

# Package ‘rjd3tramoseats’

March 11, 2026

**Type** Package

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

**Version** 3.7.1

**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), rjd3toolkit (>= 3.7.1), RProtoBuf (>= 0.4.20), stats, utils

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**SystemRequirements** Java (>= 17)

**Collate** 'deprecated.R' 'print.R' 'utils.R' 'tramoseats\_rslts.R'  
'tramoseats\_spec.R' 'revisions.R' 'seats.R' 'seats\_spec.R'  
'tramo\_generic.R' 'tramo\_outliers.R' 'tramo\_spec.R'  
'tramoseats.R' 'zzz.R'

**Suggests** spelling

**Language** en-US

**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]

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

**Repository** CRAN

**Date/Publication** 2026-03-11 17:00:02 UTC

## Contents

deprecated-rjd3tramoseats . . . . .	2
jd3_utilities . . . . .	3
refresh . . . . .	4
seats_decompose . . . . .	7
set_seats . . . . .	8
terror . . . . .	10
tramo . . . . .	12
tramoseats . . . . .	13
tramoseats_dictionary . . . . .	14
tramoseats_revisions . . . . .	15
tramoseats_spec . . . . .	17
tramo_forecast . . . . .	18
tramo_outliers . . . . .	19

**Index** **21**

---

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 <a href="#">tramoseats_dictionary()</a> ).
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()`

---

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

---

**Description**

Functions `tramo_refresh()` and `tramo_refresh()` allow 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**

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

tramo_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 "TRfull" or "RSAfull" 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. This revision applies to the estimation done in Tramo (pre-adjustment phase), Seats will then run a new decomposition which might be in some (rare) cases based on a different model.

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-arma model and replacing forecasts by new raw data points.

**FixedParameters:** pre-adjustment reg-arma 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.

**Complete:** All the parameters are re-identified and re-estimated, unless constrained in the domain spec.

## Value

a new specification, an object of class "JD3\_TRAMOSEATS\_SPEC" or "JD3\_TRAMO\_SPEC".

## References

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

## 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_tramoseats_1 <- tramoseats_spec("rsafull")

# first estimation
sa_tramoseats <- tramoseats(y_raw, spec_tramoseats_1)
# refreshing the specification
current_result_spec <- sa_tramoseats$result_spec
current_domain_spec <- sa_tramoseats$estimation_spec

# policy = "Fixed"
spec_tramoseats_ref <- tramoseats_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_tramoseats_ref <- tramoseats(y_new, spec_tramoseats_ref)
# policy = "Outliers"
spec_tramoseats_ref <- tramoseats_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_tramoseats_ref <- tramoseats(y_new, spec_tramoseats_ref)

# policy = "Current"
spec_tramoseats_ref <- tramoseats_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_tramoseats_ref <- tramoseats(y_new, spec_tramoseats_ref) #'
# same procedure using tramo_refresh
# specification for first estimation
spec_1 <- tramo_spec("tr3")

```

```

# first estimation
tramo_model <- tramo(y_raw, spec_1)
tramo_model$estimation_spec

# refreshing the specification
current_result_spec <- tramo_model$result_spec
current_domain_spec <- tramo_model$estimation_spec

# policy = "Fixed"
spec_1_ref <- tramo_refresh(current_result_spec, # point spec to be refreshed
                           current_domain_spec, # domain spec (set of constraints)
                           policy = "Fixed"
                           )

# 2nd estimation with refreshed specification
tramo_model_ref <- tramo(y_new, spec_1_ref)

```

---

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 backcasts (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: <ol style="list-style-type: none"> <li>1. <b>Burman</b>: 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;</li> <li>2. <b>KalmanSmoother</b>: it is not disturbed by the (quasi-) unit roots in MA.</li> </ol>
bias	TODO.

**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**

```
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 <a href="#">tramoseats_dictionary()</a> ).

**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("A0"))

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 <a href="#">tramoseats_dictionary()</a> ).

**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("A0"), 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 user-defined 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
```

**Description**

Computes revisions history

**Usage**

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

**Arguments**

<code>ts</code>	The time series used for the estimation.
<code>spec</code>	The specification used.
<code>data_ids</code>	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 <a href="#">tramoseats_dictionary()</a> ). See example.
<code>ts_ids</code>	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 <a href="#">tramoseats_dictionary()</a> ). See example.
<code>cmp_ids</code>	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 <a href="#">tramoseats_dictionary()</a> ). As many series as periods between <code>start</code> and <code>end</code> will be exported. See example.
<code>context</code>	The context of the specification.

**Value**

returns a list

**Examples**

```
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/TR3	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.
- efrac the standard deviation of the forecast of the transformed series.

**Examples**

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

---

tramo_outliers	<i>Outlier Detection with a Tramo Model</i>
----------------	---

---

**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

<code>y</code>	the dependent variable (a <code>ts</code> object).
<code>order, seasonal</code>	the orders of the ARIMA model.
<code>mean</code>	Boolean to include or not the mean.
<code>X</code>	user defined regressors (other than calendar).
<code>X.td</code>	calendar regressors.
<code>ao, ls, so, tc</code>	Boolean to indicate which type of outliers should be detected.
<code>cv</code>	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.
<code>ml</code>	Use of maximum likelihood (otherwise approximation by means of Hannan-Rissanen).
<code>clean</code>	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), 3  
.jd2r\_spec\_tramoseats (jd3\_utilities), 3  
.jtramoseats (tramoseats), 13  
.r2jd\_spec\_tramo (jd3\_utilities), 3  
.r2jd\_spec\_tramoseats (jd3\_utilities), 3  
.tramoseats\_rslts (jd3\_utilities), 3

deprecated-rjd3tramoseats, 2

fast\_tramo (deprecated-rjd3tramoseats),  
2

fast\_tramoseats  
(deprecated-rjd3tramoseats), 2

jd3\_utilities, 3

refresh, 4

rjd3toolkit::add\_outlier(), 18  
rjd3toolkit::add\_ramp(), 18  
rjd3toolkit::add\_usrdefvar(), 18  
rjd3toolkit::dictionary(), 14  
rjd3toolkit::remove\_outlier(), 18  
rjd3toolkit::remove\_ramp(), 18  
rjd3toolkit::result(), 14  
rjd3toolkit::sarima\_model(), 7  
rjd3toolkit::set\_arima(), 18  
rjd3toolkit::set\_automodel(), 18  
rjd3toolkit::set\_basic(), 18  
rjd3toolkit::set\_benchmarking(), 18  
rjd3toolkit::set\_easter(), 18  
rjd3toolkit::set\_estimate(), 18  
rjd3toolkit::set\_outlier(), 18  
rjd3toolkit::set\_tradingdays(), 18  
rjd3toolkit::set\_transform(), 18

seats\_decompose, 7  
set\_seats, 8  
set\_seats(), 18

spec\_tramo (deprecated-rjd3tramoseats),  
2

spec\_tramoseats  
(deprecated-rjd3tramoseats), 2

terror, 10  
tramo, 12  
tramo\_fast (tramo), 12  
tramo\_forecast, 18  
tramo\_outliers, 19  
tramo\_refresh (refresh), 4  
tramo\_spec (tramoseats\_spec), 17  
tramoseats, 13  
tramoseats\_dictionary, 14  
tramoseats\_dictionary(), 3, 12, 14, 16  
tramoseats\_fast (tramoseats), 13  
tramoseats\_full\_dictionary  
(tramoseats\_dictionary), 14  
tramoseats\_refresh (refresh), 4  
tramoseats\_revisions, 15  
tramoseats\_spec, 17  
tramoseats\_spec(), 10

userdefined\_variables\_tramoseats  
(deprecated-rjd3tramoseats), 2