JDemetra+ documentation

Anna Smyk (Coordinator) Tanguy Barthelemy
Karsten Webel Maria Novas Filgueira
Luis Sanguiao Sande Felix Aparicio Perez

2025-09-26

Table of contents

What is JDemetra+?

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. It has been officially recommended by Eurostat to the members of the European Statistical System since 2015. It is unique in its combination of very fast Java routines, a **Graphical User Interface** and an **ecosystem of R packages (rjdverse)**. The graphical interface provides structured visual feedback suitable for refined analysis and training. R tools allow the user to combine the capabilities of JDemetra+ with the versatility of the R world, be it for statistical functions or data wrangling. A pdf version of this documentation is available here

Main Features

JDemetra+ provides algorithms for:

- Seasonal Adjustment
- Trend estimation
- Benchmarking and temporal disaggregation
- Nowcasting
- Revision analysis

Currently available versions

The highest version currently recommended for production purposes is 2.2.4.

Version 3.x family provides, among other things, extended features for seasonal adjustment and trend estimation, including **high frequency data**. The latest version 3.2.4 was released on July 11th 2024.

Background

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 (2024)
- 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)

Useful links

To get started, you can browse and watch tutorials for JDemetra+. On this YouTube channel you will also find JDemetra+ related webinars.

If you need user support, please raise an issue in this repository

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

How to contribute

If you want to help us improve this book, you can fork this repository on GitHub and create pull requests with your contributions.

Installing the software

The sections below detail how to install JDemetra+ (Graphical user interface and Cruncher) and how to configure R to run rjd3 packages.

Graphical User Interface (GUI)

You can find the latest releases

- in the v2.x family: here

As of April 2024, the latest version is v2.2.4

Scroll down the page, download and unzip the file *jdemetra-2.2.4-bin.zip*.

To start the application, run the file **nbdemetra64.exe** located in the following subfolder ...\jdemetra-2.2.4-bin\nbdemetra\bin

Remark: You can create shortcuts to the executable files if you want to launch them from another folder (Desktop, project folder...).

- in the v3.x family: here

You should install the latest release available denoted from now on v3.x.y

Scroll down the page, download and unzip the file *jdemetra-standalone-3.x.y-windows-x86_64.zip*, if you use Windows, or the zip corresponding to you OS in the list.

To start the application, run the file **nbdemetra64.exe** located in the following subfolder ...\jdemetra-standalone-3.x.y-windows-x86_64\nbdemetra\bin

Version 3.x requires Java 17 or higher, *jdemetra-standalone-3.x.y-windows-x86_64.zip* contains a portable version of Java 21, so you don't have to deal with this issue on your computer.

R packages related to version 3.x (rjd3...) also require Java 17 or higher, you can (and should) use the portable version provided with the graphical user interface to run them, this is explained here.

Additional plugins

To benefit from extended features of the graphical user interface installing additional plug-ins is required. The list of available extensions and the installation procedure are detailed here. How to access the added features in the GUI is described here.

In the **v2.x family** some of the additional features are:

- Benchmarking and Temporal disaggregation Plug-in here
- Nowcasting Plug-in here

In the **v3.x family** some of the additional features are:

- Benchmarking and Temporal disaggregation Plug-in here
- Seasonal adjustment of high frequency data Plug-ins here
- Additional algorithms for seasonal adjustment Plug-ins here

Cruncher

JDemetra+ has an executable module, called the Cruncher, allowing to automate tasks

To install it :

- in v2, go to the **JWSACruncher** page, **download** and **unzip** the compressed folder `jwsacruncher-2.2.4-bin.zip`.
- in v3, go to the **jdplus-main** page, **download** and **unzip** the compressed folder `jwsacruncher-standalone-3.x.y-windows-x86_64.zip`.

R packages

JDemetra+ algorithms can be accessed in R, which is detailed here

You can directly head over to this GitHub page, then for each package indications for installation and basic use are provided in the readme files.

Configuration needed to run rjd3 packages

To use rjd3 packages in R, you need Java 17 or higher.

You can (and should) use the version of Java that comes with the Graphical User Interface (GUI). It is contained in the file **jdemetra-standalone-3.2.4-windows-x86_64.zip** available on this page, in the "Assets" section, which you might have to expand to be able to see all the files.

Once unzipped, the Java will be in located in C:\Software\jdemetra-standalone-3.x.y-windows-x86_64\nbdemetra\jdk-21.0.2+13-jre (path to be adapted to your computer/server): this location must be declared in R.

After unzipping the file:

- run this code at the beginning of your programs

```
Sys.setenv(JAVA_HOME = ".../jdemetra-standalone-3.x.y-windows-x86_64/nbdemetra/jdk-21.0.2+
```

or

- declare it "permanently" is the Renviron file (where environment variables specific to `R` are stored), following the steps below:
  - in the R console run `file.edit("~/.Renviron")`
  - in the file that opens add the line :

```
JAVA_HOME = "C:/Software/jdemetra-standalone-3.x.y-windows-x86_64/nbdemetra/jdk-21.0.2+1
```

  - save the file and restart R

Follow R packages installation procedure and run basic examples from the readme files of each package. They are all listed on this page

How to use this book

JDemetra+ on-line documentation provides a step-by-step guidance on how to use the algorithms featured in JDemetra+, highlighting all available options and outputs. It also describes the different tools giving access to these algorithms and provides a specific methodological background.

Structure

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

- Algorithms
- Tools
- Methods

What is new in version 3.x

Seasonal adjustment of high frequency data is the main functionality upgrade brought by version 3 but there are many others. They are pointed out in this chapter

R ecosystem vs Graphical User Interface (GUI)

The vast majority of functions have identical options and output when used via R or GUI, but there are some exceptions. Data structure and visualization can also differ. This is described in the relevant sections throughout this book.

New Features in v 3.x family

In this chapter

This chapter provides an overview of the new features in version 3.x as well as significant modification of display or content for features already available in v 2.2.4. Compared to its predecessor, version 3.x provides:

- Additional Algorithms for Seasonal Adjustment (SA), Benchmarking and Temporal Disaggregation (TD), Nowcasting, Revision Analysis
- More stand alone time series tools
- More “acceptable” frequencies in SA
- New SA (mass) production possibilities

Seasonal Adjustment and Modelling

Seasonal adjustment algorithms

Algorithm	Version 2.x		Version 3.x	
	Access in GUI	Access in R	Access in GUI	Access in R
X-13 Arima	yes	RJDemetra	yes	`rjd3x13`
Tramo-Seats	yes	RJDemetra	yes	`rjd3tramoseats`
X12plus			yes	`rjd3x11plus`
STL			yes	`rjd3stl`
BSM			yes	`rjd3sts`
SEATS+			upcoming	upcoming

Improvements on historical algorithms

Improvements X-13-ARIMA and Tramo-Seats (historical JD+)

- New acceptable data frequencies for seasonal adjustment and modelling of low frequency data:

  ```
  - In v3.x Low frequency data: $p$ in ${2,3,4,6,12}$ is admissible in all
  algorithms (historical and new)

  - In version 2, only Tramo-Seats supported all these frequencies, whereas
  X-13-ARIMA was restricted to $p$ in ${2,4,12}$
  ```

- outlier correction taken into account when selecting decomposition scheme
- ex-ante leap year correction added to Tramo-Seats (like in X-13)
- automatic trading day regressors selection from pre-defined sets built, according to groups of days
- specification: split into two distinct concepts, which can be directly manipulated by the user:
  - reference (or domain) specification: a global set of constraints inside of which estimation will be performed
  - point (or estimation) specification: contains all parameter choices resulting from estimation

The user can transform a given "estimation specification" in a user defined specification.

New algorithms in v3.x

Tramo-Seats and X-13-ARIMA share a very similar and sophisticated pre-adjustment process for the ARIMA model selection phase.

For new algorithms, the philosophy is to offer

- a simplified pre-adjustment on the ARIMA modelling side, reduced to airline model
- several enhanced decomposition options
  - stl+ ("+" stands for airline based pre-adjustment)

- x12+: airline based pre-adjustment + new trend estimation filters (Local Polynomials)
- seats+ (to come in the target v3 version): airline based pre-adjustment + AMB decomposition

SA with Basic Structural Models (BSM) available in GUI

In version 3.x, SA with Basic Structural Models is a fully integrated process with outlier detection, calendar correction and options on external regressors.

Fundamentally it is a one-step estimation, performing pre-adjustment and decomposition (with explicit components) in the same run

This makes regression variable selection more complicated:

- first step: a variable selection is performed with a Tramo like airline model regression
- second: the entire structural model is estimated

In version 3.x this process is available from the graphical user interface.

SA algorithms extended for high-frequency data

All algorithms are available via an R package and will be available in GUI (in target v 3.x version)

- Extended Airline estimation, reg-ARIMA like (`rjd3highfreq` and GUI)
- Extended Airline Decomposition, Seats like (`rjd3highfreq` and GUI)
- MX12+ (`rjd3x11plus`, GUI upcoming)
- MSTL+ (`rjd3stl` and in GUI)
- MSTS (`rjd3sts`, GUI upcoming)

Algorithm	Access in GUI	Access in R (v2)	Access in R (v3)

Modelling Algorithms

Algorithm	Access in GUI	Access in R (v2)	Access in R (v3)
Reg-ARIMA	✔	RJDemetra	rjd3x13
Tramo	✔	RJDemetra	rjd3tramoseats
Extended Airline	✔ (v3 only)	✖	rjd3highfreq
STS	✔ (v3 only)	rjdsts (deprecated)	rjd3sts

New SA (mass) production possibilities

New R Tools for wrangling workspaces

With functions for

- changing raw data path
- customizing specifications
- merging workspaces by series names, as you would do with a data table

These functions are in `rjd3providers` and `rjd3workspace` packages, (already in a v 2.x stable precursor `rjdworkspace`)

Production fully in R

without a workspace structure

- TS objects and full flexibility for customizing specifications
- new R functions enabling to apply revision policies (`rjd3x13::refresh` and `rjd3tramoseats::refresh`), with even more flexibility on data spans

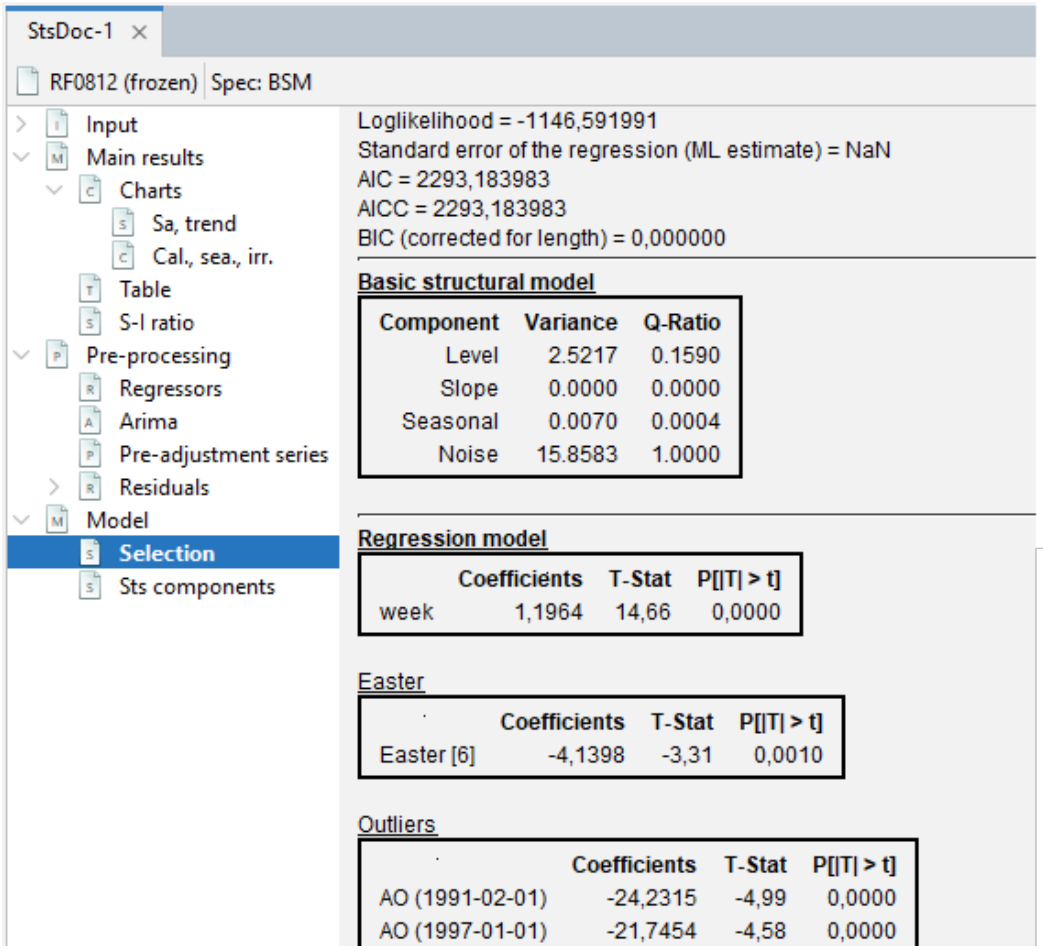


Figure 1: BSM output view in GUI

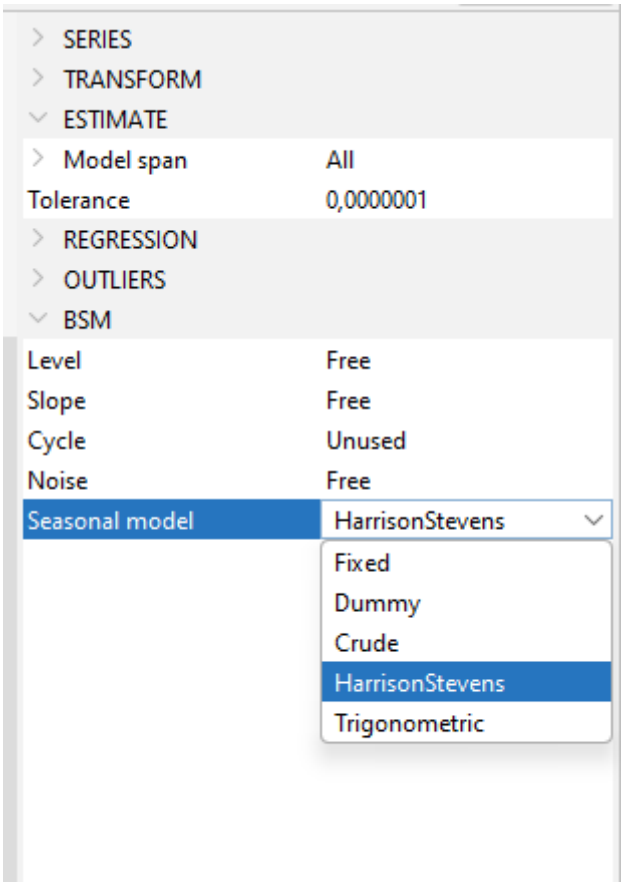


Figure 2: BSM specification box

Inherent shortcoming: data no readable by GUI, depriving of more sophisticated and visual feedback (compared to R) for manual fine tuning.

Solution : new R functions to create GUI readable dynamic workspaces on the fly (in aforementioned packages).

In the target 3.x, additional algorithms (X12+, STL+, BSM) will also be usable in production with a workspace and cruncher (on low frequency data)

Time series general purpose tools

Version 3.x offers more stand alone tools (mainly in `rjd3toolkit`)

- Tests (seasonality, auto-correlation, normality, randomness...)
- (Fast) ARIMA Modelling
- Flexible Calendar regressors generation
- Auxiliary variables for pre-adjustment
- Spectral analysis (in GUI)
- Detection of multiple seasonal patterns (Canova-Hansen test)
- State space frame work as a toolbox (`rjd3sts`)

Canova-Hansen test to identify multiple seasonal patterns

```
rjd3toolkit::seasonality_canovahansen(
    data = df_daily$births,
    p0 = min(ch.sp),
    p1 = max(ch.sp),
    np = max(ch.sp) - min(ch.sp) + 1
)
```

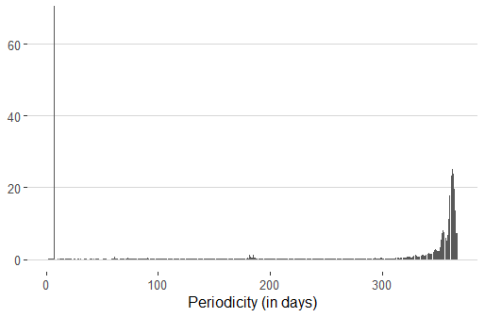


Figure 3: Canova Hansen seasonality test

Underway developments

- Moving Trading Days module integrated in all SA algorithms (for low and high frequency data), with two implementations one based on rolling windows and one on state space modelling
- Using Cubic Splines for smoother seasonal factors estimation of long periodicities ($p = 365, 25$)

Part I

Algorithms

This part provides practical guidance for using all the algorithms featured in JDemetra+, be it with the Graphical User Interface or R packages

Further methodological insights on each algorithm can be found in the Methods part of this book, whereas detailed description of all the available tools allowing to access the algorithms can be found in the Tools part.

JDemetra+ provides algorithms in the following domains:

Seasonal Adjustment (SA)

- Seasonal Adjustment overview
- Pre-treatment
- SA: X11 decomposition
- SA: Seats-decomposition
- SA: Revision policies
- SA of High-Frequency Data
- SA: STL+ and MSTL+
- X12+ and MX12+
- STS and MSTS

Modelling and Auxiliary Variables

- Reg-ARIMA modelling
- Outlier detection and external regressors
- Calendar correction

And also

- Benchmarking and temporal disaggregation
- Trend-Cycle estimation
- Revision analysis
- Nowcasting

Modelling

In this chapter

This chapter gives an overview of modelling algorithms available in JDemetra+.

Modelling features can be used stand alone or as pre-treatment (first step of seasonal adjustment).

Details on modelling (specifications and output characteristics) are provided in the pre-treatment chapter as they relate to the same procedure. High-Frequency (infra-monthly) data is tackled here.

Additional methodological explanations are covered in this chapter and in this one for high-frequency data.

Modelling Algorithms

Algorithm	Access in GUI	Access in R (v2)	Access in R (v3)
Reg-ARIMA	✔	RJDemetra	rjd3x13
Tramo	✔	RJDemetra	rjd3tramoseats
Extended Airline	✔ (v3 only)	✖	rjd3highfreq
STS	✔ (v3 only)	rjdsts (deprecated)	rjd3sts

Steps to use Reg-ARIMA and Tramo in a pre-treatment context are described here.

The possibility of saving parameters and generating output in the GUI is different when using modelling methods directly or seasonal adjustment process as explained here.

Extended Airline Model allows to handle infra-monthly series in a restricted reg-ARIMA framework, more details here.

Structural time series (STS) allow another kind of modelling using state space framework, more details here.

Practical Reg-ARIMA modelling

For the user not needing seasonal adjustment, the sections below highlight the functions or steps allowing to perform reg-ARIMA (or Tramo) as a stand alone goal, outside of a seasonal adjustment process.

In R

In version 2

```
# Reg-ARIMA
regA_v2 <- RJDemetra::regarima_x13(raw_series, spec = "RG5c")
# Tramo
tramo_v2 <- RJDemetra::regarima_tramoseats(raw_series, spec = "TRfull")
```

Full documentation of `RJDemetra::regarima()` function can be found here

Full documentation of `RJDemetra::regarima_tramoseats()` function can be found here

In version 3

```
# Reg-ARIMA
sa_regarima_v3 <- rjd3x13::regarima(raw_series, spec = "RG5c")

# Tramo
sa_tramo_v3 <- rjd3tramoseats::tramo(raw_series, spec = "TRfull")
```

Full documentation of 'rjd3x13::regarima' function can be found here

Full documentation of 'rjd3tramoseats::tramo' function can be found here

GUI

In the graphical user interface modelling algorithms can be accessed in two ways described below. In both cases a document (`.xml` file) will be generated and can be save in the workspace. The output (series, parameters and diagnostics) cannot be exported as files, just directly copied from the interface. Further details on window structure in GUI are available here.

Modelling in the Workspace Window

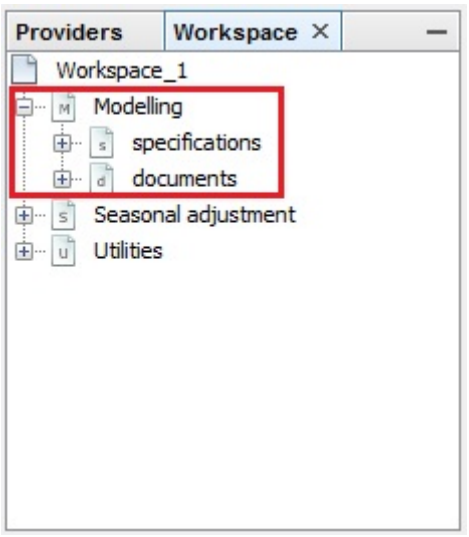


Figure 4: **The *Workspace* window with the nodes for the modelling procedure marked**

Modelling in Statistical Methods Panel

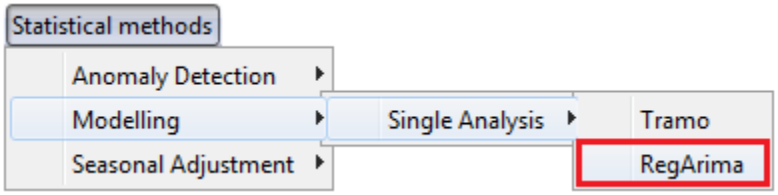


Figure 5: **The *RegARIMA* menu item**

Seasonal Adjustment (SA) Overview

In this chapter

This chapter is a first of a series focusing on the practical step by step use of JDemetra+ Seasonal Adjustment (SA) algorithms, restricted to monthly and quarterly series. For infra-monthly data see the following chapter. In the sections below an overview of the seasonal adjustment process is provided. The most widely used SA algorithms (Tramo-Seats and X-13-ARIMA) have two steps: a pre-treatment to remove (temporarily) deterministic effects and a decomposition phase to estimate the seasonal factors.

The following chapters get into the specifics of each algorithm.

- Pre-treatment
- SA: X11 decomposition
- SA: Seats-decomposition
- SA: Revision policies
- SA of High-Frequency Data
- SA: STL+ and MSTL+
- X12+ and MX12+
- STS and MSTS

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 of this book.

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, the user can refer to the Handbook on Seasonal Adjustment

SA process

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 evolutions of greater interest for the user such as short term evolution or long time trends.

When setting up the process:

- seasonality tests (can also be done in the frame of quality assessment at the end)
- trading days correction set up if relevant
  - regressors generation
  - regressors selection
- estimation with selected algorithm (see section below), might be automated with the cruncher
- quality report
- selective editing and manual fine tuning of parameters, re-estimation if needed
- updating when new data available with tailored revision policy

Seasonal Adjustment Algorithms

Algorithm	Version 2.x		Version 3.x	
	Access in GUI	Access in R	Access in GUI	Access in R
X-13 Arima	yes	RJDemetra	yes	`rjd3x13`
Tramo-Seats	yes	RJDemetra	yes	`rjd3tramoseats`
X12plus			yes	`rjd3x11plus`
STL			yes	`rjd3stl`
BSM			yes	`rjd3sts`
SEATS+			upcoming	upcoming

Two categories of algorithms :

- historical core (main): X-13-ARIMA and Tramo-Seats improved in version 3
- version 3 additional algorithms (incubator)

X-13-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+ combines STL local regression based decomposition and a simplified Reg-ARIMA pre-treatment restricted to airline models.

X12+ combines X11 (enhanced) decomposition and a simplified Reg-ARIMA pre-treatment restricted to airline models.

Seats+ combines Seats decomposition and a simplified Reg-ARIMA pre-treatment restricted to airline models.

In a Structural Time Series approach pre-treatment and decomposition are done simultaneously in a State Space Framework.

Admissible data frequencies

For low frequency data - in version 3.x p in $2, 3, 4, 6, 12$ is admissible in all algorithms - in version 3.x p in $2, 3, 4, 6, 12$ is admissible in Tramo-Seats and p in $2, 4, 12$ is admissible in X-13.

Algorithms extended for high-frequency (infra-monthly) data can be applied to "any periodicity" in their R version and to p in $7, 52.18, 365.25$ in the graphical user interface, see here for more details.

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 [1] are used in JDemetra+ :

- 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;

[1] other options as the log-additive model are also available in a more specific context described here

- 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.

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 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()`)

Full documentation of those functions can be found here

In GUI

How to perform tests in the graphical user interface is described here.

Direct or Indirect seasonal adjustment

when seasonally adjusting series which are aggregates following a given classification, the user has to chose whether to directly adjust the aggregate from its raw version or to aggregate the adjusted components.

The graphical user interface in version 2.x provides a module to compare the two options. It won't be provided in version 3.x.

SA: Pre-Treatment

In this chapter

The following sections cover pre-treatment with Reg-ARIMA (or Tramo) algorithms. Tramo and the Reg-ARIMA part of X-13-ARIMA rely on very similar principles. Thus Tramo will only be mentioned to highlight differences with the Reg-ARIMA part of X-13-ARIMA.

Reg-ARIMA modelling part can be the first of a seasonal adjustment process or run on its own. Below we focus on performing Reg-ARIMA modelling as pre-treatment in a SA processing.

More in-depth methodological explanations of the algorithms can be found in this part of the documentation.

Pre-treatment principles

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

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

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

Reallocation of pre-treatment effects

The linearised series ($Y_{lin,t}$) is be decomposed into unobservable components in the decomposition phase

$$Y_{lin} = S_{lin} + T_{lin} + I_{lin}$$

Pre-treatment effects are then reallocated to build the final components

$$S = S_{lin} + cal + out_s + reg_s$$

$$T = T_{lin} + out_t + reg_t$$

$$I = I_{lin} + out_i + reg_i$$

Where

- cal is the total calendar effect
- out_j is the total effect of outliers on component j
- reg_j is the total effect of user-defined regressors on component j

Setting Specifications

Default specifications are set for the whole SA procedure, pre-treatment and decomposition. They are slightly different for X-13-ARIMA and Tramo-Seats and can be modified with user-defined parameters.

Spec identifier	Log/level detection	Outliers detection	Calendar effects	ARIMA

Starting point for X-13-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
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 X-13-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

User-defined specifications

Principle oser 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, as detailed below.

The user may add new seasonal adjustment specifications to the *Workspace* window. To do it, go to the *Seasonal adjustment* section, right click on the *tramoseats* or *x13* item in the *specifications* node and select *New* from the local menu.

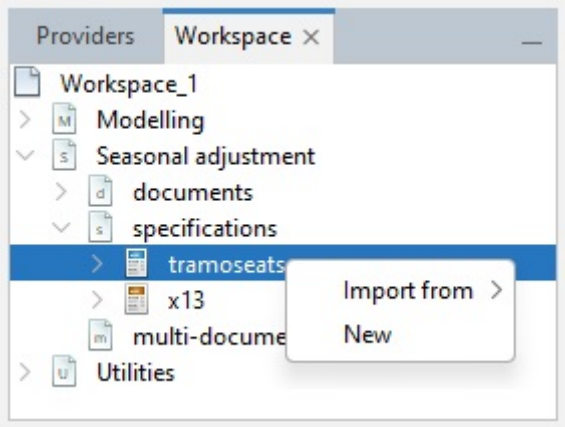


Figure 6: **Creating a new specification in the *Seasonal adjustment* section**

Next, double click on the newly created specification, change the settings accordingly and confirm with the **OK** button.

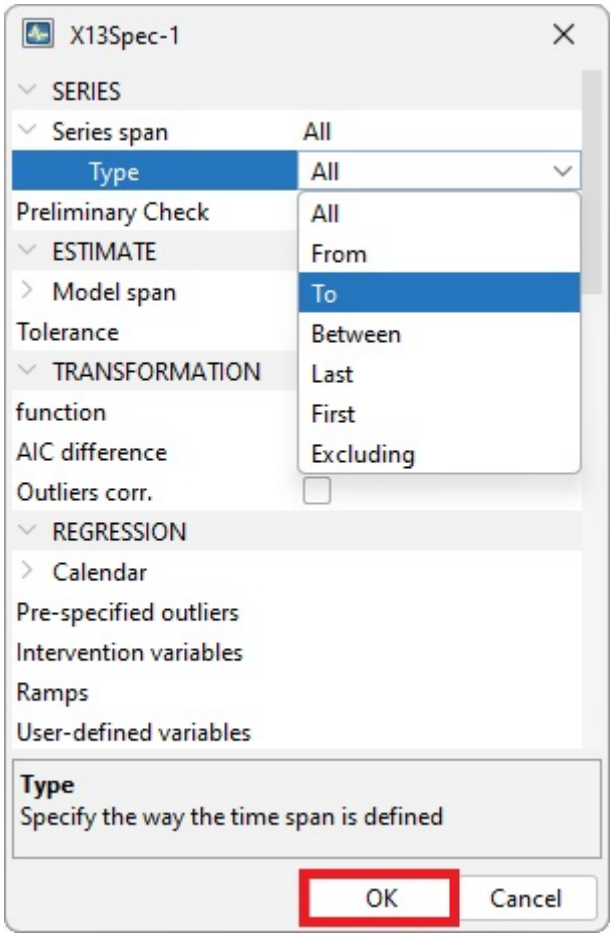


Figure 7: **Changing settings of seasonal adjustment specification**

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.

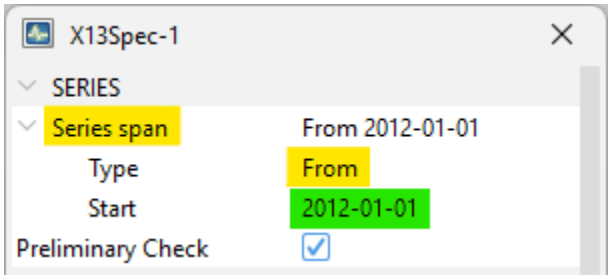


Figure 8: **Setting series span in R**

X-13 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", "RSA5"),
    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

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", "RSA5"),
```

Figure 9: **Setting in R**

```
    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

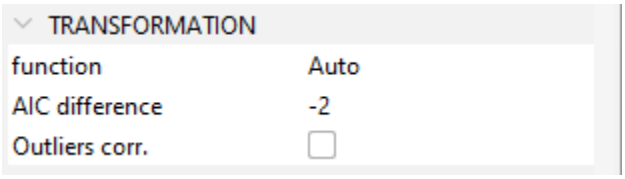


Figure 10: Model span setting

Set and in R

X-13

```
# 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", "RSA5"),
    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 X-13-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 Reg-ARIMA model after the test; "Remove" = the trading day variables belong to the initial regression model but can be removed from the Reg-ARIMA 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 Reg-ARIMA model after the test; "Remove" = the Easter effect variable belongs to the initial regression model but can be removed from the Reg-ARIMA 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

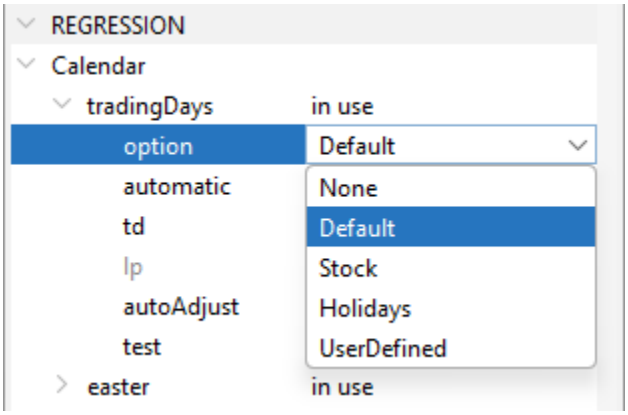


Figure 11: **STEP 1: Selection from JDemetra+ Default…or User_defined**

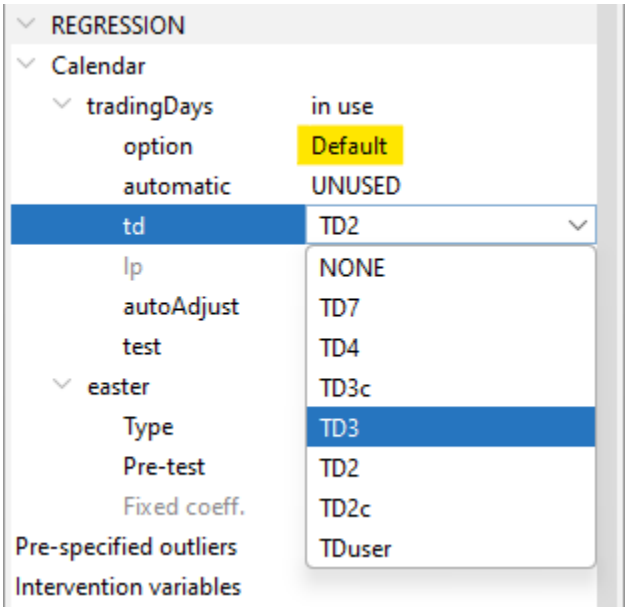


Figure 12: **STEP2: Calendar effects minus Easter are labeled *trading days***

0.0.0.0.2 * Holidays option

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

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_character_, "TradingDays", "WorkingDays", "UserDefined", "None"
tradingdays.autoadjust <- NA
tradingdays.leapyear <- c(NA_character_, "LeapYear", "LengthOfPeriod", "None")
tradingdays.stocktd <- NA_integer_
```

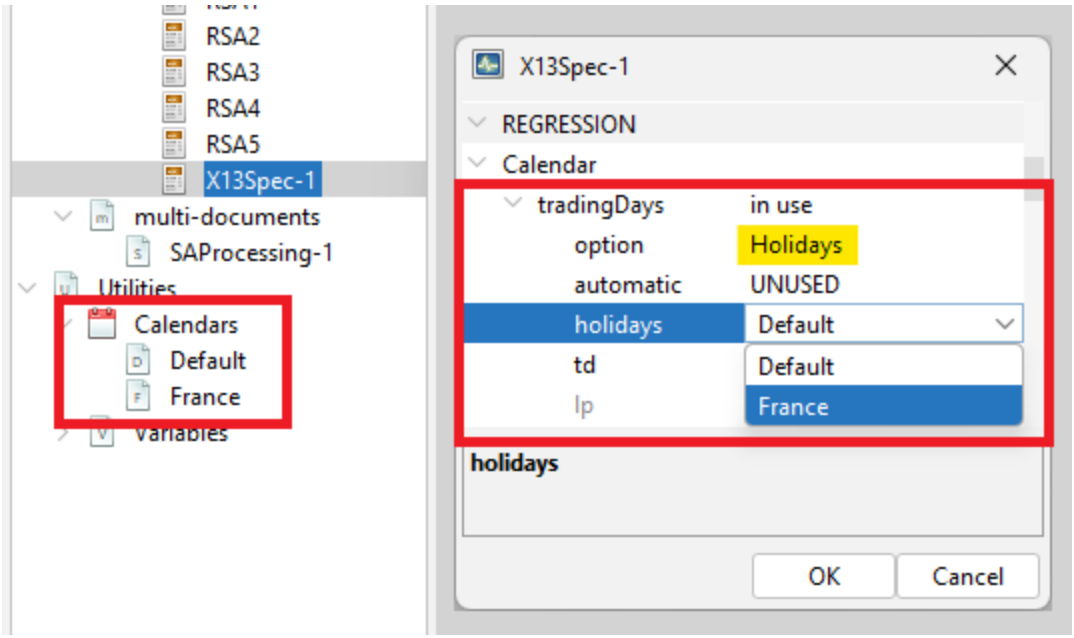


Figure 13: **The list of calendars displayed under *Holidays* option corresponds to the calendars defined in the *Workspace* window**

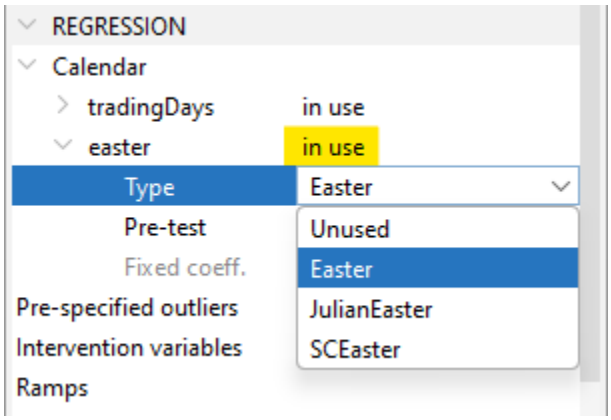


Figure 14: **easter Options**

```
tradingdays.test <- c(NA_character_, "Remove", "Add", "None")
easter.enabled <- NA
easter.julian <- NA
easter.duration <- NA_integer_
easter.test <- c(NA_character_, "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

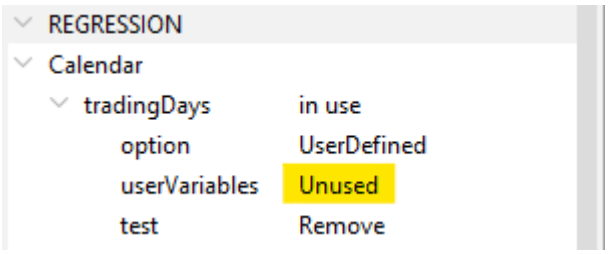
Step 1: import data set containing the regressors, general procedure explained here

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

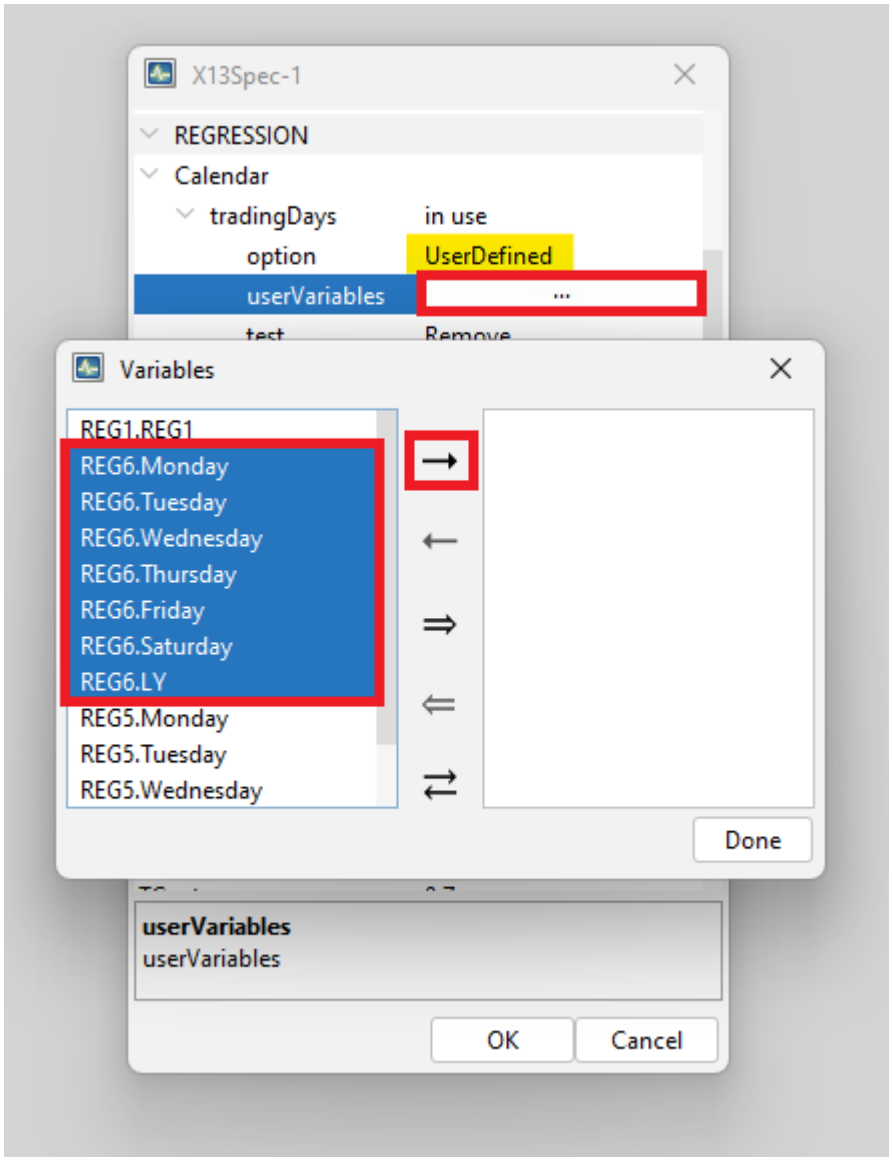
Step 3: Modify specifications Modifications are done the same way in a global specification (whole SAP) or series by series.

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

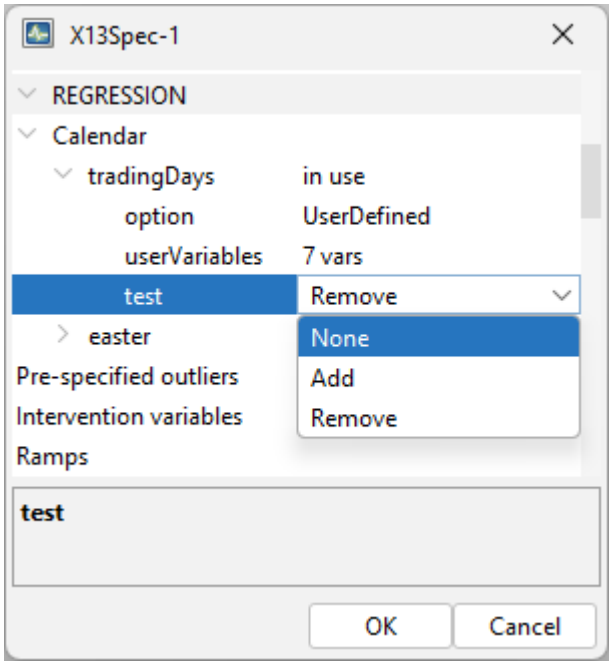
In the specification window, click right from “userVariables” on “Unused” to open the variable selection window



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

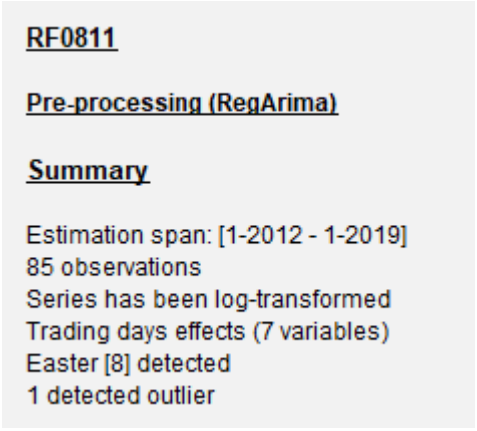
The following section details how to retrieve results (parameters, regressors, regression coefficients and tests) when using GUI or R interface.

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.



In R

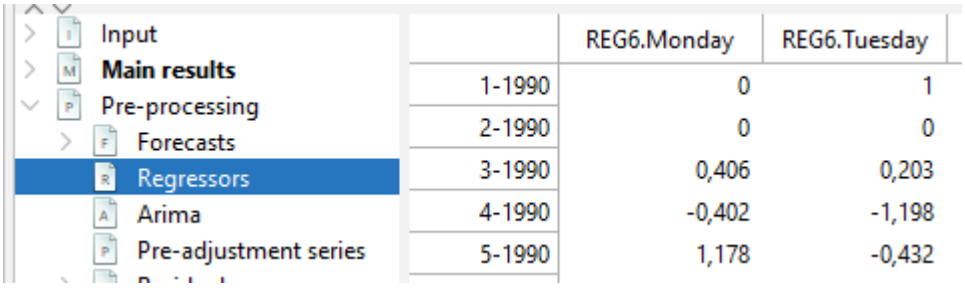
(to be added)

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.



	REG6.Monday	REG6.Tuesday
1-1990	0	1
2-1990	0	0
3-1990	0,406	0,203
4-1990	-0,402	-1,198
5-1990	1,178	-0,432

In R

(to be added)

Version 2

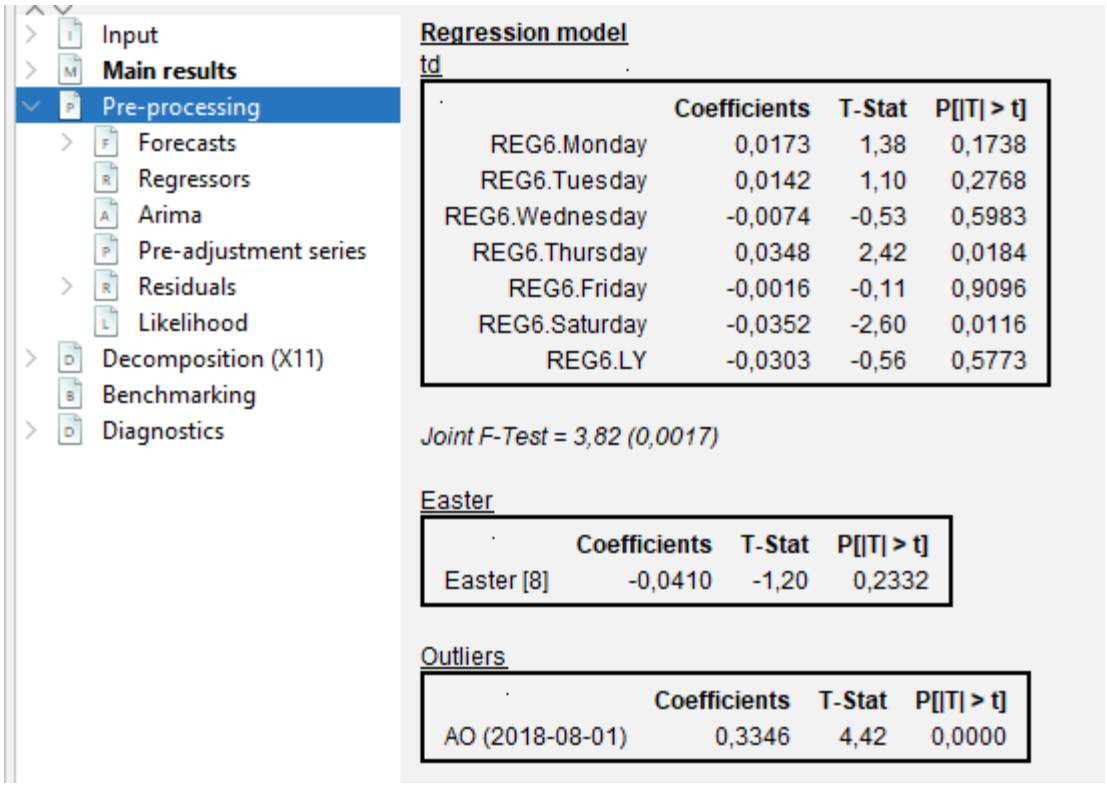
version 3

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

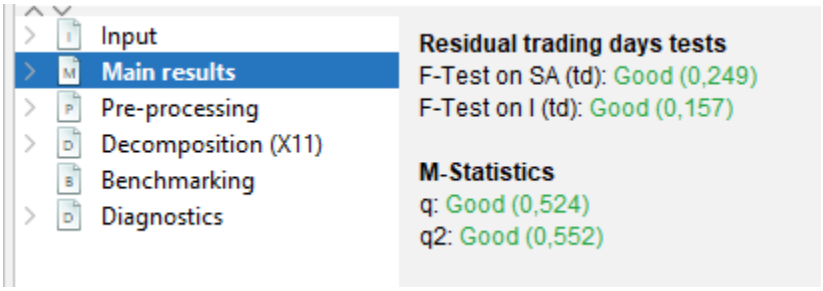


Figure 15: **F-Test results in GUI**

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 Reg-ARIMA (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 **Detection span** option. By default, the checkbox is marked, which implies that the automatic identification of outliers is enabled.

- **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

Pre-specified :

- Click on **...**
- Click on **+**
- Fill the outlier's information
- Click on **Ok**

To change the view to set the outliers, got to Tools –> Options

Then to **Demetra** –> **Pre-specified Outliers** –> **Calendar-like Grid** –> **Ok**

Then you have the calendar view (when selecting the pre-specified outliers) :

Setting in R

(to be added)

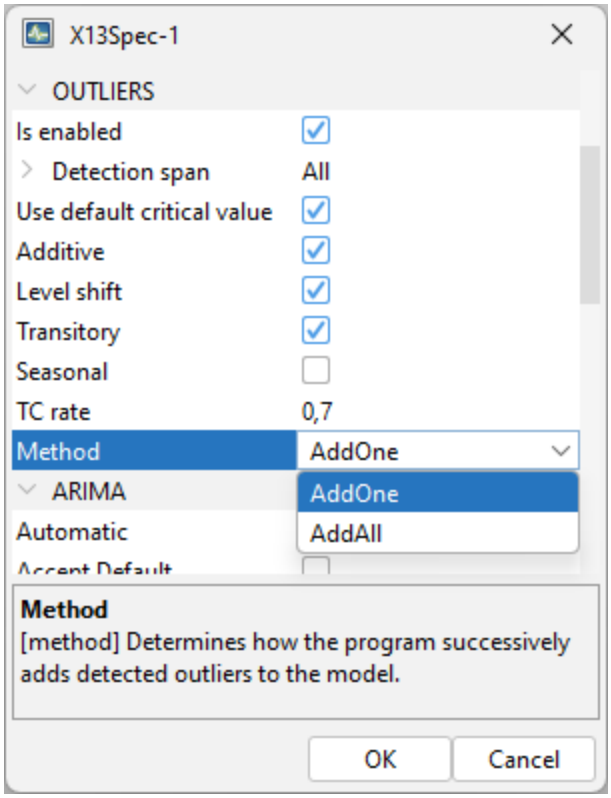


Figure 16: **Automatic detection of outliers in GUI**

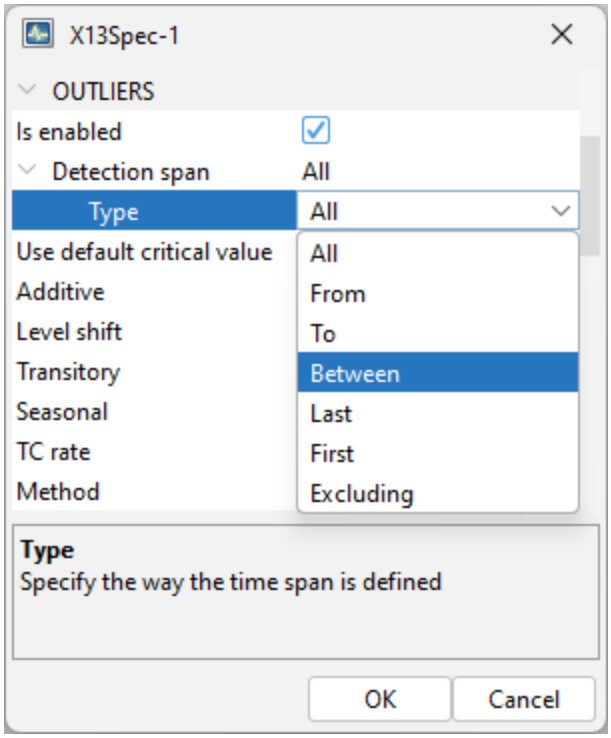


Figure 17: **Outliers types in GUI**

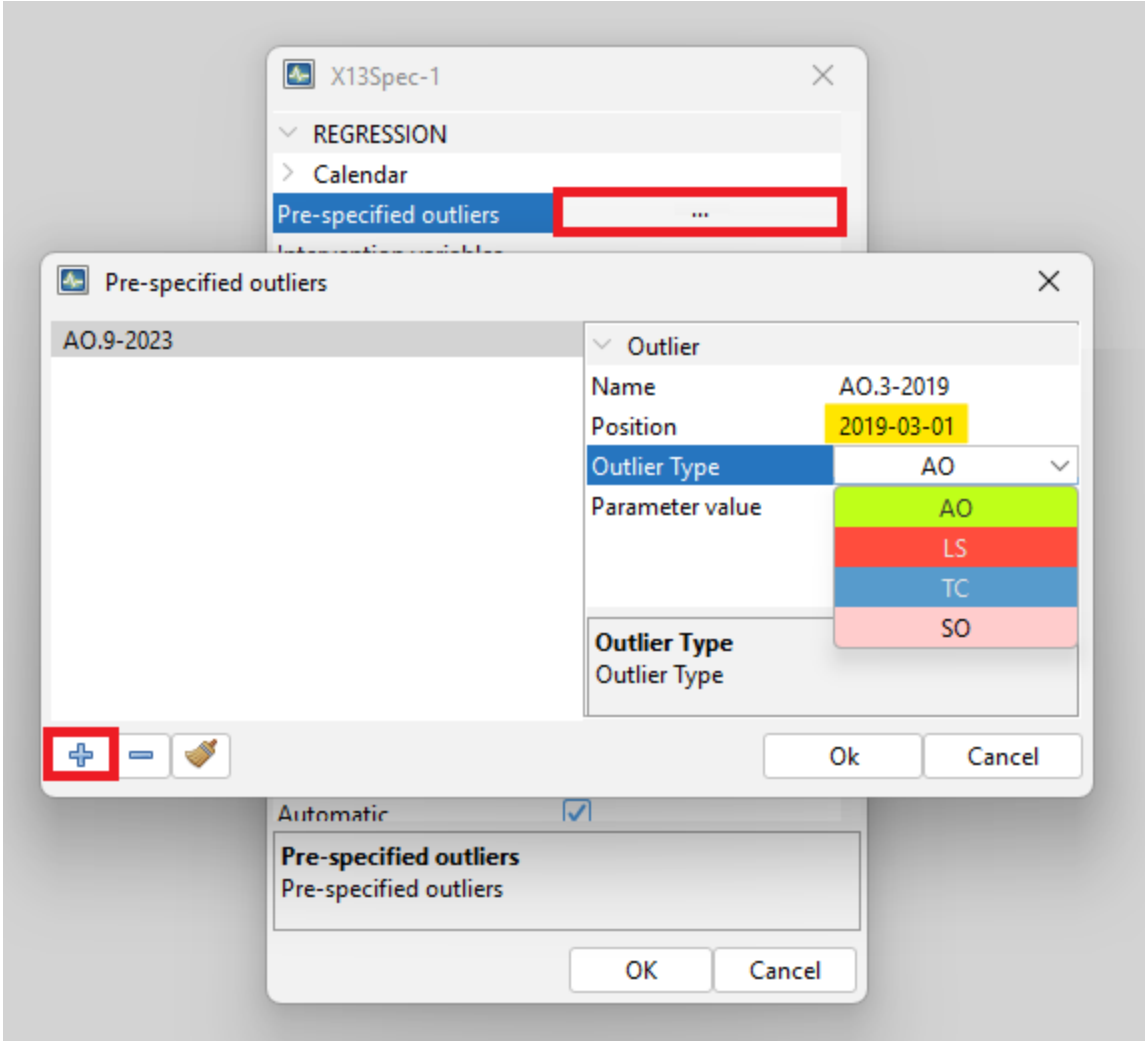


Figure 18: **Pre-specified outliers in GUI**

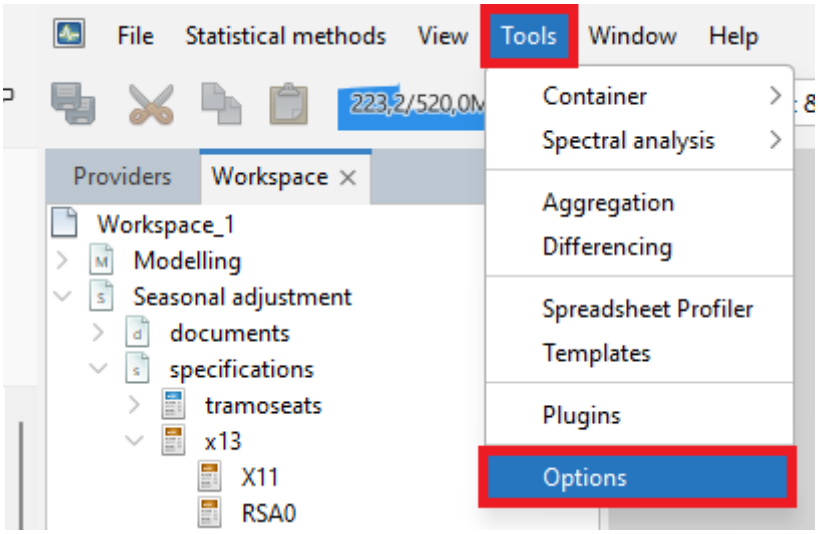


Figure 19: Options in GUI

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

In R

(to be added)

0.0.0.0.3 * Regression details

In GUI

In R

(to be added)

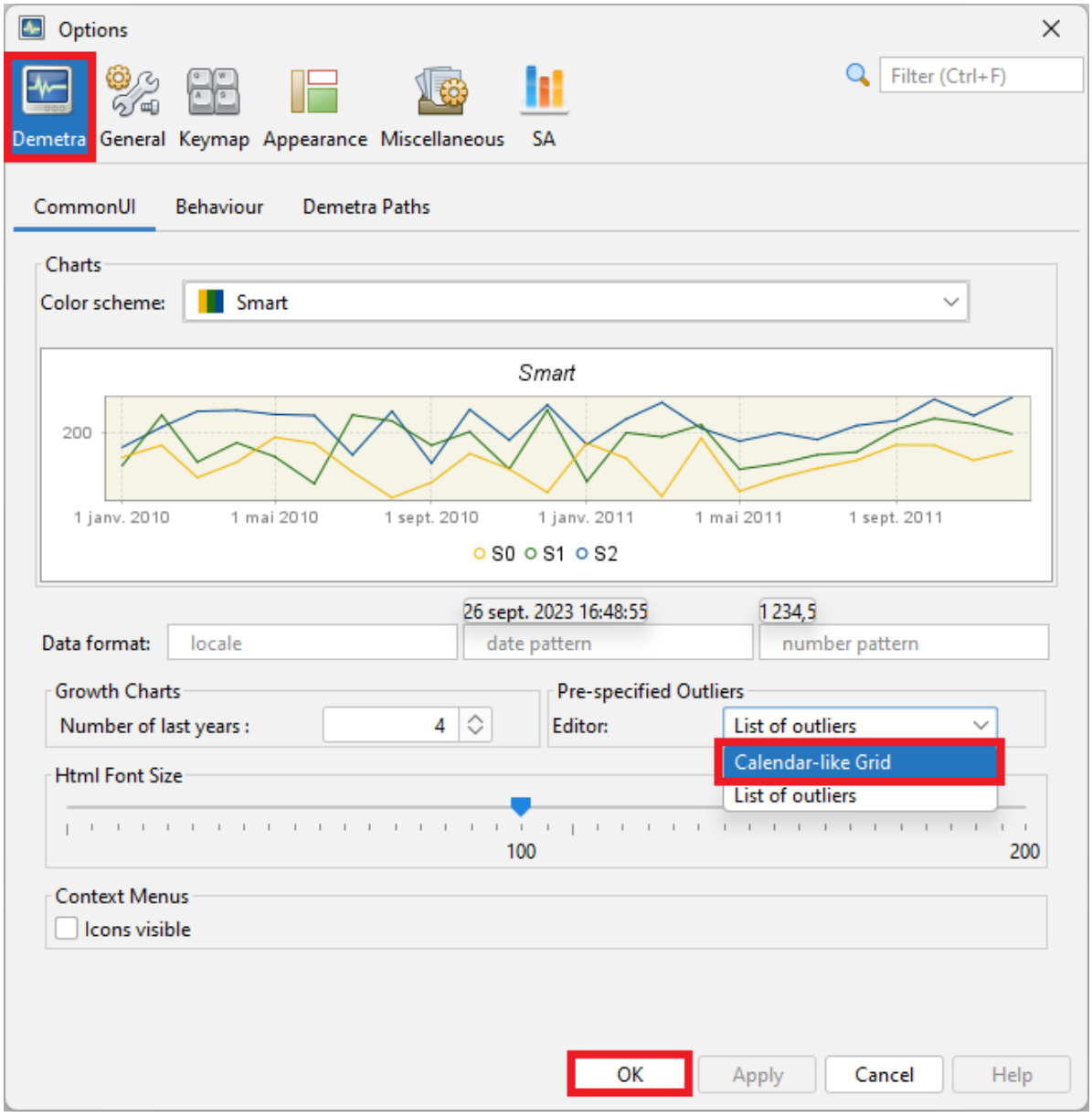


Figure 20: **Change view to calendar**

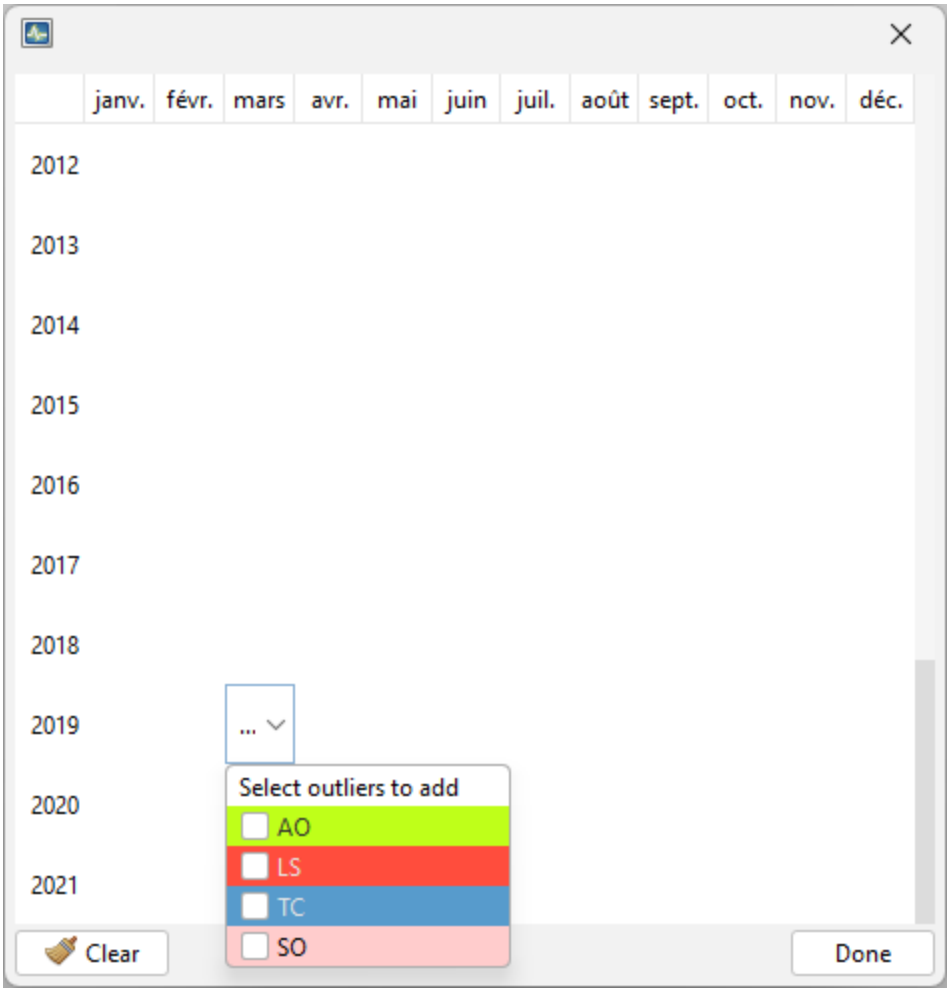


Figure 21: **Calendar view in GUI

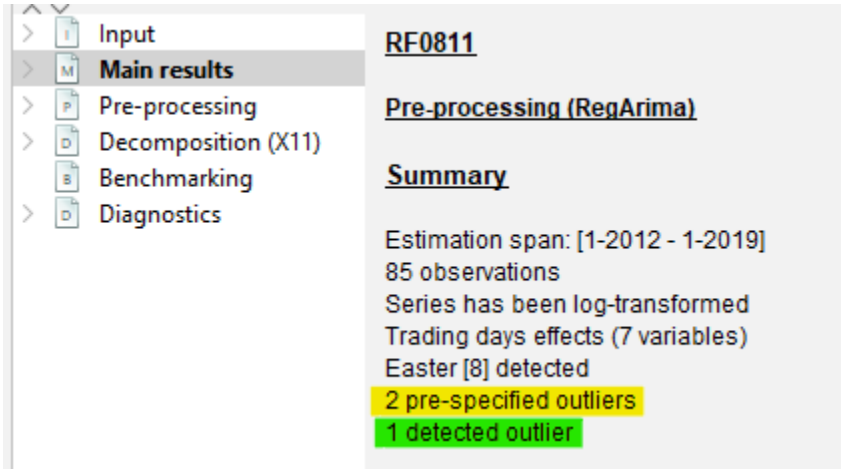
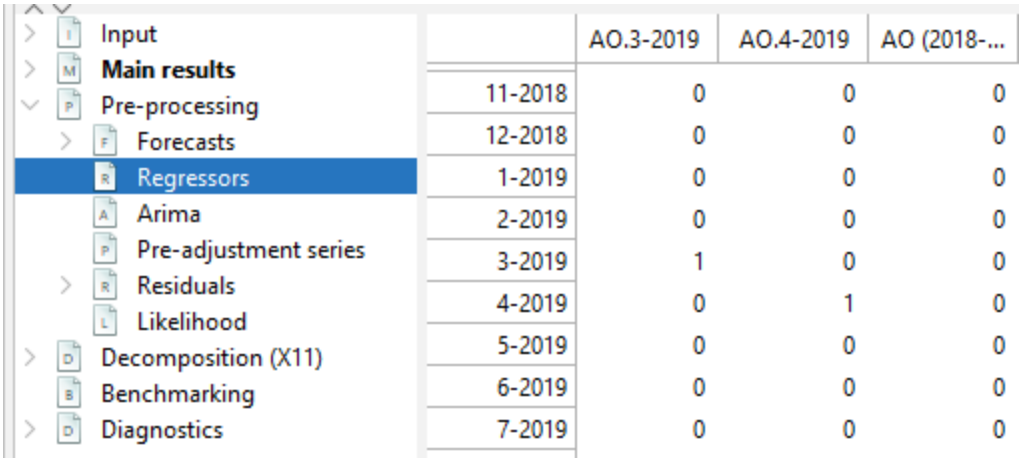


Figure 22: **Main results in GUI**



	AO.3-2019	AO.4-2019	AO (2018-...
11-2018	0	0	0
12-2018	0	0	0
1-2019	0	0	0
2-2019	0	0	0
3-2019	1	0	0
4-2019	0	1	0
5-2019	0	0	0
6-2019	0	0	0
7-2019	0	0	0

Figure 23: **Regressors in GUI**

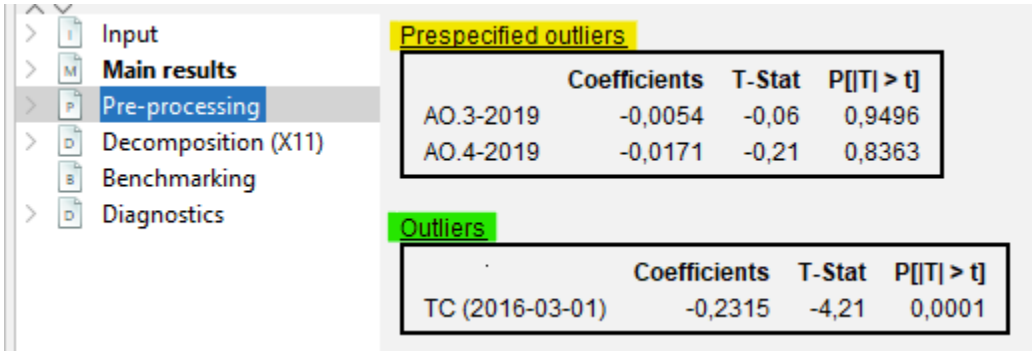


Figure 24: **Regression details in GUI**

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)

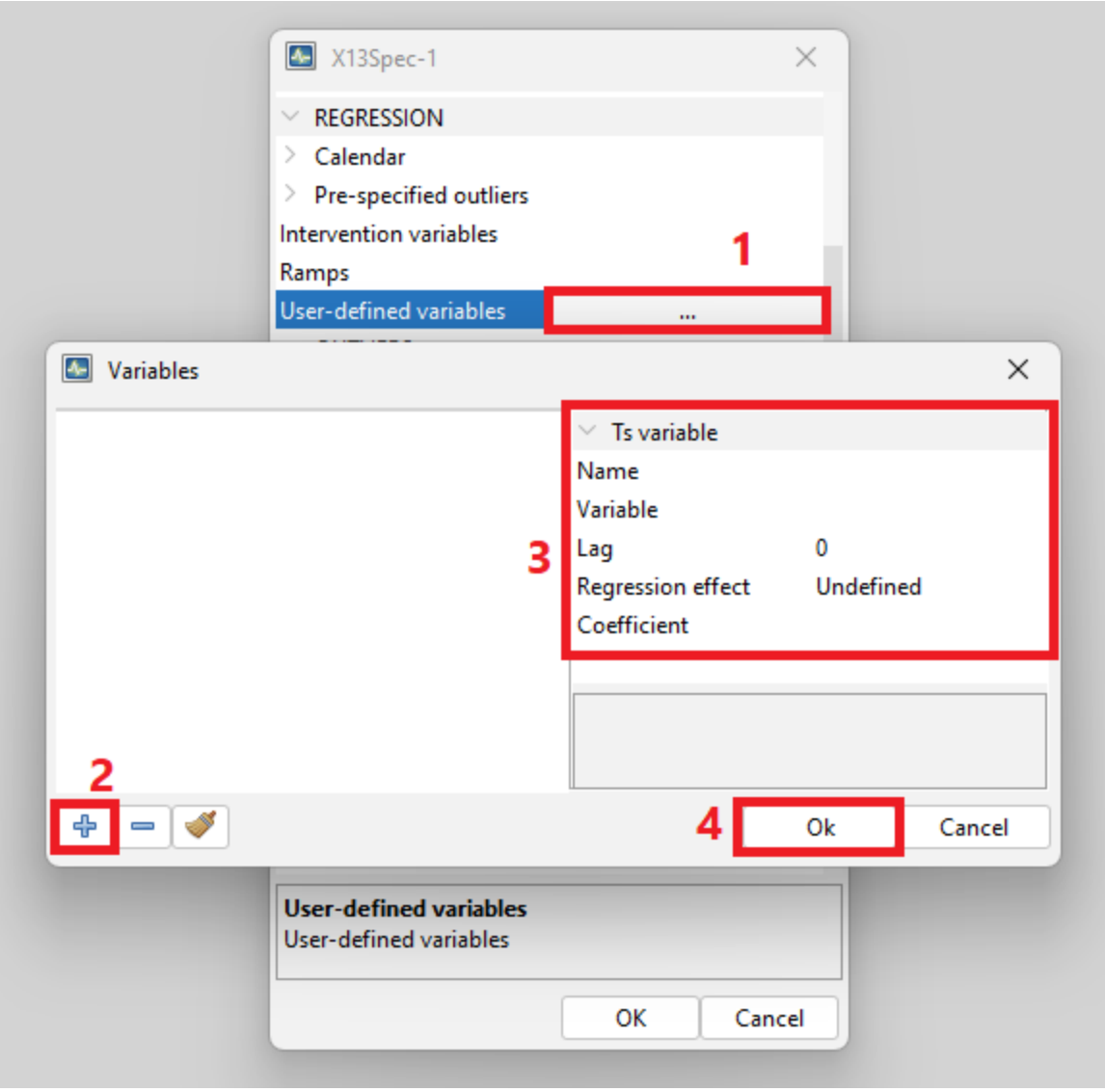


Figure 25: **Creation of user-defined variables in GUI**

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

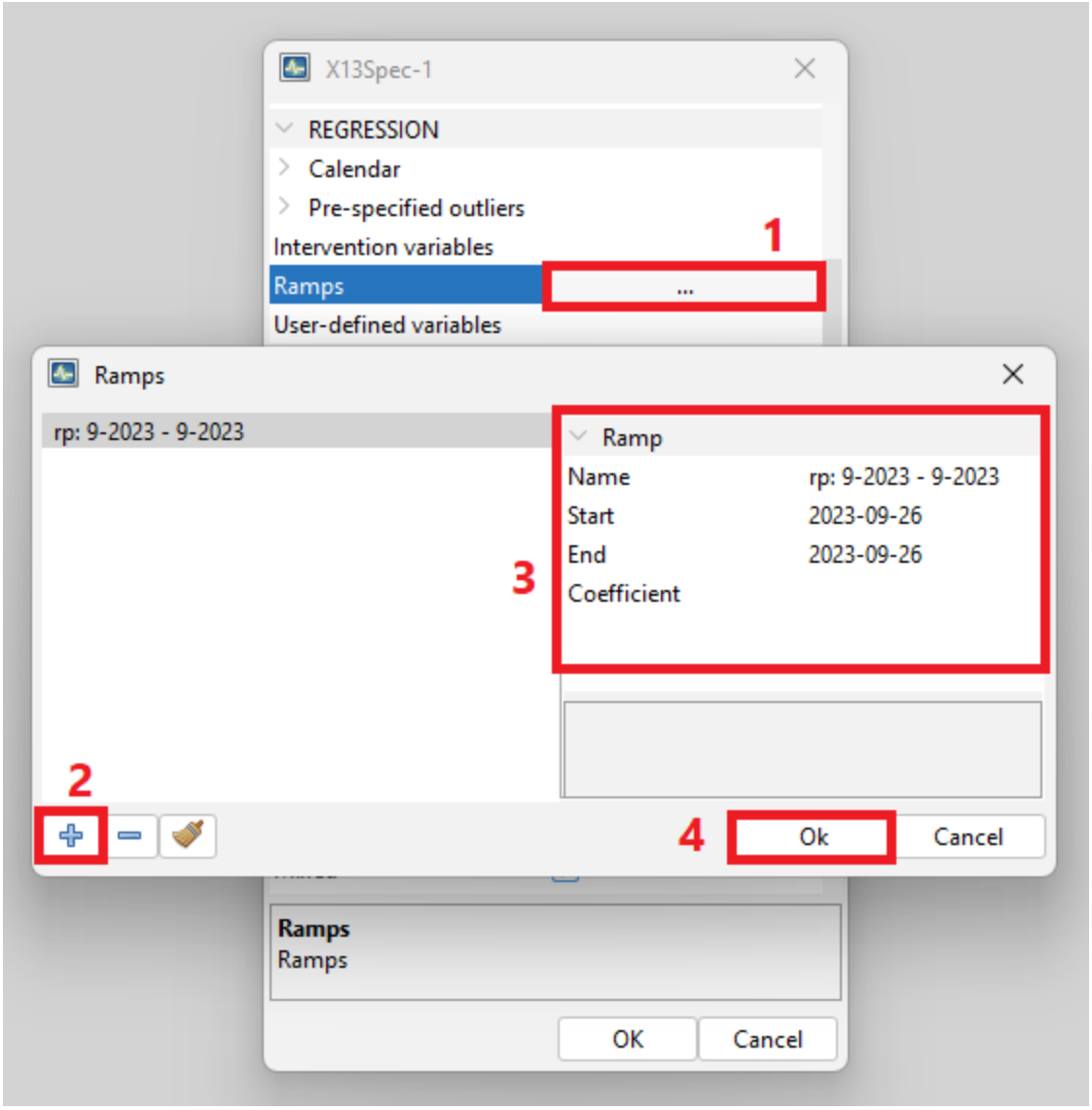


Figure 26: **Creation of ramp variable in GUI**

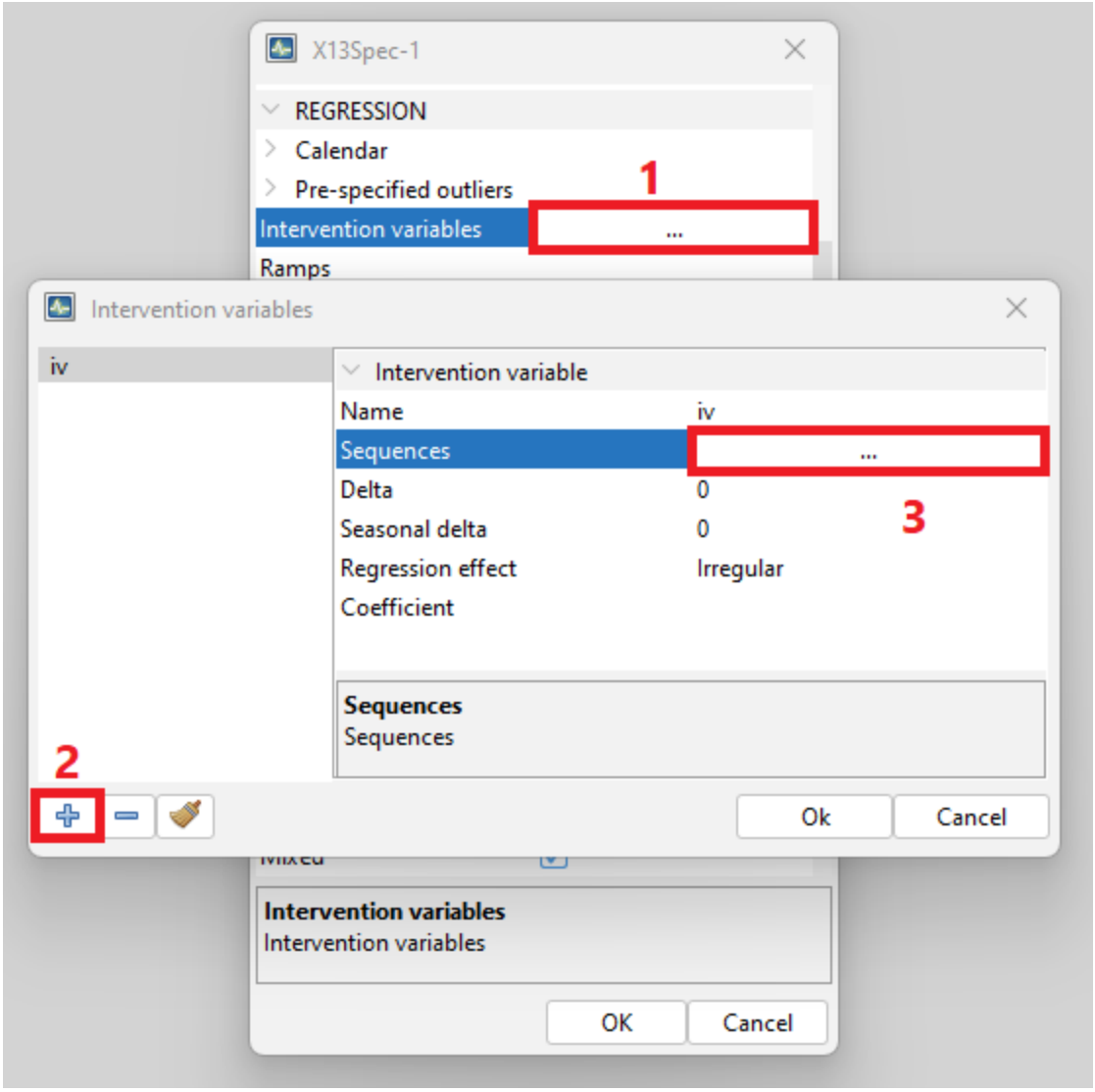


Figure 27: **Step 1**

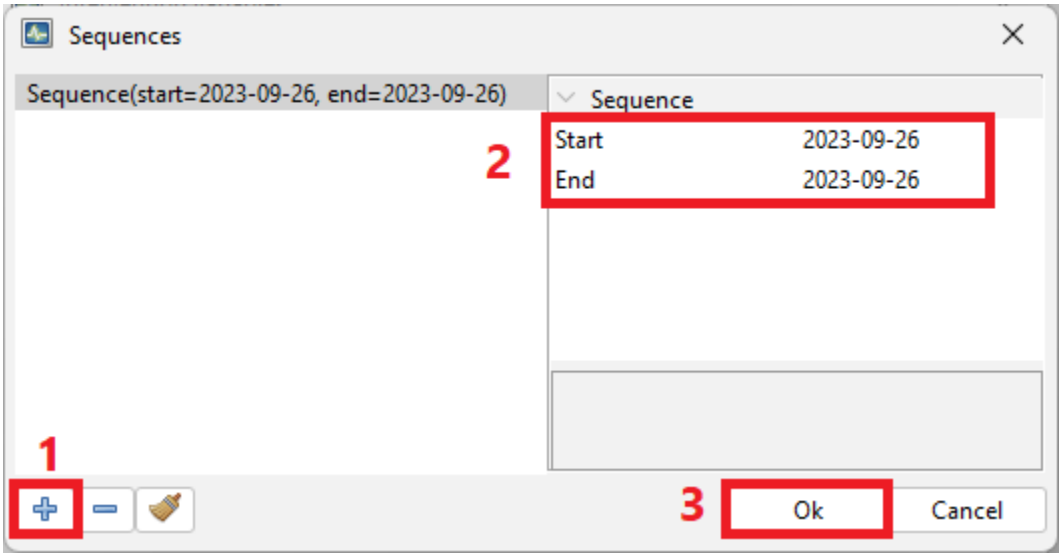


Figure 28: **Step 2**

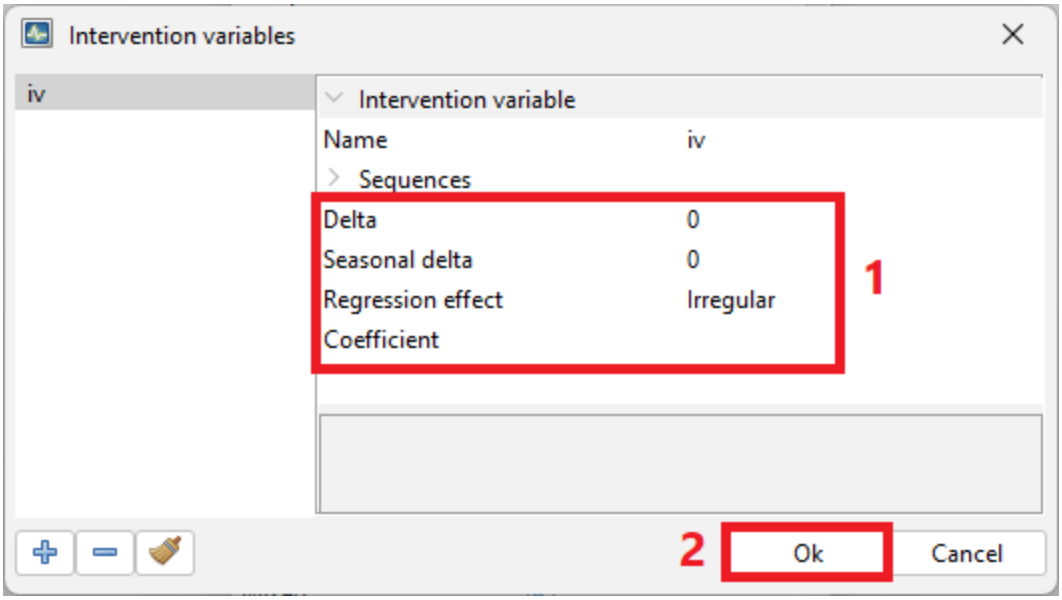


Figure 29: **Step 3**

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.

Retrieving Results

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

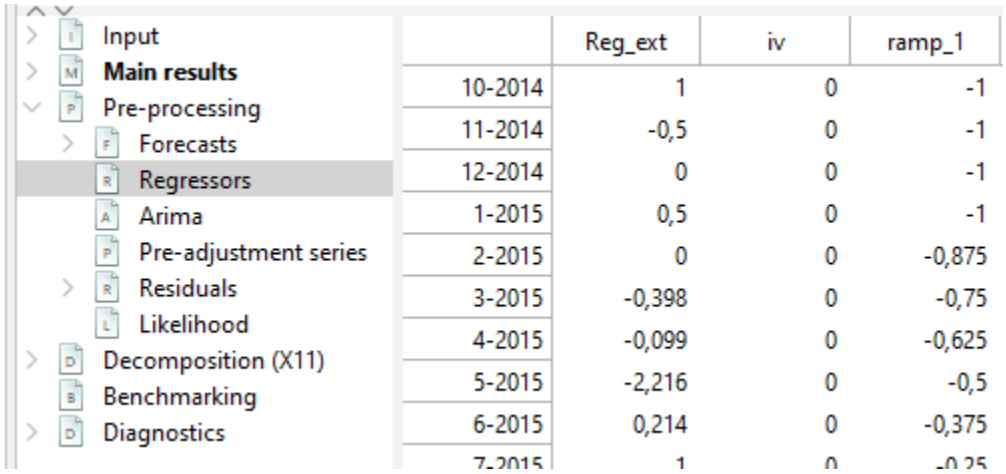
0.0.0.0.1 * Regressors

In GUI

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

In R

(to be added)



	Reg_ext	iv	ramp_1
10-2014	1	0	-1
11-2014	-0,5	0	-1
12-2014	0	0	-1
1-2015	0,5	0	-1
2-2015	0	0	-0,875
3-2015	-0,398	0	-0,75
4-2015	-0,099	0	-0,625
5-2015	-2,216	0	-0,5
6-2015	0,214	0	-0,375
7-2015	1	0	-0,25

Figure 30: **Regressors in GUI**

0.0.0.0.2 * Regression details

In GUI

Regression details are in the pre-processing pane.

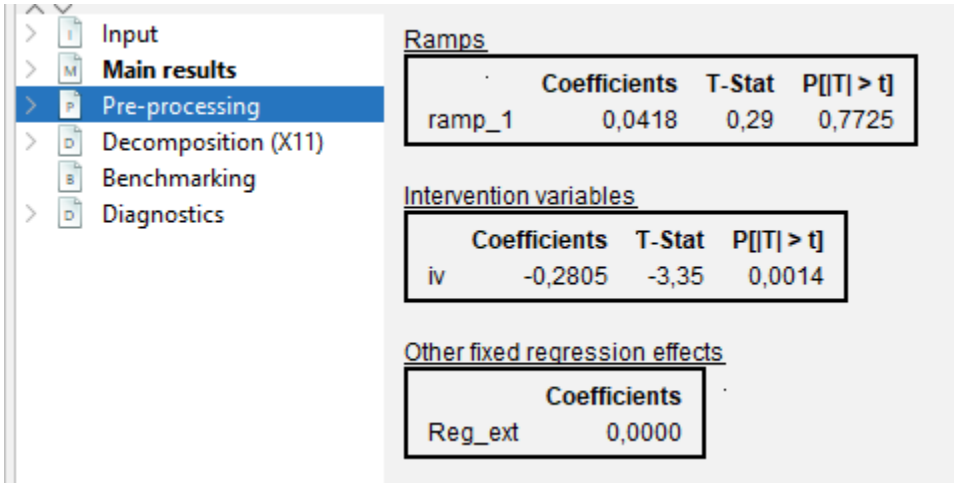


Figure 31: **Regression details**

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.

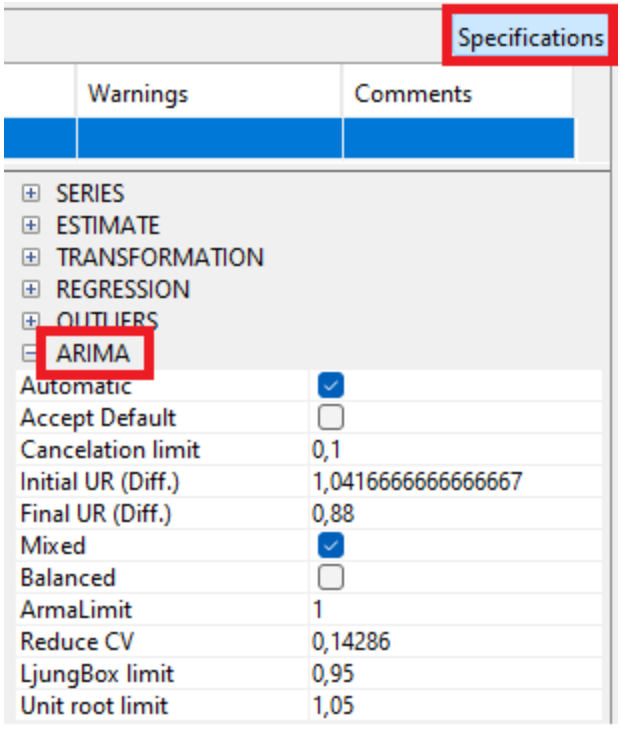


Figure 32: **Setting in GUI in v2**

0.0.0.1 v2

0.0.0.2 v3

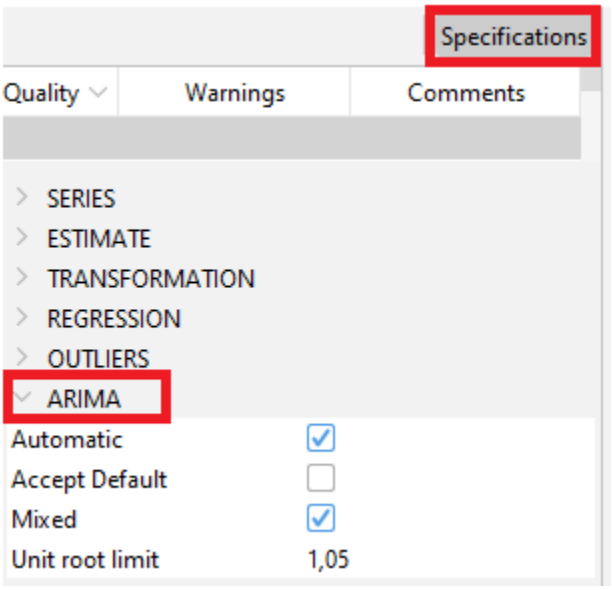


Figure 33: **Setting in GUI in v3**

Forecast horizon when using Tramo-Seats Is set in the decomposition part of the specification in GUI.

Setting in R (first template, then worked example) X-13-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_
```

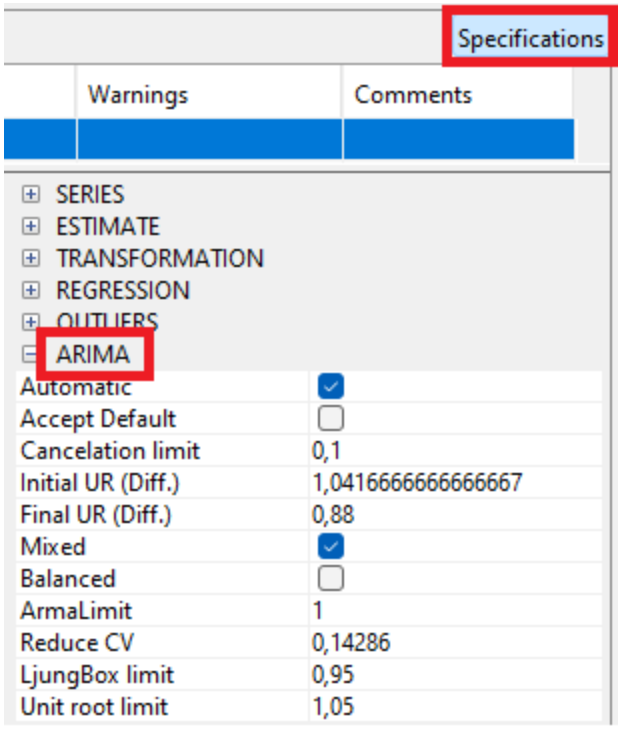


Figure 34: **Setting in GUI in v2**

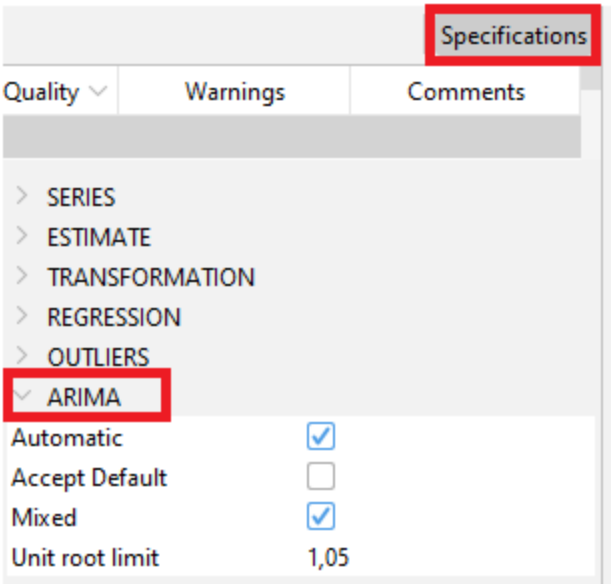


Figure 35: **Setting in GUI in v3**

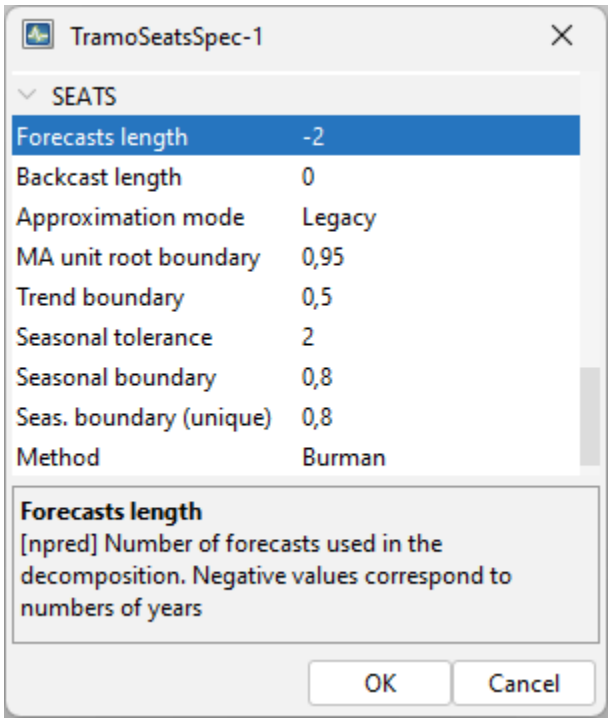


Figure 36: **Forecast horizon when using Tramo-Seats**

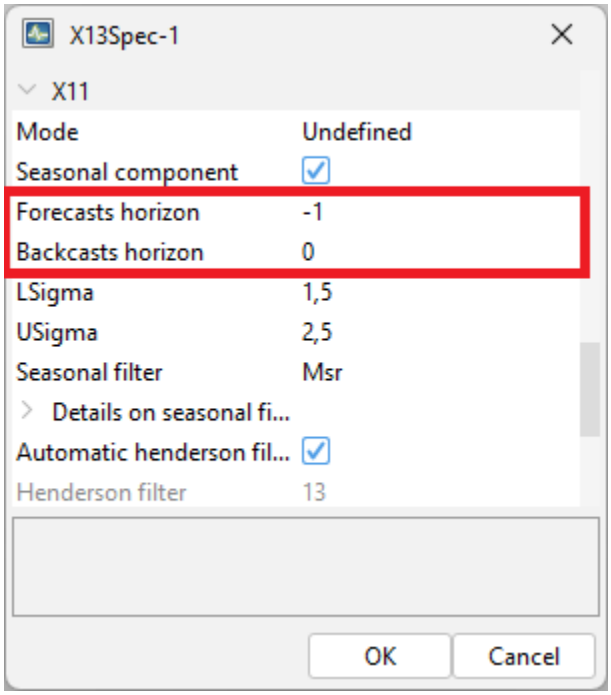


Figure 37: **Forecast horizon when using X-13-ARIMA**

```
)
```

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

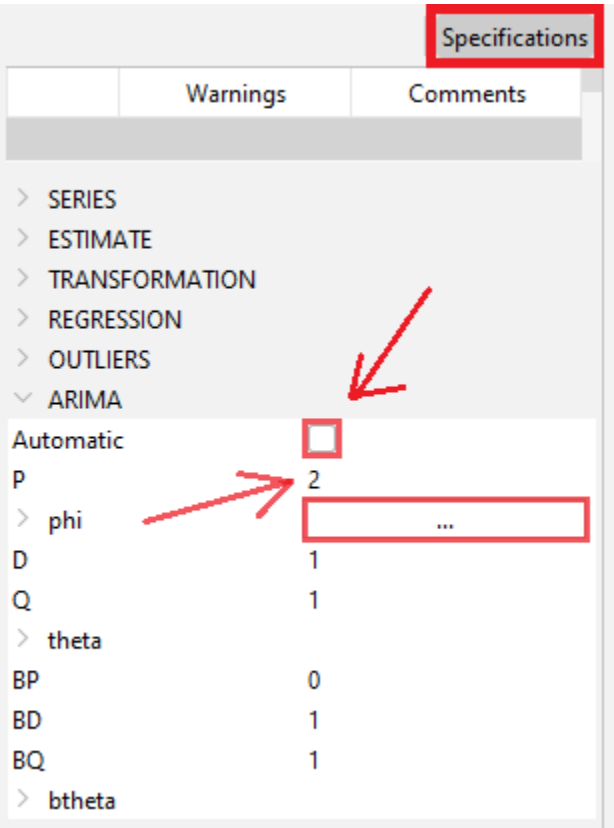


Figure 38: **Manual ARIMA modelling**

Setting in R

X-13-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_,
```

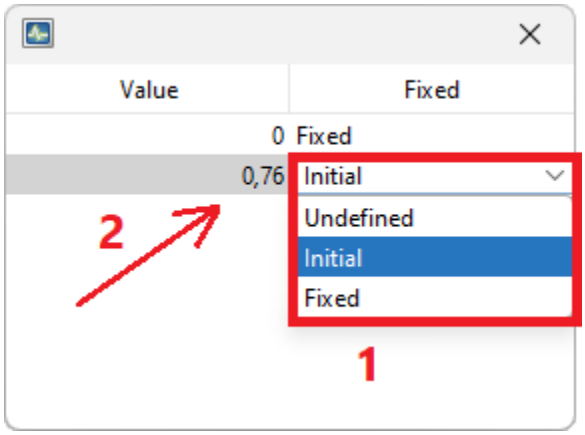


Figure 39: **Fixing coefficients**

```
    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

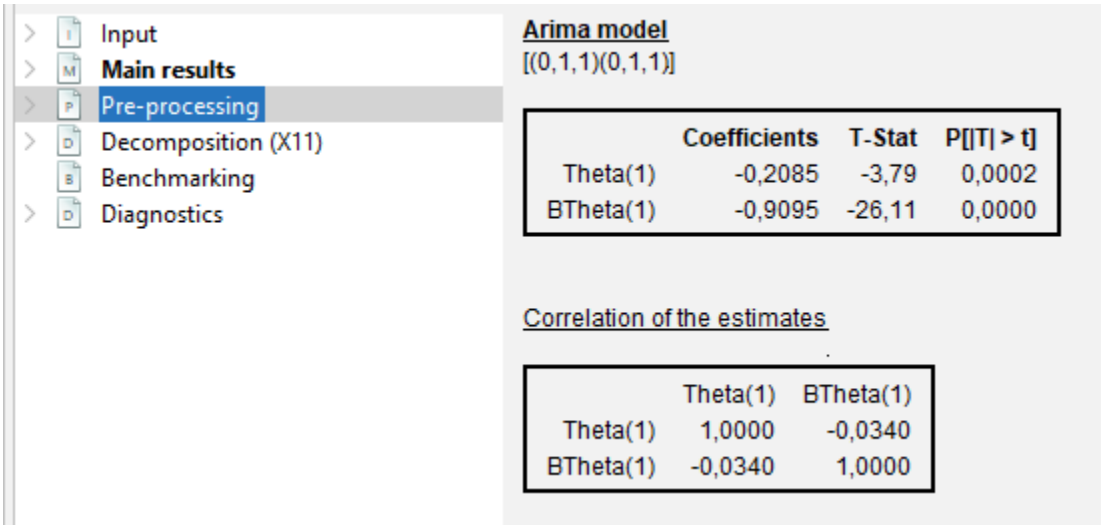


Figure 40: **Final model**

More details in Pre-processing/ARIMA Node

In residual Node

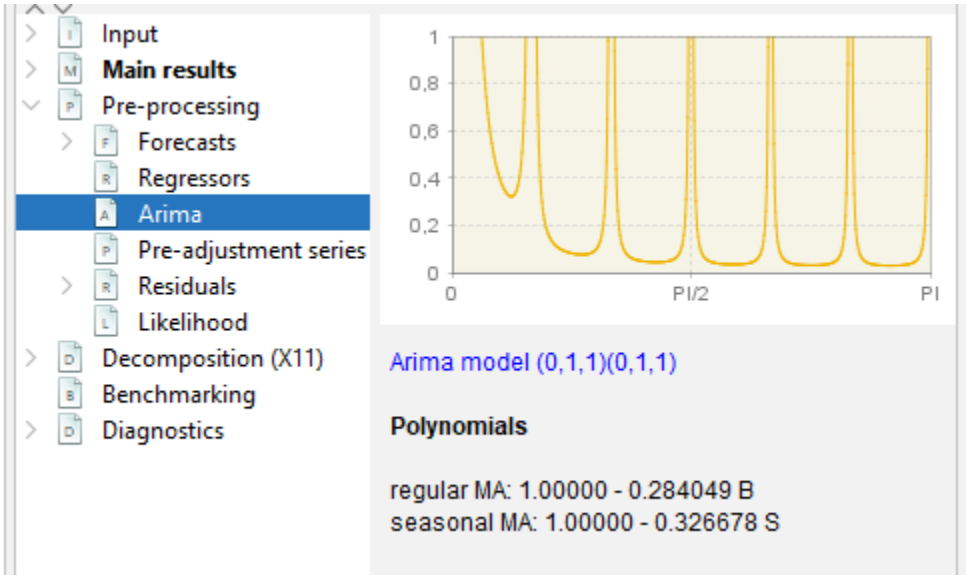


Figure 41: **ARIMA details**

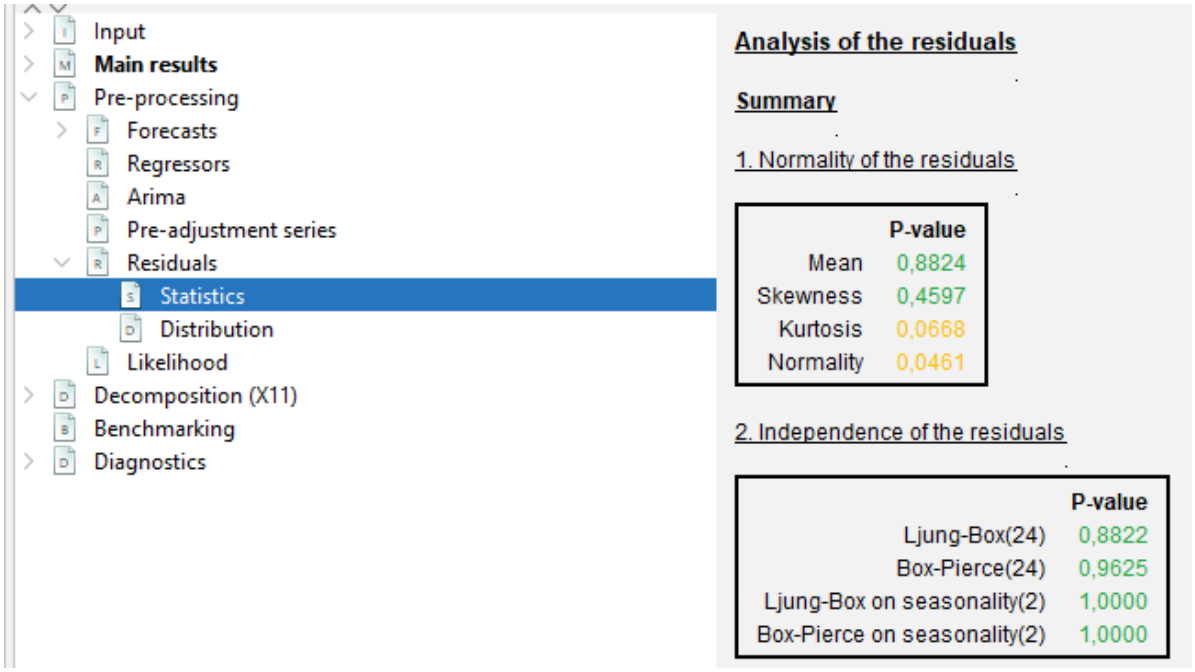


Figure 42: **Residuals**

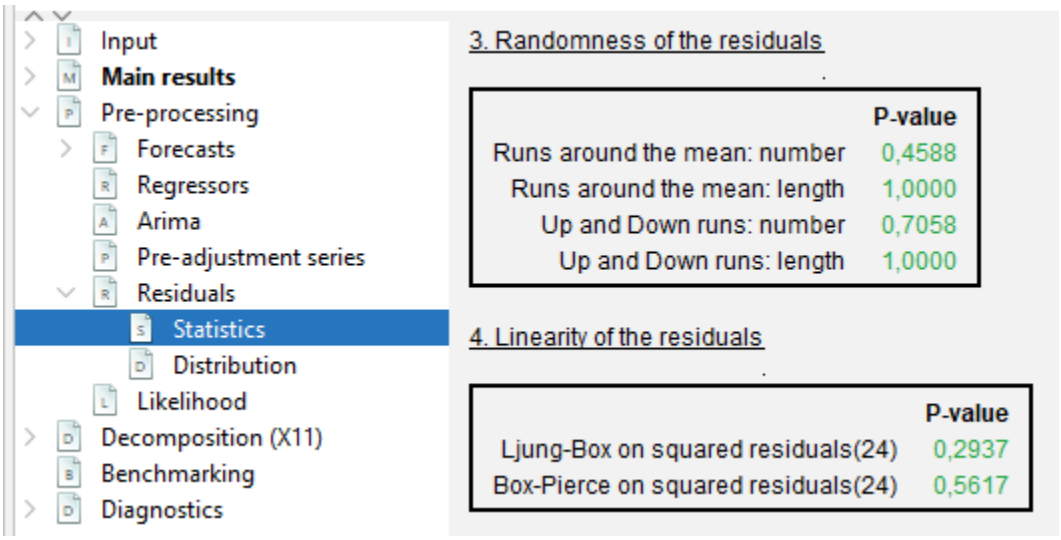


Figure 43: **Residuals**

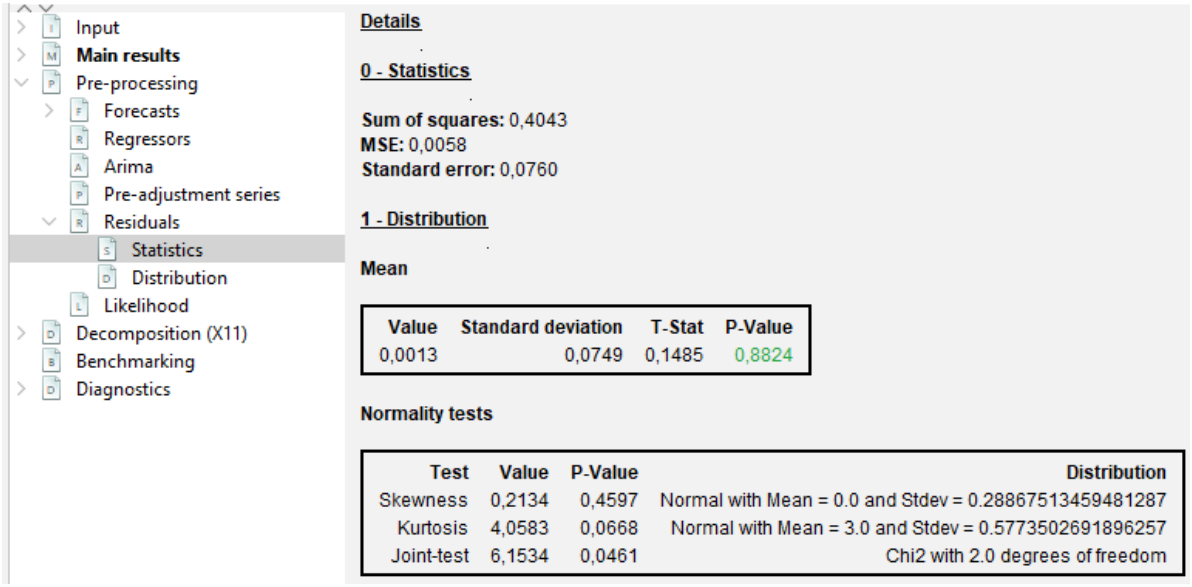
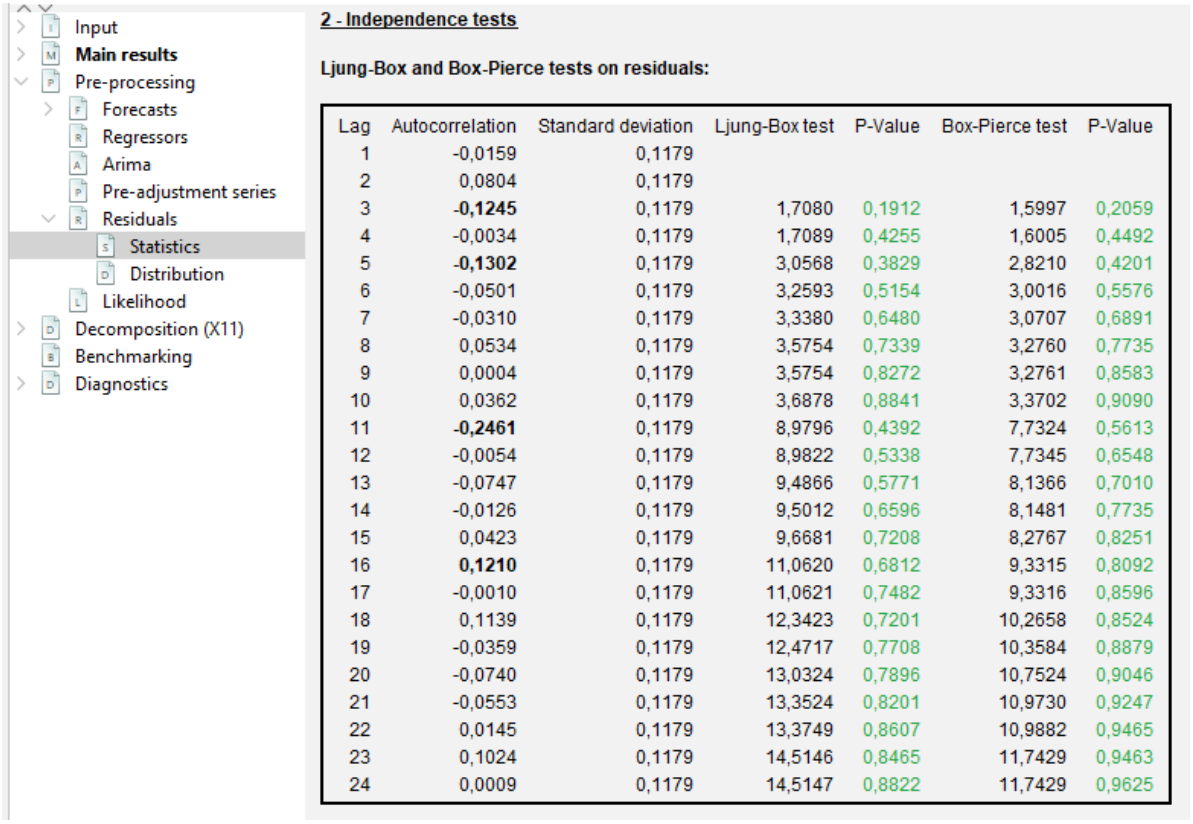


Figure 44: **Residuals**

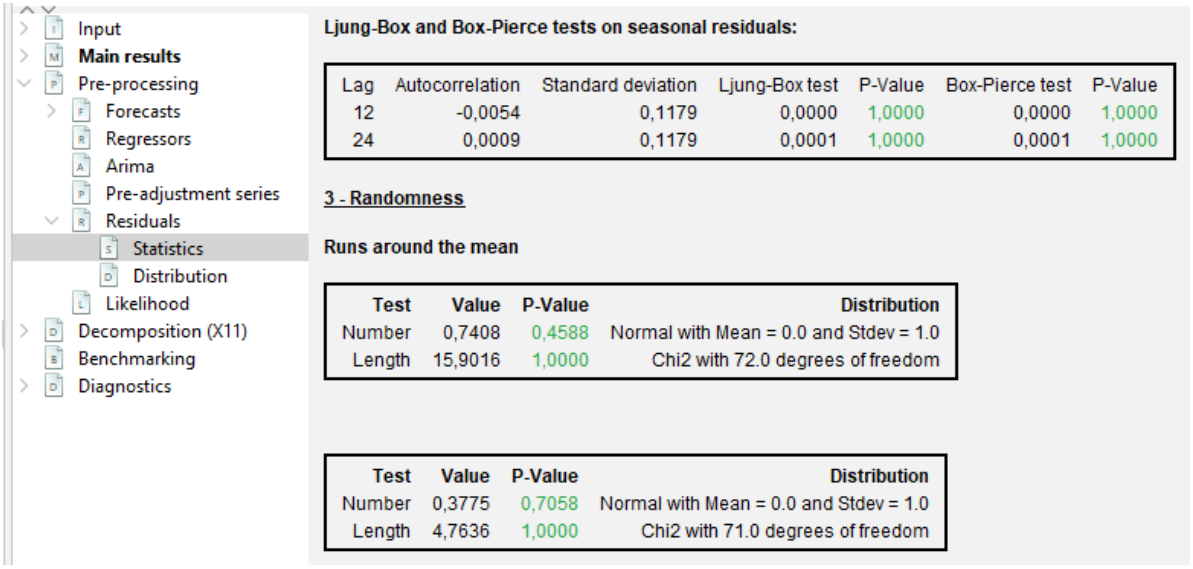


2 - Independence tests

Ljung-Box and Box-Pierce tests on residuals:

Lag	Autocorrelation	Standard deviation	Ljung-Box test	P-Value	Box-Pierce test	P-Value
1	-0,0159	0,1179				
2	0,0804	0,1179				
3	**-0,1245**	0,1179	1,7080	0,1912	1,5997	0,2059
4	-0,0034	0,1179	1,7089	0,4255	1,6005	0,4492
5	**-0,1302**	0,1179	3,0568	0,3829	2,8210	0,4201
6	-0,0501	0,1179	3,2593	0,5154	3,0016	0,5576
7	-0,0310	0,1179	3,3380	0,6480	3,0707	0,6891
8	0,0534	0,1179	3,5754	0,7339	3,2760	0,7735
9	0,0004	0,1179	3,5754	0,8272	3,2761	0,8583
10	0,0362	0,1179	3,6878	0,8841	3,3702	0,9090
11	**-0,2461**	0,1179	8,9796	0,4392	7,7324	0,5613
12	-0,0054	0,1179	8,9822	0,5338	7,7345	0,6548
13	-0,0747	0,1179	9,4866	0,5771	8,1366	0,7010
14	-0,0126	0,1179	9,5012	0,6596	8,1481	0,7735
15	0,0423	0,1179	9,6681	0,7208	8,2767	0,8251
16	**0,1210**	0,1179	11,0620	0,6812	9,3315	0,8092
17	-0,0010	0,1179	11,0621	0,7482	9,3316	0,8596
18	0,1139	0,1179	12,3423	0,7201	10,2658	0,8524
19	-0,0359	0,1179	12,4717	0,7708	10,3584	0,8879
20	-0,0740	0,1179	13,0324	0,7896	10,7524	0,9046
21	-0,0553	0,1179	13,3524	0,8201	10,9730	0,9247
22	0,0145	0,1179	13,3749	0,8607	10,9882	0,9465
23	0,1024	0,1179	14,5146	0,8465	11,7429	0,9463
24	0,0009	0,1179	14,5147	0,8822	11,7429	0,9625

Figure 45: **Residuals**



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

Lag	Autocorrelation	Standard deviation	Ljung-Box test	P-Value	Box-Pierce test	P-Value
12	-0,0054	0,1179	0,0000	1,0000	0,0000	1,0000
24	0,0009	0,1179	0,0001	1,0000	0,0001	1,0000

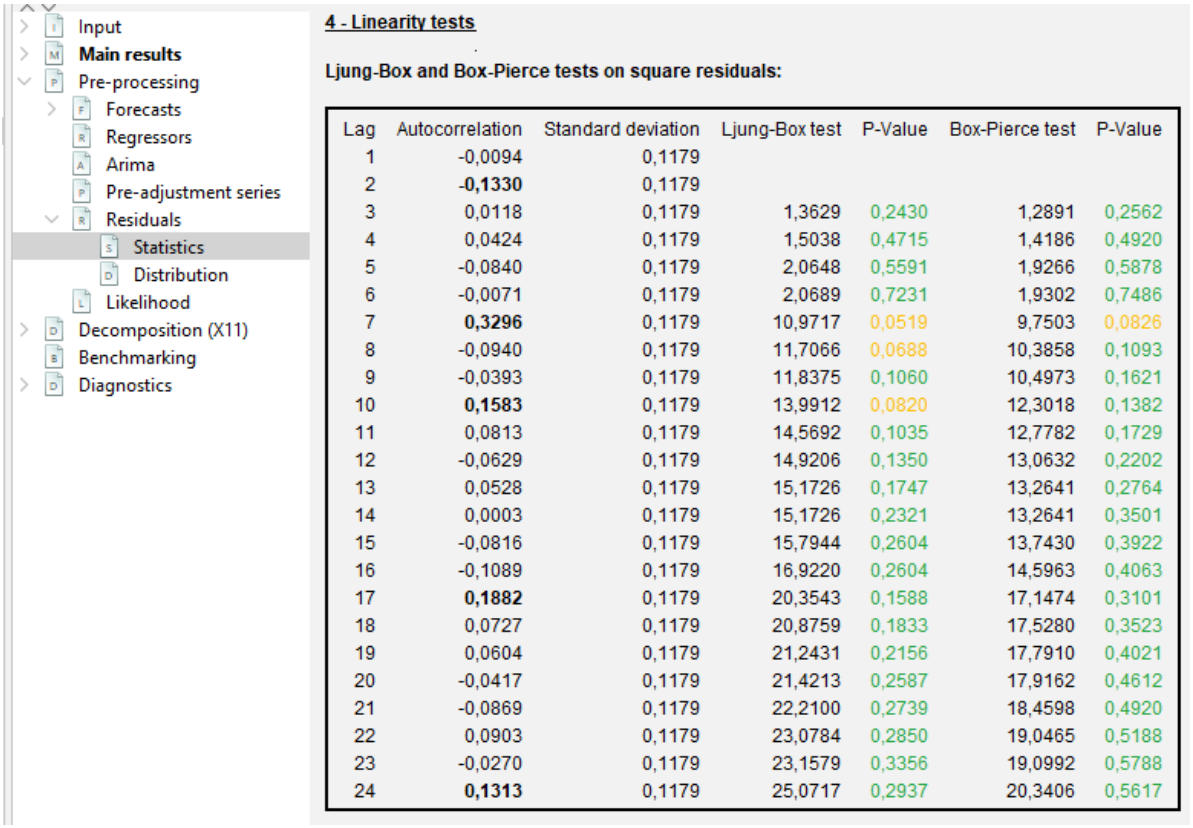
3 - Randomness

Runs around the mean

Test	Value	P-Value	Distribution
Number	0,7408	0,4588	Normal with Mean = 0.0 and Stdev = 1.0
Length	15,9016	1,0000	Chi2 with 72.0 degrees of freedom

Test	Value	P-Value	Distribution
Number	0,3775	0,7058	Normal with Mean = 0.0 and Stdev = 1.0
Length	4,7636	1,0000	Chi2 with 71.0 degrees of freedom

Figure 46: **Residuals**



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	-0,0094	0,1179				
2	**-0,1330**	0,1179				
3	0,0118	0,1179	1,3629	0,2430	1,2891	0,2562
4	0,0424	0,1179	1,5038	0,4715	1,4186	0,4920
5	-0,0840	0,1179	2,0648	0,5591	1,9266	0,5878
6	-0,0071	0,1179	2,0689	0,7231	1,9302	0,7486
7	**0,3296**	0,1179	10,9717	0,0519	9,7503	0,0826
8	-0,0940	0,1179	11,7066	0,0688	10,3858	0,1093
9	-0,0393	0,1179	11,8375	0,1060	10,4973	0,1621
10	**0,1583**	0,1179	13,9912	0,0820	12,3018	0,1382
11	0,0813	0,1179	14,5692	0,1035	12,7782	0,1729
12	-0,0629	0,1179	14,9206	0,1350	13,0632	0,2202
13	0,0528	0,1179	15,1726	0,1747	13,2641	0,2764
14	0,0003	0,1179	15,1726	0,2321	13,2641	0,3501
15	-0,0816	0,1179	15,7944	0,2604	13,7430	0,3922
16	-0,1089	0,1179	16,9220	0,2604	14,5963	0,4063
17	**0,1882**	0,1179	20,3543	0,1588	17,1474	0,3101
18	0,0727	0,1179	20,8759	0,1833	17,5280	0,3523
19	0,0604	0,1179	21,2431	0,2156	17,7910	0,4021
20	-0,0417	0,1179	21,4213	0,2587	17,9162	0,4612
21	-0,0869	0,1179	22,2100	0,2739	18,4598	0,4920
22	0,0903	0,1179	23,0784	0,2850	19,0465	0,5188
23	-0,0270	0,1179	23,1579	0,3356	19,0992	0,5788
24	**0,1313**	0,1179	25,0717	0,2937	20,3406	0,5617

Figure 47: **Residuals**

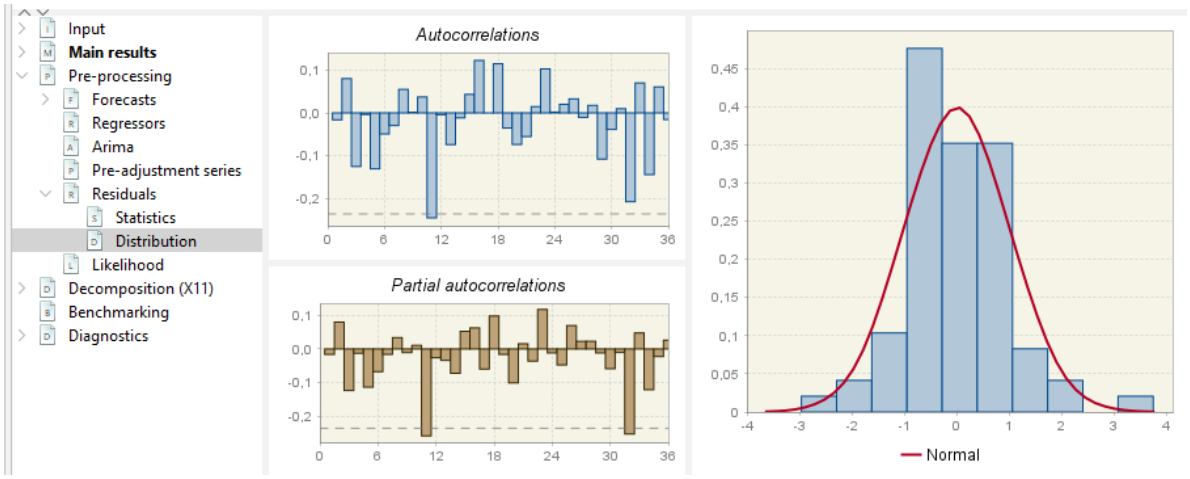


Figure 48: **Distribution**

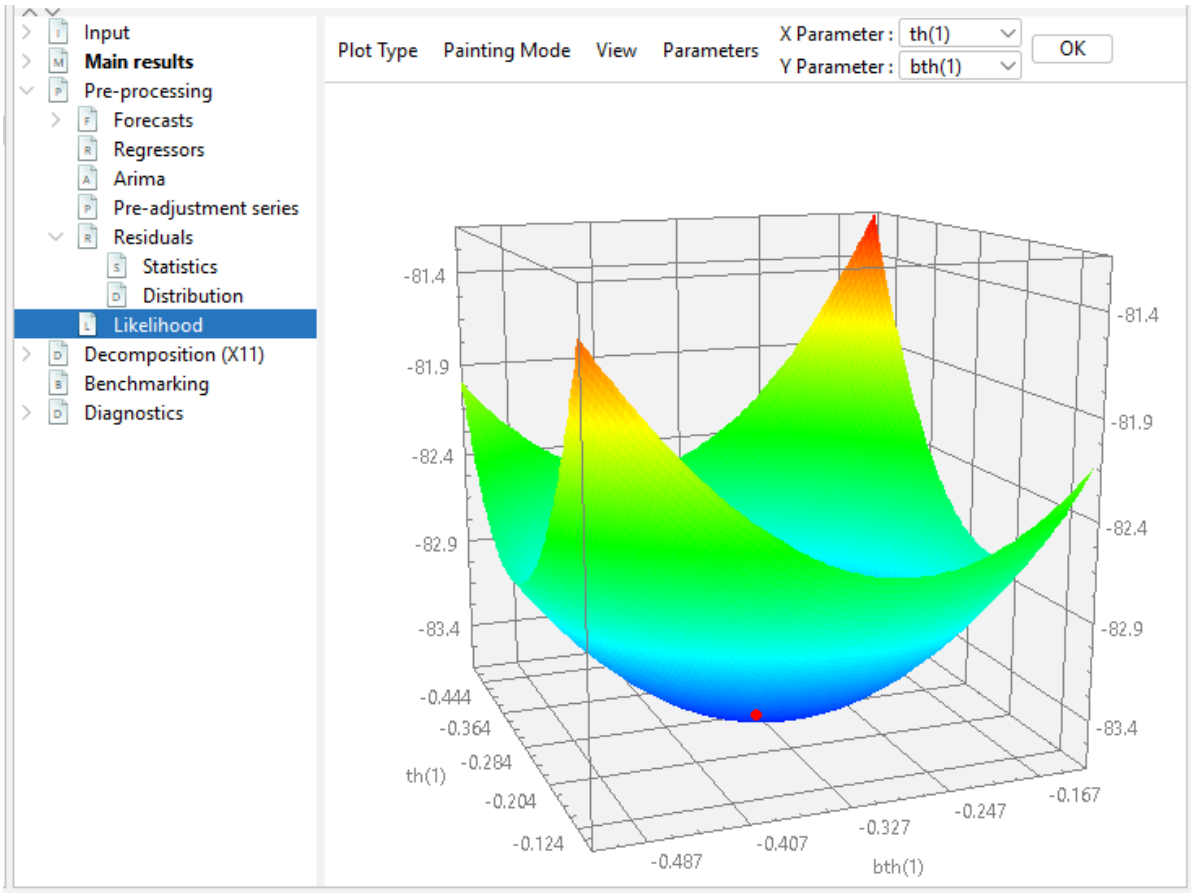
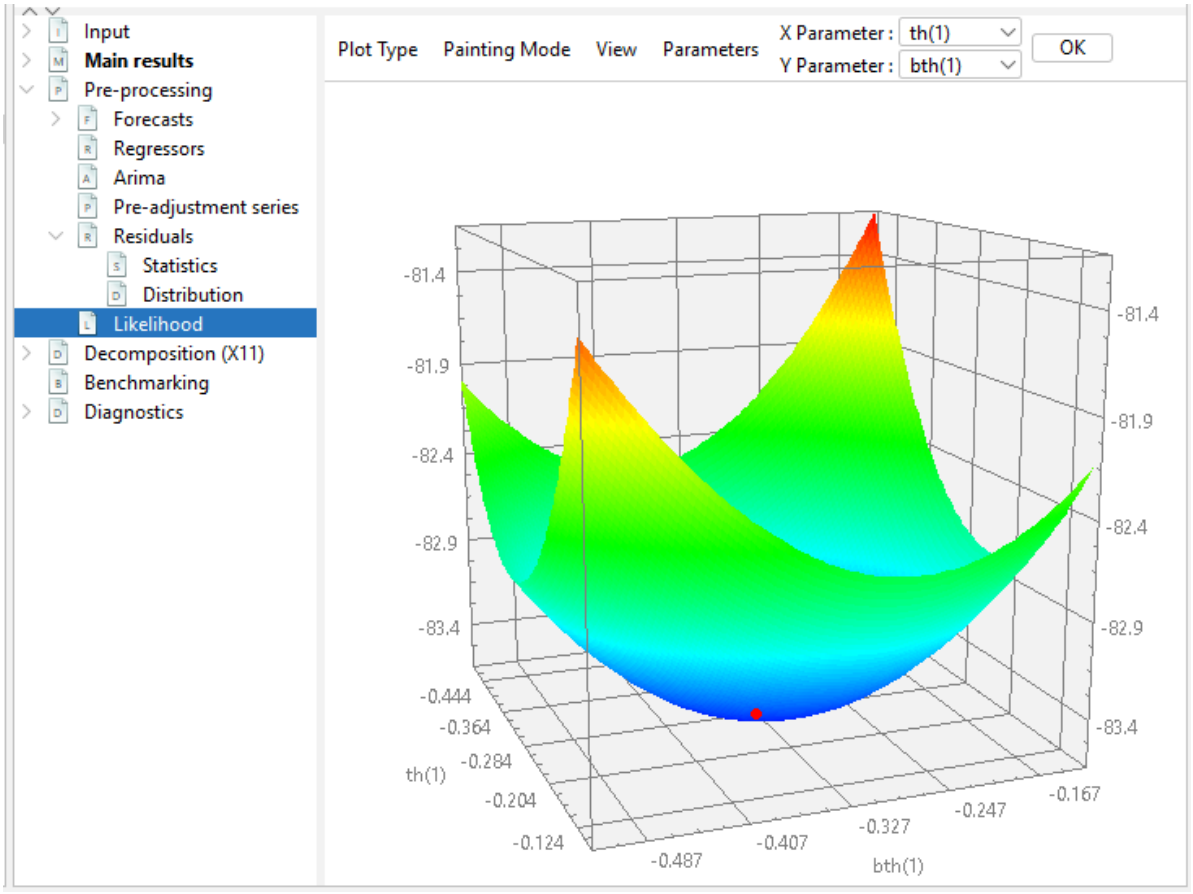


Figure 49: **Likelihood**



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

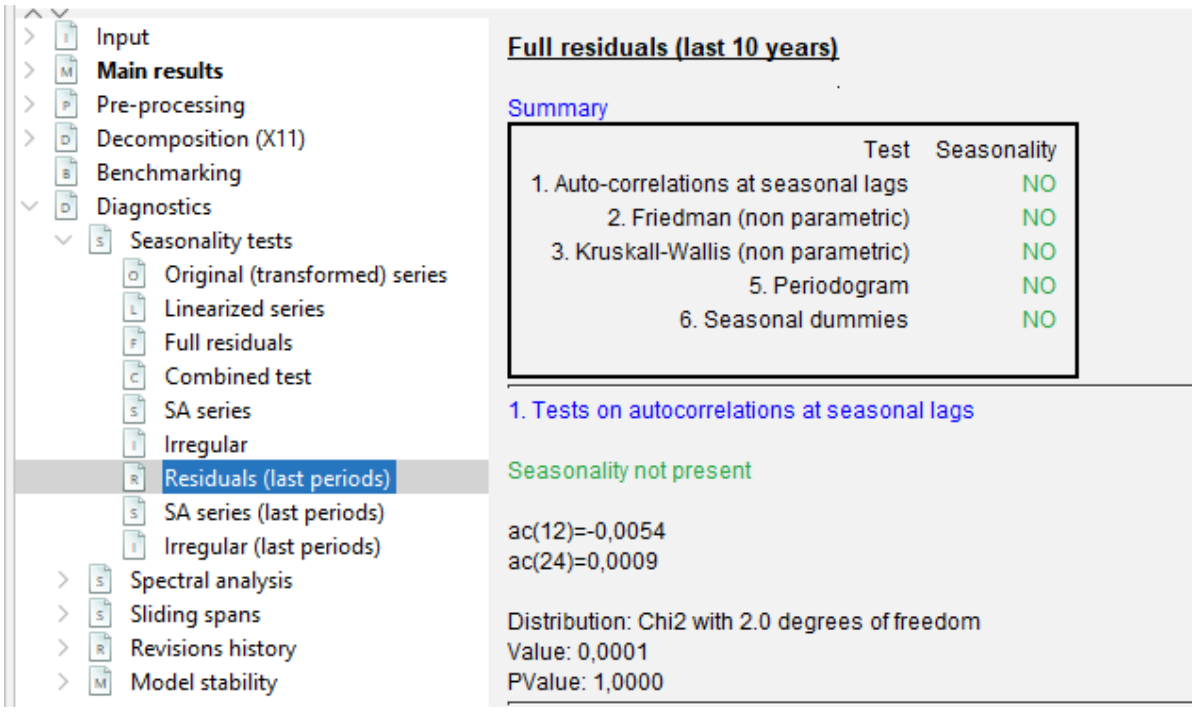


Figure 50: **Residuals**

SA: X11 Decomposition

In this chapter

This chapter focuses on practical implementation of an X11 decomposition using the graphical user interface GUI and R using R packages in version 2.x and 3.x. More explanations on X11 algorithm can be found here.

In recent years X11 has been tailored in JDemetra+ to handle high-frequency (infra-monthly) data, which is described here with more methodological details here.

The sections below will describe

- specifications needed to run X11
- generated output
- series
- diagnostics
- final parameters
- user-defined parameters

Context of use

X11 algorithm is generally the second (decomposition) step in a seasonal adjustment processing with X-13-ARIMA, once a pre-treatment phase has been performed. In this case X11 will decompose the linearized series using iteratively different moving averages. The effects of pre-treatment will be reallocated at the end the the relevant components. X11 can also be used without pre-treatment, choosing and will decompose the raw series.

Tools for X11 decomposition

Algorithm	Access in GUI (v2 and v3)	Access in R (v2)	Access in R (v3)
X-13-ARIMA	✔	RJDemetra	rjd3x13
X12plus	v3 only	——-	rjd3x11plus
X11 decomposition only	✔	RJDemetra	rjd3x13

Available frequencies in version 2 and version 3

Version	GUI and R
v 2.x	$p = 12, 4, 2$
v 3.x	$p = 12, 6, 4, 3, 2$

Default specifications

The default specifications for X11 must be chosen at the start of the SA processing, one of the options available there is to run a X11 decomposition without pre-treatment.

They are detailed in the chapter on pre-treatment.

Quick Launch

From GUI

With a workspace open, an SAProcessing created and an open data provider: (link to GUI general process)

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

In R

In version 2 using RJDemetra

```
library("RJDemetra")
# the input series has to be a Time Series (TS) object
# specification RSA5c including pre-treatment
model_sa_v2 <- x13(raw_series, spec = "RSA5c")
# specification X11 without pre-treatment
model_sa_v2 <- x13(raw_series, spec = "X11")
```

Full documentation of 'RJDemetra::x13' function can be found here

The model_sa_v2 R object (list of lists) contains all parameters and results. Its structure is detailed here. It can be printed giving access to selected parameters, series and diagnostics.

```
print(model_sa_v2)
```

In version 3 using rjd3x13

```
library("rjd3toolkit")
library("rjd3x13")
# the input series has to be a Time Series (TS) object
model_sa_v3 <- rjd3x13::x13(y_raw, spec = "RSA5")
```

Full documentation of 'rjd3x13::x13' function can be found here and of 'rjd3x13::X11' here.

The model_sa_v3 R object (list of lists) contains all parameters and results. Its structure is detailed here.

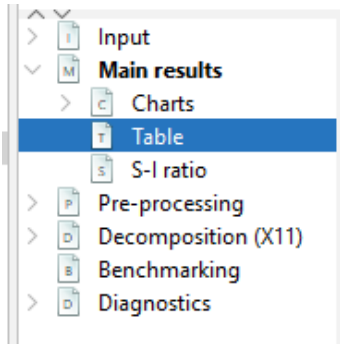
It can be printed giving access to selected parameters, series and diagnostics.

```
print(model_sa_v3)
```

Retrieve series

Display in GUI

Final components from the SA Processing are displayed in Main results. They contain the re-allocated pre-adjustment effects of outliers (link) or external regressors (link). The final seasonal components contains the calendar effects, if any.

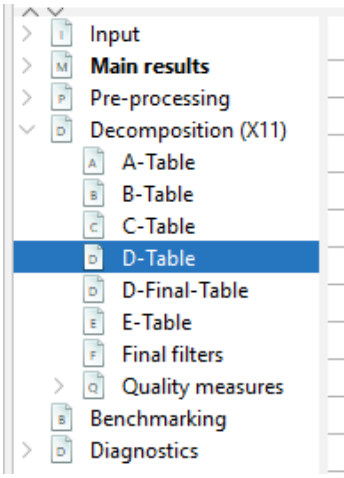


	Series	Seasonally...	Trend	Seasonal	Irregular
1-1990	248,134	236,55	232,731	1,049	1,016
2-1990	232,617	235,623	231,054	0,987	1,02
3-1990	264,746	227,346	230,228	1,165	0,987
4-1990	225,175	228,967	230,208	0,983	0,995
5-1990	220,624	223,236	231,652	0,988	0,964
6-1990	245,684	240,858	235,265	1,02	1,024
7-1990	247,358	235,835	241,197	1,049	0,978

Figure 51: **Final components in GUI**

(forecasts are added at the end of the series, values in *italic*)

Detailed results from decomposition are displayed in Decomposition (X11) node.



	d1	d4	d5	d6	d7	d
1-1990	258,072		1,043	247,373	242,627	
2-1990	247,327		0,999	247,473	240,84	
3-1990	263,728		1,117	236,085	239,929	
4-1990	246,274		1,034	238,213	240,099	
5-1990	221,835		0,964	230,066	242,045	
6-1990	267,813		1,055	253,97	246,252	
7-1990	253,286	0,995	1,009	251,07	252,57	
8-1990	201,181	0,785	0,771	261,078	259,647	
9-1990	303,845	1,175	1,144	265,592	265,608	
10-1990	292,296	1,122	1,074	272,248	269,834	
11-1990	265,577	1,013	0,989	268,619	272,058	

Figure 52: **Detailed results**

The final D Tables contain the re-allocated pre-adjustment effects.

(to be checked: v2 vs v3 on pre-treatment effects in D tables)

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

- generating output files directly with interactive menus
- running the cruncher to generate those files as described here

Retrieve in R

In version 2

```
model_sa <- x13(raw_series, spec = "RSA3") # user's spec choice
# final components
model_sa$final$series
# final forecasts y_f sa_f s_f t_f i_f
model_sa$final$forecasts
```

Detailed X11 tables have to be pre-specified from the user-defined output list.

```
# display the list of available objects (series, diagnostics, parameters)
user_defined_variables("X-13-ARIMA")
# add selected object to estimation
sa_x13_v2 <- RJDemetra::x13(myseries, myspec,
    userdefined = c("decomposition.c20", "decomposition.d1")
)
# retrieve in the user-defined sub-list
sa_x13_v2$user_defined
```

to be modified In version 3

```
# 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
```

Diagnostics

X11 produces the following type diagnostics or quality measures

SI-ratios

Display in GUI

NODE Main Results > SI-Ratios

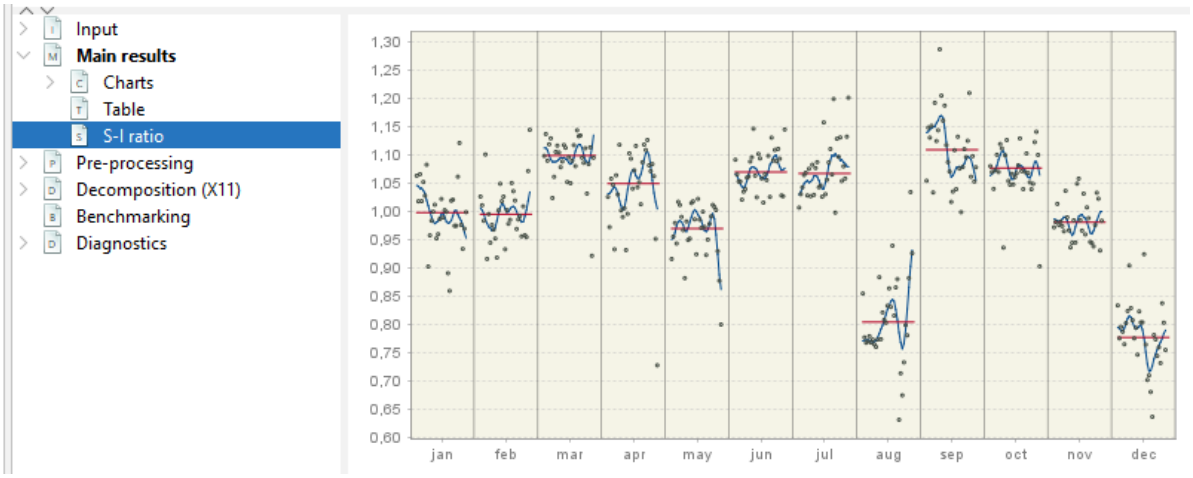


Figure 53: **S-I Ratio**

For each period (month, quarter) the final value of the seasonal factors (without calendar factors, Table D10) is plotted (blue line). The dots represent $S + I$ or $S * I$ in the multiplicative case, where I= Table D8. The red straight line is the average of the factors over the decomposition (estimation) span.

In GUI all values cannot be retrieved.

Retrieve in R

All the values and the same plot as described above can be generated in R, the span can be customized.

(to be checked the average of the seasonal factors is not recalculated if span modified)

In version 2

```
# data frame with values
model_sa_v2$decomposition$si_ratio

# customizable plot
```

```
plot(mysa2$decomposition)

plot(model_sa, type = "cal-seas-irr", first_date = c(2015, 1))
```

In version 3

```
# data frame with values
model_sa_v2$decomposition$si_ratio

# customizable plot
plot(mysa2$decomposition)

plot(model_sa, type = "cal-seas-irr", first_date = c(2015, 1))
```

M-statistics

X11 algorithm provides quality measures of the decomposition called “M statistics” (detailed here

- 11 statistics (M1 to M11)
- 2 summary indicators (Q et Q-M2)
- by design $0 < M_x < 3$ and acceptance region is $M_x \leq 1$

0.0.0.0.1 * Display in GUI

To display results in GUI, expand NODE

Decomposition(X11) > Quality Measures > Summary

Results displayed in red indicate that the test failed.

0.0.0.0.2 * Retrieve in R

In version 2

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

In version 3

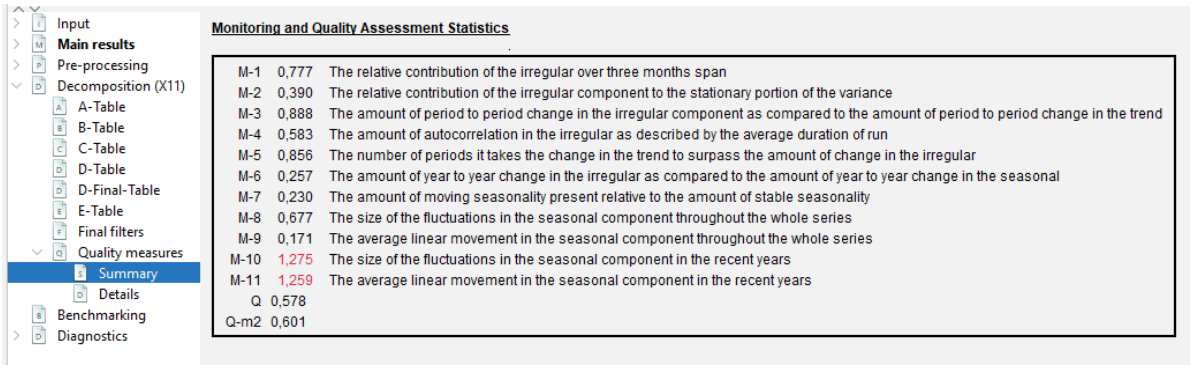


Figure 54: **Summary of the quality measures**

```
# 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(X11) > Quality Measures > Details

They are detailed in the X11 method chapter

In R (to be added): not directly available ?!

Retrieve final filters

The following parameters are automatically chosen by the software as a result of the estimation process. They have no default value but can be set by the user.

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

Display in GUI

Node Decomposition(X11) > Final Filters

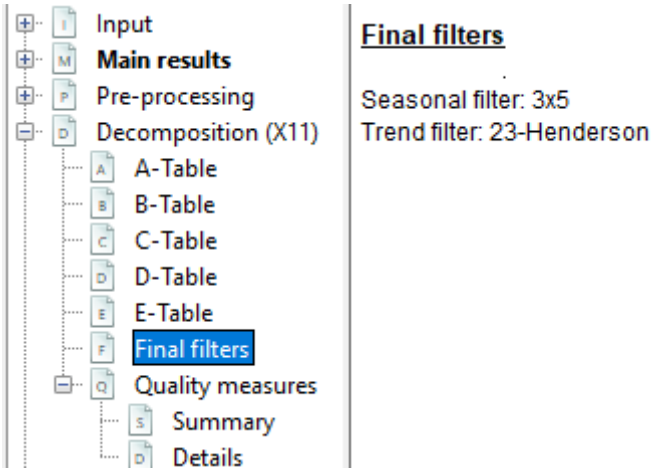


Figure 55: **Table of final filters**

Retrieve in R

In version 2

```
library("RJDemetra")
model_sa_v2 <- x13(raw_seriesa, spec = "RSA5c")
model_sa$decomposition$s_filter
model_sa$decomposition$t_filter
```

In version 3

```
library("rjd3toolkit")
library("rjd3x13")
model_sa_v3 <- rjd3x13::x13(y_raw, spec = "RSA5")
model_sa_v3$result$decomposition$final_seasonal
model_sa_v3$result$decomposition$final_henderson
```

User-defined parameters

The following parameters have default values, which will not be changed in the estimation process. They can be set by the user in a given range of admissible values.

General settings

- **Mode**

```
-   Undefined: automatically chosen between Multiplicative and Additive Options available onl
-   Additive: $Y=T+S+I$, $SA =Y-S=T+I$
-   Multiplicative $Y=T*S*I$, $SA =Y/S=T*I$
-   LogAdditive $Log(Y) = T + S + I$, $SA=exp(T+I)=Y/exp(S)$
-   PseudoAdditive $Y=T*(S+I-1)$, $SA=T*I$
```

If X11 decomposition comes after a pre-processing, **mode** is set to undefined and will correspond to decomposition choice made in the pre-treatment: multiplicative if series log- transformed, additive otherwise.

- **Seasonal component**

Option available only if no pre-processing: - yes (default), decomposition into S, T, I - no, decomposition into S, T, I

- **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 X11 procedure does not use any model-based forecasts but the original X11 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

0.0.0.0.1 * 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 - 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

0.0.0.0.2 * Seasonality extraction filters choice

- **Seasonal filter**

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

- 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

- default value: empty, same filter for all periods

0.0.0.0.3 * Trend estimation filters

- **Automatic Henderson filter** or 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

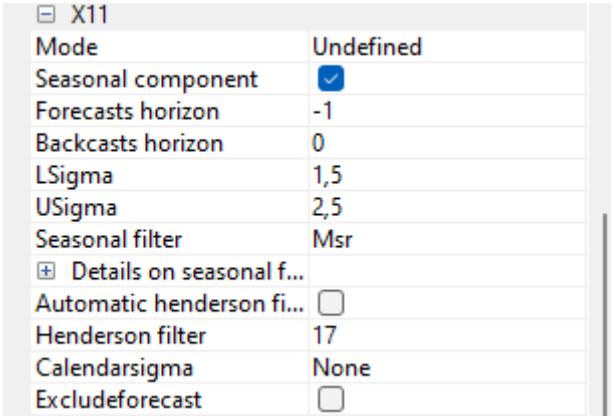


Figure 56: Text

Setting details on seasonal filters

Previously set values are displayed for each type of period, here they are all to default MSR choice.

Click on the right top button (show on image)

Another window appears in the top-left corner allowing to chose the filter period by period.

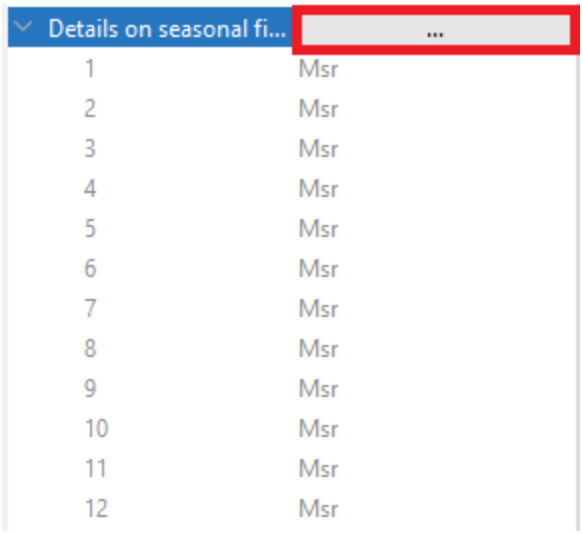


Figure 57: **Seasonnal filters**

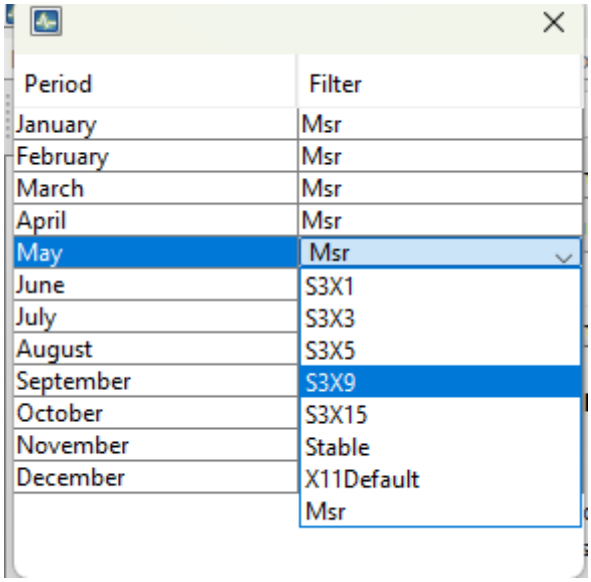


Figure 58: Text

Parameter setting in R packages

In version 2 using RJDemetra

```
current_sa_model <- x13(raw_series, spec = current_spec)
# Creating a modified specification, customizing all available X11 parameters
modified_spec <- x13_spec(current_sa_model,
    X11.mode = NA,
    X11.seasonalComp = NA,
    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)
```

In version 3 using rjd3x13

```
# Creating a modified specification, customizing all available X11 parameters
library("RJDemetra")
model_sa_v2 <- x13(raw_series, spec = "RSA5c")
# Creating a modified specification from the current estimation model
# Customizing all available X11 parameters
modified_spec <- x13_spec(model_sa_v2,
    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)


# For options available only in X11 mode
modified_spec <- x13_spec(model_sa_v2,
    # X11.mode="?",
    # X11.seasonalComp = "?",
    X11.fcasts = -2
)
```

Retrieving Parameters

How to see what parameters have actually been used.

In GUI: just open the specification box and navigate the options.

In R, print your model or navigate its elements.

SA: Seats Decomposition

In this chapter

This chapter focuses on practical implementation of a Seats decomposition using the graphical user interface GUI and R using R packages in version 2.x and 3.x. More explanations on Seats algorithm can be found here.

In recent years Seats has been tailored in JDemetra+ to handle high-frequency (infra-monthly) data, which is described here with more methodological details here.

The sections below will describe

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

Context

Seats is the second (decomposition) step in a seasonal adjustment processing with Tramo-Seats, once a pre-treatment with Tramo has been performed. Seats is an ARIMA Model Based (AMB) algorithm and will decompose the linearized series using the ARIMA model fit in Tramo.

Tools for Seats decomposition

Algorithm	Access in GUI	Access in R (v2)	Access in R v3
Tramo-Seats	✔	RJDemetra	rjd3tramoseats
Tramo only	✔	RJDemetra	rjd3tramoseats

Available frequencies in version 2 and version 3

Version	GUI and R
v 2.x	$p = 12, 6, 4, 2$
v 3.x	$p = 12, 6, 4, 3, 2$

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. {#a-sa-seats-q-start}

Quick Launch

Default specifications

The default specifications for Seats must be chosen at the starting of the SA processing. Starting point for Tramo-Seats, detailed here

Using 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 using RJDemetra

```
library("RJDemetra")
# the input series has to be a Time Series (TS) object
# specification RSAfull including pre-treatment
model_sa_v2 <- tramoseats(raw_series, spec = "RSAfull")
```

Full documentation of 'RJDemetra::tramoseats' function can be found here

The model_sa_v2 R object (list of lists) contains all parameters and results. Its structure is detailed here. It can be printed giving access to selected parameters, series and diagnostics.

```
print(model_sa_v2)
```

In version 3 using rjd3tramoseats

```
library("rjd3tramoseats")
model_sa_v3 <- tramoseats(raw_series, spec = "RSAfull")
# the input series has to be a Time Series (TS) object
```

Full documentation of 'rjd3tramoseats::tramoseats' function can be found here.

The model_sa_v3 R object (list of lists) contains all parameters and results. Its structure is detailed here.

It can be printed giving access to selected parameters, series and diagnostics.

```
print(model_sa_v3)
```

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

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

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	

Series Name	Meaning
decomposition.s_lin_ef	
decomposition.i_lin	
decomposition.i_lin_f	
decomposition.i_lin_e	
decomposition.i_lin_ef	

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(
    series = y,
    spec = "RSAfull",
    userdefined = c(
        "decomposition.y_lin", "ycal",
        "variancedecomposition.seasonality"
    )
)
# retrieve objects
m$user_defined$decomposition.y_lin
m$user_defined$ycal
m$user_defined$variancedecomposition.seasonality
```

In version 3

```
library("rjd3tramoseats")
# list of additional output objects
userdefined_variables_tramoseats("tramoseats")
# specify additional objects in estimation
m <- tramoseats(
    ts = y,
    spec = "RSAfull",
    userdefined = c(
        "decomposition.y_lin", "ycal",
```

```
        "variancedecomposition.seasonality"
    )
)
# 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

Displayed in GUI

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

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	

Series Name	Meaning
decomposition.sa_cmp_ef	
decomposition.s_cmp	
decomposition.s_cmp_f	
decomposition.s_cmp_e	
decomposition.s_cmp_ef	
decomposition.i_cmp	
decomposition.i_cmp_f	
decomposition.i_cmp_e	
decomposition.i_cmp_ef	

Retrieve in R

Same procedure as for stochastic series.

Bias correction

to be added

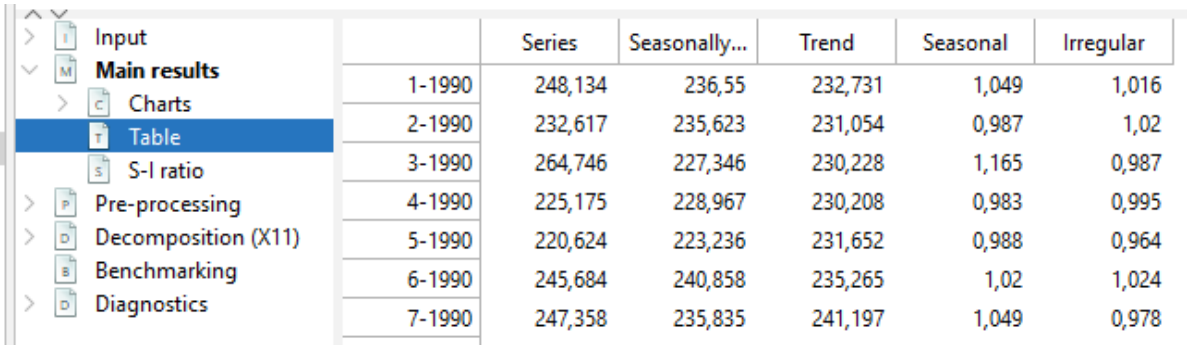
Final series

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

(to be added: reallocation of outliers effects)

Display in GUI

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



	Series	Seasonally...	Trend	Seasonal	Irregular
1-1990	248,134	236,55	232,731	1,049	1,016
2-1990	232,617	235,623	231,054	0,987	1,02
3-1990	264,746	227,346	230,228	1,165	0,987
4-1990	225,175	228,967	230,208	0,983	0,995
5-1990	220,624	223,236	231,652	0,988	0,964
6-1990	245,684	240,858	235,265	1,02	1,024
7-1990	247,358	235,835	241,197	1,049	0,978

Figure 59: **Table of final results**

Forecasts are glued at the end it *italic*

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	

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

Display in GUI

Click on the **Decomposition** NODE

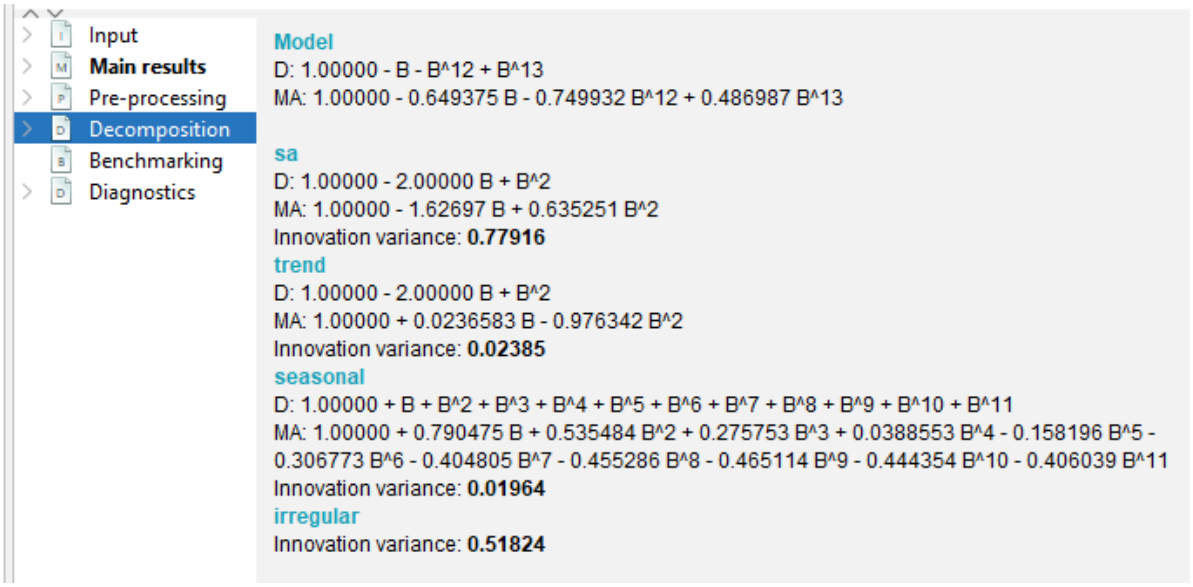


Figure 60: **Decomposition Node**

Retrieve from GUI

(add names for output and cruncher)

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

Seting parameters in GUI

In specification window corresponding to a given series:

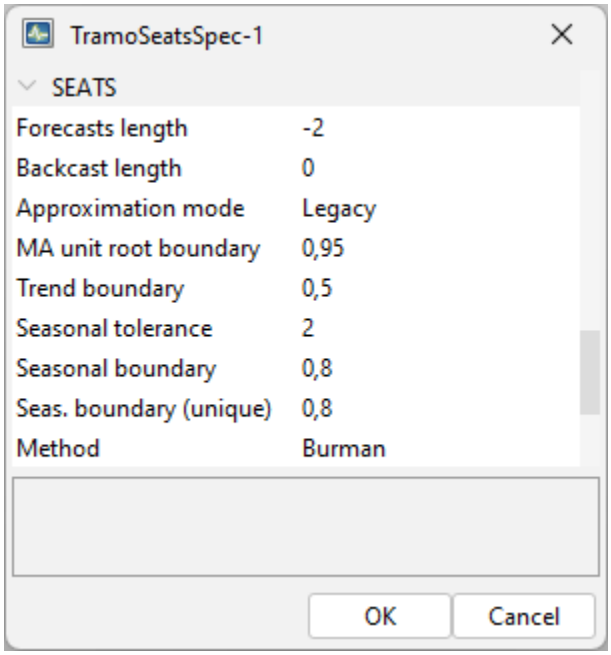


Figure 61: **TramoSeatsSpec Seats part**

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_character_, "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_character_, "Burman", "KalmanSmoother", "McElroyMatrix")
)
```

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

SA: Revision policies

In this chapter

The sections below describe:

- how to update seasonally adjusted series when new data is available
- what is a revision policy in a seasonal adjustment context
- the description of all the revision policies available in JDemetra+
- how to implement a revision policy using the Graphical User Interface, R or the Cruncher.

Revision Policies

Definition and context

When raw data has been modified (extended and/or revised), the previous seasonal adjustment estimation needs updating. It can be redone from scratch (complete re-estimation) or update keeping fixed a predefined set of parameters already estimated. Eurostat's Guidelines on seasonal adjustment (2015) recommend not to perform a complete re-estimation of the parameters on an infra-annual basis. The set of constraints on the parameters is called "revision policy" or "refresh policy".

Overview

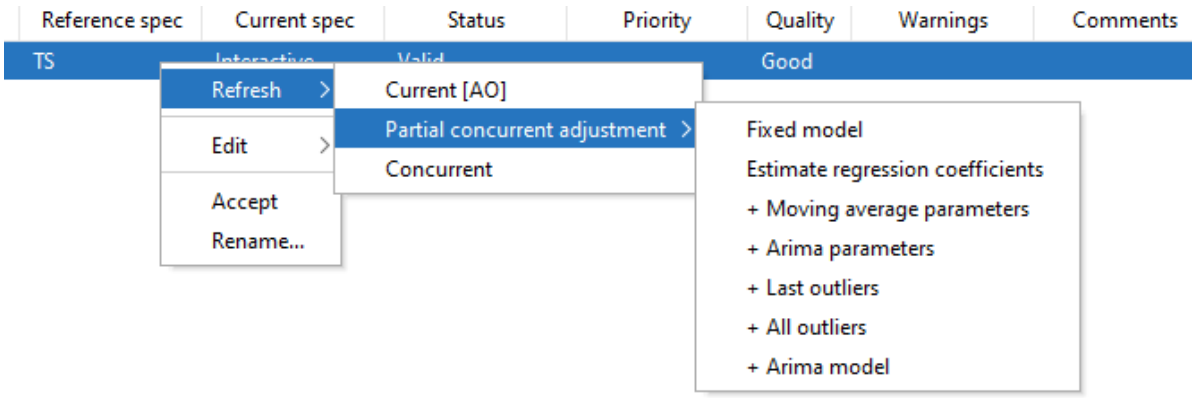
In X-13-ARIMA and Tramo-Seats revision policies are ways to impose constraints on the pre-adjustment phase, while the decomposition phase (X-11 or Seats) will be entirely re-run on new data. The changes induced by X-11 re-estimation stem only from a revised linerarized series, while in Seats they are also induced by the ARIMA model possible coefficient and/ or order changes.

The table below lists the available policies as well as their name for implementation with the graphical user interface (GUI), the cruncher or rjd3x13 ans rjd3tramoseats packages.

Revision Policy	JDemetra+ Interface (GUI)	Cruncher (via R)	Rjd3x13 / rjd3tramoseats
Applying the current model (unchanged) adding the new raw points as AO	Current adjustment (AO approach)	current (n)	current
Applying the current model (unchanged) replacing forecasts by new raw points	Fixed model	fixed(f)	fixed
Regression variables, Arima orders and coefficients are unchanged, only regression coefficents are re-estimated	Estimate regression coefficients	fixedparameters (fp)	FixedParameters
...previous + Arima model MA coefficents also re-estimated	+ Moving average parameters	FixedAutoRegressiveParameters	FixedAutoRegressiveParameters
...previous + Arima model coefficents also re-estimated	+ Arima parameters	parameters (p)	FreeParameters
...previous + outliers re-identified for the last year	+ Last outliers	lastoutliers (l)	Outliers (+span)
...previous + outliers re-identified for the whole series	+ All outliers	outliers (o)	Outliers
...previous + orders of the Arima model are re-identified	+ Arima model	stochastic (s)	Outliers_StochasticComponent
All the parameters are re-identified and re-estimated (note : any user defined variable or constraint is kept)	Concurrent	complete / concurrent (c)	complete

Implementation in GUI

To refresh results from previous estimation open your workspace, then SAprocessing click on a series to highlight it (or select several series), then right-click and choose *Refresh*, the following panel is displayed.



Display in results panel

Sections below detail, though an example, the changes in results display brought about by the use of a given revision policy.

Concurrent

Concurrent adjustment means that the model, filters, outliers, regression parameters and transformation type are all re-identified and the respective parameters and factors re-estimated every time new observations are available. This option in JDemetra+ means that a completely new model is identified, and the previous results are not taken into account, excepted for the user-defined parameters.

The picture below presents the initial model (on the left) and the results of the refreshment procedure with the *Concurrent adjustment* option (on the right). The transformation type has changed from none to log. The ARIMA model has been re-identified (it has changed from (0,1,1)(1,1,0) to (1,1,0)(0,1,1)). In contrast to the initial model, in the updated model trading day effects and a leap year effect are no longer included. Also the automatically identified outliers are not the same in both models.

Summary

Estimation span: [7-1996 - 12-2016]
246 observations
Trading days effects (7 variables)
Easter [8] detected
5 detected outliers

Final model

Likelihood statistics
Number of effective observations = 233
Number of estimated parameters = 16

Loglikelihood = -559.6616717869163
Standard error of the regression (ML estimate) = 2.669031738674505
AIC = 1151.3233435738325
AICC = 1153.841862092351
BIC (corrected for length) = 2.314356746231504

Scores at the solution
-0,000004 -0,000414 .

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4902	-8,33	0,0000
BPhi(1)	0,1680	2,45	0,0152

Correlation of the estimates

	Theta(1)	BPhi(1)
Theta(1)	1,0000	0,0784
BPhi(1)	0,0784	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5102	-1,92	0,0562
Tuesday	0,2288	0,88	0,3774
Wednesday	0,1073	0,40	0,6905
Thursday	0,2028	0,75	0,4536
Friday	0,9280	3,48	0,0006
Saturday	-0,3434	-1,27	0,2071
Sunday (derived)	-0,6134	-2,28	0,0235

Joint F-Test = 8,13 (0,0000)

Leap year

Coefficients	T-Stat	P[|T| > t]
2,6712	3,07	0,0024

Easter [8]

Coefficients	T-Stat	P[|T| > t]
1,6759	3,08	0,0023

Outliers

	Coefficients	T-Stat	P[|T| > t]
AO (4-2004)	19,8713	10,12	0,0000
LS (1-2001)	-8,5643	-4,62	0,0000
AO (4-2010)	-8,7157	-4,62	0,0000
AO (3-2004)	8,6114	4,40	0,0000
AO (12-2003)	7,2918	3,90	0,0001

Summary

Estimation span: [1-2005 - 12-2016]
144 observations
Series has been log-transformed
No trading days effects
Easter [1] detected
1 detected outlier

Final model

Likelihood statistics
Number of effective observations = 131
Number of estimated parameters = 5

Loglikelihood = 158.22155489483504
Transformation adjustment = -648.184301988354
Adjusted loglikelihood = -489.962747093519

Standard error of the regression (ML estimate) = 0.06861665286654098
AIC = 989.925494187038
AICC = 990.405494187038
BIC (corrected for length) = -5.2095790541390326

Scores at the solution
0,002923 -0,004642 .

Arima model
[(1,1,0)(0,1,1)].

	Coefficients	T-Stat	P[|T| > t]
Phi(1)	0,4240	5,25	0,0000
BTheta(1)	-0,8247	-13,50	0,0000

Correlation of the estimates

	Phi(1)	BTheta(1)
Phi(1)	1,0000	0,0857
BTheta(1)	0,0857	1,0000

Regression model
Easter [1]

Coefficients	T-Stat	P[|T| > t]
-0,0398	-1,94	0,0543

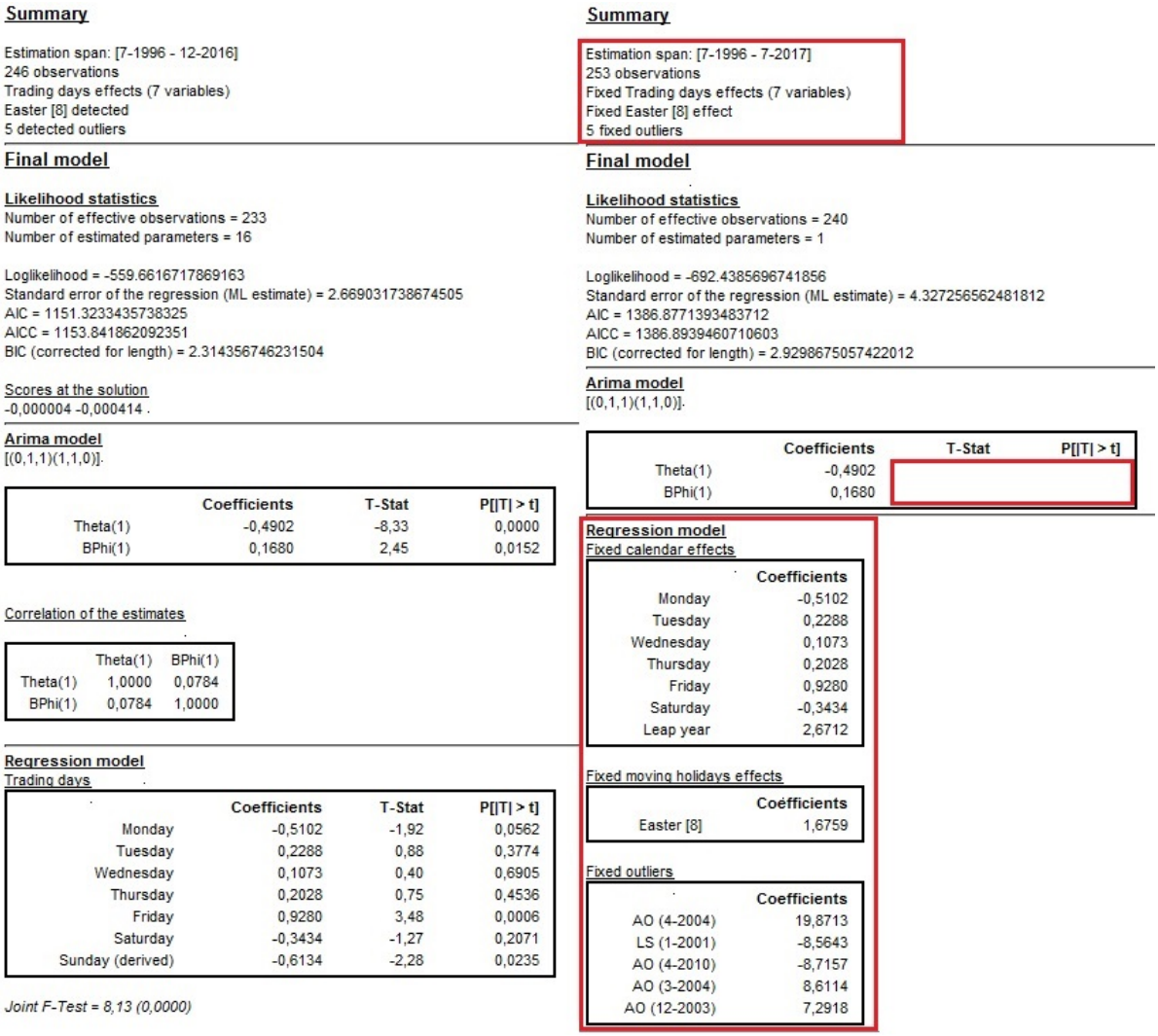
Outliers

	Coefficients	T-Stat	P[|T| > t]
TC (1-2011)	-0,4462	-7,36	0,0000

Partial concurrent adjustment → Fixed model

The *Partial concurrent adjustment → Fixed model* strategy means that the ARIMA model, outliers and other regression parameters are not re-identified and the values of the parameters are fixed. In particular, no new outliers or calendar variables are added to the model as well as no changes neither in the calendar variables nor in the outliers' types are allowed. The transformation type remains unchanged.

The picture below presents the initial model (on the left) and the results of the refreshment procedure with the *Partial concurrent adjustment → Fixed model* option (on the right). The parameters of the ARIMA part are not estimated and their values are the same as before. The trading days and outliers are fixed too and no new regression effects are identified.



Summary

Estimation span: [7-1996 - 12-2016]
246 observations
Trading days effects (7 variables)
Easter [8] detected
5 detected outliers

Final model

Likelihood statistics
Number of effective observations = 233
Number of estimated parameters = 16

Loglikelihood = -559.6616717869163
Standard error of the regression (ML estimate) = 2.669031738674505
AIC = 1151.3233435738325
AICC = 1153.841862092351
BIC (corrected for length) = 2.314356746231504

Scores at the solution
-0,000004 -0,000414 .

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4902	-8,33	0,0000
BPhi(1)	0,1680	2,45	0,0152

Correlation of the estimates

	Theta(1)	BPhi(1)
Theta(1)	1,0000	0,0784
BPhi(1)	0,0784	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5102	-1,92	0,0562
Tuesday	0,2288	0,88	0,3774
Wednesday	0,1073	0,40	0,6905
Thursday	0,2028	0,75	0,4536
Friday	0,9280	3,48	0,0006
Saturday	-0,3434	-1,27	0,2071
Sunday (derived)	-0,6134	-2,28	0,0235

Joint F-Test = 8,13 (0,0000)

Summary

Estimation span: [7-1996 - 7-2017]
253 observations
Fixed Trading days effects (7 variables)
Fixed Easter [8] effect
5 fixed outliers

Final model

Likelihood statistics
Number of effective observations = 240
Number of estimated parameters = 1

Loglikelihood = -692.4385696741856
Standard error of the regression (ML estimate) = 4.327256562481812
AIC = 1386.8771393483712
AICC = 1386.8939460710603
BIC (corrected for length) = 2.9298675057422012

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4902		
BPhi(1)	0,1680		

Regression model
Fixed calendar effects

	Coefficients
Monday	-0,5102
Tuesday	0,2288
Wednesday	0,1073
Thursday	0,2028
Friday	0,9280
Saturday	-0,3434
Leap year	2,6712

Fixed moving holidays effects

	Coefficients
Easter [8]	1,6759

Fixed outliers

	Coefficients
AO (4-2004)	19,8713
LS (1-2001)	-8,5643
AO (4-2010)	-8,7157
AO (3-2004)	8,6114
AO (12-2003)	7,2918

Partial concurrent adjustment → Estimate regression coefficients

The *Partial current adjustment → Estimate regression coefficients* option means that the ARIMA model, outliers and other regression parameters are not re-identified. The coefficients of the ARIMA model are fixed, other coefficients are re-estimated. In particular, no new outliers or calendar variables are added to the model as well as no changes neither in the calendar variables nor in the outliers' types are allowed. The transformation type remains unchanged.

The picture below presents the initial model (on the left) and the results of the refreshment procedure with the *Partial concurrent adjustment → Estimate regression coefficients* option (on the right). The number of estimated parameters is 16 in the initial model and 14 in the revised model (the parameters of the ARIMA model are not estimated.

Summary

Estimation span: [7-1996 - 12-2016]
246 observations
Trading days effects (7 variables)
Easter [8] detected
5 detected outliers

Final model

Likelihood statistics
Number of effective observations = 233
Number of estimated parameters = 16

Loglikelihood = -559.6616717869163
Standard error of the regression (ML estimate) = 2.669031738674505
AIC = 1151.3233435738325
AICC = 1153.841862092351
BIC (corrected for length) = 2.314356746231504

Scores at the solution
-0,000004 -0,000414

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4902	-8,33	0,0000
BPhi(1)	0,1680	2,45	0,0152

Correlation of the estimates

	Theta(1)	BPhi(1)
Theta(1)	1,0000	0,0784
BPhi(1)	0,0784	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5102	-1,92	0,0562
Tuesday	0,2288	0,88	0,3774
Wednesday	0,1073	0,40	0,6905
Thursday	0,2028	0,75	0,4536
Friday	0,9280	3,48	0,0006
Saturday	-0,3434	-1,27	0,2071
Sunday (derived)	-0,6134	-2,28	0,0235

Joint F-Test = 8,13 (0,0000)

Leap year

	Coefficients	T-Stat	P[|T| > t]
	2,6712	3,07	0,0024

Easter [8]

	Coefficients	T-Stat	P[|T| > t]
	1,6759	3,08	0,0023

Outliers

	Coefficients	T-Stat	P[|T| > t]
AO (4-2004)	19,8713	10,12	0,0000
LS (1-2001)	-8,5643	-4,62	0,0000
AO (4-2010)	-8,7157	-4,62	0,0000
AO (3-2004)	8,6114	4,40	0,0000
AO (12-2003)	7,2918	3,90	0,0001

Summary

Estimation span: [7-1996 - 7-2017]
253 observations
Trading days effects (7 variables)
Easter [8] detected
5 pre-specified outliers

Final model

Likelihood statistics
Number of effective observations = 240
Number of estimated parameters = 14

Loglikelihood = -665.9671390063976
Standard error of the regression (ML estimate) = 3.875350557203808
AIC = 1359.9342780127952
AICC = 1361.8009446794617
BIC (corrected for length) = 3.0061401918583264

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4902		
BPhi(1)	0,1680		

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5065	-1,34	0,1819
Tuesday	0,3467	0,94	0,3463
Wednesday	0,1225	0,32	0,7479
Thursday	0,2916	0,75	0,4525
Friday	0,7808	2,07	0,0393
Saturday	-0,6722	-1,74	0,0840
Sunday (derived)	-0,3629	-0,95	0,3437

Joint F-Test = 3,52 (0,0024)

Leap year

	Coefficients	T-Stat	P[|T| > t]
	1,4721	1,23	0,2218

Easter [8]

	Coefficients	T-Stat	P[|T| > t]
	1,5126	2,01	0,0451

Prespecified outliers

	Coefficients	T-Stat	P[|T| > t]
AO (4-2004)	38,7128	13,68	0,0000
LS (1-2001)	-8,7841	-3,28	0,0012
AO (4-2010)	-8,8868	-3,27	0,0012
AO (3-2004)	8,0340	2,85	0,0048
AO (12-2003)	6,9950	2,59	0,0101

The *Partial concurrent adjustment→ Estimate regression coefficients* revision policy results

Partial concurrent adjustment → Estimate regression coefficients + ARIMA parameters

The *Partial concurrent adjustment → Estimate regression coefficients + ARIMA parameters* strategy means that the ARIMA model, outliers and other regression parameters are not re-identified. All parameters of the Reg-ARIMA model are re-estimated but the explanatory variables remain the same. The transformation type remains unchanged.

The picture below presents the initial model (on the left) and the results of the refreshment procedure with the *Partial concurrent adjustment → Estimate regression coefficient + ARIMA parameters* option (on the right). The parameters of the ARIMA part have been re-estimated and their values have been updated. Also regression coefficients have been re-estimated and the number of estimated coefficients in the revised model is the same as in the initial model (i.e. 16 estimated coefficients). The structure of the model remains unchanged while all coefficients have been updated.

Summary

Estimation span: [7-1996 - 12-2016]
246 observations
Trading days effects (7 variables)
Easter [8] detected
5 detected outliers

Final model

Likelihood statistics
Number of effective observations = 233
Number of estimated parameters = 16

Loglikelihood = -559.6616717869163
Standard error of the regression (ML estimate) = 2.669031738674505
AIC = 1151.3233435738325
AICC = 1153.841862092351
BIC (corrected for length) = 2.314356746231504

Scores at the solution
-0,000004 -0,000414 .

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4902	-8,33	0,0000
BPhi(1)	0,1680	2,45	0,0152

Correlation of the estimates

	Theta(1)	BPhi(1)
Theta(1)	1,0000	0,0784
BPhi(1)	0,0784	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5102	-1,92	0,0562
Tuesday	0,2288	0,88	0,3774
Wednesday	0,1073	0,40	0,6905
Thursday	0,2028	0,75	0,4536
Friday	0,9280	3,48	0,0006
Saturday	-0,3434	-1,27	0,2071
Sunday (derived)	-0,6134	-2,28	0,0235

Joint F-Test = 8,13 (0,0000)

Summary

Estimation span: [7-1996 - 7-2017]
253 observations
Trading days effects (7 variables)
Easter [8] detected
5 pre-specified outliers

Final model

Likelihood statistics
Number of effective observations = 240
Number of estimated parameters = 16

Loglikelihood = -653.0614639550819
Standard error of the regression (ML estimate) = 3.668082944975884
AIC = 1338.1229279101638
AICC = 1340.5623897935718
BIC (corrected for length) = 2.9418782672541677

Scores at the solution
-0,000018 -0,000041 .

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,2036	-3,12	0,0020
BPhi(1)	0,3022	3,48	0,0006

Correlation of the estimates

	Theta(1)	BPhi(1)
Theta(1)	1,0000	0,0334
BPhi(1)	0,0334	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5098	-1,54	0,1252
Tuesday	0,4101	1,28	0,2024
Wednesday	0,1146	0,34	0,7327
Thursday	0,2243	0,65	0,5135
Friday	0,7773	2,32	0,0210
Saturday	-0,5492	-1,61	0,1078
Sunday (derived)	-0,4672	-1,38	0,1676

Joint F-Test = 5,80 (0,0000)

Partial concurrent adjustment → Estimate regression coefficients + outliers

The *Partial concurrent adjustment → Estimate regression coefficients + outliers* option means that the ARIMA model and regression parameters, except outliers, are not re-identified. The parameters of these variables, however, are re-estimated. All outliers are re-identified, i.e. the previous outcome of the outlier detection procedure is not taken into account and all outliers are identified and estimated once again. The transformation type remains unchanged.

The picture below presents the initial model (on the left) and the results of the refreshment procedure with the *Partial concurrent adjustment → Estimate regression coefficients + outliers* option (on the right). The parameters of the ARIMA part

have been re-estimated and their values have been updated. Also regression coefficients for the calendar variables have been re-estimated. In the revised model there is no *Prespecified outliers* section. Instead, the outliers were re-identified.

Summary

Estimation span: [7-1996 - 12-2016]
246 observations
Trading days effects (7 variables)
Easter [8] detected
5 detected outliers

Final model

Likelihood statistics

Number of effective observations = 233
Number of estimated parameters = 16

Loglikelihood = -559.6616717869163
Standard error of the regression (ML estimate) = 2.669031738674505
AIC = 1151.3233435738325
AICC = 1153.841862092351
BIC (corrected for length) = 2.314356746231504

Scores at the solution
-0,000004 -0,000414 .

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4902	-8,33	0,0000
BPhi(1)	0,1680	2,45	0,0152

Correlation of the estimates

	Theta(1)	BPhi(1)
Theta(1)	1,0000	0,0784
BPhi(1)	0,0784	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5102	-1,92	0,0562
Tuesday	0,2288	0,88	0,3774
Wednesday	0,1073	0,40	0,6905
Thursday	0,2028	0,75	0,4536
Friday	0,9280	3,48	0,0006
Saturday	-0,3434	-1,27	0,2071
Sunday (derived)	-0,6134	-2,28	0,0235

Joint F-Test = 8,13 (0,0000)

Leap year

	Coefficients	T-Stat	P[|T| > t]
	2,6712	3,07	0,0024

Easter [8]

	Coefficients	T-Stat	P[|T| > t]
	1,6759	3,08	0,0023

Outliers

	Coefficients	T-Stat	P[|T| > t]
AO (4-2004)	19,8713	10,12	0,0000
LS (1-2001)	-8,5643	-4,62	0,0000
AO (4-2010)	-8,7157	-4,62	0,0000
AO (3-2004)	8,6114	4,40	0,0000
AO (12-2003)	7,2918	3,90	0,0001

Summary

Estimation span: [7-1996 - 7-2017]
253 observations
Trading days effects (7 variables)
Easter [8] detected
6 detected outliers

Final model

Likelihood statistics

Number of effective observations = 240
Number of estimated parameters = 17

Loglikelihood = -573.4952681957561
Standard error of the regression (ML estimate) = 2.6361922842978505
AIC = 1180.9905363915123
AICC = 1183.747293148269
BIC (corrected for length) = 2.3040470471520735

Scores at the solution
-0,000185 -0,000728 .

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4908	-8,49	0,0000
BPhi(1)	0,1679	2,53	0,0120

Correlation of the estimates

	Theta(1)	BPhi(1)
Theta(1)	1,0000	0,0615
BPhi(1)	0,0615	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5212	-2,01	0,0454
Tuesday	0,2349	0,93	0,3516
Wednesday	0,1042	0,40	0,6899
Thursday	0,2102	0,79	0,4294
Friday	0,9059	3,51	0,0005
Saturday	-0,3332	-1,25	0,2117
Sunday (derived)	-0,6007	-2,29	0,0230

Joint F-Test = 8,39 (0,0000)

Leap year

	Coefficients	T-Stat	P[|T| > t]
	2,5121	3,04	0,0026

Easter [8]

	Coefficients	T-Stat	P[|T| > t]
	1,6820	3,27	0,0012

Outliers

	Coefficients	T-Stat	P[|T| > t]
AO (4-2004)	38,9692	20,10	0,0000
LS (1-2017)	38,7633	16,13	0,0000
LS (1-2001)	-8,5669	-4,68	0,0000
AO (4-2010)	-8,6967	-4,67	0,0000
AO (3-2004)	8,5865	4,45	0,0000
AO (12-2003)	7,2801	3,94	0,0001

Partial concurrent adjustment → Estimate regression coefficients + ARIMA model

The *Partial concurrent adjustment → Estimate regression coefficients + ARIMA model* option means that the ARIMA model, outliers and regression variables (except the calendar variables) are re-identified. All parameters are re-estimated. The transformation type remains unchanged.

The picture below presents the initial model (on the left) and the results of the refreshment procedure with the *Partial concurrent adjustment → Estimate regression coefficients + ARIMA model* option (on the right). The ARIMA part has been re-identified (a change from (2,1,0)(0,1,1) to (0,1,1)(1,1,1)). Also the regression coefficients for the calendar variables have been re-estimated. In the revised model there is no *Prespecified outliers* section. Therefore, the outliers were re-identified.

Summary

Estimation span: [1-2005 - 12-2016]
144 observations
Series has been log-transformed
Series has been corrected for leap year
Trading days effects (6 variables)
Easter [15] detected
4 detected outliers

Final model

Likelihood statistics
Number of effective observations = 131
Number of estimated parameters = 15

Loglikelihood = 330.49158009584664
Transformation adjustment = -608.1459835096218
Adjusted loglikelihood = -277.6544034137752

Standard error of the regression (ML estimate) = 0.01896469203769424
AIC = 585.3088068275504
AICC = 589.4827198710286
BIC (corrected for length) = -7.409339230469036

Scores at the solution
-0,004391 0,000967 -0,012902

Arima model
[(2,1,0)(0,1,1)].

	Coefficients	T-Stat	P[|T| > t]
Phi(1)	0,5040	5,65	0,0000
Phi(2)	0,2895	3,27	0,0014
BTheta(1)	-0,6188	-8,29	0,0000

Correlation of the estimates

	Phi(1)	Phi(2)	BTheta(1)
Phi(1)	1,0000	0,3982	0,1323
Phi(2)	0,3982	1,0000	0,0791
BTheta(1)	0,1323	0,0791	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	0,0000	0,00	0,9975
Tuesday	-0,0046	-1,49	0,1386
Wednesday	0,0044	1,45	0,1505
Thursday	-0,0032	-1,03	0,3070
Friday	0,0103	3,37	0,0010
Saturday	-0,0031	-0,98	0,3313
Sunday (derived)	-0,0038	-1,20	0,2308

Joint F-Test = 3,86 (0,0015)

Easter [15]

Coefficients	T-Stat	P[|T| > t]
0,0831	12,38	0,0000

Outliers

	Coefficients	T-Stat	P[|T| > t]
AO (12-2006)	0,1164	7,00	0,0000
AO (12-2015)	-0,0990	-5,79	0,0000
AO (11-2015)	-0,0727	-4,23	0,0000
TC (1-2009)	0,0854	5,20	0,0000

Summary

Estimation span: [1-2005 - 12-2017]
156 observations
Series has been log-transformed
Series has been corrected for leap year
Trading days effects (6 variables)
Easter [15] detected
1 detected outlier

Final model

Likelihood statistics
Number of effective observations = 143
Number of estimated parameters = 13

Loglikelihood = 383.2719601133891
Transformation adjustment = -713.7404772425816
Adjusted loglikelihood = -330.4685171291925

Standard error of the regression (ML estimate) = 0.015644939706339882
AIC = 686.937034258385
AICC = 689.7587396847416
BIC (corrected for length) = -7.89875302500467

Scores at the solution
0,001814 -0,001954 -0,000274

Arima model
[(0,1,1)(1,1,1)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4311	-5,41	0,0000
BPhi(1)	-0,3549	-3,54	0,0006
BTheta(1)	-0,9507	-12,77	0,0000

Correlation of the estimates

	Theta(1)	BPhi(1)	BTheta(1)
Theta(1)	1,0000	0,0569	0,4292
BPhi(1)	0,0569	1,0000	-0,0063
BTheta(1)	0,4292	-0,0063	1,0000

Regression model
Mean

	Coefficient	T-Stat	P[|T| > t]
mu	-0,0006	-2,10	0,0380

Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,0027	-1,14	0,2578
Tuesday	0,0015	0,64	0,5209
Wednesday	0,0023	0,96	0,3401
Thursday	0,0006	0,26	0,7982
Friday	0,0087	3,77	0,0002
Saturday	-0,0033	-1,37	0,1743
Sunday (derived)	-0,0072	-2,96	0,0036

Joint F-Test = 8,70 (0,0000)

Easter [15]

Coefficients	T-Stat	P[|T| > t]
0,0197	3,92	0,0001

Outliers

	Coefficients	T-Stat	P[|T| > t]
AO (4-2010)	-0,0554	-4,02	0,0001

Partial concurrent adjustment → Estimate regression coefficients + Last outliers

The *Partial concurrent adjustment → Estimate regression coefficients + Last outliers* strategy means that the ARIMA model, outliers (except for the last year of the sample) and other regression parameters are not re-identified. All parameters of the Reg-ARIMA model are re-estimated. The software tests for outliers in the last year of the data span and will include in the model those which are statistically significant. The transformation type remains unchanged.

The picture below presents the initial model (on the left) and the results of the refreshment procedure with the *Partial concurrent adjustment → Estimate regression coefficients + Last outliers* option (on the right). The parameters of the ARIMA part have been re-estimated and their values have been updated. Also the regression coefficients have been re-estimated. The number of estimated coefficients in the revised model is larger than the initial model because an additional outlier has been identified in the last year of the data span.

Summary

Estimation span: [7-1996 - 12-2016]
246 observations
Trading days effects (7 variables)
Easter [8] detected
5 detected outliers

Final model

Likelihood statistics
Number of effective observations = 233
Number of estimated parameters = 16

Loglikelihood = -559.6616717869163
Standard error of the regression (ML estimate) = 2.669031738674505
AIC = 1151.3233435738325
AICC = 1153.841862092351
BIC (corrected for length) = 2.314356746231504

Scores at the solution
-0,000004 -0,000414 .

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4902	-8,33	0,0000
BPhi(1)	0,1680	2,45	0,0152

Correlation of the estimates

	Theta(1)	BPhi(1)
Theta(1)	1,0000	0,0784
BPhi(1)	0,0784	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5102	-1,92	0,0562
Tuesday	0,2288	0,88	0,3774
Wednesday	0,1073	0,40	0,6905
Thursday	0,2028	0,75	0,4536
Friday	0,9280	3,48	0,0006
Saturday	-0,3434	-1,27	0,2071
Sunday (derived)	-0,6134	-2,28	0,0235

Joint F-Test = 8,13 (0,0000)

Leap year

Coefficients	T-Stat	P[|T| > t]
2,6712	3,07	0,0024

Easter [8]

Coefficients	T-Stat	P[|T| > t]
1,6759	3,08	0,0023

Outliers

	Coefficients	T-Stat	P[|T| > t]
AO (4-2004)	19,8713	10,12	0,0000
LS (1-2001)	-8,5643	-4,62	0,0000
AO (4-2010)	-8,7157	-4,62	0,0000
AO (3-2004)	8,6114	4,40	0,0000
AO (12-2003)	7,2918	3,90	0,0001

Summary

Estimation span: [7-1996 - 7-2017]
253 observations
Trading days effects (7 variables)
Easter [8] detected
5 pre-specified outliers
1 detected outlier

Final model

Likelihood statistics
Number of effective observations = 240
Number of estimated parameters = 17

Loglikelihood = -573.4952681950751
Standard error of the regression (ML estimate) = 2.6361922969352976
AIC = 1180.9905363901503
AICC = 1183.747293146907
BIC (corrected for length) = 2.3040470567397255

Scores at the solution
-0,000012 -0,000444 .

Arima model
[(0,1,1)(1,1,0)].

	Coefficients	T-Stat	P[|T| > t]
Theta(1)	-0,4908	-8,49	0,0000
BPhi(1)	0,1679	2,53	0,0120

Correlation of the estimates

	Theta(1)	BPhi(1)
Theta(1)	1,0000	0,0615
BPhi(1)	0,0615	1,0000

Regression model
Trading days

	Coefficients	T-Stat	P[|T| > t]
Monday	-0,5212	-2,01	0,0454
Tuesday	0,2349	0,93	0,3516
Wednesday	0,1042	0,40	0,6899
Thursday	0,2102	0,79	0,4294
Friday	0,9059	3,51	0,0005
Saturday	-0,3332	-1,25	0,2117
Sunday (derived)	-0,6007	-2,29	0,0230

Joint F-Test = 8,39 (0,0000)

Leap year

Coefficients	T-Stat	P[|T| > t]
2,5121	3,04	0,0026

Easter [8]

Coefficients	T-Stat	P[|T| > t]
1,6820	3,27	0,0012

Prespecified outliers

	Coefficients	T-Stat	P[|T| > t]
AO (4-2004)	38,9692	20,10	0,0000
LS (1-2001)	-8,5669	-4,68	0,0000
AO (4-2010)	-8,6967	-4,67	0,0000
AO (3-2004)	8,5865	4,45	0,0000
AO (12-2003)	7,2801	3,94	0,0001

Outliers

	Coefficients	T-Stat	P[|T| > t]
LS (1-2017)	38,7633	16,13	0,0000

Implementation with the cruncher

In a production process, it might be suitable to use the cruncher in order to automatically update workspaces. When using an R package (`rjwsacruncher`` or`JDCruncheR‘) to do so, you will just need to specify the policy’s name as shown below. Available policies and names are detailed in the #Overview section.

```
cruncher_and_param(
    workspace = "D:/my_folder/my_ws.xml",
    rename_multi_documents = FALSE,
    policy = "stochastic", # name of the revision policy
    log = my_log_file.txt
)
```

Implementation in R

Implementing refresh policies is a new v3.x feature. Two options are available

- using rjd3x13 or rjd3tramoseats directly on TS objects in R
- refreshing a workspace with rjd3workspace

When performing seasonal adjustment directly in R with `rjd3x13` or `rjd3tramoseats`, you will need to refresh the “result_spec” yielded by the previous estimation with the selected policy.

Available policies and names are detailed in here.

More explanations and full documentation of

- `rjd3x13::x13_refresh` function can be found here
- `rjd3tramoseats::tramoseats_refresh` function can be found here

SA of high-frequency data

In this chapter

The sections below provide 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

Up coming content:

- Graphical User Interface 3.x 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 15: 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.x	R package
Pre-treatment	Extended Airline Model	✔	rjd3highfreq
Decomposition	Extended Seats Extended Airline Model	✔	rjd3highfreq
	Extended X-11	✔	rjd3x11plus
	Extended STL	✖	rjd3stl
One-Step	SSF Framework	✖	rjd3sts

SA algorithms extended for high-frequency data

All algorithms are available via an R package and will be available in GUI (in target v 3.x version)

- Extended Airline estimation, reg-ARIMA like (`rjd3highfreq` and GUI)
- Extended Airline Decomposition, Seats like (`rjd3highfreq` and GUI)
- MX12+ (`rjd3x11plus`, GUI upcoming)
- MSTL+ (`rjd3stl` and in GUI)
- MSTS (`rjd3sts`, GUI upcoming)

Data frequencies and seasonal patterns

In the Graphical User Interface (display constraints)

Input data: daily, weekly

Seasonal patterns: weekly ($p = 7$) or yearly ($p = 365.25$ or $p = 52.18$)

In corresponding R packages:

- no constraint on data input as no TS structure (numeric vector)
- any seasonal patters, positive numbers

Unobserved Components

Raw series decomposition

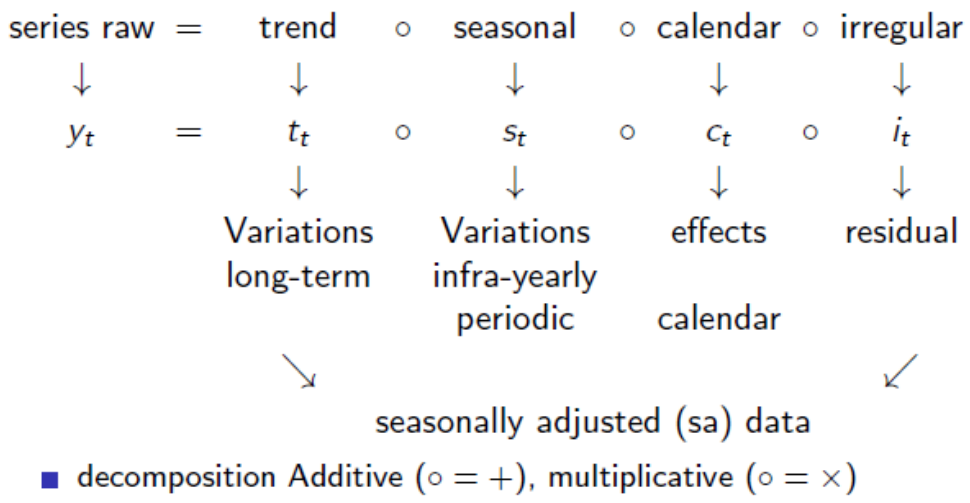


Figure 62: **Decomposition diagram**

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 X-13-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:

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

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

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

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

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

$$\epsilon_t \overset{iid}{\sim} \mathsf{N}(0, \sigma_\epsilon^2)$$

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(
    calendar = frenchCalendar,
    start = "1968-01-01",
    length = length(df_daily$births),
    type = "All",
    nonworking = 7L
)

# 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:

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

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: $p = 7 \rightarrow l = 7$, $p = 12 \rightarrow l = 13$, $p = 365.25 \rightarrow l = 367$, $p = 52.18 \rightarrow l = 53$
- 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 $M_{2\times12}$ 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 $M_{3\times3}$

$$M_{3\times3}X = \frac{1}{9}(X_{t-2p}) + \frac{2}{9}(X_{t-p}) + \frac{3}{9}(X_t) + \frac{2}{9}(X_{t+p}) + \frac{1}{9}(X_{t+2p})$$

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 $p = 365.25$

```
# extraction of day-of-the-week pattern (dow)
x11.dow <- rjd3x11plus::x11plus(y_linearized,
    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

# extraction of day-of-the-week pattern (doy)
x11.doy <- rjd3x11plus::x11plus(x11.dow$decomposition$sa, # previous sa
    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 pattern in the final SA data.

SA: X12+ and MX12+

In this chapter

Additional SA algorithms available only in v3.x.

- X12+: Airline based pre-adjustment and extended X11 decomposition
- MX12+: Extended Airline Estimation and Extended X11 Decomposition

Up coming content.

SA: STL+ and MSTL+

STL is a Loess (Weighted local regression) based decomposition algorithm used on linearized data, no integrated pre-adjustment.

In this Chapter

We will cover how to use classic STL JDemetra+. M-STL functions for tackling multiple periodicities (with rounded frequencies) in infra-monthly data will be described in the high-frequency data related chapter.

More methodological details will be provided here

Tools for access

In JDemetra+ STL is only available through rjd3stl package.

SA: STS and MSTS

Basic Structural models (BSM) allow to decompose a time series with explicit models for its components (S,T,I) while running pre-treatment in one single step. It also allows to integrate time varying trading-day correction.

In this Chapter

We will cover how to perform seasonal adjustment using BSM in JDemetra+.

How to tackle multiple periodicities (with rounded frequencies) in infra-monthly data will be described in the high-frequency data related chapter.

More methodological details will be provided here

Tools for access

In JDemetra+ Basic Structural Models are only available through rjd3sts package.

Outlier detection and external regressors

In this chapter

The following sections describe

- 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 the Graphical User Interface (GUI) or R packages((#t-r-packs).

How to use the generated regressors, or any user-defined variable, in a seasonal adjustment or Reg-ARIMA modelling process is discussed in the pre-treatment chapter for classic SA and SA of High-frequency chapter for infra-monthly 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 exclude calendar correction which is detailed here

Generating external regressors

Outliers

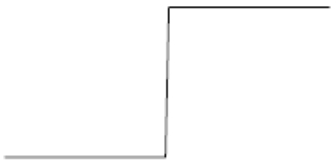
Types

The following outliers are available for automatic detection

ADD: - specifics for HF data (wo outiler)

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.

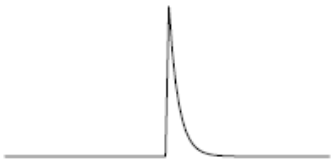


Figure 63: **Outliers type**

Seasonal Outlier (SO)

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

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

Figure 64: **Seasonal outlier**

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 pre-treatment in SA and SA of High-frequency data.

Generating regressors for outliers

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

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

```
# Outliers in February 2002, for monthly data
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.

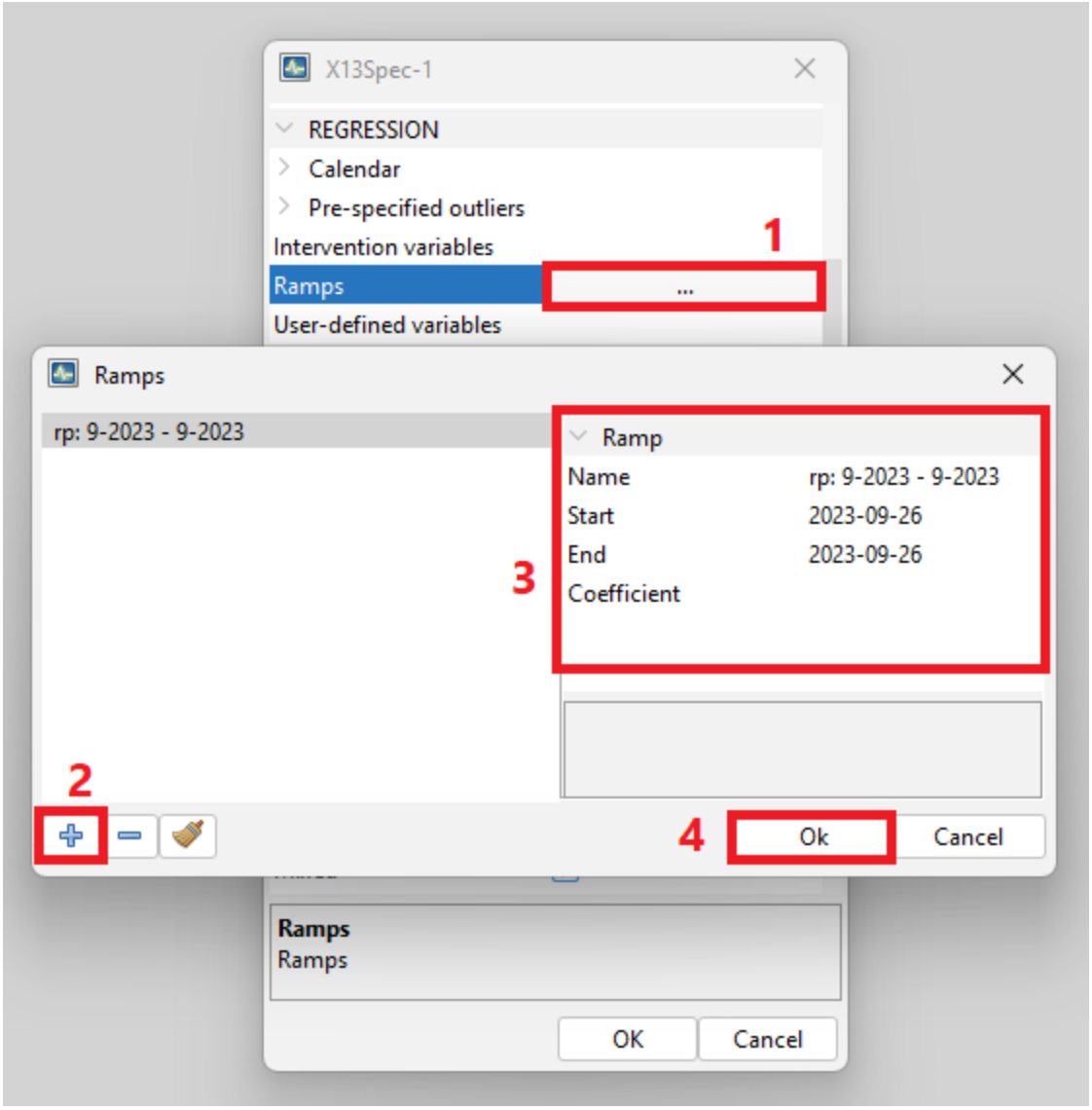


Figure 65: **Add ramp in GUI**

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

User-defined variables

User defined variables are simply time series used as explanatory regressors in the Reg-ARIMA and the Tramo models. Although JDemetra+ allows the user to indicate any time series as a variable to avoid misleading or erroneous results, the following rules should be kept:

- User-defined regression variables are used for measuring abnormalities and therefore they should not contain a seasonal pattern.
- JDemetra+ assumes that user-defined regressors are already in an appropriately centred form.

Therefore the mean of each user-defined regressor needs to be subtracted from the regressor or means for each calendar period (month or quarter) need to be subtracted from each of the user-defined regressors.

JDemetra+ considers two kinds of user-defined regression variables:

- **Static variables**, usually imported directly from external software (by drag/drop or copy/paste). The observations for static variables cannot be changed. The only way to update static series is to remove them from the list and to re-import them with the same names.
- **Dynamic variables** that are imported into the *Variables* panel by dragging and dropping series from a browser of the application, available in the *Providers* window. Dynamic variables are automatically updated each time the application is re-opened. Therefore, it is a convenient solution for creating user-defined variables.

In GUI

To create a dynamic variable first right-click on the *Variables* node in the *Workspace* window and chose the option **New**.

Next, double click on the newly created *Vars-1* item to display it in the *Results* panel. By default, JDemetra+ uses the conventions Vars_#number to name the tabs under the *Variables* node.

Then, go to *Providers* window and open your file that contains external variables. Drag and drop your external regressors from the *Providers* window to the *Vars-1* window.

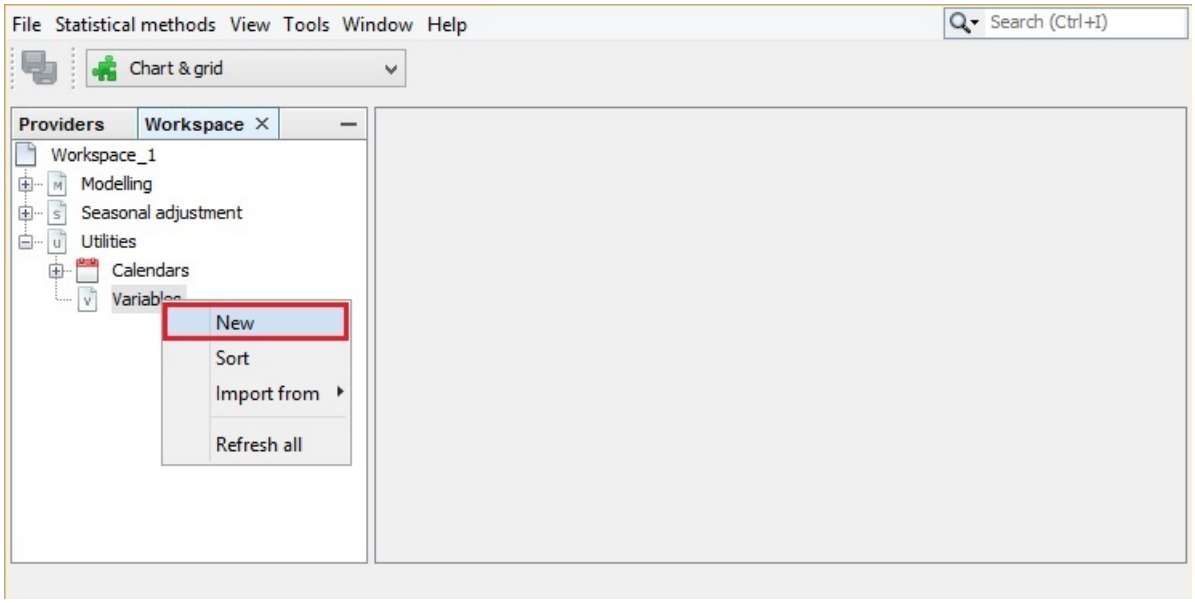


Figure 66: **Creating an empty dataset for the user-defined variables**

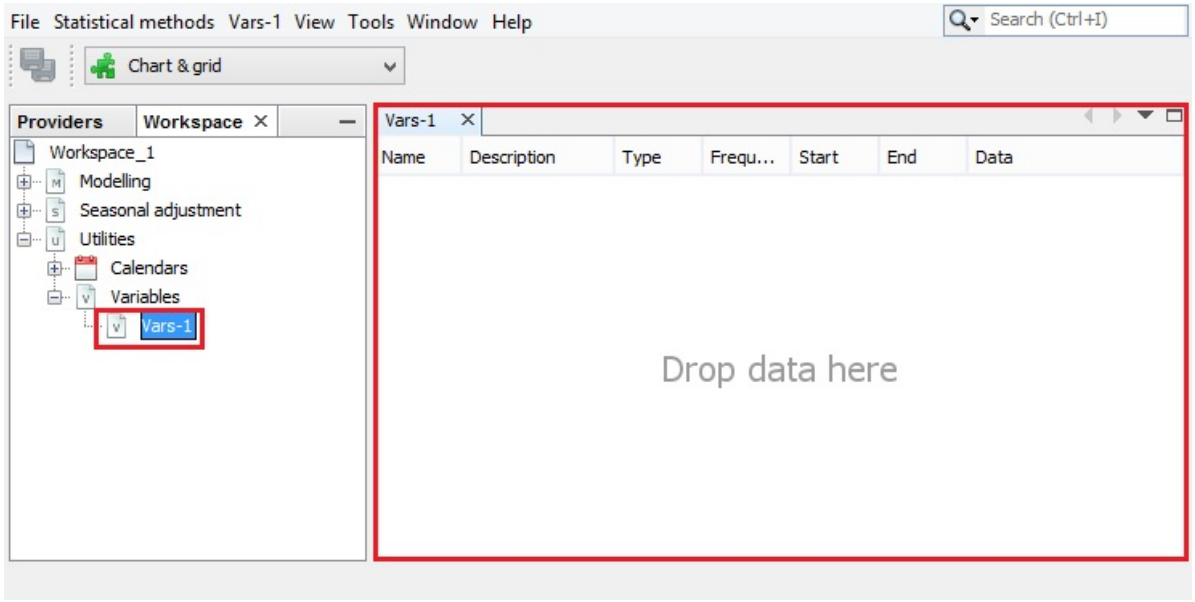


Figure 67: **Activation of an empty dataset for the user-defined variables**

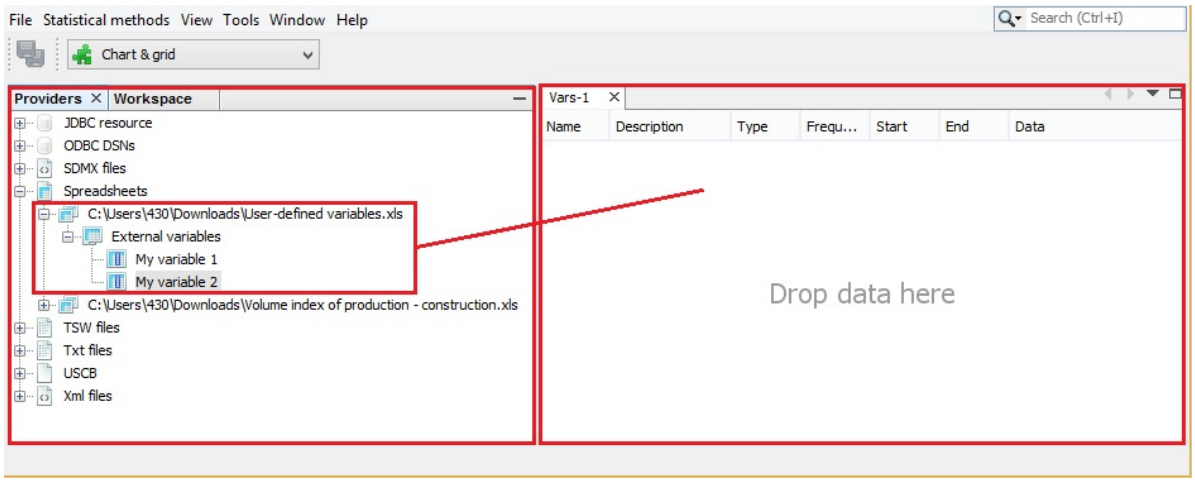


Figure 68: **Importing the user-defined variables to JDemetra+**

The original name of the series is recorded in the *Description* column of the *Variables* window.

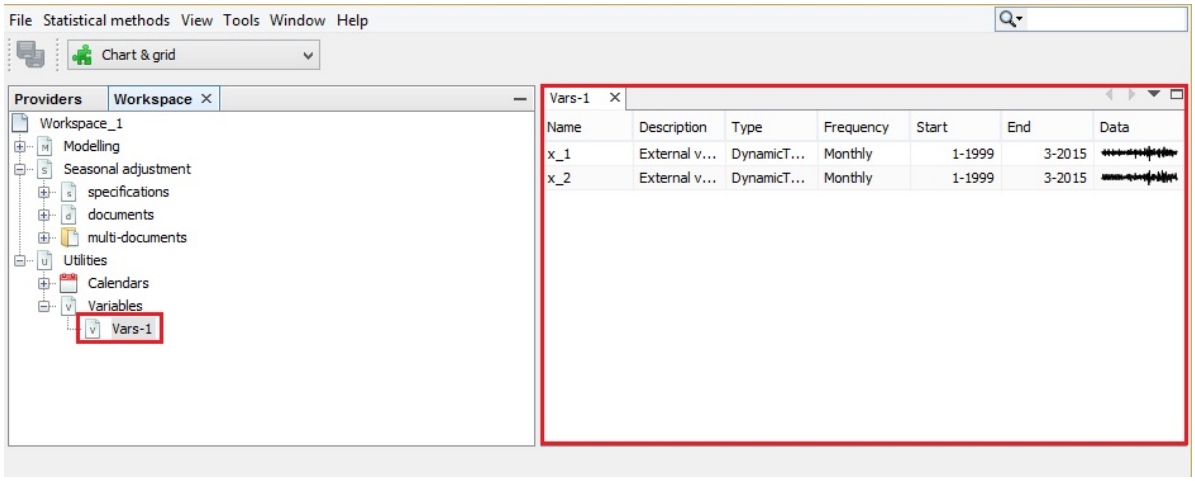


Figure 69: **Assigning regressors from the *Providers* window to the user-defined variables**

In order to rename the series in the *Variables* window, right click on the series and chose **Rename**.

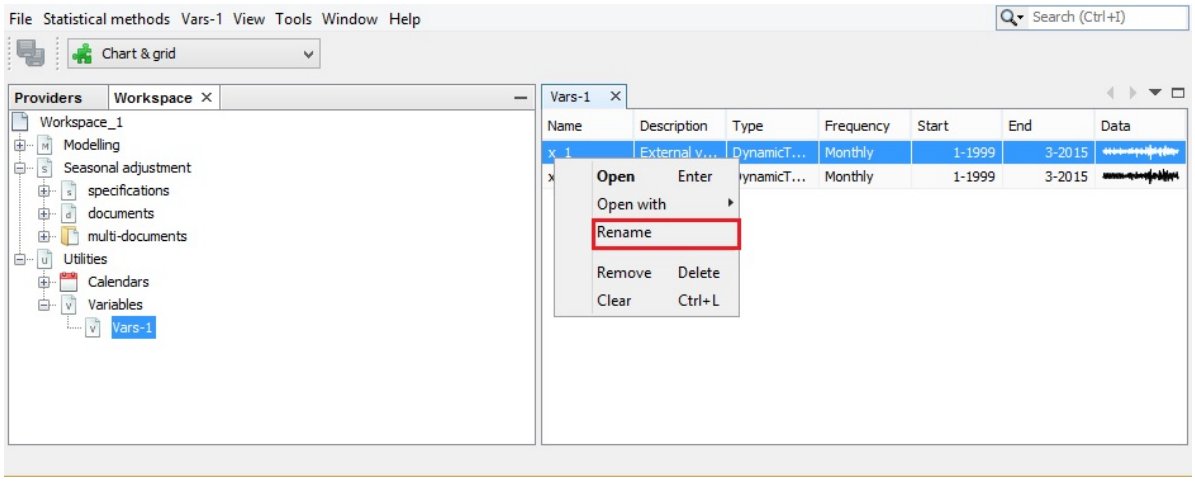


Figure 70: **A local menu for the user-defined variables**

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

In version 3;x, R packages rjd3x13 and rjd3tramoseats provide functions for detecting outliers with Reg-ARIMA (tramo) algorithms.

Example using `regarima_outliers`in rjd3x13:

```
library(rjd3x13)
?regarima_outliers
regarima_outliers(rjd3toolkit::ABS$X0.2.09.10.M,
    order = c(1, 1, 1), seasonal = c(0, 1, 1),
```

```
    mean = FALSE,
    X = NULL, X.td = NULL,
    ao = TRUE, ls = FALSE, tc = TRUE, so = TRUE, cv = 4
)
```

Example wit rjd3tramoseats::tramo_outliers

```
library(rjd3tramoseats)
?tramo_outliers
tramo_outliers(rjd3toolkit::ABS$X0.2.09.10.M,
    order = c(1, 1, 1), seasonal = c(0, 1, 1),
    mean = FALSE,
    X = NULL, X.td = NULL,
    ao = TRUE, ls = FALSE, tc = TRUE, so = TRUE, cv = 4
)
```

Specific TERROR tool

Up coming content

With structural models (BSM)

Up coming content

Calendar correction

In this Chapter

This chapter is divided in two parts. The first one (theory) outlines the rationale for 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 X-13-ARIMA and Tramo-Seats it boils down to adding suitable regressors to the pre-treatment 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

- 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 pre-treatment and SA of High-Frenquency data.

How to generate other types of regressors is described here and how to plug them into Reg-ARIMA models is detailed here

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 $\bar{D_{i,J}^y}$ 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:

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

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 $(1 - \frac{w_i}{w_K})(D_{tK} - \bar{D_{K,J}^y})$, 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 Reg-ARIMA models

In the context of Reg-ARIMA modelling, 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.

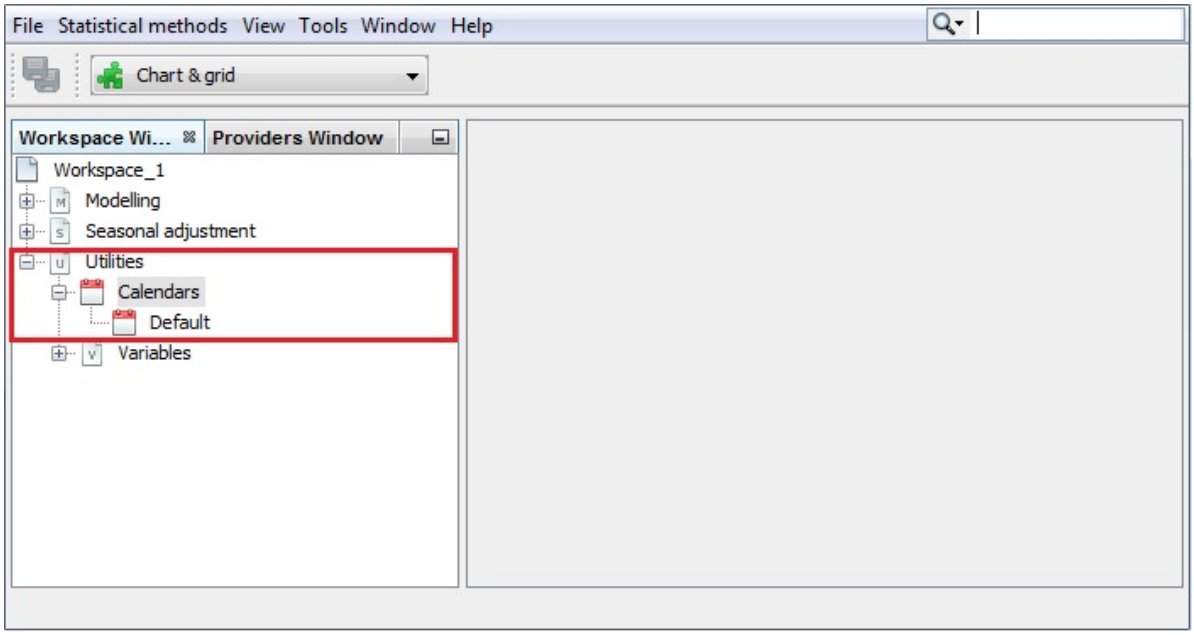


Figure 71: Text

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

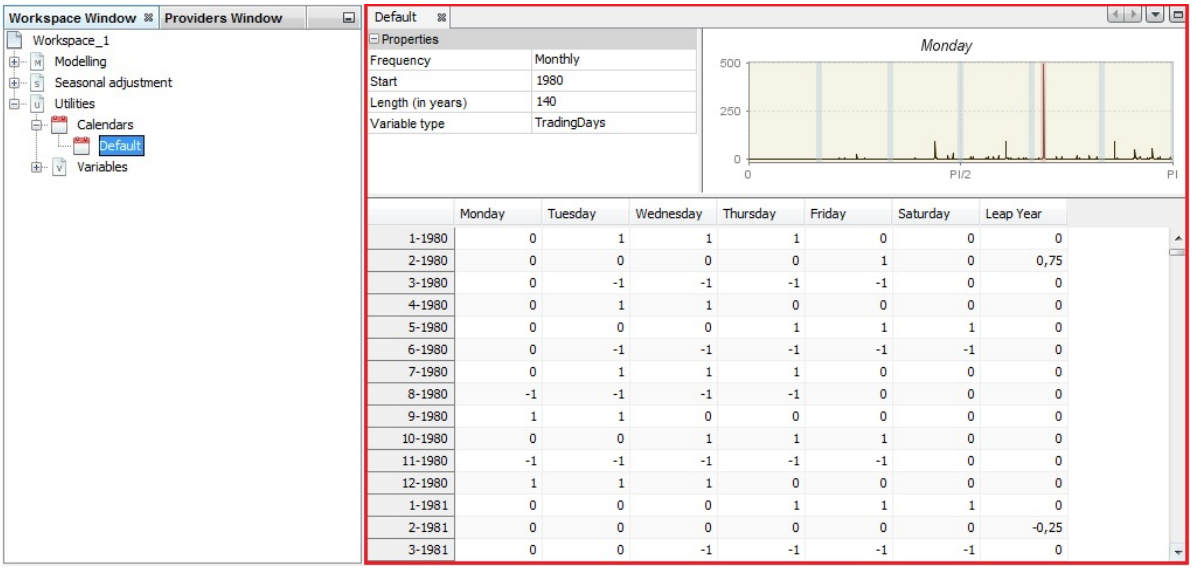


Figure 72: Text

0.0.0.0.2 * Set Properties

In the *Properties* panel the user can set:

- 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.

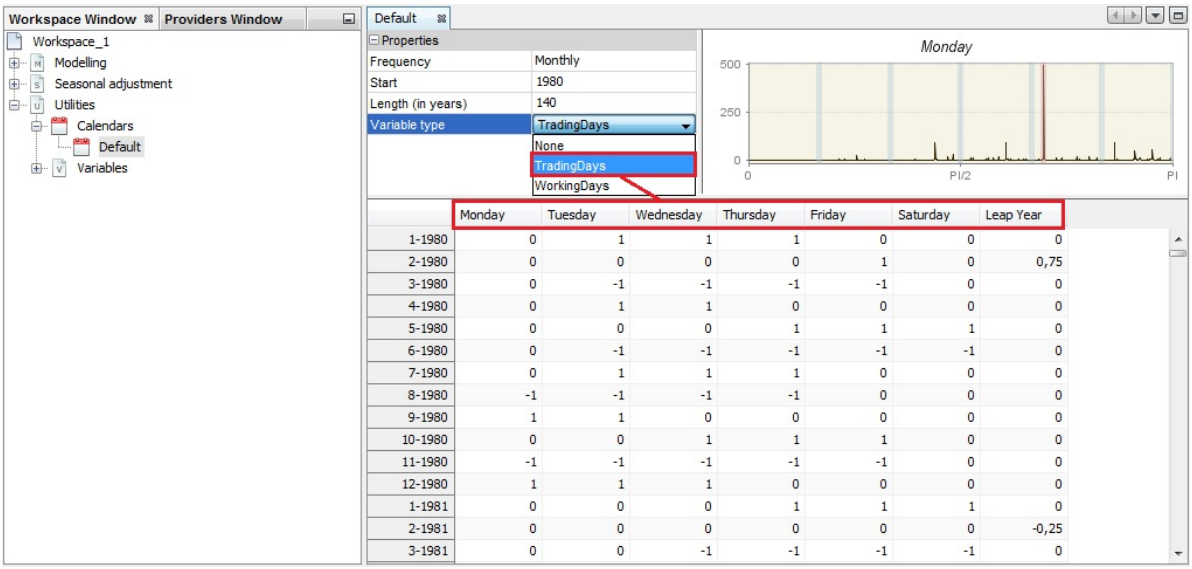


Figure 73: Text

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

Modify an existing Calendar

- click the option *Edit* from the context menu
- the list of holidays defined for this calendar is displayed
- To add a holiday unfold the **+** menu
- To remove a holiday click on it and choose the **-** button

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**

Three options are available:

- *National calendars*: allows to include country-specific holidays

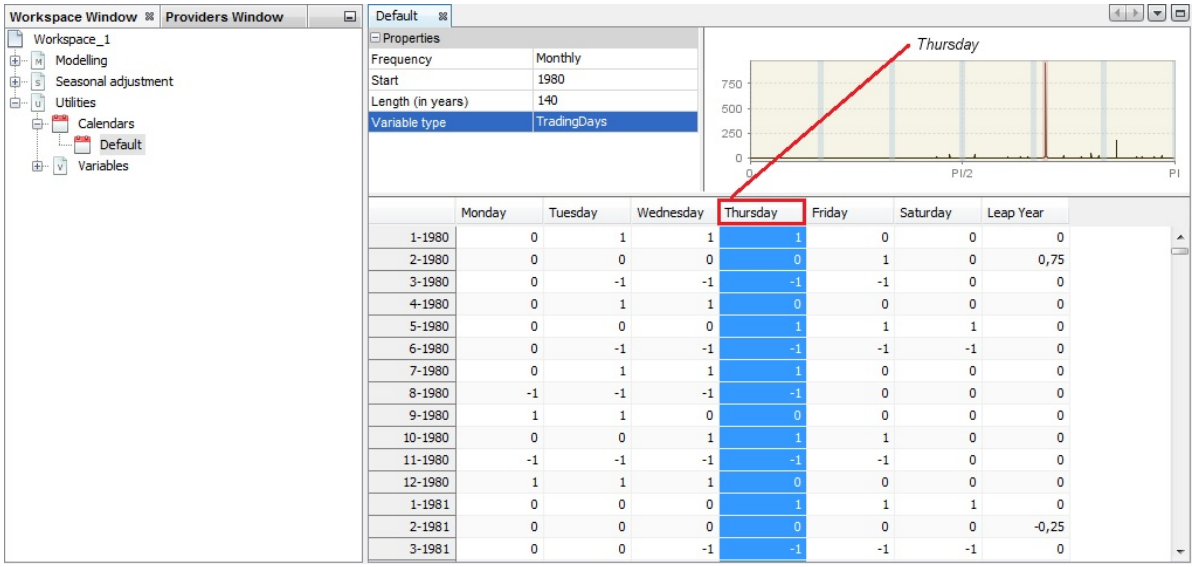


Figure 74: Text

- *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

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

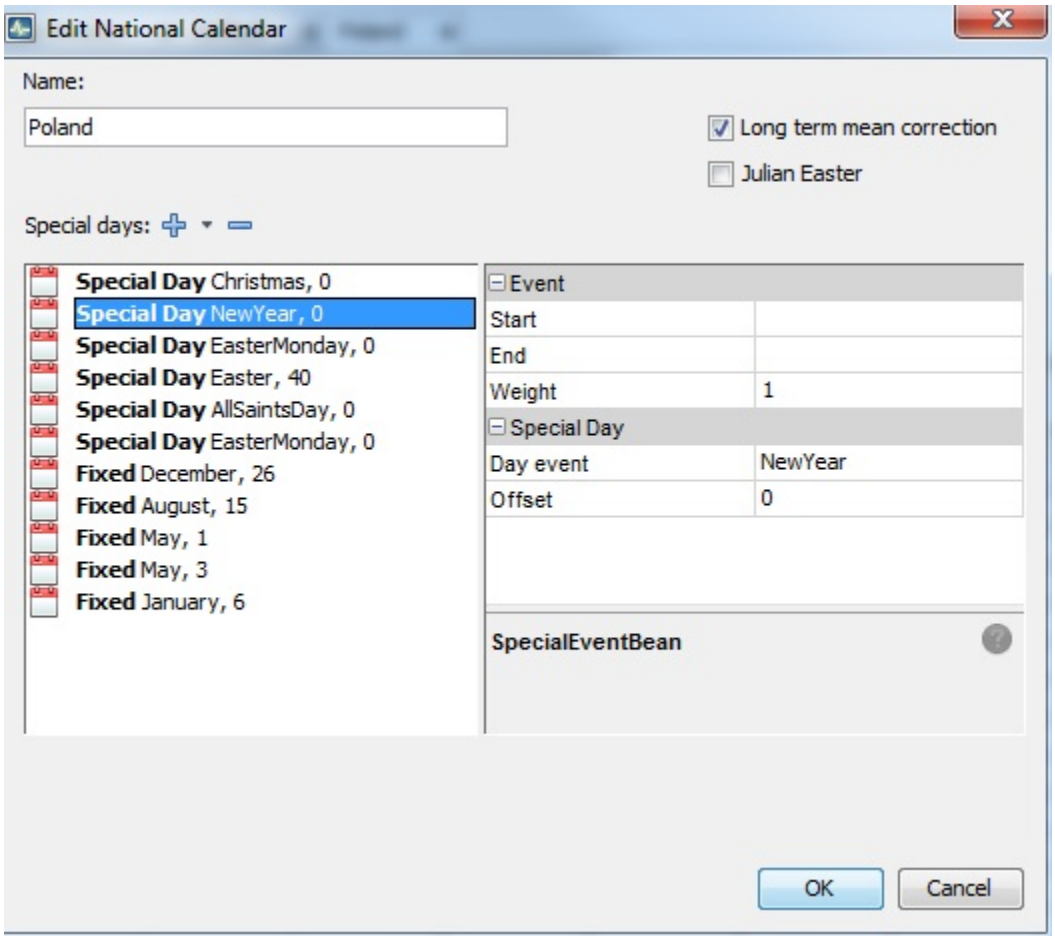


Figure 75: **Edit a calendar window**

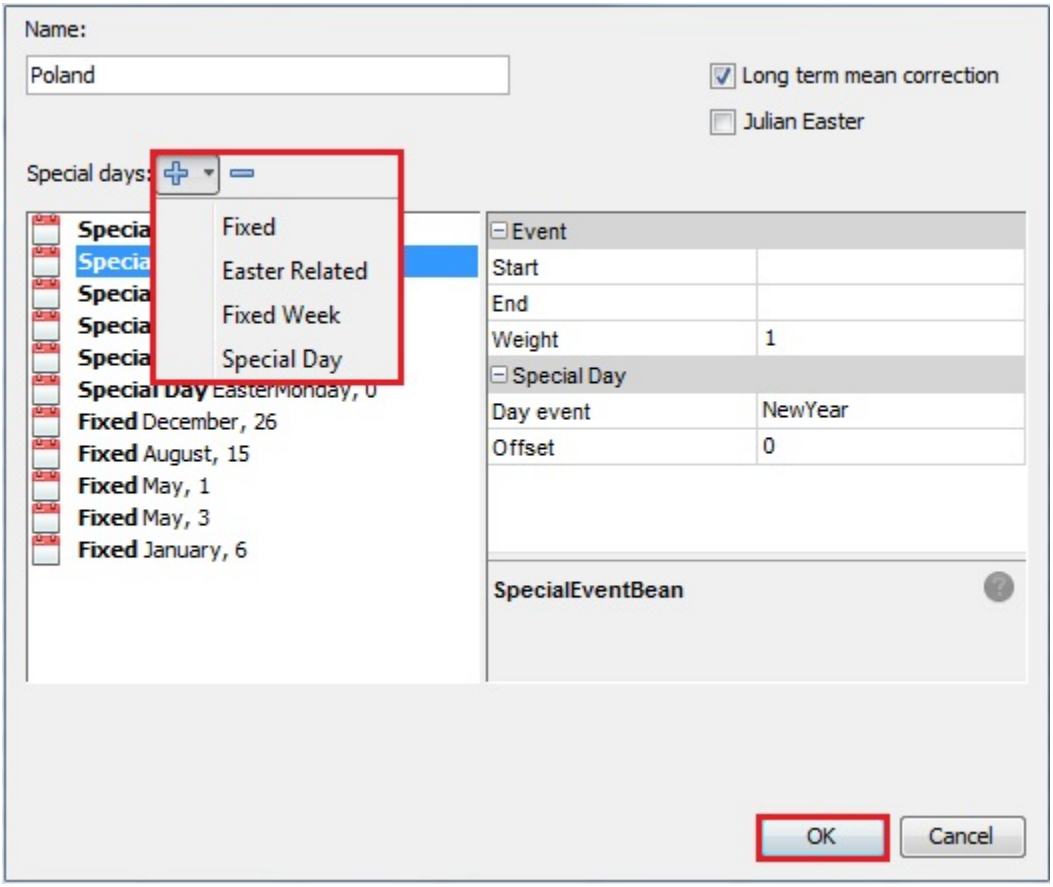


Figure 76: **Removing a holiday from the calendar**

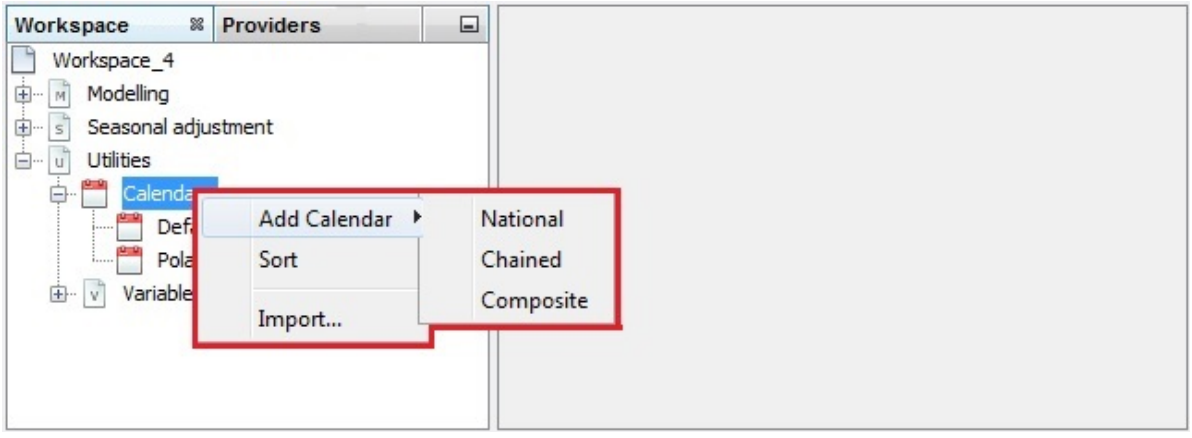


Figure 77: Text

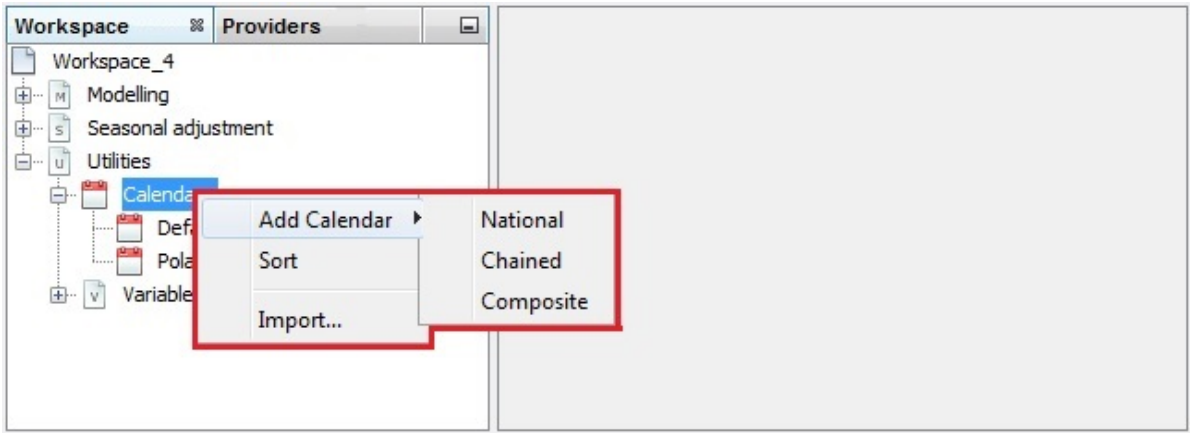


Figure 78: Text

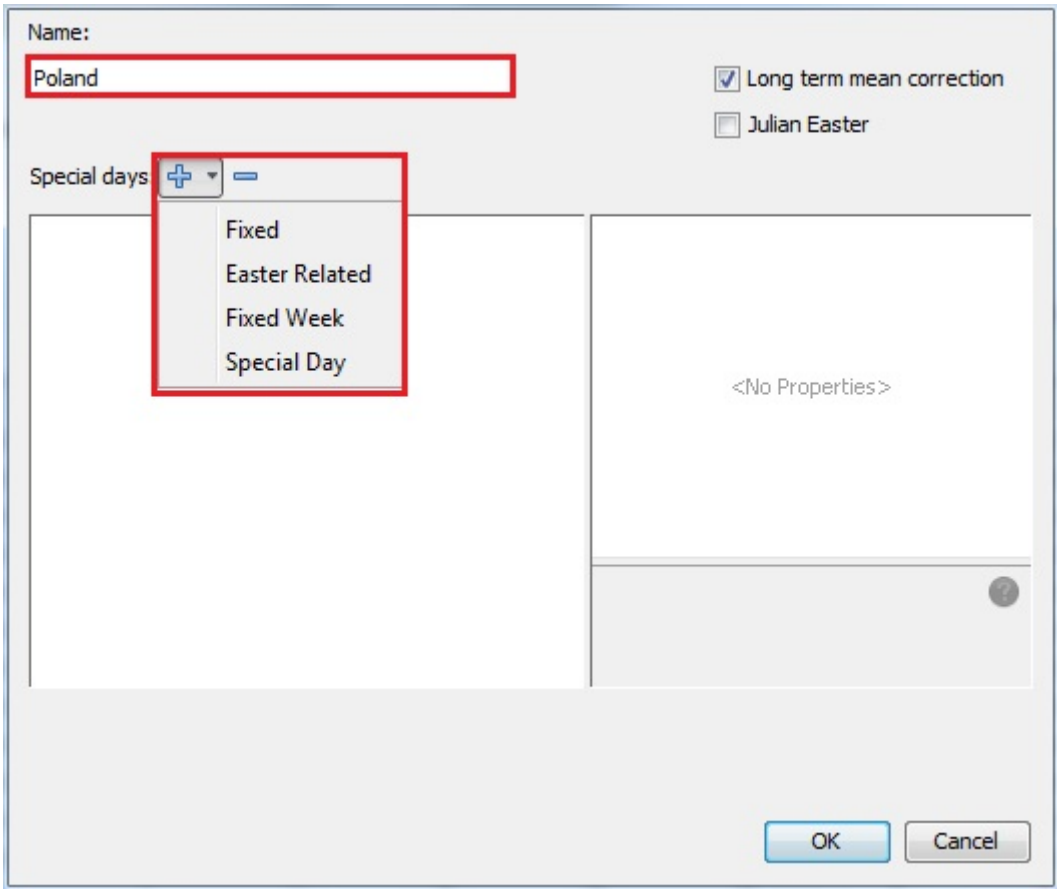


Figure 79: Text

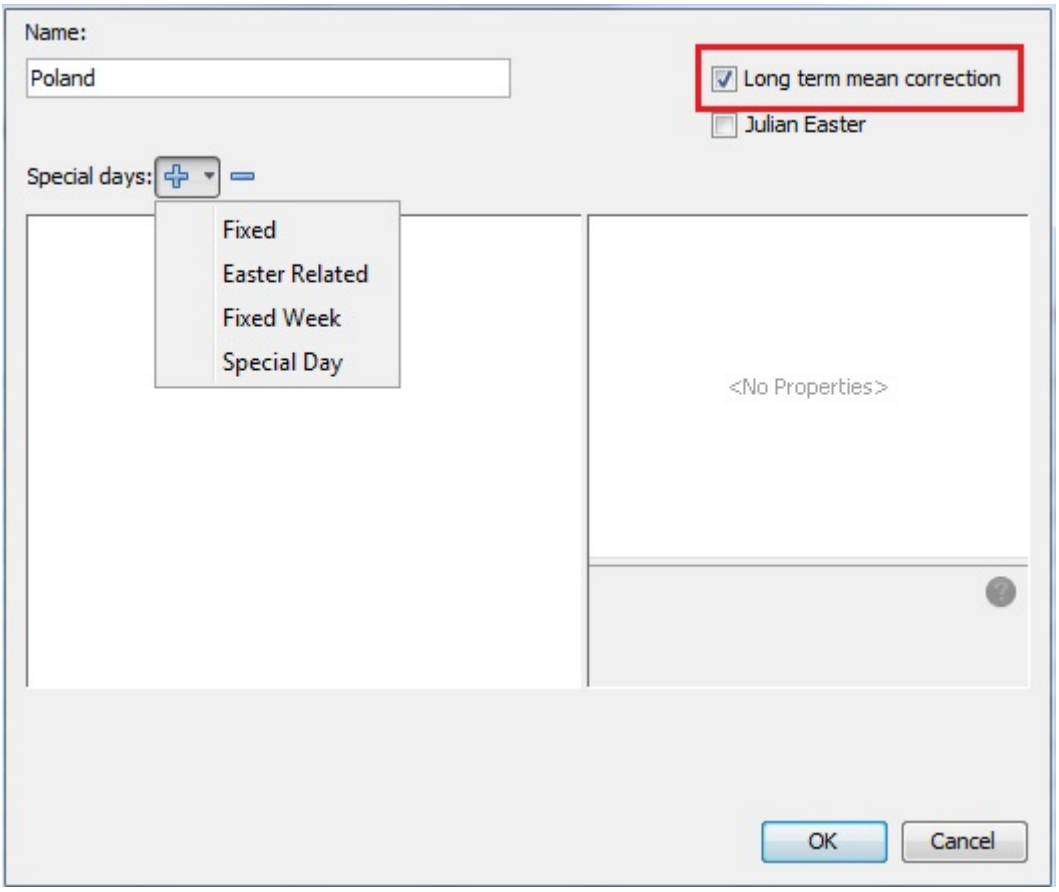


Figure 80: Text

- to use Julian Easter

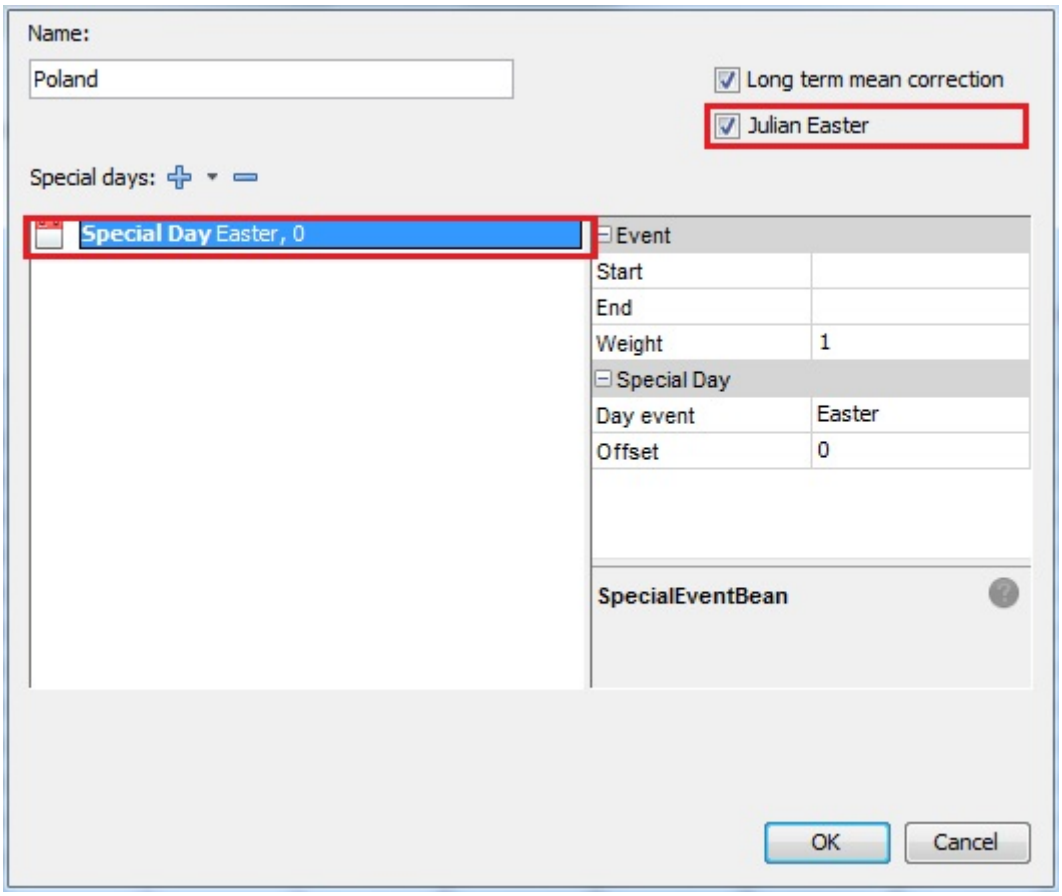


Figure 81: Text

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

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

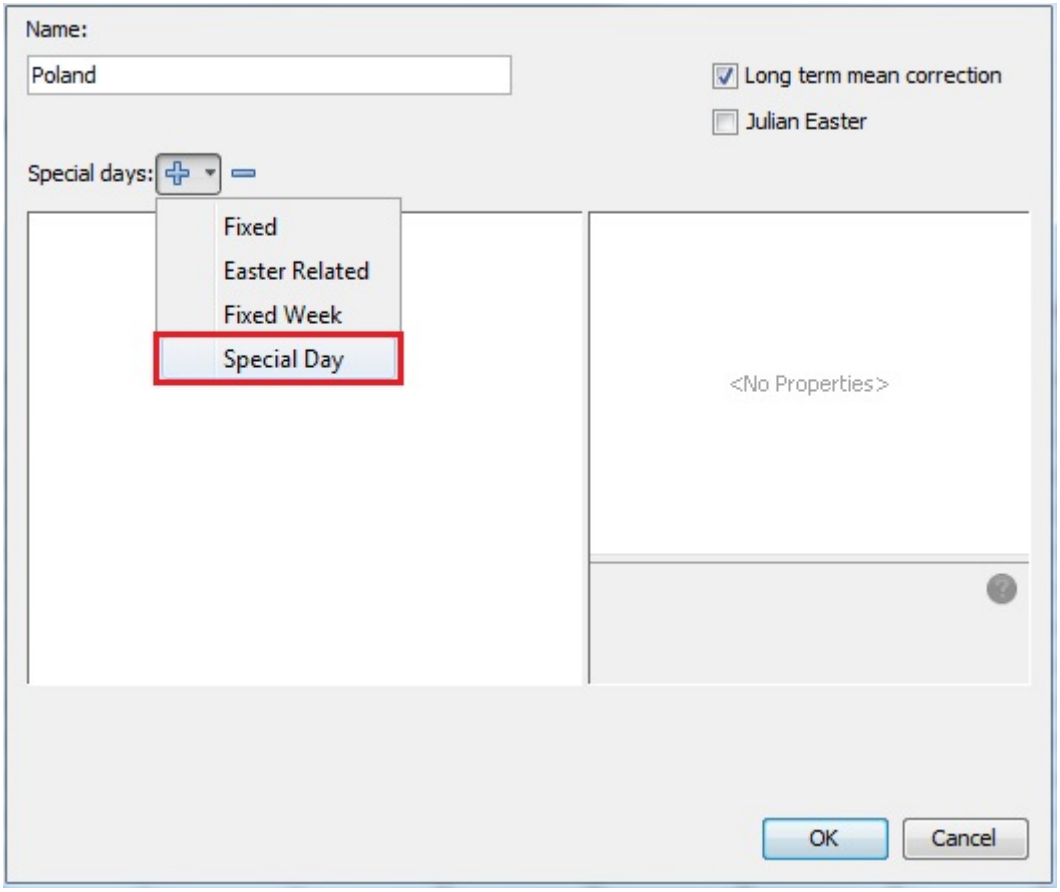


Figure 82: **Adding a pre-defined holiday to the calendar**

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.

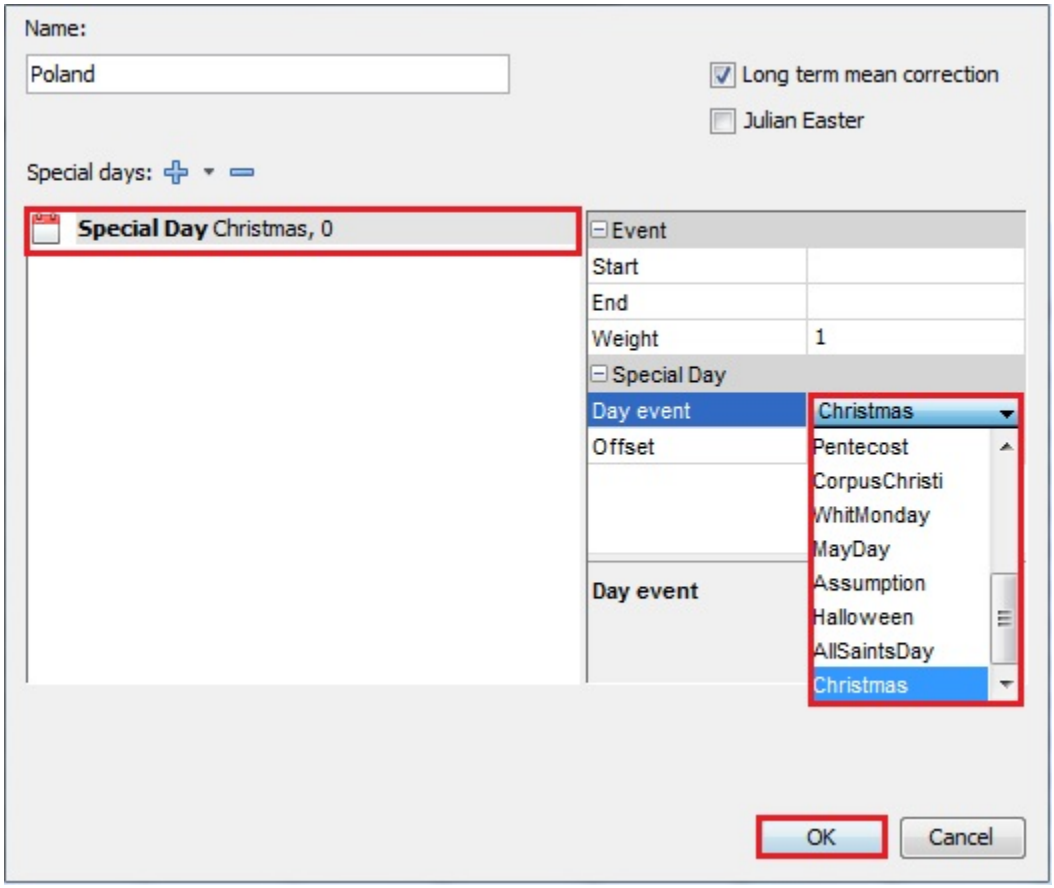


Figure 83: **Choosing a pre-defined holiday from the list**

- 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.

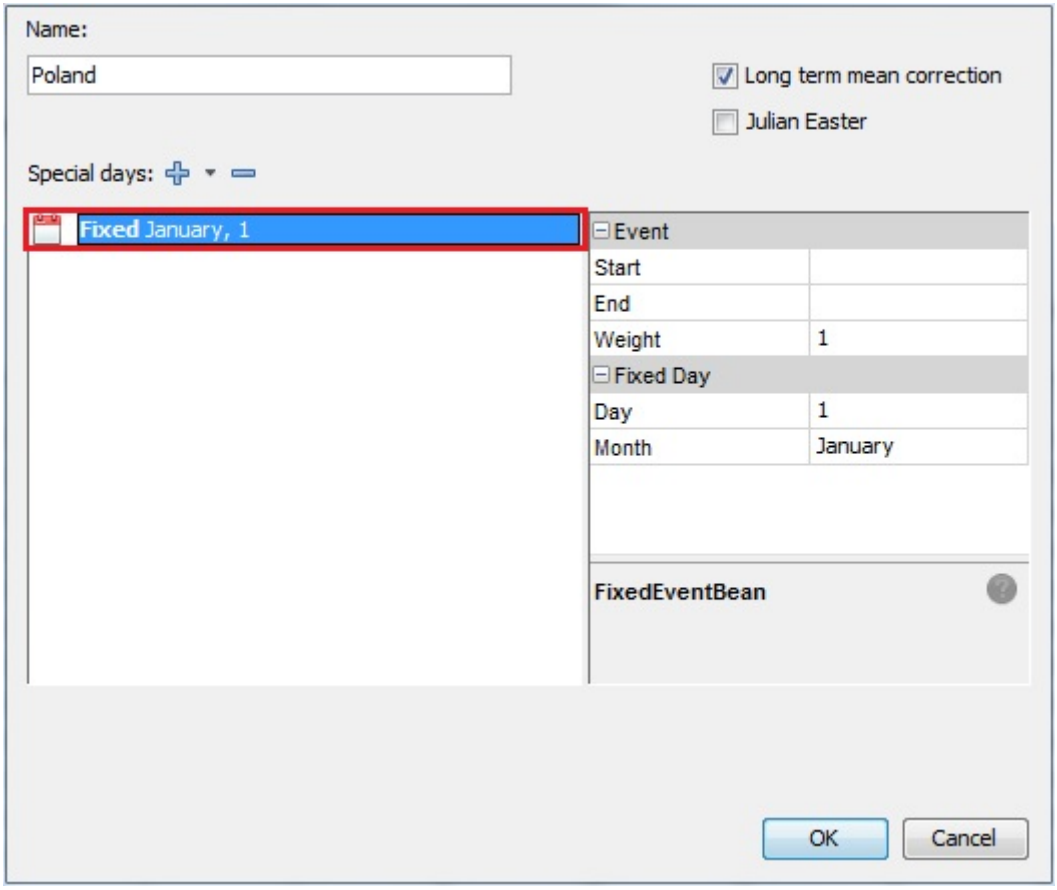
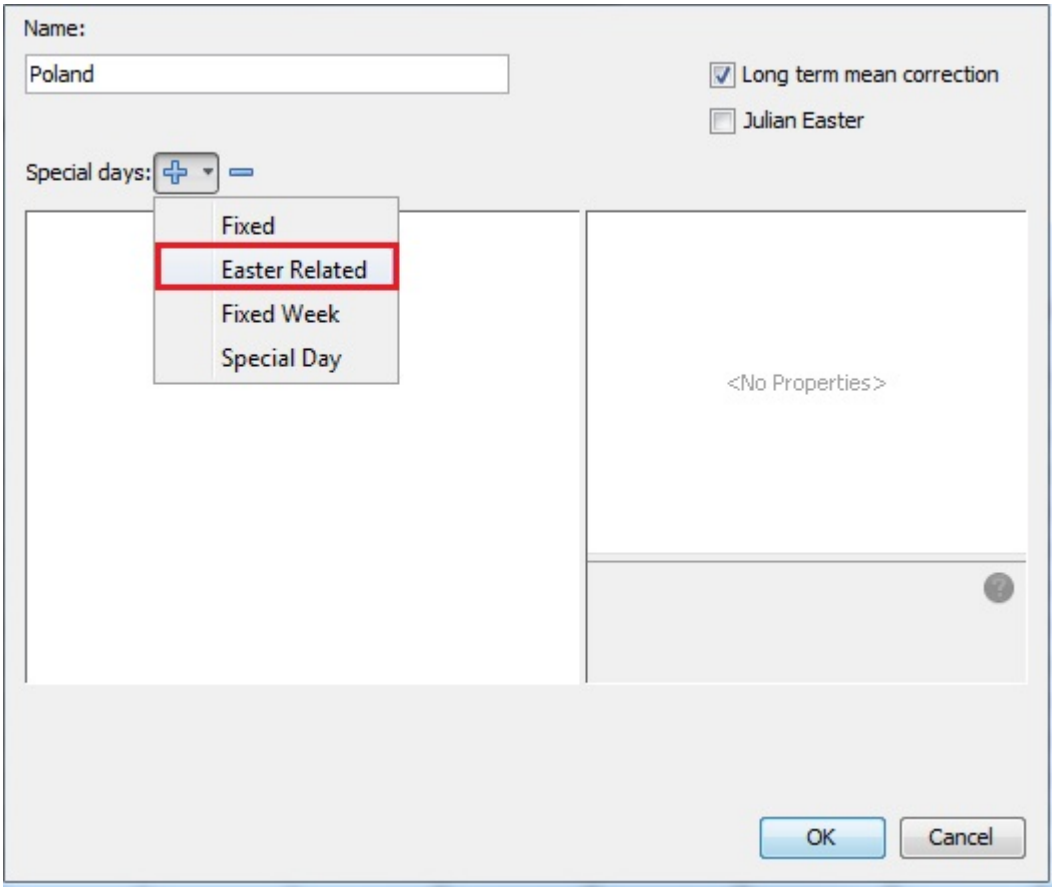


Figure 84: **Options for a fixed holiday**

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

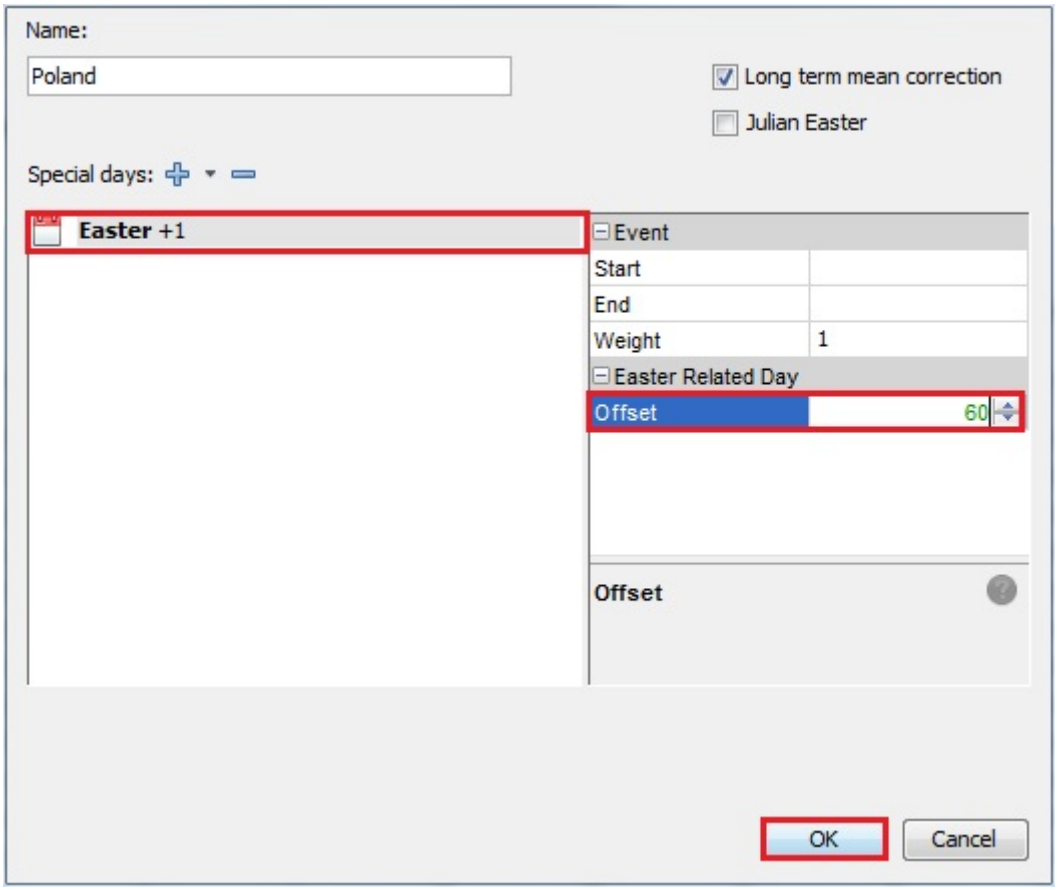


Figure 85: Text

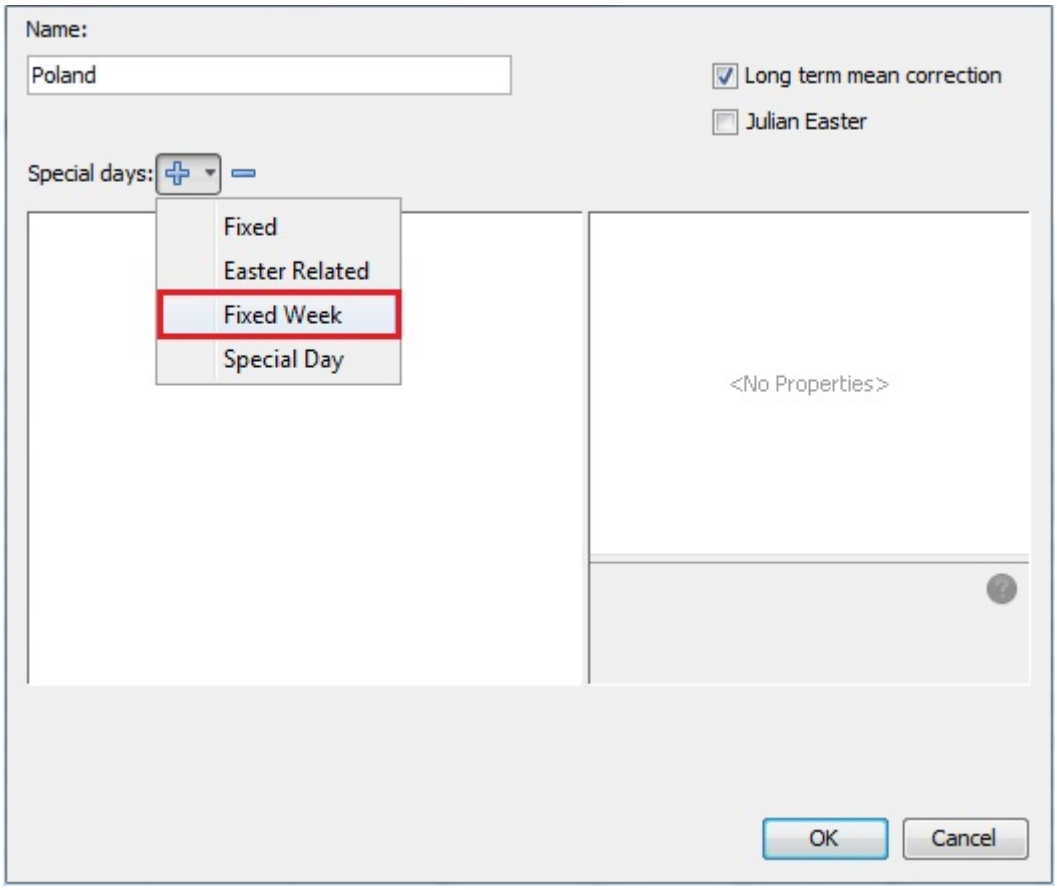


Figure 86: Text

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

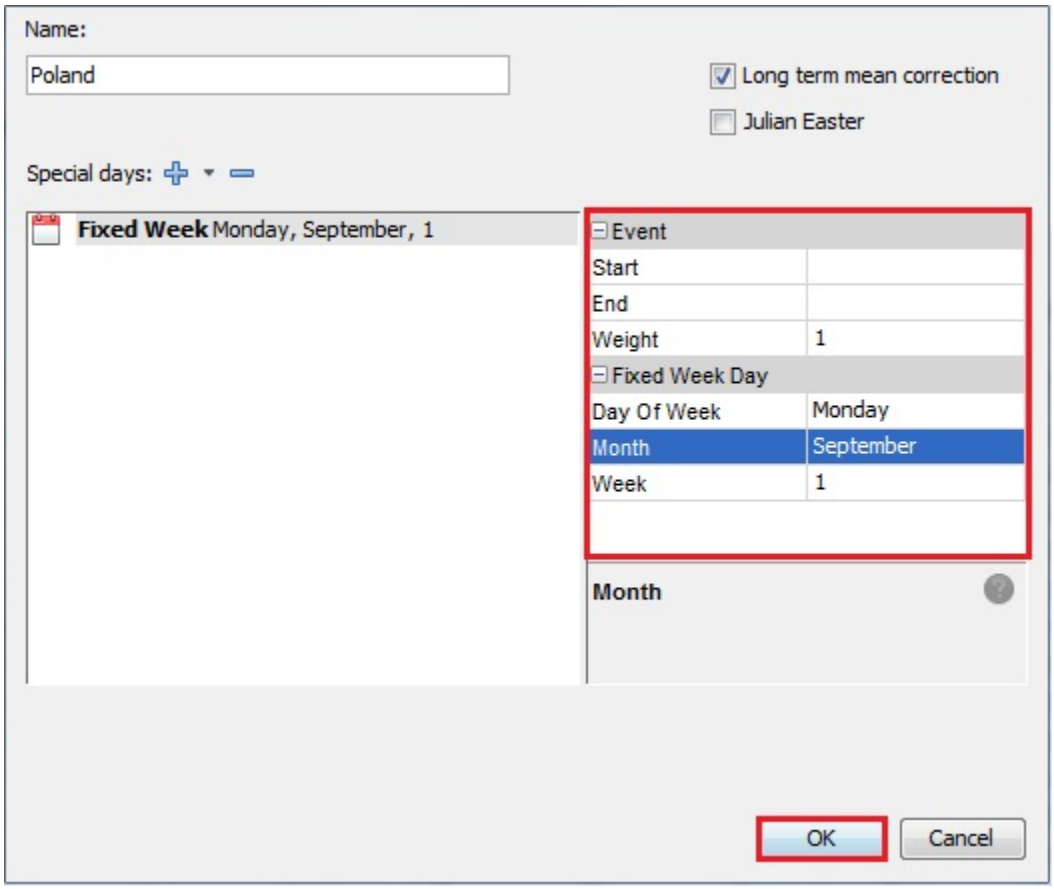


Figure 87: Text

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

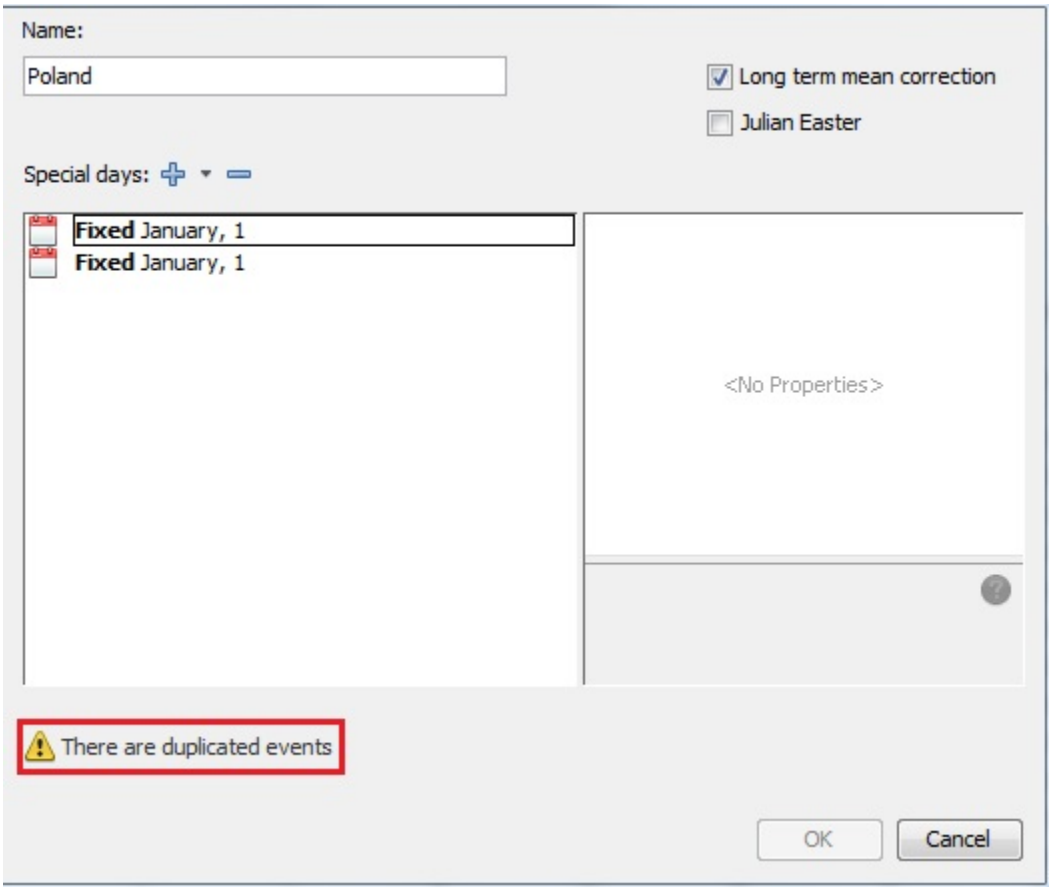


Figure 88: Text

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

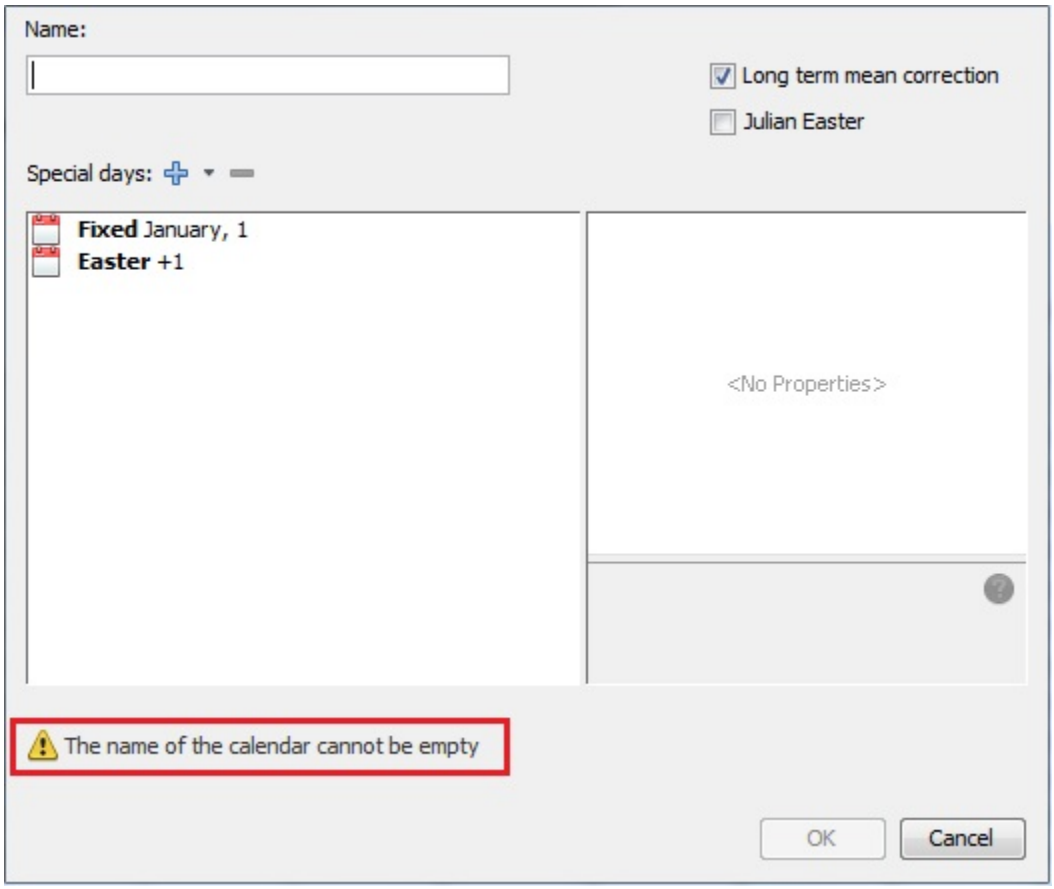


Figure 89: Text

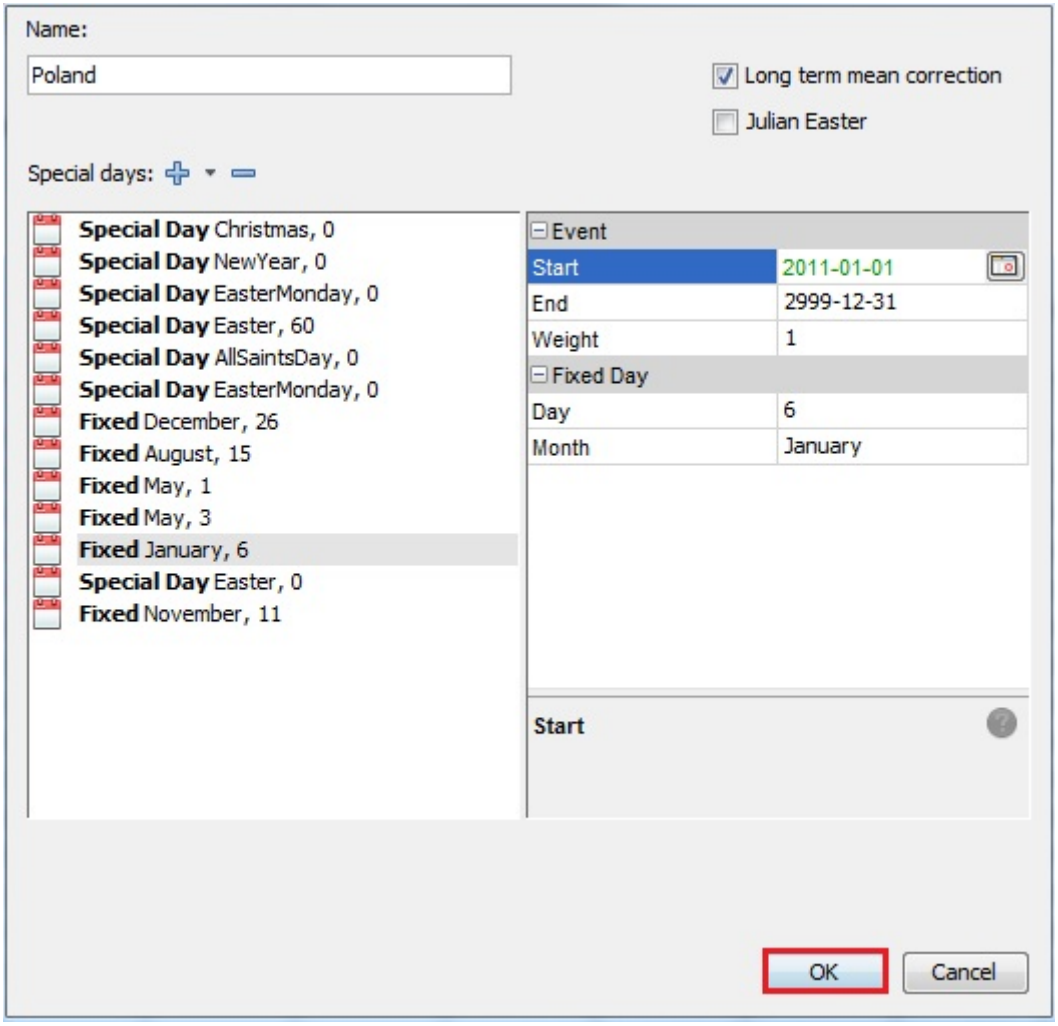


Figure 90: Text

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

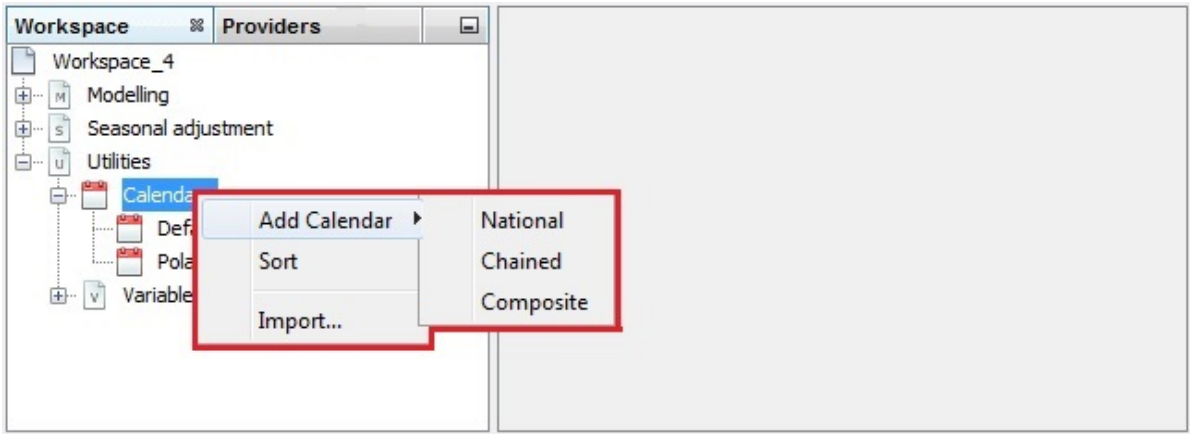


Figure 91: Text

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

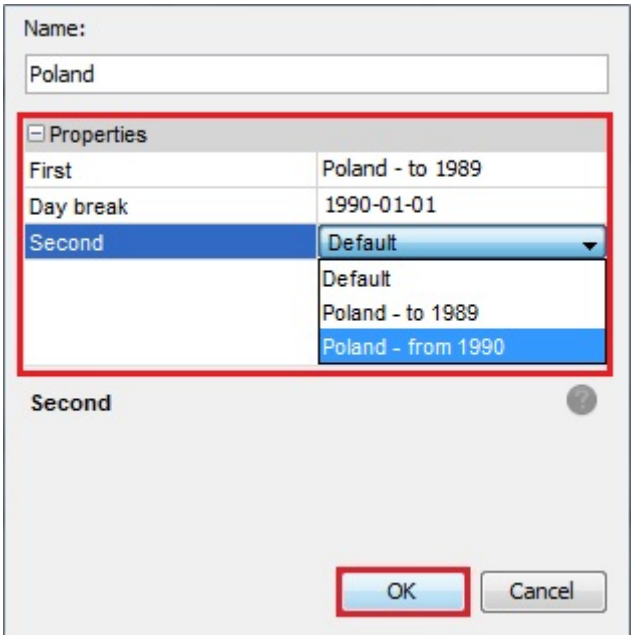


Figure 92: Text

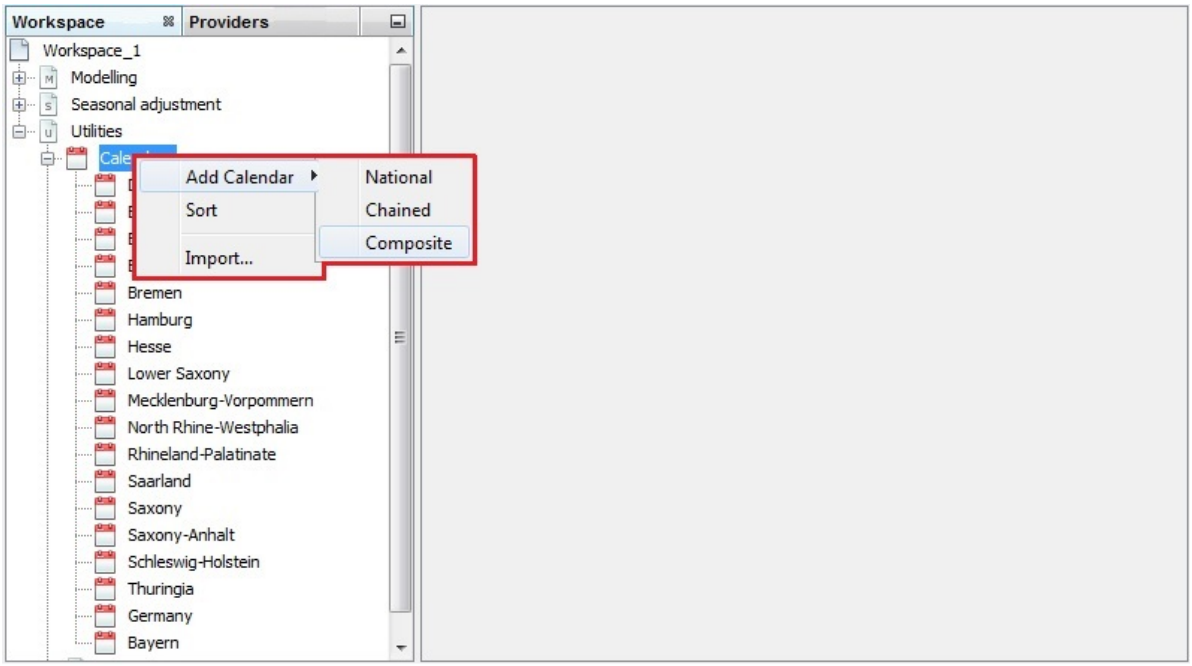


Figure 93: Text

- Assign a weight to each calendar.

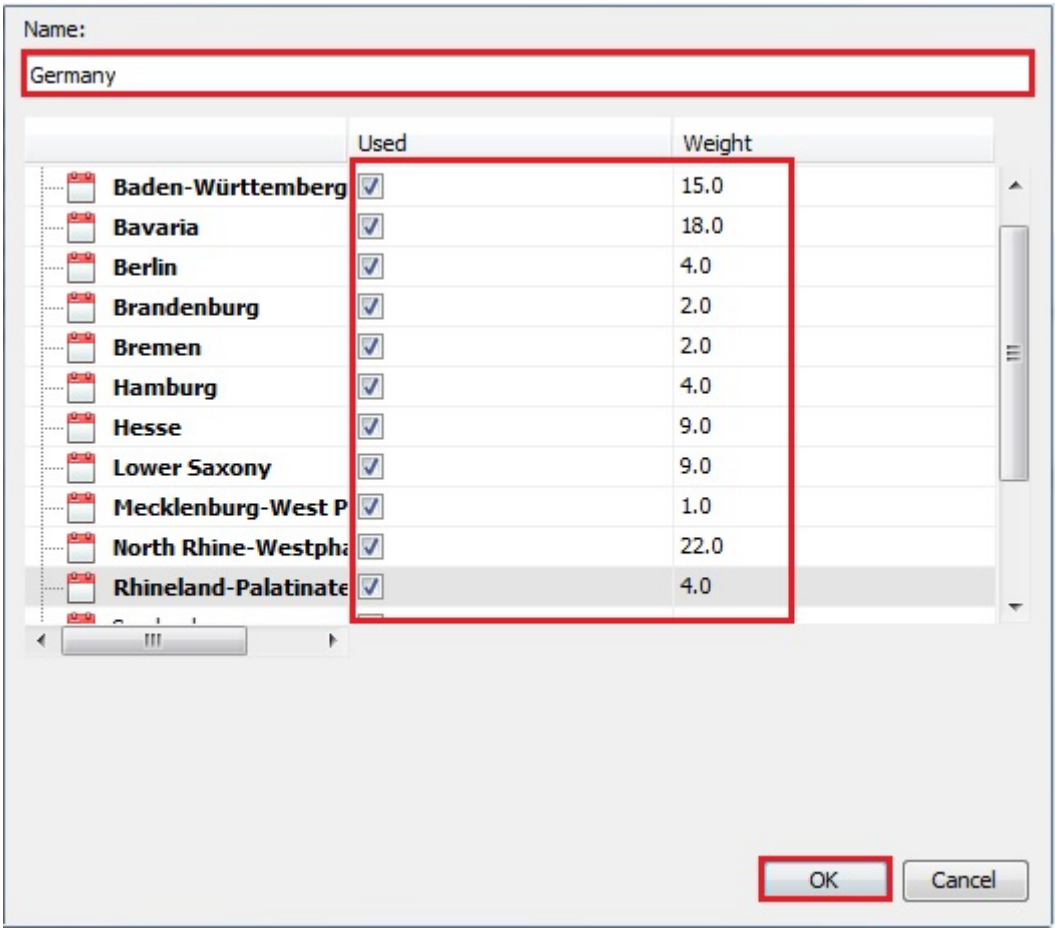


Figure 94: Text

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.

- choose the appropriate file and open it

JDemetra+ adds it to the calendars list

0.0.0.0.6 * Example of a calendar file

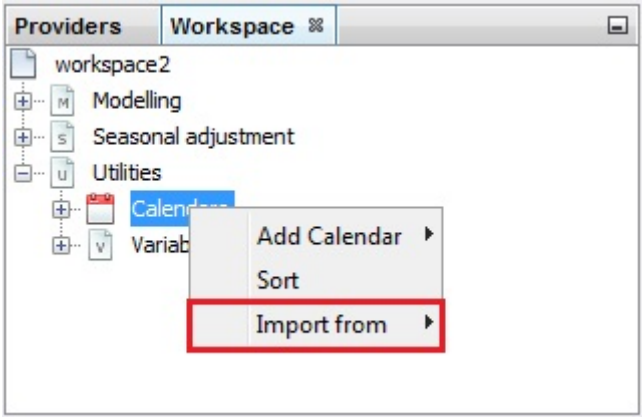


Figure 95: **Importing a calendar to JDemetra+**

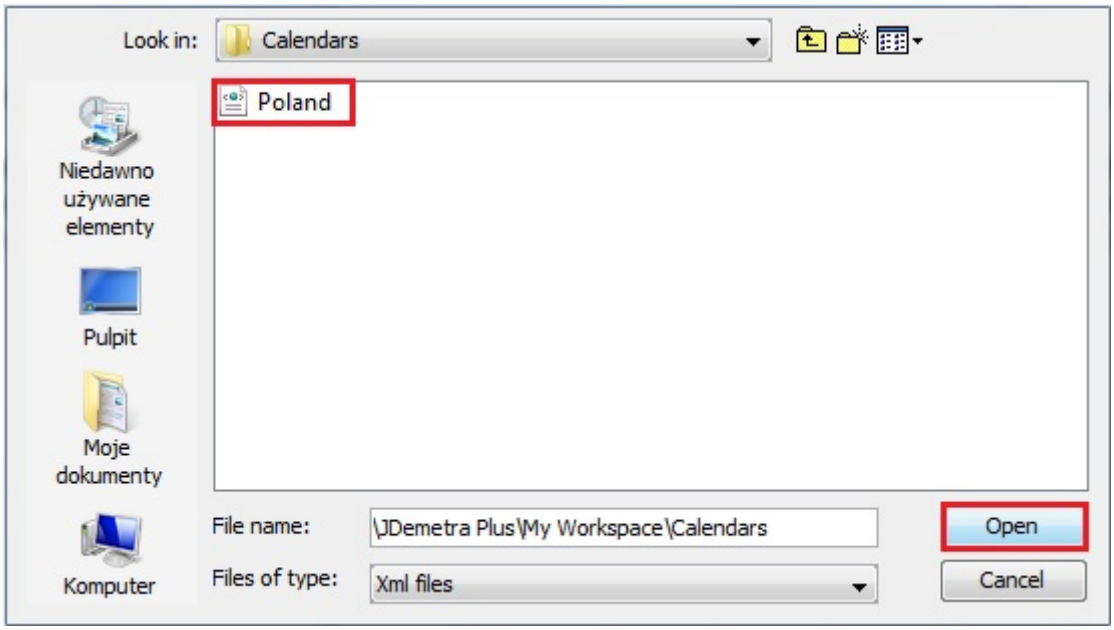


Figure 96: **Choosing the file**

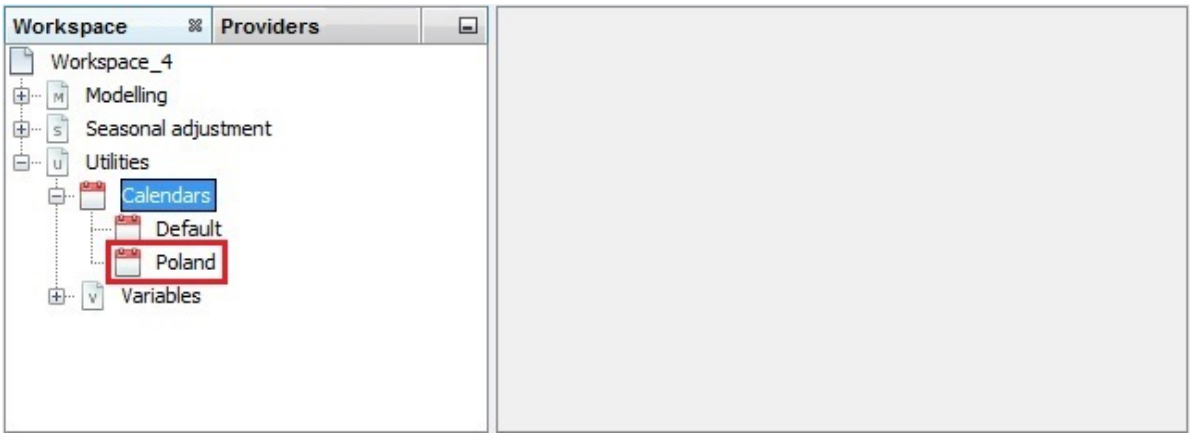


Figure 97: **A list of calendars with a newly imported calendar**

```
<?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 = 0.9,
    validity = list(start = "1968-02-01", end = "2010-01-01")
)
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( # Corpus Christi
        offset = 60,
        julian = FALSE,
        weight = 0.5,
        validity = list(start = "2000-01-01", end = "2020-12-01")
    ),
    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 =
```

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 =
```

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 disagreggation

In this chapter

The sections below provide guidance on how to implement algorithms on

- Benchmarking of seasonally adjusted data
- Benchmarking high to low frequency data
- Temporal Disaggregation

Using the GUI with a plug-in or rjd3bench package package.

Algorithms overview

Benchmarking

Method	GUI Plug-in for V 2.x	GUI Plug-in for V 3.x	In R rjd3bench
Denton	✔	✔	✔
Cholette	✔	✔	✔
Cholette Multi-variate	✔	✖	✔
Cubic Splines	✖	✔	✔
GRP (Growth Rate Preservation)	✖	✔	✔
Calendarization	✔	✖	✔

Temporal Disaggregation

Method	GUI Plug-in for V 2.x	GUI Plug-in for V 3.x	In R rjd3bench
Regression Models*	✔	✔	✔
Model-based Denton	✖	✔	✔
ADL (Autoregressive Distributed Lag Models)	✖	✖	✔

*Regression models: several structures of residuals

- Ar1: Chow-Lin
- Rw: Fernandez
- RwAr1: Litterman

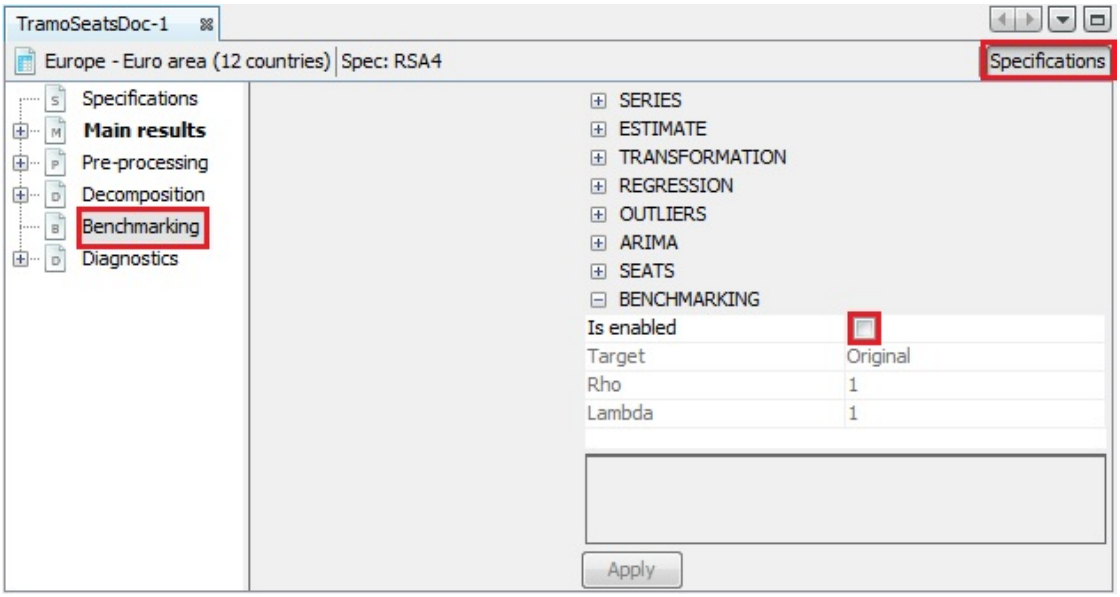
Benchmarking seasonally adjusted data

The goal here is to enforce identical annual totals on the seasonally adjusted series as on the raw or calendar adjusted series.

Using the GUI

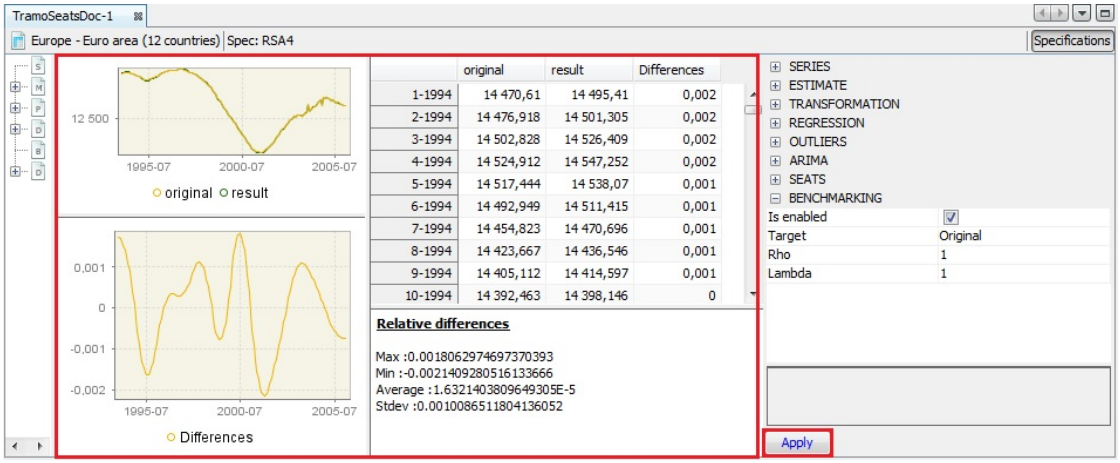
When running a seasonal adjustment process

1. With the pre-defined specifications the benchmarking functionality is not applied by default following the *ESS Guidelines on Seasonal Adjustment* (2024) recommendations. It means that once the user has seasonally adjusted the series with a pre-defined specification the *Benchmarking* node is empty. To execute benchmarking click on the *Specifications* button and activate the checkbox in the *Benchmarking* section.

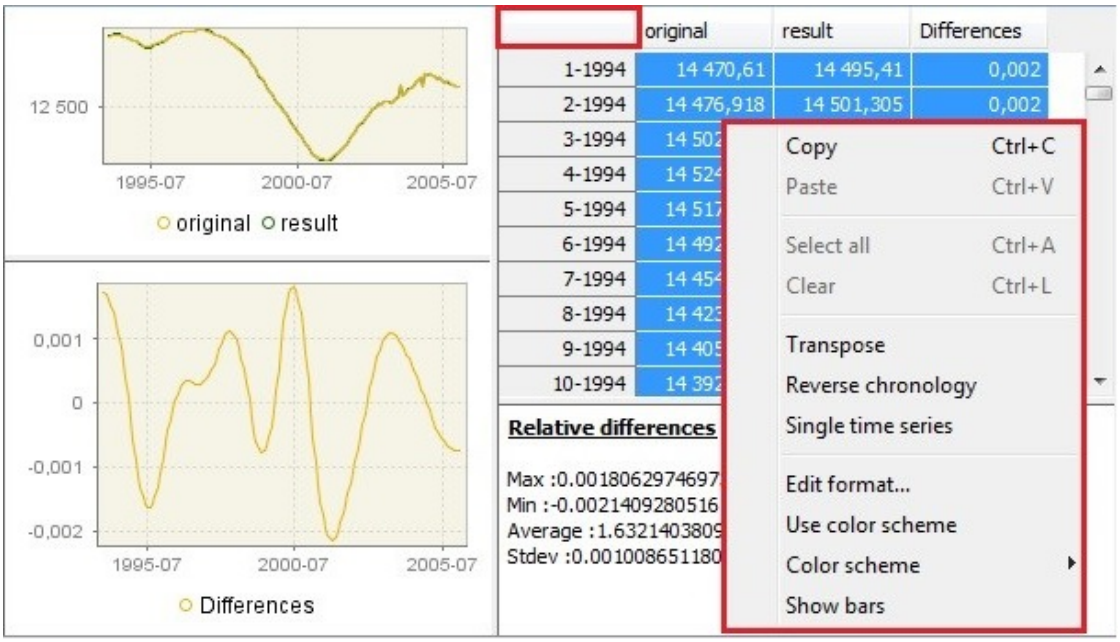


2. Three parameters can be set here. *Target* specifies the target variable for the benchmarking procedure. It can be either the *Original* (the raw time series) or the *Calendar Adjusted* (the time series adjusted for calendar effects). *Rho* is a value of the AR(1) parameter (set between 0 and 1). 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.

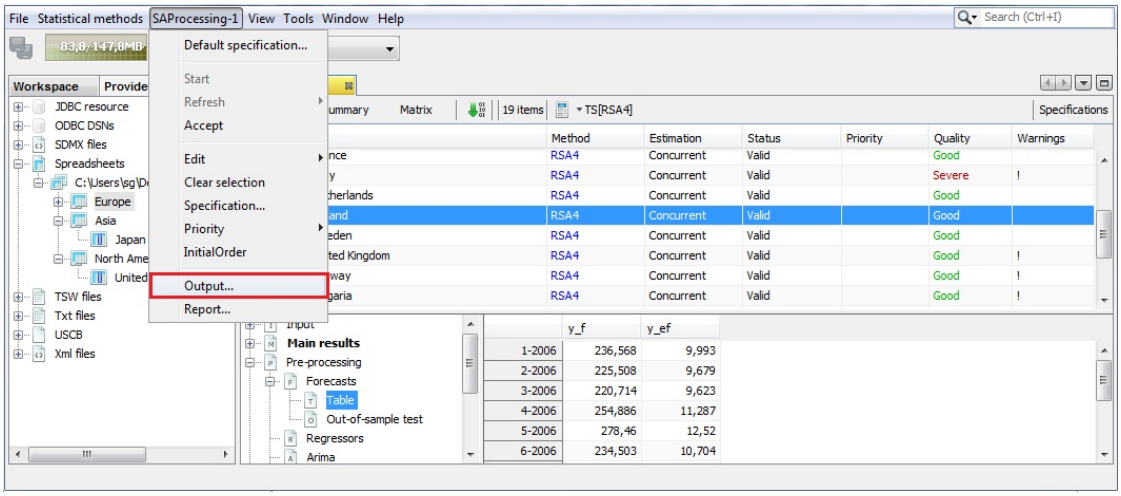
3. To launch the benchmarking procedure click on the **Apply** button. The results are displayed in four panels. The top-left one compares the original output from the seasonal adjustment procedure with the result from applying a benchmarking to the seasonal adjustment. The bottom-left panel highlights 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.



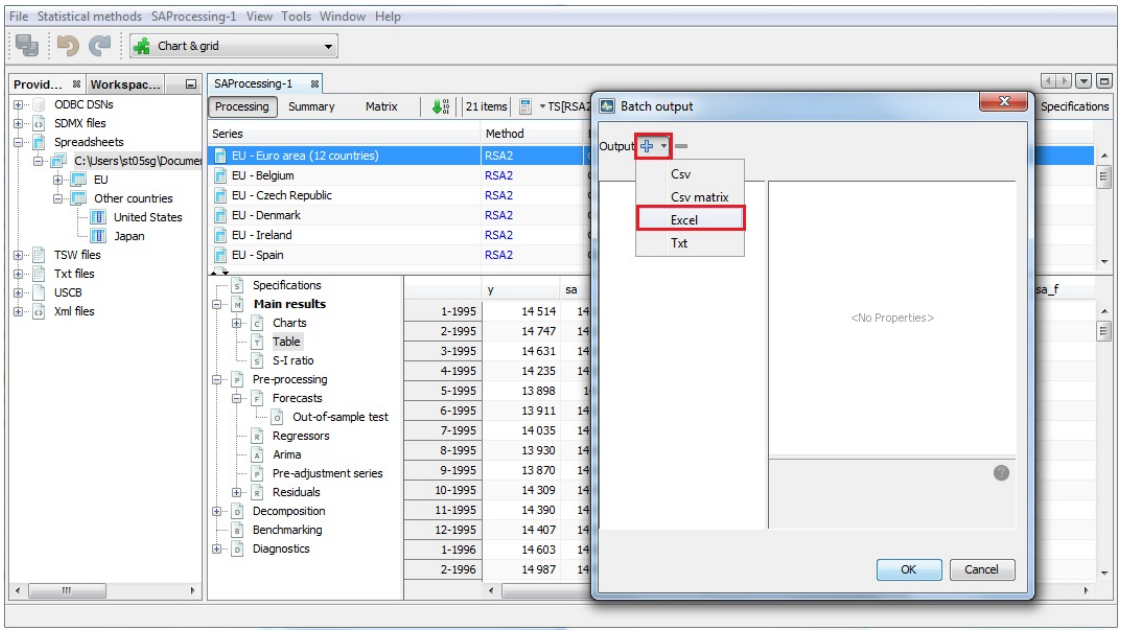
4. Both pictures and the table can be copied the usual way



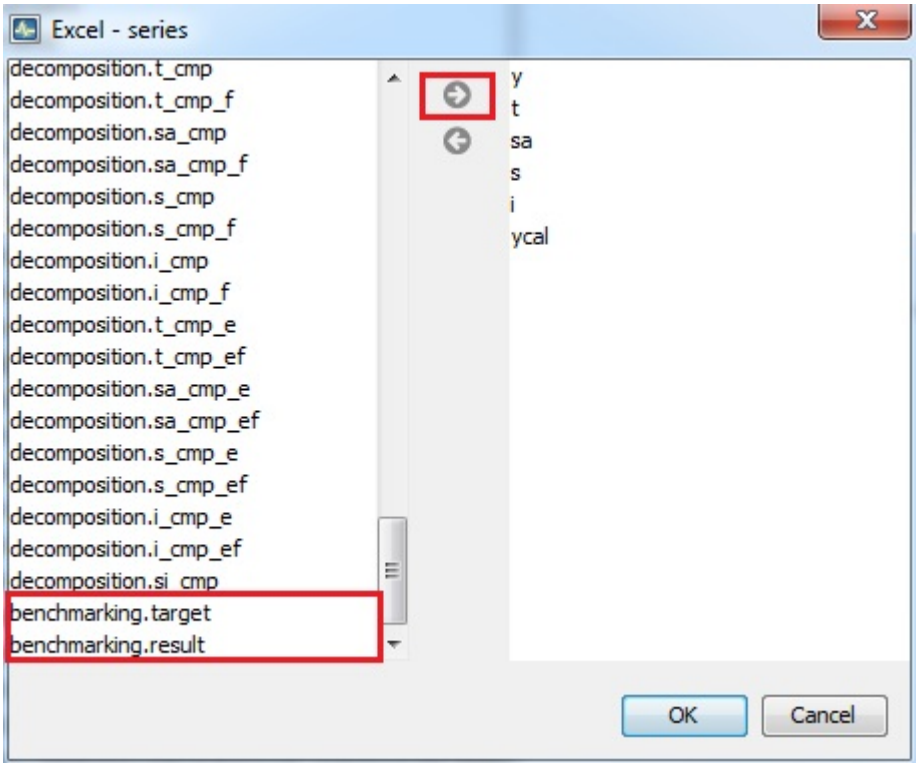
5. To export the result of the benchmarking procedure (*benchmarking.result*) and the target data (*benchmarking.target*) one needs to once execute the seasonal adjustment with benchmarking



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.



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**.



In R with `rjd3x13` and `rjd3tramoseats`

When performing seasonal adjustment with `rjd3x13` and `rjd3tramoseats`, the current (or default) specification has to be customized using the function `rjd3toolkit::set_benchmarking` documented on this GitHub page

```
init_spec <- rjd3x13::spec_x13("RSA5c")
new_spec <- set_benchmarking(
    init_spec,
    enabled = TRUE,
    target = "Normal",
    rho = 0.8,
    lambda = 0.5,
    forecast = FALSE,
    bias = "None"
)
```

More information on R packages for JDemetra+ and installation procedures is provided in this chapter

Benchmarking with different frequencies

These methods provide a high-frequency series (input series) modified so that it fulfils a linear relationship, with another series of low frequency (benchmark), both series measure the same target variable. An example of the relation to be fulfilled could be that the low frequency series (quarterly frequency) coincides with the quarterly sum of the high frequency series (monthly frequency).

Multivariate benchmarking also forces contemporary linear relations between high frequency series. If these relations do not exist, benchmarking could be carried out for each series separately. Normally contemporary relations are linear and the relations of aggregation are also linear and the same for all series, so the contemporary relations between low frequency series are fulfilled.

The benchmarking methods available in the benchmarking and time disaggregation plug-in are: Denton, Cholette, and Cholette multivariate.

Using the plug-in for GUI (version 2.x)

Download the plug-in for GUI as explained here and install it as detailed here

Once the plugin is installed, two more options appear in the Workspace window: Benchmarking and Temporal Disaggregation.

Figure 98: Text

Univariate: Denton and Cholette

To run Denton univariate case select:

Statistical **Methods** → Benchmarking → Denton or Cholette

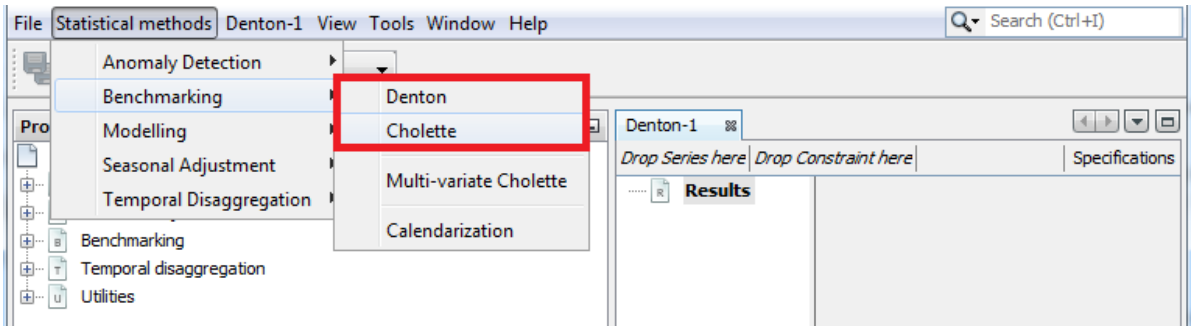


Figure 99: **Benchmarking tab**

In both cases, a new window is displayed to launch one of the methods with the series selected. In the upper left side, drag the high frequency series from the Providers window and drop it in **Drop Series here** and the low frequency series in **Drop Constraint here**.

Denton

In the top right of the screen, select the **Specifications** button to set the specifications to apply each method. See below for a description of the available options on Denton method:

1. **Type**: Aggregation function (Sum, Average, Last or First). This forces the low-frequency series to match the aggregation function selected of the high frequency series.
2. **Multiplicative**: if the checkbox is selected, the proportional Denton method is applied. Otherwise, additive Denton is applied.
3. **Modified Denton**: if the checkbox is selected, the modified Denton method is applied. Otherwise, original Denton is applied. It is recommended to select it; as original Denton perform a special treatment on the first observation.
4. **Differencing**: Number of regular differences. By default 1.
5. **Default frequency**: periodicity of the low frequency data. The options are: Yearly, HalfYearly, QuadriMonthly, Quarterly, Bimonthly and Monthly.

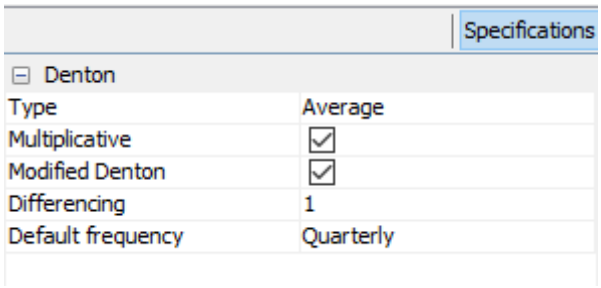


Figure 100: Denton Specifications

Cholette

See below for a description of the available options on Cholette method:

1. **Type**: Aggregation function (Sum, Average, Last or First). This forces the low-frequency series to match the aggregation function selected of the high frequency series.
2. **Aggregation frequency**: periodicity of the low frequency data. The options are: Yearly, HalfYearly, QuadriMonthly, Quarterly, Bimonthly and Monthly.
3. **Rho**: value between -1 and 1. It is the coefficient of an AR(1) model that follows the error term. The default value is 1, equivalent to applying Denton.
4. **Lambda**: value between 0 and 1. It is the parameter λ of the following function to be minimized in Cholette method:

$$\sum_t \left(\frac{x_t - z_t}{|z_t|^\lambda} - \rho \frac{x_{t-1} - z_{t-1}}{|z_{t-1}|^\lambda} \right)^2$$

   Usually lambda is 0 or 1 equivalent to applying additive benchmarking and proportional benchmarking method respectively.

In both cases, Denton and Cholette methods, the output is a graph with the original series and the benchmarked series. There is no table with the results, but it is very easy to create one from the graph. Select the graph and select copy, then paste the values in excel (control-V).

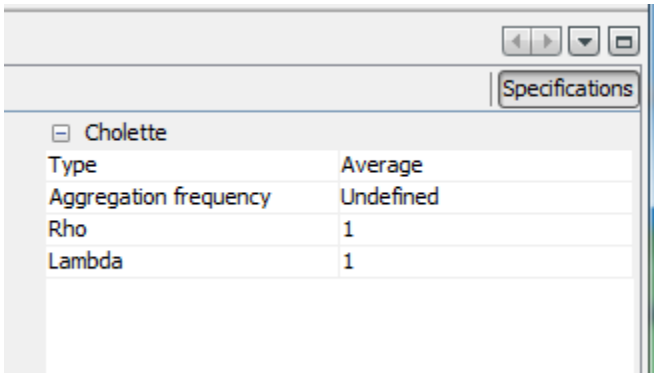


Figure 101: Cholette Specifications

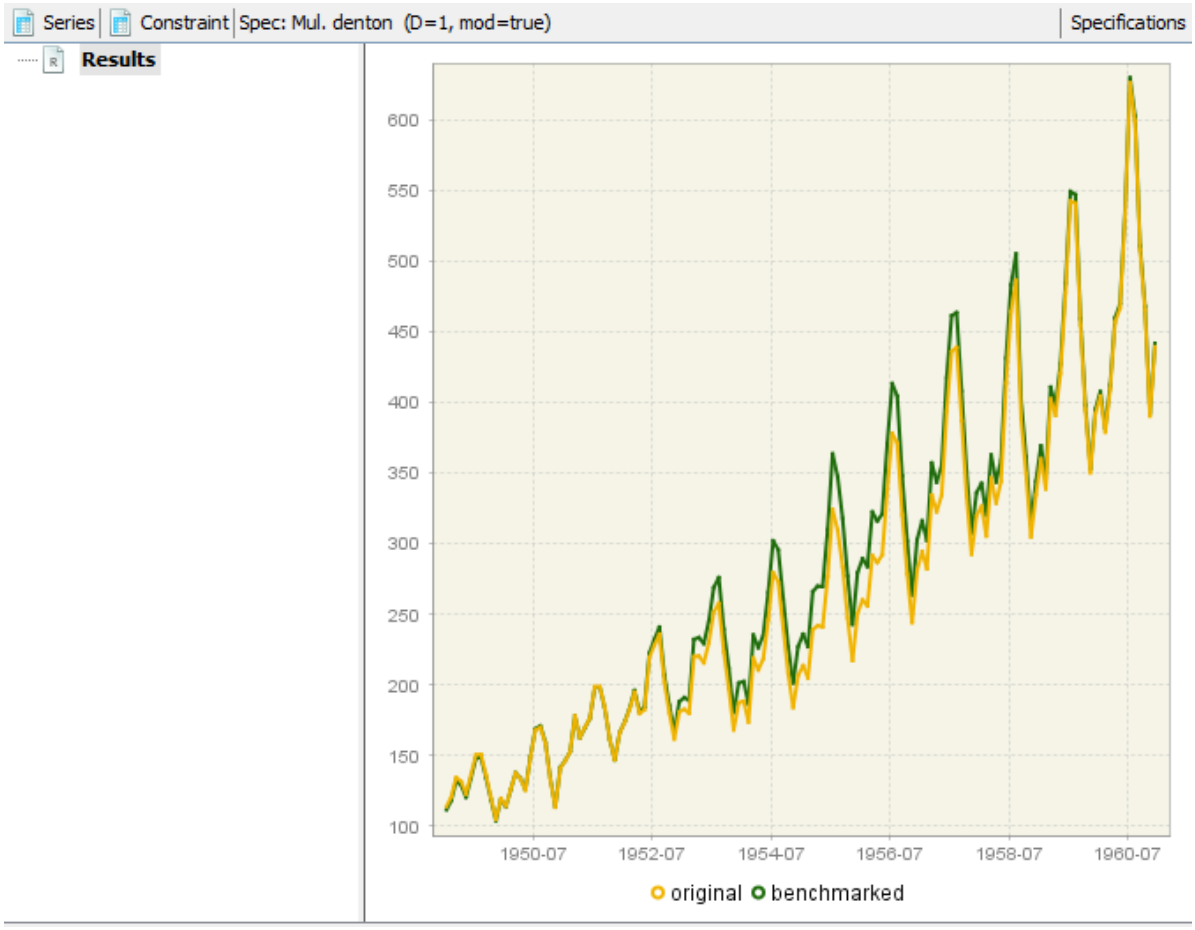


Figure 102: Denton output

Multi-variate Cholette

The only multi-variate benchmarking method available for the version 2 plugin, is multi-variate Cholette.

The input for this method are a set of time series with different frequencies and a set of constraints, both contemporary and intertemporal. The output is a new set of time series, that corresponds to the former, now fulfilling the constraints.

To run multi-variate Cholette select *Statistical methods* → *Benchmarking* → *Multi-variate Cholette*. Then, a box appears, where we can drop time series.

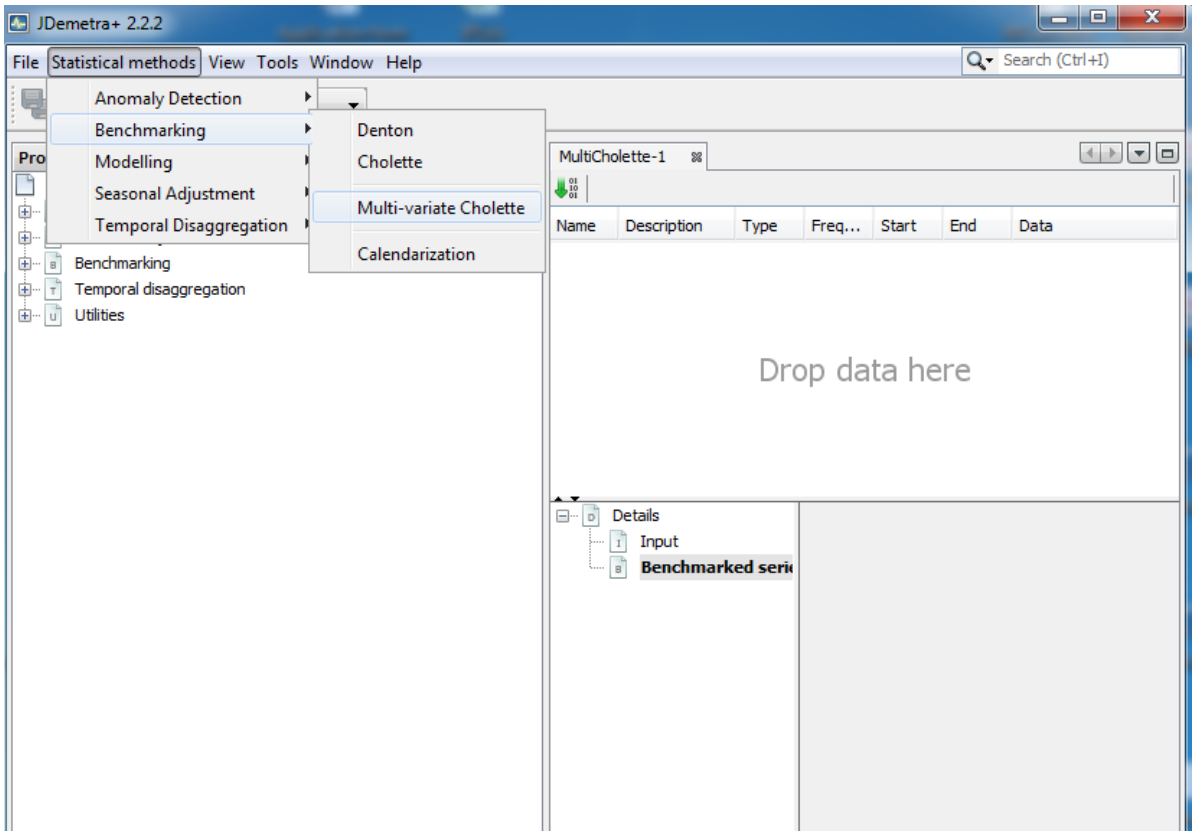


Figure 103: Multi-variate Cholette input

The specification properties can be set by selecting *Window* → *Properties*. There are only three elements in the form: *Rho*, *Lambda* and *Constraints*. The two first are analogous to their univariate Cholette counterparts. The third one allows to set constraints, both contemporary and intertemporal.

Clicking the three dots button from *Constraints*, opens the constraints list box.

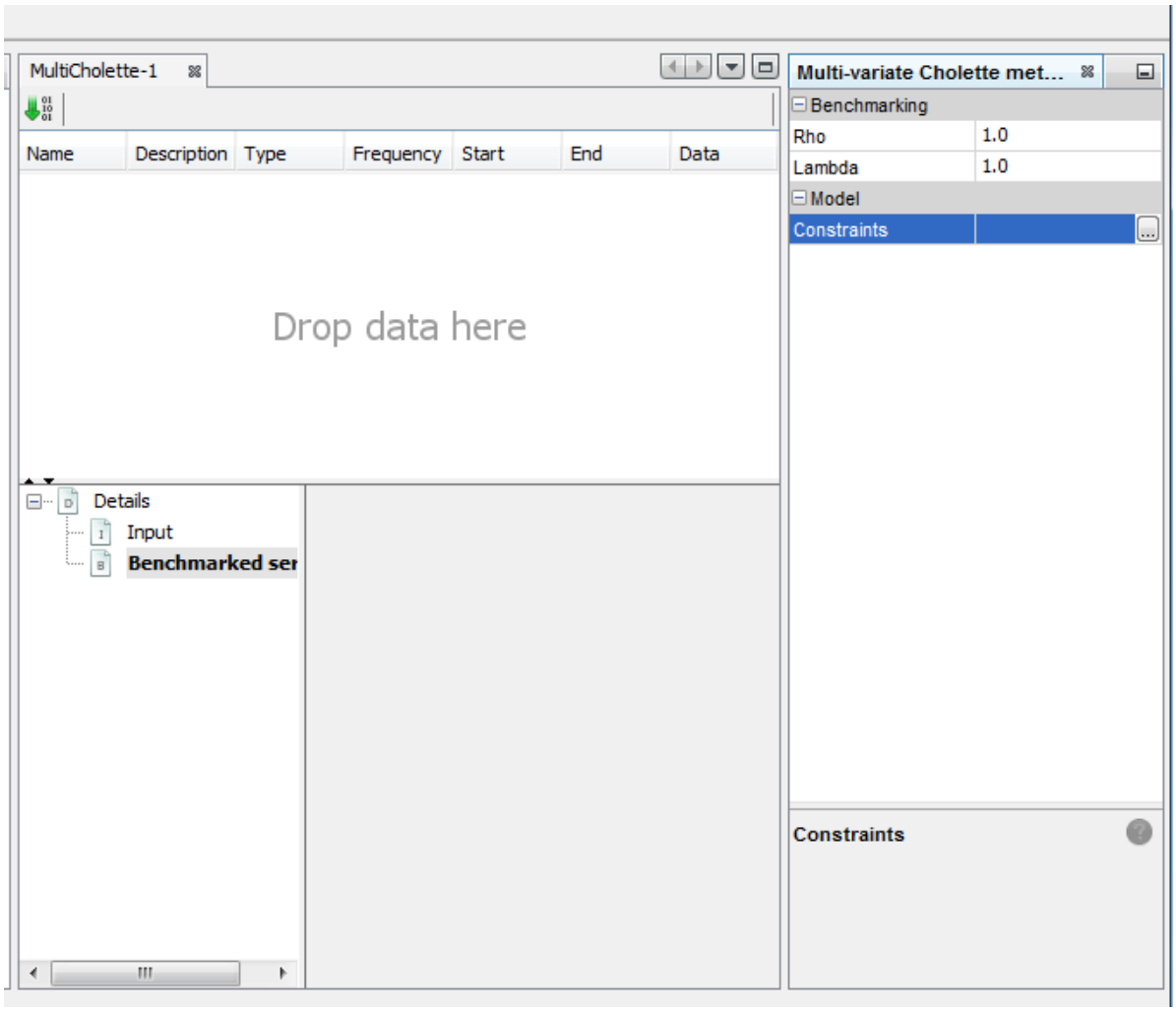


Figure 104: Multi-variate Cholette specification

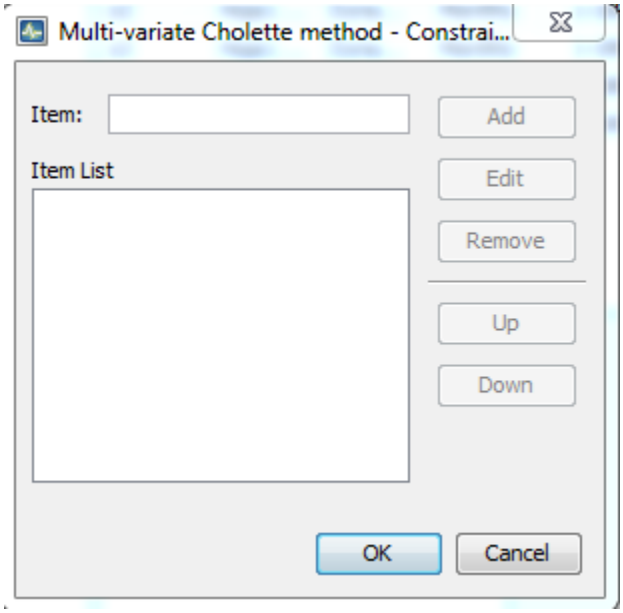


Figure 105: Constraints list

From this box, the set of constraints can be added, just typing the constraint inside item and clicking *Add* button. The constraints must be written as follows:

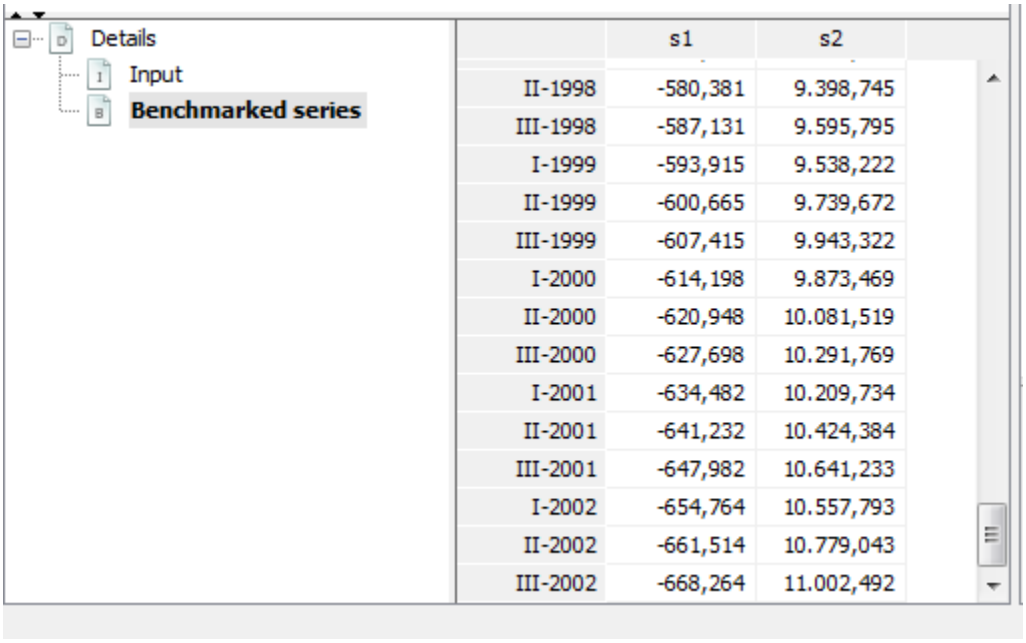
- $y = a_1 * x_1 + \cdots + a_n * x_n$ where $y, x_1, \ldots, x_n$ are same frequency time series and $a_1, \ldots, a_n$ are constant.
- $c = a_1 * x_1 + \cdots + a_n * x_n$ where $x_1, \ldots, x_n$ are same frequency time series and $c, a_1, \ldots, a_n$ are constant.
- $c = x_1 + \cdots + x_n$ can be shortened as $c = x?$.
- $S = \text{sum}(s)$ where S is low-frequency and s is high-frequency.

Note that any time series put on the left hand side can't appear on the right hand side of any other constraint. This is because left hand side quantities are fixed while right hand side quantities are adjusted so the equality holds.

The output are the benchmarked high-frequency time series, that can be found in *Details* $\rightarrow$ *Benchmarked series*.

Using the plug-in for GUI (version 3.x)

Practical use of the plug-in for v 3.x is quasi identical to the one in v2.x described above. Some methods are not available yet in v 3.x but the latter which contains Model



	s1	s2
II-1998	-580,381	9.398,745
III-1998	-587,131	9.595,795
I-1999	-593,915	9.538,222
II-1999	-600,665	9.739,672
III-1999	-607,415	9.943,322
I-2000	-614,198	9.873,469
II-2000	-620,948	10.081,519
III-2000	-627,698	10.291,769
I-2001	-634,482	10.209,734
II-2001	-641,232	10.424,384
III-2001	-647,982	10.641,233
I-2002	-654,764	10.557,793
II-2002	-661,514	10.779,043
III-2002	-668,264	11.002,492

Figure 106: Output multi-variate Cholette

Based Denton not included in v2.x, as stated here

In R with `rjd3bench`

Use the [rjd3bench](https://github.com/rjdverse/rjd3bench) package and see its documentation pages. Browse its documentation on this GitHub page.

To get started browse the vignette

More information on R packages for JDemetra+ and installation procedures is provided in this chapter

Temporal Disaggregation

These methods are used to disaggregate a series from low frequency to high frequency. Temporal disaggregation methods developed in the plug-in are Chow-Lin, Fernández and Litterman.

When there are high frequency related indicators, these methods provide high frequency estimations for a series whose sums, averages, first or last values are

consistent with the observed low frequency series, applying a regression model where it is assumed that the high frequency series to be estimated follows a multiple regression with p related series (indicators).

See Methods→Temporal disaggregation for more theoretical detail.

Using the plug-in for GUI

Temporal disaggregation in the GUI is available with the same plug-in as benchmarking (described in the sections above)

To run Temporal Disaggregation methods select Temporal disaggregation→ Regression Model:

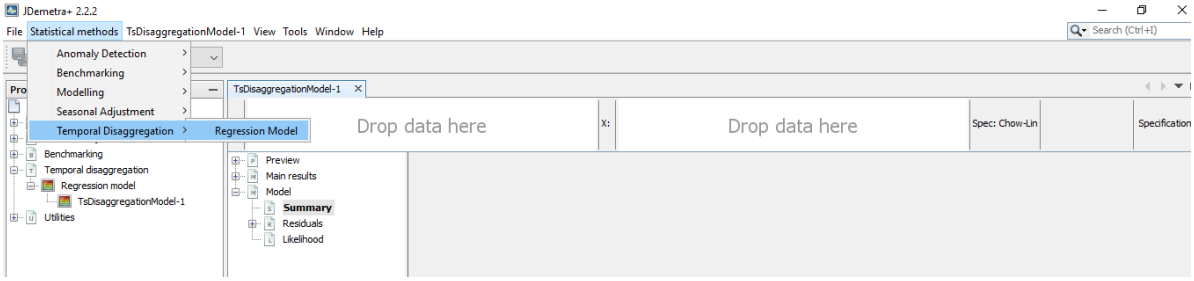


Figure 107: Temporal Disggregation

A new window is displayed to launch one of the methods with the series selected. In the upper left side drag the low frequency series from the Providers window and drop it in **Y box** and the proxy series or indicator with high frequency series in **X box**.

In the top right of the screen, select **Specifications** to set the specifications to apply each method. Here is a description of the available options on Temporal Disaggregation methods:

1. **Estimation span**: Specifies the span (data interval) of the time series to be used in the temporal disaggregation process. The user can restrict the span. The common settings are:

Option	Description (expected format)
All	default
From	first observation included (yyyy-mm-dd)
To	last observation included (yyyy-mm-dd)

Option	Description (expected format)
Between	interval [from ; to] included (yyyy-mm-dd to yyyy-mm-dd)
First	number of observtions from the beginning of the series included (dynamic) (integer)
Last	number of observations from the end of the series (dynamic)(integer)
Excluding	excluding N first observation and P last observation 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)

2. **Error**: determines the method to be applied and it refers to the model that follows the error term.

Option	Description
Ar1	Chow-Lin method (default)
Wn	Classical Regression model
Rw	Fernández
RwAr1	Litterman
I2	Integrated order 2
I3	Integrated order 3

3. **Parameter**: Coefficient of the AR(1) of the innovations model. It has a value between -1 and 1. This parameter exists only if RWar1 or Ar1 is selected in the error parameter.
4. **Constant**: a constant is included in the model if it is selected.
5. **Trend**: a linear trend is included in the model if it is selected.
6. **Type**: Aggregation function (Sum, Average, Last or First). This forces the low-frequency series to match the aggregation function selected of the high frequency series.
7. **Default frequency**: it is the frequency of the output series.
8. **Advanced options**: These parameters are related to state space model and the algorithm used to obtain the estimations.

   8.1. **Diffuse regression coefficient**: Indicates if the coefficients of the regression model are diffuse (T) or fixed unknown (F, default).

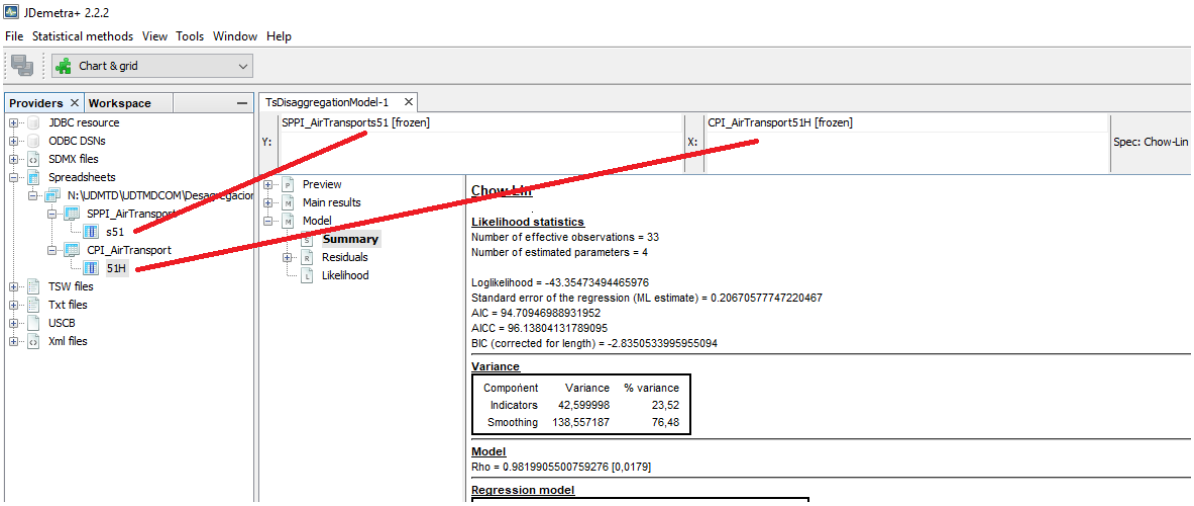


Figure 108: Temporal Disggregation

Here are the results:

Select **Model**→Summary to see the estimation of rho (coefficient of the AR(1) model) and the coefficient of the regression model. Additionally the BIC, AIC and AICC. It is also showed the variance decomposition in Indicators and Smoothing. Ideally, if the indicator adequately approximates the aggregate in the observable domain (low frequency model), the residuals of the low frequency model will be small and the indicator term will dominate. \

To confirm that the model works well, select **Model**→Residuals→Statistics and see the tests on the residuals of the model:

Select MainResults→Table to obtain the disaggregated series and standard deviation.

Select **MainResults**→Chart to see a graph of the disaggregated series and the confidence interval.

In R with `rjd3bench`

Use the [rjd3bench](https://github.com/rjdverse/rjd3bench) package and see its documentation pages. Browse its documentation on this GitHub page.

To get started browse the vignette

More information on R packages for JDemetra+ and installation procedures is provided in this chapter

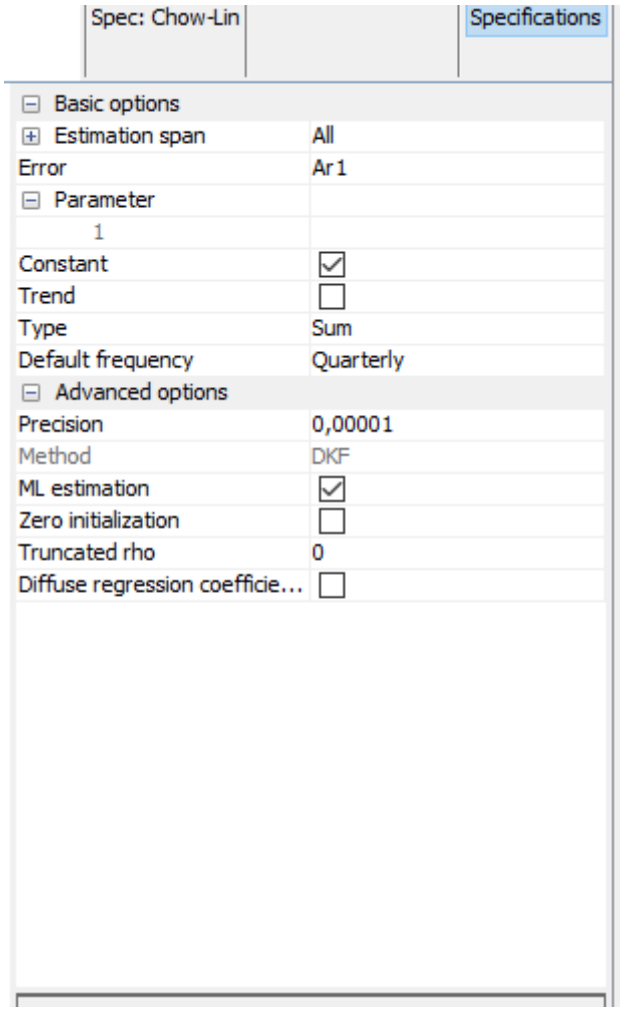


Figure 109: Temporal Disggregation

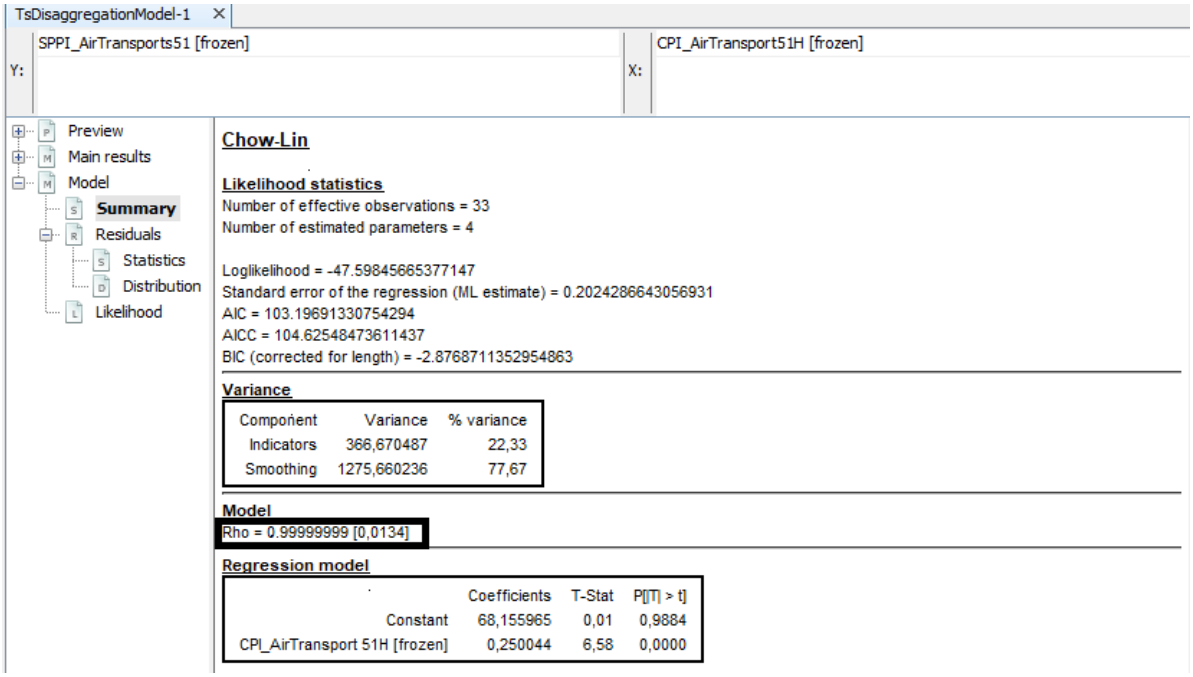


Figure 110: Temporal Disaggregation

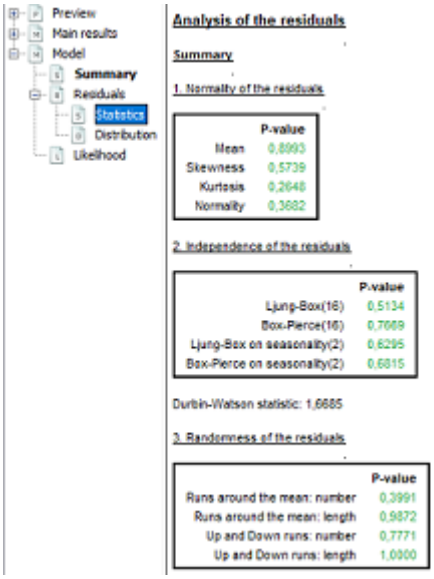
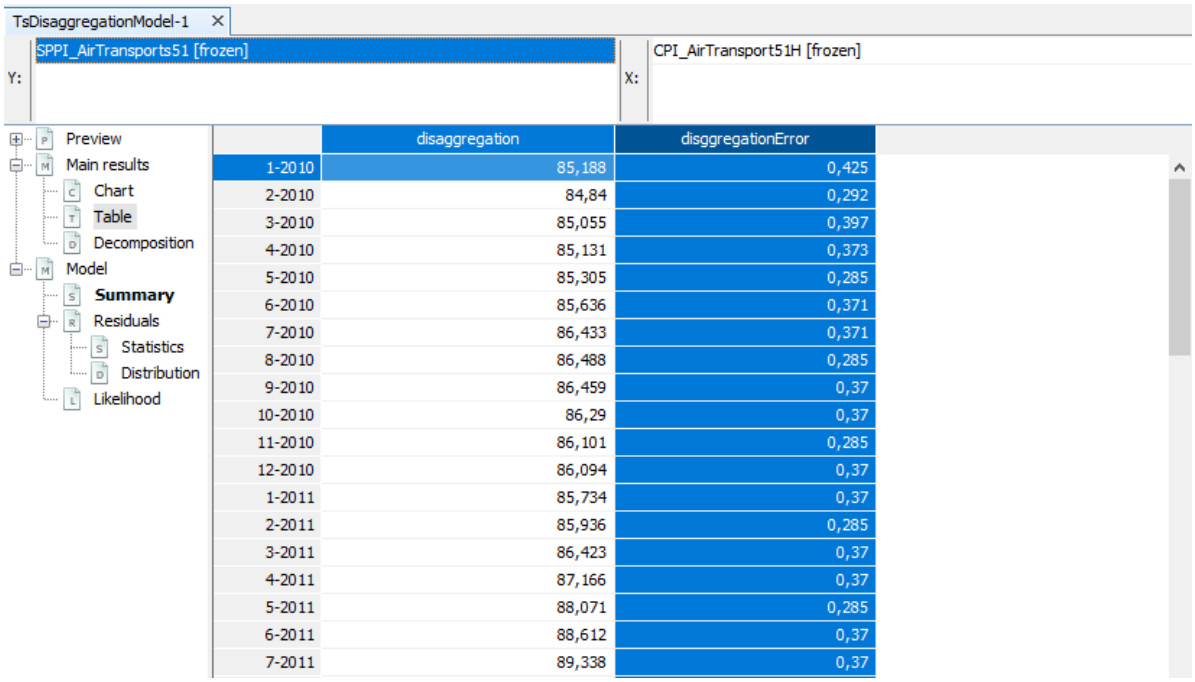


Figure 111: Temporal Disggregation



	disaggregation	disggregationError
1-2010	85,188	0,425
2-2010	84,84	0,292
3-2010	85,055	0,397
4-2010	85,131	0,373
5-2010	85,305	0,285
6-2010	85,636	0,371
7-2010	86,433	0,371
8-2010	86,488	0,285
9-2010	86,459	0,37
10-2010	86,29	0,37
11-2010	86,101	0,285
12-2010	86,094	0,37
1-2011	85,734	0,37
2-2011	85,936	0,285
3-2011	86,423	0,37
4-2011	87,166	0,37
5-2011	88,071	0,285
6-2011	88,612	0,37
7-2011	89,338	0,37

Figure 112: Temporal Disggregation

Temporal Disaggregation

To perform Temporal Disaggregation methods use the function **temporaldisaggregation**:

```
output <- rjd3bench::temporaldisaggregation(
    series = y, indicators = x, model = "Rw", freq = 12,
    conversion = "Average", diffuse.algorithm = "Diffuse"
)
```

The input parameters are the same as in the GUI, see the R Documentation of the rjd3bench package for the description.

The output is a list containing 3 elements:

1. **Regression**: contains information about:
    - Type of method applied:

```
output$regression$type
```

    - The model (coefficient estimation, standard deviation and T-statistic):

```
output$regression$model
```

- Conversion:Aggregation function (Sum, Average, Last or First):

```
output$regression$conversion
```

2. **Estimation**: contains information about:
   - The disaggregated series:

```
output$estimation$disagg
```

   - The standard deviation of the disaggregated series:

```
output$estimation$edisagg
```

   - The regressor effect:

```
output$estimation$regeffect
```

   - The smoothing part:

```
output$estimation$smoothingpart
```

   - The rho estimation (coefficient of the AR(1) model): This parameter exists only if RWar1 or Ar1 is selected in the model.

```
output$estimation$parameter
```

   - The standard deviation of the AR(1) coefficient:

```
output$estimation$eparameter
```

3. **Likelihood**: Contains information about the loglikelihood(ll), sum of squares of the residuals of the model (ssq), number of observations (nobs), number of parameters to be estimated (nparams), degrees of freedom (df), Akaike Information Criteria (aic), Akaike Information Criteria Corrected (aicc), Bayesian Information Criteria (bic), Bayesian Information Criteria Corrected (bic2).

Trend-cycle estimation

In this Chapter

This chapter will cover the implementation of trend estimation methods available in JDemetra+ v 3.

More methodological details will be provided here

Tools for access

For the time being this algorithms are available in two R packages.

rjd3filters

rjd3filters is available here, with useful information in the Readme file and function documentation.

rjd3x11plus

rjd3x11plus is available here, with (up coming) useful information in the Readme file and function documentation.

(Up coming content)

Revision Analysis

Tool

rjd3revisions package allows to perform revision analysis.

More information is available directly in the package documentation and vignette.

```
library("rjd3revisions")
browseVignettes("rjd3revisions")
```

Up coming content.

Nowcasting

Nowcasting is often defined as the prediction of the present, the very near future and the very recent past. The plug-in developed at the National Bank of Belgium helps to operationalize the process of nowcasting. It can be used to specify and estimate dynamic factor models and visualize how the real-time dataflow updates expectations, as for instance in Banbura and Modugno (2010). The software can also be used to perform pseudo out-of-sample forecasting evaluations that consider the calendar of data releases, contributing to the formalization of the nowcasting problem originally proposed by Giannone, et al. (2008) or Evans (2005).

In the meantime the user can refer to the documentation provided with the nowcasting plug-in for version 2.2 and later.

Part II

Tools

This part describes the tools allowing to access JDemetra+ algorithms: Graphical User Interface (GUI) extended with Plug-ins, Cruncher and R packages.

Practical guidance on algorithms per se is provided here and methodological details on each algorithm can be found in the here.

In this part:

Graphical User Interface (GUI)

- Overview
- Data visualization and time series tools
- Seasonal Adjustement and Modelling features
- Output: series, parameters and diagnostics

GUI extensions

- Plug-ins for GUI

Production Module

- Cruncher and quality report

R ecosystem:

- R packages

Graphical User Interface (GUI): Overview

In this chapter

This chapter provides general information about using the Graphical User Interface (GUI). Specific indications related to a given algorithm (X-13-ARIMA, Tramo-Seats, Benchmarking...) are displayed in the relevant chapters, listed here.

Contents:

- Available algorithms
- Installation and launch
- Importing data
- General window and menu structure

Additional chapters related to GUI features, provide information on:

- Data visualization and generic time series tools
- Specific Seasonal Adjustement and Modelling features
- Output: series, parameters and diagnostics

Available algorithms

0.0.0.1 v2

The Graphical User Interface in the 2.x family gives access to:

- Seasonal adjustment (SA) algorithms
  - X-13-ARIMA
  - Tramo-Seats
  - Direct-indirect SA comparisons
- Outlier detection (TERROR)
- Benchmarking

0.0.0.2 v3

The Graphical User Interface in the 3.x family gives access **in addition** to extended SA algorithms for high-frequency data (HF).

The Graphical User Interface in the 2.x family gives access to:

- Seasonal adjustment (SA) algorithms
  - X-13-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

The installation procedure is detailed here in the introductory chapter of this book. Should you need more configuration details you will find specific Sheets in the JD-Tutorials GitHub repository

Launching 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).

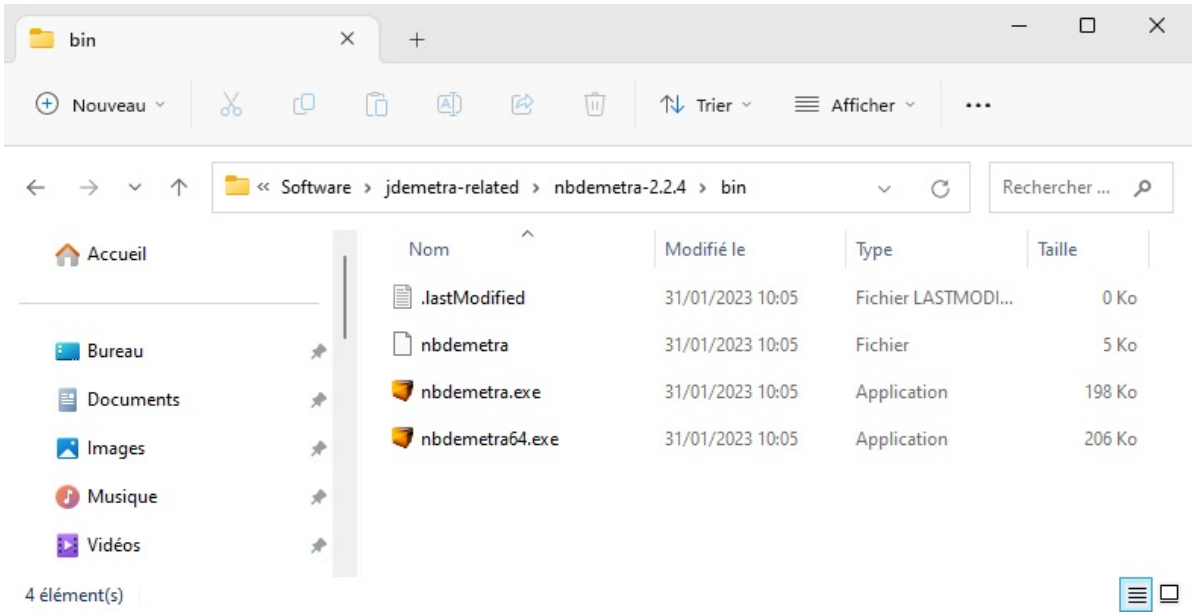


Figure 113: **Launching JDemetra+**

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.

Starting Window

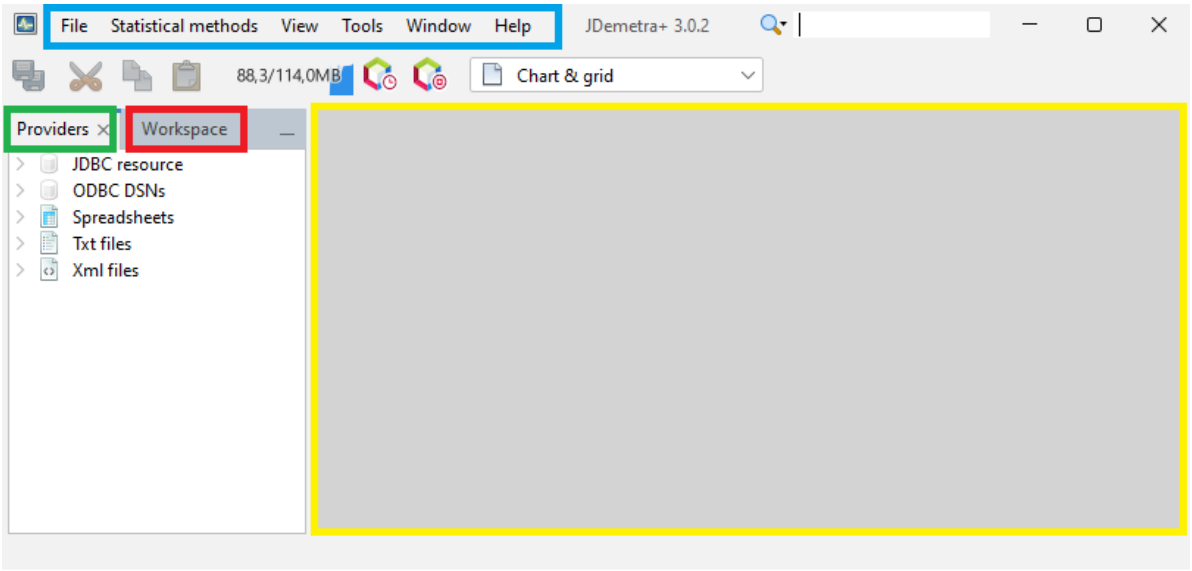


Figure 114: **JDemetra+ default window**

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;
- 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

By default, JDemetra+ supports the following data sources:

- JDBC;
- ODBC;
- SDMX;
- Excel spreadsheets;

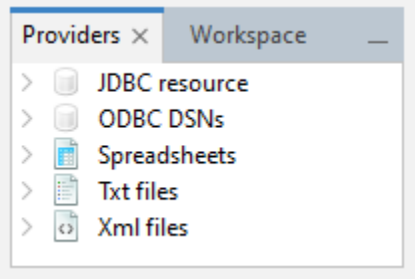


Figure 115: **The *Providers* window**

- 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).

Import data

To import data from a given data source:

- click on this data source in the *Providers* window shown below
- choose *Open* option and specify the import details, such as a path to a data file.

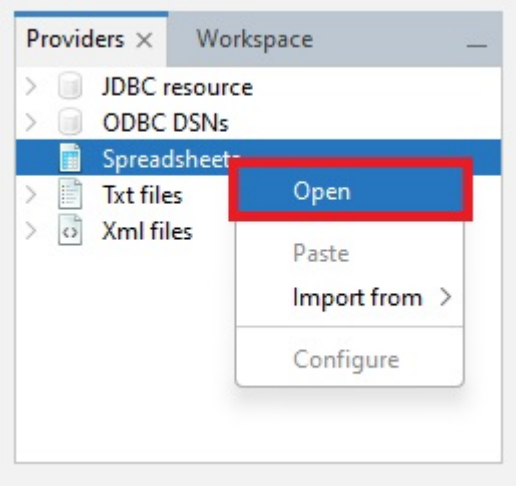
These details vary according to data providers.

https://www.youtube.com/watch?v=KYBcKx1e8ys&pp=ygUUaW1wb3J0IGRhdGEgamRlbWV0cmE%3D

0.0.0.1 Spreadsheet

The example below show how to import the data from an Excel file.

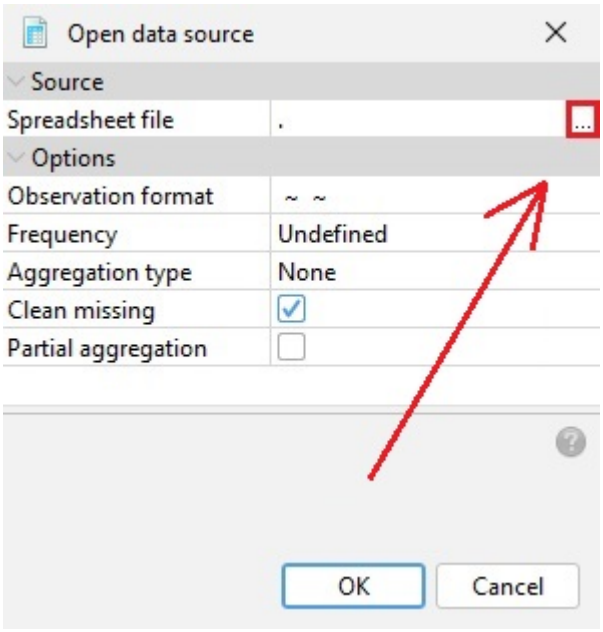
1. From the *Providers* window **right-click** on the *Spreadsheets* branch and choose *Open* option.



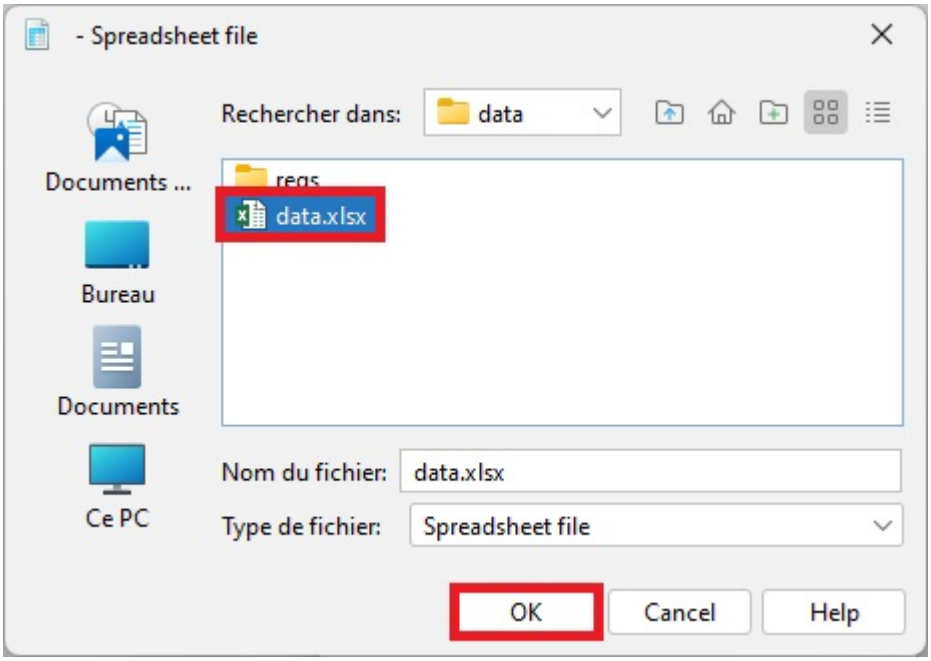
2. The *Open data source* window contains the following options:

   - **Spreadsheet file** – a path to access the Excel file.
   - **Data format** (or **Observartion format** in v3) – the data format used to read dates and values. It includes three fields: *locale* (country), *date pattern* (data format, e.g. *yyyy-mm-dd*), *number pattern* (a metaformat of numeric value, e.g. 0.## represents two digit number).
   - **Frequency** – time series frequency. This can be undefined, yearly, half-yearly, four-monthly, quarterly, bi-monthly, or monthly. When the frequency is set to undefined, JDemetra+ determines the time series frequency by analysing the sequence of dates in the file.
   - **Aggregation type** – the type of aggregation (over time for each time series in the dataset) for the imported time series. This can be *None*, *Sum*, *Average*, *First*, *Last*, *Min* or *Max*. The aggregation can be performed only if the *frequency* parameter is specified. For example, when frequency is set to *Quarterly* and aggregation type is set to *Average*, a monthly time series is transformed to quarterly one with values that are equal to the one third of the sum of the monthly values that belong to the corresponding calendar quarter.
   - **Clean missing** – erases missing values at the start of the series.

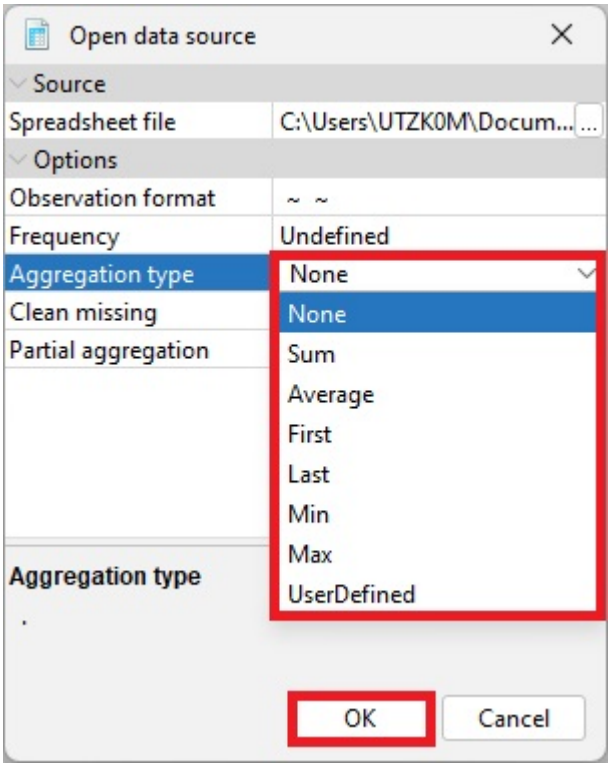
   Next, in the *Source* section click the grey "…" button to open the file.



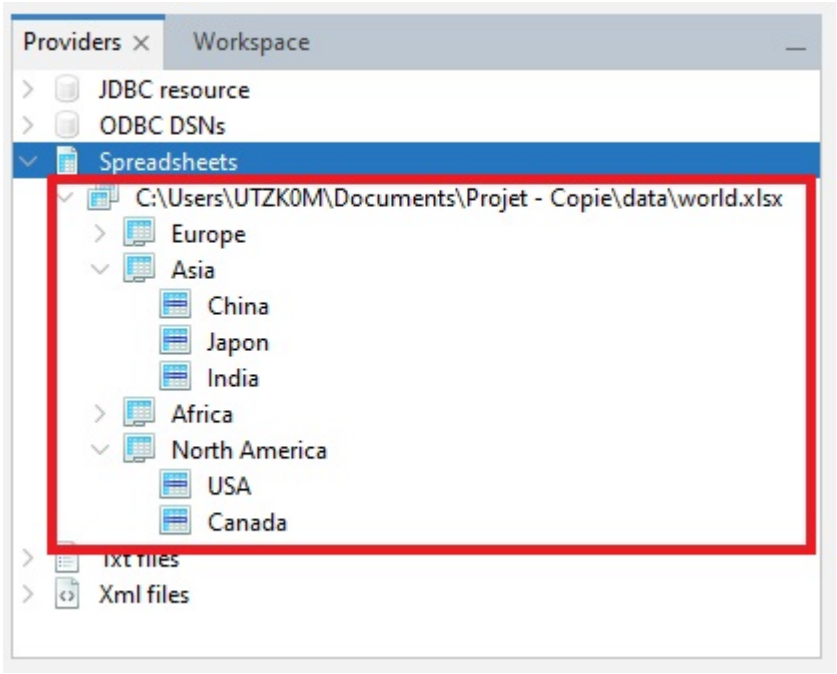
3. Choose a file and click *OK*.



4. The user may specify *Data format*, *Frequency* and *Aggregation type*, however this step is not compulsory. When these options are specified JDemetra+ is able to convert the time series frequency. Otherwise, the functionality that enables the time series frequency to be converted will not be available.



5. The data are organized in a tree structure.



Once imported, your s is visible as a “node” structure

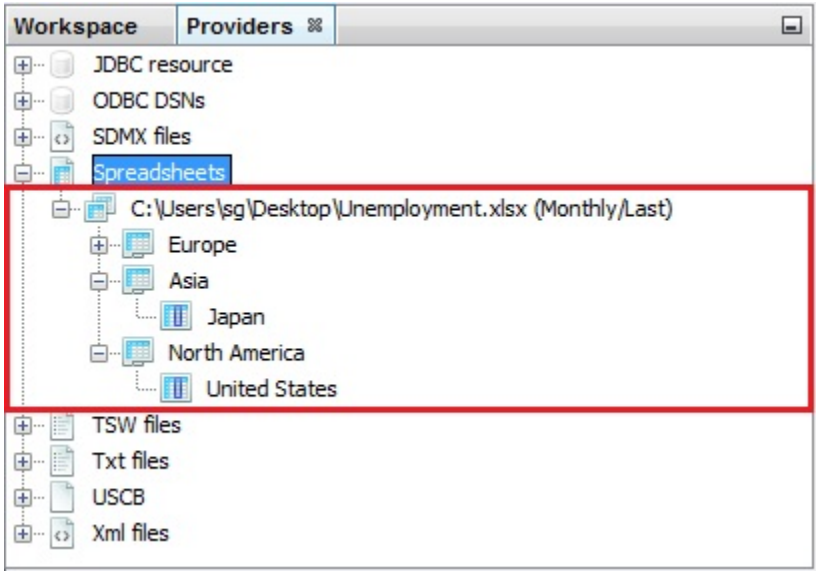


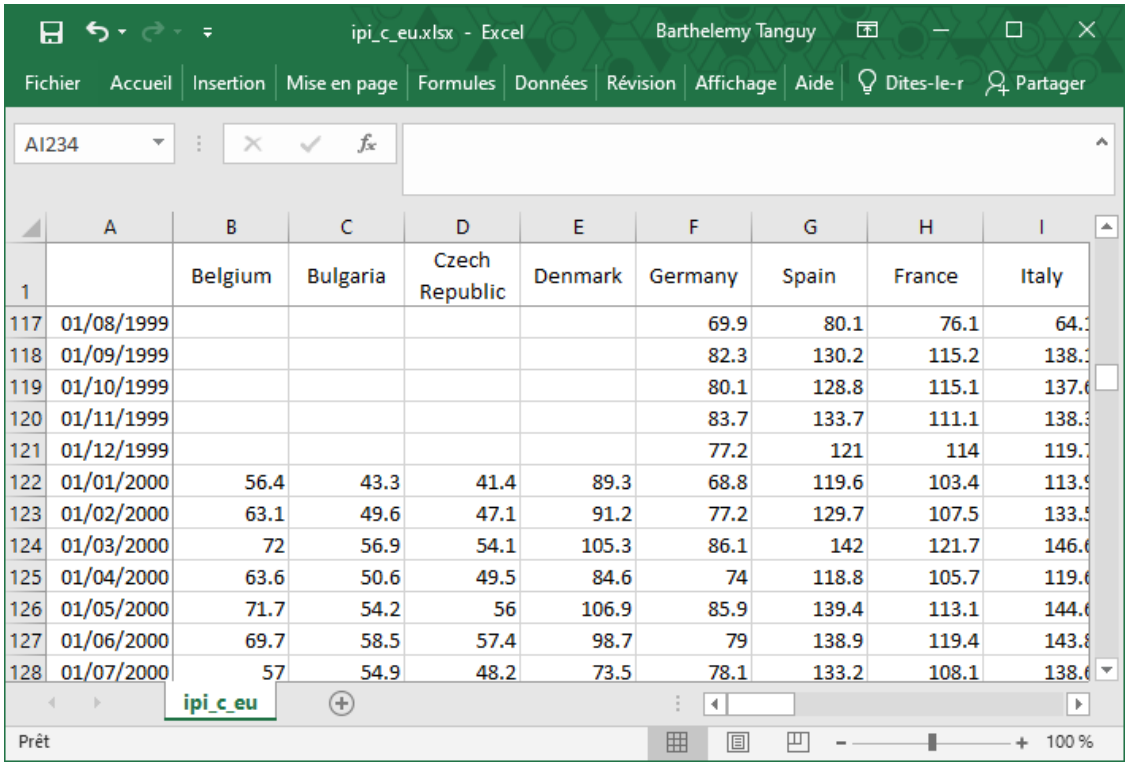
Figure 116: **A structure of a dataset**

Accepted formats

In v2, the formats `.xls` and `.xlsx` are accepted.
In v3, only the format `.xlsx` is accepted (`.xls` files are no longer supported).

How to set-up your spreadsheet

- **Dates** in Excel date format, in the first column (or in the first row)
- **Titles** of the series in the corresponding cell of the first row (or in the first column)
- **Top-left cell** $A1$ can include text or it can be left empty
- **Empty cells** are interpreted by JDemetra+ as missing values
- If empty cells are at the beginning of the series they can be ignored using the option **clean-missing**.



		Belgium	Bulgaria	Czech Republic	Denmark	Germany	Spain	France	Italy
117	01/08/1999					69.9	80.1	76.1	64.1
118	01/09/1999					82.3	130.2	115.2	138.1
119	01/10/1999					80.1	128.8	115.1	137.6
120	01/11/1999					83.7	133.7	111.1	138.3
121	01/12/1999					77.2	121	114	119.7
122	01/01/2000	56.4	43.3	41.4	89.3	68.8	119.6	103.4	113.9
123	01/02/2000	63.1	49.6	47.1	91.2	77.2	129.7	107.5	133.5
124	01/03/2000	72	56.9	54.1	105.3	86.1	142	121.7	146.6
125	01/04/2000	63.6	50.6	49.5	84.6	74	118.8	105.7	119.6
126	01/05/2000	71.7	54.2	56	106.9	85.9	139.4	113.1	144.6
127	01/06/2000	69.7	58.5	57.4	98.7	79	138.9	119.4	143.8
128	01/07/2000	57	54.9	48.2	73.5	78.1	133.2	108.1	138.6

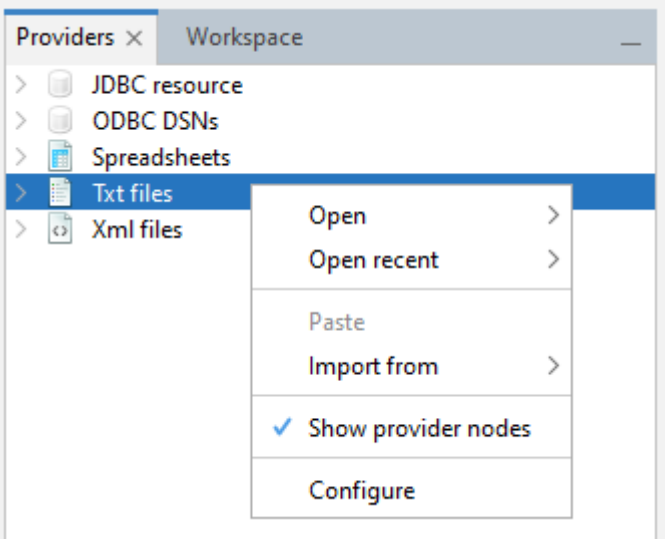
Figure 117: **Example of an Excel spreadsheet that can be imported to JDemetra+**

In Excel files, series are identified by their names (colnames) in the file.

0.0.0.2 `.txt` or `.csv` file

The example below show how to import the data from an Excel file.

1. From the *Providers* window **right-click** on the *Txt files* branch and choose *Open* option.



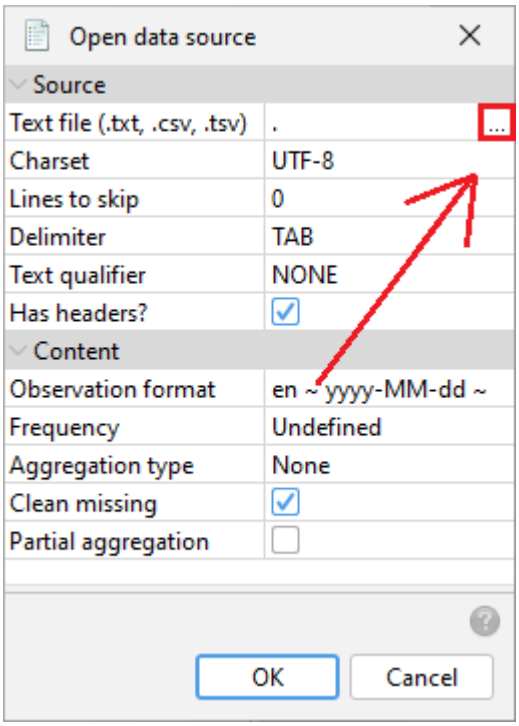
2. The *Open data source* window contains the following options:

   - `.txt` **file** – a path to access the file.
   - **Charset** – the encoding used to encode the file
   - **Lines to skip** – the number of lines to skip before reading the data
   - **Delimiter** – the character used to separate fields in the file
   - **Text qualifier** – the characters used to retrieve text fileds
   - **Has header** – check tu use the first line as header
   - **Data format** (or **Observartion format** in v3) – the data format used to read dates and values. It includes three fields: *locale* (country), *date pattern* (data format, e.g. *yyyy-mm-dd*), *number pattern* (a metaformat of numeric value, e.g. 0.## represents two digit number).
   - **Frequency** – time series frequency. This can be undefined, yearly, half-yearly, four-monthly, quarterly, bi-monthly, or monthly. When the frequency is set to undefined, JDemetra+ determines the time series frequency by analysing the sequence of dates in the file.
   - **Aggregation type** – the type of aggregation (over time for each time series in the dataset) for the imported time series. This can be *None*, *Sum*, *Average*, *First*, *Last*, *Min* or *Max*. The aggregation can be performed only if the *frequency* parameter is specified. For example, when frequency is set to *Quarterly* and aggregation type is set to *Average*, a monthly time series is transformed to quarterly one with values that are equal to the

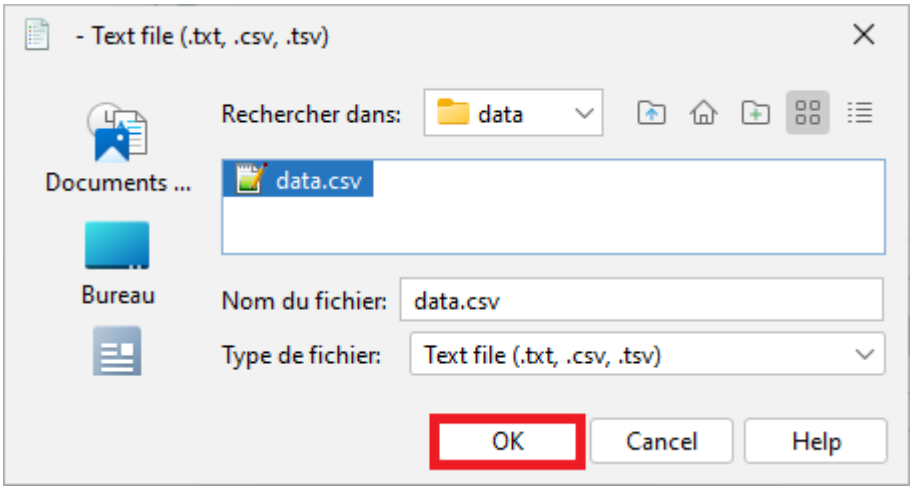
one third of the sum of the monthly values that belong to the corresponding calendar quarter.

- **Clean missing** – erases the missing values of the series.
- **Partial aggregation** – Allow partial aggregation (only with average and sum aggregation).

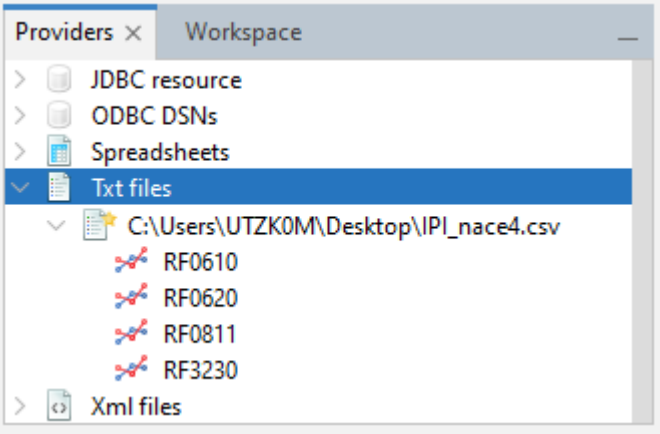
Next, in the *Source* section click the grey "…" button to open the file.



3. Choose a file and click *OK*.

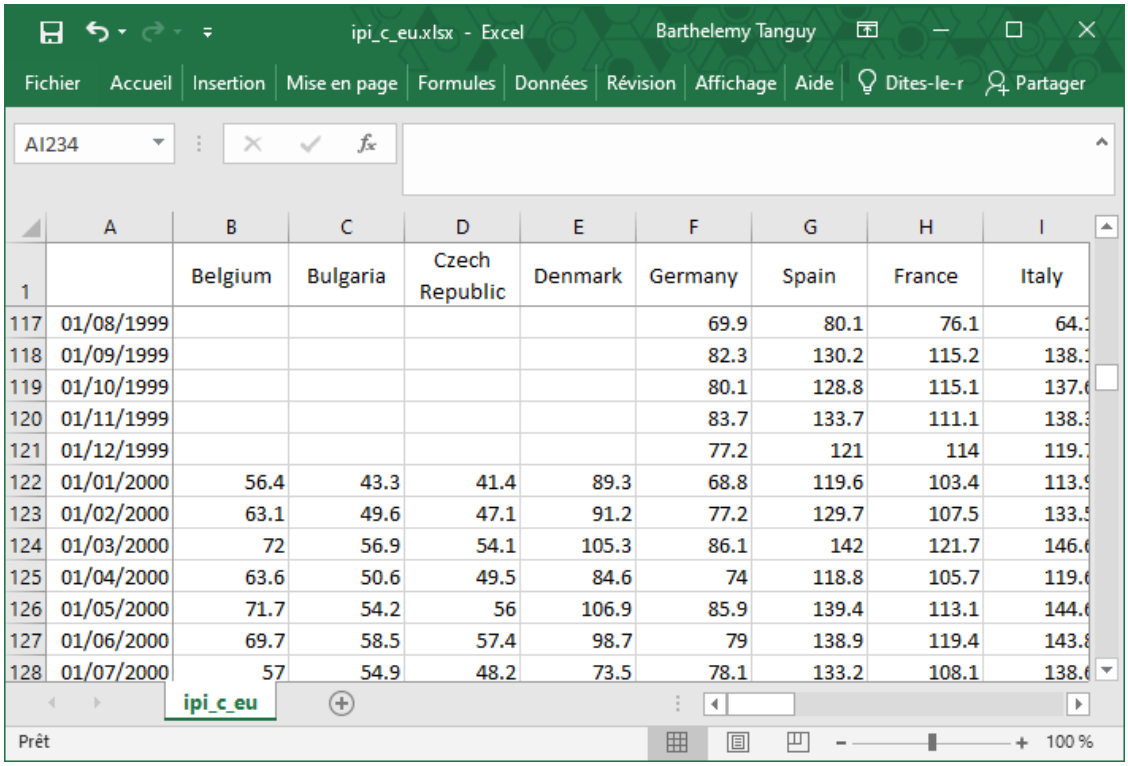


4. The data are organized in a tree structure.



i How to set up you `.txt` or `.csv` file

- **Dates** in Excel date format, in the first column (or in the first row)
- **Titles** of the series in the corresponding cell of the first row (or in the first column)
- **Top-left cell** $A1$ can include text or it can be left empty
- **Empty cells** are interpreted by JDemetra+ as missing values
- If empty cells are at the beginning of the series they can be ignored using the option **clean-missing**.



		Belgium	Bulgaria	Czech Republic	Denmark	Germany	Spain	France	Italy
117	01/08/1999					69.9	80.1	76.1	64.1
118	01/09/1999					82.3	130.2	115.2	138.1
119	01/10/1999					80.1	128.8	115.1	137.6
120	01/11/1999					83.7	133.7	111.1	138.3
121	01/12/1999					77.2	121	114	119.7
122	01/01/2000	56.4	43.3	41.4	89.3	68.8	119.6	103.4	113.9
123	01/02/2000	63.1	49.6	47.1	91.2	77.2	129.7	107.5	133.5
124	01/03/2000	72	56.9	54.1	105.3	86.1	142	121.7	146.6
125	01/04/2000	63.6	50.6	49.5	84.6	74	118.8	105.7	119.6
126	01/05/2000	71.7	54.2	56	106.9	85.9	139.4	113.1	144.6
127	01/06/2000	69.7	58.5	57.4	98.7	79	138.9	119.4	143.8
128	01/07/2000	57	54.9	48.2	73.5	78.1	133.2	108.1	138.6

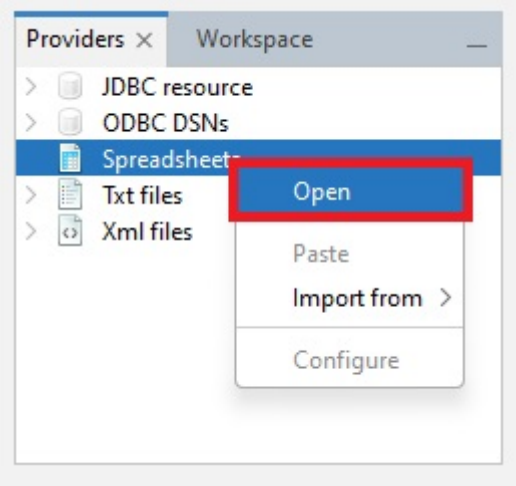
Figure 118: **Example of an Excel spreadsheet that can be imported to JDemetra+**

In text files, series are identified by their position in the file.

Spreadsheet

The example below show how to import the data from an Excel file.

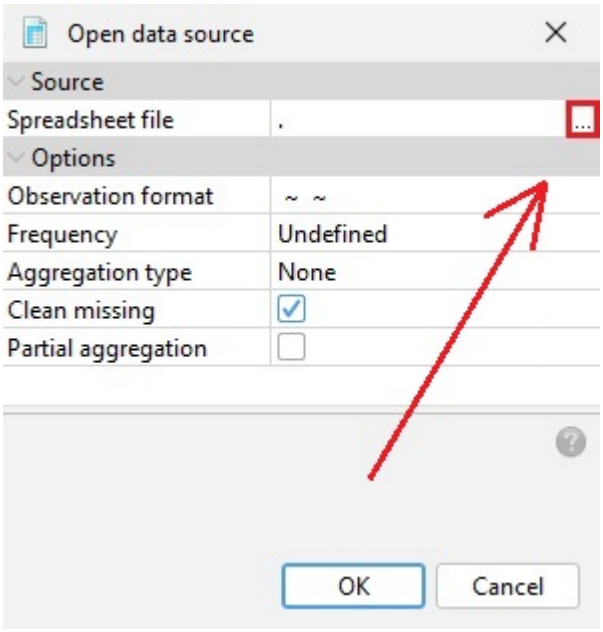
1. From the *Providers* window **right-click** on the *Spreadsheets* branch and choose *Open* option.



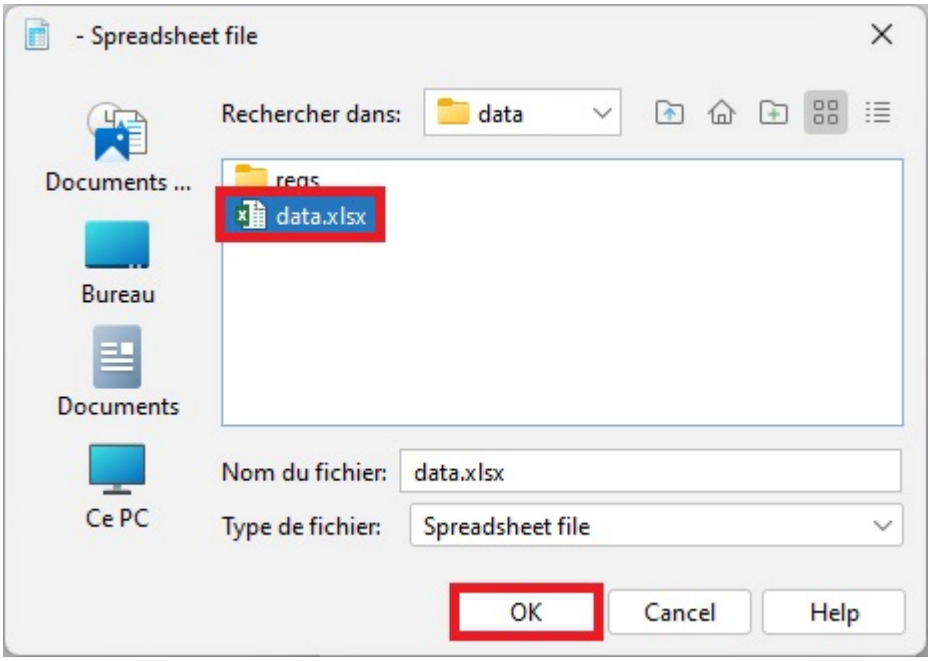
2. The *Open data source* window contains the following options:

   - **Spreadsheet file** – a path to access the Excel file.
   - **Data format** (or **Observartion format** in v3) – the data format used to read dates and values. It includes three fields: *locale* (country), *date pattern* (data format, e.g. *yyyy-mm-dd*), *number pattern* (a metaformat of numeric value, e.g. 0.## represents two digit number).
   - **Frequency** – time series frequency. This can be undefined, yearly, half-yearly, four-monthly, quarterly, bi-monthly, or monthly. When the frequency is set to undefined, JDemetra+ determines the time series frequency by analysing the sequence of dates in the file.
   - **Aggregation type** – the type of aggregation (over time for each time series in the dataset) for the imported time series. This can be *None*, *Sum*, *Average*, *First*, *Last*, *Min* or *Max*. The aggregation can be performed only if the *frequency* parameter is specified. For example, when frequency is set to *Quarterly* and aggregation type is set to *Average*, a monthly time series is transformed to quarterly one with values that are equal to the one third of the sum of the monthly values that belong to the corresponding calendar quarter.
   - **Clean missing** – erases the missing values of the series.

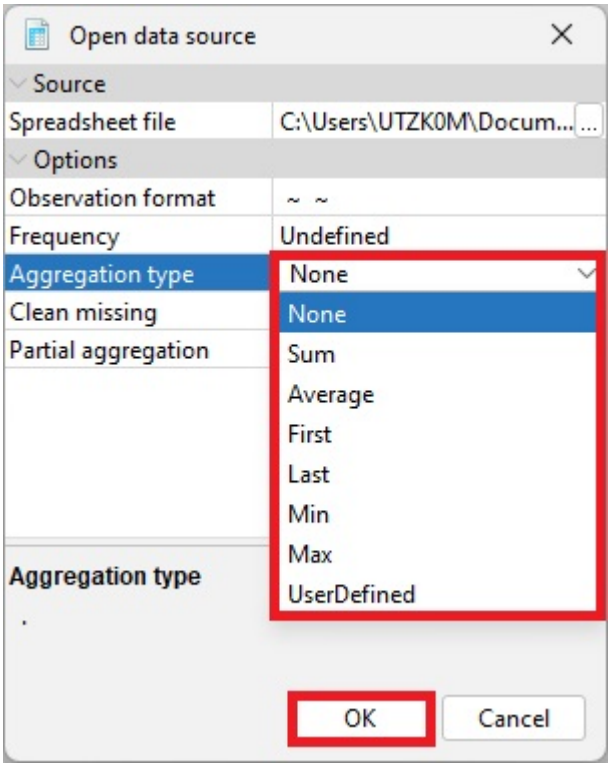
   Next, in the *Source* section click the grey "..." button to open the file.



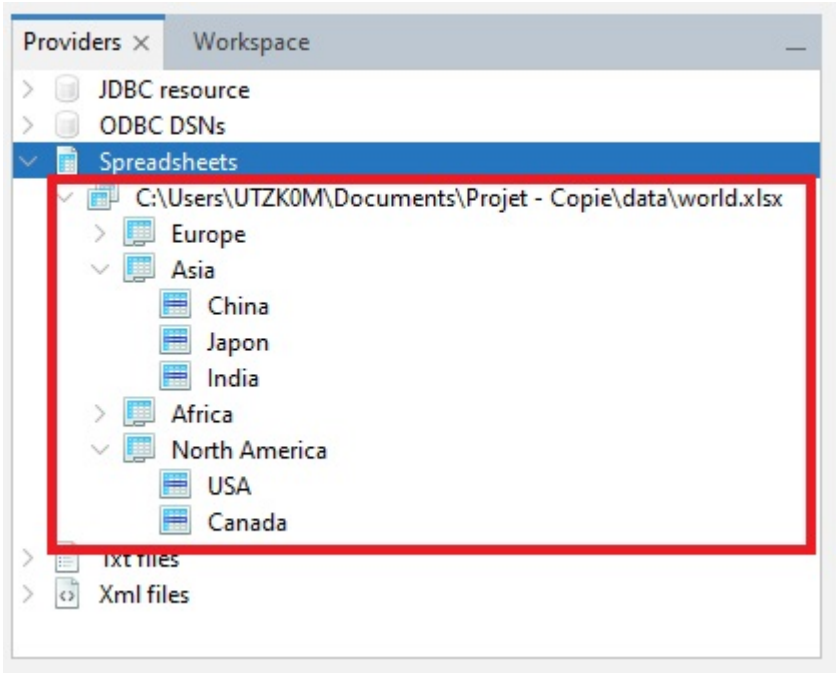
3. Choose a file and click *OK*.



4. The user may specify *Data format*, *Frequency* and *Aggregation type*, however this step is not compulsory. When these options are specified JDemetra+ is able to convert the time series frequency. Otherwise, the functionality that enables the time series frequency to be converted will not be available.



5. The data are organized in a tree structure.



Once imported, your spreadsheet is visible as a “node” structure

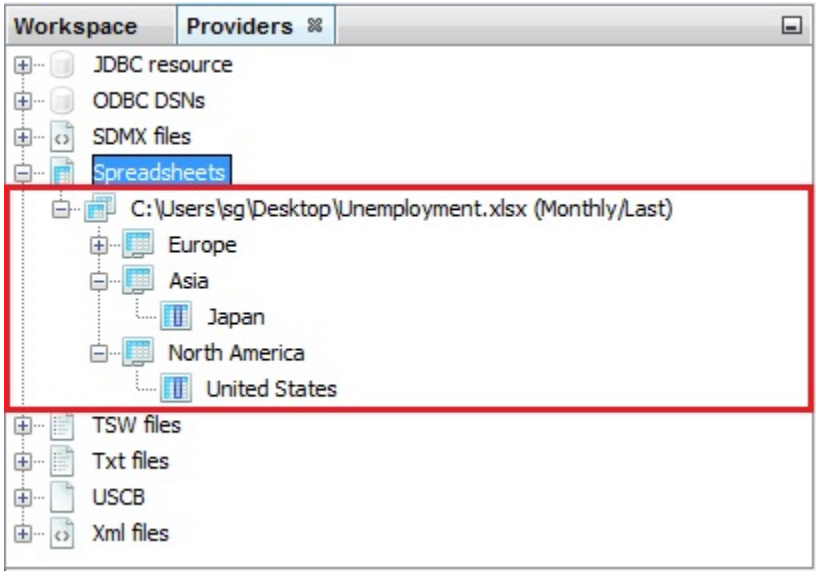


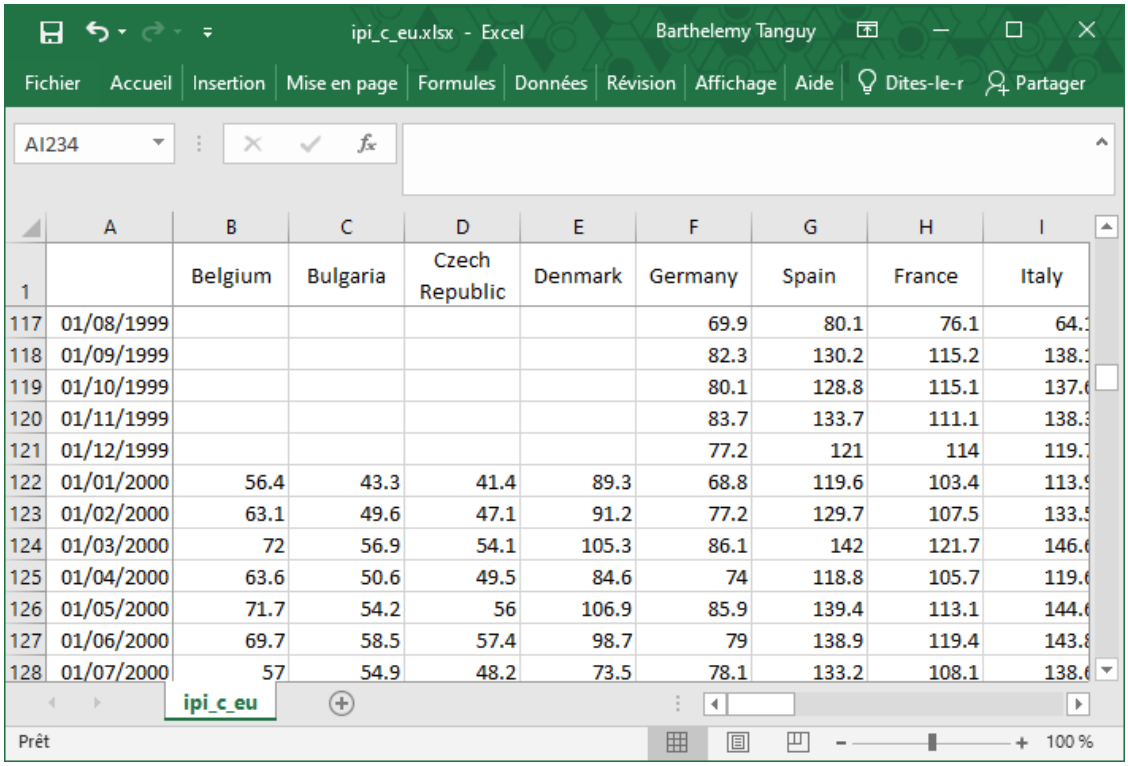
Figure 119: **A structure of a dataset**

Accepted formats

In v2, the formats `.xls` and `.xlsx` are accepted.
In v3, only the format `.xlsx` is accepted. `.xls` files are no longer supported.

How to set-up your spreadsheet

- **Dates** in Excel date format, in the first column (or in the first row)
- **Titles** of the series in the corresponding cell of the first row (or in the first column)
- **Top-left cell** $A1$ can include text or it can be left empty
- **Empty cells** are interpreted by JDemetra+ as missing values
- If empty cells are at the beginning of the series they can be ignored using the option **clean-missing**.



	A	B	C	D	E	F	G	H	I
1		Belgium	Bulgaria	Czech Republic	Denmark	Germany	Spain	France	Italy
117	01/08/1999					69.9	80.1	76.1	64.1
118	01/09/1999					82.3	130.2	115.2	138.1
119	01/10/1999					80.1	128.8	115.1	137.6
120	01/11/1999					83.7	133.7	111.1	138.3
121	01/12/1999					77.2	121	114	119.7
122	01/01/2000	56.4	43.3	41.4	89.3	68.8	119.6	103.4	113.9
123	01/02/2000	63.1	49.6	47.1	91.2	77.2	129.7	107.5	133.5
124	01/03/2000	72	56.9	54.1	105.3	86.1	142	121.7	146.6
125	01/04/2000	63.6	50.6	49.5	84.6	74	118.8	105.7	119.6
126	01/05/2000	71.7	54.2	56	106.9	85.9	139.4	113.1	144.6
127	01/06/2000	69.7	58.5	57.4	98.7	79	138.9	119.4	143.8
128	01/07/2000	57	54.9	48.2	73.5	78.1	133.2	108.1	138.6

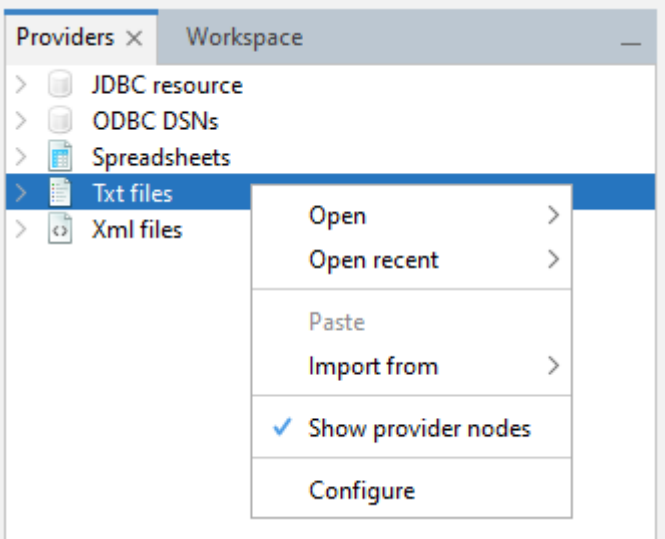
Figure 120: **Example of an Excel spreadsheet that can be imported to JDemetra+**

In Excel files, series are identified by their names (colnames) in the file.

`.txt` or `.csv` file

The example below show how to import the data from an Excel file.

1. From the *Providers* window **right-click** on the *Txt files* branch and choose *Open* option.



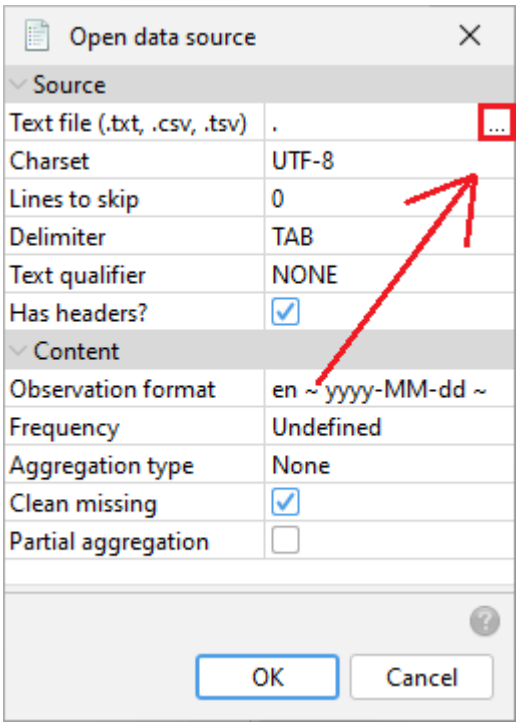
2. The *Open data source* window contains the following options:

   - **Text file** – a path to access the file.
   - **Charset** – the encoding used to encode the file
   - **Lines to skip** – the number of lines to skip before reading the data
   - **Delimiter** – the character used to separate fields in the file
   - **Text qualifier** – the characters used to retrieve text fileds
   - **Has header** – check tu use the first line as header
   - **Data format** (or **Observartion format** in v3) – the data format used to read dates and values. It includes three fields: *locale* (country), *date pattern* (data format, e.g. *yyyy-mm-dd*), *number pattern* (a metaformat of numeric value, e.g. 0.## represents two digit number).
   - **Frequency** – time series frequency. This can be undefined, yearly, half-yearly, four-monthly, quarterly, bi-monthly, or monthly. When the frequency is set to undefined, JDemetra+ determines the time series frequency by analysing the sequence of dates in the file.
   - **Aggregation type** – the type of aggregation (over time for each time series in the dataset) for the imported time series. This can be *None*, *Sum*, *Average*, *First*, *Last*, *Min* or *Max*. The aggregation can be performed only if the *frequency* parameter is specified. For example, when frequency is set to *Quarterly* and aggregation type is set to *Average*, a monthly time series is transformed to quarterly one with values that are equal to the

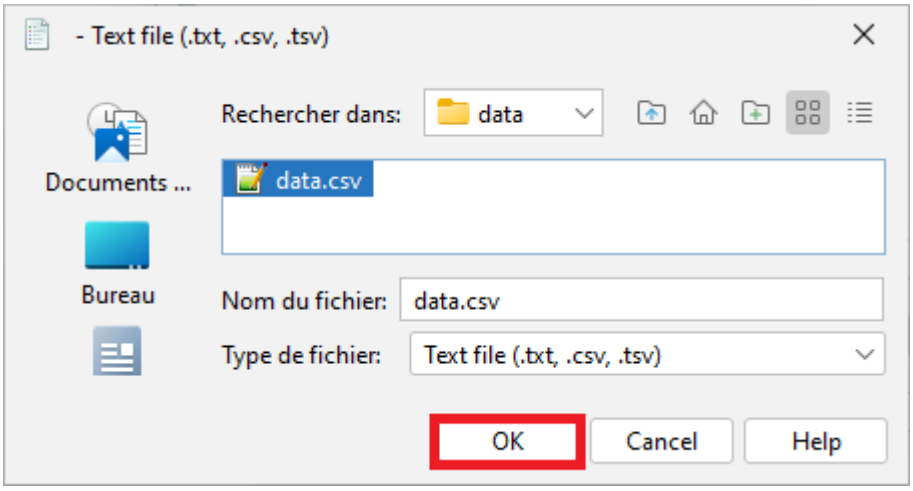
one third of the sum of the monthly values that belong to the corresponding calendar quarter.

- **Clean missing** – erases the missing values of the series.
- **Partial aggregation** – Allow partial aggregation (only with average and sum aggregation).

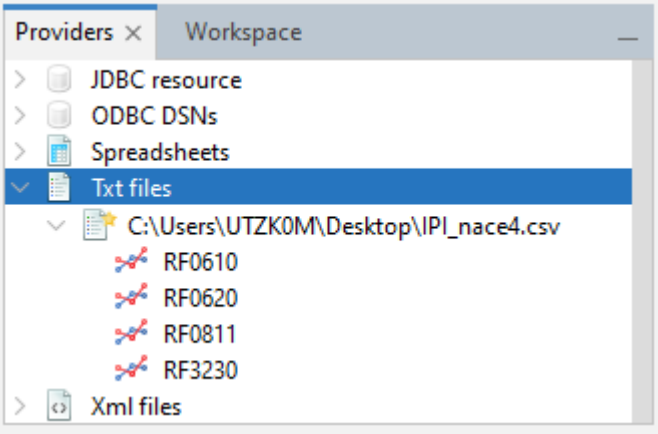
Next, in the *Source* section click the grey "…" button to open the file.



3. Choose a file and click *OK*.

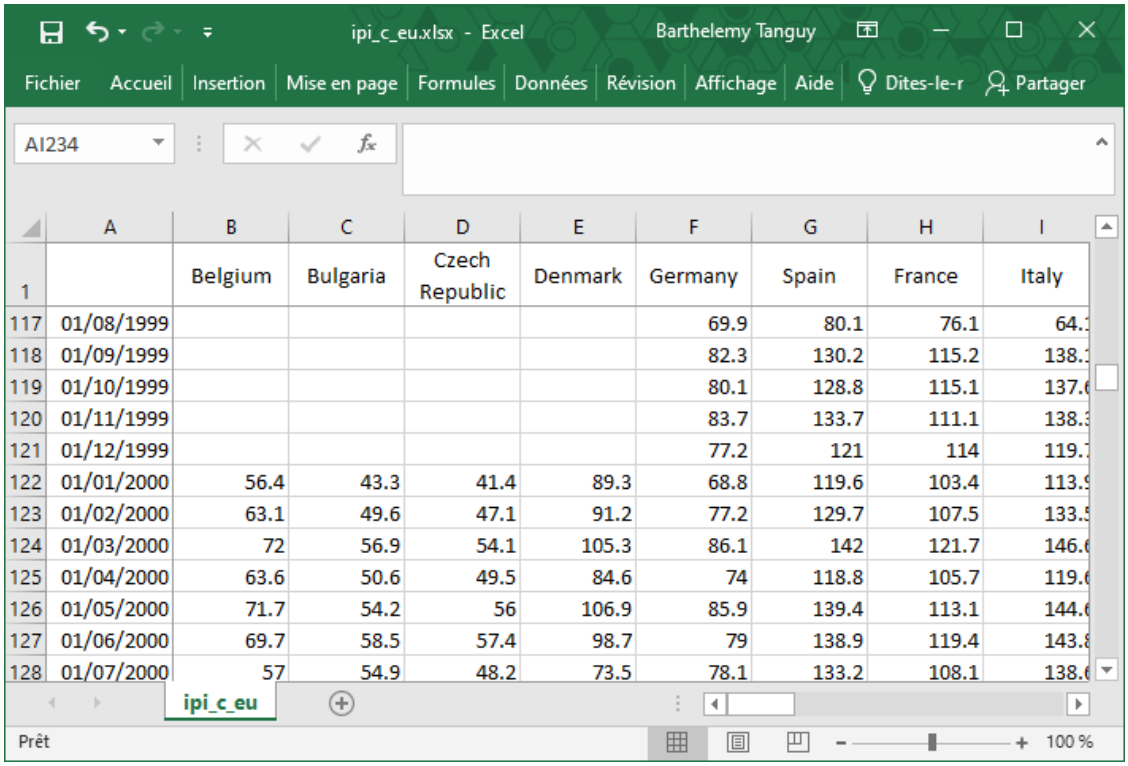


4. The data are organized in a tree structure.



i How to set up you `.txt` or `.csv` file

- **Dates** in Excel date format, in the first column (or in the first row)
- **Titles** of the series in the corresponding cell of the first row (or in the first column)
- **Top-left cell** $A1$ can include text or it can be left empty
- **Empty cells** are interpreted by JDemetra+ as missing values
- If empty cells are at the beginning of the series they can be ignored using the option **clean-missing**.



	A	B	C	D	E	F	G	H	I
1		Belgium	Bulgaria	Czech Republic	Denmark	Germany	Spain	France	Italy
117	01/08/1999					69.9	80.1	76.1	64.1
118	01/09/1999					82.3	130.2	115.2	138.1
119	01/10/1999					80.1	128.8	115.1	137.6
120	01/11/1999					83.7	133.7	111.1	138.3
121	01/12/1999					77.2	121	114	119.7
122	01/01/2000	56.4	43.3	41.4	89.3	68.8	119.6	103.4	113.9
123	01/02/2000	63.1	49.6	47.1	91.2	77.2	129.7	107.5	133.5
124	01/03/2000	72	56.9	54.1	105.3	86.1	142	121.7	146.6
125	01/04/2000	63.6	50.6	49.5	84.6	74	118.8	105.7	119.6
126	01/05/2000	71.7	54.2	56	106.9	85.9	139.4	113.1	144.6
127	01/06/2000	69.7	58.5	57.4	98.7	79	138.9	119.4	143.8
128	01/07/2000	57	54.9	48.2	73.5	78.1	133.2	108.1	138.6

Figure 121: **Example of an Excel spreadsheet that can be imported to JDemetra+**

In text files, series are identified by their position in the file.

Wrangling data

Series uploaded to the *Providers* window can be

- Displayed,
- Modified
- Tested for seasonality / white noise

or used in any available algorithm (link to list)

- Modelled Modelling
- Seasonnally adjusted
- Benchmarked

Behaviour options

Restoring data sources

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.

Add Star

You can also favorite files to find them each time you open the software.

To favorite a file:

- **right-click** on the file
- click on **Add star**

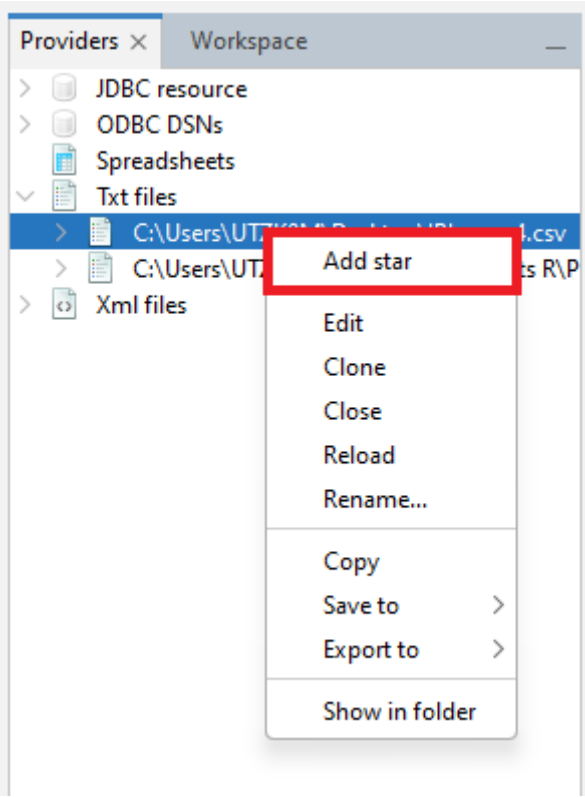


Figure 122: **Create a new favorite**

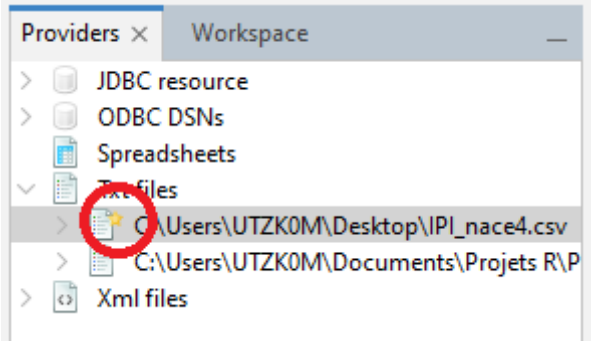


Figure 123: **Example of a favorite file**

A favorite file will have a little star on top right of the logo:

To remove a favorite:

- **right-click** on the file
- click on **Remove star**

Workspace Structure

The workspace is the main data structure used by JDemetra+.

The workspace saved by JDemetra+ includes:

- Main folder containing several folders that correspond to the different types of items created by the user and;
- The `.xml` file that enables the user to import the workspace to the application and to display its content.

The workspace can be shared with other users, which eases the burden of work with defining specifications, modelling and seasonal adjustment processes.

The main folder contains:

- a folder **SAProcessing** with all the result of the SA
- folders **TramoSeatsSpec** and/or **X-13Spec** with the custom specifications
- a folder **Variables** for external regressor and variables
- a folder **Calendars** with the calendars used to correct the trading days effect
- a folder **Output** contains all the generated output from the GUI

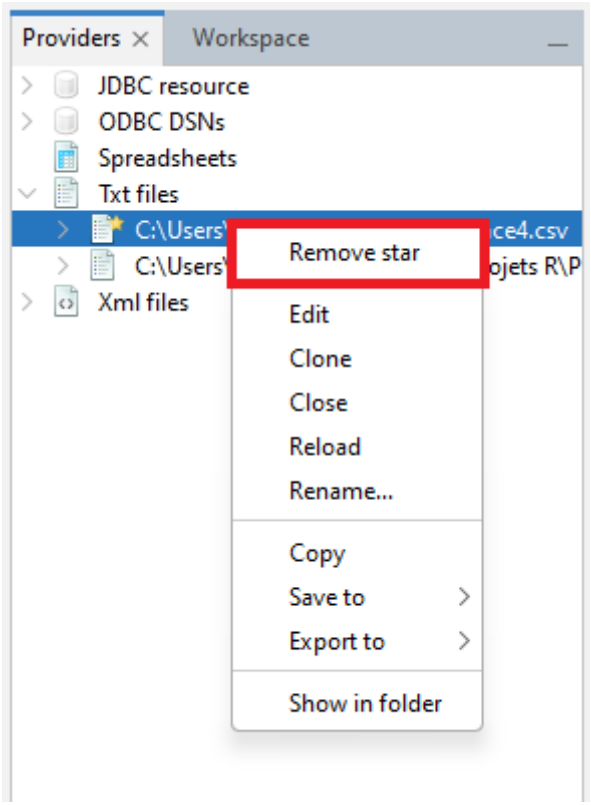


Figure 124: **Remove a favorite**

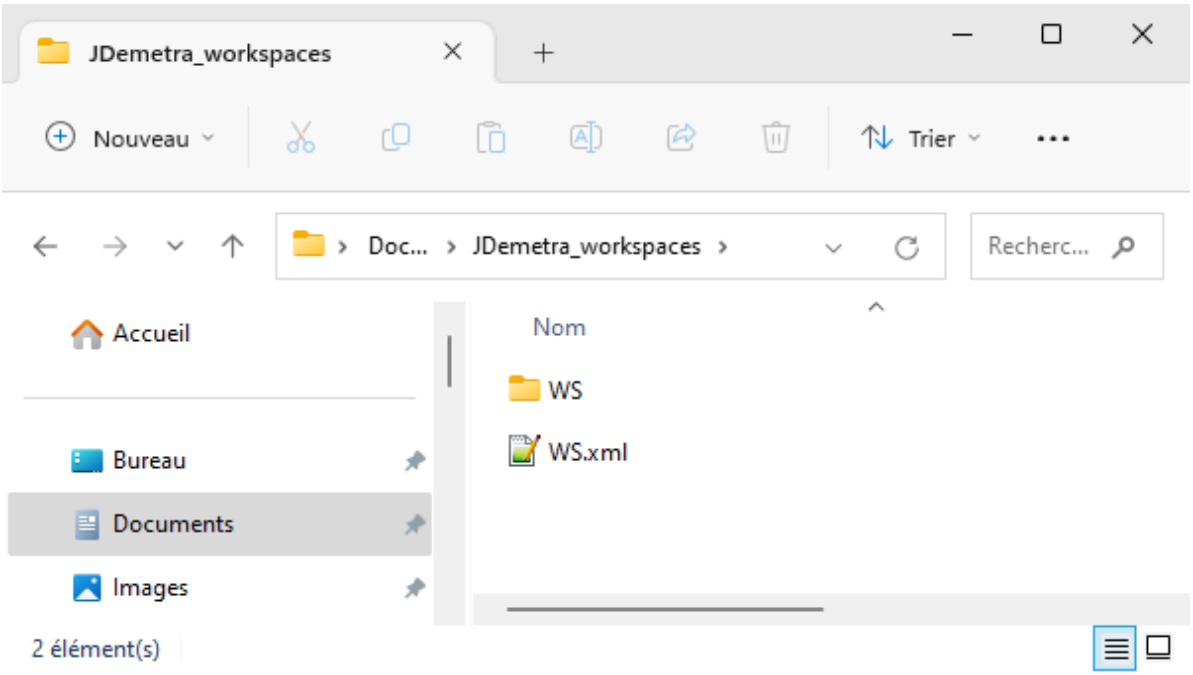


Figure 125: **Example of a workspace created by JDemetra+.**

Workspace window

The workspace window displays the **characteristics** of a workspace but ALSO gives access to other peripheric routines, the results of which won't be stored in a workspace (as data structure).

You can find:

- Modelling (contains the default and user-defined specifications for modelling; and the output from the modelling process)
- Seasonal adjustment (contains the default and user-defined specifications for seasonal adjustment and the output from the seasonal adjustment process),
- Utilities (calendars and user defined variables).

Results Panel

Results Panel of seasonal adjustment will be presented in another chapter

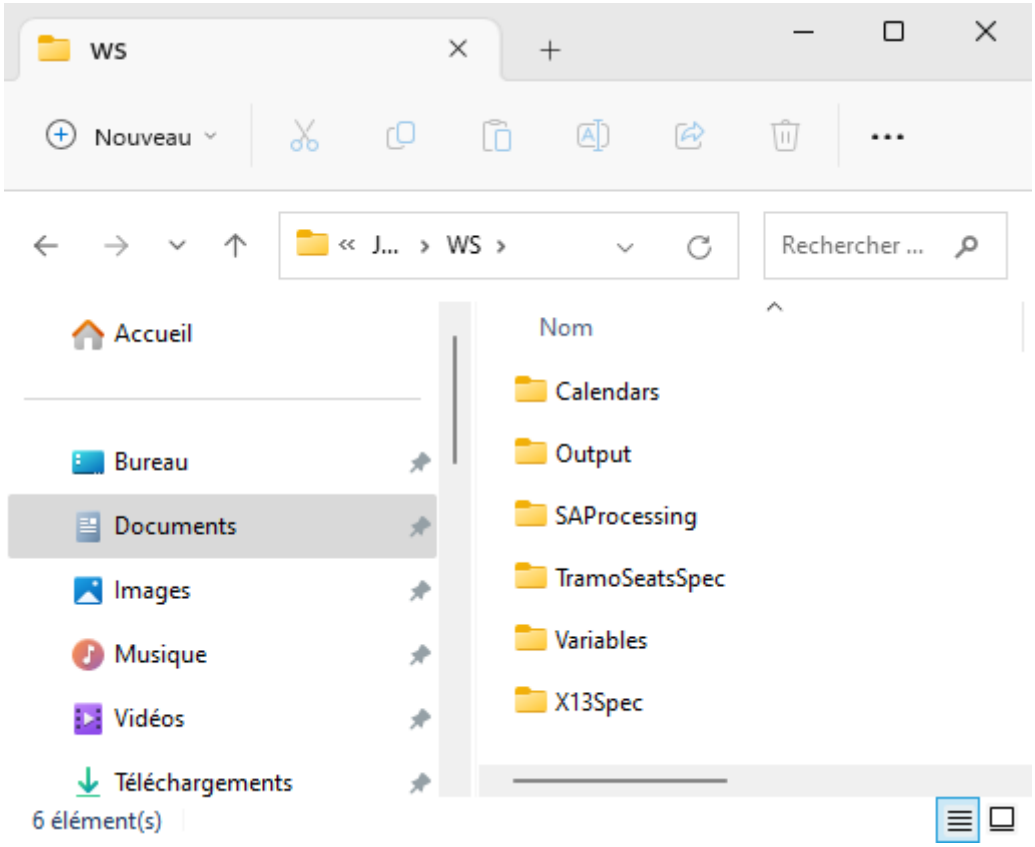


Figure 126: **Structure of a workspace**

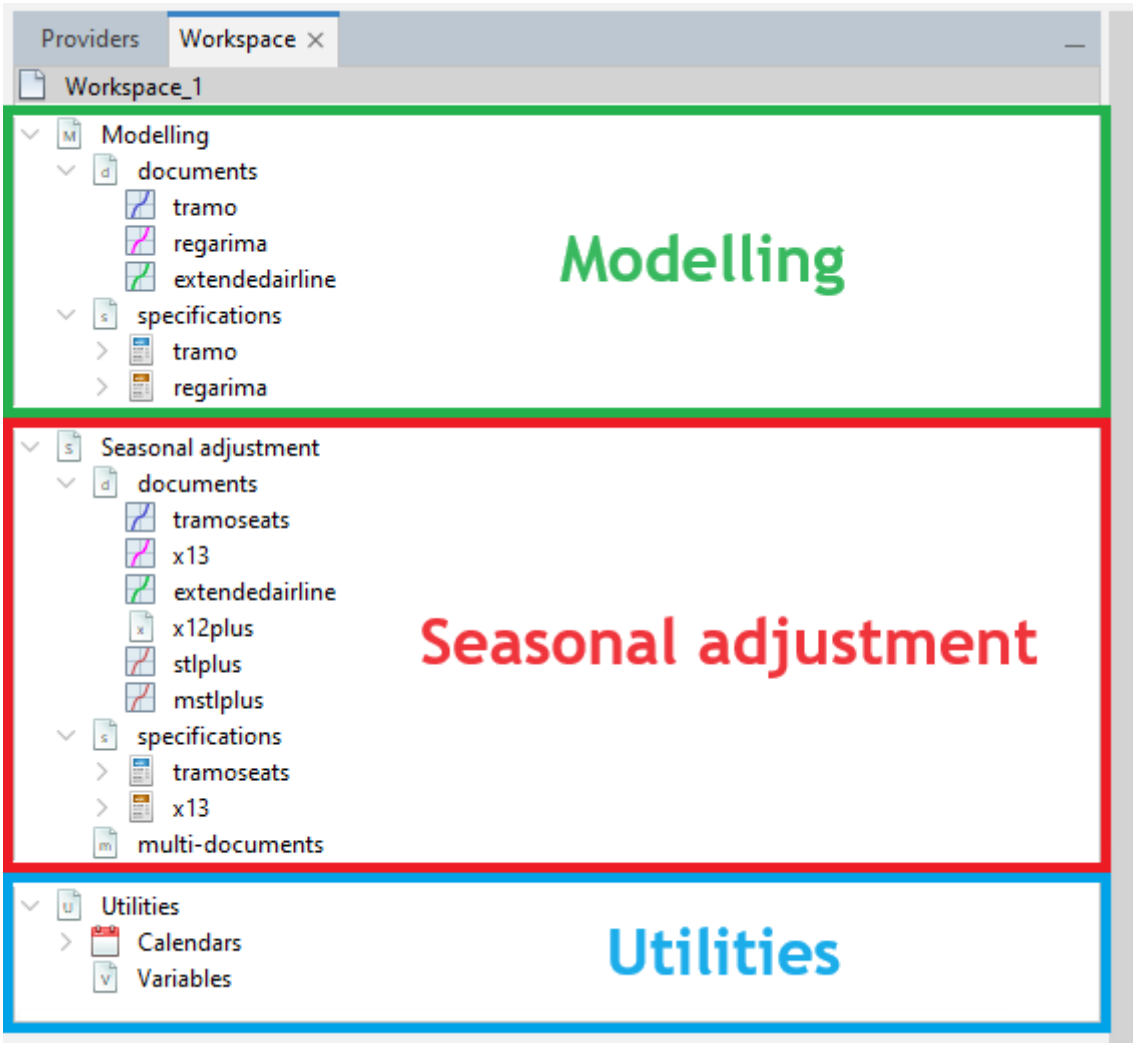


Figure 127: **The *Workspace* window**

Top Bar Menu and options

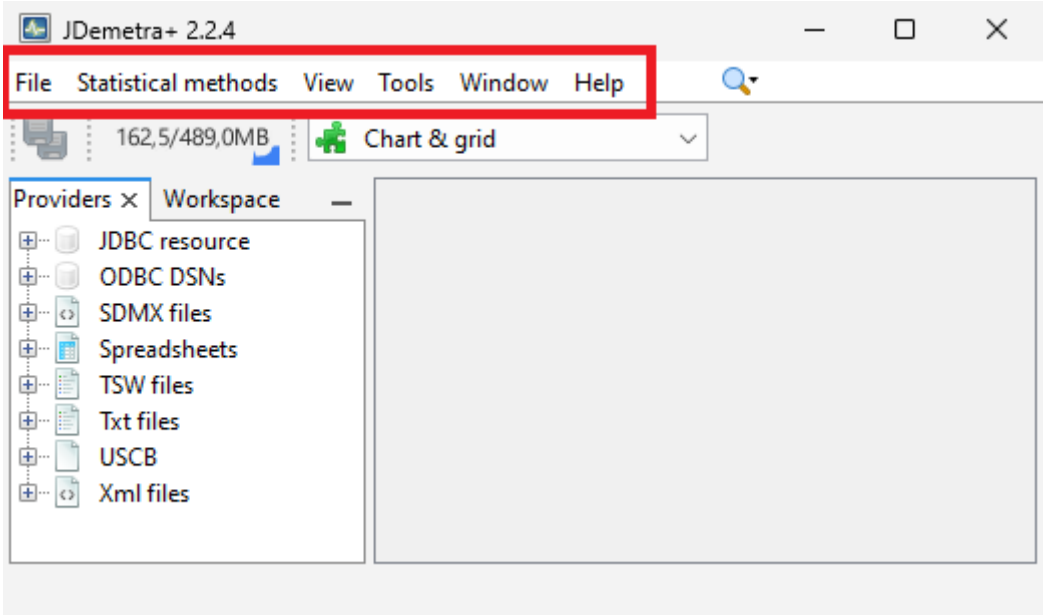


Figure 128: **The Top bar menus**

The majority of functionalities are available from the main application menu, which is situated at the very top of the main window. If the user moves the cursor to an entry in the main menu and clicks on the left mouse button, a drop-down menu will appear. Clicking on an entry in the drop-down menu selects the highlighted item.

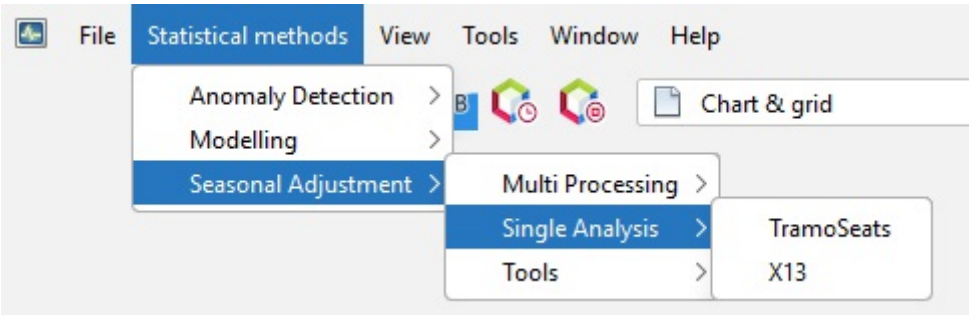


Figure 129: **The main menu with selected drop-down menu**

The functions available in the main application menu are:

- File
- Statistical methods

- View
- Tools
- Window
- Help
- RegARIMADoc
- X-13Doc
- TramoDoc
- TramoSeatsDoc
- SAProcessingDoc

File

The *File* menu is intended for working with workspaces and data sources. It offers the following functions:

- **New Workspace** – creates a new workspace and displays it in the *Workspace* window with a default name (*Workspace_#number*);
- **Open Workspace** – opens a dialog window, which enables the user to select and open an existing workspace;
- **Open Recent Workspace** – presents a list of workspaces recently created by the user and enables the user to open one of them;
- **Save Workspace** – saves the project file named by the system under the default name (*Workspace_#number*) and in a default location. The workspace can be re-opened at a later time;
- **Save Workspace As...** – saves the current workspace under the name chosen by the user in the chosen location. The workspace can be re-opened at a later time;
- **Open Recent** – presents a list of datasets recently used and enables the user to open one of them;
- **Exit** – closes an application.

Statistical Methods

Here just a hint and link to relevant chapters

The Statistical methods menu includes functionalities for modelling, analysis and the seasonal adjustment of a time series. They are divided into three groups:

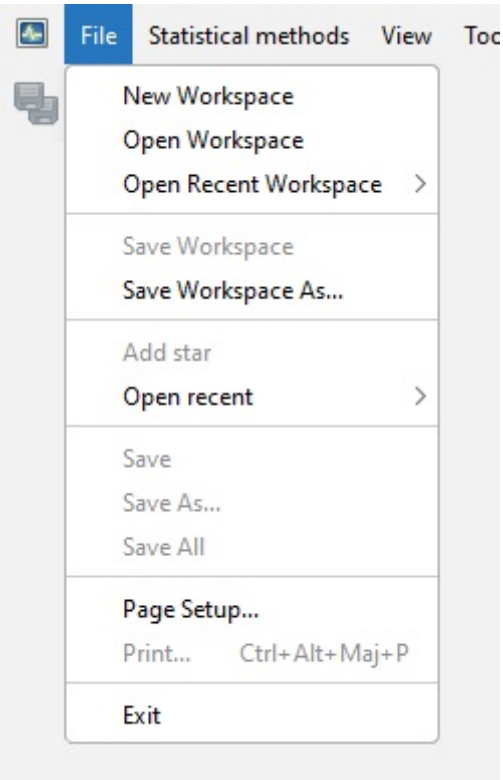


Figure 130: **The content of the *File* menu**

- Anomaly Detection – allows for a purely automatic identification of regression effects;

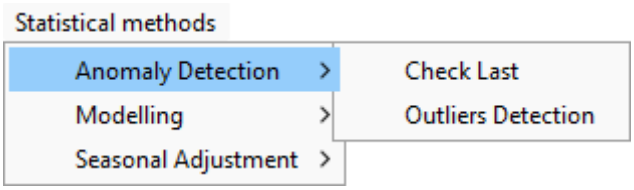


Figure 131: **The *Anomaly detection* tab.**

- Modelling – enables time series modelling using the Tramo and Reg-ARIMA models;

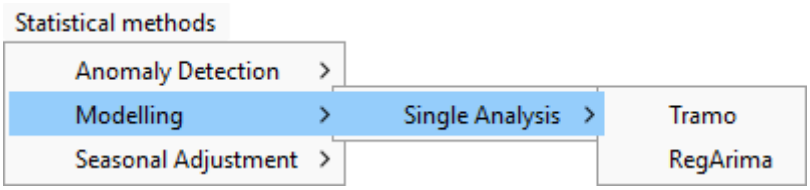


Figure 132: **The *Modelling* tab.**

- Seasonal adjustment – intended for the seasonal adjustment of a time series with the Tramo-Seats and X-13ARIMA-Seats methods.

By default, the *Seasonal adjustment* tab has 3 sub-tabs:

- *Multiprocessing*,
- *Single Analysis*,
- and *Tools*.

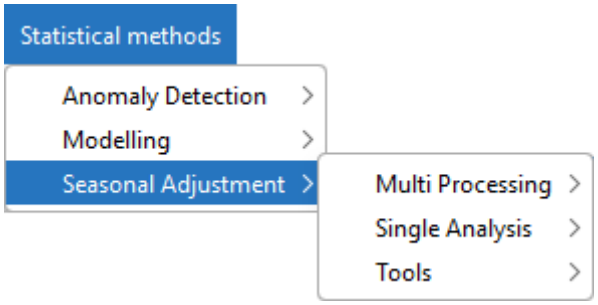


Figure 133: **The *Seasonal adjustment* tab in v2**

0.0.0.1 v2

In v2, the *Tools* sub-tabs contains 2 functionalities:

- *Seasonality Tests*
- *Direct-Indirect Seasonal Adjustment*

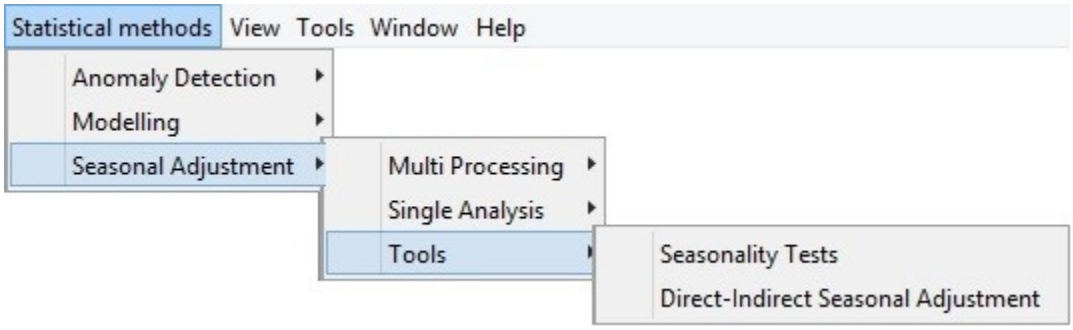


Figure 134: **The *Seasonal adjustment* tab in v2**

0.0.0.2 v3

In v3, the *Tools* sub-tabs contains 2 functionalities:

- *Seasonality Tests*
- *CanovaHansen*

But no direct-indirect analysis.

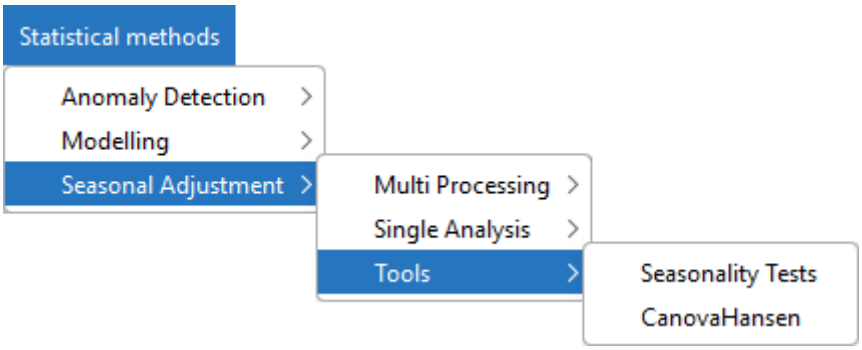


Figure 135: **The *Seasonal adjustment* tab in v3**

In v2, the *Tools* sub-tabs contains 2 functionalities:

- *Seasonality Tests*

- *Direct-Indirect Seasonal Adjustment*

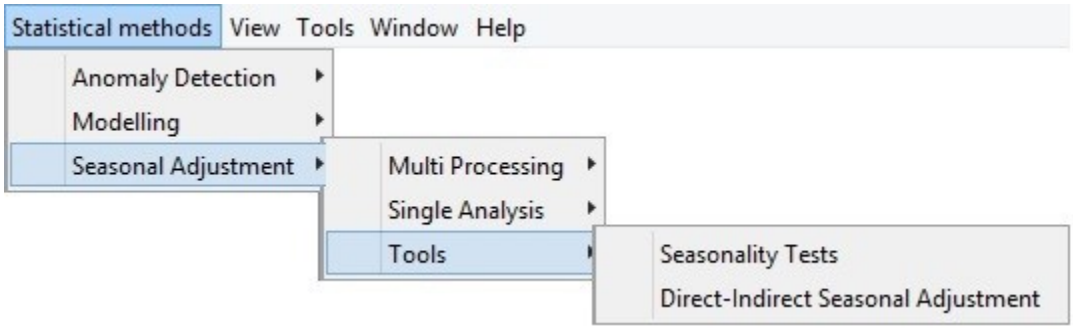


Figure 136: **The *Seasonal adjustment* tab in v2**

In v3, the *Tools* sub-tabs contains 2 functionalities:

- *Seasonality Tests*
- *CanovaHansen*

But no direct-indirect analysis.

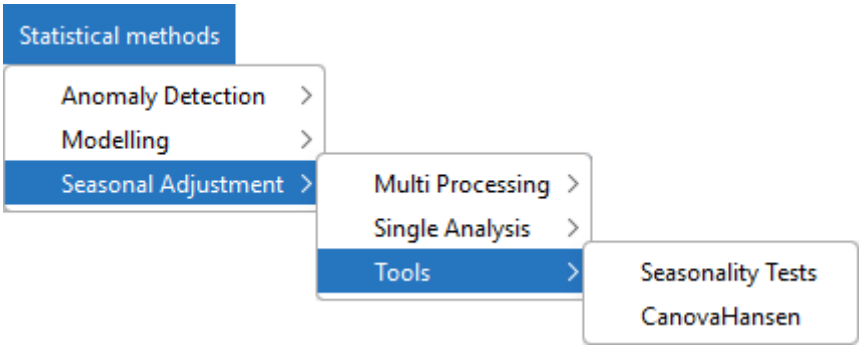


Figure 137: **The *Seasonal adjustment* tab in v3**

Tools menu

The following functionalities are available from the *Tools* menu:

- *Container* – includes several tools for displaying data in a time domain;
- *Spectral analysis* – contains tools for the analysis of a time series in a frequency domain;
- *Aggregation* – enables the user to investigate a graph of the sum of multiple time series;

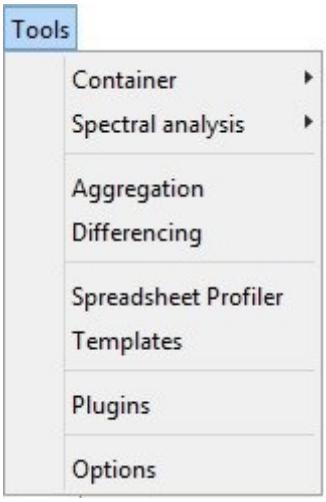


Figure 138: **The *Tools* menu**

- *Differencing* – allows for the inspection of the first regular differences of the time series;
- *Spreadsheet profiler* – offers an Excel-type view of the `.xls` file imported to JDemetra+.
- *Plugins* – allows for the installation and activation of plugins, which extend JDemetra+ functionalities.
- *Options* – presents the default interface settings and allows for their modification.

Container

Container includes basic tools to display the data. The following items are available: *Chart*, *Grid*, *Growth Chart* and *List*.

detailed in data visualization part (link to set up)

Spectral analysis

The *Spectral analysis* section provides three spectral graphs that allows an in-depth analysis of a time series in the frequency domain. These graphs are the *Auto-regressive Spectrum*, the *Periodogram* and the *Tukey Spectrum*.

For more information the user may refer to the spectral analysis chapter and to the sprectral graphs section.

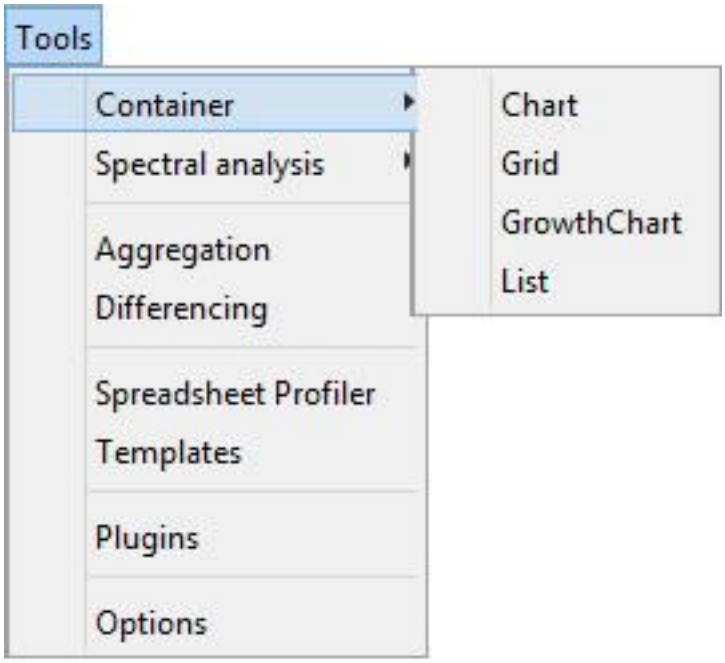


Figure 139: **The *Container* menu**

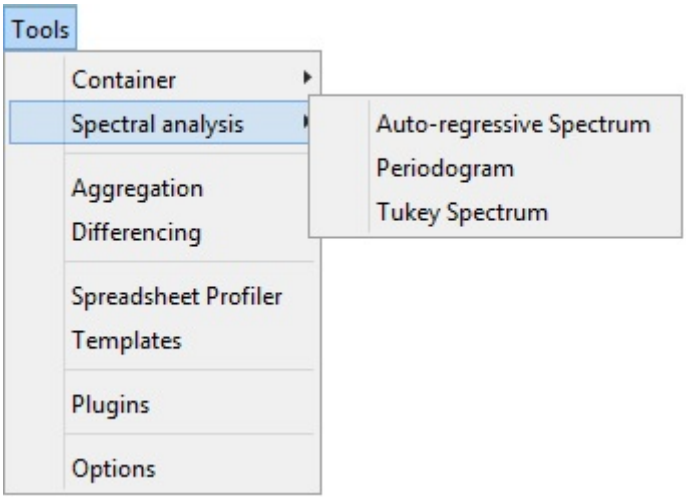


Figure 140: **Tools for spectral analysis**

Aggregation

Aggregation calculates the sum of the selected series and provides basic information about the selected time series, including the start and end date, the number of observations and a sketch of the data graph.

link to data visu chap

Differencing

The *Differencing* window displays the first regular differences for the selected time series together with the corresponding periodogram and the PACF function.

link to data visu chap

Spreadsheet profiler

The *Spreadsheet profiler* offers an Excel-type view of the `.xls` file imported to JDemetra+. To use this functionality drag the file name from the *Providers* window and drop it to the empty *Spreadsheet profiler* window.

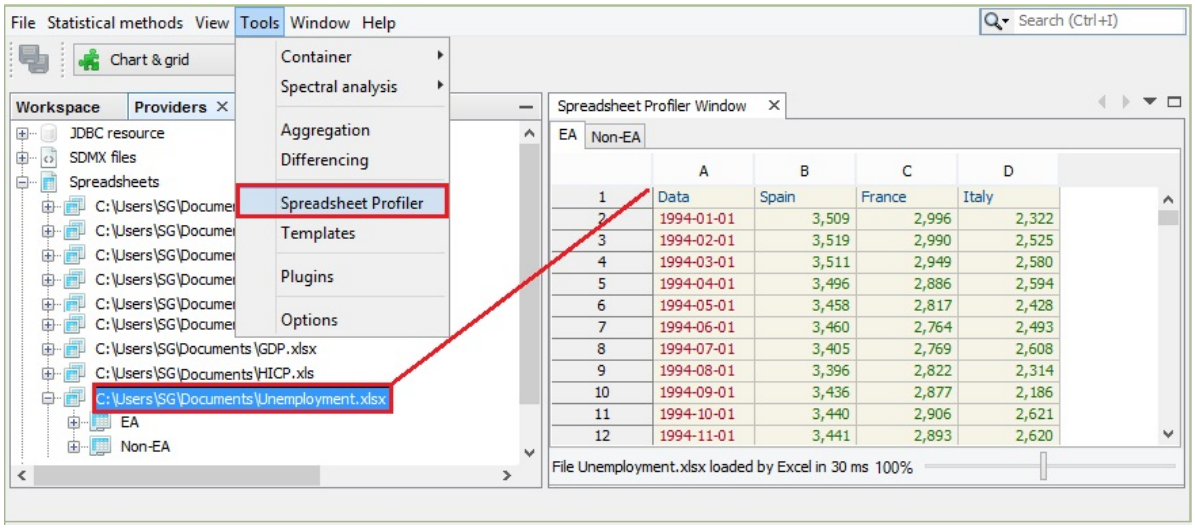


Figure 141: **The *Spreadsheet Profiler* window**

Plugins

Installation an functionalities of plugins are described in the related chapter.

View

The View menu contains functionalities that enable the user to modify how JDemetra+ is viewed. It offers the following items:

- **Split** – the function is not operational in the current version of the software.
- **Toolbars** – displays selected toolbars under the main menu. The *File* toolbar contains the *Save all* icon. The *Performance* toolbar includes two icons: one to show the performance of the application, the other to stop the application profiling and taking a snapshot. The *Other* toolbar determines the default behaviour of the program when the user double clicks on the data. It may be useful to plot the data, visualise it on a grid, or to perform any pre-specified action, e.g. execute a seasonal adjustment procedure.
- **Show Only Editor** – displays only the *Results* panel and hides other windows (e.g. *Workspace* and *Providers*).
- **Full Screen** – displays the current JDemetra+ view in full screen.

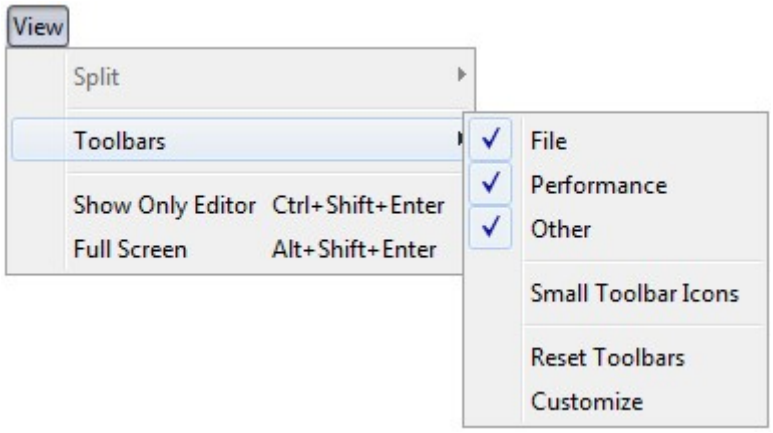


Figure 142: **The *View* menu**

Window menu

The *Window* menu offers several functions that facilitate the analysis of data and enables the user to adjust the interface view to the user's needs.

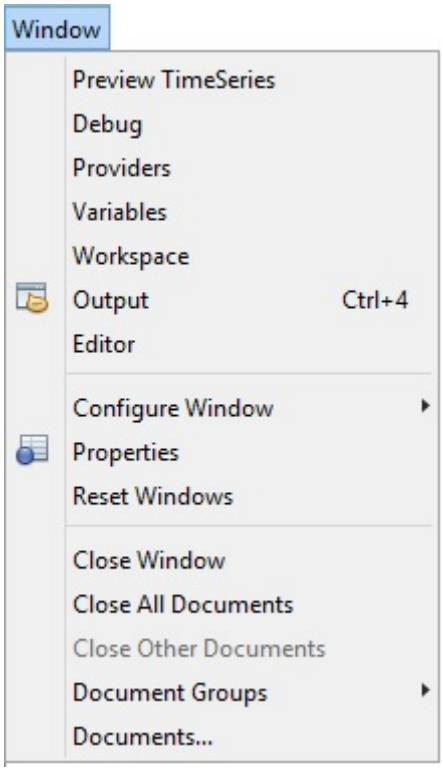


Figure 143: **The *Window* menu**

- **Preview Time Series** – opens a window that plots any of the series the user selects from *Providers*.
- **Debug** – opens a *Preview Time Series* window that enables a fast display of the graphs for time series from a large dataset. To display the graph click on the series in the *Providers* window.
- **Providers** – opens (if closed) and activates the *Providers* window.
- **Variables** – opens (if closed) and activates the *Variable* window.
- **Workspace** – opens (if closed) and activates the *Workspace* window.
- **Output** – a generic window to display outputs in the form of text; useful with certain plug-ins (e.g. tutorial descriptive statistics).

- **Editor** – activates the editor panel (and update the main menu consequently).
- **Configure Window** – enables the user to change the way that the window is displayed (maximise, float, float group, minimise, minimise group). This option is active when some window is displayed in the JD+ interface.
- **Properties** – opens the *Properties* window and displays the properties of the marked item (e.g. time series, data source).
- **Reset Windows** – restores the default JDemetra+ view.
- **Close Window** – closes all windows that are open.
- **Close All Documents** – closes all documents that are open.
- **Close Other Documents** – closes all open documents except for the one that is active (which is the last activated one).
- **Document Groups** – enables the user to create and manage document groups.
- **Documents** – lists all active documents.

Help menu

All TS&view

Search option

Options

The *Options* window includes five main panels:

- *Demetra*,
- *General*,
- *Keymap*,
- *Appearance*
- and *Miscellaneous*.

They are visible in the very top of the *Options* window.

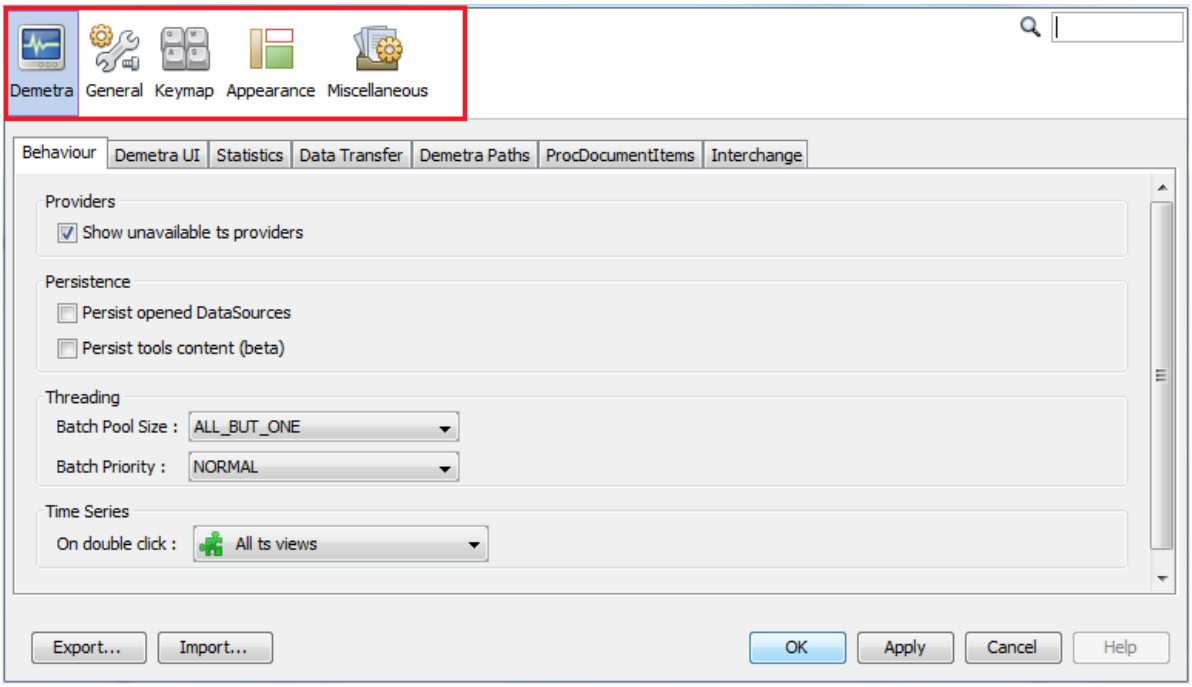


Figure 144: **Main sections of the *Options* window**

Demetra panel

0.0.0.1 v2

By default, the *Demetra* panel is shown. It is divided into seven tabs:

- *Behaviour*,
- *Demetra UI*,
- *Statistics*,
- *Data transfer*,
- *Demetra Paths*,
- *ProcDocumentItems*,
- and *Interchange*.

0.0.0.2 v3

By default, the *Demetra* panel is shown. It is divided into three tabs::

- *Common UI*,
- *Behaviour*,

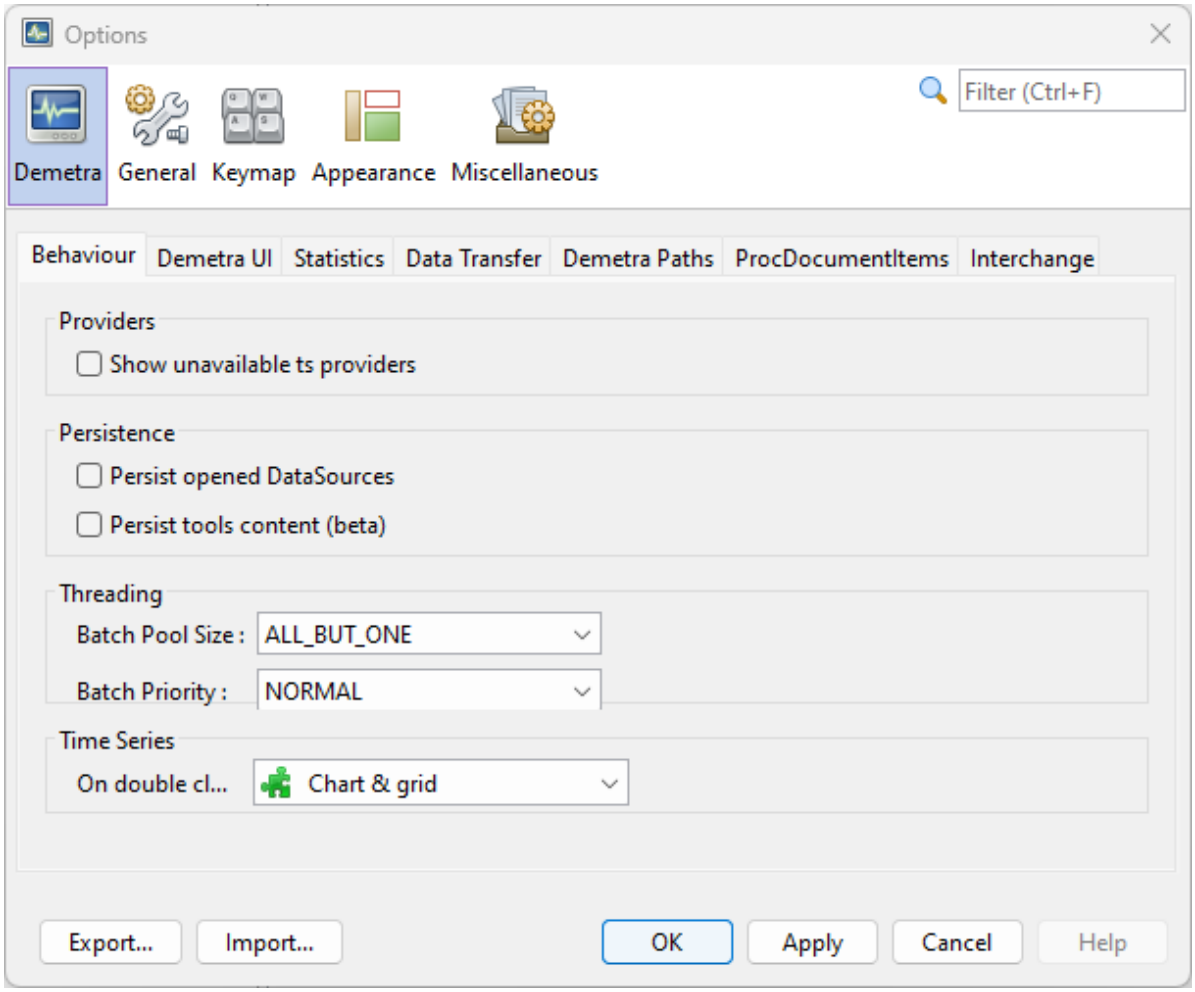


Figure 145: **Demetra panel in v2**

- and *Demetra Paths*.

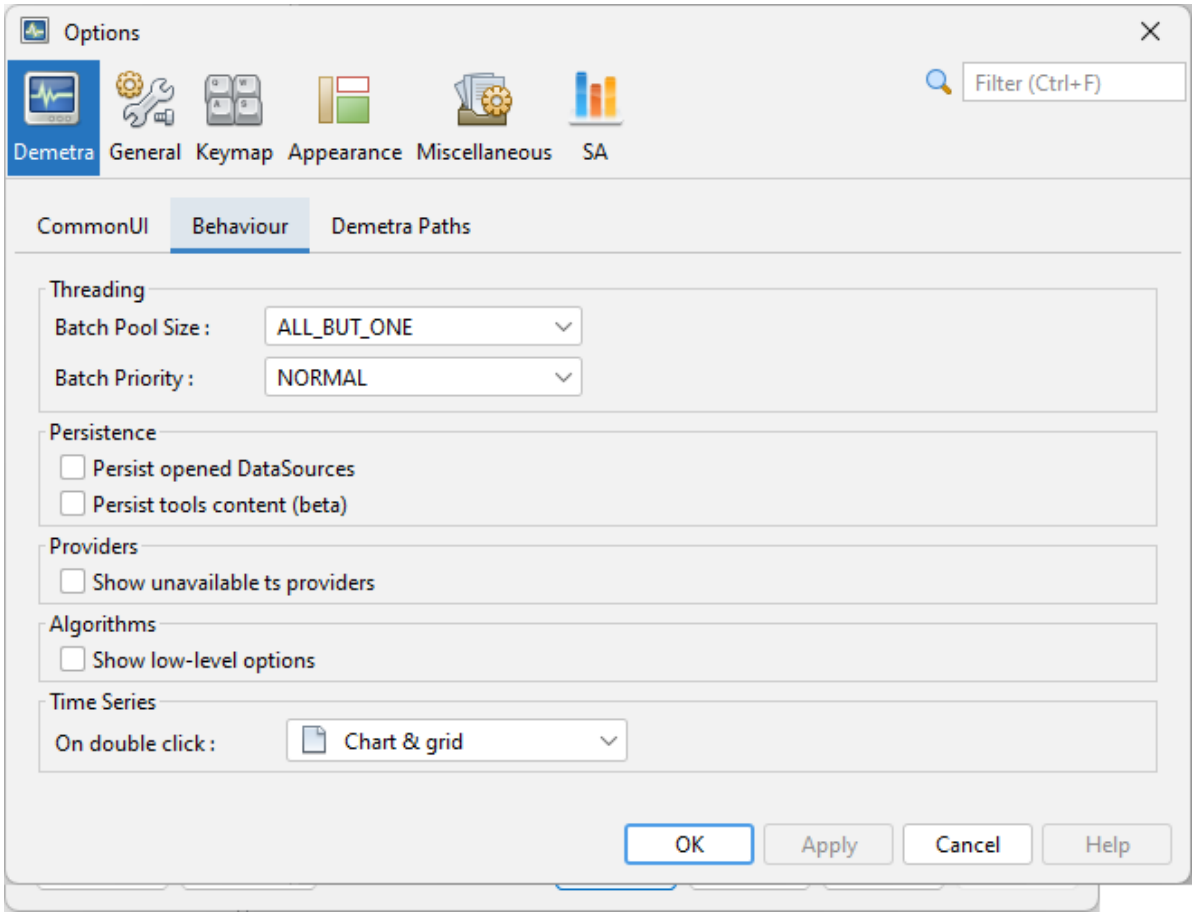


Figure 146: **Demetra panel in v3**

By default, in v2, the *Demetra* panel is shown. It is divided into seven tabs:

- *Behaviour*,
- *Demetra UI*,
- *Statistics*,
- *Data transfer*,
- *Demetra Paths*,
- *ProcDocumentItems*,
- and *Interchange*.

In v3, you will just find three tabs:

- *Common UI*,
- *Behaviour*,

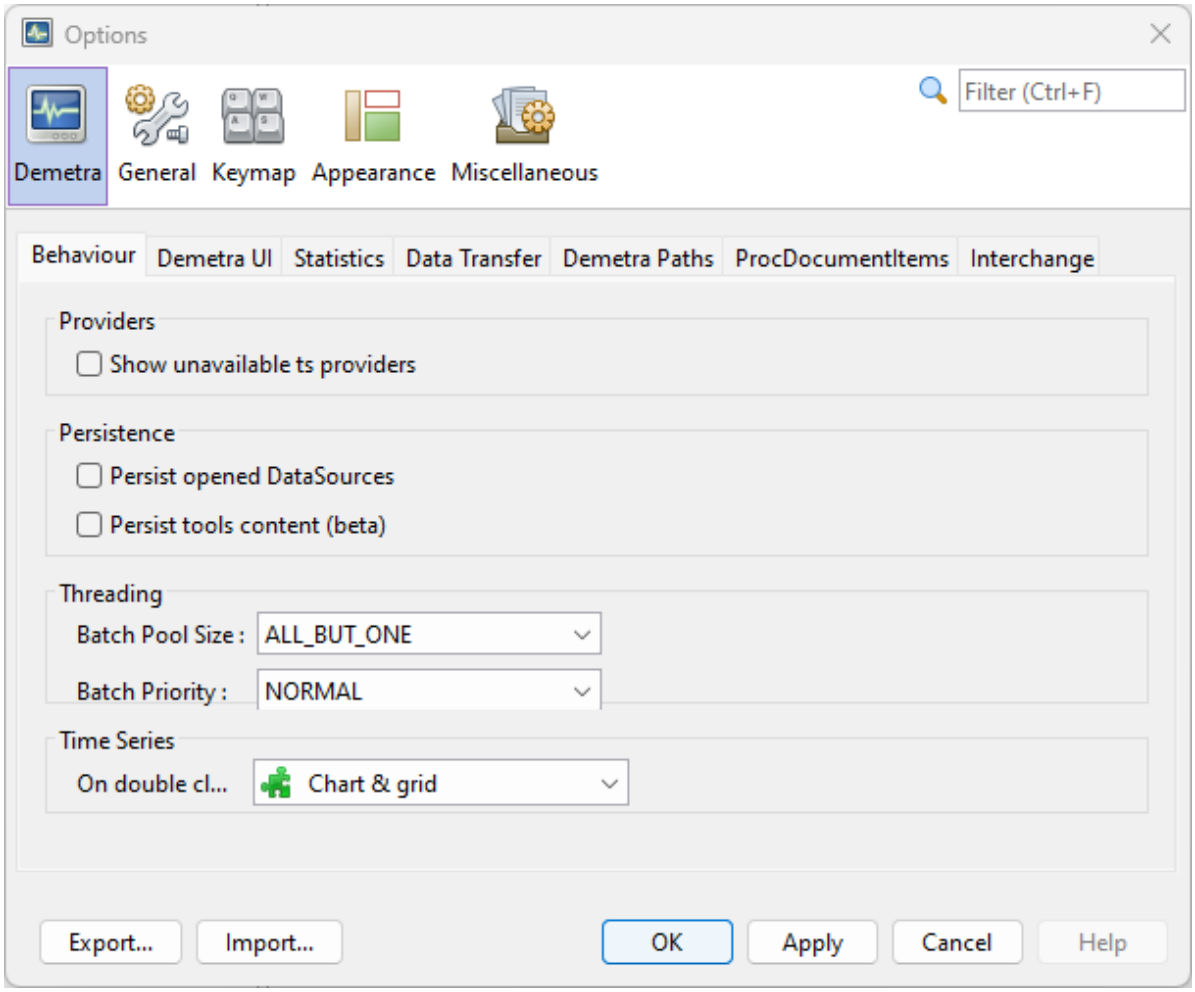


Figure 147: **Demetra panel in v2**

- and *Demetra Paths*.

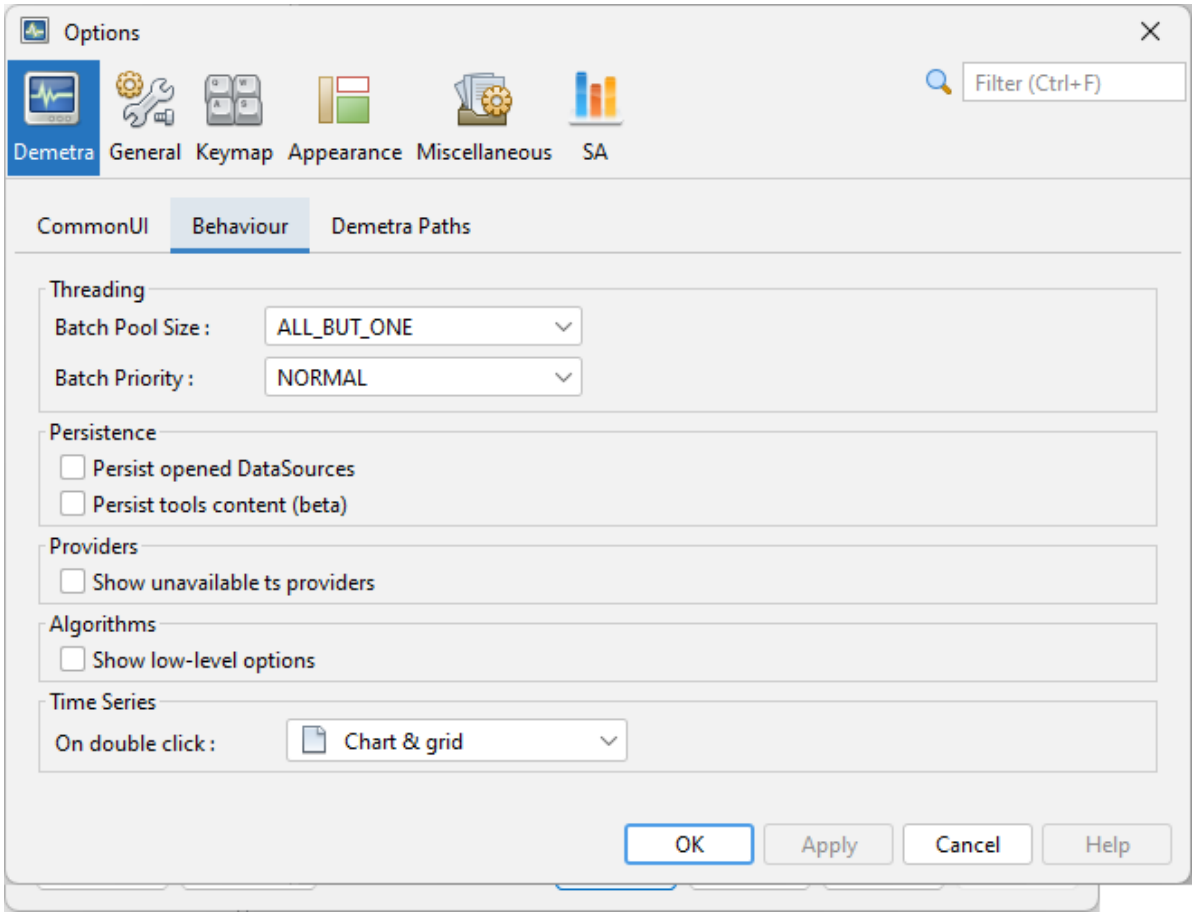


Figure 148: **Demetra panel in v3**

Behaviour tab

The tab *Behaviour* defines the default reaction of JDemetra+ to some of the actions performed by the user.

- **Providers** – an option to show only the data providers that are currently available.

- **Persistence** – an option to restore the data sources after re-starting the application so that there is no need to fetch them again (**Persist opened DataSources**) and an option to restore all the content of the chart and grid tools (**Persist tools content**).

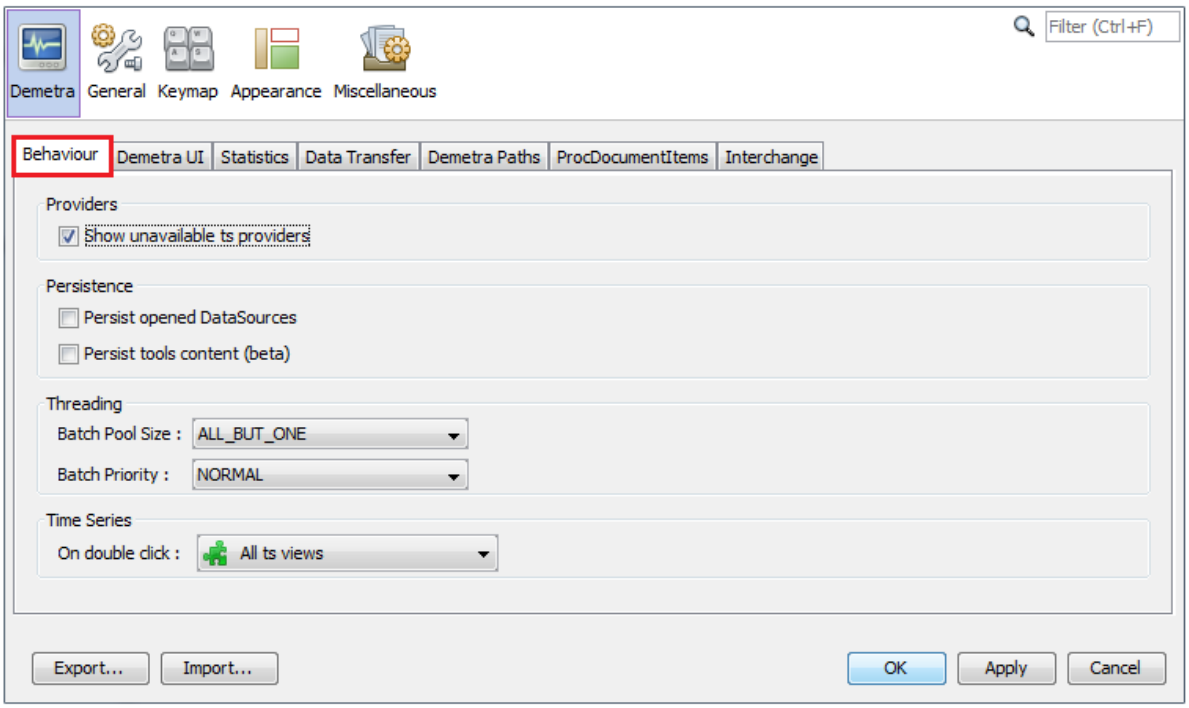


Figure 149: **The content of the *Behavior* tab**

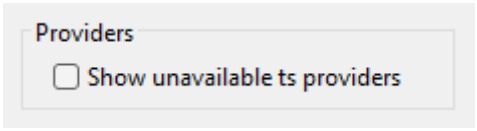


Figure 150: **The option Providers**

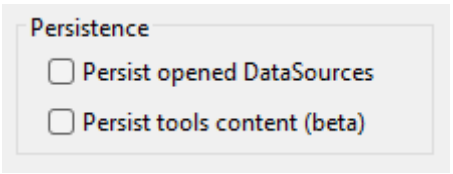


Figure 151: **The option Persistence**

- **Threading** – defines how resources are allocated to the computation (**Batch Pool Size** controls the number of cores used in parallel computation and **Batch Priority** defines the priority of computation over other processes). Changing these values might improve computation speed but also reduce user interface responsiveness.

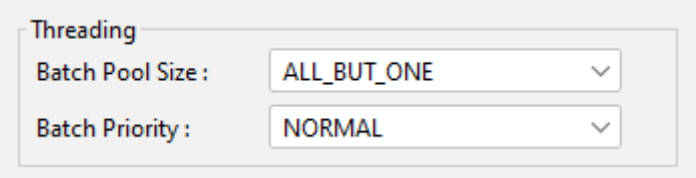


Figure 152: **The option Threading**

- **Time Series** – determines the default behaviour of the program when the user double clicks on the data. It may be useful to plot the data, visualise it on a grid, or to perform any pre-specified action, e.g. execute a seasonal adjustment procedure.

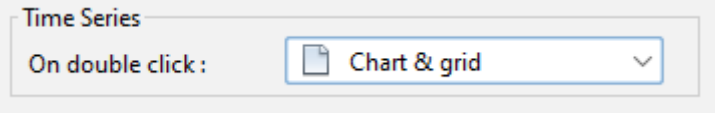


Figure 153: **The option Time Series**

> **i** Low-level options
>
> In v3, the option **Show low-level options** unable the user to access more settings in the specification.

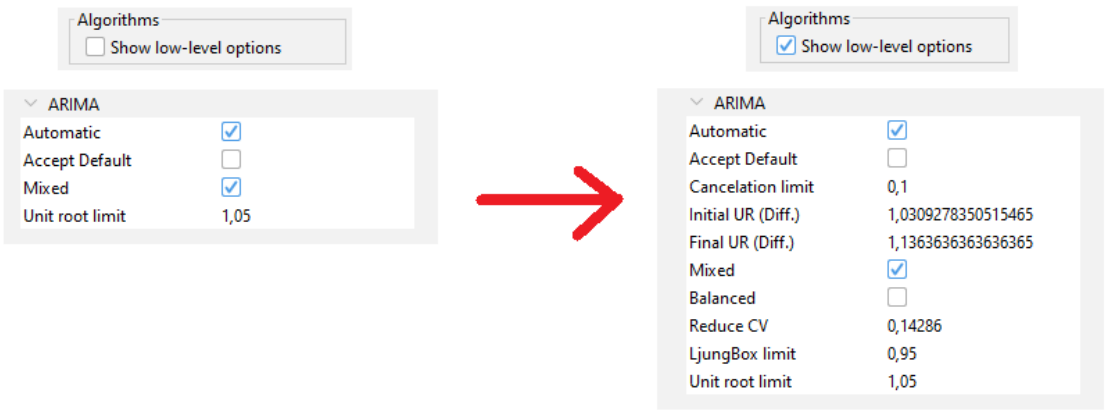


Figure 154: **Low level options on ARIMA specification**

Demetra UI / CommonUI tab

0.0.0.3 v2

In v2, this panel is called **Demetra UI**.

0.0.0.4 v3

In v3, this panel is called **CommonUI**.

In v2, this panel is called **Demetra UI**.

In v3, this panel is called **CommonUI**.

The *Demetra UI* tab enables the setting of:

- A default colour scheme for the graphs (**Color scheme**).
- The data format (uses MS Excel conventions). For example, ###,###.#### implies the numbers in the tables and the y-axis of the graphs will be rounded up to four decimals after the decimal point (**Data format** (or **Observation format** in v3)).
- The default number of last years of the time series displayed in charts representing growth rates (**Growth rates**).

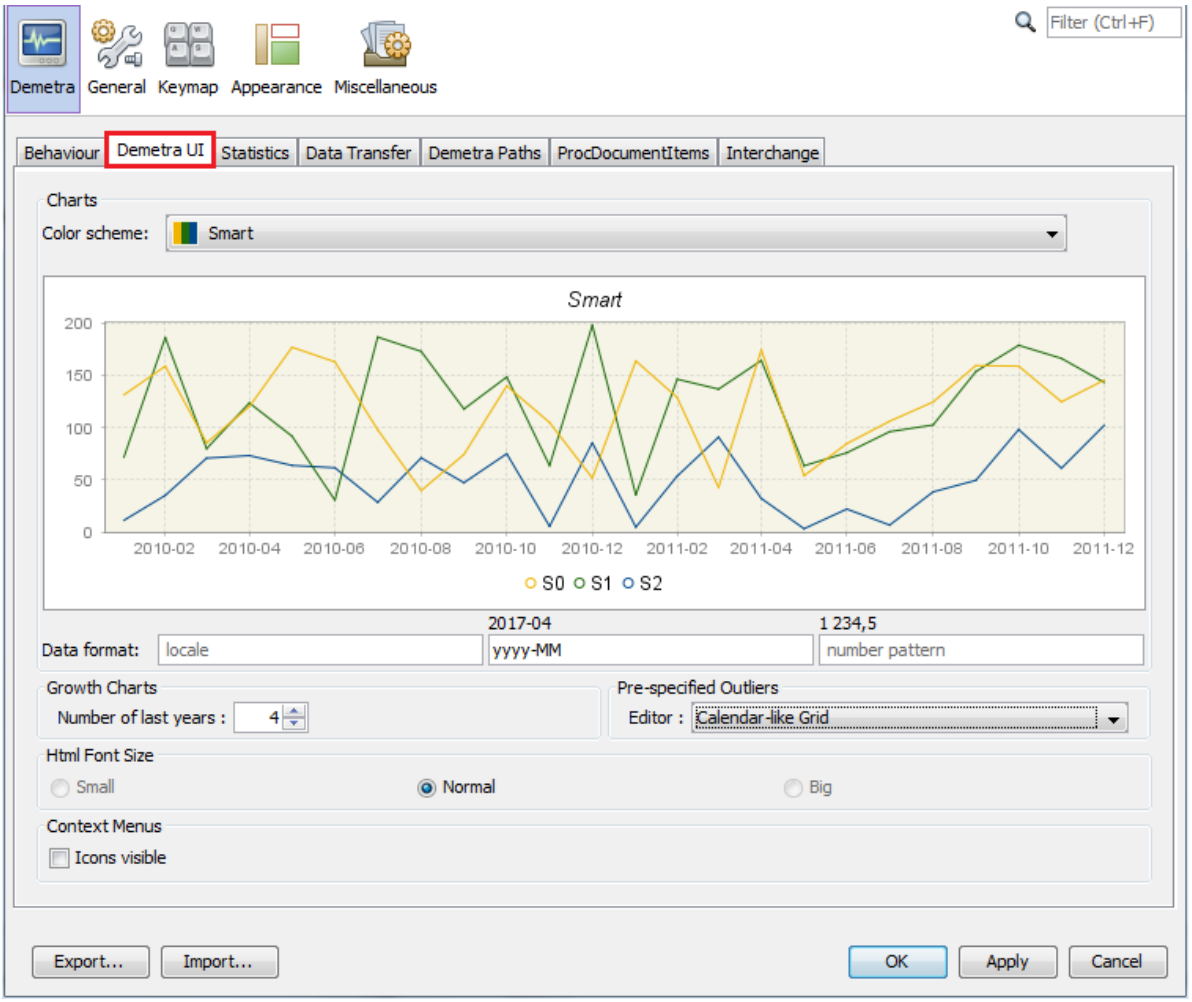


Figure 155: **The content of the *Demetra UI* tab**

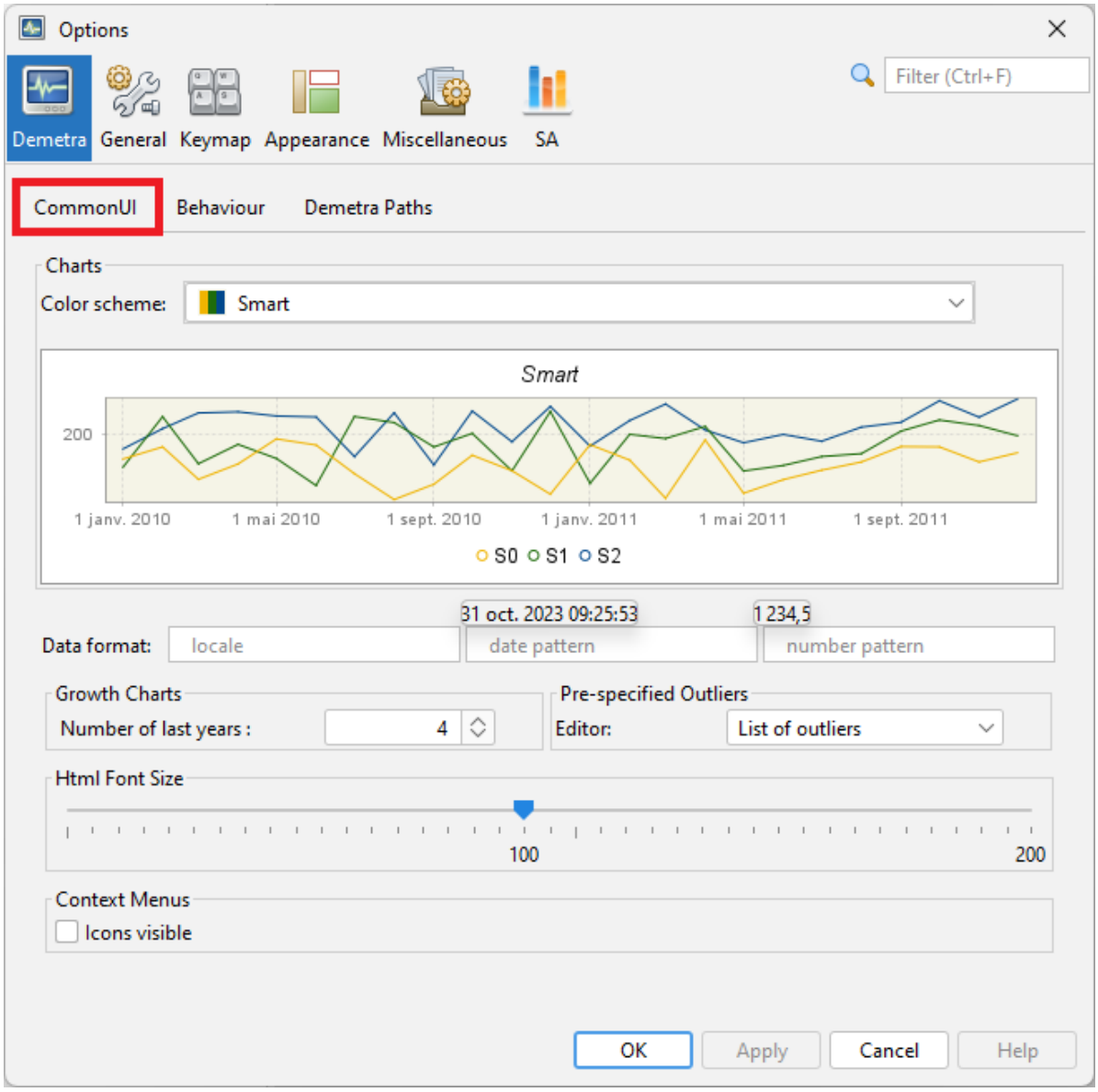


Figure 156: **CommonUI tab in v3**

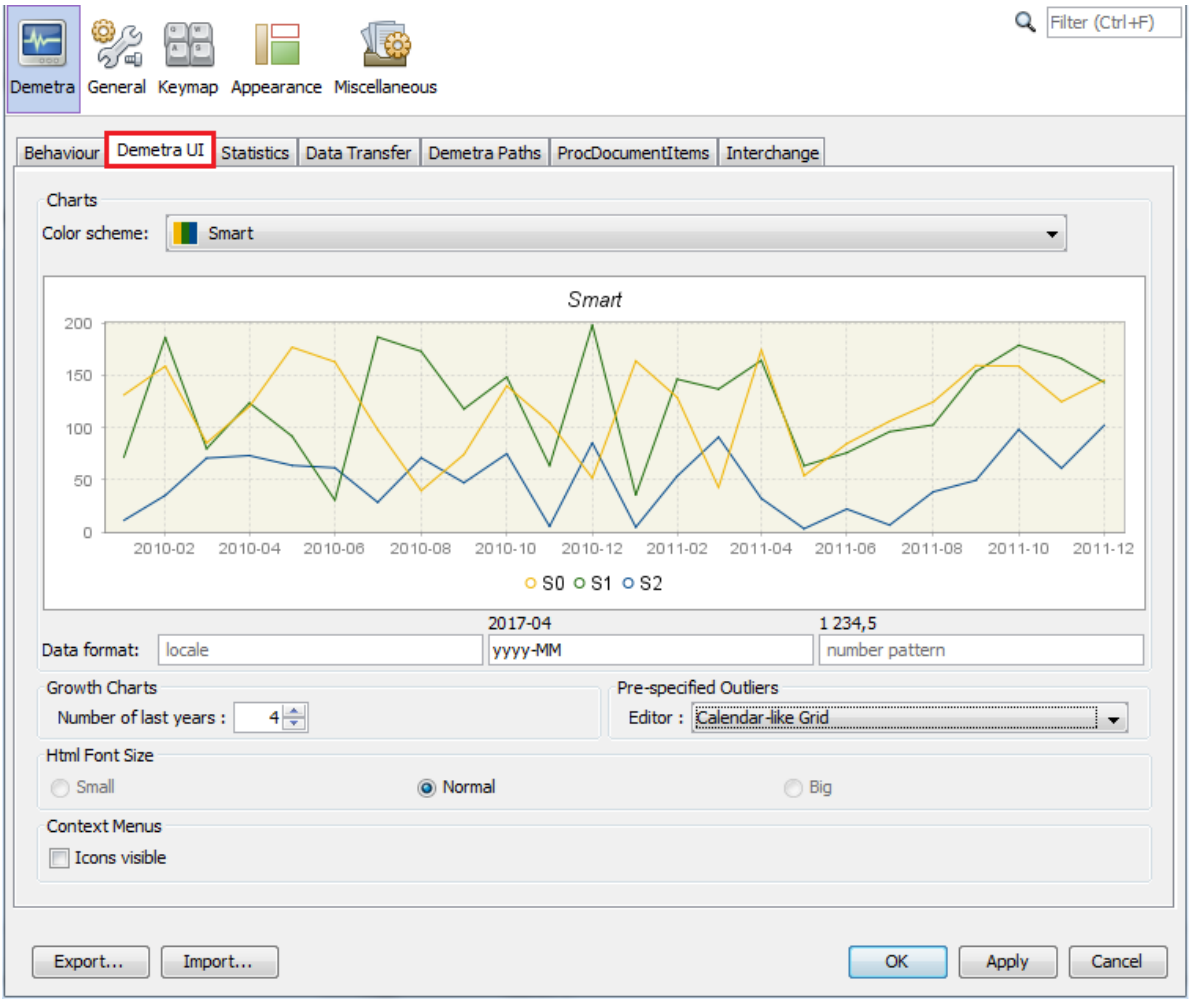


Figure 157: **The content of the *Demetra UI* tab in v2**

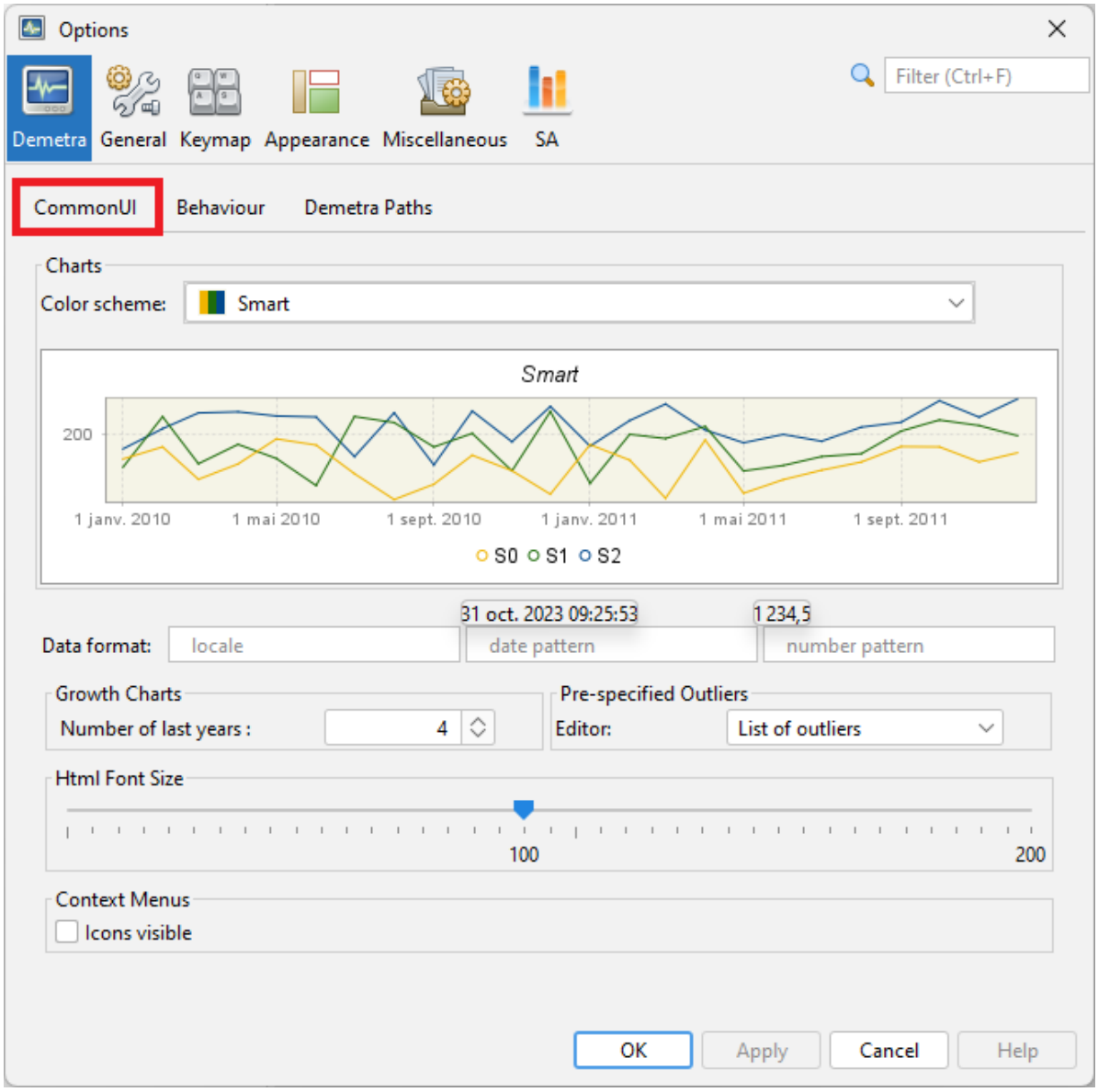


Figure 158: **CommonUI tab in v3**

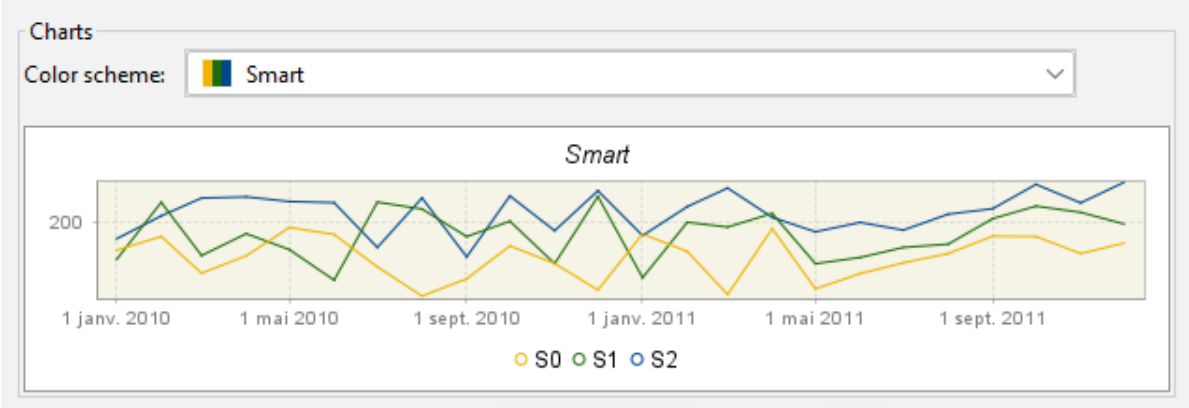


Figure 159: **The option Charts**

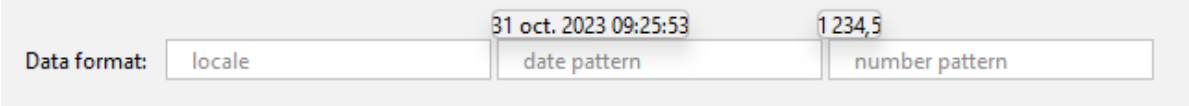


Figure 160: **The option Data format**

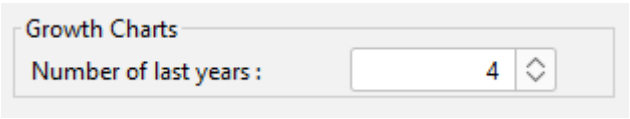


Figure 161: **The option Growth rates**

- The control of the view of the window for adding pre-specified outliers. (**Pre-specified Outliers**).

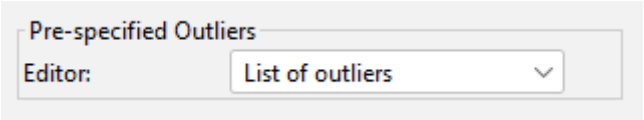


Figure 162: **The option Pre-specified Outliers**

- The visibility of the icons in the context menus (**Context Menus**).

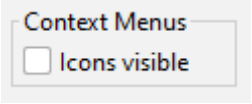


Figure 163: **The option Context Menus**

Demetra path tab

Demetra Paths allows the user to specify the relative location of the folders where the data can be found. In this way, the application can access data from different computers. Otherwise, the user would need to have access to the exact path where the data is located. To add a location, select the data provider, click the "**+**" button and specify the location.

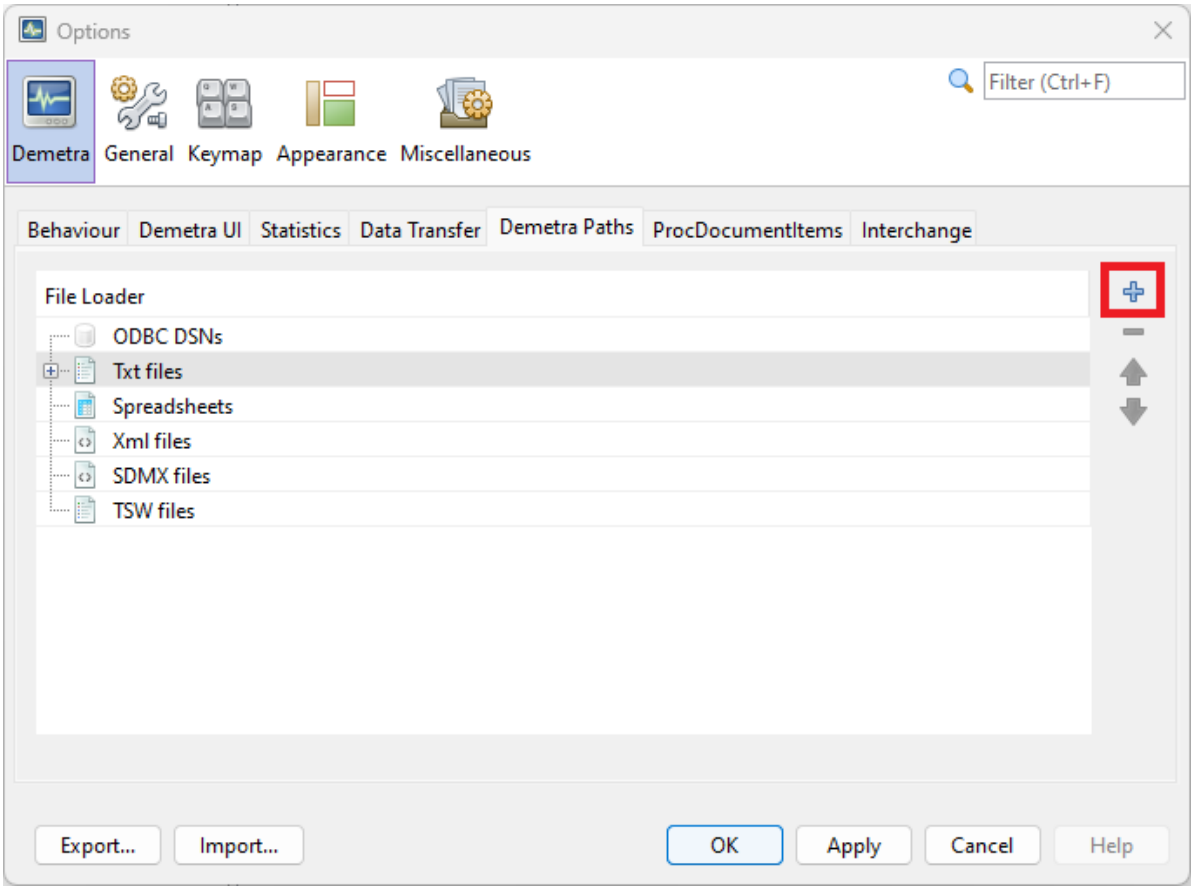


Figure 164: **The content of the *Demetra Paths* tab in v2**

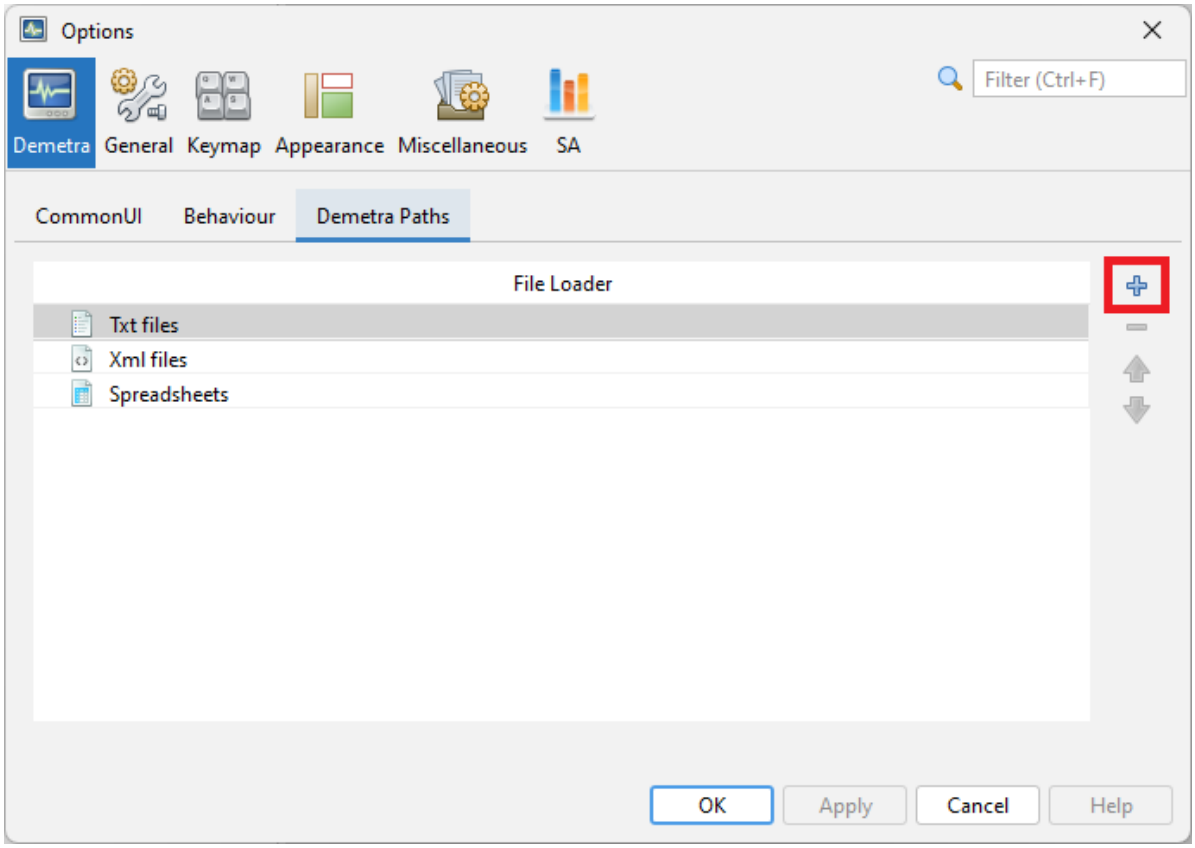
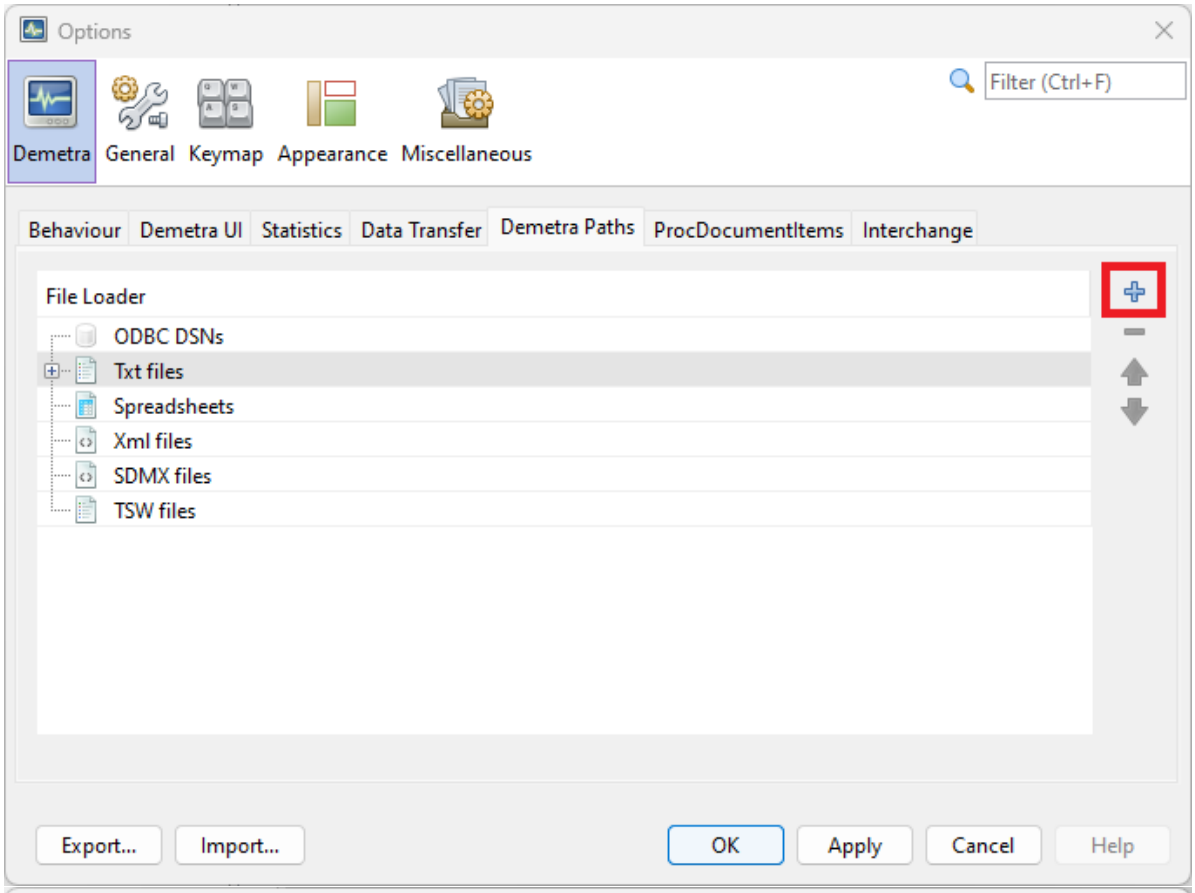
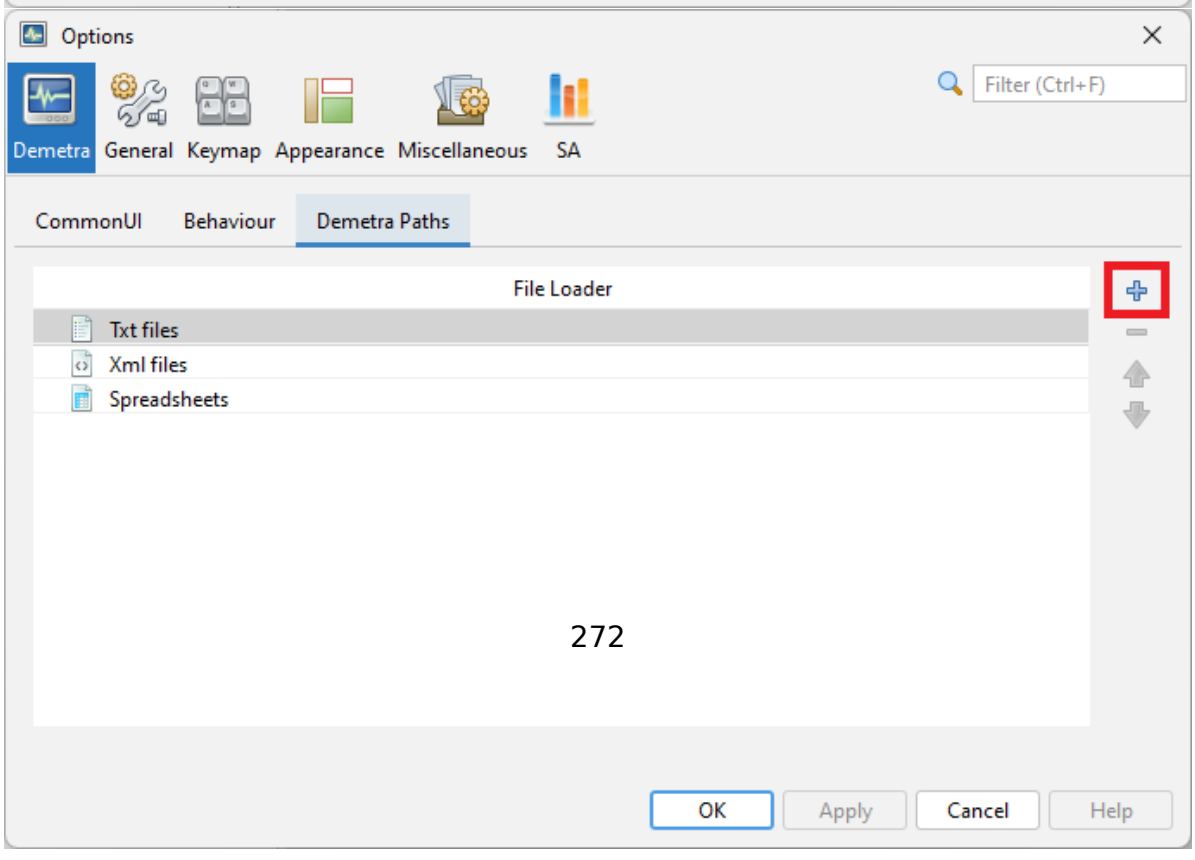


Figure 165: **The content of the *Demetra Paths* tab in v3**

0.0.0.5 v2

0.0.0.6 v3





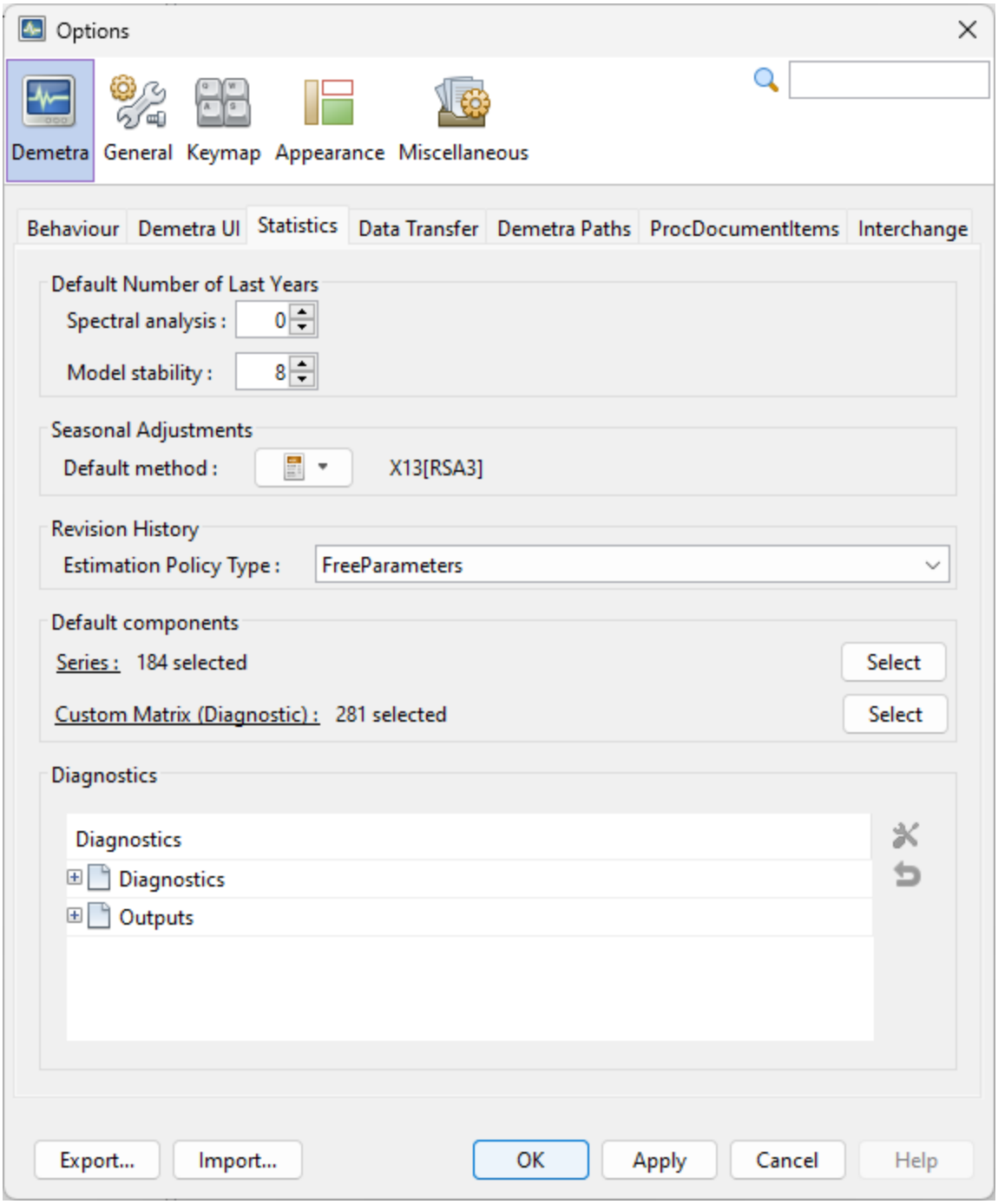


Figure 166: **Statistics tab in v2**

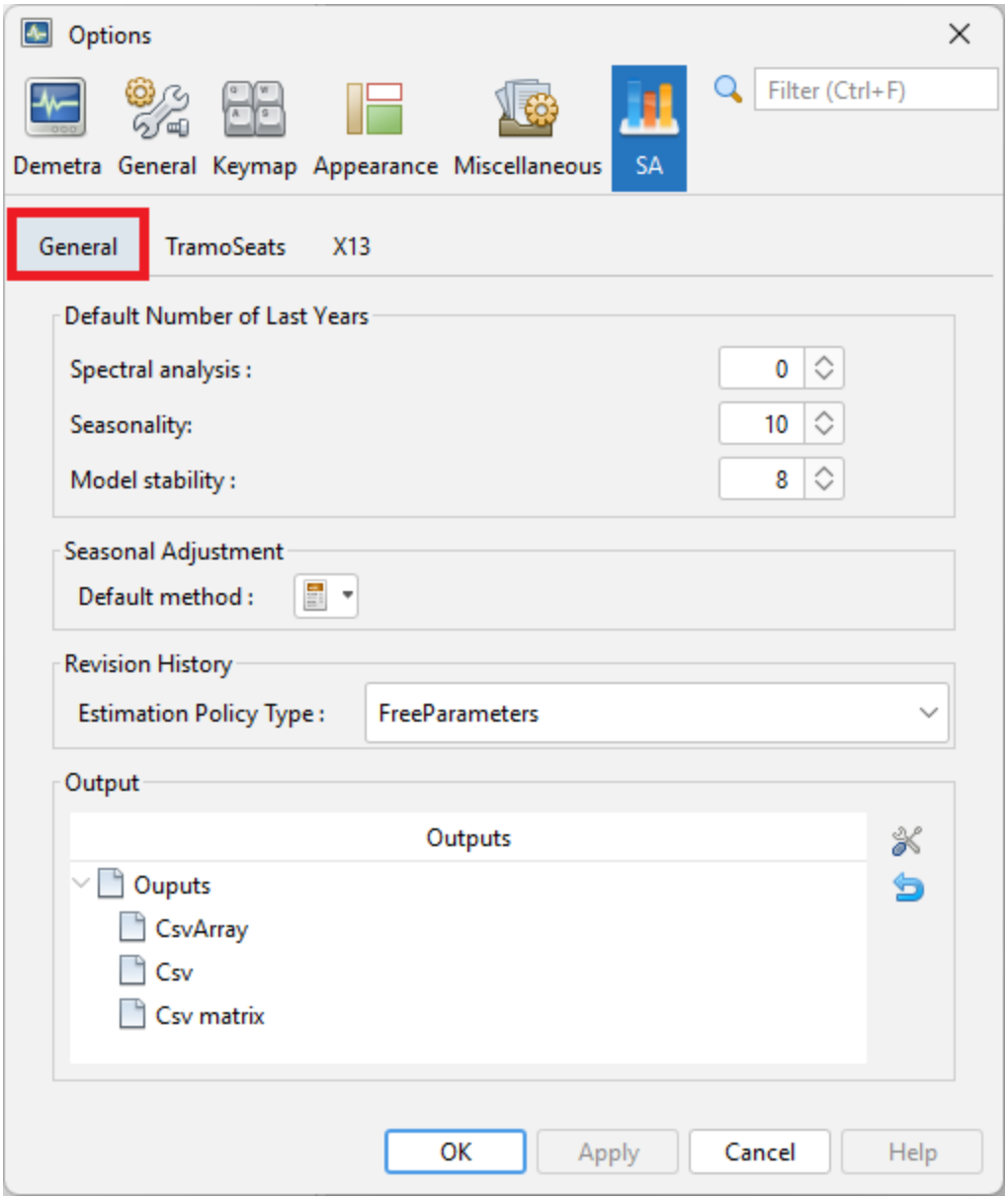
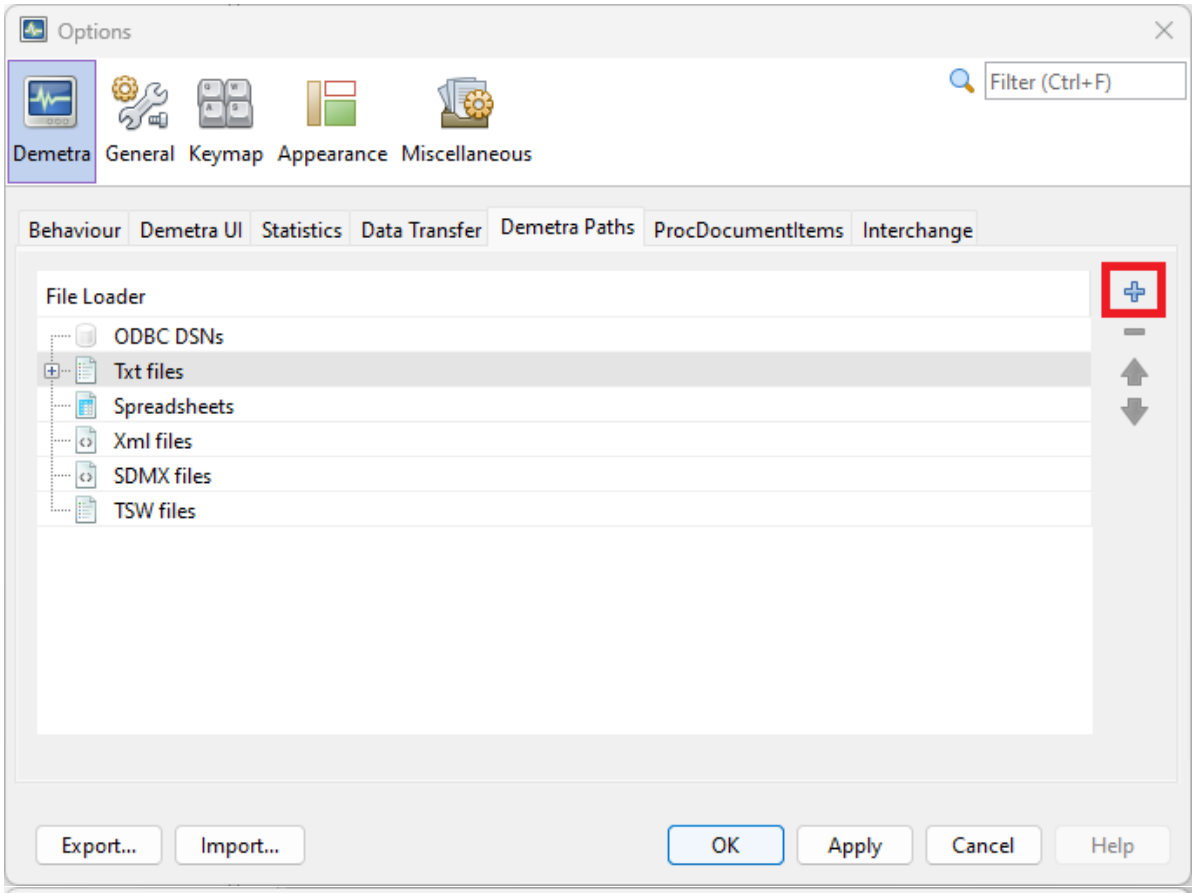


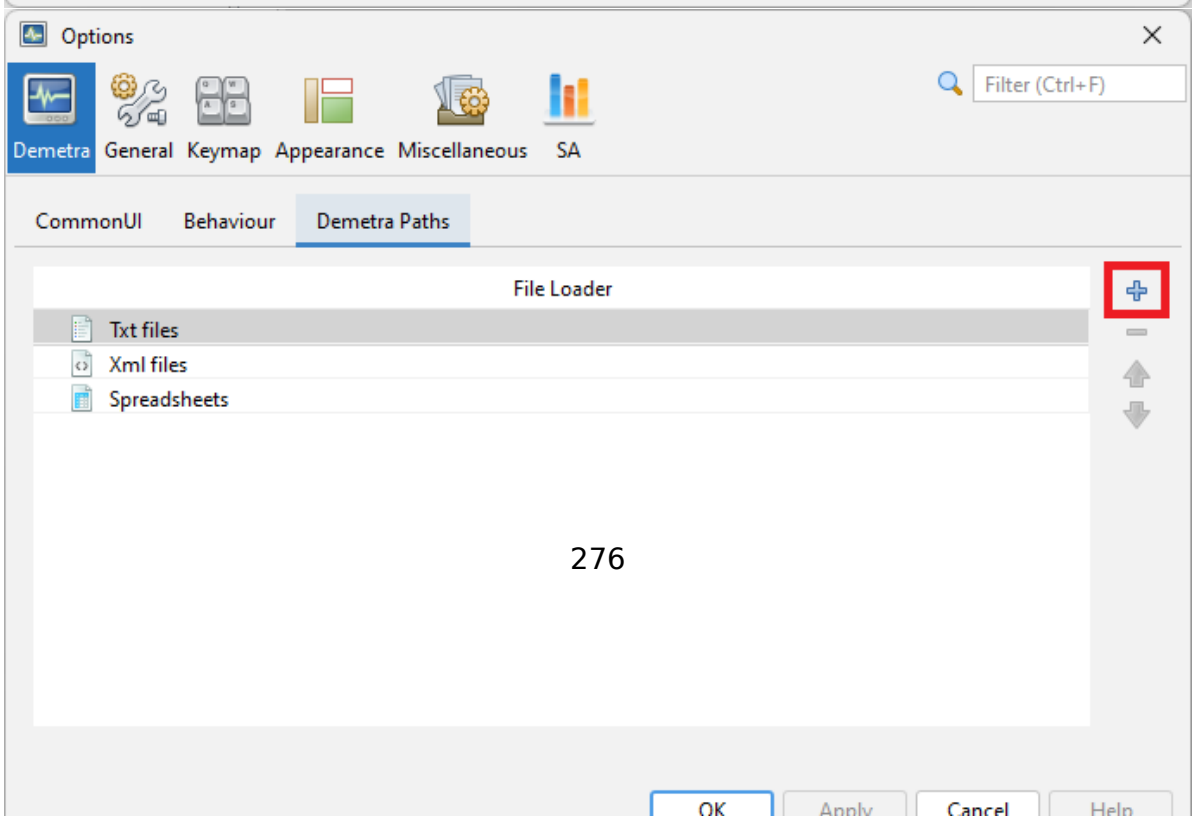
Figure 167: **SA panel**

Statistics tab

0.0.0.7 v2

0.0.0.8 v3





The *Statistics* tab includes options to control:

- The number of years used for spectral analysis and for model stability (**Default Number of Last Years**);

0.0.0.9 v2

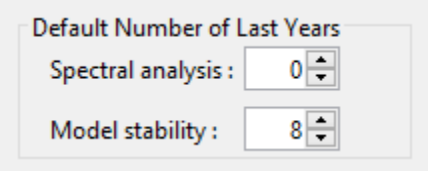


Figure 168: **Default Number of Last Years option in v2**

0.0.0.10 v3

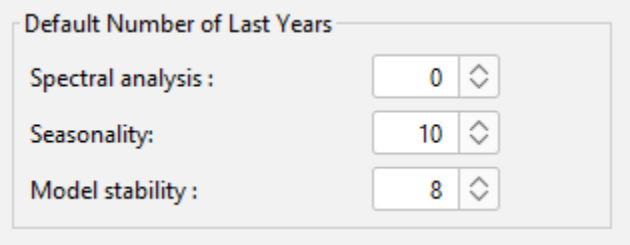
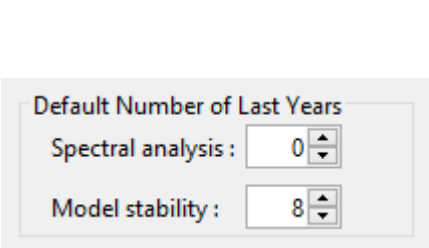
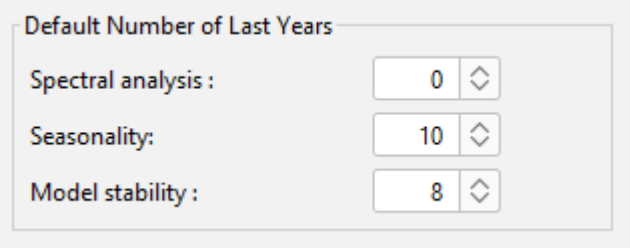


Figure 169: **Default Number of Last Years option in v3**





- The default pre-defined specification for seasonal adjustment (**Seasonal Adjustment**);
- The type of the analysis of revision history (**Revision History**):

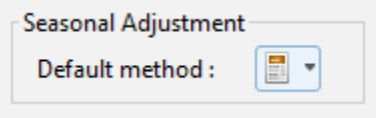


Figure 170: **Seasonal Adjustment option**

- *FreeParameters* – the Reg-ARIMA model parameters and regression coefficients of the Reg-ARIMA model will be re-estimated each time the end point of the data is changed. This argument is ignored if no Reg-ARIMA model is fit to the series.
- *Complete* – the whole Reg-ARIMA model together with regressors will be re-identified and re-estimated each time the end point of the data is changed. This argument is ignored if no Reg-ARIMA model is fitted to the series.
- *None* – the ARIMA parameters and regression coefficients of the Reg-ARIMA model will be fixed throughout the analysis at the values estimated from the entire series (or model span).

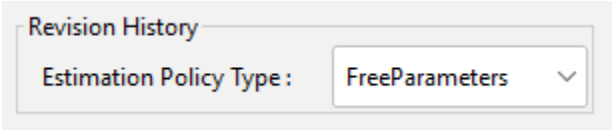


Figure 171: **Revision History option in v2**

- The settings for the quality measures and tests used in a diagnostic procedure:
  - **Default components** – a list of series and diagnostics that are displayed in the **SAProcessing** \(→\) **Output** window. The list of default items can be modified with the respective **Select** button (see figure below)

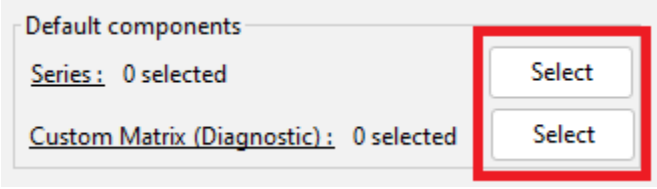


Figure 172: **The *Default components* section on the *Statistics* tab**

  - **Diagnostics** – a list of diagnostics tests, where the user can modify the default settings (see figure "*The panel for modification of the settings for the tests in the Basic checks section*" below).