# Package 'rjd3tramoseats'

September 22, 2026

**Type** Package

**Title** Seasonal Adjustment with TRAMO-SEATS in 'JDemetra+' 3.x

**Version** 3.9.0

**Description** Interface to 'JDemetra+' 3.x (<https://github.com/jdemetra>) time series analysis software. It offers full access to options and outputs of 'TRAMO-SEATS' (Time series Regression with ARIMA noise, Missing values and Outliers - Signal Extraction in ARIMA Time Series), including 'TRAMO' modelling (ARIMA model with outlier detection and trading days adjustment). ARIMA = AutoRegressive Integrated Moving Average.

**License** EUPL

**URL** https://github.com/rjdverse/rjd3tramoseats, https://rjdverse.github.io/rjd3tramoseats/

**BugReports** https://github.com/rjdverse/rjd3tramoseats/issues

**Depends** R (>= 4.1.0)

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

**Suggests** spelling

**Config/roxygen2/version** 8.1.0

**Encoding** UTF-8

**Language** en-GB

**SystemRequirements** Java (>= 21)

**Collate** 'deprecated.R' 'print.R' 'utils.R' 'tramoseats_rslts.R' 'tramoseats_spec.R' 'revisions.R' 'rjd3tramoseats-package.R' 'seats.R' 'seats_spec.R' 'tramo_generic.R' 'tramo_outliers.R' 'tramo_spec.R' 'tramoseats.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],
Anna Smyk [aut],
Eulalie Delaune [aut]

**Maintainer** Tanguy Barthelemy <timeserieswithjdemetraandr@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-09-22 10:00:02 UTC

# Contents

deprecated-rjd3tramoseats . . . . . 2
get_tramoseats_option . . . . . 3
jd3_utilities . . . . . 4
refresh . . . . . 5
seats_decompose . . . . . 9
set_seats . . . . . 10
terror . . . . . 12
tramo . . . . . 13
tramoseats . . . . . 14
tramoseats_dictionary . . . . . 16
tramoseats_option . . . . . 17
tramoseats_revisions . . . . . 18
tramoseats_spec . . . . . 19
tramo_forecast . . . . . 20
tramo_outliers . . . . . 21

**Index** **23**

---

`deprecated-rjd3tramoseats` *Deprecated functions*

---

## Description

Deprecated functions

## Usage

```
fast_tramoseats(
  ts,
  spec = c("rsafull", "rsa0", "rsa1", "rsa2", "rsa3", "rsa4", "rsa5"),
  context = NULL,
  userdefined = NULL
)

fast_tramo(
  ts,
```

```
  spec = c("trfull", "tr0", "tr1", "tr2", "tr3", "tr4", "tr5"),
  context = NULL,
  userdefined = NULL
)

spec_tramoseats(
  name = c("rsafull", "rsa0", "rsa1", "rsa2", "rsa3", "rsa4", "rsa5")
)

spec_tramo(name = c("trfull", "tr0", "tr1", "tr2", "tr3", "tr4", "tr5"))

userdefined_variables_tramoseats(x = c("TRAMO-SEATS", "TRAMO"))
```

## Arguments

| | |
|---|---|
| ts | a univariate time series. |
| spec | the model specification. Can be either the name of a predefined specification or a user-defined specification. |
| context | the dictionary of variables. |
| userdefined | a vector containing the additional output variables (see tramoseats_dictionary()). |
| name | the name of a predefined specification. |
| x | useless parameter |

## Value

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

- spec_tramoseats() returns the same value as tramoseats_spec()
- spec_tramo() returns the same value as tramo_spec()
- fast_tramoseats() returns the same value as tramoseats_fast()
- fast_regarima() returns the same value as regarima_fast()
- .jtramoseats() returns the same value as jtramoseats()
- userdefined_variables_tramoseats() returns the same value as tramoseats_dictionary()

---

# get_tramoseats_option *Set an option for tramoseats*

---

## Description

Set an option for tramoseats

## Usage

```
get_tramoseats_option(name)
```

### Arguments

| | |
|---|---|
| name | Name of the option |

### Value

The requested option or NULL if it doesn't exist

### Examples

```
tramoseats_option("test", "DUMMY")
get_tramoseats_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

```
.tramoseats_rslts(jrslts)

.jd2r_spec_tramo(jspec)

.r2jd_spec_tramo(spec)

.jd2r_spec_tramoseats(jspec)

.r2jd_spec_tramoseats(spec)
```

### Arguments

| | |
|---|---|
| spec, jspec, jrslts | parameters. |

### Value

These functions return specification in Java, proto or R.

refresh *Refresh a specification with constraints*

## Description

Functions tramoseats_refresh and tramo_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

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

tramoseats_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_tramoseats_SPEC" or "JD3_tramo_SPEC", can be obtained as an output of tramoseats_spec or tramo_spec and customised with set_ functions, see tramoseats_spec documentation |
| refspec | reference specification By default "rsa4" or "tr5" specification. Object of class "JD3_tramoseats_SPEC" or "JD3_tramo_SPEC", can be obtained as an output of tramoseats_spec or tramo_spec and customised with set_ functions, see tramoseats_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 and handling the new raw data points, or any sub-span of the series as Additive Outliers (defined as new intervention variables); SEATS and Benchmarking part parameters are untouched.
2. **Fixed**: applying the current pre-adjustment reg-arima model and replacing forecasts by new raw data points; SEATS 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; SEATS 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; SEATS 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; SEATS 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; SEATS and Benchmarking part parameters are untouched.
8. **Complete**: All the parameters are re-identified and re-estimated, unless constrained in the domain spec. SEATS and Benchmarking part parameters are entirely reset to values in the reference spec.

## Value

a new specification, an object of class `"JD3_tramoseats_SPEC"` or `"JD3_tramo_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<- tramoseats_spec("rsa3")
# Customize this spec
## Tramo 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_seats(user_spec,
                     fcasts = -2,
                     bcasts = -1,
                     trend.boundary=0.6,
                     seas.boundary=0.75,
                     algorithm= "KalmanSmoother")
#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"
tramoseats_spec_ref <- tramoseats_refresh(spec= user_spec,
                            refspec= rsa3,
                            policy = "Outliers_StochasticComponent"
```

```
)
# print(tramoseats_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, SEATS and Benchmarking parameters
# are left unchanged
# Use policy: "Complete"
tramoseats_spec_ref <- tramoseats_refresh(spec= user_spec,
                          refspec= rsa3,
                          policy = "Complete"
)
# print(tramoseats_spec_ref)
# all user defined parameters are reset and replaced with "rsa3" parameters,
# Example 2 : practical re-estimation use-case
sa_tramoseats <- tramoseats(y_raw, user_spec)
# refreshing the specification resulting from the first estimation
# to partially adapt to new data
spec_to_refresh <- sa_tramoseats$result_spec
reference_spec <- sa_tramoseats$estimation_spec
# policy = "Fixed"
spec_tramoseats_ref <- tramoseats_refresh(spec_to_refresh,
                           reference_spec,
                           policy = "Fixed"
)
# 2nd estimation with refreshed specification
sa_tramoseats_ref <- tramoseats(y_new, spec_tramoseats_ref)
# policy = "Outliers"
spec_tramoseats_ref <- tramoseats_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_tramoseats_ref <- tramoseats(y_new, spec_tramoseats_ref)

# policy = "Current"

spec_tramoseats_ref <- tramoseats_refresh(spec_to_refresh,
                          reference_spec,
                          policy = "Current",
                          period = 12,
                          start = c(2017, 1),
                          end = end(y_new)
)
sa_tramoseats_ref <- tramoseats(y_new, spec_tramoseats_ref)

# 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
```

seats_decompose *SEATS Decomposition*

## Description

SEATS Decomposition

## Usage

```
seats_decompose(
  sarima,
  seas.tolerance = 2,
  trend.boundary = 0.5,
  seas.boundary = 0.8,
  seas.boundary.unique = 0.8,
  approximation = c("None", "Legacy", "Noisy")
)
```

## Arguments

| | |
|---|---|
| sarima | SARIMA model (see rjd3toolkit::sarima_model()). |
| seas.tolerance | numeric: the seasonal tolerance (epsphi). The tolerance (measured in degrees) to allocate the AR non-real roots to the seasonal component (if the modulus of the inverse complex AR root is greater than the trend boundary and the frequency of this root differs from one of the seasonal frequencies by less than Seasonal tolerance) or the transitory component (otherwise). Possible values in [0,10]. Default value 2. |
| trend.boundary | numeric: the trend boundary (rmod). The boundary beyond which an AR root is integrated in the trend component. If the modulus of the inverse real root is greater than the trend boundary, the AR root is integrated in the trend component. Below this value, the root is integrated in the transitory component. Possible values [0,1]. Default = 0.5. |
| seas.boundary | numeric: the seasonal boundary (sbound). The boundary beyond which a real negative AR root is integrated in the seasonal component. If the modulus of the inverse negative real root is greater (or equal) than Seasonal boundary, the AR root is integrated into the seasonal component. Otherwise the root is integrated into the trend or transitory component. Possible values [0,1]. Default=0.8. |
| seas.boundary.unique | numeric: the seasonal boundary (unique), (sboundatpi). The boundary beyond which a negative AR root is integrated in the seasonal component, when the root is the unique seasonal root. If the modulus of the inverse negative real root is greater (or equal) than Seasonal boundary, the AR root is integrated into the seasonal component. Otherwise the root is integrated into the trend or transitory component. Possible values [0,1]. Default=0.8. |

approximation character: the approximation mode. When the ARIMA model estimated by TRAMO does not accept an admissible decomposition, SEATS: "None" - performs an approximation; "Legacy" - replaces the model with a decomposable one; "Noisy" - estimates a new model by adding a white noise to the non-admissible model estimated by TRAMO. Default="Legacy".

## Value

returns a "JD3_UCARIMA" object

## Examples

```
seats_decompose(rjd3toolkit::sarima_model(period = 12, phi = c(0, 1), bd = 1))
```

---

# set_seats *Set Seats Specification*

---

## Description

Function allowing to customize parameters in the decomposition part (Seats) of a Tramo-Seats seasonal adjustment process. (Seats is an Arima Model Based decomposition algorithm working in conjunction with Tramo.)

## Usage

```
set_seats(
  x,
  approximation = c(NA, "None", "Legacy", "Noisy"),
  trend.boundary = NA,
  seas.boundary = NA,
  seas.boundary.unique = NA,
  seas.tolerance = NA,
  ma.boundary = NA,
  fcasts = NA,
  bcasts = NA,
  algorithm = c(NA, "Burman", "KalmanSmoother"),
  bias = NA
)
```

## Arguments

x the specification to be modified, object of class "JD3_TRAMOSEATS_SPEC", has to be generated with tramoseats_spec() function

approximation character: the approximation mode. When the ARIMA model estimated by TRAMO does not accept an admissible decomposition, SEATS: "None" - performs an approximation; "Legacy" - replaces the model with a decomposable one; "Noisy" - estimates a new model by adding a white noise to the non-admissible model estimated by TRAMO. Default="Legacy".

`trend.boundary` numeric: the trend boundary (rmod). The boundary beyond which an AR root is integrated in the trend component. If the modulus of the inverse real root is greater than the trend boundary, the AR root is integrated in the trend component. Below this value, the root is integrated in the transitory component. Possible values [0,1]. Default = 0.5.

`seas.boundary` numeric: the seasonal boundary (sbound). The boundary beyond which a real negative AR root is integrated in the seasonal component. If the modulus of the inverse negative real root is greater (or equal) than Seasonal boundary, the AR root is integrated into the seasonal component. Otherwise the root is integrated into the trend or transitory component. Possible values [0,1]. Default=0.8.

`seas.boundary.unique` numeric: the seasonal boundary (unique), (sboundatpi). The boundary beyond which a negative AR root is integrated in the seasonal component, when the root is the unique seasonal root. If the modulus of the inverse negative real root is greater (or equal) than Seasonal boundary, the AR root is integrated into the seasonal component. Otherwise the root is integrated into the trend or transitory component. Possible values [0,1]. Default=0.8.

`seas.tolerance` numeric: the seasonal tolerance (epsphi). The tolerance (measured in degrees) to allocate the AR non-real roots to the seasonal component (if the modulus of the inverse complex AR root is greater than the trend boundary and the frequency of this root differs from one of the seasonal frequencies by less than Seasonal tolerance) or the transitory component (otherwise). Possible values in [0,10]. Default value 2.

`ma.boundary` numeric: the MA unit root boundary. When the modulus of an estimated MA root falls in the range [xl, 1], it is set to xl. Possible values [0.9,1]. Default=0.95.

`bcasts`, `fcasts` numeric: the number of backasts (`bcasts`) or forecasts (`fcasts`) used in the decomposition in periods (positive values) or years (negative values). Default `bcasts` = 0. Default `fcasts` = 0.

`algorithm` character: the estimation method for the unobserved components. The choice can be made from:

1. **Burman**: the default value. May result in a significant underestimation of the components' standard deviation, as it may become numerically unstable when some roots of the MA polynomial are near 1;
2. **KalmanSmoother**: it is not disturbed by the (quasi-) unit roots in MA.

`bias` boolean. If `TRUE` (default) and decomposition is multiplicative a correction is applied when computing final components (S, T, I) in level from components estimated in log. Average of S (on complete years) and I (on the whole series) is set to 1, the correction is allocated to T.

## Value

an object of class `"JD3_TRAMOSEATS_SPEC"`.

## References

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

## See Also

tramoseats_spec().

## Examples

```
init_spec <- tramoseats_spec("rsafull")
new_spec <- set_seats(init_spec,
    approximation = "Legacy",
    trend.boundary = 0.8,
    seas.boundary = 0.5,
    fcasts = -3,
    algorithm = "KalmanSmoother",
    bias = TRUE
)
y <- rjd3toolkit::ABS$X0.2.09.10.M

sa <- tramoseats(y, spec = new_spec)
```

---

# terror *TERROR Quality Control of Outliers*

---

## Description

TRAMO for ERRORs (TERROR) controls the quality of the data by checking outliers at the end of the series

## Usage

```
terror(
  ts,
  spec = c("trfull", "tr0", "tr1", "tr2", "tr3", "tr4", "tr5"),
  nback = 1,
  context = NULL
)
```

## Arguments

| | |
|---|---|
| ts | a univariate time series. |
| spec | the model specification. Can be either the name of a predefined specification or a user-defined specification. |
| nback | number of last observations considered for the quality check. |
| context | the dictionary of variables. |

## Value

a `mts` object with 7 variables:

1. **actual**: the actual data at the end of the series;
2. **forecast**: the forecast of the actual data at the end of the series;
3. **error**: the absolute errors (= observed - forecasts);
4. **rel.error**: relative errors ("scores") : ratios between the forecast errors and the standard deviation of the forecasts of the last observations (positive values mean under-estimation);
5. **raw**: the transformed series. More especially, if the chosen model implies a log-transformation, the values are obtained after a log-transformation. Other transformations, such leap year corrections or length-of periods corrections may also be used;
6. **fraw**: the forecast of the transformed series.;
7. **efraw**: the absolute errors of the transformed series.

## Examples

```
if (rjd3jars::check_java_version(silent = TRUE)) {
  terror(rjd3toolkit::ABS$X0.2.09.10.M, nback = 2)
}
```

---

`tramo` *TRAMO model, pre-adjustment in TRAMO-SEATS*

---

## Description

allows to model the series with a Reg-Arima model, estimate outlier, calendar or other regression effects and produce forecasts

## Usage

```
tramo(
  ts,
  spec = c("trfull", "tr0", "tr1", "tr2", "tr3", "tr4", "tr5"),
  context = NULL,
  userdefined = NULL
)

tramo_fast(
  ts,
  spec = c("trfull", "tr0", "tr1", "tr2", "tr3", "tr4", "tr5"),
  context = NULL,
  userdefined = NULL
)
```

## Arguments

| | |
|---|---|
| ts | a univariate time series. |
| spec | the model specification. Can be either the name of a predefined specification or a user-defined specification. |
| context | the dictionary of variables. |
| userdefined | a vector containing the additional output variables (see tramoseats_dictionary()). |

## Value

the tramo() function returns a list with the results ("JD3_tramo_rslts" object), the estimation specification and the result specification, while tramo_fast() is a faster function that only returns the results.

## Examples

```
library("rjd3toolkit")

y <- rjd3toolkit::ABS$X0.2.09.10.M
sp <- tramo_spec("trfull")
sp <- add_outlier(sp,
    type = c("AO"), c("2015-01-01", "2010-01-01")
)

tramo_fast(y, spec = sp)

sp <- set_transform(
    set_tradingdays(
        set_easter(sp, enabled = FALSE),
        option = "workingdays"
    ),
    fun = "None"
)

tramo_fast(y, spec = sp)

sp <- set_outlier(sp, outliers.type = c("AO"))

tramo_fast(y, spec = sp)
```

---

# tramoseats *Seasonal Adjustment with TRAMO-SEATS*

---

## Description

Seasonal Adjustment with TRAMO-SEATS

## Usage

```
tramoseats(
  ts,
  spec = c("rsafull", "rsa0", "rsa1", "rsa2", "rsa3", "rsa4", "rsa5"),
  context = NULL,
  userdefined = NULL
)

tramoseats_fast(
  ts,
  spec = c("rsafull", "rsa0", "rsa1", "rsa2", "rsa3", "rsa4", "rsa5"),
  context = NULL,
  userdefined = NULL
)

.jtramoseats(
  ts,
  spec = c("rsafull", "rsa0", "rsa1", "rsa2", "rsa3", "rsa4", "rsa5"),
  context = NULL,
  userdefined = NULL
)
```

## Arguments

| | |
|---|---|
| ts | a univariate time series. |
| spec | the model specification. Can be either the name of a predefined specification or a user-defined specification. |
| context | the dictionary of variables. |
| userdefined | a vector containing the additional output variables (see tramoseats_dictionary()). |

## Value

The tramoseats() function returns a list with the results, the estimation specification and the result specification, while tramoseats_fast() is a faster function that only returns the results. The .jtramoseats() functions only results the java object to custom outputs in other packages (use rjd3toolkit::dictionary() to get the list of variables and rjd3toolkit::result() to get a specific variable).

## Examples

```
library("rjd3toolkit")

sp <- tramoseats_spec("rsafull")
y <- rjd3toolkit::ABS$X0.2.09.10.M

tramoseats(y, spec = sp)
tramoseats_fast(y, spec = sp)
```

```
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"
)

tramoseats(y, spec = sp)
tramoseats_fast(y, spec = sp)
```

---

`tramoseats_dictionary` *TRAMO-SEATS dictionary*

---

## Description

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

## Usage

```
tramoseats_dictionary()

tramoseats_full_dictionary()
```

## Details

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

## Value

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

## Examples

```
# Visualize the dictionary
print(tramoseats_dictionary())
summary(tramoseats_dictionary())
```

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

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

# Generate the corresponding output in an estimation
y <- rjd3toolkit::ABS$X0.2.09.10.M

m <- tramoseats(y, "rsafull", userdefined=user_defined_output)

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

---

`tramoseats_option` *Set an option for tramoseats*

---

## Description

Set an option for tramoseats

## Usage

```
tramoseats_option(name, obj)
```

## Arguments

| | |
|---|---|
| `name` | Name of the option |
| `obj` | Option |

## Value

Invisibly returns `NULL`

## Examples

```
tramoseats_option("test", "DUMMY")
```

`tramoseats_revisions` *Revisions History*

## Description

Computes revisions history

## Usage

```
tramoseats_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 `tramoseats_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 `tramoseats_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 `tramoseats_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

```
if (rjd3jars::check_java_version(silent = TRUE)){
    s <- rjd3toolkit::ABS$X0.2.09.10.M
    sa_mod <- tramoseats(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 <- tramoseats_revisions(s, sa_mod$result_spec, data_ids, ts_ids, cmp_ids)
}
```

---

# tramoseats_spec *TRAMO/TRAMO-SEATS Default Specification*

---

## Description

Set of functions(`tramoseats_spec()`,`tramo_spec()`) to create default specifications associated with the TRAMO-SEATS seasonal adjustment method. Specification creation can be restricted to the tramo part with the `tramo_spec()` function.

## Usage

```
tramo_spec(name = c("trfull", "tr0", "tr1", "tr2", "tr3", "tr4", "tr5"))

tramoseats_spec(
  name = c("rsafull", "rsa0", "rsa1", "rsa2", "rsa3", "rsa4", "rsa5")
)
```

## Arguments

| | |
|---|---|
| `name` | the name of a predefined specification. |

## Details

Without argument `tramo_spec()` yields a TR5 specification

without argument `tramoseats_spec()` yields a RSA5 specification

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

| Identifier | Log/level detection | Outliers detection | Calendar effects | ARIMA |
|---|---|---|---|---|
| RSA0/TR0 | *NA* | *NA* | *NA* | Airline(+mean) |
| RSA1/TR1 | automatic | AO/LS/TC | *NA* | Airline(+mean) |
| RSA2/TR2 | automatic | AO/LS/TC | 2 td vars + Easter | Airline(+mean) |
| RSA3/TR3 | automatic | AO/LS/TC | *NA* | automatic |
| RSA4/TR4 | automatic | AO/LS/TC | 2 td vars + Easter | automatic |
| RSA5/TR5 | automatic | AO/LS/TC | 7 td vars + Easter | automatic |
| RSAfull/TRfull | automatic | AO/LS/TC | automatic | automatic |

## Value

an object of class `"JD3_TRAMOSEATS_SPEC"` (`tramoseats_spec()`) or `"JD3_TRAMO_SPEC"` (`tramo_spec()`).

## See Also

1. 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()`;
2. To set the decomposition parameters: `set_seats()`;
3. To set the benchmarking parameters: `rjd3toolkit::set_benchmarking()`.

## Examples

```
init_spec <- tramoseats_spec()
init_spec <- tramo_spec()
init_spec <- tramoseats_spec("rsa3")
init_spec <- tramo_spec("tr3")
```

---

# tramo_forecast *Forecasts with TRAMO*

---

## Description

Forecasts with TRAMO

### Usage

```
tramo_forecast(
  ts,
  spec = c("trfull", "tr0", "tr1", "tr2", "tr3", "tr4", "tr5"),
  nf = -1,
  context = NULL
)
```

### Arguments

| | |
|---|---|
| ts | a univariate time series. |
| spec | the model specification. Can be either the name of a predefined specification or a user-defined specification. |
| nf | the forecasting horizon (numeric). The forecast length is in periods (positive values) or years (negative values). By default, the program generates a one-year forecast (nf = -1). |
| context | the dictionary of variables. |

### Value

a mts object with 7 variables:

- forecast the forecast of the actual data at the end of the series.
- error standard deviation of the forecast.
- fraw the forecast of the transformed series.
- efraw the standard deviation of the forecast of the transformed series.

### Examples

```
if (rjd3jars::check_java_version(silent = TRUE)){
    tramo_forecast(rjd3toolkit::ABS$X0.2.09.10.M)
}
```

---

## tramo_outliers *Outlier Detection with a Tramo Model*

---

### Description

Tramo is a particular regarima model estimation algorithm, mainly used to linearized the series before performing a decomposition with Seats

## Usage

```
tramo_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,
  ml = FALSE,
  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 outliers' detection procedure. If equal to 0 the critical value for the outliers' detection procedure is automatically determined by the number of observations. |
| ml | Use of maximum likelihood (otherwise approximation by means of Hannan-Rissanen). |
| 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.

## Examples

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

# Index


.jd2r_spec_tramo *(jd3_utilities)*, 4
.jd2r_spec_tramoseats *(jd3_utilities)*, 4
.jtramoseats *(tramoseats)*, 14
.r2jd_spec_tramo *(jd3_utilities)*, 4
.r2jd_spec_tramoseats *(jd3_utilities)*, 4
.tramoseats_rslts *(jd3_utilities)*, 4

deprecated-rjd3tramoseats, 2

fast_tramo *(deprecated-rjd3tramoseats)*, 2
fast_tramoseats *(deprecated-rjd3tramoseats)*, 2

get_tramoseats_option, 3

jd3_utilities, 4

refresh, 5
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::sarima_model(), *9*
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*

seats_decompose, 9
set_seats, 10
set_seats(), *20*
spec_tramo *(deprecated-rjd3tramoseats)*, 2
spec_tramoseats *(deprecated-rjd3tramoseats)*, 2

terror, 12
tramo, 13
tramo_fast *(tramo)*, 13
tramo_forecast, 20
tramo_outliers, 21
tramo_refresh *(refresh)*, 5
tramo_spec *(tramoseats_spec)*, 19
tramoseats, 14
tramoseats_dictionary, 16
tramoseats_dictionary(), *3*, *14*, *15*, *18*
tramoseats_fast *(tramoseats)*, 14
tramoseats_full_dictionary *(tramoseats_dictionary)*, 16
tramoseats_option, 17
tramoseats_refresh *(refresh)*, 5
tramoseats_revisions, 18
tramoseats_spec, 19
tramoseats_spec(), *12*

userdefined_variables_tramoseats *(deprecated-rjd3tramoseats)*, 2