JDemetra+ documentation

Anna Smyk Alain Quartier-la-Tente Tanguy Barthelemy
Karsten Webel

2023-09-15

Table of contents

Preface

Welcome to the JDemetra+ on-line documentation.

JDemetra+ is an open-source software for **seasonal adjustment and time series analysis**, developed in the framework of Eurostat's "Centre of Excellence on Statistical Methods and Tools" by the National Bank of Belgium with the support of the Bundesbank and Insee.

To learn more about this project you can visit Eurostat CROS Portal.

To keep up with all JDemetra+ related news head to the **JDemetra+ Universe Blog**

This website is under construction, in the meantime you can fill a large number of the gaps by referring to the previous version of the on-line documentation, coordinated by Sylwia Grudkowska-Kubik(National Bank of Poland).

Eurostat's recommendations on the statistical processes described in this documentation are outlined in:

- Eurostat's Guidelines on seasonal adjustment (2015)
- Eurostat's Guidelines on temporal disaggregation, benchmarking and reconcilation (2018)

Key methodological explanations and state-of-the-art description and references can be found in:

- Handbook on seasonal adjustment (2018)
- Handbook on rapid estimates (2017)

Time Series Software
for Official Statistics

JDemetra+ Software

Introduction

JDemetra+ is an open source software for **seasonal adjustment** and time series analysis. It has been officially recommended by Eurostat to the European Statistical System members since 2015. It is unique in its combination of very fast Java routines, a graphical user interface and a family of R packages. The graphical interface offers a structured and visual feedback, suitable for refined analysis and training. R tools allow the user to mix the capabilities of JDemetra+ with the versatility of the R world, be it for mathematical functions or data wrangling.

Version 2.x and version 3

Version 3.0 to be released by the summer of 2023, fills several critical gaps in the tool box of a time series analyst providing extended features for seasonal adjustment and trend estimation, including **high frequency data** and production tools.

Structure of this book

This book is divided in 3 parts, allowing the user to access the resources from different perspectives.

Algorithms

This part provides a step by step guidance for using of all the algorithms featured in JDemetra+:

- Seasonal Adjustment
- Seasonal Adjustment of high-frequency data
- Reg-Arima modelling
- Outlier detection and external regressors
- Calendar correction
- Benchmarking and temporal disaggregation
- Trend-Cycle estimation

- Nowcasting

Output series, diagnostics, as well as parameters (automatically estimated or user-defined) are detailed in the relevant chapters.

Tools

This part describes the tools allowing to access JDemetra+ algorithms:

- Graphical User Interface GUI
- …enhanced with additional plug-ins
- ..and a Cruncher for mass production
- R packages

Methods

This part describes in greater detail the core algorithms and their underlying statistical methods:

- Reg-Arima modelling
- X-11: moving average based decomposition
- SEATS: Arima model based decomposition
- STL: Loess based decomposition
- Benchmarking and temporal disaggregation
- Spectral analysis tools
- Trend Estimation
- Tests for seasonality and residuals
- Structural time series and state space framework

How JDemetra+ came to be

(under construction)

In this chapter

- history of the project, people who built the software
- evolution of the software

Time line

Part I

Algorithms

This part describes the algorithms featured in JDemetra+:

- Seasonal Adjustment
- Seasonal Adjustment of high-frequency data
- Reg-Arima modelling
- Outlier detection and external regressors
- Calendar correction
- Benchmarking and temporal disaggregation
- Trend-Cycle estimation
- Nowcasting

Seasonal Adjustment

Chapter's overview

The goal of seasonal adjustment is to remove seasonal fluctuations from a time series. Seasonal fluctuations are quasi-periodic infra-annual movements. They can mask evolution of greater interest for the user such as short term evolution or long time trends.

This chapter focuses on the **practical step by step** use of JDemetra+ algorithms, restricted to monthly and quarterly series. For infra-monthly data see the following chapter. The use of graphical user interface and R packages is described simultaneously whenever relevant.

In-depth methodological explanations of the algorithms are covered in separated chapters, in the Methods part.

More information on the steps and best practices of a seasonal adjustment process can be found in the Eurostat guidelines on seasonal adjustment

For an overview on the algorithms and methodological issues, please refer to the Handbook on Seasonal Adjustment

Seasonal Adjustment Algorithms

Table 1: X13-Arima and Tramo-Seats are two-step algorithms with a pre-treatment phase (Reg-Arima or Tramo) and a decomposition phase (X11 and Seats). STL is a local regression (Loess) based decomposition, without pre-treatment. In a Structural Time Series approach pre-treatment and decomposition are done simultaneously in a State Space Framework (SSF).

Algorithm	Access in GUI	Access in R (v2)	Access in R v3
X-13 Arima	yes	RJDemetra	rjd3x13
Reg-Arima only	yes	RJDemetra	rjd3x13
X11 decomposition only	yes	RJDemetra	rjd3x13
Tramo-Seats	yes	RJDemetra	rjd3tramoseats
Tramo only	yes	RJDemetra	rjd3tramoseats
STL	no	no	rjd3stl

Algorithm	Access in GUI	Access in R (v2)	Access in R v3
STS	no	no	rjd3sts

Decomposition in unobserved components

To seasonally adjust a series, seasonal factors S_t will be estimated and removed from the original raw series: $Y_{sa} = Y_t/S_t$ or $Y_{sa} = Y_t - S_t$. To do so the series is first decomposed into unobservable components. Two decomposition models:

- The additive model: $X_t = T_t + S_t + I_t$;
- The multiplicative model: $X_t = T_t \times S_t \times I_t$.

The main components, each representing the impact of certain types of phenomena on the time series (X_t), are:

- The trend (T_t) that captures long-term and medium-term behaviour;
- The seasonal component (S_t) representing intra-year fluctuations, monthly or quarterly, that are repeated more or less regularly year after year;
- The irregular component (I_t) combining all the other more or less erratic fluctuations not covered by the previous components.

In general, the trend consists of 2 sub-components:

- The long-term evolution of the series;
- The cycle, that represents the smooth, almost periodic movement around the long-term evolution of the series. It reveals a succession of phases of growth and recession. Trend and cycle are not separated in SA algorithms.

Pre-treatment principles

The goal of this step is to remove deterministic effects (calendar and outliers) in order to improve the decomposition.

$$Y_t = \sum \alpha_i O_{it} + \sum \beta_j C_{jt} + \sum \gamma_i U_{it} + Y_{lin,t}$$

- O_{it} are the i final outliers (AO, LS, TC)
- C_{it} are the calendar regressors (automatic or user-defined) (link to calendar chap)
- U_{it} are all the other user-defined regressors (link to outliers and regressors chap)
- $Y_{lin,t} \sim ARIMA(p,d,q)(P,D,Q)$

Detecting seasonal patterns

A large number of seasonality tests are available in JDemetra+. They can be accessed in the graphical user interface or via R.

In rjd3toolkit package:

- Canova-Hansen (`seasonality.canovahansen()`)
- X-12 combined test (`seasonality.combined()`)
- F-test on seasonal dummies (`seasonality.f()`)
- Friedman Seasonality Test (`seasonality.friedman()`)
- Kruskall-Wallis Seasonality Test (`seasonality.kruskalwallis()`)
- Periodogram Seasonality Test (`seasonality.periodogram()`)
- QS Seasonality Test (`seasonality.qs()`)

Pre-treatment: Reg-Arima or Tramo

The following sections cover how to perform pre-treatment with Reg-ARIMA (or Tramo) algorithms. Tramo and the Reg-Arima part of X13-Arima rely on very similar principles: Reg-Arima modelling. Thus Tramo will be mentioned only to highlight differences with the Reg-Arima part of X13-Arima. Reg-Arima modelling part can be of a seasonal adjustment process or run on its own, we focus on performing pre-treatment as part of a SA processing.

Default Specifications

Default specifications are set for the whole SA procedure, pre-treatment and decomposition. They are slightly different for X13-ARIMA and Tramo-Seats and can be modified with user defined parameters.

Starting point for X13-ARIMA

Spec identifier	Log/level detection	Outliers detection	Calendar effects	ARIMA
RSA0	*NA*	*NA*	*NA*	Airline(+mean)
RSA1	automatic	AO/LS/TC	*NA*	Airline(+mean)
RSA2c	automatic	AO/LS/TC	2 TD vars+Easter	Airline(+mean)
RSA3	automatic	AO/LS/TC	*NA*	automatic
RSA4c	automatic	AO/LS/TC	2 TD vars+Easter	automatic

Spec identifier	Log/level detection	Outliers detection	Calendar effects	ARIMA
RSA5	automatic	AO/LS/TC	7 TD vars+Easter	automatic
X-11	*NA*	*NA*	*NA*	*NA*

explanations:

- *NA:* non applied, for example in RSA3 there is no calendar effect correction
- automatic: test is performed

outliers detection: AO/LS/TC type of outliers automatically detected under a critical T-Stat value (default value=4)

calendar:

- 2 regressors: weekdays vs week-ends + LY
- 7 regressors: each week day vs Sundays + LY
- always tested
- easter tested (default length = 6 days in Tramo, 8 days in X13-Arima)

Starting point for Tramo-Seats

Spec 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
RSA5	automatic	AO/LS/TC	6 TD vars+Easter	automatic
RSAfull	automatic	AO/LS/TC	automatic	automatic

Principle of user setting parameters: can be done from one of the default specifications or any specification in a "save" as" mode very similar in GUI and R, see below.

Spans

Estimation span

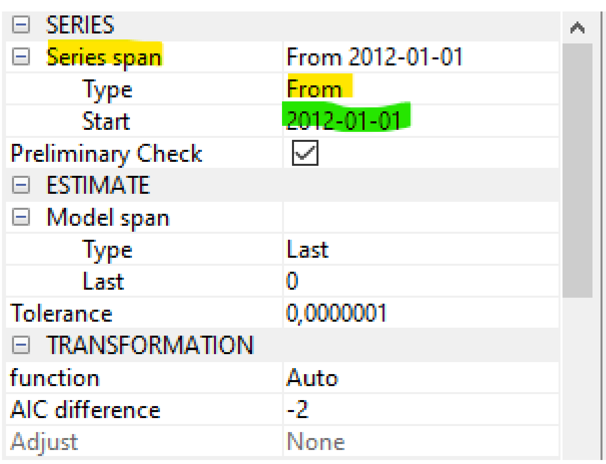
Specifies the span (data interval) of the time series to be used in the seasonal adjustment process. The user can restrict the span

Common settings

Option	Description (expected format)
All	default
From	first observation included (yyyy-mm-dd)
To	last observation included (yyyy-mm-dd)
Between	interval [from ; to] included (yyyy-mm-dd to yyyy-mm-dd)
First	number of obs from the beginning of the series included (dynamic) (integer)
Last	number of obs from the end of the series (dynamic)(integer)
Excluding	excluding N first obs and P last obs from the computation,dynamic) (integer)
Preliminary check	check to exclude highly problematic series e.g. the series with a number of identical observations and/or missing values above pre-specified threshold values. (True/False)

Setting series span in GUI

Use the specification window for a given series and expand the nodes.



Setting series span in R

x13 in version 2

```
library("RJDemetra")
# estimation interval: option with static dates
user_spec_1<-x13_spec(spec = c("RSA5c", "RSA0", "RSA1", "RSA2c",
                              "RSA3", "RSA4c", "X11"),
preliminary.check = TRUE,
estimate.from = "2012-06-01",
estimate.to = "2019-12-01")

# estimation interval: option with dynamic numbers of observations

#
# spec can be applied on different series and therefore exclude different dates
user_spec_2<-x13_spec(spec = c("RSA5c", "RSA0", "RSA1", "RSA2c", "RSA3", "RSA4c", "X11"),
estimate.first = 12)

# eestimation on the last 120 obs
user_spec_3<-x13_spec(spec = c("RSA5c", "RSA0", "RSA1", "RSA2c", "RSA3", "RSA4c", "X11"),
estimate.last = 120)

#excluding first 24 and last 36 observations
user_spec_4<-x13_spec(spec = c("RSA5c", "RSA0", "RSA1", "RSA2c", "RSA3", "RSA4c", "X11"),
estimate.exclFirst = 24,
estimate.exclLast = 36)

# Retrieve settings
```

For comprehensive details about x13_spec function see RJDemetra R help pages.

Tramo-Seats in version 2

```
#excluding first 24 and last 36 observations
user_spec_1<-tramoseats_spec( spec = c("RSAfull", "RSA0", "RSA1", "RSA2", "RSA3", "RSA4",
estimate.exclFirst = 24,
estimate.exclLast = 36)
```

For comprehensive details about tramoseats_spec function see RJDemetra R help pages.

Setting model span

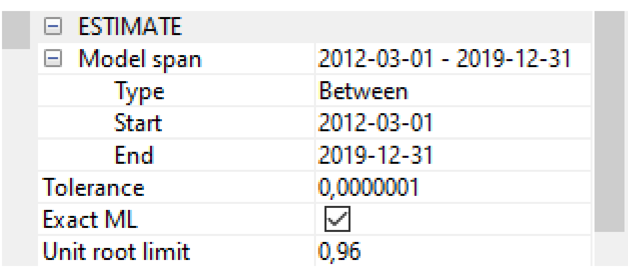
The user can also specify the span (data interval) of the time series to be used for the estimation of the Reg-ARIMA model coefficients. It allows to impede a chosen part of the data from influencing the regression estimates. Setting works the same way as setting series (estimation) span described above.

Additional (vs series span setting) parameters are described below:

Tolerance	Convergence tolerance for the non-linear estimation. The absolute changes in the log-likelihood are compared to Tolerance to check the convergence of the estimation iterations. The default setting is 0.0000001.
Tramo specific parameters	
Exact ML	When this option is marked, an exact maximum likelihood estimation is performed. Alternatively, the Unconditional Least Squares method is used. However, in the current version of JDemetra+ it is not recommended to change this parameter's value
Unit Root Limit	Limit for the autoregressive roots. If the inverse of a real root of the autoregressive polynomial of the ARIMA model is higher than this limit, the root is set equal to 1. The default parameter value is 0.96.

Setting model span in GUI:

Use the specification window



Setting in R

Tramo example in version 2

```
#excluding first 24 and last 36 observations
user_spec_1<-tramoseats_spec( spec = c("RSAfull", "RSA0", "RSA1", "RSA2", "RSA3", "RSA4",
estimate.tol = 0.0000001,
estimate.eml = FALSE,
estimate.urfinal = 0.98)
```

Decomposition Scheme

Parameters

Transformation test: a test is performed to choose between an additive decomposition (no transformation) (link to reg A chap to detail this)

Settings

Function

transform {function=}

Transformation of data. 2 The user can choose between:

None – no transformation of the data;

Log – takes logs of the data;

Auto – the program tests for the log-level specification. This option is recommended for automatic modelling of many series.

The default setting is Auto.

Reg-Arima specific settings

AIC difference

transform {aicdiff=}

Defines the difference in AICC needed to accept no transformation over a log transformation when the automatic transformation

selection option is invoked. The option is disabled when Function is not set to Auto. The default AIC difference value is -2.

Adjust

transform {adjust=}

Options for proportional adjustment for the leap year effect. The option is available when Function is set to Log. Adjust can be set to:

LeapYear – performs a leap year adjustment of monthly or quarterly data;

LengthofPeriod – performs a length-of-month adjustment on monthly data or length-of-quarter adjustment on quarterly data;

None – does not include a correction for the length of the period.

The default setting is None

Tramo specific settings

Fct
Transformation; fct

Controls the bias in the log/level pre-test (the function is active when **Function** is set to *Auto*); **Fct** >1 favours levels, **Fct** <1 favors logs. The default setting is 0.95.

Set in GUI

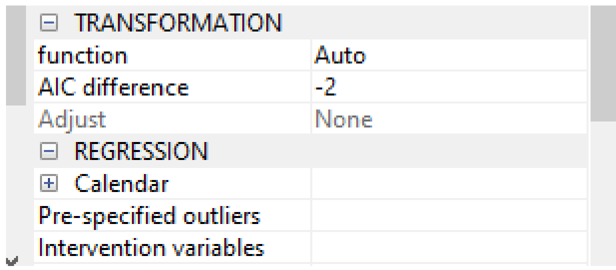


Figure 1: Model span setting

Set and in R

X13

```
#excluding first 24 and last 36 observations
user_spec <-x13_spec(spec = c("RSA5c", "RSA0", "RSA1", "RSA2c", "RSA3", "RSA4c", "X11"),
transform.function ="Log", # choose from: c(NA, "Auto", "None", "Log"),
transform.adjust = "LeapYear", #c(NA, "None", "LeapYear", "LengthOfPeriod"),
transform.aicdiff = -3)
#Retrieve settings: to complete*
```

Tramo-Seats settings

```
#transfo
user_spec_1<-tramoseats_spec( spec = c("RSAfull", "RSA0", "RSA1", "RSA2", "RSA3", "RSA4",
transform.function = "Auto", #c(NA, "Auto", "None", "Log"),
transform.fct = 0.5)
# Retrieve settings: to complete
```

Calendar correction

Some calendar correction options included in the starting specifications for X13-Arima or Tramo-Seats, they can be fine-tuned by modifying specifications. The following section lists all the available options, illustrates how to set them in GUI or R and shows have to retrieve used parameters, regressors as well as results.

JDemetra+ offers two default options for calendar correction working days regressors and trading days regressors, with Leap-year effect if needed. Those options don't take into account national calendars (link) and their specific holidays. There are two ways to change this:

- user-defined regressors (link)
- customized calendars (link)

Overview: what you can do

Need 1: correct for working days, trading days (+ easter) not taking national calendars

Need 2: taking national calendar into account Solutions

- add a work of means of allocating regressors to the calendar component

Available Options

0.0.0.0.1 * Trading Days

"Trading Days" has two meanings: general calendar correction process (here without easter effect) and one of the options of this correction (see below)

```
-    "None": no correction for trading days and working days effects

-    "Default": JDemetra + built regressors (xorking days or trading days)

-    "Holidays": same as above but taking into accoutn a national calendar,

-     "UserDefined": user-defined trading days regressors (see below)

-    (if NONE) indicating the day of the month when inventories and other stock are reported
```

0.0.0.0.2 * Leap Year effect

Autoadjust

If enabled, the program corrects automatically for the leap year effect.. When is the option available Modifications of this variable are taken into account only when transform.function is set to "Auto".

Leapyear

to specify whether or not to include the leap-year effect in the model: - "LeapYear": leap year effect; - "LengthOfPeriod": length of period, - "None" = no effect included.

The leap-year effect can be pre-specified in the model only if the input series hasn't been pre-adjusted (transform.adjust set to "None") and if the automatic correction for the leap-year effect isn't selected (tradingdays.autoadjust set to FALSE).

Test

Test: defines the pre-tests for the significance of the trading day regression variables based on the AICC statistics: "Add" = the trading day variables are not included in the initial regression model but can be added to the RegARIMA model after the test; "Remove" = the trading day variables belong to the initial regression model but can be removed from the RegARIMA model after the test; "None" = the trading day variables are not pre-tested and are included in the model.

0.0.0.0.1 * Easter

Easter.enabled a logical. If TRUE, the program considers the Easter effect in the model.

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

easter.duration a numeric indicating the duration of the Easter effect (length in days, between 1 and 20).

easter.test defines the pre-tests for the significance of the Easter effect based on the t-statistic (the Easter effect is considered as significant if the t-statistic is greater than 1.96): "Add" = the Easter effect variable is not included in the initial regression model but can be added to the RegARIMA model after the test; "Remove" = the Easter effect variable belongs to the initial regression model but can be removed from the RegARIMA model after the test; "None" = the Easter effect variable is not pre-tested and is included in the model.

A user-defined regressor can also be used, see chapter on calendar correction

(to be added: additional options in Tramo)

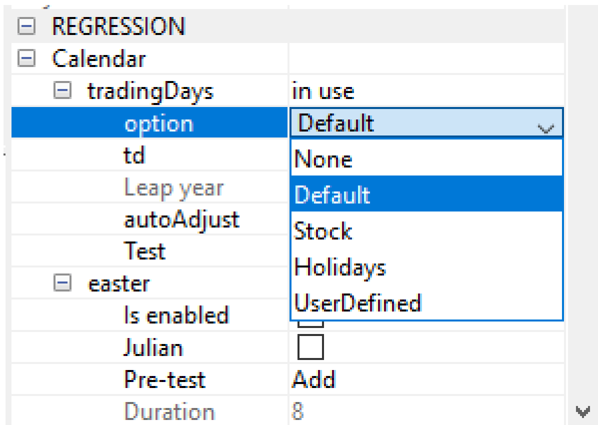
Setting Calendar correction in GUI

0.0.0.0.1 * Using default options (without national calendars)

In GUI Use the specification window

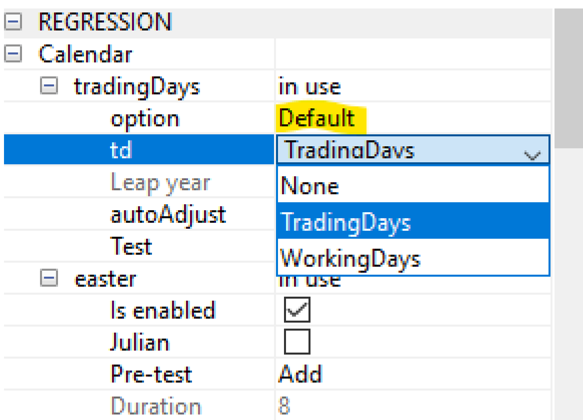
Calendar effects

STEP 1: Selection from JDemetra+ Default…or User_defined



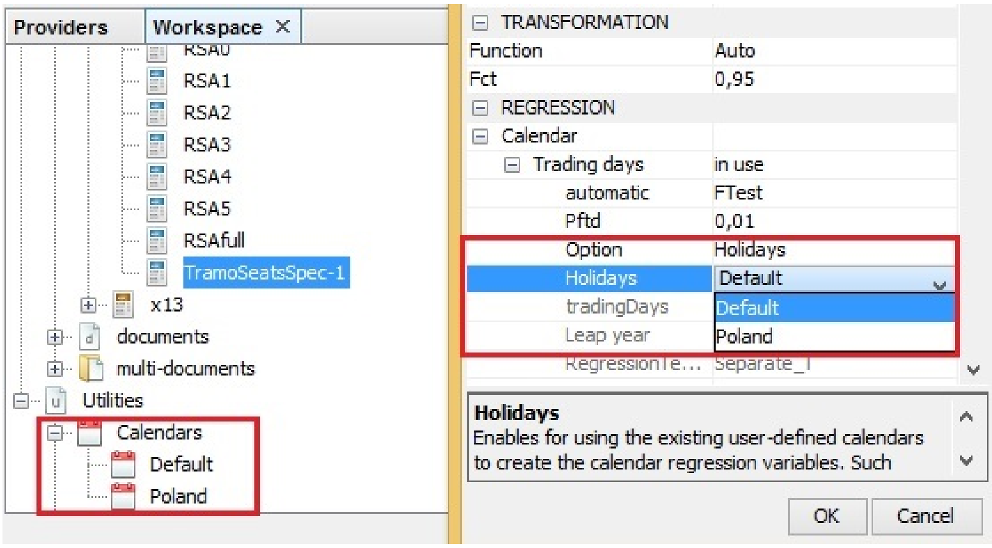
SA_REGA_calendar_options.png

STEP2: Calendar effects minus Easter are labeled **trading days**



0.0.0.0.2 * Holidays option

using a customized calendar just show how to fetch it building process in calendar chapter



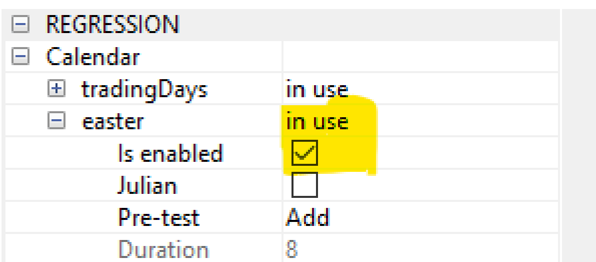
The list of calendars displayed under *Holidays* option corresponds to the calendars defined in the *Workspace* window

Missing: stock td option, length-of-period

User-defined regressors: adding see below

Link to Import data Once data imported: here explain how to link variables

0.0.0.0.3 * Easter



Setting Calendar correction in R

In version 2

```
# Parameter choice NA=...
  tradingdays.option = c(NA, "TradingDays", "WorkingDays", "UserDefined", "None"),
  tradingdays.autoadjust = NA,
  tradingdays.leapyear = c(NA, "LeapYear", "LengthOfPeriod", "None"),
```

```
    tradingdays.stocktd = NA_integer_,
    tradingdays.test = c(NA, "Remove", "Add", "None"),
    easter.enabled = NA,
    easter.julian = NA,
    easter.duration = NA_integer_,
    easter.test = c(NA, "Add", "Remove", "None"),
  # example
```

In version 3 (Under construction)

User defined regressors

If **User Defined** options is used for trading days, regressors have to be provided by the user.

Building Regressors The underlying methodology and implementation in JDemetra+ to build these regression variables are provided here

0.0.0.0.1 * Adding Regressors in GUI

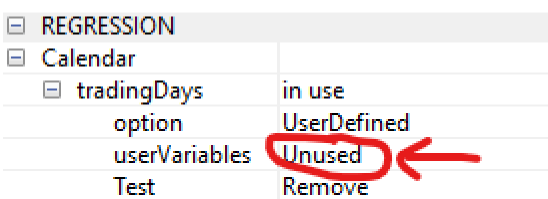
Step 1: import data set containing the regressors, general procedure explained here

Step 2: Link the regressors to the workspace, procedure detailed here

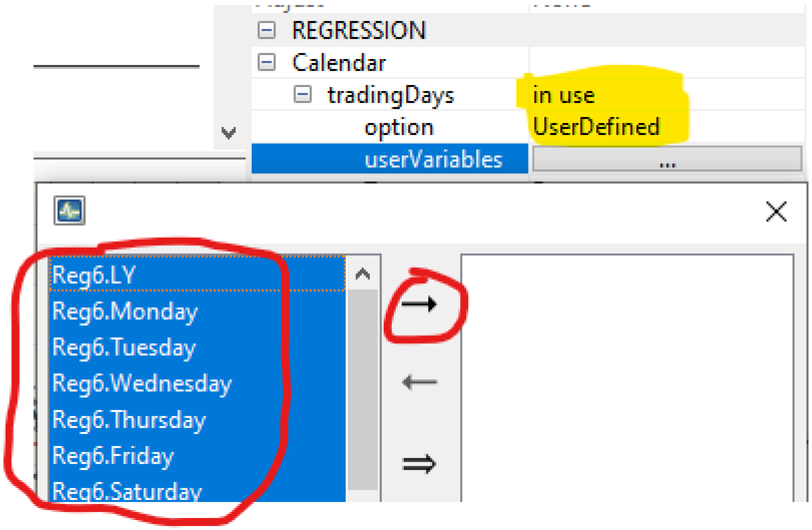
Step 3: Modify specifications Modifications are done the same way in a global specification (whole SAP) or series by series.

- select trading days **User-defined option** and select variables

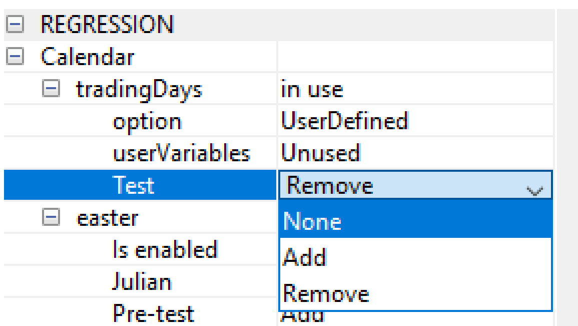
In the specification window, click right from "userVariables" on "Unused" to open the variable selection window



Move right the chosen regressors



- set TEST option (expl)



0.0.0.0.2 * Adding Regressors in R

"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).

```
# example
spec_4 <- x13_spec(spec = spec_1,
                   tradingdays.option = "UserDefined",
                   tradingdays.test = "None",
                   usrdef.varEnabled = TRUE,
                   usrdef.varType = "Calendar",
                   usrdef.var = reg3) # set of regressors in TS format
```

Retrieving Results

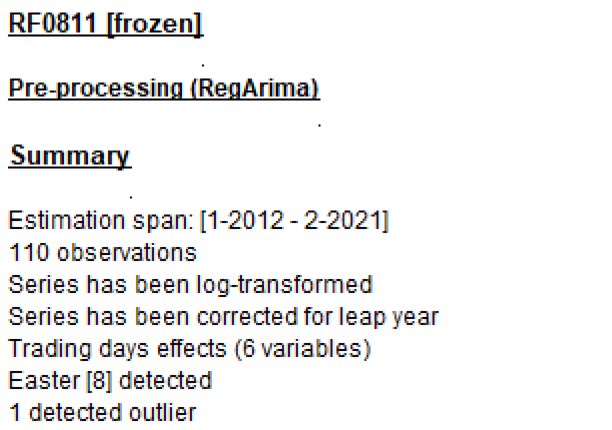
The following section details how to retrieve results (parameters, regressors, regression coefficients and tests) when using GUI or R interface.

0.0.0.0.1 * Parameters

Parameters are regressors used in fine. If non test options, parameters are known If test options are selected by the algorithm.

In GUI

Automatically chosen or user-defined calendar options (as well as other pre-adjustment options) are displayed at the top of the MAIN Results NODE displayed by clicking on a given series name in the SAProcessing panel.


RF0811 [frozen]

Pre-processing (RegArima)

Summary

Estimation span: [1-2012 - 2-2021]
110 observations
Series has been log-transformed
Series has been corrected for leap year
Trading days effects (6 variables)
Easter [8] detected
1 detected outlier

In R

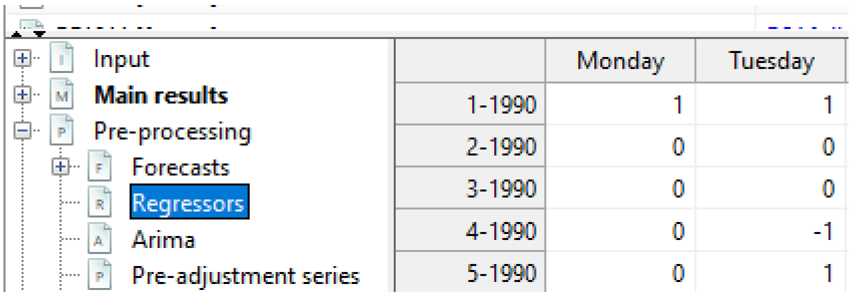
(to be added)

version 2: RJDemetra

version 3: rjd3x13 or rjd3tramoseats

0.0.0.0.2 * Regressors

In GUI All regressors in the pre-adjustment phase (calendar, outliers, external) are displayed in the pre-processing-regressors node.



In R

(to be added)

Version 2

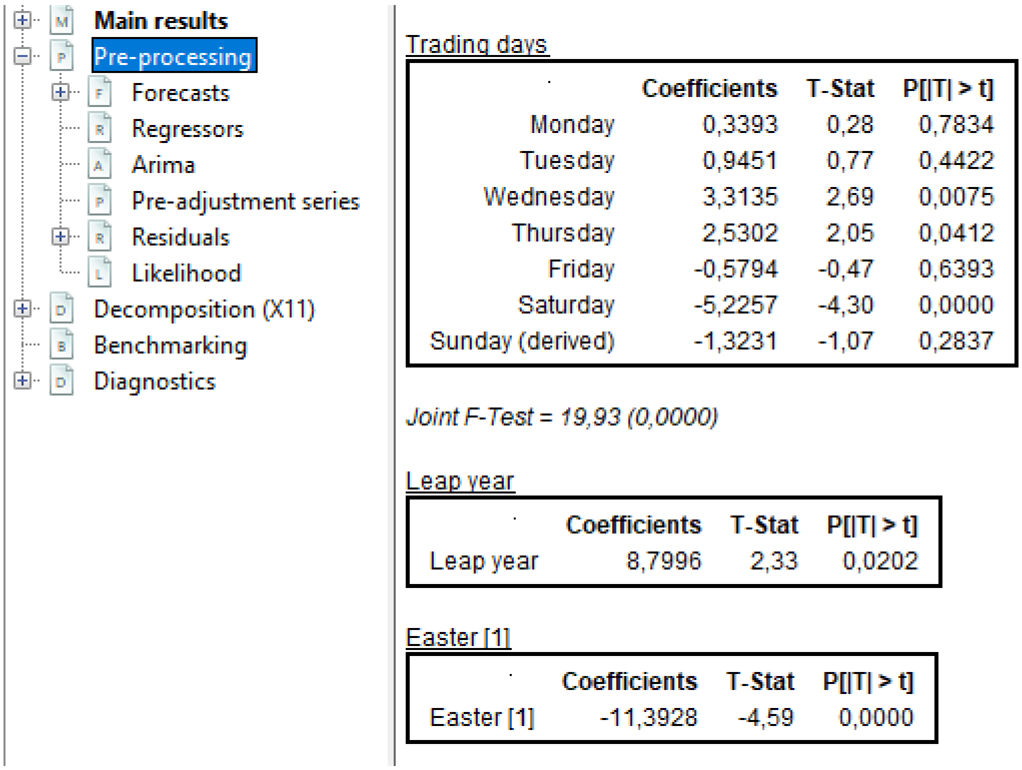
version 3

0.0.0.0.3 * Regression results

Regressions results

In GUI

The results of the whole Reg-Arima regression (link to last section) including calendar effects (below) are displayed in the pre-processing panel.



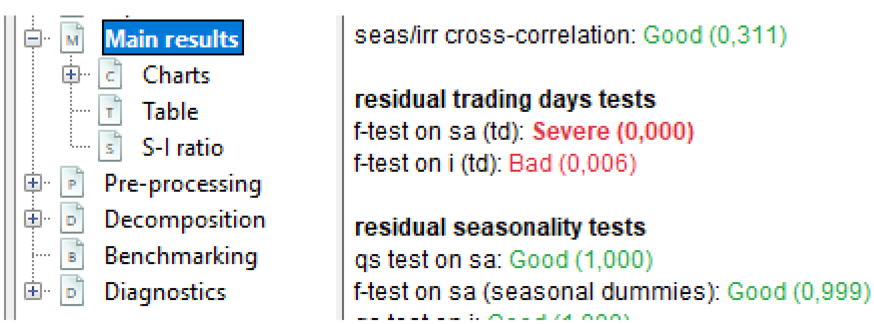
In R

0.0.0.0.4 * Test for residual trading-days effects

Residual calendar effects are tested with A F-Test 7 regressors and no national calendar, on sa final series and on irregular component (link to calendar chapter for test details)

In GUI

F-Test results are displayed at the bottom of **Main Results** NODE in the SAProcessing panel



In R

Customizing Calendars

The following describes how to take a national calendar into account.

Solution 1: if working with GUI build a new calendar in GUI (here

(to be added: GUI: how to use it or customize HTML file structure explanation)

set this option in GUI

(to be added: image: spec window calendar / holidays / choice of calendars)

set this option in R (to be added) version 2:

version 3:

solution 2: import external regressors, which can be built with rjd3toolkit (link) which can then be used in via are or imported via GUI

set this option in GUI how to import variables into JD+ / set utility (in interface chapter) classical user defined

set this option in R version 2:

version 3

Once the calendar regressors are set, the RegArima (tramo) model will be estimated globally with all the other regression variables and taking into account Arima model specificities as well. That is why diagnostics are all jointly displayed at the end of the process. (link)

(to be added: worked example: french calendar in R)

Outliers

The sections below focus on

- outlier detection parameters (type and critical value)
- pre-specifying outliers in a seasonal adjustement (reg-arima modelling) process

Additional information can be found in this chapter.

Options for automatic detection

*Is enabled** *outliers; iatip*

```
Enables/disables the automatic detection of outliers in the span determined by the **Detec
```

- **Use default critical value** *outliers; va*

  The critical value is automatically determined by the number of observations in the interval specified by the **Detection span** option. When **Use default critical value** is disabled, the procedure uses the critical value inputted in the **Critical value** item (see below). Otherwise, the default value is used (the first case corresponds to "*critical = xxx*"; the second corresponds to a specification without the critical argument). It should be noted that it is not possible to define a separate critical value for each outlier type. By default, the checkbox is marked, which implies that the automatic determination of the critical value is enabled.

- **Critical value** *outliers; va*

  The critical value used in the outlier detection procedure. The option is active once **Use default critical value** is disabled. By default, it is set to 3.5.

- **Detection span** $→$ **type** *outliers; int1, int2**

  A span of the time series to be searched for outliers. The possible values of the parameter are:

  - *All* – full time series span is considered in the modelling;
  - *From* – date of the first time series observation included in the pre-processing model;
  - *To* – date of the last time series observation included in the pre-processing model;
  - *Between* – date of the first and the last time series observations included in the pre-processing model;
  - *Last* – number of observations from the end of the time series included in the pre-processing model;
  - *First* – number of observations from the beginning of the time series included in the pre-processing model;
  - *Excluding* – number of observations excluded from the beginning (specified in the *first* field) and/or end of the time series (specified in the *last* field) of the pre-processing model.

  With the options *Last*, *First*, *Excluding* the span can be computed dynamically on the series. The default setting is *All*.

- **Additive** *outliers; aio**

  Automatic identification of additive outliers. By default, this option is enabled.

- **Level shift** *outliers; aio**

  Automatic identification of level shifts. By default, this option is enabled.

- **Transitory change** *outliers; aio**

  Automatic identification of transitory changes. By default, this option is enabled.

- **Seasonal outlier** *outliers; aio**

  Automatic identification of seasonal outliers. By default, this option is disabled. Tramo specific

- **EML estimation** *outliers; imvx*

  The estimation method used in the automatic model identification procedure. By default, the fast method of Hannan-Rissanen is used for parameter estimation in the intermediate steps of the automatic detection and correction of outliers. When the checkbox is marked the exact maximum likelihood estimation method is used.

- **TC rate***outliers; deltatc*

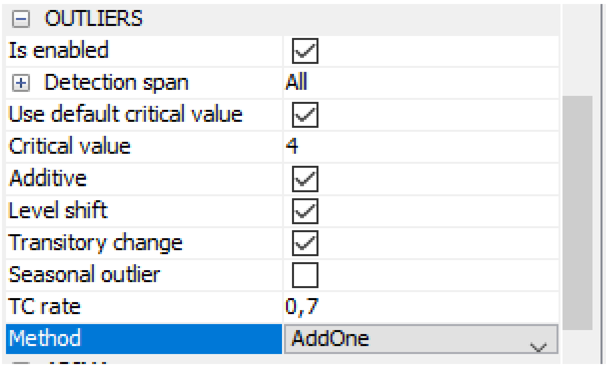
  The rate of decay for the transitory change outlier. It takes values between 0 and 1. The default value is 0.7.

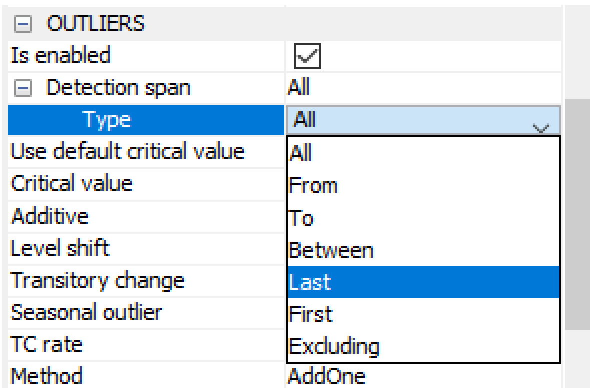
Options for pre-specified outliers

User-defined outliers are used when prior knowledge suggests that certain effects exist at known time points[^14]. Four pre-defined outlier types, which are simple forms of intervention variables, are implemented: * Additive Outlier (AO); * Level shift (LS); * Temporary change[^15] (TC); * Seasonal outliers (SO).

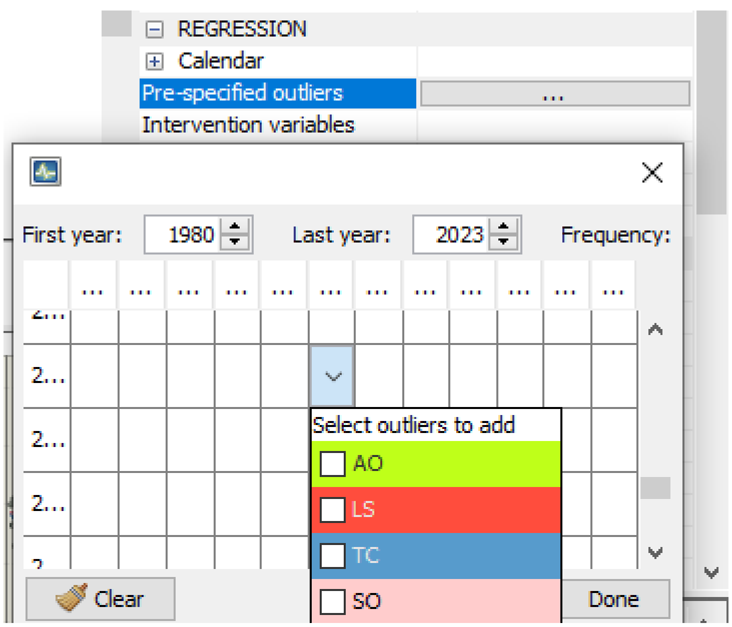
Setting in GUI

Automatic detection





Pre-specified



Setting in R

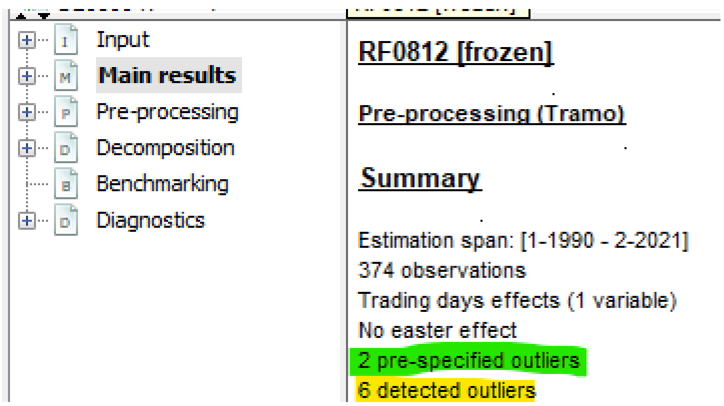
(to be added)

Retrieving results

0.0.0.0.1 * Parameters

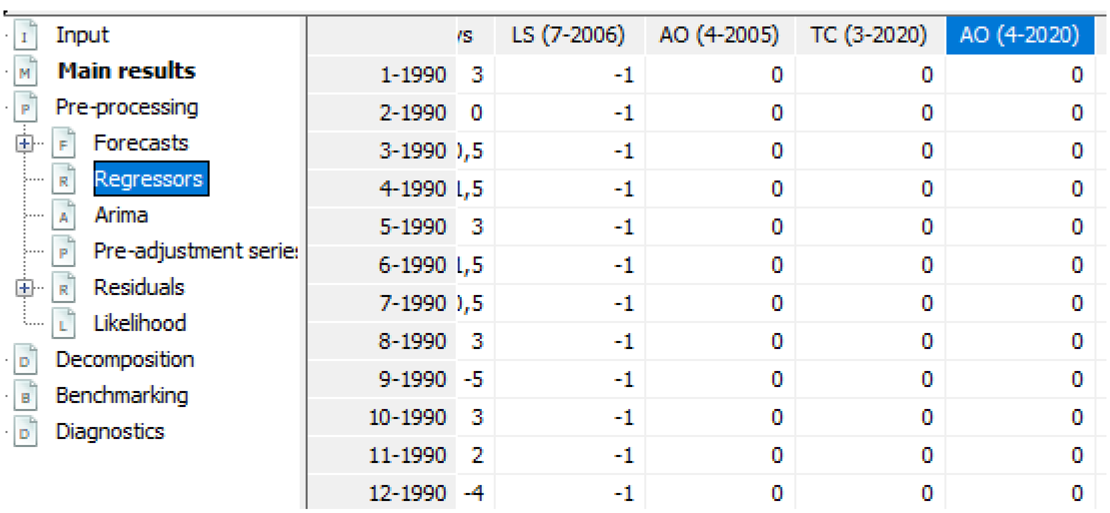
In GUI

In main results NODE (same info at top of pre-processing NODE)



0.0.0.0.2 * Regressors

In GUI



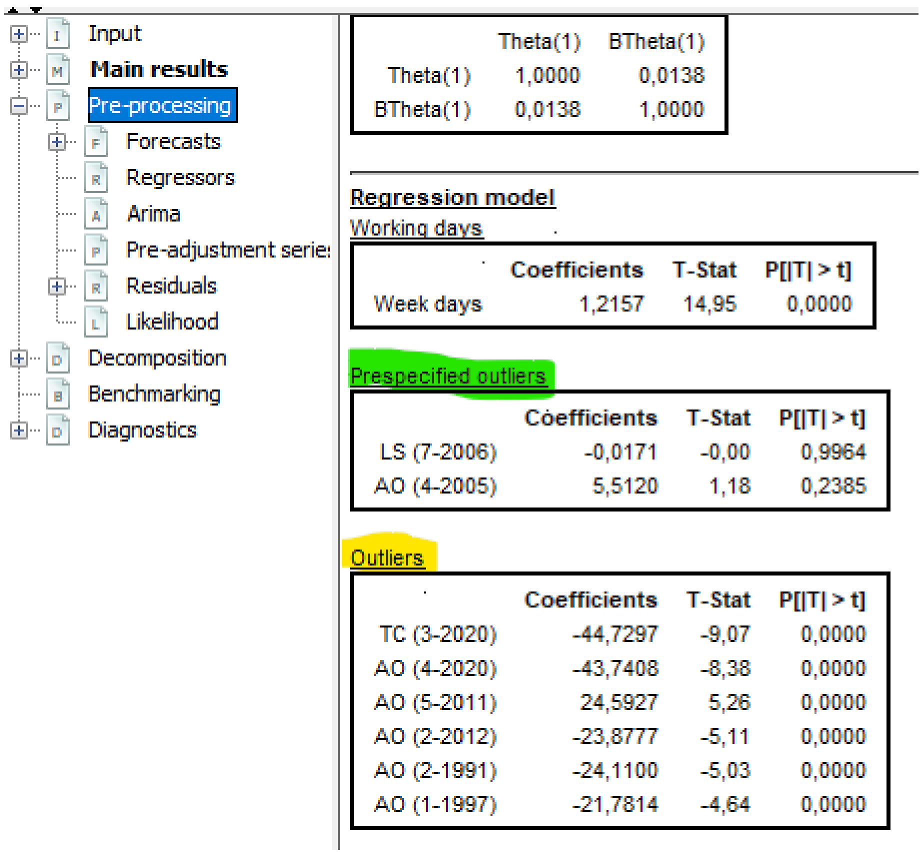
	/s	LS (7-2006)	AO (4-2005)	TC (3-2020)	AO (4-2020)
1-1990	3	-1	0	0	0
2-1990	0	-1	0	0	0
3-1990	),5	-1	0	0	0
4-1990	l,5	-1	0	0	0
5-1990	3	-1	0	0	0
6-1990	l,5	-1	0	0	0
7-1990	),5	-1	0	0	0
8-1990	3	-1	0	0	0
9-1990	-5	-1	0	0	0
10-1990	3	-1	0	0	0
11-1990	2	-1	0	0	0
12-1990	-4	-1	0	0	0

In R

(to be added)

0.0.0.0.3 * Regression details

In GUI



In R

(to be added)

User-defined regressors

(to be added)

- rationale
- parameters: assign to a component

Pre-treatment regression with additional outliers

$$Y_t = \sum \hat{\alpha}_i O_{it} + \sum \hat{\beta}_j C_{jt} + \sum \hat{\gamma}_k Reg_{kt} + y_{lin_t}$$

0.0.0.0.1 * Allocation to components

$reg = reg_i + reg_t + reg_s + ...$ The user-defined regression variable associated to a specific component should not contain effects that have to be associated with another component. Therefore, the following rules should be observed: * The variable assigned to the trend or to the seasonally adjusted series should not contain a seasonal pattern; * The variable assigned to the seasonal should not contain a trend (or level); * The variable assigned to the irregular should contain neither a seasonal pattern nor a trend (or level). - no external regressors can be assigned to calendar component. It has to be be done via user defined calendar regressors specific part (link)

Ramps and intervention variables are Specific cases of external regressors

Setting in GUI

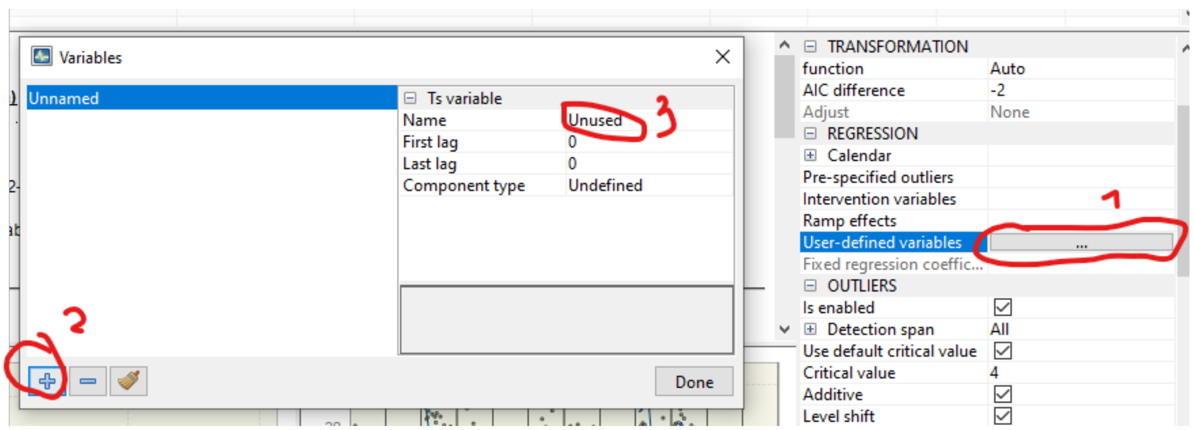
User-defined variables

Step 1: import data set containing the regressors, general procedure explained here

Step 2: Link the regressors to the workspace, procedure detailed here

Step 3: Modify specifications via window

Modifications are done the same way in a global specification (whole SAP) or series by series.



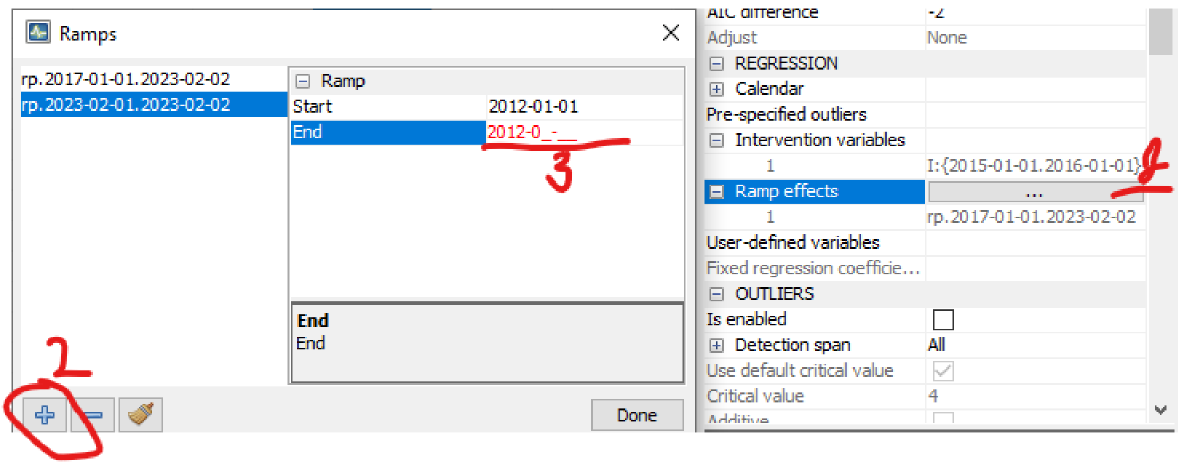
Setting in R

(to be added)

Special case 1: Ramp effects

A ramp effect means a linear increase or decrease in the level of the series over a specified time interval t_0 to t_1. All dates of the ramps must occur within the time series span. (tested: not true). Ramps can overlap other ramps, additive outliers and level shifts.

0.0.0.0.0.1 * Creation in GUI



0.0.0.0.0.2 * Allocation to components

allocation when intervention or ramps ? in test allocated to trend ? (reg)

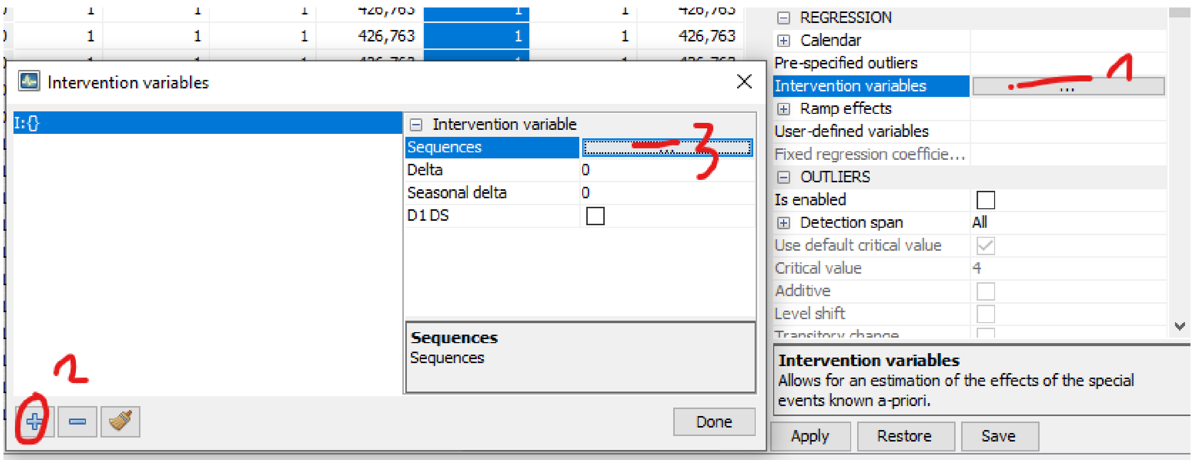
impossible (?) to create several intervention variables

Special case 2: Intervention variables

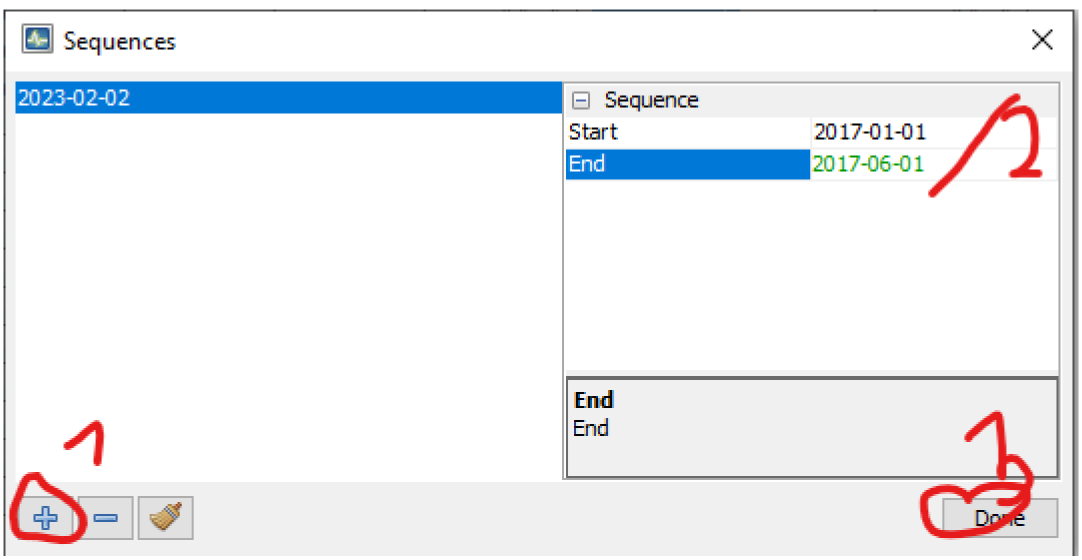
Intervention variables are modeled as any possible sequence of ones and zeros, on which some operators may be applied. They are built as combinations of the following basic structures: * Dummy variables[^17]; * Any possible sequence of ones and zeros; * $\frac{1}{(1-\delta B)}$, * $(0 < \delta \leq 1)$; * $\frac{1}{(1-\delta_s B^s)}$, * $(0 < \delta_s \leq 1)$; * $\frac{1}{(1-B)(1-B^s)}$; where B is backshift operator (i.e. $B^k X_t = X_{t-k}$) and s is frequency of the time series ($s = 12$for a monthly time series, $s = 4$for a quarterly time series).

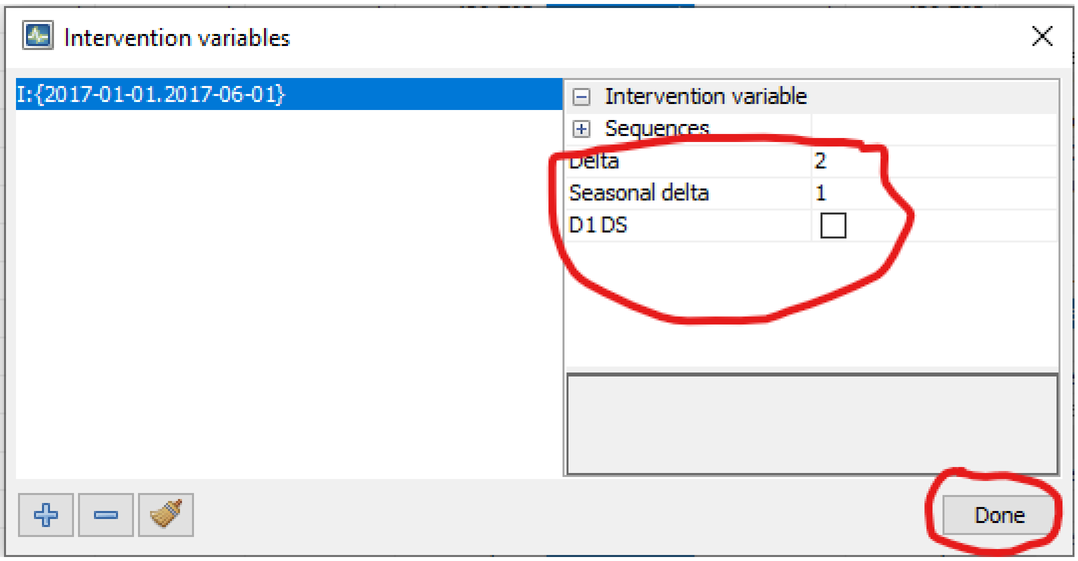
These basic structures enable the generation of not only AO, LS, TC, SO and RP outliers but also sophisticated intervention variables that are well-adjusted to the particular case.

0.0.0.0.0.1 * Creation in GUI

step 1 

Step 2:



step 3: 

0.0.0.0.0.2 * Creation in R

(to be added)

0.0.0.0.0.3 * Allocation to components

allocation (to be added)

fixed coefficient options

- **Fixed regression coefficients** *regression variables;* –

  For the pre-specified regression variables this option specifies the parameter estimates that will be held fixed at the values provided by the user. To fix a coefficient the user should undertake the following actions:

  - Choose the transformation (log or none).
  - Define some regression variables in the *Regressors* specification.
  - Push on the fixed regression coefficients editor button in the **User-defined variables** row.
  - Select the regression variable from the list for which the coefficient will be fixed.
  - Save the new setting with the **Done** button.

Overview: differences GUI set up vs R set up

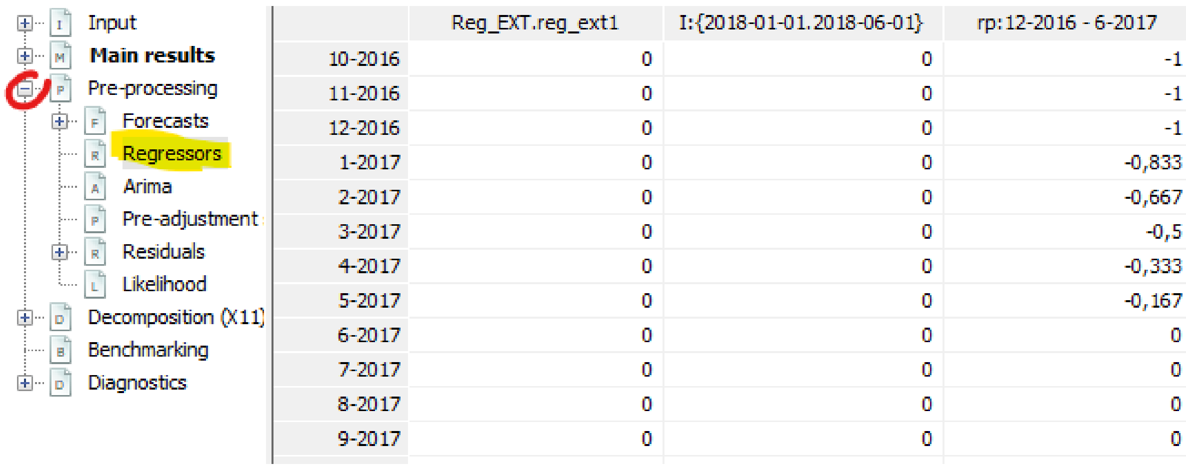
Retrieving Results

For all types of external regressors: user-defined, ramps or intervention variables.

0.0.0.0.1 * Regressors

In GUI

To retrieve regressors that were actually used, expand pre-processing NODE and click on Regressors pane.



	Reg_EXT.reg_ext1	I:{2018-01-01.2018-06-01}	rp:12-2016 - 6-2017
10-2016	0	0	-1
11-2016	0	0	-1
12-2016	0	0	-1
1-2017	0	0	-0,833
2-2017	0	0	-0,667
3-2017	0	0	-0,5
4-2017	0	0	-0,333
5-2017	0	0	-0,167
6-2017	0	0	0
7-2017	0	0	0
8-2017	0	0	0
9-2017	0	0	0

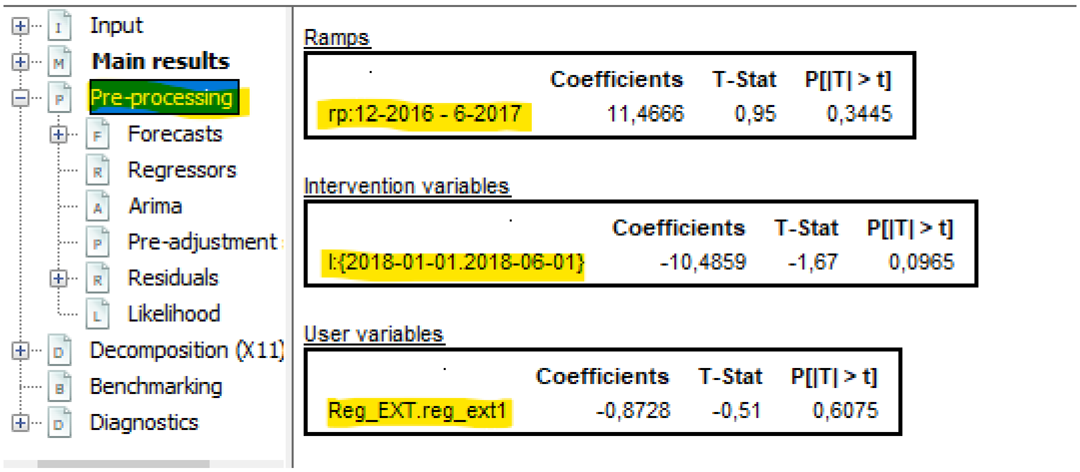
In R

(to be added)

0.0.0.0.2 * Regression details

In GUI

Regression details are in the pre-processing pane.



IN R

Arima Model

Key specifications on Arima modelling are embedded in default specifications: airline (default model) or full automatic research.(links)

Two kinds of interventions are available to the user

- modify automatic detection parameters
- set a user defined Arima model

In both cases forecast horizon can also be set (link)

Options for modifying automatic detection

automdl.enabled If TRUE, the automatic modelling of the ARIMA model is enabled. (If FALSE, the parameters of the ARIMA model can be specified, see below)

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

automdl.acceptdefault a 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 the cancellation limit (numeric). 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 cancel out.

automdl.ub1 the first unit root limit (numeric). 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 the first unit root limit in modulus, it is set equal to unity.

automdl.ub2 the second unit root limit (numeric). 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 cancelled (see automdl.cancel). If there is no cancellation, the AR root is set equal to unity (i.e. the differencing order changes).

automdl.mixed a logical. This variable controls whether ARIMA models with non-seasonal AR and MA terms or seasonal AR and MA terms will be considered in the automatic model identification procedure. If FALSE, a model with AR and MA terms in both the seasonal and non-seasonal parts of the model can be acceptable, provided there are no AR or MA terms in either the seasonal or non-seasonal terms.

automdl.balanced a logical. If TRUE, the automatic model identification procedure will have a preference for balanced models (i.e. models for which the order of the combined AR and differencing operator is equal to the order of the combined MA operator).

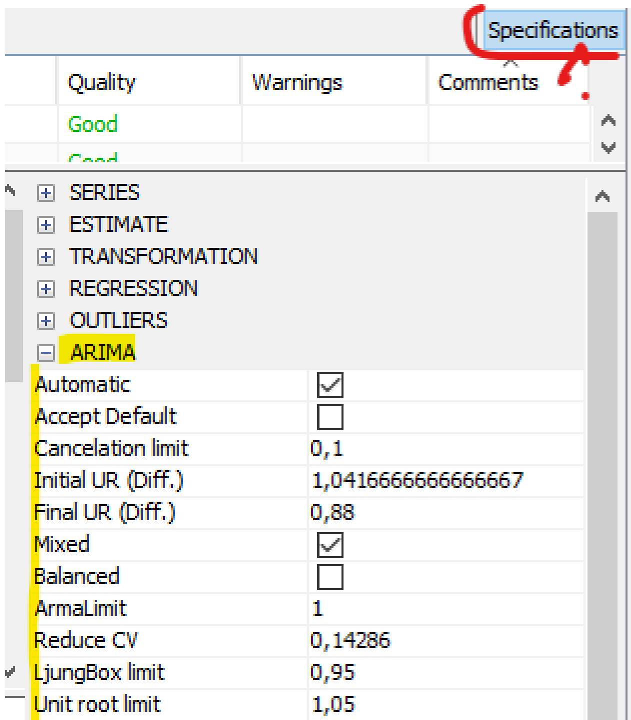
automdl.armalimit the ARMA limit (numeric). It is the threshold value for t-statistics of ARMA coefficients and constant term used for the final test of model parsimony. If the highest order ARMA coefficient has a t-value smaller than this value in magnitude, the order of the model is reduced. If the constant term t-value is 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's 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)*CV, where CV is the original critical value.

automdl.ljungboxlimit the Ljung Box limit (numeric). 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 the 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.

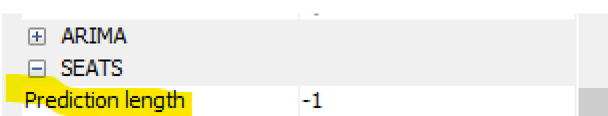
automdl.ubfinal numeric, final unit root limit. The threshold value for the final unit root test. If the magnitude of an AR root for the final model is smaller than the final unit root limit, then a unit root is assumed, the order of the AR polynomial is reduced by one and the appropriate order of the differencing (non-seasonal, seasonal) is increased. The parameter value should be greater than one.

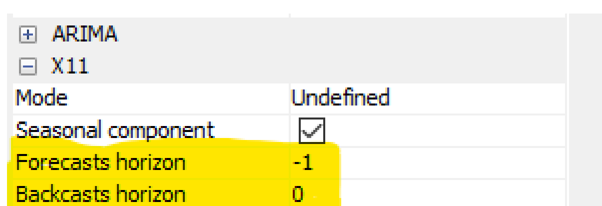
(for both options) *fcst.horizon* the forecasting horizon (numeric). The forecast length generated by the Reg-Arima model in periods (positive values) or years (negative values). By default, the program generates a two-year forecast (fcst.horizon set to -2). Defaults different in GUI and R.



Setting in GUI

Forecast horizon when using tramo seats Is set in the decomposition part of the specification in GUI.





Forecast horizon when using X13-ARIMA

Setting in R (first template, then worked example) X13-Arima template in version 2

```
spec_2 <- x13_spec(spec = spec_1,
automdl.enabled = NA,
  automdl.acceptdefault = NA,
  automdl.cancel = NA_integer_,
  automdl.ub1 = NA_integer_,
  automdl.ub2 = NA_integer_,
  automdl.mixed = NA,
  automdl.balanced = NA,
  automdl.armalimit = NA_integer_,
  automdl.reducecv = NA_integer_,
  automdl.ljungboxlimit = NA_integer_,
  automdl.ubfinal = NA_integer_)
```

add worked example in version 2

in version 3

add worked example in version 3

Options for setting a user-defined Arima model

Control variables for the non-automatic modelling of the ARIMA model (when *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.

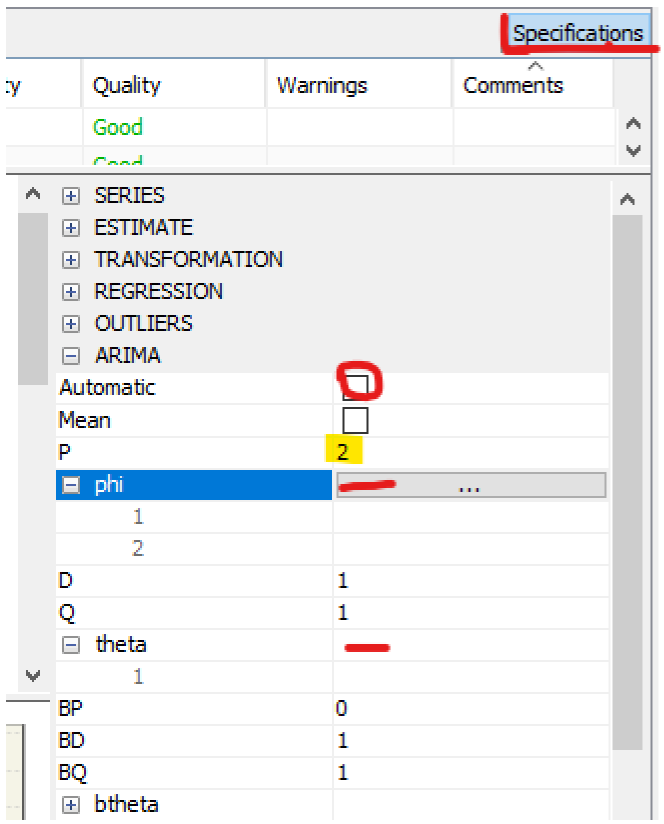
arima.bq numeric. The order of the seasonal moving average (MA) polynomial.

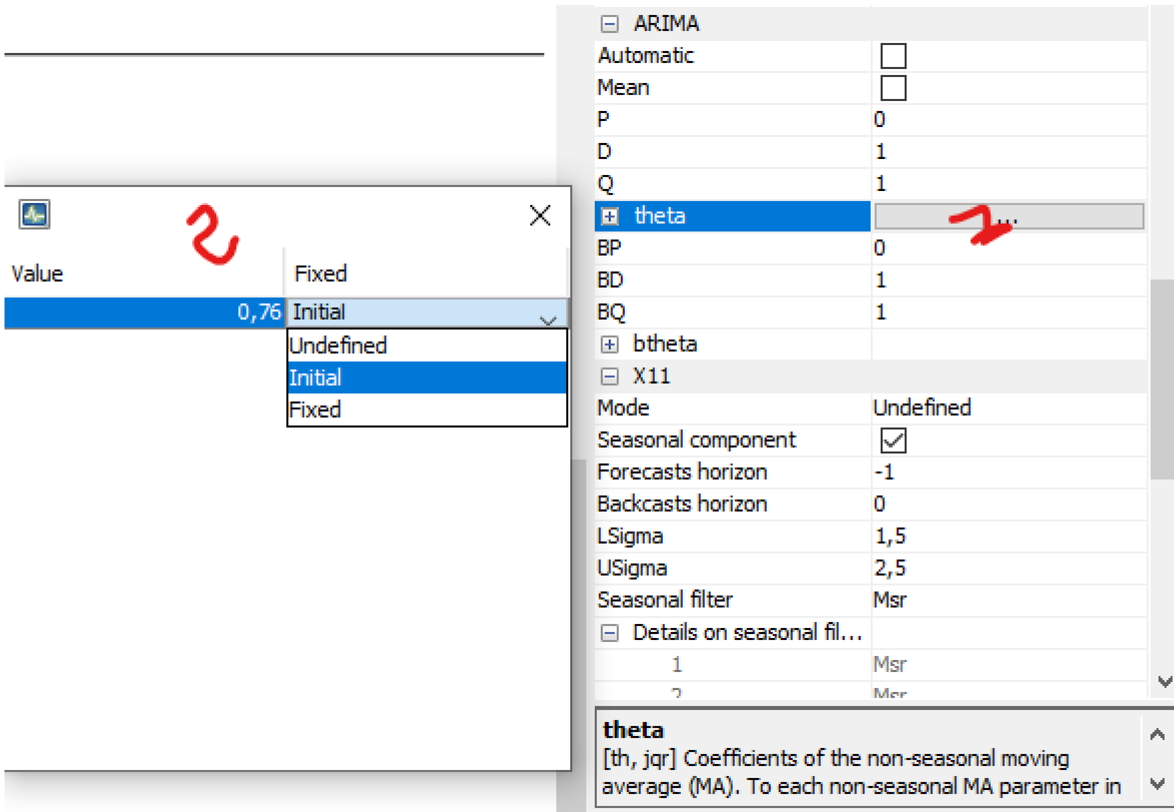
Control variables for the user-defined ARMA coefficients. Coefficients can be defined for the regular and seasonal autoregressive (AR) polynomials and moving average (MA) polynomials. The model considers the coefficients only if the procedure for their estimation (arima.coefType) is provided, and the 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 vector length 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 a vector defining the ARMA coefficients estimation procedure. Possible procedures are: "Undefined" = no use of any user-defined input (i.e. coefficients are estimated), "Fixed" = the coefficients are fixed at the value provided by the user, "Initial" = the value defined by the user is used as the 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").

Setting in GUI 



Fixing coefficients

Setting in R

X13-Arima template in version 2

```
spec_2 <- x13_spec(spec = spec_1,
automdl.enabled = FALSE,

  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_)
```

in version 3

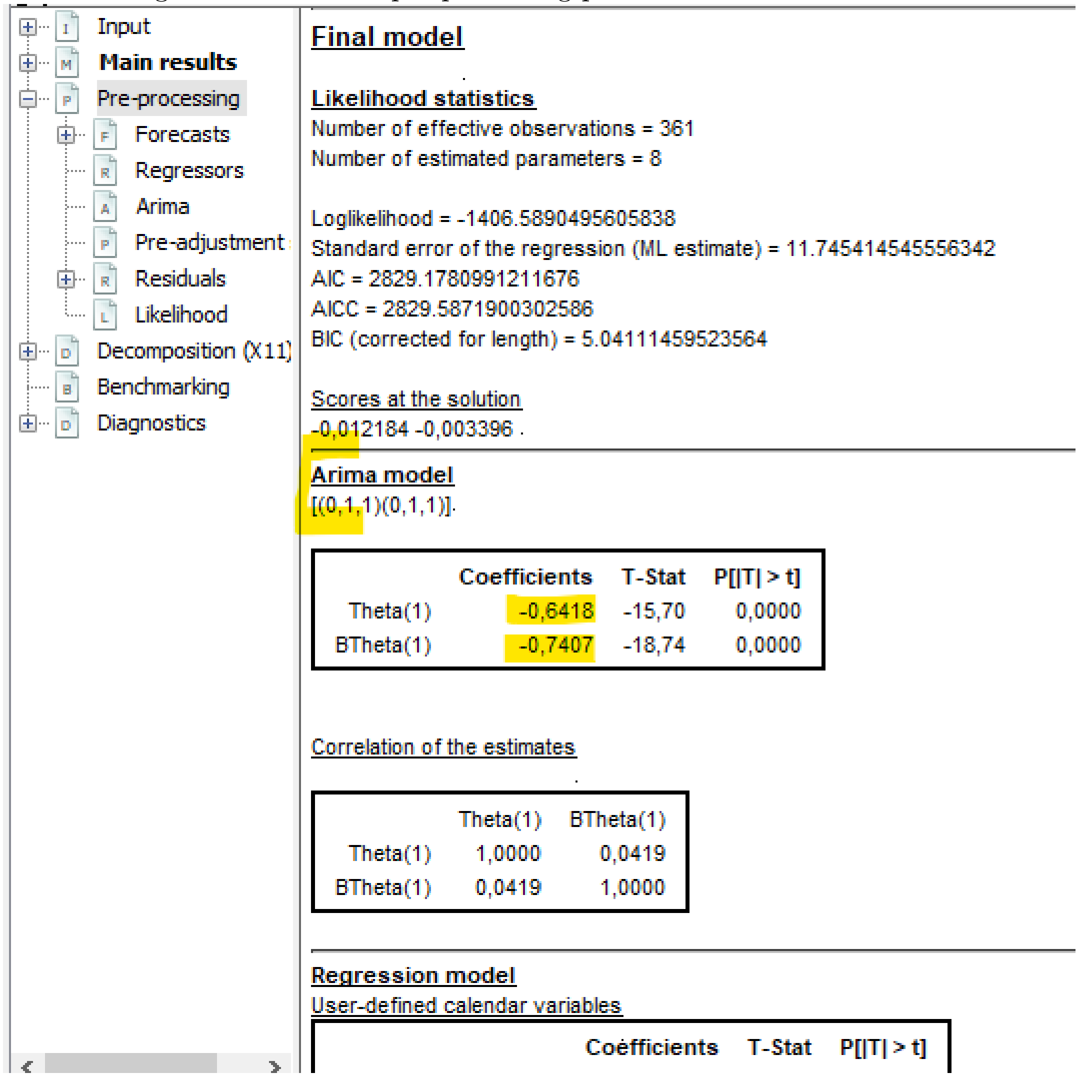
Reg-Arima model Results and Diagnostics

Type of results (including Tramo addenda)

- all regressors used (shown above)
- regression details: explanatory variables (above)
- Arima model specific results
- additional diagnostics on residuals
- likelihood
- seasonality tests on residuals

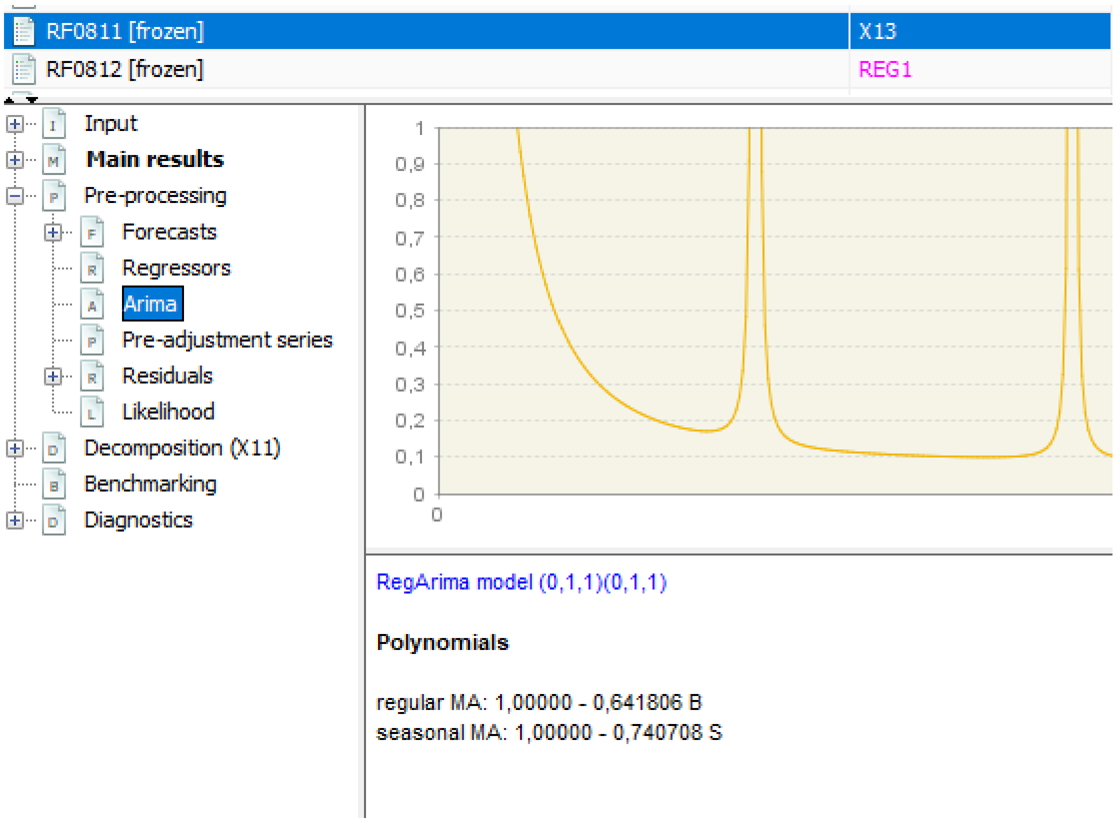
Display in GUI

Reg-Arima model detail with other regression results in pre-processing pane. with number of

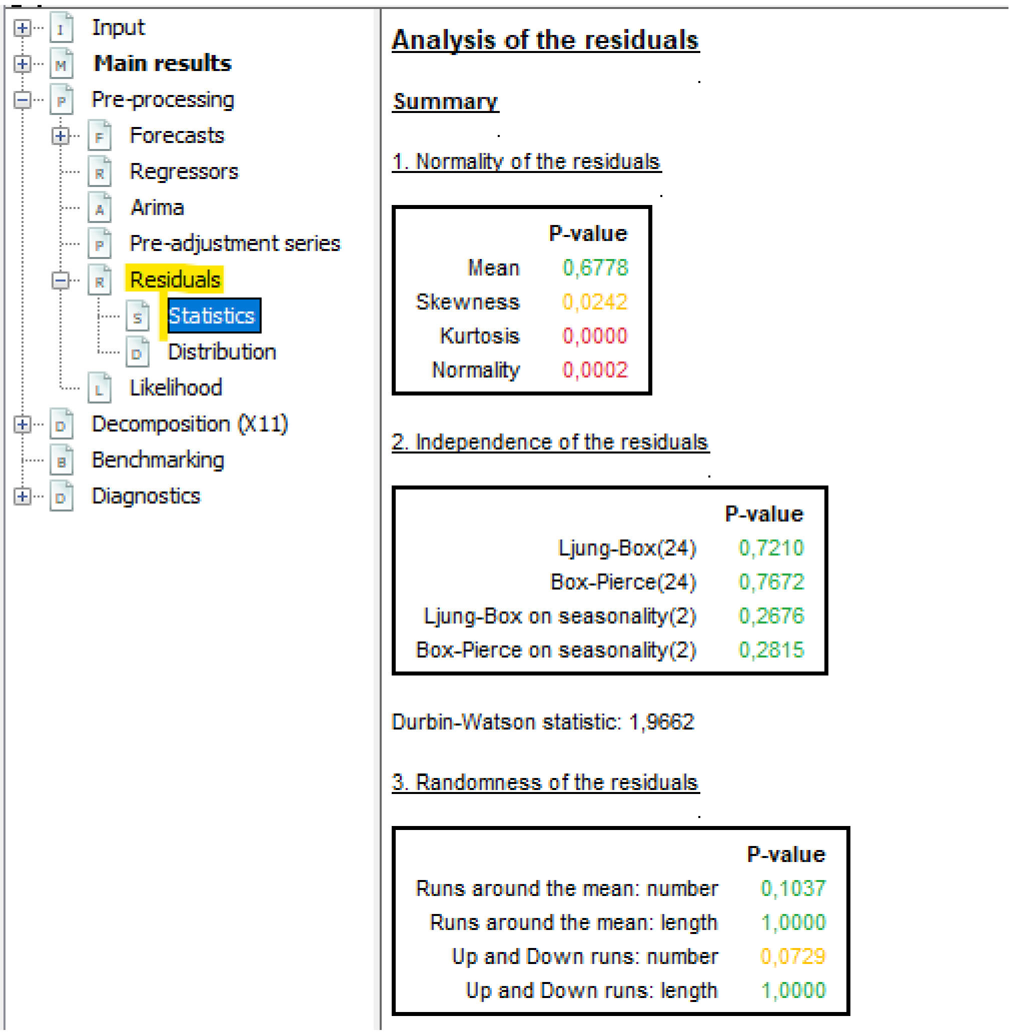


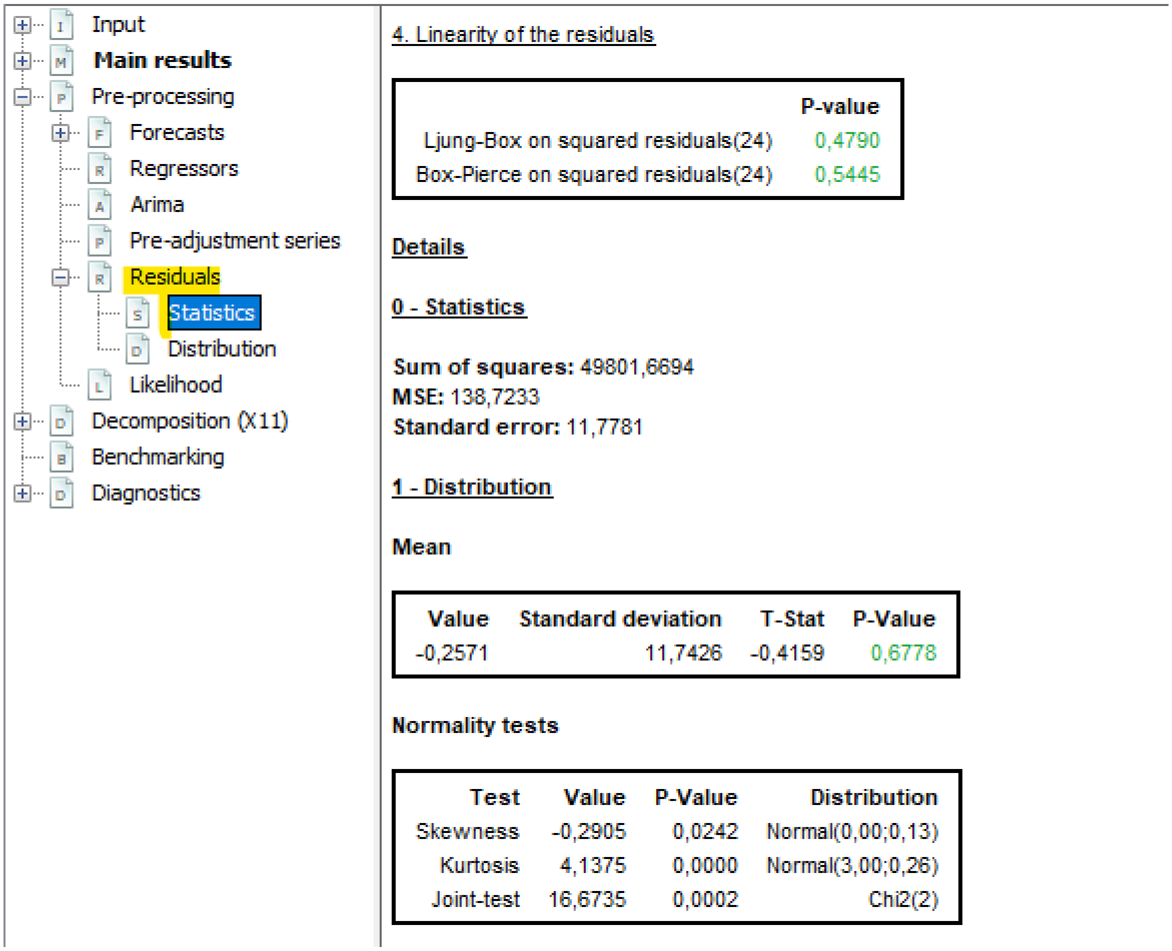
observations.. parameters

More details in Pre-processing/Arima Node



In residual Node


Input
Main results
Pre-processing
Forecasts
Regressors
Arima
Pre-adjustment series
Residuals
Statistics
Distribution
Likelihood
Decomposition (X11)
Benchmarking
Diagnostics
Analysis of the residuals
Summary
1. Normality of the residuals
P-value
Mean 0,6778
Skewness 0,0242
Kurtosis 0,0000
Normality 0,0002
2. Independence of the residuals
P-value
Ljung-Box(24) 0,7210
Box-Pierce(24) 0,7672
Ljung-Box on seasonality(2) 0,2676
Box-Pierce on seasonality(2) 0,2815
Durbin-Watson statistic: 1,9662
3. Randomness of the residuals
P-value
Runs around the mean: number 0,1037
Runs around the mean: length 1,0000
Up and Down runs: number 0,0729
Up and Down runs: length 1,0000

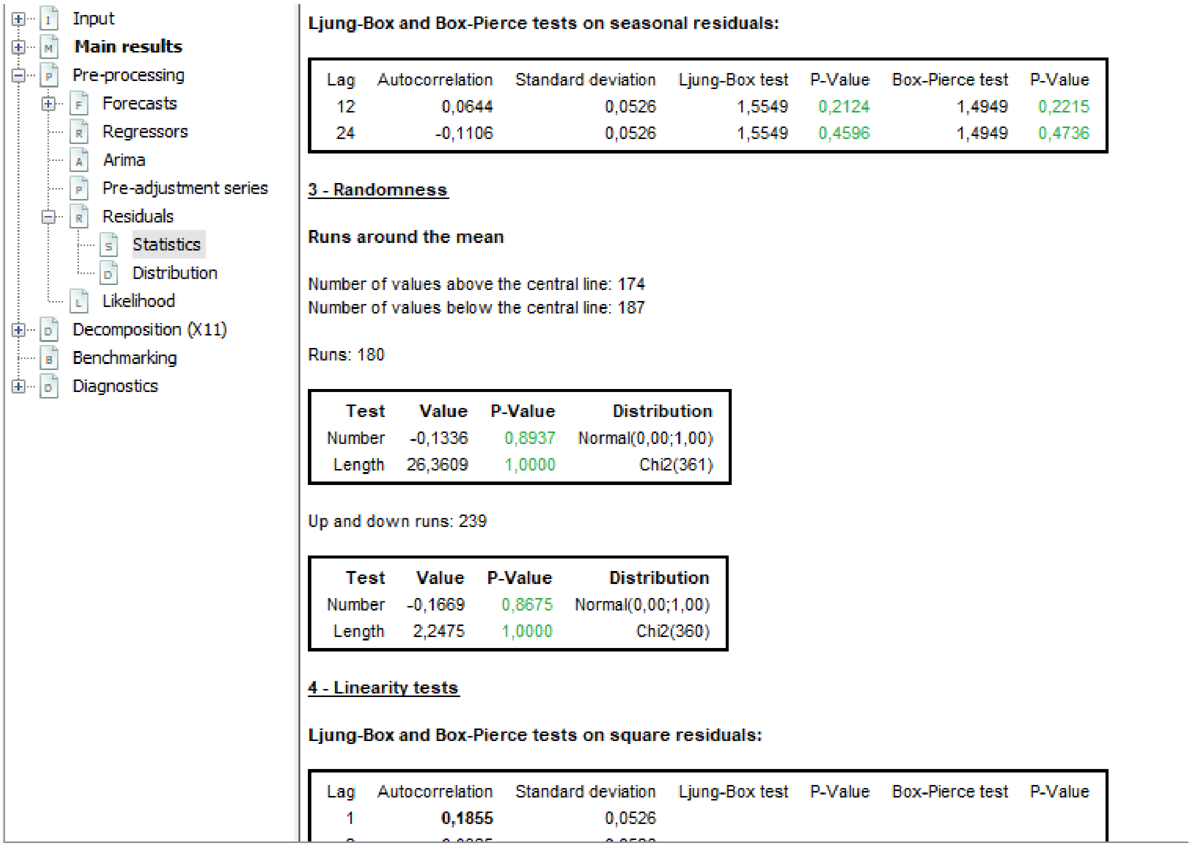

Input
Main results
Pre-processing
Forecasts
Regressors
Arima
Pre-adjustment series
Residuals
Statistics
Distribution
Likelihood
Decomposition (X11)
Benchmarking
Diagnostics
4. Linearity of the residuals
P-value
Ljung-Box on squared residuals(24) 0,4790
Box-Pierce on squared residuals(24) 0,5445
Details
0 - Statistics
Sum of squares: 49801,6694
MSE: 138,7233
Standard error: 11,7781
1 - Distribution
Mean
Value Standard deviation T-Stat P-Value
-0,2571 11,7426 -0,4159 0,6778
Normality tests
Test Value P-Value Distribution
Skewness -0,2905 0,0242 Normal(0,00;0,13)
Kurtosis 4,1375 0,0000 Normal(3,00;0,26)
Joint-test 16,6735 0,0002 Chi2(2)

- Input
- **Main results**
- Pre-processing
  - Forecasts
  - Regressors
  - Arima
  - Pre-adjustment series
  - Residuals
    - Statistics
    - Distribution
  - Likelihood
- Decomposition (X11)
- Benchmarking
- Diagnostics

2 - Independence tests

Ljung-Box and Box-Pierce tests on residuals:

Lag	Autocorrelation	Standard deviation	Ljung-Box test	P-Value	Box-Pierce test	P-Value
1	0,0157	0,0526				
2	0,0143	0,0526				
3	-0,0165	0,0526	0,2645	0,6070	0,2616	0,6090
4	-0,0024	0,0526	0,2666	0,8752	0,2636	0,8765
5	-0,0315	0,0526	0,6323	0,8890	0,6223	0,8913
6	-0,0212	0,0526	0,7983	0,9387	0,7846	0,9405
7	**-0,0957**	0,0526	4,1867	0,5229	4,0890	0,5367
8	-0,0492	0,0526	5,0866	0,5328	4,9641	0,5484
9	-0,0344	0,0526	5,5264	0,5960	5,3906	0,6124
10	-0,0335	0,0526	5,9461	0,6533	5,7964	0,6700
11	0,0275	0,0526	6,2289	0,7168	6,0691	0,7330
12	**0,0838**	0,0526	8,8655	0,5449	8,6040	0,5700
13	0,0134	0,0526	8,9329	0,6281	8,6686	0,6525
14	-0,0076	0,0526	8,9548	0,7068	8,6895	0,7292
15	-0,0038	0,0526	8,9603	0,7759	8,6948	0,7956
16	-0,0436	0,0526	9,6829	0,7850	9,3816	0,8059
17	-0,0090	0,0526	9,7139	0,8373	9,4110	0,8551
18	0,0226	0,0526	9,9089	0,8713	9,5952	0,8869
19	**-0,0912**	0,0526	13,0975	0,7296	12,5994	0,7626
20	0,0058	0,0526	13,1105	0,7850	12,6115	0,8141
21	-0,0126	0,0526	13,1717	0,8297	12,6688	0,8551
22	**-0,0854**	0,0526	15,9923	0,7171	15,3030	0,7588
23	0,0071	0,0526	16,0121	0,7690	15,3214	0,8065
24	**-0,0668**	0,0526	17,7457	0,7210	16,9308	0,7672



Ljung-Box and Box-Pierce tests on seasonal residuals:

Lag	Autocorrelation	Standard deviation	Ljung-Box test	P-Value	Box-Pierce test	P-Value
12	0,0644	0,0526	1,5549	0,2124	1,4949	0,2215
24	-0,1106	0,0526	1,5549	0,4596	1,4949	0,4736

3 - Randomness

Runs around the mean

Number of values above the central line: 174
Number of values below the central line: 187

Runs: 180

Test	Value	P-Value	Distribution
Number	-0,1336	0,8937	Normal(0,00;1,00)
Length	26,3609	1,0000	Chi2(361)

Up and down runs: 239

Test	Value	P-Value	Distribution
Number	-0,1669	0,8675	Normal(0,00;1,00)
Length	2,2475	1,0000	Chi2(360)

4 - Linearity tests

Ljung-Box and Box-Pierce tests on square residuals:

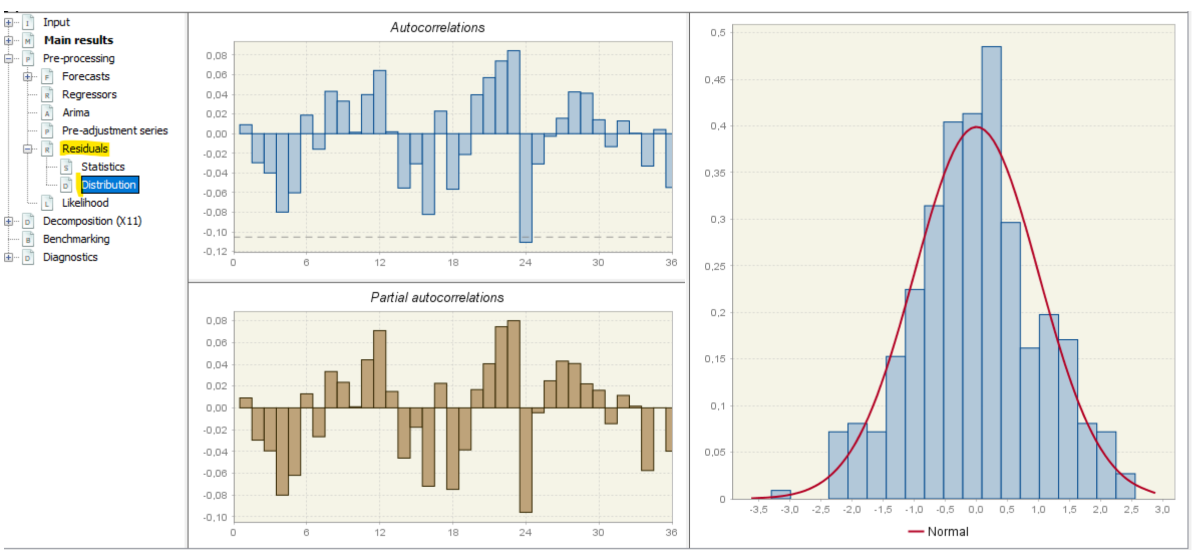
Lag	Autocorrelation	Standard deviation	Ljung-Box test	P-Value	Box-Pierce test	P-Value
1	**0,1855**	0,0526				

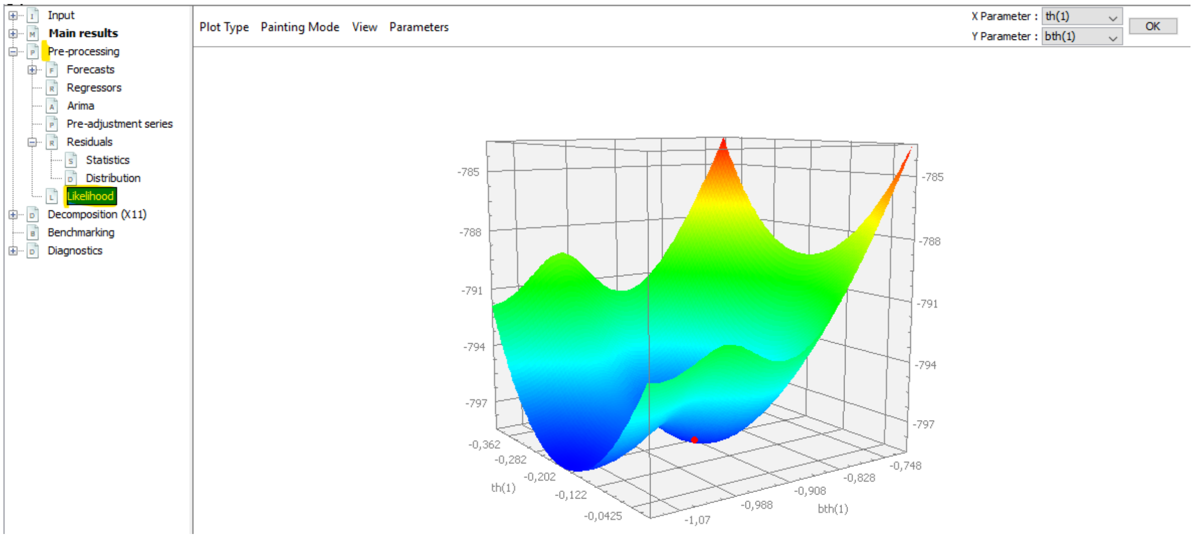
- Input
- **Main results**
- Pre-processing
  - Forecasts
  - Regressors
  - Arima
  - Pre-adjustment series
  - Residuals
    - Statistics
    - Distribution
  - Likelihood
- Decomposition (X11)
- Benchmarking
- Diagnostics

4 - Linearity tests

Ljung-Box and Box-Pierce tests on square residuals:

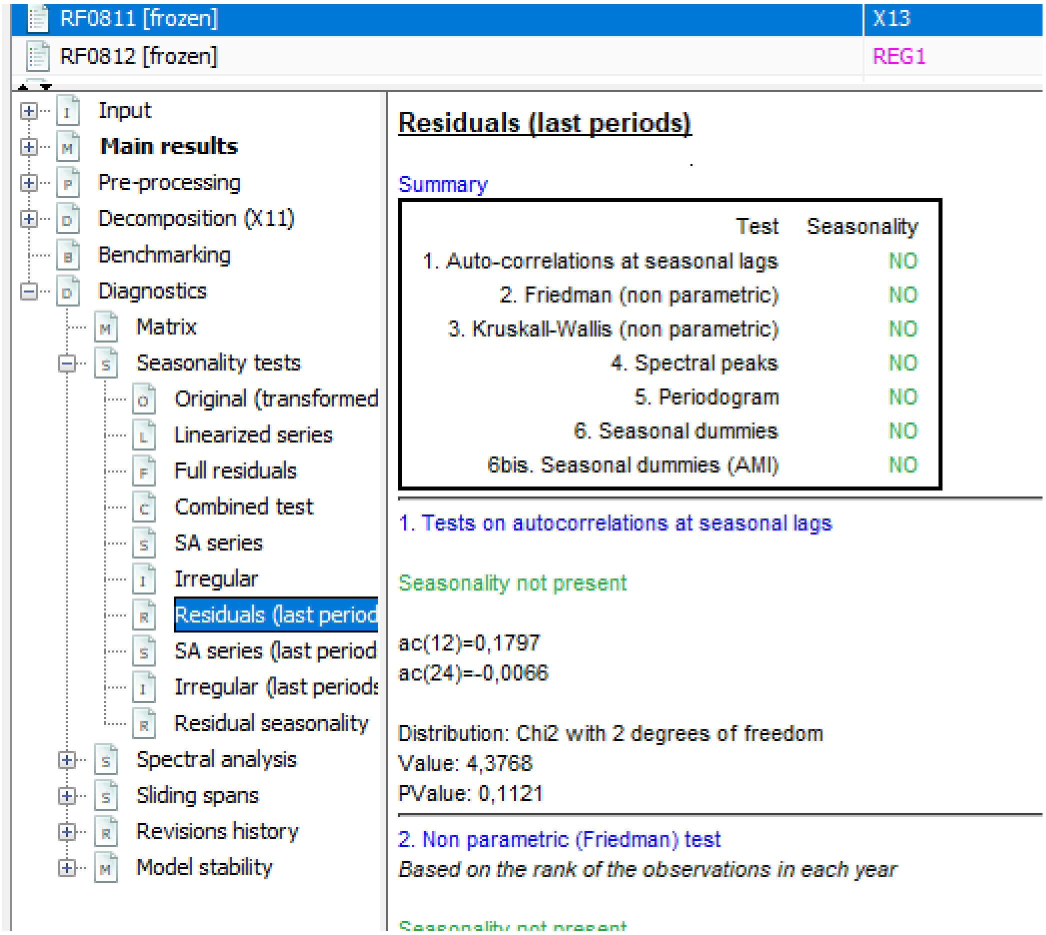
Lag	Autocorrelation	Standard deviation	Ljung-Box test	P-Value	Box-Pierce test	P-Value
1	**0,1855**	0,0526				
2	0,0365	0,0526				
3	**0,0989**	0,0526	16,5952	0,0000	16,4370	0,0001
4	-0,0255	0,0526	16,8333	0,0002	16,6712	0,0002
5	**-0,0587**	0,0526	18,1029	0,0004	17,9163	0,0005
6	**0,0819**	0,0526	20,5819	0,0004	20,3406	0,0004
7	**0,0988**	0,0526	24,1971	0,0002	23,8662	0,0002
8	-0,0044	0,0526	24,2043	0,0005	23,8732	0,0006
9	0,0427	0,0526	24,8840	0,0008	24,5323	0,0009
10	-0,0072	0,0526	24,9032	0,0016	24,5509	0,0019
11	0,0070	0,0526	24,9214	0,0031	24,5685	0,0035
12	-0,0207	0,0526	25,0819	0,0052	24,7227	0,0059
13	0,0080	0,0526	25,1061	0,0088	24,7459	0,0099
14	0,0034	0,0526	25,1106	0,0143	24,7502	0,0161
15	**-0,0870**	0,0526	27,9796	0,0091	27,4849	0,0107
16	-0,0390	0,0526	28,5583	0,0120	28,0349	0,0141
17	0,0071	0,0526	28,5774	0,0182	28,0530	0,0212
18	0,0309	0,0526	28,9424	0,0243	28,3979	0,0283
19	0,0275	0,0526	29,2320	0,0324	28,6707	0,0377
20	**0,0957**	0,0526	32,7548	0,0179	31,9801	0,0221
21	**0,1016**	0,0526	36,7313	0,0086	35,7045	0,0115
22	-0,0136	0,0526	36,8033	0,0124	35,7718	0,0164
23	-0,0361	0,0526	37,3094	0,0155	36,2430	0,0205
24	0,0337	0,0526	37,7520	0,0195	36,6539	0,0258





Seasonality tests on residuals in the **Diagnostics NODE**

(link to test chapter for tests details)



Retrieve in R

(to be added)

X-11 Decomposition

This part explains how to use X-11 decomposition algorithm, via R as well as via GUI. The algorithm itself is explained in more details here

In a nutshell, X-11 will de decompose the **linearized series** using iteratively different moving averages. The effects of pre-treatment will be reallocated at the end.

The sections below (will) describe

- specifications needed to run X-11

- generated output
- series
- diagnostics
- final parameters
- user-defined parameters

Default specifications

The default specifications for X-11 must be chosen at the starting of the SA processing. They are detailed in the Reg-Arima part. X-11 can be run without pre-treatment

Quick Launch

From GUI

With a workspace open, an SAProcessing created and open data provider:

- choose a default specification
- drop your data and press green arrow

In R

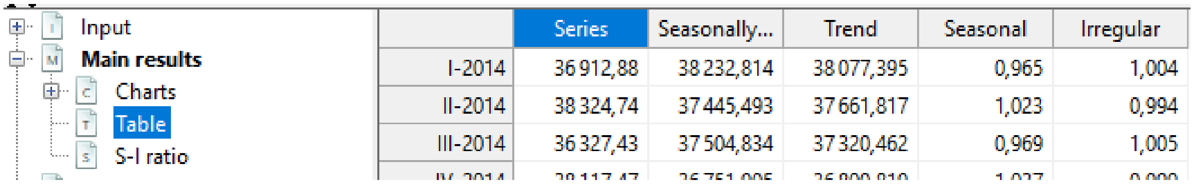
In version 2

```
library("RJDemetra")
model_sa <- x13(raw_series, spec ="RSA5c")
```

The model_sa R object (list of lists) contains all parameters and results. It will be progressively detailed below.

Retrieve series

Display in GUI

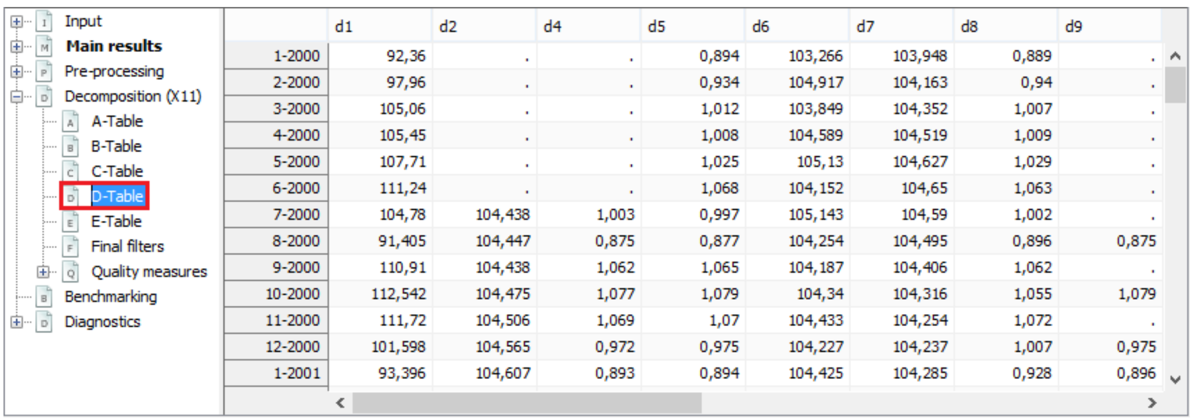


	Series	Seasonally...	Trend	Seasonal	Irregular
I-2014	36 912,88	38 232,814	38 077,395	0,965	1,004
II-2014	38 324,74	37 445,493	37 661,817	1,023	0,994
III-2014	36 327,43	37 504,834	37 320,462	0,969	1,005

Main results

(forecasts glued, values in *italic*)

X-11 Tables



	d1	d2	d4	d5	d6	d7	d8	d9
1-2000	92,36	.	.	0,894	103,266	103,948	0,889	.
2-2000	97,96	.	.	0,934	104,917	104,163	0,94	.
3-2000	105,06	.	.	1,012	103,849	104,352	1,007	.
4-2000	105,45	.	.	1,008	104,589	104,519	1,009	.
5-2000	107,71	.	.	1,025	105,13	104,627	1,029	.
6-2000	111,24	.	.	1,068	104,152	104,65	1,063	.
7-2000	104,78	104,438	1,003	0,997	105,143	104,59	1,002	.
8-2000	91,405	104,447	0,875	0,877	104,254	104,495	0,896	0,875
9-2000	110,91	104,438	1,062	1,065	104,187	104,406	1,062	.
10-2000	112,542	104,475	1,077	1,079	104,34	104,316	1,055	1,079
11-2000	111,72	104,506	1,069	1,07	104,433	104,254	1,072	.
12-2000	101,598	104,565	0,972	0,975	104,227	104,237	1,007	0,975
1-2001	93,396	104,607	0,893	0,894	104,425	104,285	0,928	0,896

Figure 2: Text

Output series can be exported out of GUI by two means:

- generating output files
- running the cruncher to generate those files as described here

Retrieve in R

In version 2

```
# final components
model_sa$final$series
# final forecasts y_f sa_f s_f t_f i_f
model_sa$final$forecasts
# from user defined output
```

Retrieve Diagnostics

X11 produces the following type diagnostics or quality measures

SI-ratios

0.0.0.0.1 * Display in GUI

NODE Main Results > SI-Ratios SA_MainResults_SI_ratios.png

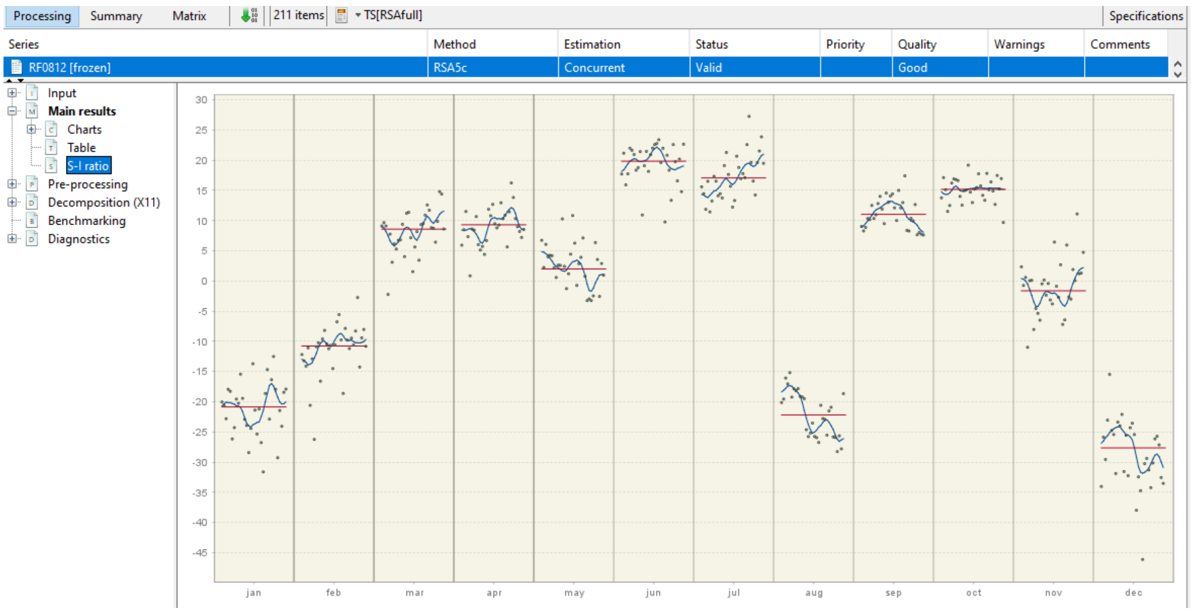


Figure 3: Text

In GUI all values cannot be retrieved

0.0.0.0.2 * Retrieve in R

In version 2

```
# data frame with values
model_sa$decomposition$si_ratio
# customizable plot
plot(model_sa, type= "cal-seas-irr",first_date = c(2015, 1))
```

M-statistics

At the end of the decomposition, X-11 algorithm provides quality measure of the decomposition called "M statistics": 11 statistics (M1 to M11) and 2 summary indicators (Q et Q-M2). By design $0 < M_x < 3$ and acceptance region is $M_x \leq 1$

- **M1** The relative contribution of the irregular over three months span
- **M2** The relative contribution of the irregular component to the stationary portion of the variance
- **M3** The amount of month to month change in the irregular component as compared to the amount of month to month change in the trend-cycle (I/C-ratio)
- **M5** MCD (Months for Cyclical Dominance): The number of months it takes the change in the trend-cycle to surpass the amount of change in the irregular
- **M6** The amount of year to year change in the irregular as compared to the amount of year to year change in the seasonal (only valid for 3x5 seasonal filter)
- **M7** The amount of moving seasonality present relative to the amount of stable seasonality
- **M8** The size of the fluctuations in the seasonal component throughout the whole series
- **M9** The average linear movement in the seasonal component throughout the whole series
- **M10** Same as 8, calculated for recent years only (4 years, N-2 to N-5)
- **M11** Same as 9, calculated for recent years only

The Q statistic is a composite indicator calculated from the M statistics.

$$Q = \frac{10M1 + 11M2 + 10M3 + 8M4 + 11M5 + 10M6 + 18M7 + 7M8 + 7M9 + 4M10 + 4M11}{100}$$

$Q = Q - M2$ (also called $Q2$) is the Q statistic for which the $M2$ statistics was excluded from the formula, i.e.:

$$Q - M2 = \frac{10M1 + 10M3 + 8M4 + 11M5 + 10M6 + 18M7 + 7M8 + 7M9 + 4M10 + 4M11}{89}$$

If a time series does not cover at least 6 years, the $M8$, $M9$, $M10$ and $M11$ statistics cannot be calculated. In this case the Q statistic is computed as:

$$Q = \frac{14M1 + 15M2 + 10M3 + 8M4 + 11M5 + 10M6 + 32M7}{100}$$

The model has a satisfactory quality if the Q statistic is lower than 1.

0.0.0.0.1 * Display in GUI

To display results in GUI, expand NODE

Decomposition(X-11) > Quality Measures > Summary

Results displayed in red indicate that the test failed.

Monitoring and Quality Assessment Statistics

M-1	0.547	The relative contribution of the irregular over three months span
M-2	0.020	The relative contribution of the irregular component to the stationary portion of the variance
M-3	0.000	The amount of period to period change in the irregular component as compared to the amount of period to period change in the trend-cycle
M-4	1.744	The amount of autocorrelation in the irregular as described by the average duration of run
M-5	0.152	The number of periods it takes the change in the trend- cycle to surpass the amount of change in the irregular
M-6	0.209	The amount of year to year change in the irregular as compared to the amount of year to year change in the seasonal
M-7	1.379	The amount of moving seasonality present relative to the amount of stable seasonality
M-8	1.743	The size of the fluctuations in the seasonal component throughout the whole series
M-9	0.339	The average linear movement in the seasonal component throughout the whole series
M-10	2.986	The size of the fluctuations in the seasonal component in the recent years
M-11	2.891	The average linear movement in the seasonal component in the recent years
Q	0.863	
Q-m2	0.967	

Figure 4: Text

0.0.0.0.2 * Retrieve in R

In version 2

```
# this code snippet is not self-sufficient
model_sa$decomposition$mstats
```

Detailed Quality measures

In GUI all the diagnostics below can be displayed expanding the NODE

Decomposition(X-11) > Quality Measures > Details

They are detailed in the X-11 method chapter

In R (to be added)

Retrieve final parameters

This section describes the parameters which are automatically chosen by the software as a result of the estimation process. They have no default value.

Final trend filter: length of Henderson filter applied for final estimation (in the second part of the D step).

Final seasonal filer: length of Henderson filter applied for final estimation (in the second part of the D step).

Display in GUI

Node Decomposition(X11) > Final Filters

Retrieve in R

In version 2

```
model_sa$decomposition$s_filter
model_sa$decomposition$t_filter
```

User-defined parameters

The following sections describe how to change default values or automatic choices.

0.0.0.0.1 * General settings

- **Mode**
- **Seasonal component**
- **Forecasts horizon**

Length of the forecasts generated by the Reg-Arima model - in months (positive values) - years (negative values) - if set to is set to 0, the X-11 procedure does not use any model-based forecasts but the original X-11 type forecasts for one year. - default value: -1, thus one year from the Arima model

- **Backcasts horizon**

Length of the backcasts generated by the Reg-Arima model - in months (positive values) - years (negative values) - default value: 0

0.0.0.0.2 * Irregular correction

- **LSigma**
  - sets lower sigma (standard deviation) limit used to down-weight the extreme irregular values in the internal seasonal adjustment iterations
  - values in $[0, Usigma]$
  - default value is 1.5
- **USigma**
  - sets upper sigma (standard deviation)
  - values in $[Lsigma, +\infty]$
  - default value is 2.5
- **Calendarsigma**
  - allows to set different **LSigma** and **USigma** for each period
  - values
    * None (default)
    * All: standard errors used for the extreme values detection and adjustment computed separately for each calendar month/quarter
    * Signif: groups determined by Cochran test (check)
    * Sigmavec: set two customized groups of periods
- **Excludeforecasts**
  - ticked: forecasts and backcasts from the Reg-Arima model not used in Irregular Correction
  - unticked (default): forecasts and backcasts used

0.0.0.0.3 * Seasonality extraction filters choice

Specifies which be used to estimate the seasonal factors for the entire series.

- **Seasonal filter**
- default value: *MSR* (Moving seasonality ratio), automatic choice of final seasonal filter, initial filters are 3×3
- choices: 3×1, 3×3, 3×5, 3×9, 3×15 or Stable
- “Stable”: constant factor for each calendar period (simple moving average of a all $S + I$ values for each period)

User choices will be applied to final phase D step.

The seasonal filters can be selected for the entire series, or for a particular month or quarter.

- **Details on seasonal filters**

Sets different seasonal filters by period in order to account for seasonal heteroskedasticity (link to M chapter)

- default value: empty

0.0.0.0.4 * Trend estimation filters

- **Automatic Henderson filter** our user-defined
  - default: length 13
  - unticked: user defined length choice
- **Henderson filter** length choice
  - values: odd number in $[3, 101]$
  - default value: 13

Check: will user choice be applied to all steps or only to final phase D step

Parameter setting in GUI

All the parameters above can be set with in the specification box.

Parameter setting in R packages

In version 2

```
#Creating a modified specification, customizing all available X11 parameters
modified_spec<- x13_spec(current_sa_model,
    #x11.mode="?",
    #x11.seasonalComp = "?",
    x11.fcasts = -2,
    x11.bcasts = -1,
    x11.lsigma = 1.2,
    x11.usigma = 2.8,
    x11.calendarSigma = NA,
      x11.sigmaVector = NA,
    x11.excludeFcasts = NA
```

```
    # filters
    x11.trendAuto = NA,
    x11.trendma = 23,
    x11.seasonalma = "S3X9)

#New SA estimation: apply modified_spec

modified_sa_model<-x13(raw_series,modified_spec)
```

SEATS Decomposition

SEATS algorithm will decompose the linearized series, in level or in logarithm, using the Arima model fitted by Tramo in the pre-treatment phase.

The sections below will describe

- specifications needed to run SEATS
- generated output
- series
- diagnostics
- final parameters
- user-defined parameters

Default specifications

The default specifications for SEATS must be chosen at the starting of the SA processing. They are detailed in the RegArima part Starting point for Tramo-Seats

Quick Launch

From GUI

With a workspace open, an SAProcessing created and open data provider:

- choose a default specification (link)
- drop your data and press green arrow (link)

In R

In version 2

```
library("RJDemetra")
model_sa <- tramoseats(raw_series, spec ="RSAfull")
```

In version 3

```
library("rjd3tramoseats")
model_sa <- tramoseats(raw_series, spec ="RSAfull")
```

The model_sa R object (list of lists) contains all parameters and results. It will be progressively detailed below.

More details on the functions are to be found in R help pages.

Retrieve Series

This section outlines how to retrieve the different kinds of output series from GUI or in R.

- final components (including reallocation of pre-adjustment effects)
- components in level
- components in level or log

Stochastic series

Decomposition of the linearized series or of its logarithm (in case of a multiplicative model)

y_lin is split into components: t_lin, s_lin, i_lin

suffixes: - _f stands for forecast - _e stands for - _ef stands for

0.0.0.0.1 * Display in GUI

NODE Decomposition>Stochastic series - Table with series and its standard error image

- Plot of Trend with confidence interval image
- Plot of Seasonal component with confidence interval image

0.0.0.0.2 * Retrieve from GUI

Generating output from GUI (link) or from Cruncher (link), stochastic series, their standard errors, forecasts and forecasts errors can be accessed with the following names

Series Name	Meaning
decomposition.y_lin	
decomposition.y_lin_f	
decomposition.y_lin_ef	
decomposition.t_lin	
decomposition.t_lin_f	
decomposition.t_lin_e	
decomposition.t_lin_f	
decomposition.sa_lin	
decomposition.sa_lin_f	
decomposition.sa_lin_e	
decomposition.sa_lin_ef	
decomposition.s_lin	
decomposition.s_lin_f	
decomposition.s_lin_e	
decomposition.s_lin_ef	
decomposition.i_lin	
decomposition.i_lin_f	
decomposition.i_lin_e	
decomposition.i_lin_ef	

0.0.0.0.3 * Retrieve in R

In version 2

```
library("RJDemetra")
# list of additional output objects
user_defined_variables("TRAMO-SEATS")
# specify additional objects in estimation
m <- tramoseats(y,"RSAfull", userdefined=c( "decomposition.y_lin", "ycal","variancedecompo
# retrieve objects
m$user_defined$decomposition.y_lin
m$user_defined$ycal
m$user_defined$variancedecomposition.seasonality
```

In version 3

```r
library("rjd3tramoseats")
# list of additional output objects
userdefined_variables_tramoseats("tramoseats")
# specify additional objects in estimation
m <- tramoseats(y,"RSAfull", userdefined=c("decomposition.y_lin","ycal","variancedecomposi
# retrieve objects
m$user_defined$decomposition.y_lin
m$user_defined$ycal
m$user_defined$variancedecomposition.seasonality
```

Components (Level)

Decomposition of the linearized series, back to level in case of a multiplicative model.

y_lin is split into components: t_lin, s_lin, i_lin

suffixes: - _f stands for forecast - _e stands for - _ef stands for

0.0.0.0.1 * Displayed in GUI

NODE Decomposition>Components - Table with series and its standard error image

0.0.0.0.2 * Retrieve from GUI

Generating output from GUI (link) or from Cruncher (link), component series, their standard errors, forecasts and forecasts errors can be accessed with the following names

Series Name	Meaning
decomposition.y_cmp	
decomposition.y_cmp_f	
decomposition.y_cmp_ef	
decomposition.t_cmp	
decomposition.t_cmp_f	
decomposition.t_cmp_e	
decomposition.t_cmp_f	
decomposition.sa_cmp	
decomposition.sa_cmp_f	
decomposition.sa_cmp_e	
decomposition.sa_cmp_ef	
decomposition.s_cmp	
decomposition.s_cmp_f	

Series Name	Meaning
decomposition.s_cmp_e	
decomposition.s_cmp_ef	
decomposition.i_cmp	
decomposition.i_cmp_f	
decomposition.i_cmp_e	
decomposition.i_cmp_ef	

0.0.0.0.3 * Retrieve in R

Same procedure as for stochastic series.

0.0.0.0.4 * Bias correction to be added

Final series

Series	Final SEATS components	Final Results	Reallocation of pre-adjustment effects
Raw series (forecasts)		y (y_f)	
Linearized series	B1		none
Final seasonal component	D16	s (s_f)	
Final trend	D12	t (t_f)	
Final irregular	D13	i (i_f)	
Calendar component			
Seasonal without calendar	D10		

(to be added: reallocation of outliers effects)

0.0.0.0.1 * Display in GUI

Final results are displayed for each series in the NODE MAIN>Table

Forecasts are glued at the end it *italic*

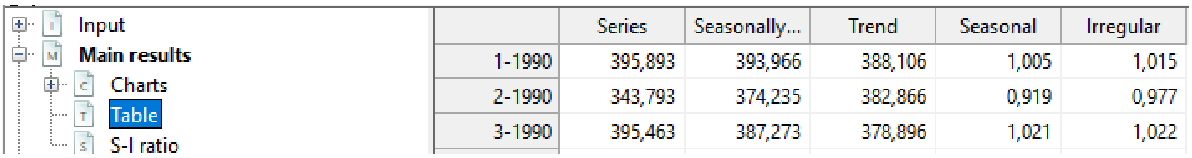


Figure 5: Text

0.0.0.0.2 * Retrieve from GUI

Generating output from GUI (link) or from Cruncher (link), component series, their standard errors, forecasts and forecasts errors can be accessed with the following names

Series Name	Meaning
y	
y_f	
t	
t_f	
sa	
sa_f	
s	
s_f	
i	
i_f	

0.0.0.0.3 * Retrieve in R

In version 2

```
library("RJDemetra")
sa_model <- RJDemetra::tramoseats(y,"RSAfull")
sa_model$final$series
sa_model$final$forecasts
# for additional results call user-defined output as explained above
```

In version 3

```
library("rjd3tramoseats")
sa_model <- tramoseats(y,spec="RSAfull")
# final series can be accessed here
sa$result$final$sa
# for additional results call user-defined output as explained above
```

Retrieve Diagnostics

- WK analysis

components final estimators

- Error analysis autocorrelation of the errors (sa, trend) revisions of the errors
- Growth rates
- Model based tests
- Significant seasonality
- Stationary variance decomposition

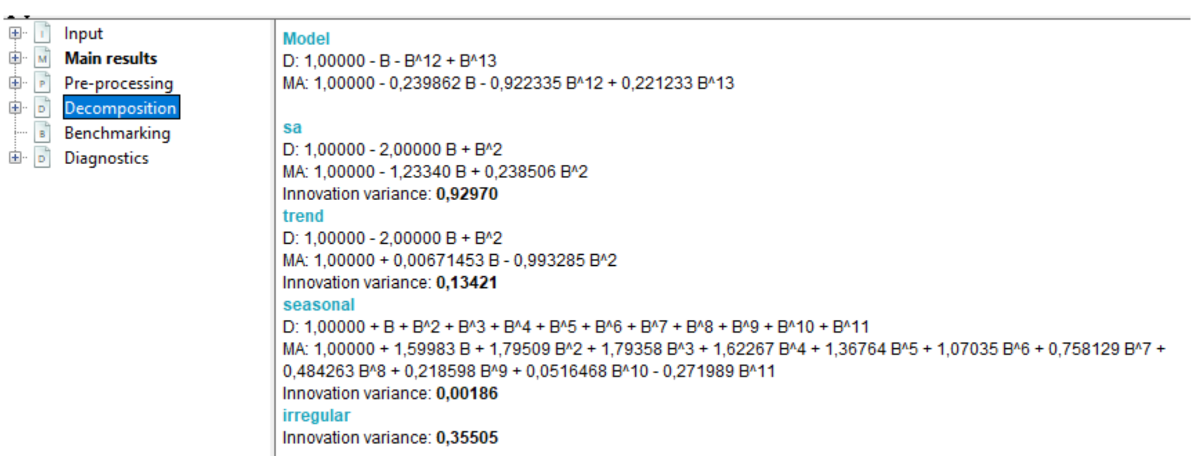
Retrieve Final Parameters

Relevant if parameters not set manually, or any parameters automatically selected by the software without having a fixed default value. (The rest of the parameters is set in the specification) To manually set those parameters and see all the fixed default values see Specifications / parameters section

Arima Models for components

0.0.0.0.1 * Display in GUI

Click on the **Decomposition** NODE



0.0.0.0.2 * Retrieve from GUI

(add names for output and cruncher)

0.0.0.0.3 * Display in R

(display or retrieve)

version 2

version 3

Other final parameters

Final parameters which can be fine-tuned be the user are described in User-defined specifications section below

Setting user-defined parameters

The section below explains how the user can fine-tune some seats parameters, which are put in context in the corresponding method chapter.the default value is indicated in ().

- Prediction length

Forecast span used in the decomposition default: one year (-1) (years are set in negative values, positive values indicate number of periods)

- Approximation Mode

Modification type for inadmissible models None (default) Legacy Noisy

- MA unit root boundary

Modulus threshold for resetteing MA "near-unit" roots [0,1] default (0.95)

- Trend Boundary Modulus threshold for assigning positive real AR Roots [0,1] default (0.5)
- Seasonal Tolerance Degree threshold for assigning complex AR roots [0,10] default (2)
- Seasonal Boundary (unique) Modulus threshold for assigning negative real AR roots [0,1] default (0.8)
- Seasonal Boundary (unique) Same modulus threshold unique seasonal AR roots [0,1] default (0.8)
- Method

Algorithm used for estimation of unobserved components

Burman (default)

KalmanSmoother

McEllroyMatrix

0.0.0.0.1 * Seting parameters in GUI

In specification window corresponding to a given series:

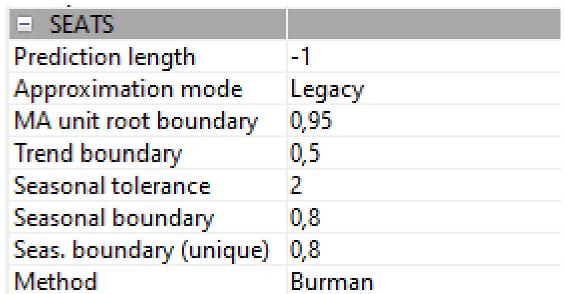


Figure 6: Text

0.0.0.0.2 * Set in R

version 2 (RJDemetra)

```
tramoseats_spec(
 spec = c("RSAfull", "RSA0", "RSA1", "RSA2", "RSA3", "RSA4", "RSA5"),
  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")
)
```

in version 3 with {rjd3tramoseats} (to be added)

STL

Loess based decomposition algorithm used on linearized data data, no pre-adjustment.

Not currently available. Under construction.

Basic Structural Models

Not currently available. Under construction.

Seasonal adjustment of high-frequency data

Overview

This chapter provides guidance on seasonal adjustment of infra-monthly, or high-frequency (HF), time-series data with JDemetra+ tailored algorithms.

Currently available topics:

- description of HF data specificities
- R functions for pre-treatment, extended X-11 and extended Seats

Topics under construction

- graphical user interface 3.0 functionalities for HF data
- STL functions
- State space framework

Data specificities

HF data often display multiple seasonal patterns with potentially non-integer periodicities which cannot be modeled with classical SA algorithms. JD+ provides tailored versions of these algorithms.

Table 10: Periodicities (number of observations per cycle)

Data	Day	Week	Month	Quarter	Year
quarterly					4
monthly				3	12
weekly			4.3481	13.0443	52.1775
daily		7	30.4368	91.3106	365.2425
hourly	24	168	730.485	2191.4550	8765.82

Tailored algorithms in JDemetra+

Col1	Algorithm	GUI v 3.0	R package
Pre-treatment	Extended Airline Model	yes	rjd3highfreq
Decomposition	Extended SEATS Extended Airline Model	yes	rjd3highfreq
	Extended X-11	yes	rjd3x11plus
	Extended STL	no	rjd3stl
One-Step	SSF Framework	no	rjd3sts

Unobserved Components

Raw series decomposition

$$
\begin{array}{ccccccccc}
\text{series raw} & = & \text{trend} & \circ & \text{seasonal} & \circ & \text{calendar} & \circ & \text{irregular} \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\
y_t & = & t_t & \circ & s_t & \circ & c_t & \circ & i_t \\
 & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\
 & & \text{Variations} & & \text{Variations} & & \text{effects} & & \text{residual} \\
 & & \text{long-term} & & \text{infra-yearly} & & & & \\
 & & & & \text{periodic} & & \text{calendar} & & \\
 & & \searrow & & & & & & \swarrow \\
 & & & & \multicolumn{3}{c}{\text{seasonally adjusted (sa) data}} & &
\end{array}
$$

- decomposition Additive ($\circ = +$), multiplicative ($\circ = \times$)

Multiple seasonal patterns

HF data often contain multiple seasonal patterns. For example, daily economic time series often display strong infra-weekly and infra-yearly seasonality. An infra-monthly seasonal pattern may also be present, but its strength is usually less pronounced in practice. In theory, the full decomposition of the seasonal component in daily data is given by:

$$S_t = S_{t,7} \circ S_{t,30.44} \circ S_{t,365.25}$$

The decomposition is done iteratively periodicity by periodicity starting with the smallest one (highest frequency) as:

- highest frequencies usually display the biggest and most stable variations
- cycles of highest frequencies can mix up with lower ones

Identifying seasonal patterns

JDemetra+ provides the Canova-Hansen test in the rjd3toolkit package.

Pre-adjustment

In classical X13-Arima and Tramo-Seats, a pre-adjustment step is performed to remove deterministic effects, such as outliers and calendar effects, with a Reg-Arima model. In the extended version for HF data, it is also the case with an **extended Airline model**.

A general Reg-ARIMA model is written as follows:

$$\left(Y_t - \sum \alpha_i X_{it}\right) \sim ARIMA(p,d,q)(P,D,Q)$$

These models contain seasonal backshift operators $B^s(y_t) = y_{t-s}$. Here s can be non-integer. JDemetra+ will rely on a modified version of a frequently used Arima model: the "Airline" model:

$$(1-B)(1-B^s)y_t = (1-\theta_1 B)(1-\theta_2 B^s)\epsilon_t \qquad \epsilon_t \sim \text{NID}(0, \sigma^2_\epsilon)$$

For HF data, the potentially non-integer periodicity s will be written: $s = s' + \alpha$, with $\alpha \in [0, 1)$ (for example $52.18 = 52 + 0.18$ is the yearly periodicity for weekly data)

Taylor series development around 1 of $f(x) = x^\alpha$

$$\begin{array}{rcl} x^\alpha & = & 1 + \alpha(x-1) + \frac{\alpha(\alpha+1)}{2!}(x-1)^2 + \frac{\alpha(\alpha+1)(\alpha+2)}{3!}(x-1)^3 + \cdots \\ B^\alpha & \cong & (1-\alpha) + \alpha B \end{array}$$

Approximation of $B^{s+\alpha}$ in an extended Airline model

$$B^{s+\alpha} \quad \cong \quad (1-\alpha)B^s + \alpha B^{s+1}$$

Example for a daily series displaying infra-weekly ($p_1 = 7$) and infra-yearly ($p_2 = 365.25$) seasonality:

$$(1-B)(1-B^7)(1-B^{365.25)}(Y_t - \sum \alpha_i X_{it}) = (1-\theta_1 B)(1-\theta_2 B^7)(1-\theta_3 B^{365.25})\epsilon_t$$

$$\epsilon_t \overset{iid}{\sim} \mathrm{N}(0, \sigma^2_\epsilon)$$

with

$$1 - B^{365.25} = 1 - (0.75B^{365} + 0.25B^{366})$$

Calendar correction

Calendar regressors can be defined with the rjd3toolkit package and added to pre-treatment function as a matrix.

```
# Create a calendar with rjd3toolkit
# Define a national calendar
frenchCalendar <- national_calendar(days = list(
  fixed_day(7, 14), # Bastille Day
  fixed_day(5, 8, validity = list(start = "1982-05-08")), # Victory Day
  special_day('NEWYEAR'),
  special_day('CHRISTMAS'),
  special_day('MAYDAY'),
  special_day('EASTERMONDAY'),
  special_day('ASCENSION'),
  special_day('WHITMONDAY'),
  special_day('ASSUMPTION'),
  special_day('ALLSAINTSDAY'),
  special_day('ARMISTICE'))
)
# Generrate calendar regressors
q<-holidays(frenchCalendar, "1968-01-01", length = length(df_daily$births), type="All",
            nonworking = as.integer(7))

# Argument type = All : taking all holidays into account
# Argument type = Skip : taking into account only the holidays falling on a week day
```

Outliers and intervention variables

Outliers detection is available in the pre-treatment function. Detected outliers are AO, LS and WO. Critical value can be computed by the algorithm or user-defined.

Linearization

Example using rjd3highfreq::fractionalAirlineEstimation function:

```
pre_adjustment<- rjd3highfreq::fractionalAirlineEstimation(y_raw,
                 x = q, # q = daily calendar regressors
                 periods = c(7, 365.25),
                 ndiff = 2, ar = FALSE, mean = FALSE,
                 outliers = c("ao","ls", "wo"),
                 criticalValue = 0, # computed in the algorithm
                 precision = 1e-9, approximateHessian = TRUE)
```

"pre_adjustment" R object is a list of lists in which the user can retrieve input series, parameters and output series. For more details see chapter on R packages and rjd3highfreq help pages R, where all parameters are listed.

Decomposition

Extended X-11

X-11 is the decomposition module of X-13-Arima, the linearized series from the pre-adjustment step is split into seasonal (S), trend (T) and irregular (I) components. In case of multiple periodicities the decomposition is done periodicity by periodicity starting with the smallest one. Global structure of the iterations is the same as in "classical" X-11 but modifications were introduced for tackling non-integer periodicities. They rely on the Taylor approximation for the seasonal backshift operator:

$$B^{s+\alpha} \quad \cong \quad (1-\alpha)B^s + \alpha B^{s+1}$$

Modification of the first trend filter for removing seasonality

The first trend estimation is thanks to a generalization of the centred and symmetrical moving averages with an order equal to the periodicity p.

- filter length l: smallest odd integer greater than p

- examples: $p = 7 \to l = 7$, $p = 12 \to l = 13$, $p = 365.25 \to l = 367$, $p = 52.18 \to l = 53$
- central coefficients $1/p$ (1/12,1/7, 1/365.25)
- end-point coefficients $\mathbb{I}\{E(p) \text{ even}\} + (p - E(p))/2p$
- example for $p = 12$: (1/12 and 1/24) (we fall back on $M_{2\times 12}$ of the monthly case
- example for $p = 365.25$: $(1/365.25$ and $0.25/(2 * 365.25))$

Modification of seasonality extraction filters

Computation is done on a given period

Example $M_{3\times 3}$

$$M_{3\times 3}X = \frac{1}{9}(X_{t-2p}) + \frac{2}{9}(X_{t-p}) + \frac{3}{9}(X_t) + \frac{2}{9}(X_{t+p}) + \frac{1}{9}(X_{t+2p})$$

if p integer: no changes needed

if p non-integer: Taylor approximation of the backshift operator

Modification of final trend estimation filter

As seasonality has been removed in the first step, there is no non-integer periodicity issue in the final trend estimation, but extended X-11 offers additional features vs classic X-11, in which final trend is estimated with Henderson filters and Musgrave asymmetrical surrogates. In extended X-11, a generalization of this method with local polynomial approximation is available.

Example of decomposition

Here the raw series is daily and displays two periodicities $p = 7$ and $p = 365.25$

```
<<<<<<< HEAD
# extraction of day-of-the-week pattern (dow)
x11.dow <- rjd3x11plus::x11plus(y_linearized,
=======
# extraction of day-of-the-week (DOW) pattern
x11.dow <- rjd3highfreq::x11(y_linearized,
>>>>>>> 86a324e971dea668791bcfb00a39798e7f9ac253
        period = 7,                 # DOW pattern
```

```
        mul = TRUE,
        trend.horizon = 9,  # 1/2 Filter length : not too long vs p
        trend.degree = 3,                      # Polynomial degree
        trend.kernel = "Henderson",            # Kernel function
        trend.asymmetric = "CutAndNormalize",  # Truncation method
        seas.s0 = "S3X9", seas.s1 = "S3X9",    # Seasonal filters
        extreme.lsig = 1.5, extreme.usig = 2.5)  # Sigma-limits


<<<<<<< HEAD
# extraction of day-of-the-week pattern (doy)

x11.doy <- rjd3x11plus::x11plus(x11.dow$decomposition$sa,  # previous sa
=======
# extraction of day-of-the-year (DOY) pattern
x11.doy <- rjd3highfreq::x11(x11.dow$decomposition$sa,  # previous sa
>>>>>>> 86a324e971dea668791bcfb00a39798e7f9ac253
                  period = 365.2425,        # DOY pattern
                  mul = TRUE,
                  trend.horizon = 371, # 1/2 final filter length
                  trend.degree = 3,
                  trend.kernel = "Henderson",
                  trend.asymmetric = "CutAndNormalize",
                  seas.s0 = "S3X15", seas.s1 = "S3X5",
                  extreme.lsig = 1.5, extreme.usig = 2.5)
```

Arima Model Based (AMB) Decomposition (Extended Seats)

Example

```
# extracting DOY pattern
amb.doy <- rjd3highfreq::fractionalAirlineDecomposition(
  amb.dow$decomposition$sa,  # DOW-adjusted linearised data
  period = 365.2425,         # DOY pattern
  sn = FALSE,                # Signal (SA)-noise decomposition
  stde = FALSE,              # Calculate standard deviations
  nbcasts = 0, nfcasts = 0)  # Numbers of back- and forecasts
```

Summary of the process

For the time being, seasonal adjustment processing in rjd3highfreq cannot be encompassed by one function like for lower frequency, e.g rjd3x13::x13(y_raw)

The user has to run the steps one by one, here is an example with $p = 7$ and $p = 365.25$

- computation of the linearized series $Y_{lin} = ExtendedAirline(Y)$
- computation of the calendar corrected series Y_{cal}
- computation of S_7 by decomposition of the linearized series
- computation of $S_{365.25}$ by decomposition of the seasonally adjusted series with $p = 7$
- finally adjusted series $sa_{final} = Y_{cal}/S_7/S_{365.25}$ (if multiplicative model)

STL decomposition

Not currently available. Under construction.

State Space framework

Not currently available. Under construction.

Quality assessment

Residual seasonality

JDemetra+ provides the Canova-Hansen test in rjd3toolkit package which allows to check for any remaining seasonal periodicity in the final SA data.

Outlier detection and external regressors

Chapter's overview

The chapter describes

- how to generate useful external regressors for improving seasonal adjustment or reg-arima modelling
- JDemetra+ solutions for outlier detection and in a time series.
  These routines can be used stand alone or as part of a seasonal adjustment process. They can be accessed via GUI or R packages.

How to use the generated regressors, or any user-defined variable, in a seasonal adjustment or reg-arima modelling process is discussed in the relevant chapters on SA and SA of High-frequency data. There you will also find out how to fix the corresponding coefficients and how to allocate the effects to the selected component.

The external regressors described below are not meant for calendar correction which is detailed here

(ADD TABLE by data type)

External regressors using R packages vs GUI: quick contrasting

Oulier detection using R packages vs GUI: quick contrasting

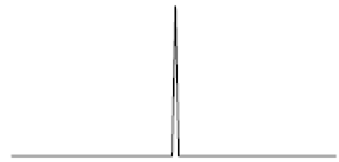
Generating external regressors

Outliers

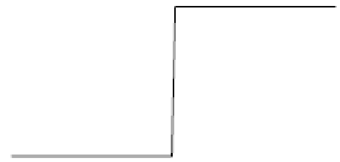
Outlier Types

The following outliers are available for automatic detection

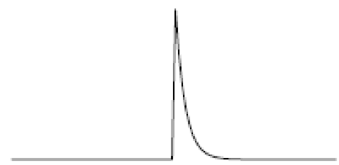
Additive outlier (AO)
Allocated at the end, after the decomposition, to the Irregular component.



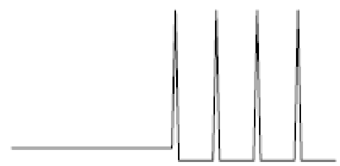
Level Shift (LS)
Allocated at the end, after the decomposition, to the Trend.



Transitory Change (TC)
Allocated at the end, after the decomposition, to the Irregular component.



Seasonal Outlier (SO)



Very rare, not automatically detected by default in JDemetra+.

Allocated at the end, after the decomposition to the Seasonal component.

ADD: - specifics for HF data (wo outiler)

Pre-specifying outliers

Outliers are well-defined types of auxiliary variables, therefore when they are used (reg-arima or tramo modelling) they don't need to be explicitly generated beforehand. Pre-specifying outliers is detailed in chapters on SA and SA of High-frequency data.

ADD:

- in v2 fixing coefficients impossible (window displayed as a calendar)
- in v3 calendar window or different window, allowing to fix coefficients

Generating regressors for outliers

Nevertheless, explicit regressors corresponding to outliers can be generated with `rjd3toolkit` functions for independent use. Further details rjd3toolkit help pages.

```
#Outliers in February 2002, for monthly data
library("rjd3toolkit")
ao <- ao_variable(frequency=12, c(2000,1), length = 12*4, date = "2002-02-01")
ls <- ls_variable(12, c(2000,1), length = 12*4, date = "2002-02-01")
tc <- tc_variable(12, c(2000,1), length = 12*4, date = "2002-02-01")
so <- so_variable(12, c(2000,1), length = 12*4, date = "2002-02-01")
```

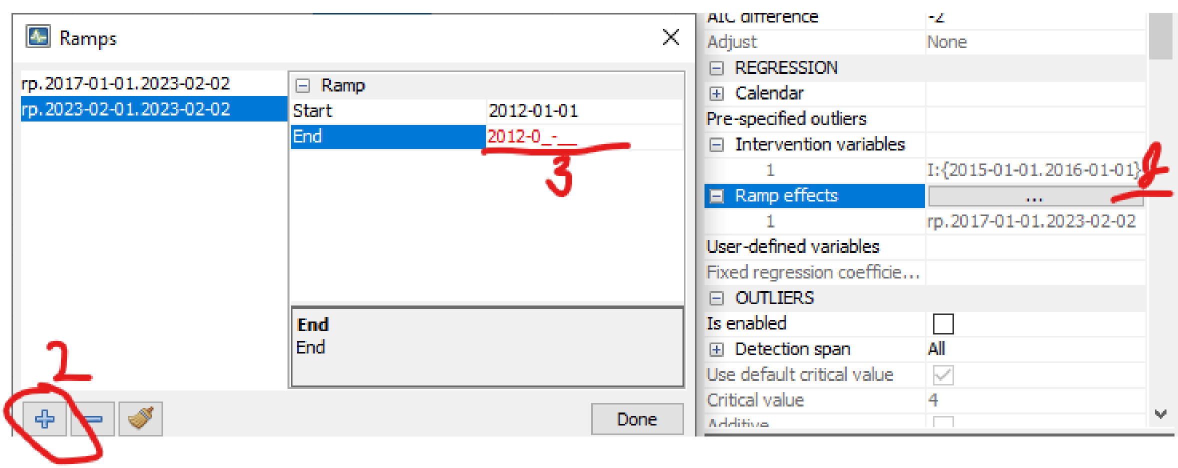
Ramps

A ramp effect means a linear increase or decrease in the level of the series over a specified time interval t_0 to t_1. Ramps can overlap other ramps, additive outliers and level shifts. In seasonal adjustment their effected will be allocated to the trend.

Adding ramps to a seasonal adjustment (or reg-arima/tramo) specification happens in one step in GUI as well as in R, where ramp regressors can nevertheless be independently generated.

Adding ramps in GUI

In the specification window



The effect of the ramps is stored in reg_t pre-adjustment series.

Adding ramps in R

Use the function `add_ramp`

```
# create a specification from a default specification
init_spec <- rjd3x13::spec_x13("RSA5c")

# add ramp on year 2012
new_spec <- rjd3toolkit::add_ramp(init_spec,start="2012-01-01",end="2012-12-01")
```

Generating ramp regressors in R

Use `ramp_variable` function in rjd3toolkit:

```
?ramp_variable
# Ramp variable from January 2001 to September 2001 for a monthly series
rp <- ramp_variable(frequency=12, c(2000,1), length = 12*4, range = c(13, 21))
# Or equivalently
rp <- ramp_variable(12, c(2000,1), length = 12*4, range = c("2001-01-01", "2001-09-02"))
plot.ts(rp)
```

More details rjd3toolkit pages.

Intervention variables

Intervention variables are modelled as any possible sequence of ones and zeros, on which differencing (regular and seasonal) can be applied.

Adding intervention variables to a seasonal adjustment (or reg-arima/tramo) specification happens in one step when using the GUI, whereas two steps are required in R: generating the regressors and the adding them as an user-defined variable.

Adding intervention variables in GUI

step 1:

Step 2:

Generating intervention variables in R

Using `intervention_variable` function in rjd3toolkit

```
library("rjd3toolkit")
? intervention_variable
iv<-intervention_variable(frequency=12, start=c(2000, 1), length=60,
                      starts = "2001-01-01", ends = "2001-12-01")
iv
plot(iv)

iv<-intervention_variable(12, c(2000, 1), 60,
                      starts = "2001-01-01", ends = "2001-12-01", delta = 1)
iv
plot(iv)

iv<-intervention_variable(12, c(2000, 1), 60,
                          starts = "2001-01-01", ends = "2001-12-01",
                          delta =0, seasonaldelta=1)
iv
plot(iv)
```

More details rjd3toolkit help pages.

Adding intervention variables in R

Intervention variables can be added to a specification like any other external regressor using the `add_usrdefvar`. They also need to be declared in a “context” using the `modelling_context` function.

```
# creating one or several external regressors (TS objects),
# which will be gathered in one or several groups
iv1<-intervention_variable(12, c(2000, 1), 60,
starts = "2001-01-01", ends = "2001-12-01")
iv2<- intervention_variable(12, c(2000, 1), 60,
starts = "2001-01-01", ends = "2001-12-01", delta = 1)
# regressors as a list of two groups (lists) reg1 and reg2
vars<-list(reg1=list(iv1 = iv1),reg2=list(iv2 = iv2) )
# to use those regressors, input : name=reg1.iv1 and name=reg2.iv2 in add_usrdefvar functi
# creating the modelling context
my_context<-modelling_context(variables=vars)
```

```
# customize a default specification
init_spec <- rjd3x13::spec_x13("RSA5c")
# regressors have to be added one by one
new_spec<- add_usrdefvar(init_spec,name= "reg1.iv1", regeffect="Trend")
new spec<- add_usrdefvar(new_spec,name = "reg2.iv2", regeffect="Trend", coef=0.7)
# modelling context is needed for the estimation phase
# raw series
y<-rjd3toolkit::ABS$X0.2.09.10.M
sa_x13<- rjd3x13::x13(y, new_spec, context = my_context)
```

Periodic dummies and contrasts

Generating regressors in R

dummies :as many time series as type of periods in a year (4,12)

```
## periodic dummies : add explanations and examples
p<-periodic.dummies(4, c(2000,1), 60)
head(p)
class(p)
q<-periodic.contrasts(4, c(2000,1), 60)
q[1:9,]
```

Trigonometric variables

Correction for stable seasonality.

Generating in R

Outlier Detection

With Reg Arima models

Within an SA processing

In a seasonal adjustment estimation or reg-arima modelling outliers are detected by default. This process can be customized by selecting the type of outliers to be taken into account and the critical values to be used for selection. See the relevant chapters on SA and SA of High-frequency data

Stand alone

R packages rjd3x13 and rjd3tramoseats provide functions for detecting outlier with reg-arima (tramo) algorithms.

using `regarima_outliers`in rjd3x13:

```
?regarima_outliers
regarima_outliers(rjd3toolkit::ABS$X0.2.09.10.M, order=c(1,1,1), seasonal=c(0,1,1),
                  mean=F,
                  X=NULL, X.td=NULL,
                  ao=T, ls=F, tc=T, so=T, cv=4)
```

ADD example wit rjd3tramoseats::tramo_outliers

Specific TERROR tool

With structural models (BSM)

Calendar correction

Chapter's overview

This chapter is divided in two parts. The first one (theory) outlines the rationale for correcting calendar correction and the underlying modelling. The second part (practice) describes how relevant regressors for calendar correction are built in JDemetra+.

As calendar effects are deterministic, they can be corrected with a regression model. In the algorithms X13-Arima and Tramo-Seats it boils down to adding suitable regressors to the reg-arima modelling (pre-adjustment) phase. This chapter will describe how to generate a set of regressors corresponding to the desired correction, which will happen according to the following steps:

- step 1: generate a calendar (usually national calendar of interest). If this step is skipped a default calendar, not taking into account country-specific holidays will be used.
- step 2: generate regressors based on the above defined calendar

Regressors will have the same frequency as the raw data, thus an aggregation process will be defined unless the data is daily.

- step 2b: a specific variable for modelling the easter effect (or any other moving holiday effect like Ramadan) can also be defined

Most of the functions are designed for quarterly and monthly data. What applies to daily and weekly data will be highlighted.

Regressors are corrected for deterministic seasonality through a long-term mean correction (see below)

- step 3: these regressors have to be plugged-in in pre-adjustment phase of a seasonal adjustment estimation. How to do this is detailed in chapters on SA or SA of HF data.

How to add other types of regressors is described here

Calendar correction using R packages vs GUI: quick contrasting

- is group and reference day choice as flexible in GUI as in R ?
- in GUI (v3) automatic detection, what in R ?
- are the same data frequencies available in R and in GUI (divisors of 12 and…)
- calendar correction HF data: R vs GUI ? including weekly data

Rationale for Calendar correction

A calendar is heterogeneous, it at least composed of:

- trading days: days usually worked, taking into account the company's sector. (Most frequently Mondays through Fridays when not bank holidays).
- week-ends
- bank holidays

For a given year as well as throughout the years, every month doesn't have the same number of days per day-type, which implies that all months/quarters aren't "equal", even for a given type of month or quarter. This causes **calendar effects** which have to be removed to allow sounder comparisons following the same principle as seasonality correction.

Two types of effects result from this heterogeneity:

- length of period (month/quarter) (leap-year or direct correction)
- composition of period (type of day)

This second effect is also relevant for daily (and weekly) data.

An additional easter effect can be modelled, as for some series, variations linked to Easter can be seen over a few days prior or following Easter. For example, flowers and chocolate sales might rise significantly as Easter approaches. (in practice this effect is very rare, it is better to deactivate by default detection)

Modelling calendar effects

Regression Model for type of days

For each period t, the days are divided in K groups $\{D_{t1}, \dots, D_{tK}\}$.

The groups of days can be anything (trading days, working days, Sundays + holidays assimilated to Sundays…) ADD

We write $N_t = \sum_1^K D_{ti}$, the number of days of the period t

Two terms appear:

- the specific effect of a type of day i as a contrast between the number of days i and the number of Sundays and bank holidays
- the effect of the month's (or period's) length.

Once seasonally adjusted, this term comes down to the leap year effect:

- for all months except Februaries $\bar{N}_t = N_t$
- for Februaries $\bar{N}_t = 28.25$ and $N_t = 28$ or $N_t = 29$

The effect of one day of the group i is measured by α_i, so that the global effect of the group i for the period t is $\alpha_i D_{ti}$

The global effect of all the days for the period t is

$$\sum_{i=1}^{K} \alpha_i D_{ti} = \bar{\alpha} N_t + \sum_{i=1}^{K} (\alpha_i - \bar{\alpha}) D_{ti}$$

where $\bar{\alpha} = \sum_{i=1}^K w_i \alpha_i$ with $\sum_{i=1}^K w_i = 1$

So,

$$\sum_{i=1}^{K} (\alpha_i - \bar{\alpha}) w_i = \sum_{i=1}^{K} \alpha_i w_i - \bar{\alpha} \sum_{i=1}^{K} w_i = 0$$

LEAP YEAR part to comment

We focus now on $\sum_{i=1}^K (\alpha_i - \bar{\alpha}) D_{ti}$, the actual trading days effects (excluding the length of period effect).

Writing $\alpha_i - \bar{\alpha} = \beta_i$ and using that $\sum_{i=1}^K \beta_i w_i = 0$, we have that

$$\sum_{i=1}^{K}\beta_i D_{ti} = \sum_{i=1}^{K}\beta_i(D_{ti} - \frac{w_i}{w_K}D_{tK}) = \sum_{i=1}^{K-1}\beta_i(D_{ti} - \frac{w_i}{w_K}D_{tK})$$

Note that the relationship is valid for any set of weights w_i. It is also clear that the contrasting group of days can be any group:

$$\sum_{i=1}^{K-1}\beta_i(D_{ti} - \frac{w_i}{w_K}D_{tK}) = \sum_{i=1}^{K,i\neq J}\beta_i(D_{ti} - \frac{w_i}{w_J}D_{tJ})$$

The "missing" coefficient is easily derived from the others:

$$\beta_K = -\frac{1}{w_K}\sum_{i=1}^{K-1}\beta_i w_i$$

Correction for deterministic seasonality

In the case of seasonal adjustment, we further impose that the regression variables don't contain deterministic seasonality. That is achieved by removing from each type of period (month, quarter…) its long term average. We write $\bar{D_i^y}$ the long term average of the yearly number of days in the group i and $\bar{D^y_{i,J}}$ the long term average of the number of days in the group i for the periods J (for instance, average number of Mondays in January…).

The corrected contrast for the time t belonging to the period J is:

$$C_{ti} = D_{ti} - \bar{D^y_{i,J}} - \frac{w_i}{w_K}(D_{tK} - \bar{D^y_{K,J}})$$

How is the long term mean computed? Probabilistic approach (more on this soon)

Weights for different groups of days

We can define different sets of weights. The usual one consists in giving the same weight to each type of days. w_i is just proportional to the number of days in the group i. In the case of "week days", $w_0 = \frac{5}{7}$ (weeks) and $w_1 = \frac{2}{7}$ (week-ends). In the case of "trading days", $w_i = \frac{1}{7}$ … Another approach consists in using the long term yearly averages, taking into account the actual holidays. We get now that $w_i = \frac{\bar{D_i^y}}{365.25}$.

After the removal of the deterministic seasonality, the variables computed using the two sets of weights considered above are very similar. In the case of the "trading days", the difference for the time t, belonging to the period J, and for the day i with contrast K is $(1-\frac{w_i}{w_K})(D_{tK}-\bar{D^y_{K,J}})$, which is usually small. By default, JD+ uses the first approach, which is simpler. The second approach is implemented in the algorithmic modules, but not available through the graphical interface.

Use in RegArima models

In the context of RegArima modeling, we can also observe that the global effect of the trading days doesn't depend neither on the used weights (we project on the same space) nor on the contrasting group (see above) nor on the long term corrections (removed by differencing).

The estimated coefficients slightly change if we use different weights (not if we use a different contrasting group). It must also be noted that the choices affect the T-Stat of the different coefficients (not the joint F-Test), which can lead to other solutions when those T-Stats are used for selecting the regression variables (Tramo). Considering that the leap year/length of period variable is nearly independent of the other variables, the test on that variable is not very sensitive to the various specifications.

Interpretation

The use of different specifications of the trading days doesn't impact the final results (except through some automatic selection procedure). It just (slightly) changes the way we interpret the estimated coefficients.

Easter effect

Stock series

Generating Regressors for calendar correction

The following parts details how to build customized regressors for calendar correction using

- graphical user interface (GUI)
- rjd3toolkit package.

To take specific holidays into account a calendar has to be defined, regressors will be built subsequently.

As regressors have the same data frequency as the input series, several cases:

- for daily series : regressors are dummies representing each holidays

- for weekly, monthly and quarterly series regressors are aggregated indicators, the way of grouping different types of days and holidays has to be specified.

In GUI

Creating calendars

The customized calendar can be directly linked to the calendar correction option in GUI while specifying a seasonal adjustment process. See chapters on SA or SA of HF data.

0.0.0.0.1 * Default Calendar

In the graphical user interface, calendars in are stored in the *Workspace* window in the *Utilities* section. In the default calendar, country-specific national holidays are not taken into account, it reflects only the usual composition of the weeks in the calendar periods.

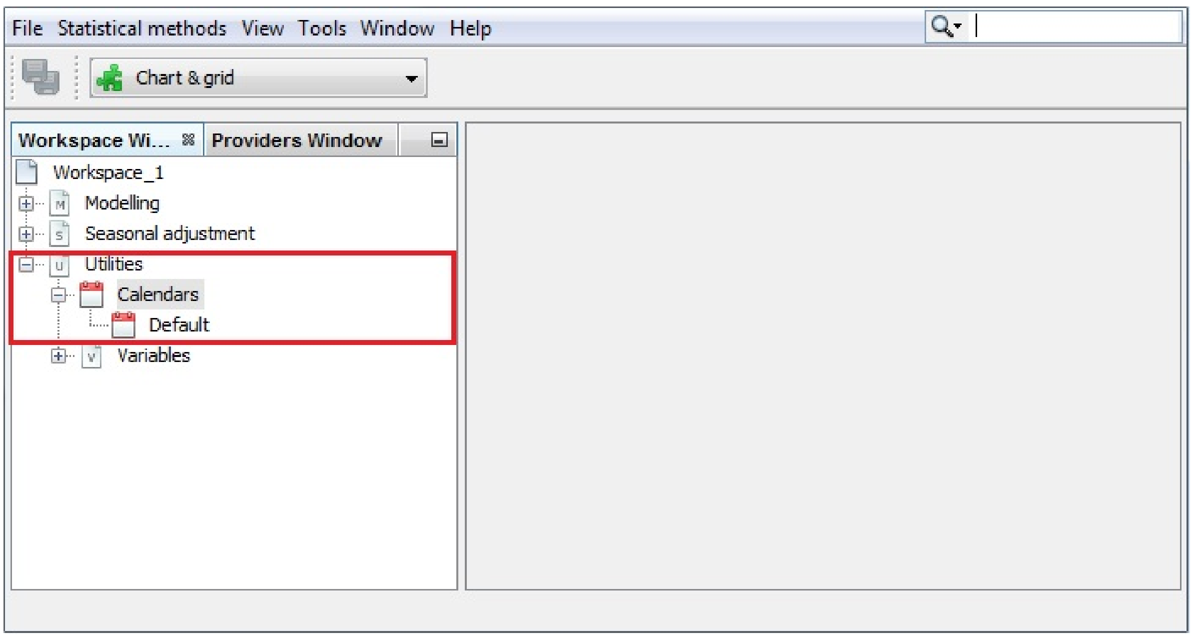


Figure 7: Text

To view the details of the default calendar: double click on it

0.0.0.0.2 * Set Properties

In the *Properties* panel the user can set:

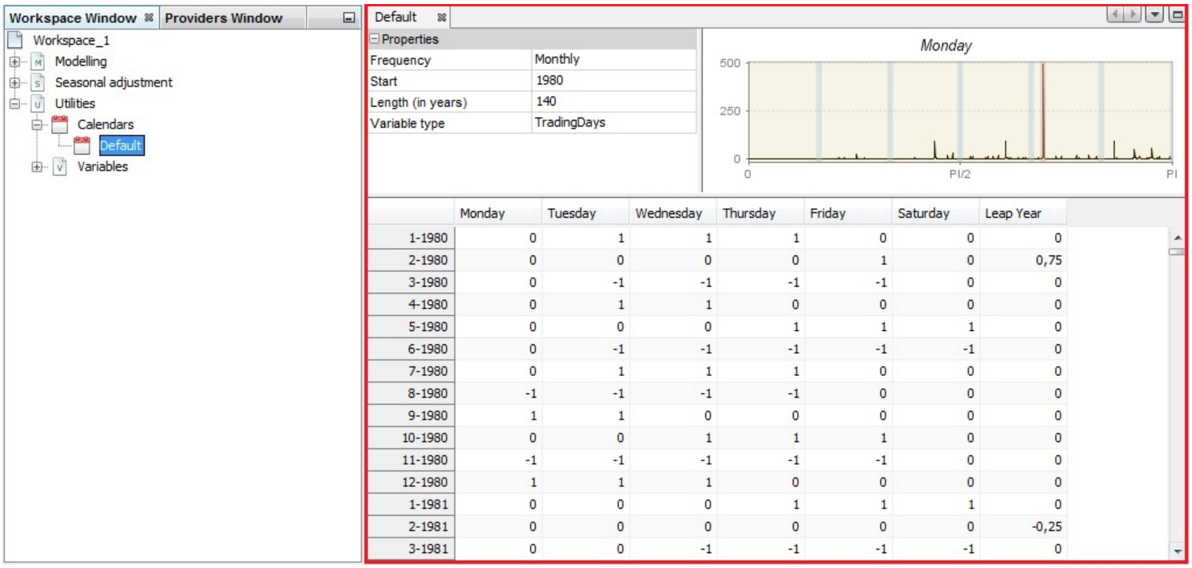


Figure 8: Text

- Frequency (monthly, quarterly..)
- Trading days or working days regressors

Trading days: 6 contrast variables

number of Mondays - number of Sundays

and one regressor for the leap year effect.

Working Days: 1 contrast variable ($number\ of\ workingdays(mondaytofriday) - number\ of\ SaturdaysandSundays,...$) and one regressor for the leap year effect.

Modification of the initial settings for the Default calendar

0.0.0.0.3 * Spectrum visualization

The top-right panel displays the spectrum for the given calendar variable. By default, the first variable from the table is shown.

- To change it, click on the calendar variable header.

Calendar variables shouldn't have a peak neither at a zero frequency (trend) nor the seasonal frequencies.

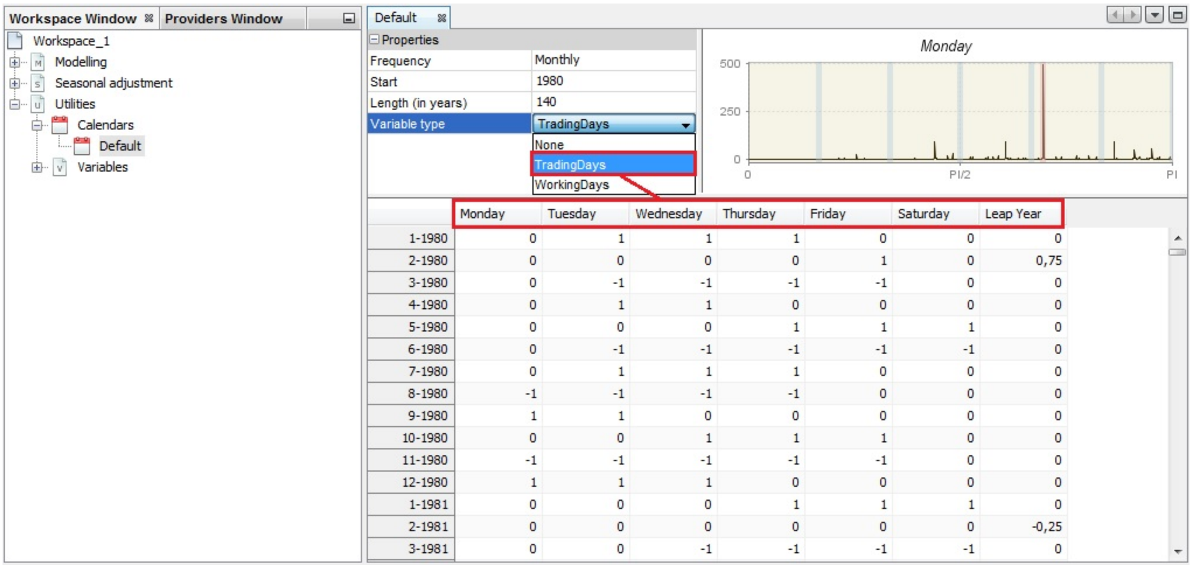


Figure 9: Text

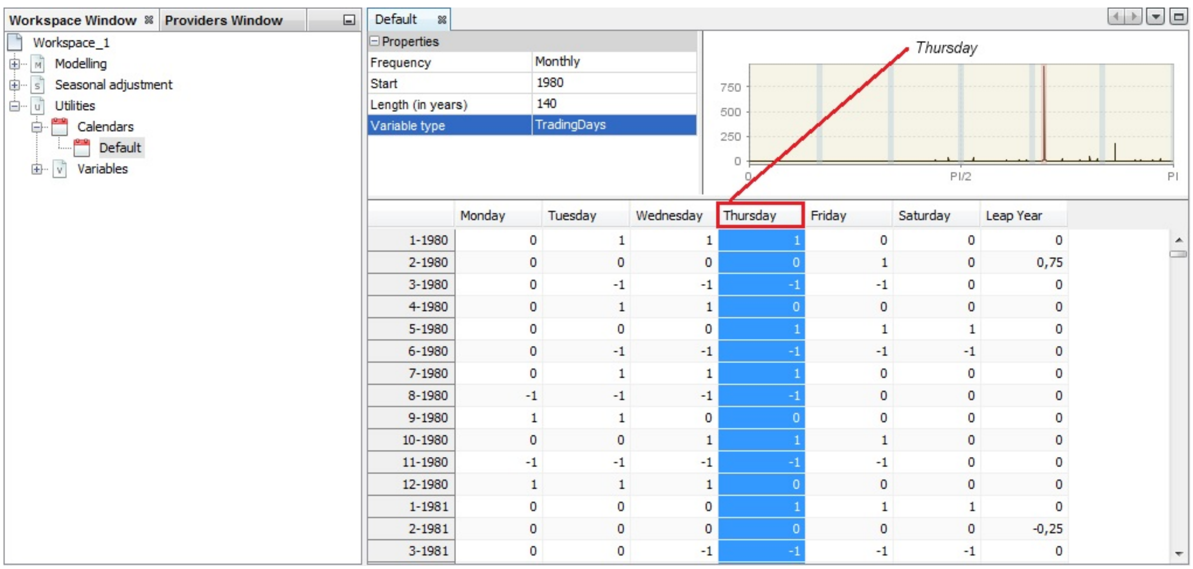


Figure 10: Text

Modify an existing Calendar

- click the option *Edit* from the context menu
- the list of holidays defined for this calendar is displayed

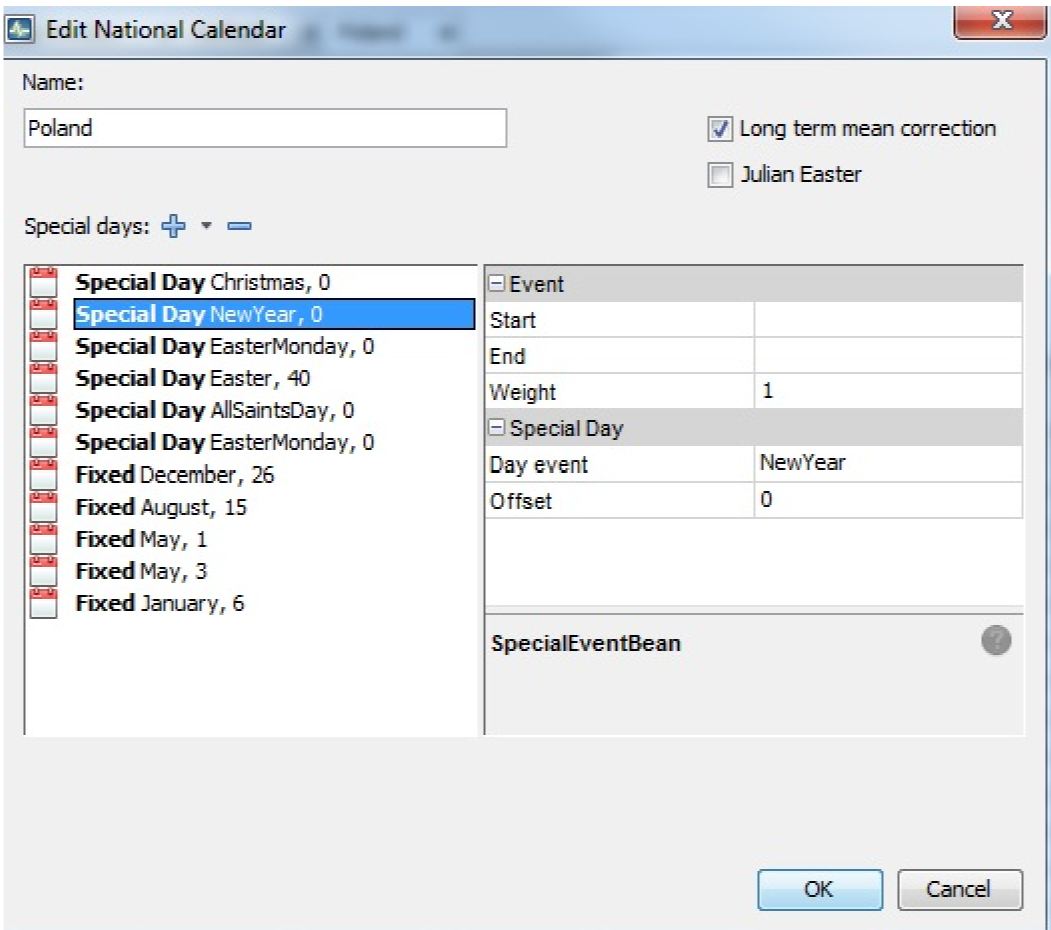


Figure 11: Text

Edit a calendar window

- To add a holiday unfold the + menu
- To remove a holiday click on it and choose the - button

Removing a holiday from the calendar

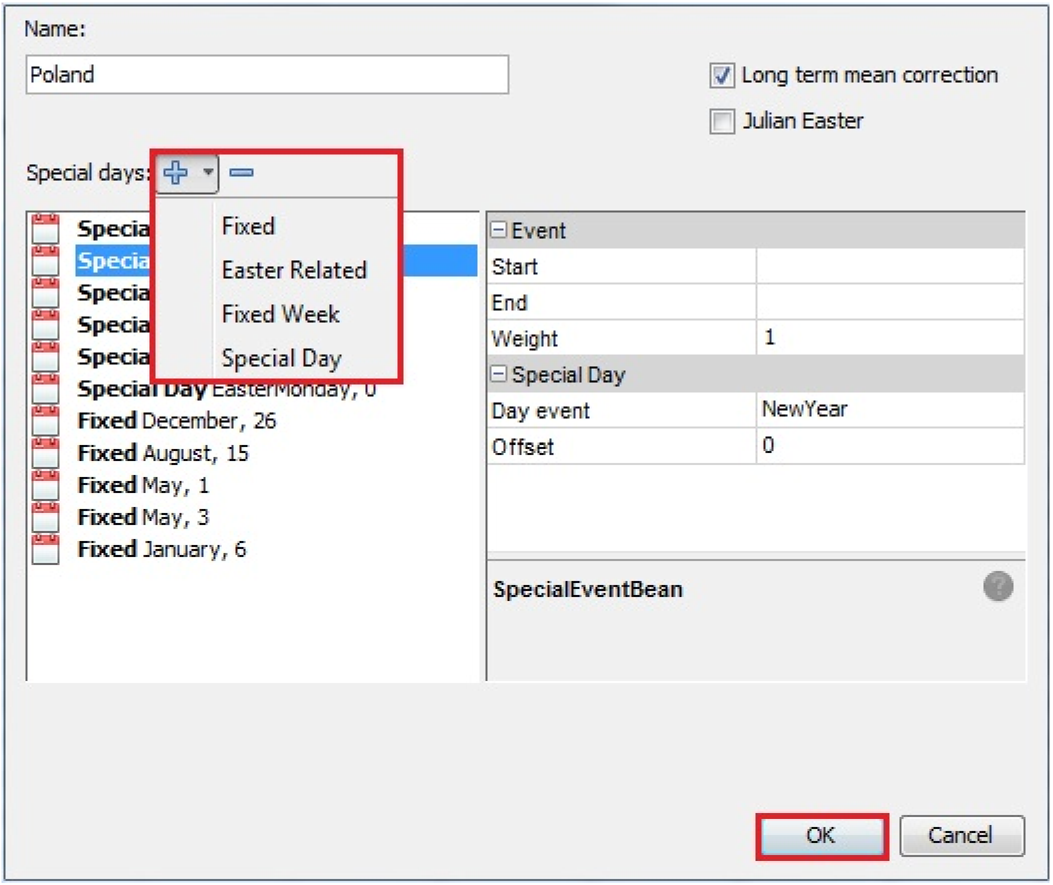


Figure 12: Text

0.0.0.0.1 * Creating a new calendar

An appropriate calendar, containing the required national holidays, needs to be created to adjust a series for country-specific calendar effects.

- right click on the *Calendar* item from the *Workspace* window and choose **Add**

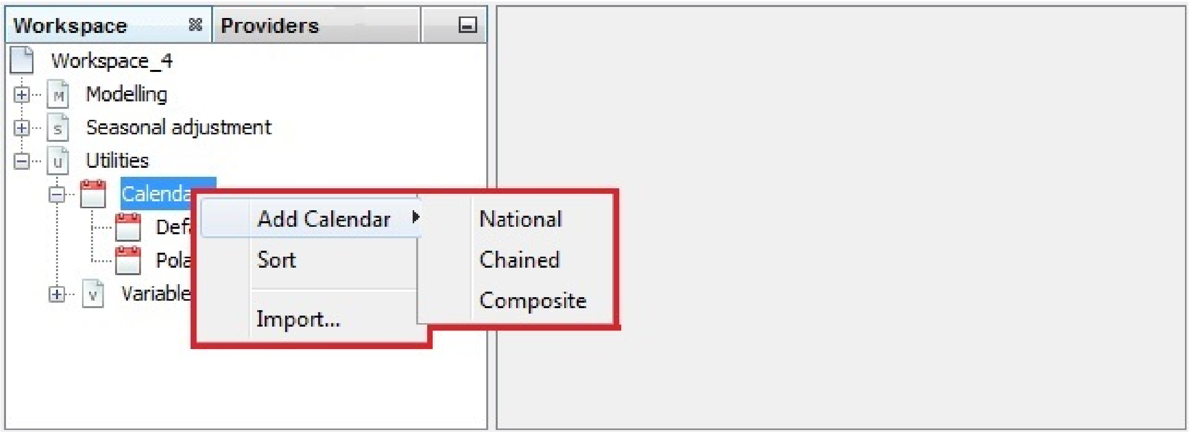


Figure 13: Text

Three options are available:

- *National calendars*: allows to include country-specific holidays
- *Composite calendars* : creates calendar as a weighted sum of several national calendars
- *Chained calendars* : allows to chain two national calendars before and after a break

0.0.0.0.2 * National Calendar

To define a national calendar: right click on Calendar item in the Utility panel of the workspace window

- To add a holiday unfold the + menu
- To remove a holiday from the list click on it and choose the **-** button.

Four options are available here:

```
- **Fixed** : holiday occurring at the same date

- **Easter Related**: holiday that depends on Easter Sunday date

- **Fixed Week**: fixed holiday that always falls in a specific week of a given month
```

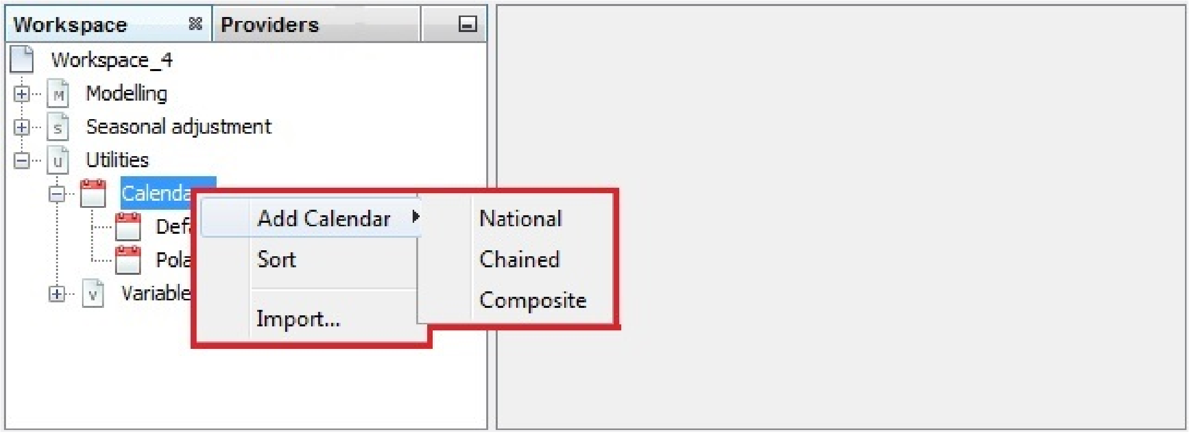


Figure 14: Text

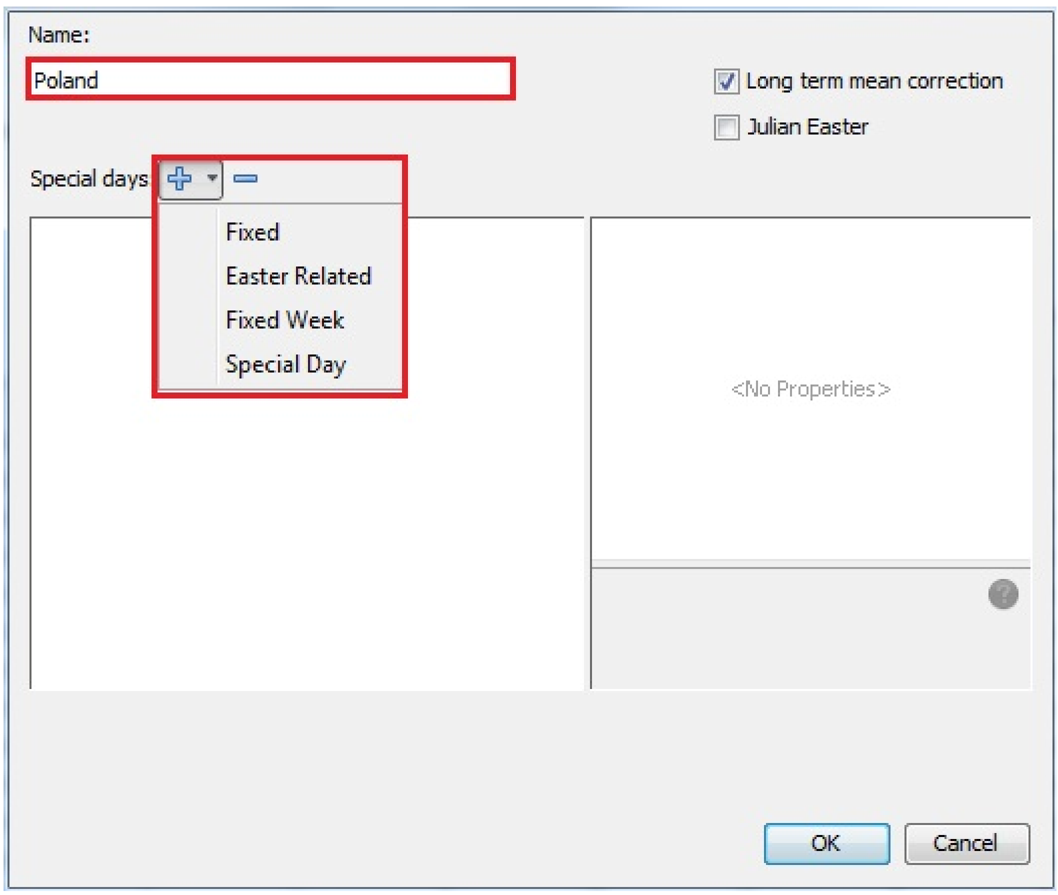


Figure 15: Text

- **Special Day**: choose a holiday from a list of pre-defined holidays (link to table)

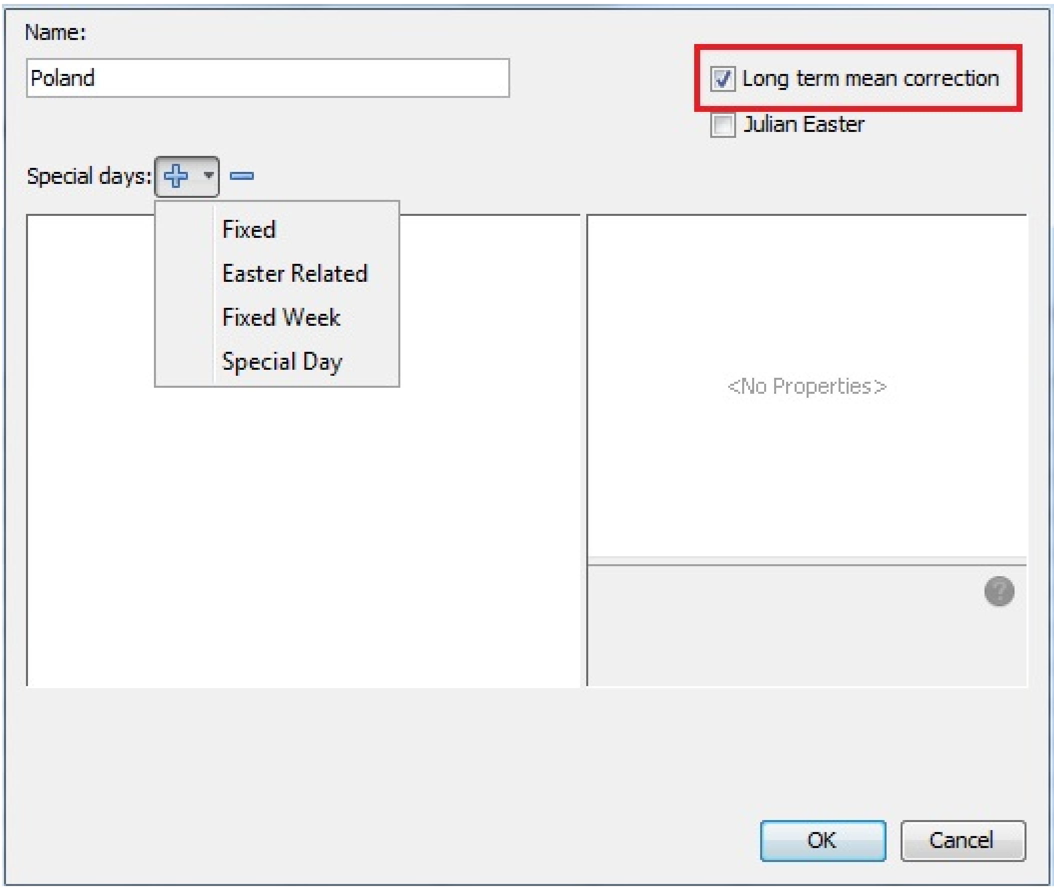


Figure 16: Text

- to use Julian Easter

To add a holiday from this list to the national calendar, choose the *Special day* item from the *Special days* list.

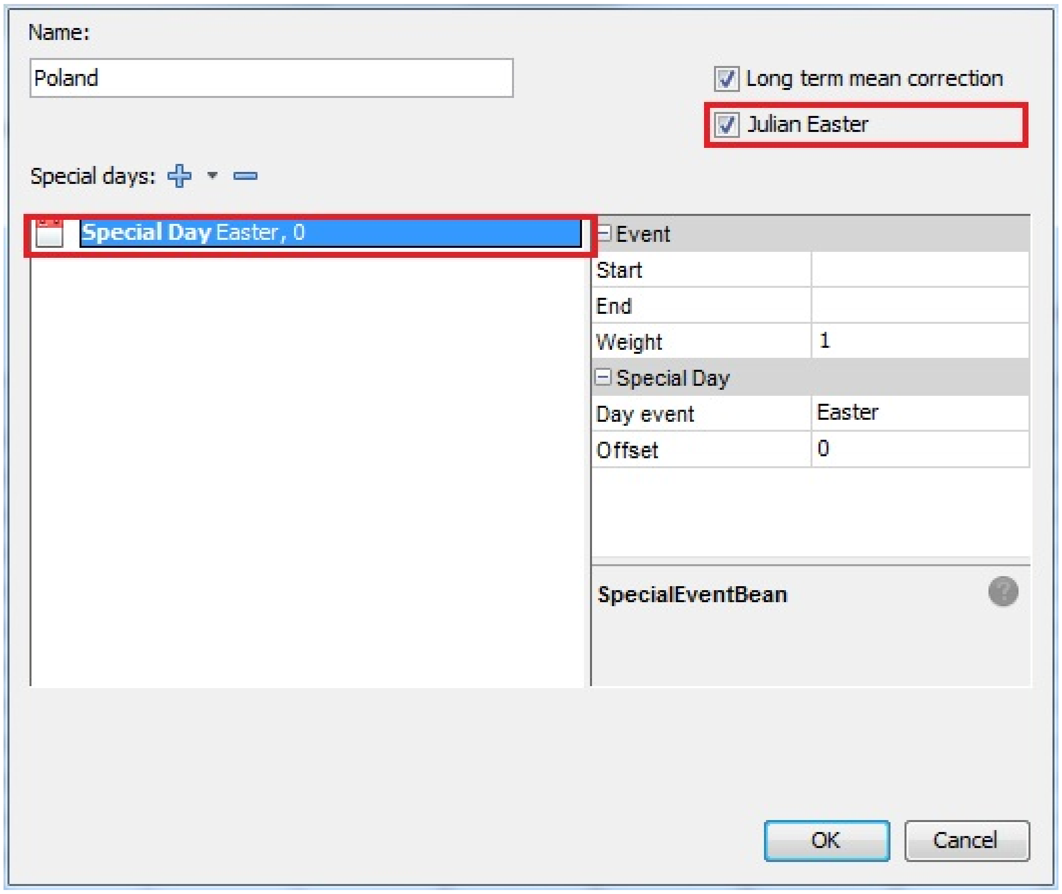
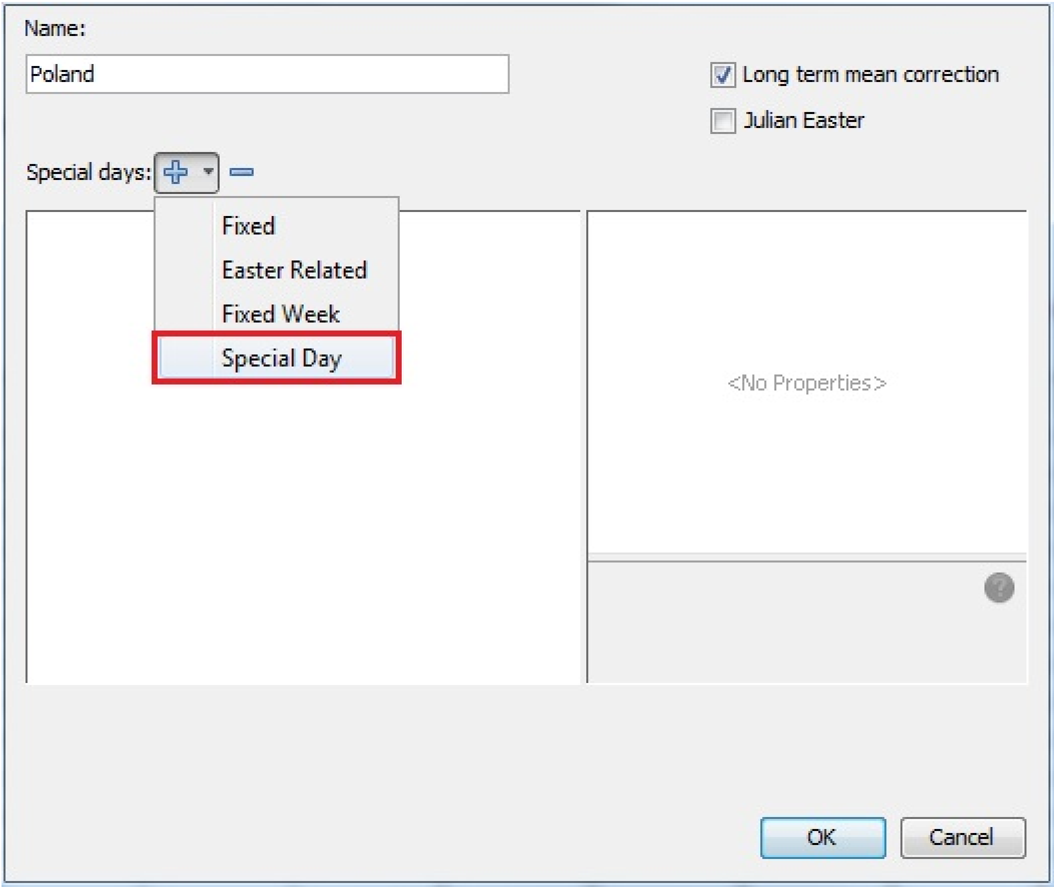


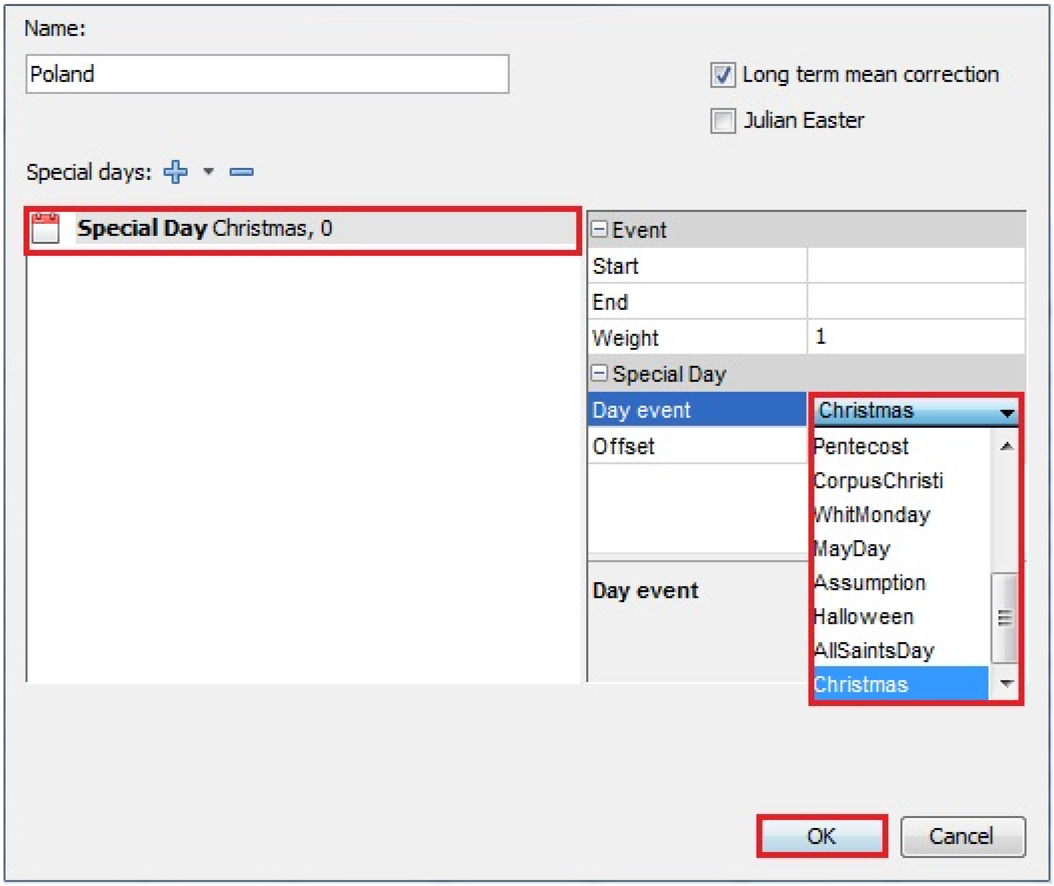
Figure 17: Text



Adding a pre-defined holiday to the calendar

By default, when the *Special Days* option is selected, JDemetra+ always adds *Christmas* to the list of selected holidays. The user can change this initial choice by specifying the settings in the panel on the right and clicking *OK*. The settings that can be changed include:

- **Start**: starting date for the holiday (expecting *yyyy-mm-dd*) Default is the starting date of the calendar (empty cell).
- **End**: same as start
- **Weight** : specifies the impact of the holiday on the series. The default weight is 1 (full weight) assuming that the influence of the holiday is the same as a regular Sunday. If less the a value between 0 and 1 can be assigned.
- **Day event**: a list of pre-defined holidays (link to table)
- **Offset**: allows to set a holiday as related to a pre-specified holiday by specifying the distance in days (e.g Easter Sunday). Default offset is 1. It can be positive or negative. Positive offset: defines a holiday following the pre-specified holiday. Negative offset: defines a holiday preceding the selected pre-specified.



Choosing a pre-defined holiday from the list

- To define a fixed holiday not included in the list of pre-defined holidays:
  - choose *Fixed* from the *Special days* list: by default January, 1 is displayed. Specify the settings:
- **Start**: starting date for the holiday (expecting *yyyy-mm-dd*) Default is the starting date of the calendar (empty cell).
- **End**: same as start
- **Weight** : specifies the impact of the holiday on the series. The default weight is 1 (full weight) assuming that the influence of the holiday is the same as a regular Sunday. If less the a value between 0 and 1 can be assigned.
- **Day**: day of month when the fixed holiday is celebrated.
- **Month**: month, in which the fixed holiday is celebrated.

Options for a fixed holiday

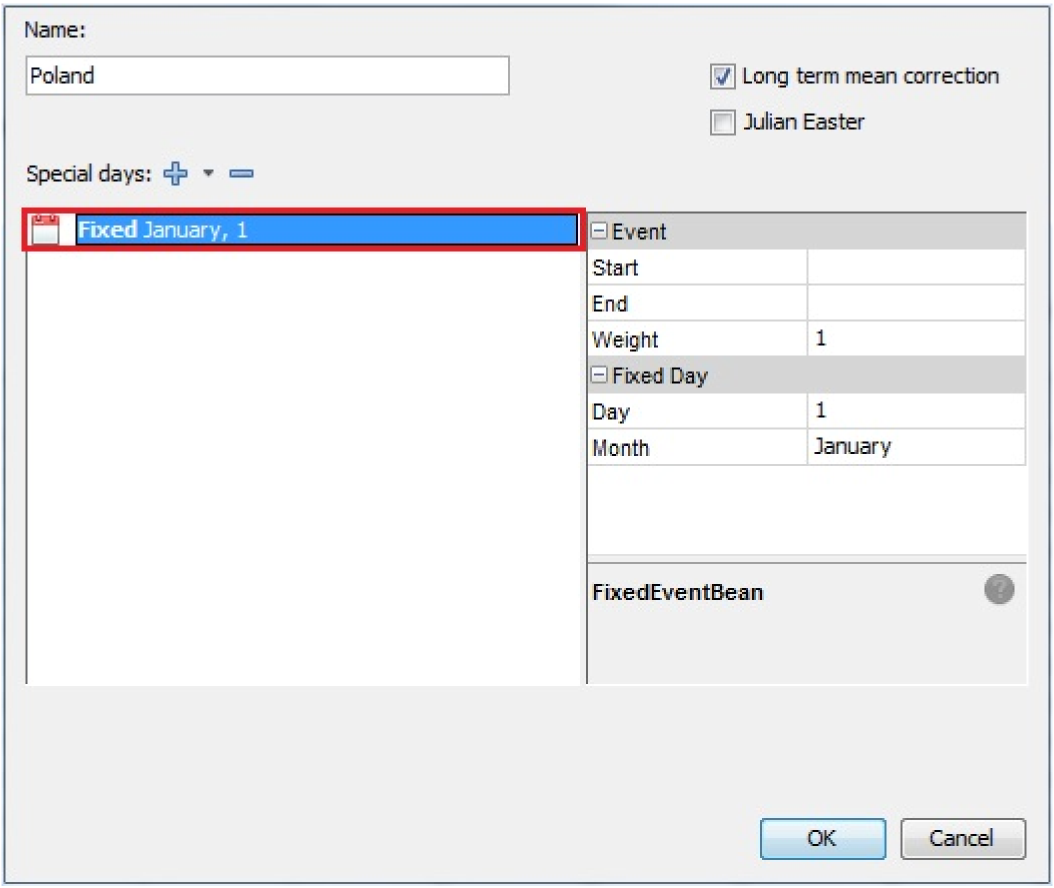
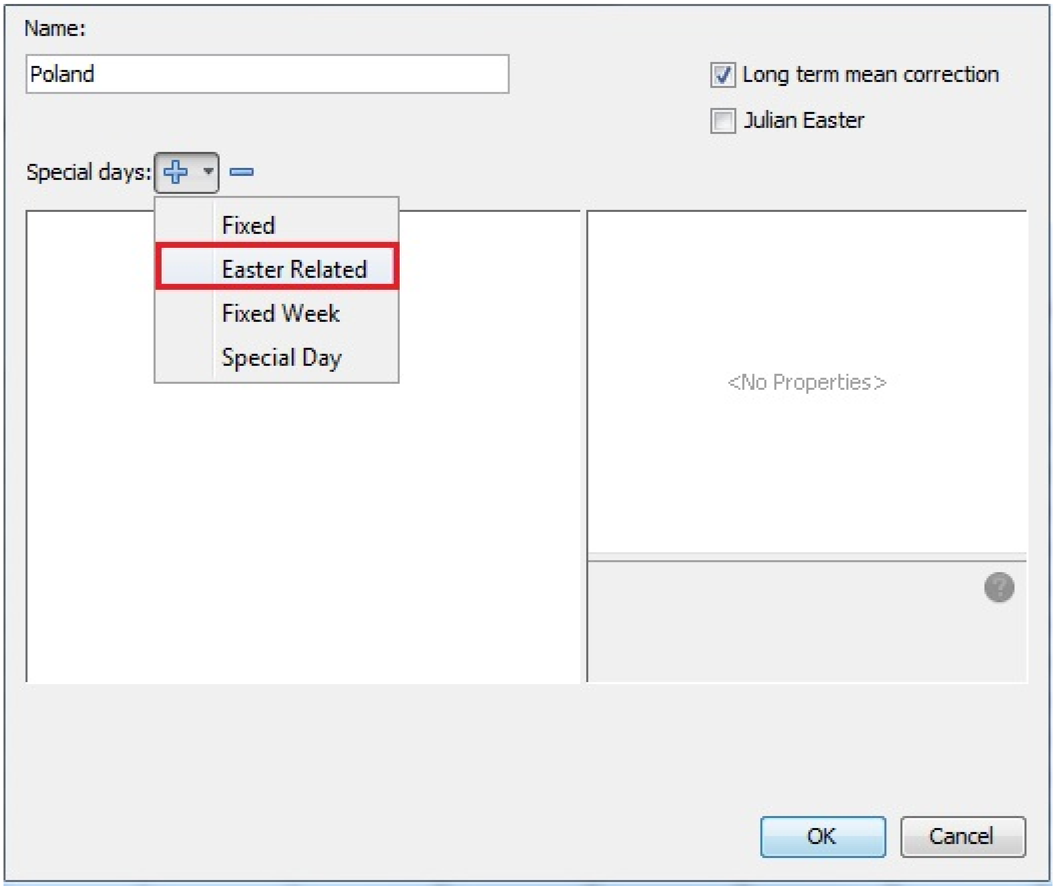


Figure 18: Text

- Add *Corpus Christi*: example of an Easter related holiday not included in the special day list(link to table). It is is a moving holiday celebrated 60 days after Easter
  - choose the *Easter related* item from the *Special days* list.



By default *Easter + 1* is displayed. Setting can be changed :

- **Start**: starting date for the holiday (expecting *yyyy-mm-dd*) Default is the starting date of the calendar (empty cell).
- **End**: same as start
- **Weight** : specifies the impact of the holiday on the series. The default weight is 1 (full weight) assuming that the influence of the holiday is the same as a regular Sunday. If less the a value between 0 and 1 can be assigned.
- **Offset**: To define Corpus Christi enter **60**, as it is celebrated 60 days after Easter Sunday.
- Fixed week option: when dealing with holidays occurring on the same week of a given month. Example: Labour Day in the USA and Canada, celebrated on the first Monday of September in Canada

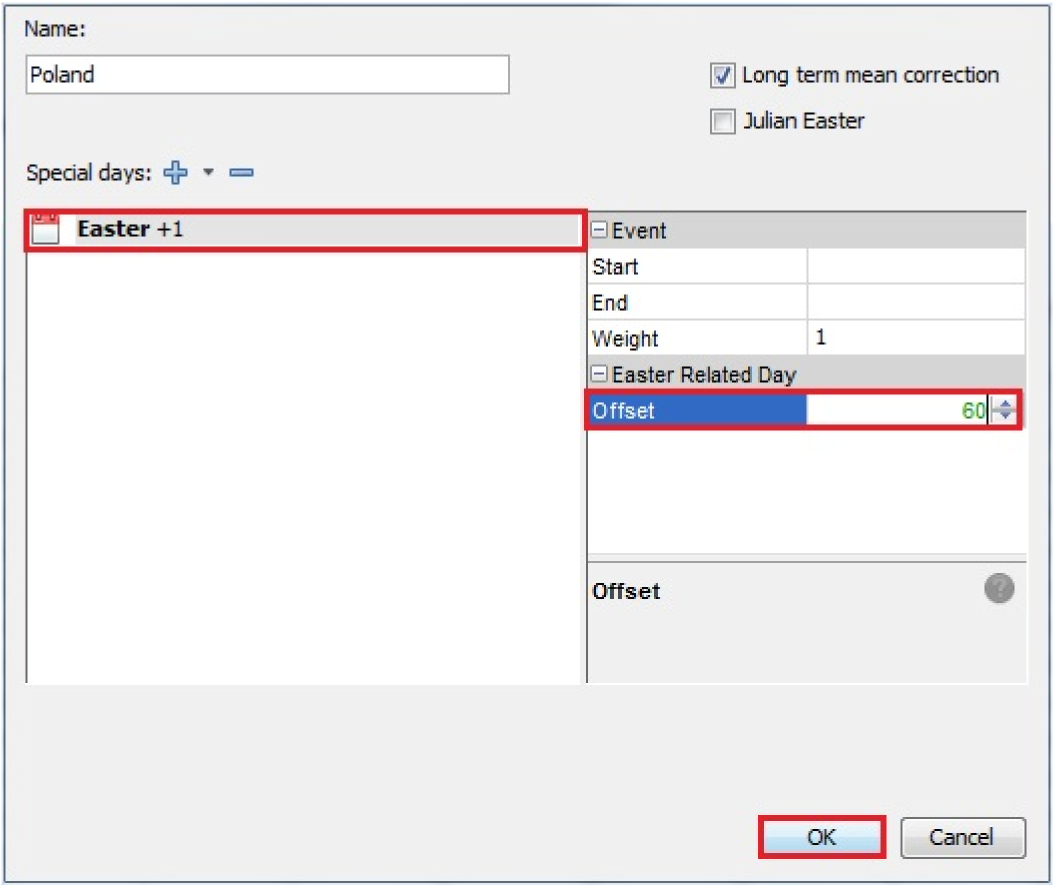


Figure 19: Text

– choose *Fixed Week* from the *Special days* list.

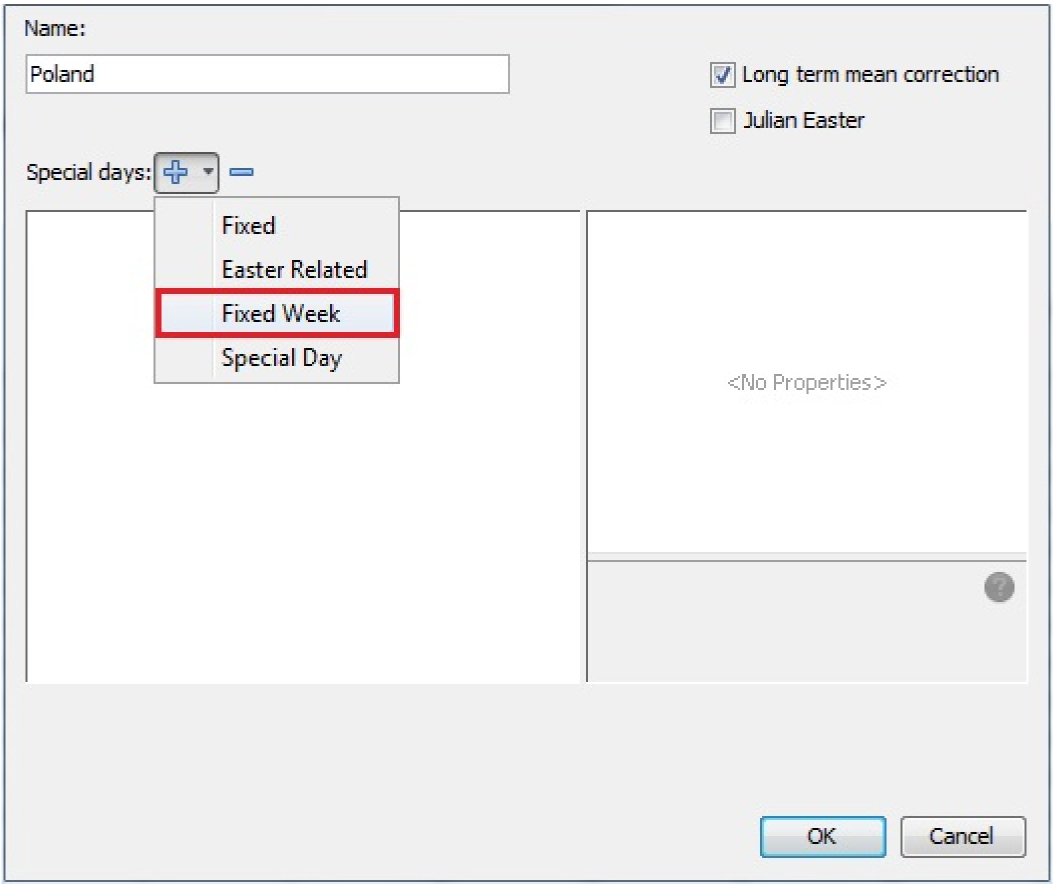


Figure 20: Text

Available settings are:

- **Start**: starting date for the holiday (expecting *yyyy-mm-dd*) Default is the starting date of the calendar (empty cell).
- **End**: same as start
- **Weight** : specifies the impact of the holiday on the series. The default weight is 1 (full weight) assuming that the influence of the holiday is the same as a regular Sunday. If less the a value between 0 and 1 can be assigned
- **Day of Week**: day of week when the holiday is celebrated each year
- **Month**: month, in which the holiday is celebrated each year
- **Week**: number denoting the place of the week in the month: between 1 and 5

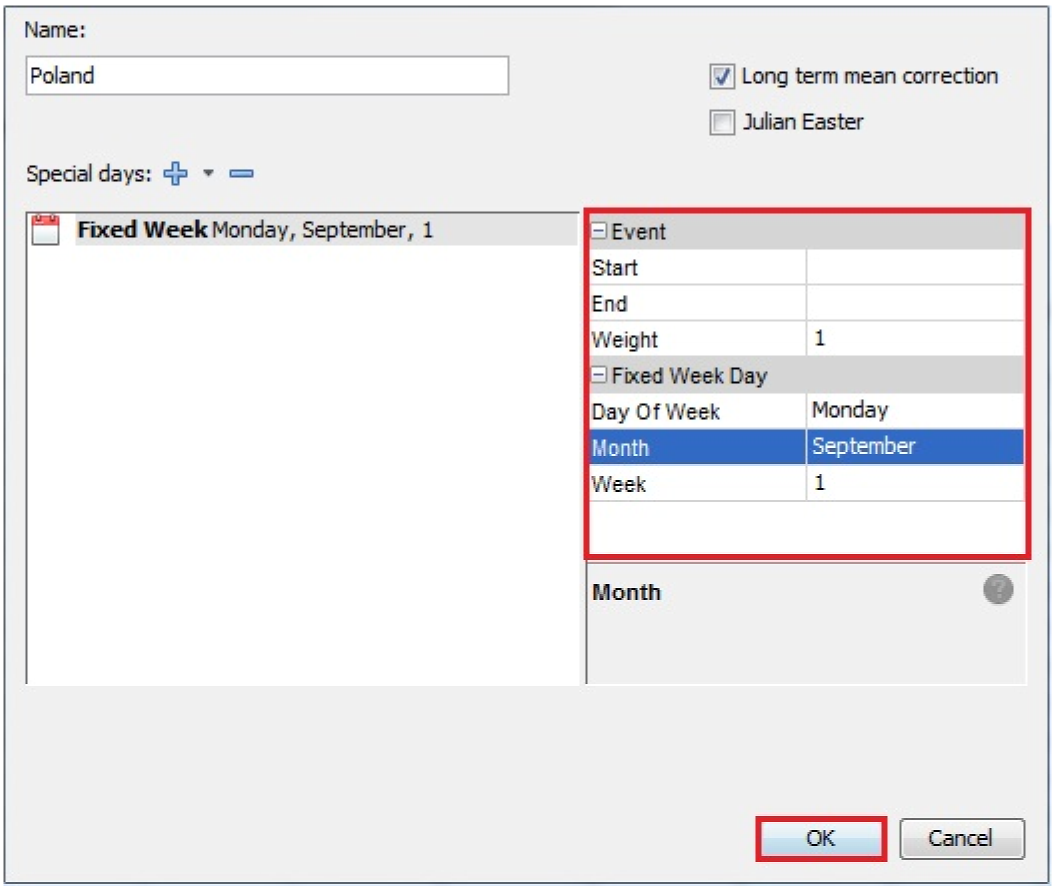


Figure 21: Text

The list of the holidays should contain only unique entries. Otherwise, a warning, as shown in the picture below, will be displayed.

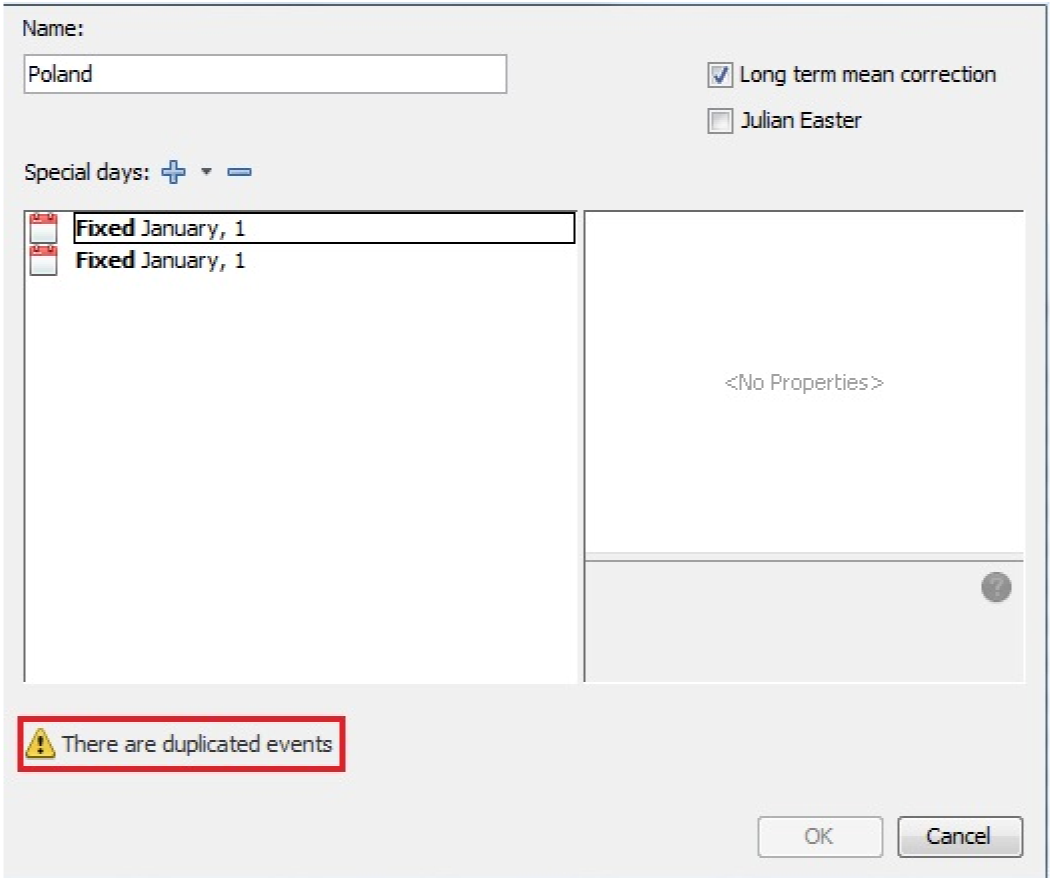


Figure 22: Text

A calendar without a name cannot be saved. Fill the *Name* box before saving the calendar.

Example : final view of a properly defined calendar for Poland

The calendar is visible in the *Workspace* window

- To display the available options right-click on it

A national calendar can be edited, duplicated (to create another calendar) and/or analysed (double click to display it in the panel on the right) or deleted.

0.0.0.0.3 * Chained Calendar

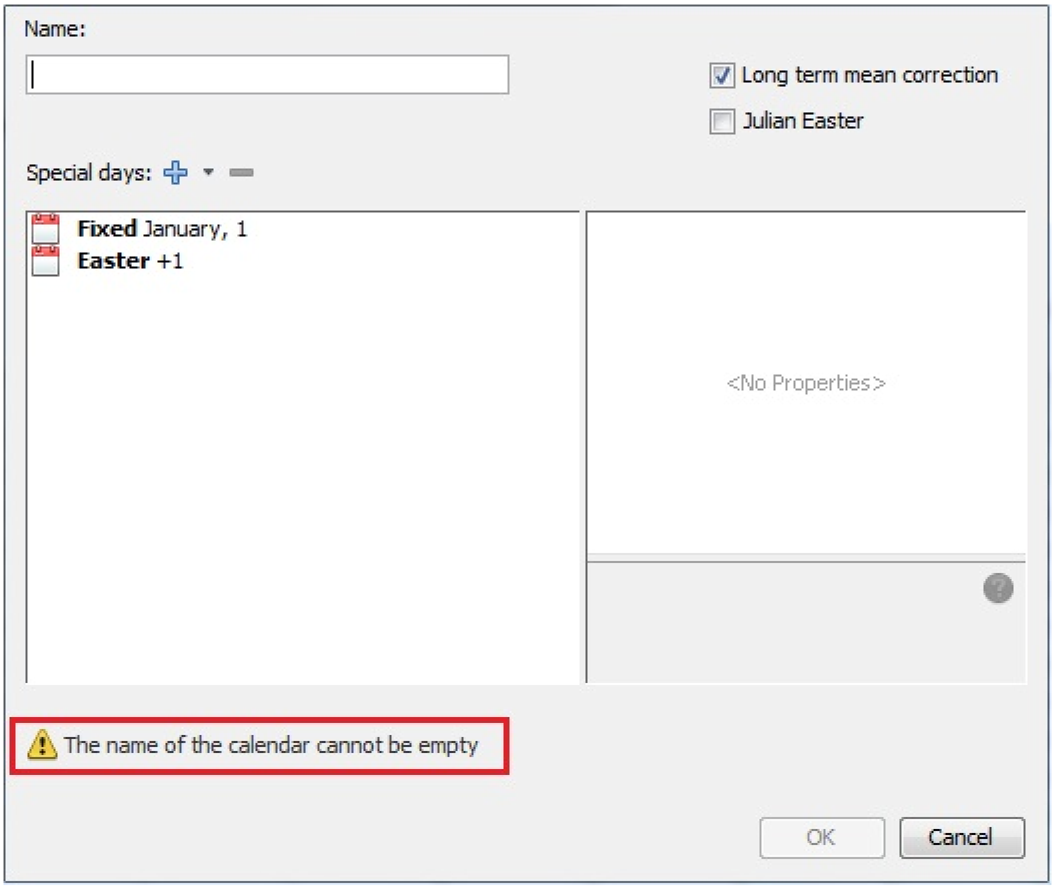


Figure 23: Text

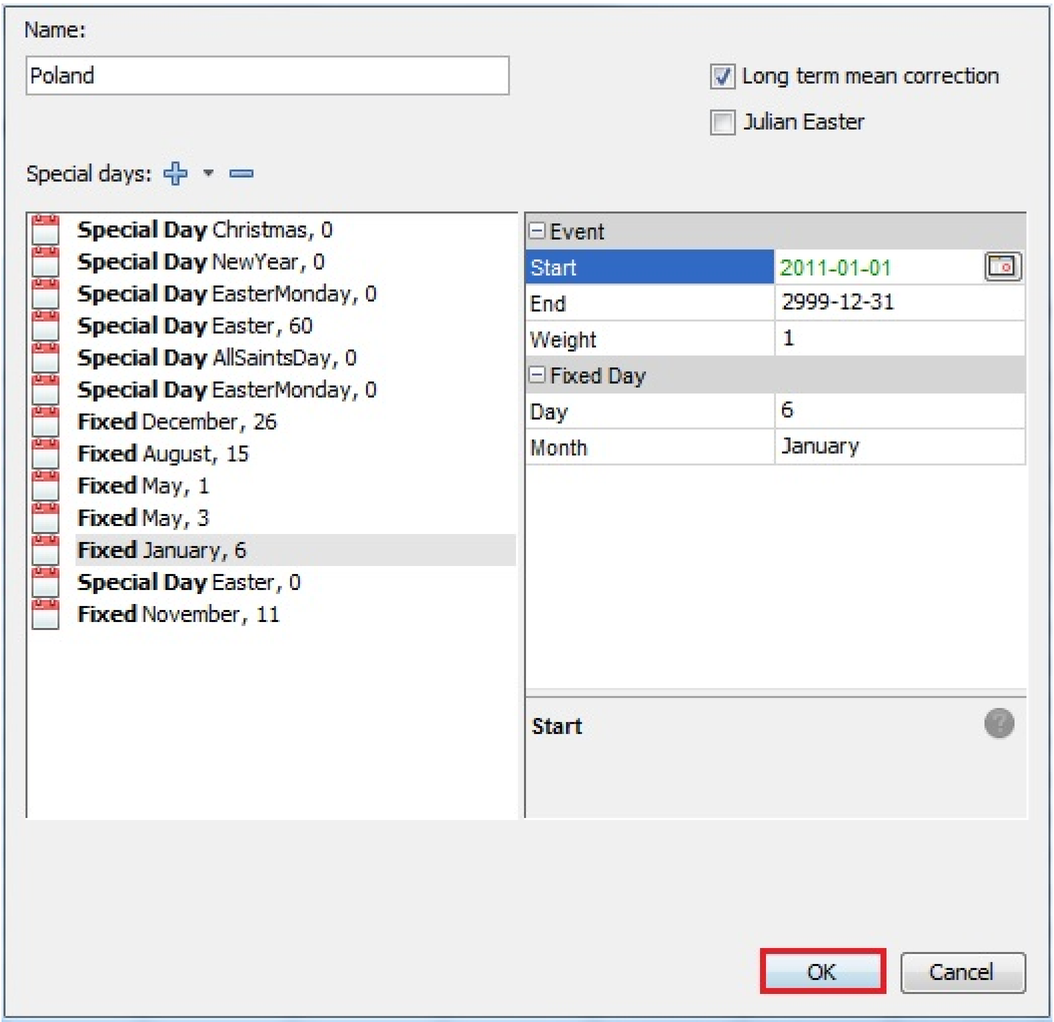


Figure 24: Text

Creating a chained calendar is relevant when a major break occurs in the definition of the country-specific holidays.

First define the 2 (or N) national calendars corresponding to each regime as explained in the section above.

To define a chained calendar: right click on Calendar item in the Utility panel of the workspace window

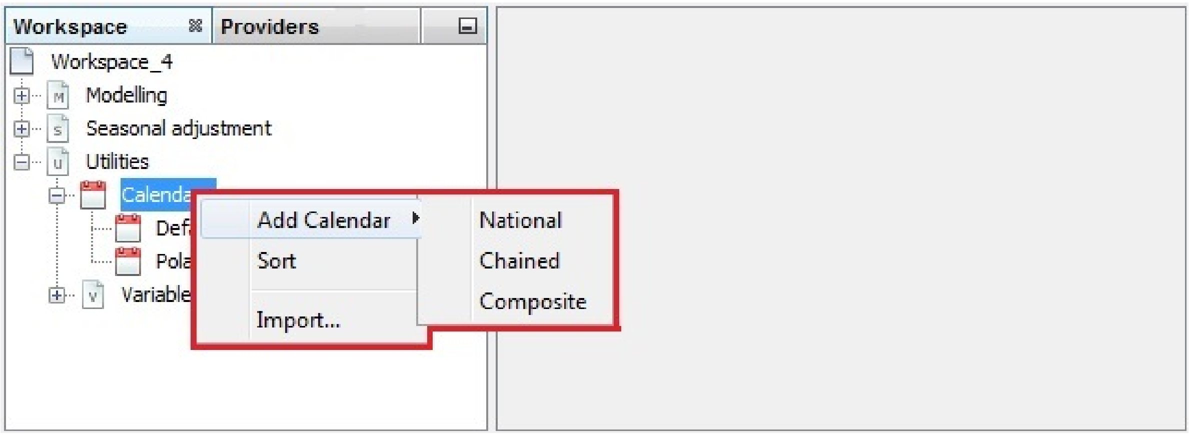


Figure 25: Text

In the *Properties* panel specify:

- first and the second calendar
- break date

0.0.0.0.4 * Composite Calendar

Creating a composite calendar is relevant when correcting series which include data from more than one country/region. This option can be used, for example, to create the calendar for the European Union or to create the national calendar for a country, in which regional holidays are celebrated.

First define the relevant national calendars corresponding to each member state/region as explained above.

To define a chained calendar: right click on Calendar item in the Utility panel of the workspace window

- Fill the name box
- Mark the regional calendars to be used

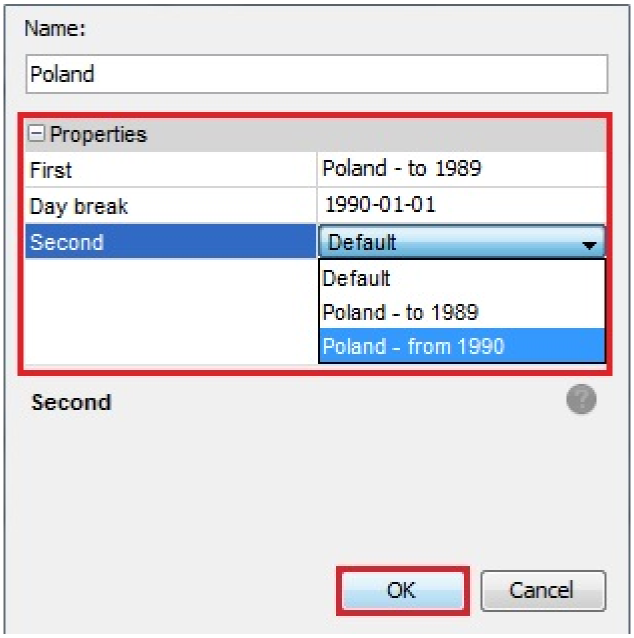


Figure 26: Text

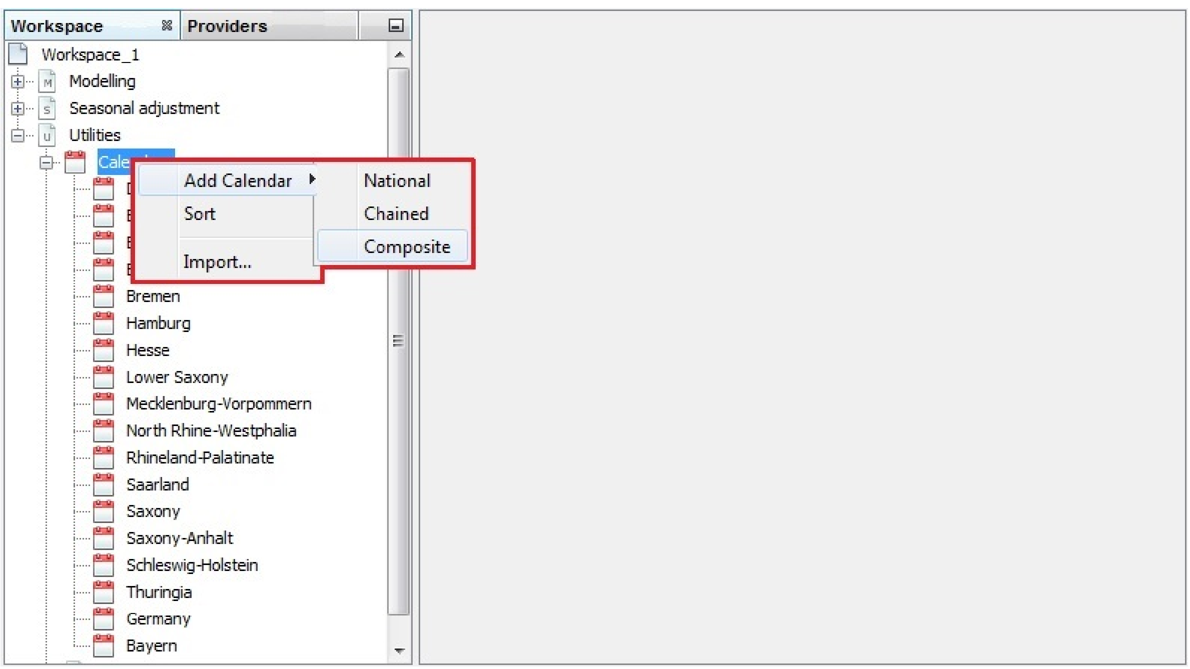


Figure 27: Text

- Assign a weight to each calendar.

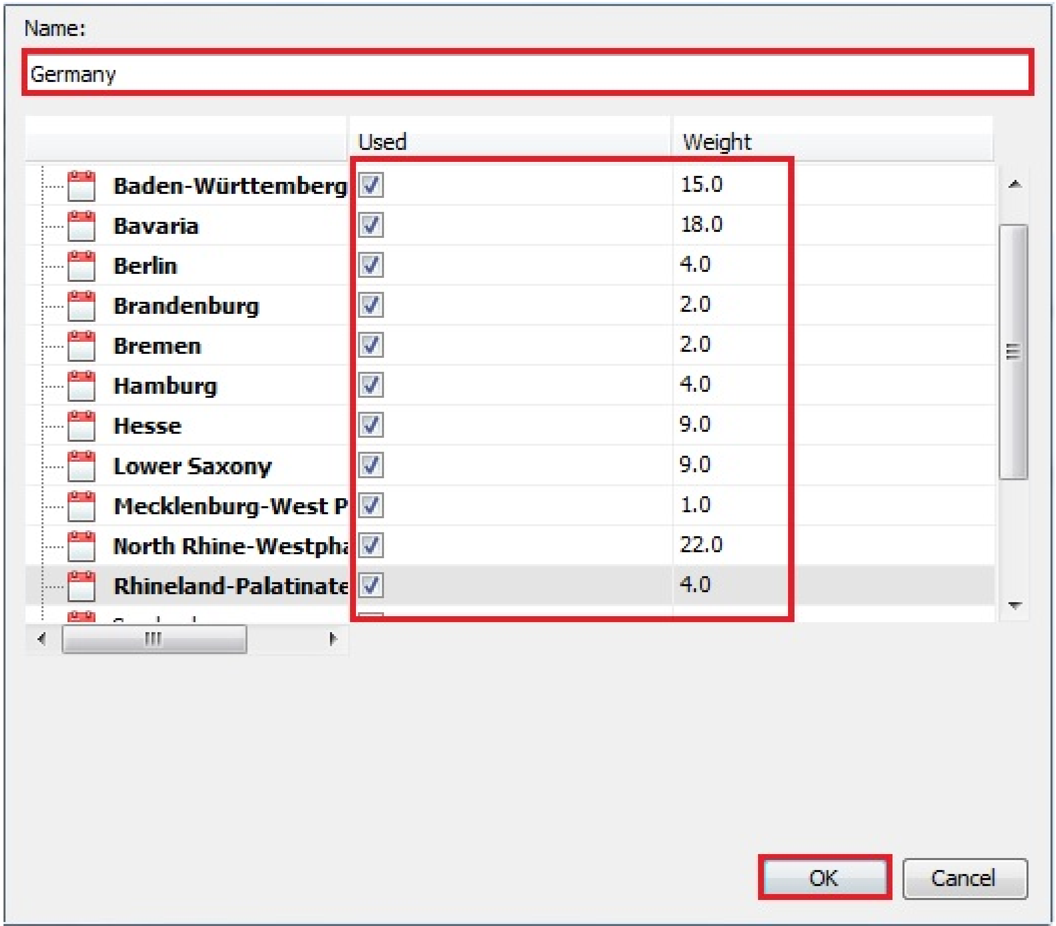


Figure 28: Text

0.0.0.0.5 * Importing an existing calendar from a file

Right click on the *Calendar* item from the *Workspace* window and choose the *Import* item from the menu.

Importing a calendar to JDemetra+

- choose the appropriate file and open it

Choosing the file

JDemetra+ adds it to the calendars list

A list of calendars with a newly imported calendar

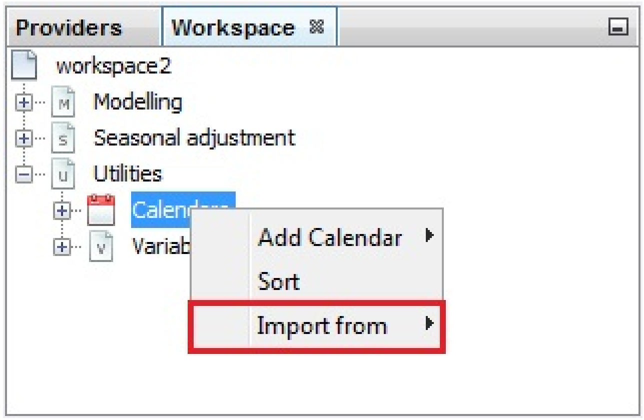


Figure 29: Text

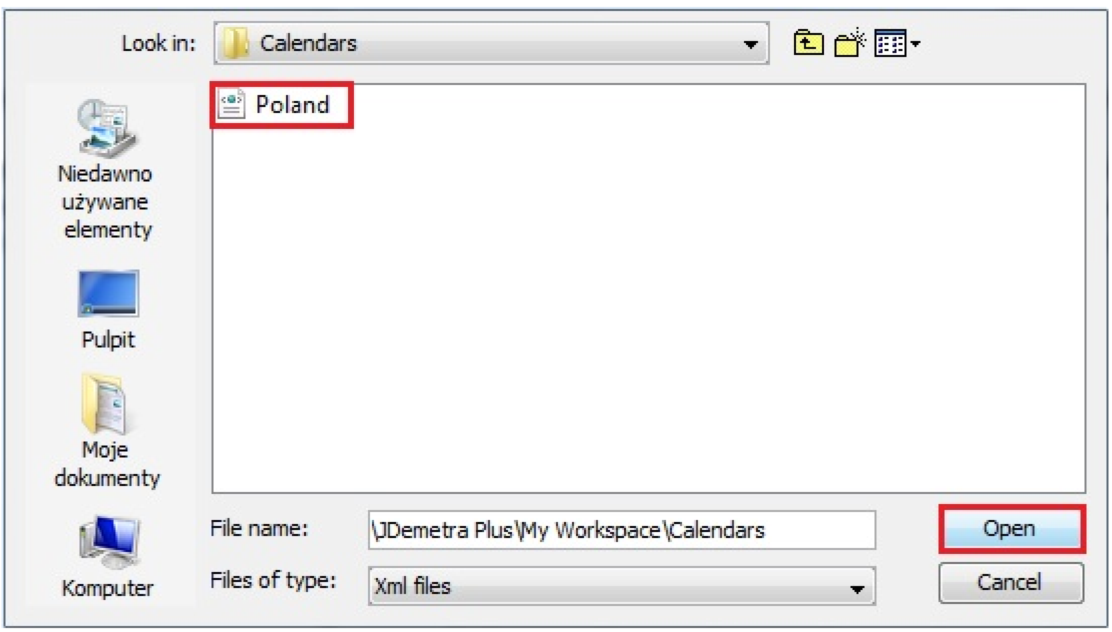


Figure 30: Text

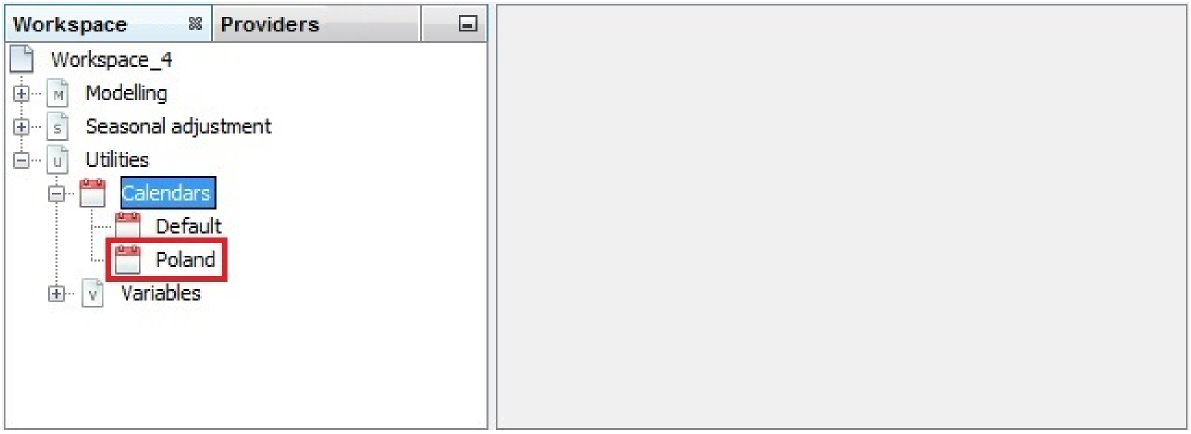


Figure 31: Text

0.0.0.0.6 * Example of a calendar file

example of a html file containing a calendar

```
  <?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <calendars xmlns="ec/tss.core">
  - <nationalCalendar name="Poland">
    - <specialDayEvent>
      - <fixedDay>
          <month>January</month>
          <day>1</day>
        </fixedDay>
      </specialDayEvent>
    - <specialDayEvent>
      - <fixedDay>
          <month>January</month>
          <day>6</day>
        </fixedDay>
        <validityperiod end="2999-12-31" start="2011-01-01"/>
      </specialDayEvent>
    - <specialDayEvent>
      - <specialCalendarDay>
          <event>Christmas</event>
        </specialCalendarDay>
      </specialDayEvent>
    - <specialDayEvent>
      - <easterRelatedDay>
          <offset>1</offset>
        </easterRelatedDay>
      </specialDayEvent>
    </nationalCalendar>
  </calendars>
```


####

Generating regressors

0.0.0.0.7 * Type of days

0.0.0.0.8 * Leap year

0.0.0.0.9 * Length of Period

(adjust param)

0.0.0.0.10 * Easter

0.0.0.0.11 * stock TD

In R with rjd3toolkit

Version 3 of JDemetra+ allows to build calendar regressors using the `rjd3toolkit` package.

The underlying concepts are identical to those available in the graphical user interface (GUI) as described above. R functions replicate the same process and all arguments and outputs are detailed in `rjd3toolkit` help pages. The sections below provide basic examples.

Note that, RJDemetra package based on version 2 of JDemetra+, doesn't allow to build calendars and generate regressors. Thus, two approaches are possible when using version 2

- use built in regressors ("working days" or "trading days") not taking into account national holidays
- import user defined calendar regressors

Creating calendars

0.0.0.0.1 * National Calendar

Creating a national calendar with rjd3toolkit.

```
## French calendar
frenchCalendar <- national_calendar(days = list(
  fixed_day(7, 14), # Bastille Day
  fixed_day(5, 8, validity = list(start = "1982-05-08")),# End of 2nd WW
  special_day('NEWYEAR'),
  special_day('CHRISTMAS'),
  special_day('MAYDAY'),
  special_day('EASTERMONDAY'),
  special_day('ASCENSION'), #
  special_day('WHITMONDAY'),
  special_day('ASSUMPTION'),
  special_day('ALLSAINTSDAY'),
  special_day('ARMISTICE'))
)
```

Holidays can be created with the following ways:

- as fixed days (falling on the exact same date every year)

```
day <- fixed_day(month= 12, day=25,weight= .9, validity = list(start="1968-02-01", end = "
day # December 25th, with weight=0.9, from February 1968 until January 2010
```

- as special days, when on the list of common holidays, which is available in the function's halp page.

```
# Get the list
?special_day
# To define a holiday for the day after Christmas, with validity and weight
special_day("CHRISTMAS", offset = 1, weight = 0.8,
validity = list(start="2000-01-01", end = "2020-12-01"))
```

- as a fixed week day

```
fixed_week_day(7, 2, 3) # second Wednesday of July
```

- as an easter related holiday

```
     easter_day(1), # Easter Monday
     easter_day(-2), # Good Friday
```

An example of calendar bringing together all options

```
MyCalendar <- national_calendar(list(
    fixed_day(7,21),
     special_day('NEWYEAR'),
     special_day('CHRISTMAS'),
     fixed_week_day(7, 2, 3), # second Wednesday of July
     special_day('MAYDAY'),
     easter_day(1), # Easter Monday
     easter_day(-2), # Good Friday
     fixed_day(5, 8, validity = list(start = "1982-05-08")), # End of 2nd WW
     single_day("2001-09-11"), # appearing once
     special_day('ASCENSION'),
     easter_day(offset=60,julian=FALSE, weight=0.5, validity = list(start="2000-01-01", en
     special_day('WHITMONDAY'),
     special_day('ASSUMPTION'),
     special_day('ALLSAINTSDAY'),
     special_day('ARMISTICE')))
```

For any defined calendar, it is possible to retrieve the long term-mean correction values which would be applied on a given set of regressors.

```
### Long-term means of a calendar
BE <- national_calendar(list(
    fixed_day(7,21),
    special_day('NEWYEAR'),
    special_day('CHRISTMAS'),
    special_day('MAYDAY'),
    special_day('EASTERMONDAY'),
    special_day('ASCENSION'),
    special_day('WHITMONDAY'),
    special_day('ASSUMPTION'),
    special_day('ALLSAINTSDAY'),
    special_day('ARMISTICE')))
class(BE)
lt <- long_term_mean(BE,12,
    groups = c(1,1,1,1,1,0,0),
    holiday = 7)
```

0.0.0.0.2 * Chained Calendar

Creating a chained calendar is relevant when a major break occurs in the definition of the country-specific holidays.

First define the 2 (or N) national calendars corresponding to each regime as explained in the section above.

```
Belgium <- national_calendar(list(special_day('NEWYEAR'),fixed_day(7,21)))
France <- national_calendar(list(special_day('NEWYEAR'),fixed_day(7,14)))
chained_cal<-chained_calendar(France, Belgium, "2000-01-01")
```

0.0.0.0.3 * Composite Calendar

Creating a composite calendar is relevant when correcting series which include data from more than one country/region. This option can be used, for example, to create the calendar for the European Union or to create the national calendar for a country, in which regional holidays are celebrated.

```
Belgium <- national_calendar(list(special_day('NEWYEAR'),fixed_day(7,21)))
France <- national_calendar(list(special_day('NEWYEAR'),fixed_day(7,14)))
composite_calendar<- weighted_calendar(list(France,Belgium), weights = c(1,2))
```

Generating regressors

First for monthly, Q, bi monthly…(set this right)

0.0.0.0.1 * Type of days

This section describes how to generate regressors to correct for type of days effects. They can be based on a default calendar (no specific holidays taken into account) or on a customized calendar.

0.0.0.0.1.1 * Trading day regressors without holidays using rjd3toolkit::td function

```
# Monthly regressors for Trading Days: each type of day is different
# contrasts to Sundays (6 series)
?td
regs_td<- td(frequency=12,c(2020,1),60, groups = c(1, 2, 3, 4, 5, 6, 0), contrasts = TRUE)
```

The `groups` argument allows to build groups of days, as daus belonging to the same group will be identified by the same number, and to set a reference for contrasts with the number `0`.

0.0.0.0.1.2 * Trading day regressors with pre-defined holidays using rjd3toolkit::calendar_td function

The rjd3toolkit::calendar_td function

```
?calendar_td
# first define a calendar
BE <- national_calendar(list(
    fixed_day(7,21),
    special_day('NEWYEAR'),
    special_day('CHRISTMAS'),
    special_day('MAYDAY'),
    special_day('EASTERMONDAY'),
    special_day('ASCENSION'),
    special_day('WHITMONDAY'),
    special_day('ASSUMPTION'),
    special_day('ALLSAINTSDAY'),
    special_day('ARMISTICE')))
# generate regressors
calendar_td(BE, frequency=12, c(1980,1), 240, holiday=7, groups=c(1,1,1,2,2,3,0),
```

```
contrasts = FALSE)
# here three groups and one reference are defined
# Mondays = Tuesdays= Wednesdays (`1`)
# Thursdays= Fridays (`2`)
# Saturdays (`3`)
# Sundays and all holidays (`0`)
```

0.0.0.0.2 * Leap year

0.0.0.0.3 * Length of Period

adjust param

0.0.0.0.4 * Easter Regressor

Create a regressor for modelling the easter effect:

```
#Monthly regressor, five-year long, duration 8 days, effect finishing on Easter Monday
ee <- easter_variable(frequency=12, c(2020,1),length=5*12,duration=8, endpos=1)
```

Display Easter Sunday dates in given period

The function below allows to display the date of Easter Sunday for each year, in the defined period. Dates are displayed in "YYYY-MM-DD" format and as a number of days since January 1st 1970.

```
#Dates from 2018(included) to 2023 (included)
easter_dates(2018, 2023)
```

0.0.0.0.5 * stock TD

0.0.0.0.6 * Daily data (dummies)

```
## dummies corresponding to holidays
q <- holidays(BE, "2020-01-01",365.25, type="All")
tail(q)
```

0.0.0.0.6.1 * Weekly data

Test for Residual Calendar effects

(To be added: where exactly to find the tests in GUI and R)

We consider below tests on the seasonally adjusted series (sa_t) or on the irregular component (irr_t). When the reasoning applies on both components, we will use y_t. The functions *stdev* stands for "standard deviation" and *rms* for "root mean squares"

The tests are computed on the log-transformed components in the case of multiplicative decomposition.

TD are the usual contrasts of trading days, 6 variables (no specific calendar).

Non significant irregular

When irr_t is not significant, we don't compute the test on it, to avoid irrelevant results. We consider that irr_t is significant if $stdev(irr_t) > 0.01$ (multiplicative case) or if $stdev(irr_t)/rms(sa_t) > 0.01$ (additive case).

F test

The test is the usual joint F-test on the TD coefficients, computed on the following models:

0.0.0.0.1 * Autoregressive model (AR modelling option)

We compute by OLS:

$$y_t = \mu + \alpha y_{t-1} + \beta TD_t + \epsilon_t$$

0.0.0.0.2 * Difference model

We compute by OLS:

$$\Delta y_t - \overline{\Delta y_t} = \beta TD_t + \epsilon_t$$

So, the latter model is a restriction of the first one ($\alpha = 1, \mu = \mu = \overline{\Delta y_t}$)

The tests are the usual joint F-tests on $\beta \quad (H_0 : \beta = 0)$.

By default, we compute the tests on the 8 last years of the components, so that they might highlight moving calendar effects.

Remark:

In Tramo, a similar test is computed on the residuals of the Arima model. More exactly, the F-test is computed on $e_t = \beta TD_t + \epsilon_t$, where e_t are the one-step-ahead forecast errors.

Benchmarking and temporal disaggregation

Benchmarking overview

Often one has two (or multiple) datasets of different frequency for the same target variable. Sometimes, however, these data sets are not coherent in the sense that they don't match up. Benchmarking[^1] is a method to deal with this situation. An aggregate of a higher-frequency measurement variables is not necessarily equal to the corresponding lower-frequency less-aggregated measurement. Moreover, the sources of data may have different reliability levels. Usually, less frequent data are considered more trustworthy as they are based on larger samples and compiled more precisely. The more reliable measurements, hence often the less frequent, will serve as benchmark.

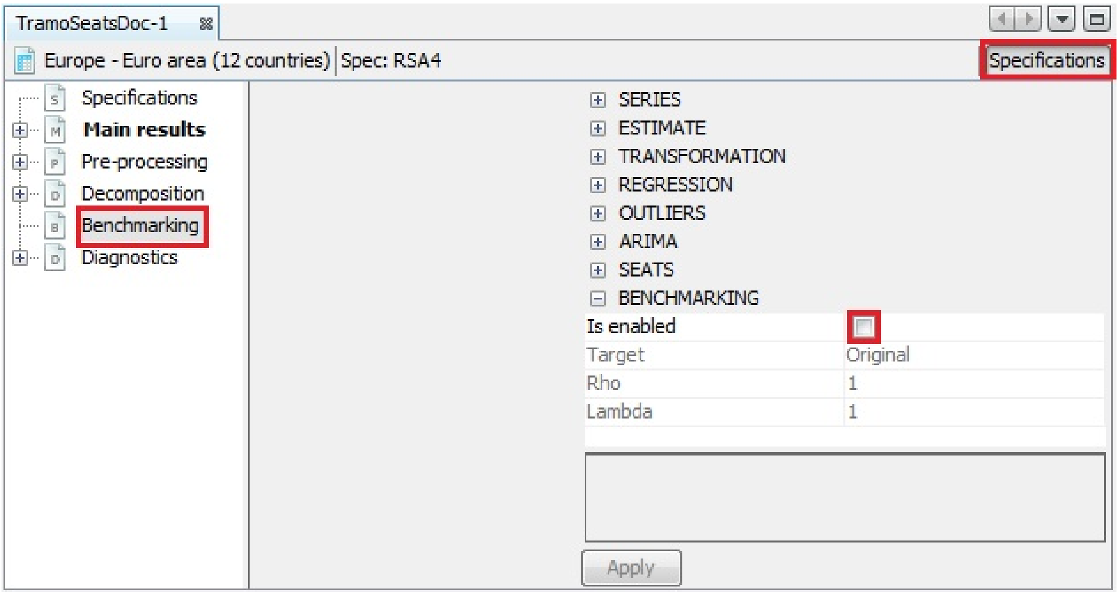
In seasonal adjustment methods benchmarking is the procedure that ensures the consistency over the year between adjusted and non-seasonally adjusted data. It should be noted that the [ESS Guidelines on Seasonal Adjustment (2015)] (https://ec.europa.eu/eurostat/documents/3859598/6830795/K GQ-15-001-EN-N.pdf/d8f1e5f5-251b-4a69-93e3-079031b74bd3), do not recommend benchmarking as it introduces a bias in the seasonally adjusted data. The U.S. Census Bureau also points out that "*forcing the seasonal adjustment totals to be the same as the original series annual totals can degrade the quality of the seasonal adjustment, especially when the seasonal pattern is undergoing change. It is not natural if trading day adjustment is performed because the aggregate trading day effect over a year is variable and moderately different from zero*"[^2]. Nevertheless, some users may need that the annual totals of the seasonally adjusted series match the annual totals of the original, non-seasonally adjusted series[^3].

According to the [ESS Guidelines on Seasonal Adjustment (2015)] (https://ec.europa.eu/eurostat/documents/385 GQ-15-001-EN-N.pdf/d8f1e5f5-251b-4a69-93e3-079031b74bd3), the only benefit of this approach is that there is consistency over the year between adjusted and the non-seasonally adjusted data; this can be of particular interest when low-frequency (e.g. annual) benchmarking figures officially exist (e.g. National Accounts, Balance of Payments, External Trade, etc.) and where users' needs for time consistency are stronger.

Tools

Benchmarking with GUI

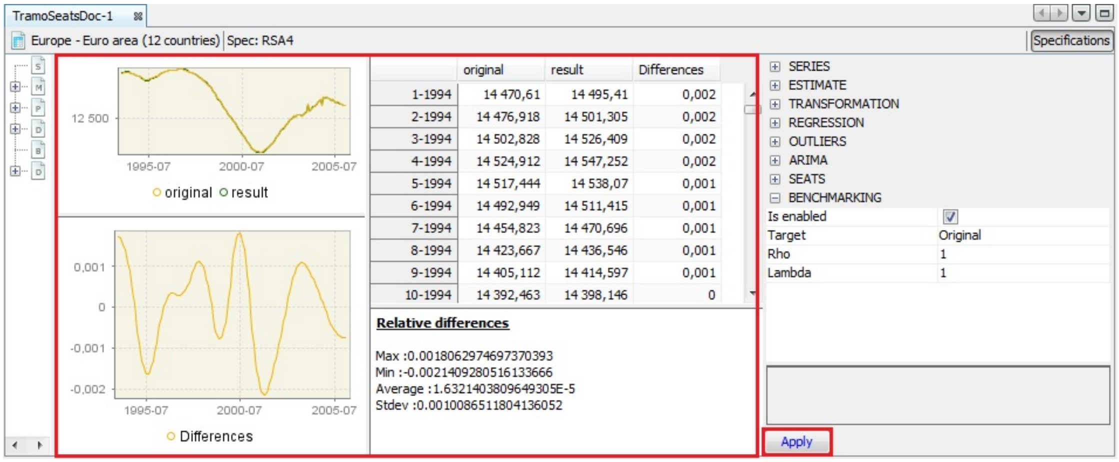
1. With the pre-defined specifcations the benchmarking functionality is not applied by default following the *ESS Guidelines on Seasonal Adjustment* (2015) recommendations. It means that once the user has seasonally adjustd the series with a pre-defined specifcation the *Benchmarking* node is empty. To execute benchmarking click on the *Specifications* button and activate the checkbox in the *Benchmarking* section.

   Benchmarking option – a default view

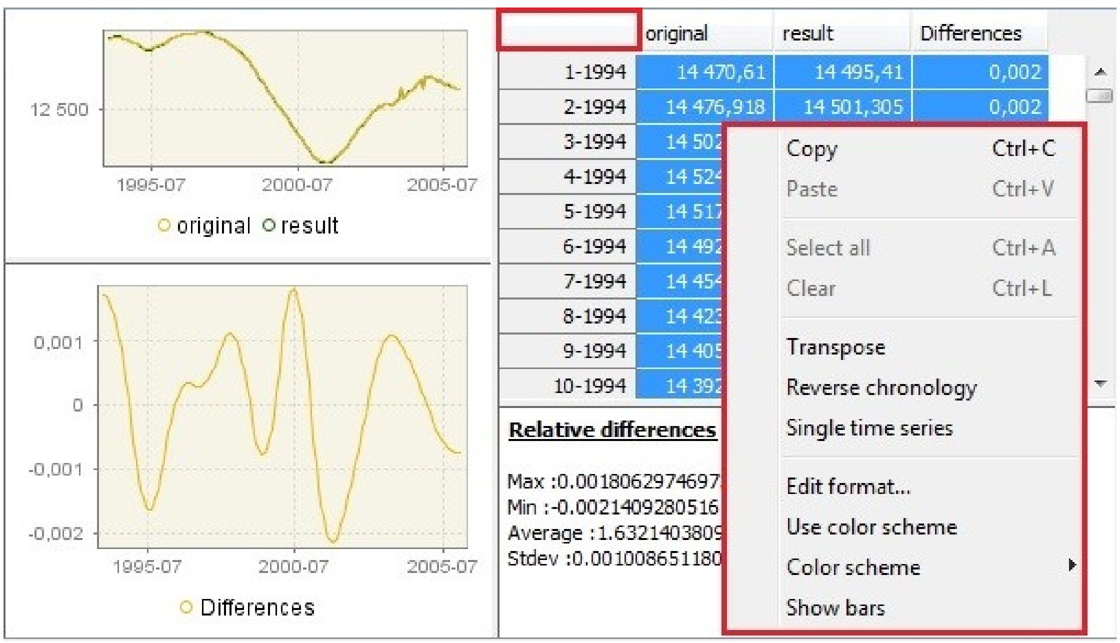
2. Three parameters can be set here. *Target* specifies the target variable for the benchmarking procedure. It can be either the *Original* (the raw time series) or the *Calendar Adjusted* (the time series adjusted for calendar effects). *Rho* is a value of the AR(1) parameter (set between 0 and 1). By default it is set to 1. Finally, *Lambda* is a parameter that relates to the weights in the regression equation. It is typically equal to 0 (for an additive decomposition), 0.5 (for a proportional decomposition) or 1 (for a multiplicative decomposition). The default value is 1.

3. To launch the benchmarking procedure click on the **Apply** button. The results are displayed in four panels. The top-left one compares the original output from the seasonal adjustment procedure with the result from applying a benchmarking to the seasonal adjustment. The bottom-left panel highlights the differences between these two results. The outcomes are also presented in a table in the top-right panel. The relevant statistics concerning relative differences are presented in the bottom-right panel.



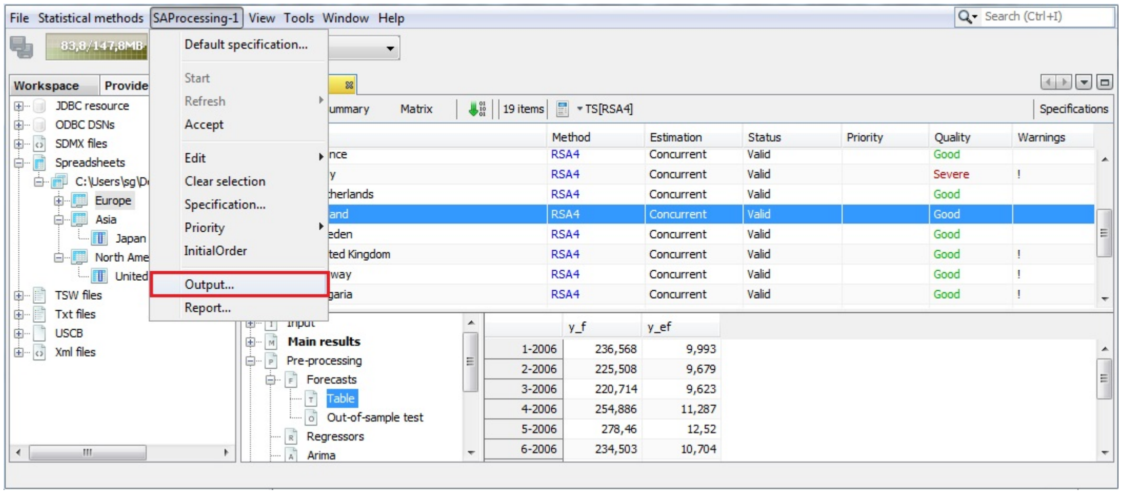
The results of the benchmarking procedure

4. Both pictures and the table can be copied the usual way (see the *Simple seasonal adjustment of a single time series* scenario).



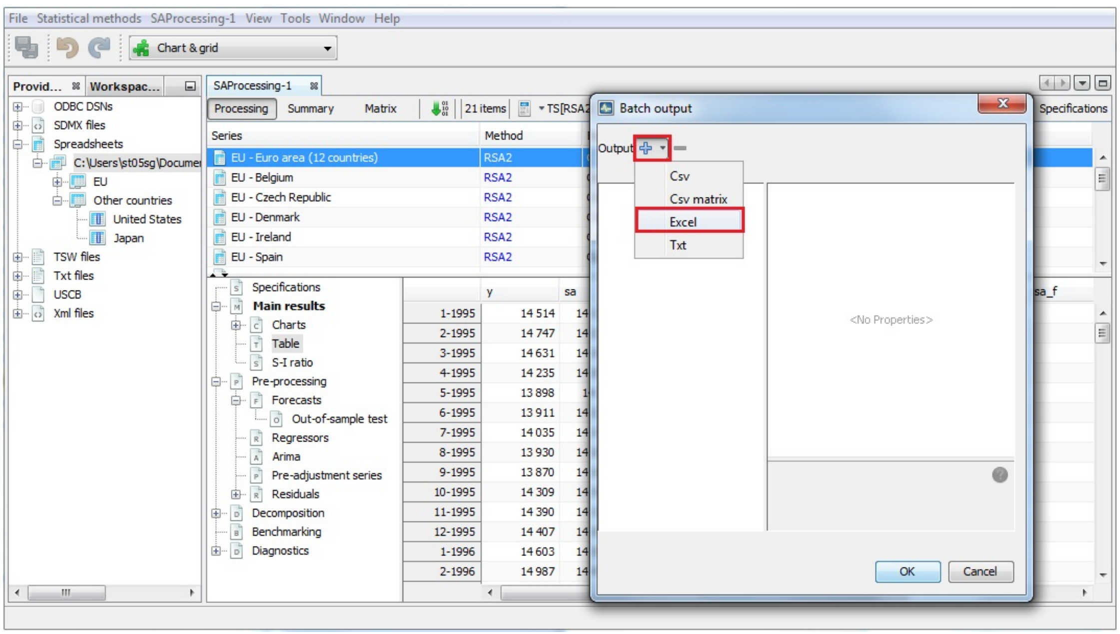
Options for benchmarking results

5. To export the result of the benchmarking procedure (*benchmarking.result*) and the target data (*benchmarking.target*) one needs to once execute the seasonal adjustment with benchmarking using the muli-processing option (see the *Simple seasonal adjustment of multiple time series* scenario. Once the muli-processing is executed, select the *Output* item from the *SAProcessing* menu.



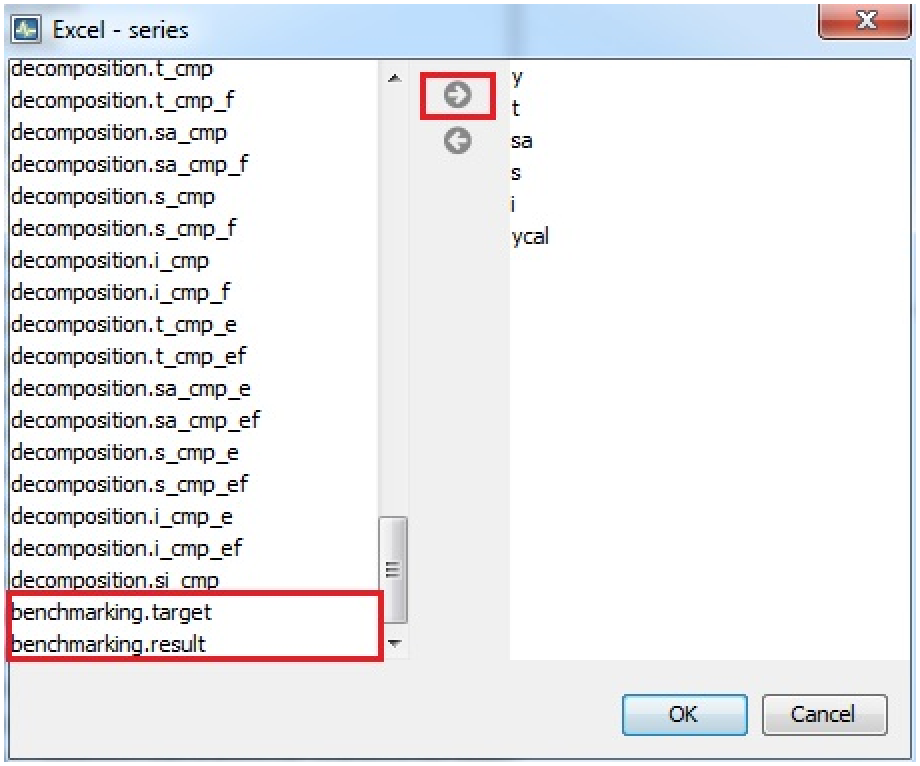
The *SAProcessing* menu

6. Expand the "+" menu and choose an appropriate data format (here Excel has been chosen). It is possible to save the results in TXT, XLS, CSV, and CSV matrix formats. Note that the available content of the output depends on the output type.



Exporting data to an Excel file

7. Chose the output items that refer to the results from the benchmarking procedure, move them to the window on the right and click **OK**.



Exporting the results of the benchmarking procedure

Benchmarking in R

See package **rjd3bench** and its documentation pages in R.

Trend-cycle estimation

Under construction

Overview

Estimation Methods

Tools

rjd3highfeq package

rjdfilters package

Nowcasting

Under construction.

Part II

Tools

The different tools described in this part:

- Graphical User Interface GUI
- …enhanced with additional plug-ins
- ..and a Cruncher for mass production
- R packages

Graphical User Interface

Overview

This chapter provides general information about using the graphical interface (GUI). Specific indications related to a given algorithm (X13-Arima, Tramo-seats, Benchmarking…) are displayed in the relevant chapters.

Available algorithms

The graphical user interface in the 2.x family gives access to:

- Seasonal adjustment (SA) algorithms
  - X13-Arima
  - Tramo-Seats
  - Direct-indirect SA comparisons
- Outlier detection (TERROR)
- Benchmarking

The graphical user interface in the 3.x family gives access **in addition** to extended SA algorithms for high-frequency data (HF).

Available Time Series tools

The graphical user interface in the 2.x and 3.x family give access to generic time series tools:

- Graphics
  - time domain
  - spectral analysis
- Tests
  - seasonality tests
  - autocorrelation, normality, randomness tests

Installation Procedure

JDemetra+ is a stand-alone application packed in a zip package. To run JDemetra+ the Java RE 8 or higher is needed. Java RE can be downloaded from Oracle website.

The official release of JDemetra+ is accessible at a dedicated Github page. The site presents all available releases - both official releases (labelled in green as latest releases) and pre-releases (labelled in red) - packed in zip packages. From the *Latest release* section either choose the installer appropriate for your operating system (Windows, Linux, Mac OS, Solaris) or take the portable zip-file. The installation process is straightforward and intuitive. For example, when the zip-file is chosen and downloaded, then under Windows OS the application can be found in the "bin"-folder of the installation/unpacked zip.

Running JDemetra+

To open an application, double click on *nbdemetra.exe* or *nbdemetra64.exe* depending on the system version (*nbdemetra.exe* for the 32-bit system version and *nbdemetra64.exe* for the 64-bit system version).

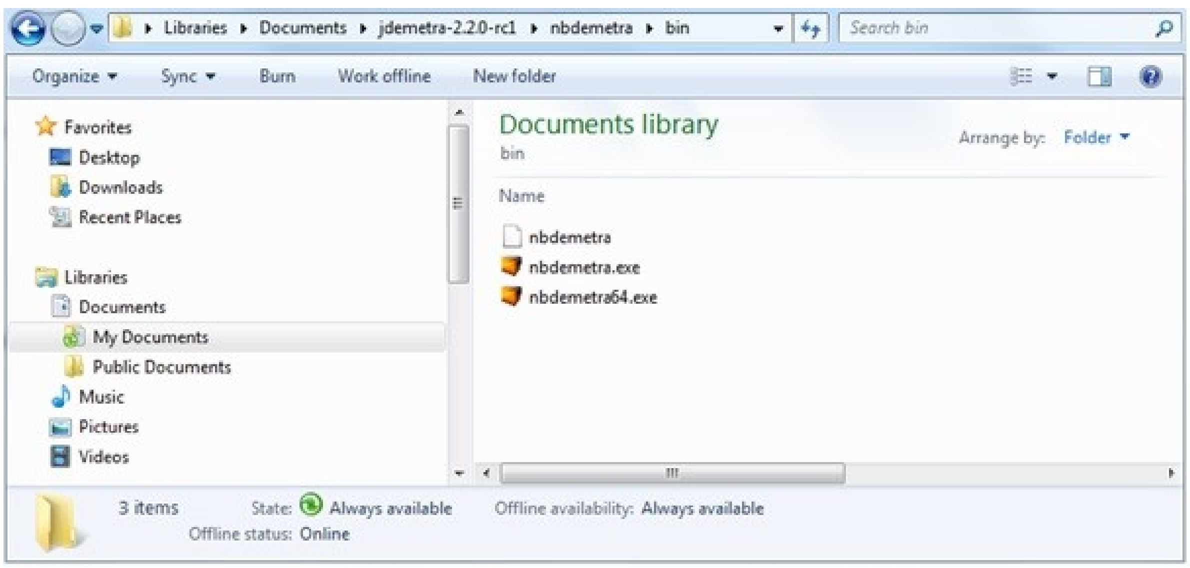


Figure 32: **Launching JDemetra+**

If the launching of JDemetra+ fails, you can try the following operations:

- Check if Java SE Runtime Environment (JRE) is properly installed by typing in the following command in a terminal: *java –version*

- Check the logs in your home directory:
  - %appdata%/.nbdemetra/dev/var/log/ for Windows;
  - ~/.nbdemetra/dev/var/log/ for Linux and Solaris;
  - ~/Library/Application Support/.nbdemetra/dev/var/log/ for Mac OS X.

In order to remove a previously installed JDemetra+ version, the user should delete an appropriate JDemetra+ folder.

Closing JDemetra+

To close the application, select *File* → *Exit* from the File menu.

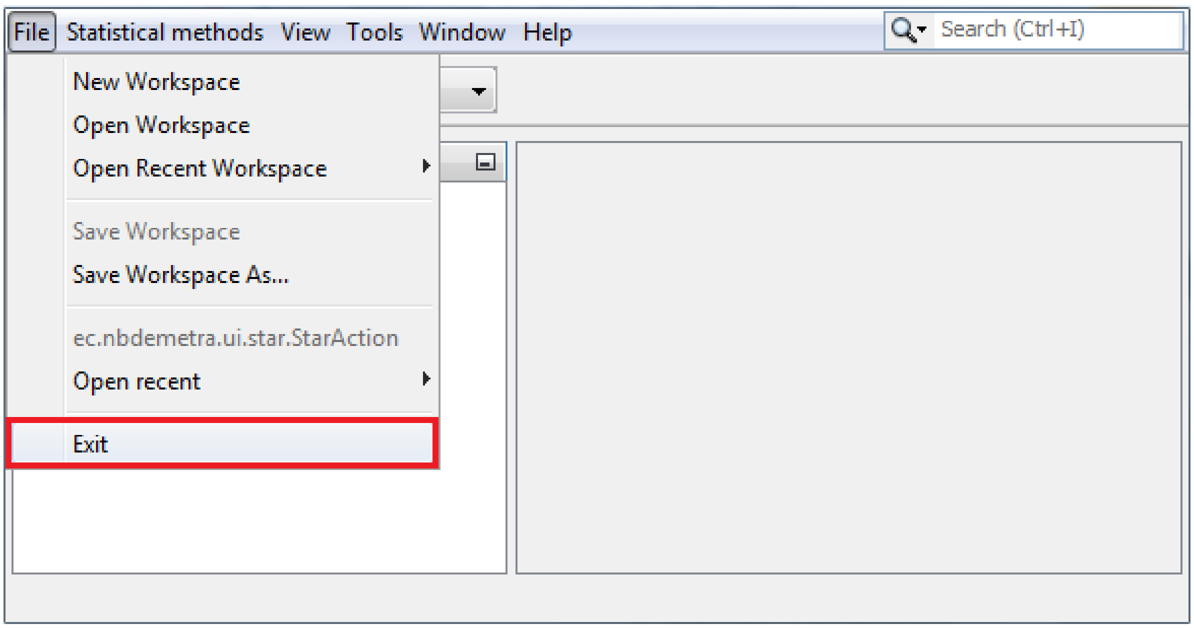


Figure 33: **Closing JDemetra+**

The other way is to click on the close box in the upper right-hand corner of the JDemetra+ window.

If there is any unsaved work, JDemetra+ will display a warning and provide the user with the opportunity to save it. The message box is shown below.

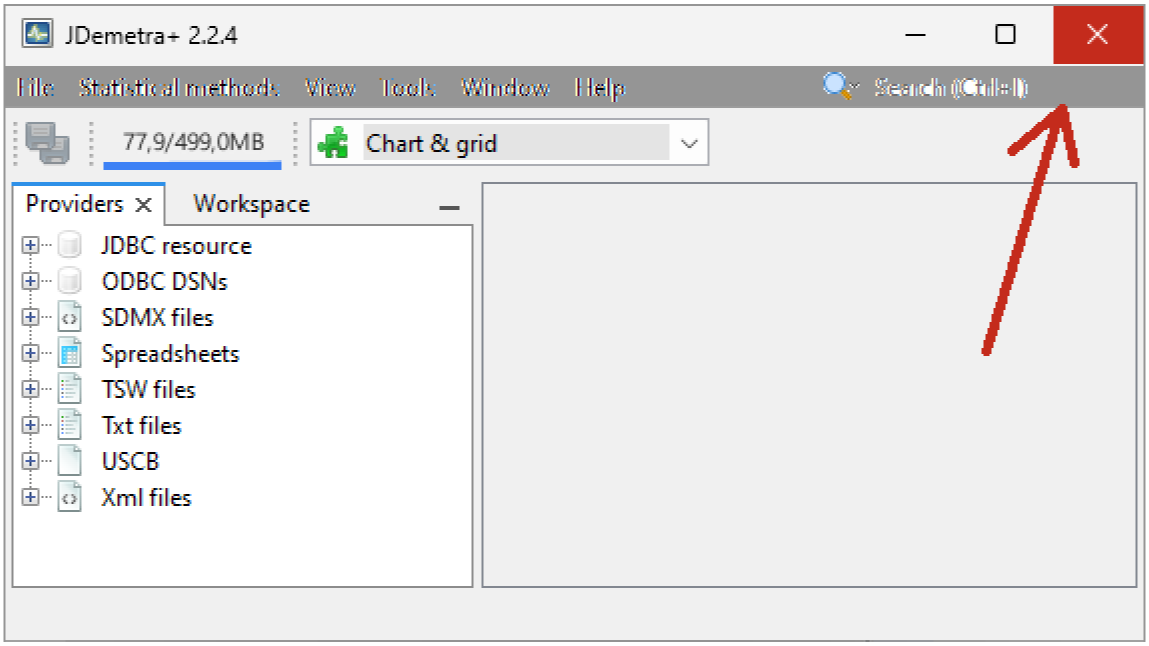


Figure 34: **Closing JDemetra+ with the close box**

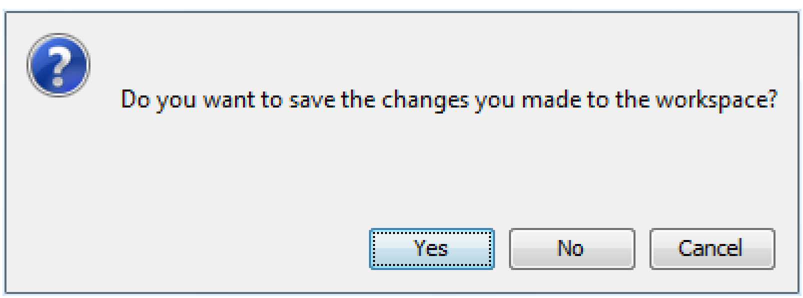


Figure 35: **The warning from leaving JDemetra+ without saving the workspace**

Interface Starting Window

The default view of the JDemetra+ window, which is displayed after launching the program, is clearly divided into several panels.

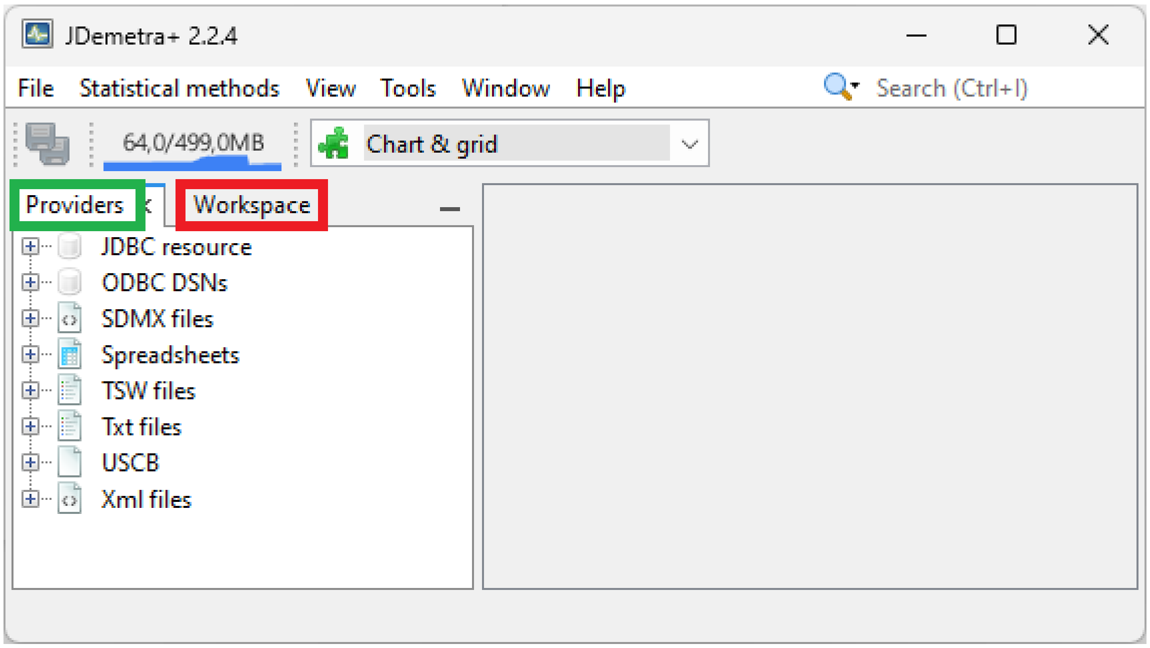


Figure 36: **JDemetra+ default window**

By default, on the left hand side of the window two panels are visible:

- the *Workspace* panel stores the results generated by the software as well as settings used to create them;
- and the *Providers* panel organises the imported raw data within each data provider;

The other key parts of the user interface are:

- The application menu.
- A central empty zone for presenting the actual analyses further called the Results panel.

Providers window

The *Providers* window presents the list of the data sources and organises the imported series within each data provider.

By default, JDemetra+ supports the following data sources:

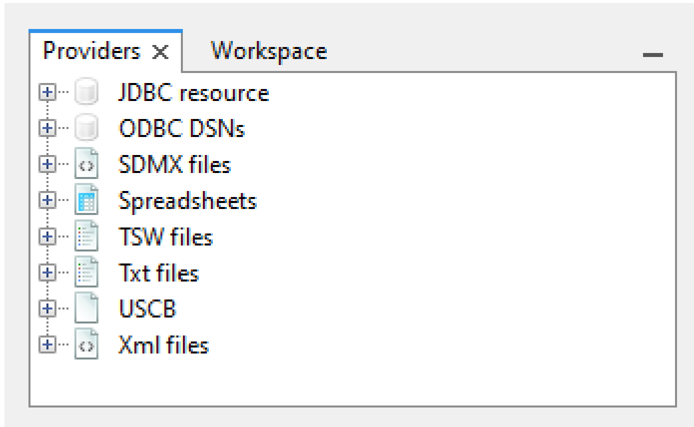


Figure 37: **The *Providers* window**

- JDBC;
- ODBC;
- SDMX;
- Excel spreadsheets;
- TSW (input files for the Tramo-Seats-Windows application by the Bank of Spain);
- TXT;
- USCB (input files for the X-13-ARIMA-SEATS application by the U.S. Census Bureau);
- XML.

All standard databases (Oracle, SQLServer, DB2, MySQL) are supported by JDemetra+ via JDBC, which is a generic interface to many relational databases. Other providers can be added by users by creating plugins (see *Plugins* section in the Tools menu).

The *Providers* window organises data in a tree structure reflecting the manner in which data are presented in the original source.

The picture below presents how JDemetra+ visualises the imported spreadsheet file. If the user expands all the pluses under the spreadsheet all the series within each sheet that has been loaded are visible. Here two time series are visible: *Japan* (under the *Asia* branch) and *United States* (under the *North America* branch) while the *Europe* branch is still folded. The names of the time series have been taken from the column headings of the spreadsheet while the names of the branches come from sheets' names.

Series uploaded to the *Providers* window can be displayed, modified and tested for seasonality and used in estimation routines (see Modelling and Seasonal adjustment). The data sources can be restored after re-starting the application so that there is no need to get them again. This functionality can be set in the *Behaviour* tab available at the *Option* item from the *Tools* menu.