JDemetra+ documentation

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

2027-08-01

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 latest release in version 3 family can be found here.

Version 3.x family provides, among other things, extended features for seasonal adjustment and trend estimation, including **high frequency data**.

The latest release in version 2 family can be found here.

The highest version using Java 8 is v2.2.4.

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 reconciliation (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

To install 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 your 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, 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 Plug-ins

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.x-bin.zip`, corresponding to the version of your Graphical User Interface.
- in v3, go to the **jdplus-main** page, **download** and **unzip** the compressed folder `jwsacruncher-standalone-3.x.y-windows-x86_64.zip`, corresponding to the version of your Graphical User Interface.

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.x.y-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

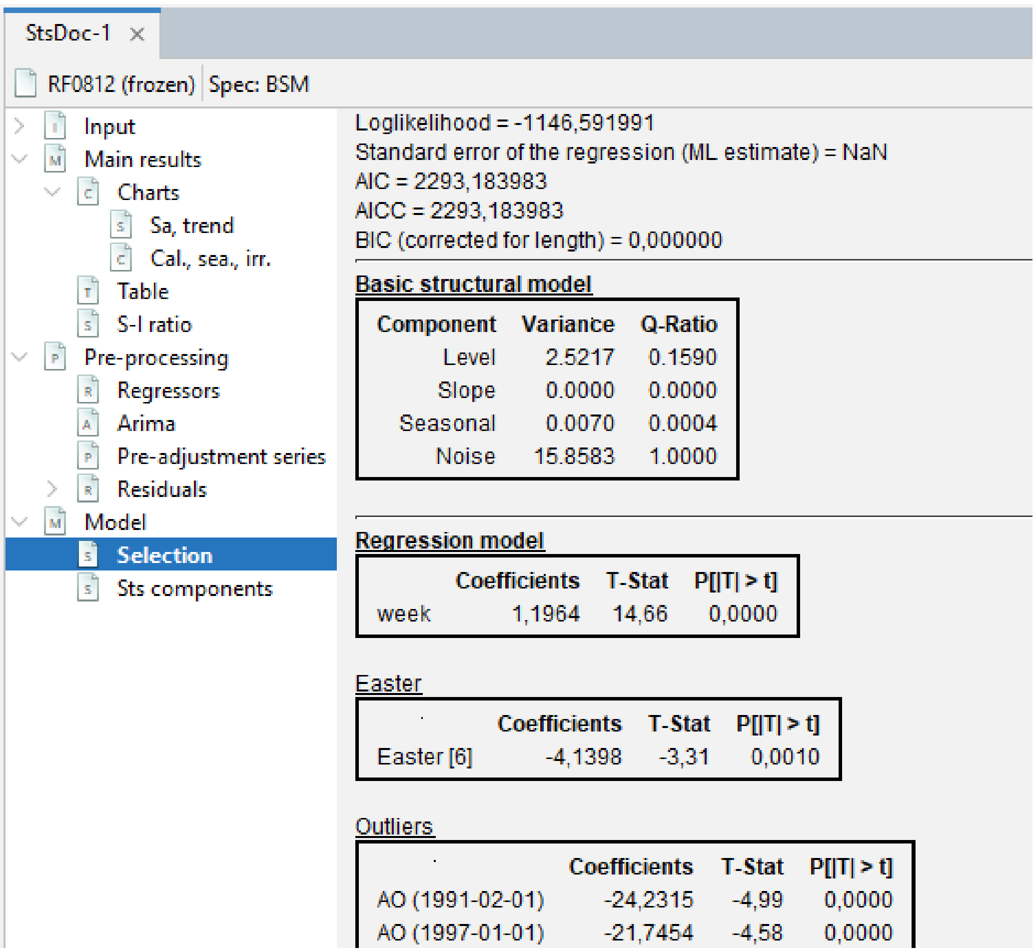


Figure 1: BSM output view in GUI

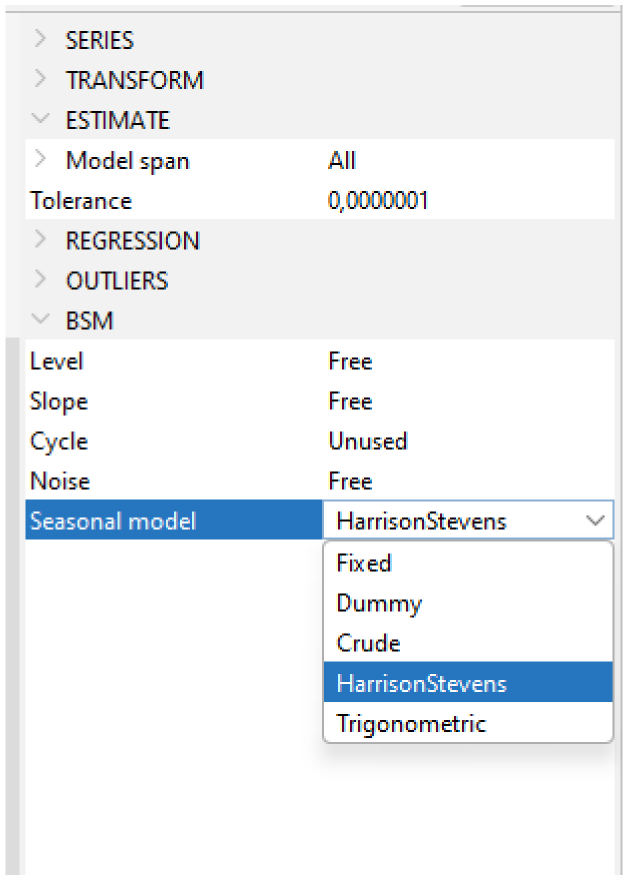


Figure 2: BSM specification box

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

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

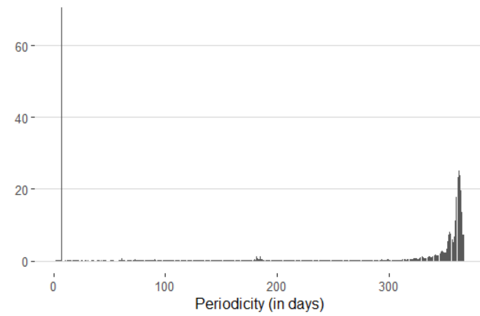


Figure 3: Canova Hansen seasonality test

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

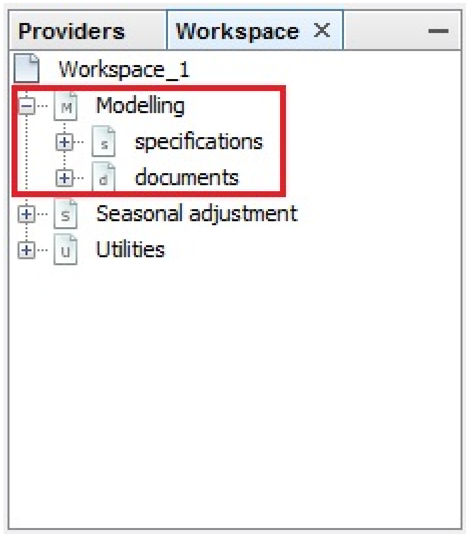


Figure 4: **The *Workspace* window with the nodes for the modelling procedure marked**

Modelling in Statistical Methods Panel

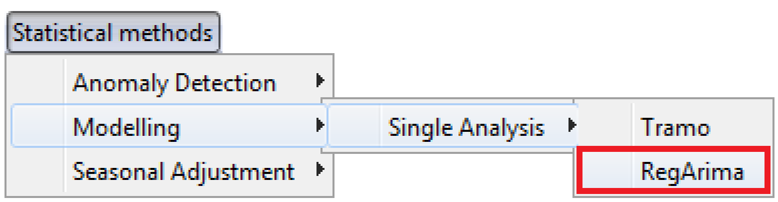


Figure 5: **The *RegARIMA* menu item**

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.

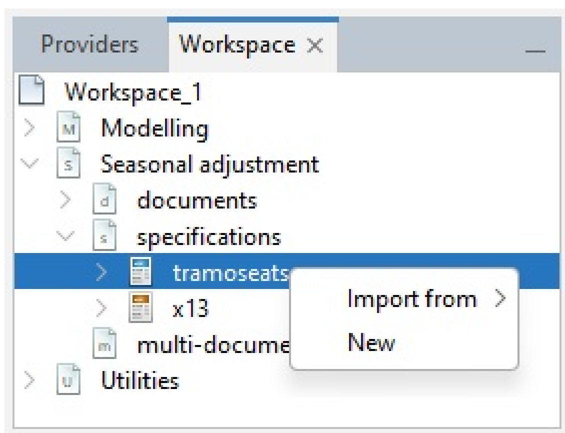


Figure 6: **Creating a new specification in the *Seasonal adjustment* section**

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

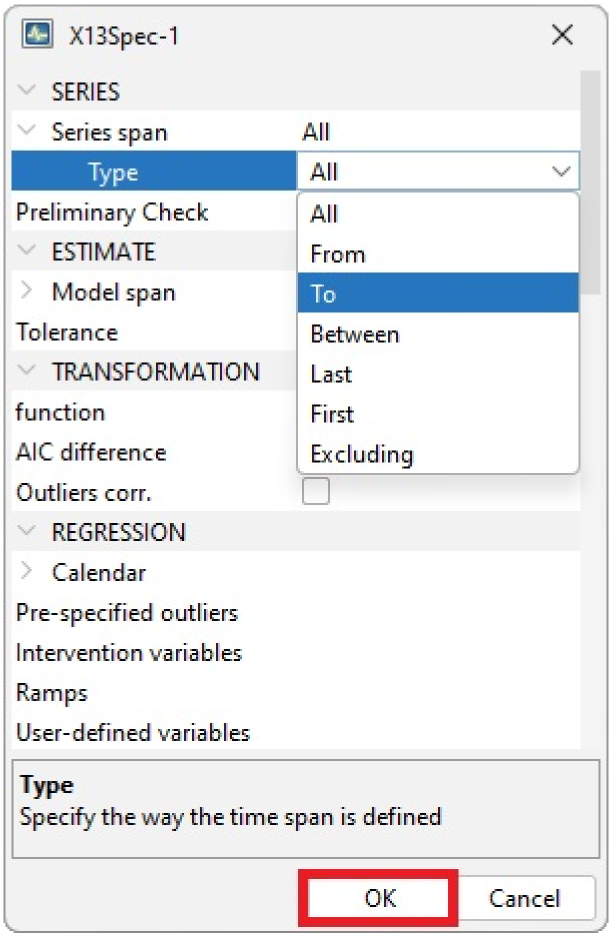


Figure 7: **Changing settings of seasonal adjustment specification**

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.

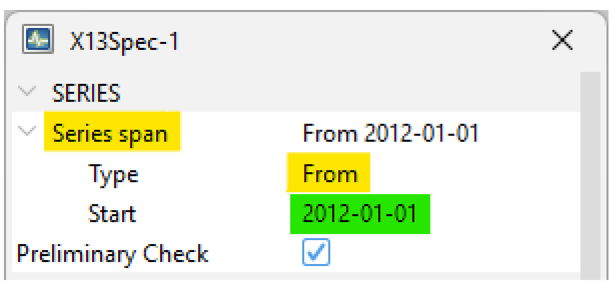


Figure 8: **Setting series span in R**

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"),
```

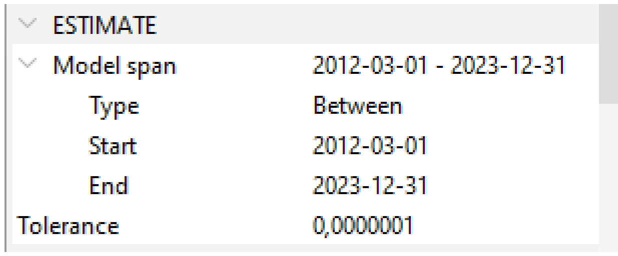


Figure 9: **Setting in R**

```
    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

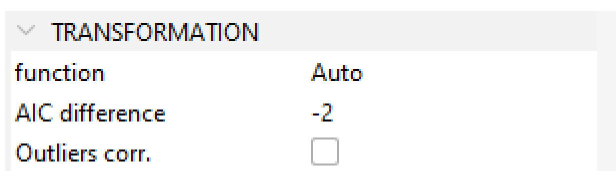


Figure 10: Model span setting

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

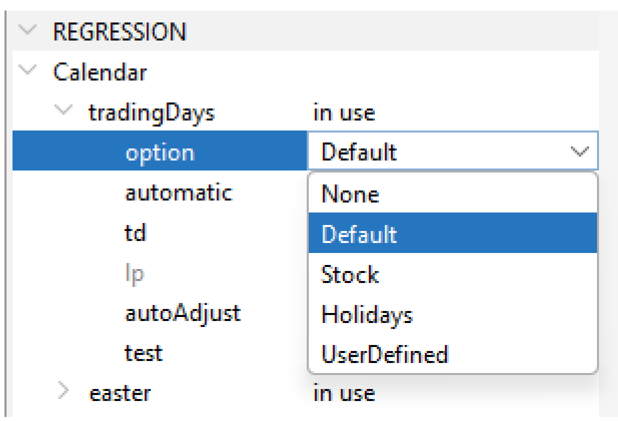


Figure 11: **STEP 1: Selection from JDemetra+ Default...or User_defined**

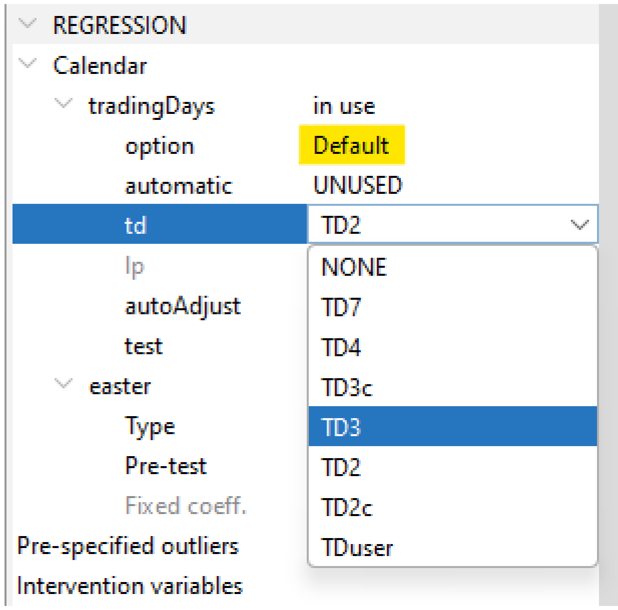


Figure 12: **STEP2: Calendar effects minus Easter are labeled *trading days***

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_
```

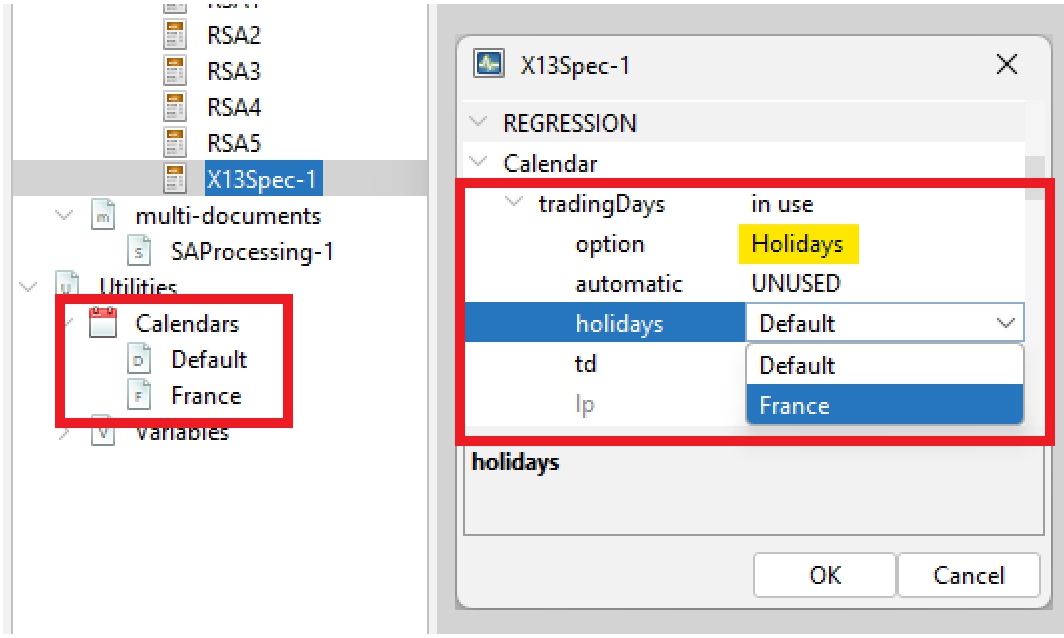


Figure 13: **The list of calendars displayed under *Holidays* option corresponds to the calendars defined in the *Workspace* window**

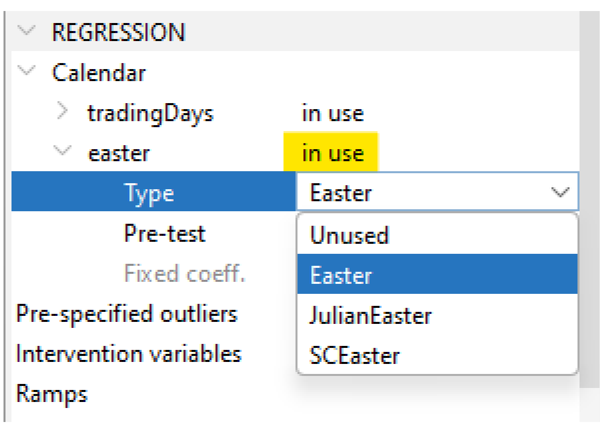


Figure 14: **easter Options**

```
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

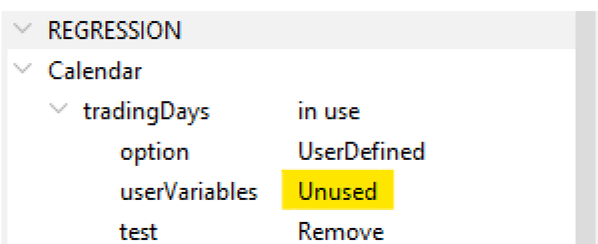
Step 1: import data set containing the regressors, general procedure explained here

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

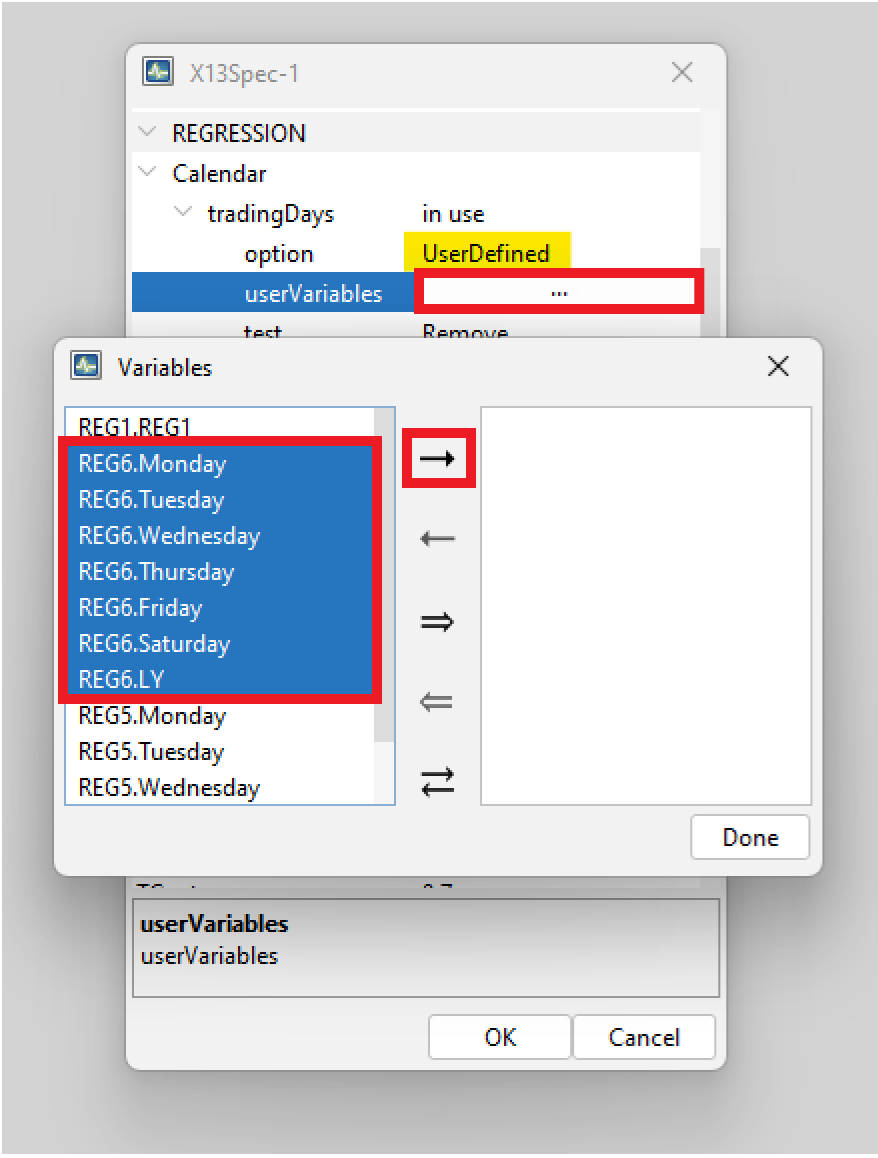
Step 3: Modify specifications Modifications are done the same way in a global specification (whole SAP) or series by series.

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

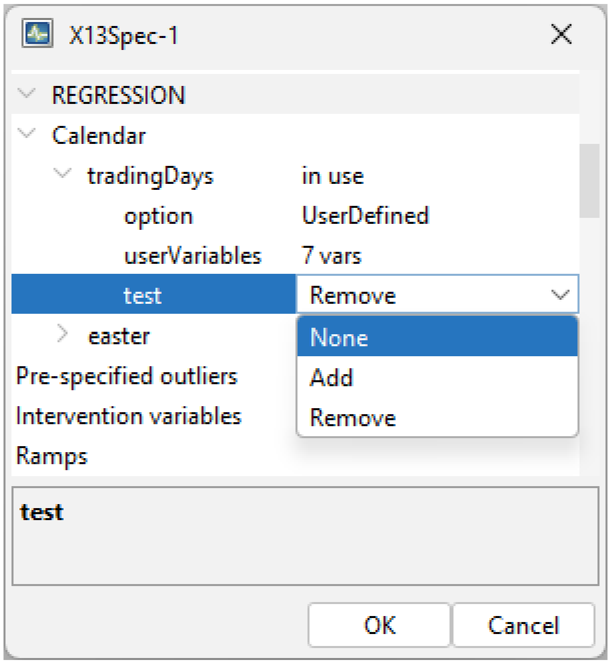
In the specification window, click right from "userVariables" on "Unused" to open the variable selection window



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.

RF0811

Pre-processing (RegArima)

Summary

Estimation span: [1-2012 - 1-2019]
85 observations
Series has been log-transformed
Trading days effects (7 variables)
Easter [8] detected
1 detected outlier

In R

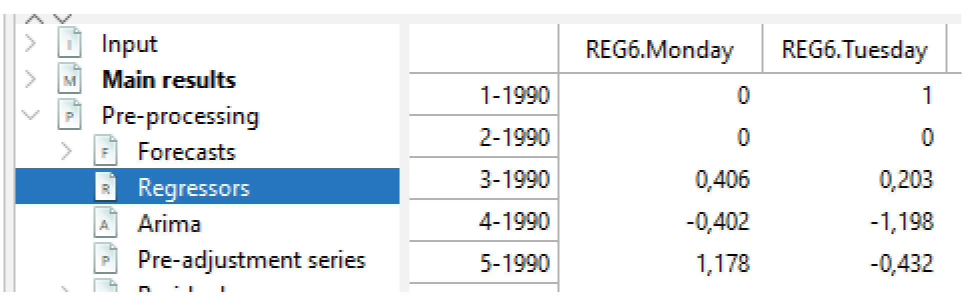
(to be added)

version 2: RJDemetra

version 3: rjd3x13 or rjd3tramoseats

0.0.0.0.2 * Regressors

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



In R

(to be added)

Version 2

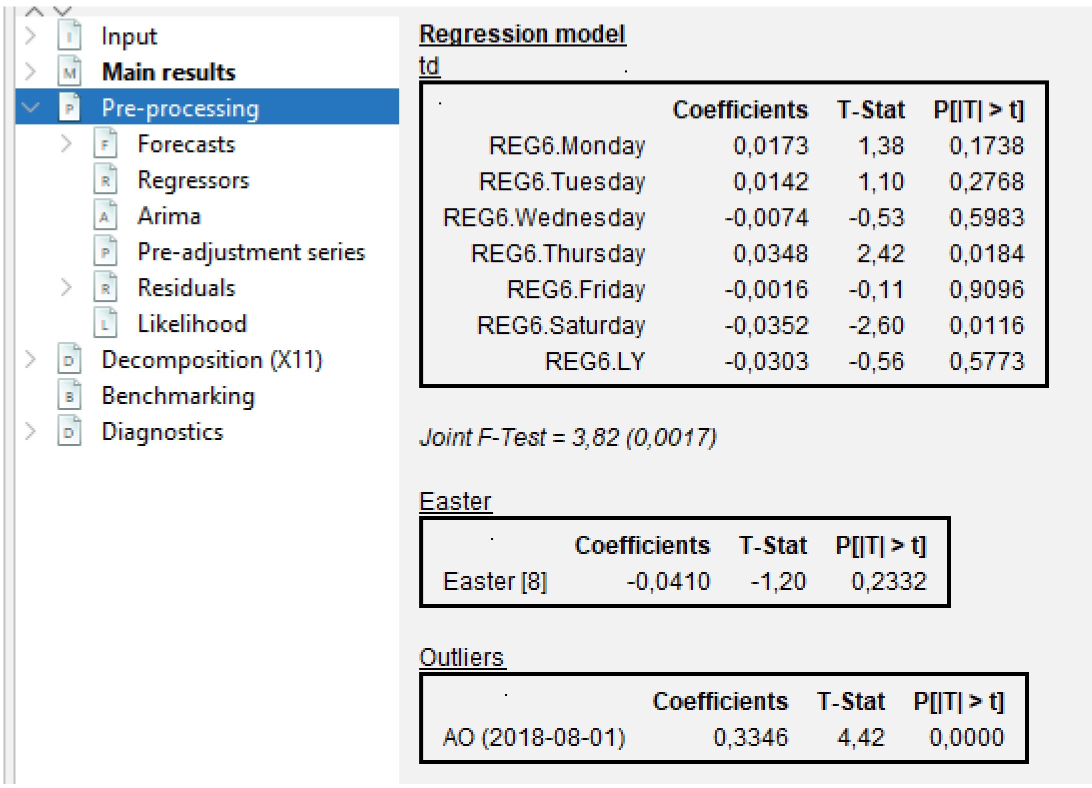
version 3

0.0.0.0.3 * Regression results

Regressions results

In GUI

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



In R

0.0.0.0.4 * Test for residual trading-days effects

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

In GUI

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

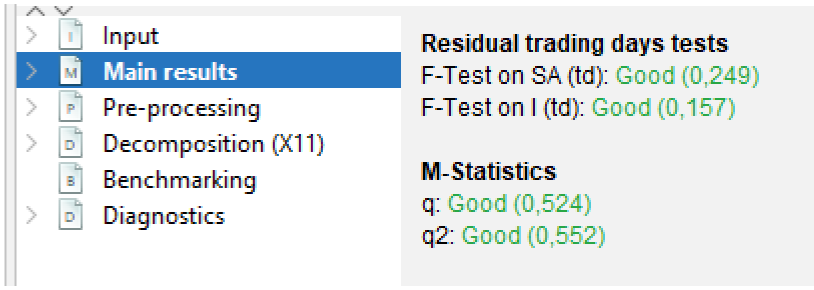


Figure 15: **F-Test results in GUI**

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)

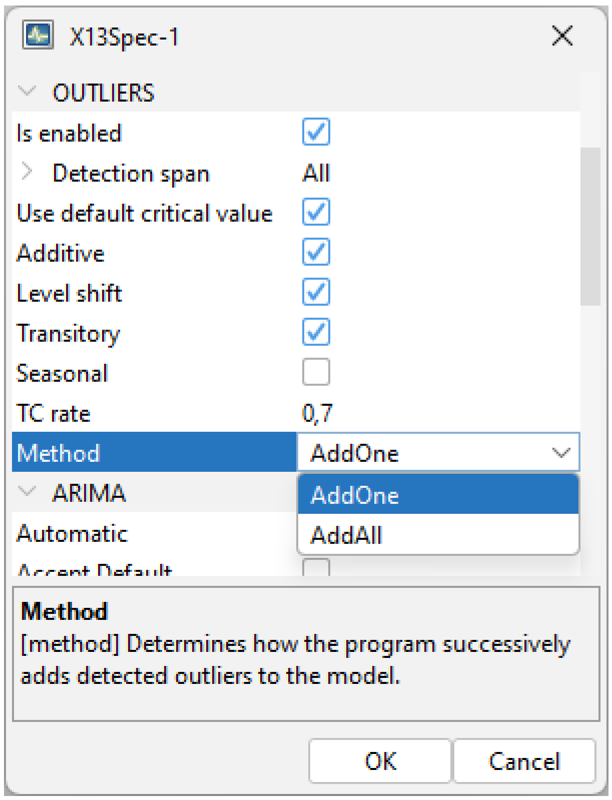


Figure 16: **Automatic detection of outliers in GUI**

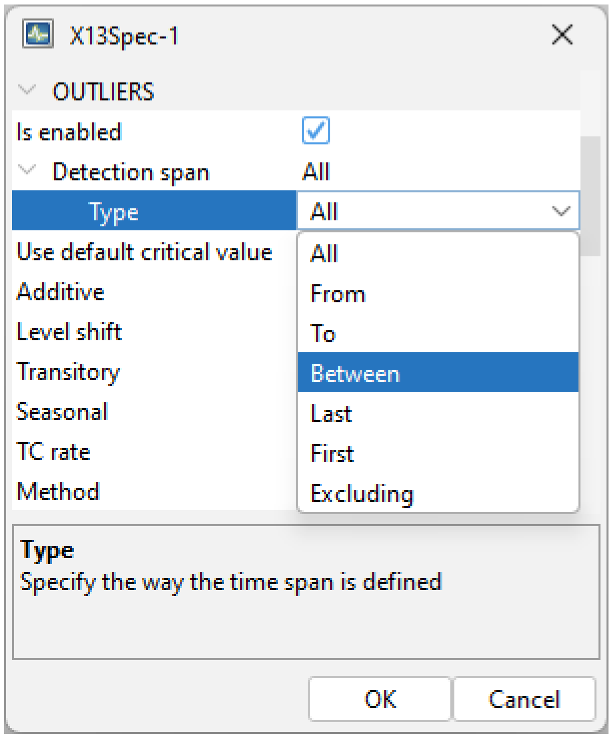


Figure 17: **Outliers types in GUI**

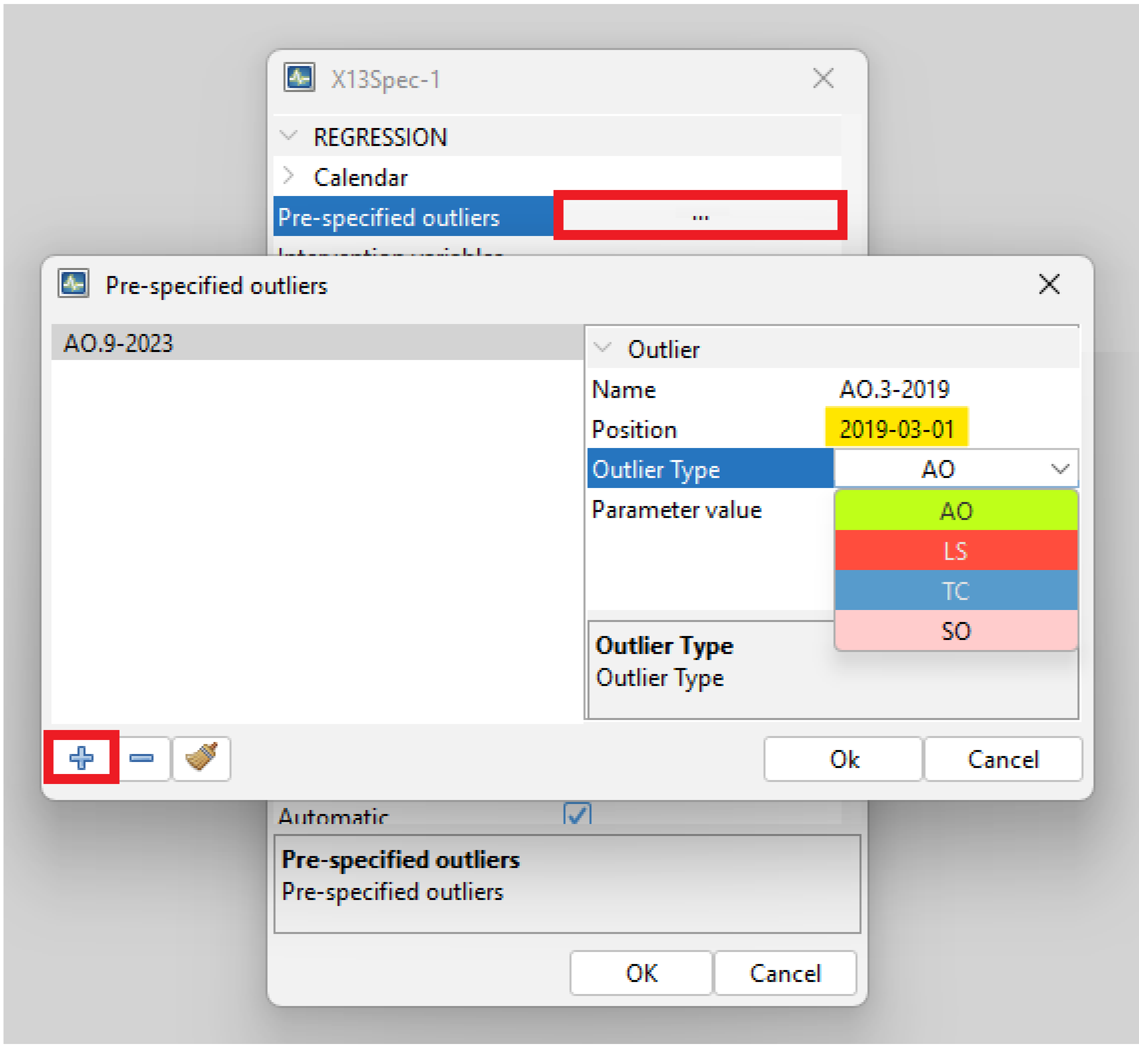


Figure 18: **Pre-specified outliers in GUI**

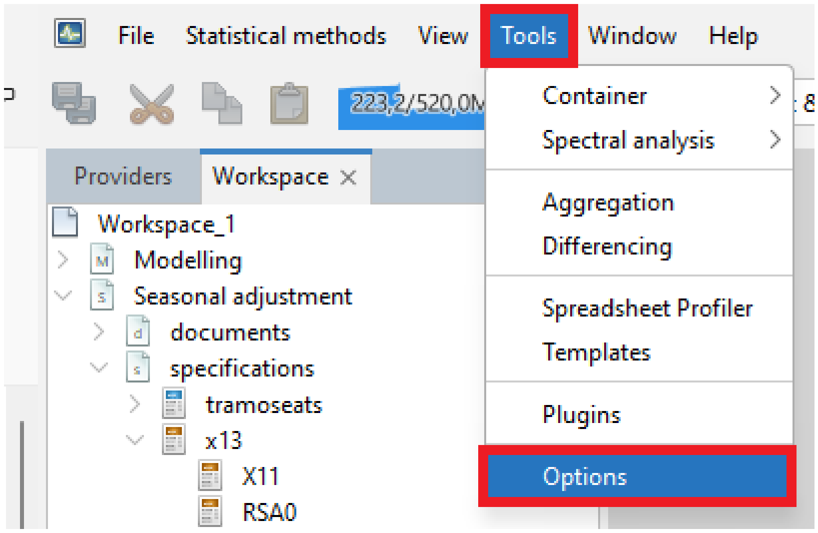


Figure 19: Options in GUI

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)

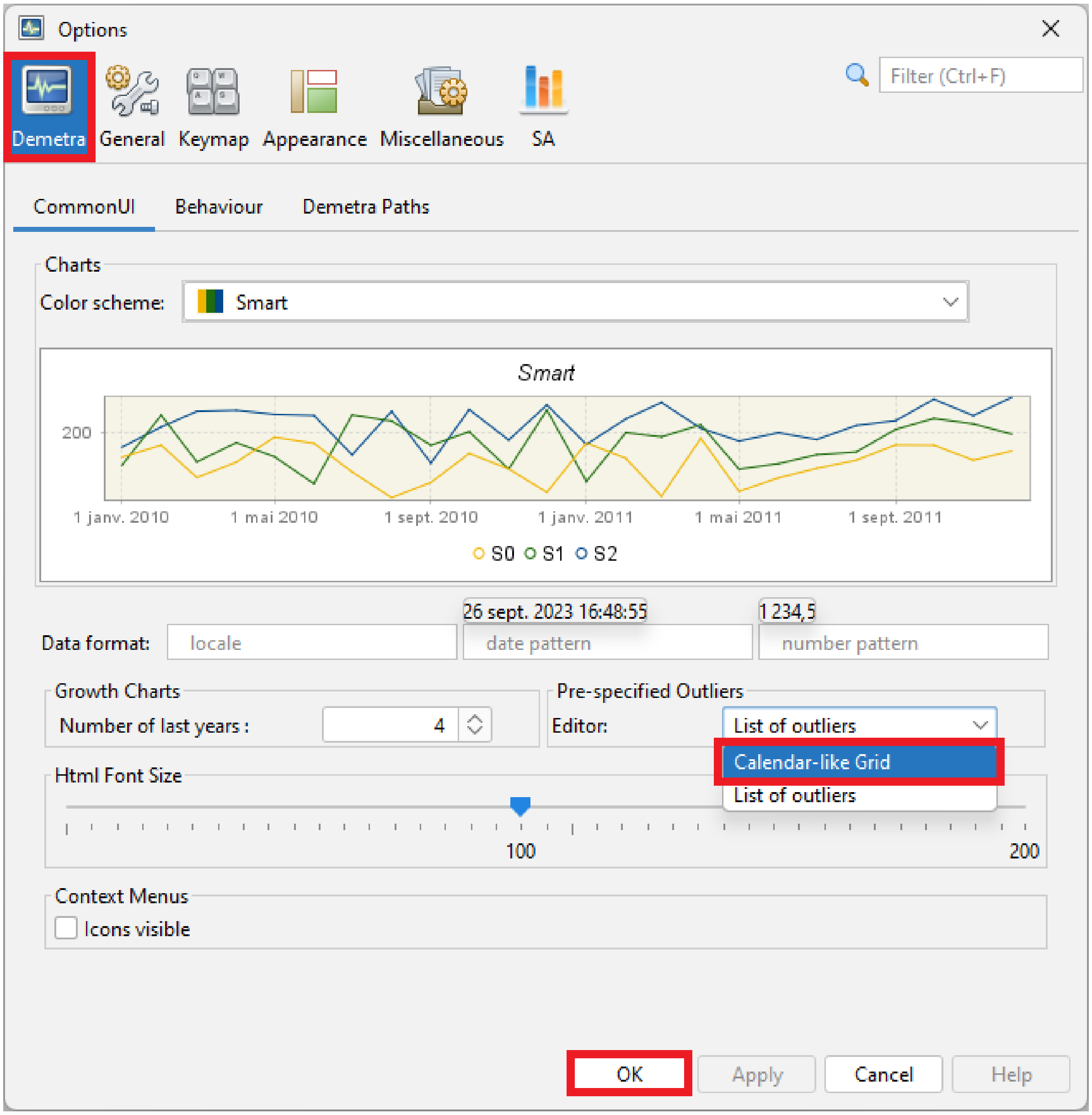


Figure 20: **Change view to calendar**

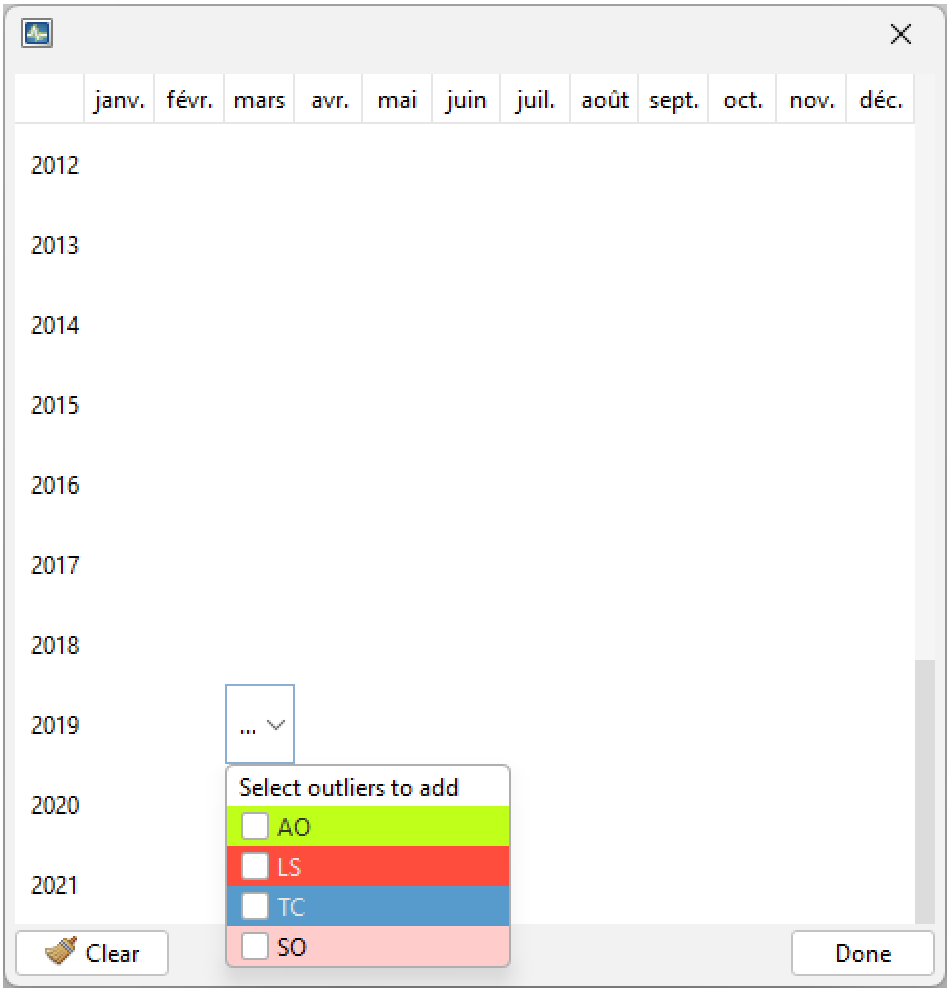


Figure 21: **Calendar view in GUI

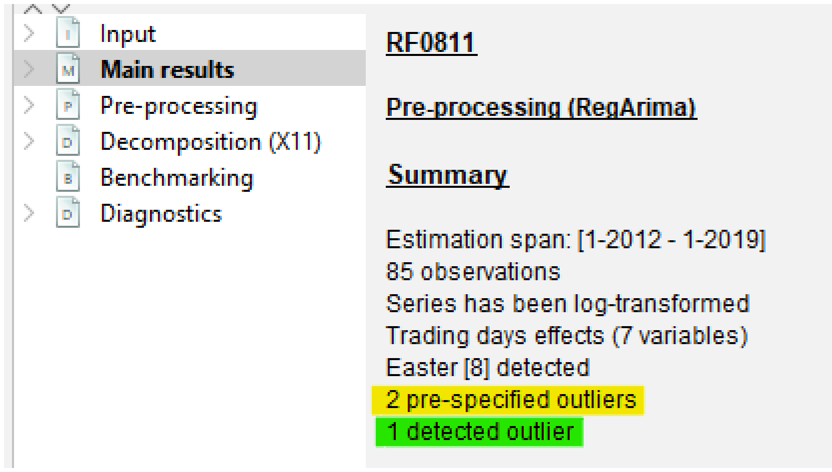


Figure 22: **Main results in GUI**

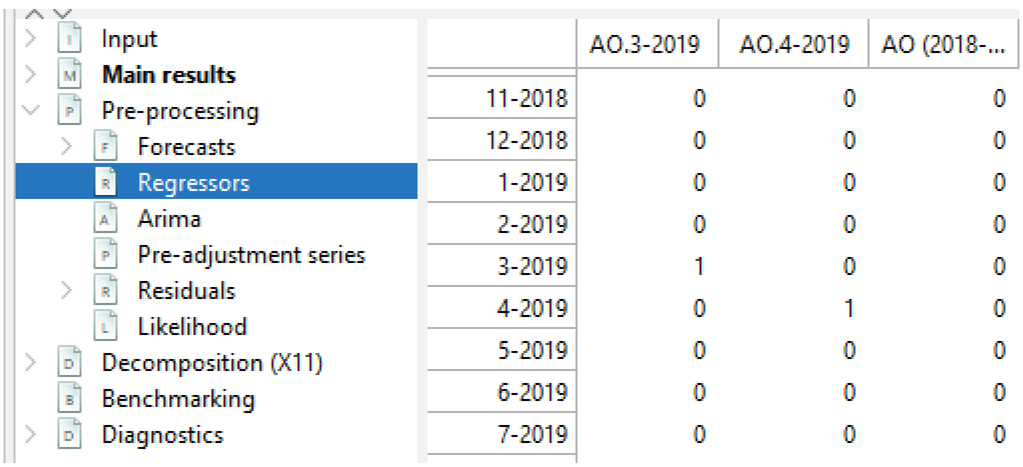


Figure 23: **Regressors in GUI**

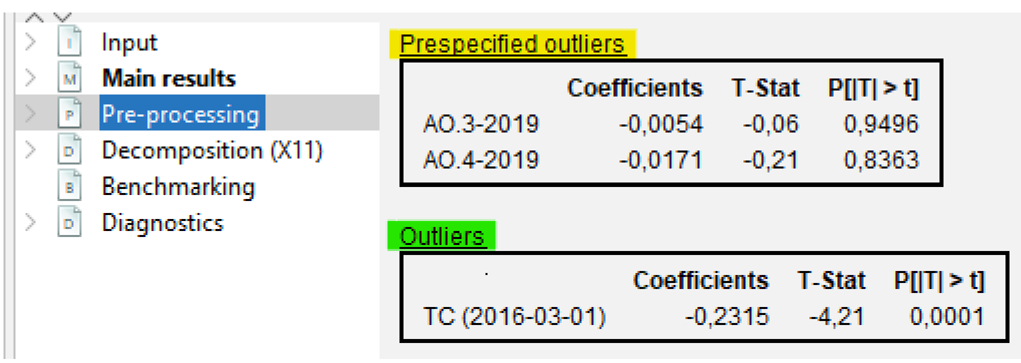


Figure 24: **Regression details in GUI**

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)

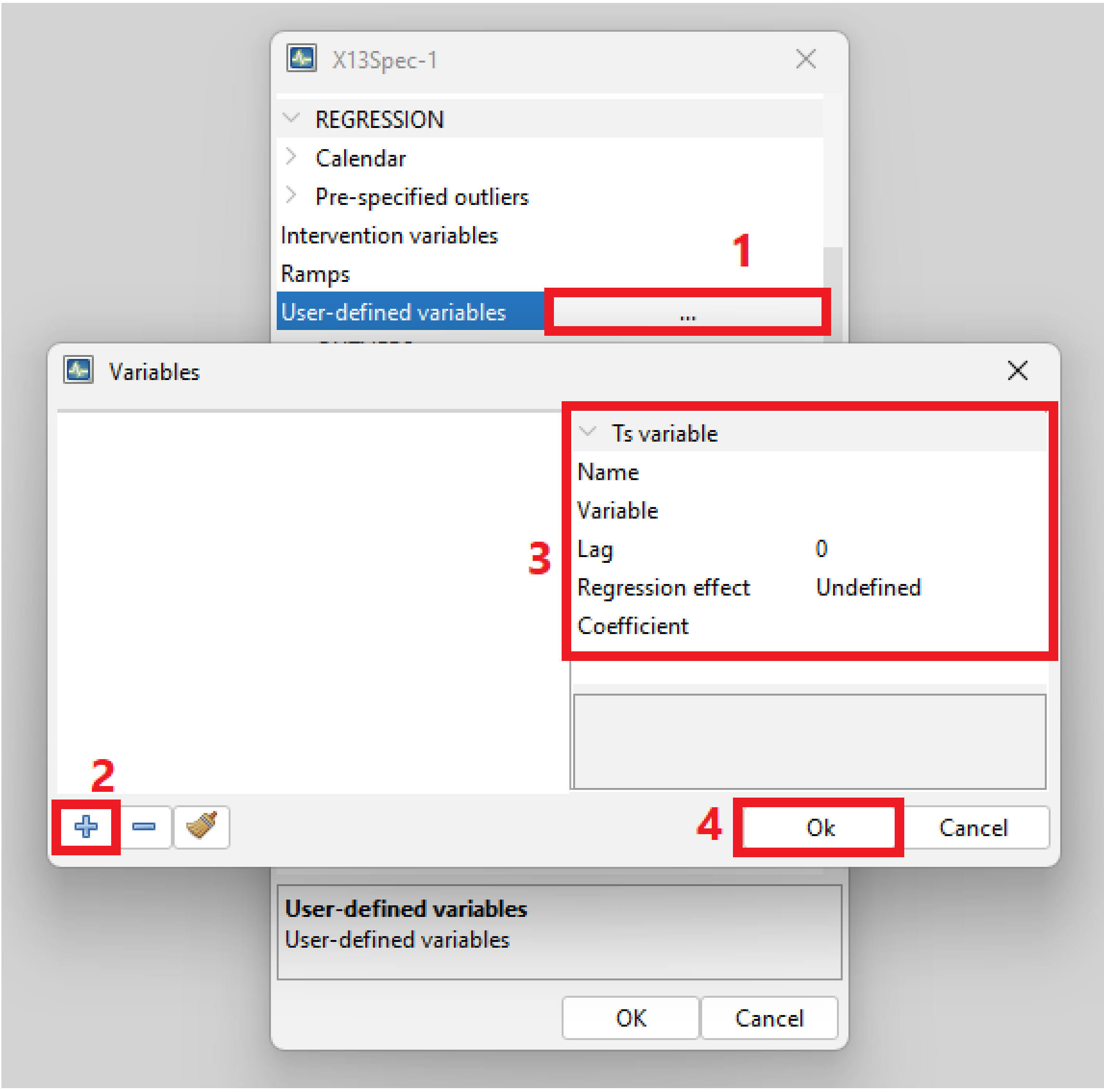


Figure 25: **Creation of user-defined variables in GUI**

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

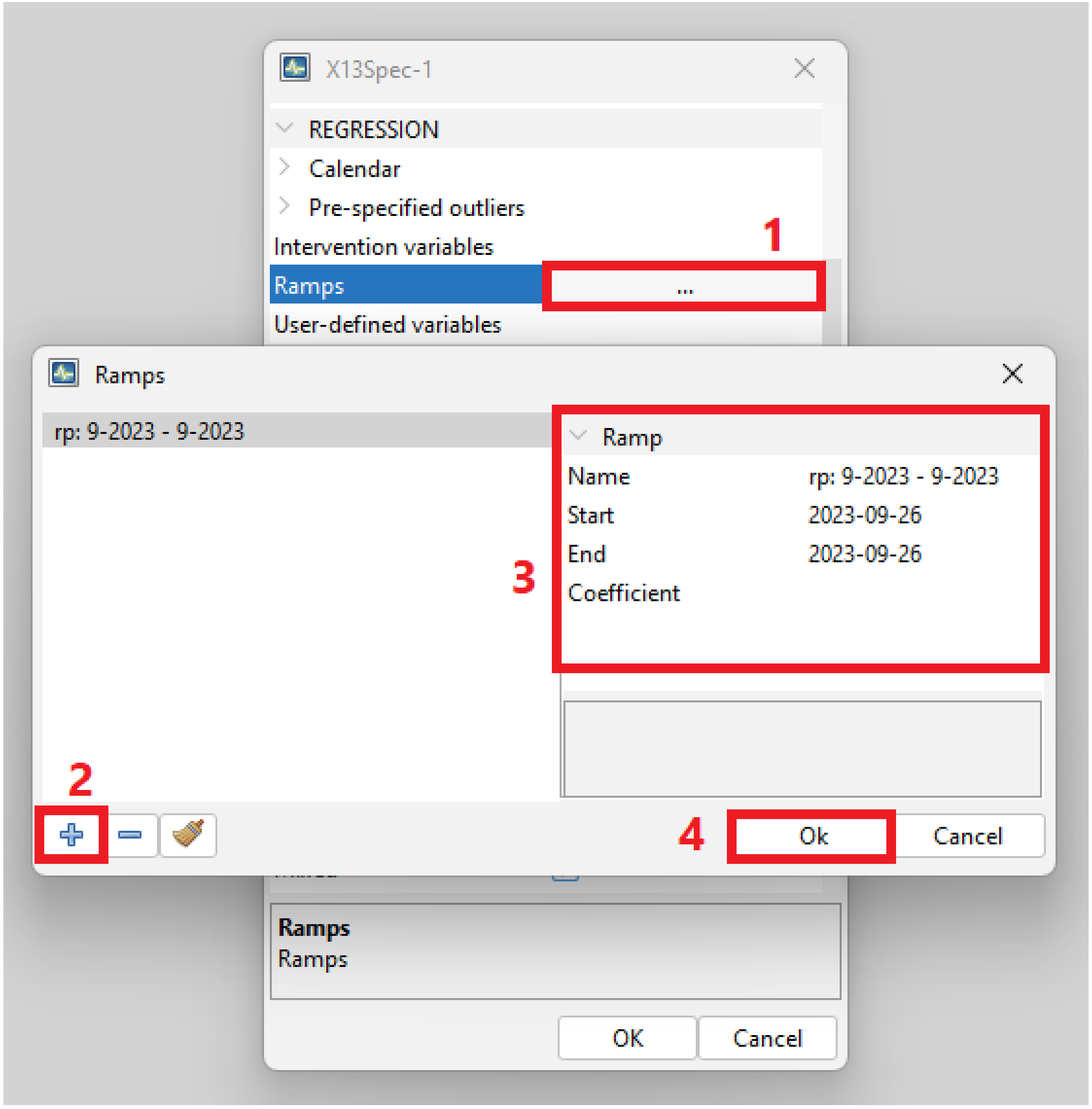


Figure 26: **Creation of ramp variable in GUI**

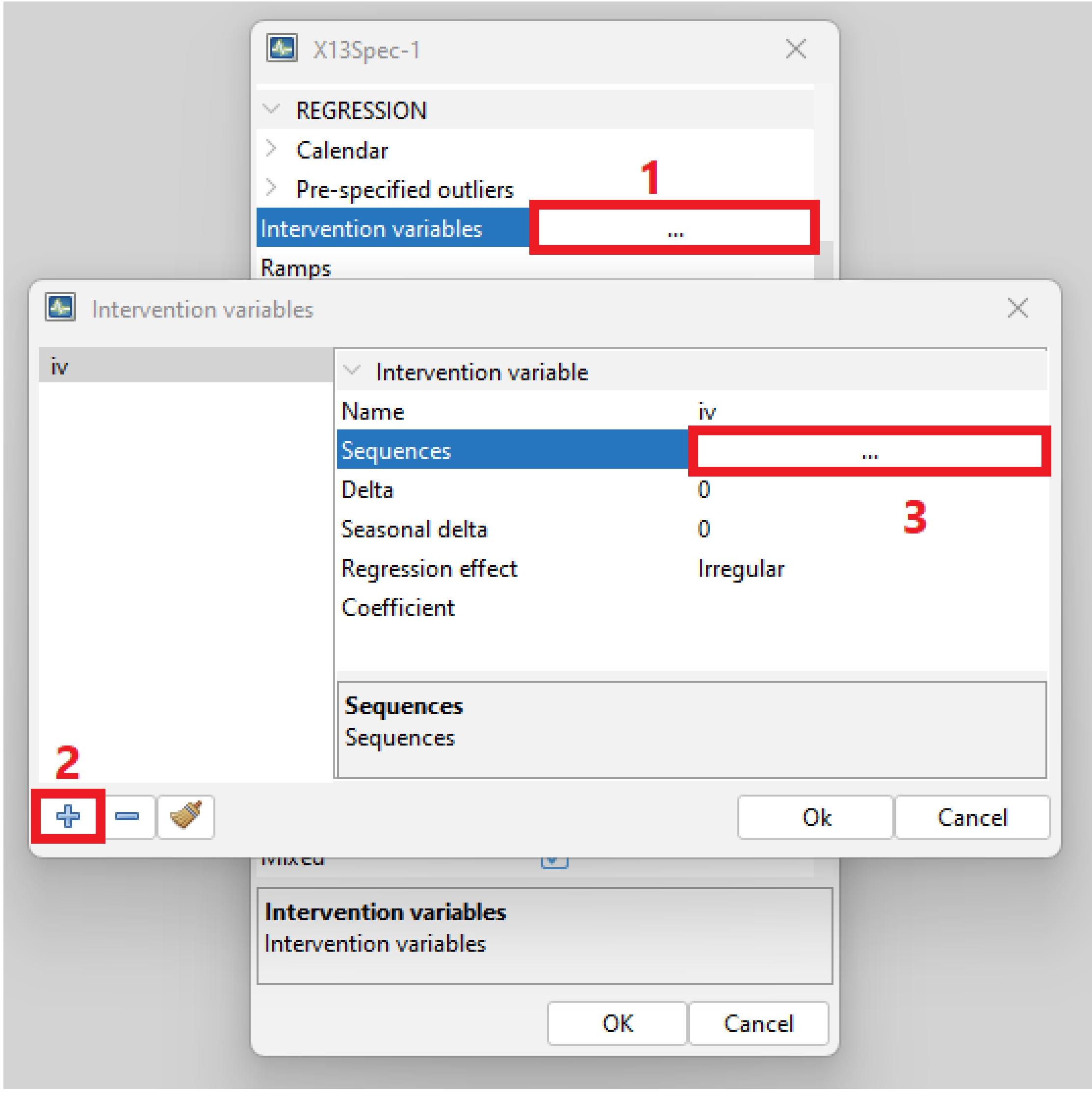


Figure 27: **Step 1**

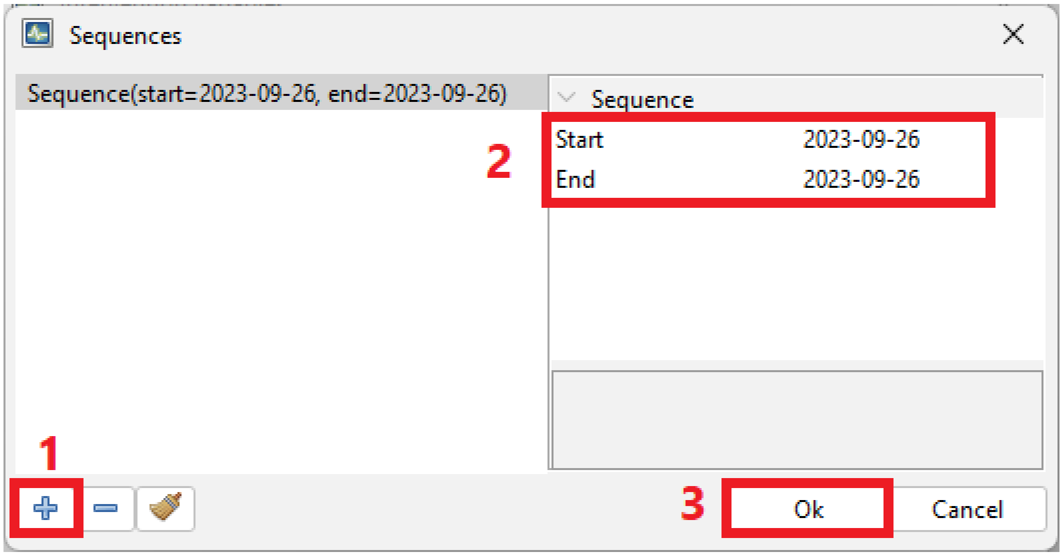


Figure 28: **Step 2**

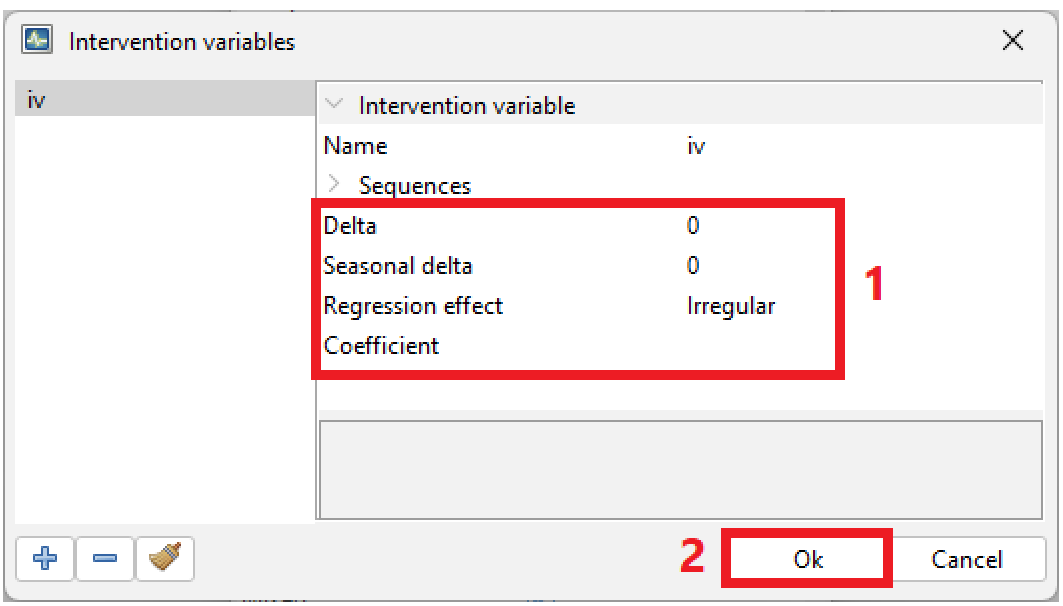


Figure 29: **Step 3**

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)

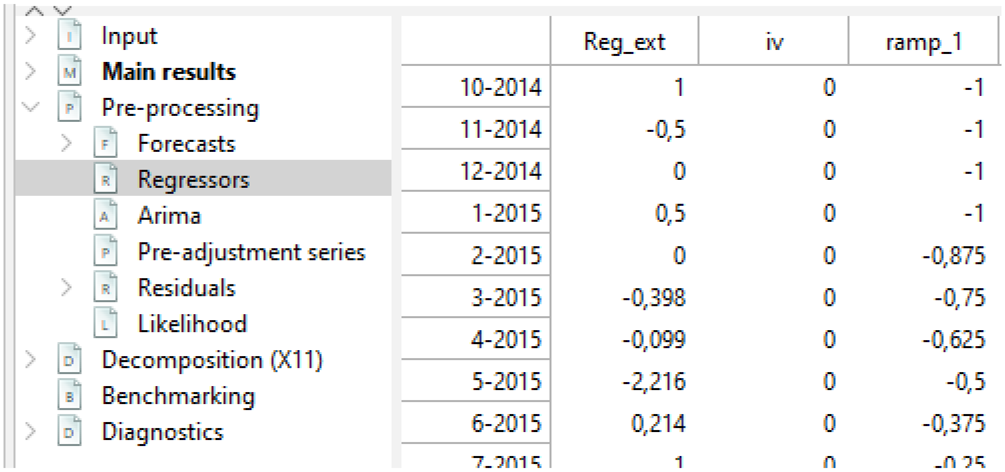


Figure 30: **Regressors in GUI**

0.0.0.0.2 * Regression details

In GUI

Regression details are in the pre-processing pane.

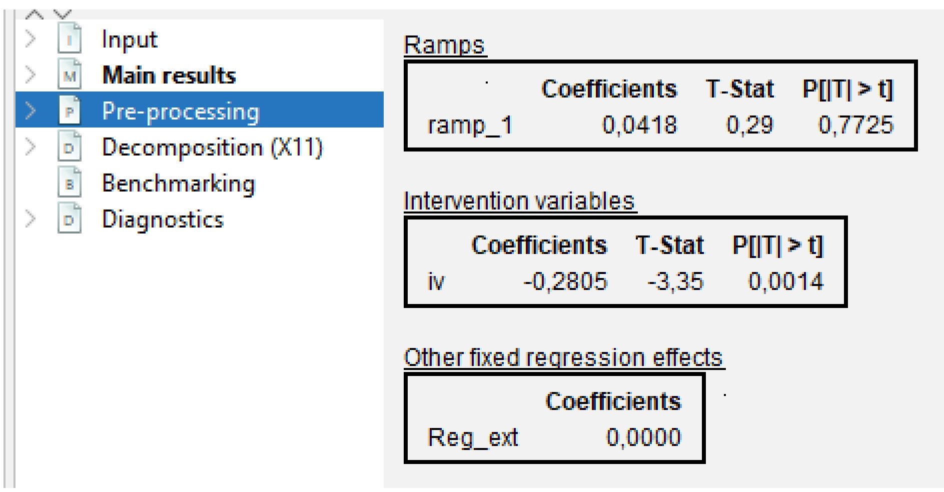


Figure 31: **Regression details**

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.

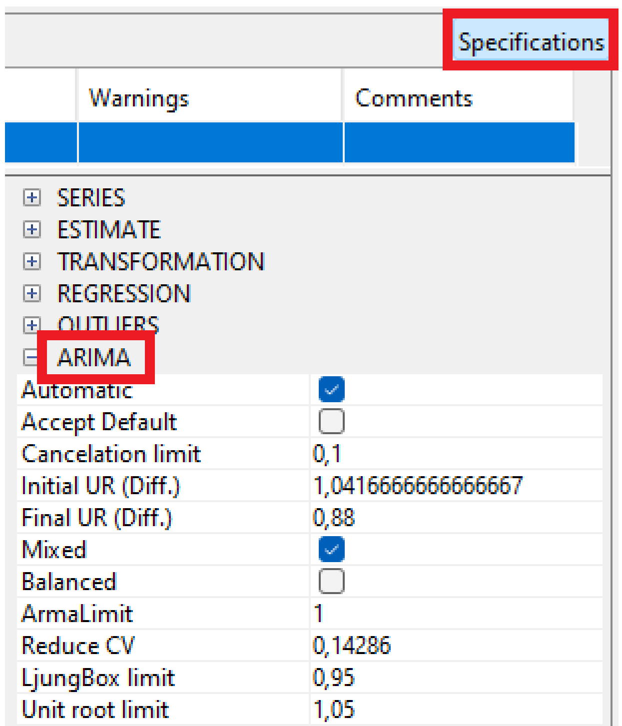


Figure 32: **Setting in GUI in v2**

0.0.0.1 v2

0.0.0.2 v3

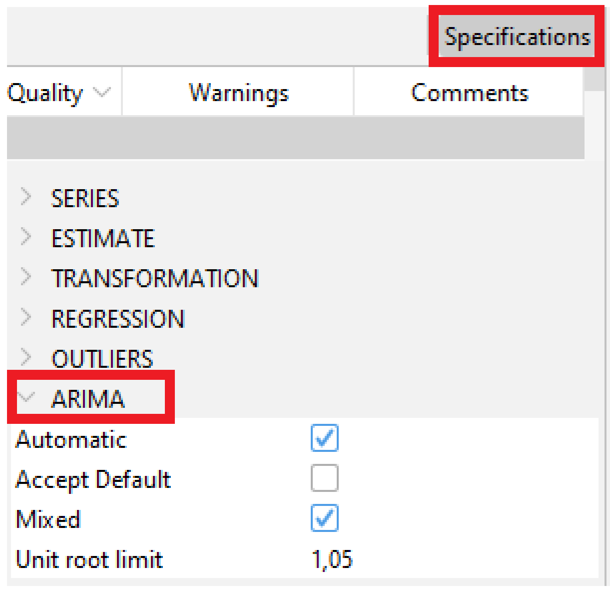


Figure 33: **Setting in GUI in v3**

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_
```

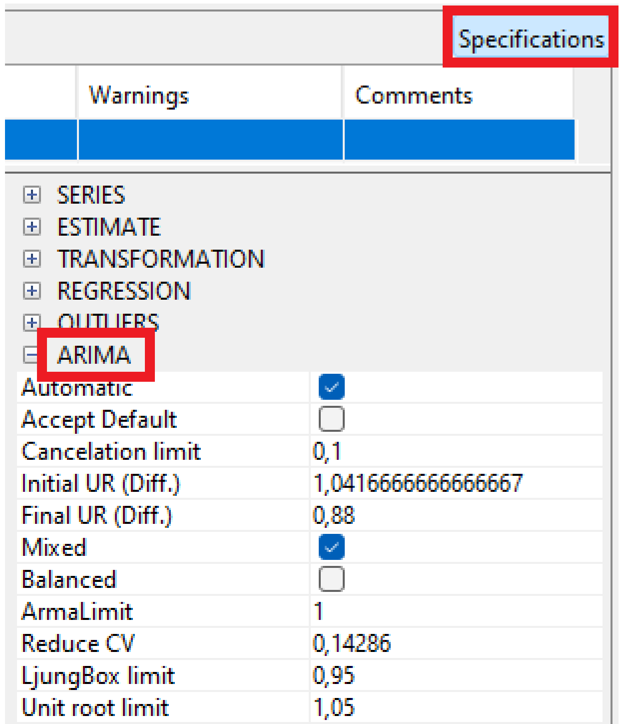


Figure 34: **Setting in GUI in v2**

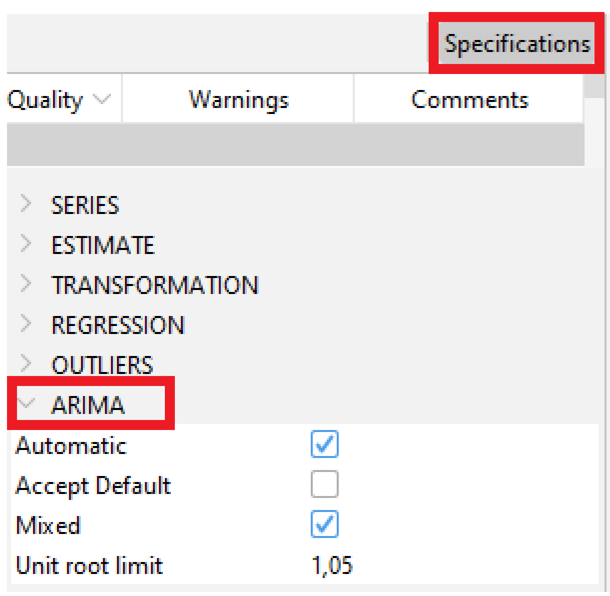


Figure 35: **Setting in GUI in v3**

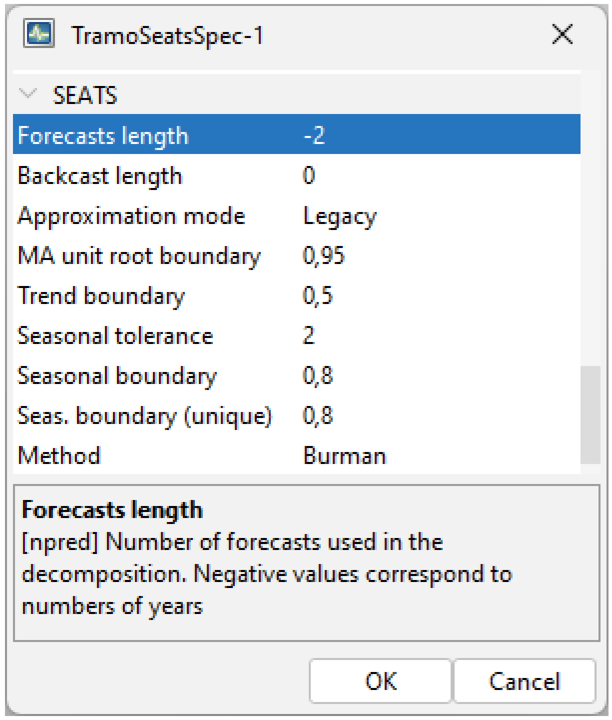


Figure 36: **Forecast horizon when using Tramo-Seats**

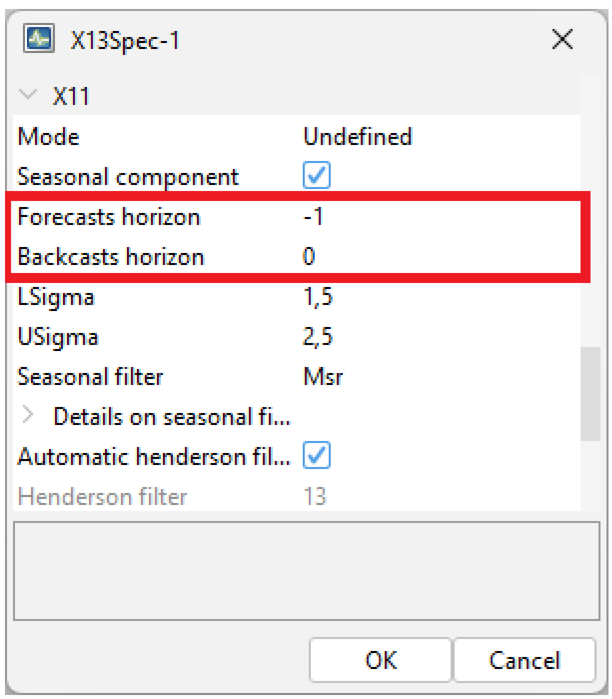


Figure 37: **Forecast horizon when using X-13-ARIMA**

```
)
```

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

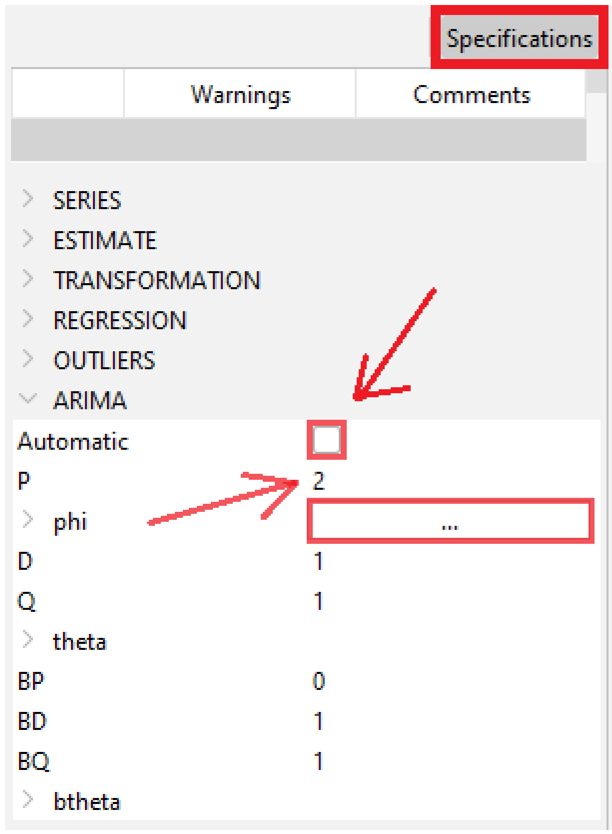


Figure 38: **Manual ARIMA modelling**

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_,
```

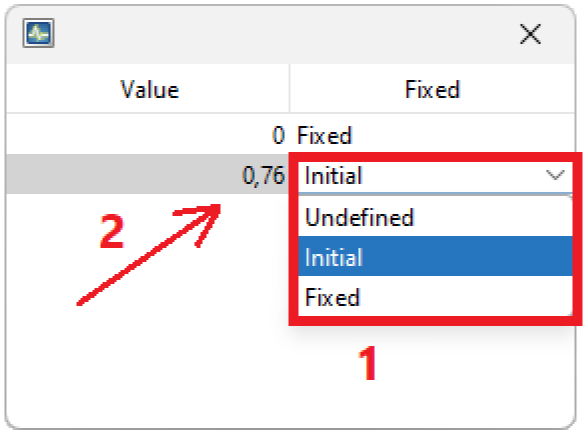


Figure 39: **Fixing coefficients**

```
    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

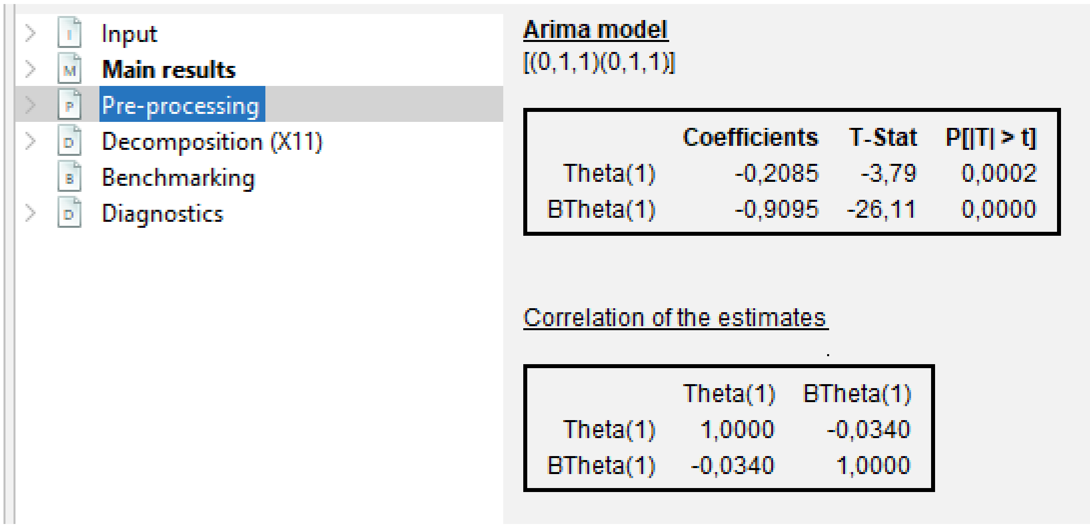


Figure 40: **Final model**

More details in Pre-processing/ARIMA Node

In residual Node

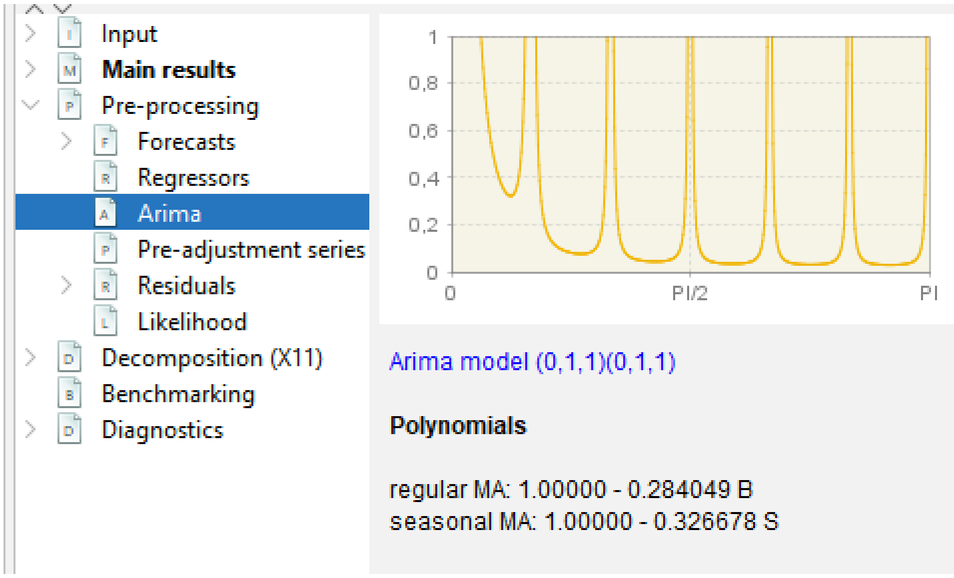


Figure 41: **ARIMA details**

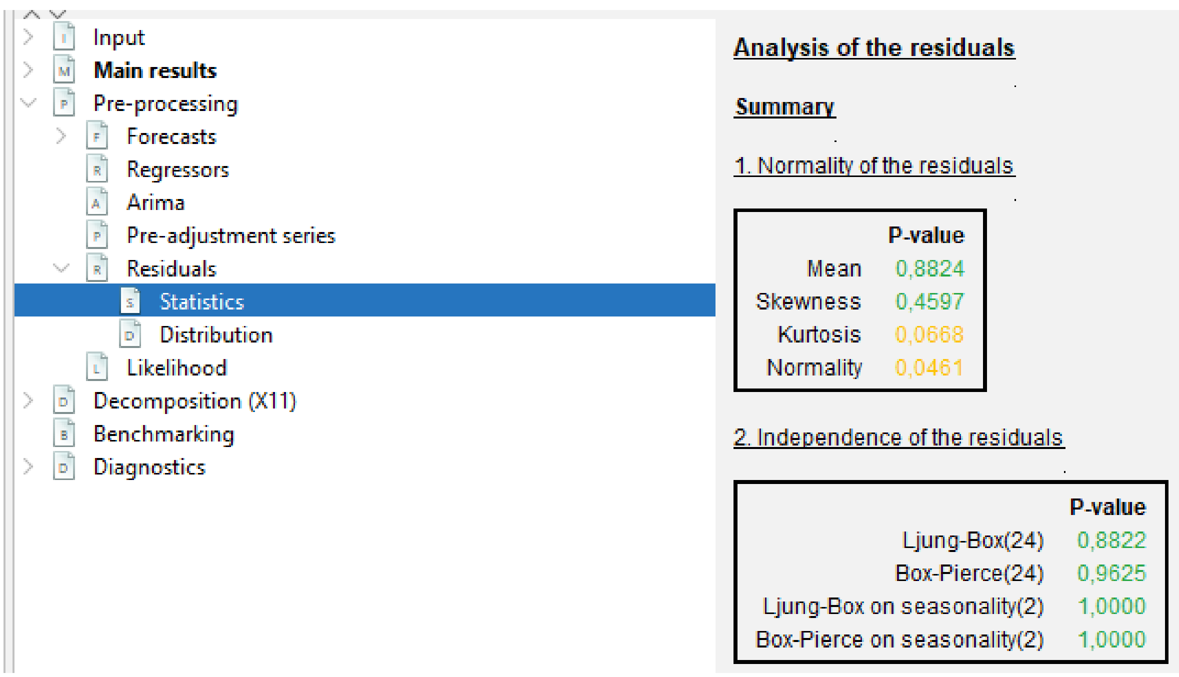


Figure 42: **Residuals**

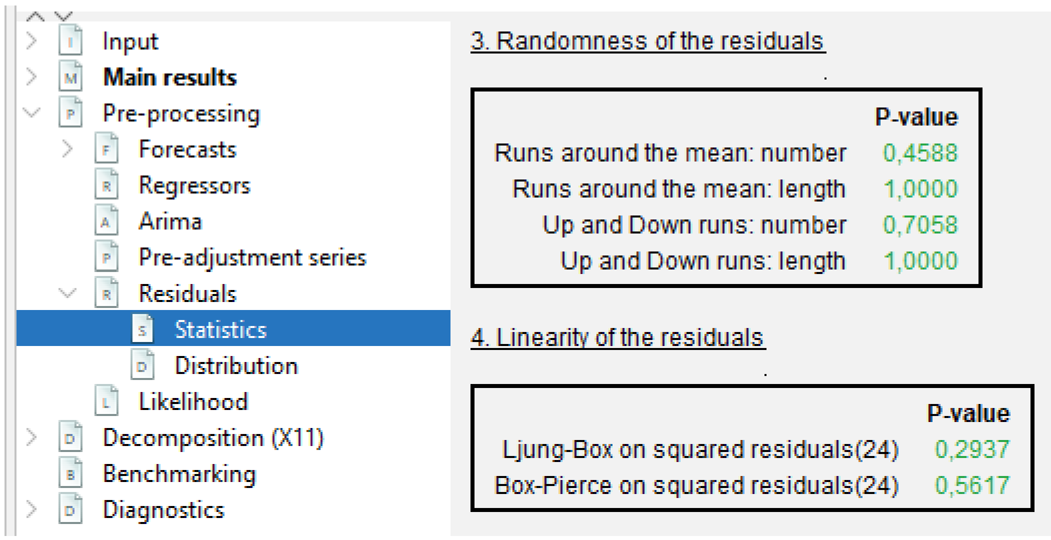


Figure 43: **Residuals**

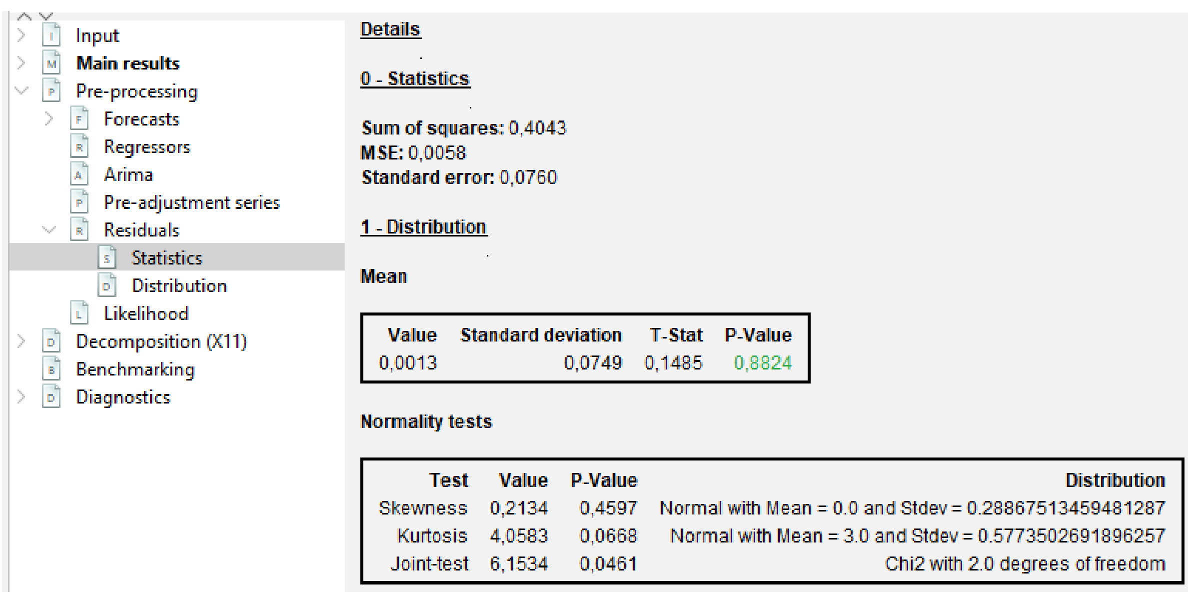
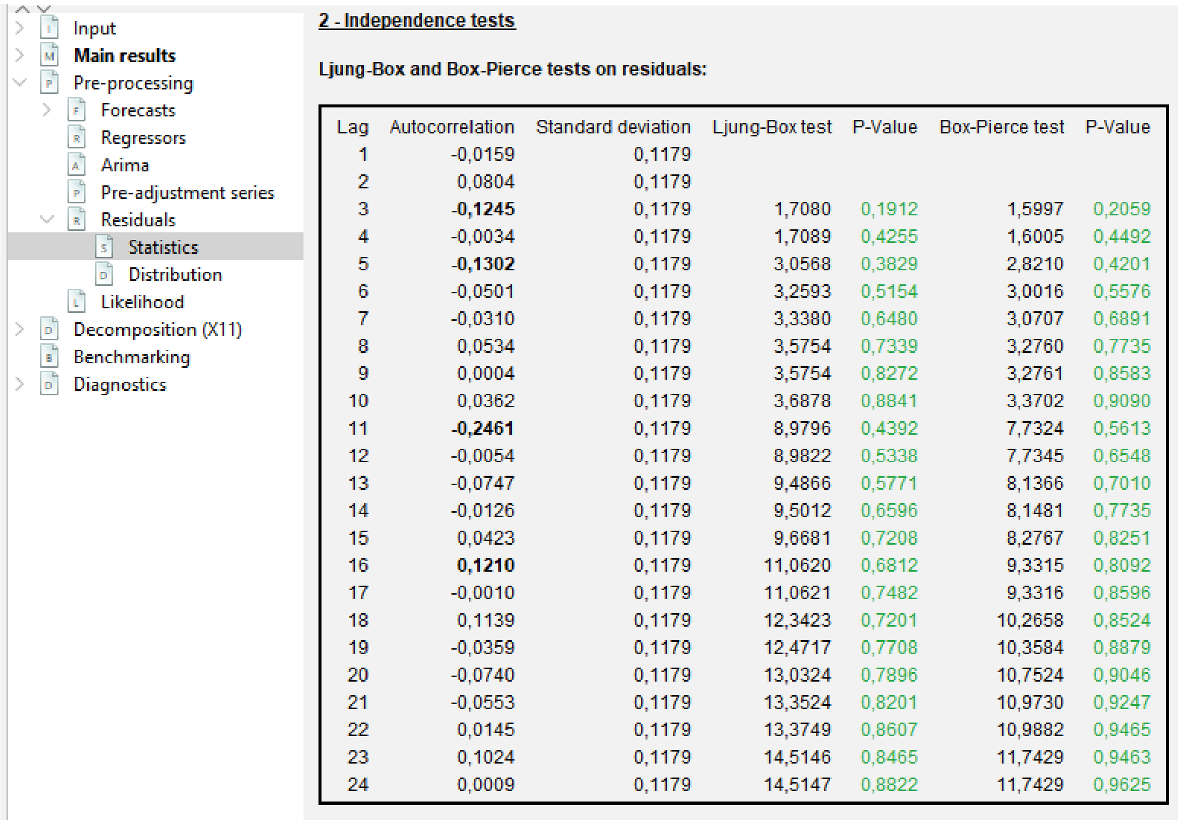


Figure 44: **Residuals**

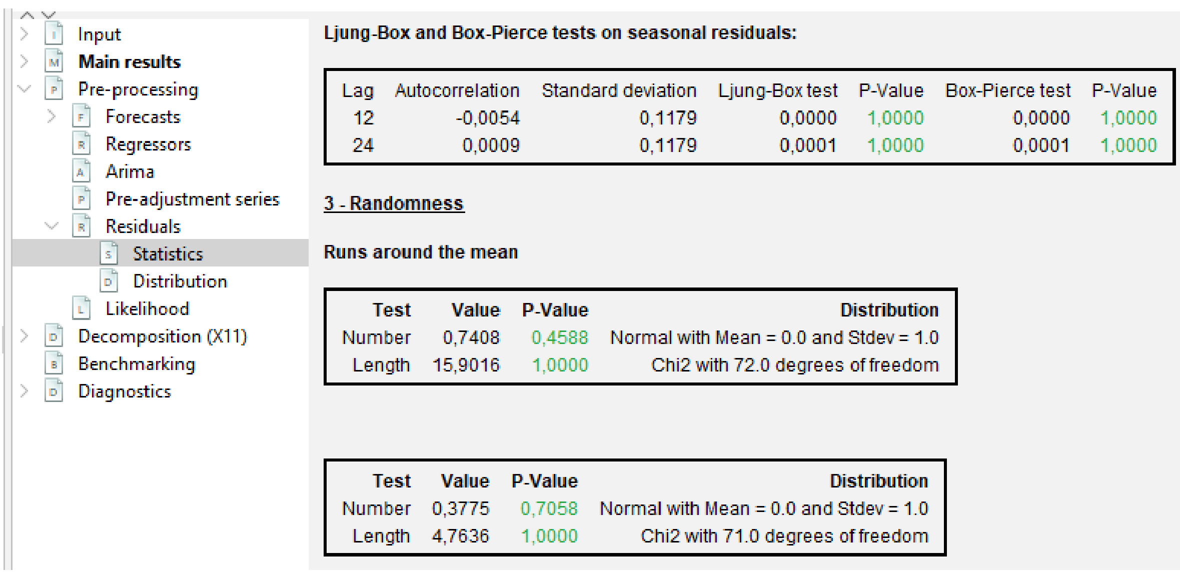


2 - Independence tests

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

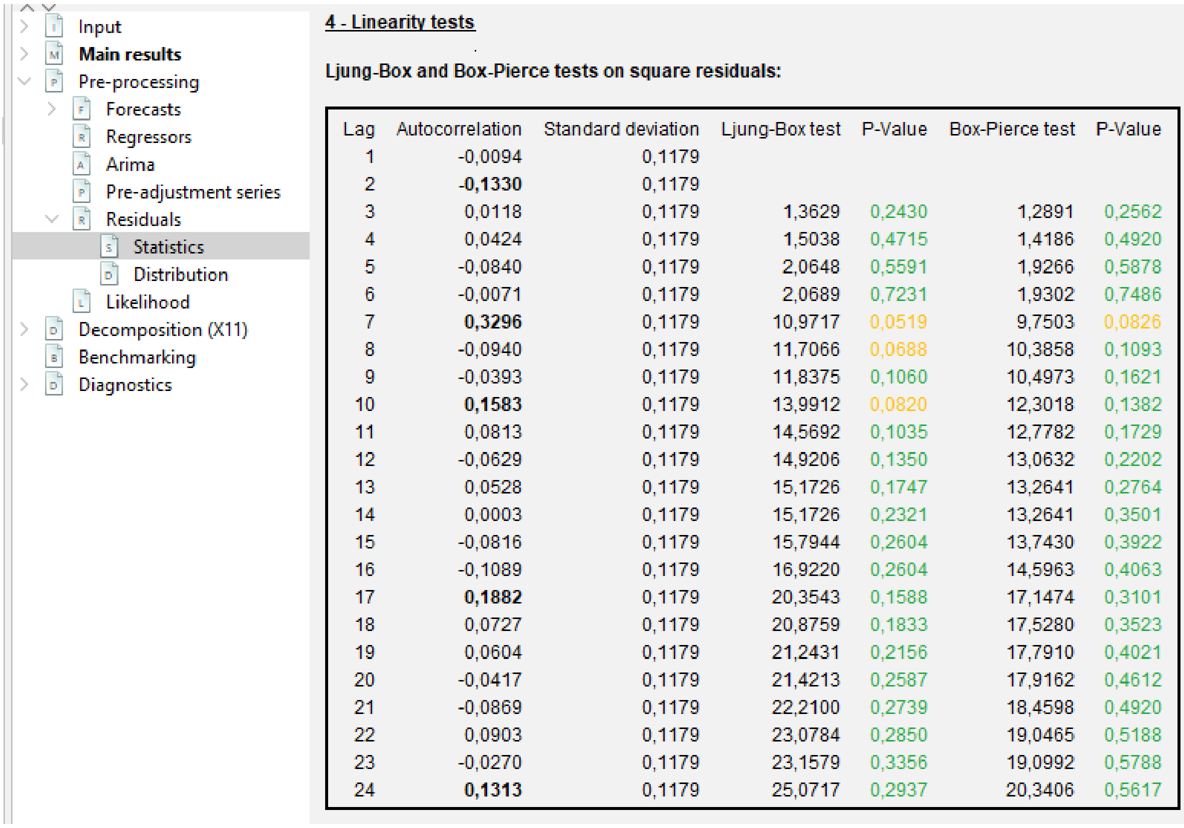
3 - Randomness

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**

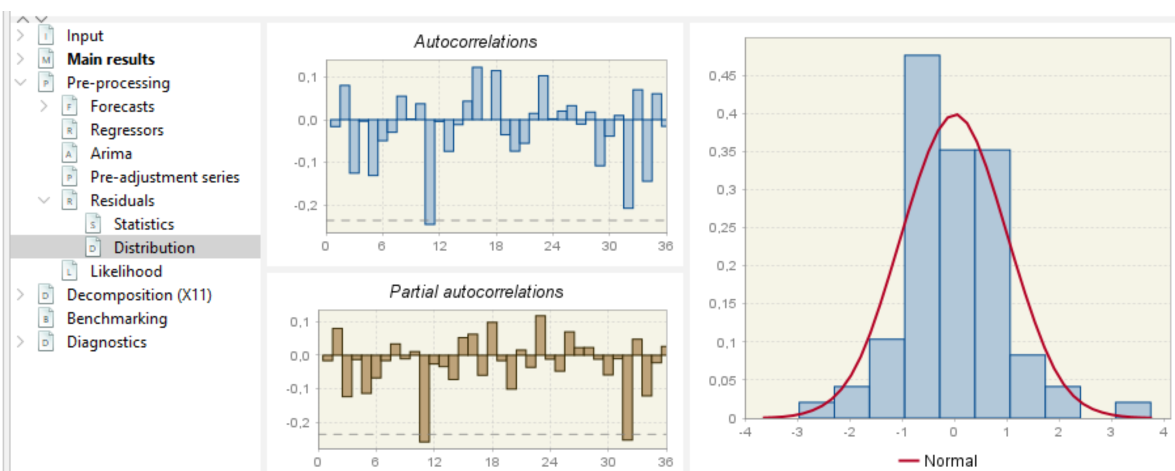


Figure 48: **Distribution**

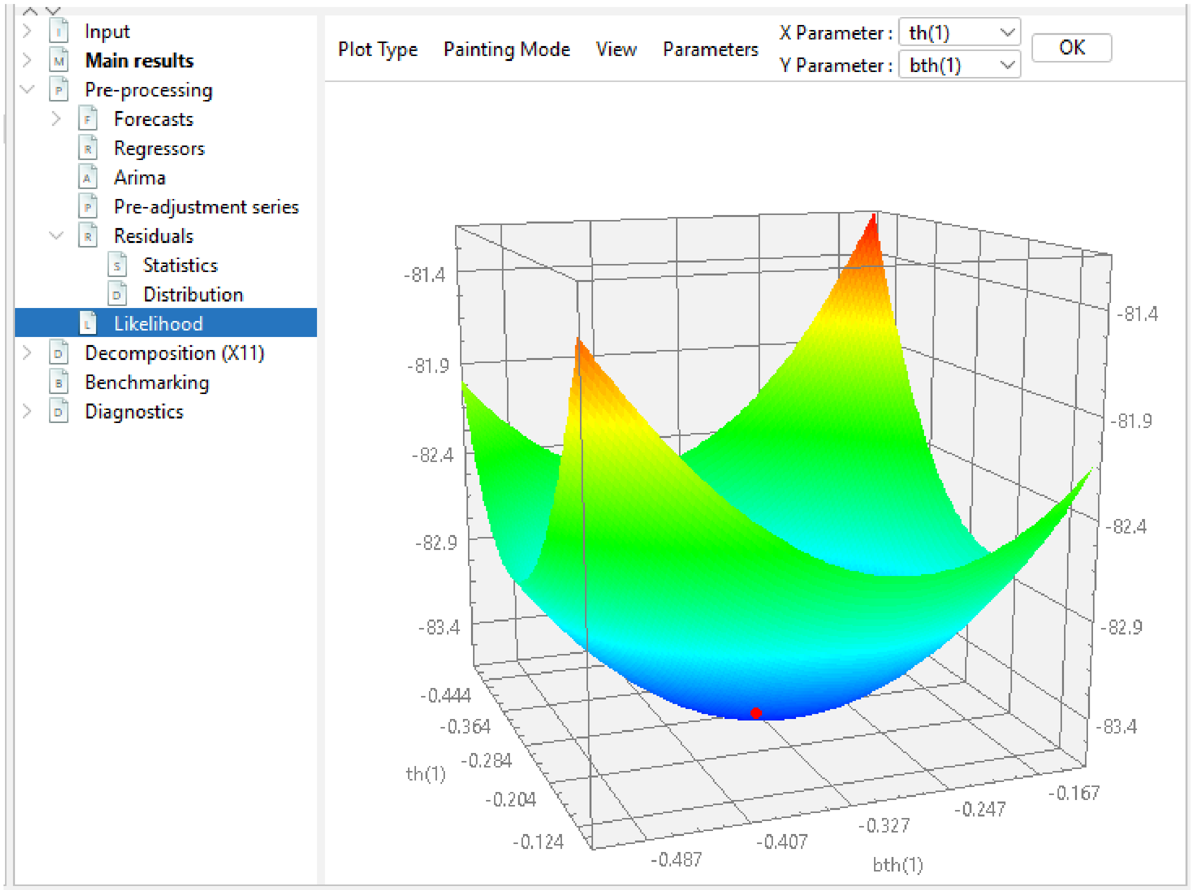
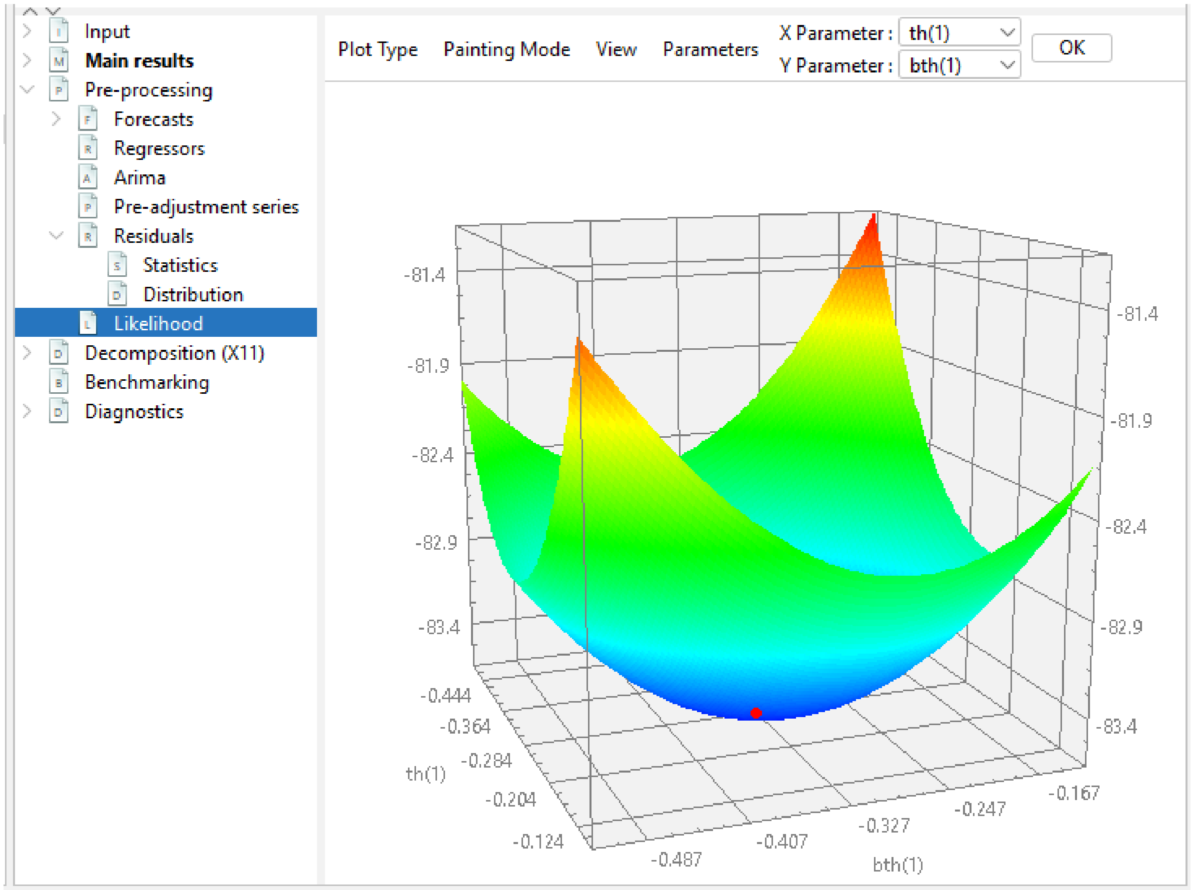


Figure 49: **Likelihood**



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

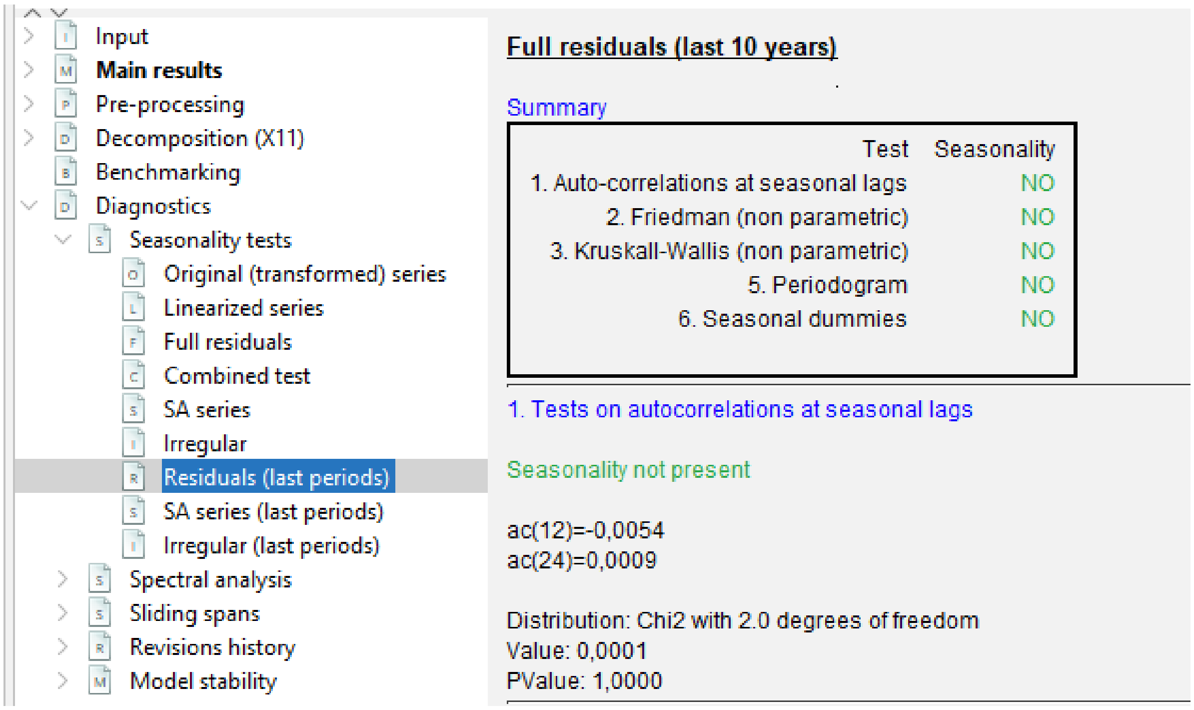


Figure 50: **Residuals**

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)
```

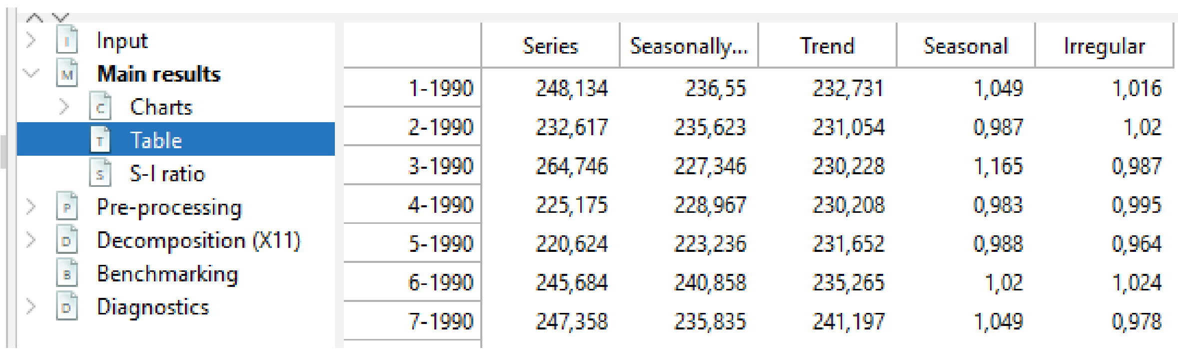
Retrieve series

All series used as an input or generated by X11 are stored into Tables A, B, C, D (and D final in v3) and E.

Detailed series names are described here in the Methods part.

Display in GUI

Final components from the SA Processing are displayed in node *Main results > Table*. They contain the re-allocated effects of outliers or external regressors. The final seasonal component 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.

In version 3

- D-Table contains the final components stemming from the decomposition of the linearized series (B1 or ylin in *Pre-processing > Pre-adjustment* series node)
- D-Final-Table contains the final components including pre-adjustment effects (equal to the series contained in node *Main results > Table*)

In version 2:

- In D-table series D10, D10a, D11, D11a, D12, D12a, D13 are the final components including pre-adjustment effects (equal to the series contained in in node *Main results > Table*)

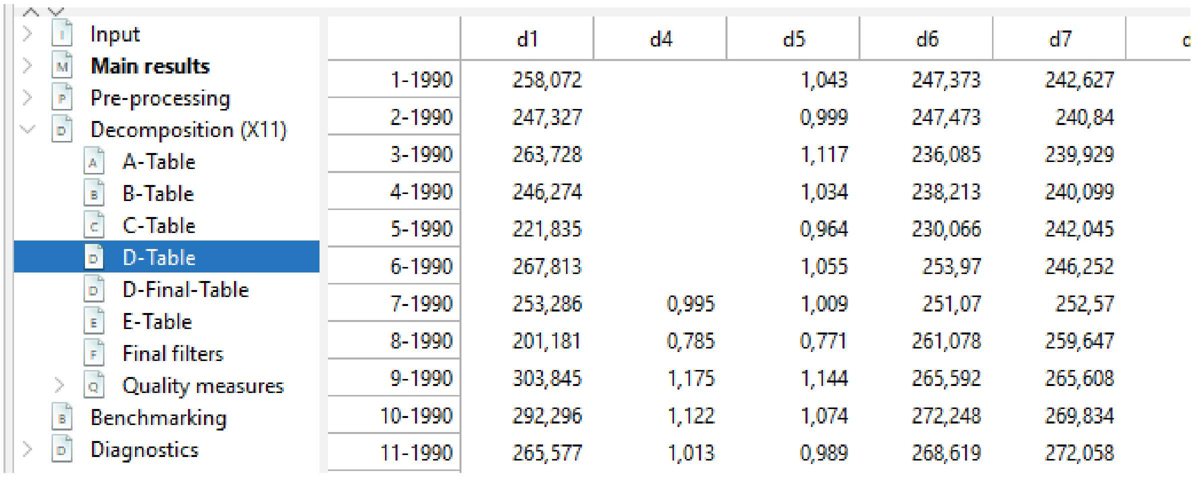


Figure 52: **Detailed results**

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
```

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

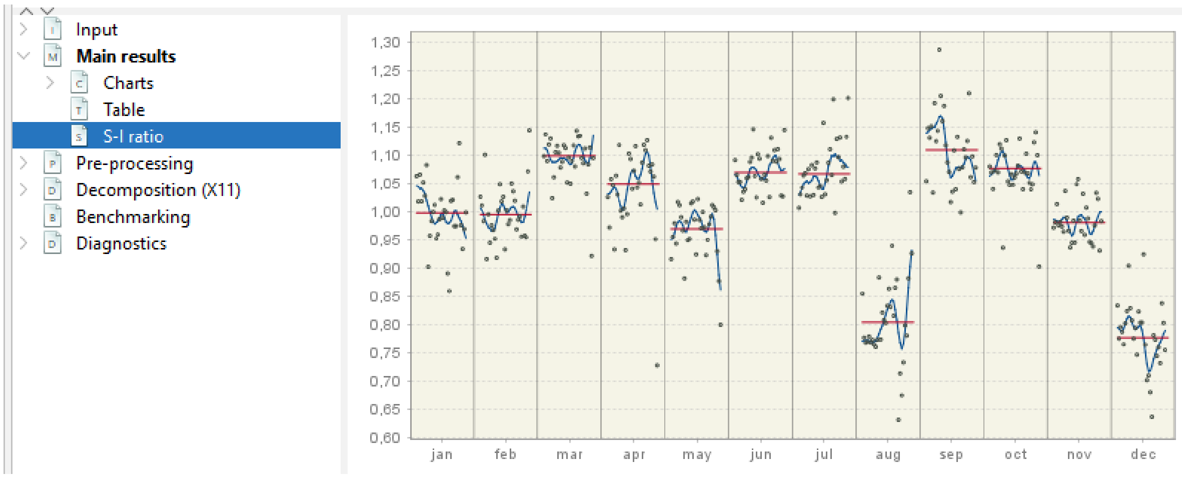


Figure 53: **S-I Ratio**

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 (Series 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.

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.

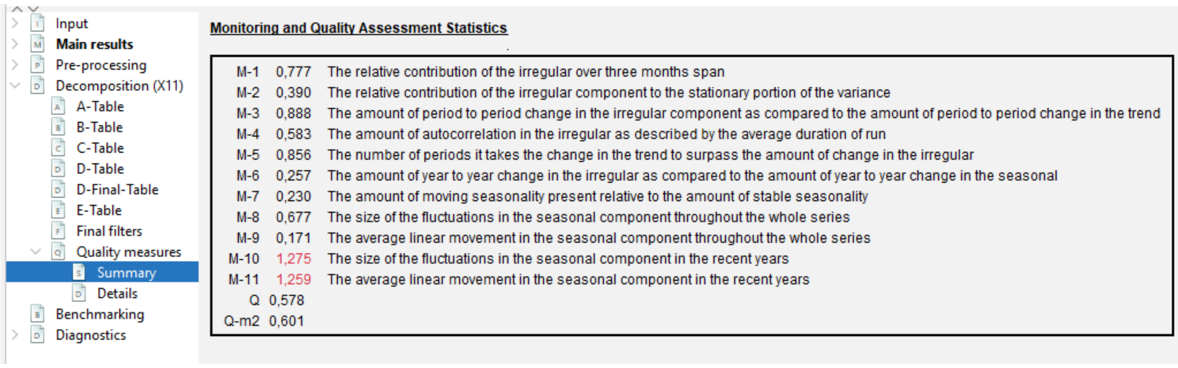


Figure 54: **Summary of the quality measures**

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

```
# 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

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

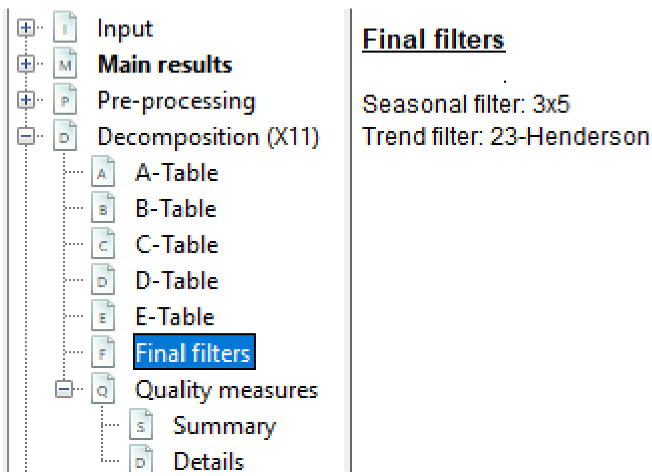


Figure 55: **Table of final filters**

Retrieve in R

In version 2

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

In version 3

```r
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

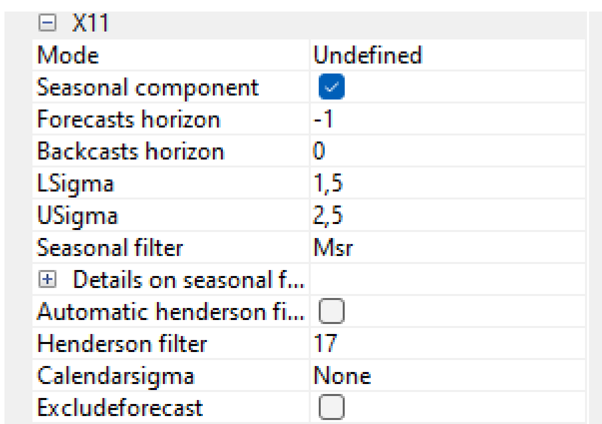


Figure 56: Text

Setting details on seasonal filters

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

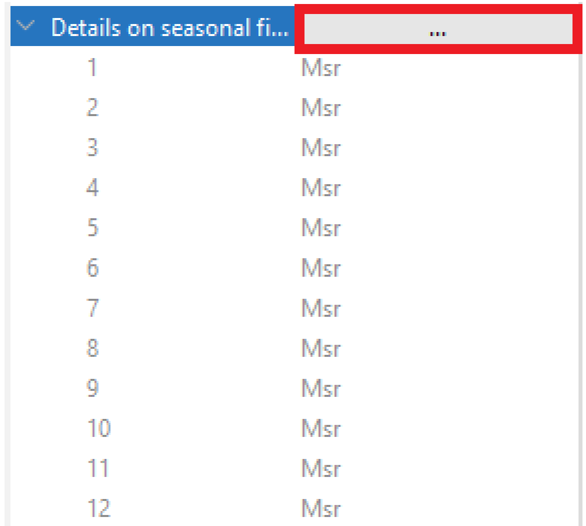


Figure 57: **Seasonnal filters**

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.

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,
```

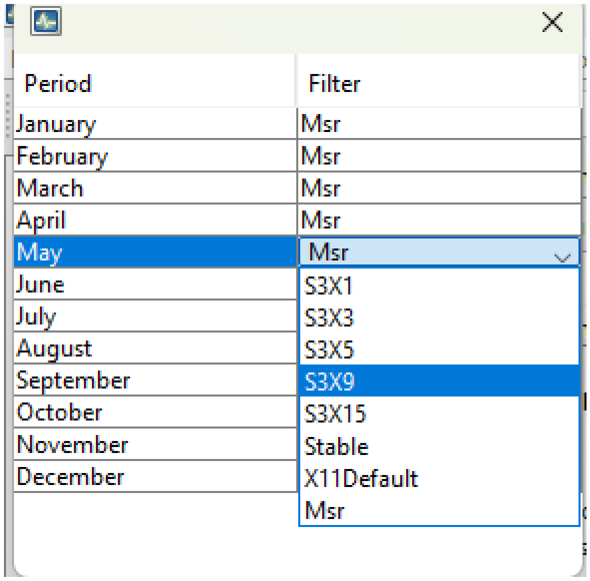


Figure 58: Text

```
    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

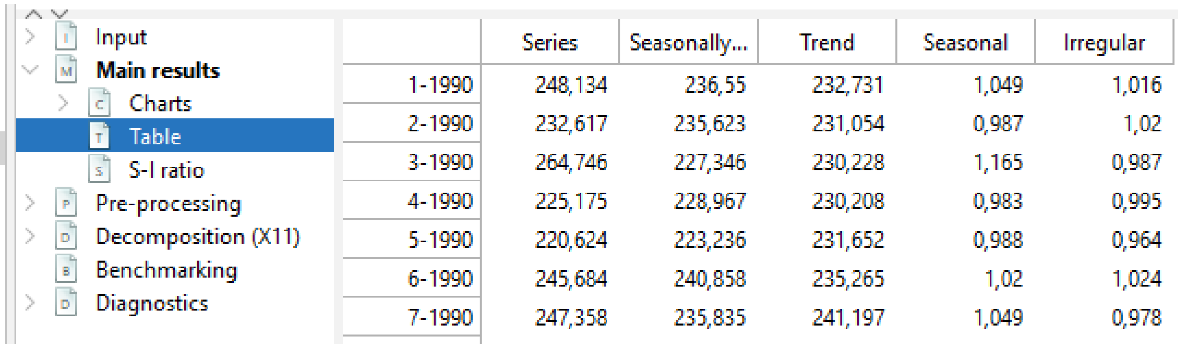
Final series

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

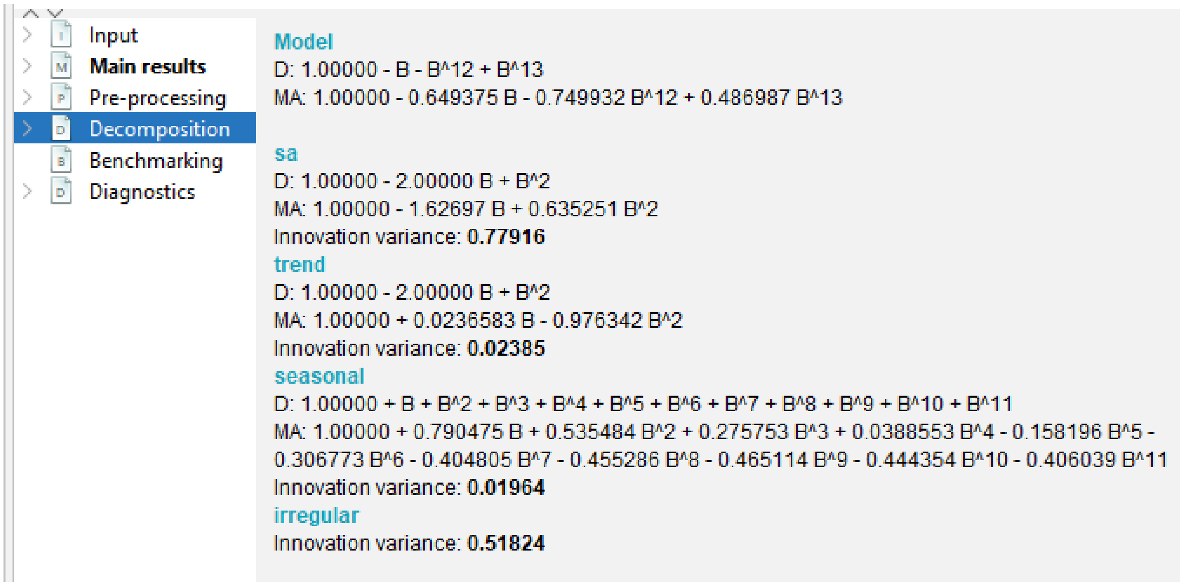


Figure 60: **Decomposition Node**

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

McElroyMatrix

Seting parameters in GUI

In specification window corresponding to a given series:

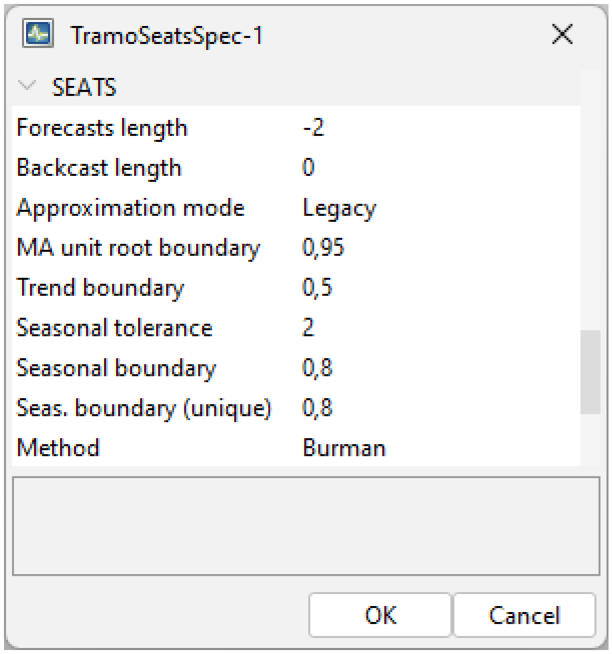


Figure 61: **TramoSeatsSpec Seats part**

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

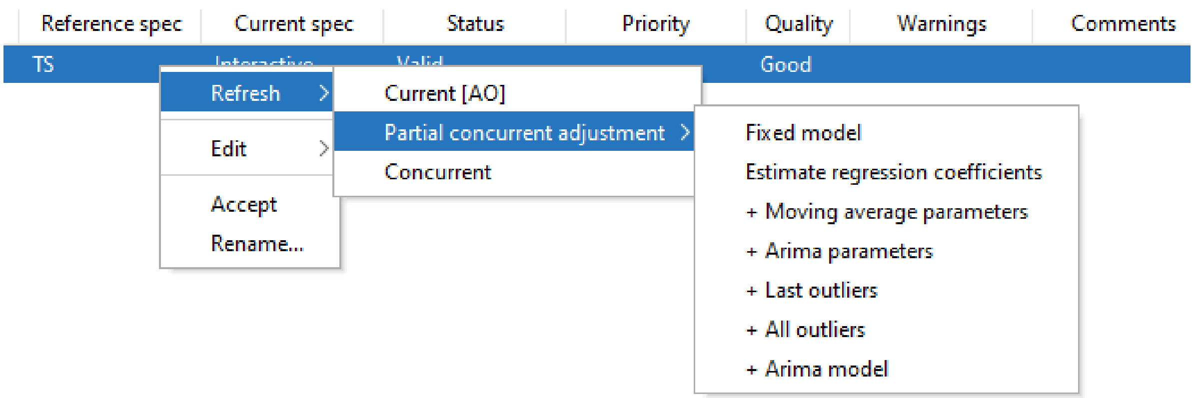
In X-13-ARIMA and Tramo-Seats revision policies are ways to impose constraints on the pre-adjustment phase, while the decomposition phase (X-11 or Seats) will be entirely re-run on new data. The changes induced by X-11 re-estimation stem only from a revised linerarized series, while in Seats they are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

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

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

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:

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

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

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

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

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:

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

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

with

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

Calendar correction

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

```
# Create a calendar with rjd3toolkit
# Define a national calendar
frenchCalendar <- national_calendar(days = list(
    fixed_day(7, 14), # Bastille Day
    fixed_day(5, 8, validity = list(start = "1982-05-08")), # Victory Day
    special_day("NEWYEAR"),
    special_day("CHRISTMAS"),
    special_day("MAYDAY"),
    special_day("EASTERMONDAY"),
    special_day("ASCENSION"),
    special_day("WHITMONDAY"),
    special_day("ASSUMPTION"),
    special_day("ALLSAINTSDAY"),
    special_day("ARMISTICE")
))
# Generrate calendar regressors
q <- holidays(
    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 $\mathbb{I}\{E(p)$ even$\} + (p - E(p))/2p$
- example for $p = 12$: ($1/12$ and $1/24$) (we fall back on $M_{2\times 12}$ of the monthly case
- example for $p = 365.25$: ($1/365.25$ and $0.25/(2 * 365.25)$)

Modification of seasonality extraction filters

Computation is done on a given period

Example $M_{3\times 3}$

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

if p integer: no changes needed

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

Modification of final trend estimation filter

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

Example of decomposition

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

```
# 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)

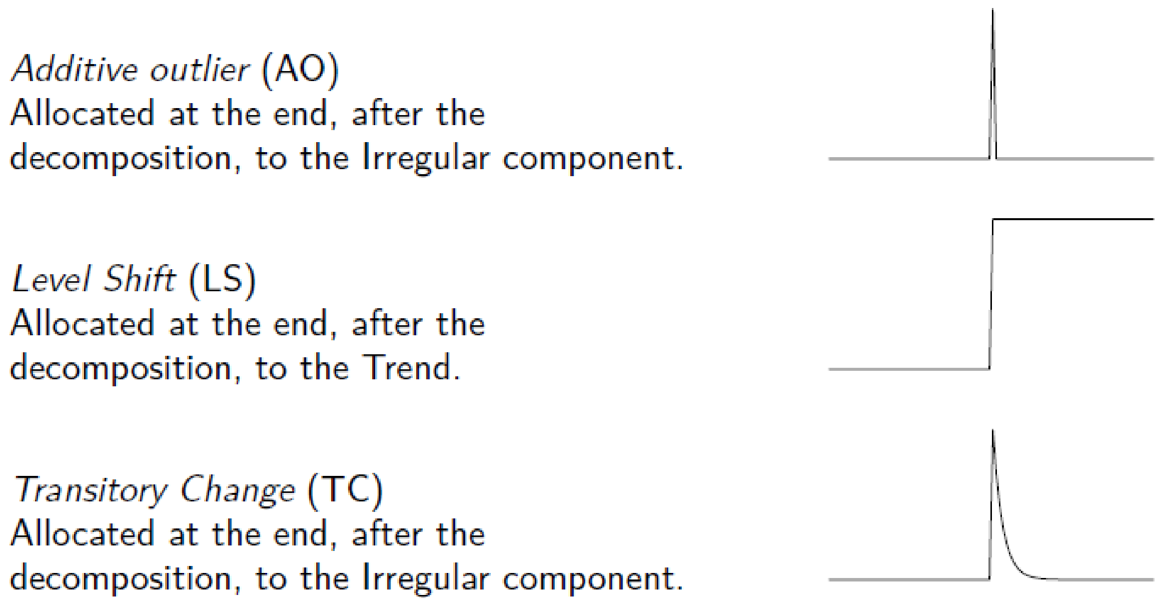


Figure 63: **Outliers type**

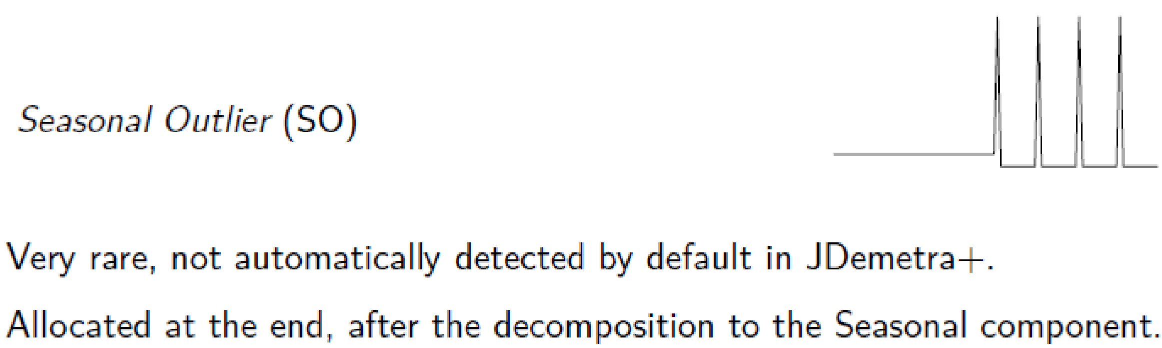


Figure 64: **Seasonal outlier**

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.

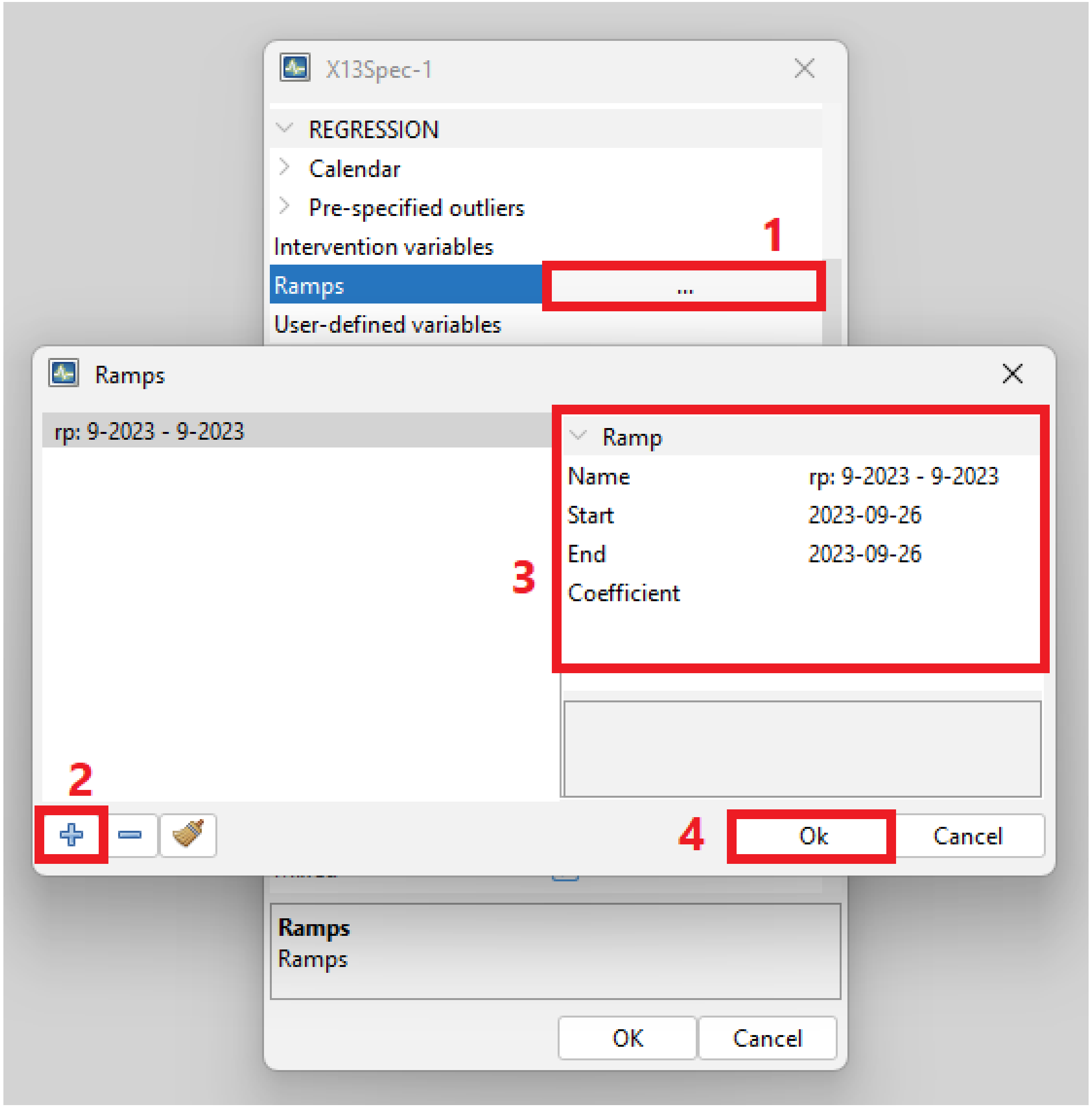


Figure 65: **Add ramp in GUI**

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.

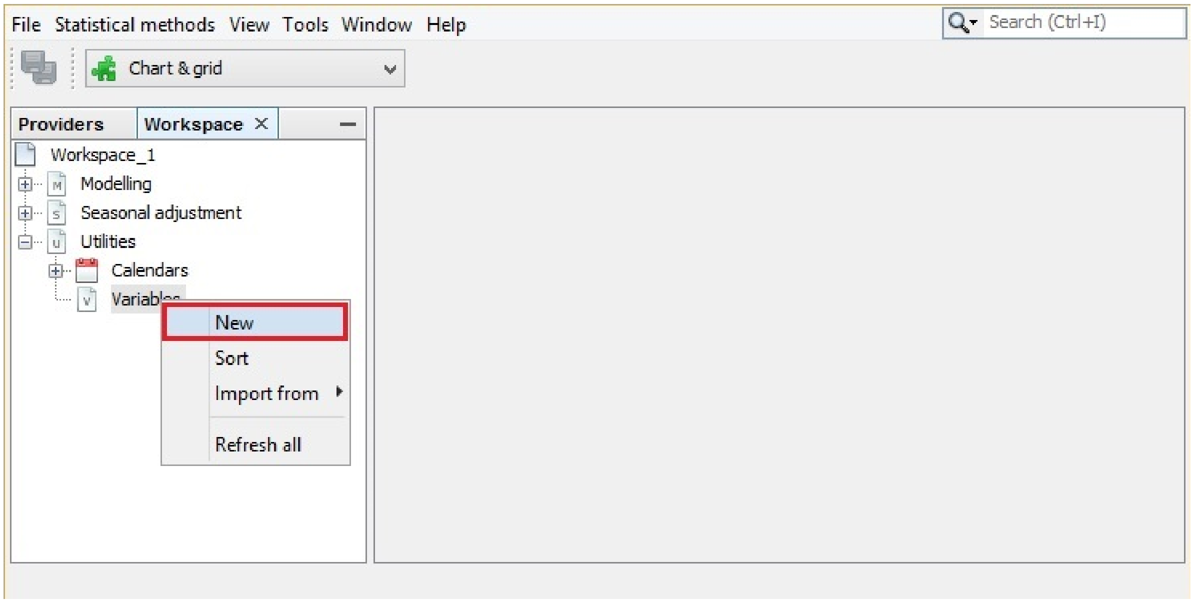


Figure 66: **Creating an empty dataset for the user-defined variables**

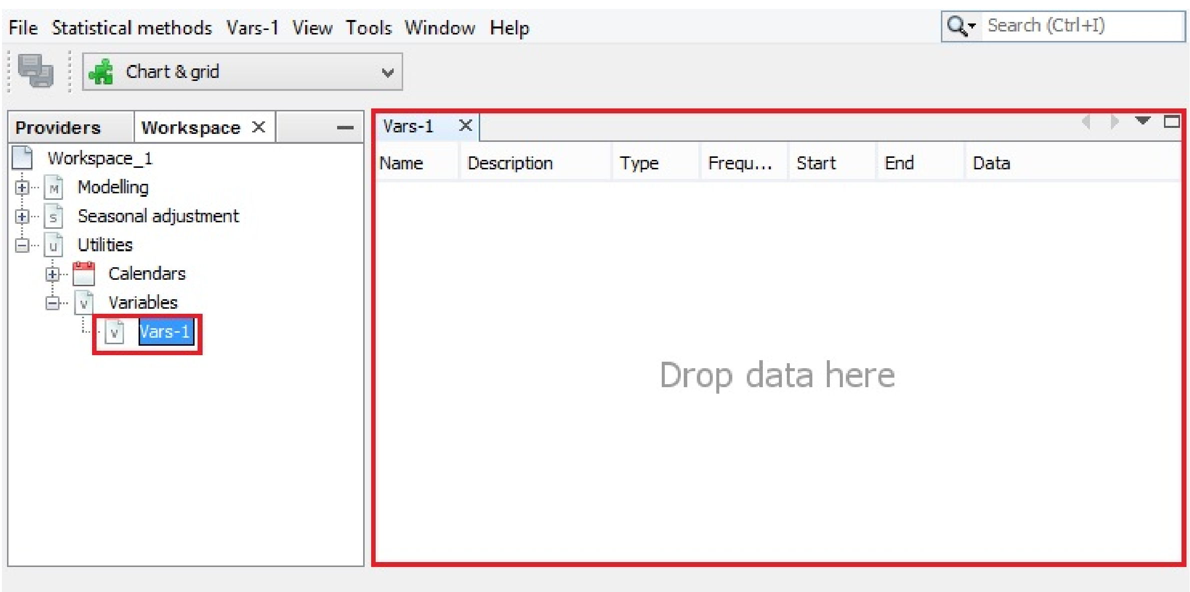


Figure 67: **Activation of an empty dataset for the user-defined variables**

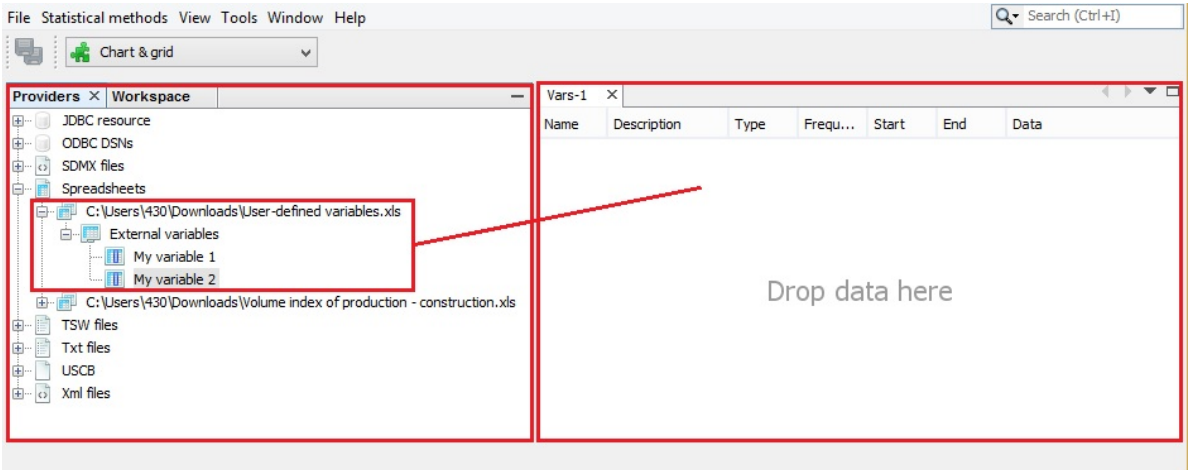


Figure 68: **Importing the user-defined variables to JDemetra+**

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

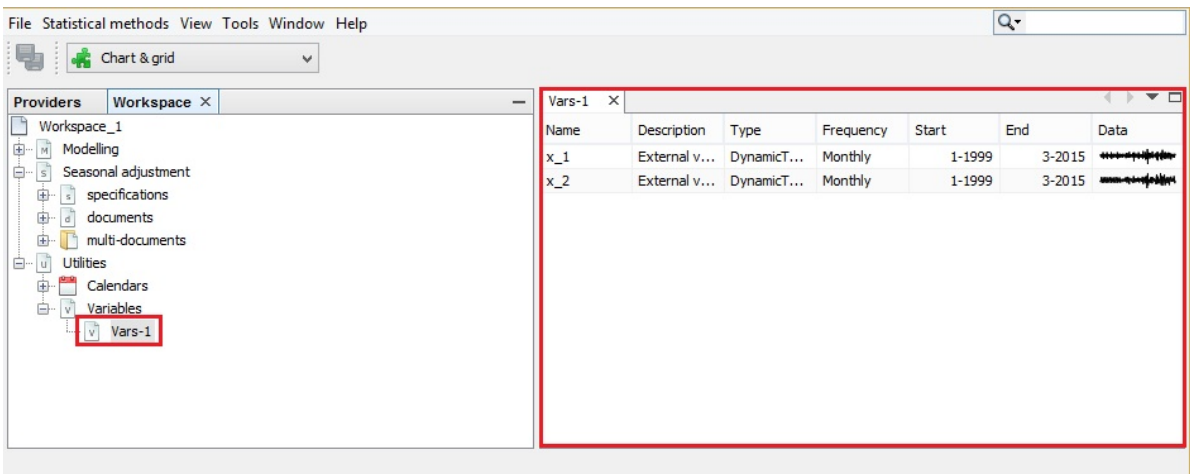


Figure 69: **Assigning regressors from the *Providers* window to the user-defined variables**

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

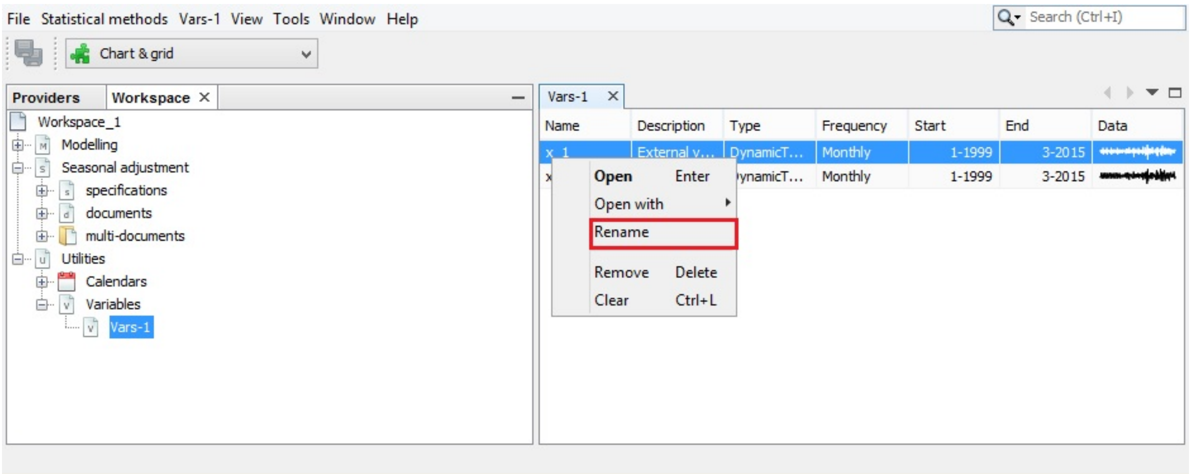


Figure 70: **A local menu for the user-defined variables**

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),
```

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

Example wit rjd3tramoseats::tramo_outliers

```r
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 $\bar{D^y_{i,J}}$ the long term average of the number of days in the group i for the periods J (for instance, average number of Mondays in January...).

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

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

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

Weights for different groups of days

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

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

Use in 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.

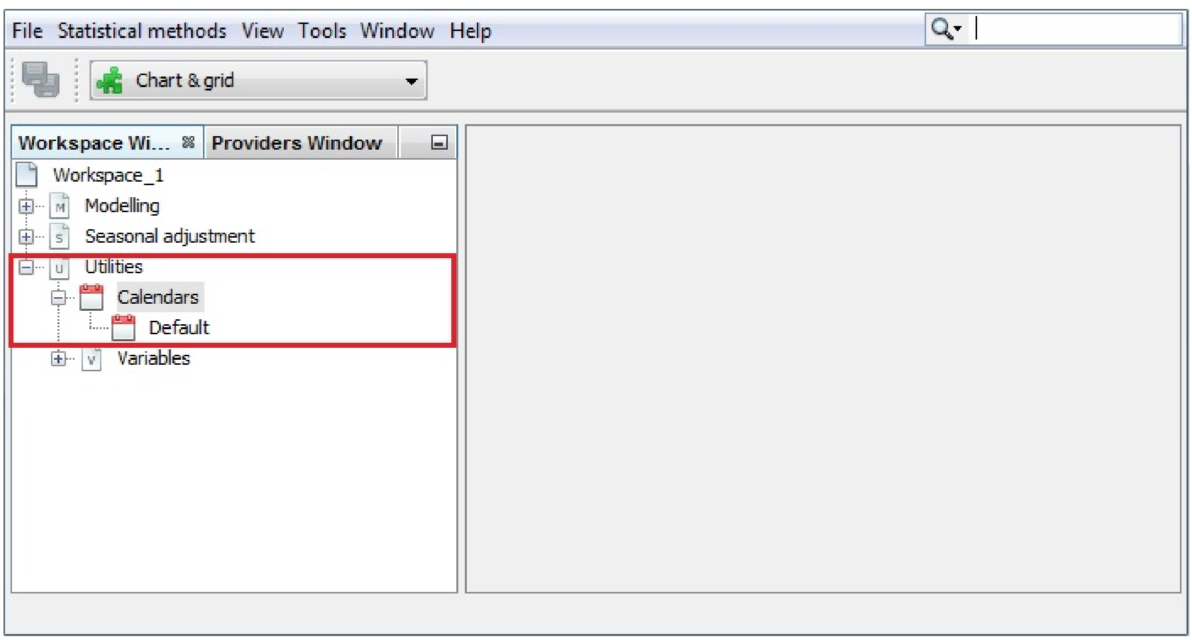


Figure 71: Text

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

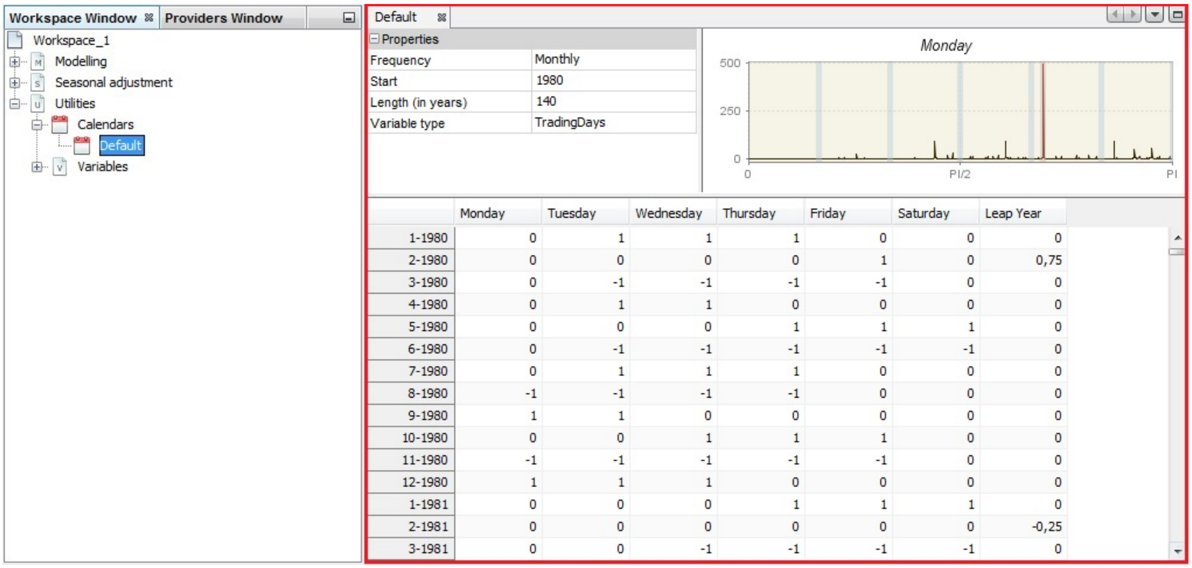


Figure 72: Text

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.

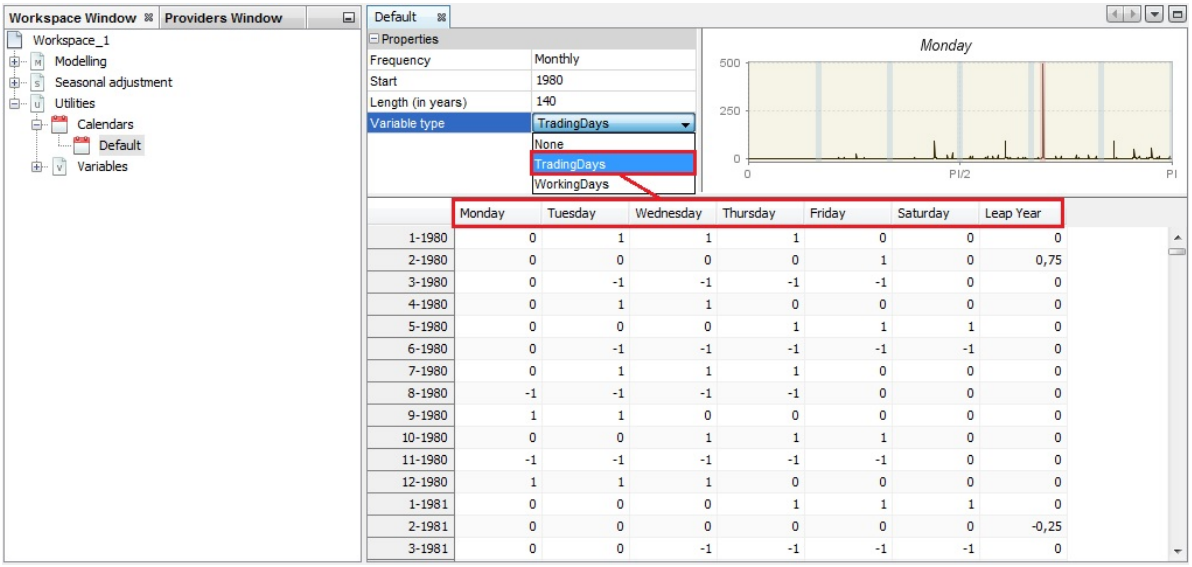


Figure 73: Text

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

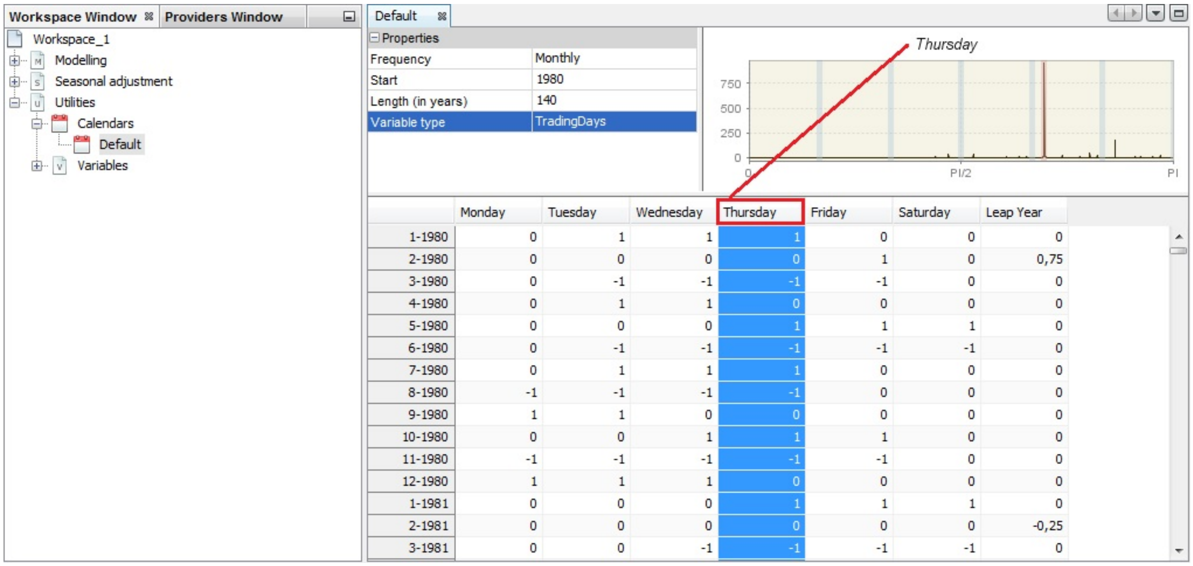


Figure 74: Text

- *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)
```

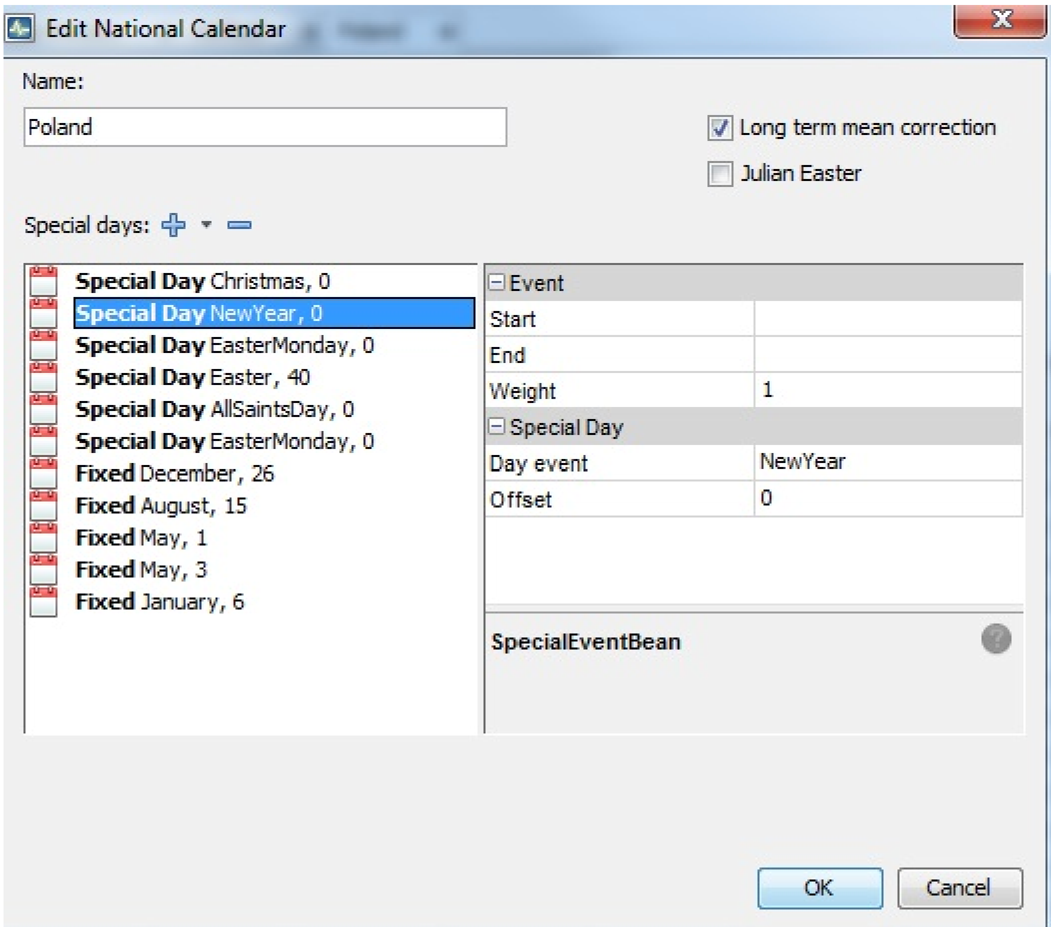


Figure 75: **Edit a calendar window**

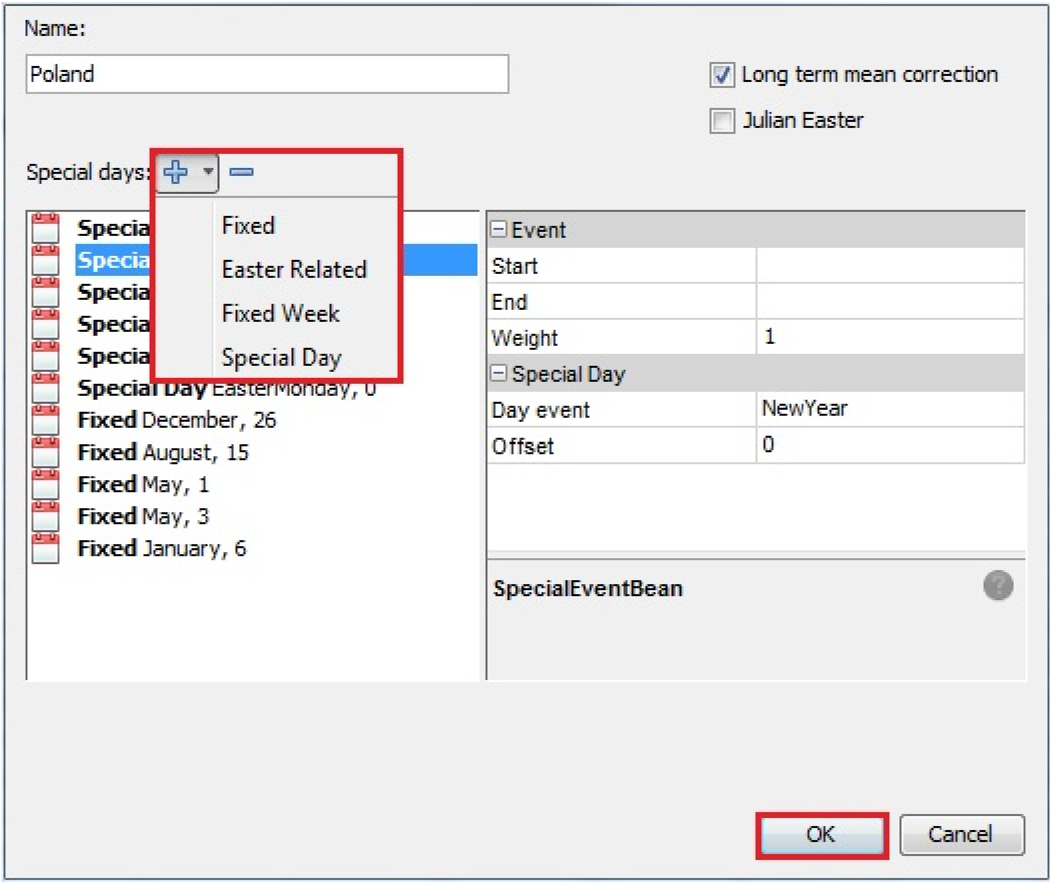


Figure 76: **Removing a holiday from the calendar**

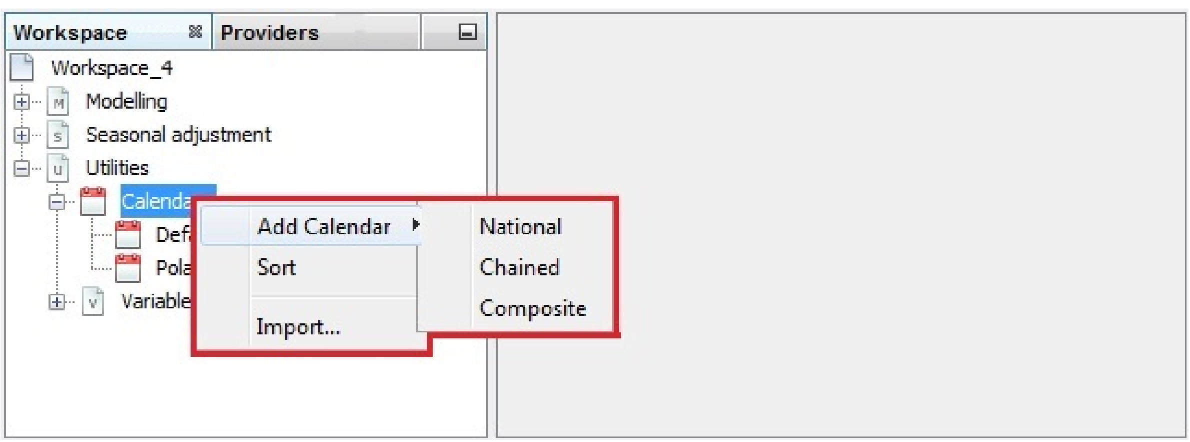


Figure 77: Text

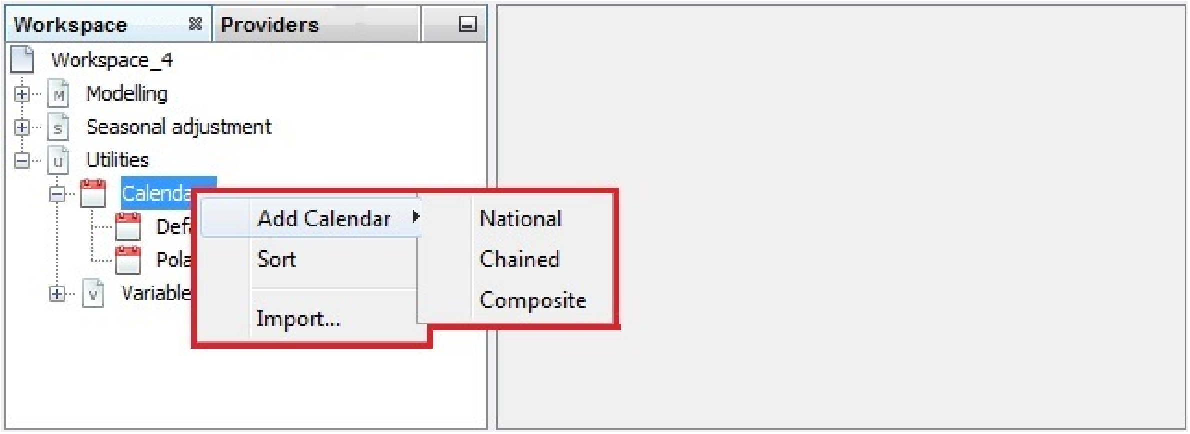


Figure 78: Text

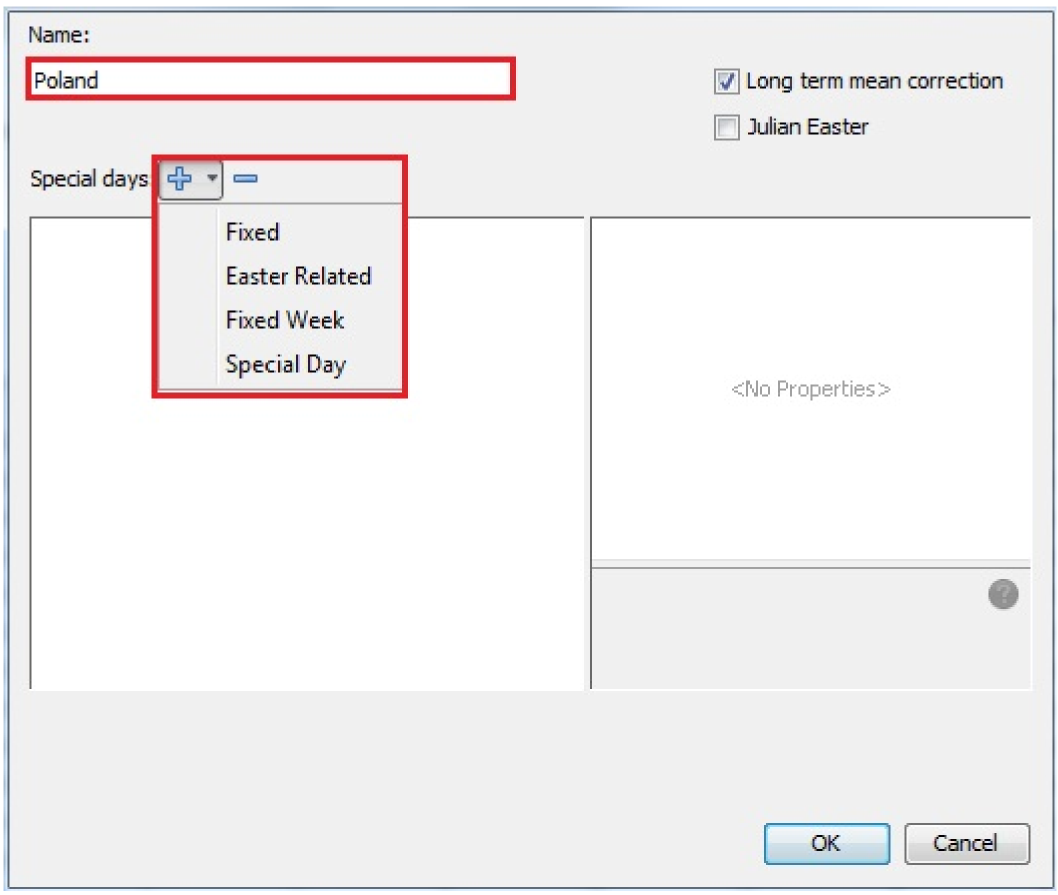


Figure 79: Text

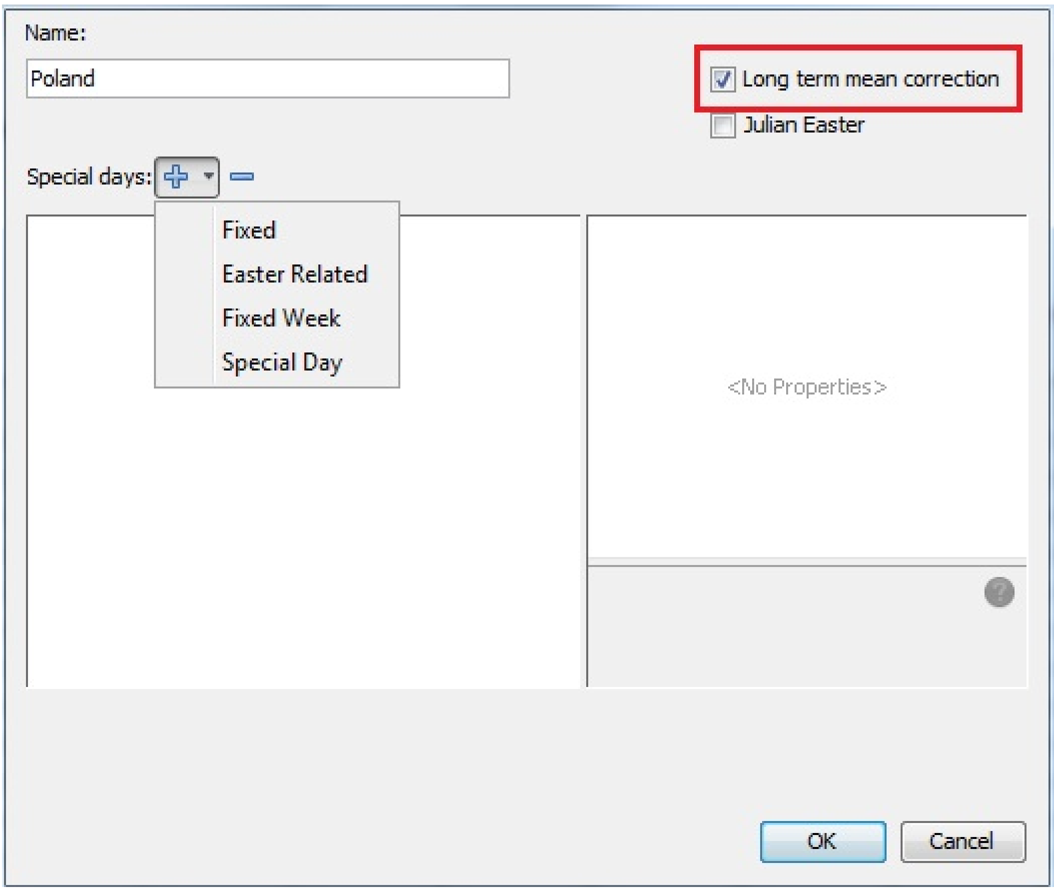


Figure 80: Text

- to use Julian Easter

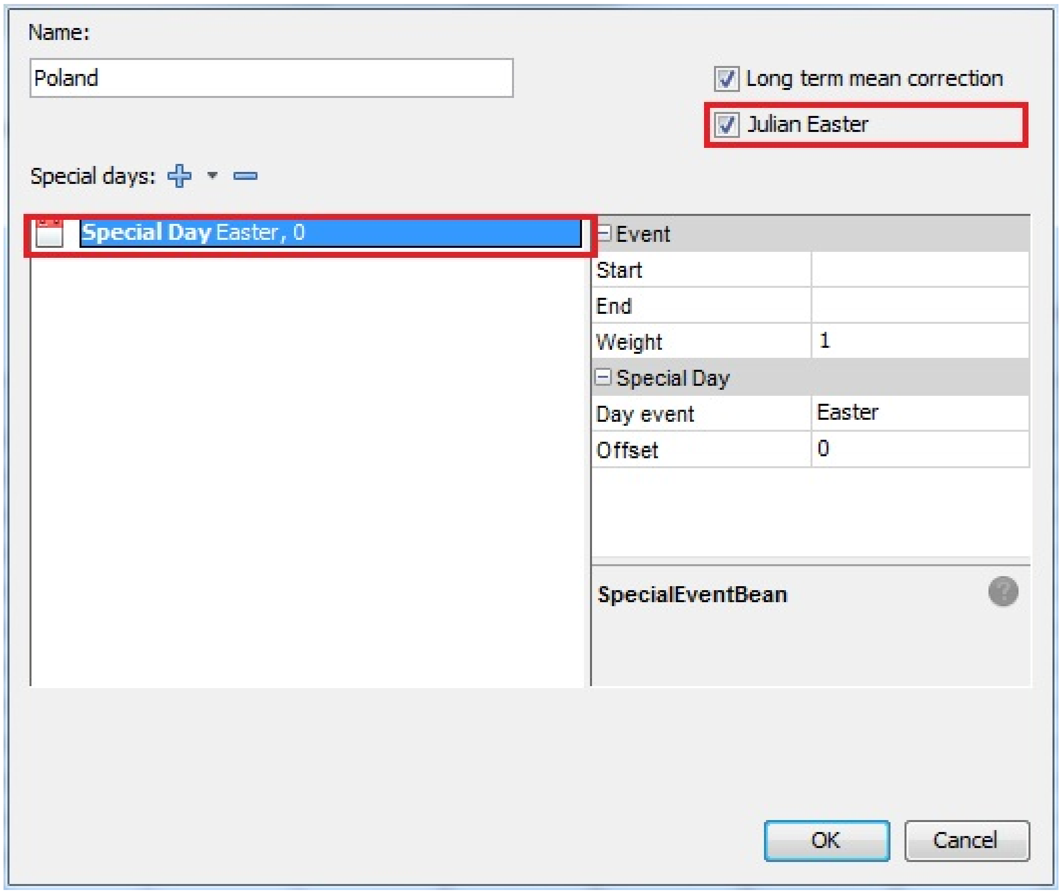


Figure 81: Text

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

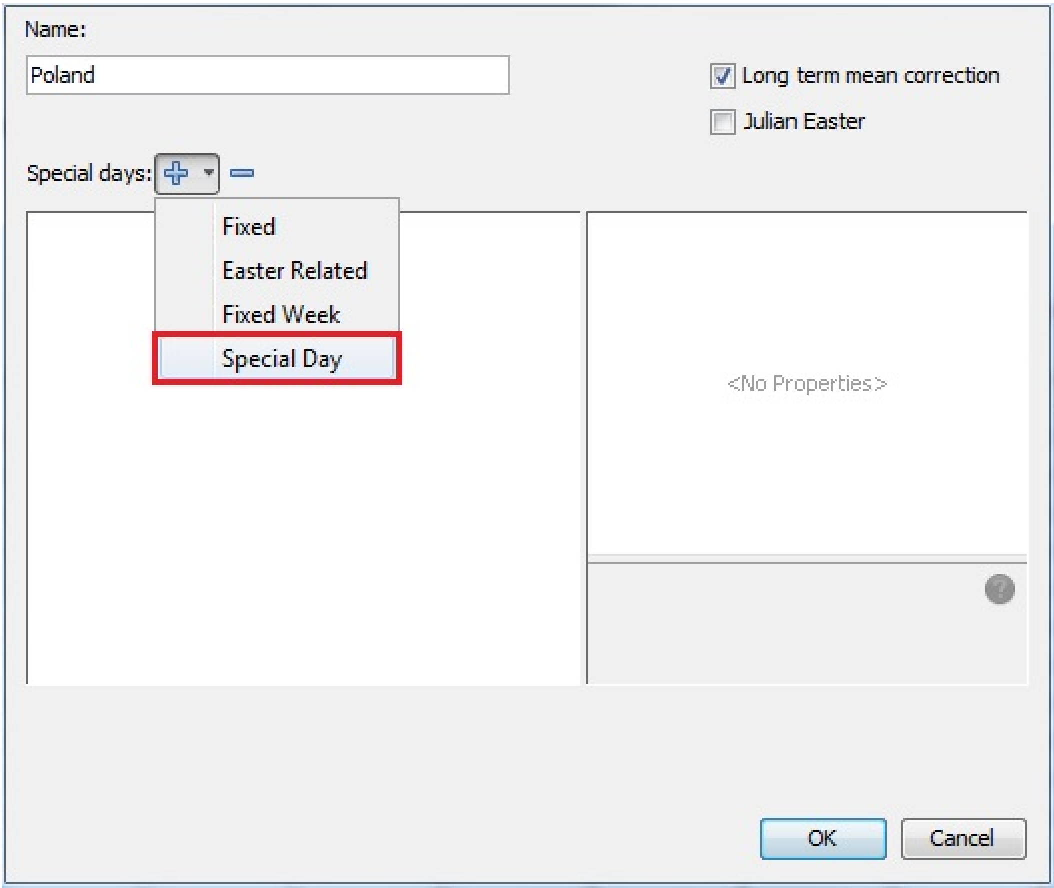


Figure 82: **Adding a pre-defined holiday to the calendar**

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.

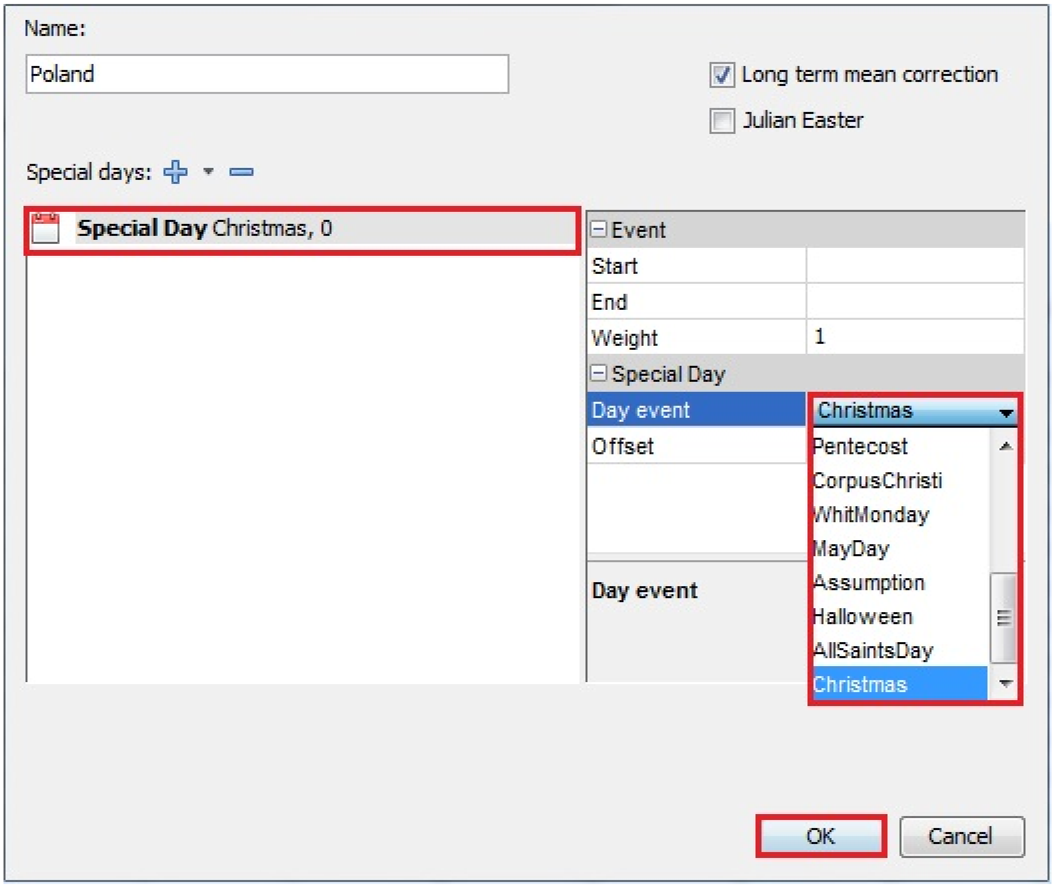


Figure 83: **Choosing a pre-defined holiday from the list**

- 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.

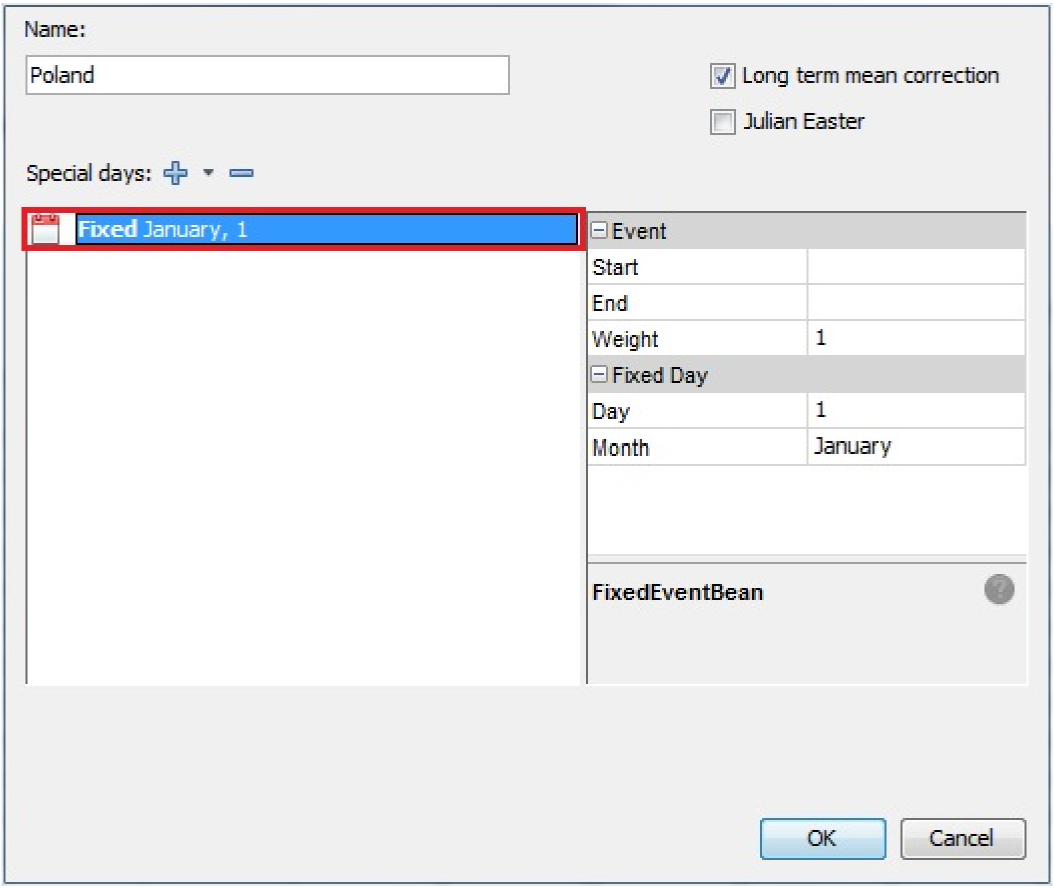
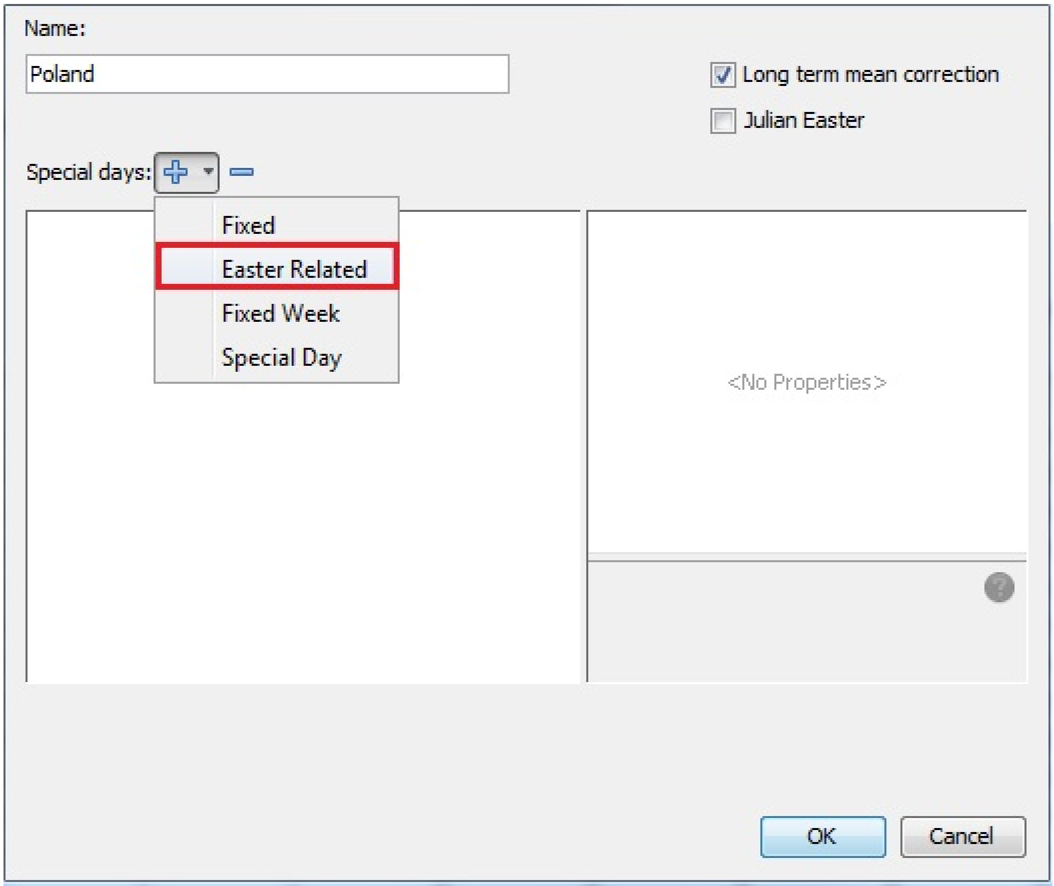


Figure 84: **Options for a fixed holiday**

- 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.

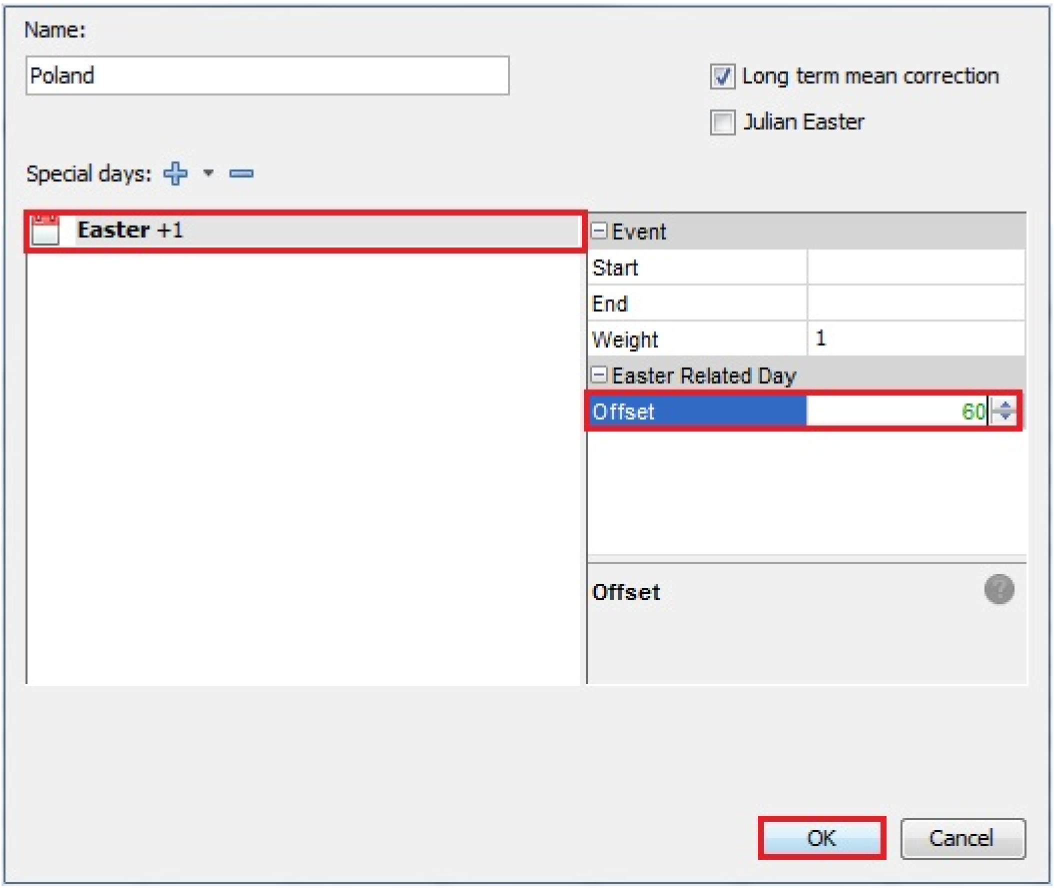


Figure 85: Text

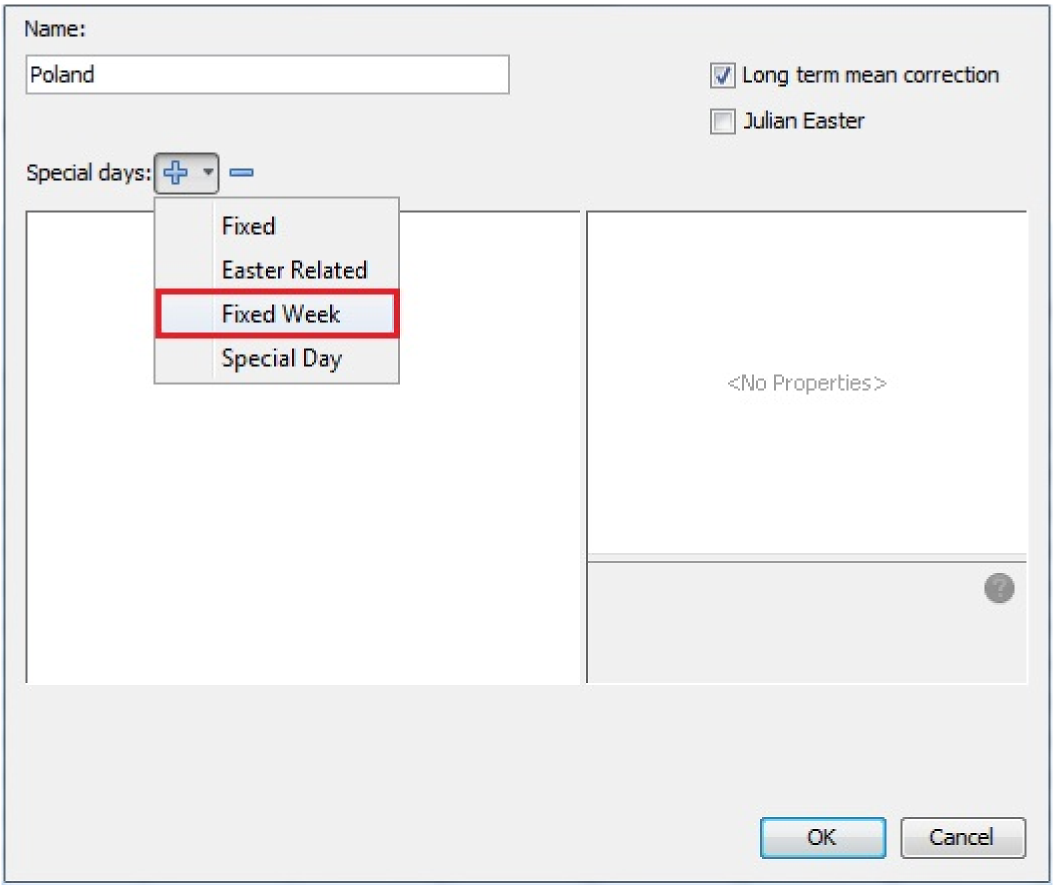


Figure 86: Text

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

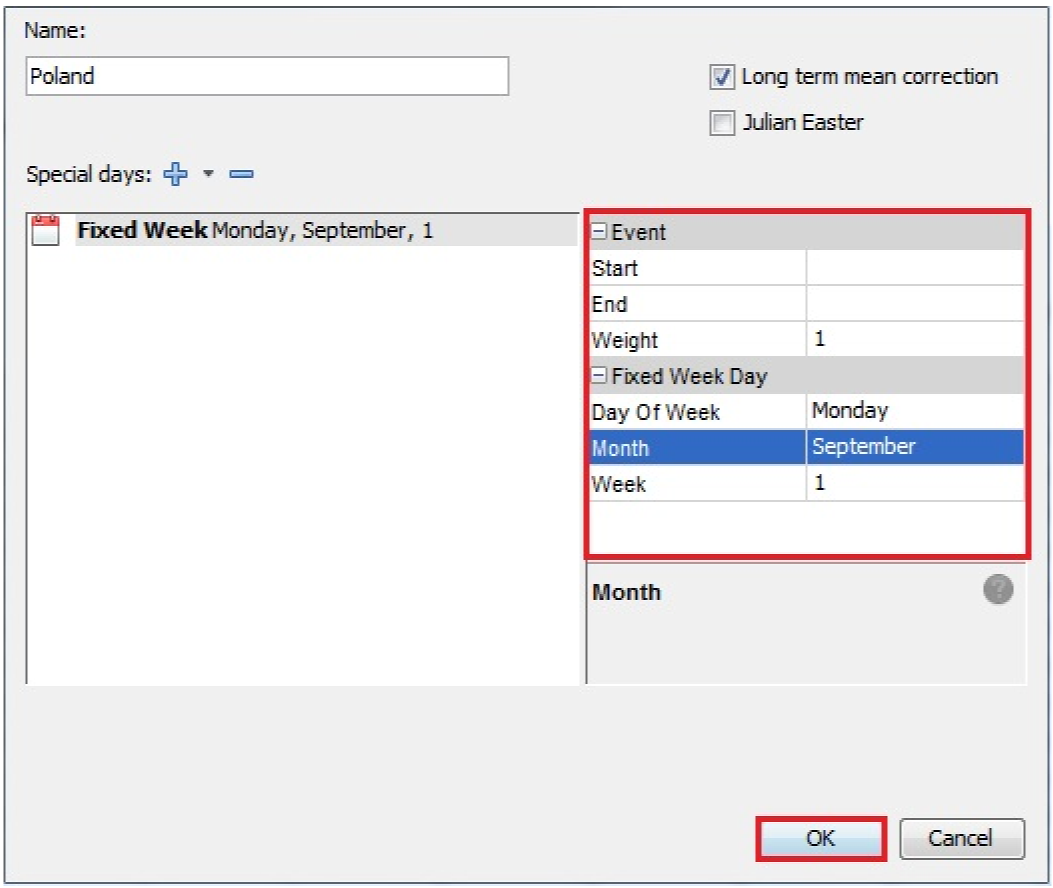


Figure 87: Text

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

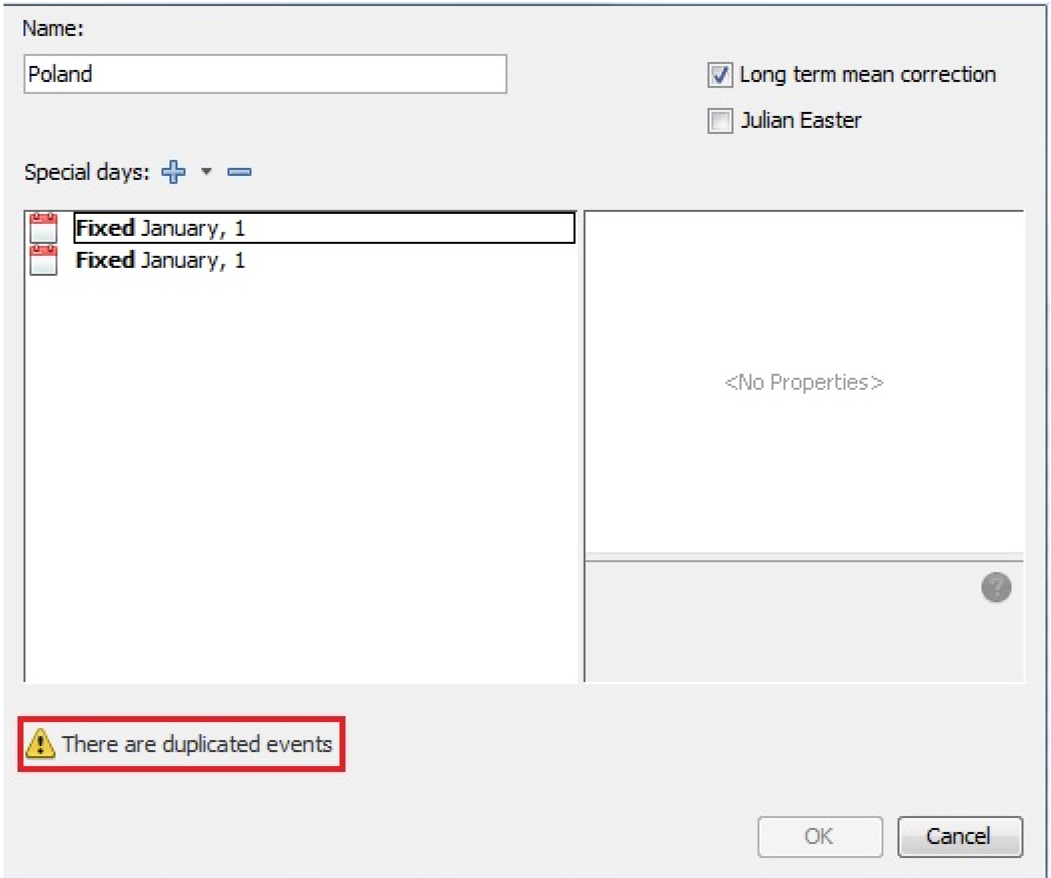


Figure 88: Text

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

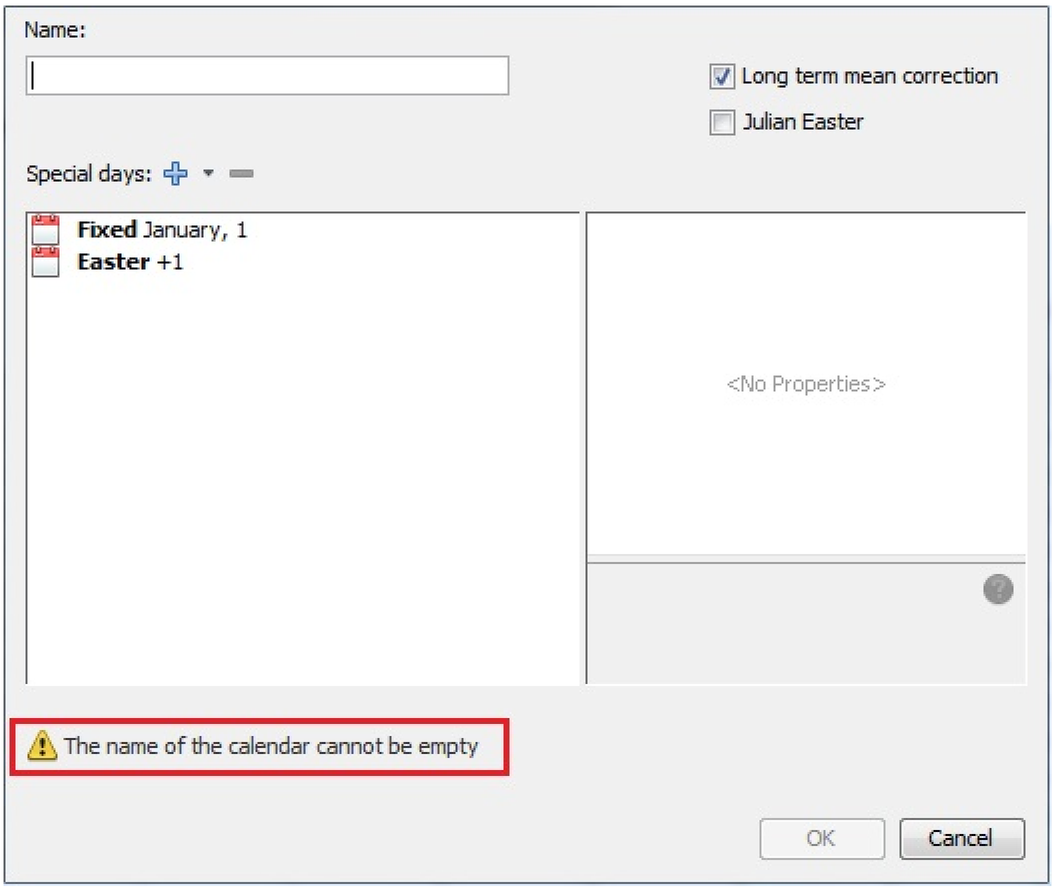


Figure 89: Text

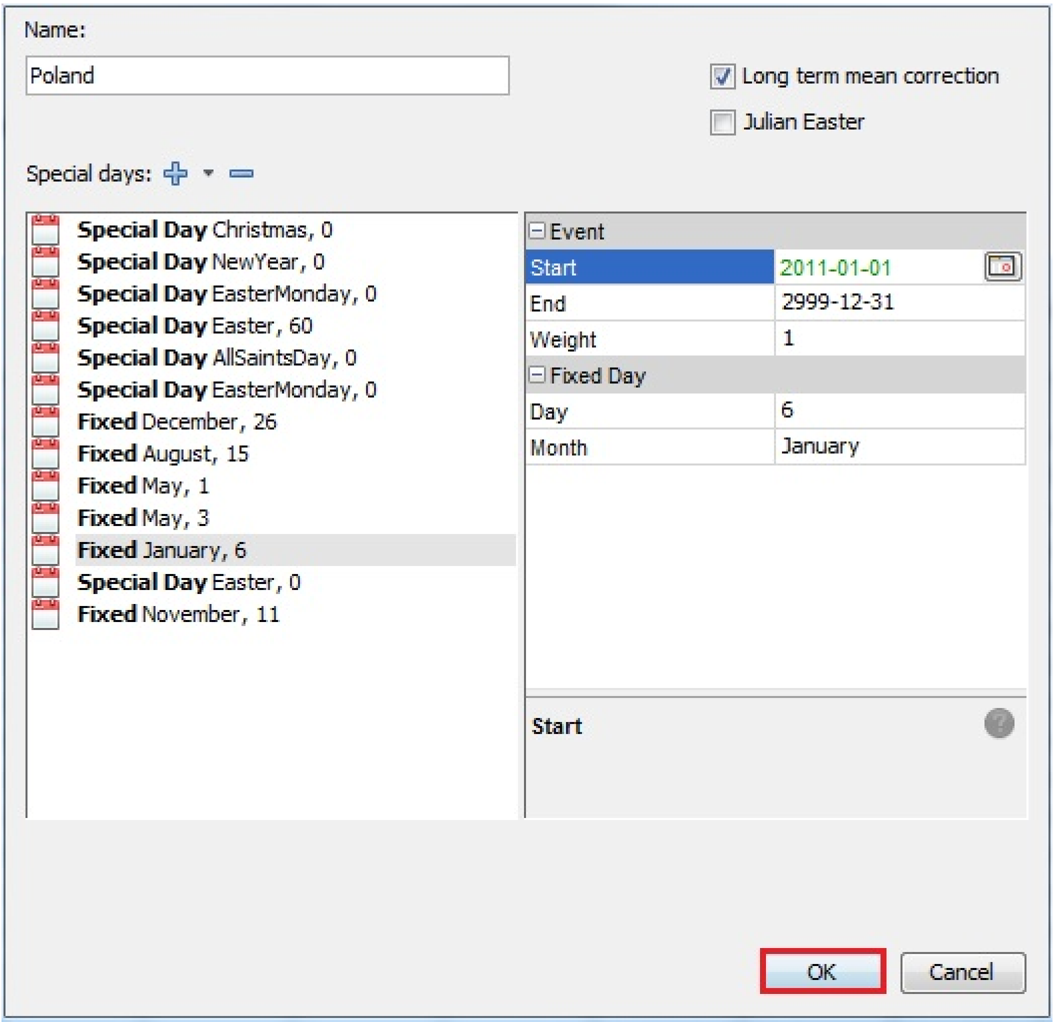


Figure 90: Text

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

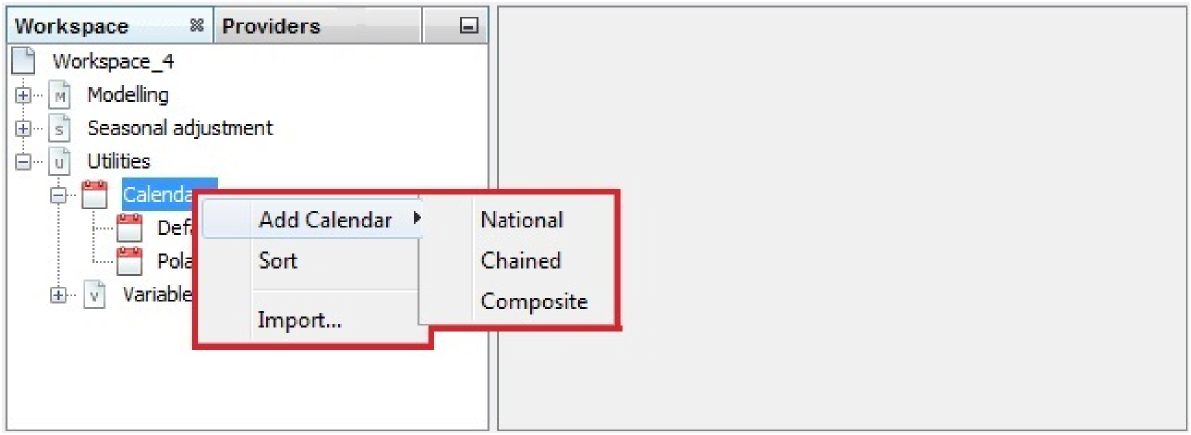


Figure 91: Text

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

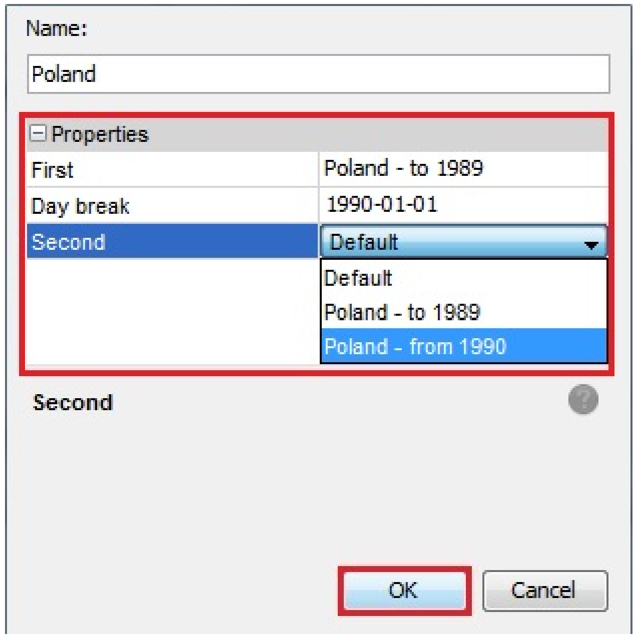


Figure 92: Text

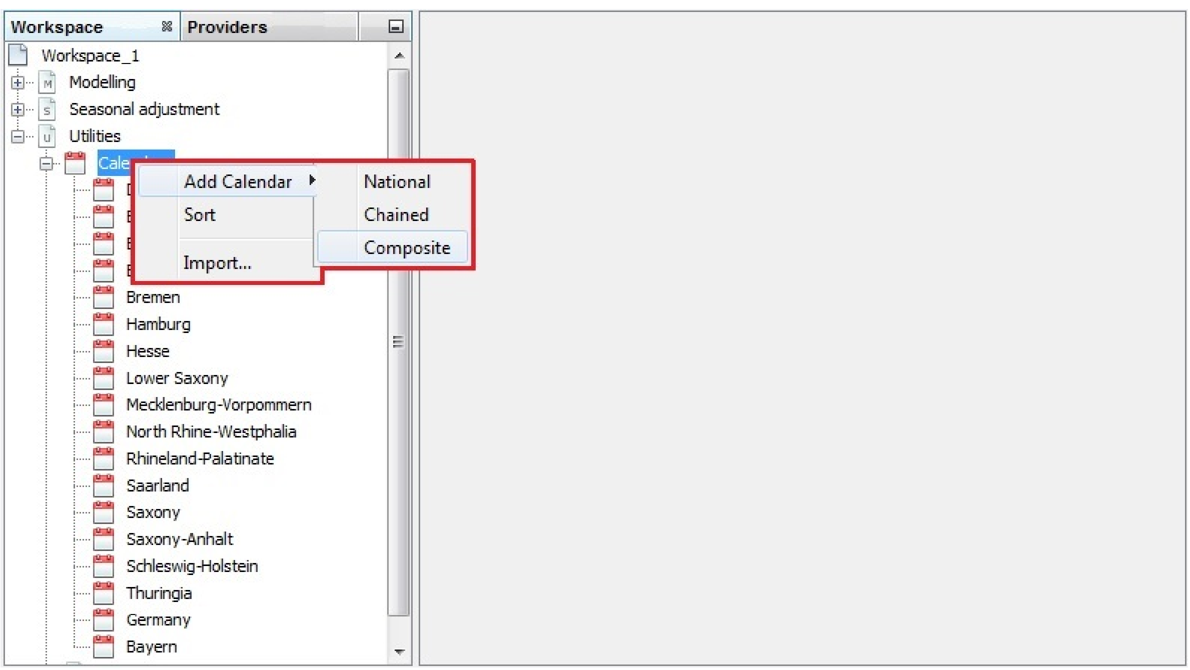


Figure 93: Text

- Assign a weight to each calendar.

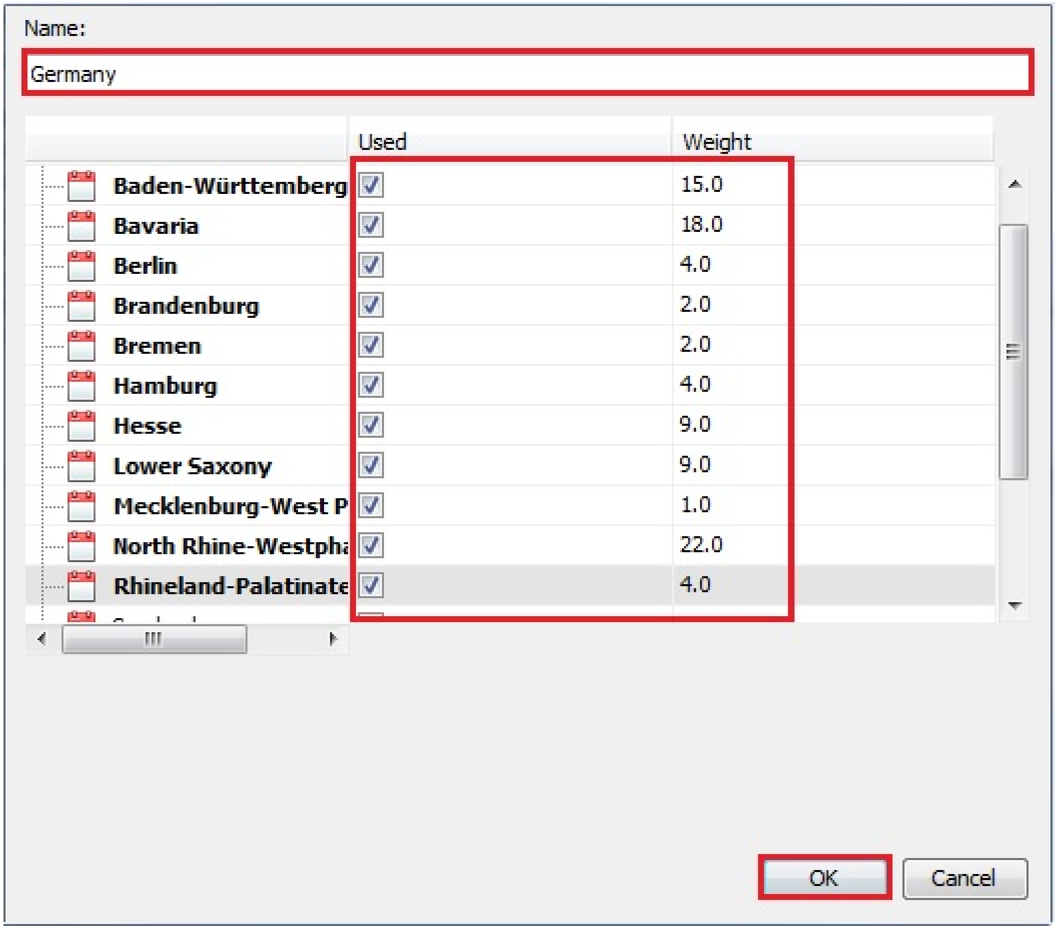


Figure 94: Text

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

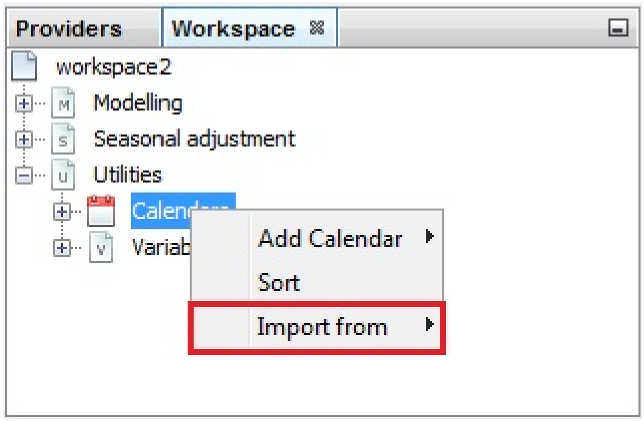


Figure 95: **Importing a calendar to JDemetra+**

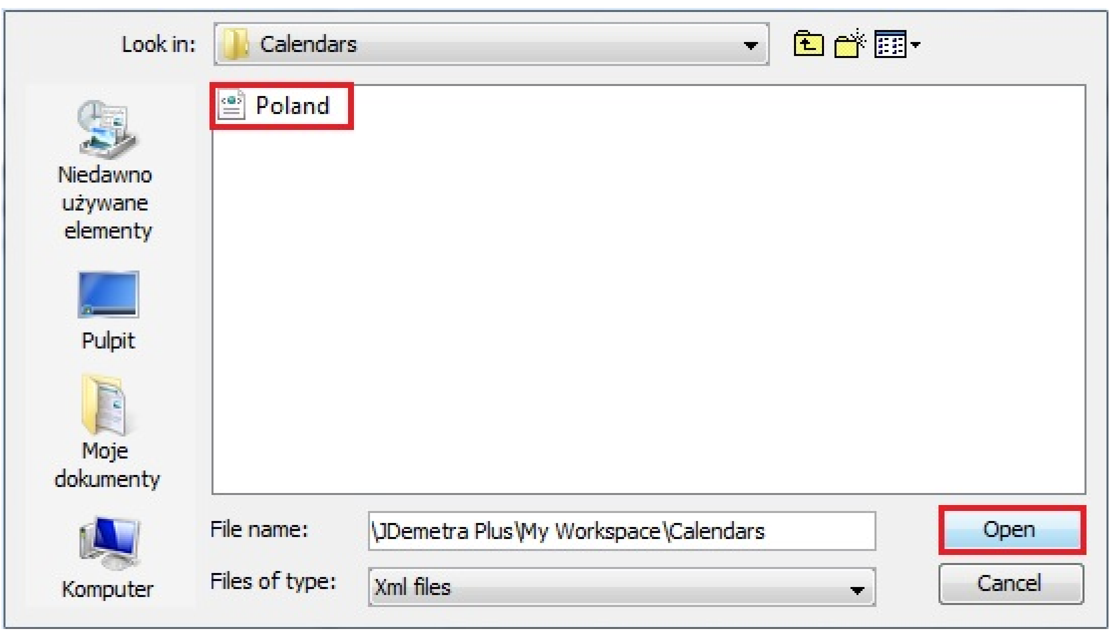


Figure 96: **Choosing the file**

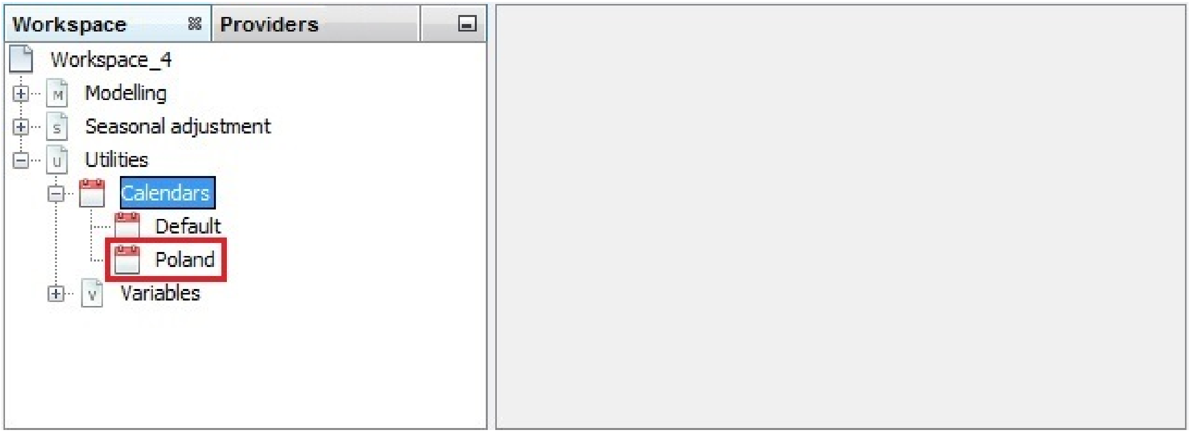


Figure 97: **A list of calendars with a newly imported calendar**

```
  <?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 disaggregation

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

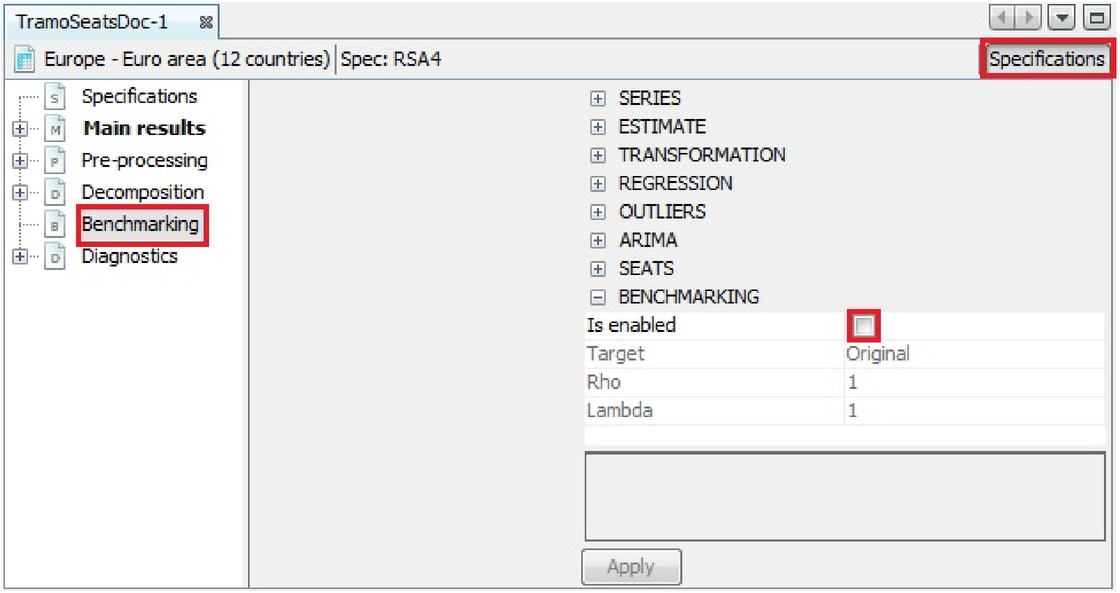
Benchmarking seasonally adjusted data

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

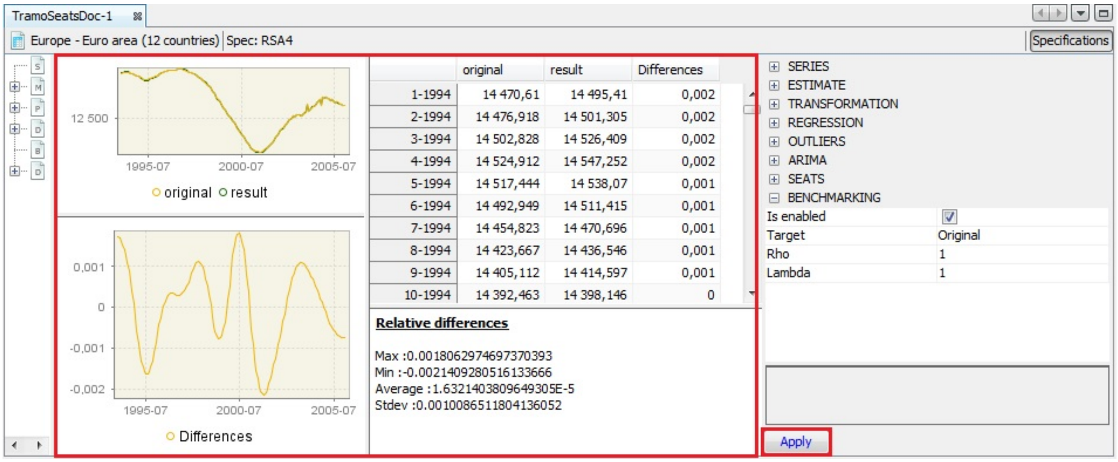
When running a seasonal adjustment process

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.

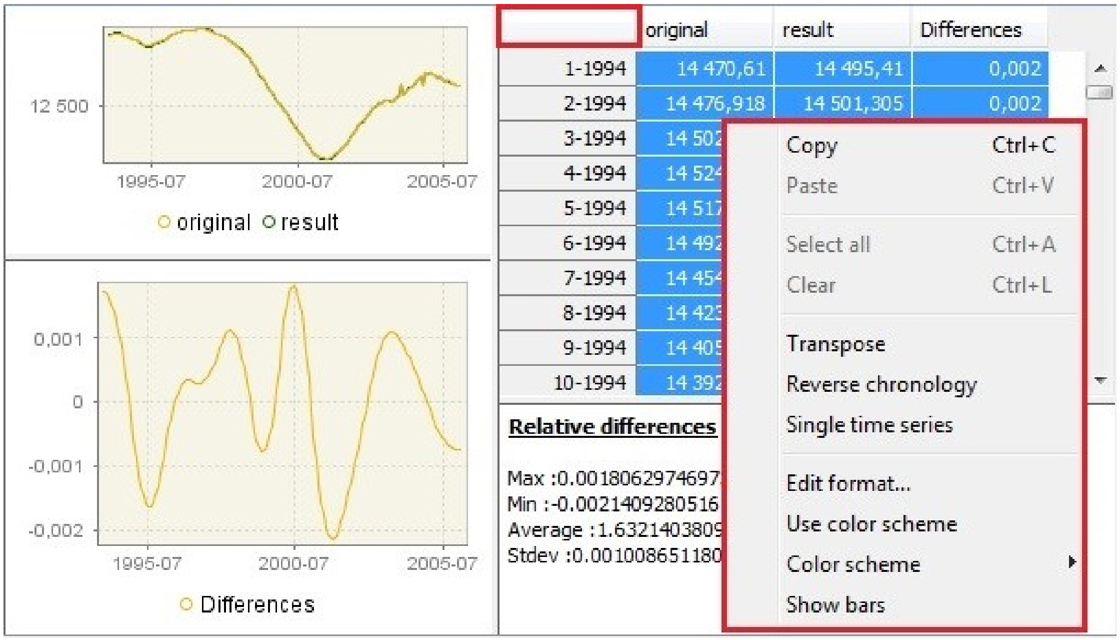


2. Three parameters can be set here. *Target* specifies the target variable for the benchmarking procedure. It can be either the *Original* (the raw time series) or the *Calendar Adjusted* (the time series adjusted for calendar effects). *Rho* is a value of the AR(1) parameter (set between 0 and 1).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

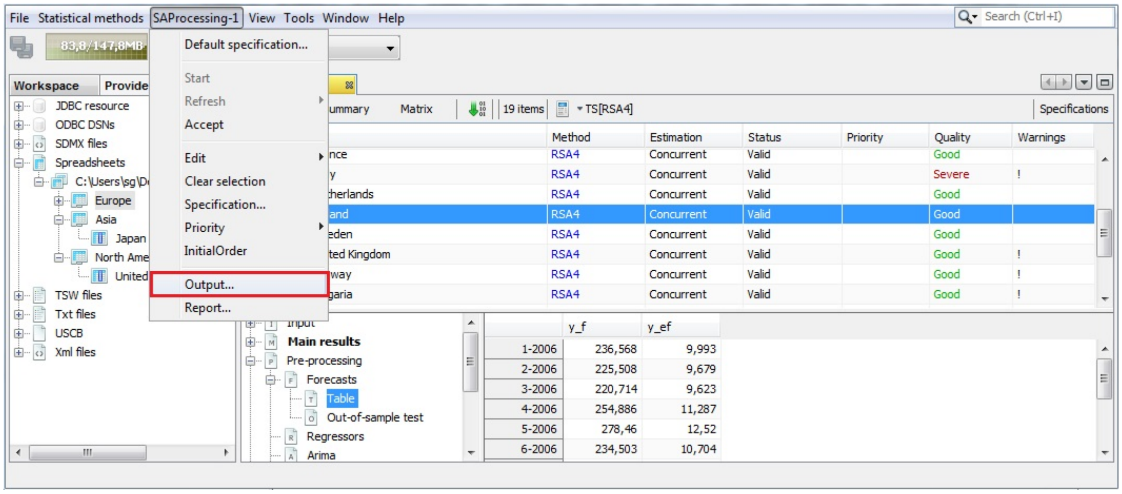
3. To launch the benchmarking procedure click on the **Apply** button. The results are displayed in four panels. The top-left one compares the original output from the seasonal adjustment procedure with the result from applying a benchmarking to the seasonal adjustment. The bottom-left panel highlights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



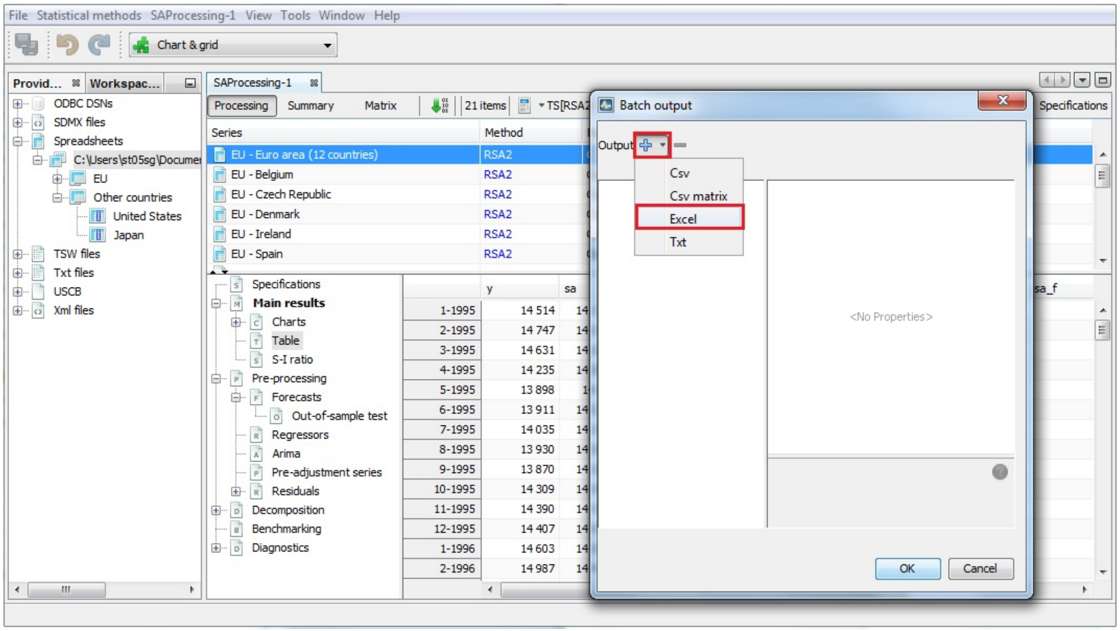
4. Both pictures and the table can be copied the usual way



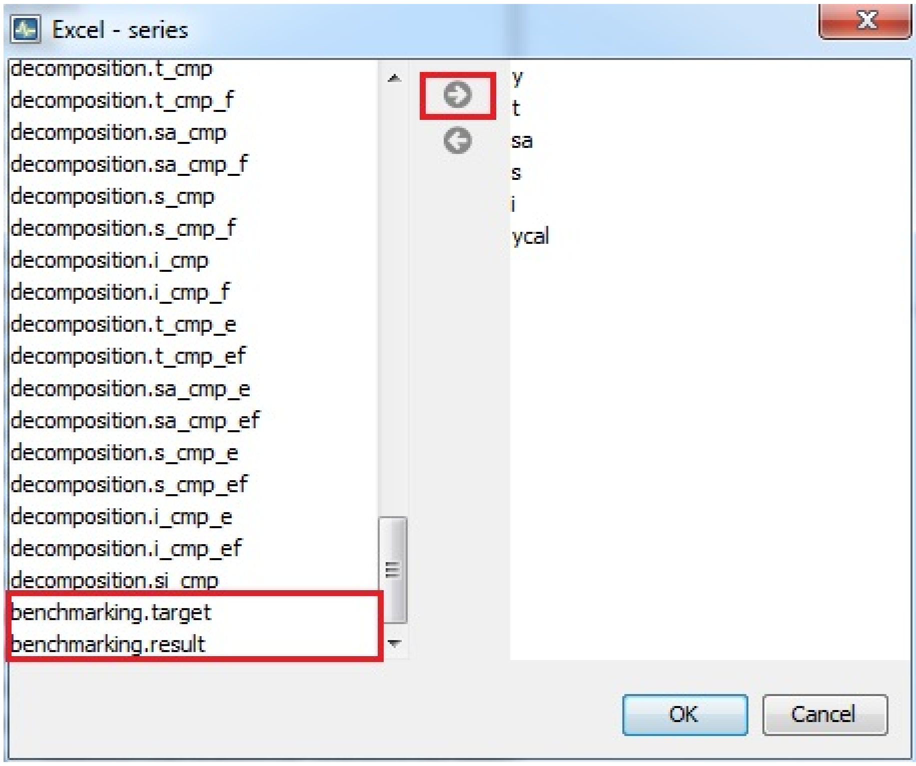
5. To export the result of the benchmarking procedure (*benchmarking.result*) and the target data (*benchmarking.target*) one needs to once execute the seasonal adjustment with benchmarking



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.

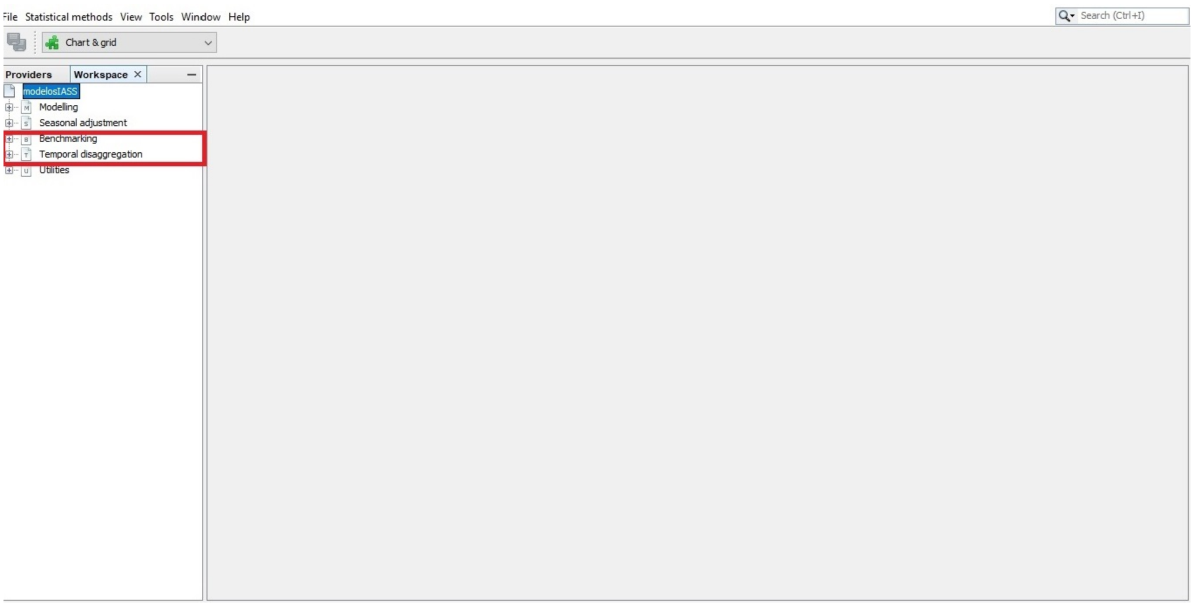


Figure 98: Text

Univariate: Denton and Cholette

To run Denton univariate case select:

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

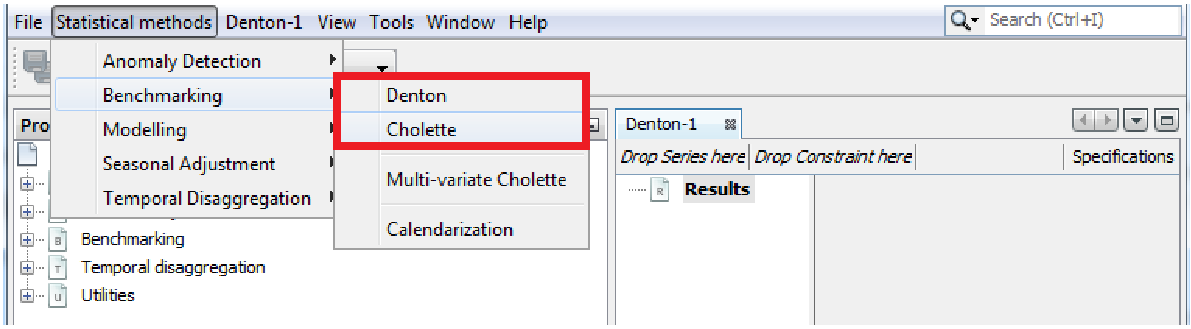


Figure 99: **Benchmarking tab**

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.

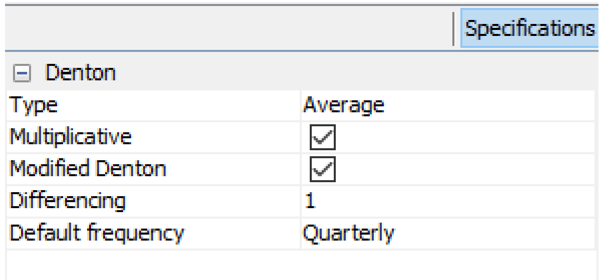


Figure 100: Denton Specifications

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).

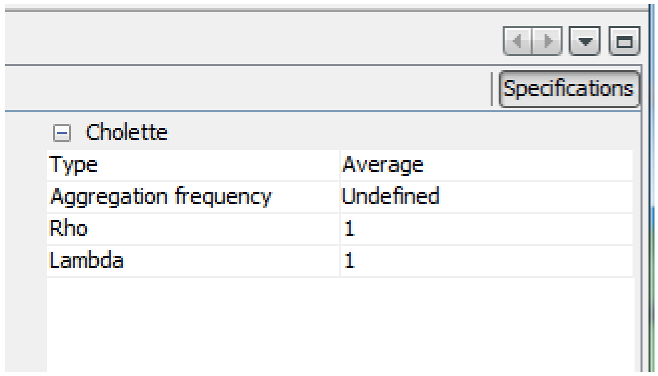


Figure 101: Cholette Specifications

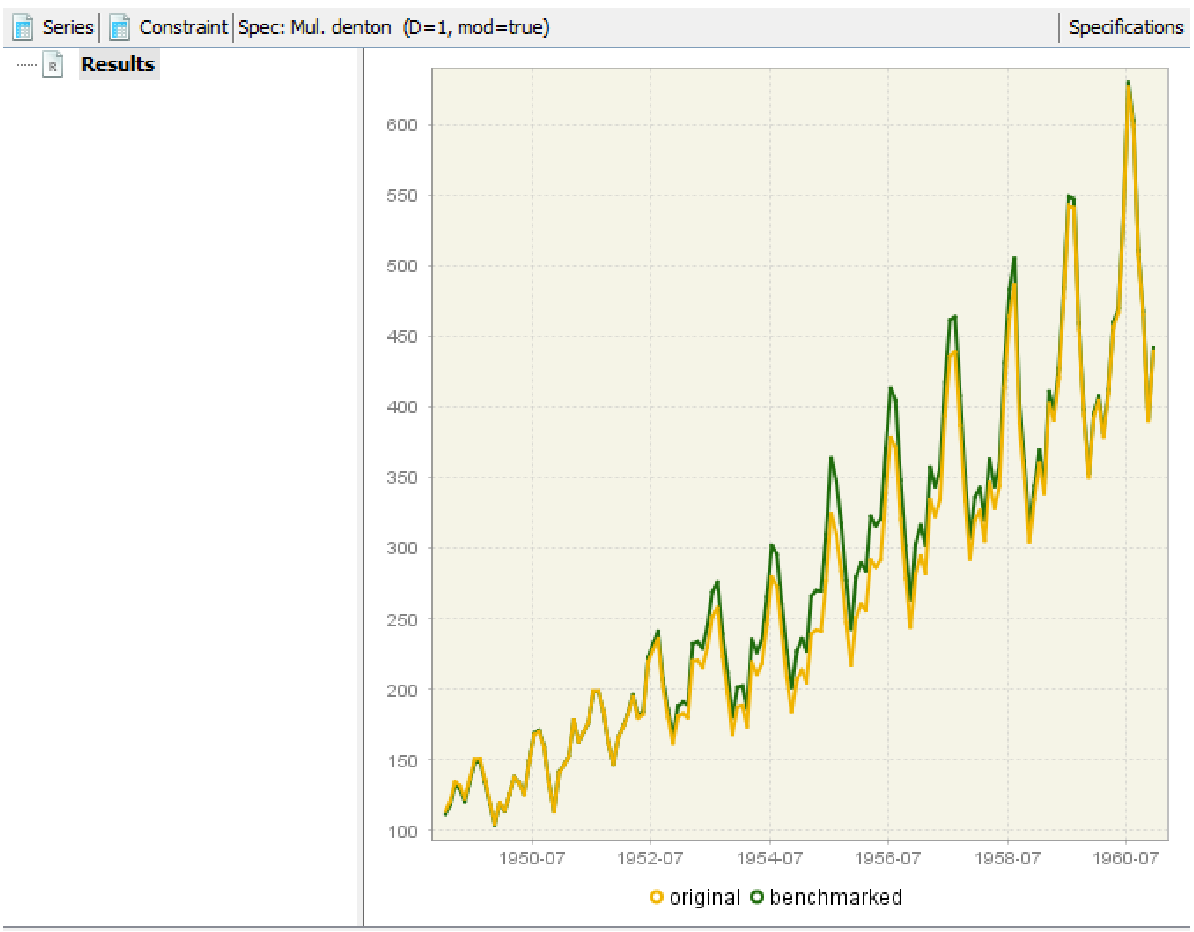


Figure 102: Denton output

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.

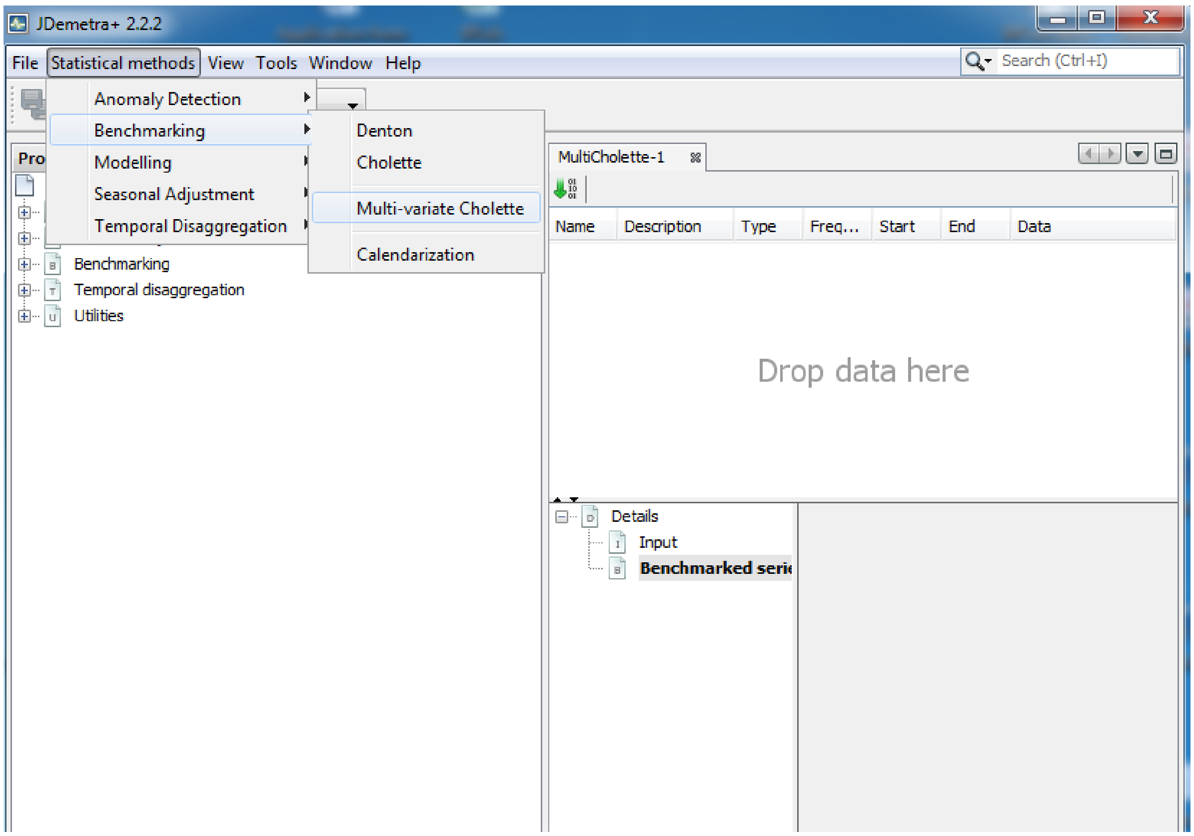


Figure 103: Multi-variate Cholette input

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.

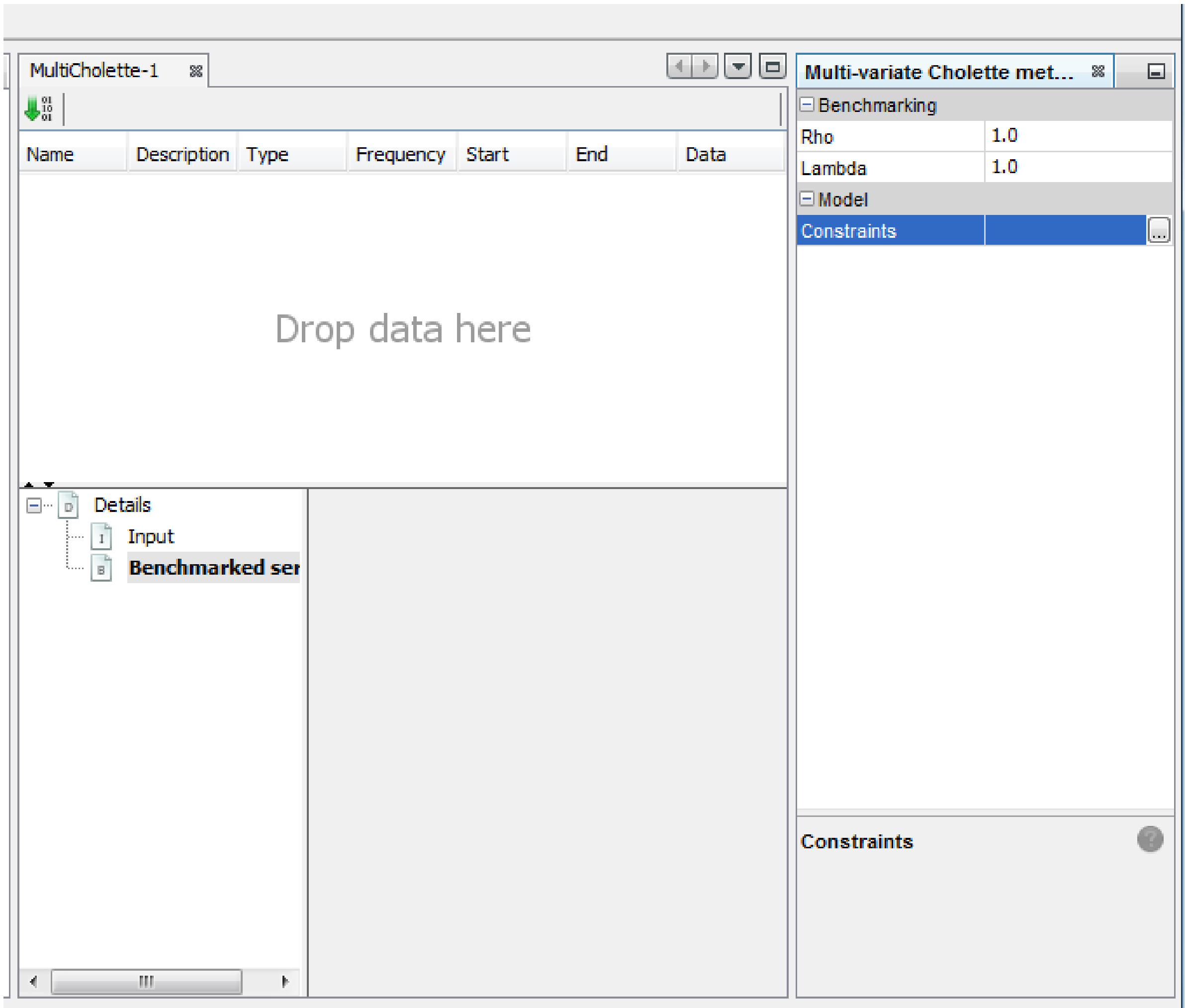


Figure 104: Multi-variate Cholette specification

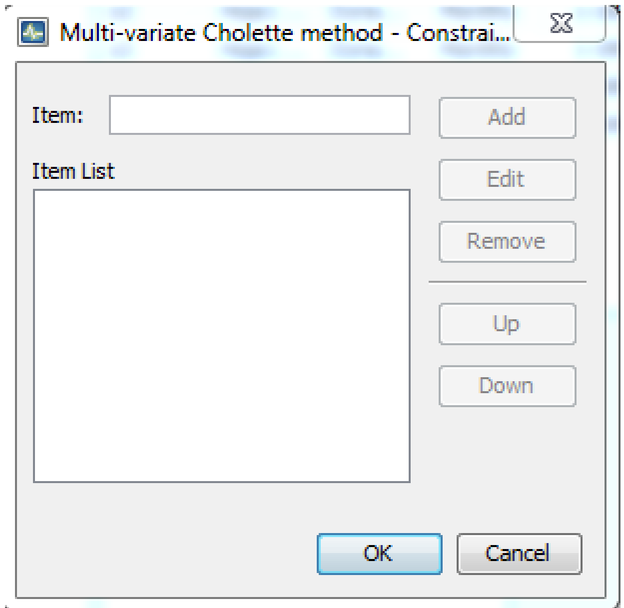


Figure 105: Constraints list

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:

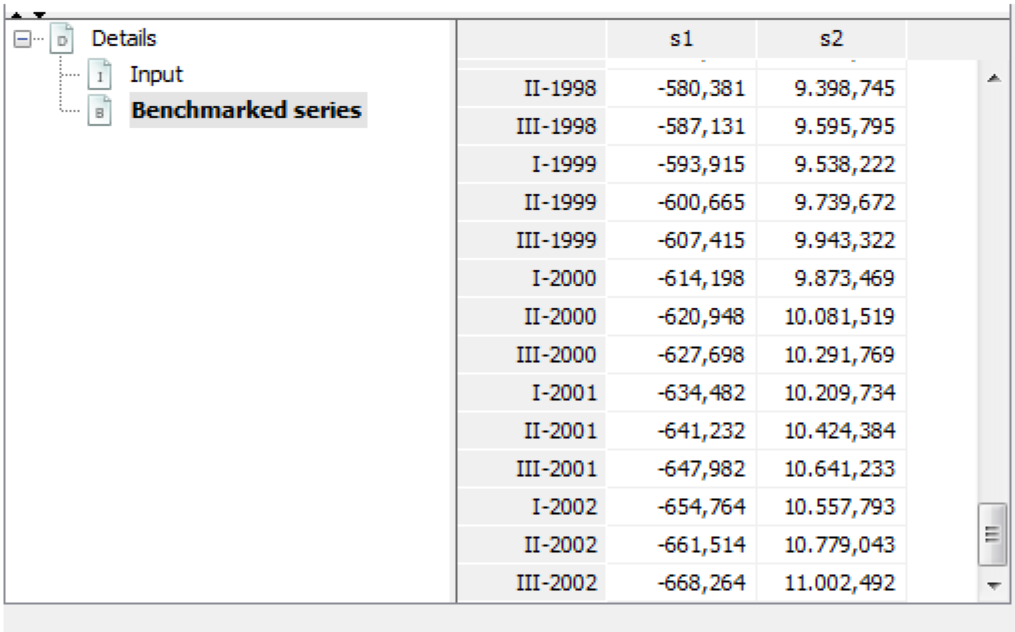
- $y = a_1 * x_1 + \cdots + a_n * x_n$ where $y, x_1, \ldots, x_n$ are same frequency time series and $a_1, \ldots, a_n$ are constant.
- $c = a_1 * x_1 + \cdots + a_n * x_n$ where $x_1, \ldots, x_n$ are same frequency time series and $c, a_1, \ldots, a_n$ are constant.
- $c = x_1 + \cdots + x_n$ can 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



Details
Input
Benchmarked series

	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:

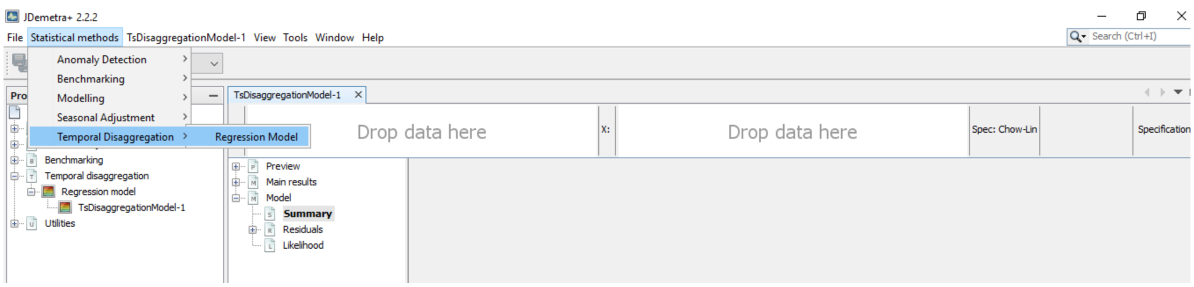


Figure 107: Temporal Disggregation

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).

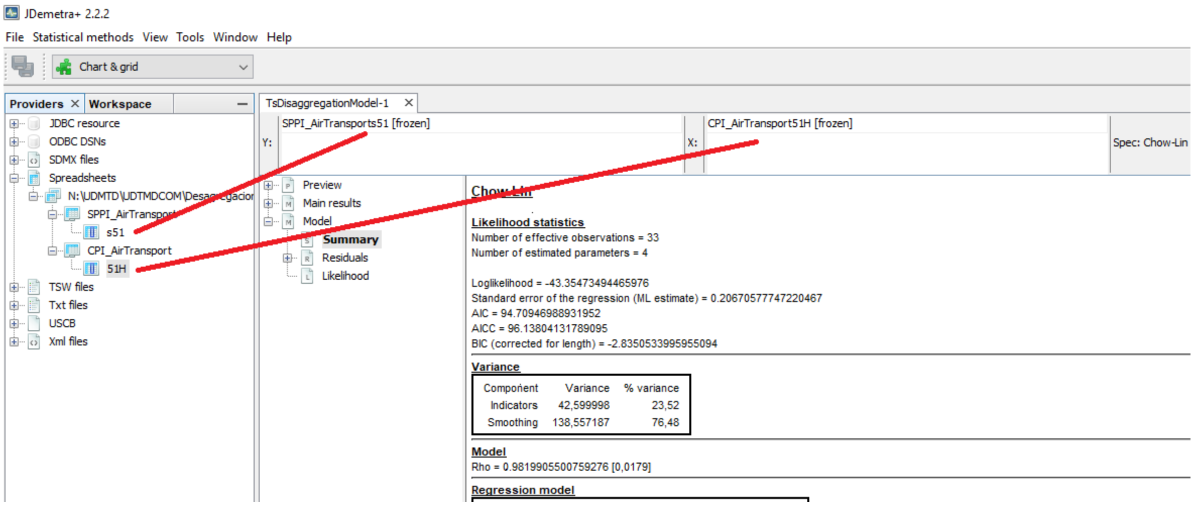


Figure 108: Temporal Disggregation

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

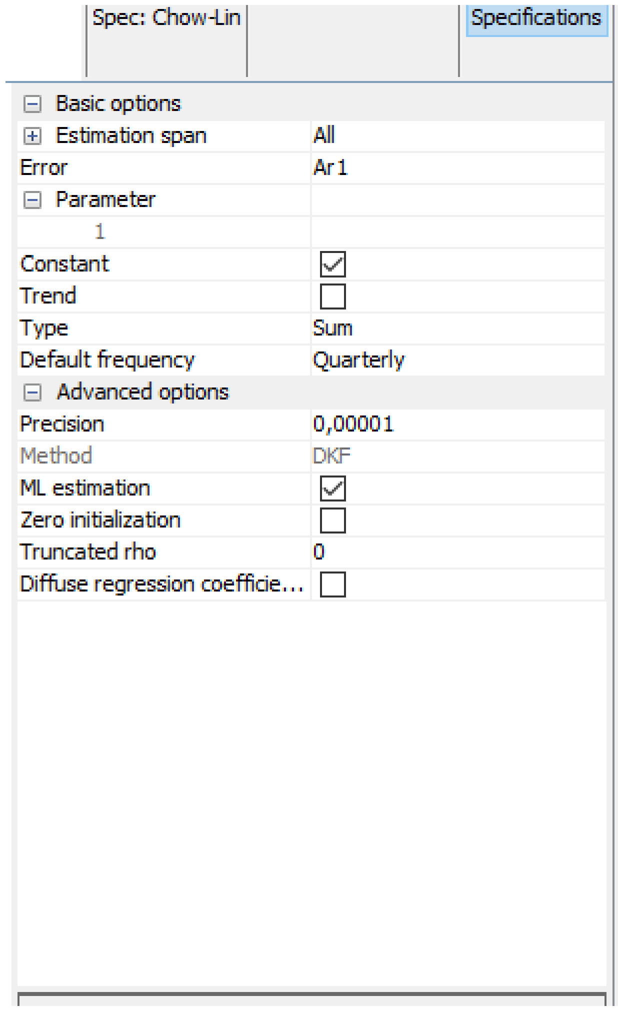


Figure 109: Temporal Disggregation

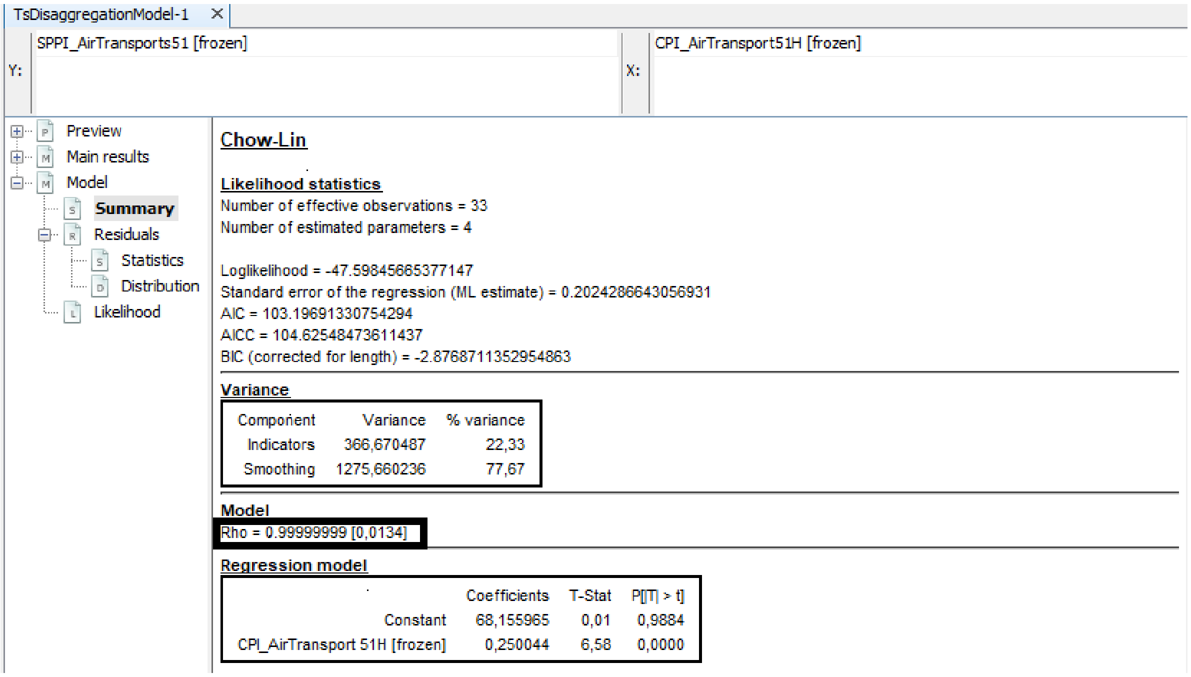


Figure 110: Temporal Disaggregation

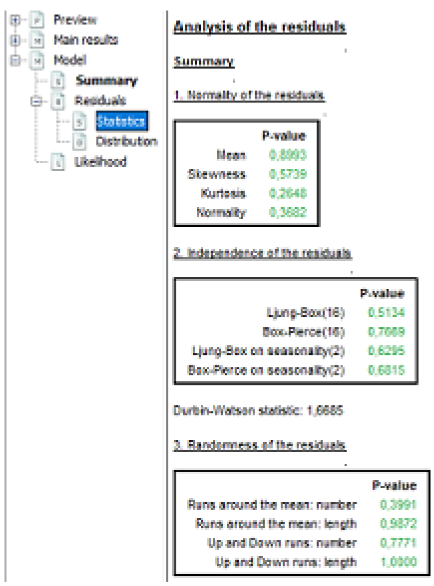


Figure 111: Temporal Disggregation

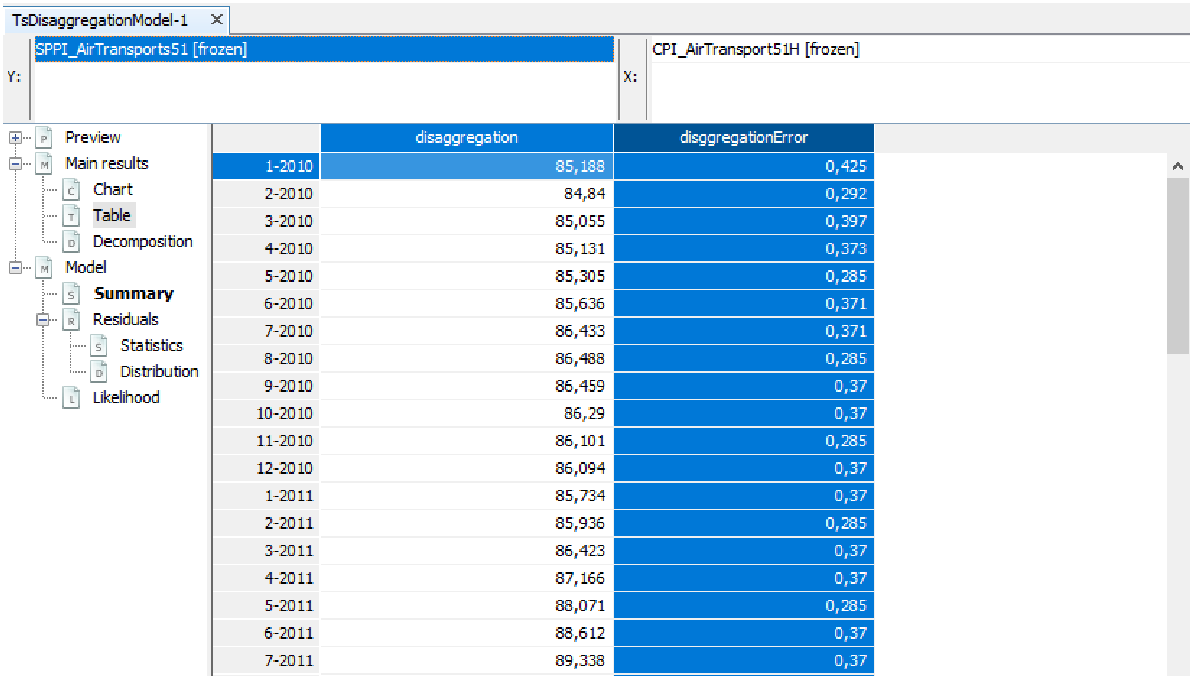


Figure 112: Temporal Disggregation

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).

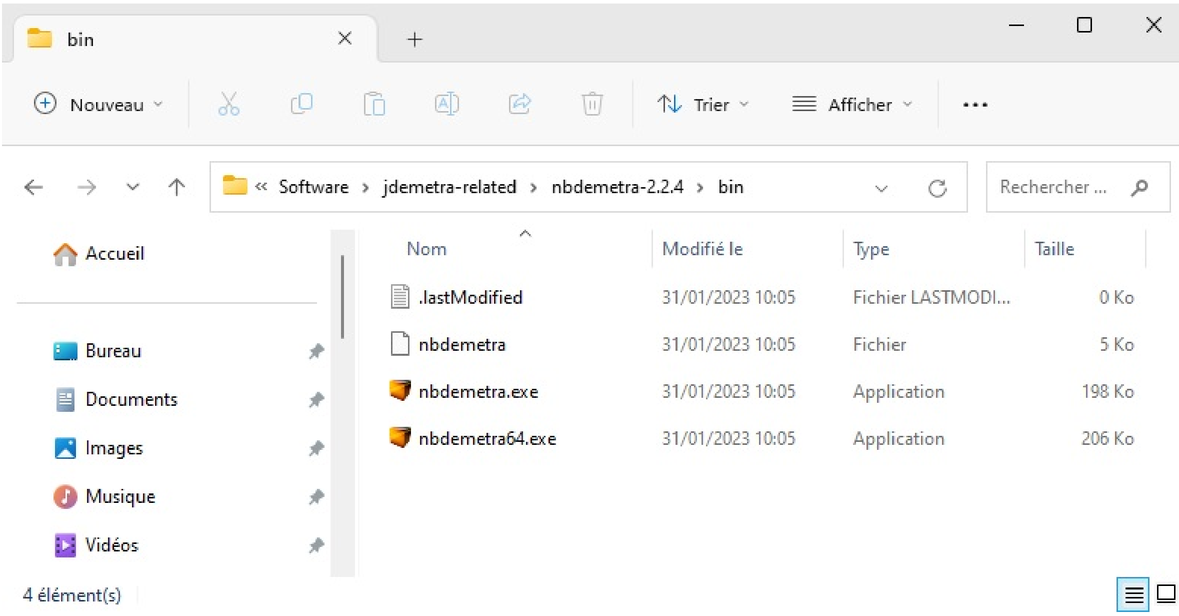


Figure 113: **Launching JDemetra+**

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

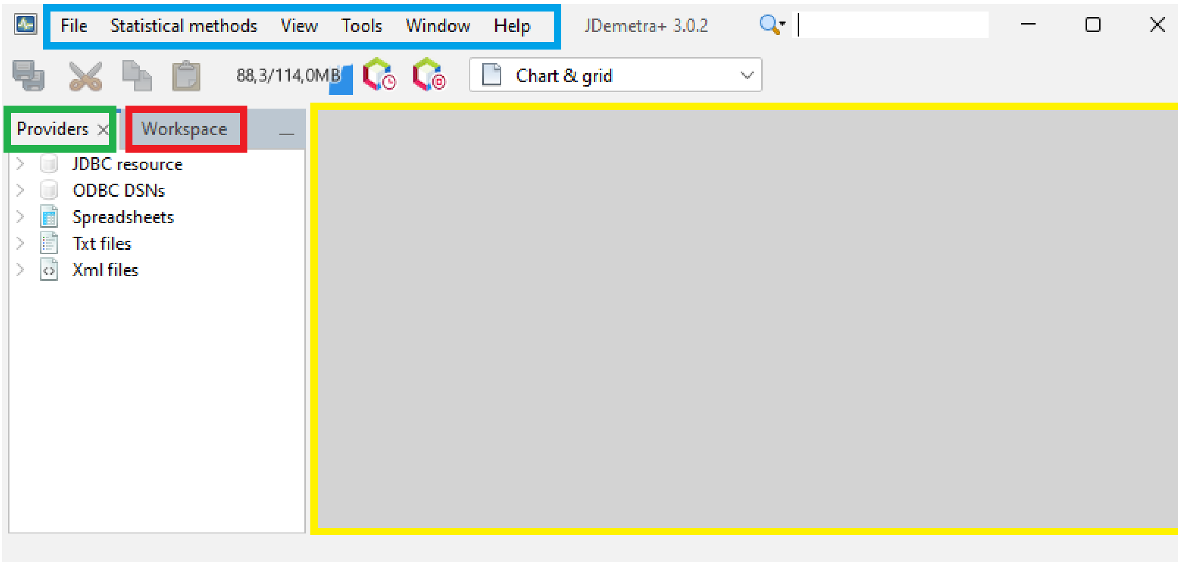


Figure 114: **JDemetra+ default window**

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;

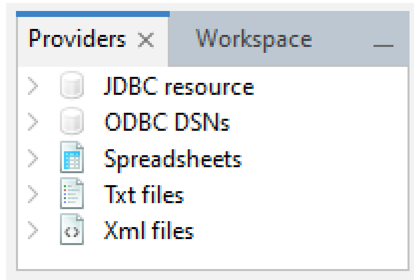


Figure 115: **The *Providers* window**

- 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.

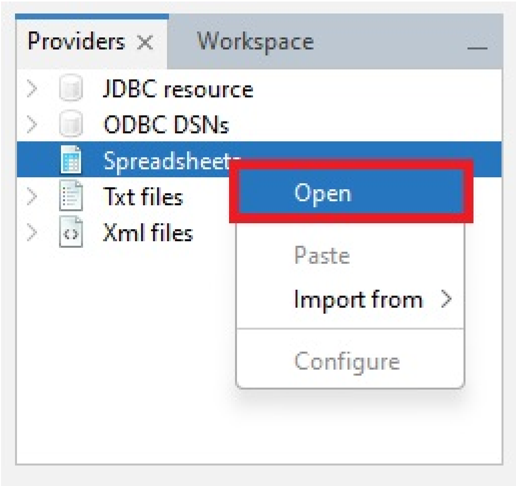
These details vary according to data providers.

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.

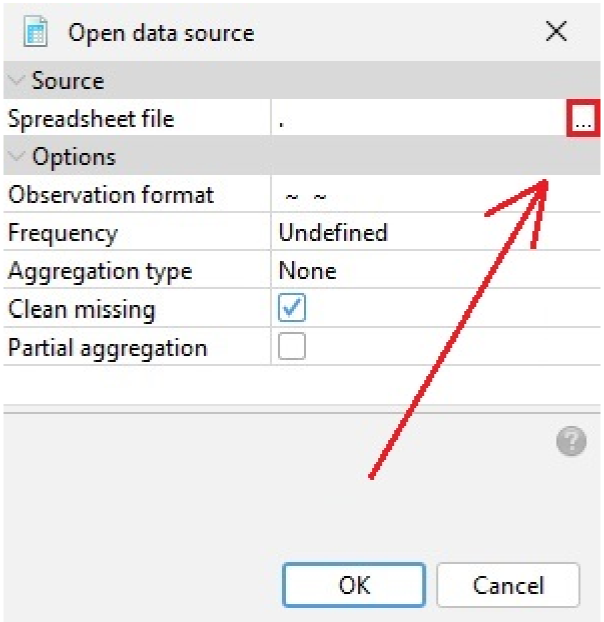
1. From the *Providers* window **right-click** on the *Spreadsheets* branch and choose *Open* option.



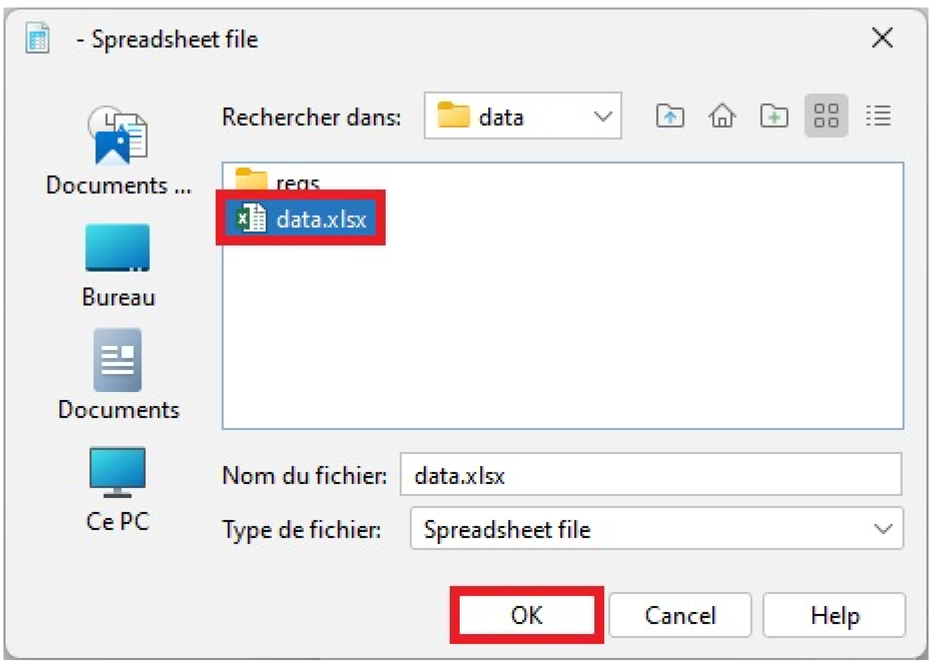
2. The *Open data source* window contains the following options:

   - **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.

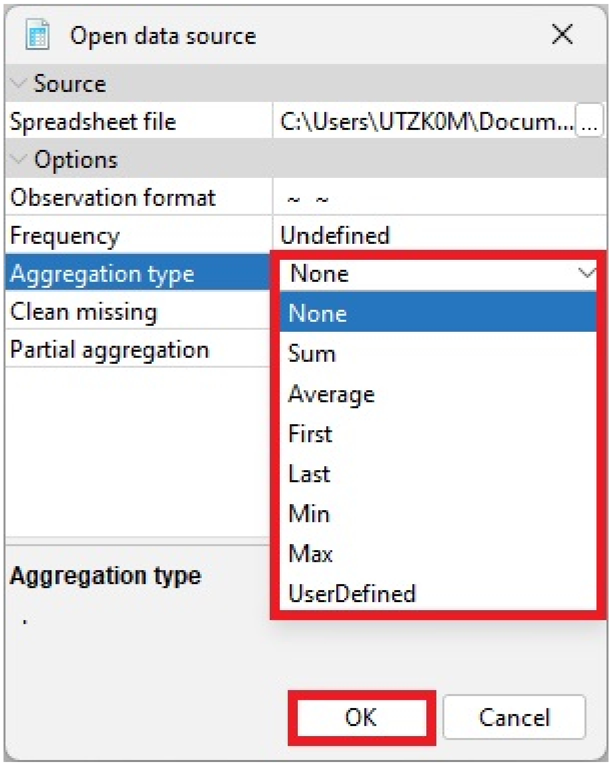
   Next, in the *Source* section click the grey "…" button to open the file.



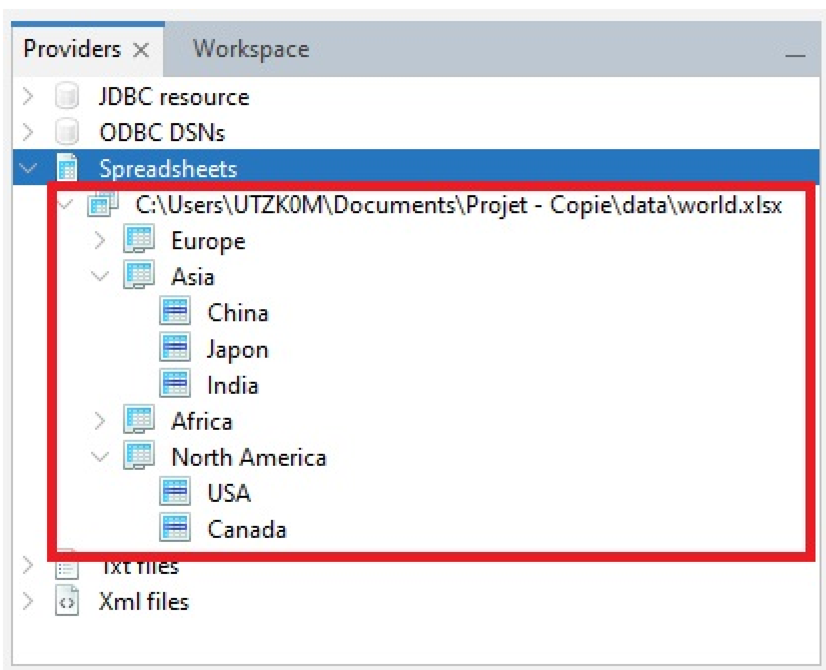
3. Choose a file and click *OK*.



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

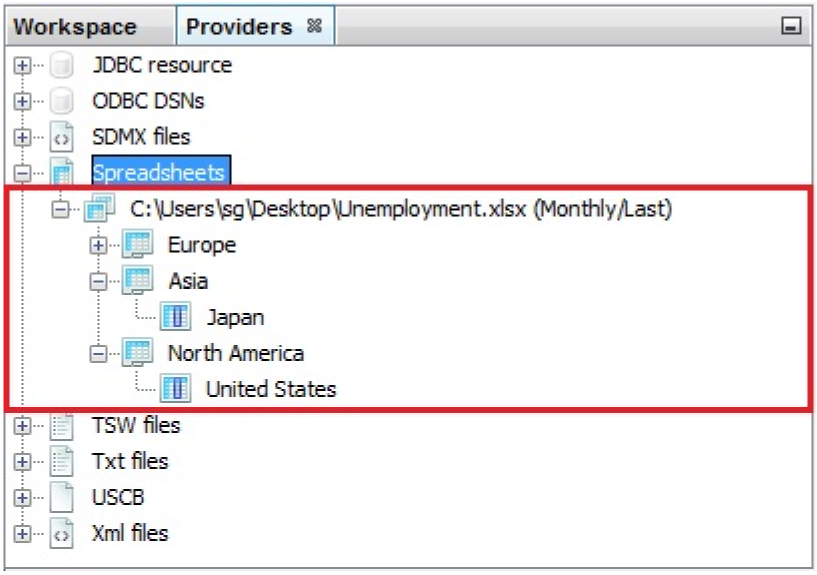


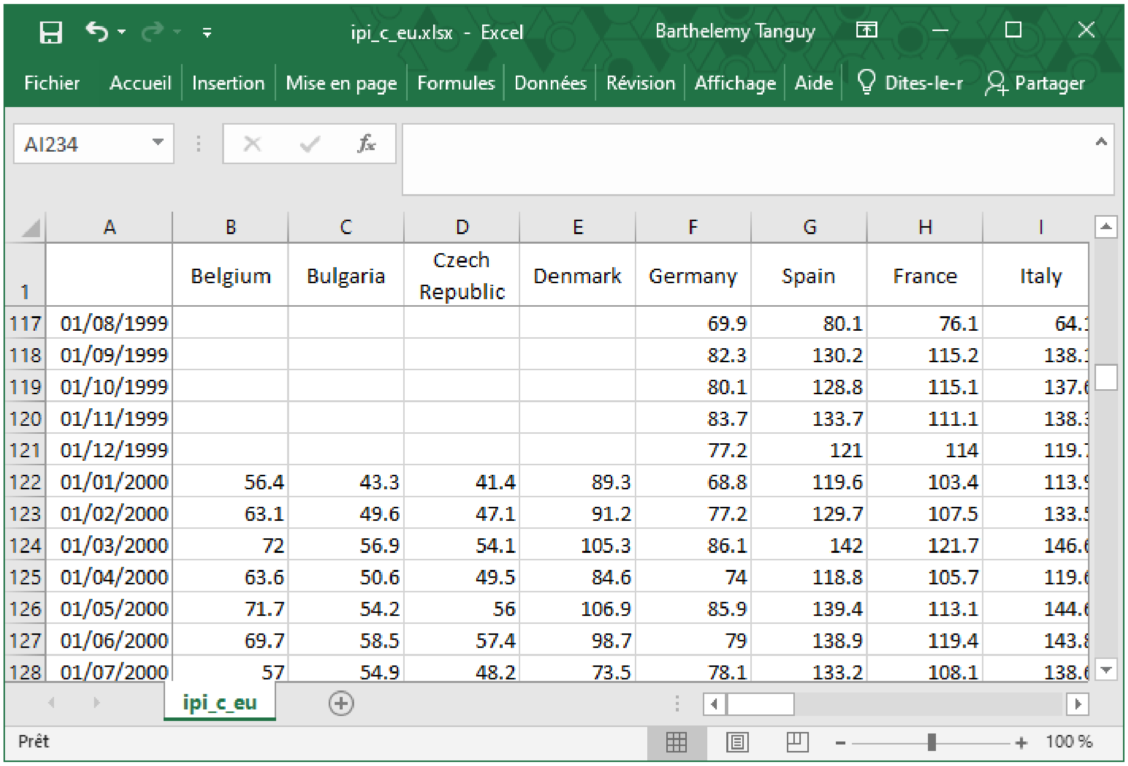
Figure 116: **A structure of a dataset**

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**.



	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

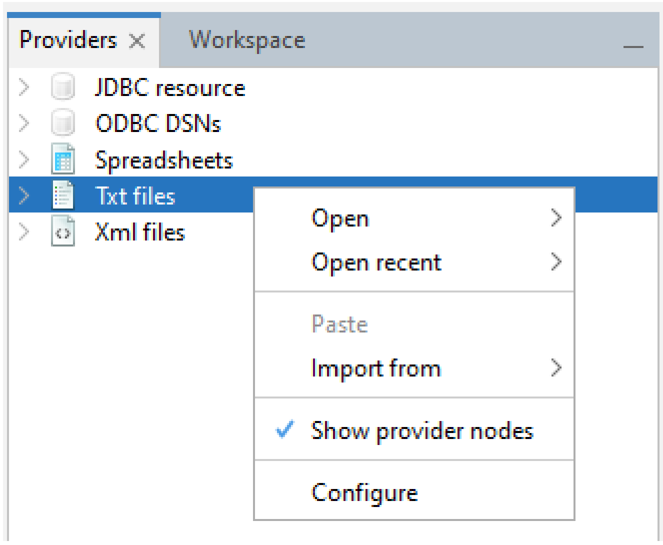
Figure 117: **Example of an Excel spreadsheet that can be imported to JDemetra+**

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.



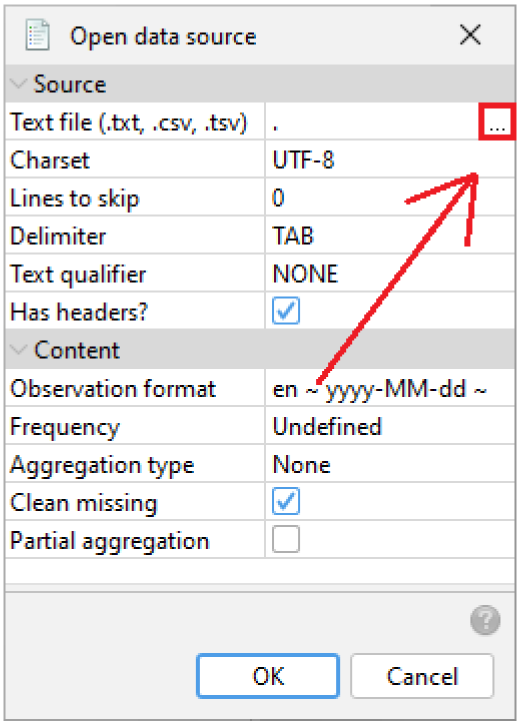
2. The *Open data source* window contains the following options:

   - `.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

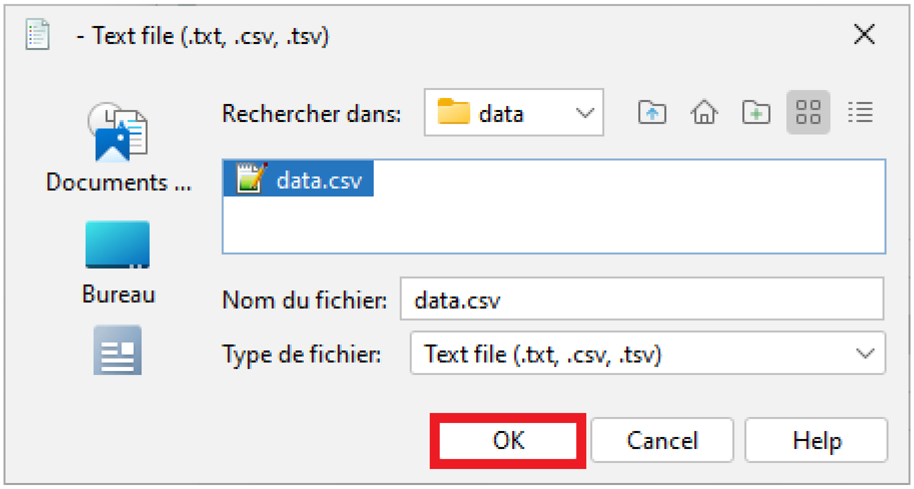
one third of the sum of the monthly values that belong to the corresponding calendar quarter.

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

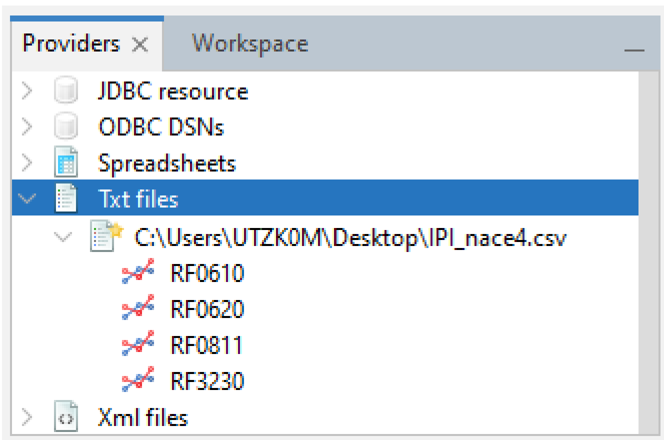
Next, in the *Source* section click the grey "…" button to open the file.



3. Choose a file and click *OK*.

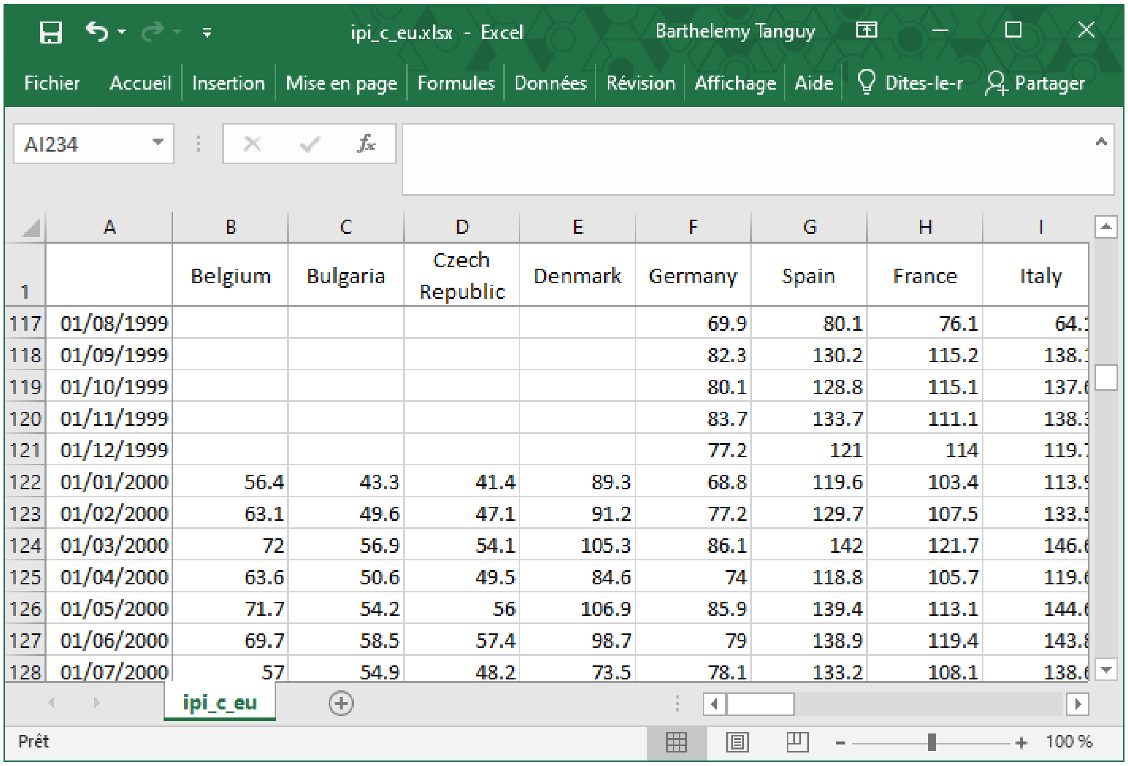


4. The data are organized in a tree structure.



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**.



	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.
118	01/09/1999					82.3	130.2	115.2	138.
119	01/10/1999					80.1	128.8	115.1	137.
120	01/11/1999					83.7	133.7	111.1	138.
121	01/12/1999					77.2	121	114	119.
122	01/01/2000	56.4	43.3	41.4	89.3	68.8	119.6	103.4	113.
123	01/02/2000	63.1	49.6	47.1	91.2	77.2	129.7	107.5	133.
124	01/03/2000	72	56.9	54.1	105.3	86.1	142	121.7	146.
125	01/04/2000	63.6	50.6	49.5	84.6	74	118.8	105.7	119.
126	01/05/2000	71.7	54.2	56	106.9	85.9	139.4	113.1	144.
127	01/06/2000	69.7	58.5	57.4	98.7	79	138.9	119.4	143.
128	01/07/2000	57	54.9	48.2	73.5	78.1	133.2	108.1	138.

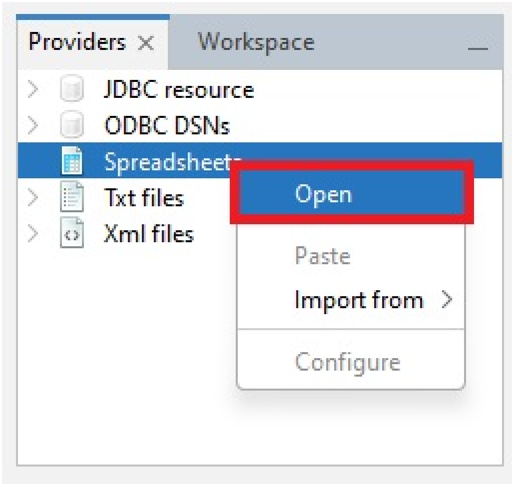
Figure 118: **Example of an Excel spreadsheet that can be imported to JDemetra+**

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.

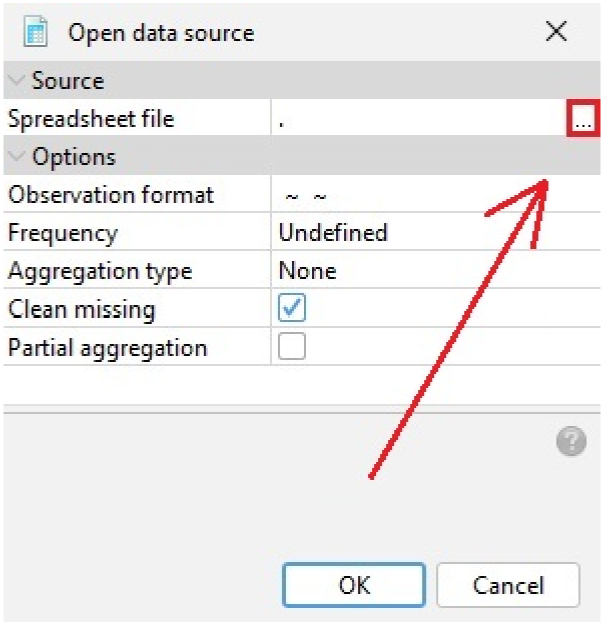
1. From the *Providers* window **right-click** on the *Spreadsheets* branch and choose *Open* option.



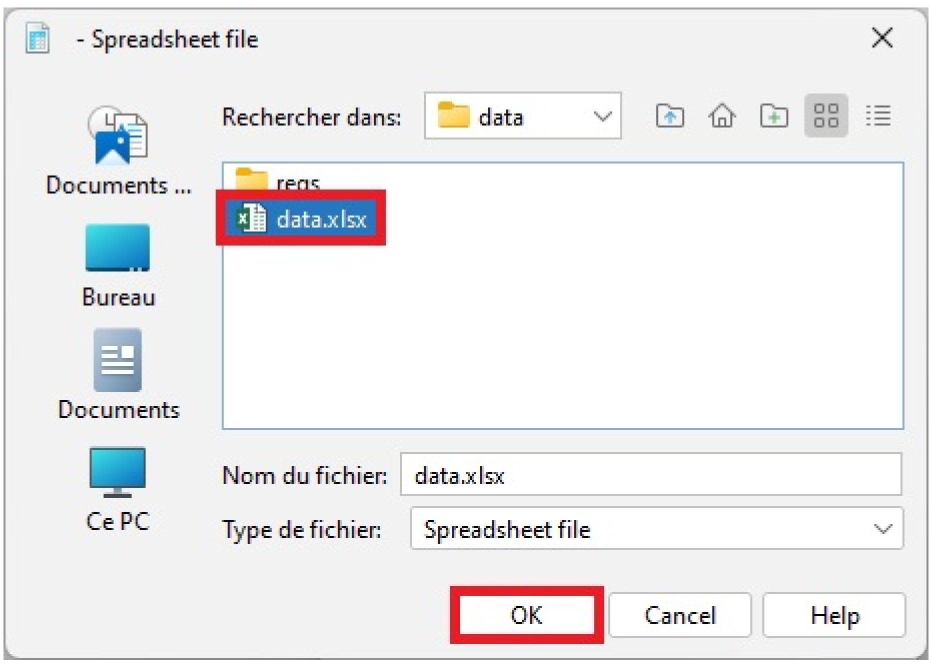
2. The *Open data source* window contains the following options:

   - **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.

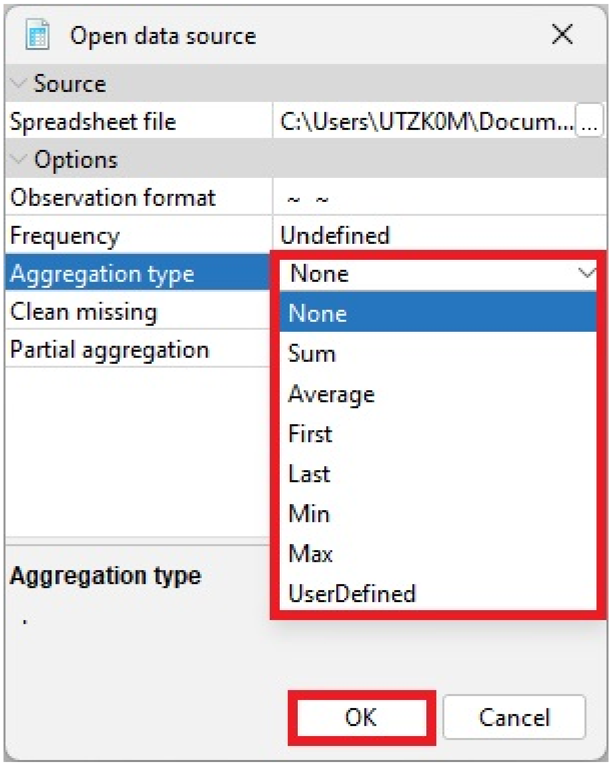
   Next, in the *Source* section click the grey “...” button to open the file.



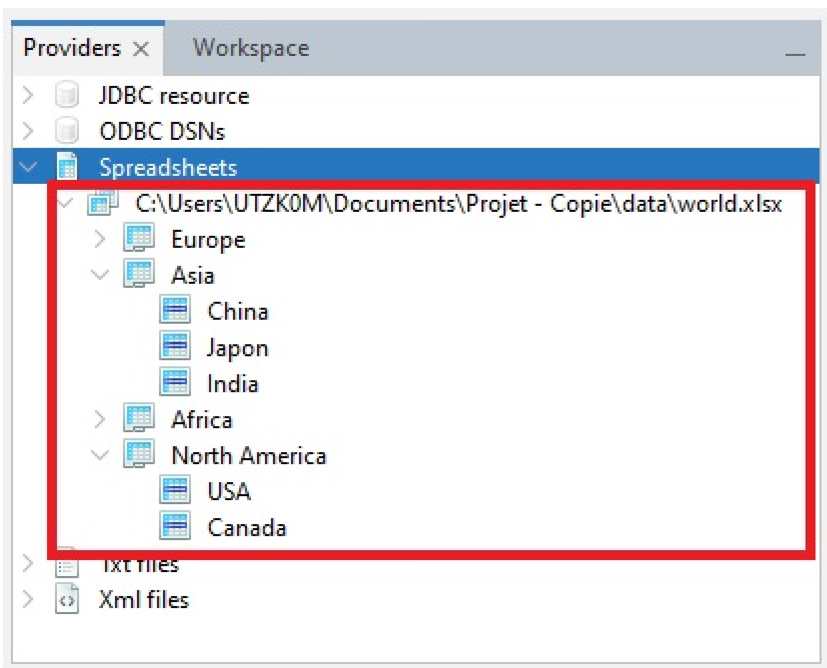
3. Choose a file and click *OK*.



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

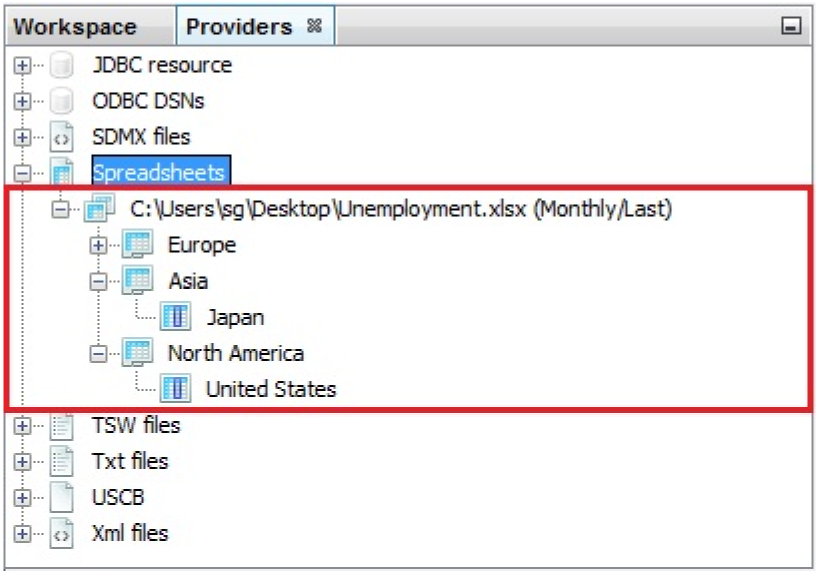


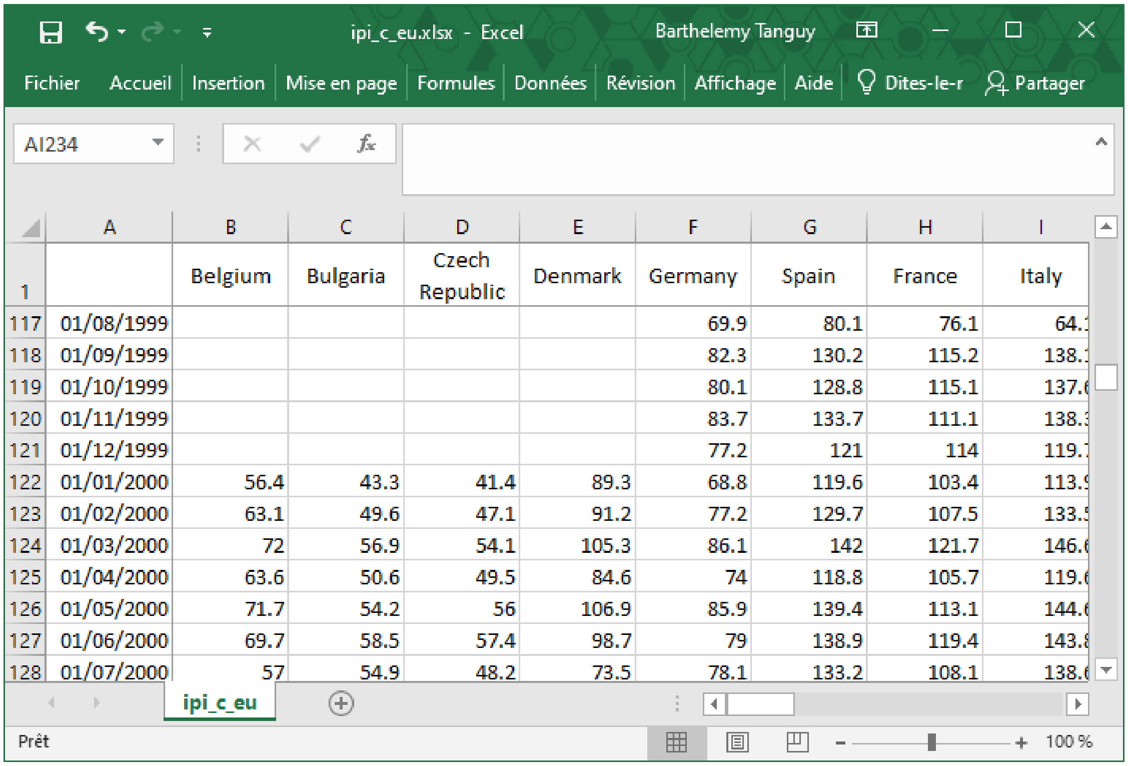
Figure 119: **A structure of a dataset**

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

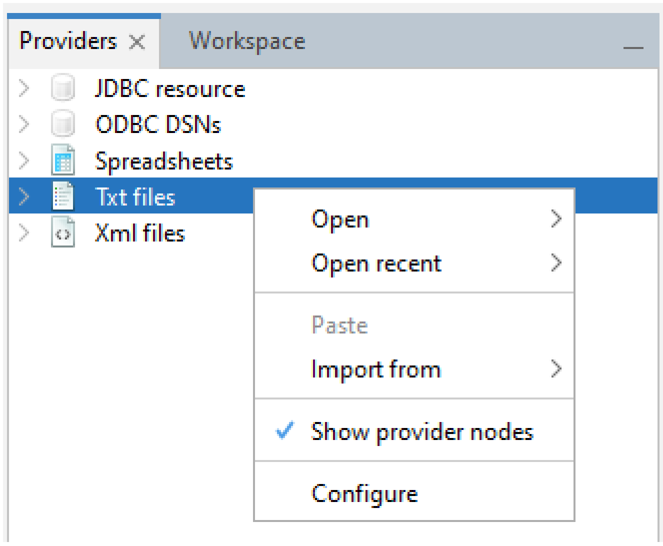
Figure 120: **Example of an Excel spreadsheet that can be imported to JDemetra+**

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.



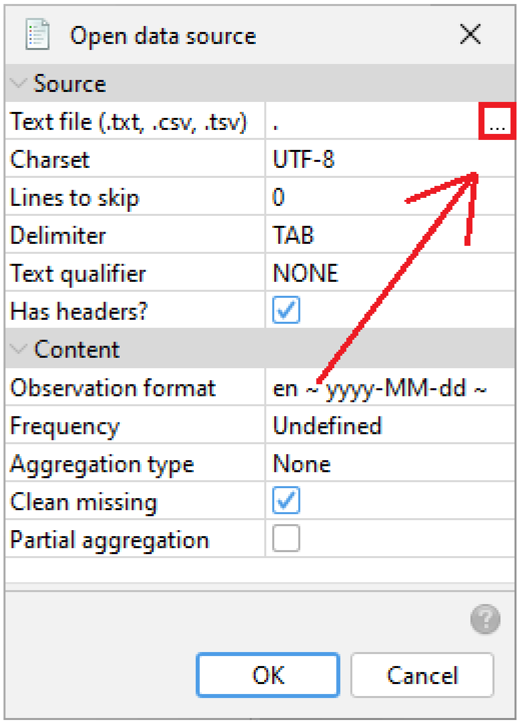
2. The *Open data source* window contains the following options:

   - **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

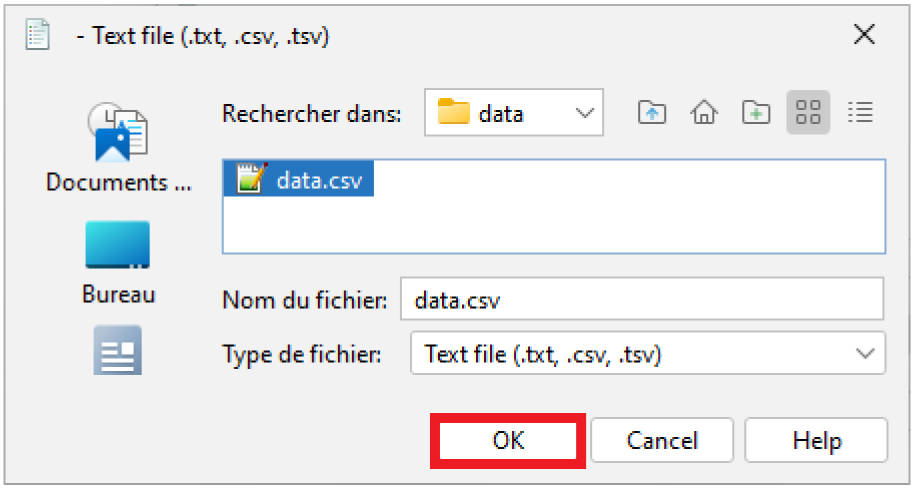
one third of the sum of the monthly values that belong to the corresponding calendar quarter.

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

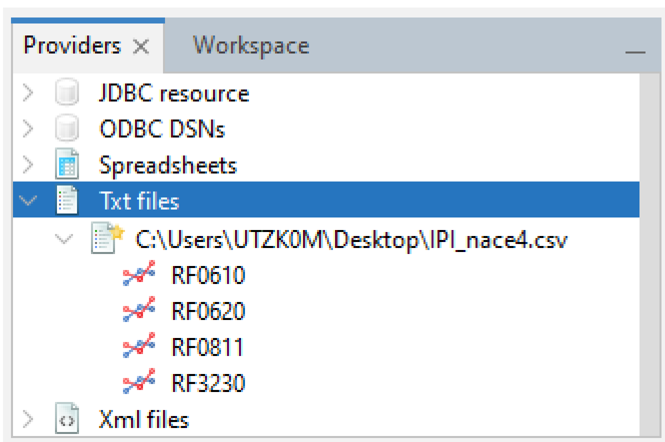
Next, in the *Source* section click the grey "…" button to open the file.



3. Choose a file and click *OK*.

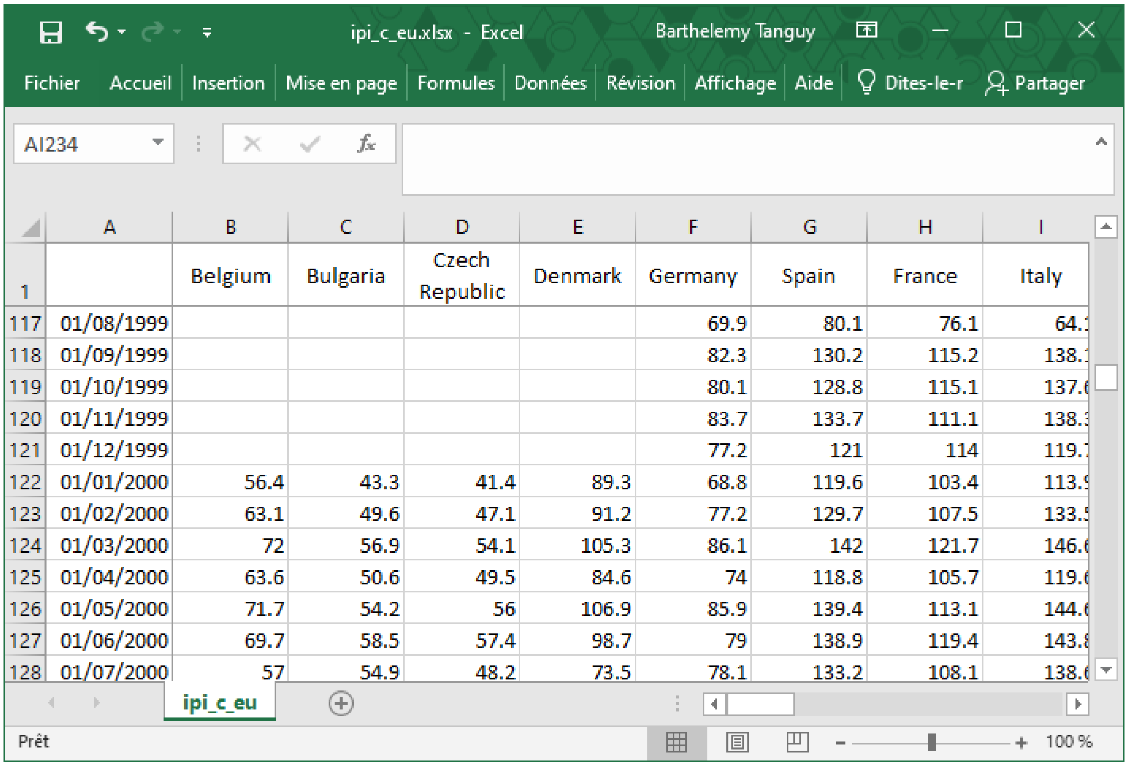


4. The data are organized in a tree structure.



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**

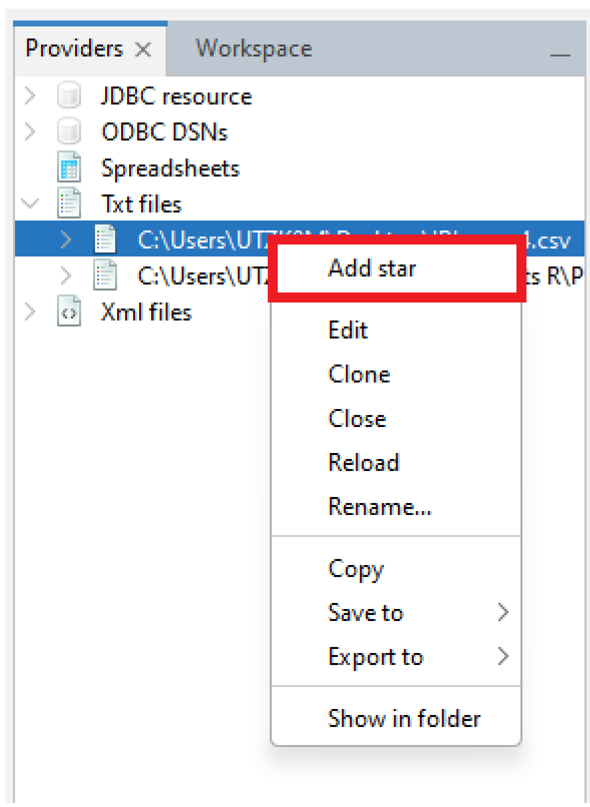


Figure 122: **Create a new favorite**

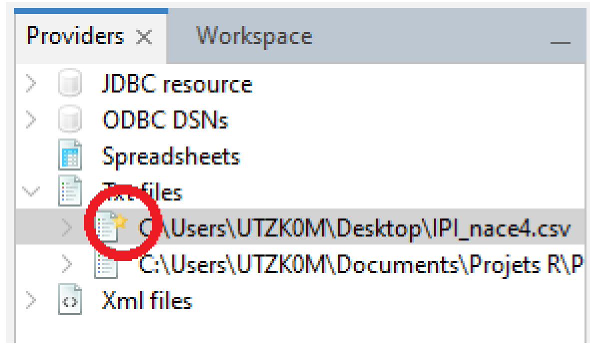


Figure 123: **Example of a favorite file**

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

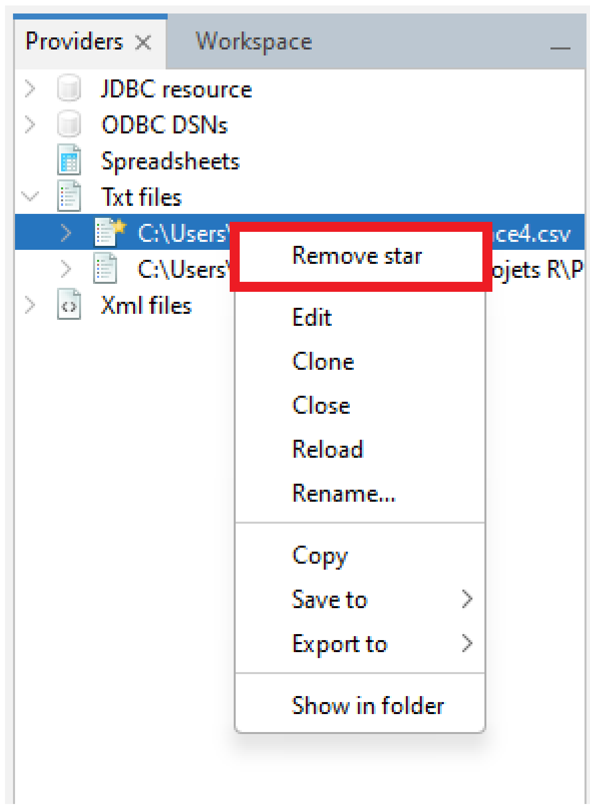


Figure 124: **Remove a favorite**

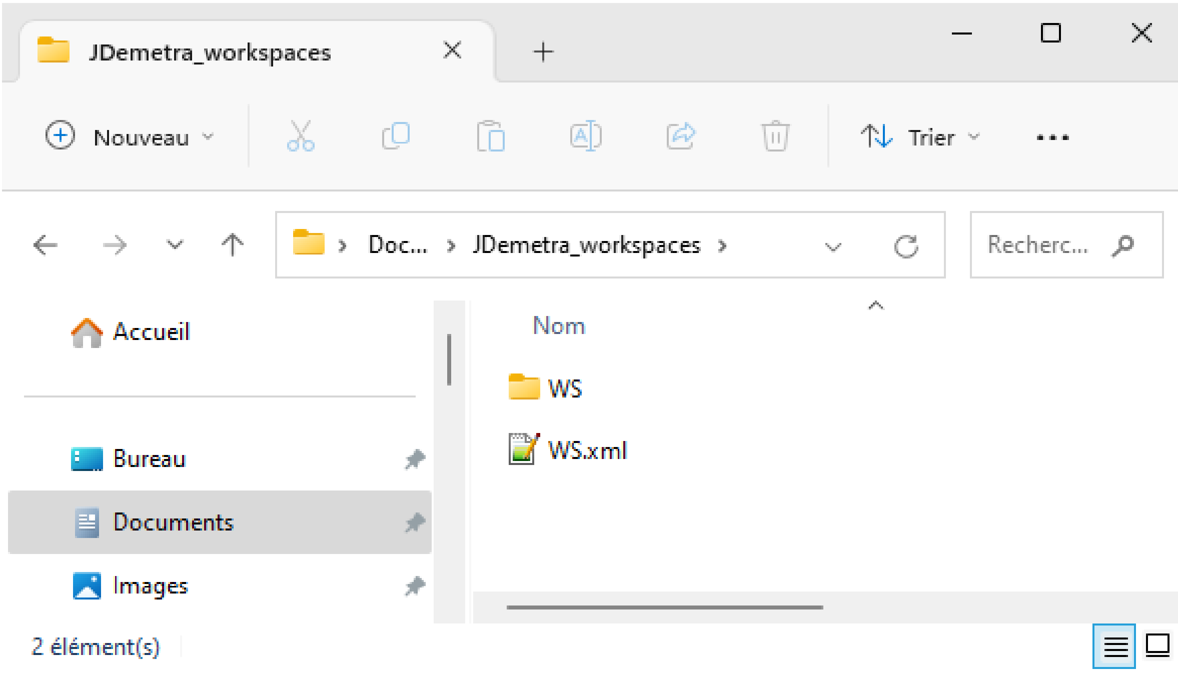


Figure 125: **Example of a workspace created by JDemetra+.**

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

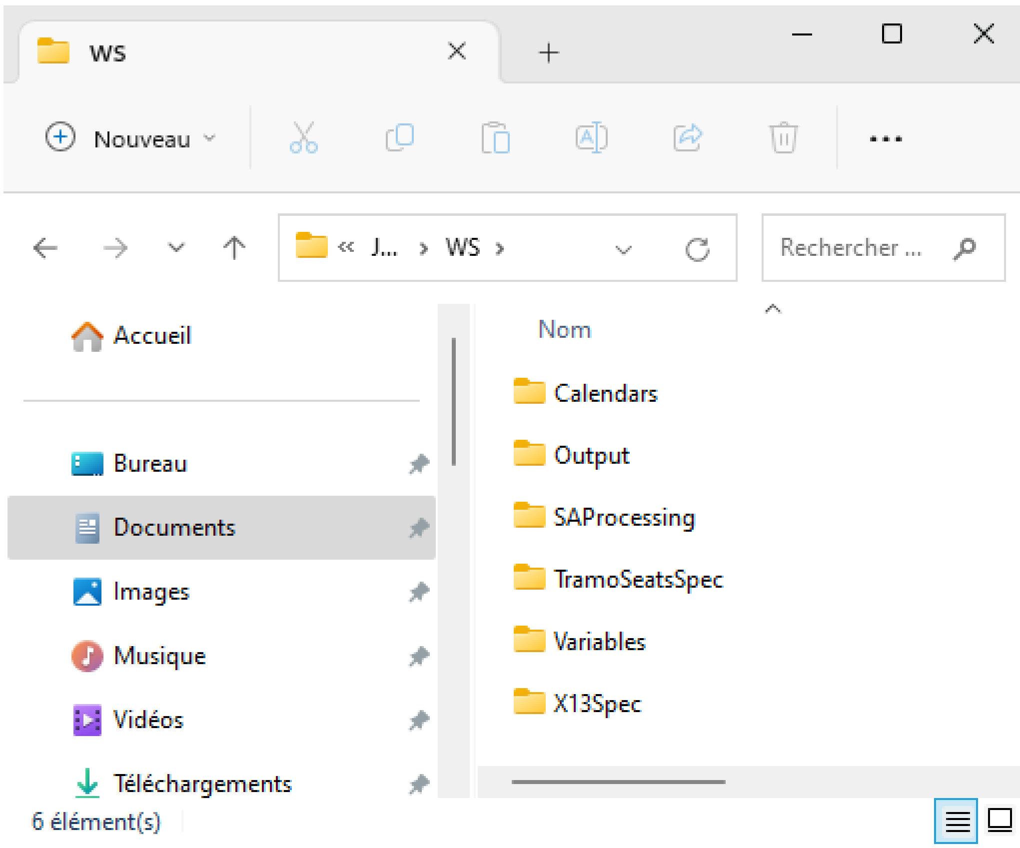


Figure 126: **Structure of a workspace**

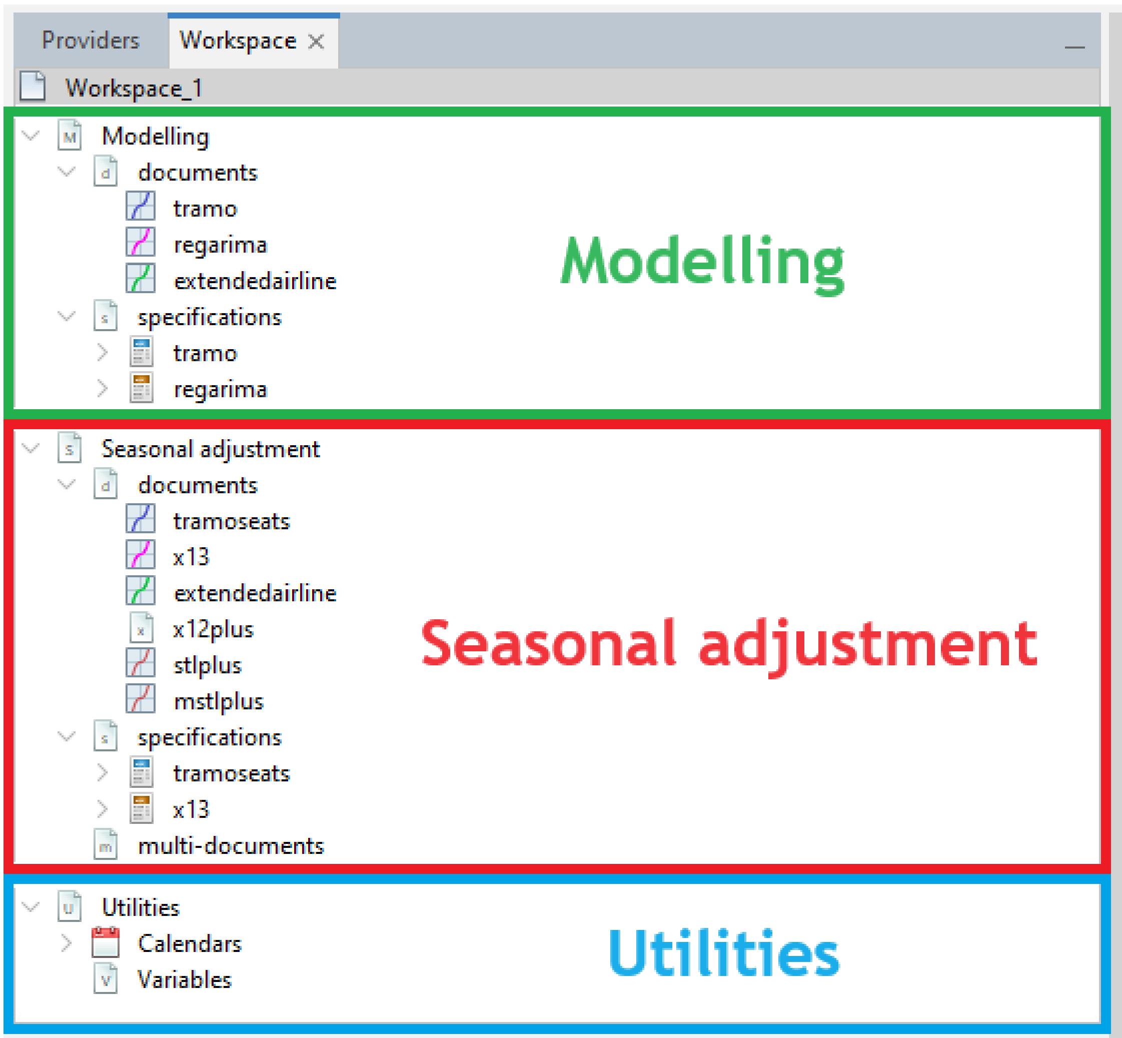


Figure 127: **The *Workspace* window**

Top Bar Menu and options

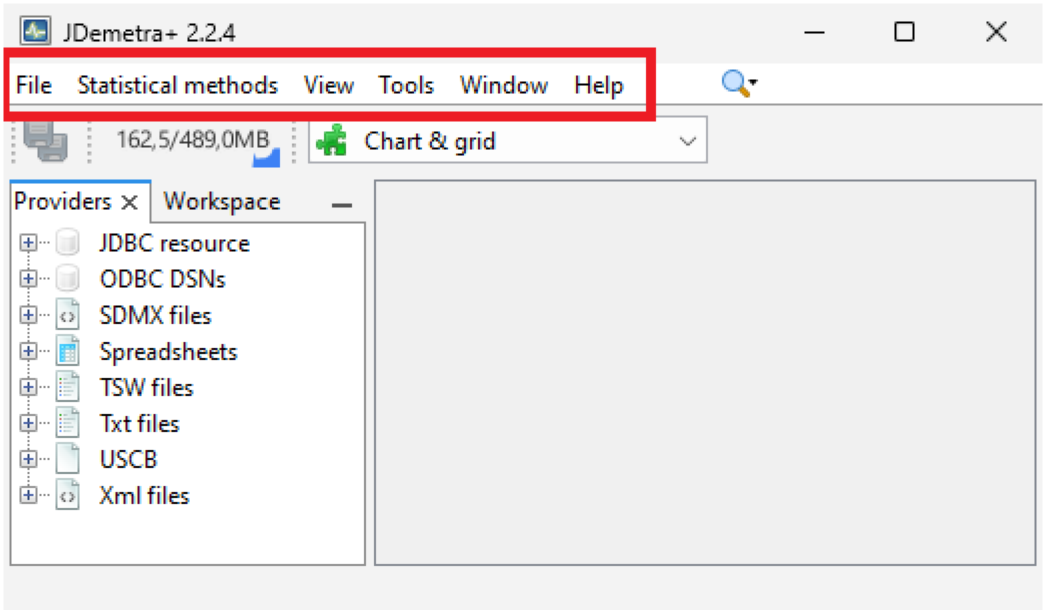


Figure 128: **The Top bar menus**

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.

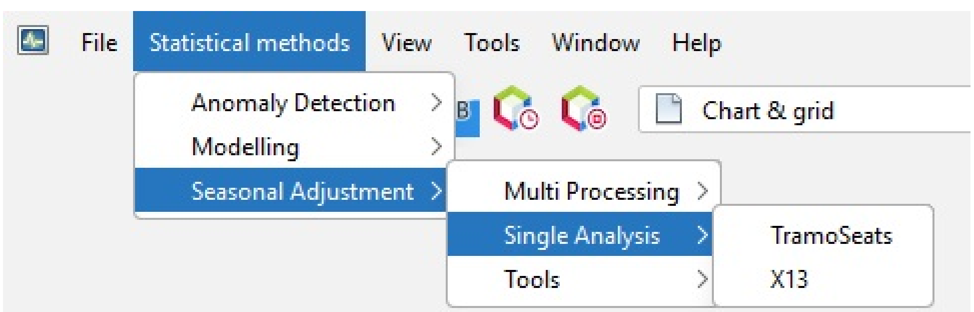


Figure 129: **The main menu with selected drop-down menu**

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:

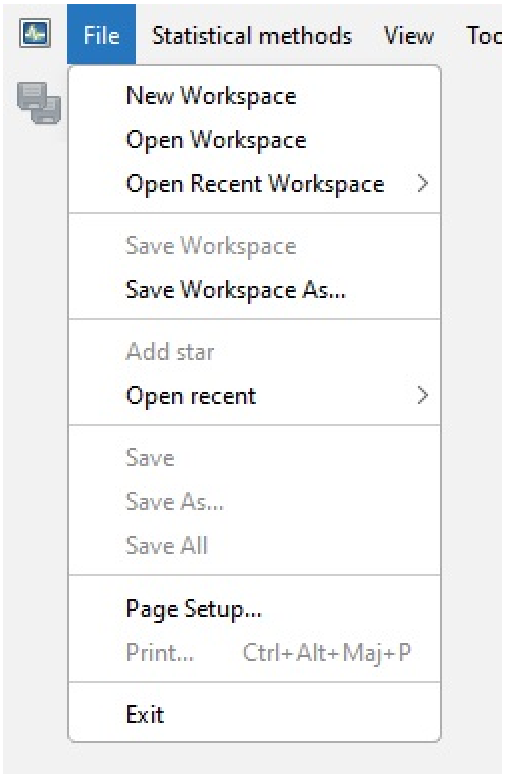


Figure 130: **The content of the *File* menu**

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

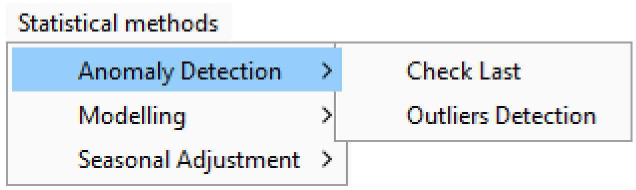


Figure 131: **The *Anomaly detection* tab.**

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

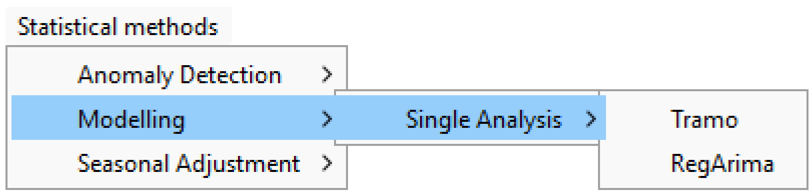


Figure 132: **The *Modelling* tab.**

- 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*.

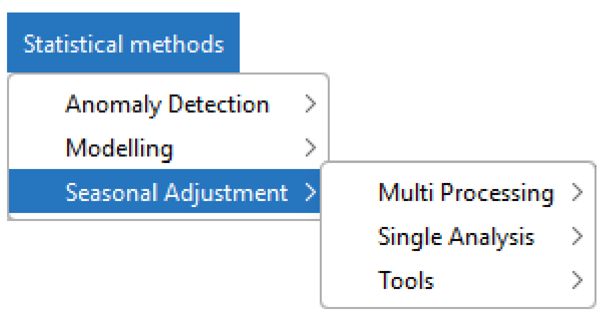


Figure 133: **The *Seasonal adjustment* tab in v2**

0.0.0.1 v2

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

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

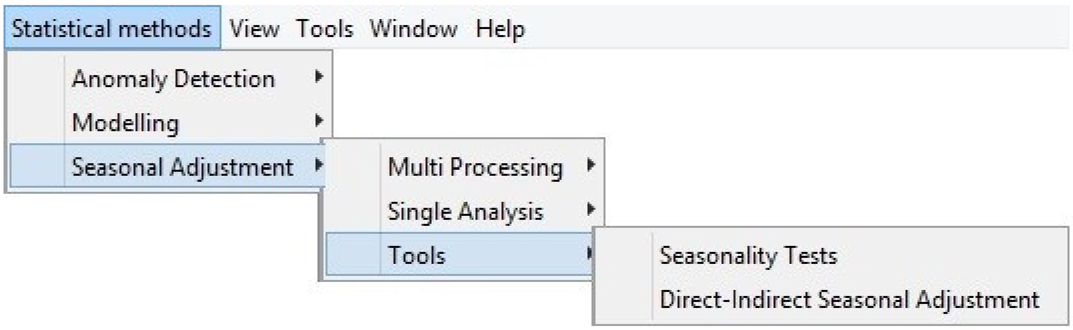


Figure 134: **The *Seasonal adjustment* tab in v2**

0.0.0.2 v3

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

- *Seasonality Tests*
- *CanovaHansen*

But no direct-indirect analysis.

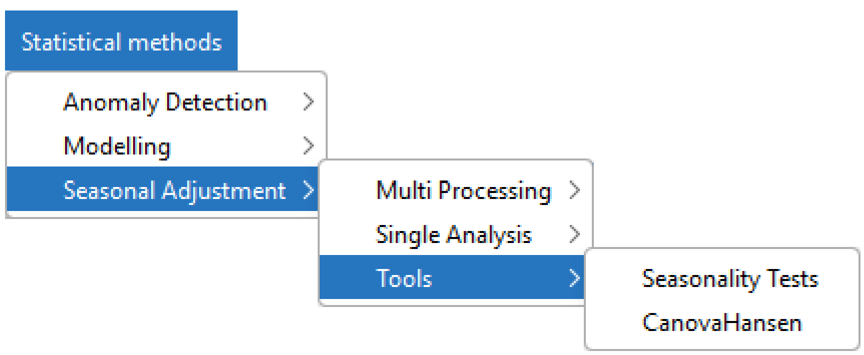


Figure 135: **The *Seasonal adjustment* tab in v3**

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

- *Seasonality Tests*

- *Direct-Indirect Seasonal Adjustment*

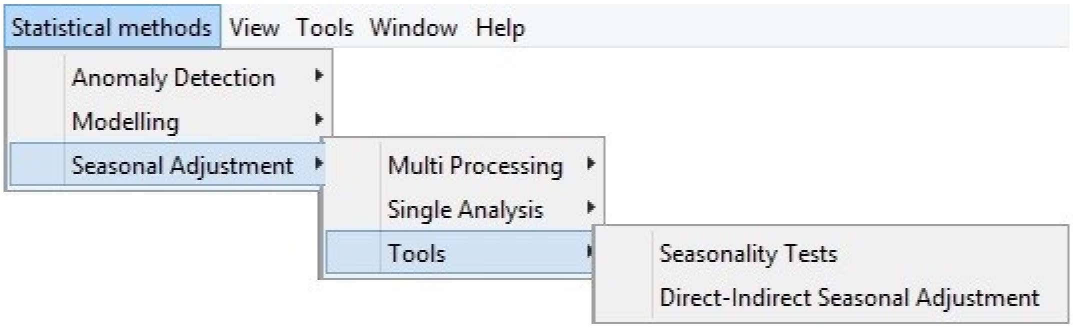


Figure 136: **The *Seasonal adjustment* tab in v2**

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

- *Seasonality Tests*
- *CanovaHansen*

But no direct-indirect analysis.

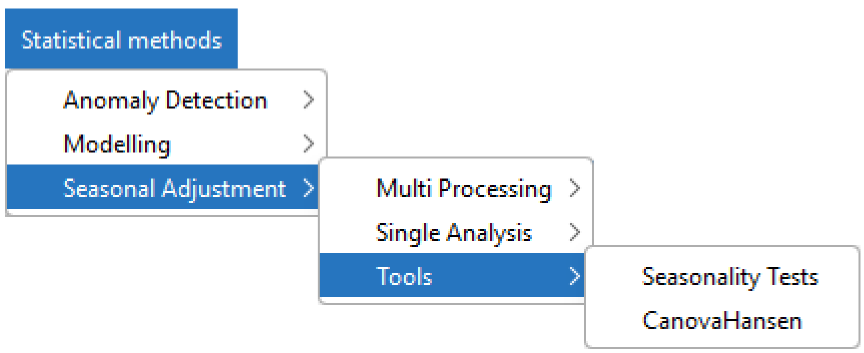


Figure 137: **The *Seasonal adjustment* tab in v3**

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;

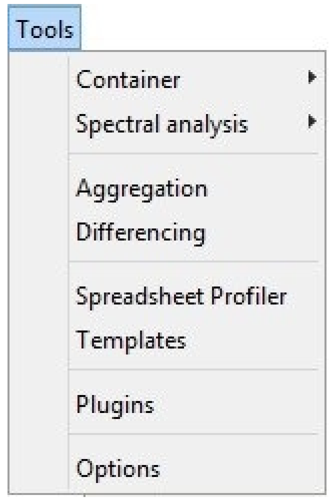


Figure 138: **The *Tools* menu**

- *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.

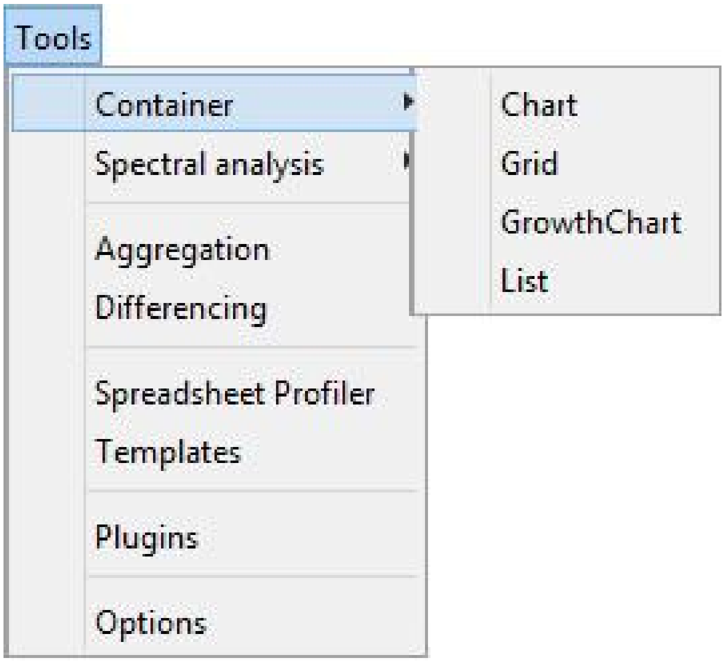


Figure 139: ***The** **Container** **menu***

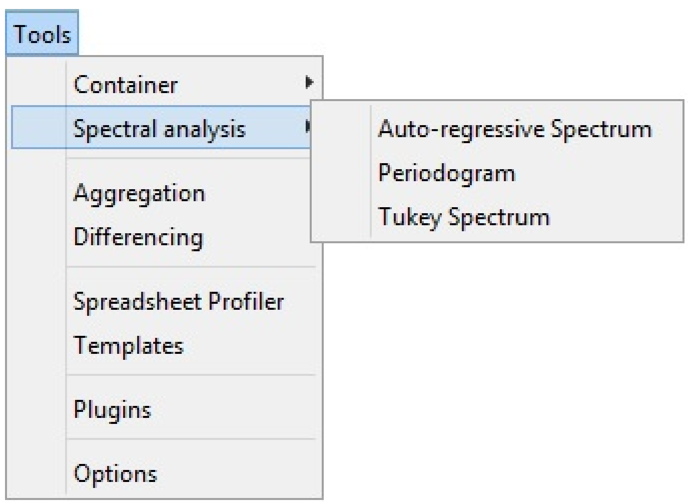


Figure 140: **Tools for spectral analysis**

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.

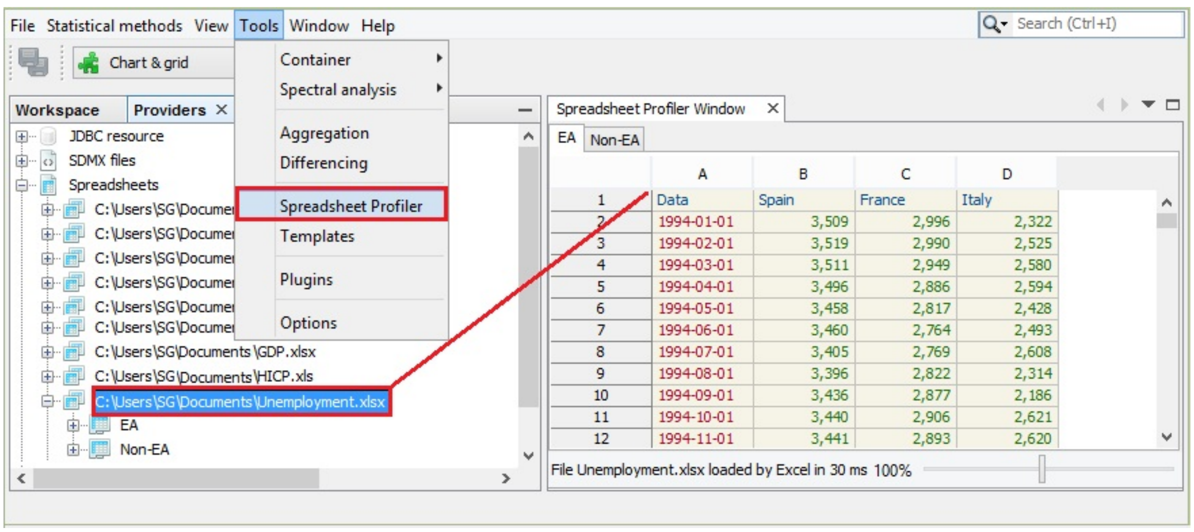


Figure 141: **The *Spreadsheet Profiler* window**

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.

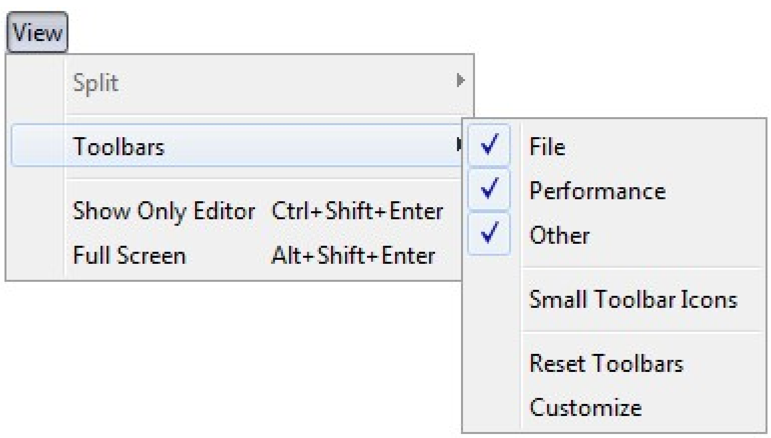


Figure 142: **The *View* menu**

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.

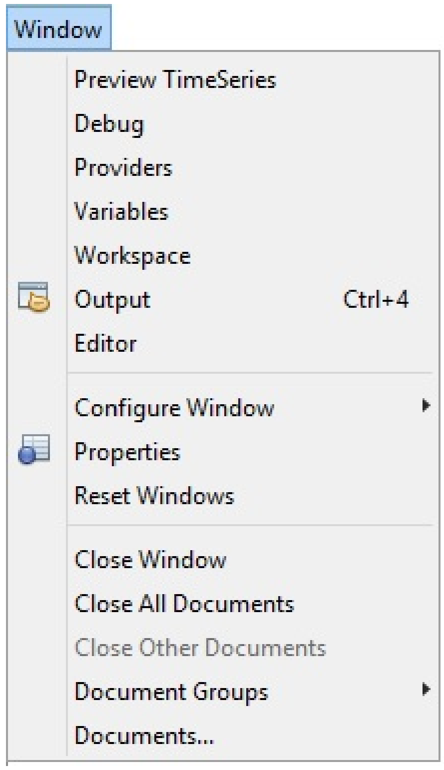


Figure 143: **The *Window* menu**

- **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.

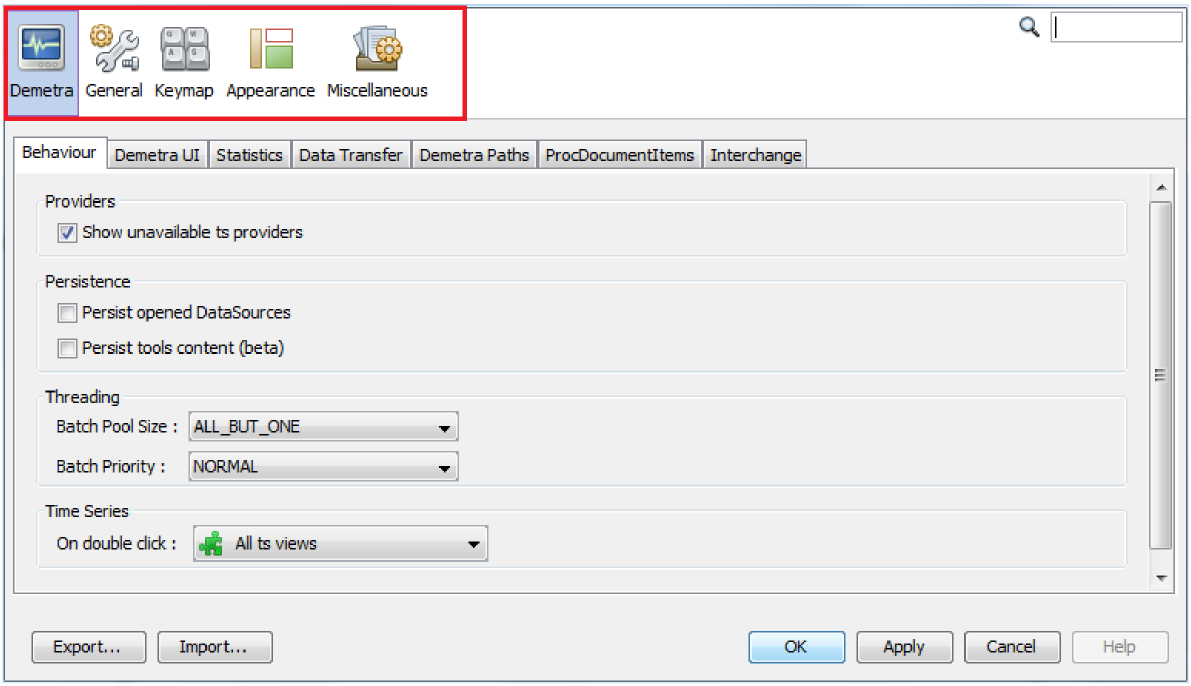


Figure 144: **Main sections of the *Options* window**

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*,

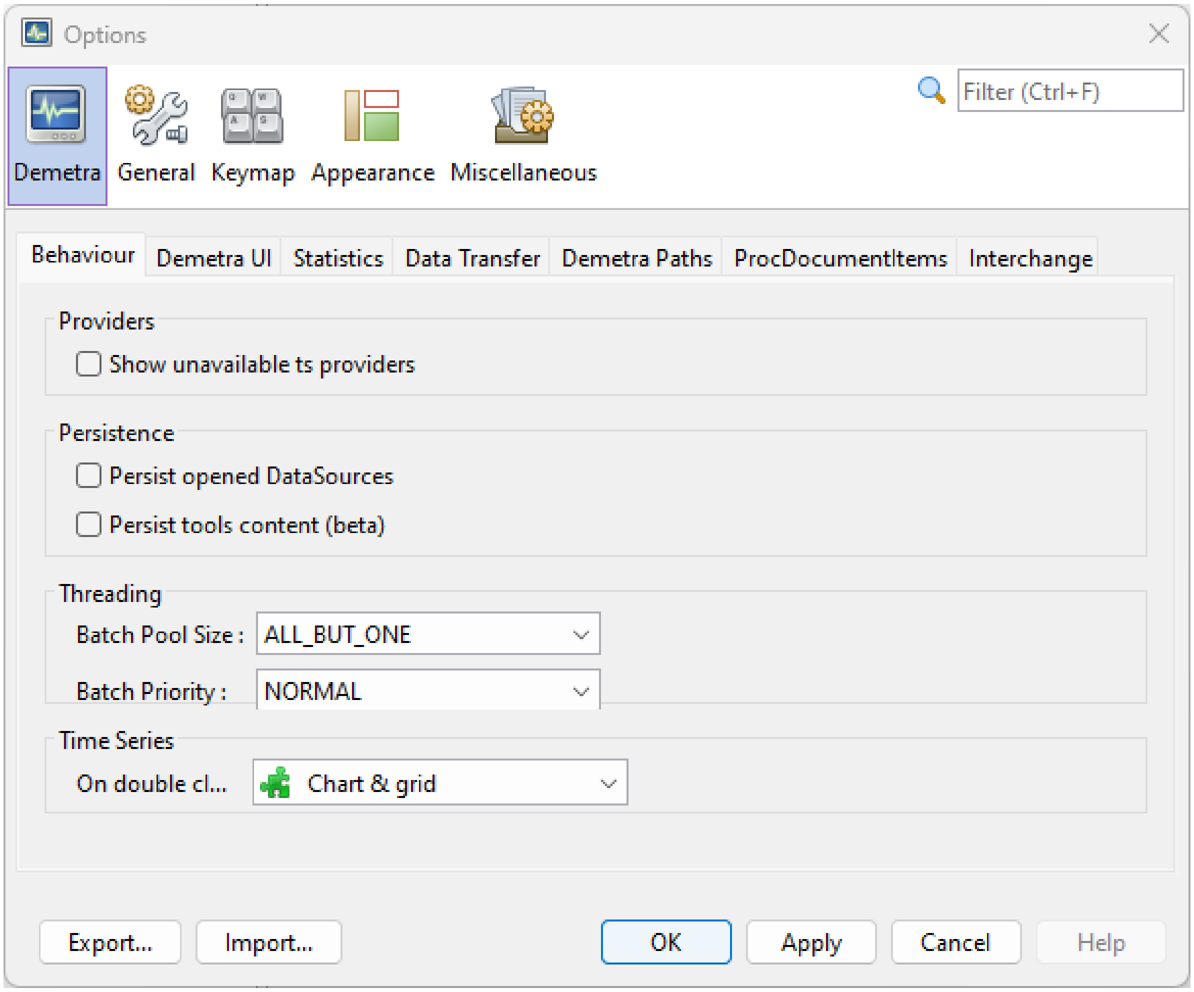


Figure 145: **Demetra panel in v2**

- and *Demetra Paths*.

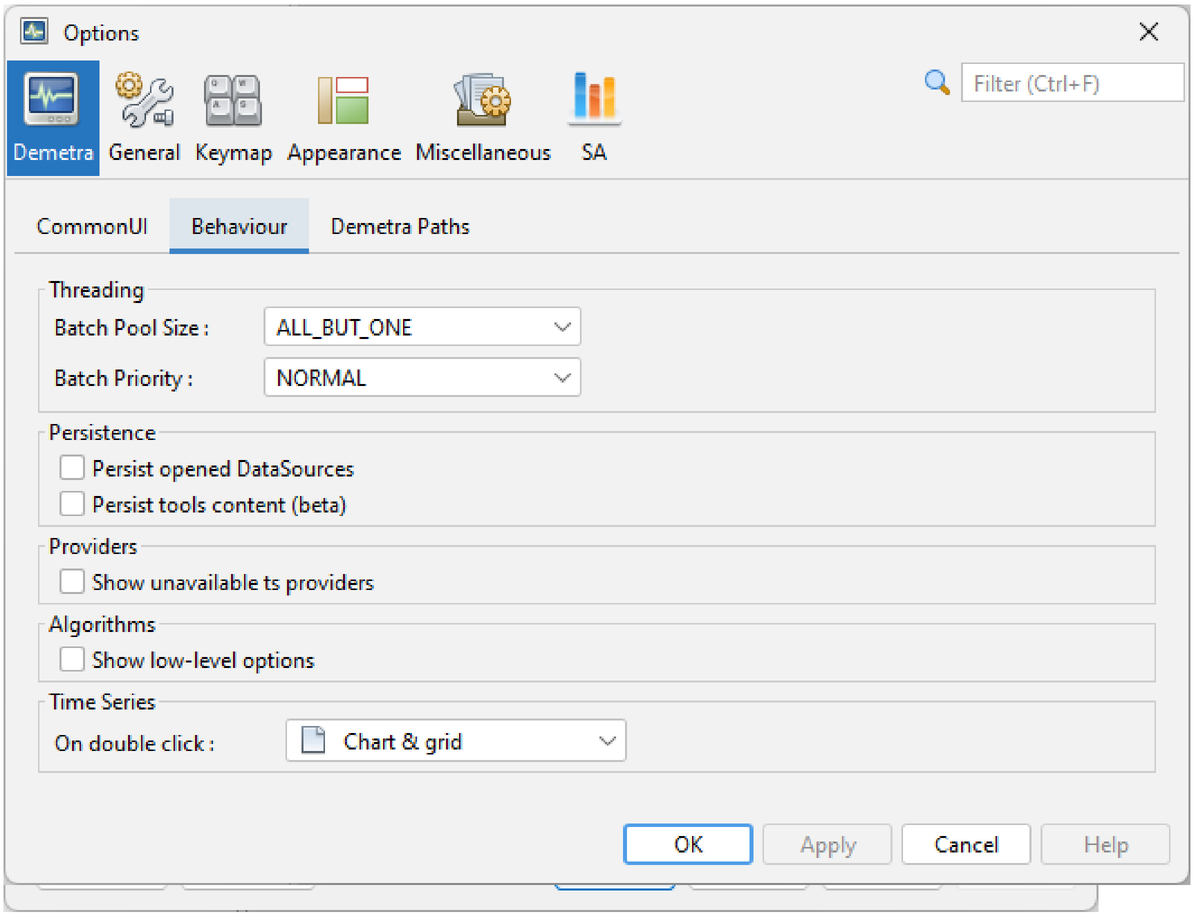


Figure 146: **Demetra panel in v3**

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*,

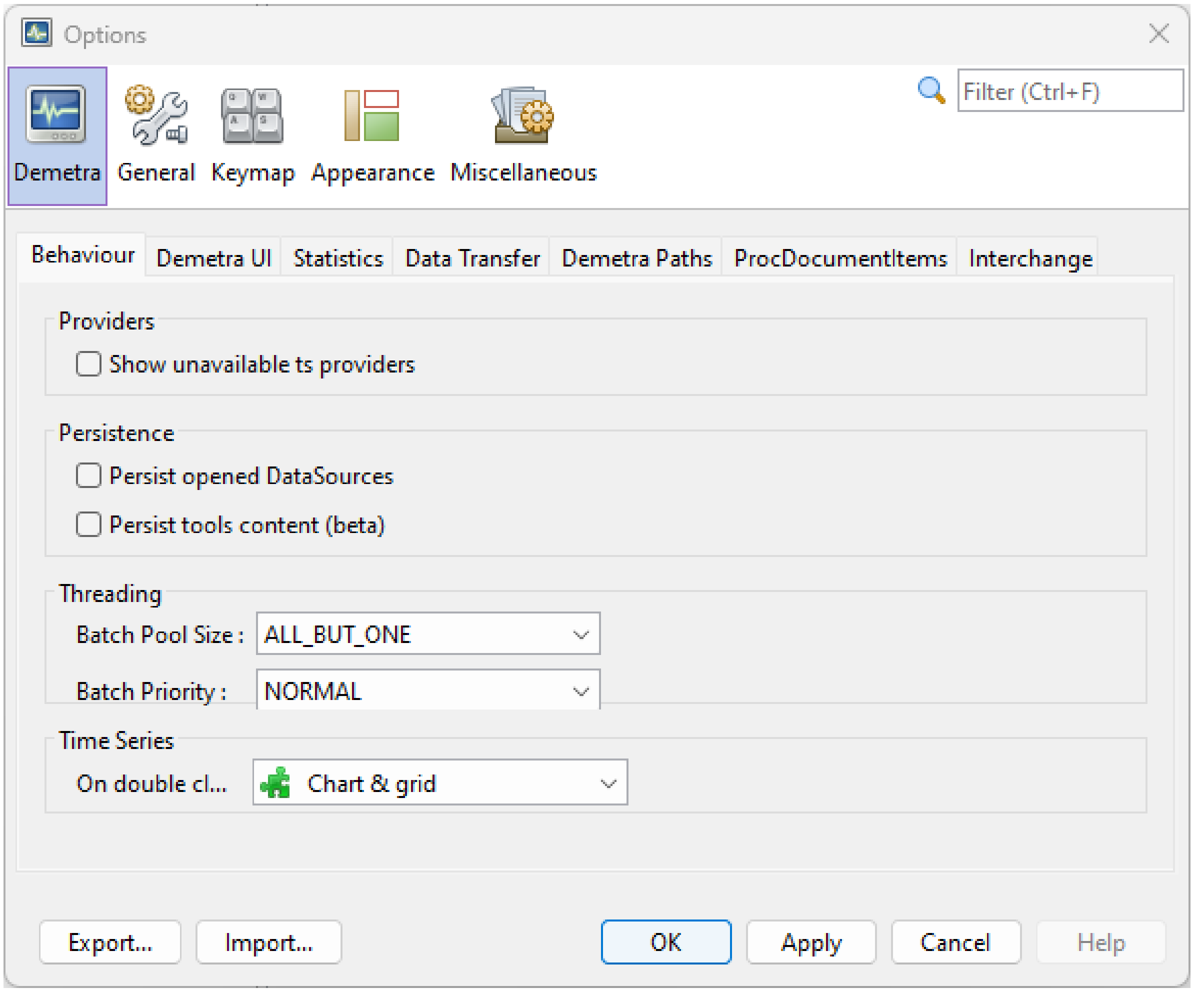


Figure 147: **Demetra panel in v2**

- and *Demetra Paths*.

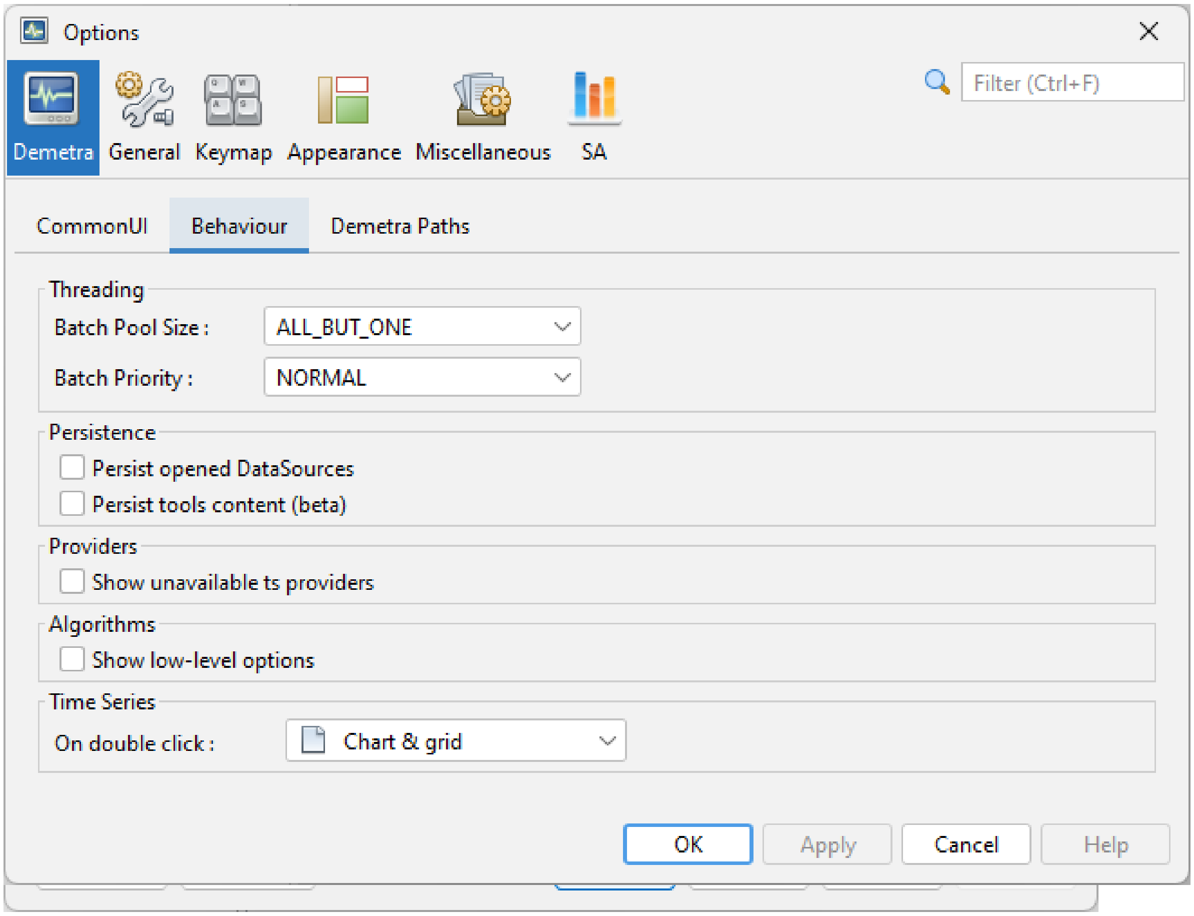


Figure 148: **Demetra panel in v3**

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**).

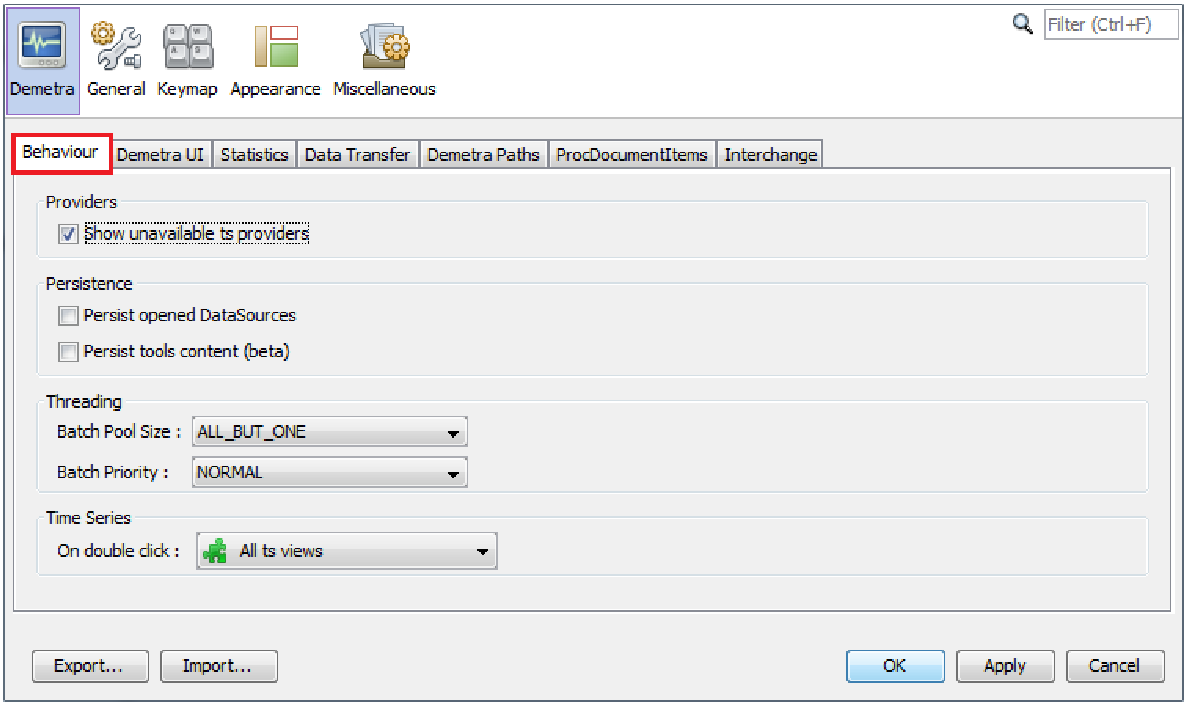


Figure 149: **The content of the *Behavior* tab**

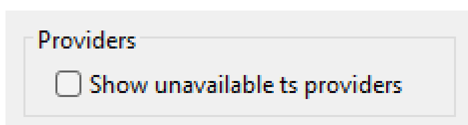


Figure 150: **The option Providers**

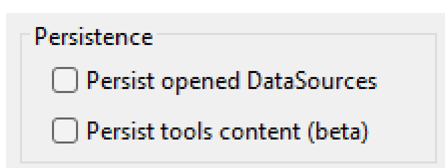


Figure 151: **The option Persistence**

- **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.

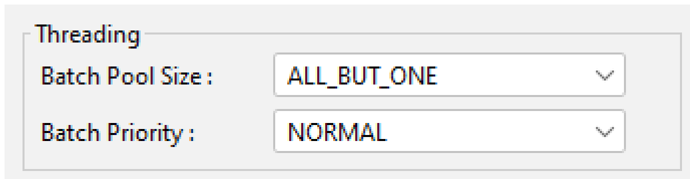


Figure 152: **The option Threading**

- **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.

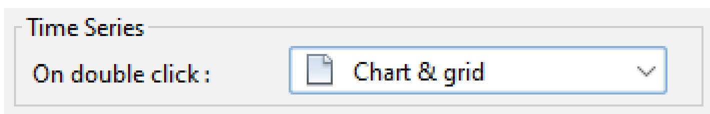


Figure 153: **The option Time Series**

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

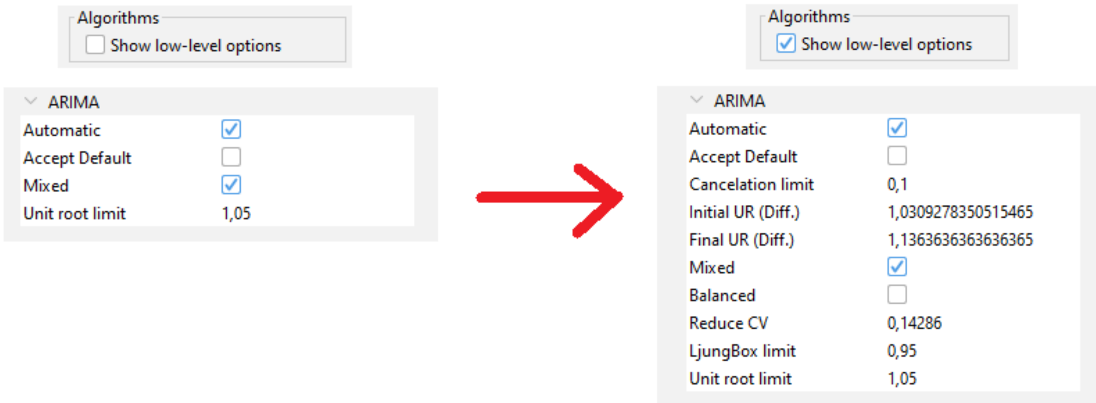


Figure 154: **Low level options on ARIMA specification**

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**).

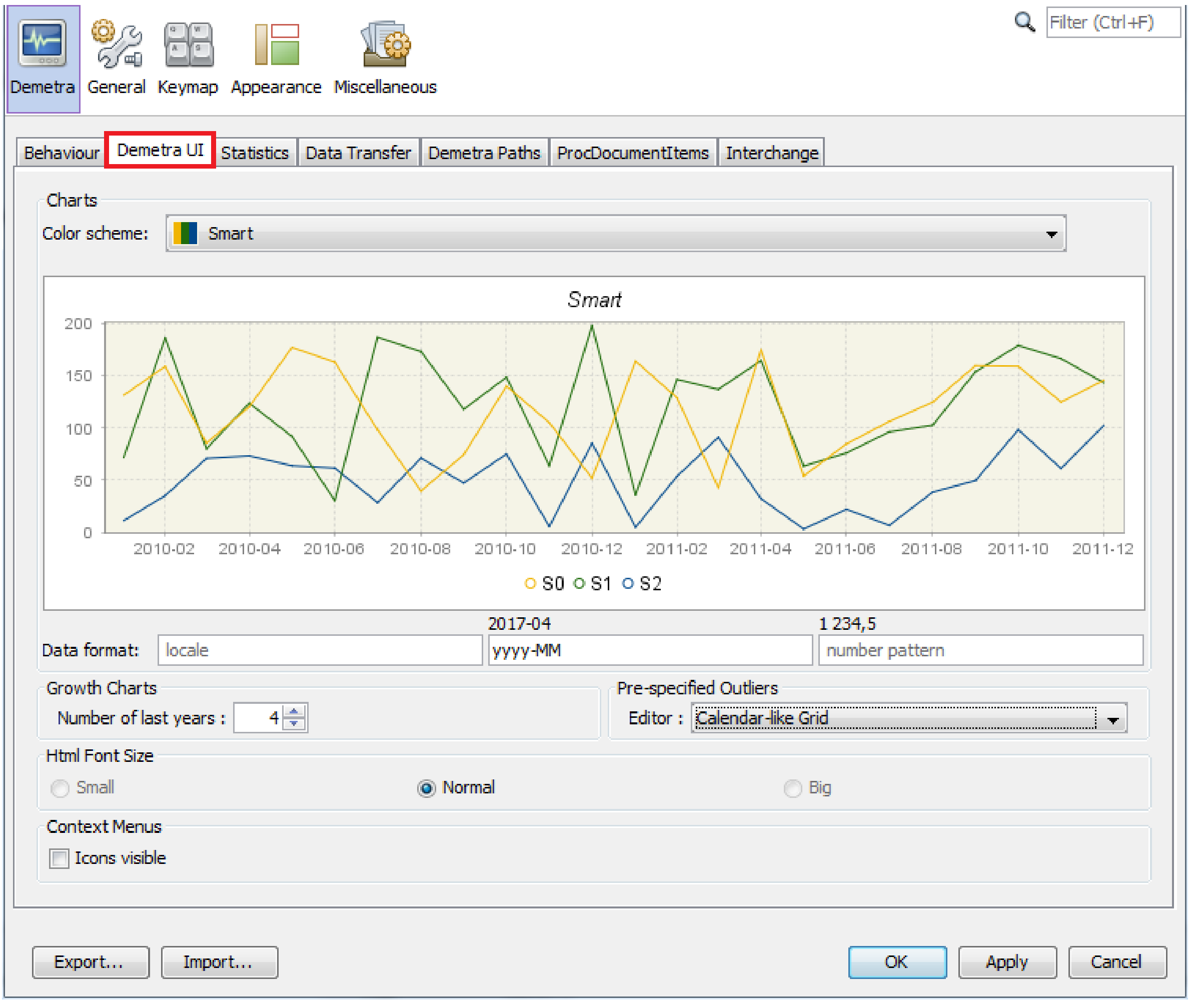


Figure 155: **The content of the *Demetra UI* tab**

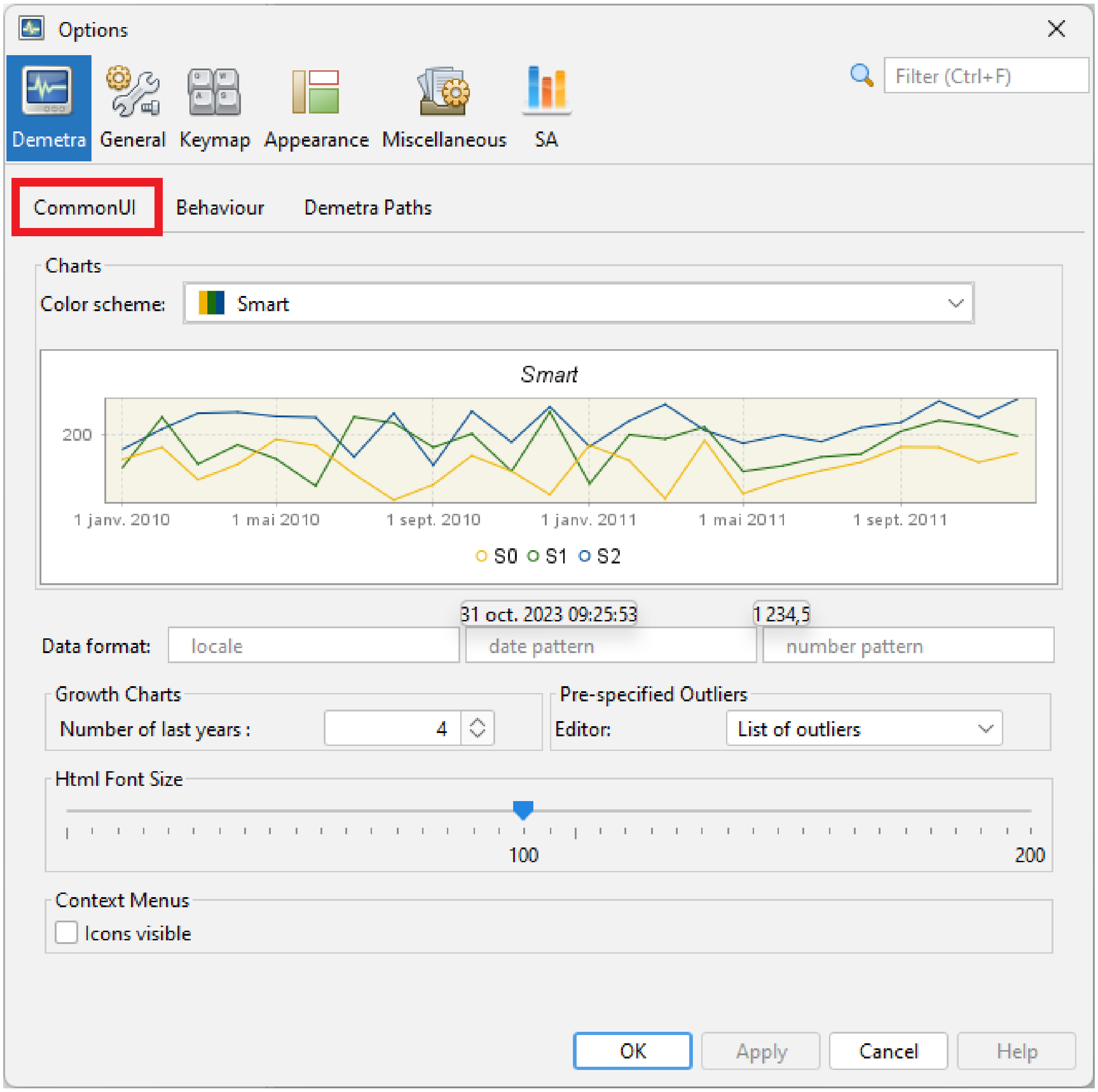


Figure 156: **CommonUI tab in v3**

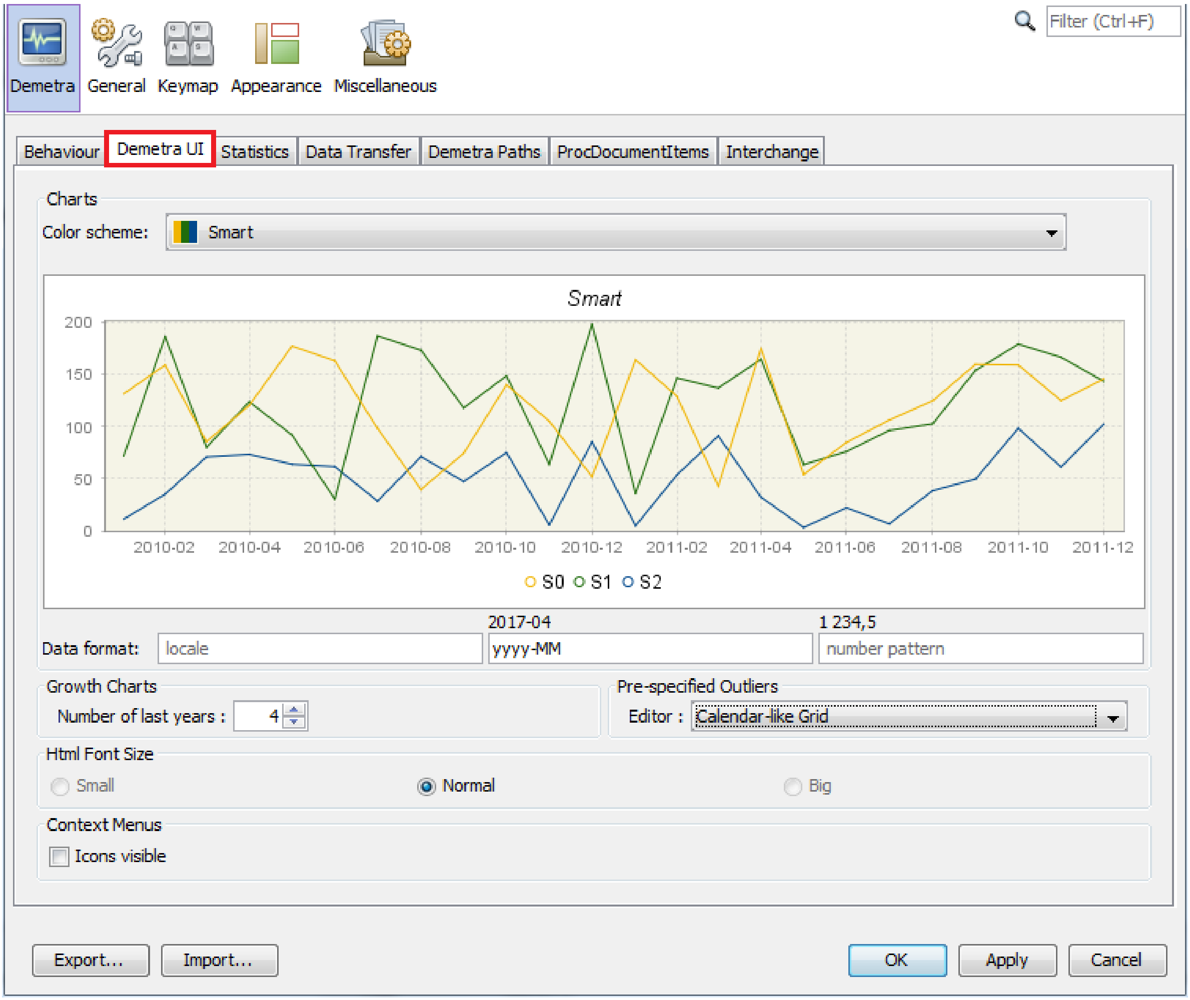


Figure 157: **The content of the *Demetra UI* tab in v2**

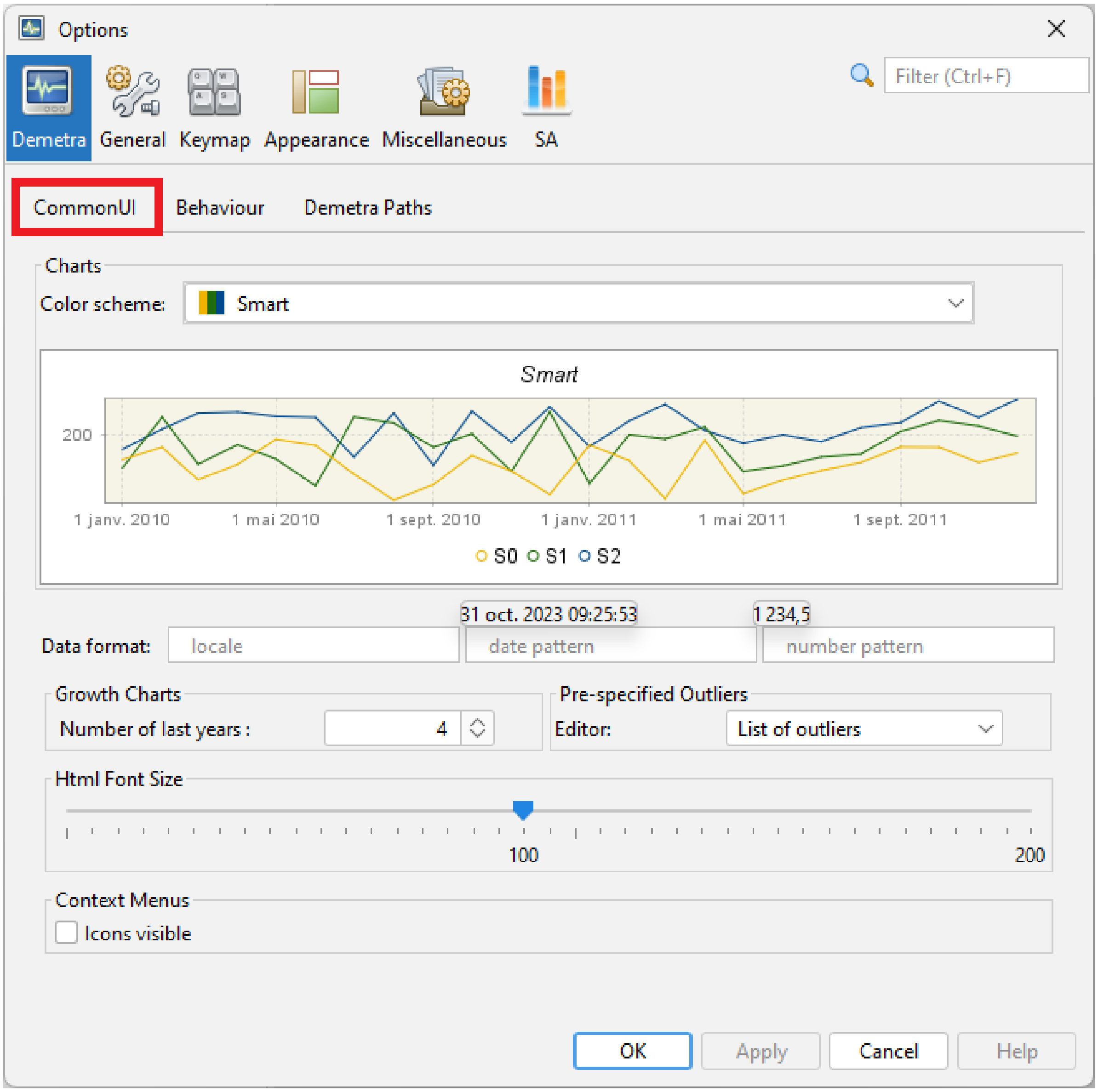


Figure 158: **CommonUI tab in v3**

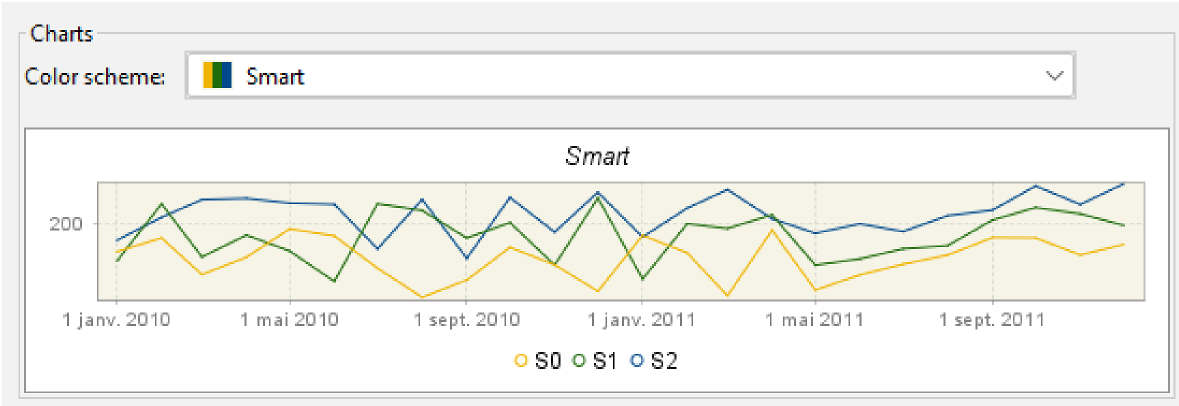


Figure 159: **The option Charts**

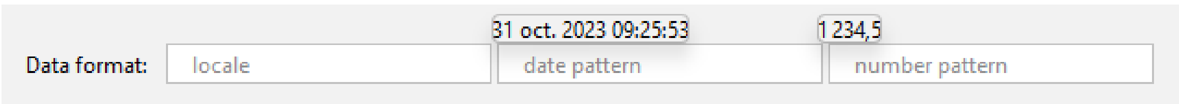


Figure 160: **The option Data format**

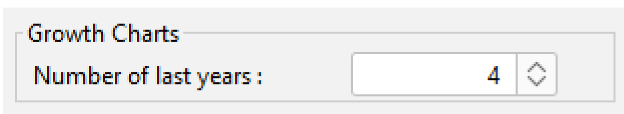


Figure 161: **The option Growth rates**

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

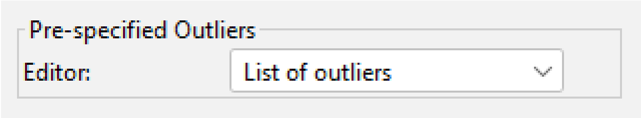


Figure 162: **The option Pre-specified Outliers**

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

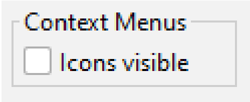


Figure 163: **The option Context Menus**

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.

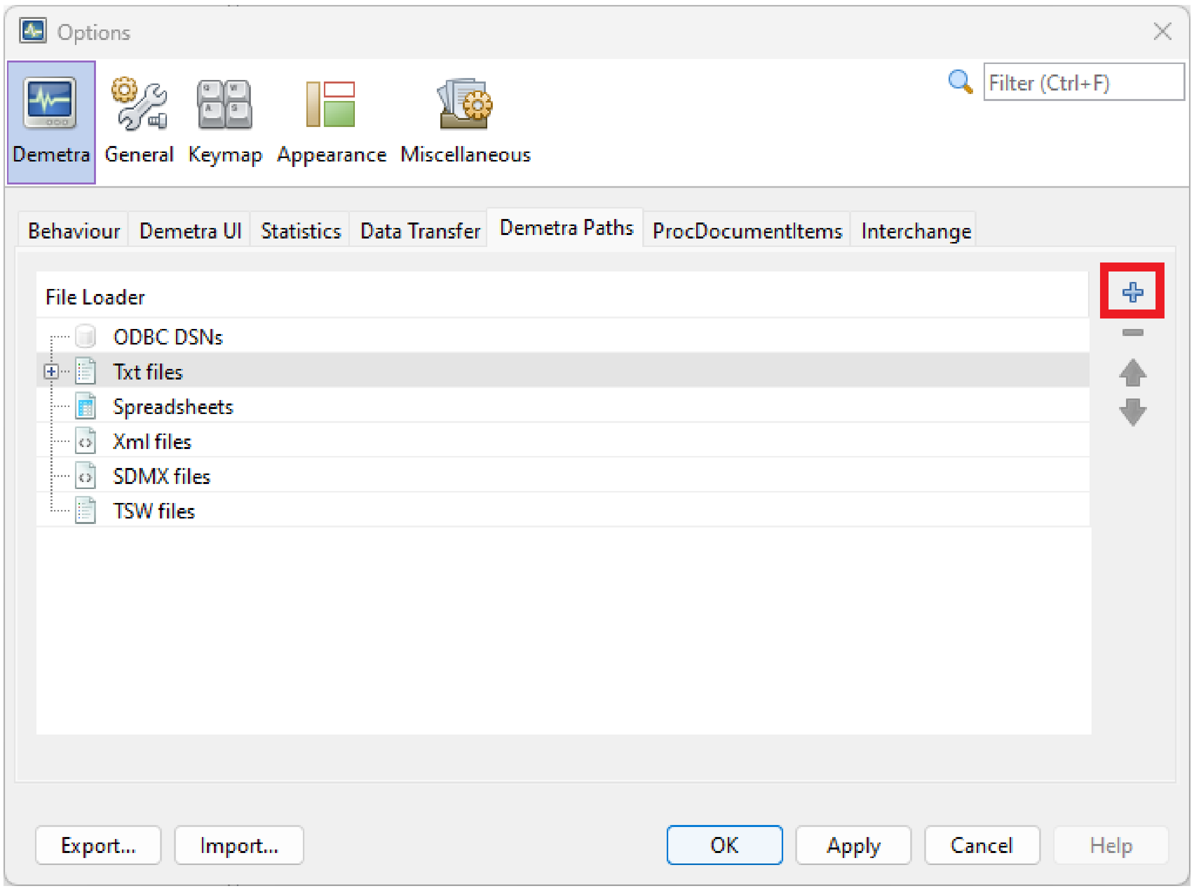


Figure 164: **The content of the *Demetra Paths* tab in v2**

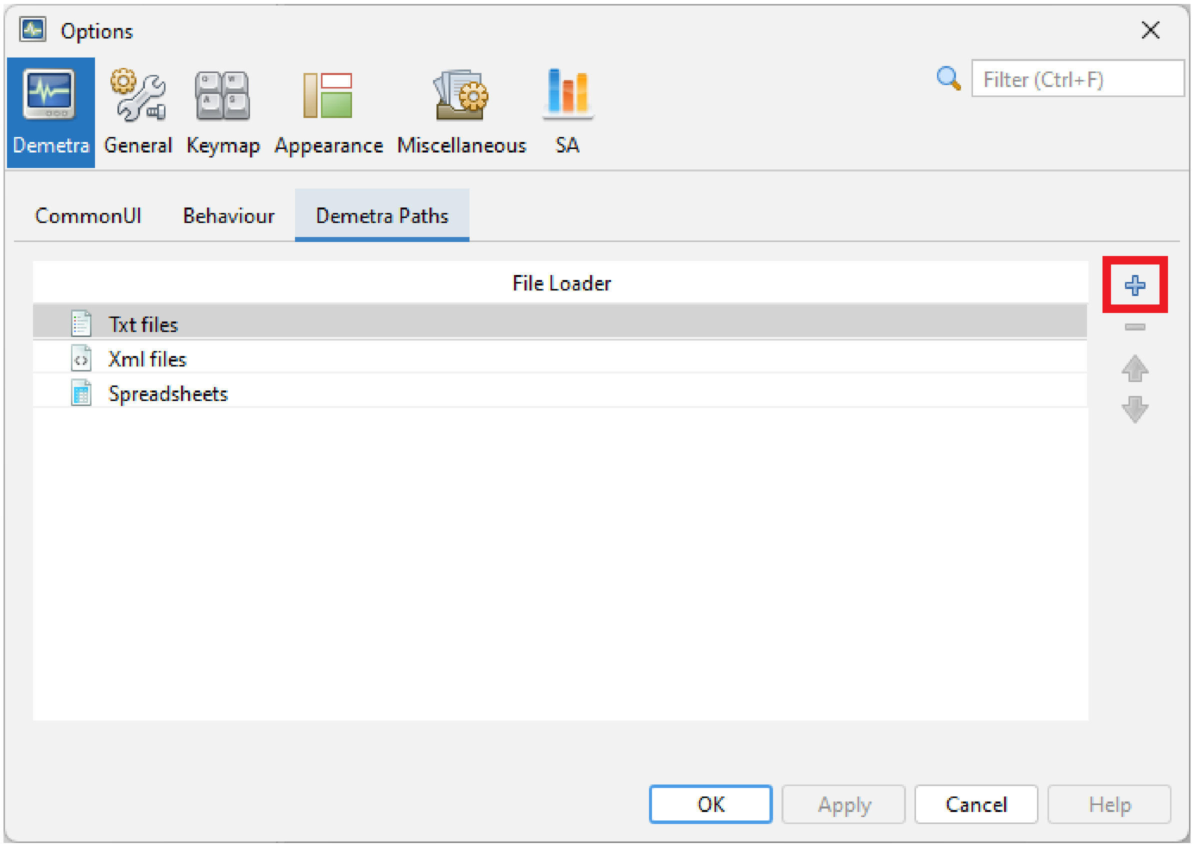
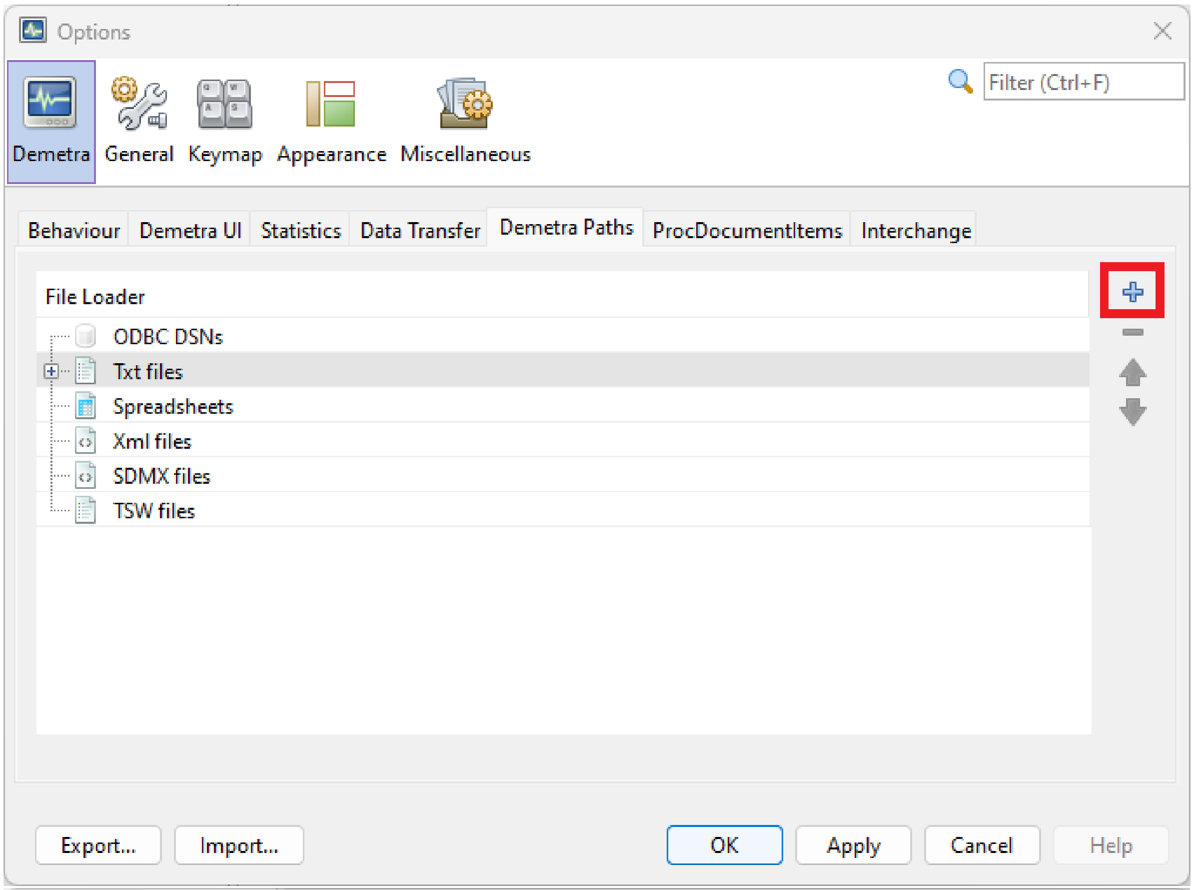
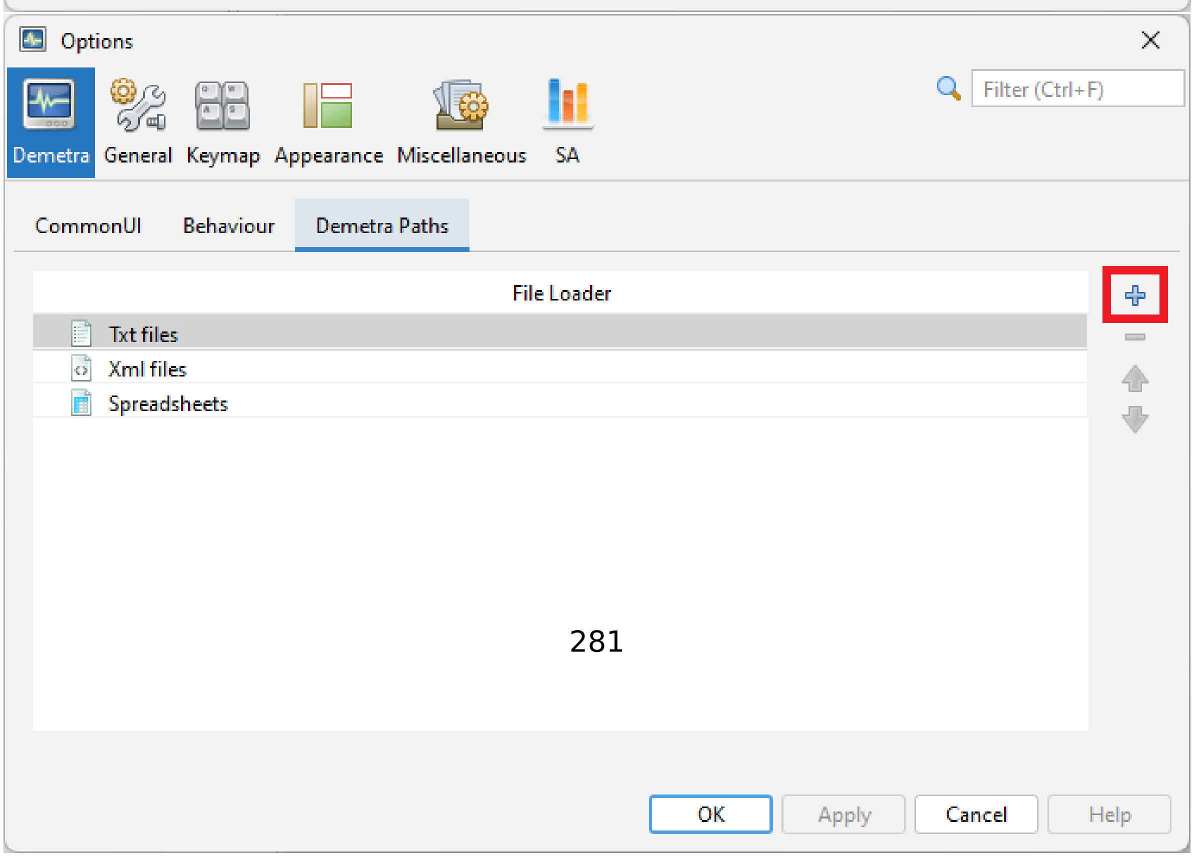


Figure 165: **The content of the *Demetra Paths* tab in v3**

0.0.0.5 v2

0.0.0.6 v3





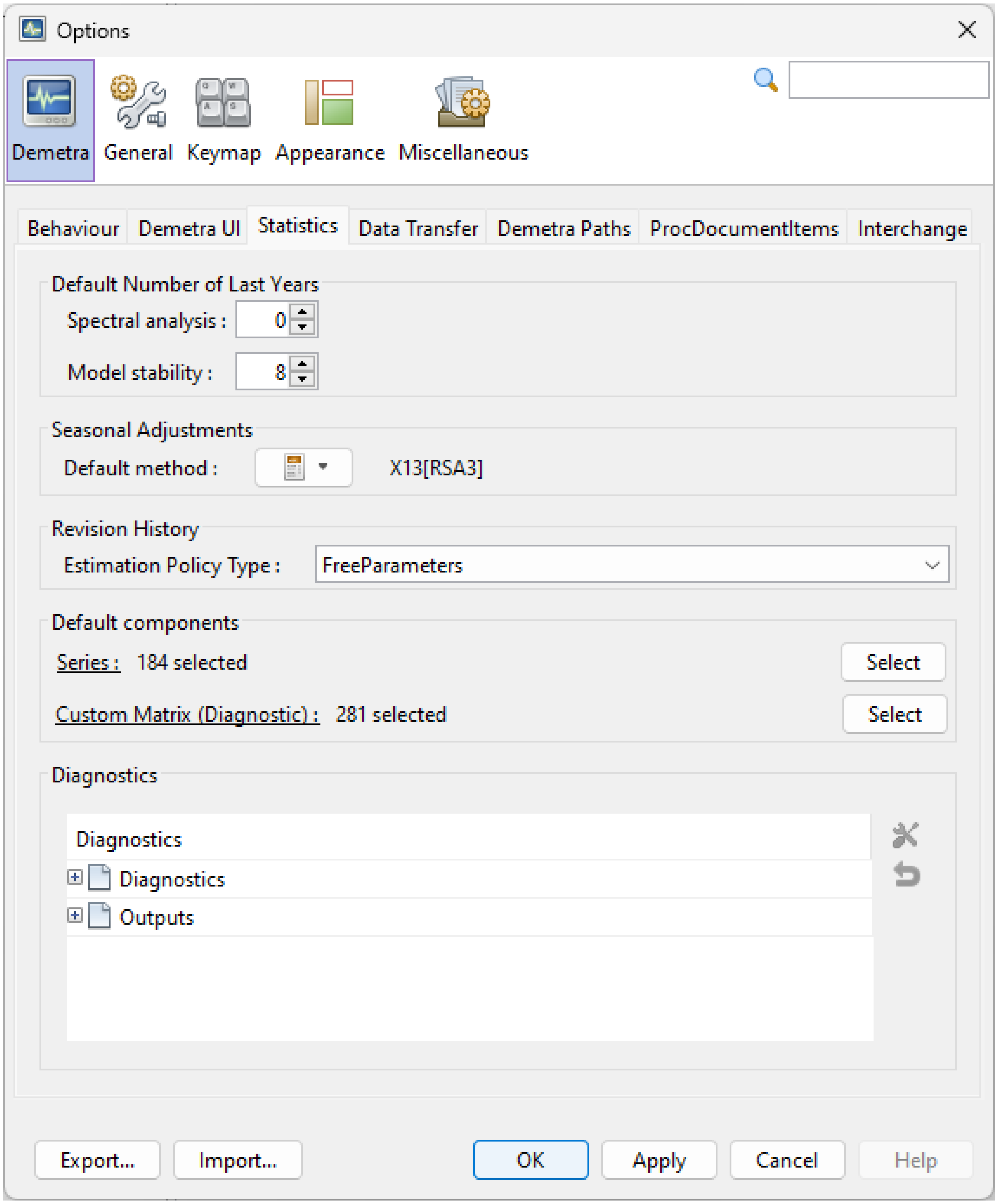


Figure 166: **Statistics tab in v2**

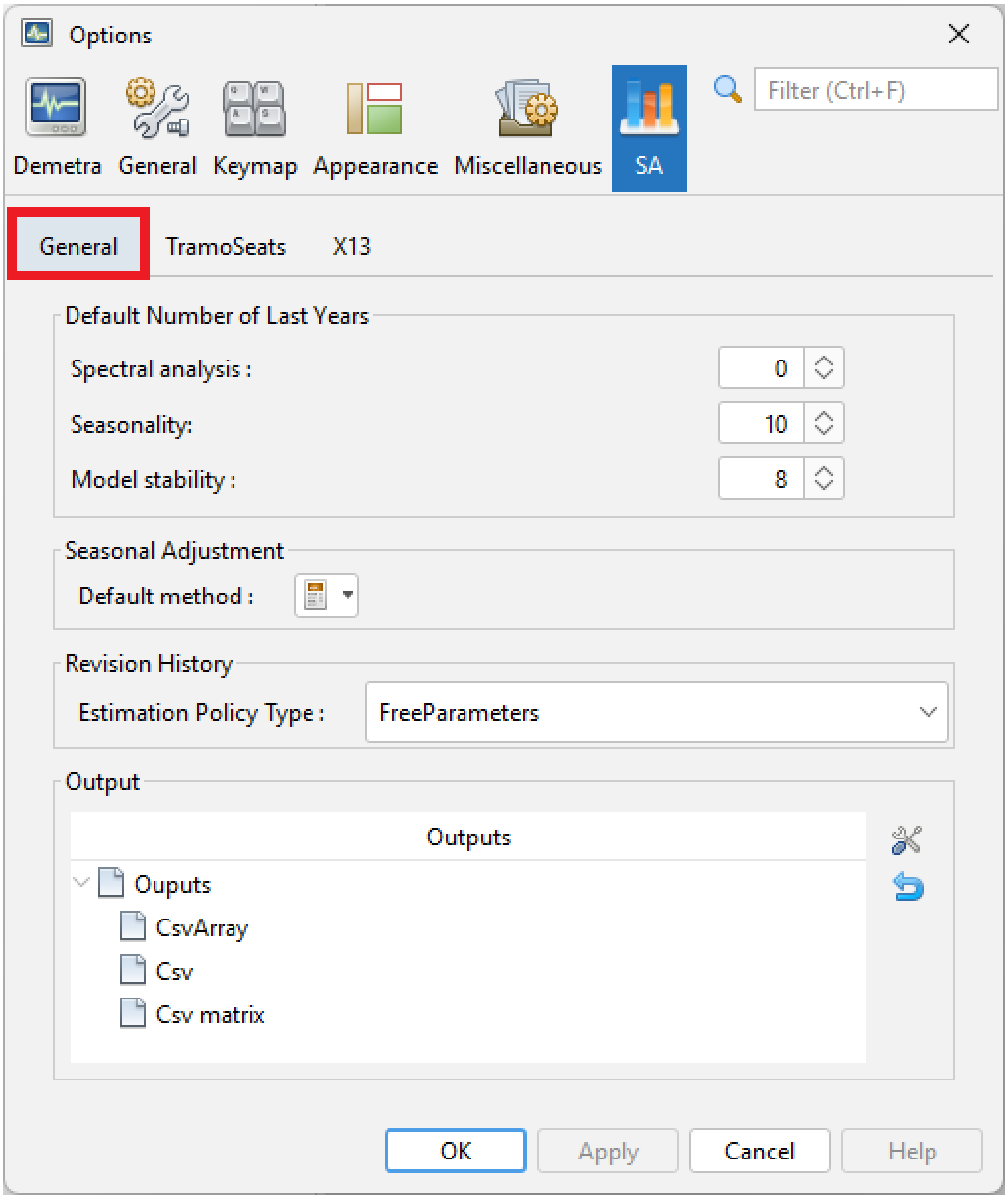
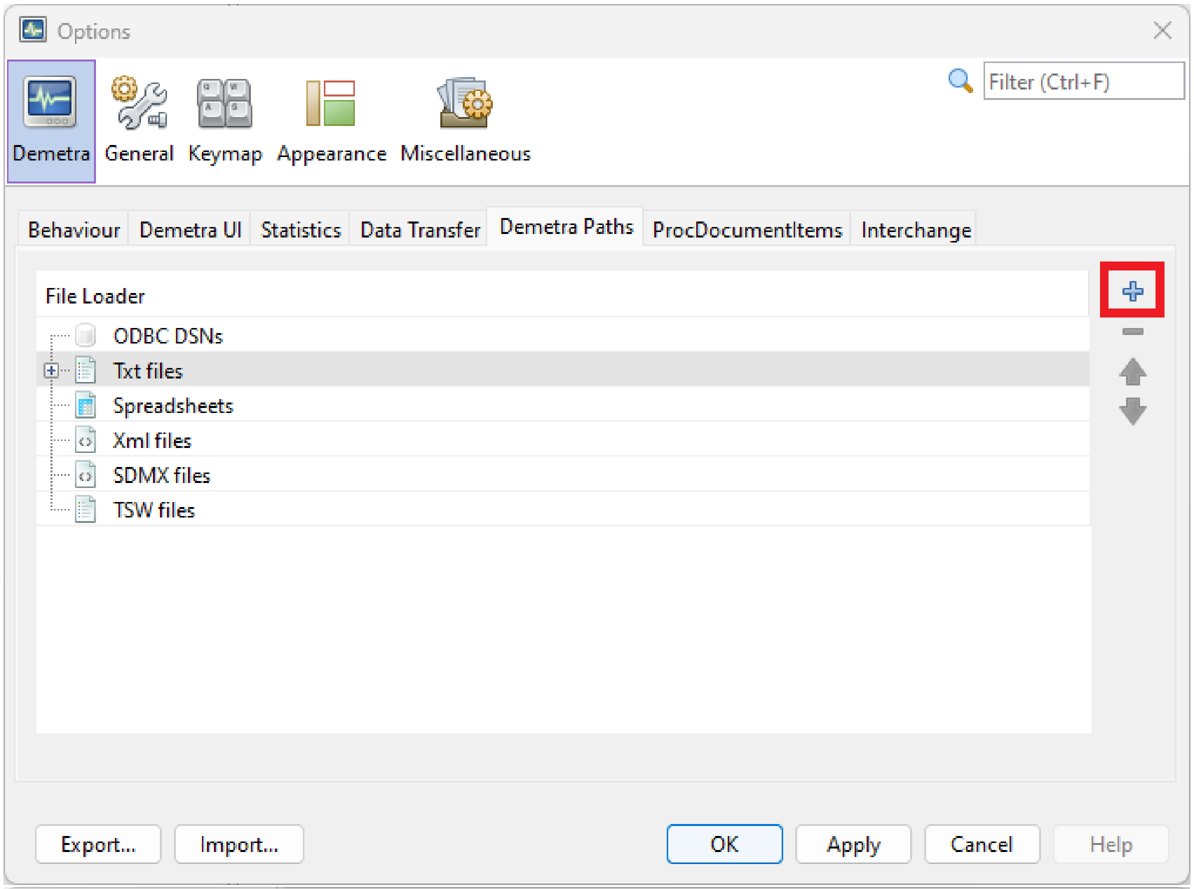


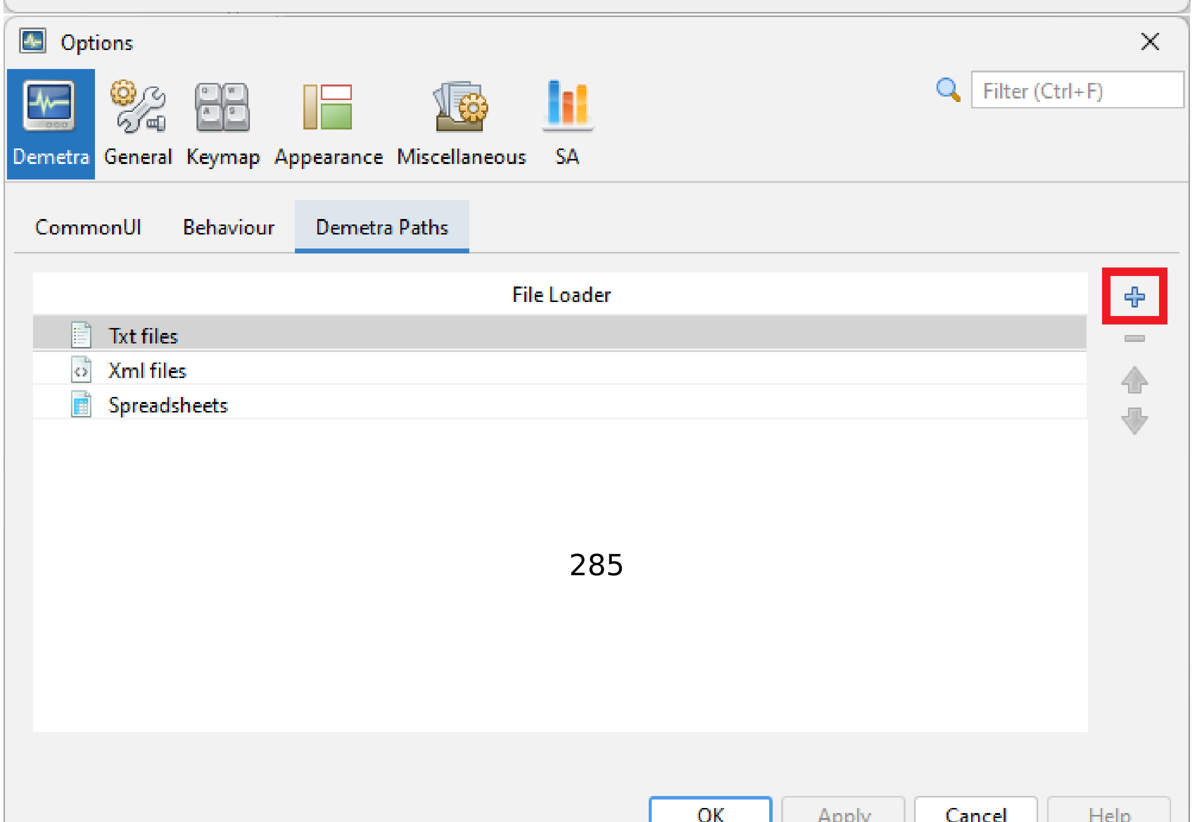
Figure 167: **SA panel**

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

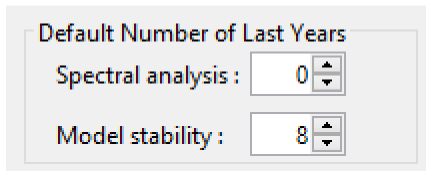


Figure 168: **Default Number of Last Years option in v2**

0.0.0.10 v3

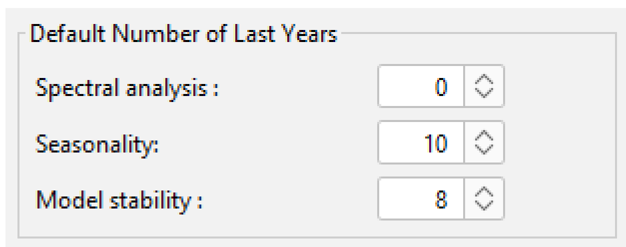
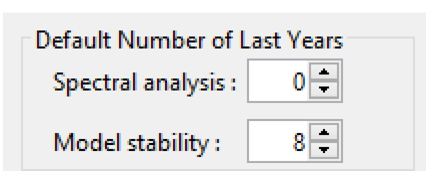
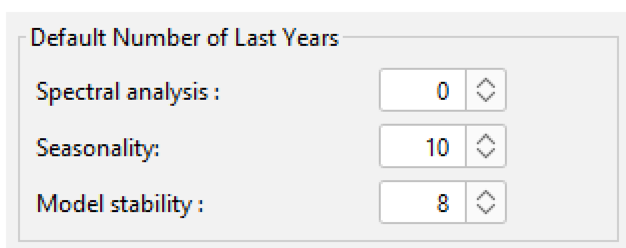


Figure 169: **Default Number of Last Years option in v3**





- The default pre-defined specification for seasonal adjustment (**Seasonal Adjustment**);

- The type of the analysis of revision history (**Revision History**):

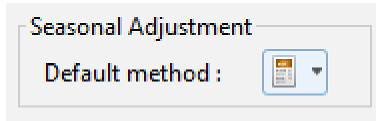


Figure 170: **Seasonal Adjustment option**

- *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).

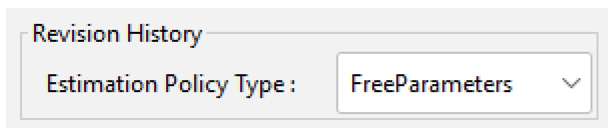


Figure 171: **Revision History option in v2**

- 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)

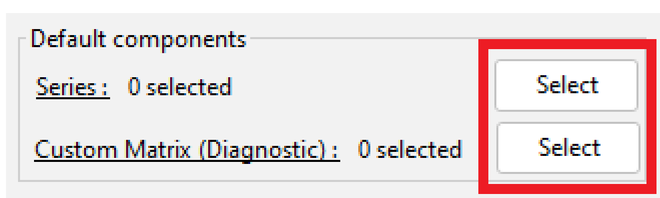


Figure 172: **The *Default components* section on the *Statistics* tab**

  - **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).

0.0.0.11 v2

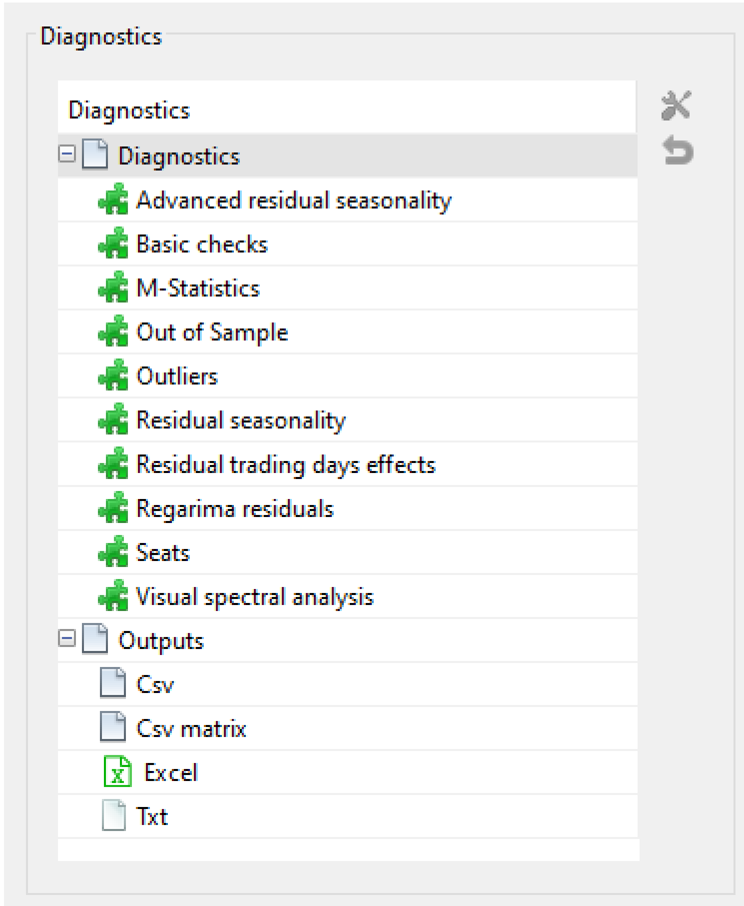


Figure 173: **The *Diagnostics* section in v2**

0.0.0.12 v3

In v3, you can find this settings in the SA panel in the tabs TramoSeats and X-13:

In v3, you can find this settings in the SA panel in the tabs TramoSeats and X-13:

To modify the settings for a particular measure, double click on a selected row (select the test's name from the list and click on the working tools button), introduce changes in the pop-up window and click the **OK** button.

To reset the default settings for a given test, select this test from the list and click on the backspace button situated below the working tools button. The description

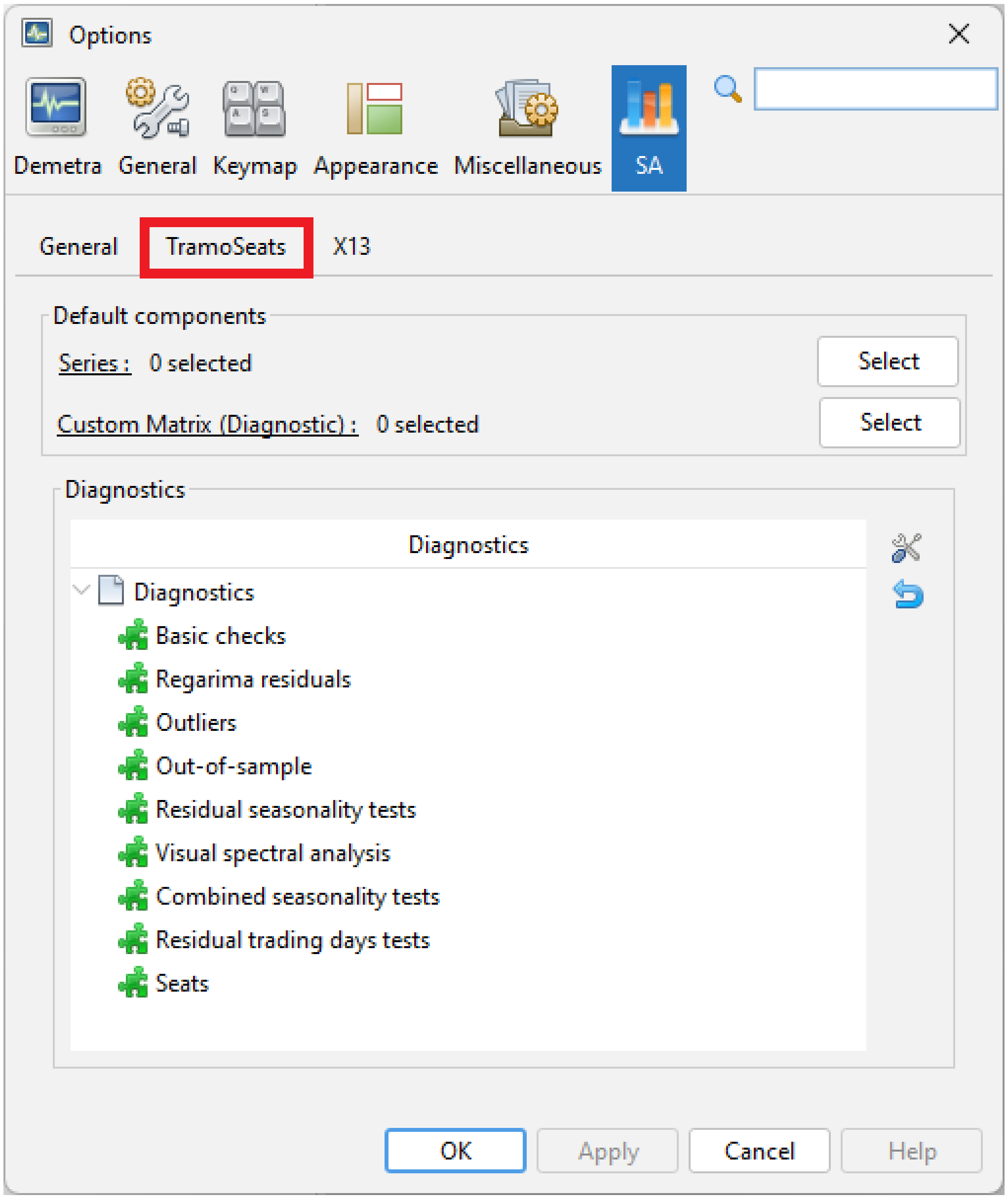


Figure 174: **Settings for the quality measures and tests in Tramoseats in v3**

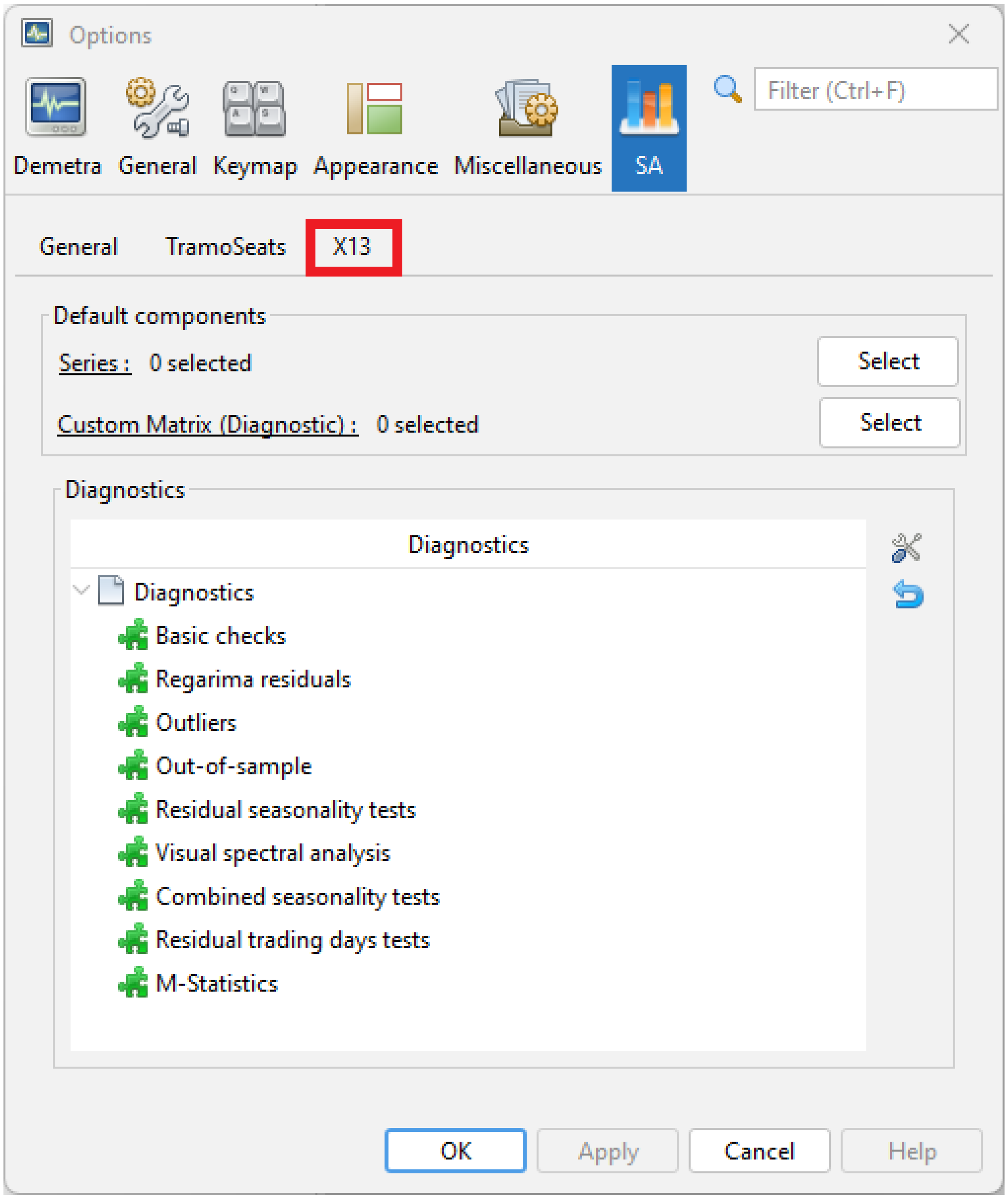


Figure 175: **Settings for the quality measures and tests in X-13 in v3**

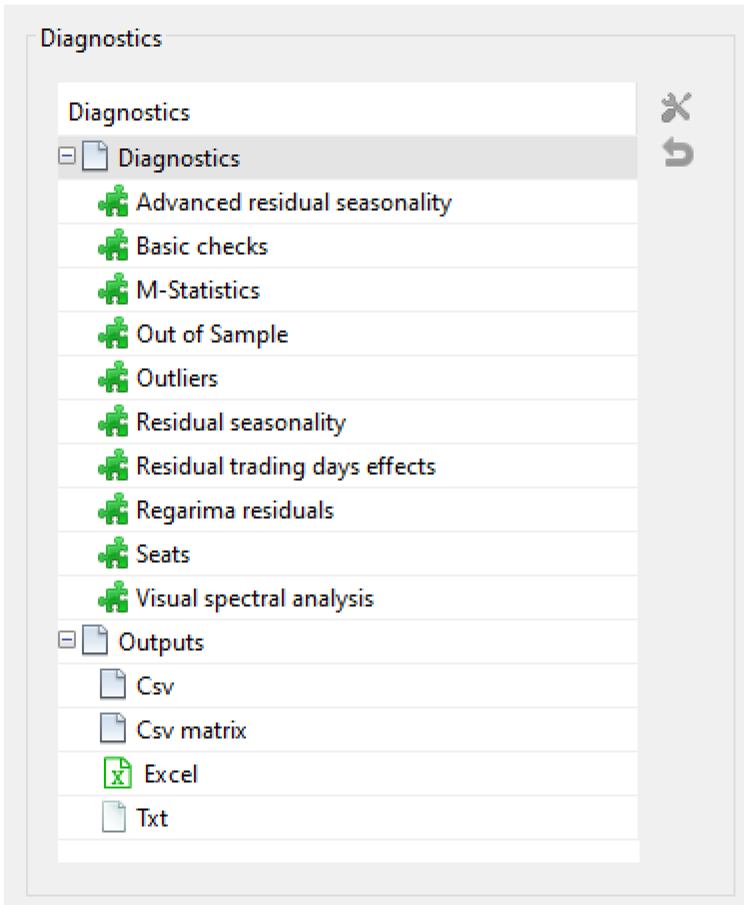


Figure 176: **The *Diagnostics* section in v2**

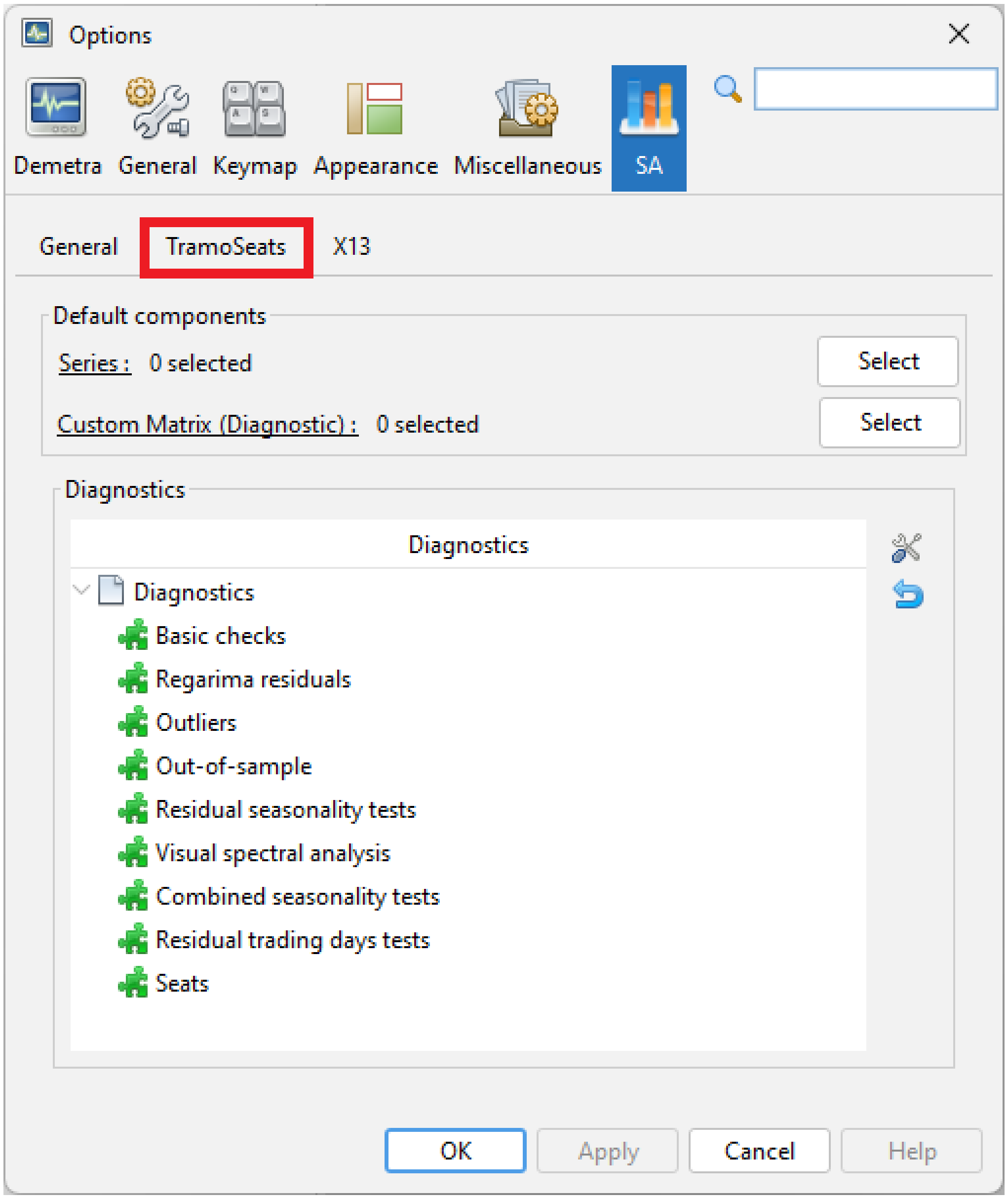


Figure 177: **Settings for the quality measures and tests in Tramoseats in v3**

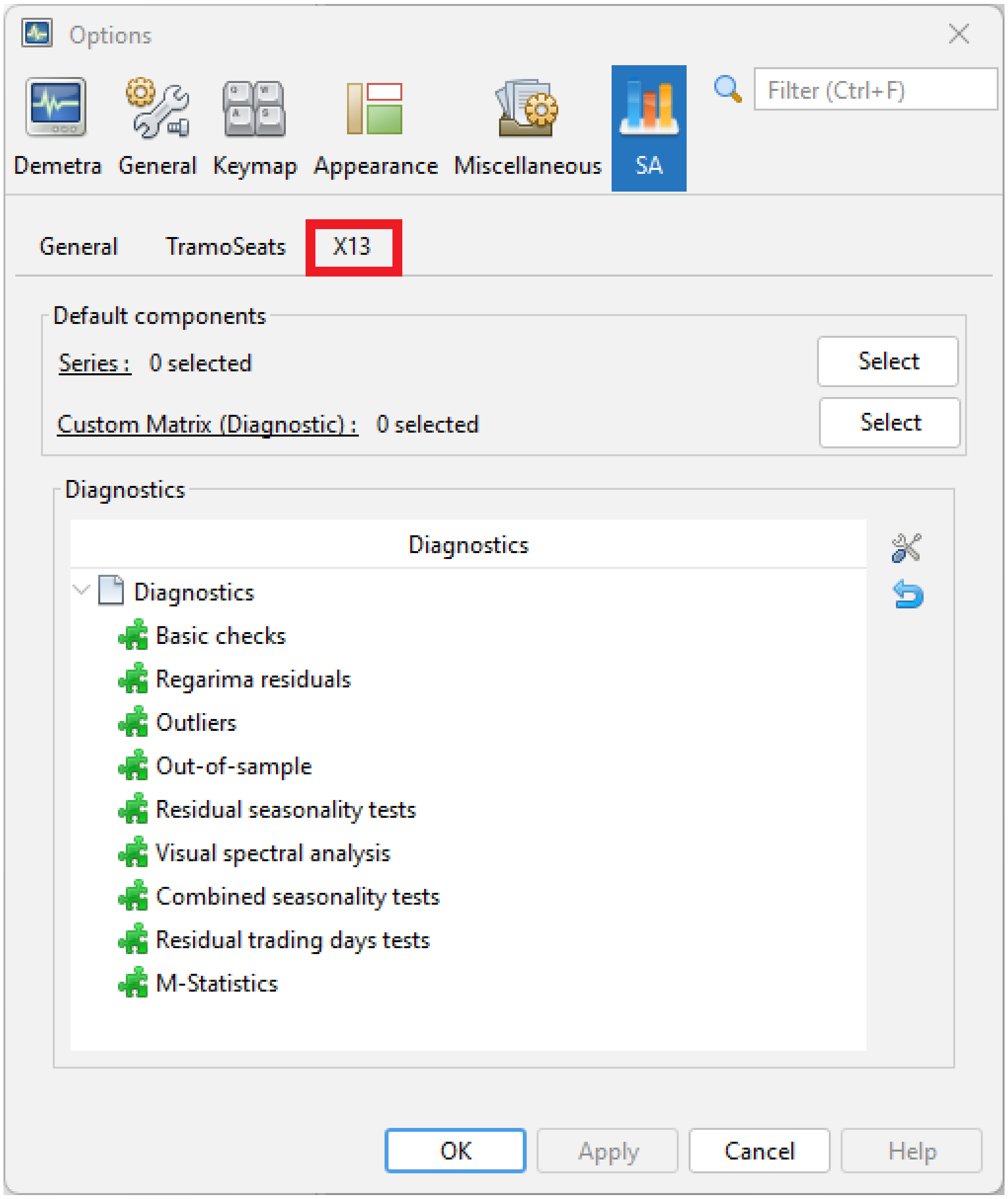


Figure 178: **Settings for the quality measures and tests in X-13 in v3**

of the parameters for each quality measure and test used in a diagnostic procedure can be found in the output from modelling and the output from seasonal adjustment nodes.

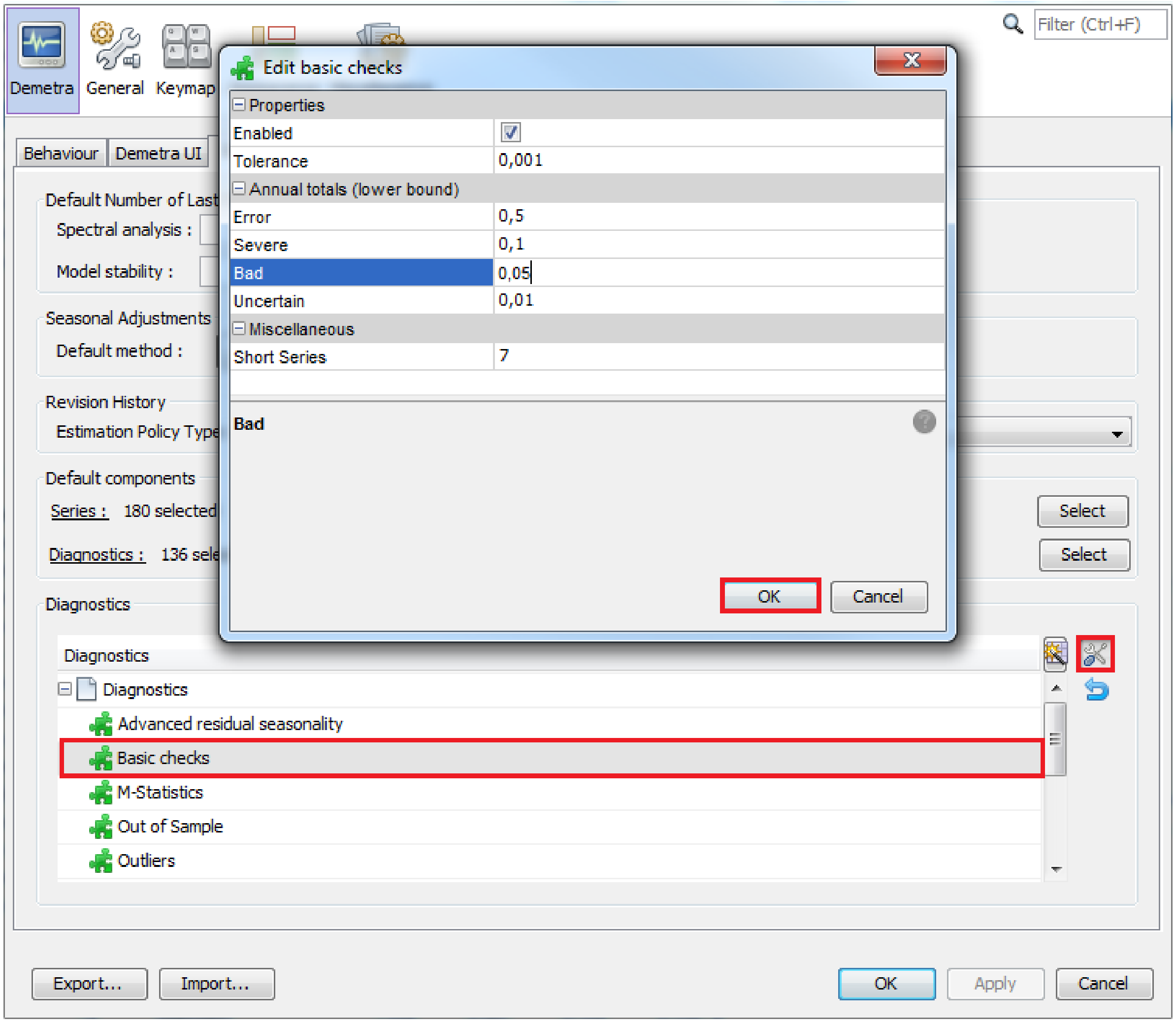


Figure 179: **The panel for modification of the settings for the tests in the *Basic checks* section**

The users can customize the diagnostics and they can specify the default settings for different outputs. Their preferences are saved between different sessions of JDemetra+. This new feature is accessible in the *Statistics* tab of the *Options* panel.

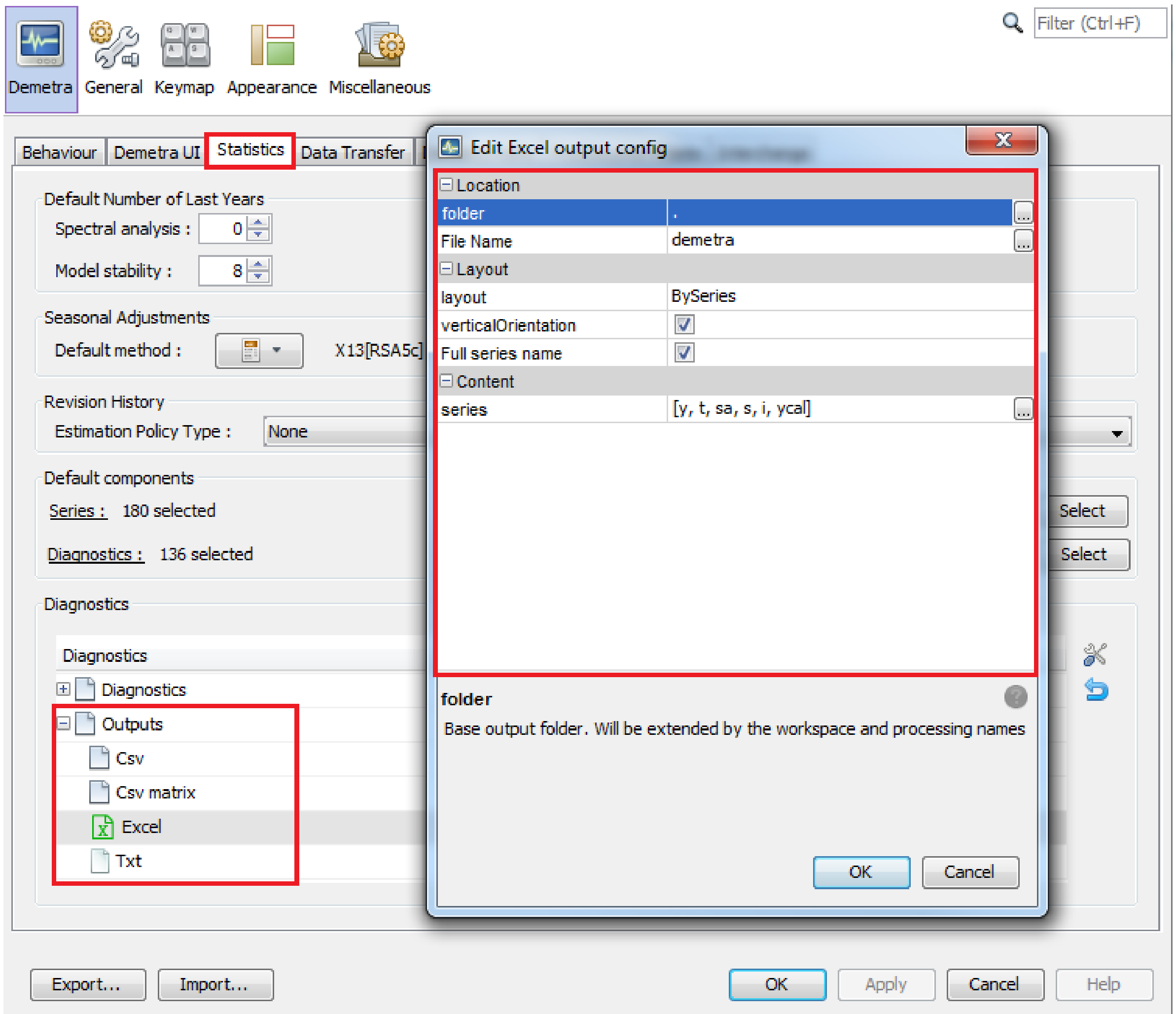


Figure 180: **The settings of the output files**

Data Transfer tab

0.0.0.13 v2

The *Data Transfer* tab contains multiple options that define the behaviour of the drag and drop and copy-paste actions. To change the default settings, double click on the selected item. Once the modifications are introduced, confirm them with the **OK** button.

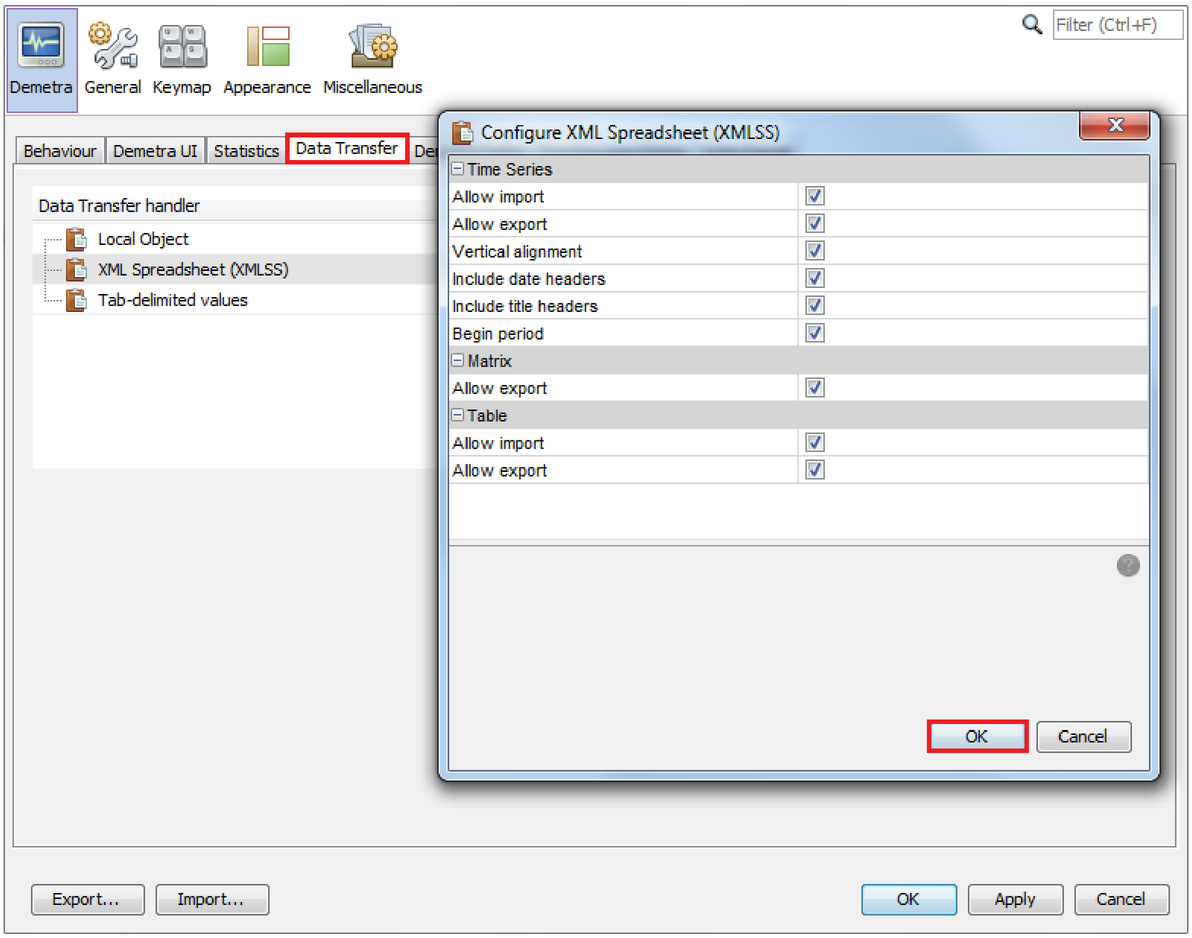


Figure 181: **The content of the *Data Transfer* tab in v2**

0.0.0.14 v3

In v3, there is no equivalent of the Data Transfer tab.

In v2, the *Data Transfer* tab contains multiple options that define the behaviour of the drag and drop and copy-paste actions. To change the default settings, double click on the selected item. Once the modifications are introduced, confirm them with the **OK** button.

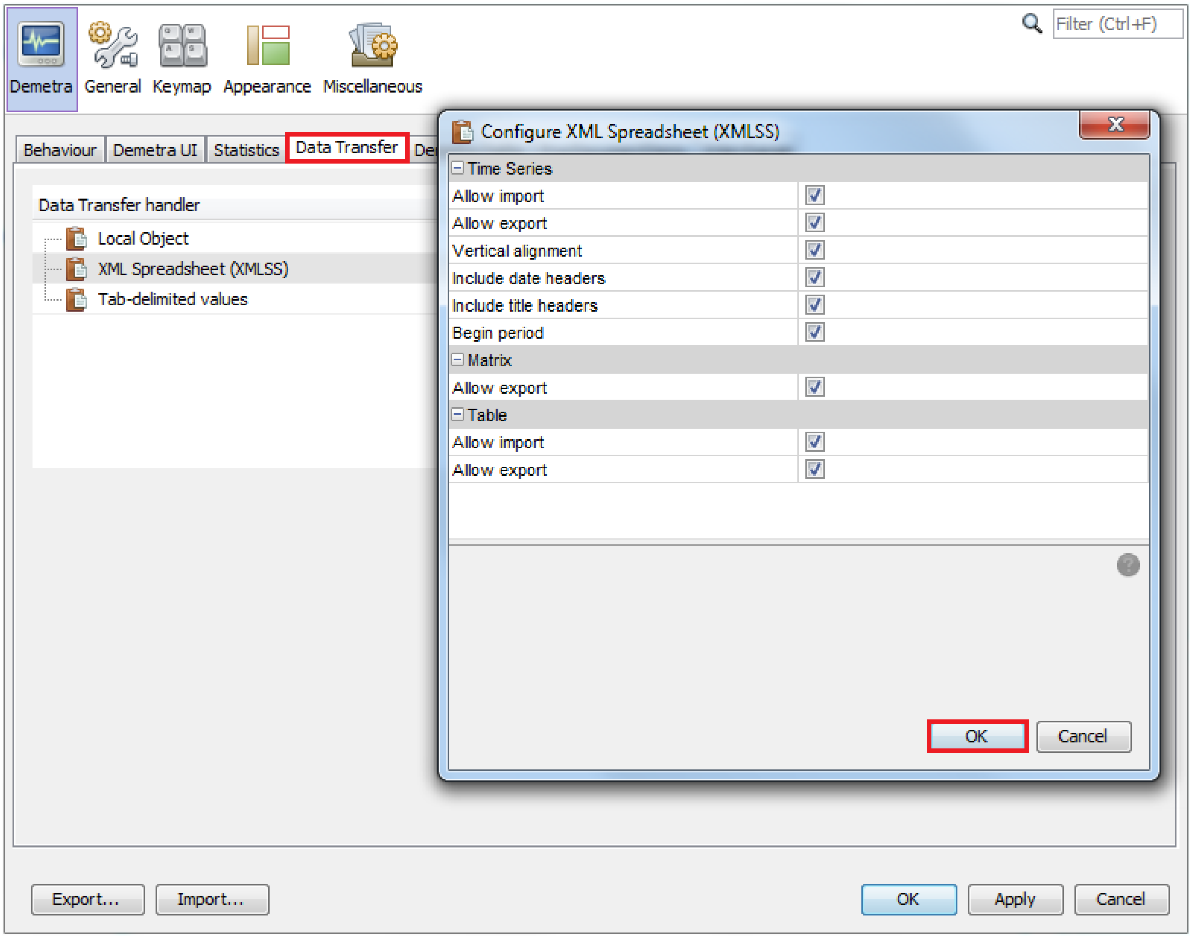


Figure 182: **The content of the *Data Transfer* tab in v2**

In v3, there is no equivalent of the *Data Transfer* tab.

ProcDocumentItems tab

0.0.0.15 v2

ProcDocumentItems includes a list of all reports available for processed documents like seasonal adjustment.

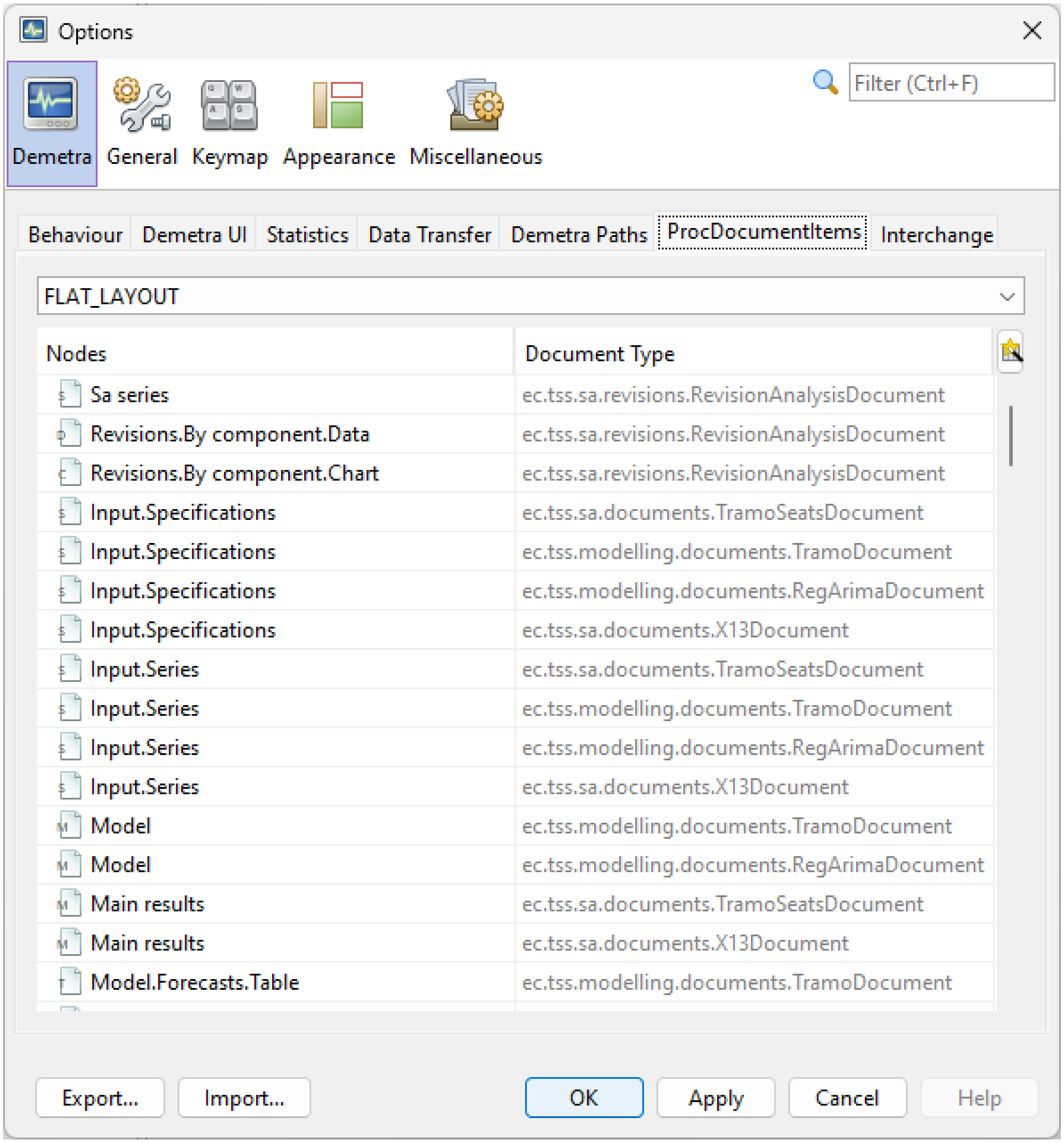


Figure 183: **The content of the *ProcDocumentItems* tab in v2**

0.0.0.16 v3

In v3, there is no equivalent of the *ProcDocumentItems* tab.

In v2, *ProcDocumentItems* includes a list of all reports available for processed documents like seasonal adjustment.

In v3, there is no equivalent of the *ProcDocumentItems* tab.

Interchange tab

0.0.0.17 v2

The *Interchange* tab lists the protocols that can be used to export/import information like calendars, specifications, etc.. For the time being, the user cannot customize the way the standard exchanges are done. However, such features could be implemented in plug-ins.

0.0.0.18 v3

In v3, there is no equivalent of the *Interchange* tab.

In v2, the *Interchange* tab lists the protocols that can be used to export/import information like calendars, specifications, etc.. For the time being, the user cannot customize the way the standard exchanges are done. However, such features could be implemented in plug-ins.

In v3, there is no equivalent of the *Interchange* tab.

General panel

The next section, *General*, allows for the customisation of the proxy settings. A proxy is an intermediate server that allows an application to access the Internet. It is typically used inside a corporate network where Internet access is restricted. In JDemetra+, the proxy is used to get time series from remote servers like .Stat.

Keymap panel

Keymap provides a list of default key shortcuts to access some of the functionalities and it allows the user to edit them and to define additional shortcuts.

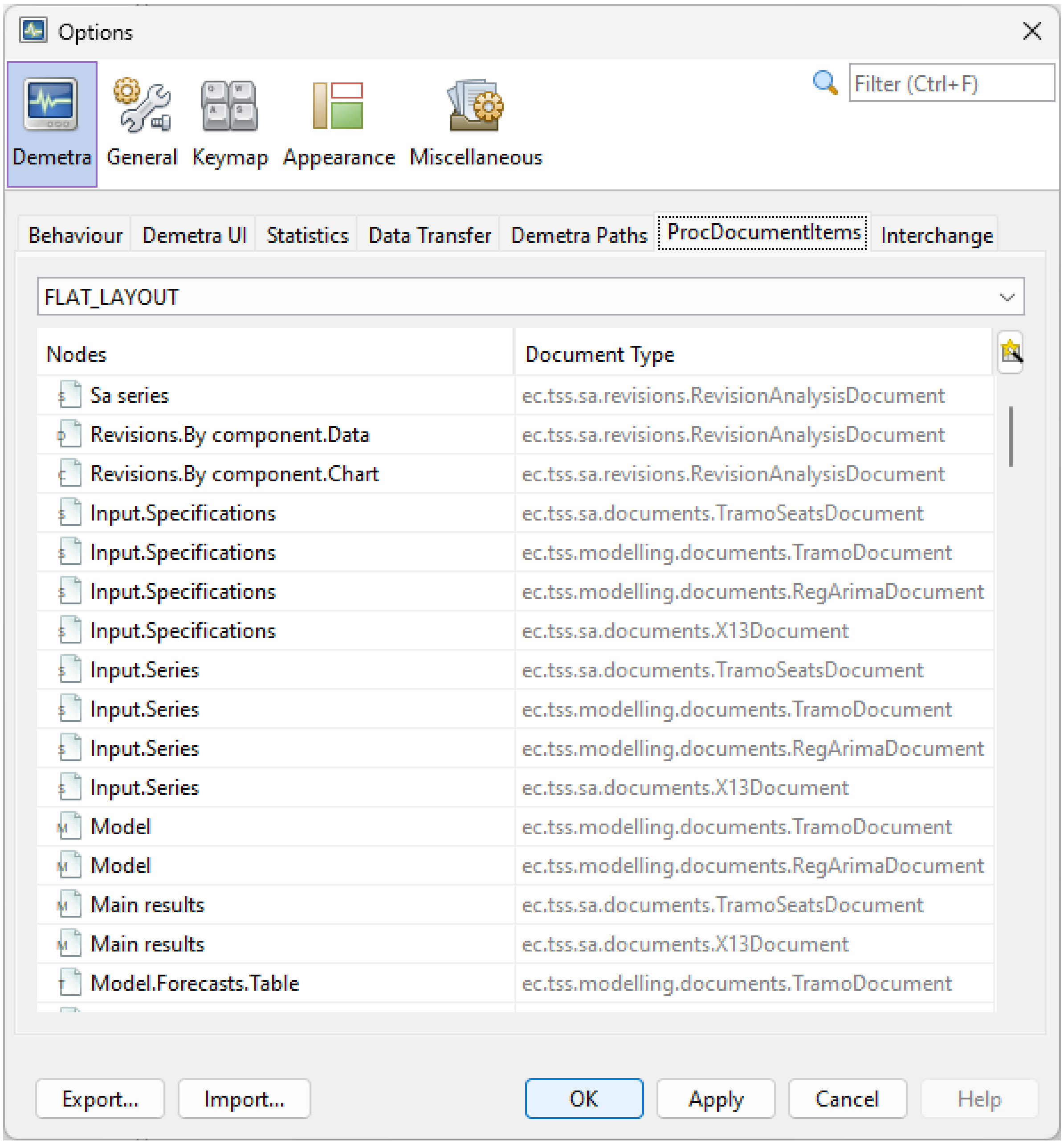


Figure 184: **The content of the *ProcDocumentItems* tab in v2**

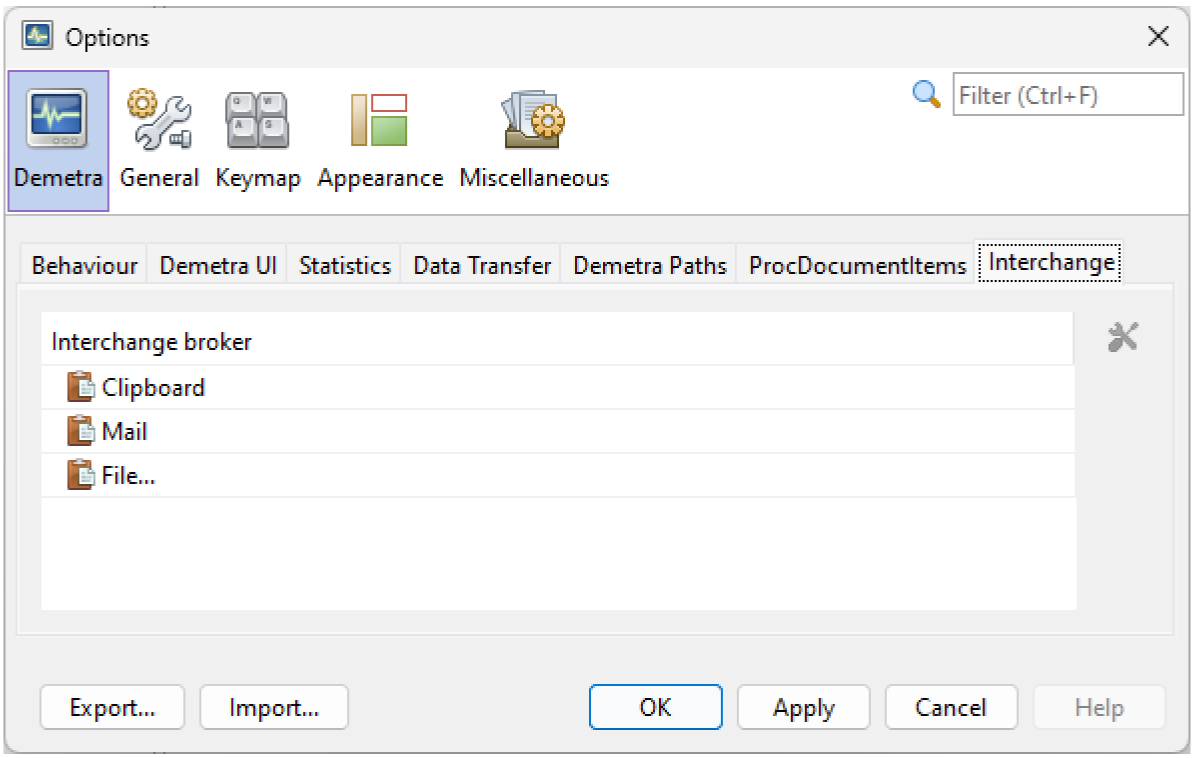


Figure 185: **The content of the *Interchange* tab in v2**

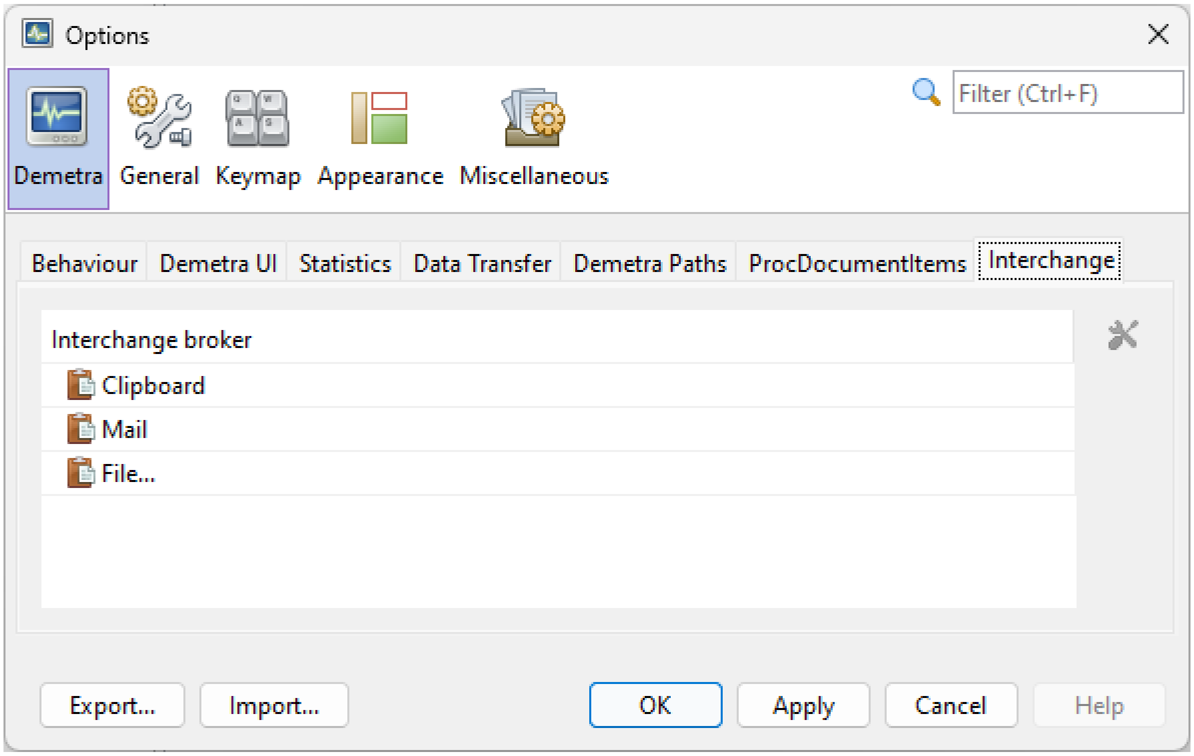


Figure 186: **The content of the *Interchange* tab in v2**

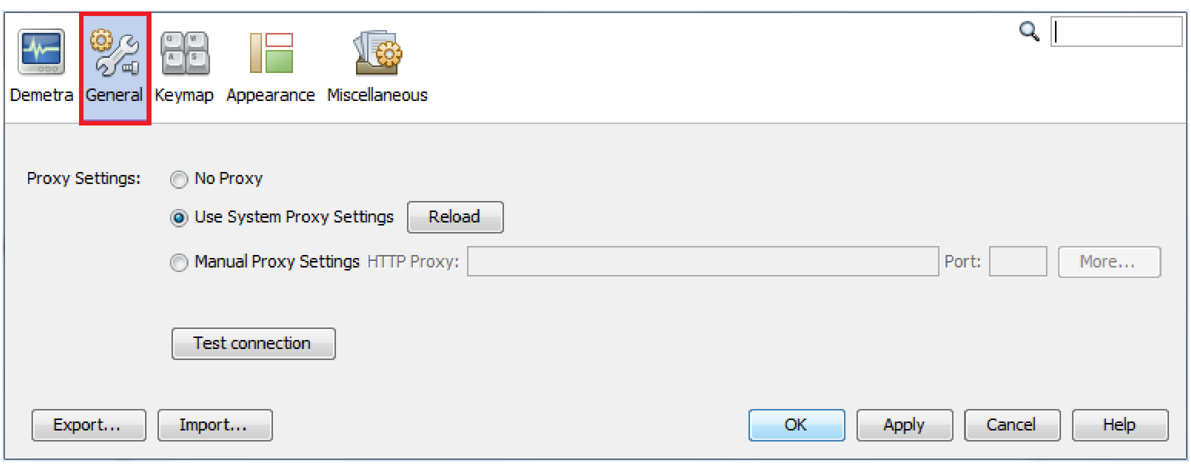


Figure 187: **The *General* tab**

0.0.0.1 v2

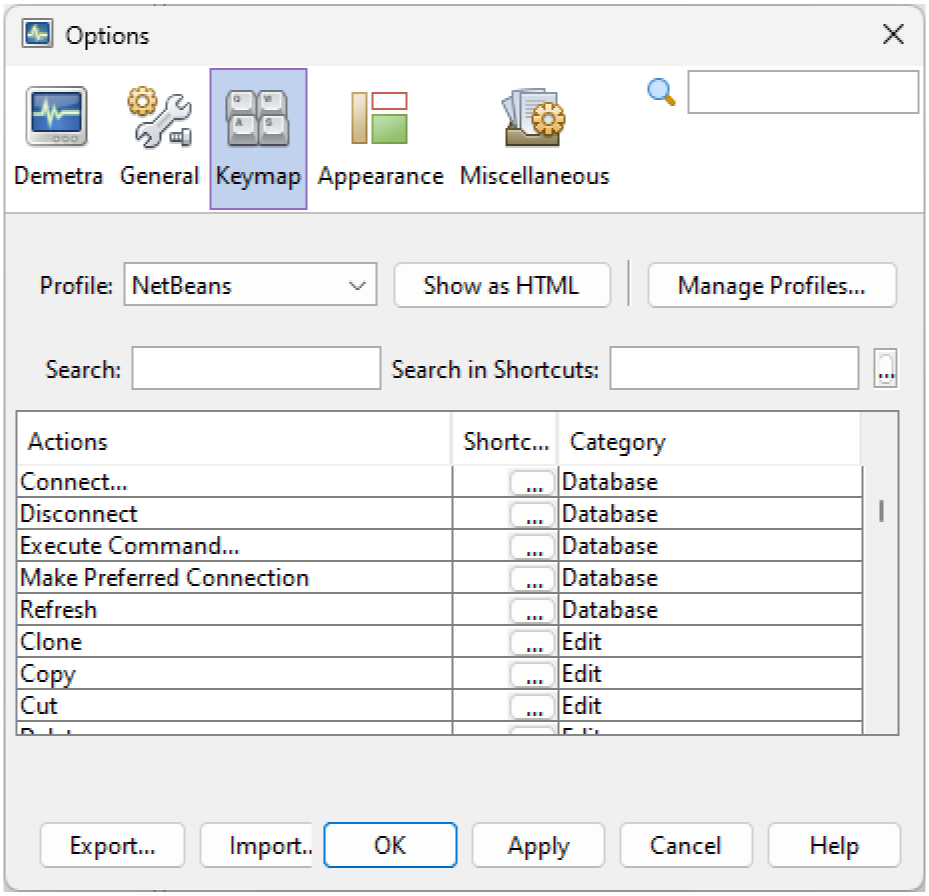


Figure 188: **The *Keymap* tab in v2**

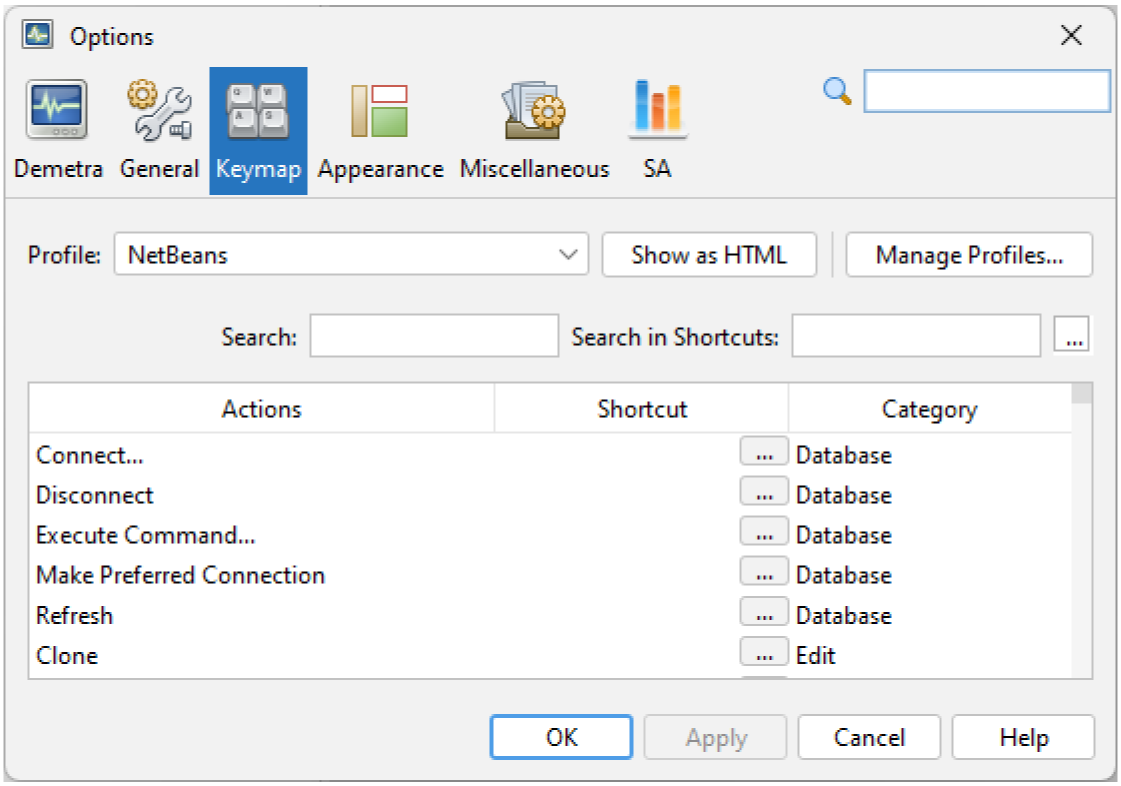
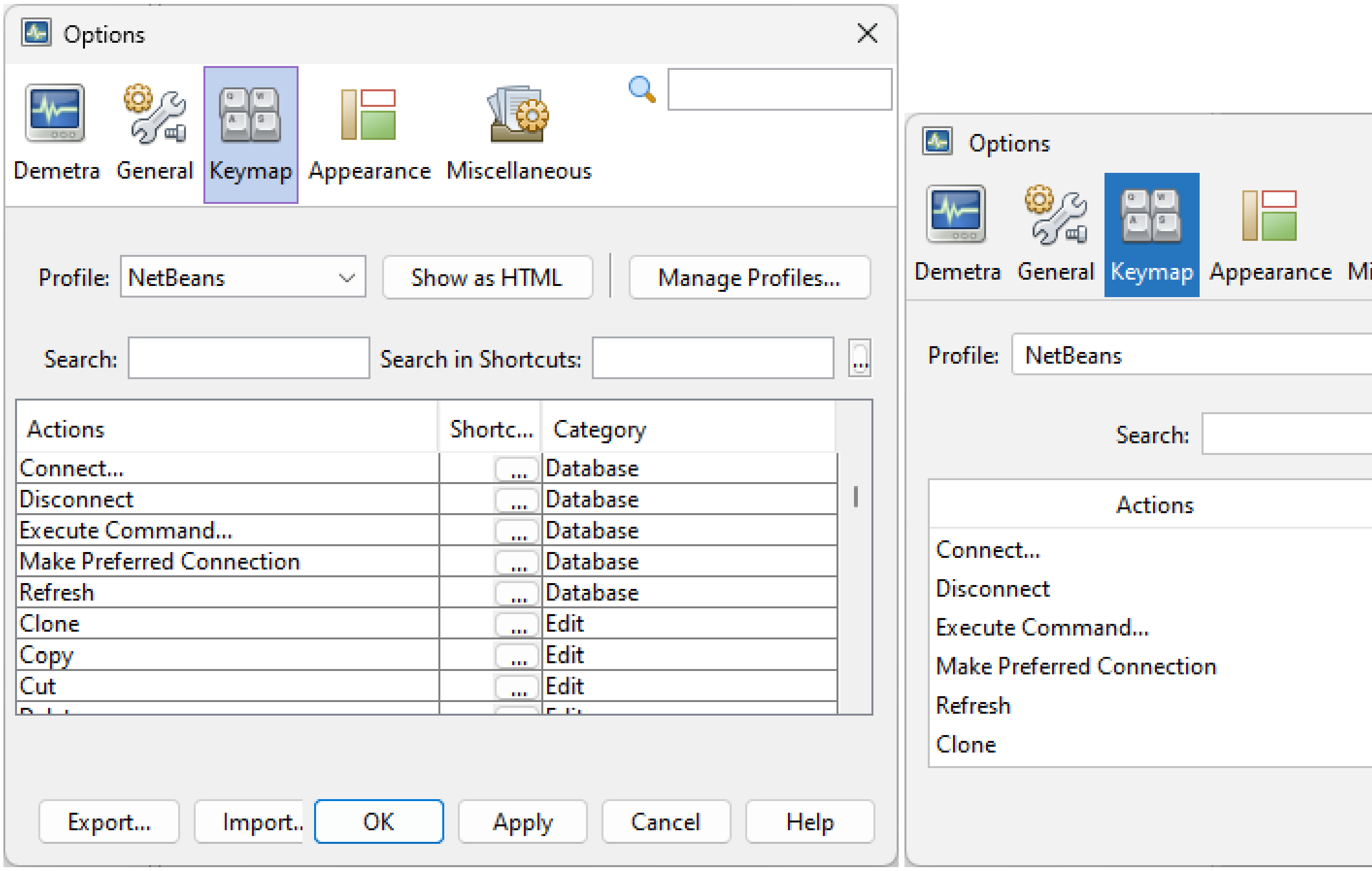


Figure 189: **The *Keymap* tab in v3**

0.0.0.2 v3



SA panel

The SA panel is only available in v3.

General tab

The General tab correspond to the Statistics tab from the Demetra panel in v2.

TramoSeats and X-13 tabs

The TramoSeats and X-13 tabs correspond to the settings for the quality measures and tests used in a diagnostic procedure in v2.

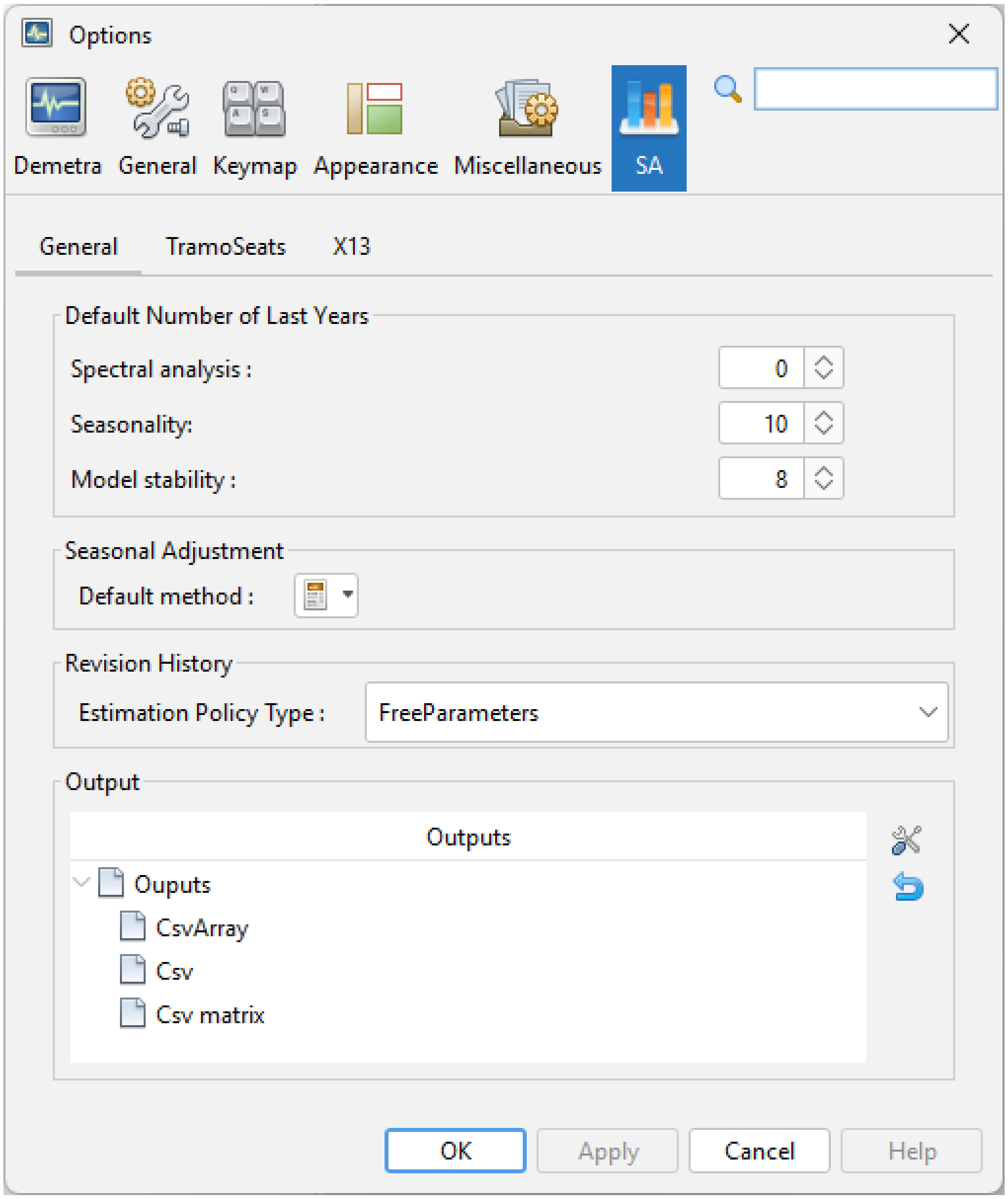


Figure 190: **SA panel**

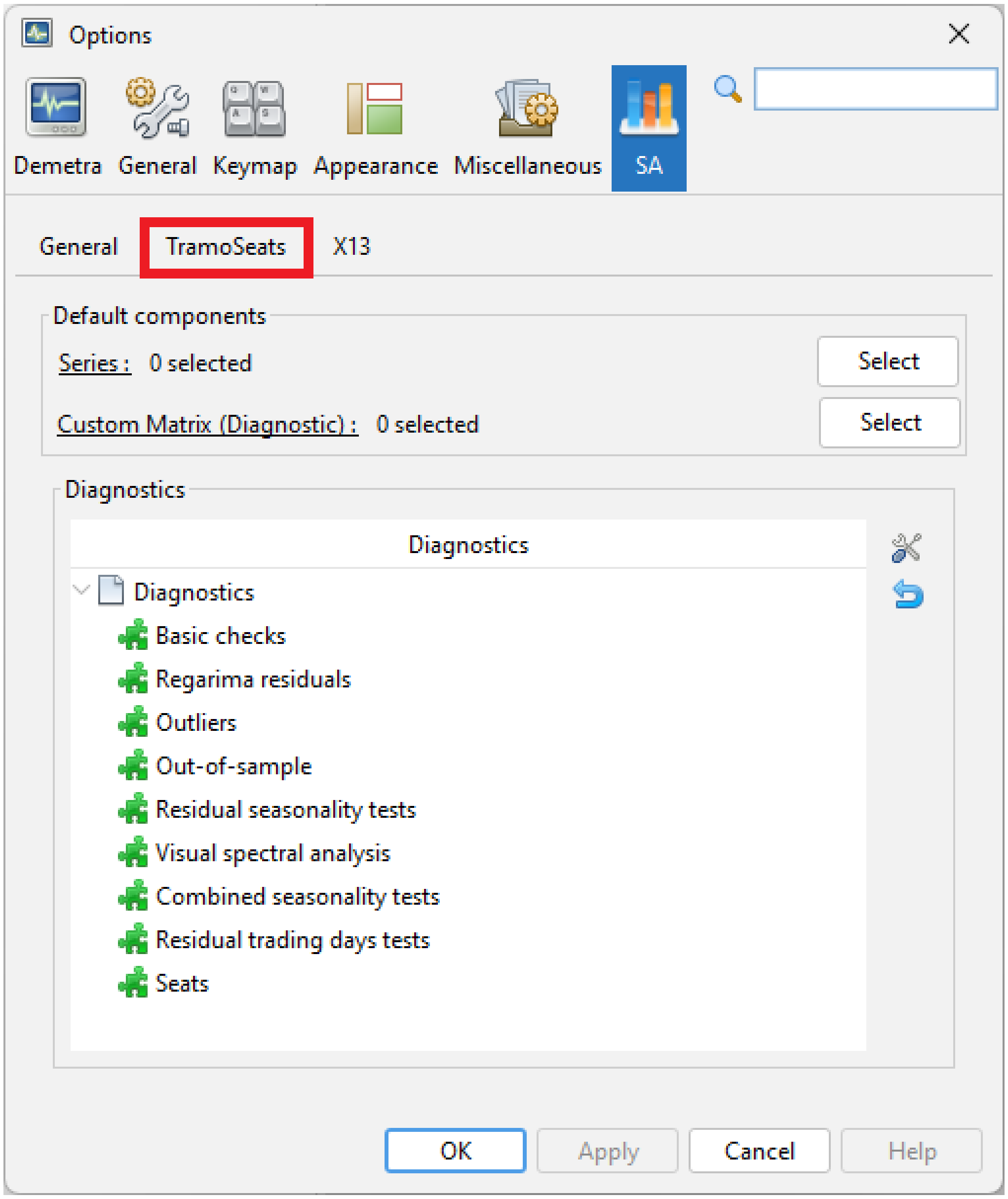


Figure 191: **Tramoseats tab**

Other panels

The *Appearance* and *Miscellaneous* panels are tabs automatically provided by the Netbeans platform. They are not used by JDemetra+.

GUI: data visualization and time series tools

In this chapter

This chapter describes time series generic tools available in the Graphical User Interface:

- data visualization
- spectral analysis tools
- aggregation
- differencing
- tests

Additional chapters related to GUI features, provide information on:

- Overview
- Specific Seasonal Adjustment and Modelling features
- Output: series, parameters and diagnostics

Data visualization

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

Several containers can be opened at the same time. Each of them may include multiple time series.

Chart plots the time series as a graph. This function opens an empty window. To display a given series drag and drop the series from the *Providers* window into the empty window. More than one series can be displayed on one graph. The chart is automatically rescaled after adding a new series.

The series to be viewed can be also dragged from the other windows (e.g. from the *Variables* window) or directly from the windows that display the results of the estimation procedure.

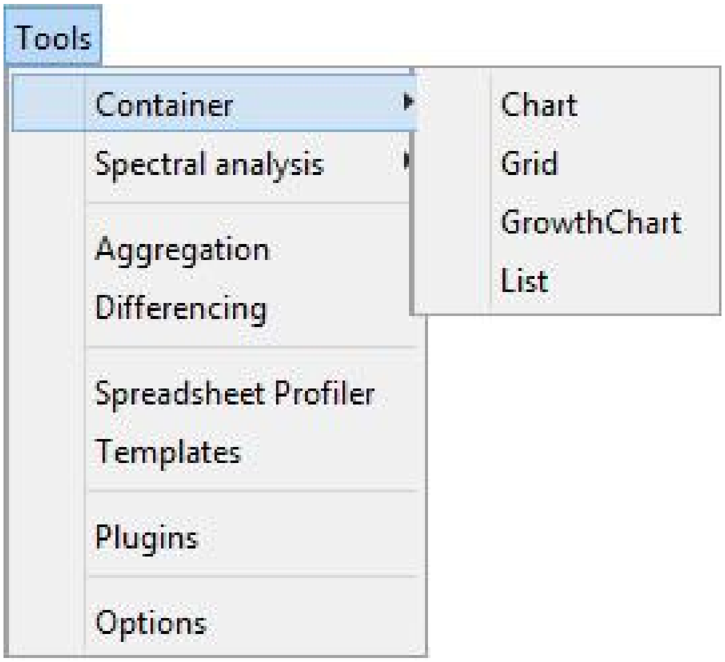


Figure 192: ** *Container* menu**

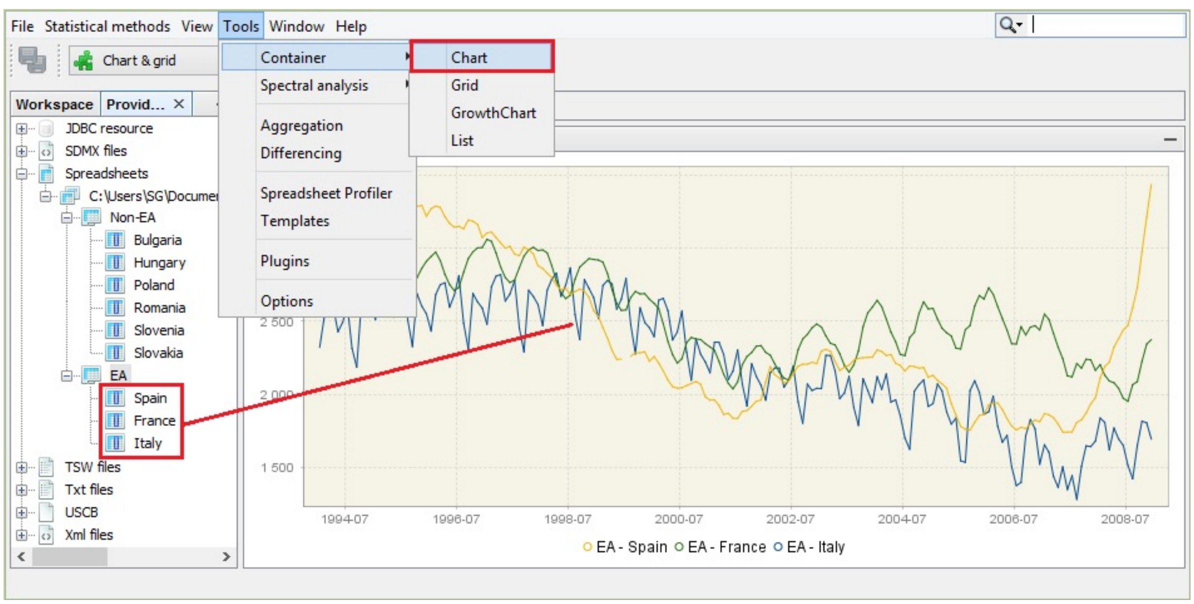


Figure 193: **Launching the *Chart* functionality**

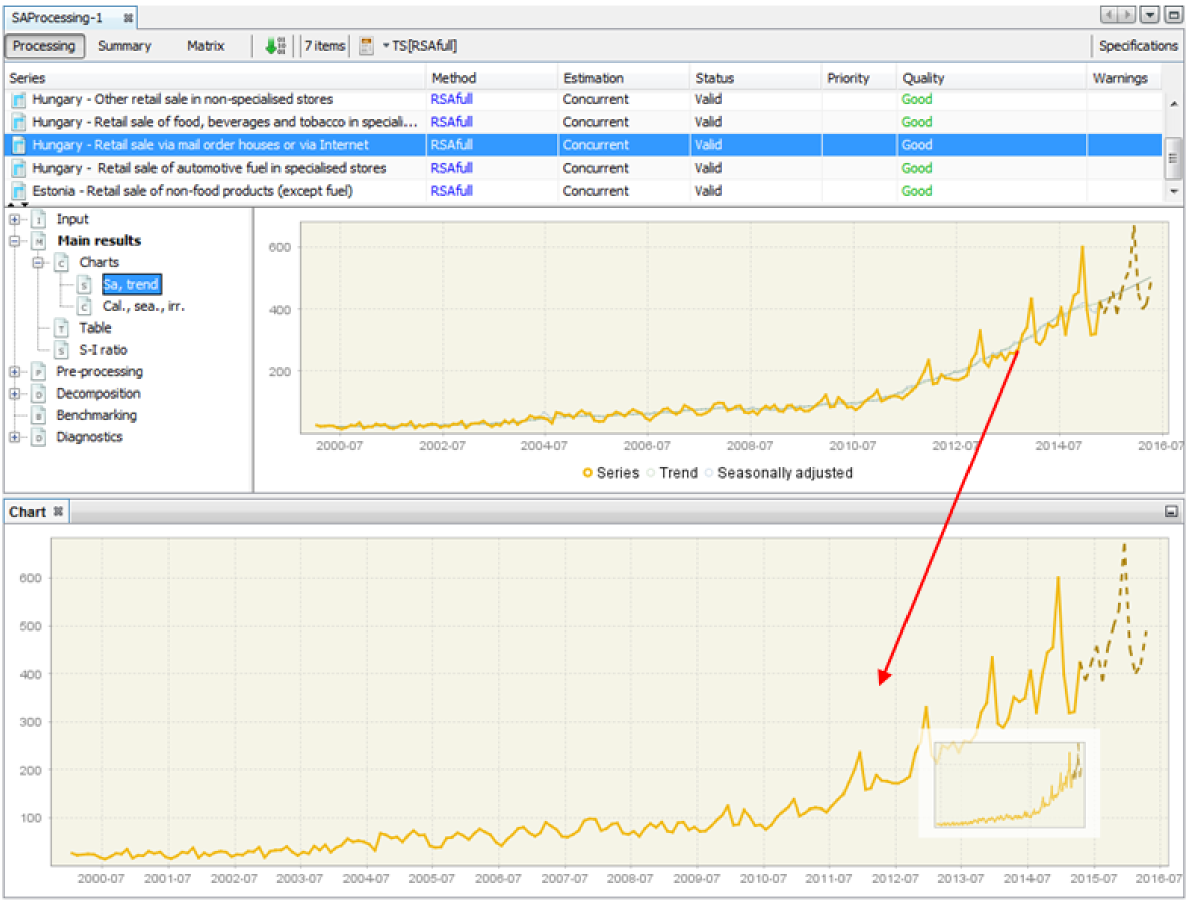


Figure 194: **Displaying the seasonally adjusted series on a separate chart**

To adjust the view of the chart and save it to a given location use the local menu, which is displayed after right-clicking on the chart. The explanation of the functions available for the local menu is given below.

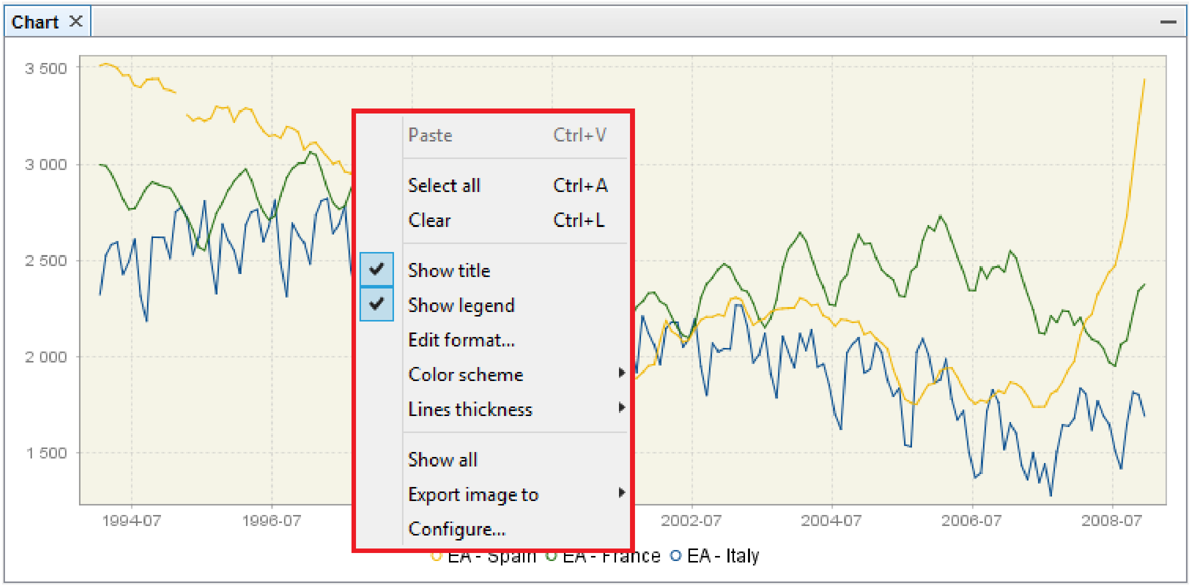


Figure 195: **Local menu basic options for the time series graph**

To display the time series value at a given date, hover over it with the cursor. Once the time series is selected by clicking on it with the right mouse button, the options dedicated to this series are available.

A list of possible actions includes:

- **Open** – opens selected time series in a new window that contains *Chart* and *Grid* panels.
- **Open with** – opens the time series in a separate window according to the user choice (*Chart & grid* or *Simple chart*). The *All ts views* option is not currently available.
- **Save** – saves the marked series in a spreadsheet file or in a text file.
- **Rename** – enables the user to change the time series name.
- **Freeze** – disables modifications of the chart.
- **Copy** – copies the series and allows it to be pasted to another application e.g. into Excel.
- **Paste** – pastes the time series previously marked.

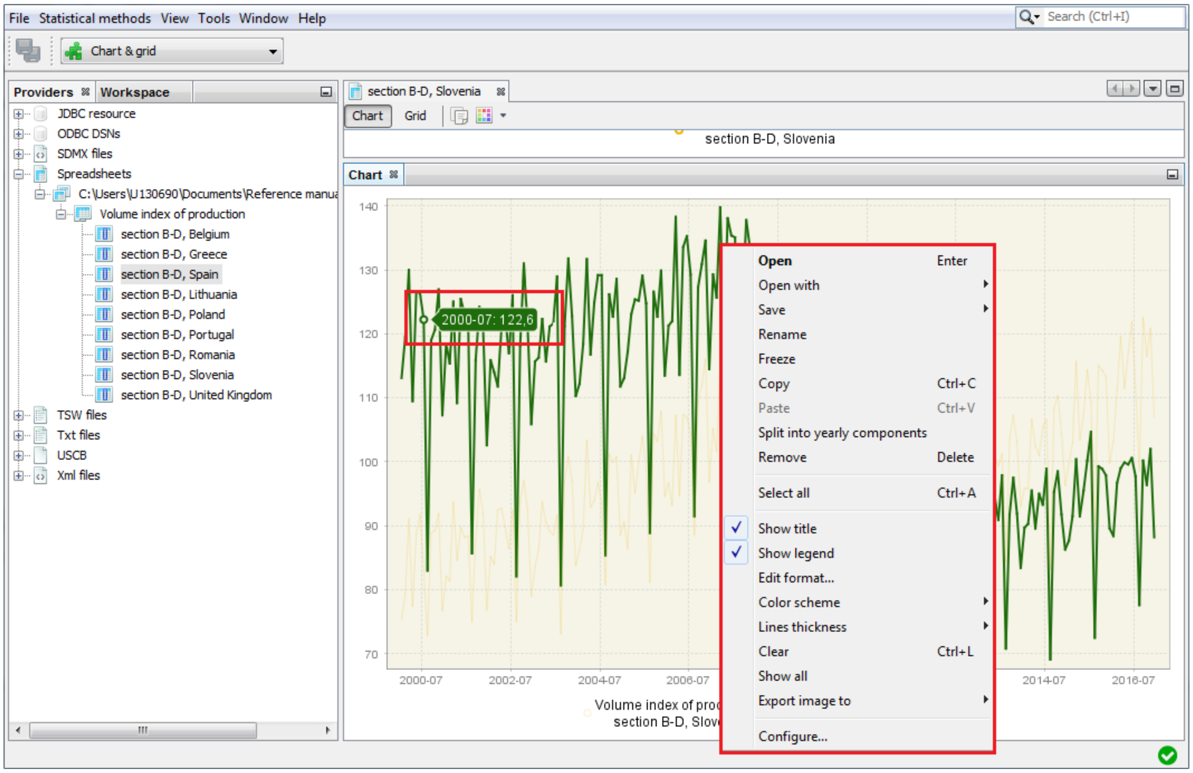


Figure 196: **Local menu options for chart**

- **Split into yearly components** – opens a window that presents the analysed series data split by year. This chart is useful to investigate the differences in time series values caused by the seasonal factors as it gives some information on the existence and size of the deterministic and stochastic seasonality in data.
- **Remove** – removes a time series from the chart.
- **Select all** – selects all the time series presented in the graph.
- **Show title** – option is not currently available.
- **Show legend** – displays the names of all the time series presented on the graph.
- **Edit format** – enables the user to change the data format.
- **Color scheme** – allows the colour scheme used in the graph to be changed.
- **Lines thickness** – allows the user to choose between thin and thick lines to be used for a graph.
- **Clear** – removes all the time series from the chart.
- **Show all** – this option is not currently available.
- **Export image to** – allows the graph to be sent to the printer and saved in the clipboard or as a file in a jpg format.
- **Configure** – enables the user to customize the chart and series display.

Grid enables the user to display the selected time series as a table. This function opens an empty window. To display a given series drag and drop the series from the *Providers* window into the empty window. More than one series can be displayed in one table.

To display options available for a given time series, left click on any time series' observation.

The options available in *Grid* are:

- **Transpose** – changes the orientation of the table from horizontal to vertical.
- **Reverse chronology** – displays the series from the last to the first observation.
- **Single time series** – removes from the table all time series apart from the selected one.
- **Use color scheme** – allows the series to be displayed in colour.
- **Show bars** – presents values in a table as horizontal bars.

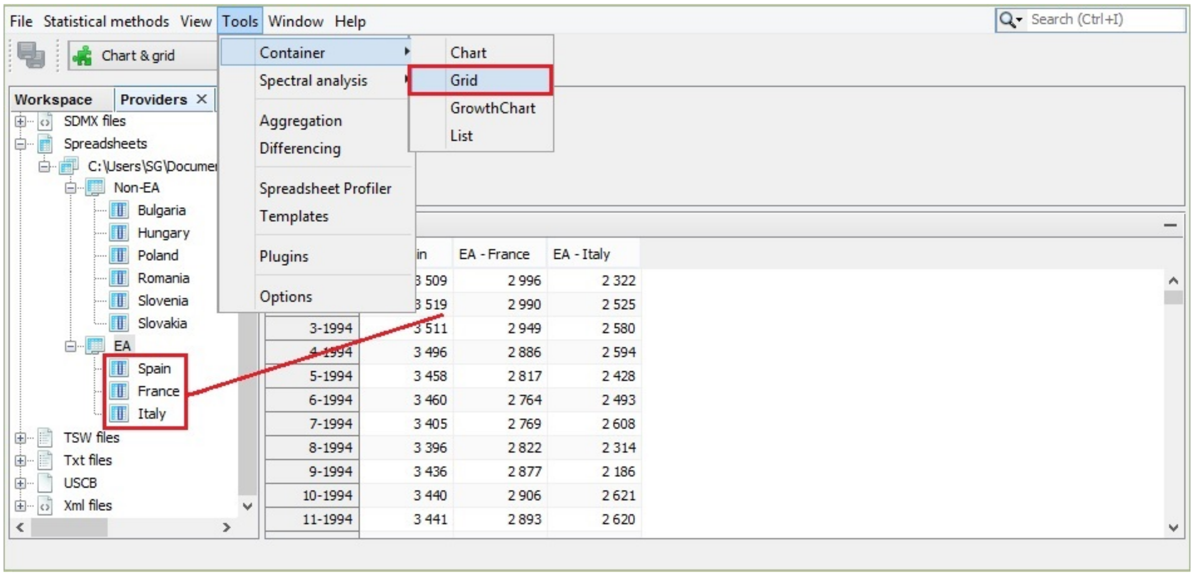


Figure 197: **Launching the *Grid* functionality**

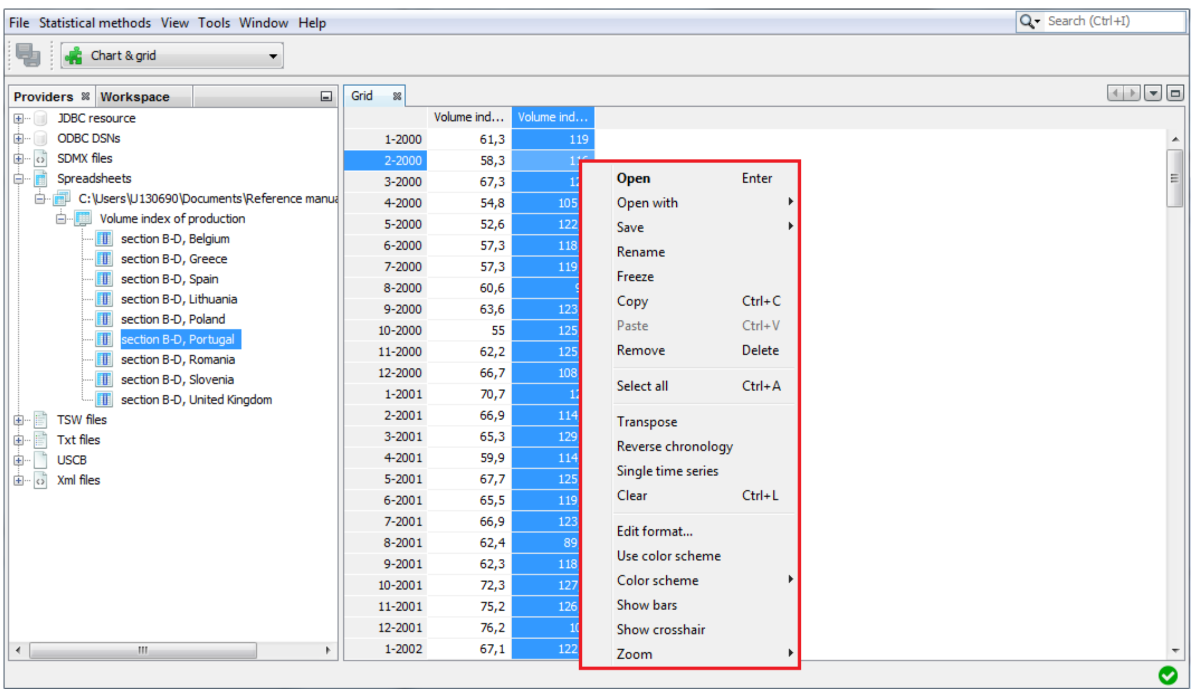


Figure 198: **Local menu options for the *Grid* view**

- **Show crosshair** – highlights an active cell.
- **Zoom** – option for modifying the chart size.

When none of the series is selected, the local menu offers a reduced list of options. The explanation of the other options can be found below in the '*Local menu options for chart*' figure in the *Container* section.

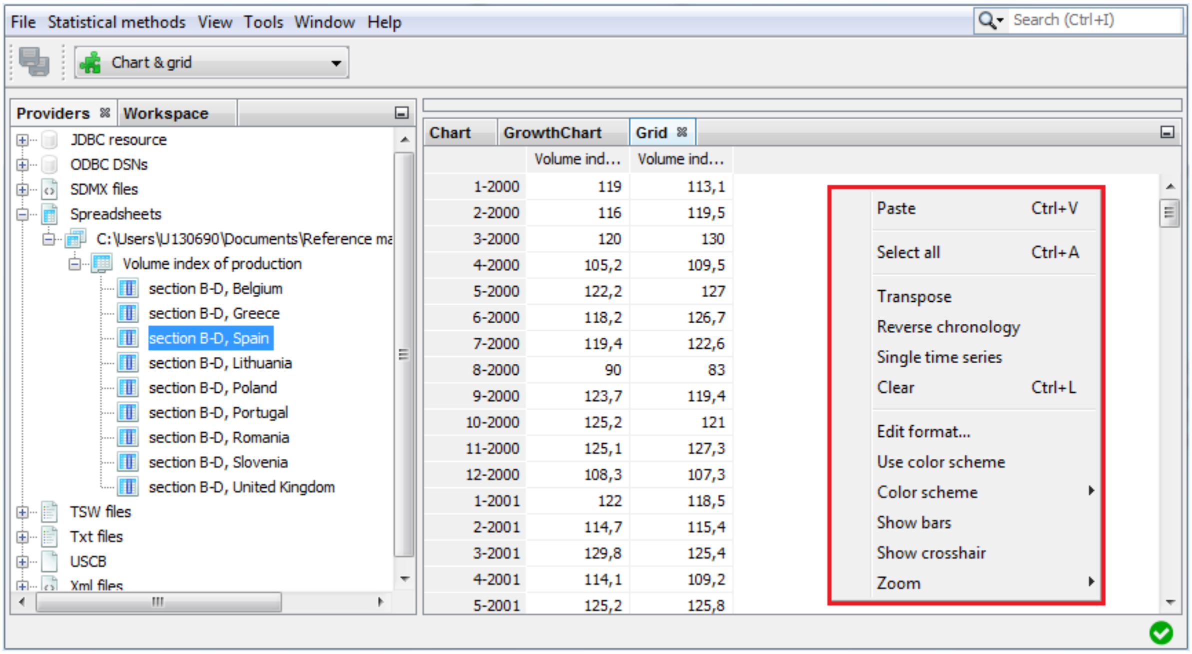


Figure 199: **A reduced list of options for *Grid***

The *Growth chart* tab opens an empty window. Once a given series is dropped into it, *Growth chart* presents the year-over-year or period-over-period growth rates for the selected time series. More than one series can be displayed in a table. The growth chart is automatically rescaled after adding a new series.

A left click displays a local menu with the available options. Those that are characteristic for the *Growth chart* are:

- **Kind** – displays m/m (or q/q) and y/y growth rates for all time series in the chart (previous period and previous year options respectively). By default, the period-over-period growth rates are shown.
- **Edit last year** – for clarity and readability purposes, only five of the last years of observations are shown by default. This setting can be adjusted in the *Options* section, if required.

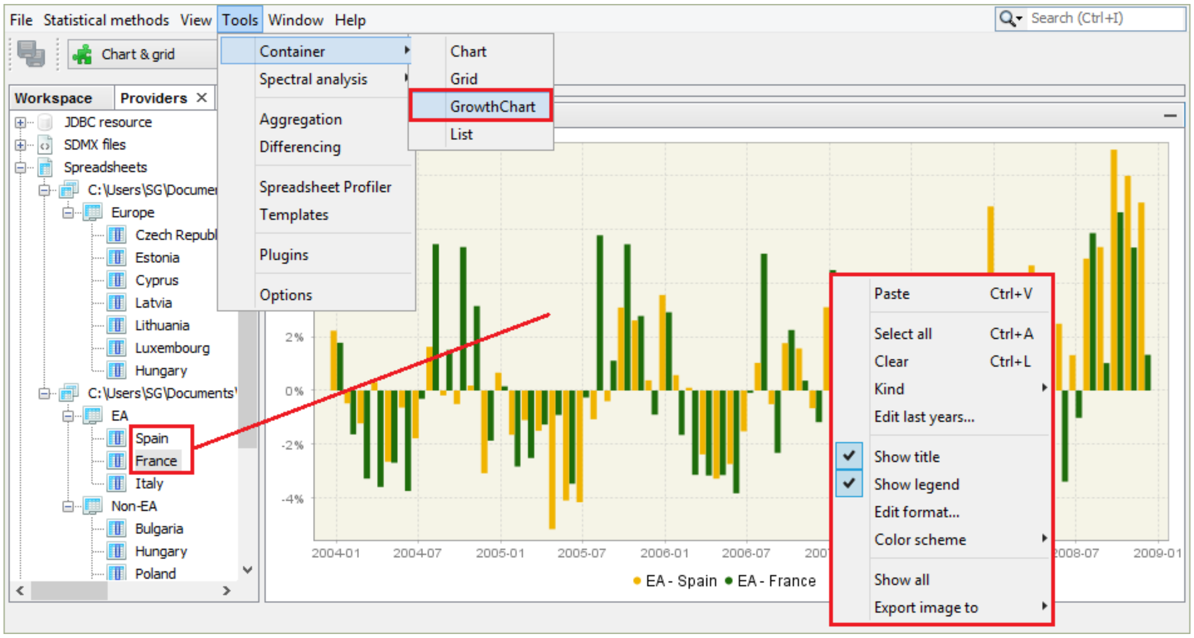


Figure 200: **The *Growth chart* view with a local menu**

The explanation of other options can be found below in the '*Local menu options for chart*' figure in the *Container* section.

The *List* tab provides basic information about the chosen time series, such as; the start and end date, the number of observations and a sketch of the data graph. This function opens an empty window. To display information, drag and drop the series from the *Providers* window into the *List* window. A right click displays the local menu with all available options. Apart from the standard options, the local menu for *List* enables marking the series that match the selected frequency (yearly, half-yearly, quarterly, monthly) by using the *Select by frequency* option. An explanation of other options can be found below in the '*Local menu options for chart*' figure in the *Container* section.

For a selected series a local menu offers an extended list of options. The explanation of the functions available for the local menu is given below in the '*Local menu options for chart*' figure in the *Container* section.

Spectral Analysis

Spectral graphs are available from: *Tools* → *Spectral analysis.

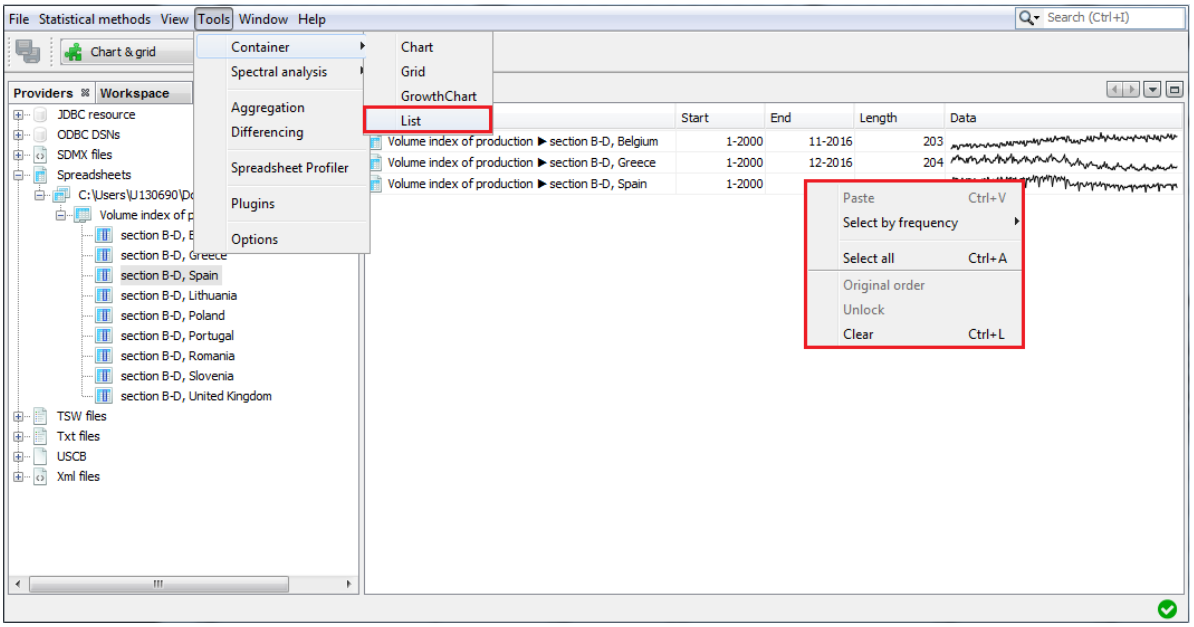


Figure 201: **A view of a list of series**

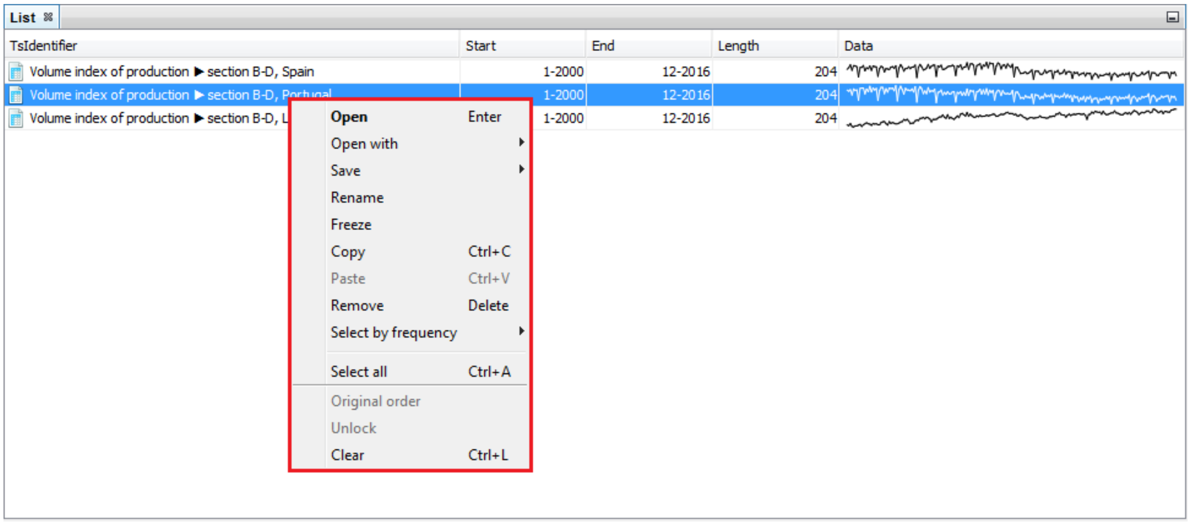


Figure 202: **Options available for a selected series from the list**

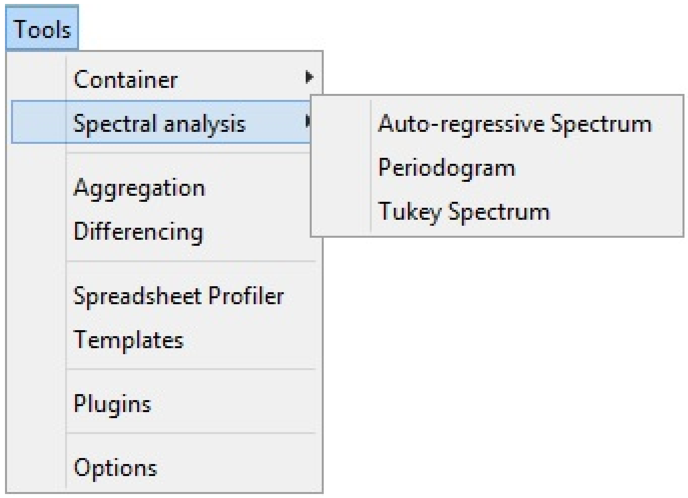


Figure 203: **Tools for spectral analysis**

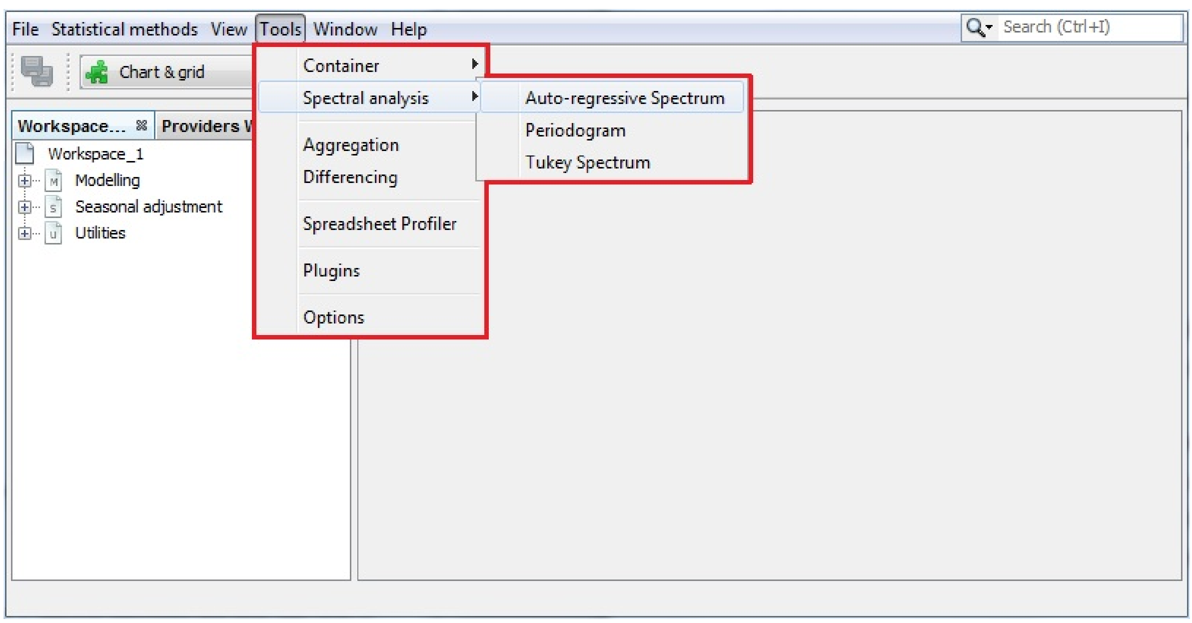


Figure 204: Tools for spectral analysis

Auto-regressive spectrum

When the first option is chosen JDemetra+ displays an empty *Auto-regressive spectrum* window. To start an analysis drag a single time series from the *Providers* window and drop it into the *Drop data here* area.

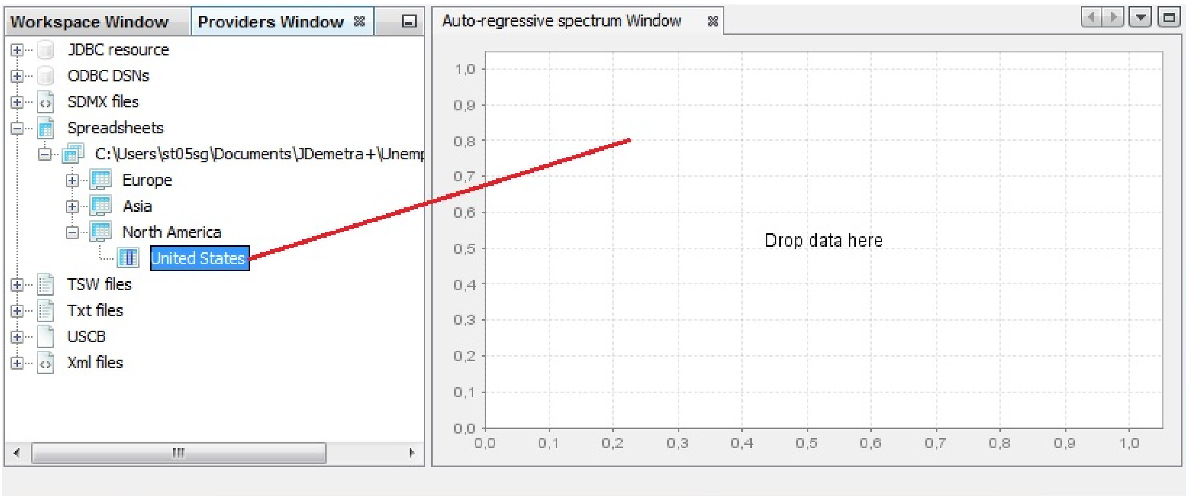


Figure 205: Launching an auto-regressive spectrum

When displaying an Auto-regressive spectrum the number of observations, data transformations and other options such as the specification of the frequency grid and the order of the autoregressive polynomial (30 by default) can be specified by opening the *Window* → *Properties* from the main menu.

The *Auto-regressive-Properties* window contains the following options:

- **Log** - log transformation of a time series;
- **Differencing**-transforms a data by calculating a regular (order 1,2..) or seasonal (order 4, 12, depending on the time series frequency) differences;
- **Differencing lag**-the number of lags that the program will use to take differences. For example, if *Differencing lag=3* then the differencing filter does not apply to the first lag (default) but to the third lag.
- **Last years**-a number of years at the end of the time series taken to produce autoregressive spectrum. By default, it is 0, which means that the whole time series is considered.
- **Auto-regressive polynomial order**-the number of lags in the AR model that is used to estimate the spectral density. By default, the order of the autoregressive polynomial is set to 30 lags.

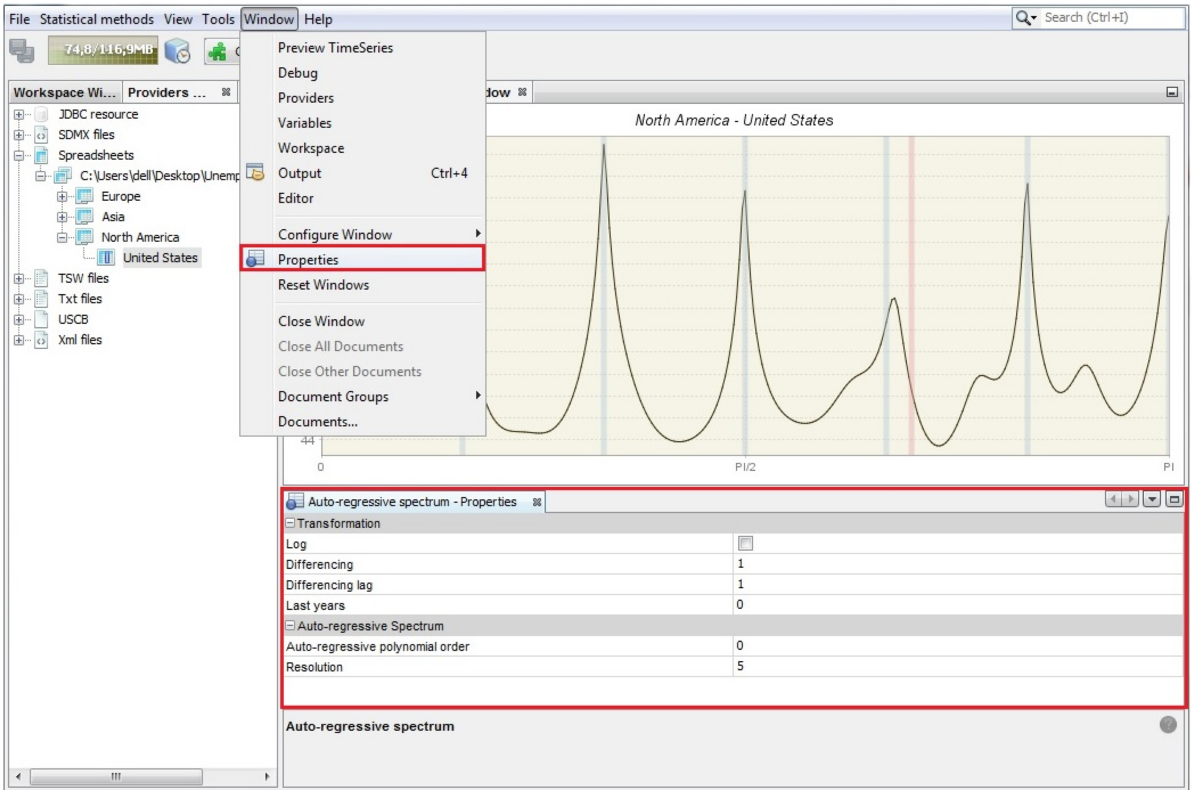


Figure 206: **Auto-regressive spectrum's properties**

- **Resolution**-the value 1 plots the spectral density estimate for the frequencies $\omega_j = \frac{2\pi j}{n}$, where $n \in (-\pi;\pi)$ is the size of the sample used to estimate the AR model. Increasing this value, which is set to 5 by default, will increase the precision of this grid.

Periodogram

Choose *Tools →Spectral analysis → Periodogram* and drag and drop a series from the *Providers* window to the empty *Periodogram* window.

`![Launching a periodogram](All_images/image5_342.jpeg)`

The sample size and data transformations can be specified by opening the *Window → Properties*, in the main menu. The *Periodogram- Properties* window contains the following options:

- **Log** - log transformation of a time series;
- **Differencing**-transforms the data by calculating regular (order 1,2..) or seasonal (order 4, 12, depending on the time series frequency) differences;
- **Differencing lag**-the number of lags that you will use to take differences. For example, if *Differencing lag=3* then the differencing filter does not apply to the first lag (default) but to the third lag.
- **Last years**-the number of years at the end of the time series taken to produce periodogram. By default it is 0, which means that the whole time series is considered.

Tukey spectrum

Choose *Tools → Spectral analysis → Tukey spectrum* and drag and drop a single series from the *Providers* window to the empty *Periodogram* window.

`![Launching a Tukey spectrum](All_images//image8_342.jpeg)`

The Tukey spectrum estimates the spectral density by smoothing the periodogram.

`![An example of a Tukey spectrum](All_images/image9_342.jpeg)`

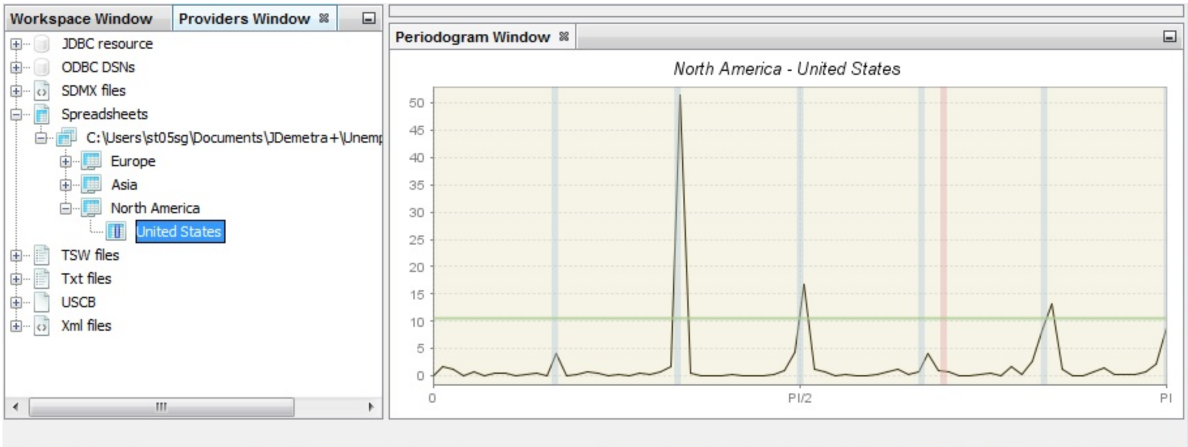


Figure 207: Example of a periodogram

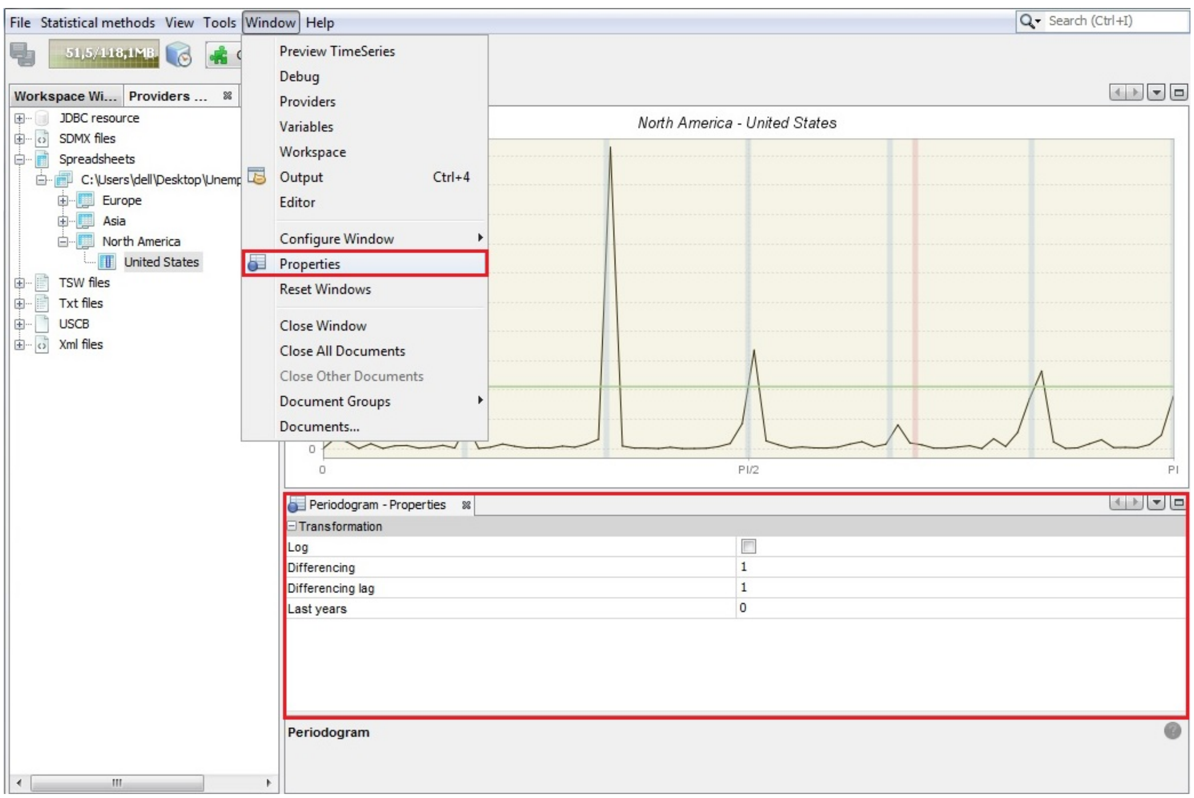


Figure 208: Periodogram's properties

Options for the Tuckey window can be specified by opening the *Window → Properties* from the main menu. The *Periodogram- Properties* window contains the following options:

- **Log** - log transformation of a time series.
- **Differencing** -transforms the data by calculating regular (order 1, 2..) or seasonal (order 4, 12, depending on the time series frequency) differences.
- **Differencing lag**-the number of lags that you will use to take differences. For example, if *Differencing lag=3* then the differencing filter does not apply to the first lag (default) but to the third lag.
- **Taper part**–parameter larger than 0 and smaller or equal to one that shapes the curvature of the smoothing function that is applied to the auto-covariance function.
- **Window length**–the size of the window that is used to smooth the auto-covariance function. A value of zero includes the whole series.
- **Window type**–it refers to the weighting scheme that it is used to smooth the auto-covariance function. Available windows types are: *Square*, *Welch*, *Tukey*, *Barlett*, *Hamming* and *Parzen*.

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, in the same way as in the *List* functionality. *Aggregation* opens an empty window. To sum the selected series, drag and drop them from the *Providers* window into the *Aggregation* window. Right click displays the local menu with the available options. The content of the local menu depends on the panel chosen (the panel on the left that contains the list of the series and the panel on the right that presents the graph of an aggregate). The local menu for the list of series offers the option *Select by frequency*, which marks all the series on the list that are yearly, half-yearly, quarterly or monthly (depending on the user's choice). The explanation of the other options can be found below in the '*Local menu options for chart*' figure in the *Container* section. The local menu for the panel on the left offers functionalities that are analogous to the ones that are available for the *List* functionalities, while the options available for the local menu in the panel on the left are the same as the ones available in *Chart* (see *Container*).

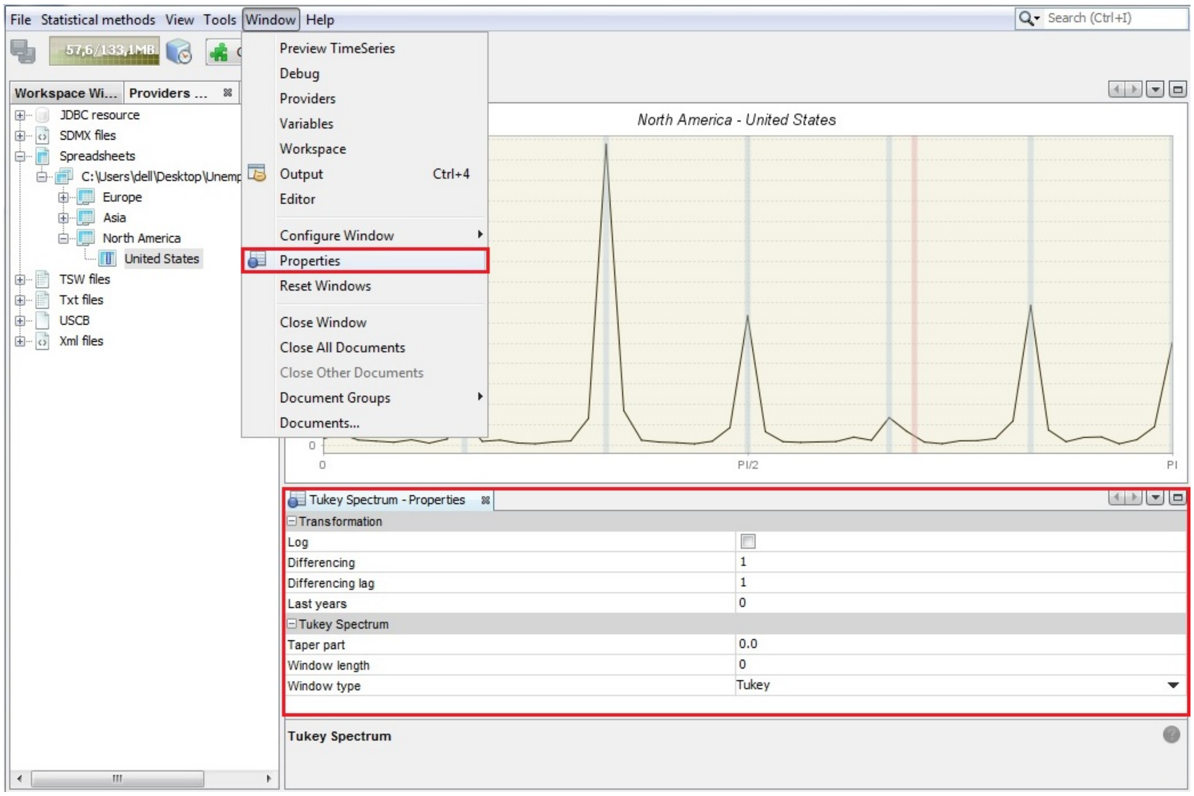


Figure 209: Tukey spectrum’s properties

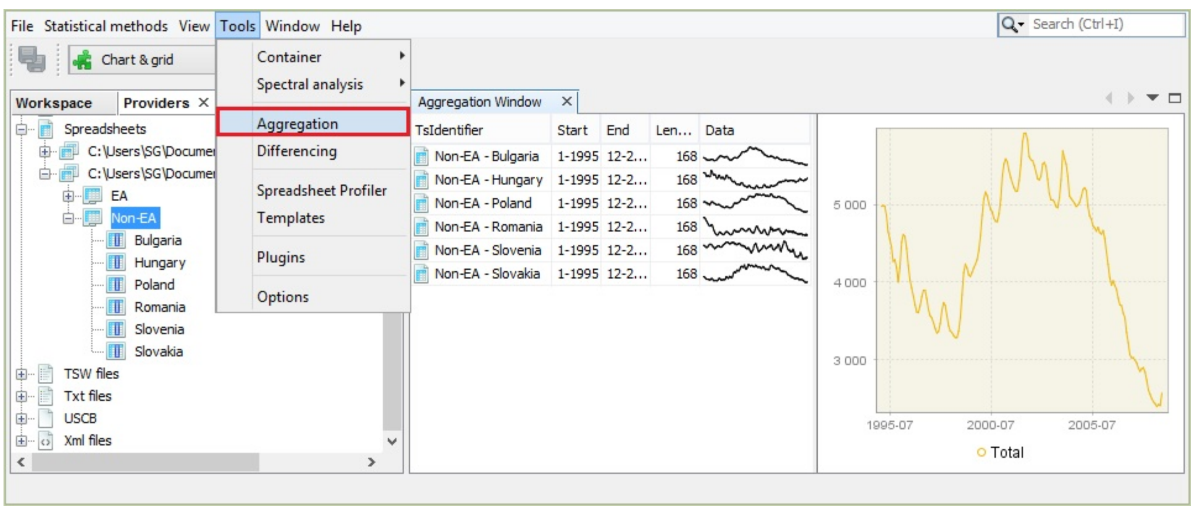


Figure 210: **The *Aggregation* tool**

Differencing

The *Differencing* window displays the first regular differences for the selected time series together with the corresponding periodogram and the PACF function. By default, the window presents the results for non-seasonally and seasonally differenced series (($d = 1, D = 1$)). These settings can be changed through the *Properties* window (*Tools* → *Properties*). A description of a periodogram and the PACF function can be found here.

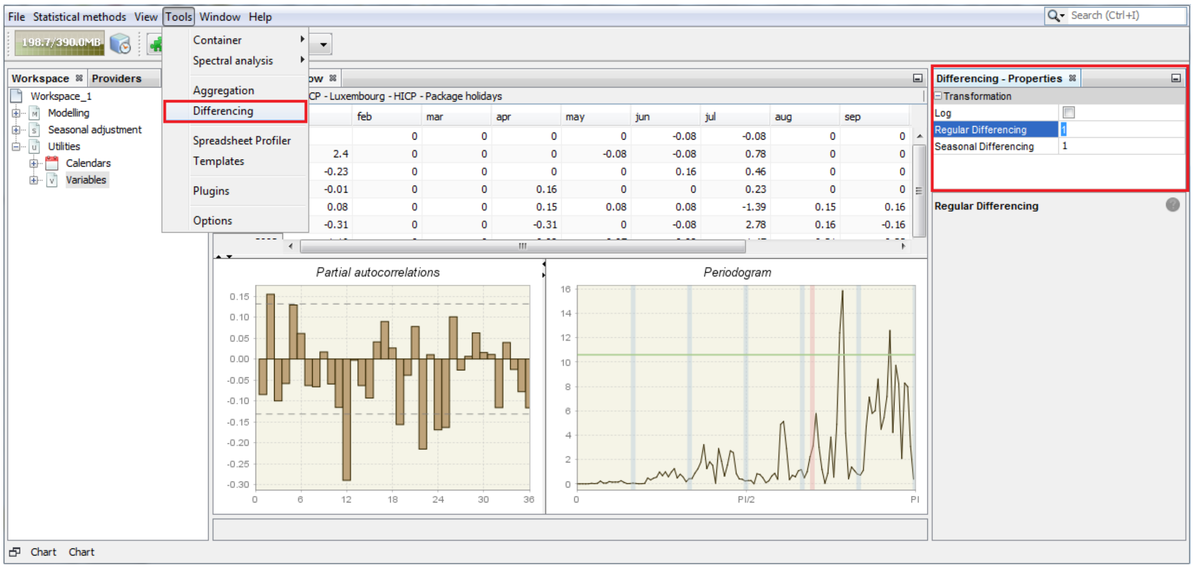


Figure 211: **The properties of the *Differencing* tool**

Typical results are shown below. The bottom left graph presents the partial autocorrelation coefficients (vertical bars) and the confidence intervals. The right-click local menu offers several functionalities for a differenced series. An explanation of the available options can be found below in the "*Local menu options for chart*" figure in the *Container* section.

For the *Partial autocorrelation* and the *Periodogram* panels the right-button menu offers "a copy series" option that allows data to be exported to another application and a graph to be printed and saved to a clipboard or as a .jpg file.

Tests

Here we describe describe the GUI access to and display of tests. The underlying methods are detailed in this chapter

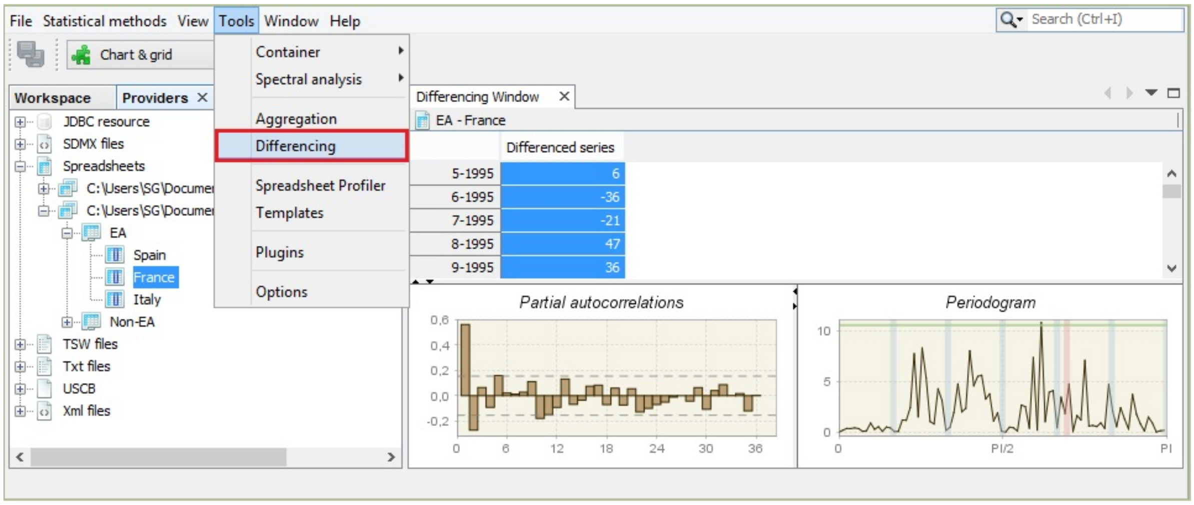


Figure 212: The *Differencing tool

Seasonality Tests

QS test

The test can be applied directly to any series by selecting the option *Statistical Methods » Seasonal Adjustment » Tools » Seasonality Tests*. This is an example of how results are displayed for the case of a monthly series:

1. Tests on autocorrelations at seasonal lags

Seasonality present

ac(12)=0.8238
ac(24)=0.7006

Distribution: Chi2 with 2 degrees of freedom
Value: 258.5028
PValue: 0.0000

Figure 213: qs

It is also visible in Main results panel and in Diagnostics node when an SA processing has been carried out.

Friedman test for stable seasonality

The test can be applied directly to any series by selecting the option *Statistical Methods » Seasonal Adjustment » Tools » Seasonality Tests*. This is an example of how results are displayed for the case of a monthly series:

2. Non parametric (Friedman) test
Based on the rank of the observations in each year

Seasonality present
Distribution: Chi2 with 11 degrees of freedom
Value: 142.8654
PValue: 0.0000

Figure 214: friedman

If the null hypothesis of no stable seasonality is rejected at the 1% significance level, then the series is considered to be seasonal and the outcome of the test is displayed in green.

It is also visible in Diagnostics node when an SA processing has been carried out.

Identification of spectral peaks

0.0.0.0.1 * In a Tukey spectrum

The test can be applied directly to any series by selecting the option *Statistical Methods » Seasonal Adjustment » Tools » Seasonality Tests*. This is an example of how results are displayed for the case of a monthly series:

4. Identification of seasonal peaks in a Tukey periodogram and in an auto-regressive spectrum

Seasonality present

T or t for Tukey periodogram, A or a for auto-regressive spectrum; 'T' or 'A' for very signficant peaks, 't' or 'a' for signficant peaks, '_' otherwise

AT.AT.AT.AT.AT.A-

Figure 215: tktest

JDemetra+ considers critical values for $\alpha = 1\%$ (code “T”) and $\alpha = 5\%$ (code “t”) at each one of the seasonal frequencies represented in the table below, e.g. frequencies $\frac{\pi}{6}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$, $\frac{2\pi}{3}$ and $\frac{5\pi}{6}$ corresponding to 1, 2, 3, 4, 5 and 6 cycles per year in this example, since we are dealing with monthly data.

0.0.0.0.2 * In an AR Spectrum

The test can be applied directly to any series by selecting the option *Statistical Methods » Seasonal Adjustment » Tools » Seasonality Tests*. This is an example of how results are displayed for the case of a monthly series:

4. Identification of seasonal peaks in a Tukey periodogram and in an auto-regressive spectrum

Seasonality present

T or t for Tukey periodogram, A or a for auto-regressive spectrum; 'T' or 'A' for very signficant peaks, 't' or 'a' for signficant peaks, '_' otherwise

AT.AT.AT.AT.AT.A-

Figure 216: artest

JDemetra+ considers critical values for $\alpha = 1\%$ (code "A") and $\alpha = 5\%$ (code "a") at each one of the seasonal frequencies represented in the table below, e.g. frequencies $\frac{\pi}{6}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$, $\frac{2\pi}{3}$ and $\frac{5\pi}{6}$ corresponding to 1, 2, 3, 4, 5 and 6 cycles per year in this example, since we are dealing with monthly data.

0.0.0.0.3 * In a Periodogram

The test can be applied directly to any series by selecting the option *Statistical Methods » Seasonal Adjustment » Tools » Seasonality Tests*.

5. Periodogram
Test on the sum of the values of a periodogram at seasonal frequencies

Seasonality present

Distribution: F with 11 degrees of freedom in the nominator and 180 degrees of freedom in the denominator
Value: 45.1387
PValue: 0.0000

Figure 217: periodtest

GUI: SA and Modelling Features

In this chapter

This chapter covers specific Seasonal Adjustment and Modelling features. Modelling refers to reg-ARIMA or Tramo when used stand alone and not as the first (pre-adjustment) step in seasonal adjustment.

Note that the menu and window structure as well as options and results are almost identical in both cases.

Additional chapters related to GUI features, provide information on:

- Overview
- Data visualization and generic time series tools
- Output: series, parameters and diagnostics

Currently, this chapter is widely incomplete, additional content will be uploaded in the coming weeks.

How to create a Workspace?

When you open a new session of the GUI JDemetra, a new workspace called *Workspace_1* is created.

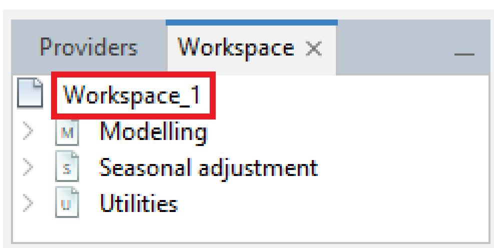


Figure 218: **Initial Workspace_1**

To create a Workspace:

1. Click on the File button
2. Then, click on New Workspace

A new Workspace called *Workspace_2* is created.

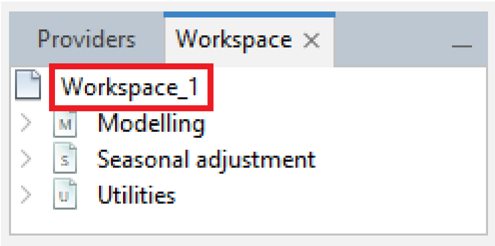


Figure 219: **Initial Workspace_1**

Workspace Window

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

Content of *Workspace* window, divided into three sections:

- 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).

Modelling

In the modelling section, you can create documents based on the reg-ARIMA or Tramo only specification (which are the same as the ones used for the pre-treatment phase of seasonal adjustment with X-13-ARIMA (Reg-ARIMA) and Tramo-Seats (Tramo) and thus described in the SA chapter).

You can also use the extended airline algorithm with the plugin jdplus-highfreq-desktop-plugin to do the pre-treatment of high frequency data.

The specifications and output of the modelling procedure are displayed in the *Workspace* window.

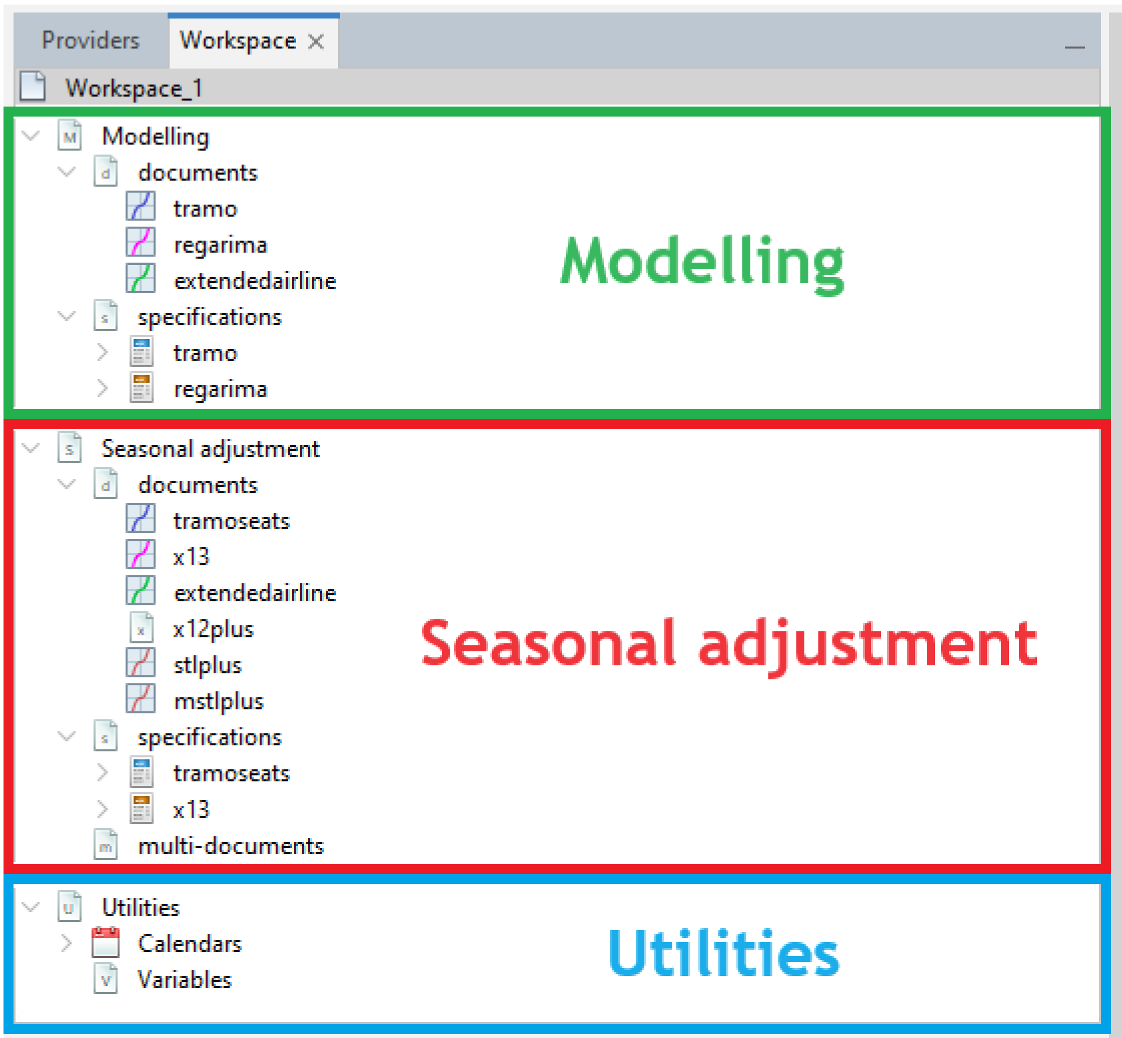


Figure 220: **The *Workspace* window**

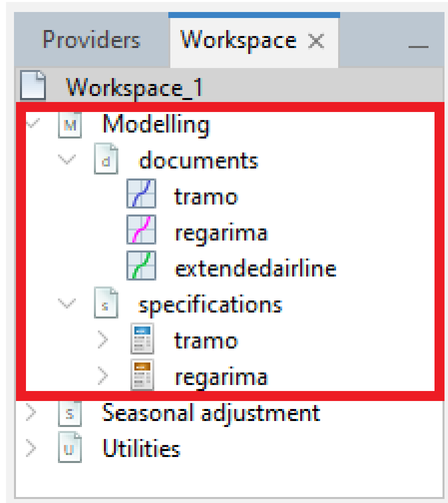


Figure 221: **The *Workspace* window with the nodes for the modelling procedure marked**

Seasonal adjustment

This window allows to set up and launch a seasonal adjustment process.

Some plugins give you additional features:

- jdplus-highfreq-desktop-plugin give you access to the extended airline algorithm
- jdplus-x12plus-desktop give you access to the x12plus algorithm
- jdplus-sts-desktop and jdplus-stl-desktop give you access to the stlplus and mstlplus algorithm

This section is divided into two parts:

- Specifications, which presents parameters of the seasonal adjustment procedure.
- Output, which explains a typical output produced by the seasonal adjustment procedure.

The specifications and output for the seasonal adjustment procedure are displayed in the *Workspace* window under the *Seasonal adjustment* item.

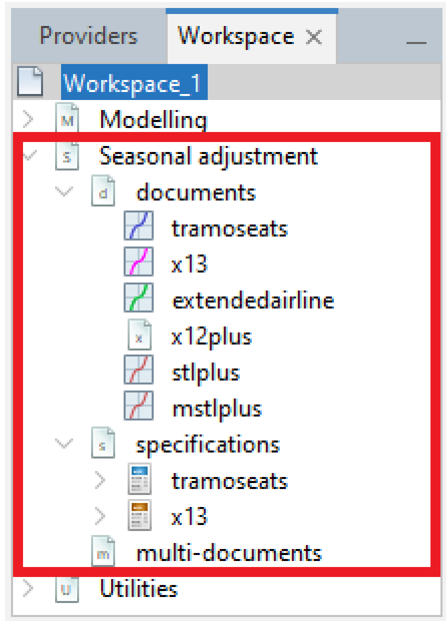


Figure 222: **The *Workspace* window with the nodes for the seasonal adjustment procedure marked**

Results panel

The blank zone in the figure above (on the right of the view) is the location where JDemetra+ displays various windows. More than one window can be displayed at the same time. Windows can overlap with each other with the foremost window being the one in focus or active. The active window has a darkened title bar. The windows in the results panel can be arranged in many different ways, depending on the user's needs. The example below shows one of the possible views of this panel. The results of the user's analysis are displayed in an accompanying window. The picture below shows two panels – a window containing seasonal adjustment results (upper panel) and another one containing an autoregressive spectrum (lower panel).

Statistical Methods Menu

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

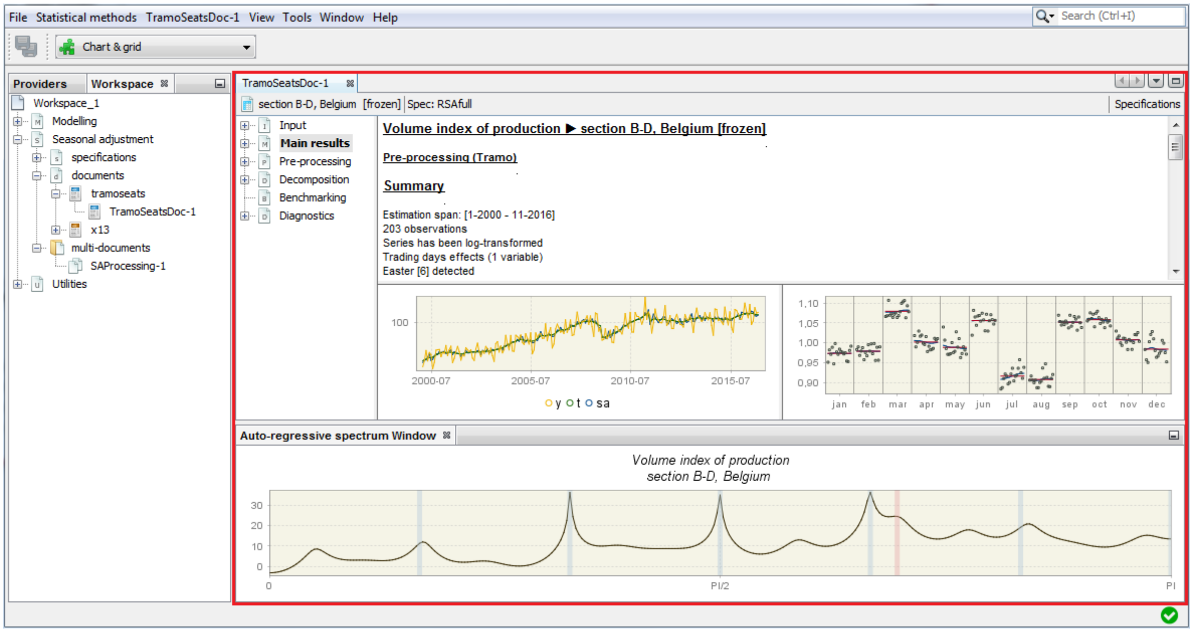


Figure 223: **The *Results* panel filled with two windows**

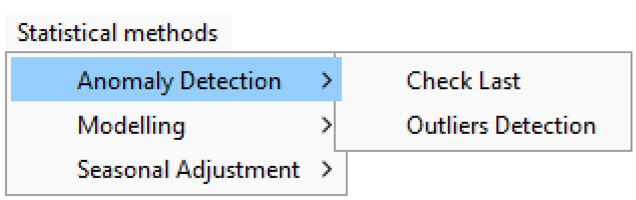


Figure 224: **The *Anomaly detection* tab.**

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

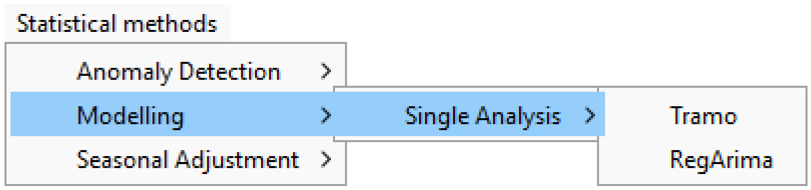


Figure 225: **The *Modelling* tab.**

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

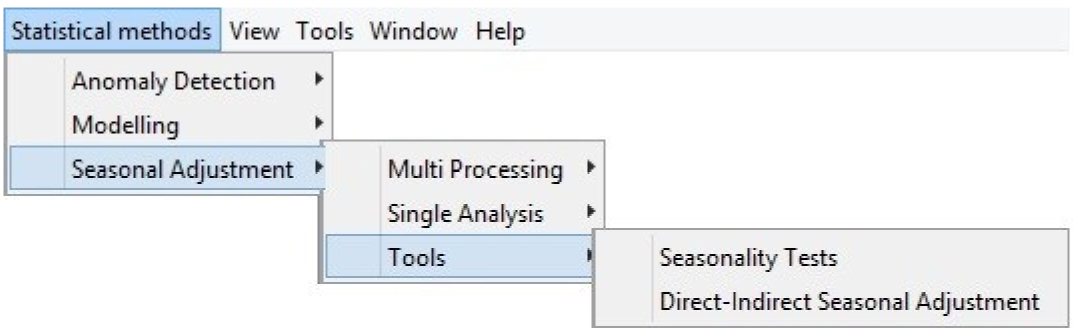


Figure 226: **The *Seasonal adjustment* tab.**

- Tools – provides access to:
  - seasonality tests
  - and to Direct-Indirect Seasonal adjustment tools

Direct-Indirect method

Context

Economic time series are often computed and reported according to a certain classification or a breakdown.

For example, in National Accounts total consumption expenditures are a sum of individual consumption expenditures and General Government & NPISHs consumption expenditures. Therefore, the seasonally adjusted aggregates can be computed either by aggregating the seasonally adjusted components (indirect adjustment) or adjusting the aggregate and the components independently (direct adjustment). The point is that these two strategies result in different seasonally adjusted aggregates. As neither theoretical nor empirical evidence uniformly favours one approach over the other, the choice of the seasonal adjustment strategy concerning aggregated series depends on the user. Guidance in this field is given in the Eurostat guidelines on seasonal adjustment (**guidelines2015?**).

In GUI v2

JDemetra+ offers a *Direct–Indirect Seasonal Adjustment* functionality that facilitates the comparison of the results from these two strategies, which is launched from the main menu.

> 🔥 Only in v2
>
> This Direct-Indirect method is not implemented in v3.

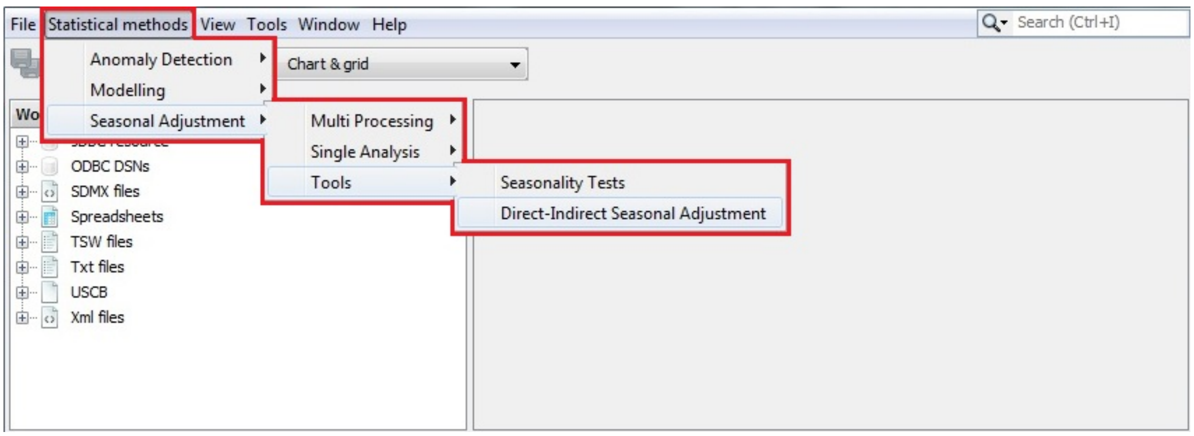


Figure 227: **The *Direct-Indirect Seasonal Adjustment* tool**

To start the analysis drag and drop time series to the top-left panel. The panel on the right presents the sum of selected series.

By going to the main menu and clicking on Window → Properties, one can specify benchmarking options for direct-indirect comparison.

Be aware that the properties window displays the properties of an active item. Therefore, first click on the time series graph in the picture below and then activate the *Properties* window.

By default, the pre-defined Tramo-Seats specification is used (RSAfull) for seasonal adjustment of a dataset. To change it, click on the button marked in the picture

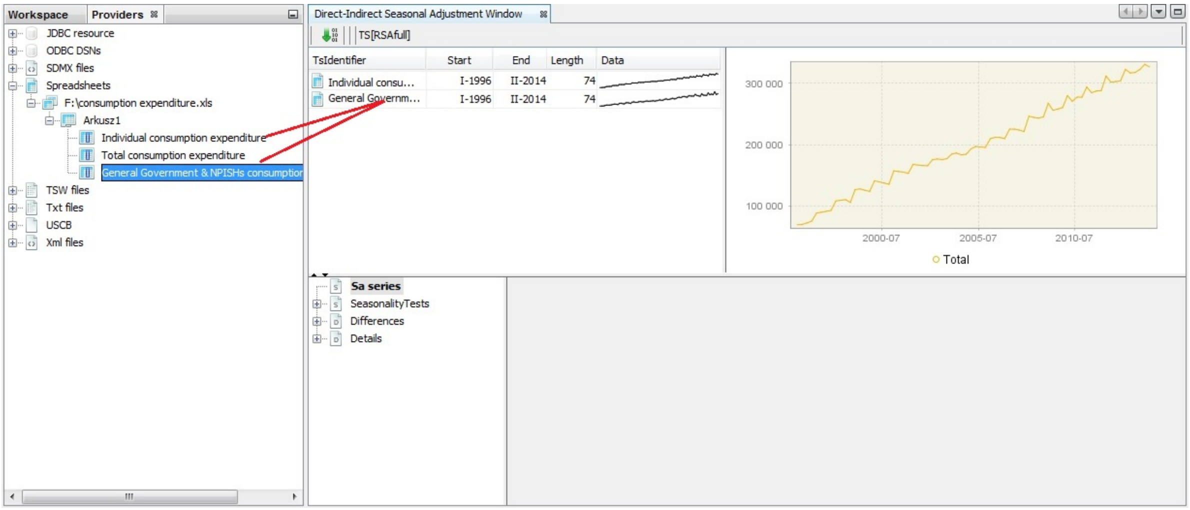


Figure 228: **Choosing series for an analysis**

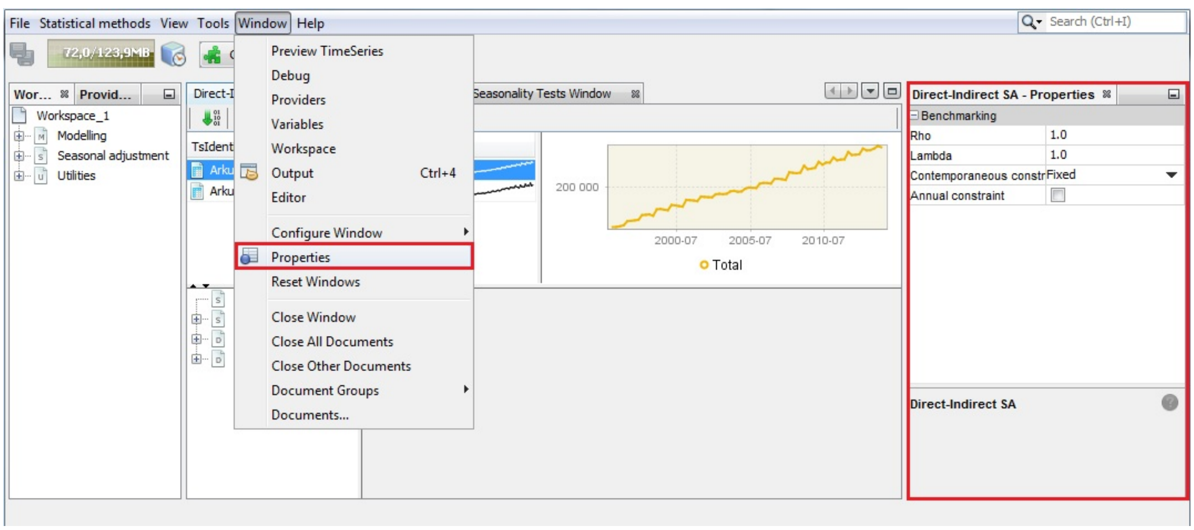


Figure 229: **The properties of the *Direct - Indirect seasonal adjustment* functionality**

below. This will provide you with the alternative specifications. Here the user defined specification named *My spec* is chosen.

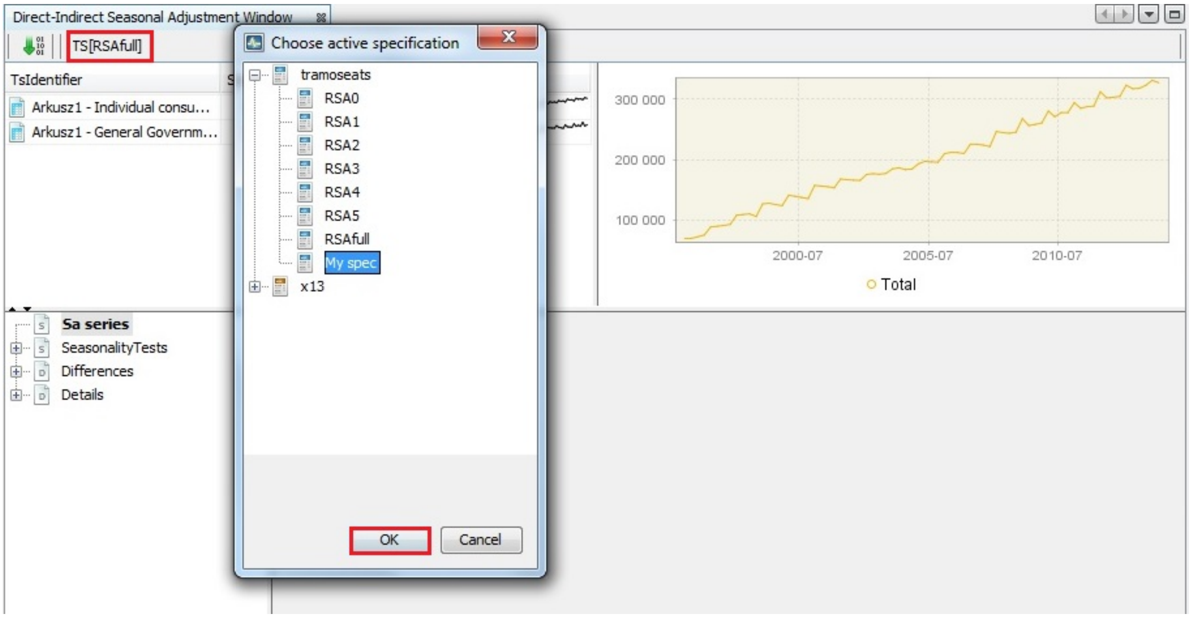


Figure 230: **Choosing a specification for the analysis**

Next, run the process by clicking the button with the green arrow.

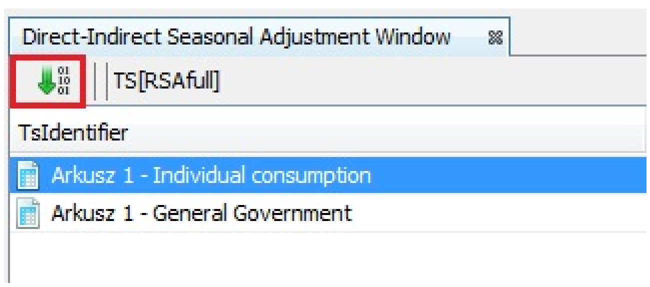


Figure 231: **Running a process**

The bottom panel presents the detailed results. The seasonality test node presents the outcome of the seasonality tests performed for the aggregated series adjusted directly (*Direct sa*) and indirectly (*Indirect sa*). The reason for presenting these tests here is that the presence of residual seasonality and calendar effects should be monitored, especially in the indirectly adjusted series (see (**guidelines2015?**)). It might happen that the seasonality is successfully removed from the components but it is still present in the aggregated series.

The *Differences* node presents selected different results between the direct and

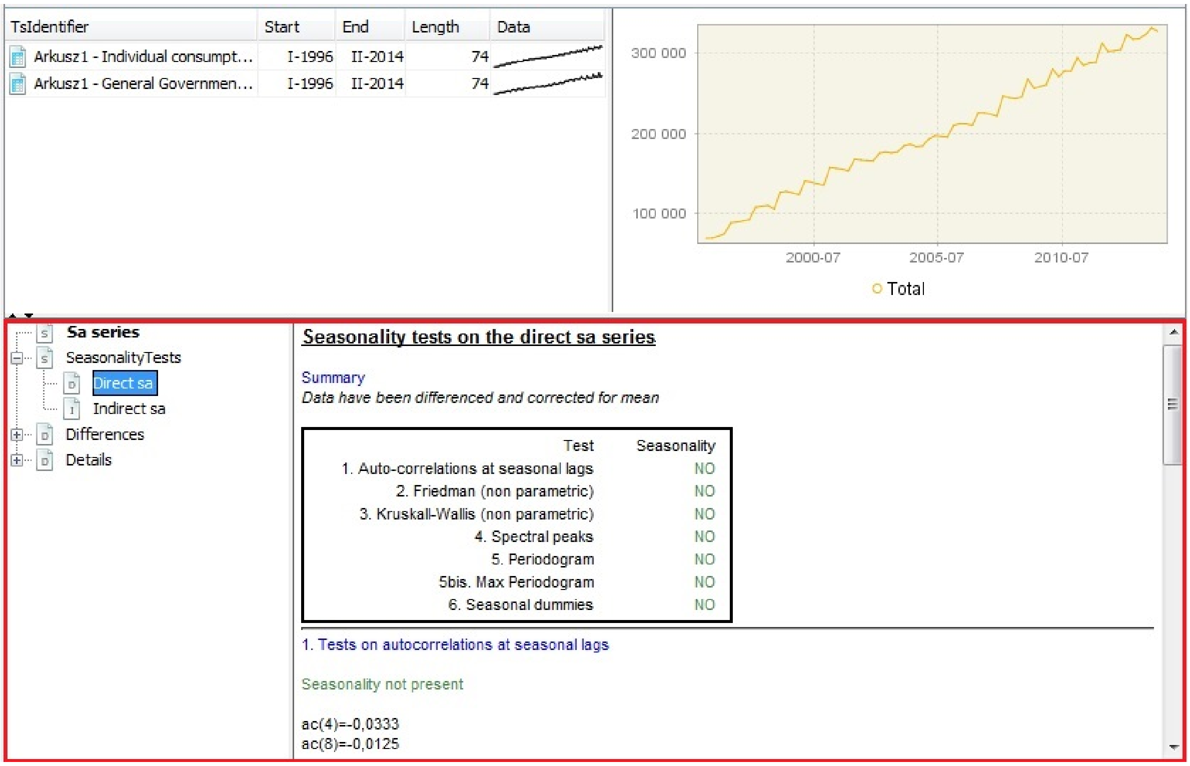


Figure 232: **Seasonality tests' results for a direct seasonal adjustment**

the indirect seasonal adjustment approaches. The *Statistics* section shows basic statistics (average, standard deviation, minimum and maximum) for the relative differences (%) between the direct and the indirect SA series. *Chart* contains the graph of the differences, while *Table* includes the actual values. The *Periodogram* section presents graphs for two spectral estimators – the periodogram and the auto-regressive spectrum.

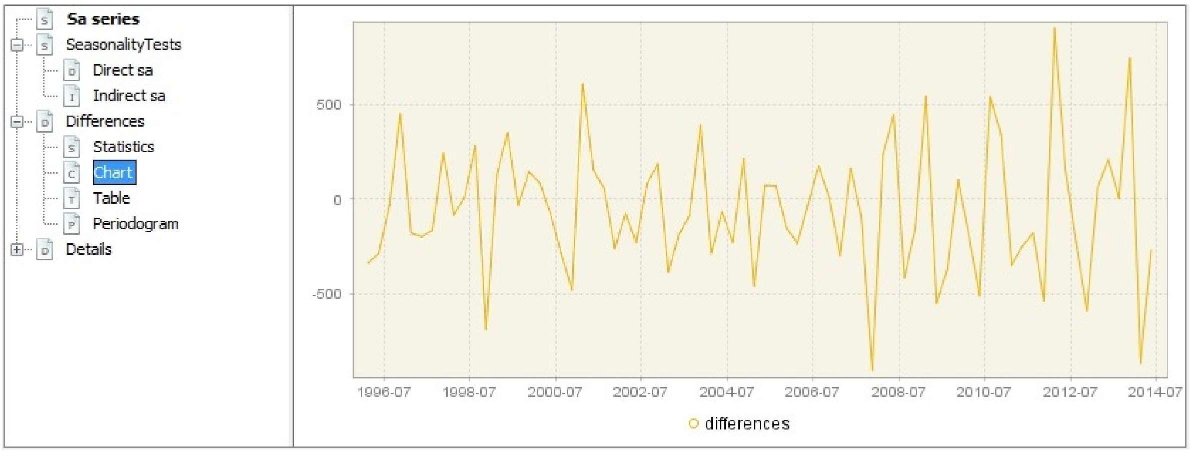


Figure 233: **Graph presenting the differences between direct and indirect seasonal adjustment results**

The *Details* node include the basic statistics for the relative differences between the benchmarked and original series as well as the actual time series adjusted directly (*Sa series*) and indirectly (*Benchmarked Sa series*).

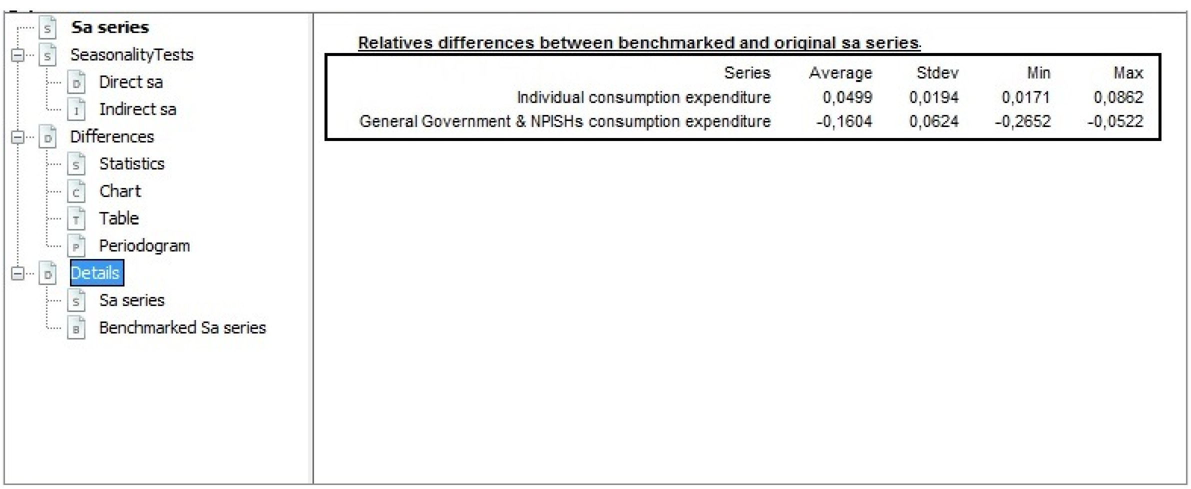


Figure 234: **Details of the differences between direct and indirect seasonal adjustment results**

GUI: Generating output

In this chapter

This chapter describes how to generate (export) output (series, parameters, diagnostics) directly from the Graphical User interface:

When running a SA-processing in GUI, series, parameters, diagnostics can be also generated without opening it, using a production module called the cruncher.

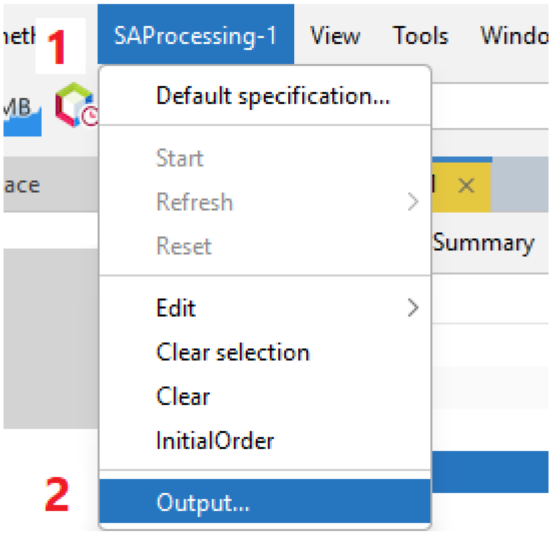
Additional chapters related to GUI features, provide information on:

- Overview
- Data visualization and generic time series tools
- Specific Seasonal Adjustement and Modelling features

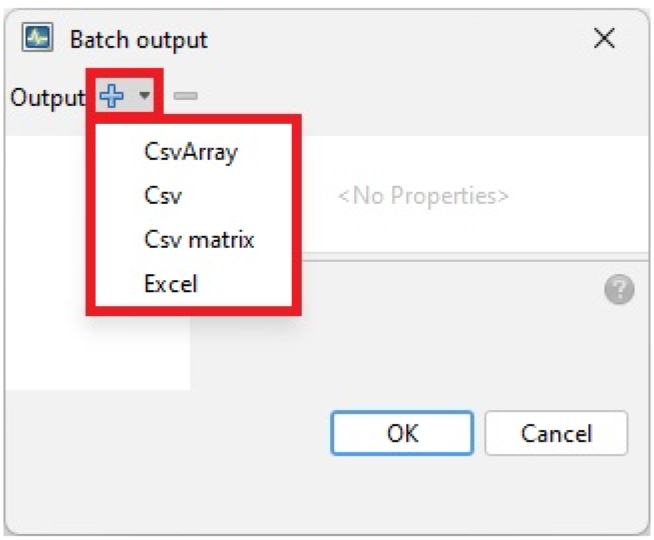
Output from SA Processing

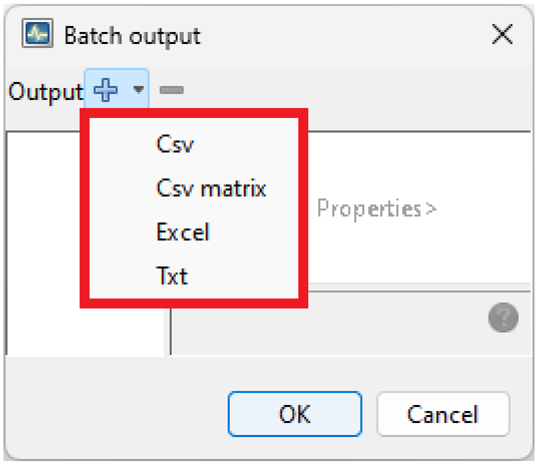
Steps

1. Once a seasonal adjustment process for the dataset is performed Go to the TOP menu bar and follow the path: *SAProcessing → Output...*

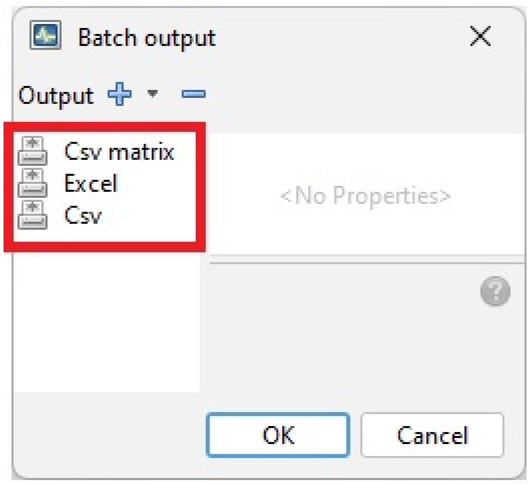


2. In the *Batch output* window the user can specify which output items will be saved and the folder in which JDemetra+ saves the results. It is possible to save the results in the `.txt`, `.csv`, `.xls`, and `.csv` matrix formats. In the first step the user should choose the output format from the list.





3. The user may choose more than one format as the output can be generated in different formats at the same time.



4. To display and modify the settings click on the given output format on the list. The available options depend on the output format.

The different ouput formats

0.0.0.1 `.csv`

For `.csv` format the following options are available:

- *folder* (location of the file),
- *file prefix* (name of the file),
- *presentation* (controls how the output is divided into separate files)

- and *series* (series included in the file).

These options are presented in the next points of this case study.

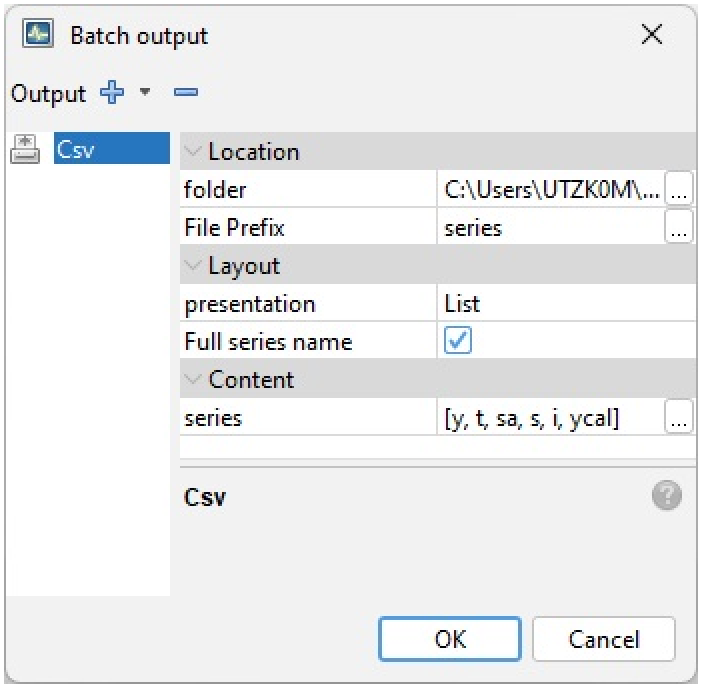


Figure 235: **Options for a `.csv` format**

The user can define the folder in which the selected results and components will be saved (click the *folder* item and choose the final destination).

With the option *File Prefix* the user can modify the default name of the output saved in the `.csv` file.

Presentation controls how the output is divided into separate files. Expand the list to display available options:

- *HTable* – the output series will be presented in the form of horizontal tables (time series in rows).
- *VTable* – the output series will be presented in the form of vertical tables (time series in columns).
- *List* – the output series will be presented in the form of vertical tables (time series in rows). Apart from that, for each time series each file contains in separate columns: the data frequency, the first year and stimation span, the first period (month or quarter) of observation span and the number of observations. The files do not include dates.

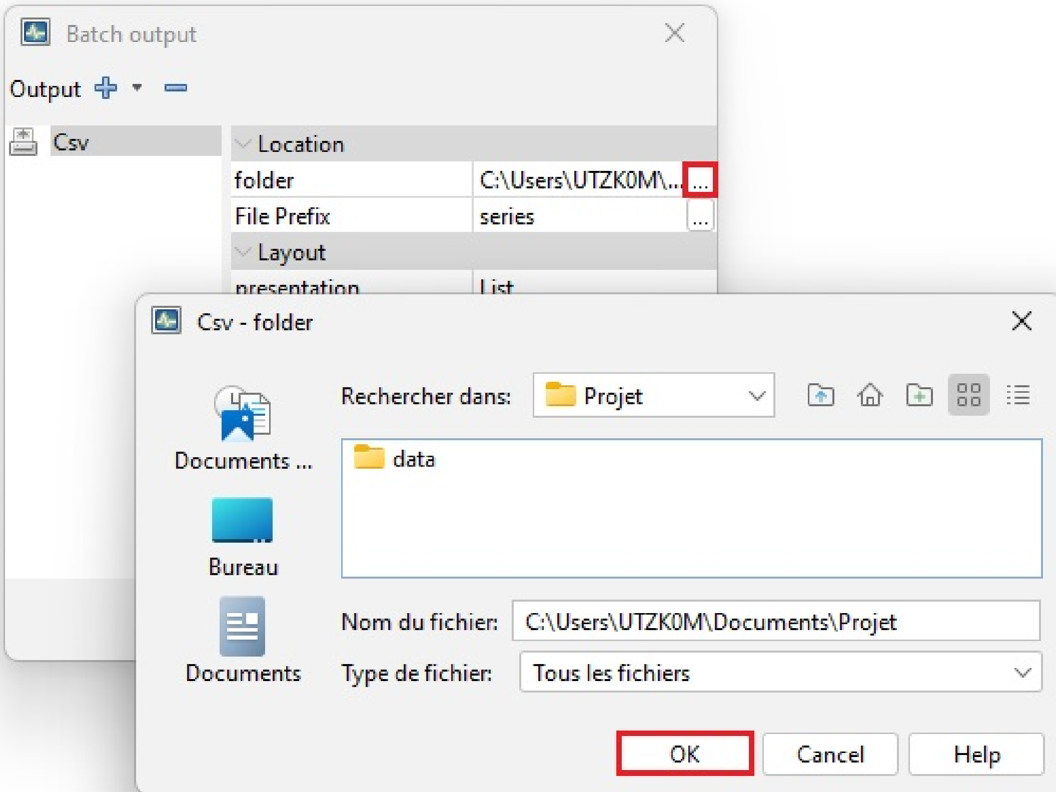


Figure 236: **Specifying a destination folder**

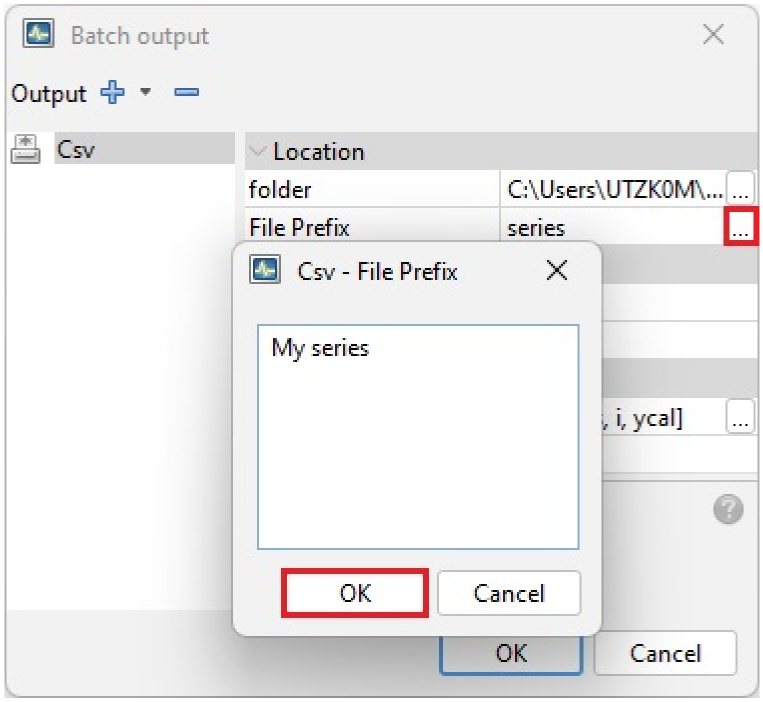


Figure 237: **Setting a *File Prefix* option**

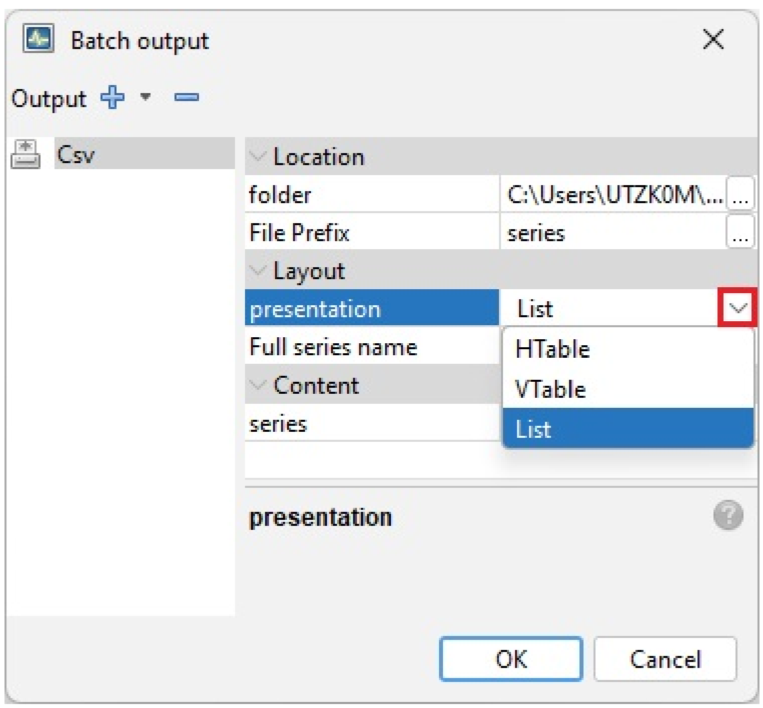


Figure 238: **Layout options for a `.csv` format**

The option *Full series name* is used to use the fully qualified name of the series (workbook + sheet + name) in the output.

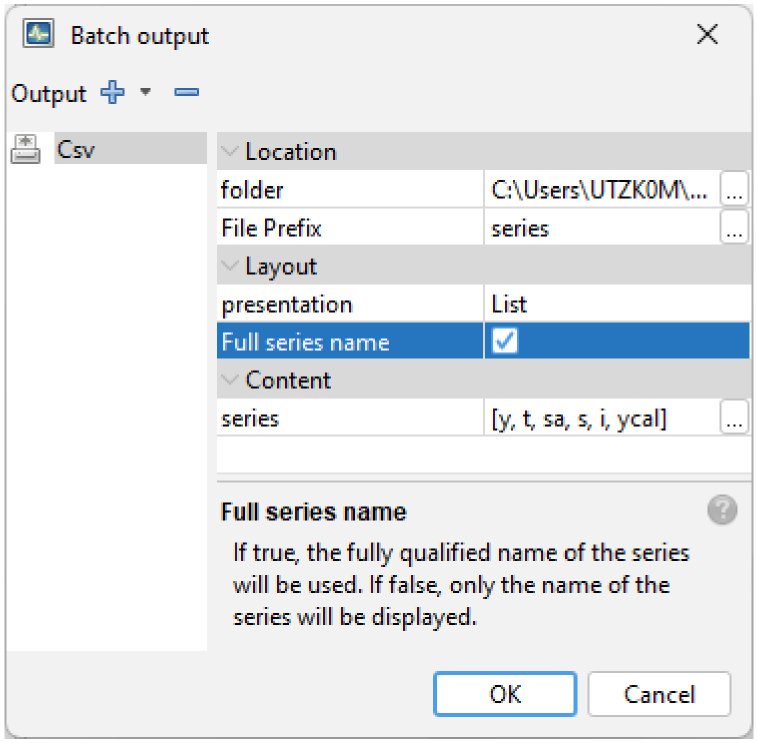


Figure 239: **The *Full series name* option**

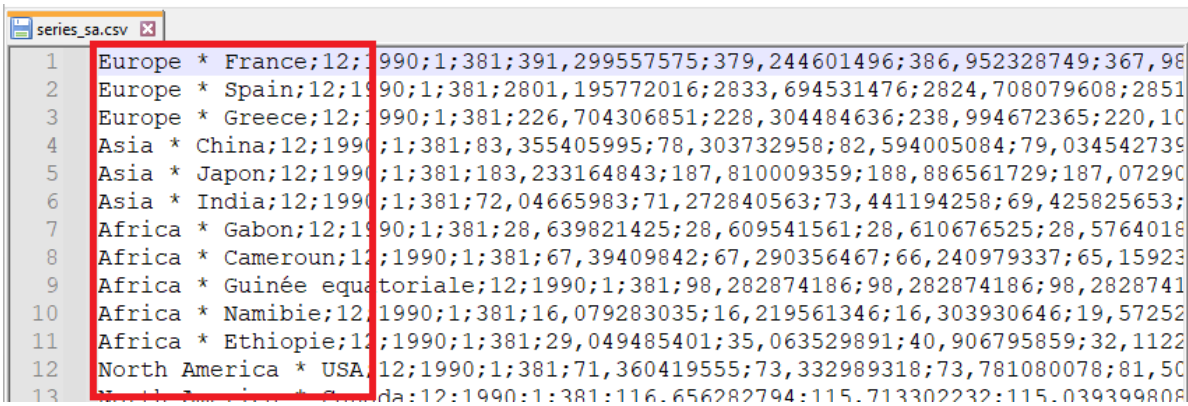
The *Content* section presents a list of series that will be included into a set of output files. To modify the initial settings click on the grey button in the *Content* section. The *CVS-series* window presents two panels:

- the panel on the left includes a list of all valuable output items
- the panel on the right presents the selected output items

Mark the series and use the arrows to change the settings. Confirm your choice with the *OK* button.

0.0.0.2 `.csv` matrix

The `.csv` matrix produces the `.csv` file containing information about the model and quality diagnostics of the seasonal adjustment. The user may generate the list of default items or create their own quality report.



```
Europe * France;12;1990;1;381;391,299557575;379,244601496;386,952328749;367,98
Europe * Spain;12;1990;1;381;2801,195772016;2833,694531476;2824,708079608;2851
Europe * Greece;12;1990;1;381;226,704306851;228,304484636;238,994672365;220,10
Asia * China;12;1990;1;381;83,355405995;78,303732958;82,594005084;79,03454273
Asia * Japon;12;1990;1;381;183,233164843;187,810009359;188,886561729;187,0729
Asia * India;12;1990;1;381;72,04665983;71,272840563;73,441194258;69,425825653;
Africa * Gabon;12;1990;1;381;28,639821425;28,609541561;28,610676525;28,576401
Africa * Cameroun;12;1990;1;381;67,39409842;67,290356467;66,240979337;65,15923
Africa * Guinée equatoriale;12;1990;1;381;98,282874186;98,282874186;98,2828741
Africa * Namibie;12;1990;1;381;16,079283035;16,219561346;16,303930646;19,57252
Africa * Ethiopie;12;1990;1;381;29,049485401;35,063529891;40,906795859;32,1122
North America * USA;12;1990;1;381;71,360419555;73,332989318;73,781080078;81,50
```

Figure 240: **The generated output**

The user can define the folder in which the selected results and components will be saved (click the *folder* item and choose the final destination).

With the option *File Name* the user can modify the default name of the output saved in the `.csv` file.

By default, all the available items are included in the output.

Once the output settings are selected, click the *OK* button.

0.0.0.3 *Excel*

Options available for the `.xls` format are the same as for the `.txt` format with an exception of the *Layout* section.

- *BySeries* – all results for a given time series are placed in one sheet;
- *ByComponent* – results are grouped by components. Each component type is saved in a separate sheet;
- *OneSheet* – all results are saved in one sheet.

If the user sets the option layout to *BySeries*, the output will be generated as follows:

- Series are placed in separate sheets
- Components are placed in different columns

If the user sets the option layout to *ByComponent*, the output will be generated as follows:

- Components are placed in separate sheets

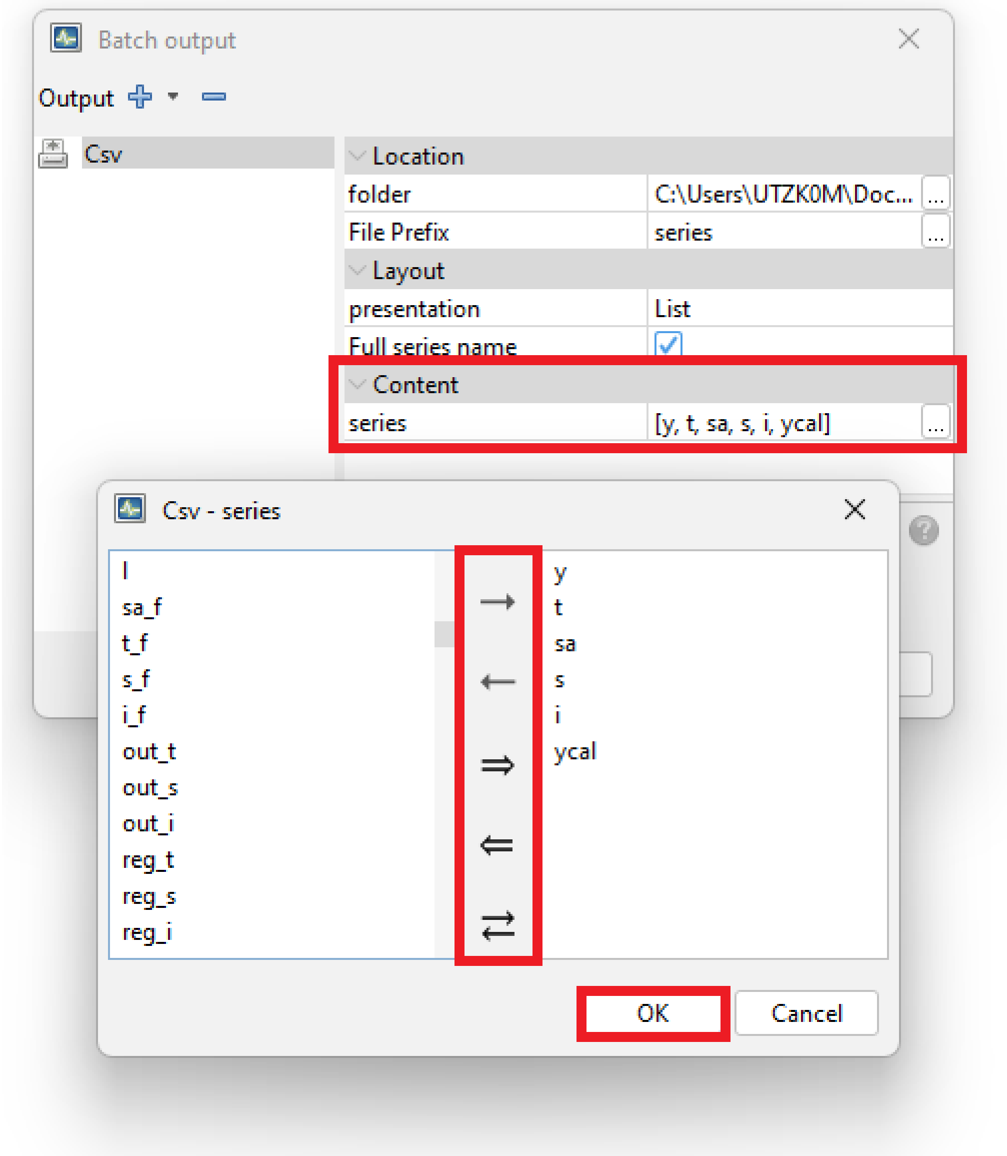


Figure 241: **Specifying a content of the output file**

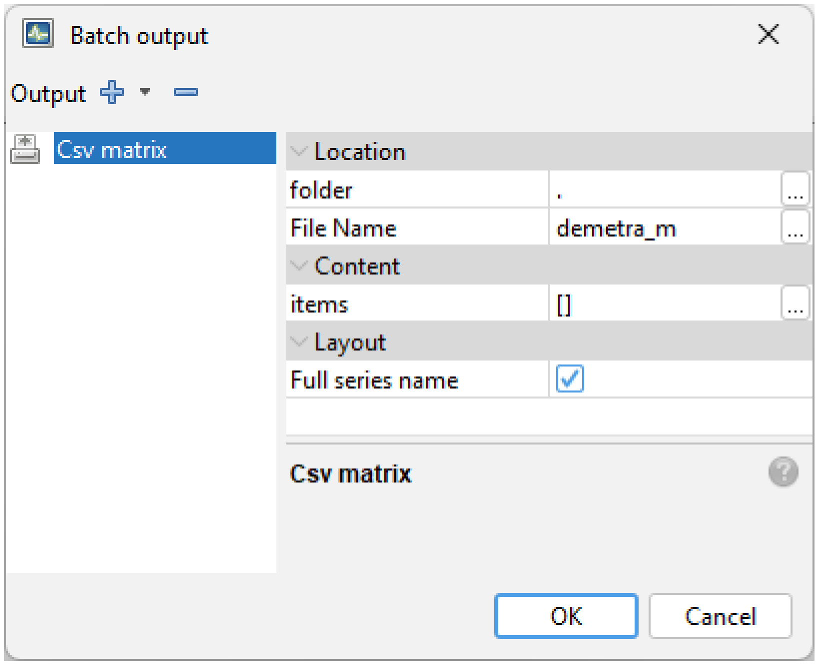


Figure 242: **Options for a `.csv` matrix format**

- Series are placed in different columns

The option *OneSheet* will produce the following `.xls` file:

- Components and Sheets are crossed in different columns of the same sheet

By default, the series in the Excel output files are organised vertically. When the user unmarks the check box the horizontal orientation is used.

The option *Full series name* is used to use the fully qualified name of the series (workbook + sheet + name).

This option will produce the following `.xls` file:

The *Content* section presents a list of series that will be included into a set of output files. To modify the initial settings click on the grey button in the *Content* section. The *Excel-series* window presents two panels:

- the panel on the left includes a list of all valuable output items
- the panel on the right presents the selected output items

Mark the series and use the arrows to change the settings. Confirm your choice with the *OK* button.

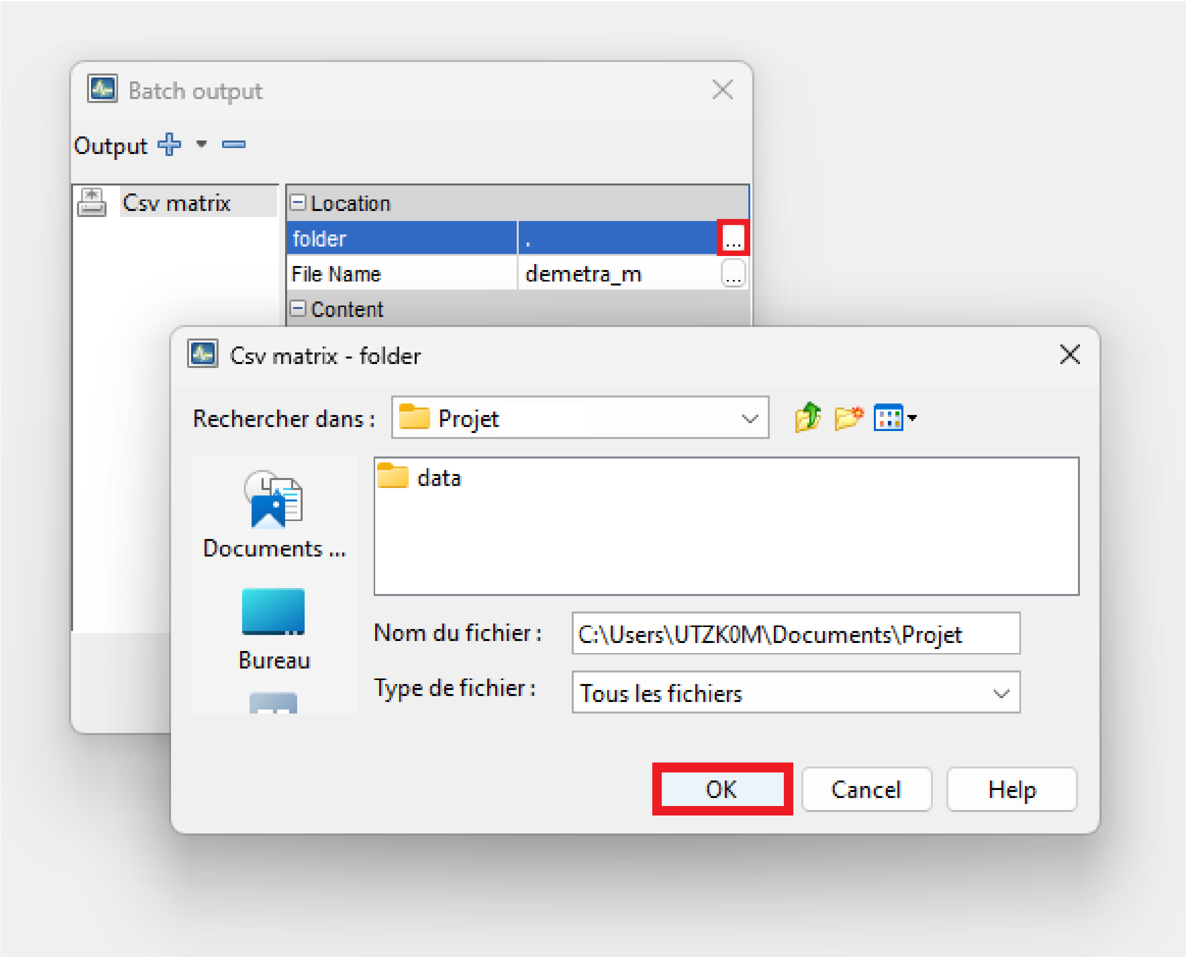


Figure 243: **Specifying a destination folder**

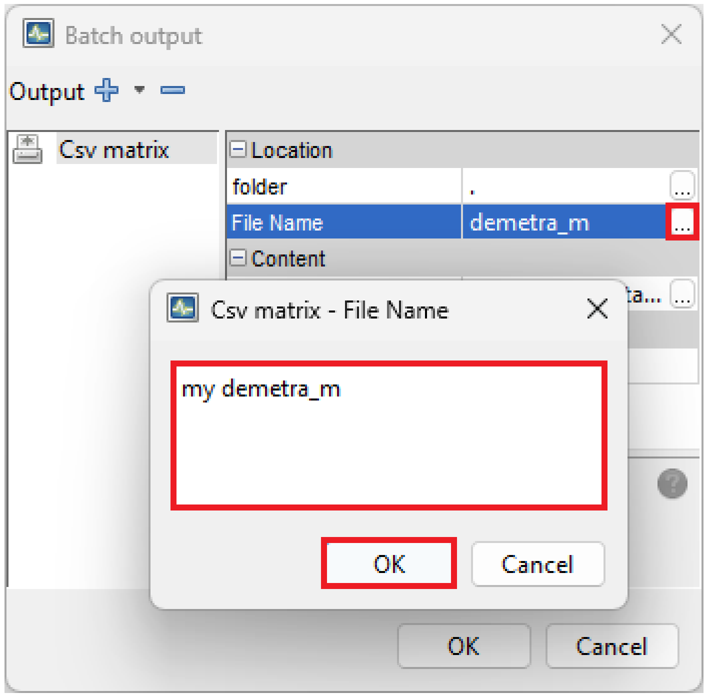


Figure 244: **Setting a *File Name* option**

0.0.0.4 `.txt` (only in v2)

In the case of the `.txt` format the only available options are *folder* (location of the file) and *series* (results included in the output file).

0.0.0.5 *CvsArray* (only in v3)

For each output JDemetra+ provides information on the status of the operation. An example is presented below.

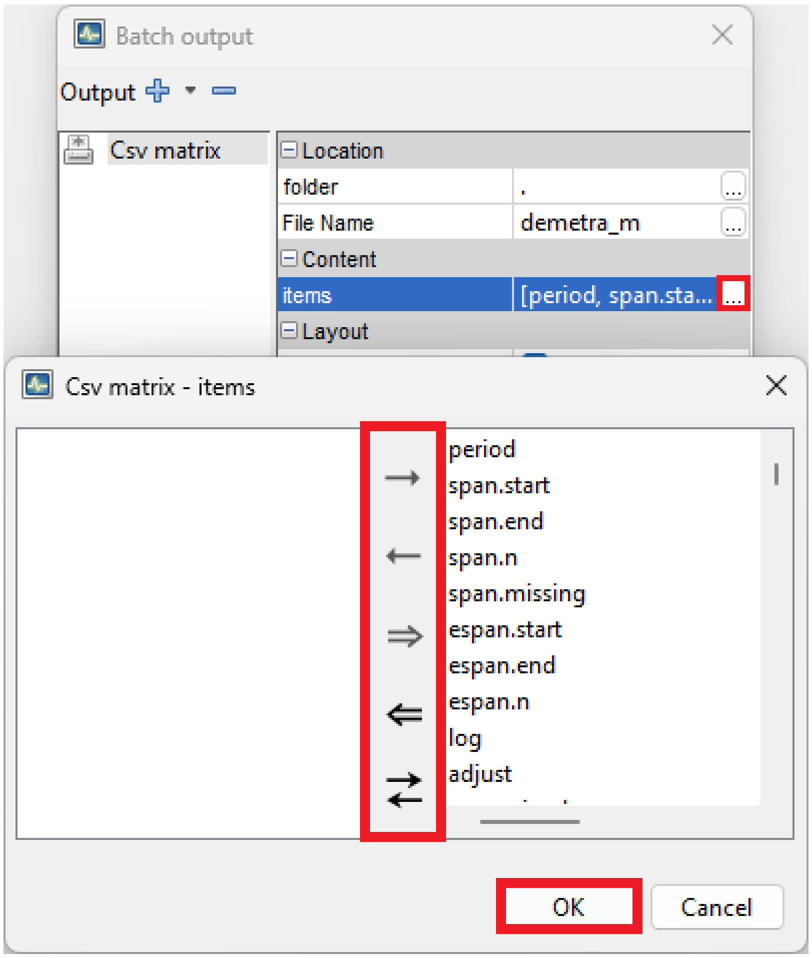


Figure 245: **List of items available for the** `.csv` **matrix output type**

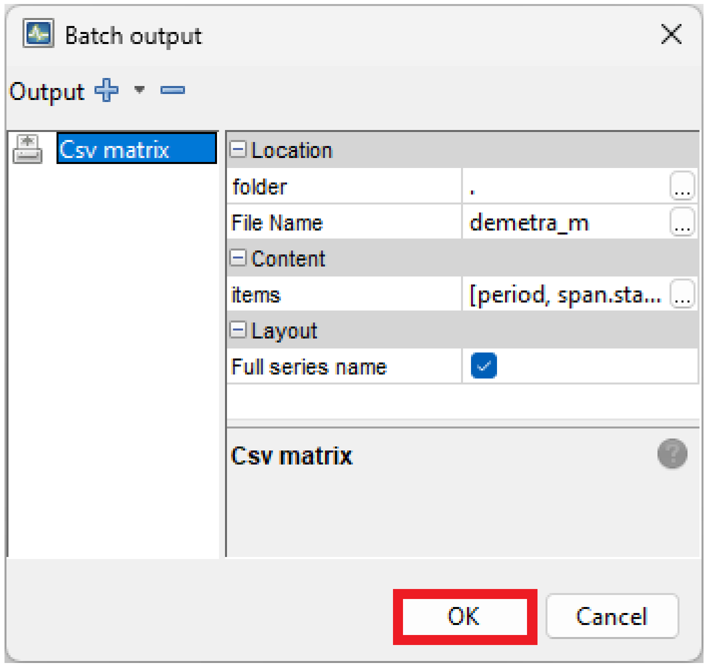


Figure 246: **Validation of the** `.csv` **matrix output**

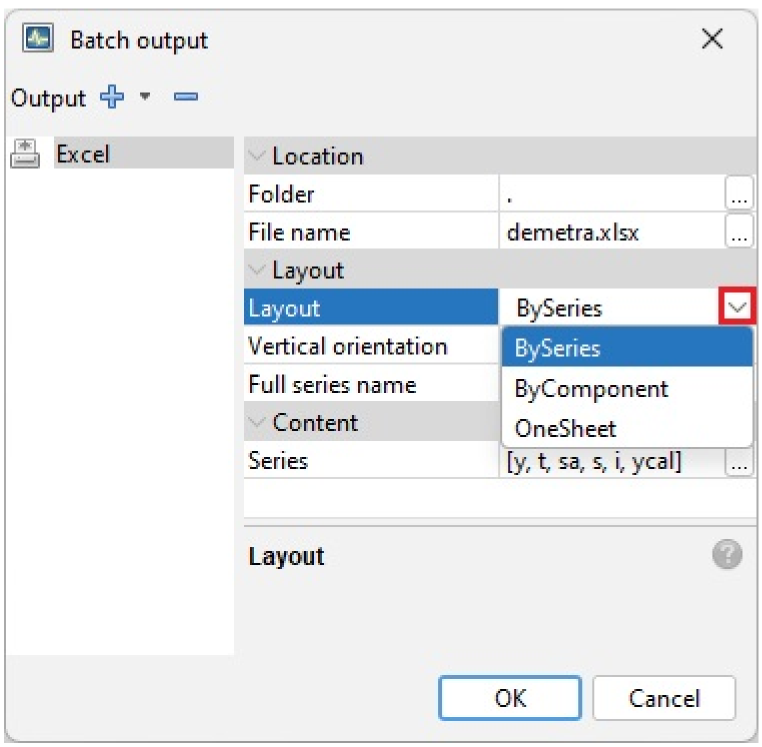


Figure 247: **Layout options for an *Excel* format**

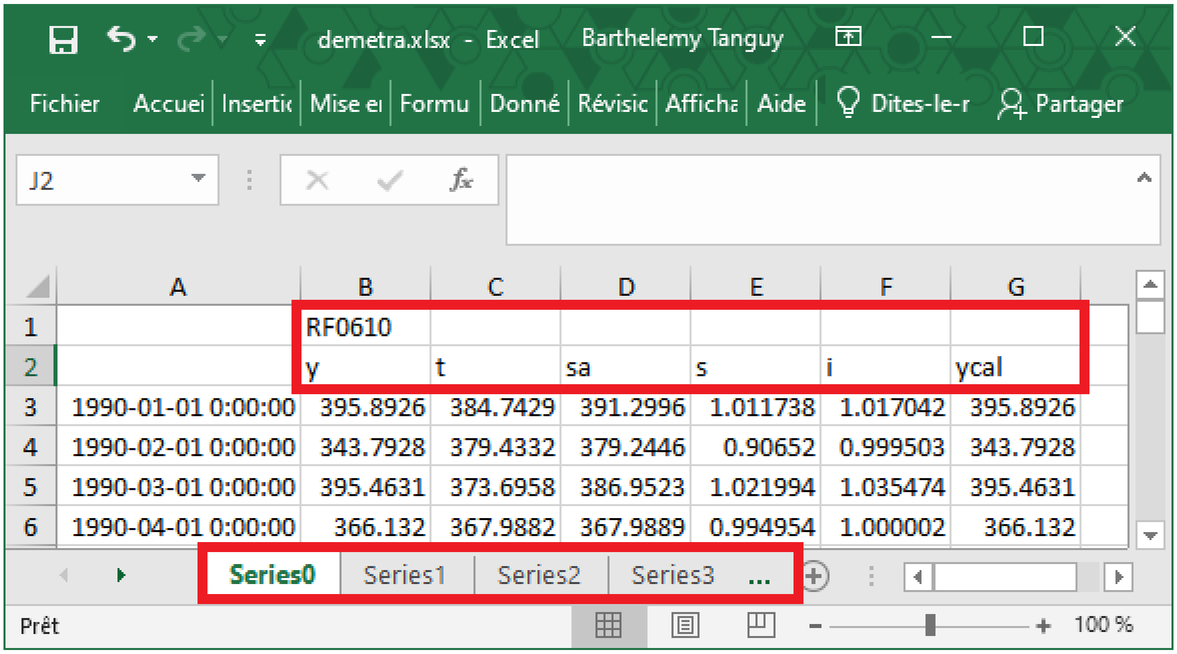


Figure 248: **An Excel file view for the *BySeries* option**

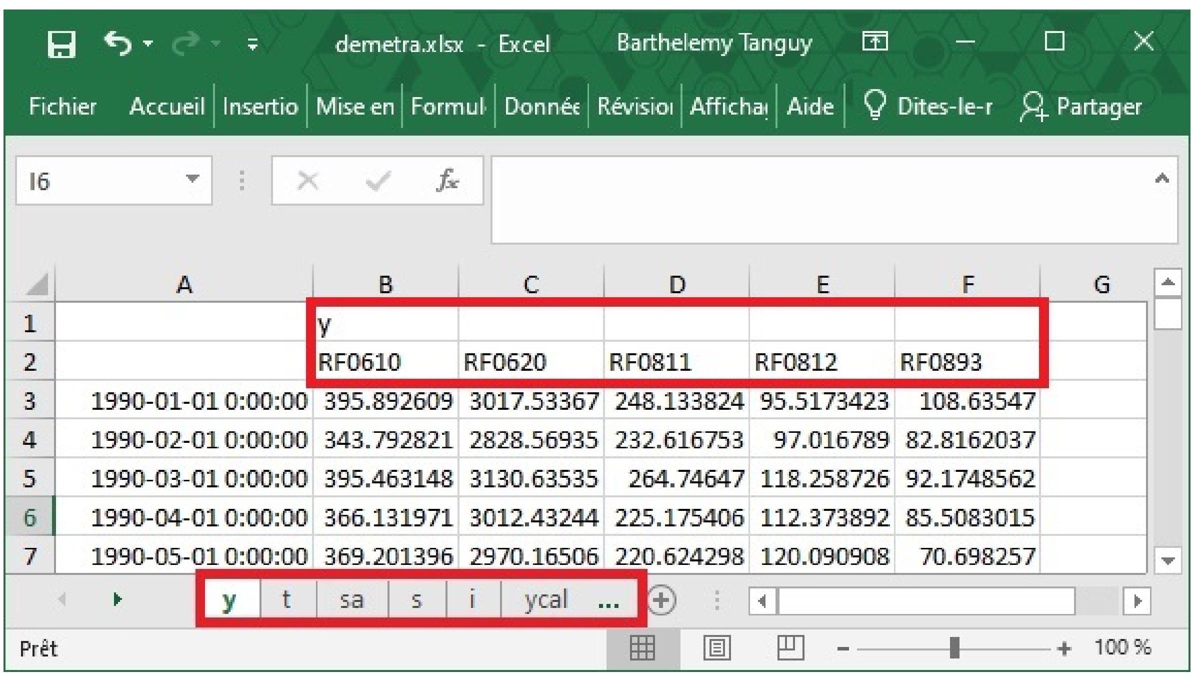


Figure 249: **An Excel file view for the *ByComponent* option**

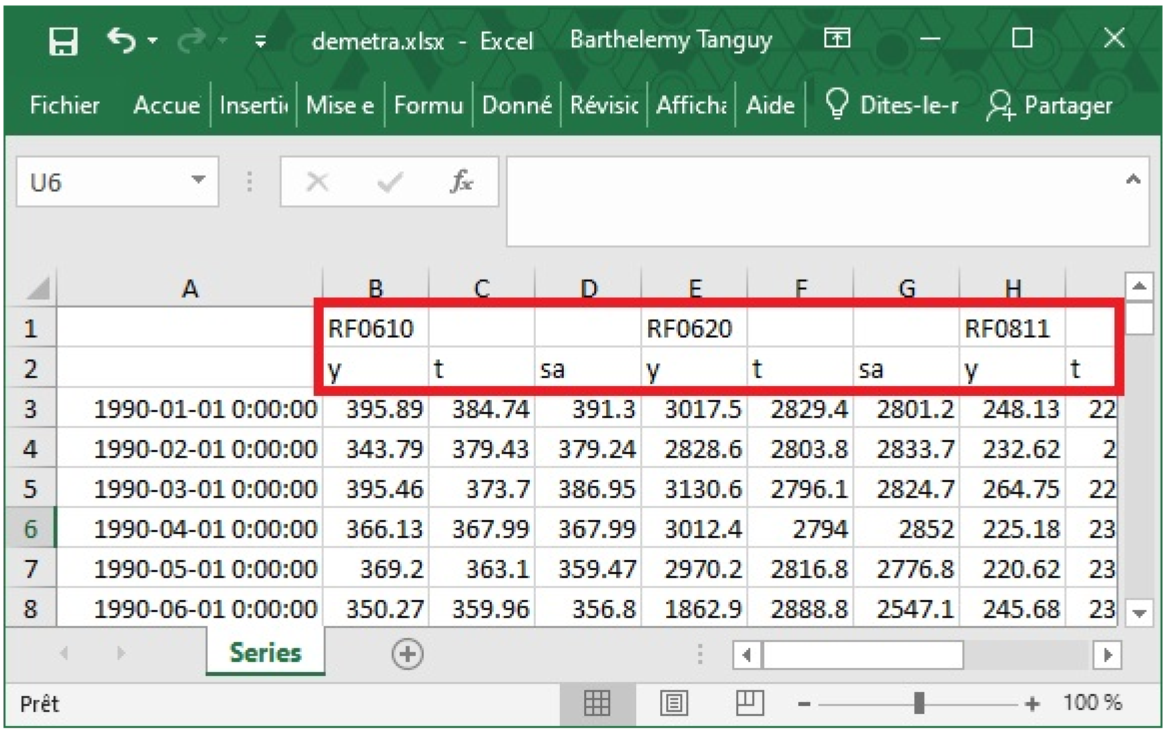


Figure 250: **An Excel file view for the *OneSheet* option**

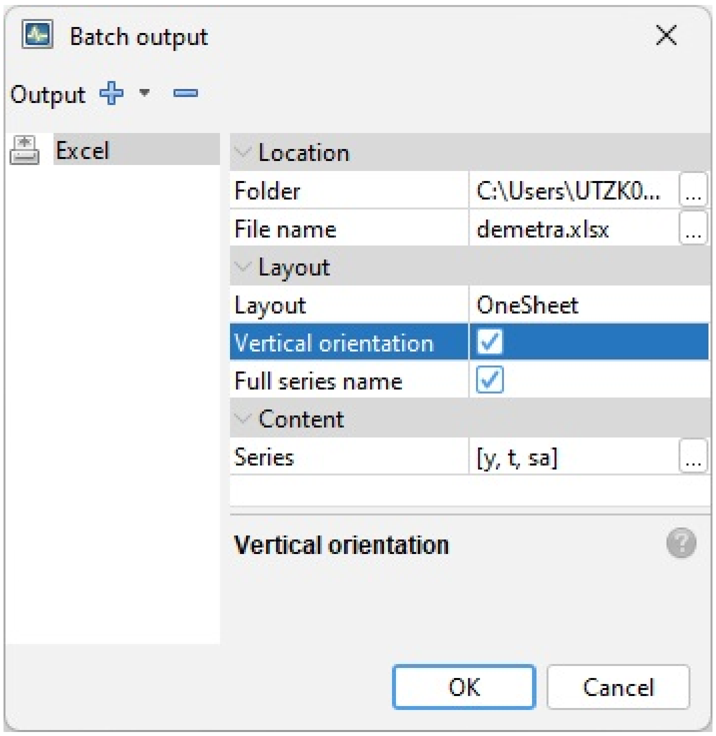


Figure 251: **The *VerticalOrientation* option**

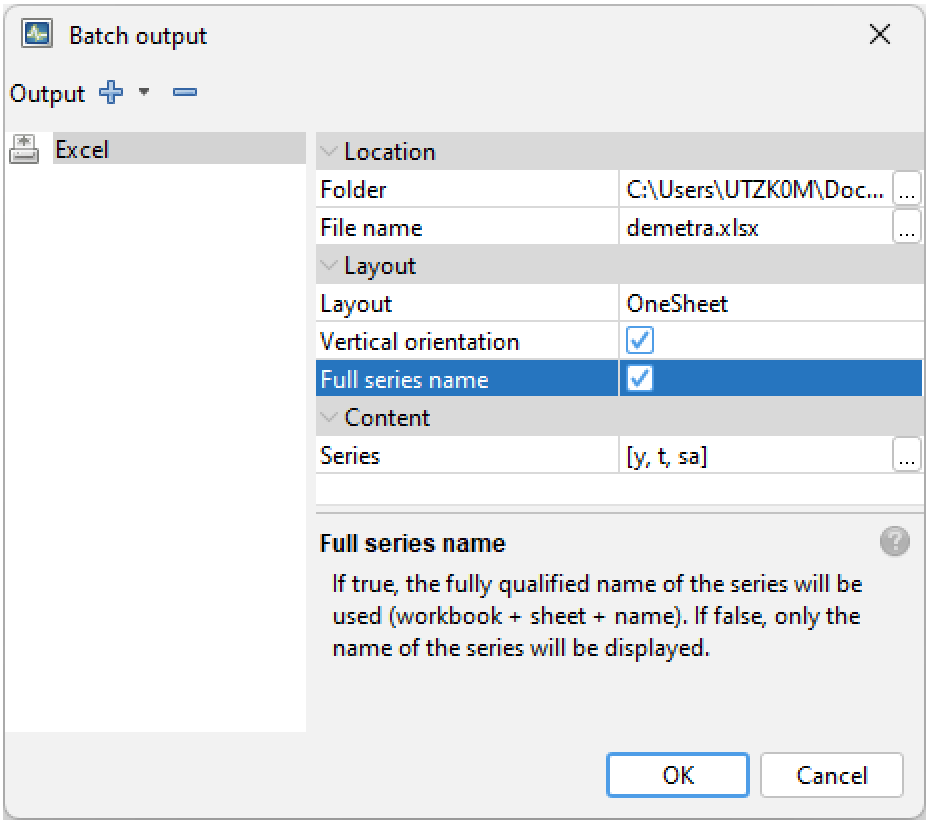


Figure 252: **The *Full series name* option**

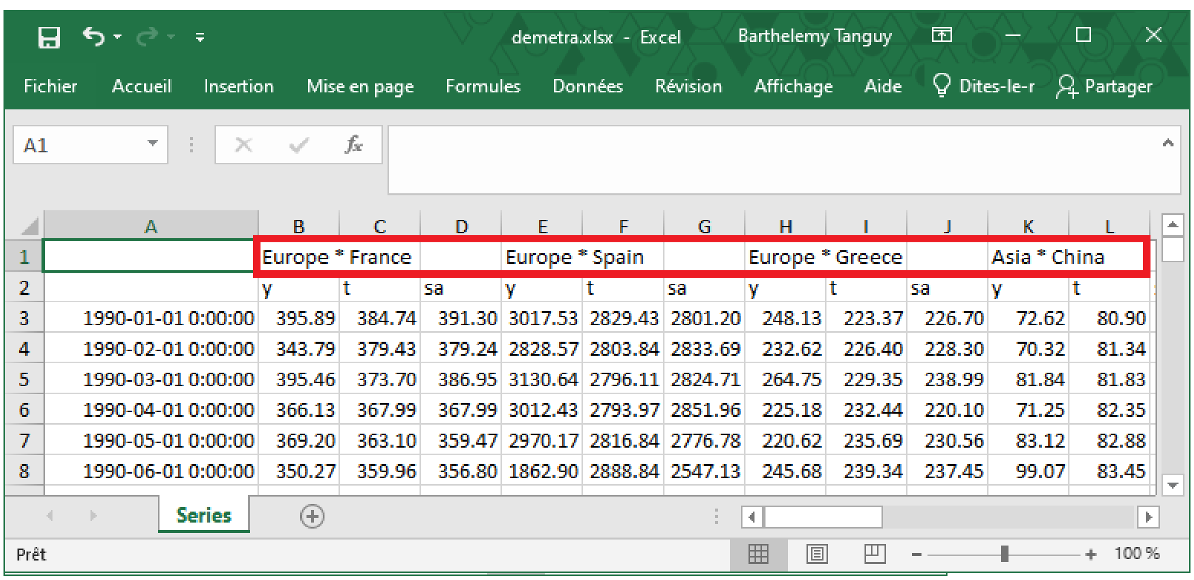


Figure 253: **The generated output**

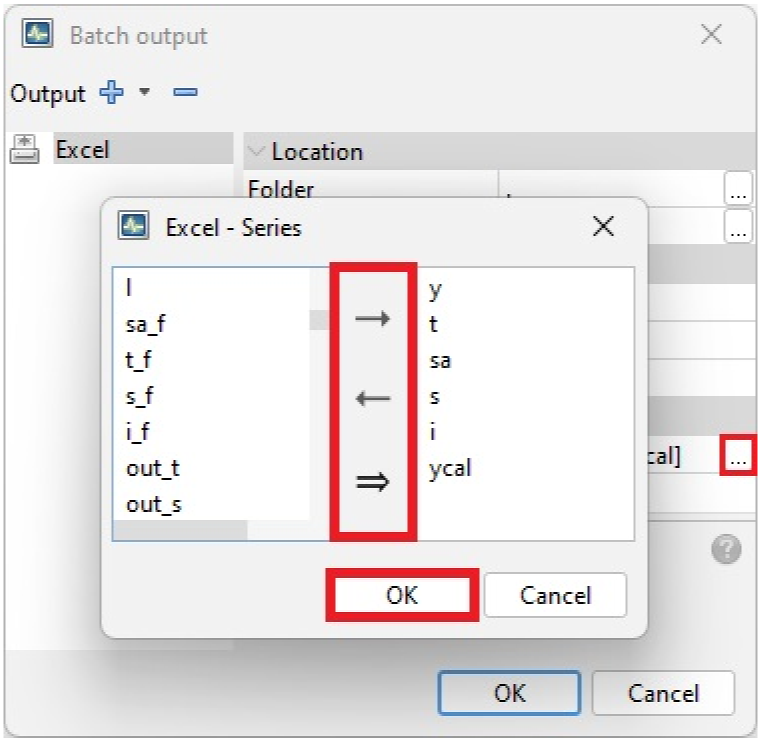


Figure 254: **Specifying a content of the output file**

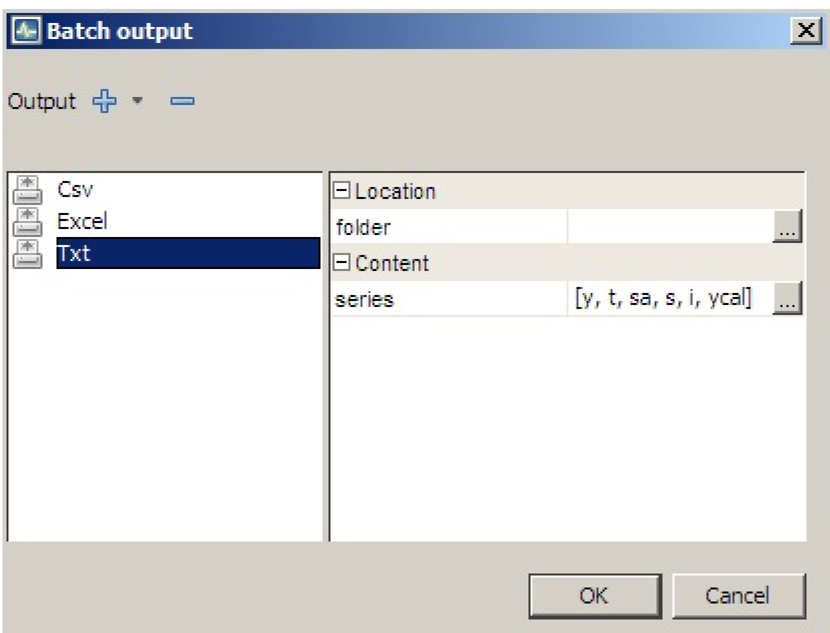


Figure 255: **Options for the** `.txt` **output**

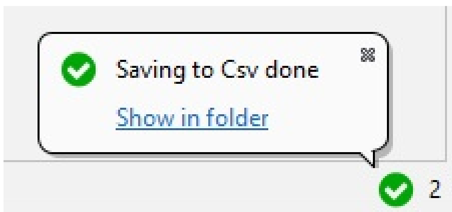


Figure 256: **Generating output - status information (here for `.csv` format)**

Plug-ins (GUI extensions)

JDemetra+ Graphical User Interface can be extended with plug-ins, which are components adding specific features to the **main** software.

In this chapter, we cover

- Algorithms available via plug-ins
- Utility plug-ins: data providers, data formatting
- Installation procedure which consists of downloading the appropriate "*.nbm" file and installation it with the GUI.

Algorithms available with plug-ins

Here is an overview of the algorithms which can be added by installing additional plug-ins to the Graphical User Interface. For all the mentioned files corresponding number of version is denoted x.y.z.

Seasonal Adjustment of Low frequency data

For low frequency (monthly, quarterly, half-yearly, quadri-monthly) data the following additional algorithms are available with plug-ins. Corresponding files are located in the Incubator (latest releases). Installation of auxiliary plug-in "jdplus-advancedsa-desktop-plugin-x.y.z.nbm" also located in the incubator is required.

Algorithm	Plugin File Name	Documentation pages
X12+ Airline based pre-adjustment and extended X11 decomposition	jdplus-x12plus-desktop-plugin-x.y.z.nbm	Usage in GUI described here
STL+ Airline based pre-adjustment and STL (Loess) decomposition	jdplus-stl-desktop-plugin-x.y.z.nbm	Usage in GUI described here

Algorithm	Plugin File Name	Documentation pages
STS+ Airline based pre-adjustment for outlier detection and calendar estimation, followed by BSM modelling	jdplus-sts-desktop-plugin-x.y.z.nbm	Usage in GUI described here
Seats+ Airline based pre-adjustment and Seats decomposition	upcoming	upcoming

Seasonal Adjustment of High-Frequency data

For low frequency (monthly, quarterly, half-yearly, quadri-monthly) data the following algorithms are available with plug-ins. Corresponding files are located in the Incubator (latest releases). Installation of auxiliary plug-in "jdplus-advancedsa-desktop-plugin-x.y.z.nbm" also located in the incubator is required.

Algorithm	Plugin File Name	Repo / Comments
Extended Airline Estimation (Reg-ARIMA Modelling)	jdplus-highfreq-desktop-plugin-x.y.z.nbm	Usage in GUI described here
Extended Airline Decomposition (Extended Seats)	jdplus-highfreq-desktop-plugin-x.y.z.nbm	Usage in GUI described here
MX12+ Extended Airline Estimation and Extended X11 Decomposition	upcoming	upcoming
MSTL+ Extended Airline Estimation and STL Decomposition	upcoming	upcoming
MSTS Basic Structural Models	upcoming	upcoming

Advanced Seasonal adjustment features for v 2.x

SA Advanced provides some experimental seasonal adjustment methods (with Reg-ARIMA preprocessing), basic structural models, generalized airline models and airline + seasonal noise models (called mixed airline).

- `gairline: generalized airline model`
- `mairline: mixed airline model`
- `mixedfreq: mixed frequencies seasonal adjustment`
- `sssts: Seasonal specific structural time series`
- `sts: Structural time series`

Benchmarking

The plug-in is common for benchmarking and temporal disaggregation.

In version 3.x, the plug-in file ("jdplus-benchmarking-desktop-plugin-x.y.z.nbm") is available here for download. Usage in GUI is described in this chapter.

In version 2.x, the plug-in file ("nbdemetra-benchmarking-2.2.2.nbm") is available here for download. Usage in GUI is described in this chapter.

Method	Plug-in for Version 2	Plug-in for version 3
Denton	✔	✔
Cholette	✔	✔
Cholette Multi-variate	✔	✖
Cubic Splines	✖	✔
Grp	✖	✔
Calendarization	✔	✖

Temporal Disaggregation

The plug-in is common for benchmarking and temporal disaggregation.

Method	Plug-in for Version 2	Plug-in for Version 3
Regression Models*	✔	✔
Model-based Denton	✖	✔

In version 3.x, the plug-in file ("jdplus-benchmarking-desktop-plugin-x.y.z.nbm") is available here for download. Usage in GUI is described in this chapter

In version 2.x, the plug-in file ("nbdemetra-benchmarking-2.2.2.nbm") is available here for download. Usage in GUI is described in this chapter

Nowcasting

In version 3.x, nowcasting are available via the rjd3nowcasting package.

In version 2.x, the plug-in file ("nbdemetra-dfm-2.2.3.nbm") is available here for download. Usage in GUI is described on this wiki page

Utility plug-ins

Fetching time series from web services (SDMX)

The SDMX plug-in aloows to import time series from SDMX website to JDemetra+ by querying web services or parsing files.

Documentation (made for version 2. but quasi-identical in version 3.x) is available on this wiki page.

In version 3.x, the plug-in file ("jdplus-sdmx-desktop-plugin-x.y.z.nbm") is available here for download.

In version 2.x, the plug-in file ("jdplus-sdmx-desktop-plugin-x.y.z.nbm") is available here for download.

Manual fine tuning in Seasonal Adjustment

The following plug-in was developed by the Deutsche Bundesbank (BBK). For the moment it is only available for version 2.x.

- CompRes: The plug-in CompRes supports the controlled current adjustment approach. It pro the storage of the current components and offers graphical tools to compare forecasted and re-estimated figures. Furthermore, a pre-defined summary of the output containing the most important quality measures can be exported to HTML files.

Chain linked Indices

The following plug-ins were developed by the Deutsche Bundesbank (BBK). For the moment they are only available for version 2.x.

- KIX: The plug-in KIX (German for chain-linked index) has been designed to facilitate the handling of this index type. It offers addition and subtraction of two or more chain-linked time series as well as the computation of contributions of growth.
- KIX2.0: KIX 2.0 offers addition and subtraction of two or more chain-linked time series as well as the computation of contributions of growth following the concept of annual overlap. Contributions to growth are calculated with the partial contribution to growth approach.
- KIXE: KIX_E offers addition and subtraction of two or more chain-linked time series as well as the computation of contributions of growth following the concept of one-period overlap. Contributions to growth are calculated with the aid of the Ribe (1999) contribution to growth approach.
- KIX: The program KIX-CC offers for continuously chain-linked indices the aggregation or disaggregation of two or more indices, or the calculation of contributions to growth.

Calendar regressors wrangling

The following plug-in was developed by the Deutsche Bundesbank (BBK). For the moment it is only available for version 2.x.

- TransReg: The plug-in TransReg allows the user to carry out grouping and centring of user-defined regression variables in JD+.

Specification conversion

The following plug-in was developed by the Deutsche Bundesbank (BBK). For the moment it is only available for version 2.x.

- Xlsx2Ws: The plug-in Xlsx2Ws allows the converting of specific workspace information to a xlsx file and vice versa.

Quality report

The Quality Report (QR) Eurostat plug-in is available on this GtHub page: download the latest version of the *.nbm file and proceed with installation as indicated in this chapter

It is commonly used by Labor Cost Index (LCI) producers across eurosystem countries.

Data base access

The following plug-ins was developed by the National bank of Belgium (NBB). It is only available for version 2.x.

- AccessThis JDemetra+ extension is a pure java library for reading time series from MS Access databases. It currently supports versions 2000-2016 read/write and 97 read-only.Being a pure Java library, you don't need MS Access installed in order to read Access files.

Installation procedure

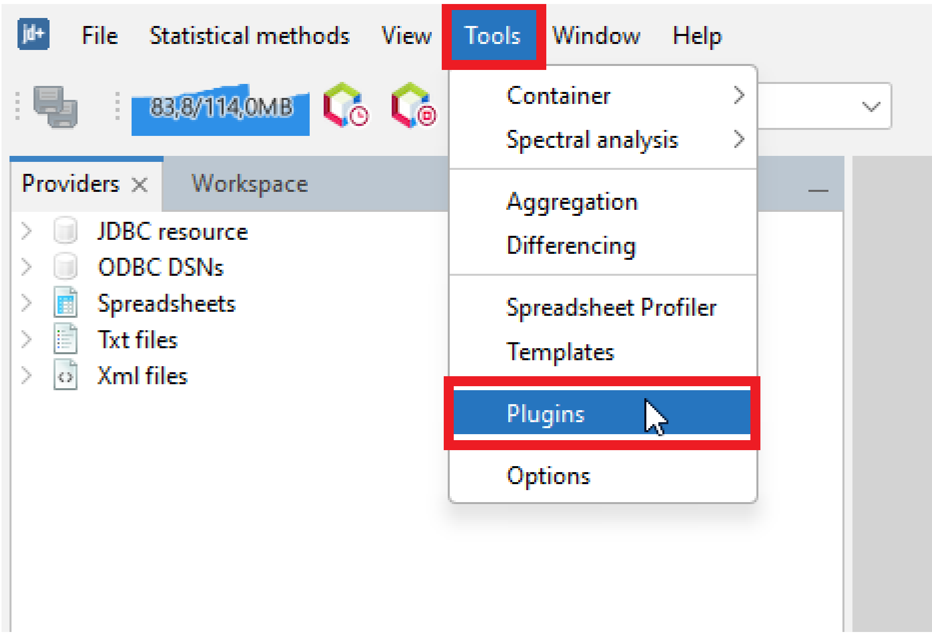
Installation from GUI

menu>tools> plug-ins

The *Plugins* window includes five panels: *Updates*, *Available plugins*, *Downloaded*, *Installed* and *Settings*, some of them however are not operational in the current version of the software.

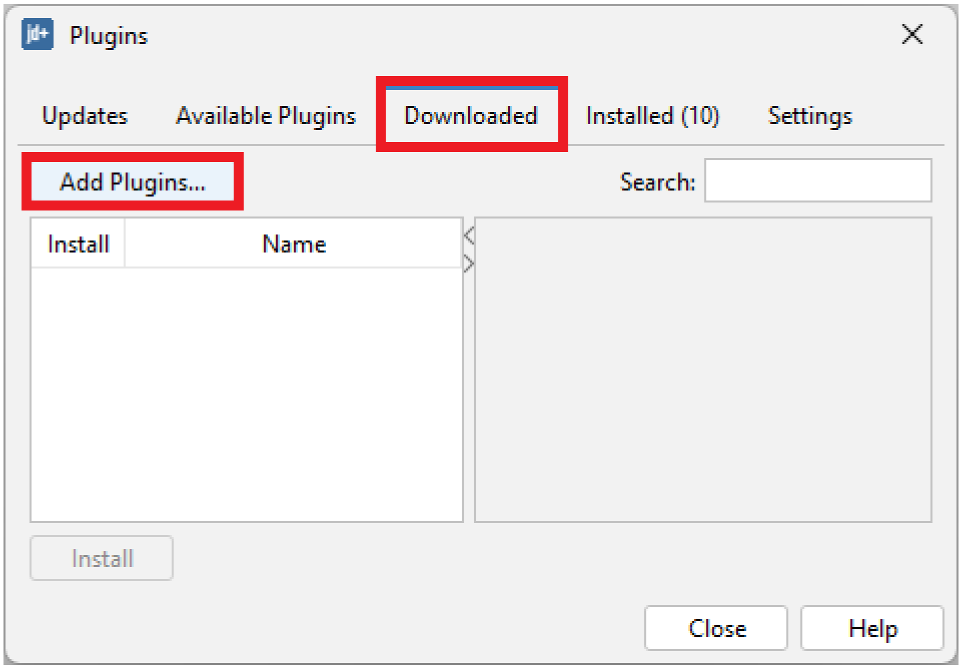
- The *Updates* panel offers the user the option to manually check if some updates of the already installed plugins are available. This functionality, however, is currently not operational for the JDemetra+ plugins.
- The *Available plugins* panel allows the downloading of all plugins that are related to JDemetra+. This functionality, however, is currently not operational for the JDemetra+ plugins.
- The *Downloaded* panel is designed for the installation of new plugins from a local machine. This process in explained in more detail below.
- The *Settings* panel is designated for adding update centres, which are the locations that hold plugins. For each centre the user can specify proxy settings and a time interval to automatically check for any updates. At the moment this functionality is not operational for the JDemetra+ plugins.

Installation of the new plugins from the local machine can be done from the *Plugin* functionality activated from the *Tools* menu.

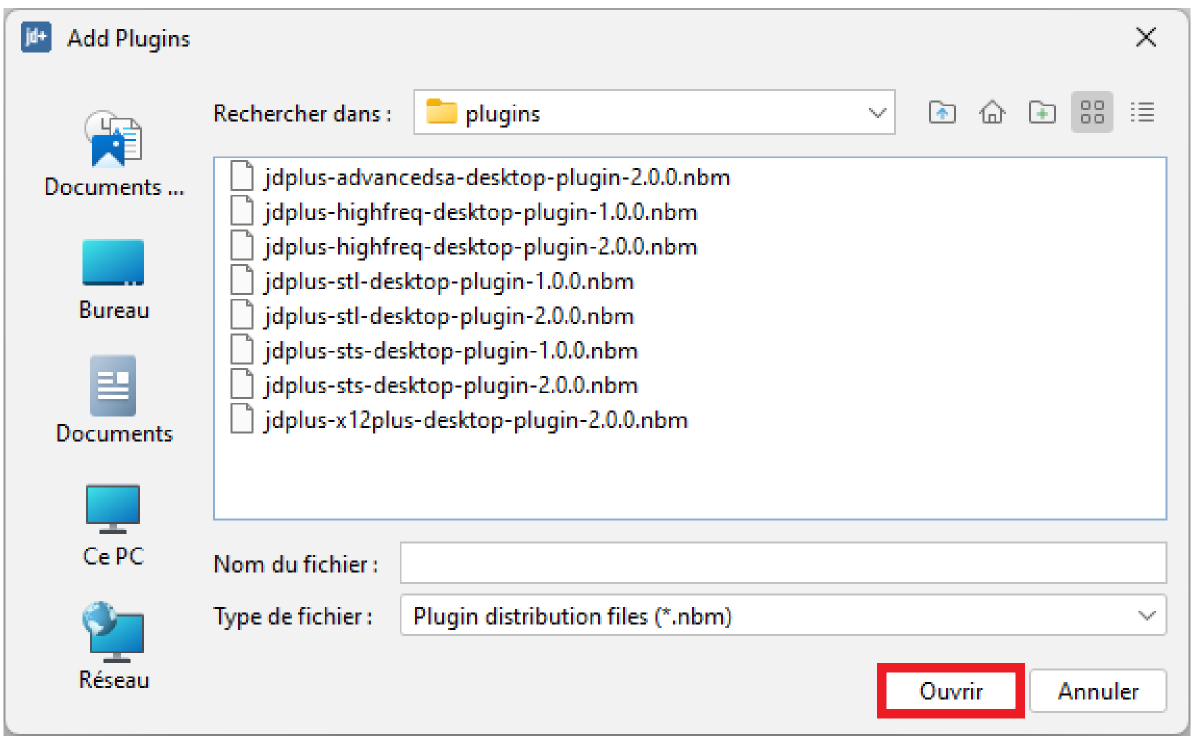


Activation of the *Plugin* functionality from the *Tools* menu

To start the process, go to the *Downloaded* panel and click on the **Add Plugins...** option. Next the user should select the plugins from the folder in which the plugins have been saved and click the **OK** button.

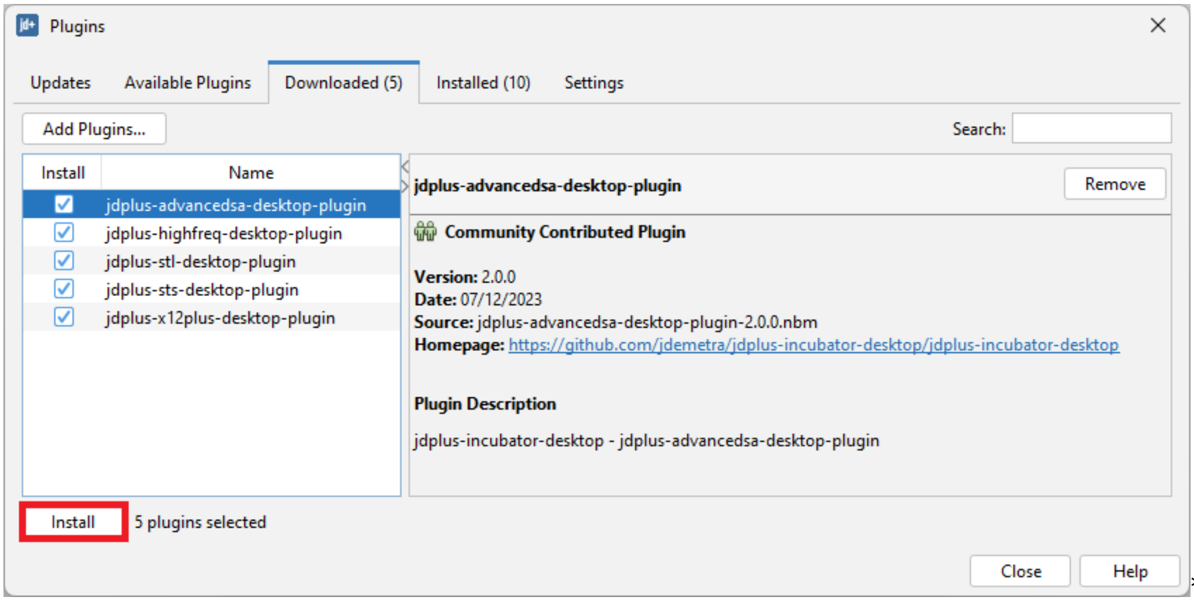


The *Downloaded* panel



Choice of available plugins

The new plugin is now visible in the panel.

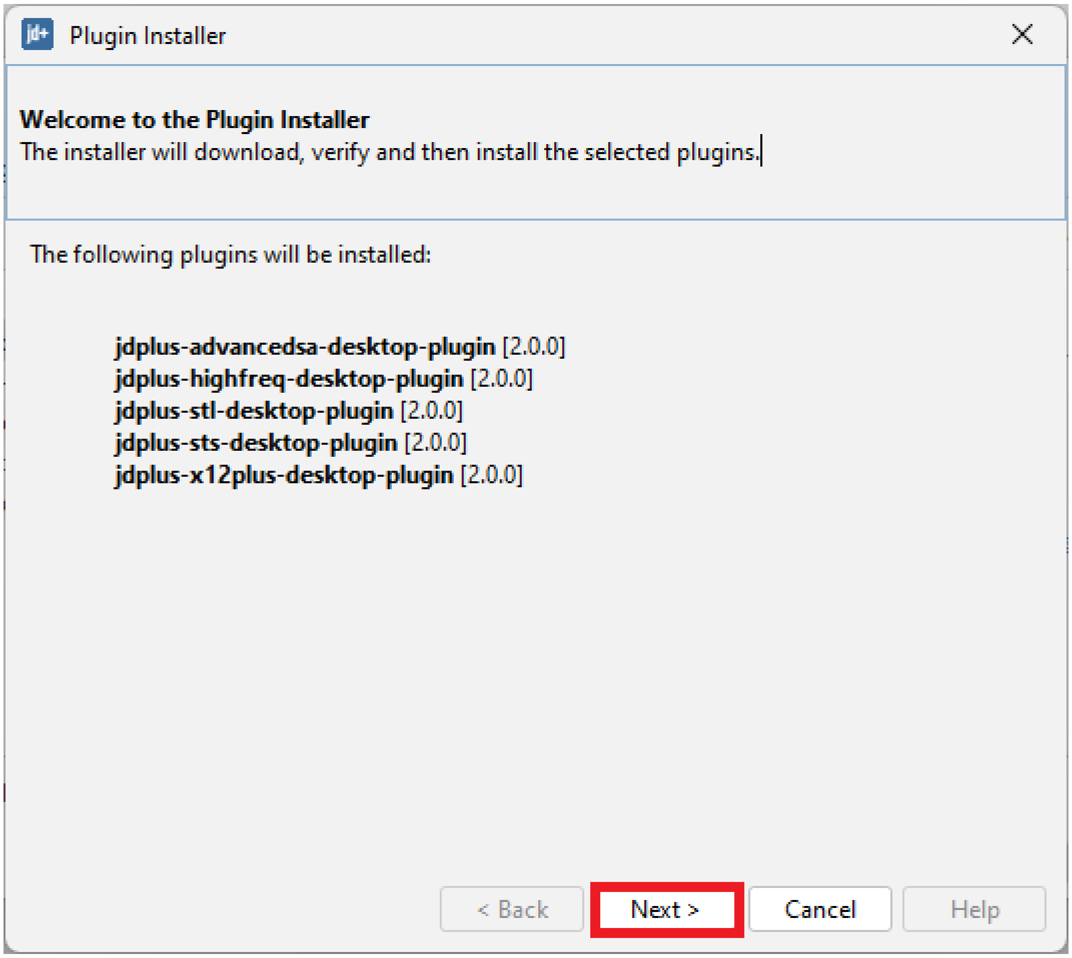


*

A downloaded plugin

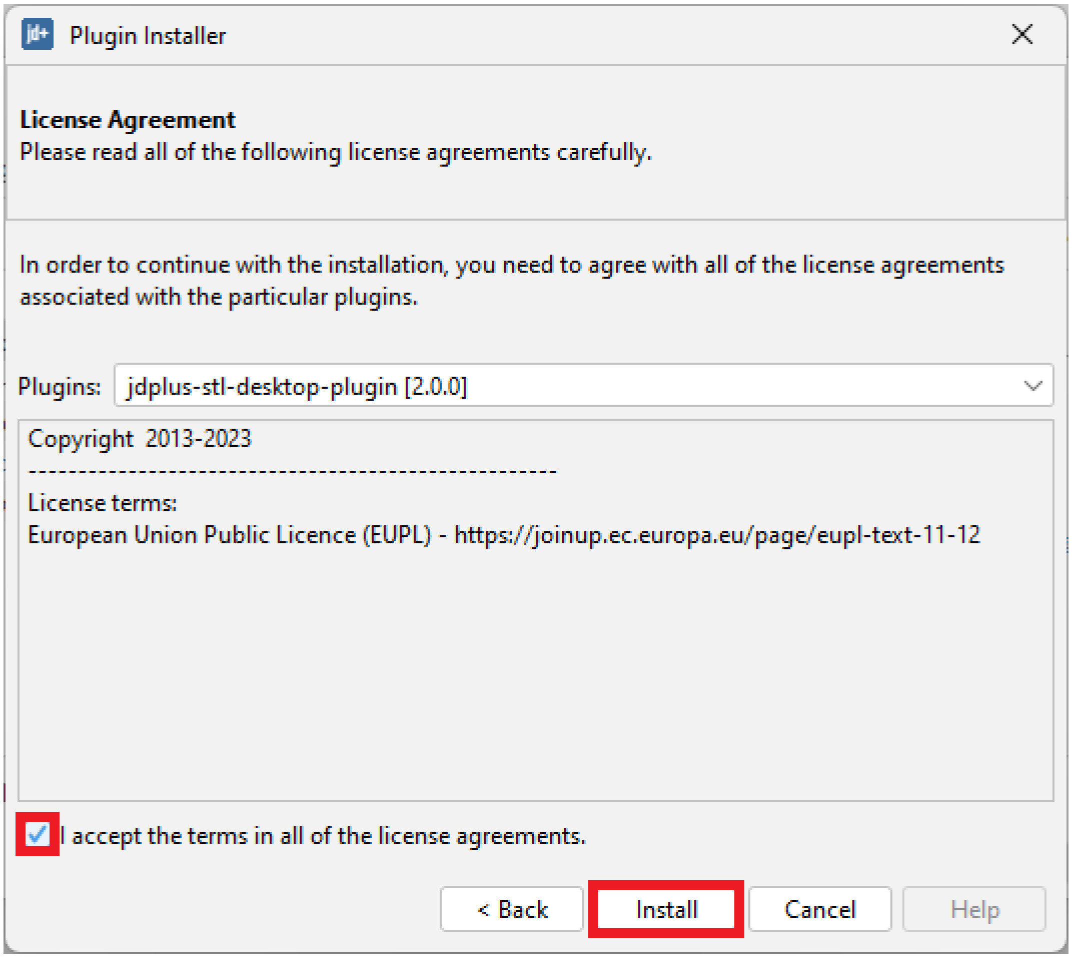
Click on it and choose the **Install** button to start the installation procedure.

There is a wizard that allows the user to install the marked plugin(s). In the first step choose **Next** to continue or **Cancel** to terminate the process.



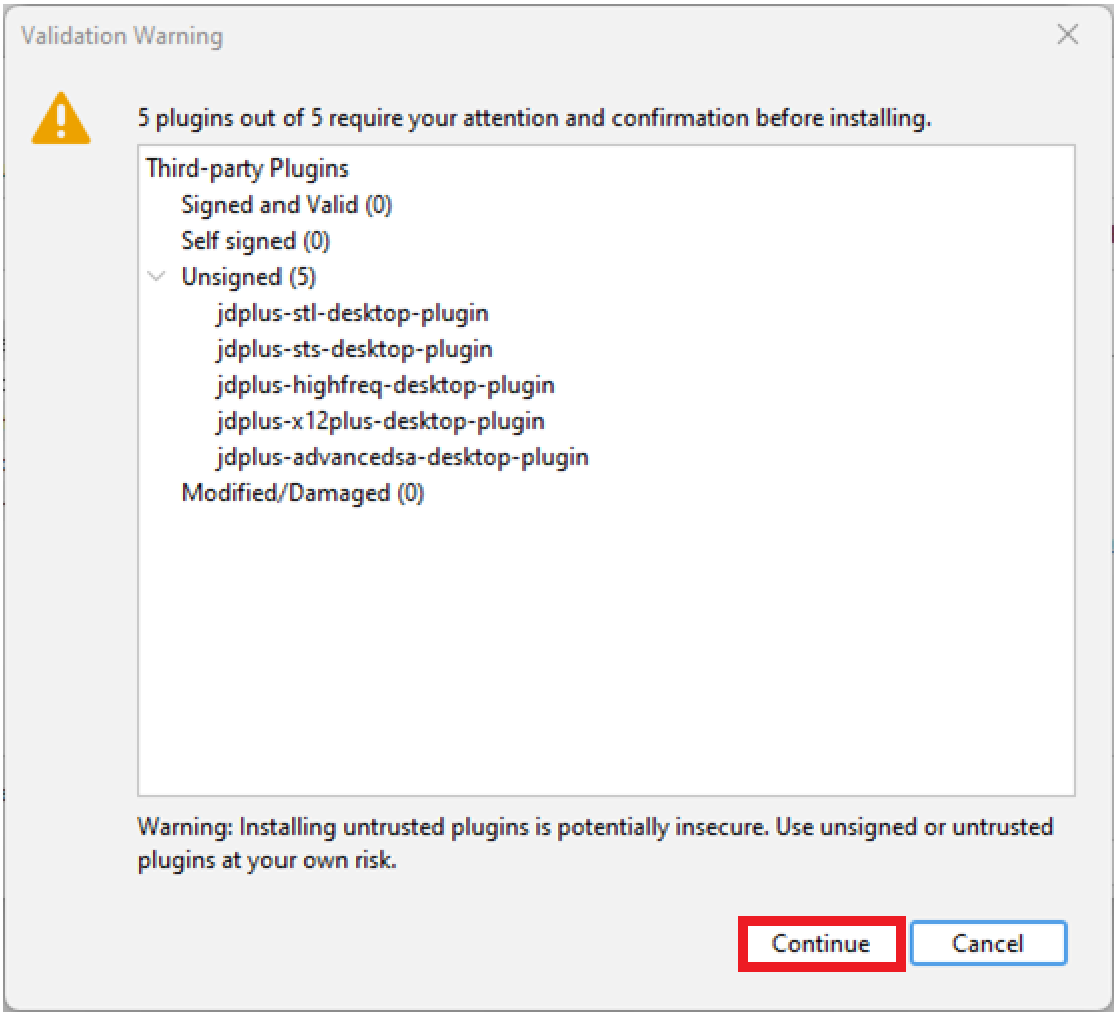
Installation wizard window

Next, mark the terms of agreements and choose **Install**.



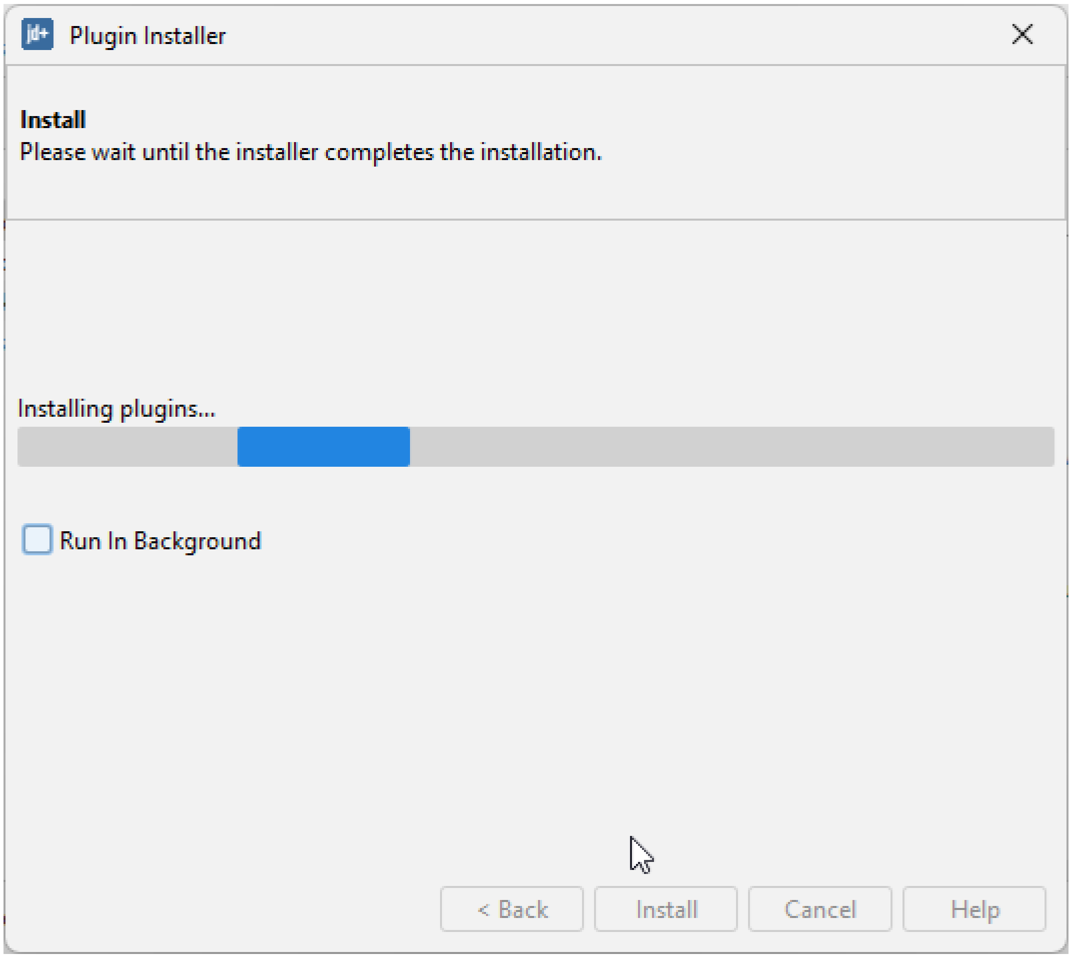
Initiating installation process

If you encounter a warning with the plungins **unsigned**. It's normal, you can click on **Continue**



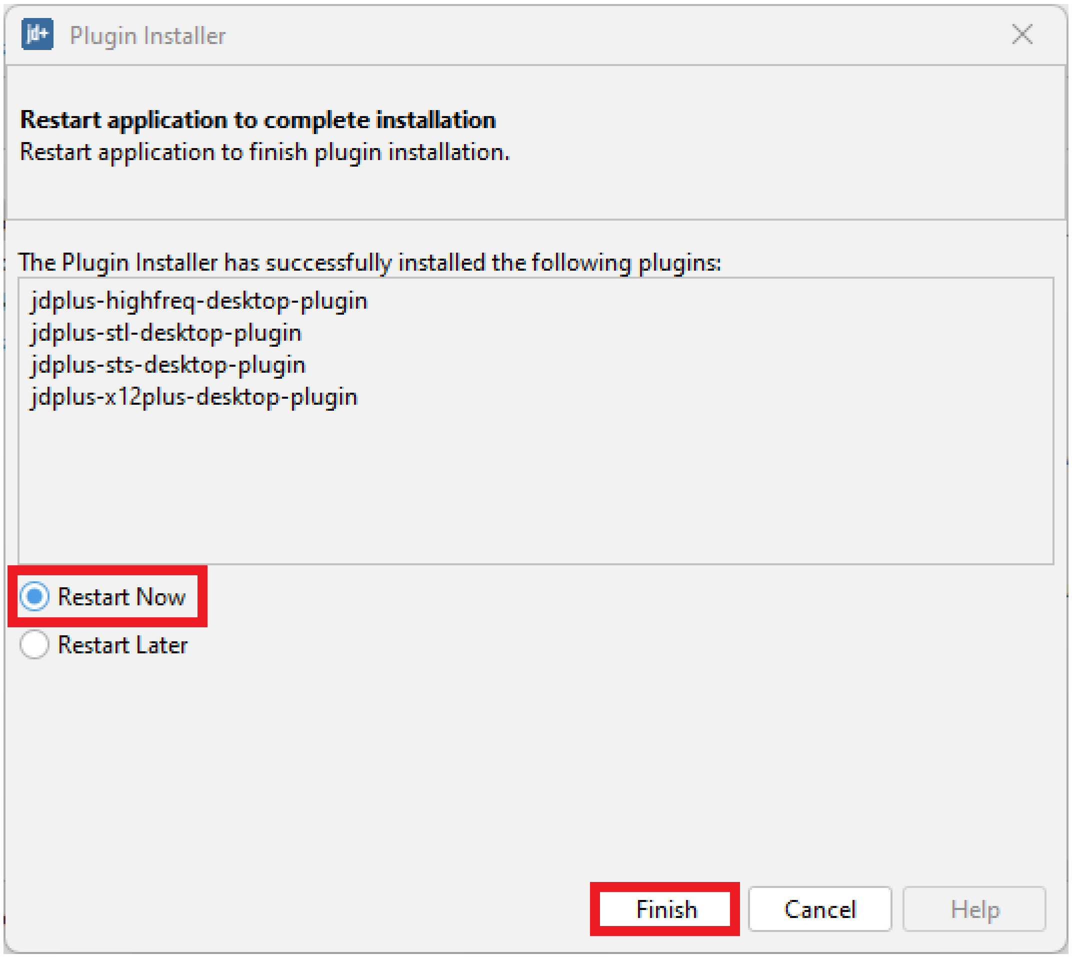
Warning with the unsigned plungins

Then the process is started.



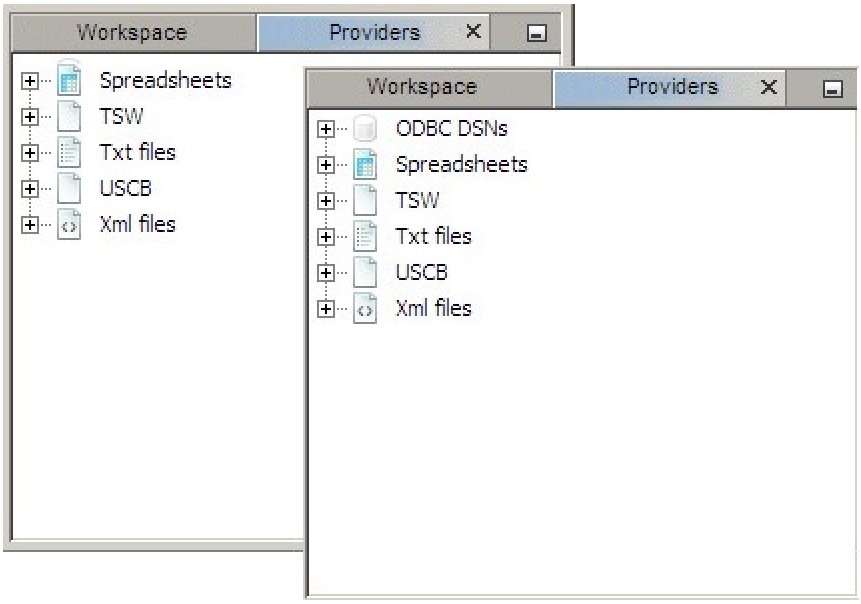
Installation in progress

After a while JDemetra+ will provide an update in the installation process. Click **Finish** to close the window and restart JDemetra+.



Installation completed

Once the process is finished, the newly installed plugin is automatically integrated within the software. The picture below compares the view of the *Workspace* window before (on the left) and after (on the right) the installation of the NbDemetra-ODBC plugin.



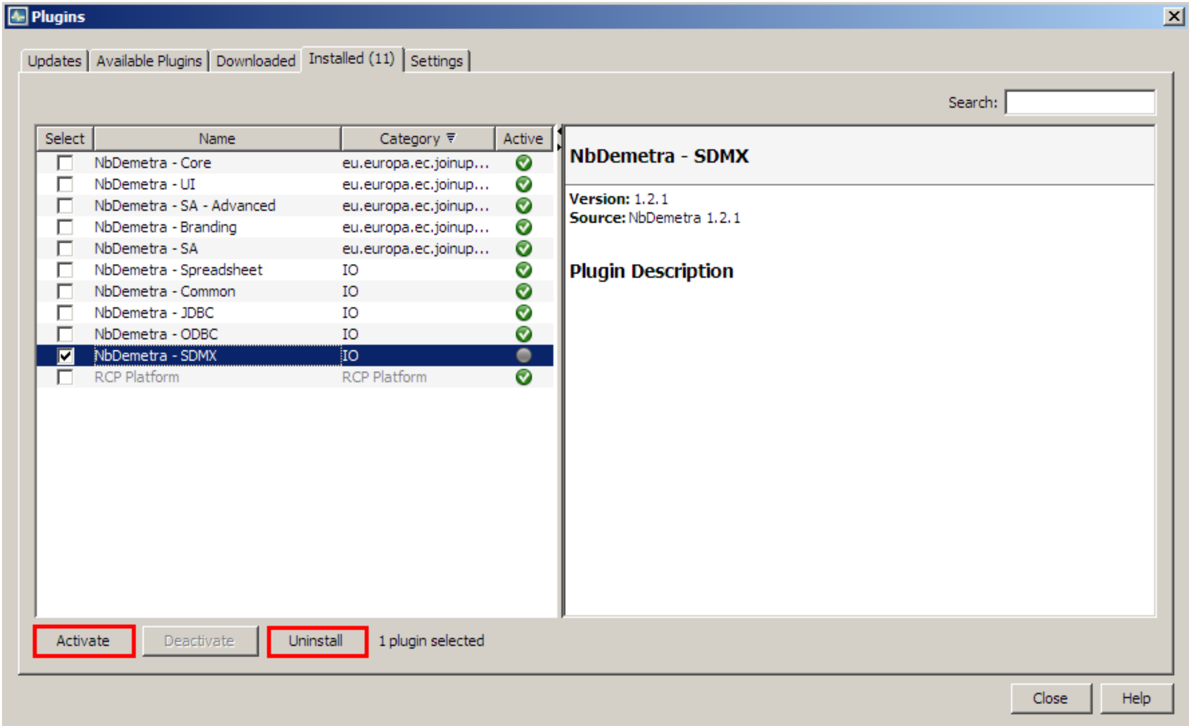
The impact of the plugin on the interface

The list of all installed plugins is displayed in the fourth panel. To modify the current settings mark the plugin (by clicking the checkbox in the *Select* column) and chose an action.

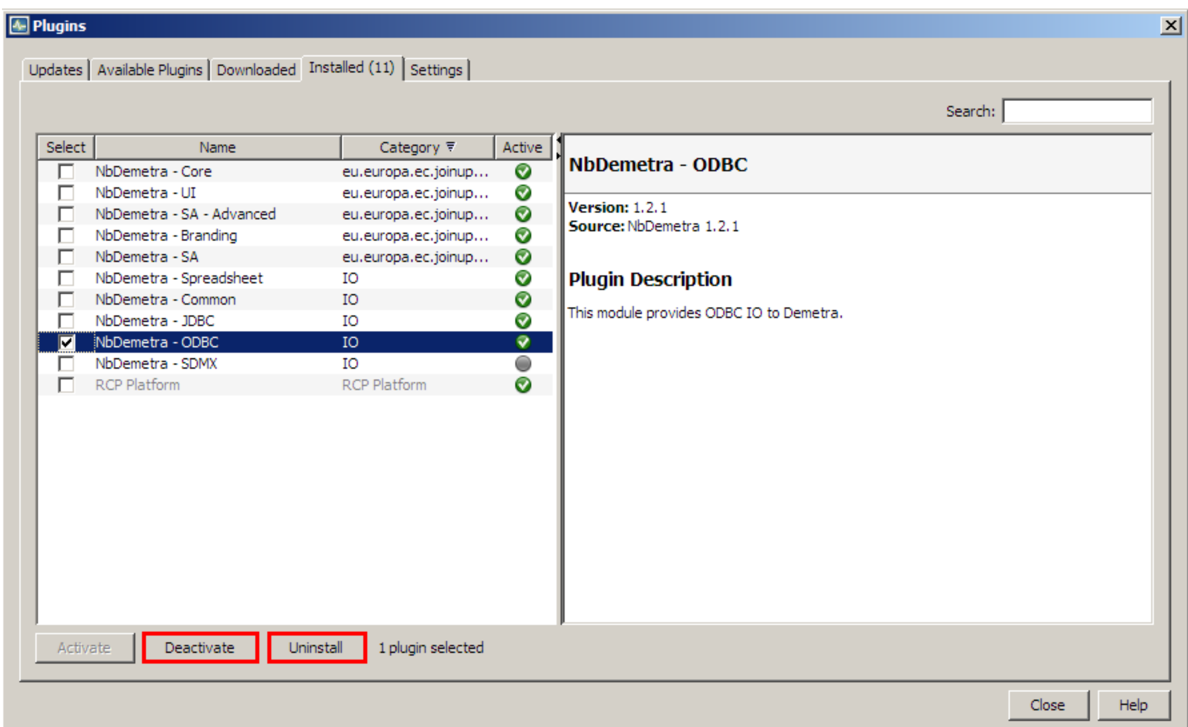
The following options are available:

- **Activate** – activates the marked plugin if it is currently inactive. The option is available for inactive plugins (see the picture below);
- **Deactivate** – deactivates the marked plugin if it is currently active. The option is available for active plugins (see the picture below);
- **Uninstall** – uninstalls the marked plugin.

Inactive plugins can be activated or uninstalled.

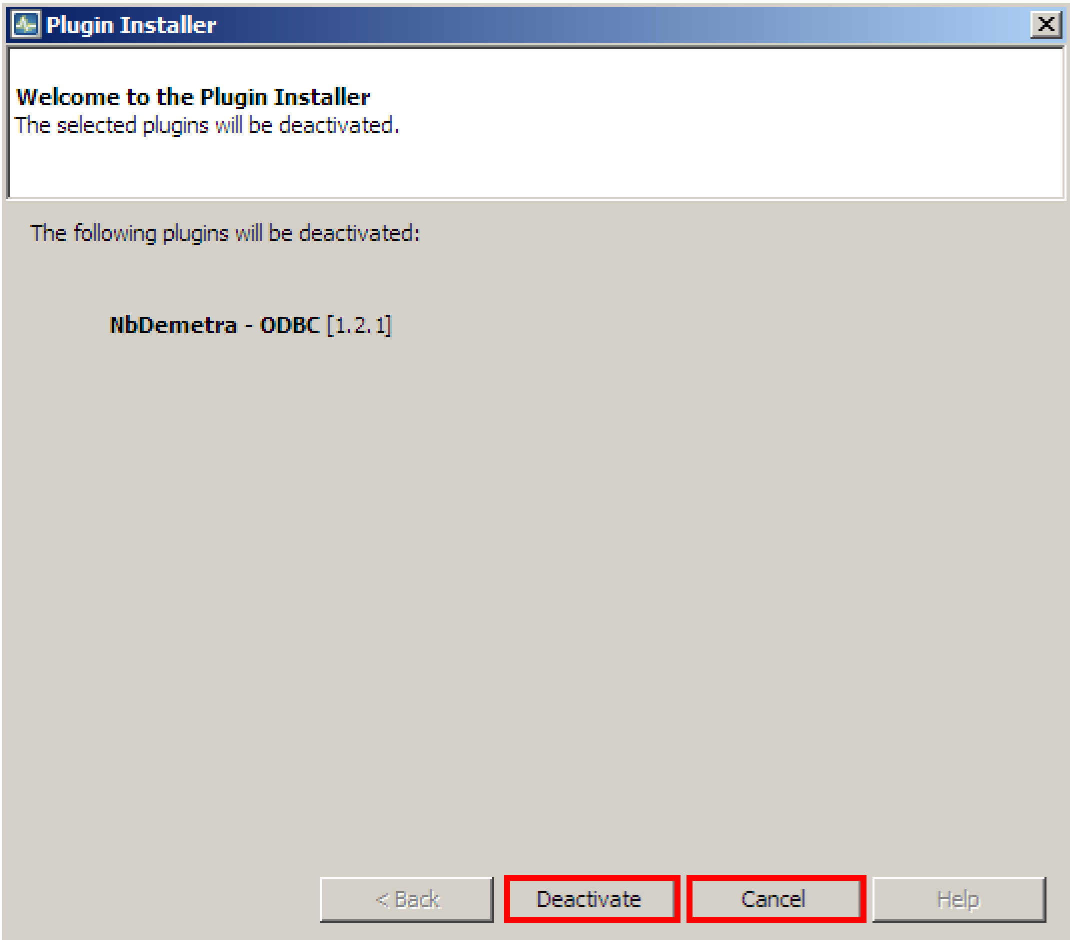


Active plugins can be deactivated or uninstalled



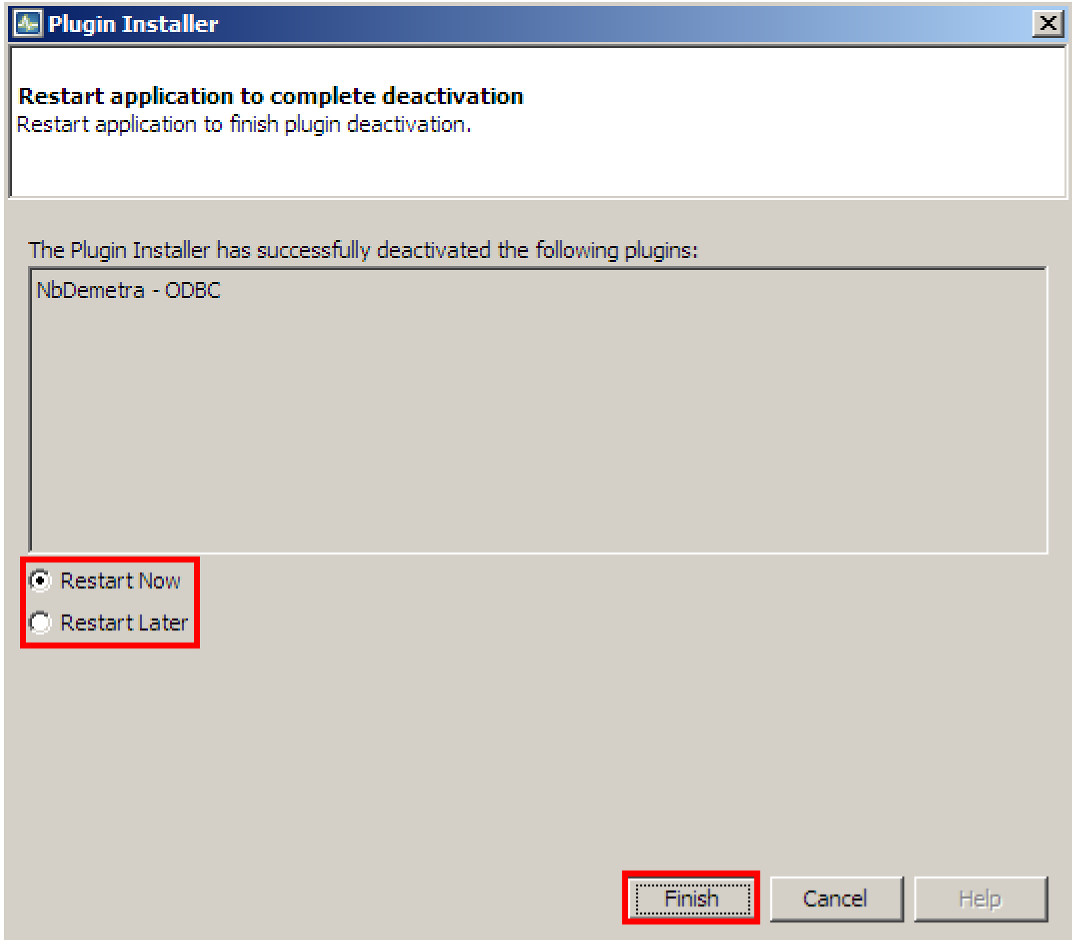
List of plugins – deactivation

There is a wizard that allows the user to activate/deactivate/uninstall the marked plugin(s). The example below illustrates the deactivation process. In the first step the user is expected to confirm or cancel the deactivation.



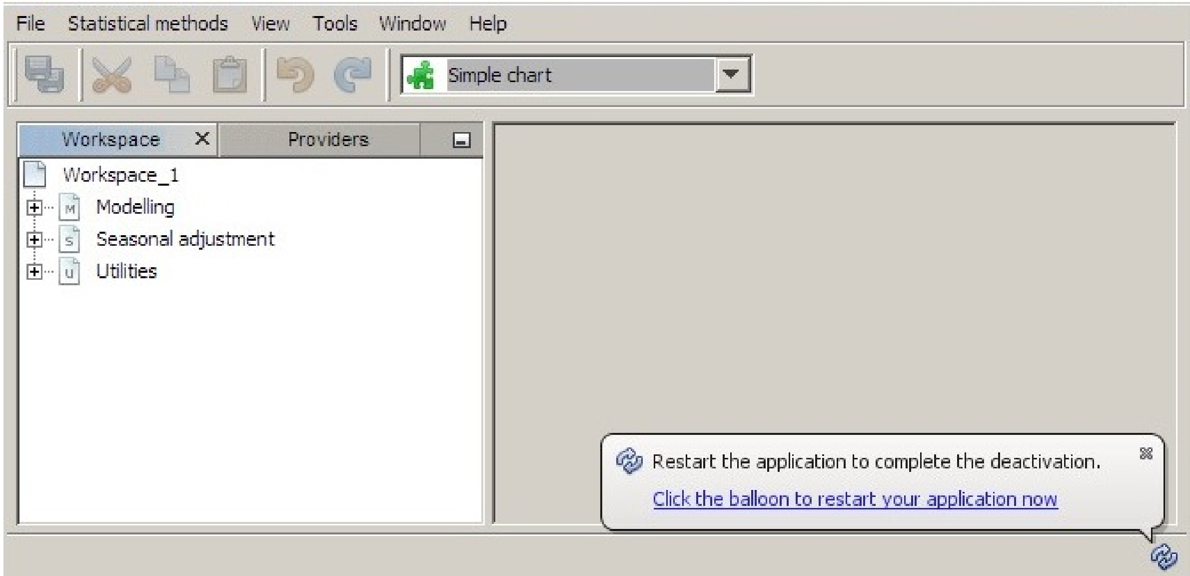
Plugin's deactivation process

In the second step the user should decide if the software will be restarted immediately after the uninstallation is completed or not.



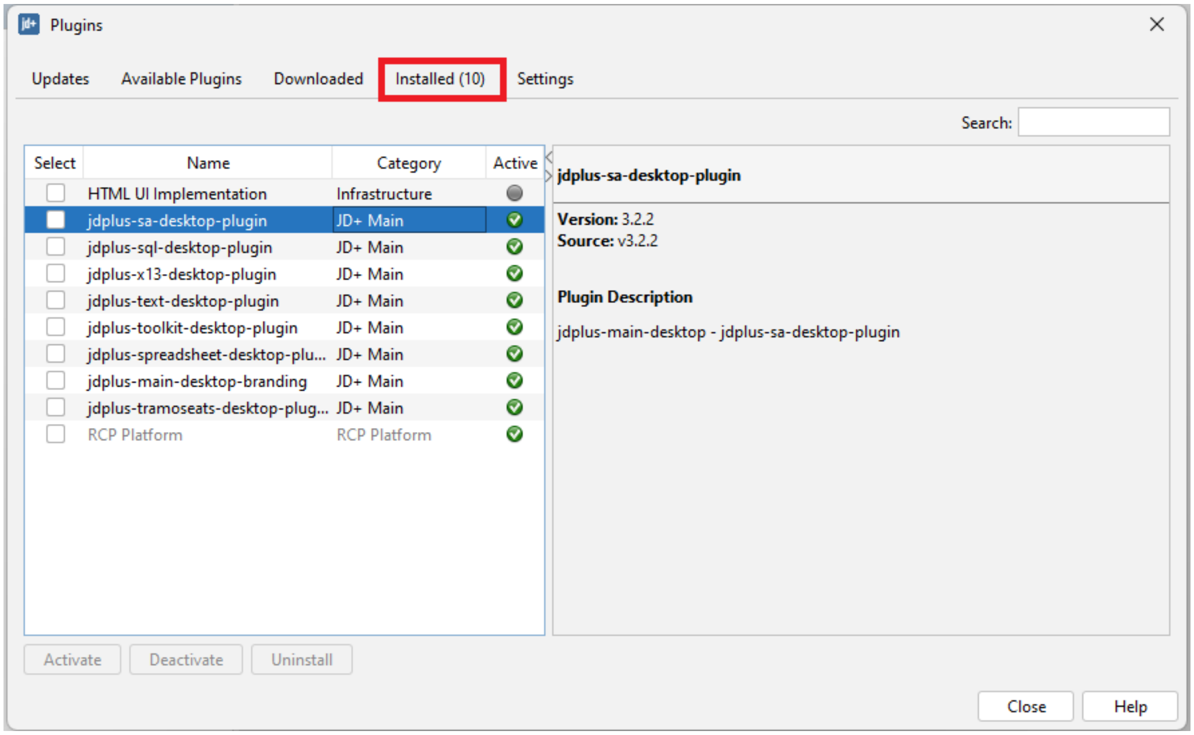
The final step of the installation procedure

It is possible to delay the restart of the application, although the restart is necessary to complete the process.


File Statistical methods View Tools Window Help
Simple chart
Workspace
Providers
Workspace_1
Modelling
Seasonal adjustment
Utilities
Restart the application to complete the deactivation.
Click the balloon to restart your application now

Default Plugins

Default Plugins in v 3.x



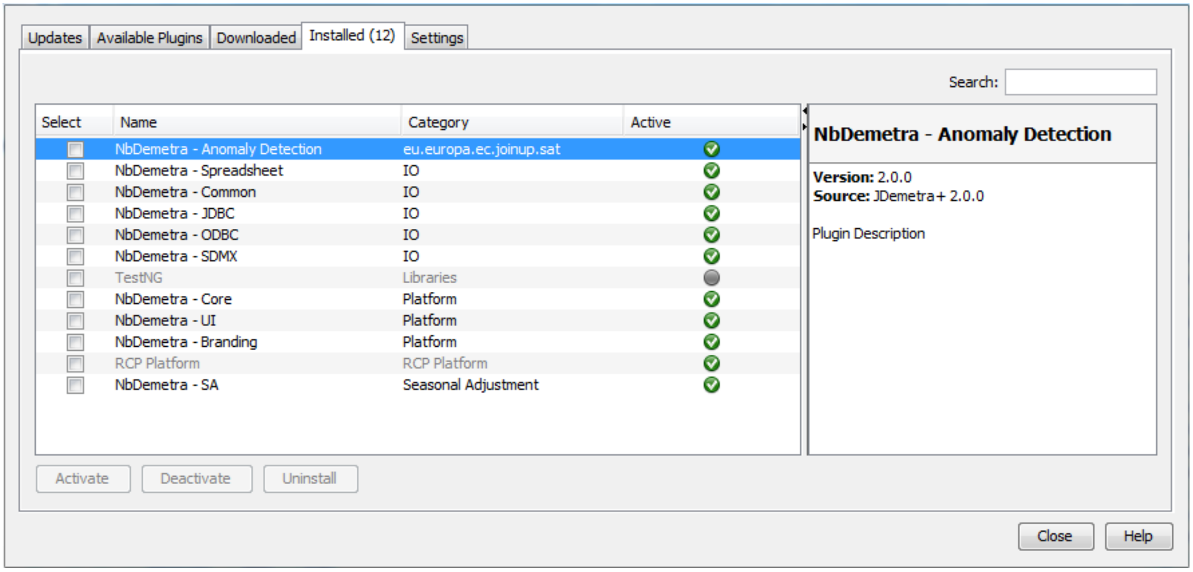
Plug-ins-default-v3.png

Default Plugins in v 2.2.4

Name	Category	Description
NbDemetra – Anomaly detection	SA core algorithms	Identification of outliers
NbDemetra – Spreadsheet	IO (Input/output)	Time series providers for spreadsheet (Excel, OpenOffice)
NbDemetra – Common	IO (Input/output)	Common time series providers, like `.xml` and `.txt`

Name	Category	Description
NbDemetra – JDBC	IO (Input/output)	Time series provider for the JDBC sources
NbDemetra – ODBC	IO (Input/output)	Time series provider for the ODBC sources
NbDemetra – SDMX	IO (Input/output)	Time series provider for SDMX files
NbDemetra – Core	SA core algorithms	Encapsulation of the core algorithms
NbDemetra – UI	SA core algorithms	Basic graphical components
NbDemetra – Branding	SA core algorithms	
NbDemetra – SA	SA core algorithms	Default SA framework, including Tramo-Seats and X-13ARIMA-Seats. This implementation can lead to small differences in comparison with the original programs.

This list is displayed in the *Installed* panel. This panel is available from the *Plugin* functionality and it is activated from the *Tools* menu.


Updates
Available Plugins
Downloaded
Installed (12)
Settings
Search:
Select
Name
Category
Active
NbDemetra - Anomaly Detection
eu.europa.ec.joinup.sat
NbDemetra - Spreadsheet
IO
NbDemetra - Common
IO
NbDemetra - JDBC
IO
NbDemetra - ODBC
IO
NbDemetra - SDMX
IO
TestNG
Libraries
NbDemetra - Core
Platform
NbDemetra - UI
Platform
NbDemetra - Branding
Platform
RCP Platform
RCP Platform
NbDemetra - SA
Seasonal Adjustment
NbDemetra - Anomaly Detection
Version: 2.0.0
Source: JDemetra+ 2.0.0
Plugin Description
Activate
Deactivate
Uninstall
Close
Help

Cruncher and quality report

In this chapter

The sections below describe how to

- automate a Seasonal adjustment estimation process
- update a workspace when new data is available
- export output (series, diagnostics, parameters)
- generate a quality report usable for selective editing (manual fine tuning)

Automate estimation with the cruncher

The cruncher is an additional "executable" module. It can be launched via R or SAS for example.

Objectives of the cruncher:

- update a JDemetra+ workspace (with a selected revision policy)
- export the results (series,diagnostics and parameters)

without having to open the graphical interface and operate manually. Suitable for a production process.

Installation procedure

- Download the cruncher

Available here https://github.com/jdemetra/jwsacruncher/releases

Click on the zip code line of the latest release

- Unzip locally (or on server)

Help pages

Documentation is available here or click on the wiki icon on the GitHub page https://github.com/jdemetra/jwsacruncher/wiki

Running the cruncher in R

Two R packages are currently available

- rjwsacruncher (on CRAN): workspace update and output production
- JDCruncheR (on CRAN): adds a quality report

Installation

```
install.packages(c("rjwsacruncher", "JDCruncheR"))
```

Loading

```
library("rjwsacruncher")
library("JDCruncheR")
```

Connecting the Cruncher module

To connect the cruncher to the R package, the path to the bin directory containing the **cruncher.bat** file must be specified. This directory is available once the zip file has been unzipped. For example:

```
options(cruncher_bin_directory = "C:/Software/jwsacruncher-3.4.0/jdemetra-cli-3.4.0/bin")
```

Updating a workspace with rjwsacruncher

Running estimations

General context: two use cases

- Run first or complete estimation of seasonally adjusted series (from raw series and parameters contained in the workspace)
- Apply a revision policy to updated raw series

The function `cruncher_and_param()` allows to do that:

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

To use the documentation, compute `help()` or `?function`:

```
?cruncher_and_param
help(cruncher_and_param)
```

Before running SA estimations, set the export options.

Additional options

The function `cruncher_and_param()` calls the cruncher with the param file created by the function `create_param_file()`. So you can add arguments to this function according to the needs of your workspace.

For example:

- `paths_path` to add relative paths
- `v3` to specify if your workspace was created in v3
- others arguments you can find here

Configuring output options

After updating the workspace with the selected revision policies, the cruncher generates output - series (`.csv` files) - diagnostics and parameters (demetra_m.csv file)

These files will be created in the workspace's repository, sub-repository 'Output"

```
path <- "My_Workspace/Output/SAProcessing"
```

Selecting time series to export

```
# returns names of the currently exported series
getOption("default_tsmatrix_series")
# example of setting this option
options(default_tsmatrix_series = c("sa", "sa_f"))
# only seasonally adjusted series ("sa") and its forecasts ("sa_f") will be exported
```

Selecting diagnostics and parameters to exoprt

```
# returns names of the currently exported diagnostics and parameters
getOption("default_matrix_item")
# example of setting this option
options(default_matrix_item = c(
    "likelihood.aic",
    "likelihood.aicc",
    "likelihood.bic",
    "likelihood.bicc"
))
```

Quality Report with JDCruncheR

The JDCruncheR package:

- computes a quality score from the diagnostics produced by JDemetra+
- creates a quality report

Main steps

The three main functions of the package are:

- `extract_QR()` to extract the quality report from the `.csv` file (demetra_m.csv) that contains all JD+ diagnostics;
- `compute_score()` to compute a weighted score based on the diagnostics
- `export_xlsx()` to export the quality report.

```r
# choose the demetra_m.csv file generated by the cruncher
QR <- extract_QR()
QR

?compute_score # to see how the score is calculated (formula)
QR <- compute_score(QR,
    n_contrib_score = 3
)

QR

QR <- sort(QR, decreasing = TRUE, sort_variables = "score")
export_xlsx(QR,
    file_name = "U:/quality_report.xls"
)
```

Piling up results

When working with several workspaces or Seasonal adjustment processings (SAP) within a given workspace, quality reports can be piled up with the function `rbind()` or by creating a mQR_matrix object with the function `mQR_matrix()`

```r
QR1 <- extract_QR()
QR2 <- extract_QR()
mQR <- mQR_matrix(QR1, QR2)
mQR
# naming each object
names(mQR) <- c("report_1", "report_2")
# Equivalent to:
mQR <- mQR_matrix(report_1 = QR1, report_2 = QR2)
mQR
```

```r
# score calculation for all reports
mQR <- compute_score(mQR,
    n_contrib_score = 3
)
export_xlsx(mQR,
    export_dir = "U:/"
)
```

Conditionnal score

Missing values can be ignored and conditions can be set for indicators:

```r
# oos_mse weight reduced to 1 when the other
# indicators are "Bad" ou "Severe"
condition1 <- list(
    indicator = "oos_mse",
    conditions = c(
        "residuals_independency",
        "residuals_homoskedasticity",
        "residuals_normality"
    ),
    conditions_modalities = c("Bad", "Severe")
)
BQ <- compute_score(BQ,
    n_contrib_score = 5,
    conditional_indicator = list(condition1),
    na.rm = TRUE
)
```

Customize the score computation

Practical steps if you want to customize the score computation (see package documentation in R)

- select your indicators of interest
- adjust “good”, “uncertain”, “bad”, “severe”
- by default good=0, uncertain=1, bad or severe=3

List of exportable series

Some available output series will be different when using X-13-ARIMA or Tramo-Seats.

List of exportable diagnostics and parameters

Some parameters and available diagnostics will be different when using X-13-ARIMA or Tramo-Seats.

```
options(
    default_matrix_item =
        c(
            "period", "span.start", "span.end", "span.n", "span.missing",
            "espan.start", "espan.end", "espan.n", "log", "adjust", "regression.lp",
            "regression.ntd", "regression.nmh", "regression.td-derived",
            "regression.td-ftest", "regression.easter", "regression.nout",
            "regression.noutao", "regression.noutls", "regression.nouttc",
            "regression.noutso", "regression.td(*):4", "regression.out(*)",
            "regression.user(*)", "likelihood.neffectiveobs", "likelihood.np",
            "likelihood.logvalue", "likelihood.adjustedlogvalue", "likelihood.ssqerr",
            "likelihood.aic", "likelihood.aicc", "likelihood.bic", "likelihood.bicc",
            "residuals.ser", "residuals.ser-ml", "residuals.mean", "residuals.skewness:3",
            "residuals.kurtosis:3", "residuals.dh", "residuals.lb", "residuals.lb2:3",
            "residuals.seaslb", "residuals.bp", "residuals.bp2", "residuals.seasbp",
            "residuals.nudruns", "residuals.ludruns", "residuals.nruns",
            "residuals.lruns", "arima", "arima.mean", "arima.p", "arima.d",
            "arima.q", "arima.bp", "arima.bd", "arima.bq", "arima.phi(*)",
            "arima.bphi(*)", "arima.th(*)", "arima.bth(*)", "decomposition.seasonality",
            "decomposition.parameters_cutoff", "decomposition.model_changed",
            "decomposition.tvar-estimator", "decomposition.tvar-estimate",
            "decomposition.tvar-pvalue", "decomposition.savar-estimator",
            "decomposition.savar-estimate", "decomposition.savar-pvalue",
            "decomposition.svar-estimator", "decomposition.svar-estimate",
            "decomposition.svar-pvalue", "decomposition.ivar-estimator",
            "decomposition.ivar-estimate", "decomposition.ivar-pvalue", "decomposition.tsc
            "decomposition.tscorr-estimate", "decomposition.tscorr-pvalue",
            "decomposition.ticorr-estimator", "decomposition.ticorr-estimate",
            "decomposition.ticorr-pvalue", "decomposition.sicorr-estimator",
```

```
"decomposition.sicorr-estimate", "decomposition.sicorr-pvalue",
"decomposition.ar_root(*)", "decomposition.ma_root(*)", "method",
"variancedecomposition.cycle", "variancedecomposition.seasonality",
"variancedecomposition.irregular", "variancedecomposition.tdh",
"variancedecomposition.others", "variancedecomposition.total",
"diagnostics.logstat", "diagnostics.levelstat", "diagnostics.fcast-insample-me
"diagnostics.fcast-outsample-mean", "diagnostics.fcast-outsample-variance",
"diagnostics.seas-lin-f", "diagnostics.seas-lin-qs", "diagnostics.seas-lin-kw"
"diagnostics.seas-lin-friedman", "diagnostics.seas-lin-periodogram",
"diagnostics.seas-lin-spectralpeaks", "diagnostics.seas-si-combined",
"diagnostics.seas-si-evolutive", "diagnostics.seas-si-stable",
"diagnostics.seas-res-f", "diagnostics.seas-res-qs", "diagnostics.seas-res-kw"
"diagnostics.seas-res-friedman", "diagnostics.seas-res-periodogram",
"diagnostics.seas-res-spectralpeaks", "diagnostics.seas-res-combined",
"diagnostics.seas-res-combined3", "diagnostics.seas-res-evolutive",
"diagnostics.seas-res-stable", "diagnostics.seas-i-f", "diagnostics.seas-i-qs"
"diagnostics.seas-i-kw", "diagnostics.seas-i-periodogram", "diagnostics.seas-i
"diagnostics.seas-i-combined", "diagnostics.seas-i-combined3",
"diagnostics.seas-i-evolutive", "diagnostics.seas-i-stable",
"diagnostics.seas-sa-f", "diagnostics.seas-sa-qs", "diagnostics.seas-sa-kw",
"diagnostics.seas-sa-friedman", "diagnostics.seas-sa-periodogram",
"diagnostics.seas-sa-spectralpeaks", "diagnostics.seas-sa-combined",
"diagnostics.seas-sa-combined3", "diagnostics.seas-sa-evolutive",
"diagnostics.seas-sa-stable", "diagnostics.seas-sa-ac1", "diagnostics.td-sa-al
"diagnostics.td-sa-last", "diagnostics.td-i-all", "diagnostics.td-i-last",
"diagnostics.td-res-all", "diagnostics.td-res-last", "diagnostics.ic-ratio-hen
"diagnostics.ic-ratio", "diagnostics.msr-global", "diagnostics.msr(*)",
"decomposition.trendfilter", "decomposition.seasfilter", "m-statistics.m1",
"m-statistics.m2", "m-statistics.m3", "m-statistics.m4", "m-statistics.m5",
"m-statistics.m6", "m-statistics.m7", "m-statistics.m8", "m-statistics.m9",
"m-statistics.m10", "m-statistics.m11", "m-statistics.q", "m-statistics.q-m2",
"diagnostics.basic checks.definition:2", "diagnostics.basic checks.annual tota
"diagnostics.visual spectral analysis.spectral seas peaks", "diagnostics.visua
"diagnostics.regarima residuals.normality:2", "diagnostics.regarima residuals.
"diagnostics.regarima residuals.spectral td peaks:2", "diagnostics.regarima re
"diagnostics.outliers.number of outliers:2", "diagnostics.out-of-sample.mean:2
"diagnostics.out-of-sample.mse:2", "diagnostics.m-statistics.q:2",
"diagnostics.m-statistics.q-m2:2", "diagnostics.seats.seas variance:2",
"diagnostics.seats.irregular variance:2", "diagnostics.seats.seas/irr cross-co
"diagnostics.residual seasonality tests.qs test on sa:2", "diagnostics.residua
"diagnostics.residual seasonality tests.f-test on sa (seasonal dummies):2",
```

```
            "diagnostics.residual seasonality tests.f-test on i (seasonal dummies):2",
            "diagnostics.combined seasonality test.combined seasonality test on sa:2",
            "diagnostics.combined seasonality test.combined seasonality test on sa (last 3
            "diagnostics.combined seasonality test.combined seasonality test on irregular:
            "diagnostics.residual trading days tests.f-test on sa (td):2",
            "diagnostics.residual trading days tests.f-test on i (td):2",
            "diagnostics.quality"
        )
)
```

R packages

In this chapter

Core JDemetra+ Java algorithms can be accessed in R. This chapter provides an overview of the suite of R packages related to JDemetra+: rjdverse. R Help pages, Readme pages and vignettes relative to each package provide a detailed description of all available functions: objectives, arguments, output and examples. The corresponding GitHub pages are directly linked to this documentation whenever their use in a selected algorithm is mentioned. More details on specific functions are available in the relevant chapters in the Algorithm part

Installing Packages

All R packages related to JDemetra+ (v2 or v3) are available from the rjdverse Github page.

They can be installed from - GitHub - r-universe - CRAN (for some of them)

As explained in the corresponding Readme files on Github.

Configuration needed

- To run packages based on version 3.x: Java 17 or higher: how to get a portable Java and link it with R on your computer is explained here
- To run packages based on version 2.x: Java 8 or higher

Algorithms available in R

Seasonal adjustment

Using JDemetra+ version 2.x

Algorithm	Package	Comments
X-13-ARIMA	RJDemetra	Reg-ARIMA and X-11 decomposition available independently
Tramo-Seats	RJDemetra	Tramo available independently

Using JDemetra+ version 3.x

Algorithm	Package	Comments
X-13-ARIMA	rjd3x13	Reg-ARIMA and X-11 decomposition available independently
Extended X-11	rjd3x11plus	Extended for high-frequency (infra-monthly) data and Trend estimation with local polynomial filters
Tramo-Seats	rjd3tramoseats	Tramo available independently
Extended Tramo	rjd3highfreq	Extended for high-frequency data
Extended Seats	rjd3highfreq	Extended for high-frequency data
STL+	rjd3stl	Airline based preadjustment and high-frequency data extension
Basic Structural Models	rjd3sts	State space framework

Version 3.x includes Revision Policies in X-13 and Tramo-Seats.

More details on functions parameters and retrieving output in the chapter dedicated to Seasonal Adjustment

Filtering and Trend estimation

Algorithm	Package
Moving average functions	rjd3filters
Local Polynomial Trend Estimation	rjd3filters, rjd3x11plus

Benchmarking and Temporal disaggregation

Algorithm	Package
Denton	rjd3bench
Cholette	rjd3bench
Cubic splines	rjd3bench
Temporal Disaggregation	rjd3bench

Utility functions available in R

The packages listed below contain utility functions useful when running a production process with massive datasets.

Running the cruncher and generating a quality report

JDemetra+ cruncher is an executable module designed for mass production of seasonally adjusted series .

Package	JD+ version	Comments
rjwsacruncher	2.x	estimation update and output
JDCruncheR	2.x	all the above + Quality Report

Wrangling JD+ workspaces

A workspace is a specific JDemetra+ data format (`.xml` files) allowing to use the graphical user interface (GUI) and the cruncher.

Package	JD+ version	Comments
rjdworkspace	2.x	update meta data, set specifications, merge workspaces
rjd3providers	3.x	update metadata
rjd3workspace	3.x	set specifications, merge workspace

Generating enhanced output in SA estimation

This additional packages produce enhanced plots and diagnostic outputs.

Package	JD+ version	Comments
rjdmarkdown	2.x	enhanced print of diagnostics
ggdemetra	3.x	plots based on ggplot
ggdemetra3	3.x	plots based on ggplot
rjdqa	2.x	visual dashboard on one series

General structure

The R object resulting from an estimation is a list of lists containing raw data, parameters, output series and diagnostics.

RJDemetra output structure

Organised by domain:

To retrieve any element just navigate this list of lists.

rjd3x13 output structure

Results and specification are separated first and then organised by domain.

```
sa_x13_v3 <- RJDemetra::x13(y_raw, spec = "RSA5")
sa_x13_v3$result
sa_x13_v3$estimation_spec
sa_x13_v3$result_spec
```

```
SA
├─ regarima (≠ X-13 and TRAMO-SEAT)
│  ├─ specification
│  └─ ...
├─ decomposition (≠ X-13 and TRAMO-SEAT)
│  ├─ specification
│  └─ ...
├─ final
│  ├─ series
│  └─ forecasts
├─ diagnostics
│  ├─ variance_decomposition
│  ├─ combined_test
│  └─ ...
└─ user_defined
```

Figure 257: V2 SA structure

```
sa_x13_v3$user_defined
```

To retrieve any element just navigate this list of lists.

rjd3 suite of packages: overview

The sections below provide an overview of each package based on version 3.x of JDemetra+. For detailed description refer to the package's own R Readme file and documentation pages as linked below.

rjd3toolkit

Contains utility functions used in other `rjd3` packages and has to be systematically installed before using any other rjd3 package. From a user point of view, it allows to:

- customize specifications in rjd3x13 and rjd3tramoseats
- generate user-defined regressors for calendar correction
- generate auxiliary variables (outliers, ramps..)
- run ARIMA model estimations

- perform tests (seasonality, normality, white noise)
- access general functions such as autocorrelations, distributions

Documentation here

rjd3x13

`rjd3x13` gives access to X-13-ARIMA seasonal adjustment algorithm.

- Specification: created with `spec_x11_default()`, `spec_x13_default()`, `spec_regarima_default()` and customized with `rjd3toolkit` functions + `set_x11()`
- Apply model with `x11()`, `x13()`, `fast.x13()`, `regarima()`, `fast.regarima()`
- Refresh policies: `regarima.refresh()` and `x13.refresh()`

Documentation here

rjd3tramoseats

`rjd3tramoseats` gives access to Tramo-Seats seasonal adjustment algorithm.

- Specification: created with `spec_tramoseats_default()`, `spec_tramo_default()` and customized with `rjd3toolkit` functions + `set_seats()`
- Apply model with `tramoseats()`, `fast.tramoseats()`, `tramo()`, `fast.tramo()`
- Refresh policies: `tramo.refresh()` and `tramoseats.refresh()`

Documentation here

rjd3sts

Gives access to structural time series and state space models.

Documentation here

rjd3highfreq

Depends on `rjd3sts`

Seasonal adjustment of high frequency (infra-monthly) data:

- fractional airline based reg-ARIMA pre-adjustment
- fractional and multi airline decomposition

Documentation here

rjd3filters

The rjd3filters package allows to:

- easily create/combine/apply moving averages `moving_average()` (much more general than `stats::filter()`) and study their properties: plot coefficients (`plot_coef()`), gain (`plot_gain()`), phase-shift (`plot_phase()`) and different statics (`diagnostic_matrix()`)
- trend-cycle extraction with different methods to treat endpoints:
- `lp_filter()` local polynomial filters of Proietti and Luati (2008) (including Musgrave): Henderson, Uniform, biweight, Trapezoidal, Triweight, Tricube, "Gaussian", Triangular, Parabolic (= Epanechnikov)

- `rkhs_filter()` Reproducing Kernel Hilbert Space (RKHS) of Dagum and Bianconcini (2008) with same kernels

- `fst_filter()` FST approach of Grun-Rehomme, Guggemos, and Ladiray (2018)

- `dfa_filter()` derivation of AST approach of Wildi and McElroy (2019)
- change the filter used in X-11 for TC extraction

Create moving averages

```
library("rjd3filters")

m1 <- moving_average(rep(1, 3), lags = 1)
m1 # Forward MA
m2 <- moving_average(rep(1, 3), lags = -1)
m2 # centred MA

m1 + m2
m1 - m2
m1 * m2
```

Can be used to create all the MA of X-11:

```
e1 <- moving_average(rep(1, 12), lags = -6)
e1 <- e1 / sum(e1)
e2 <- moving_average(rep(1 / 12, 12), lags = -5)

# used to have the 1rst estimate of the trend
tc_1 <- M2X12 <- (e1 + e2) / 2
coef(M2X12) |> round(3)
si_1 <- 1 - tc_1
M3 <- moving_average(rep(1 / 3, 3), lags = -1)
M3X3 <- M3 * M3

# M3X3 moving average applied to each month
coef(M3X3) |> round(3)
M3X3_seasonal <- to_seasonal(M3X3, 12)
coef(M3X3_seasonal) |> round(3)
s_1 <- M3X3_seasonal * si_1
s_1_norm <- (1 - M2X12) * s_1
sa_1 <- 1 - s_1_norm
henderson_mm <- moving_average(lp_filter(horizon = 6)$filters.coef[, "q=6"],
    lags = -6
)
tc_2 <- henderson_mm * sa_1
si_2 <- 1 - tc_2
M5 <- moving_average(rep(1 / 5, 5), lags = -2)
M5X5_seasonal <- to_seasonal(M5 * M5, 12)
s_2 <- M5X5_seasonal * si_2
```

```r
s_2_norm <- (1 - M2X12) * s_2
sa_2 <- 1 - s_2_norm
tc_f <- henderson_mm * sa_2
```

```r
par(mai = c(0.3, 0.3, 0.2, 0))
layout(matrix(c(1, 1, 2, 3), 2, 2, byrow = TRUE))

plot_coef(tc_f)
plot_coef(sa_2, col = "orange", add = TRUE)
legend("topleft",
    legend = c("Final TC filter", "Final SA filter"),
    col = c("black", "orange"), lty = 1
)

plot_gain(tc_f)
plot_gain(sa_2, col = "orange", add = TRUE)

plot_phase(tc_f)
plot_phase(sa_2, col = "orange", add = TRUE)
```

Apply a moving average

```r
y <- retailsa$AllOtherGenMerchandiseStores
trend <- y * tc_1
sa <- y * sa_1
plot(window(ts.union(y, trend, sa), start = 2000),
    plot.type = "single",
    col = c("black", "orange", "lightblue")
)
```

rjd3x11plus

Depends on `rjd3filters`

- Extension of X-11 decomposition with multiple non integer periodicities
- Trend estimation with local polynomial based filters

Full documentation here

rjd3stl

rjd3stl contains usual STL functions and an airline model based pre-adjustment module. Is also tailored to handle high-frequency data.

Full documentation here

ggdemetra3

ggdemetra3 uses ggplot2 to add seasonal adjustment statistics to your plot (Like ggdemetra but compatible with version 3.x.). Also compatible with high-frequency methods:

```
library("ggdemetra3")

spec <- spec_x13_default("rsa3") |> set_tradingdays(option = "WorkingDays")
p_ipi_fr <- ggplot(data = ipi_c_eu_df, mapping = aes(x = date, y = FR)) +
    geom_line() +
    labs(
        title = "SA - IPI-FR",
        x = NULL, y = NULL
    )
p_sa <- p_ipi_fr +
    geom_sa(
        component = "y_f(12)", linetype = 2,
        spec = spec
    ) +
    geom_sa(component = "sa", color = "red") +
    geom_sa(component = "sa_f", color = "red", linetype = 2)
p_sa
p_sa +
    geom_outlier(
        geom = "label_repel",
        coefficients = TRUE,
        ylim = c(NA, 65), force = 10,
        arrow = arrow(
            length = unit(0.03, "npc"),
            type = "closed", ends = "last"
        ),
        digits = 2
    )
```

rjd3bench

Tailored for Benchmarking and temporal disaggregation

rjd3revisions

Revision analysis, more information here

Part III

Methods

This part provides underlying methodological background on all the algorithms featured in JDemetra+.

Practical guidance on how to use these algorithms can be found here, whereas detailed description of all the available tools allowing to use them can be found in the Tools part of this book.

In this part:

- Spectral analysis tools
- Tests for seasonality and residuals
- Reg-ARIMA modelling
- X-11: moving average based decomposition
- Seats: ARIMA model based decomposition
- STL: Loess based decomposition
- Structural time series and state space framework
- Seasonal Adjustment of High-Frequency Data
- Trend Estimation
- Benchmarking and temporal disaggregation

Spectral Analysis Principles and Tools

In this chapter

This chapter provides some guidance on spectral analysis, which will allow to understand the principle of various spectral analysis tools available in JDemetra+, via Graphical User Interface and R packages.

- explanation of spectral graphs here, but description in GUI chap
- outputs of tests ?
- description of spectral graphs in GUI can be found here

Spectral analysis concepts

A time series x_t with stationary covariance, mean μ and k^{th} autocovariance $E((x_t - \mu)(x_{t-k}\mu)) = \gamma(k)$ can be described as a weighted sum of periodic trigonometric functions: $sin(\omega t)$ and $cos(\omega t)$, where $\omega = \frac{2*pi}{T}$ denotes frequency. Spectral analysis investigates this frequency domain representation of x_t to determine how important cycles of different frequencies are in accounting for the behaviour of x_t.

Assuming that the autocovariances $\gamma(k)$ are absolutely summable($\sum_{k=-\infty}^{\infty} |\gamma(k)| < \infty$), the autocovariance generating function, which summarizes these autocovariances through a scalar valued function, is given by Equation 0.60 (HAMILTON, J.D. (1994)).

$$acgf(z) = \sum_{k=-\infty}^{\infty} z^k \gamma(k) \tag{0.1}$$

where z denotes a complex scalar.

Once the Equation 0.60 is divided by π and evaluated at some $z= e^{-i\omega} = cos\omega - isin\omega$, where $i = \sqrt{-1}$ and ω is a real scalar, $-\infty < \omega < \infty$, the result of this transformation is called a population spectrum $f(\omega)$for x_t, given in Equation 0.61 (HAMILTON, J.D. (1994)).

$$f(\omega) = \frac{1}{\pi} \sum_{k=-\infty}^{\infty} e^{-ik\omega}\gamma(k) \tag{0.2}$$

Therefore, the analysis of the population spectrum in the frequency domain is equivalent to the examination of the autocovariance function in the time domain analysis; however it provides an alternative way of inspecting the process. Because $f(\omega)$dω is interpreted as a contribution to the variance of components with frequencies in the range $(\omega,\ \omega + d\omega)$, a peak in the spectrum indicates an important contribution to the variance at frequencies near the value that corresponds to this peak.

As $e^{-i\omega} = cos\omega - isin\omega$, the spectrum can be also expressed as in Equation 0.62.

$$f(\omega) = \frac{1}{\pi} \sum_{k=-\infty}^{\infty} (cos\omega k - isin\omega k)\gamma(k) \tag{0.3}$$

Equation 0.62 can be presented as:

$$f(\omega) = \frac{1}{\pi}[\ \gamma(0) + 2\sum_{k=1}^{\infty} \gamma(k)cos\omega k] \tag{0.4}$$

This implies that if autocovariances are absolutely summable the population spectrum exists and is a continuous, real-valued function of ω. Due to the properties of trigonometric functions $(\cos(-\omega k) = \cos(\omega k)$ and $\cos(\omega + 2\pi j)k = cos(\omega k))$ the spectrum is a periodic, even function of ω, symmetric around $\omega = 0$. Therefore, the analysis of the spectrum can be reduced to the interval $(-\pi, \pi)$. The spectrum is non-negative for all $\omega \in (-\pi, \pi)$.

The shortest cycle that can be distinguished in a time series lasts two periods. The frequency which corresponds to this cycle is $\omega = \pi$ and is called the Nyquist frequency. The frequency of the longest cycles that can be observed in the time series with n observations is $\omega = \frac{2\pi}{n}$ and is called the fundamental (Fourier) frequency.

Note that if x_t is a white noise process with zero mean and variance σ^2, then for all $|k| > 0$ $\gamma(k) = 0$ and the spectrum of x_t is constant ($f(\omega) = \frac{\sigma^2}{\pi}$) since each frequency in the spectrum contributes equally to the variance of the process (BROCKWELL, P.J., and DAVIS, R.A. (2002)).

The aim of spectral analysis is to determine how important cycles of different frequencies are in accounting for the behaviour of a time series. Since spectral analysis can be used to detect the presence of periodic components, it is a natural diagnostic tool for detecting trading day effects as well as seasonal.

Among the tools used for spectral analysis are the autoregressive spectrum and the periodogram.

The explanations given in the subsections of this node derive mainly from DE ANTONIO, D., and PALATE, J. (2015) and BROCKWELL, P.J., and DAVIS, R.A. (2006).

Spectral density of an ARIMA model

Estimation

Method 1: Periodogram

For any given frequency ω the sample periodogram is the sample analog of the sample spectrum. In general, the periodogram is used to identify the periodic components of unknown frequency in the time series. X-13ARIMA-Seats and Tramo-Seats use this tool for detecting seasonality in raw time series and seasonally adjusted series. Apart from this it is applied for checking randomness of the residuals from the ARIMA model.

To define a periodogram, first consider a vector of complex numbers

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \in \mathbb{C}^n \tag{0.5}$$

where $\mathbb{C}^n$ is the set of all column vectors with complex-valued components.

The Fourier frequencies associated with the sample size n are defined as a set of values $\omega_j = \frac{2\pi j}{n}$, $j = -[\frac{n-1}{2}], \dots, [\frac{n}{2}]$, $-\pi < \omega_j \leq \pi$, $j \in F_n$, where $[n]$ denotes the largest integer less than or equal to n. The Fourier frequencies, which are called harmonics, are given by integer multiples of the fundamental frequency $\frac{2\pi}{n}$.

Now the n vectors $e_j = n^{-\frac{1}{2}}(e^{-i\omega_j}, e^{-i2\omega_j}, \dots, e^{-in\omega_j})'$ can be defined. Vectors $e_1, \dots, e_n$ are orthonormal in the sense that:

$$\mathbf{e}_j^*\mathbf{e}_k = n^{-1}\sum_{r=1}^{n} e^{ir(\omega_j-\omega_k)} = \begin{matrix} 1,\ if\ j = k \\ 0,\ if\ j \neq k \end{matrix} \tag{0.6}$$

where $\mathbf{e}_j^*$ denotes the row vector, which k^{th} component is the complex conjugate of the k^{th} component of $\mathbf{e}_j$. These vectors are a basis of F_n, so that any $\mathbf{x} \in \mathbb{C}^n$ can be expressed as a sum of n components:

$$\mathbf{x} = \sum_{j=-[\frac{n-1}{2}]}^{[\frac{n}{2}]} a_j\mathbf{e}_j \tag{0.7}$$

where the coefficients $a_j = \mathbf{e}_j^*\mathbf{x} = n^{-\frac{1}{2}}\sum_{t=1}^{n} x_t e^{-it\omega_j}$ are derived from Equation 0.30 by multiplying the equation on the left by $\mathbf{e}_j^*$ and using Equation 0.28.

The sequence of $\{a_j, j \in F_n\}$ is referred as a discrete Fourier transform of $\mathbf{x} \in \mathbb{C}^n$ and the periodogram $I(\omega_j)$ of $\mathbf{x}$ at Fourier frequency $\omega_j = \frac{2\pi j}{n}$ is defined as the square of the Fourier transform $\{a_j\}$ of $\mathbf{x}$:

$$I(\omega_j)=|a_j|^2 \ = n^{-\ 1}|\sum_{t=1}^{n} x_t e^{-it\omega_j}|^2 \tag{0.8}$$

From Equation 0.29 and Equation 0.30 it can be shown that a periodogram decomposes the total sum of squares $\sum_{t=1}^{n}|x_t|^2$ into a sum of components associated with the Fourier frequencies ω_j:

$$\sum_{t=1}^{n}|x_t|^2 = \sum_{j=-[\frac{n-1}{2}]}^{[\frac{n}{2}]}|a_j|^2 = \sum_{j=-[\frac{n-1}{2}]}^{[\frac{n}{2}]} I(\omega_j) \tag{0.9}$$

If $\mathbf{x} \in R^n$, ω_j and $-\omega_j$ are both in $[-\pi, -\pi]$ and a_j is presented in its polar form (i.e. $a_j = r_j \exp(i\theta_j)$), where r_j is the modulus of a_j, then Equation 0.30 can be rewritten in the form:

$$\mathbf{x} = a_0\mathbf{e}_0 + \sum_{j=1}^{[\frac{n-1}{2}]} 2^{1/2} r_j(\mathbf{c}_j \cos\theta_j - \mathbf{s}_j \sin\theta_j) + a_{n/2}\mathbf{e}_{n/2} \tag{0.10}$$

The orthonormal basis for R^n is $\{\mathbf{e}_0, \mathbf{c}_1, \mathbf{s}_1, \ldots, \mathbf{c}_{[\frac{n-1}{2}]}, \mathbf{s}_{[\frac{n-1}{2}]}, \mathbf{e}_{\frac{n}{2}(excluded\ if\ n\ is\ odd)}\}$, where:

$\mathbf{e}_0$ is a vector composed of n elements equal to $n^{-1/2}$, which implies that $\mathbf{a}_0\mathbf{e}_0 = (n^{-1}\sum_{t=1}^{n} x_t, \ldots, n^{-1}\sum_{t=1}^{n} x_t)^{'}$;

$$\mathbf{c}_j = (\frac{n}{2})^{-1/2}(\cos\omega_j, \cos 2\omega_j, \ldots, \cos n\omega_j)^{'}, for 1 \leq j \leq [\frac{(n-1)}{2}]$$

$$\mathbf{s}_j = (\frac{n}{2})^{-1/2}(\sin\omega_j, \sin 2\omega_j, \ldots, \sin n\omega_j)^{'},\ for\ 1 \leq j \leq [\frac{(n-1)}{2}]$$

$$\mathbf{e}_{n/2} = (-(n^{-\frac{1}{2}}), n^{-\frac{1}{2}}, \ldots, -(n)^{-\frac{1}{2}}), n^{-\frac{1}{2}})^{'}$$

Equation 0.32 can be seen as an OLS regression of x_t on a constant and the trigonometric terms. As the vector of explanatory variables includes n elements, the number of explanatory variables in Equation 0.32 is equal to the number of observations. HAMILTON, J.D. (1994) shows that the explanatory variables are linearly independent, which implies that an OLS regression yields a perfect fit (i.e. without an error term). The coefficients have the form of a simple OLS projection of the data on the orthonormal basis:

$$\hat{a}_0 = \frac{1}{\sqrt{n}}\sum_{t=1}^{n} x_t \tag{0.11}$$

$$\hat{a}_{n/2} = \frac{1}{\sqrt{n}}\sum_{t=1}^{n} (-1)^t x_t (\text{only when n is even}) \tag{0.12}$$

$$\hat{a}_0 = \frac{1}{\sqrt{n}} \sum_{t=1}^{n} x_t \tag{0.13}$$

$$\hat{\alpha}_j = 2^{1/2} r_j \cos\theta_j = (\frac{n}{2})^{-1/2} \sum_{t=1}^{n} x_t \cos(t\frac{2\pi j}{n}), j = 1, \ldots, [\frac{n-1}{2}] \tag{0.14}$$

$$\hat{\beta}_j = 2^{1/2} r_j \sin\theta_j = (\frac{n}{2})^{-1/2} \sum_{t=1}^{n} x_t \sin(t\frac{2\pi j}{n}), j = 1, \ldots, [\frac{n-1}{2}] \tag{0.15}$$

With Equation 0.32 the total sum of squares $\sum_{t=1}^{n} |x_t|^2$ can be decomposed into $2 \times [\frac{n-1}{2}]$ components corresponding to $\mathbf{c}_j$ and $\mathbf{s}_j$, which are grouped to produce the "frequency ω_j" component for $1 \geq j \geq [\frac{n-1}{2}]$. As it is shown in the table below, the value of the periodogram at the frequency ω_j is the contribution of the j^{th} harmonic to the total sum of squares $\sum_{t=1}^{n} |x_t|^2$.

Decomposition of sum of squares into components corresponding to the harmonics

Frequency	**Degrees of freedom**	**Sum of squares decomposition**
ω_0 (mean)	1	$a_0^2 = n^{-1}(\sum_{t=1}^{n} x_t)^2 = I(0)$
ω_1	2	$2r_1^2 = 2\|a_1\|^2 = 2I(\omega_1)$
$\vdots$	$\vdots$	$\vdots$
ω_k	2	$2r_k^2 = 2\|a_k\|^2 = 2I(\omega_k)$
$\vdots$	$\vdots$	$\vdots$
$\omega_{n/2} = \pi$ (excluded if n is odd)	1	$a_{n/2}^2 = I(\pi)$
Total	**n**	$\boldsymbol{\sum_{t=1}^{n} x_t^2}$

Source: DE ANTONIO, D., and PALATE, J. (2015).

Obviously, if series were random then each component $I(\omega_j)$ would have the same expectation. On the contrary, when the series contains a systematic sine component having a frequency j and amplitude A then the sum of squares $I(\omega_j)$

increases with A. In practice, it is unlikely that the frequency j of an unknown systematic sine component would exactly match any of the frequencies, for which periodogram have been calculated. Therefore, the periodogram would show an increase in intensities in the immediate vicinity of j. (BOX, G.E.P., JENKINS, G.M., and REINSEL, G.C. (2007).

Note that in JDemetra+ the periodogram object corresponds exactly to the contribution to the sum of squares of the standardised data, since the series are divided by their standard deviation for computational reasons.

Using the decomposition presented in table above the periodogram can be expressed as:

$$I(\omega_j)=r_j^2 = \frac{1}{2}(\alpha_j^2 + \beta_j^2) = \frac{1}{n}\left(\sum_{t=1}^{n} x_t \cos\left(t\frac{2\pi j}{n}\right)\right)^2 + \frac{1}{n}\left(\sum_{t=1}^{n} x_t \sin\left(t\frac{2\pi j}{n}\right)\right)^2 \tag{0.16}$$

where $j = 0, \ldots, [\frac{n}{2}]$.

Since $\mathbf{x}-\overline{\mathbf{x}}$ are generated by an orthonormal basis, and $\overline{\mathbf{x}}=a_0\mathbf{e}_0$ Equation 0.32 can be rearranged to show that the sum of squares is equal to the sum of the squared coefficients:

$$\mathbf{x} - a_0\mathbf{e}_0 = \sum_{j=1}^{[(n-1)/2]} (\alpha_j\mathbf{c}_j + \beta_j\mathbf{s}_j) + a_{n/2}\mathbf{e}_{n/2} \tag{0.17}$$

Thus the sample variance of x_t can be expressed as:

$$n^{-1}\sum_{t=1}^{n}(x_t - \overline{x})^2 = n^{-1}\left(\sum_{k=1}^{[(n-1)/2]} 2{r_j}^2 + {a_{n/2}}^2\right) \tag{0.18}$$

where $a_{n/2}^2$ is excluded if n is odd.

The term $2{r_j}^2$ in Equation 0.41 is then the contribution of the j^{th} harmonic to the variance and Equation 0.41 shows then how the total variance is partitioned.

The periodogram ordinate $I(\omega_j)$ and the autocovariance coefficient $\gamma(k)$ are both quadratic forms of x_t. It can be shown that the periodogram and autocovariance function are related and the periodogram can be written in terms of the sample autocovariance function for any non-zero Fourier frequency ω_j (The proof is given in BROCKWELL, P.J., and DAVIS, R.A. (2006)).

$$I(\omega_j) = \sum_{|k|<n} \hat{\gamma}(k)\ e^{-ik\omega_j} = \hat{\gamma}(0)\ + 2\sum_{k=1}^{n-1} \hat{\gamma}(k) \cos{(k\omega_j)} \tag{0.19}$$

and for the zero frequency $I(0) = n|\overline{x}|^2$.

Once comparing Equation 0.42 with an expression of the spectral density of a stationary process:

$$f(\omega\ ) = \frac{1}{2\pi}\sum_{k<-\infty}^{\infty} \gamma(k)\ e^{-ik\omega}\ = \frac{1}{2\pi}[\gamma(0)\ + 2(\sum_{k=1}^{\infty} \gamma(k) \cos{(k\omega\ )})] \tag{0.20}$$

It can be noticed that a periodogram is a sample analogue of the population spectrum. In fact, it can be shown that the periodogram is asymptotically unbiased but inconsistent estimator of the population spectrum $f(\omega)$. Therefore, the periodogram is a wildly fluctuating, with high variance, estimate of the spectrum. However, the consistent estimator can be achieved by applying the different linear smoothing filters to the periodogram, called lag-window estimators. The lag-window estimators implemented in JDemetra+ includes square, Welch, Tukey, Barlett, Hanning and Parzen. They are described in DE ANTONIO, D., and PALATE, J. (2015). Alternatively, the model-based consistent estimation procedure, resulting in autoregressive spectrum estimator, can be applied.

Method 2: Autoregressive spectrum estimation

BROCKWELL, P.J., and DAVIS, R.A. (2006) point out that for any real-valued stationary process (x_t) with continuous spectral density $f(\omega)$ it is possible to find both $AR(p)$ and $MA(q)$ processes which spectral densities are arbitrarily close to $f(\omega)$. For this reason, in some sense, (x_t) can be approximated by either $AR(p)$ or $MA(q)$ process. This fact is a basis of one of the methods of achieving a consistent estimator of the spectrum, which is called an autoregressive spectrum estimation. It is based on the approximation of the stochastic process (x_t) by an autoregressive process of sufficiently high order p:

$$x_t = \mu + (\phi_1 B + ... + \phi_p B^p)x_t + \varepsilon_t \tag{0.21}$$

where ε_t is a white-noise variable with mean zero and a constant variance.

The autoregressive spectrum estimator for the series x_t is defined as (Definition from '*X-12-ARIMA Reference Manual*' (2011)).

$$\hat{s}(\omega) = 10 \times \log_{10} \frac{\sigma_x^2}{2\pi|1 - \sum_{k=1}^{p} \hat{\phi}_k e^{-ik\omega}|^2} \tag{0.22}$$

where:

- ω– frequency, $0 \leq \omega \leq \pi$;
- σ_x^2 innovation variance of the sample residuals;
- $\hat{\phi}_k$–AR(k) coefficient estimates of the linear regression of $x_t - \overline{x}$ on $x_{t-k} - \overline{x}$, $1 \leq k \leq p$.

The autoregressive spectrum estimator is used in the visual spectral analysis tool for detecting significant peaks in the spectrum. The criterion of *visual significance*, implemented in JDemetra+, is based on the range $\hat{s}^{\max} - \hat{s}^{\min}$ of the $\hat{s}(\omega)$ values, where $\hat{s}^{\max} = \max_k \hat{s}(\omega_k)$; $\hat{s}^{\min} = \min_k \hat{s}(\omega_k)$; and $\hat{s}(\omega_k)$ is k^{th} value of autoregressive spectrum estimator.

A particular value is considered to be visually significant if, at a trading day or at a seasonal frequency ω_k (other than the seasonal frequency $\omega_{60} = \pi$), $\hat{s}(\omega_k)$ is above the median of the plotted values of $\hat{s}(\omega_k)$ and is larger than both neighbouring values $\hat{s}(\omega_{k-1})$ and $\hat{s}(\omega_{k+1})$ by at least $\frac{6}{52}$ times the range $\hat{s}^{\max} - \hat{s}^{\min}$.

Following the suggestion of SOUKUP, R.J., and FINDLEY, D.F. (1999), JDemetra+ uses an autoregressive model spectral estimator of model order 30. This order yields high resolution of strong components, meaning peaks that are sharply defined in the plot of $\hat{s}(\omega)$ with 61 frequencies. The minimum number of observations needed to compute the spectrum is set to $n = 80$ for monthly data and to $n = 60$ for quarterly series while the maximum number of observations considered for the estimation is 121. Consequently, with these settings it is possible to identify up to 30 peaks in the plot of 61 frequencies. By choosing $\omega_k = \frac{\pi k}{60}$ for $k = 0, 1, ..., 60$ the density estimates are calculated at exact seasonal frequencies (1, 2, 3, 4, 5 and 6 cycles per year).

The model order can also be selected based on the AIC criterion (in practice it is much lower than 30). A lower order produces the smoother spectrum, but the contrast between the spectral amplitudes at the trading day frequencies and neighbouring frequencies is weaker, and therefore not as suitable for automatic detection.

SOUKUP, R.J., and FINDLEY, D.F. (1999) also explain that the periodogram can be used in the *visual significance* test as it has as good as those of the AR(30) spectrum abilities to detect trading day effect, but also has a greater false alarm rate, which is defined as the fraction of the 50 replicates for which a visually significant spectral peak occurred at one of the trading day frequencies being considered in the designated output spectra (SOUKUP, R.J., and FINDLEY, D.F. (1999)).

Method 3: Tukey spectrum

The Tukey spectrum belongs to the class of lag-window estimators. A lag window estimator of the spectral density $f(\omega) = \frac{1}{2\pi}\sum_{k<-\infty}^{\infty}\gamma(k)e^{ik\omega}$ is defined as follows:

$$\hat{f}_L(\omega) = \frac{1}{2\pi}\sum_{|h|\leq r} w(h/r)\hat{\gamma}(h)e^{ih\omega}$$

where $\hat{\gamma}(.)$ is the sample autocovariance function, $w(.)$ is the lag window, and r is the truncation lag. $|w(x)|$ is always less than or equal to one, $w(0) = 1$ and $w(x) = 0$ for $|x| > 1$. The simple idea behind this formula is to down-weight the autocovariance function for high lags where $\hat{\gamma}(h)$ is more unreliable. This estimator requires choosing r as a function of the sample size such that $r/narrow0$ and $r \to \infty$ when $narrow\infty$. These conditions guarantee that the estimator converges to the true density.

JDemetra+ implements the so-called Blackman-Tukey (or Tukey-Hanning) estimator, which is given by $w(h/r) = 0.5(1 + cos(\pi h/r))$ if $|h/r| \leq 1$ and 0 otherwise.

The choice of large truncation lags r decreases the bias, of course, but it also increases the variance of the spectral estimate and decreases the bandwidth.

JDemetra+ allows the user to modify all the parameters of this estimator, including the window function.

Identification of spectral peaks

The sections below describe the test, their practical implementation in the Graphical User interface can be found here

In order to decide whether a series has a seasonal component that is predictable (stable) enough, these tests use visual criteria and formal tests for the periodogram. The periodogram is calculated using complete years, so that the set of Fourier frequencies contains exactly all seasonal frequencies.

The tests rely on two basic principles:

- The peaks associated with seasonal frequencies should be larger than the median spectrum for all frequencies and;
- The peaks should exceed the spectrum of the two adjacent values by more than a critical value.

JDemetra+ performs this test on the original series. If these two requirements are met, the test results are displayed in green. The statistical significance of each of the seasonal peaks (i.e. frequencies $\frac{\pi}{6}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$, $\frac{2\pi}{3}$ and $\frac{5\pi}{6}$ corresponding to 1, 2, 3, 4 and 5 cycles per year) is also displayed. The seasonal and trading days frequencies depends on the frequency of time series. They are shown in the table below. The symbol d denotes a default frequency and is described below the table.

The seasonal and trading day frequencies by time series frequency

Number of months per period (year)	Seasonal frequency	Trading day frequency (radians)
12	$\frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}, \frac{2\pi}{3}, \frac{5\pi}{6}, \pi$	d,2.714
4	$\frac{\pi}{2}$, π	d, 1.292, 1.850, 2.128
3	π	d
2	π	d

The calendar (trading day or working day) effects, related to the variation in the number of different days of the week per period, can induce periodic patterns in the data that can be similar to those resulting from pure seasonal effects. From the theoretical point of view, trading day variability is mainly due to the fact that the average number of days in the months or quarters is not equal to a multiple of 7 (the average number of days of a month in the year of 365.25 days is equal to $\frac{365.25}{12} = 30.4375$ days). This effect occurs $\frac{365.25}{12} \times \frac{1}{7} = 4.3482$ times per month: one time for each one of the four complete weeks of each month, and a residual of 0.3482 cycles per month, i.e. $0.3482 \times 2\pi = 2.1878$ radians. This turns out to be a fundamental frequency for the effects associated with monthly data. In JDemetra+ the fundamental frequency corresponding to 0.3482 cycles per month is used in place of the closest frequency $\frac{\pi k}{60}$. Thus, the quantity $\frac{\pi \times 42}{60}$ is replaced by $\omega_{42} = 0.3482 \times 2\pi = 2.1878$. The frequencies neighbouring ω_{42}, i.e. ω_{41} and ω_{43} are set to, respectively, $2.1865 - \frac{1}{60}$ and $2.1865 + \frac{1}{60}$.

The default frequencies (d) for calendar effect are: 2.188 (monthly series) and 0.280 (quarterly series). They are computed as:

$$\omega_{\text{ce}} = \frac{2\pi}{7}(n - 7 \times [\frac{n}{7}]) \tag{0.23}$$

where $n = \frac{365.25}{s}$, $s = 4$ for quarterly series and $s = 12$ for monthly series.

Other frequencies that correspond to trading day frequencies are: 2.714 (monthly series) and 1.292, 1.850, 2.128 (quarterly series).

In particular, the calendar frequency in monthly data (marked in red on the figure below) is very close to the seasonal frequency corresponding to 4 cycles per year $\omega_{40} = \frac{2}{3}\pi = 2.0944$.

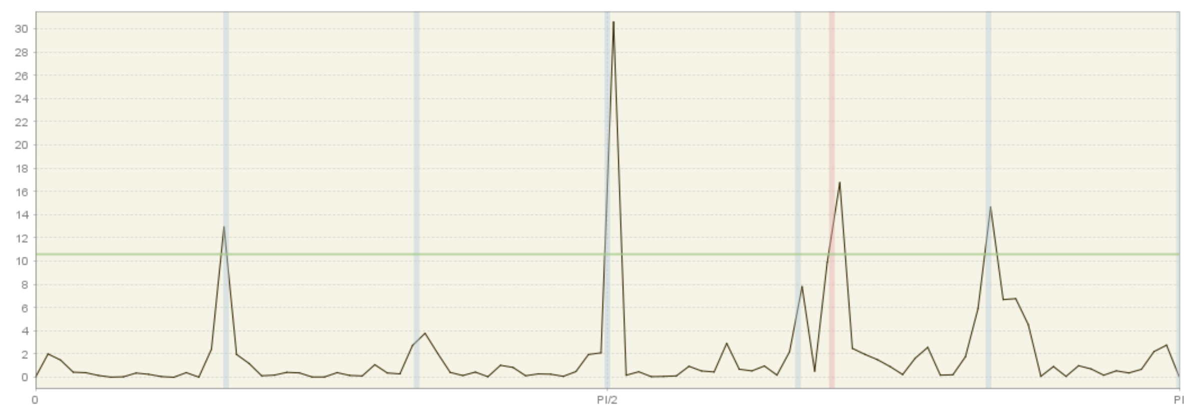


Figure 258: **Periodogram with seasonal (grey) and calendar (red) frequencies highlighted**

This implies that it may be hard to disentangle both effects using the frequency domain techniques.

In a Tukey spectrum

Current JDemetra+ implementation of the seasonality test is based on a $F(d_1, d_2)$ approximation that has been originally proposed by Maravall (2012) for Tramo-Seats. This test is has been designed for a Blackman-Tukey window based on a particular choices of the truncation lag r and sample size. Following this approach, we determine visually significant peaks for a frequency ω_j when

$$\frac{2f_x(\omega_j)}{[f_x(\omega_{j+1}) + f_x(\omega_{j-1})]} \geq CV(\omega_j)$$

where $CV(\omega_j)$ is the critical value of a $F(d_1, d_2)$ distribution, where the degrees of freedom are determined using simulations. For $\omega_j = \pi$, we have a significant peak when $\frac{f_x(\omega_{[n/2]})}{[f_x(\omega_{[(n-1)/2]})]} \geq CV(\omega_j)$

Two significant levels for this test are considered: $\alpha = 0.05$ (code "t") and $\alpha = 0.01$ (code "T").

As opposed to the AR spectrum test, which is computed on the basis of the last 120 data points, we will use here all available observations. Those critical values have been calculated given the recommended truncation lag $r = 79$ for a sample size within the interval $\in [80, 119]$ and $r = 112$ for $n \in [120, 300]$. The F approximation is less accurate for sample sizes larger than 300. For quarterly data, $r = 44$, but there are no recommendations regarding the required sample size.

Practical implementation in GUI is detailed here

In AR Spectrum definition

The estimator of the spectral density at frequency $\lambda \in [0, \pi]$ will be given by the assumption that the series will follow an AR(p) process with large p. The spectral density of such model, with an innovation variance $var(x_t) = \sigma_x^2$, is expressed as follows:

$$10 \times log_{10} f_x(\lambda) = 10 \times log_{10} \frac{\sigma_x^2}{2\pi|\phi(e^{i\lambda})|^2} = 10 \times log_{10} \frac{\sigma_x^2}{2\pi|1 - \sum_{k=1}^{p} \phi_k e^{ik\lambda})|^2}$$

where:

- ϕ_k denotes the AR(k) coefficient ;
- $e^{-ik\lambda} = cos\ (-ik\lambda) + isin\ (-ik\lambda)$.

Soukup and Findely (1999) suggest the use of p=30, which in practice much larger than the order that would result from the AIC criterion. The minimum number of observations needed to compute the spectrum is set to *n=80* for monthly data (or *n=60*) for quarterly series. In turn, the maximum number of observations considered for the estimation is *n=121*. This choice offers enough resolution, being able to identify a maximum of 30 peaks in a plot of 61 frequencies: by choosing $\lambda_j = \pi j/60$,for $j = 0, 1, ..., 60$, we are able to calculate our density estimates at exact seasonal frequencies (1, 2, 3, 4, 5 and 6 cycles per year). Note that x cycles per year can be converted into cycles per month by simply dividing by twelve, $x/12$, and to radians by applying the transformation $2\pi(x/12)$.

The traditional trading day frequency corresponding to 0.348 cycles per month is used in place of the closest frequency $\pi j/60$. Thus, we replace $\pi 42/60$ by $\lambda_{42} = 0.348 \times 2\pi = 2.1865$. The frequencies neighbouring λ_{42} are set to $\lambda_{41} = 2.1865 - 1/60$ and $\lambda_{43} = 2.1865 + 1/60$. The periodogram below illustrates the proximity of this trading day frequency λ_{42} (red shade) and the frequency corresponding to 4 cycles per year $\lambda_{40} = 2.0944$. This proximity is precisely what poses the identification problems: the AR spectrum boils down to a smoothed version of the periodogram and the contribution of the of the trading day frequency may be obscured by the leakage resulting from the potential seasonal peak at λ_{40}, and vice-versa.

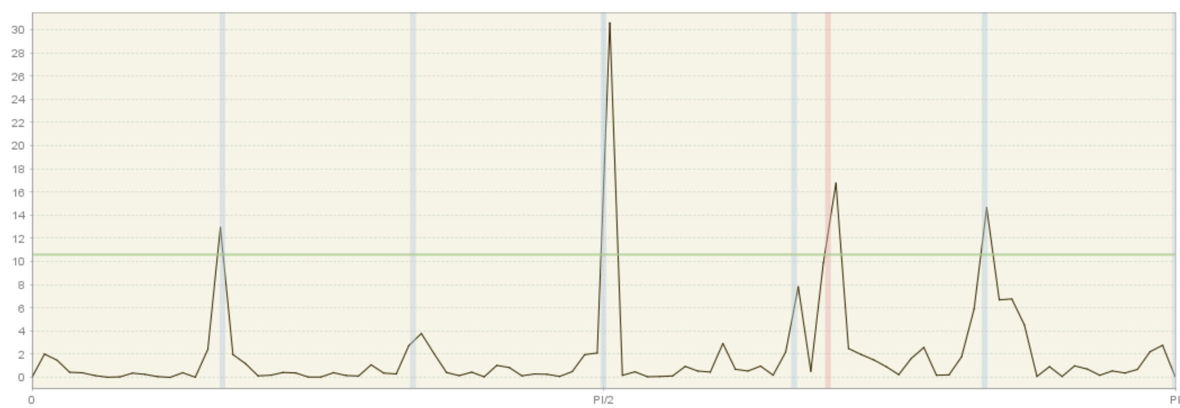


Figure 259: **Periodogram with seasonal (grey) and calendar (red) frequencies highlighted**

JDemetra+ allows the user to modify the number of lags of this estimator and to change the number of observations used to determine the AR parameters. These two options can improve the resolution of this estimator.

The statistical significance of the peaks associated to a given frequency can be informally tested using a visual criterion, which has proved to perform well in simulation experiments. Visually significant peaks for a frequency λ_j satisfy both conditions:

- $\frac{f_x(\lambda_j)-\max\{f_x(\lambda_{j+1}),f_x(\lambda_{j-1})\}}{[\max_k f_x(\lambda_k)-\min_i f_x(\lambda_i)]} \geq CV(\lambda_j)$, where $CV(\lambda_j)$ can be set equal to $6/52$ for all j
- $f_x(\lambda_j) > median_j\{f_x(\lambda_j)\}$, which guarantees $f_x(\lambda_j)$ it is not a local peak.

The first condition implies that if we divide the range $\max_k f_x(\lambda_k) - \min_i f_x(\lambda_i)$ in 52 parts (traditionally represented by stars) the height of each pick should be at least 6 stars.

Seasonal and trading day frequencies by time series frequency

Number of months per full period	Seasonal frequency	Trading day frequency (radians)
12	$\frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}, \frac{2\pi}{3}, \frac{5\pi}{6}, \pi$	d, 2.714
6	$\frac{\pi}{3}, \frac{2\pi}{3}, \pi$	d
4	$\frac{\pi}{2}, \pi$	d, 1.292, 1.850, 2.128
3	π	d
2	π	d

Currently, only seasonal frequencies are tested, but the program allows you to manually plot the AR spectrum and focus your attention on both seasonal and trading day frequencies. Agustin Maravall has conducted a simulation experiment to calculate $CV(\lambda_{42})$ (trading day frequency) and proposes to set for all j equal to the critical value associated to the trading frequency, but this is currently not part of the current automatic testing procedure of JDemetra+.

Practical implementation in GUI is detailed here

In a Periodogram

The periodogram $I(\omega_j)$ of $\mathbf{X} \in \mathbb{C}^n$ is defined as the squared of the Fourier transform

$$I(\omega_j) = a_j^2 = n^{-1} |\sum_{t=1}^{n} \mathbf{X_t} e^{-it\omega_j}|^2,$$

where the Fourier frequencies ω_j are given by multiples of the fundamental frequency $\frac{2\pi}{n}$:

$$\omega_j = \frac{2\pi j}{n}, -\pi < \omega_j \leq \pi$$

An orthonormal basis in $\mathbb{R}^n$:

$$\{e_0, \quad c_1, s_1, \quad \dots \quad , \quad c_{[(n-1)/2]}, s_{[(n-1)/2]} \quad , \quad e_{n/2}\},$$

where $e_{n/2}$ is excluded if n is odd,

can be used to project the data and obtain the spectral decomposition

Thus, the periodogram is given by the projection coefficients and represents the contribution of the jth harmonic to the total sum of squares, as illustrated by Brockwell and Davis (1991):

Source	Degrees of freedom
Frequency ω_0	1
Frequency ω_1	2
$\vdots$	$\vdots$
Frequency ω_k	2
$\vdots$	$\vdots$
Frequency $\omega_{n/2} = \pi$ (excluded if n is odd)	1
==========	======
Total	n

In JDemetra+, the periodogram of **X** $\in \mathbb{R}^n$ is computed for the standardized time series.

Defining a F-test

Brockwell and Davis (1991, section 10.2) exploit the fact that the periodogram can be expressed as the projection on the orthonormal basis defined above to derive a test. Thus, under the null hypothesis:

- $2I(\omega_k) = \|P_{\bar{sp}_{\{c_k,s_k\}}}\mathbf{X}\|^2 \sim \sigma^2\chi^2(2)$, for Fourier frequencies $0 < \omega_k = 2\pi k/n < \pi$
- $I(\pi) = \|P_{\bar{sp}_{\{e_{n/2}\}}}\mathbf{X}\|^2 \sim \sigma^2\chi^2(1)$, for π

Because $I(\omega_k)$ is independent from the projection error sum of squares, we can define our F-test statistic as follows:

- $\frac{2I(\omega_k)}{\|\mathbf{X}-P_{\bar{sp}_{\{e_0,c_k,s_k\}}}\mathbf{X}\|^2}\frac{n-3}{2} \sim F(2, n-3)$, for Fourier frequencies $0 < \omega_k = 2\pi k/n < \pi$
- $\frac{I(\pi)}{\|\mathbf{X}-P_{\bar{sp}_{\{e_0,e_{n/2}\}}}\mathbf{X}\|^2}\frac{n-2}{1} \sim F(1, n-2)$, for π

where:

-$\|\mathbf{X} - P_{\bar{sp}_{\{e_0,c_k,s_k\}}}\mathbf{X}\|^2 = \sum_{i=1}^{n}\mathbf{X_i^2} - I(0) - 2I(\omega_k) \sim \sigma^2\chi^2(n-3)$ for Fourier frequencies $0 < \omega_k = 2\pi k/n < \pi$

-$\|\mathbf{X} - P_{\bar{sp}_{\{e_0,e_{n/2}\}}}\mathbf{X}\|^2 = \sum_{i=1}^{n}\mathbf{X_i^2} - I(0) - I(\pi) \sim \sigma^2\chi^2(n-2)$ for π

Thus, we reject the null if our F-test statistic computed at a given seasonal frequency (different from π) is larger than $F_{1-\alpha}(2, n-3)$. If we consider π, our test statistic follows a $F_{1-\alpha}(1, n-2)$ distribution.

The implementation of JDemetra+ considers simultaneously the whole set of seasonal frequencies (1, 2, 3, 4, 5 and 6 cycles per year). Thus, the resulting test-statistic is:

$$\frac{2I(\pi/6) + 2I(\pi/3) + 2I(2\pi/3) + 2I(5\pi/6) + \delta I(\pi)}{\|\mathbf{X} - P_{\bar{sp}_{\{e_0,c_1,s_1,c_2,s_2,c_3,s_3,c_4,s_4,c_5,s_5,\delta e_{n/2}\}}}\mathbf{X}\|^2}\frac{n-12}{11} \sim F(11-\delta, n-12+\delta)$$

where $\delta = 1$ if n is even and 0 otherwise.

In small samples, the test performs better when the periodogram is evaluated as the exact seasonal frequencies. JDemetra+ modifies the sample size to ensure the seasonal frequencies belong to the set of Fourier frequencies. This strategy provides a very simple and effective way to eliminate the leakage problem.

Practical implementation in GUI is detailed here

Spectral graphs

The section below provides guidance on interpretation of spectral graphs, the display of which in the Graphical User Interface can be found here

The interpretation of the spectral graph is rather straightforward. When the values of a spectral graph for low frequencies (i.e. one year and more) are large in relation to its other values it means that the long-term movements dominate in the series. When the values of a spectral graph for high frequencies (i.e. below one year) are large in relation to its other values it means that the series are rather trendless and contains a lot of noise. When the values of a spectral graph are distributed randomly around a constant without any visible peaks, then it is highly probable that the series is a random process. The presence of seasonality in a time series is manifested in a spectral graph by the peaks on the seasonal frequencies.

Reg-ARIMA models

Under construction.

Overview

The primary aim of seasonal adjustment is to remove the unobservable seasonal component from the observed series. The decomposition routines implemented in the seasonal adjustment methods make specific assumptions concerning the input series. One of the crucial assumptions is that the input series is stochastic, i.e. it is clean of deterministic effects. Another important limitation derives from the symmetric linear filter used in Tramo-Seats and X-13ARIMA-Seats. A symmetric linear filter cannot be applied to the first and last observations with the same set of weights as for the central observations[^1]. Therefore, for the most recent observations these filters provide estimates that are subject to revisions.

To overcome these constrains both seasonal adjustment methods discussed here include a modelling step that aims to analyse the time series development and provide a better input for decomposition purposes. The tool that is frequently used for this purpose is the ARIMA model, as discussed by BOX, G.E.P., and JENKINS, G.M. (1970). However, time series are often affected by the outliers, other deterministic effects and missing observations. The presence of these effects is not in line with the ARIMA model assumptions. The presence of outliers and other deterministic effects impede the identification of an optimal ARIMA model due to the important bias in the estimation of parameters of sample autocorrelation functions (both global and partial)[^3]. Therefore, the original series need to be corrected for any deterministic effects and missing observations. This process is called linearisation and results in the stochastic series that can be modelled by ARIMA.

For this purpose both Tramo and Reg-ARIMA use regression models with ARIMA errors. With these models Tramo and Reg-ARIMA also produce forecasts.

X11 decomposition

X11 is the decomposition part of X-13-ARIMA seasonal adjustment algorithm originally developed by the US Census Bureau. It is a non parametric algorithm based on moving averages.

The genuine X11 algorithm was meant to decompose monthly ($p = 12$) and quarterly series ($p = 4$). In JDemetra+ half-yearly data ($p = 2$), quadri-monthly ($p = 3$) and bi-monthly data ($p = 6$) can also be handled. ($p = 3$ and $p = 6$ only in version 3.x)

In the following chapter, explanations and examples are based on monthly and quarterly data, but the principles can be extended to other supra-monthly frequencies.

In recent years, X11 implementation in JDemetra+ v 3.x has been extended to infra-monthly data (weekly, daily, hourly...). Handling this kind of data gave way to a tailored X11 algorithm whose peculiarities are described in this chapter.

In this chapter

This chapter provides details on - algorithm steps - computation stages and detailed output series - quality measures - filter length choices - extreme values correction

The practical implementation as well as all the options, using the graphical user interface or R packages are described in this chapter

Moving averages in X11

Moving averages (MA) are the building blocks of X11. They will be used successively to accomplish three goals:

```
- removing seasonality

- extracting seasonality

- estimating Trend on non seasonal series
```

The type of MA used for each of these tasks and the computation steps are described in the sections below.

Definitions

A moving average of *order* $p + f + 1$ and coefficients (θ_i) is the operator M defined as:

$$MX_t = \sum_{i=-p}^{f} \theta_i X_{t+i}$$

The series value in t is replaced by a weighted average of p past values, the current value and the f future values. If $p = f$, the moving average is *centered* and if $\theta_{-i} = \theta_i$, it is *symmetrical*.

Example of simple moving average of order 3:

$$MX_t = \frac{1}{3}(X_{t-1} + X_t + X_{t+1})$$

A moving average is a **linear operator**, $M(X_t + Y_t) = M(X_t) + M(Y_t)$.

Combined moving averages

Centred and symmetrical moving averages preserve linear trends, which is a desirable propriety. They cannot have an even order, thus for even orders they are obtained by combining simple moving averages as arithmetic means of p moving averages of the same order (ie. length): $M_{p \times order}$

Combination example for order 12:

There are two intuitive ways to create a Moving Average of order 12:

$$M1X_t = \frac{1}{12}(X_{t-6} + X_{t-5} + X_{t-4} + X_{t-3} + X_{t-2} + X_{t-1}$$

$$+X_t + X_{t+1} + X_{t+2} + X_{t+3} + X_{t+4} + X_{t+5})$$

The other being:

$$M2X_t = \frac{1}{12}(X_{t-5} + X_{t-4} + X_{t-3} + X_{t-2} + X_{t-1} + X_t$$
$$+X_{t+1} + X_{t+2} + X_{t+3} + X_{t+4} + X_{t+5} + X_{t+6})$$

A centred and symmetrical MA with an **even order** (here 12) can be created:

$$M_{2\times 12} = \frac{1}{2}(M1X_t + M2X_t)$$

which is:

$$M_{2\times 12} = \frac{1}{24}(X_{t-6}) + \frac{1}{12}(X_{t-5} + X_{t-4}$$
$$+X_{t-3} + X_{t-2} + X_{t-1} + X_t + X_{t+1} + X_{t+2}$$
$$+X_{t+3} + X_{t+4} + X_{t+5}) + \frac{1}{24}(X_{t+6})$$

Supressing locally constant seasonality

Applying a moving average of an order equal to the periodicity of the raw series removes a locally stable seasonality ($\sum_{i=1}^{12} S_{t+i} = 0$)

A moving average of order, 12 will remove a locally stable monthly seasonality: $M_{1\times 12}(S) = 0$ and also $M_{2\times 12}(S) = 0$ with linear trend preservation.

X11 algorithm steps

The X11 decomposition algorithm has eight main steps, outlined below for a monthly time series. (For a quarterly time series a 2×4 moving average would be used, instead of 2×12)

Step 1: Estimation of the **trend-cycle** with a 2×12 MA

$$TC_t^{(1)} = M_{2\times 12}(X_t)$$

Step 2: Estimation of the **seasonal+irregular** component

$$(S_t + I_t)^{(1)} = X_t - TC_t^{(1)}$$

Step 3: Estimation of the **seasonal** component by applying a 3×3 MA to **each month**

$S_t^{(1)} = M_{3\times3}\left[(S_t + I_t)^{(1)}\right]$ and normalisation $Snorm_t^{(1)} = S_t^{(1)} - M_{2\times12}\left(S_t^{(1)}\right)$

Step 4: First estimation of the seasonally adjusted series

$$Xsa_t^{(1)} = (TC_t + I_t)^{(1)} = X_t - Snorm_t^{(1)}$$

Step 5: Refined estimation of the **trend-cycle** with a Henderson filter, which yields a better approximation fo trends than 2×12 MA, but cannot be applied on a seasonal series

$$TC_t^{(2)} = H_{13}(Xsa_t^{(1)})$$

Step 6: Refined estimation of the **seasonal+irregular** part

$$(S_t + I_t)^{(2)} = X_t - TC_t^{(2)}$$

Step 7: Refined estimation of the **seasonal** component by applying a 3×5 MA (generally) to **each month/quarter**

$S_t^{(2)} = M_{3\times5}\left[(S_t + I_t)^{(2)}\right]$ and normalisation $Snorm_t^{(2)} = S_t^{(2)} - M_{2\times12}\left(S_t^{(2)}\right)$

Step 8: Final estimation of the seasonally adjusted series

$$Xsa_t^{(2)} = X_t - Snorm_t^{(2)}$$

Processing stages

To evaluate the different components of a series, while taking into account the possible presence of extreme observations, X11 will proceed iteratively: - estimation of components - search for disruptive effects in the irregular component - estimation of components over a corrected series - search for disruptive effects in the irregular component - ...

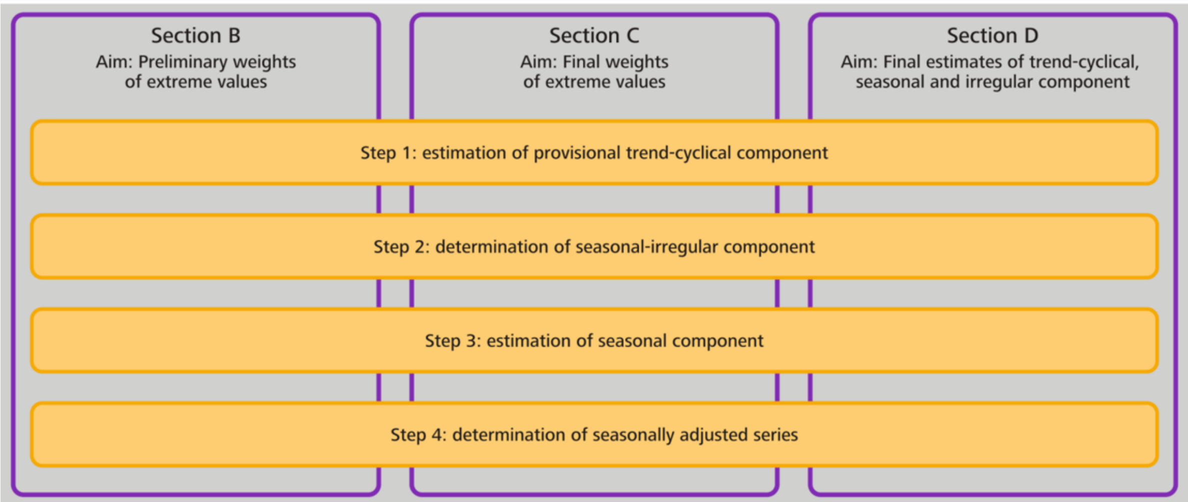
The steps described above will be run six times, at least.

The algorithm is split into

- four processing stages called A, B, C and D
- two diagnostics parts: E tables and Quality Measures (Summary and Detailed)

The basic principle of the X-11 seasonal adjustment algorithm*

Workflow diagram, simplified version

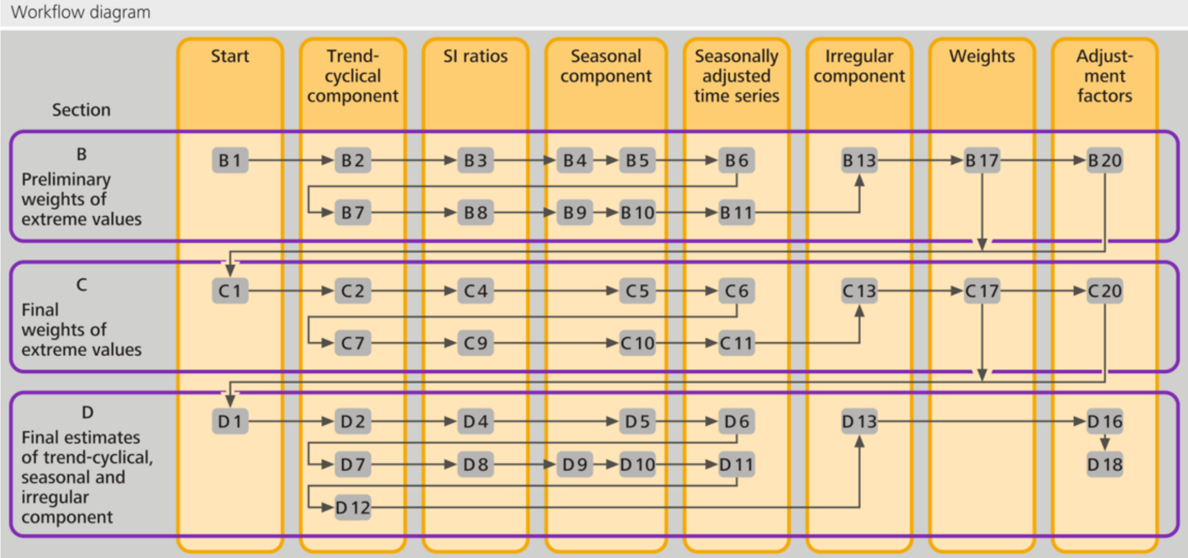


* In X-13 terminology, section A is solely devoted to the treatment of outliers and calendar effects within a regARIMA modelling framework which is done prior to the application of the X-11 core.

Deutsche Bundesbank S3PR0023.Chart

Basic principle of the X-11 seasonal adjustment algorithm in JDemetra+

Workflow diagram



Deutsche Bundesbank S3PR0037B.Chart

Series description (input and output)

Stage A: Pre-adjustment

If a pre-treatment is performed using X-13-ARIMA algorithm is A step is not used, results in A table appearing in JDemetra+ output are a copy of pre-treatment results.

Detailed series produced at the end of stage A, including estimated effect from the Reg-ARIMA part

- Table A1: Original raw series;
- Table A1a: Forecast of Original Series;
- Table A2: Leap year effect;
- Table A6: Trading Day effect (1 or 6 variables);
- Table A7: The Easter effect;
- Table A8: Total Outlier Effect;
- Table A8i: Additive outlier effect;
- Table A8t: Level shift effect;
- Table A8s: Transitory effect;
- Table A9: Effect of user-defined regression variables assigned to the seasonally adjusted series or for which the component has not been defined;
- Table 9sa: Effect of user-defined regression variables assigned to the seasonally adjusted series;
- Table9u: Effect of user-defined regression variables for which the component has not been defined.

Stage B: First automatic correction of the series

This stage consists of a first estimation and down-weighting of extreme observations. This stage is performed by applying twice the algorithm steps outlined above. It starts with the linearised or raw series copied in table B1 lead to table B20, containing adjustment values for extreme observations. B1 corrected with weights from B20 allows to compute the series C1 which will start the next stage.

Detailed series produced at the end of stage B:

- Table B1: Linearized Series (Original series after adjustment by the Reg-ARIMA model);
- Table B2: Unmodified Trend (preliminary estimation using composite moving average);
- Table B3: Unmodified Seasonal: Irregular Component (preliminary estimation);
- Table B4: Replacement Values for Extreme SI Values;
- Table B5: Seasonal Component;
- Table B6: Seasonally Adjusted Series;
- Table B7: Trend (estimation using Henderson moving average);
- Table B8: Unmodified Seasonal: Irregular Component;
- Table B9: Replacement Values for Extreme SI Values;
- Table B10: Seasonal Component;
- Table B11: Seasonally Adjusted Series;
- Table B13: Irregular Component;
- Table B17: Preliminary Weights for the Irregular;
- Table B20: Adjustment Values for the original series B1, allow to compute C1.

(Up coming here: computation steps from B1 to B20)

Stage C: Second automatic correction of the series**

Following the same steps as Stage B, this stage leads to table C20, which allows to compute the final "cleaned up" series shown in D1.

Detailed series produced at the end of stage C:

- Table C1: Modified Linearized Series;
- Table C2: Trend (preliminary estimation using composite moving average);
- Table C4: Modified Seasonal: Irregular Component;
- Table C5: Seasonal Component;
- Table C6: Seasonally Adjusted Series;
- Table C7: Trend (estimation using Henderson moving average);

- Table C9: Seasonal: Irregular Component;
- Table C10: Seasonal Component;
- Table C11: Seasonally Adjusted Series;
- Table C13: Irregular Component;
- Table C20: Adjustment Values for the original series B1, allow to compute D1.

Stage D: Final decomposition and seasonal adjustment

In this part, all algorithm steps are applied one last time, finally leading to the computation of the final components

Detailed series produced at the end of stage D:

- Table D1: Modified Raw Series;
- Table D2: Trend (preliminary estimation using composite moving average);
- Table D4: Modified Seasonal: Irregular Component;
- Table D5: Seasonal Component;
- Table D6: Seasonally Adjusted Series;
- Table D7: Trend (estimation using Henderson moving average);
- Table D8: Unmodified Seasonal: Irregular Component;
- Table D9: Replacement Values for Extreme SI Values;
- Table D10: Final Seasonal Factors of the decomposition phase, without pre-adjustment effects meant to be allocated to the seasonal component in version 3, and with those effects version 2;
- Table D10A: Forecast of Final Seasonal Factors of the decomposition phase;
- Table D11: Final Seasonally Adjusted Series (obtained with D10 factors);
- Table D11A: Forecast of Final Seasonally Adjusted Series;
- Table D12: Final Trend, without pre-adjustment effects meant to be allocated to the trend in version 3, and with those effects version 2;
- Table D12A: Forecast of Final Trend Component;
- Table D13: Final Irregular Component without pre-adjustment effects meant to be allocated to the irregular in version 3, and with those effects version 2;

D-Final-Table

In version 3 the very final components including pre-adjustment effects are stored in the D-Final-Table

- Table D11: Final Seasonally Adjusted Series (obtained with D16 factors);
- Table D11A: Forecast of Final Seasonally Adjusted Series;
- Table D12: Final Trend ;
- Table D12A: Forecast of Final Trend Component;
- Table D13: Final Irregular Component;
- Table D16: Final Seasonal Factors ;
- Table D16A: Forecast of Final Seasonal Factors;
- Table D18: Combined Calendar Effects Factors.

Stage E: Components modified for large extreme values

In this part, additional series will are computed and used in the Quality Measures part

Detailed series produced at the end of stage E:

- Table E1: Raw Series Modified for Large Extreme Values
- Table E2: SA Series Modified for Large Extreme Values
- Table E3: Final Irregular Component Adjusted for Large Extreme Values
- Table E11: Robust Estimation of the Final SA Series

Quality Measures

All the diagnostics below can be displayed in the GUI by expanding the NODES

Decomposition(X11) > Quality Measures > Summary

Decomposition(X11) > Quality Measures > Details

M-statisctics

M statistics are specific quality measures (ref: Lothian and Mory (1979))

- $0 < M_x < 3$, acceptance region $M_x \leq 1$
- 11 statistics of the decomposition quality (M1 to M11) and 2 summary indicators (Q and Q-M2)

Detailed description:

- $M1$ measures the contribution of the irregular component to the total variance. When it is above 1 some changes in outlier correction should be considered.
- $M2$, which is a very similar to $M1$, is calculated on the basis of the contribution of the irregular component to the stationary portion of the variance. When it is above 1, some changes in an outlier correction should be considered.
- $M3$ compares the irregular to the trend taken from a preliminary estimate of the seasonally adjusted series. If this ratio is too large, it is difficult to separate the two components from each other. When it is above 1 some changes in outlier correction should be considered.
- $M4$ tests the randomness of the irregular component. A value above 1 denotes a correlation in the irregular component. In such case a shorter seasonal moving average filter should be considered.
- $M5$ is used to compare the significance of changes in the trend with that in the irregular. When it is above 1 some changes in outlier correction should be considered.
- $M6$ checks the SI (seasonal: irregular components ratio). If annual changes in the irregular component are too small in relation to the annual changes in the seasonal component, the 3×5 seasonal filter used for the estimation of the seasonal component is not flexible enough to follow the seasonal movement. In such case a longer seasonal moving average filter should be considered. It should be stressed that $M6$ is calculated only if the 3×5 filter has been applied in the model.
- $M7$ is the combined test for the presence of an identifiable seasonality. The test compares the relative contribution of stable and moving seasonality[^m-X11-decomposition-2].

- $M8$ to $M11$ measure if the movements due to the short-term quasi-random variations and movements due to the long-term changes are not changing too much over the years. If the changes are too strong then the seasonal factors could be erroneous. In such case a correction for a seasonal break or the change of the seasonal filter should be considered.

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

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

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

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

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

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

Detailed Quality measures

Average percent change (or Average differences) without regard to sign over the indicated span

The first table presents the average percent change without regard to sign of the percent changes (multiplicative model) or average differences (additive model) over several periods (from 1 to 12 for a monthly series, from 1 to 4 for a quarterly series) for the following series:

- O: Original series (Table A1);
- CI: Final seasonally adjusted series (Table D11);
- I: Final irregular component (Table D13);
- C: Final trend (Table D12);

- S: Final seasonal factors (Table D10);
- P: Preliminary adjustment coefficients, i.e. regressors estimated by the Reg-ARIMA model (Table A2);
- $TD\&H$: Final calendar component (Tables A6 and A7);
- Mod.O: Original series adjusted for extreme values (Table E1);
- Mod.CI: Final seasonally adjusted series corrected for extreme values (Table E2);
- Mod.I: Final irregular component adjusted for extreme values (Table E3).

In the case of an additive decomposition, for each component the average absolute changes over several periods are calculated as:

$$\text{Component}_d = \frac{1}{n-d} \sum_{t=d+1}^{n} |Table_t - Table_{t-d}|$$

where:

d: time lag in periods (from a monthly time series d varies from to 4 or from 1 to 12);

n: total number of observations per period;

Component: the name of the component;

Table: the name of the table that corresponds to the component.

Average percent change without regard to sign over the indicated span

Span	O	CI	I	C	S	P	TD&H	Mod.O	Mod.CI	Mod.I
1	7,50	3,81	3,49	1,42	6,99	0,00	0,00	7,75	3,57	3,29
2	5,33	4,88	3,90	2,88	3,57	0,00	0,00	5,40	4,61	3,55
3	8,23	5,75	3,74	4,39	7,16	0,00	0,00	8,53	5,50	3,39
4	6,36	6,75	3,76	5,94	0,00	0,00	0,00	6,74	6,74	3,56

For the multiplicative decomposition the following formula is used:

$$\text{Component}_d = \frac{1}{n-d} \sum_{t=d+1}^{n} |\frac{\text{Table}_t}{\text{Table}_{t-d}} - 1|$$

.

Relative contribution to the variance of the differences in the components of the original series

Relative contributions of the different components to the differences (additive model) or percent changes (multiplicative model) in the original series is displayed express the relative importance of the changes in each component. Assuming that the components are independent, the following relation is valid:

$$O_d^2 \approx C_d^2 + S_d^2 + I_d^2 + P_d^2 + TD\&H_d^2$$

In order to simplify the analysis, the approximation can be replaced by the following equation:

$$O_d^{*2} = C_d^2 + S_d^2 + I_d^2 + P_d^2 + TD\&H_d^2$$

The column Total denotes total changes in the raw time series.

Data presented in Table F2B indicate the relative contribution of each component to the percent changes (differences) in the original series over each span, and are calculated as:
$\frac{I_d^2}{O_d^{*2}}$, $\frac{C_d^2}{O_d^{*2}}$, $\frac{S_d^2}{O_d^{*2}}$, $\frac{P_d^2}{O_d^{*2}}$ and $\frac{TD\&H_d^2}{O_d^{*2}}$ where: $O_d^{*2} = I_d^2 + C_d^2 + S_d^2 + P_d^2 + TD\&H_d^2$.

The last column presents the *Ratio* calculated as: $100 \times \frac{O_d^{*2}}{O_d^2}$, which is an indicator of how well the approximation $(O_d^*)^2 \approx O_d^2$ holds.

Relative contributions to the variance of the percent change in the components of the original series

Span	I	C	S	P	TD&H	Total	Ratio
1	17,53	3,27	79,20	0,00	0,00	100,00	102,79
2	37,38	24,71	37,91	0,00	0,00	100,00	115,35
3	13,97	23,47	62,56	0,00	0,00	100,00	112,79
4	26,47	73,53	0,00	0,00	0,00	100,00	105,49

Average differences with regard to sign and standard deviation over indicated span

When an additive decomposition is used, Table F2C presents the average and standard deviation of changes calculated for each time lag d, taking into consideration the sign of the changes of the raw series and its components. In case of a multiplicative decomposition the respective table shows the average percent differences and related standard deviations.

Average percent change with regard to sign and standard deviation over indicated span

	O		I		C		S		CI	
Span	Avg	S.D.	Avg	S.D.	Avg	S.D.	Avg	S.D.	Avg	S.D.
1	1,97	8,67	0,05	3,73	1,41	0,48	0,53	8,01	1,46	3,81
2	3,19	5,72	0,15	4,48	2,86	0,97	0,24	4,42	3,02	4,72
3	4,97	9,47	0,09	4,52	4,36	1,44	0,55	8,30	4,46	4,97
4	5,93	3,81	0,10	4,32	5,90	1,90	0,00	0,00	6,01	5,06

Average duration of run

Average duration of run is an average number of consecutive monthly (or quarterly) changes in the same direction (no change is counted as a change in the same direction as the preceding change). JDemetra+ displays this indicator for the seasonally adjusted series, for the trend and for the irregular component.

Average duration of run.

CI	8.44
I	1.31
C	15.20

Figure 260: **Average duration of run**

I/C ratio over indicated span and global

The $\frac{I}{C}$ ratios for each value of time lag d, presented in Table F2E, are computed on a basis of the data in Table F2A. Global IC is displayed below the table

I/C Ratio for indicated span.

1	**0.150**
2	0.052
3	0.039
4	0.031

I/C Ratio: 0.314

Figure 261: **I/C ratio**

Relative contribution to the stationary part of the variance in the original series

The relative contribution of components to the variance of the stationary part of the original series is calculated for the irregular component (I), trend made stationary (C), seasonal component (S) and calendar effects (TD&H).

The trend is made stationary by by extracting a linear trend from the trend component presented in Table D12.

Relative contribution of the components to the stationary portion of the variance in the original series.

I	0.01
C	99.56
S	0.15
P	0.00
TD&H	0.00
Total	99.72

Figure 262: **Relative contribution to the stationary part of the variance in the original series**

Autocorrelations in the irregular

The last table shows the autocorrelogram of the irregular component from Table D13. In the case of multiplicative decomposition it is calculated for time lags between 1 and the number of periods per year +2 using the formula:

$$\text{Corr}_k I = \frac{\sum_{t=k+1}^{N} (I_t - 1)(I_{t-k} - 1)}{\sum_{t=1}^{N} (I_t - 1)^2}$$

where N is number of observations in the time series and k the lag.

For the additive decomposition the formula is:

$$Corr_k I_t = \frac{\sum_{t=k+1}^{N} (I_t \times I_{t-k})}{\sum_{t=1}^{N} (I_t)^2}$$

Autocorrelation of the irregular

1	-0.601
2	0.200
3	0.019
4	-0.147
5	0.187
6	-0.138

Figure 263: **Autocorrelations in the irregular**

Heteroskedasticity

A Cochran test on equal variances within each period is performed in the extreme value detection procedure to check if the irregular component is heteroskedastic. In this procedure the standard errors of the irregular component are used for an identification of extreme values. If the null hypothesis (for all the periods (months, quarters) the variances of the irregular component are identical) is rejected, the standard errors will be computed separately for each period. This will happen only if in the option *Calendarsigma*=**signif** has been selected.

Heteroskedasticity (Cochran test on equal variances within each period)

Test statistic	Critical value (5% level)	Decision
0.1303	0.15	Null hypothesis is not rejected.

Figure 264: **Heteroskedasticity**

Moving seasonality ratios (MSR)

For each i^{th} month we will be looking at the mean annual changes for each component by calculating:

$$\overline{S}_i = \frac{1}{N_i - 1} \sum_{t=2}^{N_i} |S_{i,t} - S_{i,t-1}|$$

and

$$\overline{I}_i = \frac{1}{N_i - 1} \sum_{t=2}^{N_i} |I_{i,t} - I_{i,t-1}|$$

where N_i refers to the number of months i in the data, and the moving seasonality ratio of month i:

$$MSR_i = \frac{\overline{I}_i}{\overline{S}_i}$$

The Moving Seasonality Ratio (MSR) is used to measure the amount of noise in the Seasonal-Irregular component. By studying these values, the user can select for each period the seasonal filter that is the most suitable given the noisiness of the series.

Moving Seasonality Ratios (MSR)

Period	I	S	MSR
1	0.0597	0.0211	2.8292
2	0.0808	0.0135	5.9850
3	0.0767	0.0139	5.5038
4	0.0777	0.0262	2.9640

Figure 265: **Moving Seasonality Ratio (MSR)**

Filter length choice

Trend estimation with Henderson Moving average

In iteration B (Table B7), iteration C (Table C7) and iteration D (Table D7 and Table D12) the trend component is extracted from an estimate of the seasonally adjusted series using Henderson moving averages.

The algorithm chooses between different filter lengths automatically according to the I/C ratio, the user can modify this choice (first step is computed with H_{13})

Seasonality extraction filters

In iteration D, Table D10 shows an estimate of the seasonal factors implemented on the basis of the modified SI (Seasonal: Irregular) factors estimated in Tables D4 and D9bis. This component will be smoothed to estimate the seasonal component; depending on the importance of the irregular in the SI component

$$\frac{I}{C} = \frac{\sum_t \left| \frac{\tilde{\imath}_t}{\tilde{\imath}_{t-1}} - 1 \right|}{\sum_t \left| \frac{\tilde{t}_t}{\tilde{t}_{t-1}} - 1 \right|}, \qquad \text{with} \quad \begin{array}{l} \tilde{\imath}_t = \text{temporary irregular} \\ \tilde{t}_t = \text{temporary trend-cycled} \end{array}$$

	Decision rule		
I/C	$[0,1)$	$[1,3.5)$	$[3.5,\infty)$
Henderson filter (m)	9-term	13-term	23-term

Aim:

- dominance of irregular (I/C ratio large) → choose long filter
- dominance of trend-cycle (I/C ratio small) → choose short filter

Figure 266: **Calculation of I-C ratios**

Step 1: Estimating the irregular and seasonal components

An estimate of the seasonal component is obtained by smoothing, month by month and therefore column by column, Table D9bis using a simple 7-term moving average, i.e. of coefficients $\frac{1}{7}\{1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1\}$. In order not to lose three points at the beginning and end of each column, all columns are completed as follows. Let us assume that the column that corresponds to the month is composed of N values $\{x_1,\ x_2,\ x_3,\ \ldots x_{N-1},\ x_N\}$. It will be transformed into a series $\left\{x_{-2}, x_{-1}, x_0, x_1,\ x_2,\ x_3,\ \ldots x_{N-1},\ x_N, x_{N+1},\ x_{N+1},\ x_{N+2},\ x_{N+3}\right\}$ with $x_{-2} = x_{-1} = x_0 = \frac{x_1+x_2+x_3}{3}$ and $x_{N+1} = x_{N+2} = x_{N+3} = \frac{x_N+x_{N-1}+x_{N-2}}{3}$. We then have the required estimates: $S = M_7(D9bis)$ and $I = D9bis - S$.

Step 2: Calculating the Moving Seasonality Ratios

For each i^{th} month the mean annual changes for each component is obtained by calculating

$$\overline{S}_i = \frac{1}{N_i - 1} \sum_{t=2}^{N_i} \left| S_{i,t} - S_{i,t-1} \right|$$

and

$$\overline{I}_i = \frac{1}{N_i - 1} \sum_{t=2}^{N_i} \left| I_{i,t} - I_{i,t-1} \right|$$

where N_i refers to the number of months iin the data, and the moving seasonality ratio of month i:

$$MSR_i = \frac{\overline{I}_i}{\overline{S}_i}$$

These ratios are presented in Detailed Quality Measures

Step 3: Calculating the overall Moving Seasonality Ratio

The overall Moving Seasonality Ratio is calculated as follows:

$$\mathrm{MSR}_i = \frac{\sum_i N_i \ \overline{I}_i}{\sum_i N_i \overline{S}_i}$$

Step 4: Selecting a moving average and estimating the seasonal

component

Depending on the value of the ratio, the program automatically selects a moving average that is applied, column by column (i.e. month by month) to the Seasonal/Irregular component in Table D8 modified, for extreme values, using values in Table D9.

The default selection procedure of a moving average is based on the Moving Seasonality Ratio in the following way:

- If this ratio occurs within zone A (MSR < 2.5), a 3×3 moving average is used; if it occurs within zone C (3.5 < MSR < 5.5), a 3×5 moving average is selected; if it occurs within zone E (MSR 6.5), a 3×9 moving average is used;
- If the MSR occurs within zone B or D, one year of observations is removed from the end of the series, and the MSR is recalculated. If the ratio again occurs within zones B or D, we start over again, removing a maximum of five years of observations. If this does not work, i.e. if we are again within zones B or D, a 3×5 moving average is selected.

The chosen symmetric moving average corresponds, as the case may be 5 (3×3), 7 (3×5) or 11 (3×9 3×9) terms, and therefore does not provide an estimate for the values of seasonal factors in the first 2 (or 3 or 5) and the last 2 (or 3 or 5) years. These are then calculated using associated asymmetric moving averages.

Extreme values: identification and replacement

Though it is recommended rely on the pre-adjustment stage to correct for outliers (transparent method with explicit modelling), X11 has its own (historical) module for identification and treatment of extreme values based on a comparison between the actual and the theoretical value of I.

Stages B and C aim only at correcting for extreme values and contain several iterations of the following steps.

If the irregular is heteroskedastic the standard deviations used for identifying outliers will be computed separately period by period or for distinct groups of several periods.

Step 1: I is estimated once S has been extracted from $S + I$

- for each year the standard deviation σ is computed on the 5 neighbouring years
- I has a theoretical value m, for multiplicative model $m = 1$, $m = 0$ for an additive model
- for a given year y: any point such as $|I_t - m| > 2,5\sigma_y$ is considered as an extreme value and suppressed....
- ...all the yearly sigmas (σ_y) are computed without those points (more robust sigmas)

Step 2: The distance $|I_t - m|$ is computed for each point and evaluated with σ_y as a benchmark, a weight w_t is then assigned to each point, 3 cases:

1) value unchanged

$$|I_t - m| < 1.5\sigma_y \Rightarrow w_t = 1$$

2) value downsized

$$1,5\sigma_y < |I_t - m| < 2,5\sigma_y \Rightarrow w_t = \frac{2.5\sigma_y - |I_t - m|}{2.5\sigma_y - 1.5\sigma_y}$$

3) value removed and replaced

$$|I_t - m| > 2,5\sigma_y \Rightarrow w_t = 0$$

Step3:

Using this weights, a new value of $S + I$ will be computed:

- if $w_t = 1$, $S + I$ remains unchanged for point {t}
- if $w_t < 1$ then the new value of $S + I$ will be an average of $w_t * (S + I)_t$ and the values of $(S + I)$ of the two closest neighbours in the future and in the past with $w = 1$

Seats decomposition

Seats is an ARIMA Model Based (AMB) decomposition method which is the second part of Tramo-Seats seasonal adjustment algorithm, originally developed by the Bank of Spain (Caporello and Maravall 2004).

Seats draws from the ARIMA-model-based (AMB) decomposition framework developed by CLEVELAND, W.P., and TIAO, G.C. (1976), BURMAN, J.P. (1980), HILLMER, S.C., and TIAO, G.C. (1982), BELL, W.R., and HILLMER, S.C. (1984) and MARAVALL, A., and PIERCE, D.A. (1987).

A methodological overview of the entire process can be found in - maravall 1995 - planas book

In recent years, Seats implementation in JDemetra+ v 3.x has been extended to infra-monthly data (weekly, daily, hourly…). Handling this kind of data gave way to a tailored AMB decomposition whose peculiarities are described in this chapter.

In this chapter

This chapter provides details on seats algorithm steps.

The practical implementation as well as all the options, using the graphical user interface or R packages are described in this chapter

Seats steps

Seats decomposes the linearized series into trend, seasonal, transitory and irregular components, provides forecasts for these components, together with the associated standard errors, and finally assign the deterministic effects to each component yielding the final components.

Input from Tramo

In JDemetra+ the input for the model based signal extraction procedure is always provided by Tramo and includes the original series y_t, the linearized series x_t (i.e. the original series y_t with the deterministic effects removed), the ARIMA model for the stochastic (linearized) time series x_t and the deterministic effects (calendar effects, outliers and other regression variable effects).

ARIMA modelling of the input series

One of the fundamental assumptions made by Seats is that the linearized time series x_t follows the ARIMA model:

$$\phi(B)\delta(B)x_t = \theta(B)a_t \tag{0.24}$$

where:

- B – the backshift operator $(Bx_t = x_{t-1})$;
- $\delta(B)$ – a non-stationary autoregressive (AR) polynomial in B (unit roots);
- $\theta(B)$ – an invertible moving average (MA) polynomial in B and in B^S, which can be expressed in the multiplicative form $(1 + \vartheta_1 B + ... + \vartheta q B^q)(1 + \Theta 1 B^s + ... + \Theta_Q B^{\mathrm{sQ}})$;
- $phi(B)$ – a stationary autoregressive (AR) polynomial in B and in B^S containing regular and seasonal unit roots, with *s* representing the number of observations per year;
- a_t – a white-noise variable with the variance $V(a)$.

It should be noted that the stochastic time series can be predicted using its past observations and making an error. The variable a_t, which is assumed to be white noise, is the fundamental *innovation* to the series at time *t*, that is the part that cannot be predicted based on the past history of the series.

Denoting $\varphi(B) = \phi(B)\delta(B)$,can be written in a more concise form as

$$\varphi(B)x_t = \theta(B)a_t \tag{0.25}$$

where $\varphi(B)$ contains both the stationary and the nonstationary roots.

Derivation of the models for the components

Let us consider the additive decomposition model

$$x_t = \sum_{i=1}^{k} x_{\text{it}} \tag{0.26}$$

where *i* refers to the orthogonal components: trend, seasonal, transitory or irregular. Apart from the irregular component, supposed to be a white noise, it is assumed that each component follows the ARIMA model which can be represented, using the notation of Equation 0.61 as:

$$\varphi_i(B)\ x_{\text{it}} = \theta_i(B) a_{\text{it}} \tag{0.27}$$

where

- $\varphi_i(B) = \phi_i(B)\delta_i(B), ...x_{\text{it}}$ is the *i-th* unobserved component,
- $\varphi_i(B)$ and $\theta_i(B)$ are finite polynomials of order p_i and q_i, respectively,
- a_{it}, the disturbance associated with such component, is a white noise process with zero mean and constant variance $V(a_i)$ and a_{it} and a_{jt} are not correlated for $i \neq j$ and for any t..

These disturbances are functions of the innovations in the series and are called "pseudo-innovations" in the literature concerning the AMB decomposition as they refer to the components that are never observed In the JDemetra+ documentation the term "innovations" is used to refer to "pseudo-innovations".(GÓMEZ, V., and MARAVALL, A. (2001a).

The following assumptions hold for Equation 0.27. For each i the polynomials $\phi_i(B)$, $\delta_i(B)$ and $\theta_i(B)$ are prime and of finite order. The roots of $\delta_i(B)$ lies on the unit circle; those of $\phi_i(B)$ lie outside, while all the roots of $\theta_i(B)$ are on or outside the unit circle. This means that nonstationary and noninvertible components are allowed. Since different roots of the AR polynomial induce peaks in the spectrum of the series at different frequencies, and given that different components are associated with the spectral peaks for different frequencies, it is assumed that for $i \neq j$ the polynomials $\phi_i(B)$ and $\phi_j(B)$ do not share any common root (they are coprime). Finally, it is assumed that the polynomials $\theta_i(B),\ i = 1, \ldots, k$ are prime share no unit root in common, guaranteeing the invertibility of the overall series. In fact, since the unit root of $\theta_i(B)$ induce a spectral zero, when the polynomials $\theta_i(B),\ i=1,\dots,k$ share no unit root in common, there is no frequency for which all component spectra become zero.

For description of the spectrum see chapter on Spectral Analysis.(MARAVALL, A. (1995).

Since aggregation of ARIMA models yields ARIMA models, the series x_t will also follow an ARIMA model, as in Equation 0.61, and consequently the following identity can be derived:

$$\frac{\theta(B)}{\varphi(B)}a_t = \sum_{i=1}^{k}\frac{\theta_i(B)}{\varphi_i(B)}a_{\text{it}} \tag{0.28}$$

In the ARIMA model based approach implemented in Seats, the ARIMA model identified and estimated for the observed series x_t is decomposed to derive the models for the components. In particular, the AR polynomials for the components, $\varphi_i(B)$, are easily derived through the factorization of the AR polynomial $\varphi(B)$:

$$\varphi(B) = \prod_{i=1}^{k}\varphi_i(B) \tag{0.29}$$

while the MA polynomials for the components, together with the innovation variances $V(a_i)$, cannot simply be obtained through the relationship:

$$\theta(B)a_t = \sum_{i=1}^{k}\varphi_{\text{ni}}(B)\theta_i(B)a_{\text{it}} \tag{0.30}$$

where $\varphi_{\text{ni}}(B)$ is the product of all $\varphi_j(B),\ j=1,\dots,k$, except from $\varphi_i(B)$. Further assumptions are therefore needed to cope with the under identification problem: i) $p_i \ge q_i$ and ii) the canonical decomposition, i.e. the decomposition that allocate all additive white noise to the irregular component (yielding non invertible components except the irregular).

To understand how Seats factorizes the AR polynomials, first a concept of a root will be explored.(Description based on KAISER, R., and MARAVALL, A. (2000) and MARAVALL, A. (2008c).)

Equation 0.61 can be expressed as:

$$\psi^{-1}(B)x_t = a_t(1+\varphi_1 B + ... \varphi_p B^p)x_t = (1+\theta_1 B + ... \theta_q B^q)a_t \tag{0.31}$$

Let us now consider Equation 0.61 in the inverted form:

$$\theta(B)y_t = \varphi(B)a_t \tag{0.32}$$

If both sides of Equation 0.31 multiplied by x_{t-k} with $k > q$, and expectations are taken, the right hand side of the equation vanishes and the left hand side becomes:

$$\varphi(B)\gamma_k = \gamma_k + \varphi_1\gamma_{k-1} + ... \varphi_p\gamma_{k-p} = 0 \tag{0.33}$$

where B operates on the subindex k.

The autocorrelation function γ_k is a solution of Equation 0.33 with the characteristic equation:

$$z^p + \varphi_1 z^{p-1} + ... \varphi_{p-1}z + \varphi_p = 0 \tag{0.34}$$

If z_1,..., z_p are the roots of Equation 0.34, the solutions of Equation 0.33 can be expressed as:

$$\gamma_k = \sum_{i=1}^{p} z_i^k \tag{0.35}$$

and will converge to zero as $karrow\infty$ when $|r_i| < 1,\ i = 1, \ldots, p.$ From Equation 0.33 and Equation 0.35 it can be noticed that $z_1 = B_i^{-1}$, meaning that z_1,..., z_p are the inverses of the roots $B_1, \ldots, B_p$ of the polynomial $\varphi(B)$. The convergence of γ_k implies that the roots of the $\varphi(B)$ are larger than 1 in modulus (lie outside the unit circle). Therefore, from the Equation 0.36.

$$\varphi(B)^{-1} = \frac{1}{(1-z_1)\ldots(1-z_1)} \tag{0.36}$$

it can be derived that $\varphi(B)^{-1}$ is convergent and all its inverse roots are less than 1 in modulus.

Equation 0.34 has real and complex roots (solutions). Complex number $x = a+bi$, with a and b both real numbers, can be represented as $x = r(cos(\omega)+i\ sin(\omega))$, where i is the imaginary unit $i^2 = -1$), r is the modulus of x, that is $r = |x| = \sqrt{a^2+b^2}$ and ω is the argument (frequency). When roots are complex, they are always in pairs of complex conjugates. The representation of the complex number $x = a + bi$ has a geometric interpretation in the complex plane established by the real axis and the orthogonal imaginary axis.

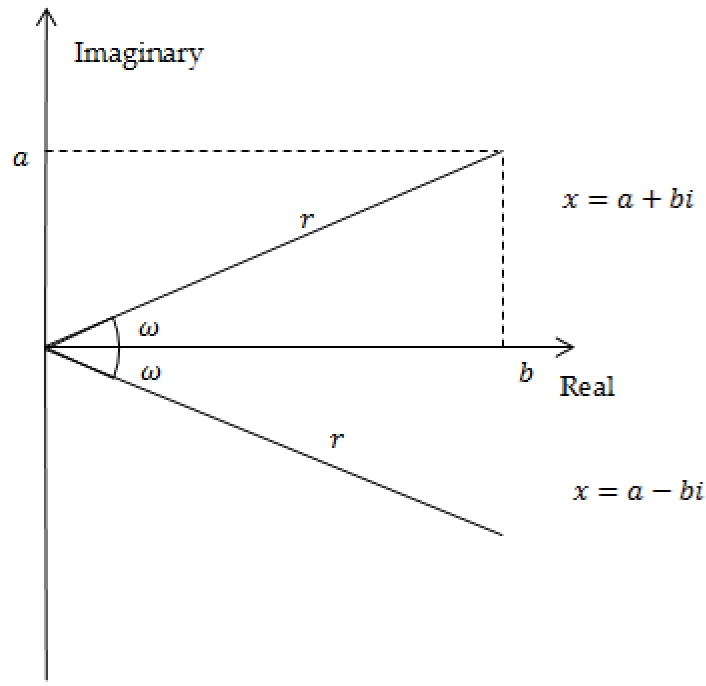


Figure 267: **Geometric representation of a complex number and of its conjugate**

Representing the roots of the characteristic Equation 0.34 in the complex plane enhances understanding how they are allocated to the components. When the modulus r of the roots in z are greater than 1 (i.e. modulus of the roots in $\varphi(B) < 1$), the solution of the characteristic equation has a systematic explosive process, which means that the impact of the given impulse on the time series is more and more pronounced in time. This behaviour is not in line with the developments that can be identified in actual economic series. Therefore, the models estimated by Tramo-Seats (and X-13ARIMA-Seats) have never inverse roots in B with modulus greater than 1.

The characteristic equations associated with the regular and the seasonal differences have roots in $\varphi(B)$ with modulus $r = 1$. They are called non-stationary roots and can be represented on the unit circle. Let us consider the seasonal differencing operator applied to a quarterly time series $(1-B^4)$. Its characteristic equation is $(z^4-1)=0$ with solutions given by$z=\sqrt[4]{1}$, i.e. $z_{1,2}=\pm 1$ and $z_{3,4}=\pm i1$. The first two solutions are real and the last two are complex conjugates. They are represented by the black points on the unit circle on the figure below.

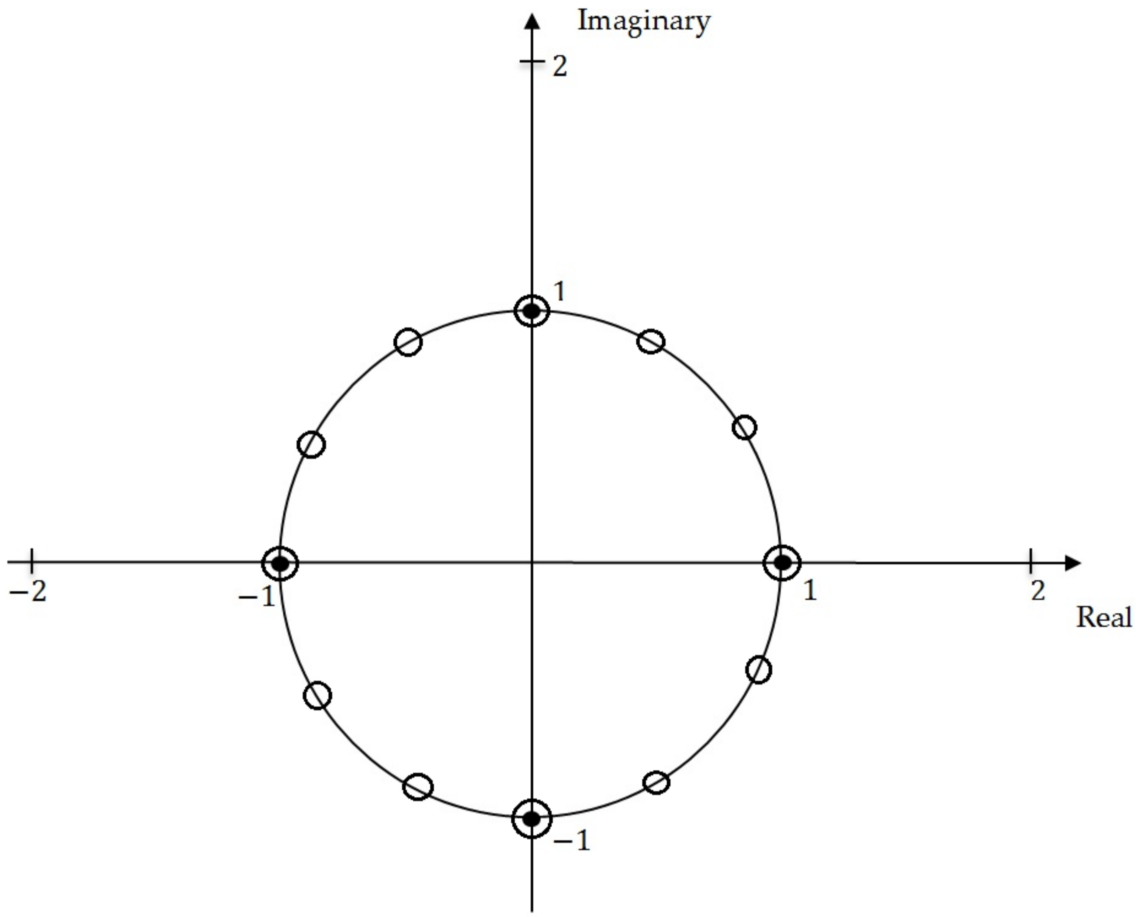


Figure 268: **Unit roots on the unit circle**

For the seasonal differencing operator $(1-B^{12})$ applied to the monthly time series the characteristic equation $(z^{12}-1)=0$ has twelve non-stationary solutions given by $z=\sqrt[12]{1}$: two real and ten complex conjugates, represented by the white circles in unit roots figure above.

The complex conjugates roots generate the periodic movements of the type:

$$z_t = A^t \cos(\omega t + W) \tag{0.37}$$

where:

- A – amplitude;
- ω – angular frequency (in radians);
- W – phase (angle at $t = 0$).

The frequency f, i.e. the number of cycles per unit time, is $\frac{\omega}{2\pi}$. If it is multiplied by *s*, the number of observations per year, the number of cycles completed in one year is derived. The period of function in Equation 0.37, denoted by τ, is the number of units of time (months/quarters) it takes for a full circle to be completed.

For quarterly series the seasonal movements are produced by complex conjugates roots with angular frequencies at $\frac{\pi}{2}$ (one cycle per year) and π (two cycles per year). The corresponding number of cycles per year and the length of the movements are presented in the table below.

Seasonal frequencies for a quarterly time series

Angular frequency (ω)	**Frequency (cycles per unit time) (f)**	**Cycles per year**	**Length of the movement measured in quarters (τ)**
$\frac{\pi}{2}$	0.25	1	4
π	0.5	2	2

For monthly time series the seasonal movements are produced by complex conjugates roots at the angular frequencies: $\frac{\pi}{6}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$, $\frac{2\pi}{3}$, $\frac{5\pi}{6}$ and π. The corresponding number of cycles per year and the length of the movements are presented in the table below:

Seasonal frequencies for a monthly time series

Angular frequency (ω)	**Frequency (cycles per time unit) (f)**	**Cycles per year**	**Length of the movement measured in months (τ)**
$\frac{\pi}{6}$	0.083	1	12
$\frac{\pi}{3}$	0.167	2	6

Angular frequency (ω)	Frequency (cycles per time unit) (f)	Cycles per year	Length of the movement measured in months (τ)
$\frac{\pi}{2}$	0.250	3	4
$\frac{2\pi}{3}$	0.333	4	3
$\frac{5\pi}{6}$	0.417	5	2.4
π	0.500	6	2

In JDemetra+ Seats assigns the roots of the AR full polynomial to the components according to their associated modulus and frequency (For details see MARAVALL, A., CAPORELLO, G., PÉREZ, D., and LÓPEZ, R. (2014))

- Roots of $(1-B)^d$ are assigned to trend component.
- Roots of $(1-B^s)^{d_s} = ((1-B)(1+B+...+B^{s-1}))^{d_s}$ are assigned to the trend component (root of $(1-B)^{d_s}$) and to the seasonal component (roots of $(1+B+...+B^{s-1})^{d_s}$).
- When the modulus of the inverse of a real positive root of $\varphi(B)$ is greater than k or equal to k, where k is the threshold value controlled by the *Trend boundary* parameter, then the root is assigned to the trend component. Otherwise it is assigned to the transitory component.
- Real negative inverse roots of $\phi_p(B)$ associated with the seasonal two-period cycle are assigned to the seasonal component if their modulus is greater than *k*, where k is the threshold value controlled by the *Seasonal boundary* and the *Seas. boundary (unique)* parameters. Otherwise they are assigned to the transitory component.
- Complex roots, for which the argument (angular frequency) is close enough to the seasonal frequency are assigned to the seasonal component. Closeness is controlled by the *Seasonal tolerance* and *Seasonal tolerance (unique)* parameters. Otherwise they are assigned to the transitory component.
- If d_s (seasonal differencing order) is present and Bphi < 0 (Bphi is the estimate of the seasonal autoregressive parameter), the real positive inverse root is assigned to the trend component and the other ($s-1$) inverse roots are assigned to the seasonal component. When $d_s = 0$, the root is assigned to the seasonal when Bphi < -0.2 and/or the overall test for seasonality indicates presence of seasonality. Otherwise it goes to the transitory component. Also, when Bphi > 0, roots are assigned to the transitory component.

It should be highlighted that when $Q > P$, where Q and P denote the orders of the polynomials $\varphi(B)$ and $\theta(B)$, the Seats decomposition yields a pure MA $(Q-P)$ component (hence transitory). In this case the transitory component will appear even when there is no AR factor allocated to it.

Once these rules are applied, the factorization of the AR polynomial presented by Equation 0.61 yields to the identification of the AR polynomials for the components which contain, respectively, the AR roots associated with the trend component, the seasonal component and the transitory component.

The AR roots close to or at the trading day frequency generates a stochastic trading day component. A stochastic trading day component is always modelled as a stationary ARMA(2,2), where the AR part contains the roots close to the TD frequency, and the MA(2) is obtained from the model decomposition (MARAVALL, A., and PÉREZ, D. (2011)). This component, estimated by Seats, is not implemented by the current version of JDemetra+.

Then with the partial fraction expansion the spectrum of the final components are obtained.

For example, the Airline model for a monthly time series:

$$(1-B)(1-B^{12})x_t = (1+\theta_1 B)(1+\Theta_1 B^{12})\, a_t \tag{0.38}$$

is decomposed by Seats into the model for the trend component:

$$(1-B)(1-B)c_t = (1+\theta_{c,1}B+\theta_{c,2}B^2)a_{c,t} \tag{0.39}$$

and the model for the seasonal component:

$$(1+B+...+B^{11})s_t = (1+\theta_{s,1}B+...+\theta_{s,11}B^{11})a_{s,t}, \tag{0.40}$$

As a result, the Airline model is decomposed as follows:

$$\frac{(1+\theta_1 B)(1+\Theta_1 B^{12})}{(1-B)(1-B)}a_t = \frac{(1+\theta_{s,1}B+...+\theta_{s,11}B^{11})}{(1+B+...+B^{11})}a_{s,t} + \frac{(1+\theta_{c,1}B+\theta_{c,2}B^2)}{(1-B)(1-B)}a_{c,t} + \tag{0.41}$$

The transitory component is not present in this case and the irregular component is the white noise.

The partial fractions decomposition is performed in a frequency domain. In essence, it consists in portioning of the pseudo-spectrum of x_t into additive spectra of the components. When the AMB decomposition of the ARIMA model results in the non-negative spectra for all components, the decomposition is called admissible. In such case an infinite number of admissible decompositions exists, i.e. decompositions that yield the non-negative spectra of all components.

Therefore, the MA polynomials and the innovation variances cannot be yet identified from the model of x_t. As sketched above, to solve this under identification problem and identify a unique decomposition, it is assumed that for each component the order of the MA polynomial is no greater than the order of the AR polynomial and the canonical solution of S.C. Hillmer and G.C. Tiao is applied, i.e. all additive white noise is added to the irregular component As a consequence all components derived from the canonical decomposition, except from the irregular, have a spectral minimum of zero and are thus non invertible.

Given the stochastic features of the series, it can be shown by that the canonical decomposition produces as stable as possible trend and seasonal components since it maximizes the variance of the irregular and minimizes the variance of the other components. However, there is a price to be paid as canonical components can produce larger revisions in the preliminary estimators of the component than any other admissible decomposition.

The term pseudo-spectrum is used for a non-stationary time series, while the term spectrum is used for a stationary time series.

If the ARIMA model estimated in Tramo does not accept an admissible decomposition, Seats replaces it with a decomposable approximation. The modified model is therefore used to decompose the series. There are also other rare situations when the ARIMA model chosen by Tramo is changed by Seats. It happens when, for example, the ARIMA models generate unstable seasonality or produce a senseless decomposition. Such examples are discussed by MARAVALL, A. (2009).

HILLMER, S.C., and TIAO, G.C. (1982).

GÓMEZ, V., and MARAVALL, A. (2001a).

HILLMER, S.C., and TIAO, G.C. (1982).

MARAVALL, A. (1986).

The figure below represents the pseudo-spectrum for the canonical trend and an admissible trend.

A pseudo-spectrum is denoted by$g_i(\omega)$, where ω represents the angular frequency. The pseudo-spectrum of x_{it} is defined as the Fourier transform of ACGF of x_t which is expressed as:

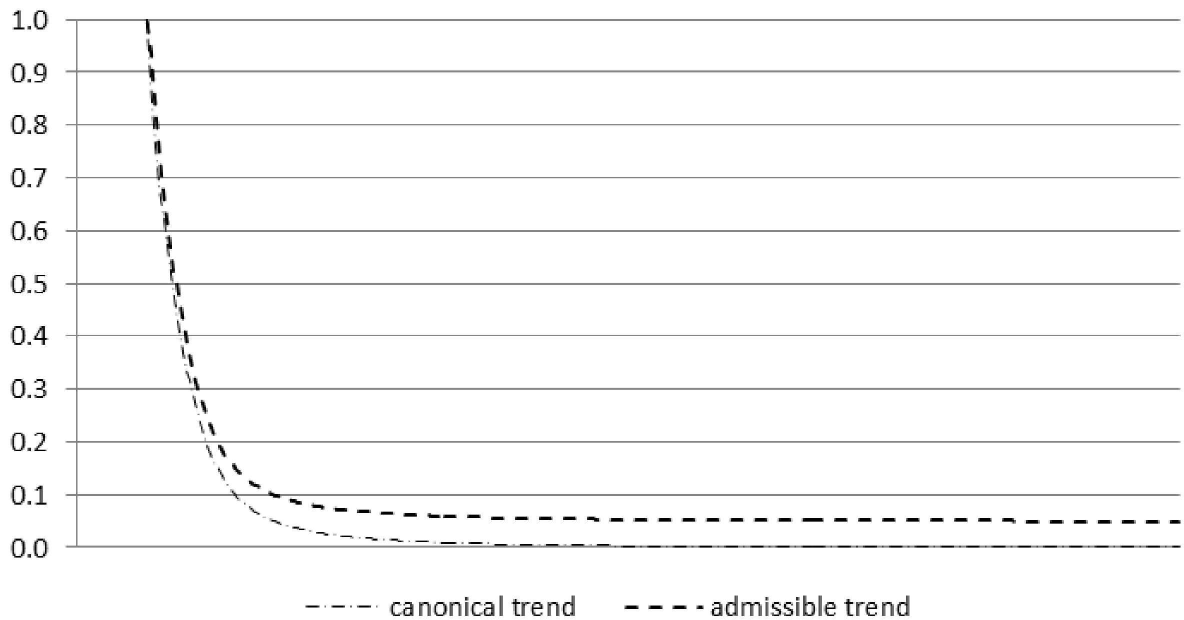


Figure 269: **A comparison of canonical trend and admissible trend**

$$\frac{\psi_i(B)\psi_i(F)}{\delta_i(B)\delta_i(F)}V(a_i) \tag{0.42}$$

where:

- $\psi_i(F) = \frac{\theta_i(F)}{\phi_i(F)}$
- $\psi_i(B) = \frac{\theta_i(B)}{\phi_i(B)}$

A pseudo-spectrum for a monthly time series x_t is presented in the figure below: The pseudo-spectrum for a monthly series. The frequency $\omega = 0$ is associated with the trend, frequencies in the range $[0 + \epsilon_1,\ \frac{\pi}{6} - \epsilon_2]$ with $[0 + \epsilon_1,\ \frac{\pi}{6} - \epsilon_2]$ $\epsilon_1,\ \epsilon_2 > 0$ and $\epsilon_1 <\ \frac{\pi}{6} - \epsilon_2$ are usually associated with the business-cycle and correspond to a period longer than a year and bounded.

The frequencies in the range $[\frac{\pi}{6}, \pi]$ are associated with the short term movements, whose cycle is completed in less than a year. If a series contains an important periodic component, its spectrum reveals a peak around the corresponding frequency and in the ARIMA model it is captured by an AR root. In the example below spectral peaks occur at the frequency $\omega = 0$ and at the seasonal frequencies ($\frac{\pi}{6}$, $\frac{2\pi}{6},\ \frac{3\pi}{6},\ \frac{4\pi}{6}, \frac{5\pi}{6}, \pi$).

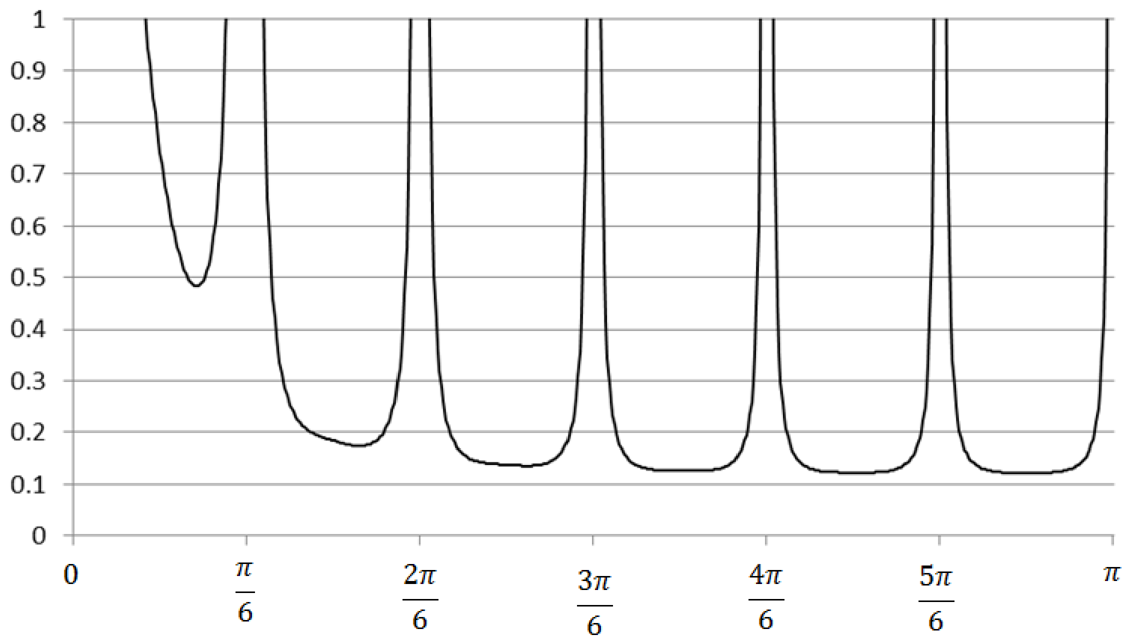


Figure 270: **The pseudo-spectrum for a monthly series**

In the decomposition procedure, the pseudo-spectrum of the time series x_t is divided into the spectra of its components (in the example figure below, four components were obtained).

Estimation of the components

The various components are estimated using Wiener-Kolmogorow (WK) filters. JDemetra+ includes three options to estimate the WK filter, namely *Burman*, *KalmanSmoother* and *MCElroyMatrix*[^m-seats-decomposition_old-12]. Here the first of above mentioned options, proposed by BURMAN, J.P. (1980) will be explained.

The estimation procedure and the properties of the WK filter are easier to explain with a two-component model. Let the seasonally adjusted series (s_t) be the signal of interest and the seasonal component (n_t) be the remainder, "the noise". The series is given by the model in Equation 0.61 and from Equation 0.27 the models for theoretical components are:

$$\varphi_s(B)s_t = \theta_s(B)a_{\text{st}} \tag{0.43}$$

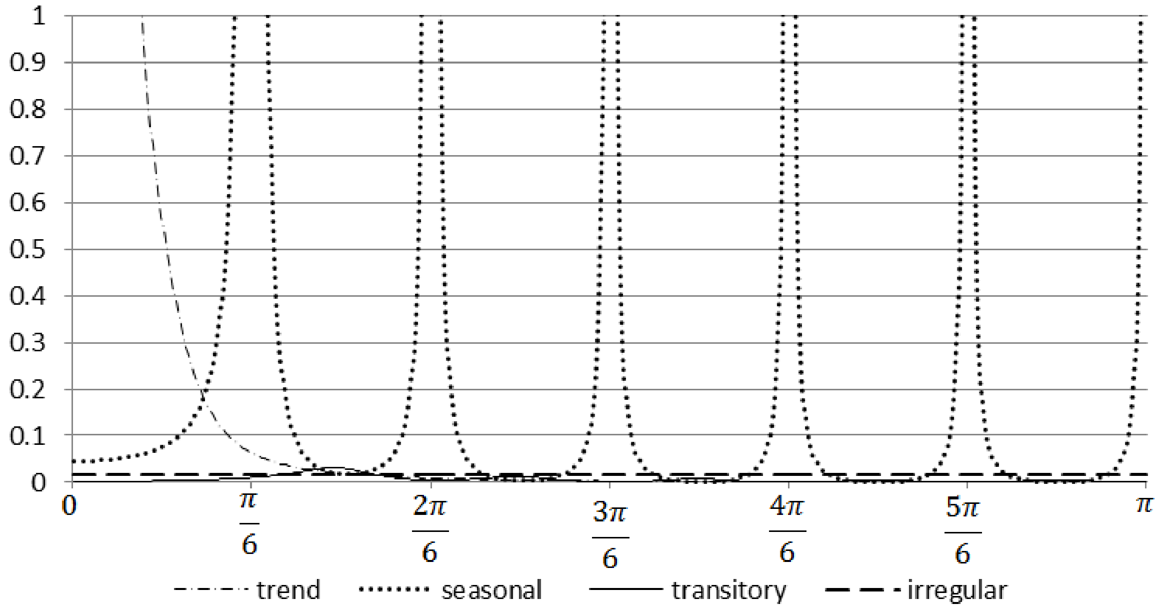


Figure 271: **The pseudo-spectra for the components**

and

$$\varphi_n(B)n_t = \theta_n(B)a_{\text{nt}} \tag{0.44}$$

From Equation 0.29 and Equation 0.30 it is clear that $\varphi(B) = \varphi_s(B)\varphi_n(B)$ and $\theta(B)a_t = \theta_s(B)a_{\text{st}} + \theta_n(B)a_{\text{nt}}$.

As the time series components are never observed, their estimators have to be used. Let us note X_T an infinite realization of the time series x_t. Seats computes the Minimum Mean Square Error (MMSE) estimator of s_t, e.g. the estimator $\widehat{s}_t$ that minimizes $E[(s_t - \widehat{s}_t)^2 | X_T)]$. Under the normality assumption $\widehat{s}_{t|T}$ is also equal to the conditional expectation $E(s_t|X_T)$, so it can be presented as a linear function of the elements in X_T.WHITTLE (1963) shows that the MMSE estimator of $\widehat{s}_t$ is:

$$\widehat{s}_t = k_s \frac{\psi_s(B)\psi_s(F)}{\psi(B)\psi(F)} x_t \tag{0.45}$$

where

- $\psi(B) = \frac{\theta(B)}{\phi(B)}$,
- $k_s = \frac{V(a_s)}{V(a)}$,

$V(a_s)$ is the variance of a_{st} and $V(a)$ is the variance of a_t.

Expressing the $\psi(B)$ polynomials as functions of the AR and MA polynomials, after cancellation of roots, the estimator of s_t can be expressed as:

$$\hat{s}_t = k_s \frac{\theta_s(B)\theta_s(F)\varphi_n(B)\delta_n(B)\varphi_n(F)\delta_n(F)}{\theta(B)\theta(F)} x_t \tag{0.46}$$

where:

$$\nu_s(B,F) = k_s \frac{\theta_s(B)\theta_s(F)\varphi_n(B)\delta_n(B)\varphi_n(F)\delta_n(F)}{\theta(B)\theta(F)} \tag{0.47}$$

is a WK filter.

Equation 0.47 shows that the WK filter is two-sided (uses observations both from the past and from the future), centred (the number of points in the past is the same as in the future) and symmetric (for any k the weight applied to x_{t-k} and x_{t+k} is the same), which allows the phase effect to be avoided. Due to invertibility of $\theta(B)$ (and $\theta(F)$) the filter is convergent in the past and in the future.

The estimator can be presented as

$$\hat{s}_t = \nu_i(B,F)x_t \tag{0.48}$$

where $\nu_i(B,F) = \nu_0 + \sum_{j=1}^{\infty} \nu_{ij}(B^j + F^j)$ is the WK filter.

The example of the WK filters obtained for the pseudo-spectra of the series illustrated above is shown on the figure below: WK filters for components.

The WK filter from Equation 0.47 can also be expressed as a ratio of two pseudo-autocovariance generating functions (p-ACGF). The p-ACGF function summarizes the sequence of absolutely summable autocovariances of a stationary process x_t (see Spectral Analysis).

The ACGF function of an ARIMA process is expressed as:

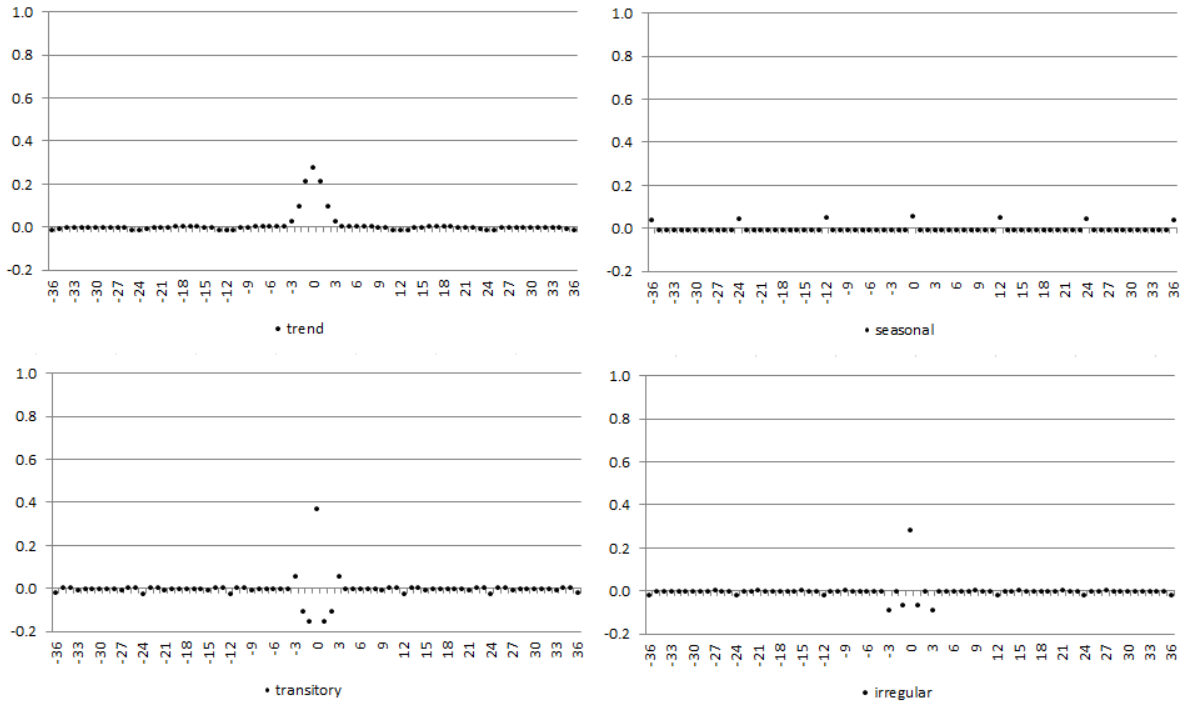


Figure 272: **WK filters for components**

$$acgf(B) = \frac{\theta(B)\theta(F)}{\phi(B)\delta(B)\phi(F)\delta(F)}V(a) \tag{0.49}$$

And, the WK filter can be rewritten as:

$$\nu_s(B,F) = \frac{\gamma_s(B,F)}{\gamma(B,F)} \tag{0.50}$$

where:

- $\gamma_s(B,F) = \frac{\theta_s(B)\theta_s(F)}{\phi_s(B)\delta_s(B)\phi_s(F)\delta_s(F)}V(a_s)$ is the p-ACGF of s_t;
- $\gamma(B,F) = \frac{\theta(B)\theta(F)}{\phi(B)\delta(B)\phi(F)\delta(F)}V(a)$ is the p-ACGF of x_t.

From Equation 0.47 it can be seen that the WK filter depends on both the component and the series models. Consequently, the estimator of the component and the WK filter reflect the characteristic of data and by construction, the WK filter adapts itself to the series under consideration. Therefore, the ARIMA model is of particular importance for the Seats method. Its misspecification results in an incorrect decomposition.

This adaptability, if the model has been correctly determined, avoids the dangers of under and overestimation with an ad-hoc filtering. For example, for the series with a highly stochastic seasonal component the filter adapts to the width of the seasonal peaks and the seasonally adjusted series does not display any spurious seasonality. Examples of WK filters for stochastic and stable seasonal components are presented on the figure below. (MARAVALL, A. (1995)).

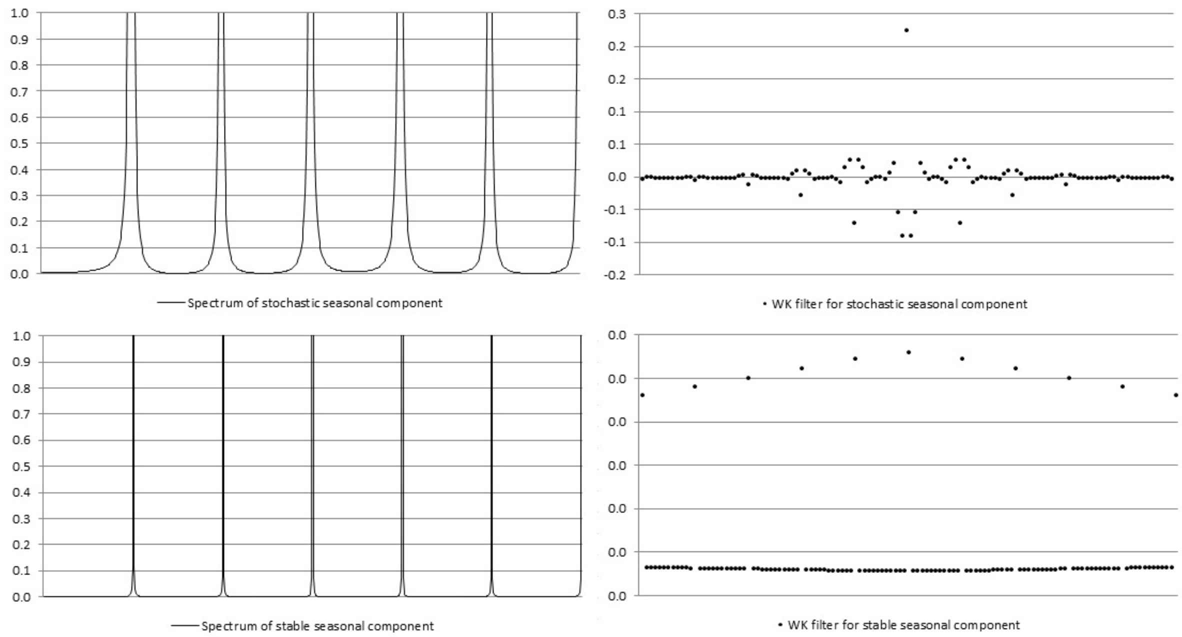


Figure 273: **WK filters for stable and stochastic seasonal components**

The derivation of the components requires an infinite realization of x_t in the direction of the past and of the future. However, the convergence of the WK filter guarantees that, in practice, it could be approximated by a truncated (finite) filter and, in most applications, for large k the estimator for the central periods of the series can be safely seen as generated by the WK filter (MARAVALL, A., and PLANAS, C. (1999)).

$$\hat{s}_t = \nu_k x_{t-k} + ... + \nu_0 x_t + ... + \nu_k x_{t+k} \tag{0.51}$$

When $T > 2L + 1$, where T is the last observed period, and L is an a priori number that typically expands between 3 and 5 years, the estimator expressed by Equation 0.46 can be assumed as the final (historical) estimator for the central observations of the series[2]. In practice, the Wiener-Kolmogorov filter is applied to x_t extended with forecasts and backcasts from the ARIMA model. The final or historical estimator of $\hat{s}_t$, is obtained with a doubly infinite filter, and therefore contains an error e_{st} called final estimation error, which is equal $e_{st} = s_t$ associated with the regular and the seasonal differences have roots in $\varphi(B)$ with modulus $r = 1$. They are called non-stationary roots and can be represented on the unit circle. Let us consider the seasonal differencing operator applied to a quarterly time series $(1 - B^4)$. Its characteristic equation is $(z^4 - 1) = 0$ with solutions given by $z = \sqrt[4]{1}$, i.e. $z_{1,2} = \pm 1$ and $z_{3,4} = \pm i1$. The first two solutions are real and the last two are complex conjugates. They are represented by the black points on the unit circle on the figure below.

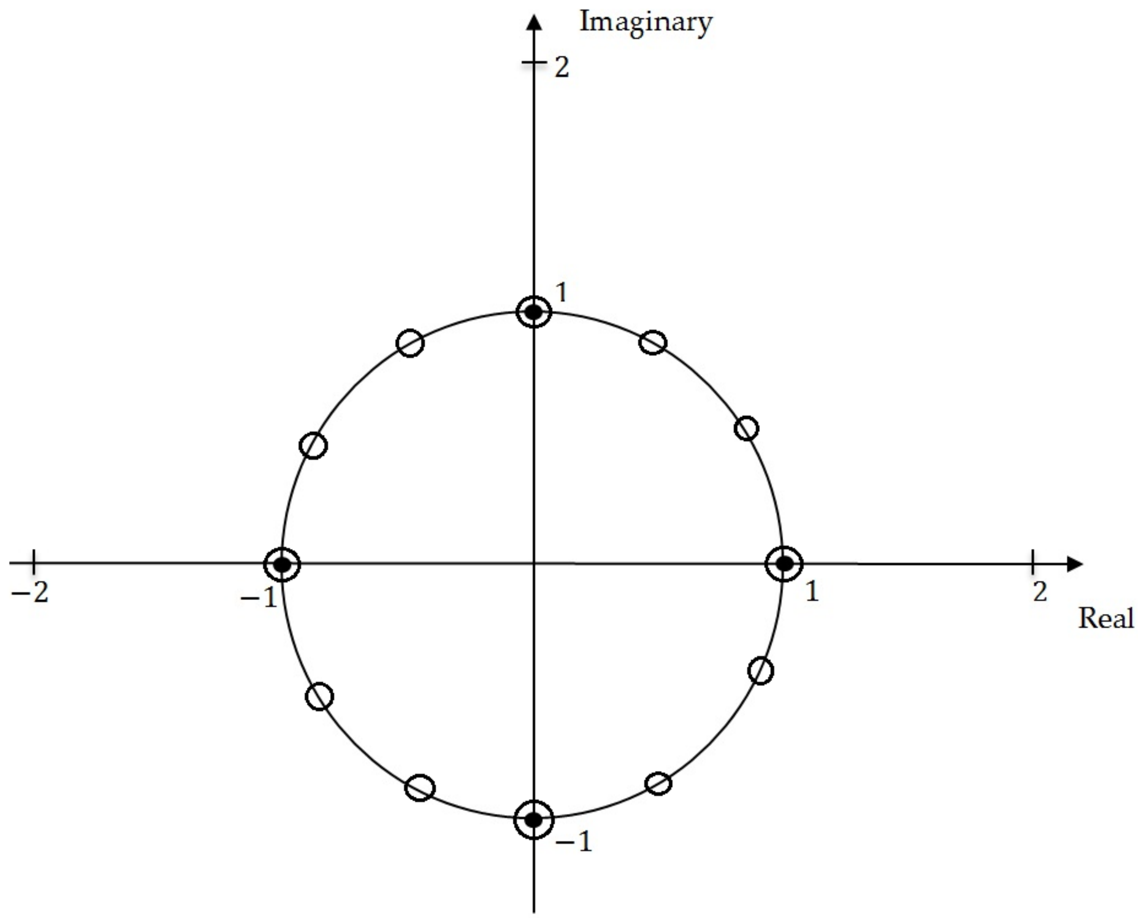


Figure 274: **Unit roots on the unit circle**

[2] MARAVALL, A. (1998).

In the frequency domain, the Wiener-Kolmogorov filter $\nu(B, F)$ that provides the final estimator of s_t is expressed as the ratio of the s_t and x_t pseudo-spectra:

$$\tilde{\nu}(\omega) = \frac{g_s(\omega)}{g_x(\omega)} \tag{0.52}$$

The function $\tilde{\nu}(\omega)$ is also referred as the gain of the filter. GÓMEZ, V., and MARAVALL, A. (2001a) show that when for some frequency the signal (the seasonally adjusted series) dominates the noise (seasonal fluctuations) the gain $\tilde{\nu}(\omega)$ approaches 1. On the contrary, when for some frequency the noise dominates the gain $\tilde{\nu}(\omega)$ approaches 0.

The spectrum of the estimator of the seasonal component is expressed as:

$$g_{\hat{s}}(\omega) = [\frac{g_s(\omega)}{g_x(\omega)}]^2 g_x(\omega) \tag{0.53}$$

where

- $[\tilde{\nu}(\omega)]^2 = [\frac{g_s(\omega)}{g_x(\omega)}]^2 = [\frac{g_s(\omega)}{g_s(\omega)+g_n(\omega)}]^2 = [\frac{1}{1+\frac{1}{r(\omega)}}]^2$ is the squared gain of the filter ;
- $r(\omega) = \frac{g_s(\omega)}{g_n(\omega)}$ represents the signal-to-noise ratio.

For each ω, the MMSE estimation gives the signal-to-noise ratio. If this ratio is high, then the contribution of that frequency to the estimation of the signal will be also high. Assume that the trend is a signal that needs to be extracted from a seasonal time series. Then $R(0) = 1$ and the frequency $\omega = 0$ will only be used for trend estimations. For seasonal frequencies $R(\omega) = 0$, so that these frequencies are ignored in computing the trend resulting in spectral zeros in $g_{\hat{s}}(\omega)$. For this reason, unlike the spectrum of the component, the component spectrum contains dips as it can be seen on the figure below: Component spectrum and estimator spectrum for trend.

From the Equation 0.52 it is clear that the squared gain of the filter determines how the variance of the series contributes to the variance of the seasonal component for the different frequencies. When $\tilde{\nu}(\omega) = 1$, the full variation of x_t for that frequency is passed to $\hat{s}_t$, while if $\tilde{\nu}(\omega) = 0$ the variation of x_t for that frequency is fully ignored in the computation of $\hat{s}_t$. These two cases are well illustrated by the figure below that shows the square gain of the WK filter for two series already analysed in the figure above (Figure: WK filters for stable and stochastic seasonal components).

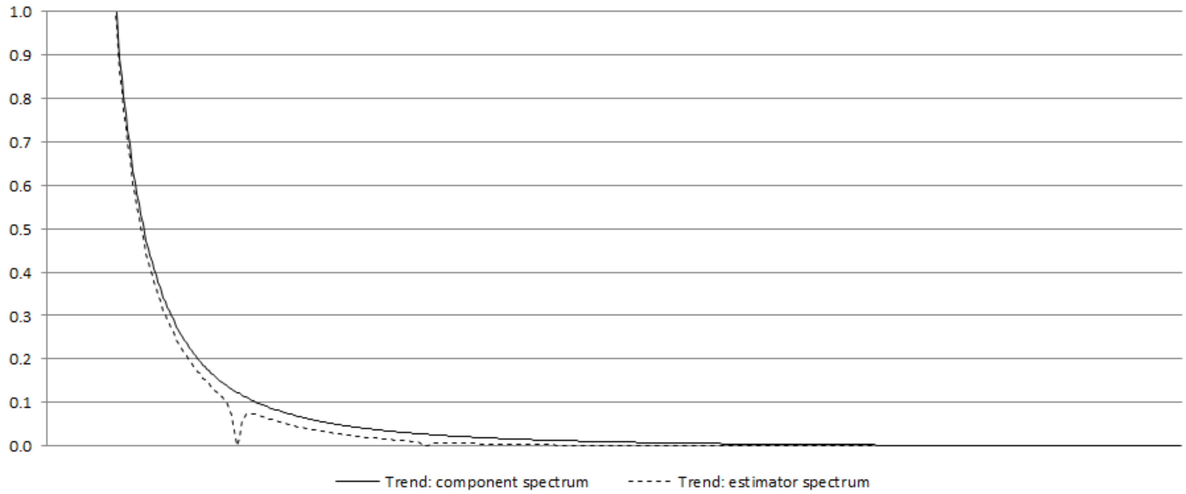


Figure 275: **Component spectrum and estimator spectrum for trend**

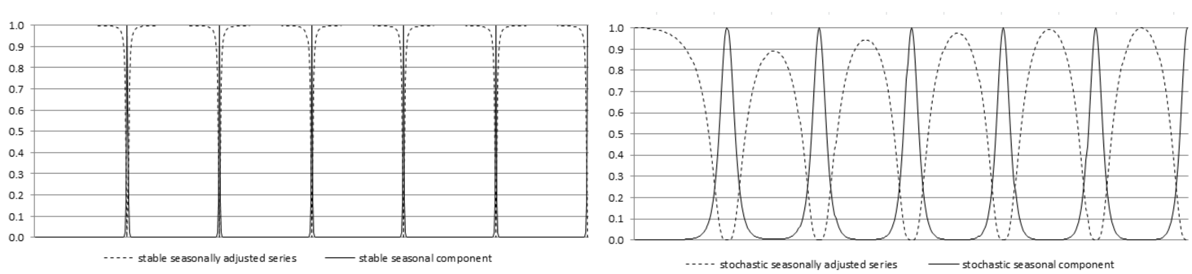


Figure 276: **The squared gain of the WK filter for stable and stochastic seasonal components.**

Since $r(\omega) \geq 0$, then $\tilde{\nu}(\omega) \leq 1$ and from Equation 0.52 it can be derived that $g_{\hat{s}}(\omega) = \tilde{\nu}(\omega) g_s(\omega)$. As a result, the estimator will always underestimate the component, i.e. it will be always more stable that the component.

Since $g_{\widehat{n}}(\omega) < g_n(\omega)$ and $g_{\hat{s}}(\omega) < g_s(\omega)$ the expression: $g_x(\omega) - [g_{\widehat{n}}(\omega) + g_{\hat{s}}(\omega)] \geq 0$ is the cross-spectrum. As it is positive, the MMSE yields correlated estimators. This effect emerges since variance of estimator is smaller than the variance of component. Nevertheless, if at least one non-stationary component exists, cross-correlations estimated by Tramo-Seats will tend to zero as cross-covariances between estimators of the components are finite. In practice, the inconvenience caused by this property will likely be of little relevance.

Preliminary estimators for the components

GÓMEZ, V., and MARAVALL, A. (2001a) point out that *the properties of the estimators have been derived for the final (or historical) estimators. For a finite (long enough) realization, they can be assumed to characterize the estimators for the central observations of the series, but for periods close to the beginning of the end the filter cannot be completed and some preliminary estimator has to be used*. Indeed, the historical estimator shown in Equation 0.51 is obtained for the central periods of the series. However, when t approaches T (last observation), the WK filter requires observations, which are not available yet. For this reason a preliminary estimator needs to be used.

To introduce preliminary estimators let us consider a semi-finite realization $[x_{-\infty}$,...$ x_{T}]$, where T is the last observed period. The preliminary estimator of x_{it} obtained at T, $(T - t = k \geq 0)$ can be expressed as

$$\hat{x}_{it|t+k} = \nu_i(B, F) x^e_{t|T} \tag{0.54}$$

where

- $\nu_i(B, F)$ is the WK filter ;
- $x^e_{t|T}$ is the extended series, such that $x^e_{t|T} = x_t$ for $t \leq T$ and $x^e_{t|T} = \hat{x}_{t|T}$ for $t > T$, where $\hat{x}_{t|T}$ denotes the forecast of x_t obtained at period T.

The future k values necessary to apply the filter are not yet available and are replaced by their optimal forecasts from the ARIMA model on x_t. When $k = 0$ the preliminary estimator becomes the concurrent estimator. As the forecasts are linear functions of present and past observations of x_t, the preliminary estimator $\hat{x}_{it}$ will be a truncated asymmetric filter applied to x_t that generates a phase effect (KAISER, R., and MARAVALL, A. (2000)).

When a new observation x_{T+1} becomes available the forecast $\hat{x}_{T+1|T}$ is replaced by the observation and the forecast $\hat{x}_{iT+j|T}$, $j > 1$ are updated to $x_{T+j|T+1}$ resulting in the revision error (MARAVALL, A. (1995)). The total error in the preliminary estimator $d_{it|t+k}$ is expressed as a sum of the final estimation error (e_{it}) and the revision error ($r_{it|t+k}$), i.e.:

$$d_{it|t+k} = x_{it} - \hat{x}_{it|t+k} = (x_{it} - \hat{x}_{it}) + (\hat{x}_{it} - \hat{x}_{it|t+k}) = e_{it} + r_{it|t+k} \quad (0.55)$$

where:

- $x_{it} - i^{th}$ component;
- $\hat{x}_{it|t+k}$- the estimator of x_{it} when the last observation is x_{t+k}.

Therefore the preliminary estimator is subject not only to the final error but also to a revision error, which are orthogonal to each other (MARAVALL, A. (2009)). The revision error decreases as k increases, until it can be assumed equal to 0 for large enough k.

It's worth remembering that Seats estimates the unobservable components of the time series so the "true" components are never observed. Therefore, MARAVALL, A. (2009) stresses that *the error in the historical estimator is more of academic rather than practical interest. In practice, interest centres on revisions. (...) the revision standard deviation will be an indicator of how far we can expect to be from the optimal estimator that will be eventually attained, and the speed of convergence of $\theta(B)^{-1}$ will dictate the speed of convergence of the preliminary estimator to the historical one.* The analysis of an error is therefore useful for making decision concerning the revision policy, including the policy for revisions and horizon of revisions.

PsiE-weights

The estimator of the component is calculated as $\hat{x}_{it} = \nu_s(B,F)x_t$. By replacing $x_{it} = \frac{\theta(B)}{\gamma(B)\delta(B)}a_t$, the component estimator can be expressed as (KAISER, R., and MARAVALL, A. (2000)):

$$\hat{x}_{it} = \xi_s(B,F)a_t \quad (0.56)$$

where $\xi_s(B,F) = ... + \xi_j B^j + ... + \xi_1 B + \xi_0 + \xi_{-1} F ... \xi_{-j} F^j + ...$

This representation shows the estimator as a filter applied to the innovation a_t, rather than on the series x_t. Hence, the filter from Equation 0.55 can be divided into two components: the first one, i.e. $\ldots + \xi_j B^j + \ldots + \xi_1 B + \xi_0$, applies to prior and concurrent innovations, the second one, i.e. $\xi_{-1} F + \ldots + \xi_{-j} F^j$ applies to future (i.e. posterior to t) innovations. Consequently, ξ_j determines the contribution of a_{t-j} to $\hat{s}_t$ while ξ_{-j} determines the contribution of a_{t+j} to $\hat{s}_t$. Finally, the estimator of the component can be expressed as:

$$\hat{x}_{it} = \xi_i(B)^{-} a_t + \xi_i(F)^{+} a_{t+1} \tag{0.57}$$

where:

- $\xi_i(B)^{-} a_t$ is an effect of starting conditions, present and past innovations in series;
- $\xi_i(F)^{+} a_{t+1}$ is an effect of future innovations.

For the two cases already presented in figure *WK filters for stable and stochastic seasonal components* and figure *The squared gain of the WK filter for stable and stochastic seasonal components* above, the psi-weights are shown in the figure below.

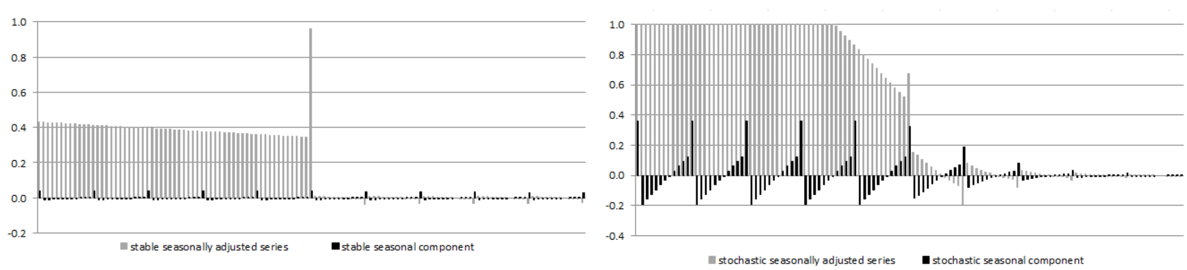


Figure 277: **WK filters and squared gain of the WK filter**

It can be shown that $\xi_{-1}, \ldots, \xi_{-j}$ are convergent and $\xi_j, \ldots, \xi_1, \xi_0$ are divergent. From Equation 0.56, the concurrent estimator is equal to:

$$\hat{x}_{it|t} = E_t x_{it} = E_t \hat{x}_{it} = \xi_i(B)^{-} a_t \tag{0.58}$$

so that the revision

$$r_{it} = \hat{x}_{it} - \hat{x}_{it|t} = \xi_i(F)^{+} a_{t+1} \tag{0.59}$$

is a zero-mean stationary MA process. As a result, historical and preliminary estimators are co-integrated. From Equation 0.48 the relative size of the full revision and the speed of convergence can be obtained.

Tests

In this chapter

This chapter describes all the tests available in JDemetra+, via Graphical User Interface and R packages. Here the underlying theoretical principles of each test are provided.

to be added: which test are done when Seats decomposition vs X-13-ARIMA.

Tests on residuals

Test	Purpose	GUI	R package
Ljung-Box	autocorrelation	✔	rjd3tookit
Box-Pierce	autocorrelation	✔	
Doornik-Hansen	normality	✔	rjd3tookit

Ljung-Box

The Ljung-Box Q-statistics are given by:

$$\text{LB}(k) = n \times (n+2) \times \sum_{k=1}^{K} \frac{\rho_{a,k}^2}{n-k}$$

where $\rho_{a,k}^2$ is the autocorrelation coefficient at lag k of the residuals $\hat{a}_t$, n is the number of terms in the differenced series, K, the maximum lag being considered, is set in JDemetra+ to 24 (monthly series) or 8 (quarterly series).

If the residuals are random (which is the case for residuals from a well specified model), they will be distributed as $\chi^2_{(K-m)}$, where m is the number of parameters in the model which has been fitted to the data. (edit: not the residuals, but $\hat{\rho}$)

Box-Pierce

The Box-Pierce Q-statistics are given by:

$$\text{BP}(k) = n\sum_{k=1}^{K}\rho_{a,k}^{2}$$

where:

- $\rho_{a,k}^{2}$ is the autocorrelation coefficient at lag k of the residuals $\hat{a}_t$.
- n is the number of terms in differenced series;
- K is the maximum lag being considered, set in JDemetra+ to 24 (monthly series) or 8 (quarterly series).

If the residuals are random (which is the case for residuals from a well specified model), they will be distributed as $\chi_{(K-m)}^{2}$ degrees of freedom, where m is the number of parameters in the model which has been fitted to the data.

Dornik-Hansen

The Doornik-Hansen test for multivariate normality (DOORNIK, J.A., and HANSEN, H. (2008)) is based on the skewness and kurtosis of multivariate data that is transformed to ensure independence.

The skewness and kurtosis are defined, respectively, as: $s = \frac{m_3}{\sqrt{m_2}^3}$ and $k = \frac{m_4}{m_2^2}$,

where:

- $m_i = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^i$;
- $\overline{x} = \frac{1}{n}\sum_{i=1}^{n}x_i$;
- n number of (non-missing) residuals.

The Doornik-Hansen test statistic derives from SHENTON, L.R., and BOWMAN, K.O. (1977) and uses transformed versions of skewness and kurtosis.

The transformation for the skewness s into z_1 is as in D'AGOSTINO, R.B. (1970):

$$\beta = \frac{3(n^2 + 27n - 70)(n+1)(n+3)}{(n-2)(n+5)(n+7)(n+9)}$$

$$\omega^2 = -1 + \sqrt{2(\beta - 1)}$$

$$\delta = \frac{1}{\sqrt{\log(\omega^2)}}$$

$$y = s\sqrt{\frac{(\omega^2 - 1)(n+1)(n+3)}{12(n-2)}}$$

$$z_1 = \delta log(y + \sqrt{y^2 - 1})$$

The kurtosis k is transformed from a gamma distribution to χ^2, which is then transformed into standard normal z_2 using the Wilson-Hilferty cubed root transformation:

$$\delta = (n-3)(n+1)(n^2 + 15n - 4)$$

$$a = \frac{(n-2)(n+5)(n+7)(n^2 + 27n - 70)}{6\delta}$$

$$c = \frac{(n-7)(n+5)(n+7)(n^2 + 2n - 5)}{6\delta}$$

$$l = \frac{(n+5)(n+7)(n^3 + 37n^2 + 11n - 313)}{12\delta}$$

$$\alpha = a + c \times s^2$$

$$\chi = 2l(k - 1 - s^2)$$

$$z_2 = \sqrt{9\alpha}(\frac{1}{9\alpha} - 1 + \sqrt[3]{\frac{\chi}{2\alpha}})$$

Finally, the Doornik-Hansen test statistic is defined as the sum of squared transformations of the skewness and kurtosis. Approximately, the test statistic follows a χ^2 distribution, i.e.:

$$DH = z_1^2 + z_2^2 \sim \chi^2(2)$$

Seasonality tests

table with all tests by purpose and accessibility

Test	Purpose	GUI	R package
QS test	Autocorrelation at seasonal lags	✔	
F-test with seasonal dummies	Stable seasonality	✔	rjd3toolkit
Canova-Hansen	Seasonal frequencies	✖	rjd3toolkit
Identification of spectral peaks	Seasonal frequencies	✔	
Friedman test	Stable seasonality	✔	rjd3toolkit
Two-way variance analysis	Moving seasonality	✔	

QS Test on autocorrelation at seasonal lags

The QS test is a variant of the Ljung-Box test computed on seasonal lags, where we only consider positive auto-correlations

More exactly,

$$QS = n(n+2)\sum_{i=1}^{k}\frac{[\text{max}(0,\hat{\gamma}_{i\cdot l})]^2}{n-i\cdot l}$$

where $k = 2$, so only the first and second seasonal lags are considered. Thus, the test would check the correlation between the actual observation and the observations lagged by one and two years. Note that $l = 12$ when dealing with monthly observations, so we consider the autocovariances $\hat{\gamma}_{12}$ and $\hat{\gamma}_{24}$ alone. In turn, $k = 4$ in the case of quarterly data.

Under H0, which states that the data are independently distributed, the statistics follows a $\chi(k)$ distribution. However, the elimination of negative correlations makes it a bad approximation. The p-values would be given by $P(\chi^2(k) > Q)$ for $k = 2$. As $P(\chi^2(2)) > 0.05 = 5.99146$ and $P(\chi^2(2)) > 0.01 = 9.21034$, $QS > 5.99146$ and $QS > 9.21034$ would suggest rejecting the null hypothesis at 95% and 99% significance levels, respectively.

Maravall (2012) proposes approximate the correct distribution (p-values) of the QS statistic using simulation techniques. Using 1000K replications of sample size 240, the correct critical values would be 3.83 and 7.09 with confidence levels of 95% and 99%, respectively (lower than the 5.99146 and 9.21034 shown above). For each of the simulated series, he obtains the distribution by assuming $QS = 0$ when $\hat{\gamma}_{12}$, so in practice this test will detect seasonality only when any of these conditions hold:

- Statistically significant positive autocorrelation at lag 12
- Non-negative sample autocorrelation at lag 12 and statistically significant positive autocorrelation at lag 24

F-test on seasonal dummies

The F-test on seasonal dummies checks for the presence of deterministic seasonality. The model used here uses seasonal dummies (mean effect and 11 seasonal dummies for monthly data, mean effect and 3 for quarterly data) to describe the (possibly transformed) time series behaviour. The test statistic checks if the seasonal dummies are jointly statistically not significant. When this hypothesis is rejected, it is assumed that the deterministic seasonality is present and the test results are displayed in green.

This test refers to Model-Based \$▯^{2}\$ and F-tests for Fixed Seasonal Effects proposed by LYTRAS, D.P., FELDPAUSCH, R.M., and BELL, W.R. (2007) that is based on the estimates of the regression dummy variables and the corresponding t-statistics of the Reg-ARIMA model, in which the ARIMA part of the model has a form (0,1,1)(0,0,0). The consequences of a misspecification of a model are discussed in LYTRAS, D.P., FELDPAUSCH, R.M., and BELL, W.R. (2007).

For a monthly time series the Reg-ARIMA model structure is as follows:

$$(1-B)(y_t - \beta_1 M_{1,t} - ... - \beta_{11} M_{11,t} - \gamma X_t) = \mu + (1-B)a_t \qquad (0.60)$$

where:

- $M_{j,t} = \begin{cases} 1 & \text{in month } j = 1, \dots, 11 \\ -1 & \text{in December} \\ 0 & \text{otherwise} \end{cases}$ -dummy variables;
- y_t – the original time series;
- B – backshift operator;
- X_t – other regression variables used in the model (e.g. outliers, calendar effects, user-defined regression variables, intervention variables);
- μ – mean effect;
- a_t – white-noise variable with mean zero and a constant variance.

In the case of a quarterly series the estimated model has this form:

$$(1-B)(y_t - \beta_1 M_{1,t} - \dots - \beta_3 M_{3,t} - \gamma X_t) = \mu + (1-B)a_t \tag{0.61}$$

where:

$$M_{j,t} = \begin{cases} 1 & \text{in quarter} j = 1, \dots, 3 \\ -1 & \text{in the fourth quarter} \\ 0 & \text{otherwise} \end{cases} \text{-dummy variables;}$$

One can use the individual t-statistics to assess whether seasonality for a given month is significant, or a chi-squared test statistic if the null hypothesis is that the parameters are collectively all zero. The chi-squared test statistic is $\hat{\chi}^2 = \hat{\beta}'[Var(\hat{\beta})\ )^{-1}]\hat{\beta}$ in this case compared to critical values from a χ^2(df)-distribution, with degrees of freedom $df = 11$ (monthly series) or $df = 3$ (quarterly series). Since the $Var(\hat{\beta})$ computed using the estimated variance of α_t may be very different from the actual variance in small samples, this test is corrected using the proposed F statistic:

$$F = \frac{\hat{\chi}^2}{s-1} \times \frac{n-d-k}{n-d}$$

where n is the sample size, d is the degree of differencing, s is time series frequency (12 for a monthly series, 4 for a quarterly series) and k is the total number of regressors in the Reg-ARIMA model (including the seasonal dummies $M_{j,t}$ and the intercept).

This statistic follows a $F_{s-1,n-d-k}$ distribution under the null hypothesis.

Friedman test for stable seasonality

The Friedman test is a non-parametric method for testing that samples are drawn from the same population or from populations with equal medians. The significance of the month (or quarter) effect is tested. The Friedman test requires no distributional assumptions. It uses the rankings of the observations. If the null hypothesis of no stable seasonality is rejected at the 0.10% significance level then the series is considered to be seasonal and the test's outcome is displayed in green.

The test statistic is constructed as follows. Consider first the matrix of data $\{x_{ij}\}_{n\times k}$ with n rows (the blocks, i.e. number of years in the sample), k columns (the treatments, i.e. either 12 months or 4 quarters, depending on the frequency of the data).

The data matrix needs to be replaced by a new matrix $\{r_{ij}\}_{n\times k}$, where the entry r_{ij} is the rank of x_{ij} within block i .

The test statistic is given by

$$Q = \frac{SS_t}{SS_e}$$

where $SS_t = n\sum_{j=1}^{k}(\overline{r}_{.j} - \overline{r})^2$ and $SS_e = \frac{1}{n(k-1)}\sum_{i=1}^{n}\sum_{j=1}^{k}(r_{ij} - \overline{r})^2$. It represents the variance of the average ranking across treatments j relative to the total.

Under the hypothesis of no seasonality, all months can be equally treated. For the sake of completeness:-$\overline{r} * .j$ is the average ranks of each treatment (month) j within each block (year)-The average rank is given by $\overline{r} = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij}$

For large n or k , i.e. $n > 15$ or $k > 4$, the probability distribution of Q can be approximated by that of a chi-squared distribution. Thus, the p-value is given by $P(\chi^2_{k-1} > Q)$.

Moving seasonality test

The evolutive seasonality test is based on a two-way analysis of variance model. The model uses the values from complete years only. Depending on the decomposition type for the Seasonal/Irregular component it uses Equation 0.60 (in the case of a multiplicative model) or Equation 0.61 (in the case of an additive model):

$$|\mathrm{SI_{ij}} - 1| = X_{\mathrm{ij}} = b_i + m_j + e_{\mathrm{ij}}$$

$$|\mathrm{SI_{ij}}| = X_{\mathrm{ij}} = b_i + m_j + e_{\mathrm{ij}}$$

where:

- m_j – monthly or quarterly effect for j-th period, $j = (1, \ldots, k)$, where $k = 12$ for a monthly series and $k = 4$ for a quarterly series;
- b_j – annual effect i, $(i = 1, \ldots, N)$ where N is the number of complete years;
- e_{ij} – residual effect.

The test is based on the following decomposition:

$$S^2 = S_A^2 + S_B^2 + S_R^2, \tag{0.62}$$

where:

- $S^2 = \sum_{j=1}^{k} \sum_{i=1}^{N} (\overline{X}_{\mathrm{ij}} - \overline{X}_{\bullet\bullet})^2$ –the total sum of squares;
- $S_A^2 = N \sum_{j=1}^{k} (\overline{X}_{\bullet j} - \overline{X}_{\bullet\bullet})^2$ – the inter-month (inter-quarter, respectively) sum of squares, which mainly measures the magnitude of the seasonality;
- $S_B^2 = k \sum_{i=1}^{N} (\overline{X}_{i\bullet} - \overline{X}_{\bullet\bullet})^2$ – the inter-year sum of squares, which mainly measures the year-to-year movement of seasonality;
- $S_R^2 = \sum_{i=1}^{N} \sum_{j=1}^{k} (\overline{X}_{\mathrm{ij}} - \overline{X}_{i\bullet} - \overline{X}_{\bullet j} - \overline{X}_{\bullet\bullet})^2$ – the residual sum of squares.

The null hypothesis H_0 is that $b_1 = b_2 = \ldots = b_N$ which means that there is no change in seasonality over the years. This hypothesis is verified by the following test statistic:

$$F_M = \frac{\frac{S_B^2}{(n-1)}}{\frac{S_R^2}{(n-1)(k-1)}}$$

which follows an F-distribution with $k-1$ and $n-k$ degrees of freedom.

Combined seasonality test

This test combines the Kruskal-Wallis test along with test for the presence of seasonality assuming stability (F_S), and evaluative seasonality test for detecting the presence of identifiable seasonality (F_M). Those three tests are calculated using the final unmodified SI component. The main purpose of the combined seasonality test is to check whether the seasonality of the series is identifiable. For example, the identification of the seasonal pattern is problematic if the process is dominated by highly moving seasonality (DAGUM, E.B. (1987)). The testing procedure is shown in the figure below.

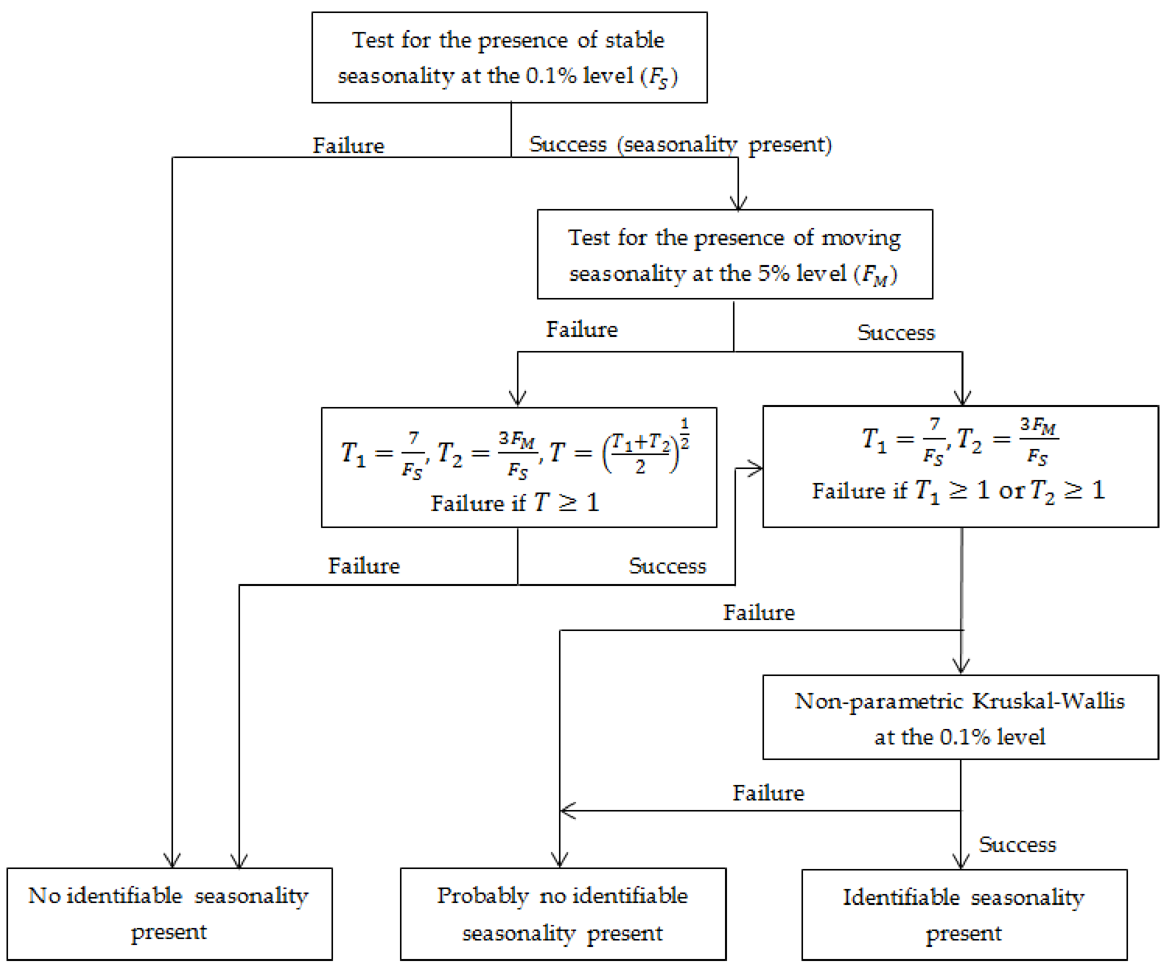


Figure 278: **Combined seasonality test, source: LADIRAY, D., QUENNEVILLE, B. (2001)**

Identification of spectral peaks

Tests related to identification of spectral peaks in a periodogram, autoregressive spectrum or Tuckey spectrum are detailed here in the chapter dedicated to spectral analysis.

STL: Local regression decomposition

Up coming content.

Benchmarking and Temporal disaggregation

Benchmarking Underlying Theory

Benchmarking[3] is a procedure widely used when for the same target variable the two or more sources of data with different frequency are available. Generally, the two sources of data rarely agree, as an aggregate of higher-frequency measurements is not necessarily equal to the less-aggregated measurement. Moreover, the sources of data may have different reliability. Usually it is thought that less frequent data are more trustworthy as they are based on larger samples and compiled more precisely. The more reliable measurement is considered as a benchmark.

Benchmarking in Seasonal adjustment

In seasonal adjustment methods benchmarking is the procedure that ensures the consistency over the year between adjusted and non-seasonally adjusted data. It should be noted that the ESS Guidelines on Seasonal Adjustment (2024), do not recommend benchmarking as it introduces a bias in the seasonally adjusted data. The U.S. Census Bureau also points out that "*forcing the seasonal adjustment totals to be the same as the original series annual totals can degrade the quality of the seasonal adjustment, especially when the seasonal pattern is undergoing change. It is not natural if trading day adjustment is performed because the aggregate trading day effect over a year is variable and moderately different from zero*"[^2]. Nevertheless, some users may need that the annual totals of the seasonally adjusted series match the annual totals of the original, non-seasonally adjusted series[^3].

According to the ESS Guidelines on Seasonal Adjustment (2015), the only benefit of this approach is that there is consistency over the year between adjusted and the non-seasonally adjusted data; this can be of particular interest when

[3] Description of the idea of benchmarking is based on DAGUM, B.E., and CHOLETTE, P.A. (1994) and QUENNEVILLE, B. et all (2003). Detailed information can be found in: DAGUM, B.E., and CHOLETTE, P.A. (2006)

low-frequency (e.g. annual) benchmarking figures officially exist (e.g. National Accounts, Balance of Payments, External Trade, etc.) and where users' needs for time consistency are stronger..

The benchmarking procedure in JDemetra+ is available for a single seasonally adjusted series and for an indirect seasonal adjustment of an aggregated series. In the first case, univariate benchmarking ensures consistency between the raw and seasonally adjusted series. In the second case, the multivariate benchmarking aims for consistency between the seasonally adjusted aggregate and its seasonally adjusted components.

Given a set of initial time series

$$\left\{z_{i,t}\right\}_{i\in I}$$

, the aim of the benchmarking procedure is to find the corresponding

$$\left\{x_{i,t}\right\}_{i\in I}$$

that respect temporal aggregation constraints, represented by $X_{i,T} = \sum_{t\in T} x_{i,t}$ and contemporaneous constraints given by

$$q_{k,t} = \sum_{j\in J_k} w_{\mathrm{kj}} x_{j,t}$$

or, in matrix form:

$$q_{k,t} = w_k x_t$$

.

The underlying benchmarking method implemented in JDemetra+ is an extension of Cholette's[4] method, which generalises, amongst others, the additive and the multiplicative Denton procedure as well as simple proportional benchmarking.

The JDemetra+ solution uses the following routines that are described in DURBIN, J., and KOOPMAN, S.J. (2001):

- The multivariate model is handled through its univariate transformation,
- The smoothed states are computed by means of the disturbance smoother.

[4] CHOLETTE, P.A. (1979).

The performance of the resulting algorithm is highly dependent on the number of variables involved in the model ($\propto\ n^3$). The other components of the problem (number of constraints, frequency of the series, and length of the series) are much less important ($\propto\ n$).

From a theoretical point of view, it should be noted that this approach may handle any set of linear restrictions (equalities), endogenous (between variables) or exogenous (related to external values), provided that they don't contain incompatible equations. The restrictions can also be relaxed for any period by considering their "observation" as missing. However, in practice, it appears that several kinds of contemporaneous constraints yield unstable results. This is more especially true for constraints that contain differences (which is the case for non-binding constraints). The use of a special square root initializer improves in a significant way the stability of the algorithm.

Temporal disaggregation

Temporal disaggregation is a process by means of which a high frequency time series is obtained from its low frequency observations and, possibly, some additional information, such as a related high frequency time series.

By low and high frequency we may refer, for example, to a time series observed yearly or quarterly (in low frequency) that we try to estimate for each month (in high frequency), or to a time series observed yearly that we try to estimate for each quarter.

There are several types of temporal disaggregation methods. We will classify them according to two criteria, their deterministic or stochastic nature and whether they use any related time series or not.

In temporal disaggregation, we use s as low frequency time index variable and t as high frequency time index variable. So, y_s is the observed low frequency time series of interest, y_t is the desired, but not observed, high frequency time series of interest, while z_s and z_t are the corresponding auxiliary time series, where, usually, z_t is observed and z_s is computed from z_t. The objective is to compute the estimates $\hat{y}_t$.

In benchmarking the notation is similar, but now y_t is observed. The purpose is to calibrate it using z_s or z_t (whichever is available). The calibrated values are the $\hat{y}_t$.

Deterministic Methods

We now briefly describe some of the deterministic methods used for temporal disaggregation and benchmarking.

Pro-rata

For temporal disaggregation, if we have y_s and z_t, we first compute z_s and then $\hat{y}_t = y_s \frac{z_t}{z_s}$ (we pro-rate y proportionally to z).

For benchmarking, if we have y_t and z_s, we first compute y_s and then $\hat{y}_t = z_s \frac{y_t}{y_s}$ (we pro-rate z_s with the ratios y_t/y_s).

The advantage of this method is that it is simple to use, but there are some other methods which have more desirable properties.

Denton

The Denton method[5] was designed to preserve the movement of the indicator in the benchmarked or disaggregated series.

For benchmarking assume that we observe $Y = (y_1, \cdots, y_T)^T$ and that we have a set of $r < T$ linear constraints on the benchmarked values $\hat{Y} = (\hat{y}_1, \cdots, \hat{y}_T)^T$ of the form

$$C\hat{Y} = d, \quad \text{that is } \begin{pmatrix} C_{11} & \cdots & C_{1T} \\ \cdots & \cdots & \cdots \\ C_{r1} & \cdots & C_{rT} \end{pmatrix} \begin{pmatrix} \hat{y}_1 \\ \cdots \\ \hat{y}_T \end{pmatrix} = \begin{pmatrix} d_1 \\ \cdots \\ d_r, \end{pmatrix}$$

For example, the y_i values could be monthly values, the $c_{i,j}$ could be all zeros and ones (twelve consecutive ones in each row) and the d_j could be accurate annual totals obtained from an external source of information. So, the restrictions would mean that we know more exact annual totals than the annual totals obtained by summing the y_i, and we require that the annual sums of the benchmarked $\hat{y}_i$ match those d_j.

[5] Denton(1971). Adjustment of Monthly or Quarterly Series to Annual Totals: An Approach Based on Quadratic Minimization. Journal of the American Statistical Association, 66(333):99-102, 1971.

There are several variations of the Denton method. The additive first differences Denton method tries, after taking regular differences once, to preserve the movement of the y_t in the benchmarked values $\hat{y}_t$. Exactly it minimizes

$$\min_{\hat{y}_t} \sum_{j=2}^{T} [(\hat{y}_t - y_t) - (\hat{y}_{t-1} - y_{t-1})]^2, \text{ subject to } C\hat{Y} = d, \tag{0.63}$$

where $(\hat{y}_t - y_t) - (\hat{y}_{t-1} - y_{t-1}) = \hat{z}_t - z_t$ and $z_t = y_t - y_{t-1}$ are the first regular differences of the y_t.

The proportional first differences Denton method is similar, but it assumes that the short term fluctuations, such as seasonal and irregular, have a multiplicative effect, instead of additive. It minimizes:

$$\min_{\hat{y}_t} \sum_{j=2}^{T} \left[\frac{\hat{y}_t}{y_t} - \frac{\hat{y}_{t-1}}{y_{t-1}}\right]^2, \text{ subject to } C\hat{Y} = d, \tag{0.64}$$

The additive and proportional second differences Denton methods are also frequently used and are similar to the first differences ones, but taking two regular differences instead of one.

There exist also some multivariate Denton methods. In them, several time series are benchmarked or disaggregated, each one with its own restrictions but, additionally, there are also some new restrictions that involve simultaneously two or more of the time series at some fixed time points. The optimization has a single objective function in which all the time series are included, and a different weight can be assigned to each series.

Stochastic methods

These methods assume some kind of statistical model involving the time series and the indicator.

Most methods in this category can be considered as particular cases of the method proposed by Stram and Wei[6] [7] [8]. There is a basic assumption made when we

[6]Stram and Wei (1986). Temporal Aggregation in the ARIMA Process. Journal of Time Series Analysis, 7(4):279-292, 1986.

[7]Stram and Wei (1986). A Methodological Note on the Disaggregation of Time Series Totals. Journal of Time Series Analysis, 7(4):293-302, 1986.

[8]Wei and Stram (1990). Disaggregation of Time Series Models. Journal of the Royal Statistical Society, Ser. B, 52(3):453-467, 1990.

use any method in this category to temporally disaggregate a time series. That assumption is that there are no hidden periodicities, and it means that if the (often unknown) high frequency model is $ARMA(p, q)$, namely $\phi(B)y_t = \theta(B)\varepsilon_t$, and if $r_1, \cdots, r_p$ are the inverses of the roots of the $\phi(B)$ polynomial, then, if for any i, j, $r_i^m = r_j^m$ this implies that $r_i = r_j$, where m is the disaggregation period (for example, m is 4 for yearly to quarterly, 12 for yearly to monthly and 3 for quarterly to monthly). Without this assumption important problems may arise, that are related to what in system theory is called lack of observability, see Gómez and Aparicio-Pérez (2009)[9] for an in-depth discussion of this subject and how to proceed when there are hidden periodicities. All the disaggregation methods used in JDemetra+ use models that assume that there are no hidden periodicities.

Chow-Lin, Litterman and Fernandez

These methods can be all expressed with the same equation, but with different models for the error term:

$$y_t = z_t\beta + \alpha_t,$$

$$\alpha_t = \phi\alpha_{t-1} + \varepsilon_t, \text{ with } |\phi| < 1 \text{ (Chow-Lin)},$$
$$\alpha_t - \alpha_{t-1} = \phi(\alpha_{t-1} - \alpha_{t-2}) + \varepsilon_t, \text{ with } |\phi| < 1 \text{ (Litterman)},$$
$$\alpha_t - \alpha_{t-1} = \varepsilon_t. \text{ (Fernandez)}.$$

It is assumed that z_t is observed in high frequency while y_t is observed only in low frequency.

As it can be seen, these methods assume a linear regresion model between y_t and z_t, with residuals that are $ARIMA(1, 0, 0)$ (Chow-Lin), $ARIMA(1, 1, 0)$ (Litterman) or $ARIMA(0, 1, 0)$ (Fernández), so Fernandez is a special case of Litterman with $\phi = 0$ and is also a limit case of Chow-Lin when $\phi \to 1$.

When using modern software like JDemetra+, these models are estimated using state-space techniques, though some older programs use regression techniques after writing the model in matrix notation, obtaining then the low frequency model, estimating it and projecting the solution into high frequency using the conditional expectation to get the $\hat{y}_t$. This latter process is also sometimes used to explain the method when no knowledge of state-space techniques is assumed.

[9] Gómez and Aparicio-Pérez (2009). A new State-space Methodology to Disaggregate Multivariate Time Series. Journal of Time Series Analysis, 30(1):97-124, 2009.

Autoregressive distributed lags models (ADL).

These models are particular cases of $ARMAX$ models, that is, autoregressive, moving average with exogenous input models. An $ARMAX(p, q, r)$ model is of the form

$$\phi(B)y_t = \omega(B)z_t + \theta(B)\epsilon_t,$$

where the $\phi(B)$, $\theta(B)$ and $\omega(B)$ polynomials have respective degrees p, q and r, ϵ_t is a white noise process with variance σ_ϵ^2 and z_t is an exogenous input series.

As a particular case, an autoregressive distributed lags $ADL(p, r)$ model is defined as a $ARMAX(p, 1, r)$ model, that is

$$\phi(B)y_t = \omega(B)z_t + \epsilon_t,$$

When, additionally, $p = 1$ is imposed, the resulting $ADL(1, r)$ model is called a distributed lags model.

The idea behind using these models for temporal disaggregation is that the inclusion of lagged dependent variables $y_{t-1}, y_{t-2}, \cdots$ may significantly reduce the autocorrelation of the residuals in models such as Chow-Lin and others.

Santos Silva and Cardoso.

In Santos, Silva and Cardoso (2001)[10] the $ADL(1, 0)$ model

$$y_t = \phi y_{t-1} + z_t\beta + \epsilon_t$$

is proposed.

Some variants of this model are also possible, for example

$$y_t = \alpha + \phi y_{t-1} + z_t\beta + \epsilon_t$$

where a constant term is included. It is also possible to work in logarithms, for example with the model $\log y_t = \alpha + \phi \log y_{t-1} + \log z_t\beta + \epsilon_t$.

[10] Santos, Silva and Cardoso (2001). The Chow-Lin Method Using Dynamic Models. Economic Modelling, 18:269-280, 2001.

Proietti

In Proietti (2006)[11] the $ADL(1,1)$ model with linear deterministic trend

$$y_t = \phi y_{t-1} + m + gt + z_t\beta + z_{t-1}\gamma + \epsilon_t$$

is considered.

As a particular case, if $m = g = 0$ and $\gamma = -\beta\phi$, we get $(1-\phi B)y_t = \beta(1-\phi B)z_t + \varepsilon_t$, that is Chow-Lin (multiplying by $(1-\phi B)$ in Chow-Lin we get exactly this model).

[11] Proietti (2006). Temporal Disaggregation by State Space Methods: Dynamic Regression Methods Revisited. Econometric Journal, 9:357-372, 2006.

Revision Analysis

In this Chapter

This chapter gives some methodological background on revision analysis. Available JDemetra+ algorithms are described in this chapter

Motivation

why ?

how to analyse the behaviour of revisions of frequently published (and revised) macro-economic indicators.

details on methods

Bell, William R, and Steven C Hillmer. 1984. "Issues Involved With the Seasonal Adjustment of Economic Time Series." *Journal of Business & Economic Statistics* 2 (4): 291–320.

Burman, J P. 1980. "Seasonal Adjustment by Signal Extraction." *Journal of the Royal Statistical Society: Series A (General)* 143 (3): 321–37.

Caporello, Gianluca, and Agustin Maravall. 2004. "Program TSW: Revised Reference Manual." {T}echnical {R}eport, {R}esearch {D}epartment. Banco de España. https://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosOcasionales/04/Fic/do0408e.pdf.

Cleveland, William P, and George C Tiao. 1976. "Decomposition of Seasonal Time Series: A Model for the Census x-11 Program." *Journal of the American Statistical Association* 71 (355): 581–87.

Hillmer, Steven C, and George C Tiao. 1982. "An ARIMA Model-Based Approach to Seasonal Adjustment." *Journal of the American Statistical Association* 77 (377): 63–70.

Maravall, Agustin, and David A Pierce. 1987. "A Prototypical Seasonal Adjustment Model." *Journal of Time Series Analysis* 8 (2): 177–93.