```
var1 <- ts(rnorm(length(myseries))*10, start = start(myseries), frequency = 12)
var2 <- ts(rnorm(length(myseries))*100, start = start(myseries), frequency = 12)
var <- ts.union(var1, var2)
myspec2 <- tramoseats_spec(myspec, tradingdays.mauto = "Unused",
                           tradingdays.option = "WorkingDays",
                           easter.type = "Standard",
                           automdl.enabled = FALSE, arima.mu = TRUE,
                           usrdef.varEnabled = TRUE, usrdef.var = var)
s_preVar(myspec2)
mysa2 <- tramoseats(myseries, myspec2,
                    userdefined = c("decomposition.sa_lin_f",
                                    "decomposition.sa_lin_e"))
mysa2
plot(mysa2)
plot(mysa2$regarima)
plot(mysa2$decomposition)
```

`tramoseats_spec` *TRAMO-SEATS model specification*

Description

Function to create (and/or modify) a `c("SA_spec", "TRAMO_SEATS")` class object with the SA model specification for the TRAMO-SEATS method. It can be done from a pre-defined 'JDemetra+' model specification (a `character`), a previous specification (`c("SA_spec", "TRAMO_SEATS")` object) or a seasonal adjustment model (`c("SA", "TRAMO_SEATS")` object).

Usage

```
tramoseats_spec(
  spec = c("RSAfull", "RSA0", "RSA1", "RSA2", "RSA3", "RSA4", "RSA5"),
  preliminary.check = NA,
  estimate.from = NA_character_,
  estimate.to = NA_character_,
  estimate.first = NA_integer_,
  estimate.last = NA_integer_,
  estimate.exclFirst = NA_integer_,
  estimate.exclLast = NA_integer_,
  estimate.tol = NA_integer_,
  estimate.eml = NA,
  estimate.urfinal = NA_integer_,
  transform.function = c(NA, "Auto", "None", "Log"),
  transform.fct = NA_integer_,
  usrdef.outliersEnabled = NA,
  usrdef.outliersType = NA,
  usrdef.outliersDate = NA,
  usrdef.outliersCoef = NA,
```

```
 usrdef.varEnabled = NA,
 usrdef.var = NA,
 usrdef.varType = NA,
 usrdef.varCoef = NA,
 tradingdays.mauto = c(NA, "Unused", "FTest", "WaldTest"),
 tradingdays.pftd = NA_integer_,
tradingdays.option = c(NA, "TradingDays", "WorkingDays", "UserDefined", "None"),
 tradingdays.leapyear = NA,
 tradingdays.stocktd = NA_integer_,
 tradingdays.test = c(NA, "Separate_T", "Joint_F", "None"),
easter.type = c(NA, "Unused", "Standard", "IncludeEaster", "IncludeEasterMonday"),
 easter.julian = NA,
 easter.duration = NA_integer_,
 easter.test = NA,
 outlier.enabled = NA,
 outlier.from = NA_character_,
 outlier.to = NA_character_,
 outlier.first = NA_integer_,
 outlier.last = NA_integer_,
 outlier.exclFirst = NA_integer_,
 outlier.exclLast = NA_integer_,
 outlier.ao = NA,
 outlier.tc = NA,
 outlier.ls = NA,
 outlier.so = NA,
 outlier.usedefcv = NA,
 outlier.cv = NA_integer_,
 outlier.eml = NA,
 outlier.tcrate = NA_integer_,
 automdl.enabled = NA,
 automdl.acceptdefault = NA,
 automdl.cancel = NA_integer_,
 automdl.ub1 = NA_integer_,
 automdl.ub2 = NA_integer_,
 automdl.armalimit = NA_integer_,
 automdl.reducecv = NA_integer_,
 automdl.ljungboxlimit = NA_integer_,
 automdl.compare = NA,
 arima.mu = NA,
 arima.p = NA_integer_,
 arima.d = NA_integer_,
 arima.q = NA_integer_,
 arima.bp = NA_integer_,
 arima.bd = NA_integer_,
 arima.bq = NA_integer_,
 arima.coefEnabled = NA,
 arima.coef = NA,
 arima.coefType = NA,
```

```
  fcst.horizon = NA_integer_,
  seats.predictionLength = NA_integer_,
  seats.approx = c(NA, "None", "Legacy", "Noisy"),
  seats.trendBoundary = NA_integer_,
  seats.seasdBoundary = NA_integer_,
  seats.seasdBoundary1 = NA_integer_,
  seats.seasTol = NA_integer_,
  seats.maBoundary = NA_integer_,
  seats.method = c(NA, "Burman", "KalmanSmoother", "McElroyMatrix")
)
```

Arguments

spec	a TRAMO-SEATS model specification. It can be the 'JDemetra+' name (character) of a predefined TRAMO-SEATS model specification (see *Details*), an object of class c("SA_spec","TRAMO_SEATS") or an object of class c("SA", "TRAMO_SEATS"). The default is "RSAfull".
preliminary.check	a logical to check the quality of the input series and exclude highly problematic series e.g. the series with a number of identical observations and/or missing values above pre-specified threshold values. The time span of the series, which is the (sub)period used to estimate the regarima model, is controlled by the following six variables: estimate.from, estimate.to, estimate.first, estimate.last, estimate.exclFirst and estimate.exclLast; where estimate.from and estimate.to have priority over the remaining span control variables, estimate.last and estimate.first have priority over estimate.exclFirst and estimate.exclLast, and estimate.last has priority over estimate.first. Default= "All".
estimate.from	a character in format "YYYY-MM-DD" indicating the start of the time span (e.g. "1900-01-01"). It can be combined with the parameter estimate.to.
estimate.to	a character in format "YYYY-MM-DD" indicating the end of the time span (e.g. "2020-12-31"). It can be combined with the parameter estimate.from.
estimate.first	numeric, the number of periods considered at the beginning of the series.
estimate.last	numeric, the number of periods considered at the end of the series.
estimate.exclFirst	numeric, the number of periods excluded at the beginning of the series. It can be combined with the parameter estimate.exclLast.
estimate.exclLast	numeric, the number of periods excluded at the end of the series. It can be combined with the parameter estimate.exclFirst.
estimate.tol	numeric, the convergence tolerance. The absolute changes in the log-likelihood function are compared to this value to check for the convergence of the estimation iterations.
estimate.eml	logical, the exact maximum likelihood estimation. If TRUE, the program performs an exact maximum likelihood estimation. If FASLE, the Unconditional Least Squares method is used.

estimate.urfinal
: numeric, the final unit root limit. The threshold value for the final unit root test for identification of differencing orders. If the magnitude of an AR root for the final model is smaller than this number, then a unit root is assumed, the order of the AR polynomial is reduced by one and the appropriate order of the differencing (non-seasonal, seasonal) is increased.

transform.function
: the transformation of the input series: "None" = no transformation of the series; "Log" = takes the log of the series; "Auto" = the program tests for the log-level specification.

transform.fct
: numeric controlling the bias in the log/level pre-test: transform.fct > 1 favours levels, transform.fct< 1 favours logs. Considered only when transform.function is set to "Auto".

Control variables for the pre-specified outliers. Said pre-specified outliers are used in the model only when enabled (usrdef.outliersEnabled=TRUE) and when the outliers' type (usrdef.outliersType) and date (usrdef.outliersDate) are provided.

usrdef.outliersEnabled
: logical. If TRUE, the program uses the pre-specified outliers.

usrdef.outliersType
: a vector defining the outliers' type. Possible types are: ("AO") = additive, ("LS") = level shift, ("TC") = transitory change, ("SO") = seasonal outlier. E.g.: usrdef.outliersType= c("AO","AO","LS").

usrdef.outliersDate
: a vector defining the outliers' date. The dates should be characters in format "YYYY-MM-DD". E.g.: usrdef.outliersDate= c("2009-10-01","2005-02-01","2003-04-01").

usrdef.outliersCoef
: a vector providing fixed coefficients for the outliers. The coefficients can't be fixed if the parameter transform.function is set to "Auto" (i.e. if the series transformation needs to be pre-defined.) E.g.: usrdef.outliersCoef= c(200,170,20).

Control variables for the user-defined variables:

usrdef.varEnabled
: logical If TRUE, the program uses the user-defined variables.

usrdef.var
: a time series (ts) or a matrix of time series (mts) containing the user-defined variables.

usrdef.varType
: a vector of character(s) defining the user-defined variables component type. Possible types are: "Undefined", "Series", "Trend", "Seasonal", "SeasonallyAdjusted", "Irregular", "Calendar". To use the user-defined calendar regressors, the type "Calendar" must be defined in conjunction with tradingdays.option = "UserDefined". Otherwise, the program will automatically set usrdef.varType = "Undefined".

usrdef.varCoef
: a vector providing fixed coefficients for the user-defined variables. The coefficients can't be fixed if transform.function is set to "Auto" (i.e. if the series transformation needs to be pre-defined).

tradingdays.mauto
: defines whether the calendar effects should be added to the model manually ("Unused") or automatically. During the automatic selection, the choice of the number of calendar variables can be based on the F-Test ("FTest") or the Wald Test ("WaldTest"); the model with higher F value is chosen, provided that it is higher than tradingdays.pftd).

tradingdays.pftd
: numeric. The p-value used in the test specified by the automatic parameter (tradingdays.mauto) to assess the significance of the pre-tested calendar effects variables and whether they should be included in the RegArima model.

Control variables for the manual selection of calendar effects variables (tradingdays.mauto is set to "Unused"):

tradingdays.option
: to choose the trading days regression variables: "TradingDays" = six day-of-the-week regression variables; "WorkingDays" = one working/non-working day contrast variable; "None" = no correction for trading days and working days effects; "UserDefined" = user-defined trading days regressors (regressors must be defined by the usrdef.var argument with usrdef.varType set to "Calendar" and usrdef.varEnabled = TRUE). "None" must also be chosen for the "day-of-week effects" correction (and tradingdays.stocktd must be modified accordingly).

tradingdays.leapyear
: logical. Specifies if the leap-year correction should be included. If TRUE, the model includes the leap-year effect.

tradingdays.stocktd
: numeric indicating the day of the month when inventories and other stock are reported (to denote the last day of the month set the variable to 31). Modifications of this variable are taken into account only when tradingdays.option is set to "None".

tradingdays.test
: defines the pre-tests of the trading day effects: "None" = calendar variables are used in the model without pre-testing; "Separate_T" = a t-test is applied to each trading day variable separately and the trading day variables are included in the RegArima model if at least one t-statistic is greater than 2.6 or if two t-statistics are greater than 2.0 (in absolute terms); "Joint_F" = a joint F-test of significance of all the trading day variables. The trading day effect is significant if the F statistic is greater than 0.95.

easter.type
: acharacter that specifies the presence and the length of the Easter effect: "Unused" = the Easter effect is not considered; "Standard" = influences the period of n days strictly before Easter Sunday; "IncludeEaster" = influences the entire period (n) up to and including Easter Sunday; "IncludeEasterMonday" = influences the entire period (n) up to and including Easter Monday.

easter.julian
: logical. If TRUE, the program uses the Julian Easter (expressed in Gregorian calendar).

easter.duration
: numeric indicating the duration of the Easter effect (length in days, between 1 and 15).

`easter.test` `logical`. If `TRUE`, the program performs a t-test for the significance of the Easter effect. The Easter effect is considered as significant if the modulus of t-statistic is greater than 1.96.

`outlier.enabled` `logical`. If `TRUE`, the automatic detection of outliers is enabled in the defined time span.

The time span of the series to be searched for outliers is controlled by the following six variables: `outlier.from`, `outlier.to`, `outlier.first`, `outlier.last`, `outlier.exclFirst` and `outlier.exclLast`; where `outlier.from` and `outlier.to` have priority over the remaining span control variables, `outlier.last` and `outlier.first` have priority over `outlier.exclFirst` and `outlier.exclLast`, and `outlier.last` has priority over `outlier.first`.

`outlier.from` a character in format "YYYY-MM-DD" indicating the start of the time span (e.g. "1900-01-01"). It can be combined with `outlier.to`.

`outlier.to` a character in format "YYYY-MM-DD" indicating the end of the time span (e.g. "2020-12-31"). It can be combined with `outlier.from`.

`outlier.first` `numeric` specifying the number of periods considered at the beginning of the series.

`outlier.last` `numeric` specifying the number of periods considered at the end of the series.

`outlier.exclFirst` `numeric` specifying the number of periods excluded at the beginning of the series. It can be combined with `outlier.exclLast`.

`outlier.exclLast` `numeric` specifying the number of periods excluded at the end of the series. It can be combined with `outlier.exclFirst`.

`outlier.ao` `logical`. If `TRUE`, the automatic detection of additive outliers is enabled (`outlier.enabled` must also be set to `TRUE`).

`outlier.tc` `logical`. If `TRUE`, the automatic detection of transitory changes is enabled (`outlier.enabled` must also be set to `TRUE`).

`outlier.ls` `logical`. If `TRUE`, the automatic detection of level shifts is enabled (`outlier.enabled` must also be set to `TRUE`).

`outlier.so` `logical`. If `TRUE`, the automatic detection of seasonal outliers is enabled (`outlier.enabled` must also be set to `TRUE`).

`outlier.usedefcv` `logical`. If `TRUE`, the critical value for the outliers' detection procedure is automatically determined by the number of observations in the outlier detection time span. If `FALSE`, the procedure uses the entered critical value (`outlier.cv`).

`outlier.cv` `numeric`. The entered critical value for the outliers' detection procedure. The modification of this variable is only taken in to account when `outlier.usedefcv` is set to `FALSE`.

`outlier.eml` `logical` for the exact likelihood estimation method. It controls the method applied for a parameter estimation in the intermediate steps of the automatic detection and correction of outliers. If `TRUE`, an exact likelihood estimation method is used. When `FALSE`, the fast Hannan-Rissanen method is used.

`outlier.tcrate` `numeric`. The rate of decay for the transitory change outlier.

`automdl.enabled`
: `logical`. If `TRUE`, the automatic modelling of the ARIMA model is enabled. If `FALSE`, the parameters of the ARIMA model can be specified.

  Control variables for the automatic modelling of the ARIMA model (`automdl.enabled` is set to `TRUE`):

`automdl.acceptdefault`
: `logical`. If `TRUE`, the default model (ARIMA(0,1,1)(0,1,1)) may be chosen in the first step of the automatic model identification. If the Ljung-Box Q statistics for the residuals is acceptable, the default model is accepted and no further attempt will be made to identify another model.

`automdl.cancel`
: `numeric`, the cancellation limit. If the difference in moduli of an AR and an MA roots (when estimating ARIMA(1,0,1)(1,0,1) models in the second step of the automatic identification of the differencing orders) is smaller than the cancellation limit, the two roots are assumed equal and canceled out.

`automdl.ub1`
: `numeric`, the first unit root limit. It is the threshold value for the initial unit root test in the automatic differencing procedure. When one of the roots in the estimation of the ARIMA(2,0,0)(1,0,0) plus mean model, performed in the first step of the automatic model identification procedure, is larger than first unit root limit in modulus, it is set equal to unity.

`automdl.ub2`
: `numeric`, the second unit root limit. When one of the roots in the estimation of the ARIMA(1,0,1)(1,0,1) plus mean model, which is performed in the second step of the automatic model identification procedure, is larger than second unit root limit in modulus, it is checked if there is a common factor in the corresponding AR and MA polynomials of the ARMA model that can be canceled (see `automdl.cancel`). If there is no cancellation, the AR root is set equal to unity (i.e. the differencing order changes).

`automdl.armalimit`
: `numeric`, the arma limit. It is the threshold value for t-statistics of ARMA coefficients and the constant term used for the final test of model parsimony. If the highest order ARMA coefficient has a t-value smaller than this value in magnitude, the order of the model is reduced. If the constant term has a t-value smaller than the ARMA limit in magnitude, it is removed from the set of regressors.

`automdl.reducecv`
: `numeric`, ReduceCV. The percentage by which the outlier critical value will be reduced when an identified model is found to have a Ljung-Box statistic with an unacceptable confidence coefficient. The parameter should be between 0 and 1, and will only be active when automatic outlier identification is enabled. The reduced critical value will be set to (1-ReduceCV)xCV, where CV is the original critical value.

`automdl.ljungboxlimit`
: `numeric`, the Ljung Box limit, setting the acceptance criterion for the confidence intervals of the Ljung-Box Q statistic. If the LjungBox Q statistics for the residuals of a final model is greater than Ljung Box limit, then the model is rejected, the outlier critical value is reduced, and model and outlier identification (if specified) is redone with a reduced value.

`automdl.compare`
: `logical`. If `TRUE`, the program compares the model identified by the automatic procedure to the default model (ARIMA(0,1,1)(0,1,1)) and the model with the

best fit is selected. Criteria considered are residual diagnostics, the model structure and the number of outliers.

Control variables for the non-automatic modelling of the ARIMA model (`automdl.enabled` is set to `FALSE`):

`arima.mu` `logical`. If `TRUE`, the mean is considered as part of the ARIMA model.

`arima.p` `numeric`. The order of the non-seasonal autoregressive (AR) polynomial.

`arima.d` `numeric`. The regular differencing order.

`arima.q` `numeric`. The order of the non-seasonal moving average (MA) polynomial.

`arima.bp` `numeric`. The order of the seasonal autoregressive (AR) polynomial.

`arima.bd` `numeric`. The seasonal differencing order.

`arima.bq` `numeric`. The order of the seasonal moving average (MA) polynomial.

Control variables for the user-defined ARMA coefficients. Such coefficients can be defined for the regular and seasonal autoregressive (AR) polynomials and moving average (MA) polynomials. The model considers the coefficients only if the procedure for their estimation (`arima.coefType`) is provided, and the number of provided coefficients matches the sum of (regular and seasonal) AR and MA orders (`p,q,bp,bq`).

`arima.coefEnabled`

`logical`. If `TRUE`, the program uses the user-defined ARMA coefficients.

`arima.coef` a vector providing the coefficients for the regular and seasonal AR and MA polynomials. The length of the vector must be equal to the sum of the regular and seasonal AR and MA orders. The coefficients shall be provided in the following order: regular AR (*Phi* - `p` elements), regular MA (*Theta* - `q` elements), seasonal AR (*BPhi* - `bp` elements) and seasonal MA (*BTheta* - `bq` elements). E.g.: `arima.coef=c(0.6,0.7)` with `arima.p=1`, `arima.q=0`,`arima.bp=1` and `arima.bq=0`.

`arima.coefType` avector defining the ARMA coefficients estimation procedure. Possible procedures are: `"Undefined"` = no use of user-defined input (i.e. coefficients are estimated), `"Fixed"` = fixes the coefficients at the value provided by the user, `"Initial"` = the value defined by the user is used as initial condition. For orders for which the coefficients shall not be defined, the `arima.coef` can be set to `NA` or `0` or the `arima.coefType` can be set to `"Undefined"`. E.g.: `arima.coef = c(-0.8,-0.6,NA)`, `arima.coefType = c("Fixed","Fixed","Undefined")`.

`fcst.horizon` `numeric`, the forecasting horizon. The length of the forecasts generated by the RegARIMA model in periods (positive values) or years (negative values). By default, the program generates two years forecasts (`fcst.horizon` set to `-2`).

`seats.predictionLength`

integer: the number of forecasts used in the decomposition. Negative values correspond to number of years. Default=-1.

`seats.approx` 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".

`seats.trendBoundary`
: numeric: the trend boundary. 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.

`seats.seasdBoundary`
: numeric: the seasonal boundary. The boundary beyond which a 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.

`seats.seasdBoundary1`
: numeric: the seasonal boundary (unique). 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.

`seats.seasTol`
: numeric: the seasonal tolerance. 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.

`seats.maBoundary`
: 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.

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

  - `"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;
  - `"KalmanSmoother"`: it is not disturbed by the (quasi-) unit roots in MA;
  - `"McElroyMatrix"`: it has the same stability issues as the Burman's algorithm.

Details

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

Identifier	Log/level detection	Outliers detection	Calendar effects	ARIMA
RSA0	*NA*	*NA*	*NA*	Airline(+mean)
RSA1	automatic	AO/LS/TC	*NA*	Airline(+mean)
RSA2	automatic	AO/LS/TC	2 td vars + Easter	Airline(+mean)
RSA3	automatic	AO/LS/TC	*NA*	automatic
RSA4	automatic	AO/LS/TC	2 td vars + Easter	automatic
RSA5	automatic	AO/LS/TC	7 td vars + Easter	automatic
RSAfull	automatic	AO/LS/TC	automatic	automatic

Value

A two-element list of class `c("SA_spec", "TRAMO_SEATS")`, containing: (1) an object of class `c("regarima_spec", "TRAMO_SEATS")` with the RegARIMA model specification, (2) an object of class `c("seats_spec", "data.frame")` with the SEATS algorithm specification. Each component refers to a different part of the SA model specification, mirroring the arguments of the function (for details see the function arguments in the description). Each lowest-level component (except span, pre-specified outliers, user-defined variables and pre-specified ARMA coefficients) is structured as a data frame with columns denoting different variables of the model specification and rows referring to:

- first row: the base specification, as provided within the argument `spec`;
- second row: user modifications as specified by the remaining arguments of the function (e.g.: `arima.d`);
- and third row: the final model specification.

  The final specification (third row) shall include user modifications (row two) unless they were wrongly specified. The pre-specified outliers, user-defined variables and pre-specified ARMA coefficients consist of a list of `Predefined` (base model specification) and `Final` values.
- `regarima`: an object of class `c("regarima_spec", "TRAMO_SEATS")`. See *Value* of the function `regarima_spec_tramoseats`.
- `seats`: a data.frame of class `c("seats_spec", "data.frame")`, containing the *seats* variables in line with the names of the arguments variables. The final values can also be accessed with the function `s_seats`.

References

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

See Also

`tramoseats`

Examples

```
myseries <- ipi_c_eu[, "FR"]
myspec1 <- tramoseats_spec(spec = c("RSAfull"))
mysa1 <- tramoseats(myseries, spec = myspec1)

# To modify a pre-specified model specification
myspec2 <- tramoseats_spec(spec = "RSAfull", tradingdays.mauto = "Unused",
                           tradingdays.option = "WorkingDays",
                           easter.type = "Standard",
                           automdl.enabled = FALSE, arima.mu = TRUE)
mysa2 <- tramoseats(myseries, spec = myspec2)

# To modify the model specification of a "SA" object
myspec3 <- tramoseats_spec(mysa1, tradingdays.mauto = "Unused",
                           tradingdays.option = "WorkingDays",
```