optionally with `LY`).

These tables are typically created with `retrieve_td()` or `import_td()`.

Workflow

The workflow typically follows these steps:

1. Extract regression information from a workspace (`retrieve_XXX()`)
2. Optionally export it to a YAML file (`export_XXX()`)
3. Import it later from the YAML file (`import_XXX()`)
4. Assign the regression specification to another workspace (`assign_XXX()`)

Other

The assignment functions (`assign_XXX()`) modify the **first SA-Processing** of the workspace.

Currently, regression information can be extracted (`retrieve_XXX()`) from the point, estimation or domainSpec, while the assignment step (`assign_XXX()`) is performed in both the domainSpec and the estimationSpec.

Examples

```
library("rjd3workspace")
library("rjd3toolkit")

my_data <- ABS[, 1:3]
jws <- create_ws_from_data(my_data)
set_context(jws, create_insee_context(start = c(2015L, 1L)))

## Outliers

# Read all the outliers from a workspace
outs <- retrieve_outliers(jws, point = TRUE, domain = FALSE)

# Export outliers
path_outs <- tempfile(pattern = "outliers-table", fileext = ".yaml")
export_outliers(outs, path_outs)

# Import outliers from a file
outs2 <- import_outliers(path_outs)

# Assign the outliers to a WS
assign_outliers(jws = jws, outliers = outs2)


## Trading day workflow

# Read all the td variables from a workspace
```

```
td <- retrieve_td(jws)

# Export td variables
path_td <- tempfile(pattern = "td-table", fileext = ".yaml")
export_td(td, path_td)

# Import td variable from a file
td2 <- import_td(path_td)

# Select td
td3 <- select_td(my_data)

# Assign the td variables to a WS
assign_td(jws = jws, td = td3)
```

`compare` *Compare series across workspaces*

Description

Reads multiple JDemetra+ workspaces and extracts comparable series (by SAI and series type), returning them in a tidy format. This is particularly useful to compare results across different specifications (e.g. RSA3 vs RSA5).

Usage

```
compare(..., series_names)
```

Arguments

`...`	character Workspace file paths.
`series_names`	character Vector of SAI names to compare.

Value

A `data.frame` with columns:

- `ws`: workspace name (derived from file basename),
- `SAI`: SAI name,
- `series`: type of series,
- `date`: observation date,
- `value`: numeric value.

Examples

```
library("rjd3toolkit")
library("rjd3x13")
library("rjd3workspace")


# Two demo workspaces (RSA3 and RSA5)
jws_rsa3 <- create_ws_from_data(ABS, x13_spec("rsa3"))
jws_rsa5 <- create_ws_from_data(ABS, x13_spec("rsa5"))

path_rsa3 <- tempfile(pattern = "ws-rsa3", fileext = ".xml")
path_rsa5 <- tempfile(pattern = "ws-rsa5", fileext = ".xml")

save_workspace(jws_rsa3, file = path_rsa3)
save_workspace(jws_rsa5, file = path_rsa5)

df <- compare(path_rsa3, path_rsa5, series_names = "X0.2.09.10.M")
head(df)
```

`create_specs_set` *Creating a set of X13 specifications*

Description

Builds a set of X13 specifications from a start date, a modelling context (explanatory variables) and outliers (optional).

Usage

```
create_specs_set(
  spec_0 = NULL,
  context = NULL,
  outliers = NULL,
  span_start = NULL
)
```

Arguments

`spec_0`	Basic specification
`context`	list Modeling context created by `rjd3toolkit::modelling_context()`.
`outliers`	[list or NULL] Optional list with elements : • `type`: vector of outlier types (e.g. "AO", "LS", "TC") • `date`: vector of corresponding dates
`span_start`	character Estimation start date (format "YYYY-MM-DD").

Value

A list of named X13 specifications (TD and variants).

Examples

```
my_context <- create_insee_context()
create_specs_set(context = my_context)
```

---

create_ws_from_data *Create a Workspace from Data*

Description

Creates a new JDemetra+ workspace with all columns of a time series object using a specified specification.

Each column of `x` is interpreted as a separate time series and added to a newly created Seasonal Adjustment Processing (SAP).

Usage

```
create_ws_from_data(x, spec = rjd3x13::x13_spec())
```

Arguments

`x`	A time series object (e.g. `ts`, `mts`, or matrix coercible to `ts`) where each column represents a series to be seasonally adjusted. Column names are used as SA-Item names.
`spec`	A JDemetra+ specification. Defaults to `rjd3x13::x13_spec()`.

Details

All series share the same specification (`spec`).

Value

A JDemetra+ workspace object (Java pointer) containing one SA-Processing with one SA-Item per column of `x`.

Examples

```
library("rjd3toolkit")

# Create workspace
ws <- create_ws_from_data(ABS)
```

get_jsai_by_name *Retrieve a SA-Item by its name*

Description

Searches a workspace for a seasonal adjustment item (SAI) whose name matches the user-supplied string and returns the corresponding object.

Usage

```
get_jsai_by_name(jws, series_name)
```

Arguments

jws	A Java Workspace object, as returned by rjd3workspace::jws_open() or rjd3workspace::jws_new()
series_name	character Name of the SAI to retrieve.

Value

A Java Seasonal Adjustment Item object (jsai).

Examples

```
library("rjd3toolkit")
library("rjd3workspace")

# Demo workspace
jws <- create_ws_from_data(ABS)
jws_compute(jws)
jsap <- jws_sap(jws, 1L)

jsai <- get_jsai_by_name(jws, "X0.2.09.10.M")
df <- get_series(jsai)
head(df)
```

get_named_variables *Retrieve all the auxiliary variables from a workspace*

Description

Lists all the variables in a modelling context.

Usage

```
get_named_variables(context = NULL)
```

Arguments

context	a modelling context

Value

a list with all the groups and named variables

Examples

```
context_FR <- create_insee_context()
get_named_variables(context_FR)
```

---

get_series *Extract all series from a SA-Item*

Description

Extracts all available time series (pre-adjustment, decomposition, and final) from a seasonal adjustment item (jsai) inside a JDemetra+ workspace.

Usage

```
get_series(x, ...)

## S3 method for class 'JD3_TRAMOSEATS_RSLTS'
get_series(x, name, ...)

## S3 method for class 'JD3_X13_RSLTS'
get_series(x, name, ...)

## S3 method for class 'jobjRef'
get_series(x, ...)
```

Arguments

x	The object to extract the series
...	Additional argument
name	Name of the SA object

Details

x can be a Java SAI object, typically obtained via rjd3workspace::jsap_sai() after opening and computing a workspace with rjd3workspace::jws_open() and rjd3workspace::jws_compute().

Value

A `data.frame` with columns:

- `SAI`: name of the SAI,
- `series`: the type of series (e.g. `"y"`, `"sa"`, `"trend"`),
- `date`: observation dates,
- `value`: numeric values of the series.

Examples

```
library("rjd3toolkit")
library("rjd3workspace")


# Demo workspace
jws <- create_ws_from_data(ABS)
jws_compute(jws)
jsap <- jws_sap(jws, 1L)
jsai <- jsap_sai(jsap, 1L)

df <- get_series(jsai)
head(df)
```

`init_env` *Initialize a seasonal adjustment project environment*

Description

This function creates a complete project structure for a seasonal adjustment production workflow. It initializes an R project, sets up useful directories, configuration files, and development tools.

The generated structure is designed for workflows based on the 'rjdverse'.

Usage

```
init_env(path, open = FALSE)
```

Arguments

`path`	A character string. Path where the project will be created.
`open`	Boolean. Should the project be opened in RStudio after creation? Default is `FALSE`.

Value

The project path invisibly.

Examples

```
project_path <- tempfile(pattern = "my-project")

## Not run:
# Create a new project
init_env(path = project_path)

## End(Not run)
```

`insee_modelling` *French modelling context, calendar and trading days regressors.*

Description

These functions allow to construct the standard regressors and modelling context used by INSEE for seasonal adjustment:

- `create_french_calendar()` creates the French national calendar.
- `create_insee_regressors()` generates trading day regressors and leap-year effect (LY).
- `create_insee_regressors_sets()` organizes these regressors into standard sets (REG1, REG2, . . . , REG6, with or without LY).
- `create_insee_context()` combines the regressors and calendar into a `modelling_context` object that can be used directly with `rjd3toolkit`.

Usage

```
create_french_calendar()

create_insee_regressors(
  start = c(1990L, 1L),
  frequency = 12L,
  length = 492L,
  s = NULL,
  cal = NULL
)

create_insee_regressors_sets(
  start = c(1990L, 1L),
  frequency = 12L,
  length = 492L,
  s = NULL,
  cal = NULL
)

create_insee_context(
```

```
  start = c(1990L, 1L),
  frequency = 12L,
  length = 492L,
  s = NULL
)
```

Arguments

start	[integer vector] Start period in the format c(year, month) (default c(1990, 1)).
frequency	integer Series frequency (default 12L).
length	integer Series length (default 492L).
s	[numeric or NULL] Optional argument for adjustment (passed to rjd3toolkit).
cal	a calendar of class JD3_CALENDAR.

Value

- create_french_calendar() returns a national_calendar object.
- create_insee_regressors() returns a matrix of regressors (working days
- LY).
- create_insee_regressors_sets() returns a list of regressor sets (REG1, REG2, ..., REG6, with or without LY).
- create_insee_context() returns a modelling_context object.

Examples

```
# 1. Create the French calendar
cal <- create_french_calendar()
cal

# 2. Generate regressors
regs <- create_insee_regressors(start = c(2000, 1), frequency = 12, length = 240)
head(regs)

# 3. Organize into standard sets
sets <- create_insee_regressors_sets(start = c(2000, 1), frequency = 12, length = 240)
names(sets)

# 4. Build a complete context for rjd3toolkit
context <- create_insee_context(start = c(2000, 1), frequency = 12, length = 240)
context
```

`make_ws_crunchable` *Make a workspace crunchable*

Description

Complete and replace the ts metadata of a WS to make it crunchable

Usage

```
make_ws_crunchable(jws, verbose = TRUE)
```

Arguments

`jws`	A Java Workspace object, as returned by `rjd3workspace::jws_open()` or `rjd3workspace::jws_new()`
`verbose`	Boolean. Print additional informations. Default is `TRUE`.

Details

New metadata are added from temporary files created on the heap. Thus, this operation is not intended to make the workspace crunchable in a stable way over time, but rather for a short period of time for testing purposes, in particular when we are sent a workspace without the raw data.

Value

A java workspace (as jws) but with new ts metadata

Examples

```
library("rjd3workspace")
library("rjd3x13")
library("rjd3toolkit")

jws <- jws_new()
jsap <- jws_sap_new(jws, "sap1")
add_sa_item(
    jsap = jsap,
    name = "series_3",
    x = ABS[, 1],
    spec = x13_spec("RSA3")
)
jws <- make_ws_crunchable(jws)
```

random-spec *Random JDemetra+ Specifications Generator*

Description

`random_spec()` allows you to create a random specification based on a set of helper functions (auxiliary functions). These specifications are created from scratch.

Usage

```
random_spec()
```

Details

The objective is to enable:

- examples
- tests of other functions (notably for reverse engineering)
- other tests and demonstrations

Value

a JD+ Specification

Examples

```
set.seed(1L)
spec <- random_spec()
```

remove_non_significative_outliers *Remove non-significant outliers from a JDemetra+ workspace*

Description

This function scans a JDemetra+ workspace (`.xml`) and removes regression outliers whose p-values are above a given threshold. Both the estimation specification and the domain specification are updated accordingly, and the workspace file is saved in place.

Typical use case: after estimation with user pre-specified outliers, outliers with weak statistical significance (e.g. `p > 0.3`) are dropped to simplify the regression specification.

Usage

```
remove_non_significative_outliers(
  ws_path,
  threshold = 0.3,
  domain = FALSE,
  estimation = FALSE,
  verbose = TRUE
)
```

Arguments

ws_path	[character] Path to a JDemetra+ workspace file (usually with extension .xml).
threshold	[numeric] Maximum p-value for keeping an outlier. Outliers with Pr(>|t|) > threshold are removed. Default is 0.3.
domain	Boolean indicating if the domain specification should be modified.
estimation	Boolean indicating if the estimation specification should be modified.
verbose	Boolean. Print additional informations. Default is TRUE.

Details

The function:

- iterates over all the series (SA-Items) in the workspace,
- identifies outliers in the regarima specification,
- checks their p-values in the pre-processing regression summary,
- removes those with p-values above the threshold from both estimationSpec and, if present, domainSpec,
- saves the workspace file.

Value

The function invisibly returns NULL, but it **modifies the workspace file in place** (saved at the same location as ws_path).

Examples

```
library("rjd3workspace")
library("rjd3x13")
library("rjd3toolkit")


new_spec <- x13_spec() |>
    add_outlier(type = "LS", date = "1990-01-01")
jws <- create_ws_from_data(x = ABS[, 1, drop = FALSE], spec = new_spec)
path_ws <- tempfile(pattern = "ws", fileext = ".xml")
save_workspace(jws, file = path_ws)
```

```
# Remove non-significant outliers (p > 0.3) from a workspace
remove_non_significative_outliers(path_ws, threshold = 0.3, domain = TRUE)
```

---

run_app *Run the Shiny comparison app*

Description

Launches an interactive Shiny application to explore and compare seasonal adjustment results stored in a `data.frame` returned by `compare()`.

Usage

```
run_app(data, ...)
```

Arguments

`data`	A `data.frame` returned by `compare()`, containing the columns `ws`, `SAI`, `series`, `date`, and `value`.
`...`	Additional arguments passed to `shiny::shinyApp()`.

Value

Runs a Shiny app in the R session (no return value).

Examples

```
library("rjd3toolkit")
library("rjd3x13")
library("rjd3workspace")


# Two demo workspaces (RSA3 and RSA5)
jws_rsa3 <- create_ws_from_data(ABS, x13_spec("rsa3"))
jws_rsa5 <- create_ws_from_data(ABS, x13_spec("rsa5"))

path_rsa3 <- tempfile(pattern = "ws-rsa3", fileext = ".xml")
path_rsa5 <- tempfile(pattern = "ws-rsa5", fileext = ".xml")

save_workspace(jws_rsa3, file = path_rsa3)
save_workspace(jws_rsa5, file = path_rsa5)

# Compare the two workspace
df <- compare(path_rsa3, path_rsa5, series_names = "X0.2.09.10.M")
head(df)
```

```
# Launch the shiny app
if (interactive()) {
    run_app(df)
}
```

`select_td` *Select Calendar Regressors for One or Multiple Series*

Description

Applies the X13 regression selection procedure to one or more time series. If multiple series are provided as columns of a matrix or data.frame, each series is processed separately. The function returns the selected set of regressors for each series.

Usage

```
select_td(series, context = NULL, ..., verbose = TRUE)
```

Arguments

`series`	[ts or mts or matrix or data.frame] A univariate time series (`ts`) or a multivariate series (columns as separate series).
`context`	list Modeling context created by `rjd3toolkit::modelling_context()`.
`...`	Additional arguments passed to `create_specs_set()` controlling the generation of X13 specifications. Possible arguments include: **outliers** Optional list of outliers with elements `type` (vector of types, e.g., "AO", "LS", "TC") and `date` (vector of dates). **span_start** Starting date of the estimation (character, format `"YYYY-MM-DD"`). **...** Other arguments accepted by `create_specs_set()`.
`verbose`	Boolean. Print additional informations. Default is `TRUE`.

Value

A data.frame with two columns:

series Name of the series (column name if `series` is multivariate).

regs Name of the selected regressor set.

Examples

```
library("rjd3toolkit")


# Single series
select_td(ABS[, 1])

# Multiple series
select_td(ABS)

# Restrict regressors sets
my_context <- create_insee_context(s = ABS)
my_context$variables <- my_context$variables[c("REG1", "REG1_LY", "REG6", "REG6_LY")]
select_td(ABS, context = my_context)
```

`set_minimum_span` *Set span minimum to a value*

Description

Set span minimum to a value

Usage

```
set_minimum_span(
  spec,
  d0,
  model_span = TRUE,
  series_span = TRUE,
  without_outliers = TRUE
)
```

Arguments

`spec`	Specification (object of class `JD3_X13_SPEC` or `JD3_TRAMOSEATS_SPEC`
`d0`	characters in the format "YYYY-MM-DD" to specify first date of the span
`model_span`	Boolean. Should the estimation (= model) span be modifed?
`series_span`	Boolean. Should the series (= basic) span be modifed?
`without_outliers`	Boolean. Should the outliers set before the starting date be removed? (Small crutch while waiting for the resolution of jdemetra/jdplus-main issue 858.)

Details

model_span = estimation_span series_span = basic_span

Value

the modify specification (an JD3_X13_SPEC or JD3_TRAMOSEATS_SPEC object).

Examples

```
library("rjd3toolkit")
library("rjd3x13")
library("rjd3workspace")


# Two demo workspaces (RSA3 and RSA5)
spec <- x13_spec("rsa3")
set_minimum_span(spec, "2012-01-01")
```

---

translate-spec *Reverse Engineering of rjd3 Specifications*

Description

This family of functions reconstructs executable R code from a X13 specification object. the generated code uses only the packages {rjd3toolkit} and {rjd3x13}.

The main entry point is `rev_spec()`, which aggregates all reverse-generating helpers.

Usage

```
rev_spec(x)
```

Arguments

x	A JDemetra+ specification object

Details

The functions are taking a specification (argument `x`) as input and returns A corresponding code that generates the object `x`.

`rev_spec()` is the main function and calls all other helper functions (`rev_XXX`). These helper functions (auxiliary functions) do NOT provide sufficient code to reproduce the specification, but only the part dedicated to them (outliers, trading days regressors, x11 filters, etc.).

The generated code is neither unique nor optimal.

That is, different codes (other than the one generated by rev_spec) can generate the same specification. It is not optimal because it does not use the default values of the functions but clearly redefines all the parameters.

Value

Each `rev_XXX()` function returns a character string containing executable R code. `rev_spec()` returns a complete multi-line pipeline.

Examples

```
spec_init <- rjd3x13::x13_spec("RSA3") |>
    rjd3toolkit::set_basic(type = "All") |>
    rjd3toolkit::set_automodel(enabled = FALSE)
code <- rev_spec(spec_init)
cat(code)
spec_rebuilt <- eval(parse(text = code))
```

Index

∗ **regression tools**
    assign_outliers, 3

add_raw_data_path, 2
assign_outliers, 3
assign_td *(*assign_outliers*)*, 3

character, *4*, *6*, *7*, *9*, *16*
compare, 6
compare(), *17*
create_french_calendar
        *(*insee_modelling*)*, 12
create_french_calendar(), *12*
create_insee_context *(*insee_modelling*)*,
        12
create_insee_context(), *12*
create_insee_regressors
        *(*insee_modelling*)*, 12
create_insee_regressors(), *12*
create_insee_regressors_sets
        *(*insee_modelling*)*, 12
create_insee_regressors_sets(), *12*
create_specs_set, 7
create_specs_set(), *18*
create_ws_from_data, 8

data.frame, *4*, *18*

export_outliers *(*assign_outliers*)*, 3
export_td *(*assign_outliers*)*, 3

get_jsai_by_name, 9
get_named_variables, 9
get_series, 10

import_outliers, *4*
import_outliers *(*assign_outliers*)*, 3
import_outliers(), *4*
import_td, *4*
import_td *(*assign_outliers*)*, 3
import_td(), *5*

init_env, 11
insee_modelling, 12
integer, *13*

list, *7*, *18*

make_ws_crunchable, *2*, 14

numeric, *13*, *16*

random-spec, 15
random_spec *(*random-spec*)*, 15
regression_tools *(*assign_outliers*)*, 3
remove_non_significative_outliers, 15
retrieve_outliers, *4*
retrieve_outliers *(*assign_outliers*)*, 3
retrieve_outliers(), *4*
retrieve_td, *4*
retrieve_td *(*assign_outliers*)*, 3
retrieve_td(), *5*
rev_spec *(*translate-spec*)*, 20
rjd3toolkit::modelling_context(), *7*, *18*
rjd3workspace::jsap_sai(), *10*
rjd3workspace::jws_compute(), *10*
rjd3workspace::jws_new(), *2*, *4*, *9*, *14*
rjd3workspace::jws_open(), *2*, *4*, *9*, *10*, *14*
run_app, 17

select_td, 18
set_minimum_span, 19
shiny::shinyApp(), *17*

translate-spec, 20
ts, *18*