Time Series Lab Manual

Rutger Lit

Time
Series
Lab ®

Preface

Time Series Lab is a platform that facilitates the analysis, modelling, and forecasting of time series in a highly interactive way with much graphical support. The users can base their analyses on a large selection of time series approaches, including Box-Jenkins models, exponential smoothing methods, score-driven location models and basic structural time series models. Therefore, users only concentrate on selecting models that fit the data best.

Furthermore, *Time Series Lab* allows the users to select a wide range of dynamic components that are related to, for example, trend, seasonal and stationary cycle processes, in order to build their own time series models.

Time Series Lab fully relies on advanced state space methods such as the Kalman filter and related smoothing algorithms. These methods have proven to be very effective in providing powerful solutions for many time series applications. For example, *Time Series Lab* can handle missing values in all model settings.

More information can be found on `https://timeserieslab.com`.

Feedback: we appreciate your feedback on the program. Please let us know by sending an email to `feedback@timeserieslab.com`.

Bugs: encountered a bug? Please let us know by sending an email to `bugs@timeserieslab.com`. Please describe the exact steps you took to reach to the point where you found the bug.

Contact: for questions about *Time Series Lab* or inquiries about commercial versions of the program, please send an email to `info@timeserieslab.com`.

Contents

Chapter 1

Getting started

If you're interested in time series analysis and forecasting, this is the right place to be. The *Time Series Lab* (TSL) software platform makes time series analysis available to anyone with a basic knowledge of statistics. Future versions will remove the need for a basic knowledge altogether by providing fully automated forecasting systems. The platform is designed and developed in a way such that results can be obtained quickly and verified easily. At the same time, many advanced time series and forecasting operations are available for the experts.

There are a few key things to know about TSL before you start. First, TSL operates using a number of different steps. Each step covers a different part of the modelling process. Before you can access certain steps, information must be provided to the program. This can be, for example, the loading of data or the selection of the dependent variable. The program warns you if information is missing and guides you to the part of the program where the information is missing. We will discuss each step of the modelling process and use example data sets to illustrate the program's functionality.

Throughout this **manual**, alert buttons like the one on the left will provide you with important information about TSL.

Furthermore, throughout the **software**, info buttons, like this blue one on the left, are positioned where additional information might be helpful. The info button displays its text by hovering the mouse over it.

TSL uses its algorithms to extract *time-varying components* from the data. In its simplest form, this is just a *Random walk* but it can be much more elaborate with combinations of *Autoregressive* components, multiple *Seasonal* components, and *Explanatory variables*. You will see examples of *time-varying* components throughout this manual. The workings and features of TSL are discussed in Chapter 3–11. If you are more interested in *Case studies* and see TSL in action, go to Chapter 12.

We refer to Appendix A – D for background and details of time series methodology.

Appendix A illustrates the strength of dynamic models and why dynamic models are often better in forecasting than static (constant over time) models. Several of the algorithms of *Time Series Lab* are based on State Space methodology, see Harvey (1990) and Durbin and Koopman (2012) and the score-driven methodology, see Creal et al. (2013) and Harvey (2013). Appendix B discusses the mathematical framework of state space models and Appendix C discusses the mathematical framework of score-driven models. Knowledge of the methodology is not required to use TSL but is provided for the interested reader. Appendix D shows that well-known models like ARMA and GARCH models are submodels of score-driven models.

1.1 Installing and starting TSL

Time Series Lab comes in two versions, the *Home* edition and the *Enterprise* edition. The Home edition can be downloaded for free from `https://timeserieslab.com`. It has almost every feature and algorithm that the Enterprise edition has. The main difference is that the Home edition is restricted to univariate time series analysis. The commercial Enterprise edition supports companies and institutions to process large volumes of time series in a sequential manner. It further allows TSL to be modified and extended on an individual basis. Also, additional modules can be added to TSL, creating a hands-on platform that is finely tuned towards the particular needs of the user(s). More information can be obtained by sending an email message to `info@timeserieslab.com`.

Windows 10 64bit and Windows 11 64bit are the supported platforms. TSL can be started by double-clicking the icon on the desktop or by clicking the Windows Start button and selecting TSL from the list of installed programs. TSL is generally light-weight under normal circumstances. It needs less than 600 MB hard-disk space and 600 MB RAM.

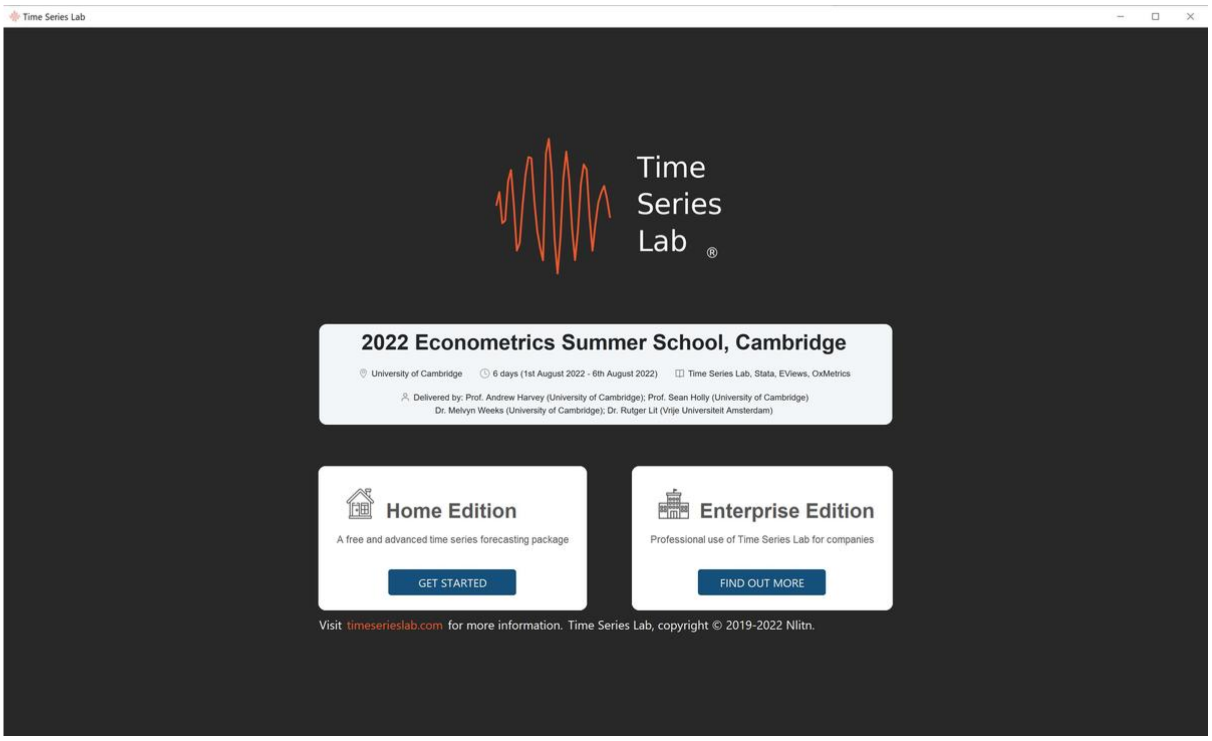
1.2 Frontpage

After starting the Home edition of TSL, you see the screen as shown in Figure 1.1. It shows TSL's logo at the top of the page. The information banner in the middle of the screen displays relevant updates on *Time Series Lab*. Examples are, information on upcoming new versions of TSL or organized courses / summer schools / winter schools which involves TSL in any way. Information is refreshed every eight seconds and currently does not hold any sponsored content.

The *Get Started* button leads you to the *Database* page where you can load your data set. The *Find out more* buttons opens the web browser and shows information of the TSL Enterprise edition.

Within TSL, you can always return to the *Front page* by clicking *File* ▶ *Front page* in the menu bar at the top of the page.

Figure 1.1
Front page of TSL Home edition



2022 Econometrics Summer School, Cambridge

University of Cambridge   6 days (1st August 2022 - 6th August 2022)   Time Series Lab, Stata, EViews, OxMetrics

Delivered by: Prof. Andrew Harvey (University of Cambridge); Prof. Sean Holly (University of Cambridge)
Dr. Melvyn Weeks (University of Cambridge); Dr. Rutger Lit (Vrije Universiteit Amsterdam)

A free and advanced time series forecasting package

Professional use of Time Series Lab for companies

Visit timeserieslab.com for more information. Time Series Lab, copyright © 2019-2022 Nlitn.

1.3 *Time Series Lab* modules

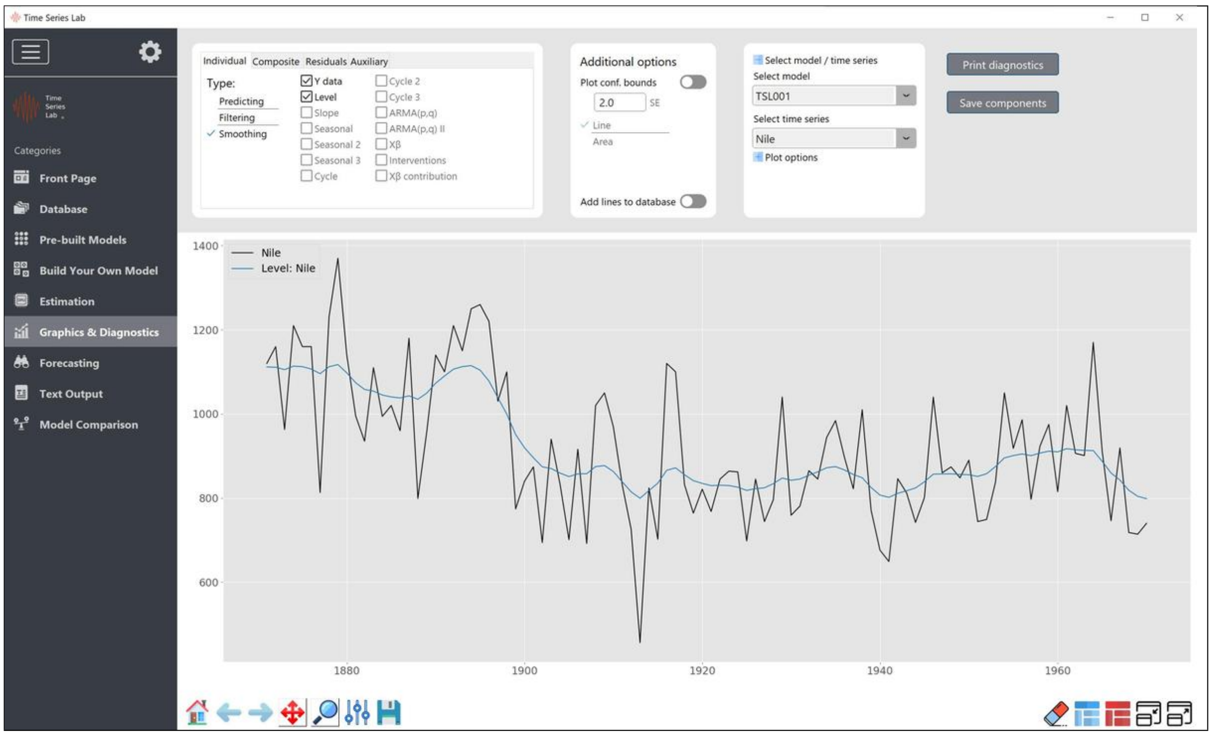
The modules of TSL can be accessed by clicking the buttons located at the left of the screen, see Figure 1.2. The modules are:

- Connect to Database
- Select & Prepare data
- Pre-built models
- Build Your Own Model
- Estimation
- Graphics & Diagnostics
- Forecasting
- Text Output
- Model Comparison
- Batch Module

All modules are described in detail in the following chapters. The *Batch module* allows you to program and schedule TSL yourself without going through all the menus and pages manually.

Figure 1.2

Modules of TSL are accessed using the buttons left of the screen



Type: ✓ Smoothing

☑ Y data

☑ Level

Plot conf. bounds 2.0 SE

✓ Line

Select model / time series

Select model: TSL001

Select time series: Nile

Chapter 2

Connect to database

There are currently two ways of getting data in TSL. The first method is with a database connection to an API server and this method is discussed in this chapter. The second method is by manually loading a data file. This option is discussed in Chapter 3. To connect to a database via an API server, navigate to the *Connect to Database* page.

2.1 Select database connection

Currently the only available database connection is to the Federal Reserve Economic Data — FRED — St. Louis Fed database. We will add more connections in the future. Let us know if you have suggestions! To connect to a database, perform the following steps:

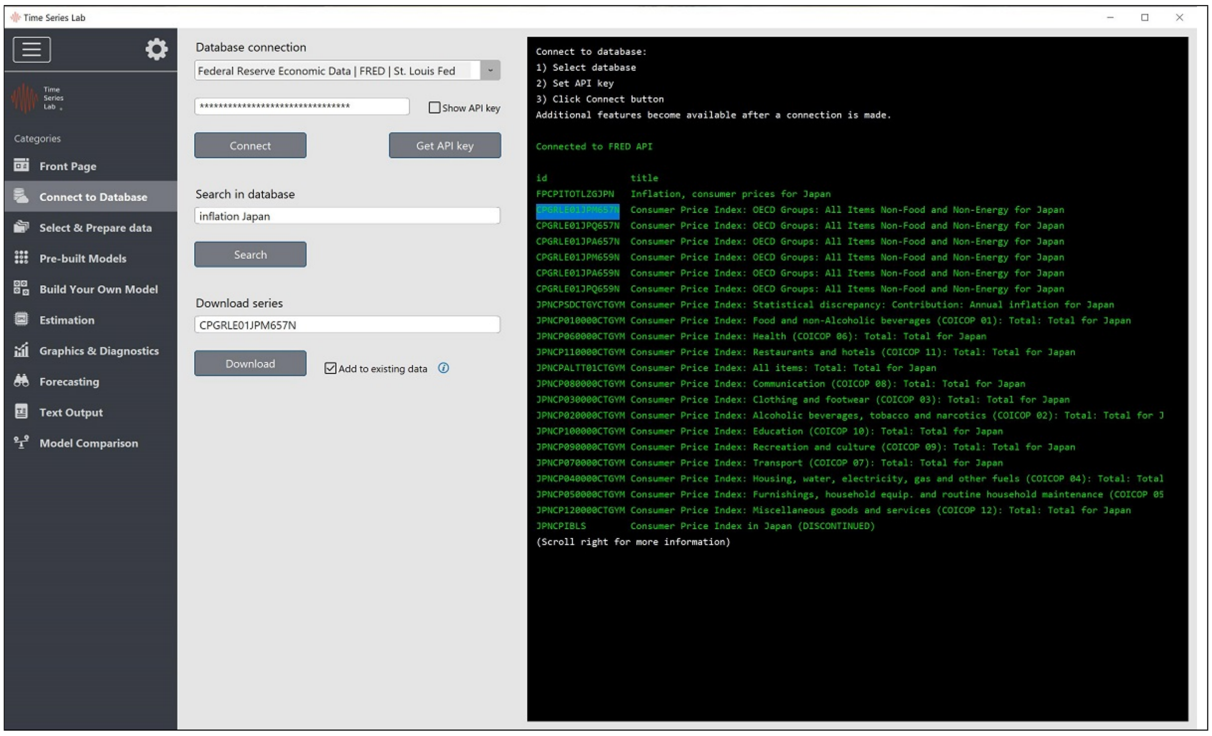
1. Select a database from the drop-down menu
2. Set the API key
3. Click the connect button

If you do not have an API key to the selected database, you can request one by clicking the *Get API key* button which will lead you to a sign-up page to request an API key. After a successful connection is made, additional features become available.

2.1.1 Search in database

The *search in database* option allows you to search for keywords in the database you are connected to. For example, in Figure 2.1 we searched for *inflation Japan* and the search results from the API server are displayed in the right side of the screen. You might need to scroll down and/or to the right to see all information that was received from the API server.

Figure 2.1
Connect to database page



Database connection
Federal Reserve Economic Data | FRED | St. Louis Fed
****
☐ Show API key

Search in database
inflation Japan

Download series
CPGRLE01JPM657N
☑ Add to existing data

```
Connect to database:
1) Select database
2) Set API key
3) Click Connect button
Additional features become available after a connection is made.

Connected to FRED API

id              title
FPCPITOTLZGJPN  Inflation, consumer prices for Japan
CPGRLE01JPM657N Consumer Price Index: OECD Groups: All Items Non-Food and Non-Energy for Japan
CPGRLE01JPQ657N Consumer Price Index: OECD Groups: All Items Non-Food and Non-Energy for Japan
CPGRLE01JPA657N Consumer Price Index: OECD Groups: All Items Non-Food and Non-Energy for Japan
CPGRLE01JPM659N Consumer Price Index: OECD Groups: All Items Non-Food and Non-Energy for Japan
CPGRLE01JPA659N Consumer Price Index: OECD Groups: All Items Non-Food and Non-Energy for Japan
CPGRLE01JPQ659N Consumer Price Index: OECD Groups: All Items Non-Food and Non-Energy for Japan
JPNCPSDCTGYCTGYM Consumer Price Index: Statistical discrepancy: Contribution: Annual inflation for Japan
JPNCP010000CTGYM Consumer Price Index: Food and non-Alcoholic beverages (COICOP 01): Total: Total for Japan
JPNCP060000CTGYM Consumer Price Index: Health (COICOP 06): Total: Total for Japan
JPNCP110000CTGYM Consumer Price Index: Restaurants and hotels (COICOP 11): Total: Total for Japan
JPNCPALTT01CTGYM Consumer Price Index: All items: Total: Total for Japan
JPNCP080000CTGYM Consumer Price Index: Communication (COICOP 08): Total: Total for Japan
JPNCP030000CTGYM Consumer Price Index: Clothing and footwear (COICOP 03): Total: Total for Japan
JPNCP020000CTGYM Consumer Price Index: Alcoholic beverages, tobacco and narcotics (COICOP 02): Total: Total for J
JPNCP100000CTGYM Consumer Price Index: Education (COICOP 10): Total: Total for Japan
JPNCP090000CTGYM Consumer Price Index: Recreation and culture (COICOP 09): Total: Total for Japan
JPNCP070000CTGYM Consumer Price Index: Transport (COICOP 07): Total: Total for Japan
JPNCP040000CTGYM Consumer Price Index: Housing, water, electricity, gas and other fuels (COICOP 04): Total: Total
JPNCP050000CTGYM Consumer Price Index: Furnishings, household equip. and routine household maintenance (COICOP 05
JPNCP120000CTGYM Consumer Price Index: Miscellaneous goods and services (COICOP 12): Total: Total for Japan
JPNCPIBLS       Consumer Price Index in Japan (DISCONTINUED)
(Scroll right for more information)
```

2.1.2 Download series

A time series can be downloaded by providing the series ID to the *Download series* entry field. The simplest way of setting the ID is by double clicking the ID in the text field (right side screen). Alternatively, you can manually type the ID or highlight the ID in the text field and click the right-mouse key followed by clicking *Select for download*.

Tick the checkbox *Add to existing data* if you want TSL to add the newly downloaded data to the existing database of previously downloaded series. Note that downloaded data can be added to the existing database only if the frequency matches (quarterly, monthly, etc.). If not, the current database is overwritten with the most recent downloaded data.

Press the *Download* button to download the series and if this is succesfull, the time series is placed in the TSL database and you are brought to the *Select & Prepare data* page for further processing.

Chapter 3

Select & prepare data

The first step in any time series analysis is the inspection and preparation of data. In TSL, clicking the button as shown on the left brings you to the data inspection and preparation page, see also Figure 3.1. You are also directed to this page after clicking the *Get Started* button on the Front page. The Nile data set[1] that comes bundled with the installation file of TSL is used as illustration.

3.1 Database

3.1.1 Load database

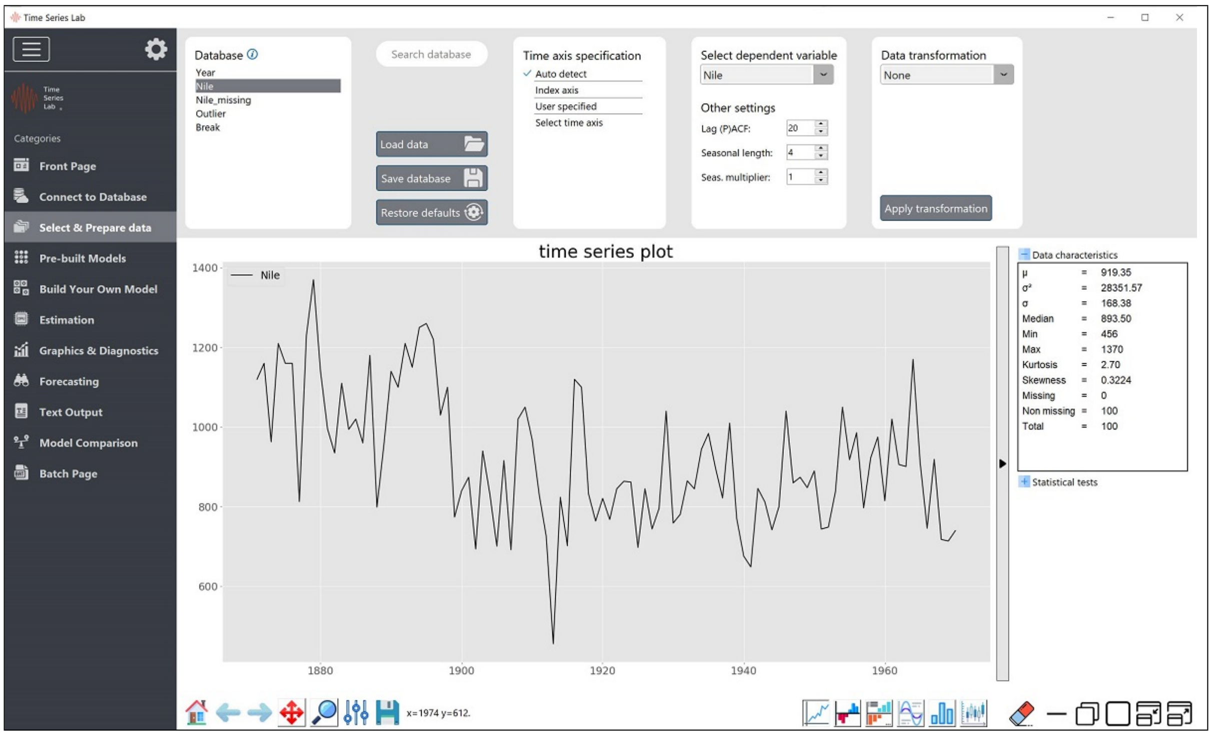
The data set is loaded and selected from the file system by pressing the *Load database* button or by clicking *File* ▶ *Load data*.

Important: The data set should be in column format with headers. The format of the data should be *.xls(x), or *.csv, *.txt with commas as field separation. The program (purposely) does not sort the data which means that the data should be in the correct time series order before loading it into the program.

After loading the data, the headers of the data columns appear in the *Database* section at the top left of the page. TSL automatically plots the second column of the database after loading. Plot a different variable by clicking on another header name. Ctrl-click or Shift-click to plot multiple variables in one graph. As shown in 3.1, the Nile data is currently highlighted and plotted.

[1]The Nile data set consists of annual averages of series of daily readings of the flow volume for the river Nile, measured at the city of Aswan, the annual time series is from 1871 to 1970.

Figure 3.1
Data inspection and preparation page



Database: Year, Nile, Nile_missing, Outlier, Break

Time axis specification: ✓ Auto detect, Index axis, User specified, Select time axis

Select dependent variable: Nile

Data transformation: None

Other settings

Lag (P)ACF:	20
Seasonal length:	4
Seas. multiplier:	1

Data characteristics

μ	=	919.35
σ^2	=	28351.57
σ	=	168.38
Median	=	893.50
Min	=	456
Max	=	1370
Kurtosis	=	2.70
Skewness	=	0.3224
Missing	=	0
Non missing	=	100
Total	=	100

x=1974 y=612.

3.1.2 Save database

The loaded data set can also be saved to the file system. This will not be useful right after loading but extracted signals from the modelling process are added to the Database at a later stage and can therefore be easily saved for further processing. Additionally, transformed variables, see Section 3.1.5, appear in the Database section as well.

3.1.3 Time axis specification

For time series analysis, time is obviously an important factor. As mentioned in Section 3.1.1, the loaded database should already be in the correct time series order before loading it in TSL. A time series axis can be specified as follows. First, TSL tries to auto detect the time axis specification (e.g. annual data, daily data) from the *first column* of the data set. In the case of the Nile data illustration in Figure 3.1, it finds an annual time axis specification. If the auto detection fails, the program selects the *Index axis* option which is just a number for each observation, $1, 2, 3, \ldots$.

You can specify the time axis manually as well via the *User specified* option or the *Select time axis* option. The *User specified* option, opens a new window in which the user can specify the date and time stamp of the first observation and the interval and frequency of the time series. The newly specified time axis shows up in the plot after pressing confirm (and exit). The *Select time axis* option allows the user to select the time axis from the loaded database. If the time axis format is not automatically found by the program the user can specify this via the *Format* input field. The text behind the info button ⓘ tells us:

```
Specify date format codes according to the 1989 C-standard convention, for example:

2020-01-27: %Y-%m-%d
2020(12): %Y(%m)
2020/01/27 09:51:43: %Y/%m/%d %H:%M:%S

Specify 'Auto' for auto detection of the date format.
```

Note that a time axis is not strictly necessary for the program to run and an *Index axis* will always do.

3.1.4 Select dependent variable

The so-called *y-variable* of the time series equation is the time series variable of interest, i.e. the time series variable you want to model, analyze, and forecast. You can specify the dependent variable by selecting it from the drop-down menu located under the *Select dependent variable* section of the Database page. Alternatively, the dependent variable is automatically selected if a variable is selected to be plotted by clicking on it. In our example, the highlighted variable Nile also appears in the *Select dependent variable* drop down menu. The dependent variable needs to be specified because without it, the software cannot estimate a model.

3.1.5 Data transformation

If needed, you can transform the data before modelling. For example if the time series consists of values of the Dow Jones Index, a series of percentage returns can be obtained by selecting *percentage change* from the *Data transformation* drop-down menu followed by clicking the *Apply transformation* button. Note that the (original) variable before transformation should be highlighted before applying the transformation to tell the program which variable to transform. An example is given in Figure 3.2 where the Nile data is transformed by taking logs. The newly transformed log variable (Nile_log) is added to the variables in the *Database* section and is automatically highlighted and plotted after the transformation. Transformations can be combined by applying transformations to already transformed variables. Newly created transformed variables can be removed from the database by *right mouse clicking* the variable and selecting the *Delete from database* option from the popup menu.
Depending on the selection, spin boxes under the *Apply transformation* button become visible to provide additional input.

Lag operator:
Lagged variables can be added to the model as well. Often these are explanatory variables, e.g. X_{t-1}. Lagging a time series means shifting it in time. The number of periods shifted can be controlled by the *Add lag* spin box which becomes visible after selecting the Lag operator from the menu. Note the text behind the information button ⓘ that says:

> Please note that values > 0 are lags and values < 0 are leads

Lag all operator:
Same as *Lag operator* but all lags in between are added to the database as well.

Differencing:
A non-stationary time series[2] can be made stationary by differencing. For example, if y_t denotes the value of the time series at period t, then the first difference of y_t at period t is equal to $y_t - y_{t-1}$, that is, we subtract the observation at time $t - 1$ from the observation at time t. TSL accommodates for this procedure since differencing is common practice among time series researchers. However, the methodology of TSL allows the user to explicitly model non-stationary time series and differencing is not strictly necessary. Note the text behind the information button ⓘ that tells us:

> Time Series Lab allows the user to explicitly model non-stationary components like trend and seasonal. However, users might prefer to make the time series stationary by taking first / seasonal differences before modelling.
>
> Please note that missing values are added to the beginning of the sample to keep the time series length equal to the original time series length before the difference operation.

Scaling:
Estimating a time series that consist of several very small or large values could potentially lead to numerical instabilities. This can be resolved by scaling the time series to more manageable numbers. For example, if sales data is in euros and numbers are high, the time series could be scaled down to model sales in millions, for example. Alternatively sales in Logs can be modelled.

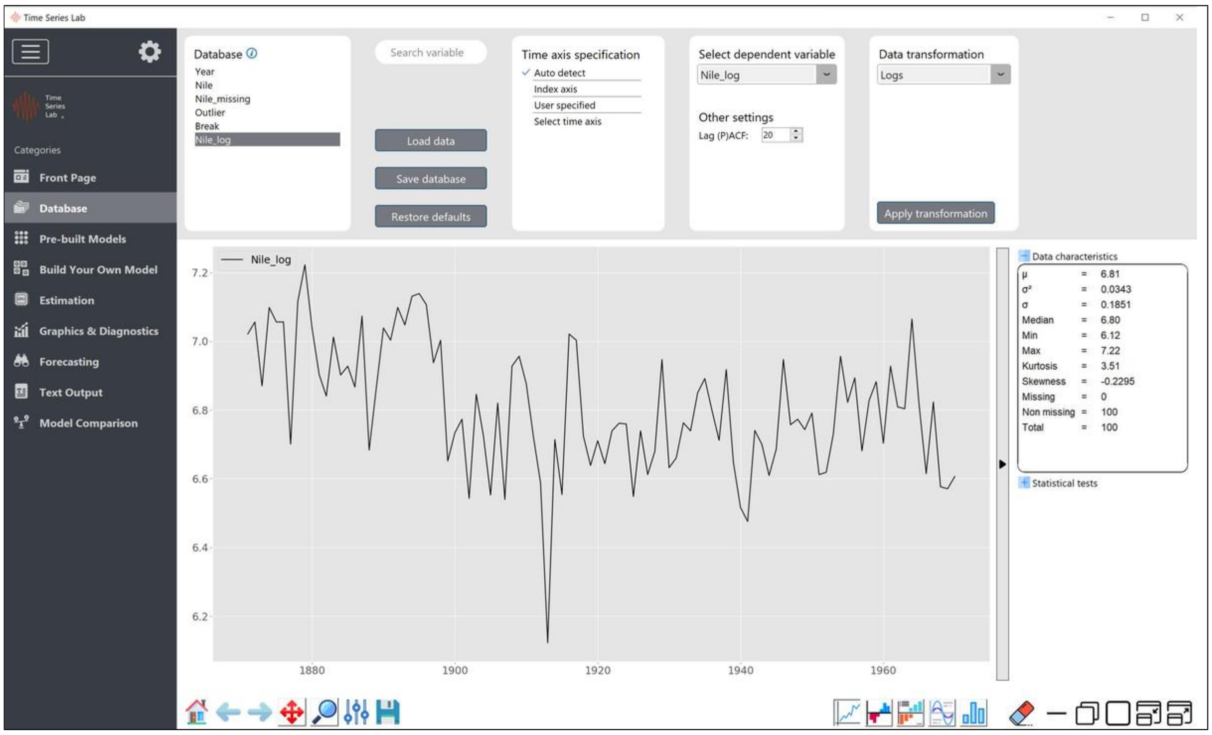
Truncate:
After selecting *Truncate*, two spinboxes appear which allows you to specify the lower and upper bound. These values are in the same units as the time series is in and set all observations

[2]A stationary time series is one whose statistical properties such as mean and variance are constant over time.

outside of the bounds to missing values. Note that missing values can easily be taken into account by TSL.

Winsorize:
After selecting *Winsorize*, two spinboxes appear that allows you to specify the lower and upper *percentage* bound. All observations outside of the percentage bounds are set to the values corresponding to the lower and upper percentages. This means that, in contrast to Truncate, they are not set to missing values.

Figure 3.2
Data inspection and preparation page: Logs of Nile data



Database
Year
Nile
Nile_missing
Outlier
Break
Nile_log

Time axis specification
✓ Auto detect
Index axis
User specified
Select time axis

Select dependent variable: Nile_log

Other settings
Lag (P)ACF: 20

Data transformation: Logs

Data characteristics

μ	=	6.81
σ²	=	0.0343
σ	=	0.1851
Median	=	6.80
Min	=	6.12
Max	=	7.22
Kurtosis	=	3.51
Skewness	=	-0.2295
Missing	=	0
Non missing	=	100
Total	=	100

3.2 Graphical inspection of the data

3.2.1 Type of plots

Different types of time series plots can be activated by selecting one of the six plot types at the bottom right of the Database page.

- Time series: this just plots the selected time series.
- Autocorrelation function (ACF): this describes how well the value of the time series at time t is related with its past values $t-1, t-2, \ldots$. This plot (in combination with the

PACF plot) is often used to determine the p and q values in ARIMA models. The lags of the (P)ACF plot can be controlled by the spinbox below *Other settings*.

- Partial autocorrelation function (PACF): this describes how well the value of the time series at time t is related with a past value with the correlation of the other lags removed. For example, it takes into account the correlation between the values at time t and time $t - 2$ without the effect of $t - 1$ on t. This plot (in combination with the ACF plot) is often used to determine the p and q values in ARIMA models.
- Spectral density: the spectral density and the autocovariance function contain the same information, but expressed in different ways. Spectral analysis is a technique that allows us to discover underlying periodicities for example to find the period of cycle components in the times series. The periodicity of the signal is 2.0 / value on the x-axis.
- Histogram plot: this plots the selected time series in histogram format.
- Seasonal subseries: this plots the seasons from a time series into a subseries, e.g. time series of only Mondays, followed by only Tuesdays. For example if your data is hourly data, set the Seasonal length spinbox to 24 and the Seas. multiplier to 1 to obtain a plot of the intraday pattern. To obtain a plot of the weekdays, set the Seasonal length spinbox to 24 and the Seas. multiplier to 7.

3.2.2 Plot area

The plot area can be controlled by the buttons on the bottom left of the Database page. The *pan/zoom*, *zoom to rectangle*, and *save figure* are the most frequently used buttons.

- Home button: reset original view.
- Left arrow: back to previous view.
- Right arrow: forward to next view.
- Pan/zoom: left mouse button pans (moves view of the graph), right button zooms in and out.
- Zoom to rectangle: this zooms in on the graph, based on the rectangle you select with the left mouse button.
- Configure subplots: this allows you to change the white space to the left, right, top, and bottom of the figure.
- Save the figure: save figure to file system (plot area only). To make a screenshot of the complete TSL window, press Ctrl-p.

Right mouse click on the graph area, opens a popup window in which you can select to set the Titles of the graph, the time-axis, add or remove the legend, and add or remove the grid of the plot area.

3.2.2.1 Data characteristics and statistical tests

When clicked, the vertical arrow bar on the right of the screen shows additional information about the selected time series. The *Data characteristics* panel shows characteristics of the selected time series. It shows statistics like mean, variance, min, and maximum value, among others characteristics. It also shows the number of missing values in the time series. It should be emphasized that:

> Missing values can easily be taken into account in TSL. Even at the beginning of the time series.

The *Statistical tests* panel shows the result of the Augmented Dickey-Fuller test and KPSS test. The null hypothesis of the Augmented Dickey-Fuller test is:

$$H_0 : \text{a unit root is present in the time series}$$

If the p-value $<$ 0.05, H_0 is rejected. For our example Nile dataset, we have a p-value of 0.0005 so we reject the null hypothesis. The null hypothesis of the KPSS test is:

$$H_0 : \text{the series is stationary}$$

If the p-value if $<$ 0.05, H_0 is rejected. For our example Nile dataset, we have a p-value of < 0.01 so we reject the null hypothesis. The results of both test may look contradicting at first but it is possible for a time series to be non-stationary, yet have no unit root and be trend-stationary. It is always better to apply both tests, so that it can be ensured that the series is truly stationary. Possible outcomes of applying these stationary tests are as follows:

Case 1: Both tests conclude that the series is not stationary - the series is not stationary

Case 2: Both tests conclude that the series is stationary - the series is stationary

Case 3: KPSS indicates stationarity and ADF indicates non-stationarity - the series is trend stationary. Trend needs to be removed to make series strict stationary. The detrended series is checked for stationarity.

Case 4: KPSS indicates non-stationarity and ADF indicates stationarity - the series is difference stationary. Differencing is to be used to make series stationary. The differenced series is checked for stationarity.

See also this link for more information. We emphasize that in TSL there is no need to make a series trend or difference stationary but you can if you prefer. One of the many advantages of TSL is that trends and other non-stationary components can be included in the model.

3.2.2.2 Undocking the plot area

The plot area can be undocked from the main window by clicking the *undock draw window* in the bottom right of the screen. Undocking can be useful to have the graph area on a different screen but most of its purpose comes from the area underneath the plot area. For the *Enterprise* edition of TSL, this area is used to select and summarize all selected time series. Since the *Home* edition of TSL is for univariate time series analysis only, this area is just blank and serves no further purpose.

Chapter 4

Pre-built models

After loading our time series data in TSL, it is time to analyze the time series and extract information from it. The fastest way to extract information from your time series is by using the *Pre-built models* page. Select the models you want to use, set the *training and validation sample*, click the green arrow which says *Process Dashboard* and let TSL do the rest. But which time series model to use?

4.1 Model selection

TSL comes with a range of pre-programmed time series models for you to choose from, all with its own characteristics and features. An overview of the available models in TSL is given in Table 4.1. This table provides a short description of the models and, for the interested reader, references to the scientific journals the models were published in. You can select one or multiple models and TSL will give you results of all the selected models in one go. The alternative is to construct your own time series model based on the selection of components. This option will be discussed in Chapter 5.

If you are not sure which time series model to select, use the following guideline. Does your time series data exhibit a trend or seasonal pattern? If neither is present in the time series, start with *Exponential Smoothing* or the *Local Level* model. If your data does contain trending behavior, use the *Holt-Winters* or the *Local Linear Trend* model. If your time series shows a seasonal pattern as well, use the *Seasonal Holt-Winters* or the *Basic Structural model*. The last two models and the *Local Level + Seasonal* model take seasonal effects into account, something which can greatly improve model fit and forecast accuracy. To set the *Seasonal period length* (s), enter a number in one of the Spinboxes located beneath one of the seasonal models. Note that the seasonal period needs to be an integer (whole number). Fractional seasonal periods can be used when you build your own model, see Chapter 5. TSL tries to determine the seasonal period from the loaded data and pre-enters it. If it cannot find the seasonal period length, it reverts to the default period length of 4. You can of course always change this number yourself. Typical examples of seasonal specifications are:

- Monthly data, s=12
- Quarterly data, s=4
- Daily data, s=7 (for day-of-week pattern)

Once the models are selected, set the length of the *training sample* by dragging the slider or by pressing one of the buttons of the pre-set training and validation sample sizes. Model parameters are optimized based on the data in the training sample and the rest of the data belongs to the *validation sample* which is used to assess out-of-sample forecast accuracy. As a rule of thumb, a model is preferred over a rival model, if model fit in the training sample (e.g. in-sample RMSE) is better (lower) AND out-of-sample forecast accuracy in the training sample is better as well. The latter is often harder to achieve compared to improving the fit in the training sample.

The *ENERGY* dataset that comes bundled with TSL has quarterly data on energy consumption. It's an old dataset but good for illustrative purposes since energy consumption changes with the four seasons. If we would like to model the ENERGY dataset with a *Seasonal Holt-Winters* model with *Additive* seasonality (period = 4), a *Basic Structural Model* (period = 4), and we want to combine the forecasts of both models by *Constrained Least Squares Model Averaging*, we set TSL as shown in Figure 4.1

4.2 Score-driven models

Many time series models are build on the assumption of Normally distributed errors. Despite its popularity, many time series require a different distribution than the Normal distribution. One of the major advantages of score-driven models is that you are not restricted to the Normal distribution, in fact you can choose almost any probability distribution. In many cases, using a different probability distribution than the Normal leads to increases in model fit and out-of-sample forecast accuracy.

Score-driven models are a class of linear and non-linear models that can be used to analyse and forecast a wide range of time series. Score-driven models are so versatile that well-known models like ARMA and GARCH models, are subclasses of score-driven models, see also Appendix D. Furthermore, the score-driven model encompasses other well-known models like the autoregressive conditional duration (ACD) model, autoregressive conditional intensity (ACI) model, and Poisson count models with time-varying mean. We refer to Appendix C, Creal et al. (2013), and Harvey (2013) for more information on score-driven models.

The Time Series Lab project started with the *Time Series Lab - Dynamic Score Edition* which focused solely on score-driven models and had many probability distributions that the user could choose from. At the time of writing of this manual, the *Time Series Lab - Dynamic*

Table 4.1

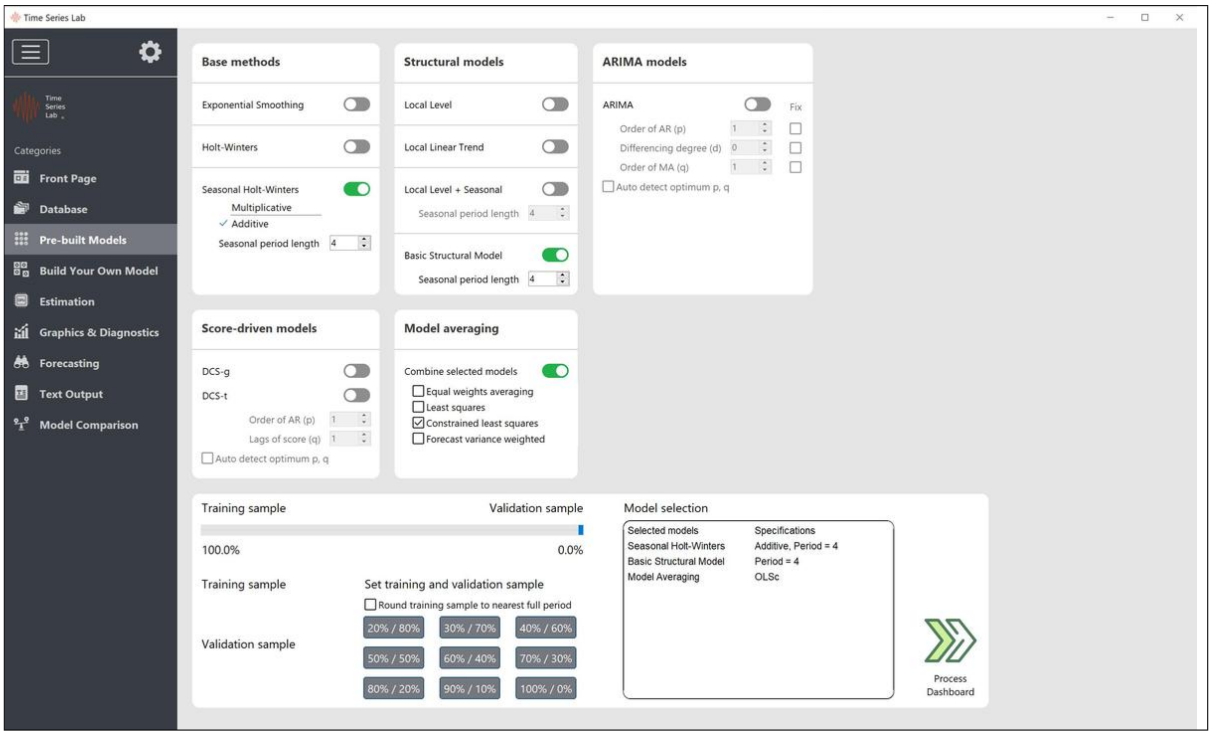
Models of TSL and references to scientific literature

The table reports the models of TSL with features and reference to the literature.

Model description	Model features	Reference
Base models		
Exponential Smoothing Holt-Winters Seasonal Holt-Winters	Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight. This framework generates reliable forecasts quickly and for a wide range of time series, which is a great advantage and of major importance to applications in industry, see https://otexts.com.	Brown (1959), Holt (2004), Winters (1960)
Structural models		
Local Level Local Linear Trend Local Level + Seasonal Basic Structural Model	By structural time series models we mean models in which the observations are made up of trend, seasonal, cycle and regression components plus error. In this approach it is assumed that the development over time of the system under study is determined by an unobserved series of vectors $\alpha_1, \dots, \alpha_n$, with which are associated a series of observations $y_1, \dots, y_n$; the relation between the α_t's and the y_t's is specified by the state space model. In TSL, complex dynamics like multiple seasonalities can modelled with state space models, more information is presented in Chapter 5. Many time series models are special cases of the state space model.	Durbin and Koopman (2012), Harvey (1990)
ARIMA models		
ARIMA(p,d,q)	As with structural time series models, ARIMA models typically regard a univariate time series y_t as made up of trend, seasonal and irregular components. However, instead of modelling the various components separately, the idea is to eliminate the trend and seasonal by differencing at the outset of the analysis. The resulting differenced series are treated as a stationary time series. ARIMA is an acronym for AutoRegressive Integrated Moving Average and any ARIMA model can be put into state space form.	Box et al. (2015)
Score-driven models		
DCS-g, DCS-t	Score-driven models are a class of linear and non-linear models that can be used to analyse and forecast a wide range of time series. Score-driven models are so versatile that well-known models like ARMA and GARCH models are subclasses of score-driven models, see Appendix D, Appendix C, and Chapter 4.2 for more information.	Creal et al. (2013), Harvey (2013)
Model averaging		
Equal weights averaging Least Squares Restricted Least Squares Forecast Variance Weighted	The idea of combining forecasts from different models as a simple and effective way to obtain improvements in forecast accuracy was introduced by Bates and Granger (1969). In almost all cases we cannot identify the true data generating process of the time series, and combining different models can play a complementary role in approximating it.	Bates and Granger (1969), Timmermann (2006), Hansen (2008)

Figure 4.1

Model settings of TSL on the Pre-built models page



Selected models	Specifications
Seasonal Holt-Winters	Additive, Period = 4
Basic Structural Model	Period = 4
Model Averaging	OLSc

Model settings of TSL for time series with quarterly data with seasonality. Additionally, model averaging of the two selected models is selected.

Score Edition is still available for downloaded but the idea is to merge all Time Series Lab projects into one time series package. That would mean that all score-driven models with their specific distributions and features will, over time, be available in the main Time Series Lab package. A start with the merging of score-driven models into the current package is made by introducing the Normal score-driven model (DCS-g) and the Student t score-driven model (DCS-t). The Student t distribution has, so called, fatter tails compared to the Normal distribution. Fatter tails mean that outliers have a higher probability of occuring. The benefit of using the Student t distribution is shown in Case study 12.5.

4.2.1 Auto detect optimum p, q

Both the ARIMA and score-driven models can easily be extended with additional lag structures. This can be done by setting the p and q parameters. For ARIMA models, there is the extra option of setting the parameter d which allows for the series to be differenced before being modelled to make the time series stationary. We refer to the literature in Table 4.1 for more information on lag structures.

Often, including more lags leads to a higher likelihood value which is a measure of model fit. However, including more lags comes at the price of more model parameters that need to be determined. The optimal number of lags p and q, based on the Akaike Information Criterion

(AIC), can be found by selecting the *Auto detect optimum p, q* option. TSL determines the optimum number of lag structures by applying the Hyndman-Khandakar algorithm, see Hyndman and Khandakar (2008).

4.3 Model averaging

The idea of combining forecasts from different models as a simple and effective way to obtain improvements in forecast accuracy was introduced by Bates and Granger (1969). In almost all cases we cannot identify the true *Data Generating Process* of the time series, and combining different models can play a role in approximating it. A popular way to combine the individual predictions is to consider the following linear model:

$$Y_t = X_t^{'}\beta + \varepsilon_t, \qquad t = 1, \ldots, T \tag{4.1}$$

where ε_t is white noise which is assumed to have a normal distribution with zero mean and unknown variance, Y_t is our observed time series, and $X_{i,t}$ the point forecast from one of our selected models for $i = 1, \ldots, k$ with k the number of selected models and T the length of our time series. The parameter vector β can be chosen in different ways each leading to a different combination of models.

4.3.1 Equal weights averaging

This is the simplest of the model averaging techniques. All weights are chosen equal, meaning that each point forecast has weight $1/k$ which gives:

$$\hat{\beta}_{EWA} = (\tfrac{1}{k}, \ldots, \tfrac{1}{k}) \qquad \text{and} \qquad \tilde{Y}_t^{EWA} = \tfrac{1}{k}\sum_{i=1}^{k} X_{i,t}.$$

4.3.2 Least squares

A natural extension to the *Equal weights averaging* method is to determine the weights by *Least squares* estimators. This OLS approach to combine forecasts was proposed by Granger and Ramanathan (1984). They used OLS to estimate the unknown parameters in the linear regression model of (4.1). The OLS estimator of the parameter vector β of the linear regression model is

$$\hat{\beta}_{OLS} = \left(X^{'}X\right)^{-1} X^{'}Y.$$

Note that the $T \times (k+1)$ matrix X has an intercept in its first column. The estimate $\hat{\beta}_{OLS}$ is unrestricted meaning that negative weights are allowed. This model often performs very well in the training set but not always for the validation set. To counter this issue of overfitting, *Restricted least squares* might be a good alternative.

4.3.3 Restricted least squares

For the restricted OLS approach, the unknown parameters in the linear regression model (4.1) are obtained by restricting each of the elements of $\hat{\beta}_{OLSc}$ to lie between 0 and 1. Furthermore, the elements of $\hat{\beta}_{OLSc}$ must sum to 1. Note that the $T \times k$ matrix X does **not** have an intercept. The estimate $\hat{\beta}_{OLSc}$ is restricted and negative weights cannot occur.

4.3.4 Forecast variance weighted

This method is also called *Bates-Granger averaging*. The idea is to weight each model by $1/\sigma_i^2$, where σ_i^2 is its forecast variance. In practice the forecast variance is unknown and needs to be estimated. This leads to

$$\hat{\beta}_{FVW,i} = \frac{1/\hat{\sigma}_i^2}{\sum_{j=1}^{k} \hat{\sigma}_i^2},$$

where $\hat{\sigma}_i^2$ denotes the forecast variance of model i which we estimate as the sample variance of the forecast error $e_{i,t} = X_{i,t} - Y_t$ within the training sample period.

Chapter 5

Build your own model

Instead of selecting a pre-defined model from the *Pre-built models* page, you can also build your own model. This requires some basic knowledge and logic but no extensive statistical knowledge is needed. With the help of this chapter, you can come a long way. If in doubt, you can always try adding or removing components to see the effect on model fit and forecast performance. TSL is written in a robust way and adding components should not break anything so you are free to experiment. If things break, please let us know so we can make TSL even more robust!

5.1 Structural time series models

The Structural Time Series Model allows the explicit modelling of the trend, seasonal and error term, together with other relevant components for a time series at hand. It aims to present the stylised facts of a time series in terms of its different dynamic features which are represented as unobserved components. In the seminal book of Harvey (1990) it is stated as follows: "The statistical formulation of the trend component in a structural model needs to be flexible enough to allow it to respond to general changes in the direction of the series. A trend is not seen as a deterministic function of time about which the series is constrained to move for ever more. In a similar way the seasonal component must be flexible enough to respond to changes in the seasonal pattern. A structural time series model therefore needs to be set up in such a way that its components are stochastic; in other words, they are regarded as being driven by random disturbances." A framework that is flexible enough to handle the above requirements is the State Space model.

The Basic structural time series model is represented as:

$$y_t = \mu_t + \gamma_t + \varepsilon_t, \qquad t = 1, \ldots, T$$

where y_t is our time series observation, μ_t is the trend component, γ_t the seasonal component, and ε_t the error term, all at time t. In TSL you can select other or additional components like cycle (ψ_t), autoregressive components (AR$_t$), Explanatory variables ($X_t\beta_x$), and Intervention variables ($Z_t\beta_z$). We discuss each dynamic component and its characteristics.

A summary of all *selected* components of the *Build your own model* page and its characteristics is given in the blue *Model specification area* of TSL.

Important: Dynamic components each have unique characteristics and can be combined to form complicated models that reveal hidden dynamics in the time series.

Intermezzo 1: Time-varying components

TSL extracts a *signal* from the observed time series. The difference between the observed time series and the *signal* is the *noise*, or the error. The methodology of TSL relies heavily on filtering time series with the aim to remove the noise from the observation and to secure the signal in the time series and possibly to identify the different components of the signal. We are interested in the *signal* because it provides us the key information from the past and current observations that is relevant for the next time period. In its simplest form, the *signal* at time t is equal to its value in $t-1$ plus some innovation. In mathematical form we have

$$\alpha_t = \alpha_{t-1} + \text{some innovation},$$

with α_t being the *signal* for $t = 1, \ldots, T$ where T is the length of the time series. The innovation part is what drives the *signal* over time. A more advanced model can be constructed by combining components, for example

$$\alpha_t = \mu_t + \gamma_t + X_t\beta,$$

where μ_t is the level component, γ_t is the seasonal component, $X_t\beta$ are explanatory variables, and where each of the components have their own updating function.

5.1.1 Level

The *Level* component μ_t is the basis of many models. If only the (*time-varying*) level is selected the resulting model is called the *Local Level model* and, informally, it can be seen as the *time-varying* equivalent of the intercept in the classical linear regression model. The *time-varying* level uses observations from a window around[1] an observation to optimize model

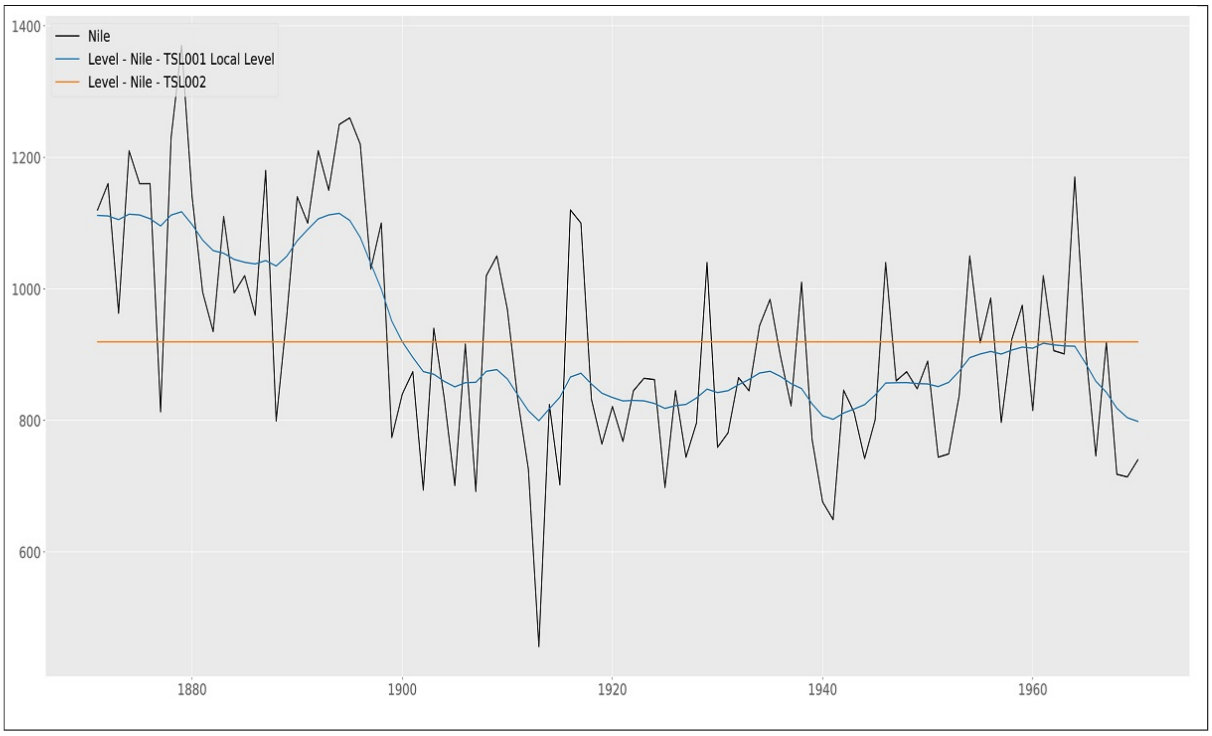
[1]We are interested in the statistical behavior of the state, α_t, (which includes the level) given a subset of the data, i.e. the data up to time $t-1$ (forecasting), the data up to time t (filtering) or the whole data set (smoothing). The choice for forecasting, filtering, or smoothing determines which window around the

fit locally. How much each observation contributes to the local level fit is optimized by the algorithms of TSL.

The *fixed* version of the level is often used in combination with other components. We will see examples of the use of a *fixed* (or static) level later in this manual. Figure 5.1 shows the result of fitting the local level to the Nile data. For completeness, *fixed* level is added for comparison. Needless to say, the *time-varying* local level model fits the data better.

Figure 5.1

Nile data with Local level model, time-varying and static



The figure shows the result of fitting the local level to the Nile data. A fixed level is added for comparison. Needless to say, the time-varying local level model fits the data better. Lines shown are from the Kalman Smoother, see Appendix B for more information on Kalman Filtering and Smoothing.

5.1.2 Slope

The *slope* component ν_t can only be selected in combination with the level component. It is used for time series that exhibits trending behavior. If both a time-varying level and time-varying trend are selected, the model selection corresponds to the *Local Linear Trend* model. Certain combinations of the level and slope component can have interesting effects on the smoothness of the extracted signal. For example, if the level is set to fixed and the slope to time-varying, often a much smoother signal is obtained. In the literature, the resulting model is known as an *Integrated Random Walk* model. Varying levels of smoothness can also be

observations we can use. For example, for forecasting we can only use the data up to time $t-1$ and therefore the *window* around the data is limited to the data before the current observation at time t only. If we would use data after time t for forecasting we would use the future to forecast the future!

achieved by setting the *order* of the trend to a higher number. In general (but definitely not always), the higher the order, the smoother the resulting signal.

5.1.3 Seasonal short

The inclusion of the *Seasonal* component γ_t^s in our model allows us to model a wide range of time series. Time series can contain a variety of seasonal effects and in TSL you can add three seasonal components to your model if needed. The info button ⓘ, next to the seasonal short component, tells us:

> Seasonal period length is the number of time points after which the seasonal repeats. Examples of seasonal specifications are:
>
> Monthly data, s = 12.
> Quarterly data, s = 4.
> Daily data, when modelling the weekly pattern, s = 7.
>
> The seasonal period can be a fractional number. For example, with daily data, specify a period of 365.25 for a seasonal that repeats each year, taking leap years into account. See the case studies on the timeserieslab.com website for more information on how to specify seasonals.
>
> Number of factors specifies the seasonal flexibility. Note that a higher number is not always better and parsimonious models often perform better in forecasting.

It's best to explain the seasonal component with an example. Let's say our time series is weekly data on *gasoline* consumption, see also Case study 12.2. With gasoline consumption, fluctuations are to be expected throughout the year due to, for example, temperature changes during the year. For the moment assume we have 52 weeks in a year and we would therefore specify s = 52.0 as the seasonal period. For the number of factors, we specify 10. As a rule of thumb, do not take the maximum amount of factors (which is s/2) because this makes the seasonal very flexible which is good for your training sample fit but often performs worse in forecasting. Another disadvantage of taking a "large" number of factors is that the model becomes slower to estimate. This is however a general guideline and experimenting might be necessary. Future version of TSL determine the optimal set of factors.

Now let's assume that our data on gasoline consumption is still weekly data but we realize that we need $s > 52.0$ since we have more than 52.0 weeks in a year. On top of that our dataset also contains a leap year. We therefore set the seasonal period to $s = 365.25/7 = 52.179$ where 365.25 is the average number of days in a year in a four year time span including one leap year and 7 the number of days in one week. This small change of $52.179 - 52.0 = 0.179$ in seasonal period length can make a big difference in forecasting as we will see in Case study 12.2.

Another example would be *hourly* electricity demand. We can expect electricity demand to change within a 24h cycle with more energy demand during the daytime and less during night time. For this example we would set $s = 24.0$ to model the 24h cycle within a day, see also Case study 12.10.

5.1.4 Seasonal medium

If we want to include only one seasonal component in our model we should take the *Seasonal short*. But if we want to model a double seasonal pattern we can include *Seasonal medium* γ_t^m as well. Continuing with our *hourly* electricity demand example, we can use the seasonal medium to model the day of week pattern on top of the 24h intraday pattern. We can expect energy demand to be lower during the weekend since, for example, many business are closed. To model this, we set the seasonal period length of the seasonal medium component to $s = 7 \times 24 = 168$. For the number of factors, we can specify a number around 20.

5.1.5 Seasonal long

Continuing with our *hourly* electricity demand example, we can use the seasonal long γ_t^l to model the demand pattern throughout the year. Since energy demand often changes with the four seasons of the year we can $s = 24 \times 365.25 = 8766.0$. Note that our time series needs to be long and preferably several times s to take the yearly pattern into account.

A combination of seasonal patterns can strongly increase forecast precision as we will see in Case study 12.10.

5.1.6 Cycle short / medium / long

The cycle and seasonal components have similarities since both components have repeating patterns. The big difference however is that the seasonal component has a fixed, user set, period while the period of the cycle components $\psi_t^s, \psi_t^m, \psi_t^l$ are determined from the data. This becomes useful if you want to, for example, model GDP and determine the length of the business cycle. Time series can contain multiple cycles as we will see in Case study 12.4 where an El Nino time series contains complex dynamics with three cycle patterns. The statistical specification of a cycle ψ_t is as follows:

$$\begin{bmatrix} \psi_t \\ \psi_t^* \end{bmatrix} = \rho \begin{bmatrix} \cos\lambda_c & \sin\lambda_c \\ -\sin\lambda_c & \cos\lambda_c \end{bmatrix} \begin{bmatrix} \psi_{t-1} \\ \psi_{t-1}^* \end{bmatrix} + \begin{bmatrix} \kappa_t \\ \kappa_t^* \end{bmatrix}, \qquad t = 1, \ldots, T \tag{5.1}$$

where λ_c is the frequency, in radians, in the range $0 < \lambda_c < \pi$, κ_t and κ_t^* are two mutually uncorrelated white noise disturbances with zero means and common variance σ_κ^2, and ρ is a damping factor with $0 \leq \rho \leq 1$. Note that the period is $2\pi/\lambda_c$. The stochastic cycle becomes a first-order autoregressive process if $\lambda_c = 0$ or $\lambda_c = \pi$. The parameters $\lambda_c, \rho, \sigma_\kappa^2$ are determined by the algorithms of TSL.

5.1.7 ARMA(p,q) I and II

An autoregressive moving average process of order p, q, ARMA(p,q), is one in which the current value is based on the previous p values and error terms occurring contemporaneously and at various times in the past. It is written as:

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_p \epsilon_{t-p} + \epsilon_t,$$

with ϵ_t being white noise, $\phi_1, \ldots, \phi_{t-p}$ being the AR coefficients that determine the persistence of the process, and $\theta_1, \ldots, \theta_{t-p}$ being the MA coefficients. The process is constrained to be stationary; that is, the AR and MA coefficients are restricted to represent a stationary process. If this were not the case there would be a risk of them being confounded with the random walk component in the trend. Since the ARMA(p, q) processes of TSL are stationary by construction, the processes fluctuates around a constant mean which is zero in our case. The persistence of fluctuations depend on the values of the ϕ parameters. When there is a high degree of persistence, shocks that occur far back in time would continue to affect y_t, but by a relative smaller amount than shocks from the immediate past. If an autoregressive process is needed that fluctuates around a number other than zero, add a **fixed** (constant) *Level* component in combination with the autoregressive process.

A second ARMA(p, q) process can be added to the model. In this setup, the first autoregressive process captures persistent and long run behavior while the second autoregressive process captures short term fluctuations.

5.1.8 Explanatory variables

Explanatory variables play an important and interesting role in time series analysis. Adding explanatory variables can significantly improve model fit and forecasting performance of your model. The difficulty often lies in finding explanatory variables that significantly contribute to model fit and forecasting performance. If we switch *Explanatory variables* on, a new window opens like the one presented in Figure 5.2. Alternatively, if the window does not popup, you can click the *Adjust selection* button to bring the window to the front. You can choose between *Manually* and *Automatically*. Manually means, all selected variables will be added to the model, automatically means, variables are selected based on a significance level that you can set, see also Section 5.1.8.3. In *Automatic* mode, a TSL algorithm adds and removes variables and iteratively re-estimates the model in between to end up with a set of explanatory variables that all have their t-stats above a specified threshold. We note that our algorithm does not simply remove variables one-by-one. Variables can re-enter the equation at later stages to increase the probability of ending up with an optimal set of explanatory variables.

Important: We are fully aware that people engage in heated debates on do or do not remove explanatory variables based on statistical relevance. We do not contribute to that debate here, you just have the option to auto-remove variables or leave them all in.

5.1.8.1 Select variables

In the explanatory variables window you see a list of all variables in the database with colored indicators in front of them. Hovering with the mouse over the info button ⓘ, you see

> To select multiple explanatory variables, click the first variable, then Ctrl-click the next variable, and so on. Click on a variable and press Ctrl-a to select all variables. A consecutive range of variables can be included by clicking the first variable and Shift-click the last variable in the range.
>
> The color in the first column indicates how likely it is that the variable contributes significantly to the overall model fit. The green variables should be considered first and the red ones last. The Indicator lights are based on a pre-analysis and only when the full dynamic model is estimated can we say something about the actual contribution of the variable.
>
> Important:
> The pre-analysis is determined based on results from the last model run.

The explanation of the indicators is in the bottom right corner of the newly opened window. The colors show the likelihood that an explanatory variable contributes significantly to the fit of the model. The more significant a variable is, the higher it is on the color ranking scale. Of course, we can only be certain after we estimate the model but it gives an idea. The colors are determined based on a regression of the currently selected time series on the rest of the variables in the database, as in

$$v_t = V_t\beta + \epsilon_t, \qquad t = 1, \ldots, T, \tag{5.2}$$

where v and V correspond to y and X with dynamics removed. The idea is to use information from the last model to remove dynamics (trend, seasonal, etc.) from the y and X data to end up with v and V. After that, a regression from v on V should now reveal relationships that cannot be explained by dynamics alone and could therefore explained by the regression variables. The more significant $\hat{\beta}_i$ is, the higher the corresponding regression variable X_i will be on the color ranking scale.

Some variables are not allowed in the model. These are variables that have values that cannot be represented by a number, like the values in the date column for example.

The black dot indicator deserves a bit more attention. It is possible to add a black indicator variable to the model. However, it should be used with a lot of caution. The reason is as follows: black indicator variables are time series that are added to the database *after* estimation. Typically, these are extracted signals like level or seasonal. Take the following basic structural model

$$y_t = \mu_t + \gamma_t + \varepsilon_t, \qquad t = 1, \dots, T \tag{5.3}$$

with level μ_t seasonal γ_t and irregular ε_t. When this model is estimated in TSL (Chapter 6 explains Estimation), the extracted components μ_t and γ_t can be added to the database and show up in the explanatory variables window where they get a black indicator. If now the following model is estimated:

$$y_t = \mu_t + X_t\beta + \varepsilon_t, \qquad t = 1, \dots, T \tag{5.4}$$

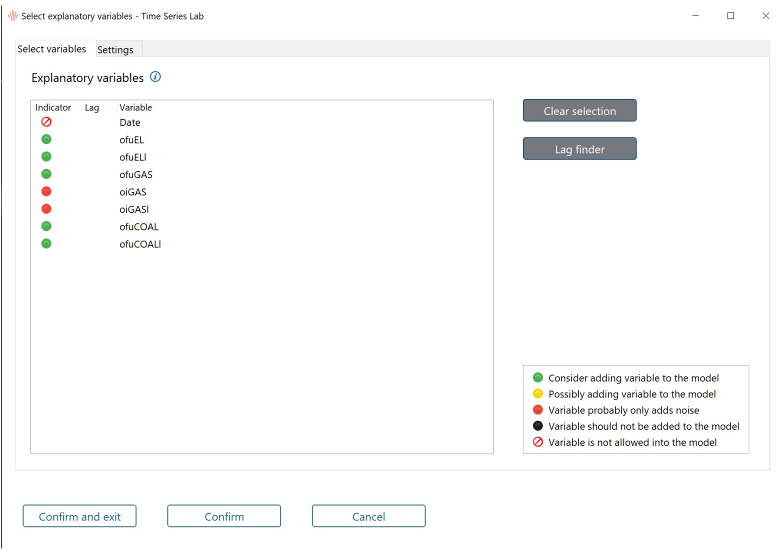
where X_t is the black indicator (Smoothed) seasonal component that was obtained from model (5.3) we could think we have the same model as in 5.3. However, this is not true and we introduced two errors in the model. The first one is that each component in a state space model has its own variance (we discuss error bounds in Chapter 7.3.1) but if extracted dynamic components are added to the model as explanatory variables, the variance gets deflated and does not represent the true uncertainty in the extracted signals anymore. The second problem is that the smoothed seasonal is based on all data so adding that to a new model for the same time series y_t is using more observations than you would normally do for prediction. You will see that you artificially increase model fit by doing this.

A reason to still include a black indicator variable is if an extracted signal is used as explanatory variable in a model with a different time series y_t.

Figure 5.2

TSL - selection of *Explanatory variables*

TSL window in which *Explanatory variables* can be selected. All variables of the database show up here, except the currently selected y variable, and each of the variables is color coded to signal its possible importance to the overall model fit.



Indicator	Lag	Variable
⊘		Date
●		ofuEL
●		ofuELI
●		ofuGAS
●		oiGAS
●		oiGASI
●		ofuCOAL
●		ofuCOALI

● Consider adding variable to the model
● Possibly adding variable to the model
● Variable probably only adds noise
● Variable should not be added to the model
⊘ Variable is not allowed into the model

5.1.8.2 Lag finder

The *Lag finder* module uses the same auxiliary regression as in (5.2) but instead of the contemporaneous X variables in the dataset, it aims to find candidates for significant lags of the X variables. In time series analysis, often one series leads another. For example, an explanatory variables at time $t-3$ can explain part of the variation in y_t. Instead of trying out all possible lags of all variables, the lag finder module aims at finding these lags for you. Just as with the regression of (5.2), we can only be certain if a lagged variable contributes significantly to the model fit after we estimate the model.

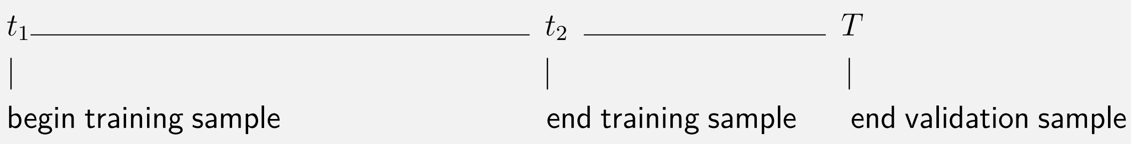
Select the X variable(s) for which you want to find significant lags and press the *Lag finder* button. If significant lags are found, they show up with a green indicator light in the explanatory variable list.

5.1.8.3 Settings

The second tab of the Explanatory variables window is the *Settings* tab. On this tab we set the t-stat after which an indicator becomes green on the Select variables tab. We can also set the *Maximum allowed p-value* which determines how strict we are in removing variables from the model in the *Automatic* removal of explanatory variables. Furthermore, you can set which method to use to replace missing values in explanatory variables and which method to use to forecast explanatory variables. The need to forecast explanatory variables is given below.

Intermezzo 2: Forecasting explanatory variables

In the diagram below, we have three time points, t_1, t_2, and T. The start and end of the training sample are denoted by t_1 and t_2. The start and end of the validation sample dataset are denoted by $t_2 + 1$ and T.



The time points t_1 and t_2 can be set on the Pre-built models page or the Estimation page. If t_2 is set to the end of the dataset (time point T), we have $t_2 = T$ and a validation sample is not specified.

Important, in the case of $t_2 < T$ (meaning we have a validation sample), we distinguish two situations:

- one-step-ahead forecasting
- multi-step-ahead forecasting

For one-step-ahead forecasting we can simply use the explanatory variables from the loaded dataset. For multi-step-ahead forecasting we need to forecast the explanatory variables for the time points $t_2 + 1$ to T. This means that we make 1-step, 2-step, 3-step, ..., n-step-ahead forecasts till we reach time point T.

You can choose one of the methods above for multi-step-ahead forecasting of the explanatory variables.

5.1.9 Intervention variables

Intervention variables are a special kind of explanatory variables. There are two types, *Outliers* and *Structural breaks*, both are very useful for anomaly detection. For example, early warning systems rely on anomaly detection, also called outlier and break detection. Could a catastrophic event have been seen in advance? Take for example sensor readings from an important piece of heavy machinery. The breaking down of this machine would cost a company a lot of money. If anomalies were detected in the sensor reading, preventive maintenance might have saved the company from a break-down of the machine.

For example, if we have the following stochastic trend plus error model

$$y_t = \mu_t + \lambda Z_t + \varepsilon_t, \qquad t = 1, \ldots, T$$

where Z_t is an intervention (dummy) variable and λ is its coefficient. If an unusual event is to be treated as an outlier, it may be captured by a pulse dummy variable at time $t = \tau$, that is

$$Z_t = \begin{cases} 0 & \text{for } t \neq \tau \\ 1 & \text{for } t = \tau. \end{cases} \tag{5.5}$$

A structural break in the level at time t may be modelled by a level shift dummy,

$$Z_t = \begin{cases} 0 & \text{for } t < \tau \\ 1 & \text{for } t \geq \tau. \end{cases} \tag{5.6}$$

If we switch *Intervention variables* on, a new window opens. Alternatively, if the window does not open, you can click the *Adjust selection* button to bring the window to the front. Just as with explanatory variables, you can choose between *Manually* and *Automatically*. If set to manual, the window opens up showing the *Select interventions* tab. Here you can specify time points where you want to position outliers and / or structural breaks. Hovering with the mouse over the info button ⓘ, we see

Select outliers and structural breaks. If an Outlier and Structural break are set at the same time point, the Outlier gets precedence.

Add an outlier or structural break by clicking one of the buttons and change the index to reflect the desired date.

If *Automatically* is selected, the *Settings* tab is shown. In the automatic case, TSL finds the outliers and structural breaks for you. The *Lowerbound t-stats* and *Stop algorithm when* both have to do with the stopping condition of the algorithm. Furthermore, the checkboxes allow you to exclude outliers or structural breaks from the algorithm. We will see an example of automatic outlier detection in Case study 12.3, among other.

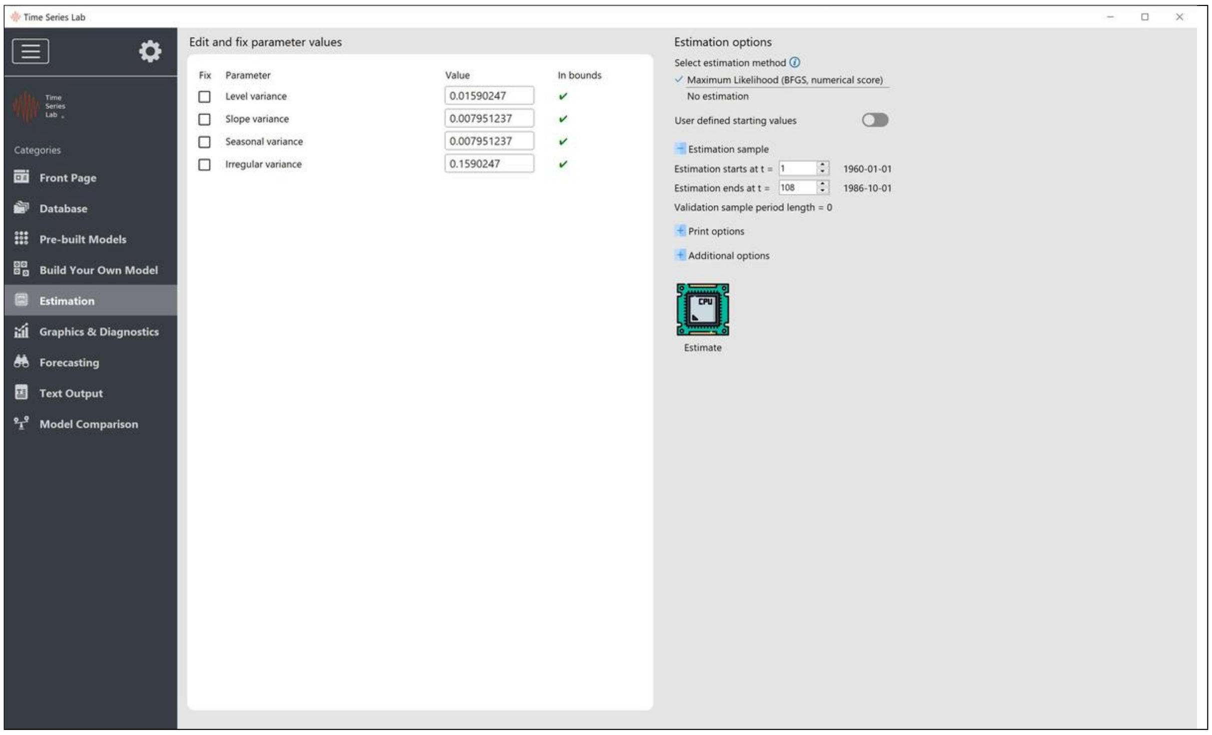
Chapter 6

Estimation

The *Estimation* page of TSL looks like Figure 6.1. Getting results from TSL can be achieved in two ways. The first one was discussed in Chapter 4 by using the *Process Dashboard* button. The second one is by using the *Estimate* button on the *Estimation* page. Before you press this button there are certain things useful to know, so let's discuss them.

Figure 6.1

Estimation page of TSL



Edit and fix parameter values

Fix	Parameter	Value	In bounds
☐	Level variance	0.01590247	✓
☐	Slope variance	0.007951237	✓
☐	Seasonal variance	0.007951237	✓
☐	Irregular variance	0.1590247	✓

Estimation options

Select estimation method

✓ Maximum Likelihood (BFGS, numerical score)

No estimation

User defined starting values

− Estimation sample

Estimation starts at t = 1 1960-01-01

Estimation ends at t = 108 1986-10-01

Validation sample period length = 0

+ Print options

+ Additional options

6.1 Edit and fix parameter values

Components that we have selected on the *Build your own model* page all have corresponding parameters that can be set, fixed, and / or estimated. After entering the Estimation page, TSL has set the parameters to rudimentary starting values based on the sample variance of the selected time series. These values will later be refined depending on what you tell TSL to do. Model output results depend strongly on the parameters as listed in the *Edit and fix parameter values* section. There are five ways in which you can influence parameter values.

(1) Don't do anything and press the *Estimate* button. TSL uses an algorithm to determine optimal starting values and estimates the model parameters. This is the default option.

(2) Fix a subset of parameters by ticking the boxes in the *Fix* column and by setting the fix values in the *Value* column followed by pressing the *Estimate* button. TSL again uses an algorithm to determine optimal starting values and estimates the model parameters but leaves the fixed parameters to the specified values. Fixing parameter can be useful to force certain smoothness of the components.

(3) Switch the *User defined starting values* button **on** and press the *Estimate* button. TSL now does **not** determine optimal starting values and instead uses the values in the value column as starting values for the optimization algorithm.

(4) Set the *Set estimation method* to *No estimation* and switch **on** *User defined starting values*. You see the *Estimate* button change to a *Process selection* button with a green arrow. After clicking it, TSL will use the parameter values in the value column and proceeds to output without optimizing the parameter values.

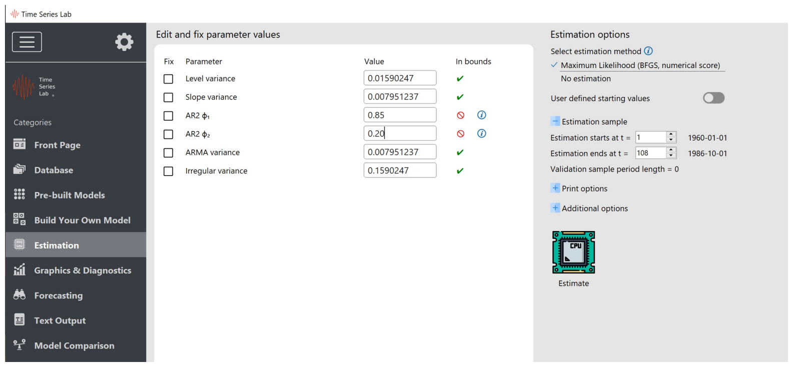
(5) Set the *Set estimation method* to *No estimation* and switch **off** *User defined starting values*. You see the *Estimate* button change to a *Process selection* button with a green arrow. After clicking it, TSL will determine optimal starting values but does **not** proceed with further optimization and instead goes straight to output.

We can say something, in general, about ordering of the five ways based on which setting obtains results fastest. If we rank the methods in order of speed starting with the fastest method we have, (4) – (5) – (2, 3) – (1). Starting values can have a big impact on the speed of estimation, therefore we cannot say if (2) is faster than (3) or vice versa. We are often interested in the best model fit for the validation sample. For this we choose method (1), which is the default.

The software checks the user input in the *Value* entry boxes because some parameters are restricted to a certain range, see for an example Figure 6.2.

Figure 6.2
TSL warning for a non-stationary *AR(2)* process

TSL warns the user via the *In bounds* column that the (user) specified values lead to an *AR(2)* process that is non-stationary. Hovering over the info button, we can read "Value is missing or outside of admissible bounds. Autoregressive processes are restricted to be stationary."



Edit and fix parameter values

Fix	Parameter	Value	In bounds
☐	Level variance	0.01590247	✓
☐	Slope variance	0.007951237	✓
☐	AR2 ϕ_1	0.85	⃠ ⓘ
☐	AR2 ϕ_2	0.20	⃠ ⓘ
☐	ARMA variance	0.007951237	✓
☐	Irregular variance	0.1590247	✓

Estimation options

Select estimation method ⓘ
[✓ Maximum Likelihood (BFGS, numerical score)]
No estimation

User defined starting values

− Estimation sample
Estimation starts at t = 1 1960-01-01
Estimation ends at t = 108 1986-10-01
Validation sample period length = 0

+ Print options

+ Additional options

6.2 Estimation options

The model parameters, also called *hyper parameters*, can be estimated by *Maximum Likelihood* with the *BFGS* algorithm or *No estimation* can be selected so that text and graphical output will be based on the provided starting values. For the majority of the models, maximizing the likelihood is a routine affair. Note however, that for more complicated models and the increase of the number of *hyper parameters*, optimization can be complex. The info button next to the *BFGS* option tell us that:

> The BFGS method falls in the category of quasi-Newton optimizers. This class of optimizers is applicable to many optimization problems and is often used to maximize a likelihood function.
>
> Please be aware that finding a global optimum is not guaranteed and trying different starting values increases the chance of finding the global optimum.

We are not restricted to estimating the full sample in our data set. If needed, we can restrict the estimation to a smaller sample by setting the *Estimation starts at t* and *Estimation ends at t* entry boxes.

During estimation results can be communicated to you via the *Text output* page (see also Chapter 9). You can choose the amount of detail that is communicated by TSL, we have

- Print model information. This prints the type of model that is currently estimated, the dependent variable, and the selection sample $(t_1 - t_2)$.
- Print optimization warnings. This prints starting values, progress of optimization, and optimized parameter values.
- Print warnings. This prints general warning messages.

Furthermore, we have three buttons under *Additional options*. The *Set default estimates* button, sets the values in the value column back to the rudimentary starting values that were initially there. The *Set default estimates* button, sets the values in the value column to the most recent optimized parameter values.

After the successful estimation of a model, a (hyper) parameter report can be generated. Clicking the *Parameter report* button brings us to the *Text output* page where the parameter report will be printed.

Important: The time it takes to generate the parameter report depends strongly on the number of *hyper parameters*, the number of model components, and the length of the time series. The generation of a parameter report can take the amount of time it takes to maximize the likelihood.

Chapter 7

Graphics and diagnostics

Graphical inspection of the estimation results is an important step in the time series modelling process. You are taken automatically to the *Graphics* page of TSL after the estimation process has finished. We can manually reach this page by clicking the button shown on the left. The default graphical output for the *Nile* data illustration is displayed in Figure 7.1.

7.1 Selecting plot components

Plotting components is very simple: it only requires ticking the check box corresponding to the component you would like to see in the graph. Note that some components are *greyed out* as they were not selected as part of the model. We have five plot tabs in the upper left corner of the program,

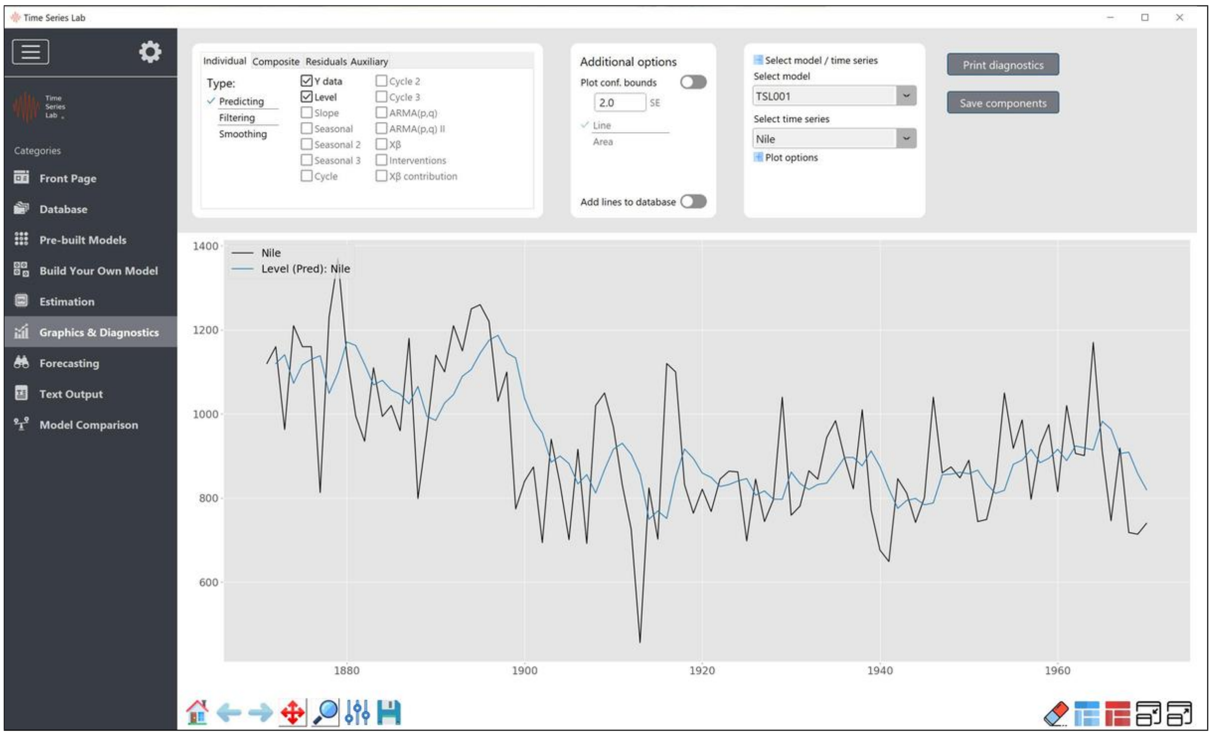
- Individual: these correspond to single components like Level, Slope, Seasonal etc. The "$X\beta$ *contribution*" component shows the contribution of each explanatory variables to the total explanatory variable signal $X\beta$. It is also called a *Sand graph*.
- Composite: these correspond to plots that consist of sums of components like for example the *Total signal* which is the sum of all selected components of the model. In Intermezzo 1, the Total signal would be the sum of the level, seasonal, and explanatory variables component, i.e. $\alpha_t = \mu_t + \gamma_t + X_t\beta$. In our Local Level model and *Nile* data illustration we have the situation that the *Level* and the *Total signal* are exactly equal due to the *Level* being the only component in the model.
- Residuals: residuals are the remainder of the time series after the signal is removed. It is the part that can not be explained by the model. Standardized residuals form the basis of *model specification* tests. For example, tests based on the autocorrelation function. The *Autocorrelation function* (ACF) plot of the *Standardized residuals* are shown in Figure 7.2. This plot is used to assess the performance of the model in explaining the autocorrelation in the time series. If we see bars larger than the confidence bounds (horizontal lines) we could try different model specifications since there is signal left

to explain. The Local Level model nicely captures the dynamics in the time series, although one spike at lag 10 is close to the confidence bound.

- Auxiliary: these are special types of residuals and are, for example, used in outlier detection.
- Transformations: Detrend or seasonally adjust Y based on the extracted components of the level and the seasonal.

Figure 7.1

Graphical output for Nile data and Local Level model

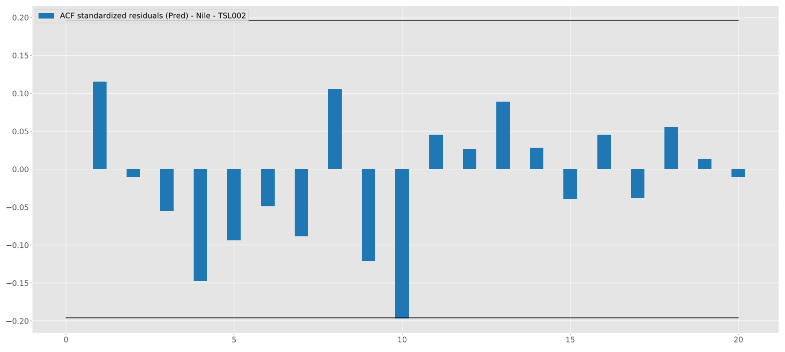

Type: ✓ Predicting
☑ Y data
☑ Level
Plot conf. bounds 2.0 SE
Select model TSL001
Select time series Nile

Despite our simple *Local Level* model, which is often a useful model, we see that the model nicely follows the data. It, of course, lags the data by one period because a predictive filter is always based on data up to time $t-1$.

Figure 7.2

Autocorrelation plot of Nile data standardized residuals

Autocorrelation plot of Nile data Standardized residuals. The Local Level model with nicely captures the dynamics in the time series.



The difference between *Predicting, Filtering* and *Smoothing* is further explained in Appendix B. These three options are available for State Space models which is every model combination on the *Build your own model* page and the Local Level, Local Linear Trend, Local Level + Seasonal, and Basic Structural Model on the *Pre-built models* page.

7.2 Plot area

The majority of the buttons *below* the plot area is explained in Chapter 3 except two new buttons. These are the *add subplot* and *remove subplot* button. Just as the name says, they add and remove subplots from the plot area. The plot area of TSL can consist of a maximum of nine subplots. Subplots can be convenient to graphically summarize model results in one single plot with multiple subplots. Notice that after clicking the *add subplot* button an *empty* subplot is added to the existing graph which corresponds to **no** check boxes being ticked.

Important: The components that are graphically represented in a subplot directly correspond to the check boxes that are ticked. Clicking on a subplot activates the current plot settings.

Notice that by clicking a subplot, a blue border appears shortly around the subplot as a sign that the subplot is active. If you hover the *Clear all* button, the tooltip window shows:

> The *Clear all* button clears everything from the figure including all subplots. To have more refined control over the subplots, right-mouse click on a subplot for more options.

If you hover the *Add subplot* button, the tooltip window shows:

> Click on a subplot to activate it. Notice that by clicking on a subplot, the checkboxes in the top left of the window change state based on the current selection of lines in the subplot.
>
> If not all checkbox settings correspond with the lines in the subplot, switch the tabs to show the rest of the selection.

7.3 Additional options

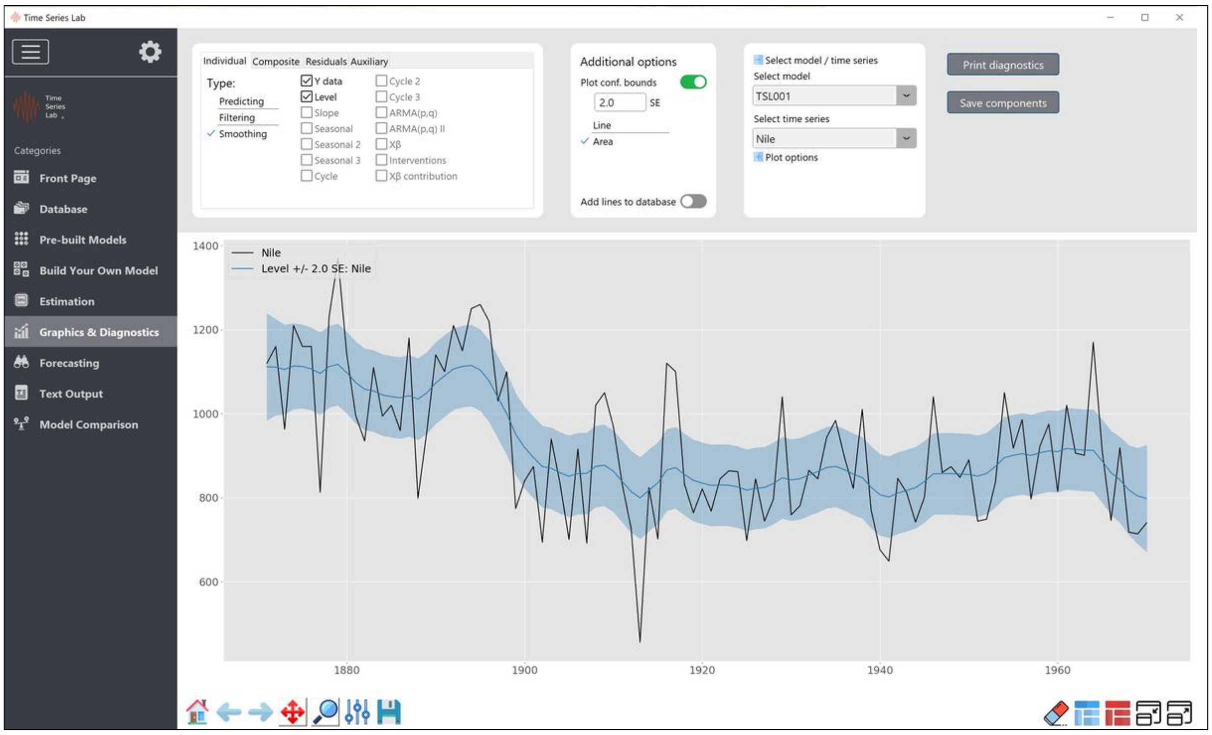
Several more plot options are available on the Graphics page. We discuss them here.

7.3.1 Plot confidence bounds

You have the option to include *confidence bounds in the plot*. The methodology that is used if you *Build your own model* is based on the *State Space model*, see also Appendix B. One advantage of the state space framework is that the statistical properties of the state, which holds all components, is fully known at each time t. This in contrast to many other time series models that do not possess this property. We can therefore include confidence bounds for all models that are based on State Space models. To add confidence bounds, *first* switch *Plot conf. bounds* on, and then select the component you want to plot. If plots are requested for other types of models, the *Plot conf. bounds* option is not available. The width of the confidence interval can be controlled by changing the number in the *SE* field. Furthermore, you can choose the *Line* option, which plots two extra lines corresponding to the bounds of the confidence interval, or select *Area* to give the whole confidence area a shaded color, see also Figure 7.3.

Figure 7.3

Graphical output for Nile data and Local Level model



Type: ✓ Smoothing
☑ Y data
☑ Level
Plot conf. bounds: on
2.0 SE
Line [✓ Area]
Add lines to database: off
Select model: TSL001
Select time series: Nile

Local Level model with confidence area based on +/- 2.0 standard errors.

7.3.2 Add lines to database

This option is switched *off* by default. If switched *on*, every following line that is plotted is added to the database of the *Database* page, see also Chapter 3. This allows you to quickly store extracted signals combined with the data you originally loaded. Use the *Save database* button on the *Database* page to store your data, see also Section 3.1.2.

7.3.3 Select model / time series

The drop-down menu corresponding to *Select model* collects all the previously estimated models. This is a powerful feature since you can now compare extracted signals from different models in the same plot. This feature becomes even more powerful in comparing forecast abilities of different models, see Chapter 10. There are a couple of important things you should know because under some circumstances models cannot be compared and the drop-down menu is reset to only the most recent model. This happens if you:

- change time series (different y) on the *Database* page and estimate a model with the newly selected time series.
- change the length of the *Training* sample and estimate a model with the newly selected training sample.
- load a new database.

In all other cases, the drop-down menu will grow with every new model you estimate.

The drop-down menu corresponding to *Select time series* holds all time series that were estimated with the model currently selected by the *Select model* drop-down menu. For the TSL *Home* edition you will see only one time series at a time.

7.3.4 Plot options

Plot options gives you control over the *line transformation*, *line type*, and *line color* of the lines you want to plot. The defaults are, line transformation: *None*, line type: *Solid*, and line color: *Default color cycle*. The latter means that for every line drawn a new color is chosen according to a pre-set color order. You can always choose a color yourself from the drop-down menu or by clicking the color circle to the right of the *Line color* drop-down menu.

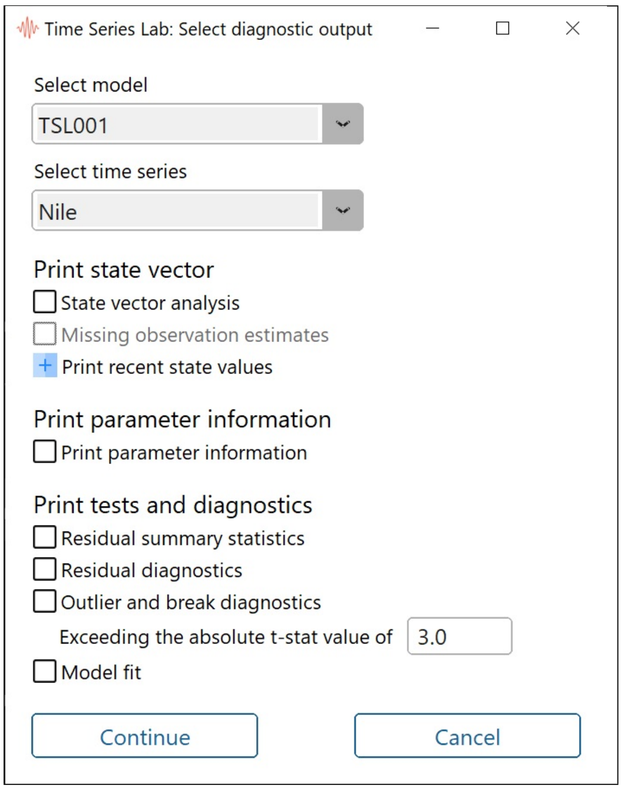
7.4 Print diagnostics

Clicking the *Print diagnostics* button in the top right corner of the *Graph* page opens the window as shown in Figure 7.4. We are presented the option to chose a model, a time series, and a selection of diagnostics, and additional output. Note that not all models have the same number of diagnostic options. Some of the options are reserved for State Space models only since they have confidence bounds at their disposal (see Section 7.3.1) which some tests are based on. All output of the diagnostic tests is printed to the *Text output* page.

7.4.1 State vector analysis

Select this option for a statistical analysis of the components of the state vector at time t_2 (last time point of training sample). TSL prints the name of the component, the value at time t_2, the standard error, the t-stat (value / standard error), and the probability corresponding to the t-stat.

Figure 7.4
TSL diagnostics window


Time Series Lab: Select diagnostic output
Select model
TSL001
Select time series
Nile
Print state vector
State vector analysis
Missing observation estimates
Print recent state values
Print parameter information
Print parameter information
Print tests and diagnostics
Residual summary statistics
Residual diagnostics
Outlier and break diagnostics
Exceeding the absolute t-stat value of
3.0
Model fit
Continue
Cancel

7.4.2 Missing observation estimates

If your time series contains missing values, select this option to give estimates for the missing values. The estimates are printed to the *Text output* page. If you want to store these values, save the Smoothed estimates via the *Save components* option, see also Section 7.5.

7.4.3 Print recent state values

Select this option to print the last X values of the state for the selected components and for *Predicting*, *Filtering*, and *Smoothing*. X is the number of recent periods which you can specify.

7.4.4 Print parameter information

Select this option to print the values of the optimized parameters. For State Space models, a column with q-values is added which are the ratios of the variance parameters. A 1.0 corresponds to the largest variance in the model.

7.4.5 Residual summary statistics

Select this option to print residual characteristics of the *standardized* residuals. TSL prints Sample size, Mean, Variance, Median, Minimum, Maximum, Skewness, and Kurtosis. For State Space models, additional residuals are available, these are *Smoothed residuals* and *Level residuals* which are used for *Outlier* and *Break* detection, see also Chapter 5.1.9 and Case study 12.3.

7.4.6 Residual diagnostics

Depending on the model specification, residuals can be assumed to come from a Normal distribution. This is for example the case with State Space models. Due to the Normal assumption, residuals can be formally tested to come from a Normal distribution. Several tests are available. There are tests that target specific parts of the Normal distribution like *Skewness* and *Kurtosis* and there are tests that combine both, for example the *Jarque-Bera* test. Other tests like *Shapiro-Wilk* and *D'Agostino's* K^2 take a different approach. TSL provides *Statistics* and *Probabilities* for each of them. In general, a *Probability* < 0.05 is considered a rejection of the null hypothesis:

$$H_0 : \text{Standardized residuals are Normally distributed.}$$

Furthermore, a *Serial correlation* test is provided to test for residual autocorrelation based on the *Ljung-Box* test. Currently the number of lags are chosen by TSL. In future versions, the lags can be set by the user. TSL provides the Lags it tested, the *Test statistic*, and the corresponding *Probability*. In general, a *Probability* < 0.05 is considered a rejection of the null hypothesis:

$$H_0 : \text{Standardized residuals are independently distributed.}$$

7.4.7 Outlier and break diagnostics

Outlier and *Break* detection, see also Chapter 5.1.9 and Case study 12.3 is based on *Auxiliary* residuals, see Durbin and Koopman (2012) p59. Outliers and breaks above a provided threshold can be printed to screen.

7.4.8 Model fit

Model fit is printed to the *Text output* page after each model estimation. This output can also be printed by selecting the *Model fit* option.

7.5 Save components

All components from the *Individual*, *Composite*, *Residuals*, and *Auxiliary* tabs on the Graph page can be saved to the file system. Click the *Save components* button in the upper right corner of the *Graph* page. Select the components you want to save or click the *Select all components* button to select all components, followed by *Save selected*.

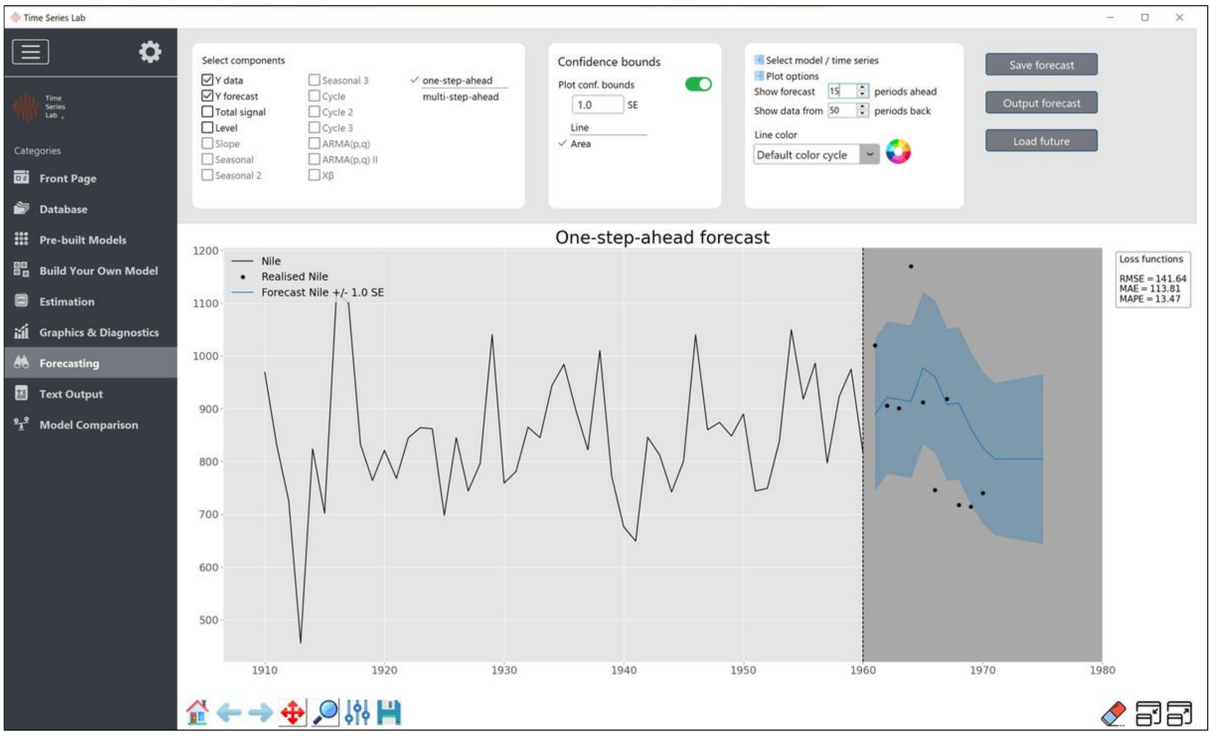
Chapter 8

Forecasting

The *Forecasting* page of TSL with the *Nile* data as illustration looks like Figure 8.1. The *Forecasting* page can be reached by using the button as shown on the left. Forecasting a time series is often of great interest for users because being able to say "something" about the future can be of great value. Forecasts need to be evaluated in some way. This is usually done by loss functions which we will discuss later in this section. A forecast comparison between different models and for different forecast horizons can be performed on the *Model comparison* page, see Chapter 10.

Figure 8.1

Forecasting page of TSL



Select components: [✓ Y data] [✓ Y forecast] Total signal, Level; [✓ one-step-ahead] multi-step-ahead

Plot conf. bounds: 1.0 SE; Line [✓ Area]

Show forecast 15 periods ahead

Show data from 50 periods back

Loss functions

RMSE = 141.64
MAE = 113.81
MAPE = 13.47

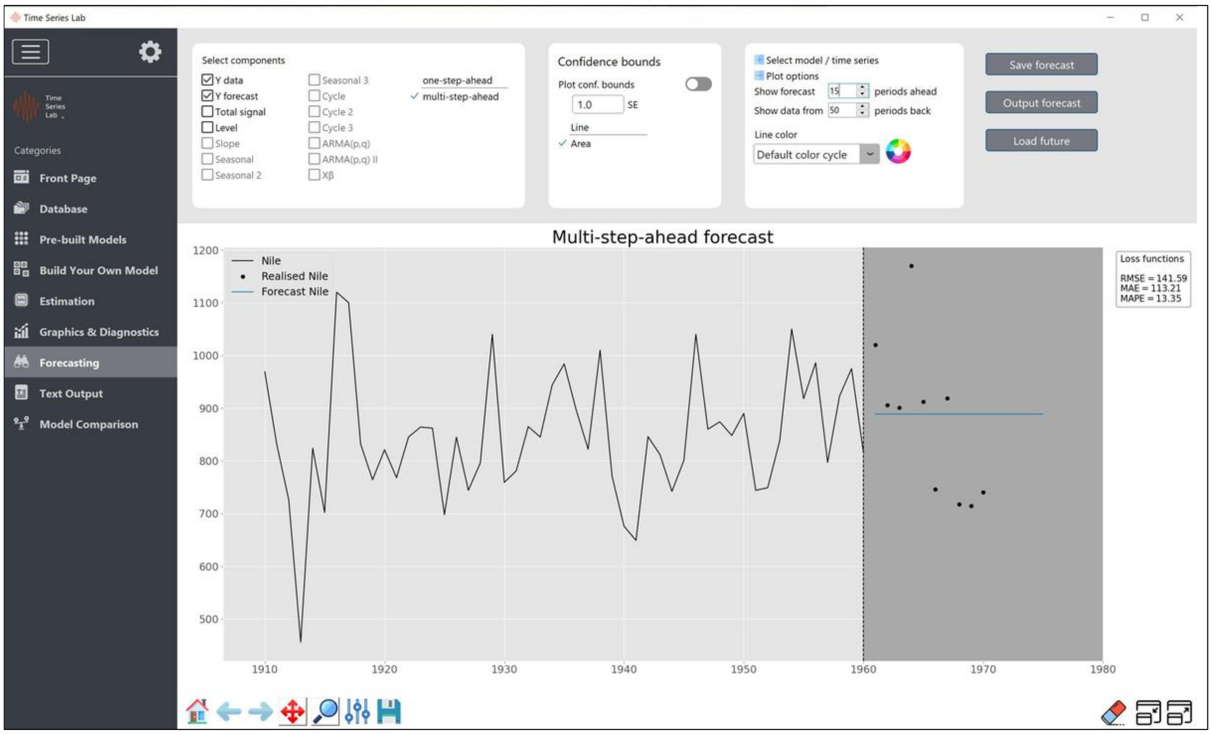
8.1 Forecast components

The available components to forecast are a subset of the components that can be plotted on the *Graphics* page with one exception, *Y forecast*. For all models, *Total signal* and *Y forecast* are exactly equal, except for their confidence intervals. Note that confidence intervals are only available for State Space models. The difference between confidence intervals for *Total signal* and *Y forecast* has to do with the extra error term that State Space models possess, see Appendix B for more background. As a result, confidence intervals for *Y forecast* are wider than for *Total signal*.

You can choose between *one-step-ahead* forecasting and *multi-step-ahead* forecasting. The difference is how many steps we forecast ahead standing at the end of the *Training* sample (time point t_2). For *one-step-ahead* forecasting, we make a forecast one step ahead, update the data and go to the next time point $(t_2 + 1)$ and the forecasting process repeats. For *multi-step-ahead* forecasting, we stay at time t_2 and forecast 1-step, 2-step, 3-step,. . .,n-step-ahead. If the *Validation* sample has size 0, meaning the *Training* sample is 100% of your data, *one-step-ahead* forecasting and *multi-step-ahead* forecasting are the same.

For the Local Level model, the *multi-step-ahead* forecasts form a straight line because we cannot *update* our model anymore with new data, see also Figure 8.2. For models with, for example, seasonal components, *multi-step-ahead* forecasts are no longer straight lines since these models have dynamics regardless if new data comes in. We see examples of this in Chapter 12.

Figure 8.2
Multi-step-ahead forecast for Local Level model



Select components
[✓] Y data
[✓] Y forecast
☐ Total signal
☐ Level
☐ Slope
☐ Seasonal
☐ Seasonal 2
☐ Seasonal 3
☐ Cycle
☐ Cycle 2
☐ Cycle 3
☐ ARMA(p,q)
☐ ARMA(p,q) II
☐ Xβ
one-step-ahead [✓ multi-step-ahead]

Confidence bounds
Plot conf. bounds
1.0 SE
Line [✓ Area]

Select model / time series
Plot options
Show forecast 15 periods ahead
Show data from 50 periods back
Line color
Default color cycle

Loss functions
RMSE = 141.59
MAE = 113.21
MAPE = 13.35

The difference between the observations in the *Training* sample and the forecasts can be quantified into a loss function. If *Y forecast* or *Total signal* are selected, three *loss functions* are added to the right side of the plot area. The loss functions are *Root Mean Squared Error* (RMSE), *Mean Absolute Error* (MAE), and *Mean Absolute Percentage Error* (MAPE). A more substantial evaluation of *Forecasting* performance is provided on the *Model comparison* page, see also Chapter 10.

8.2 Additional options

Some more plot options are available on the Forecast page. We discuss them here.

8.2.1 Plot confidence bounds

See Section 7.3.1

8.2.2 Select model / time series

See Section 7.3.3

8.2.3 Plot options

Click the spinboxes to expand or contract the part of the Training and Validation sample that is displayed on screen. For line colors see Section 7.3.4.

8.3 Load future

The *load future* option is for users who have knowledge about the future that they want to incorporate in the model forecasts. This is mainly used for explanatory variables but can be used for the y variable as well. If you make multi-step-ahead forecasts and explanatory variables are included in the model, the explanatory variables need to be forecasted as well. This is the case, for example, on the model comparison page where forecasts are made up to 50 steps ahead. TSL forecasts the explanatory variables with the method selected as described in Section 5.1.8.3. If you do not want TSL to forecast the explanatory variables you can specify them yourselves by loading a dataset with the *load future* option.

Important: The loaded future data is matches with the main data set by means of comparing column names. Only if column names of the loaded future data matches the ones (can also be a subset of the column names) in the main data set, is the future data taken into account.

8.4 Save forecast

See Section 7.5

8.5 Output forecast

See Section 7.5

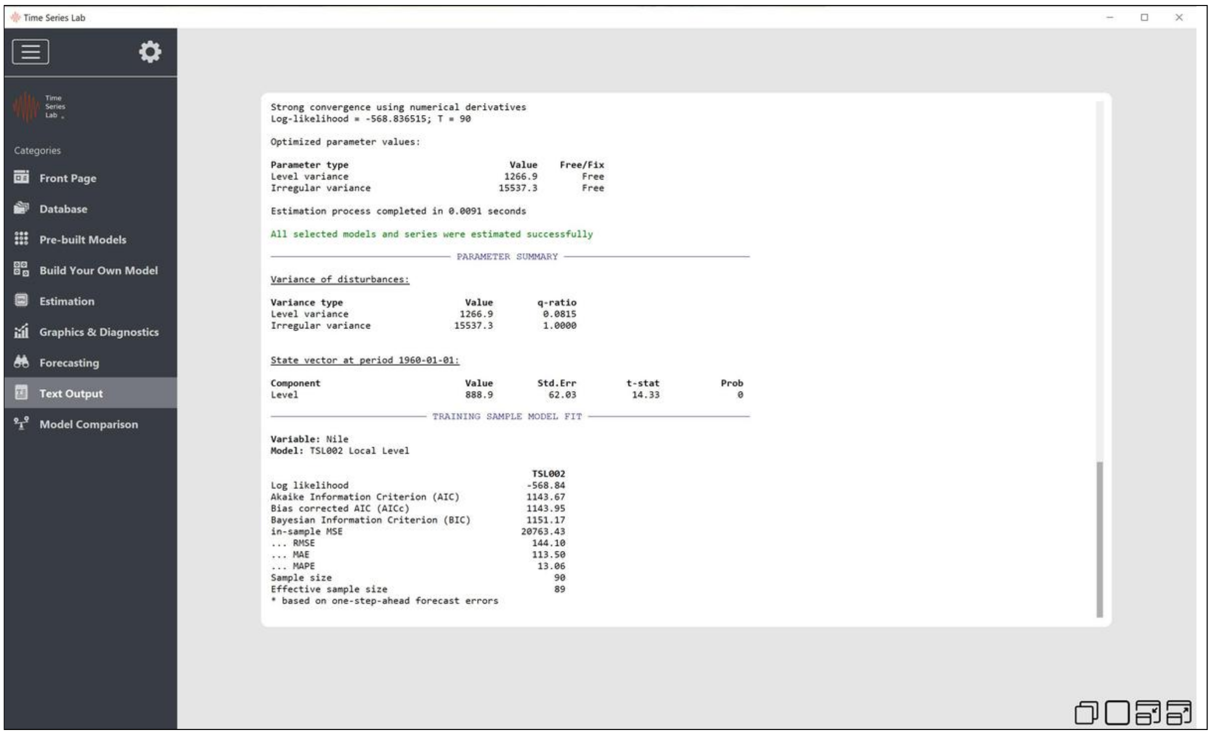
Chapter 9

Text output

Written output is provided on the *Text output* page of TSL which can be reached by using the button as displayed on the left. Written output can be anything from optimization results, warnings, to output of diagnostic tests. An example of text output printed by TSL during and after the estimation of the Local Level model and the Nile data is shown in Figure 9.1.

Figure 9.1

Text output for the Local Level model



```
Strong convergence using numerical derivatives
Log-likelihood = -568.836515; T = 90

Optimized parameter values:

Parameter type                         Value    Free/Fix
Level variance                        1266.9        Free
Irregular variance                   15537.3        Free

Estimation process completed in 0.0091 seconds

All selected models and series were estimated successfully

──────────────────────────── PARAMETER SUMMARY ────────────────────────────

Variance of disturbances:

Variance type                  Value       q-ratio
Level variance                1266.9        0.0815
Irregular variance           15537.3        1.0000


State vector at period 1960-01-01:

Component                      Value       Std.Err        t-stat          Prob
Level                          888.9         62.03         14.33             0
────────────────────────── TRAINING SAMPLE MODEL FIT ──────────────────────────

Variable: Nile
Model: TSL002 Local Level

                                          TSL002
Log likelihood                           -568.84
Akaike Information Criterion (AIC)       1143.67
Bias corrected AIC (AICc)                1143.95
Bayesian Information Criterion (BIC)     1151.17
in-sample MSE                           20763.43
... RMSE                                  144.10
... MAE                                   113.50
... MAPE                                   13.06
Sample size                                   90
Effective sample size                         89
* based on one-step-ahead forecast errors
```

Just like plot areas, the Text output are can be undocked with the buttons in the bottom right of the screen. This way the text area can be placed outside of the main TSL window on for example a different monitor.

Chapter 10

Model comparison

The *Model comparison* page of TSL can be reached by using the button as displayed on the left. It is an import page because forecasting performance of different models can be compared with each other. For each model, the page shows forecast errors up to $h = 50$ time periods ahead. This is useful to see if a model performs well on, for example, shorter or longer forecast horizons.

Important: The *Model comparison* button is only visible if you provided a Validation sample. If you would have chosen to use all data for the Training sample ($t_2 = T$), loss calculations cannot be made since forecasts cannot be compared with actual values in the Validation sample.

10.1 Loss calculation procedure

We compare models based on several *Loss functions*, for example, the *Root Mean Squared Error* (RMSE). For the RMSE we have

$$\text{RMSE}_h = \sqrt{\frac{1}{p-h+1} \sum_{t=t_2}^{t_2+p-h} (y_{t+h} - \hat{y}_{t+h|t})^2}, \tag{10.1}$$

with p the length of the Validation sample, t_2 the length of the Training sample, h the length of the forecast horizon, and $\hat{y}_{t+h|t}$ the forecast for observation y_{t+h} given the information up to time t.

Assume we estimated a model for a time series with length $T = 400$ and we took a ratio of $80\%/20\%$ for Training and Validation sample. This means we have a Training sample with length $t_2 = 320$. By repeatedly making 1-step-ahead, 2-step-ahead, ..., 50-step-ahead forecasts, according to Equation (10.1), we make a total of 80 $\times$ 1-step-ahead forecasts, 79 $\times$ 2-step-ahead forecasts, 78 $\times$ 3-step-ahead forecasts, ..., 31 $\times$ 50-step-ahead forecasts. By obtaining many forecasts we have a robust way of comparing models with each other based on forecasting performance.

10.2 Start loss calculation

Select the model and time series from the drop-down menus for which you want to calculate forecast losses. Click the *Start loss calculation* button in the top right corner of the screen to start the calculation. When the loss calculation is finished, the selected model shows up in the *User defined models* section in the top left of the screen. Select the newly appeared check box and the corresponding losses appear in the plot area for the loss functions selected under *Loss functions*. Repeat the procedure for other models and/or compare the results with losses from the *Naive forecast methods*.

- Last observation. This model always takes the last observation from the Training sample as forecast for the forecast horizons.
- Average of last X observations. This model takes the average of the last X observation from the Training sample as forecast for the forecast horizons.
- Last observation X periods back. This model takes a block of last X observations and uses that block repeatedly as forecasts till the end of the forecast horizon is reached. For example with $X = 4$ we obtain a block of observations $y_{T-3}, y_{T-2}, y_{T-1}, y_T$. The forecast for $y_{T+1}, y_{T+5}, y_{T+9}, \ldots$ is y_{T-3}, the forecast for $y_{T+2}, y_{T+6}, y_{T+10}, \ldots$ is y_{T-2}, etc.

Finally, print the selected model loss combinations to the *Text output* page by clicking the *Print model comparison* button.

Chapter 11

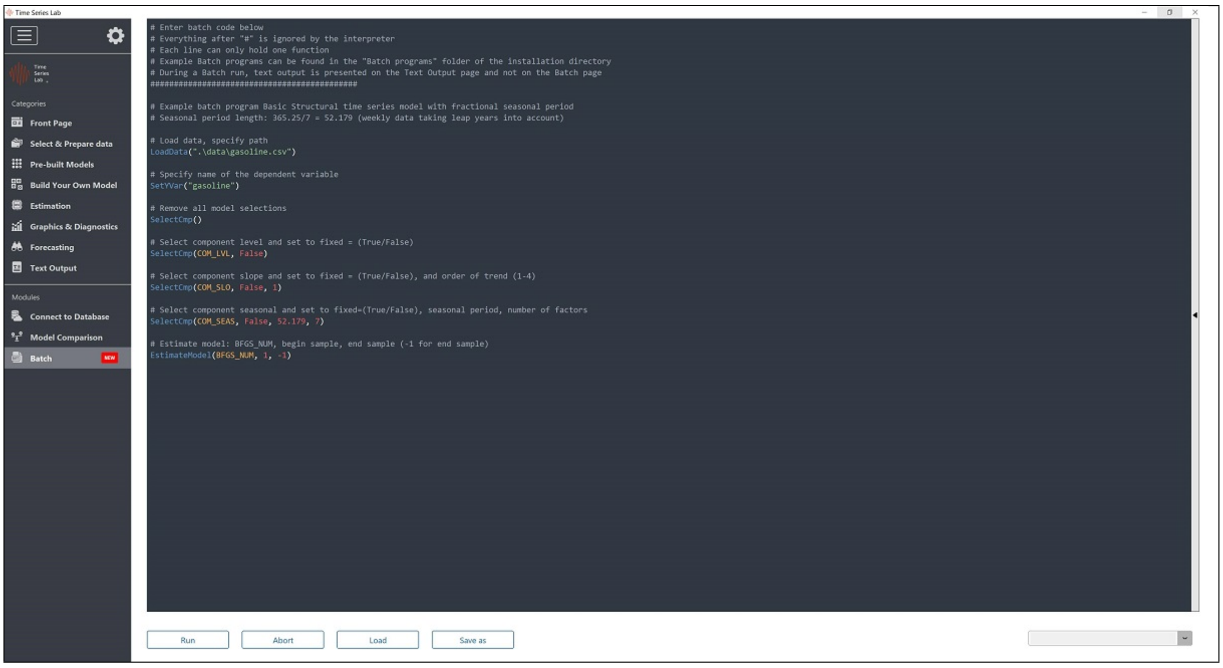
Batch module

The *Batch module* of TSL is designed to program TSL rather than go through all modelling steps and menus manually. This way, TSL can be used to automate the modelling and forecasting of time series. A *Batch program* needs to be written only once and can then be used each time again to assign tasks to TSL. It also allows you to schedule TSL, for example to run the Batch program every morning at 07:00 or to run the Batch program repeatedly every 60 seconds.

Example Batch programs can be found in the "batch_programs" folder of the installation directory

Figure 11.1

Batch module with example program



```
# Enter batch code below
# Everything after "#" is ignored by the interpreter
# Each line can only hold one function
# Example Batch programs can be found in the "Batch programs" folder of the installation directory
# During a Batch run, text output is presented on the Text Output page and not on the Batch page
############################################

# Example batch program Basic Structural time series model with fractional seasonal period
# Seasonal period length: 365.25/7 = 52.179 (weekly data taking leap years into account)

# Load data, specify path
LoadData(".\data\gasoline.csv")

# Specify name of the dependent variable
SetYVar("gasoline")

# Remove all model selections
SelectCmp()

# Select component level and set to fixed = (True/False)
SelectCmp(COM_LVL, False)

# Select component slope and set to fixed = (True/False), and order of trend (1-4)
SelectCmp(COM_SLO, False, 1)

# Select component seasonal and set to fixed=(True/False), seasonal period, number of factors
SelectCmp(COM_SEAS, False, 52.179, 7)

# Estimate model: BFGS_NUM, begin sample, end sample (-1 for end sample)
EstimateModel(BFGS_NUM, 1, -1)
```

Chapter 12

Case studies

12.1 Nile data

In this first case study we illustrate the fundamentals of TSL using observations from the river Nile. The data set consists of a series of readings of the annual flow volume at Aswan from 1871 to 1970. The Nile dataset is part of any TSL installer file and can be found in the data folder located in the install folder of TSL. Many time series concepts can be explained by the Nile time series alone.

12.1.1 Loading data

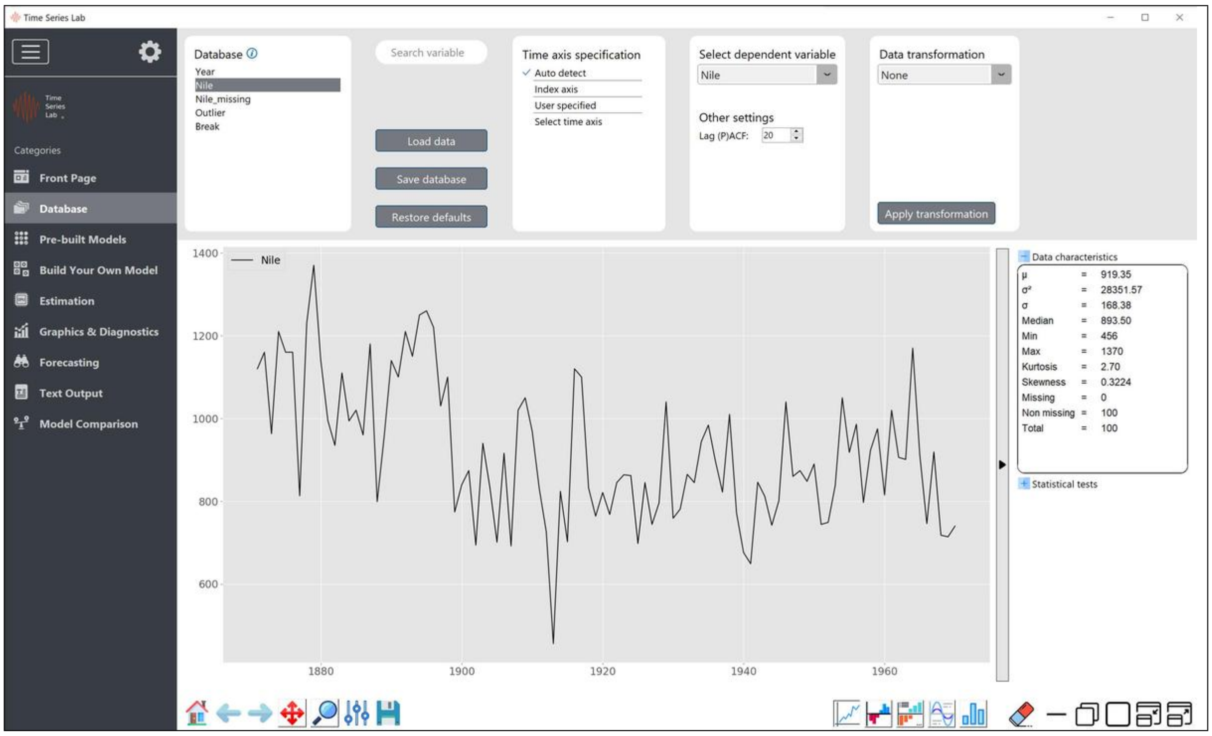
Let's start the modelling process. First go to the *Database* page of TSL by clicking the *Database* button. On this page we load, visually inspect, and prepare our data for the modelling process. The data set is loaded and selected from the file system by pressing the *Load data* button or by selecting Load data from the File menu. Locate the file *Nile.csv* in the *data* folder of the TSL install folder.

Important: The data set should be in column format with headers. The format of the data should be *.xls(x), or *.csv, *.txt with commas as field separation. The program (purposely) does not sort the data which means that the data should be in the correct time series order before loading it into the program.

After loading, click on the name *Nile* in the database field. If you click the *arrow bar* at the right side of the screen, a new area unfolds which shows us *Data characteristics* of the selected time series. It shows that the Nile time series has a length of $T = 100$ observations with 0 missing values, among other characteristics. The TSL window should look like Figure 12.1.

The highlighted variable *Nile* also appears in the *Select dependent variable* drop-down menu. This is the so-called y-variable of the time series equation and it is the time series variable of interest, i.e. the time series variable you want to model, analyse, and forecast.

Figure 12.1
Data inspection and preparation page



Database
Year
Nile
Nile_missing
Outlier
Break

Time axis specification
✓ Auto detect
Index axis
User specified
Select time axis

Select dependent variable: Nile

Other settings
Lag (P)ACF: 20

Data transformation: None

Data characteristics

μ	=	919.35
σ²	=	28351.57
σ	=	168.38
Median	=	893.50
Min	=	456
Max	=	1370
Kurtosis	=	2.70
Skewness	=	0.3224
Missing	=	0
Non missing	=	100
Total	=	100

Optionally, a time series axis can be specified. The program's algorithm tries to auto-detect the time axis specification (e.g. annual data, daily data) from the first column of the data set. In the case of the Nile data illustration, it finds an annual time axis specification. If auto-detection fails, the program selects the Index axis option which is just a number for each observation, 1,2,3,..., see also Chapter 3.1.3.

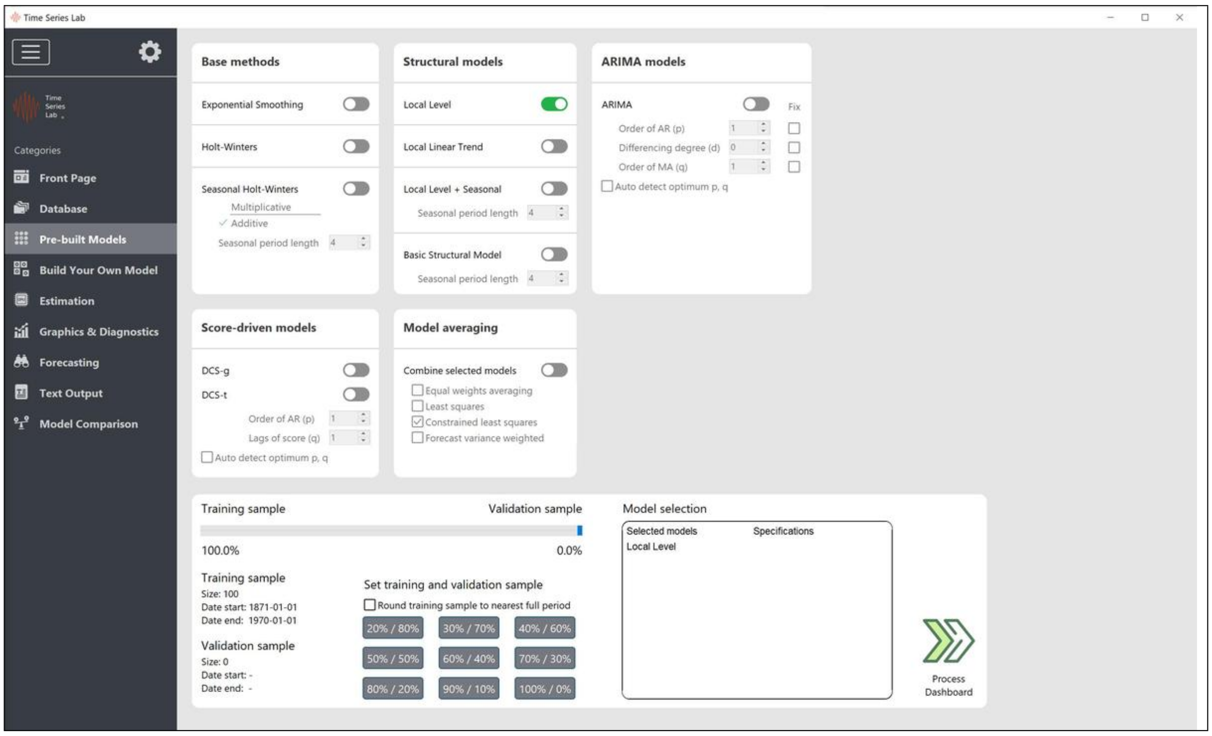
12.1.2 Pre-built models

Click on the *Pre-built models* button in the button bar at the left of your screen. Switch on the Local Level model. Make sure this is the only selected model, see also the *Model selection* summary in the blue pane in the bottom of the screen. Select an 100%/0% ratio for *Training* and *Validation* sample. The settings are shown in Figure 12.2. Click the *Process Dashboard* button which is the green arrow located at the bottom right of your screen. After pressing this button, two things happen:

- TSL estimates the selected models and prints results to the *Text output* page. The results are: progress results from the optimizer and *model fit* of the selected models.
- Once processing of the selected models is complete, TSL plots the information it found and shows the *Graphics page*.

Figure 12.2

Model selection page with Local Level model selected



Training sample 100.0% — Validation sample 0.0%

Training sample
Size: 100
Date start: 1871-01-01
Date end: 1970-01-01

Validation sample
Size: 0
Date start: -
Date end: -

Selected models: Local Level

12.1.3 Graphical output

After processing the selected models, TSL automatically takes you to the Graphics page. Components, or combinations of components, can be easily plotted and removed from the plot by checking or unchecking the tickboxes in the top left corner of the page. You can add subplots as well to create a grid of plots.

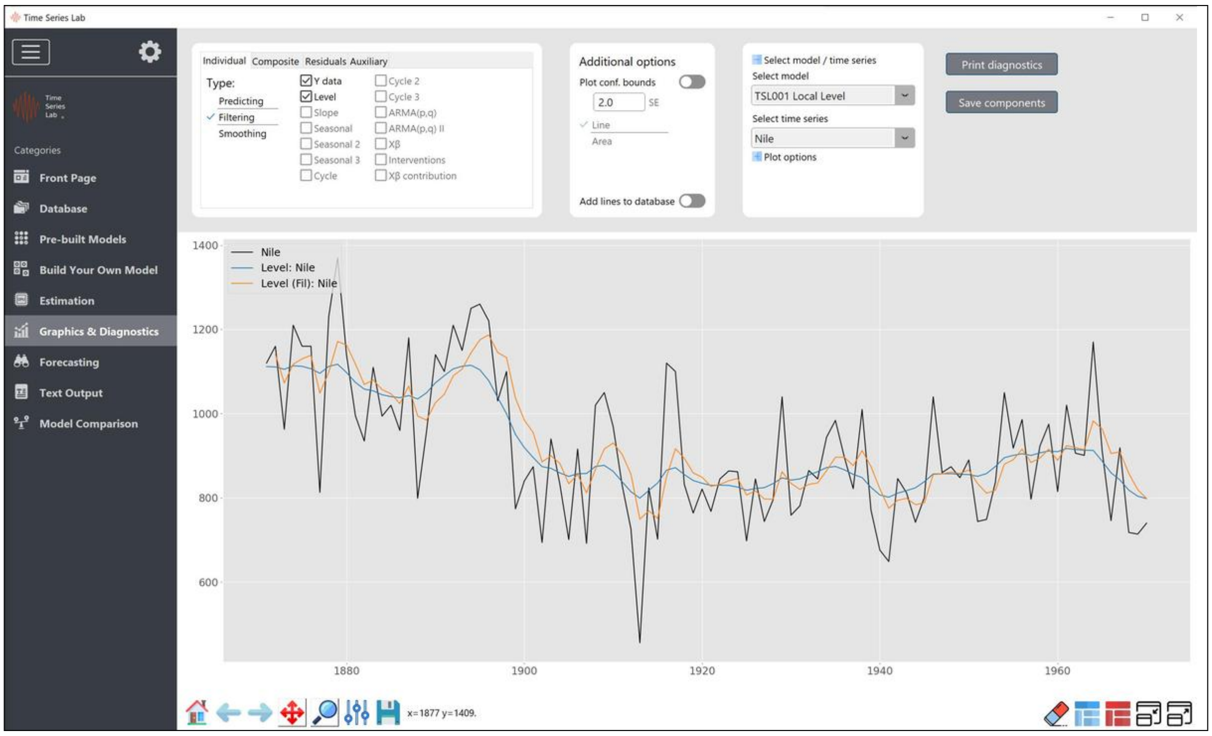
Click on a (sub)plot to activate it. Notice that by clicking on a subplot, the checkboxes in the top left of the window correspond to the current selection of lines in the subplot. If not all checkbox settings correspond with the lines in the subplot, switch tabs to show the rest of the selection.

To see what is meant by the text above: switch from Smoothing to Filtering. You now see that the Level checkbox is **unchecked** because the level that is currently plotted corresponds to the Smoothed level and not the Filtered level. The reason the Smoothed level in the plot does not automatically switch to a Filtered level on changing is that you sometimes want to compare Smoothed, Filtered, and Predicted components in one plot. If you click on level, the resulting graph should look like the one in Figure 12.3.

For State Space models, confidence intervals can be included in the plot as well. A major benefit of State Space models is that the error bounds can easily be obtained. More on State Space models in Appendix B. You can also choose between Predicting, Filtering, and Smoothing by changing the *type* in the top left corner. The difference between Smoothed, Filtered, and Predicted components has to do with the subset of the data used to determine the statistical properties of the components, i.e. the data up to time $t-1$ (forecasting), the data up to time t (filtering) or the whole data set (smoothing), see also Appendix B.

Figure 12.3

Time Series Lab Graph page



12.1.4 Missing data

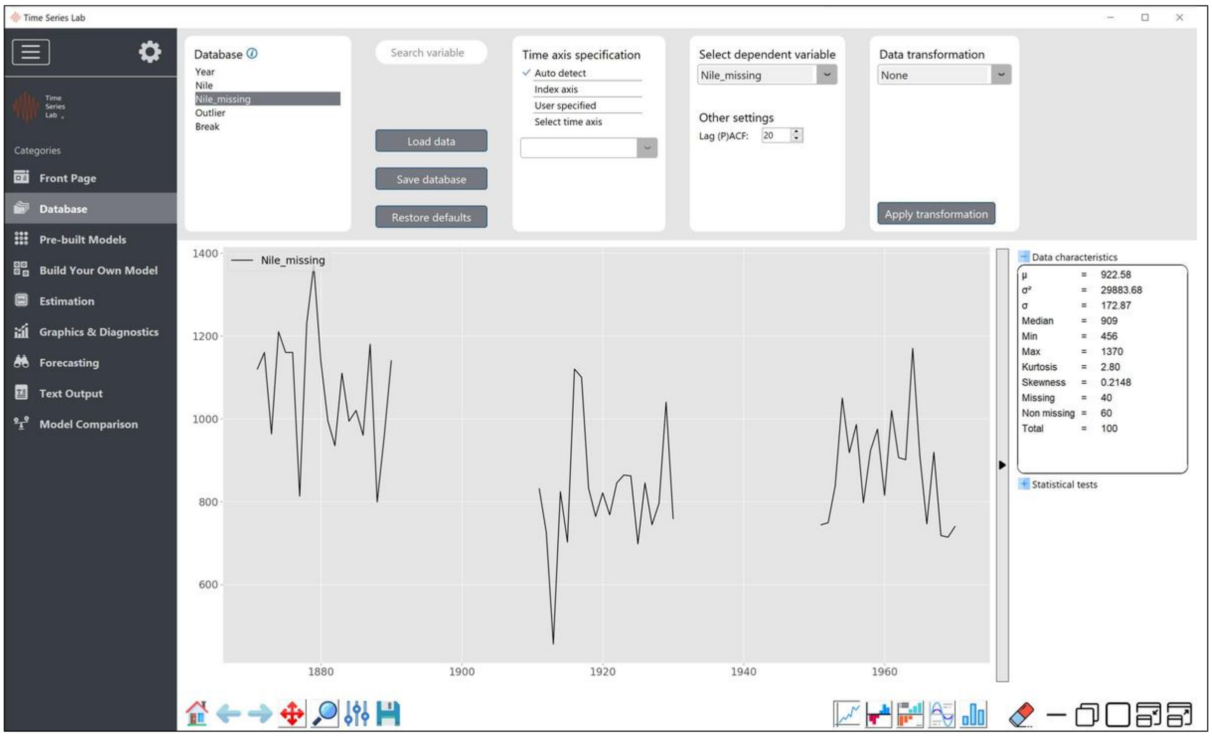
We continue this Case study with a version of the Nile data with *missing values* to illustrate one of the many advantages of using TSL, namely the capability of **easily** handling missing values. Missing values in time series can occur due a variety of reasons and for some time series algorithms it is problematic.

Missing data can cause problems for some time series algorithms. These algorithms often revert to deleting the missing values or the missing values are filled with certain values. In TSL there is no need to rely on such drastic measures. Missing values are part of time series analysis and they should be handled in a correct manner.

Go back to the *Database* page of TSL and select the *Nile_missing* time series by clicking on the name. You see that the *Data characteristics* are updated by selecting the new time series. It shows us 40 missing values, among other characteristics. The TSL window should now look like Figure 12.4.

Figure 12.4

Data inspection and preparation page



Database: Year, Nile, Nile_missing, Outlier, Break

Time axis specification: ✓ Auto detect, Index axis, User specified, Select time axis

Select dependent variable: Nile_missing

Other settings: Lag (P)ACF: 20

Data transformation: None

Data characteristics

μ	=	922.58
σ^2	=	29883.68
σ	=	172.87
Median	=	909
Min	=	456
Max	=	1370
Kurtosis	=	2.80
Skewness	=	0.2148
Missing	=	40
Non missing	=	60
Total	=	100

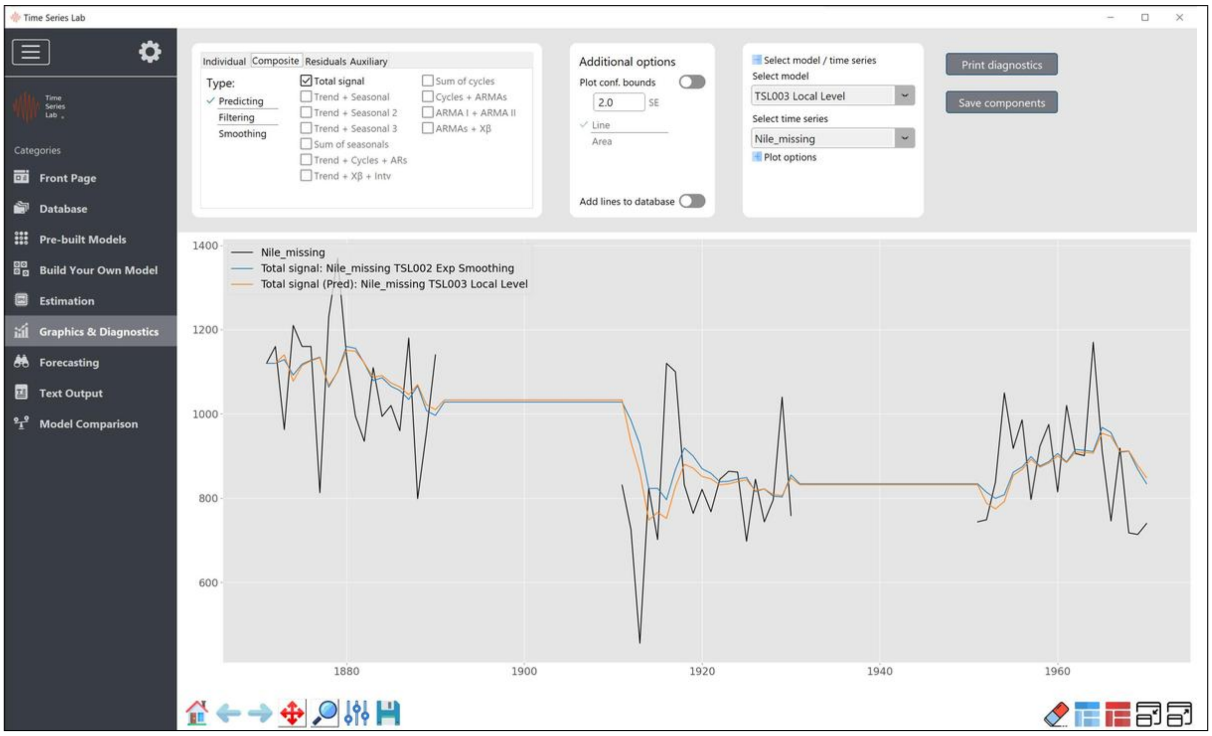
We will estimate and compare two models with each other. Click on the *Pre-built models* button in the button bar at the left of your screen and switch on, the models *Exponential Smoothing* and *Local Level*. Make sure these are the only selected models, see also the *Model selection* summary in the blue pane in the bottom of the screen. Select an $100\%/0\%$ ratio for *Training* and *Validation sample* and click the *Process Dashboard* button which is the green arrow located at the bottom right of your screen.

12.1.5 Comparing results

Go to the *Graphics and diagnostics* page and click the *Clear all* button (eraser icon, bottom right) to start with a clean graph window. From the *Individual* tab select Y data to plot the Nile_missing time series. From the drop-down menu select the Exp Smoothing model and plot the *Total signal* from the *Composite* tab. Next, from the drop-down menu select the Local Level model and make sure the *Type* in the top left corner says **Predicting**, followed by plotting the *Total signal* from the *Composite* tab. The resulting graph should look the one in Figure 12.5. We see that the Local Level model reacts stronger to changes in the time series after missing values periods.

Figure 12.5

TSL Graph page



We can also see the difference in model fit expressed in numbers. Go to the *Text output* page where at the end of the estimation, model fit of the selected models is summarized. Looking at *in-sample MSE* we see that the loss of the Local Level model is lower.

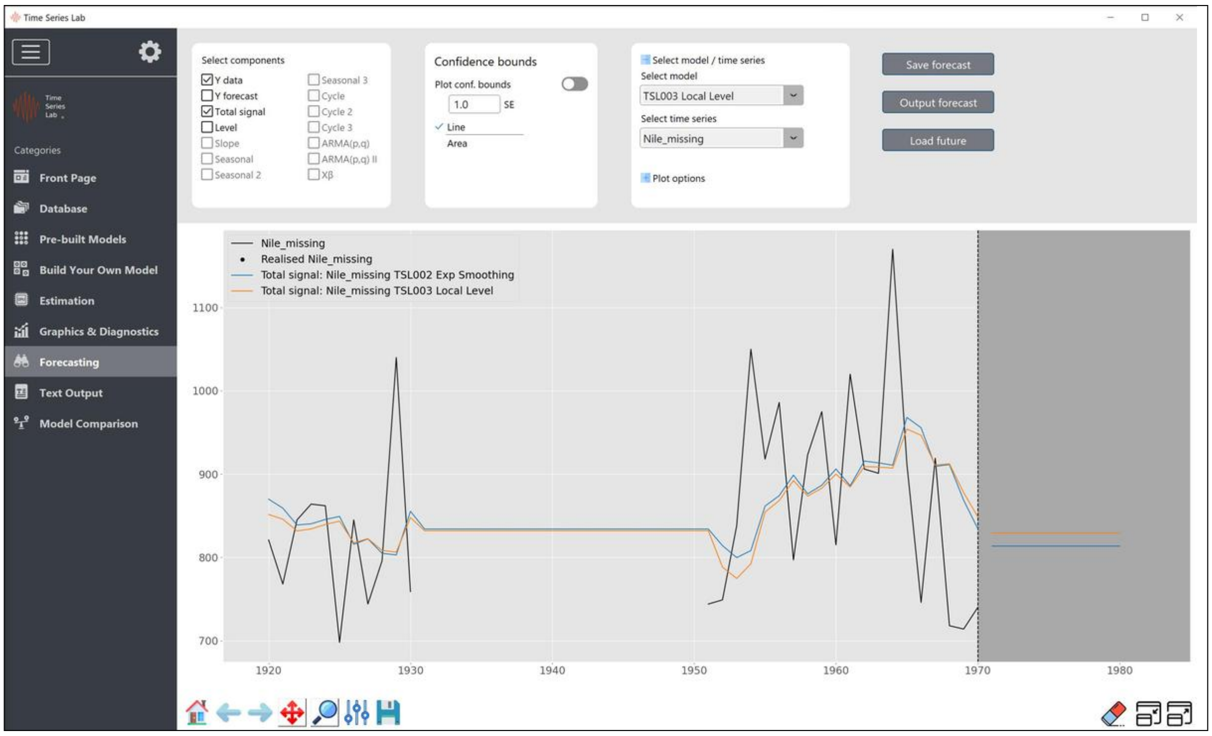
```
Variable: Nile_missing
Model(s):
TSL005 Exp Smoothing
TSL006 Local Level


                                            TSL005       TSL006
Log likelihood                                   -      -380.01
Akaike Information Criterion (AIC)               -       766.02
Bias corrected AIC (AICc)                        -       766.44
Bayesian Information Criterion (BIC)             -       772.30
in-sample MSE                             23735.33     23069.80
... RMSE                                    154.06       151.89
... MAE                                     119.86       118.60
... MAPE                                     14.06        13.70
Sample size                                    100          100
Effective sample size                           99           99
* based on one-step-ahead forecast errors
```

If you want to compare models and conclude something like "model A is better than model B", it is important to note that only looking at in-sample (Training sample) model fit can be misleading. It is often a good idea to take *forecast performance* into account as well. If model A performs better on both model fit and *forecast performance*, it is a good indication of model A being preferred over model B. We see examples of comparing forecast performance in other Case studies.

The forecasts of both our models can be visually inspected on the *Forecasting* page. Figure 12.6 plots the forecasts of both model in one graph. Since no new data is coming in, the forecasts are just straight lines but the level (height) of the lines differ per model. Note that the local level model is not just a theoretical model, it has practical value as well. For example for inflation modelling, the local level model is a strong contender. We will see more complex forecasting patterns in other case studies.

Figure 12.6

TSL forecast page



Select components

[✓] Y data
[ ] Y forecast
[✓] Total signal
[ ] Level
[ ] Slope
[ ] Seasonal
[ ] Seasonal 2
[ ] Seasonal 3
[ ] Cycle
[ ] Cycle 2
[ ] Cycle 3
[ ] ARMA(p,q)
[ ] ARMA(p,q) II
[ ] Xβ

Confidence bounds

Plot conf. bounds: 1.0 SE

✓ Line
Area

Select model / time series

Select model: TSL003 Local Level

Select time series: Nile_missing

12.1.6 Outliers and Structural breaks

Intervention analysis, also called *anomaly detection*, is an important part of time series analysis. We distinguish two types of anomalies, *Outliers* and *Structural breaks*. For example, early warning systems rely on outlier and break detection. Could a catastrophic event have been seen in advance? Take for example sensor readings from an important piece of heavy machinery. The breaking down of this machine would cost a company a lot of money. If anomalies were detected in the sensor reading, preventive maintenance might have saved the company from a break-down of the machine.

Intervention variables are dummy (or indicator) variables which are used to take account of outlying observations and structural breaks. These data irregularities are usually thought of as arising from a specific event, for example a strike in the case of an outlier or a change in policy in the case of a structural break. An outlier can be thought of as an unusually large value of the irregular disturbance at a particular time. It can be captured by an impulse intervention variable which takes the value one at the time of the outlier and zero elsewhere. A structural break in which the level of the series shifts up or down is modelled by a step intervention variable which is zero before the event and one after. Alternatively it can be modelled in exactly the same way by adding an outlying intervention to the level equation. In other words the break is identified with an unusually large value of the level disturbance.

TSL is able to propose a set of potential outliers and structural breaks for time series. It is an effective multi-step procedure based on the auxiliary residuals, see also Harvey and Koopman (1992) for details. First the selected model is estimated and the diagnostics are investigated. Then a first (larger) set of potential outliers and trend breaks are selected from the auxiliary residuals. After re-estimation of the model, only those interventions survive that are sufficiently significant. After the automatic selection, the results are reported. All considered outliers and breaks are kept in the intervention dialog and they can be deleted from the model or added to the model.

The Nile time series has some interesting features with regard to Intervention analysis. To see this, go back to the *Database* page and select the Nile time series again without missing values. Next, go to the *Build your own model* page and select a time-varying level and time-varying slope. These two model components correspond to a model with the name *Local Linear Trend* model. On top of that, select *Intervention variables* with the *automatic setting*. Next, go to the *Estimation* page, make sure the sample starts at $t = 1$ and ends at $t = 100$ and click the green *Estimate* button. Once TSL is done estimating, you should see the graph as presented in Figure 12.7.

We see from Figure 12.7 that TSL finds a structural break and an outlier. We can also inspect these in more detail by looking at the *Text output* page where we see

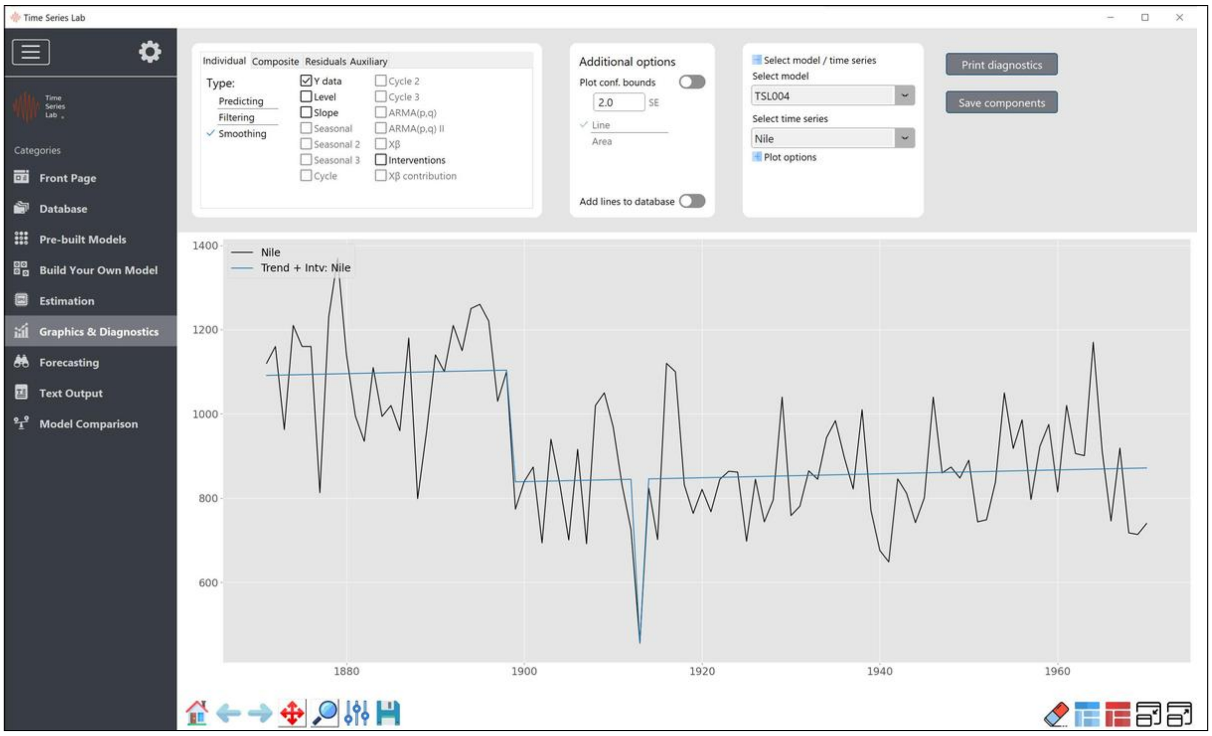
```
Intervention coefficients:

Beta                              Value      Std.Err       t-stat         Prob
beta_outlier_1913-01-01          -389.4       123.92       -3.143       0.0022
beta_break_1899-01-01            -265.5        43.67       -6.079   2.4458e-08
```

TSL finds the location of the structural break at 1899 which is very plausible since the year 1899 corresponds to the building of a dam at Aswan.

Interestingly, the addition of the outlier and structural break remove certain dynamics from the data which we can see from the straight lines in the graph which are the result of the (close to) zero variances from the Level and Slope component.

Figure 12.7

Graph page with structural break in Nile data



Select model: TSL004

Select time series: Nile

Plot conf. bounds: 2.0 SE

Type: ✓ Smoothing; ✓ Y data

12.1.7 Further exploration

- On the graph page, plot the Autocorrelation Function (ACF) of the *Predicted standardized residuals* for the Local Level model. Are all plotted lags within the confidence bounds?
- Performing diagnostic tests can be done via the *Print diagnostics* button located on the Graph page. Can you print the *Residual diagnostics* for the Exponential Smoothing model? Are all *Probabilities* for the *Normality* test above 0.05? See also Section 7.4.6.
- Outliers and structural breaks can be added (and removed) manually to (from) the model by selecting the *Manual* option of the Intervention variables. Estimate a Local Level model with only the structural break.

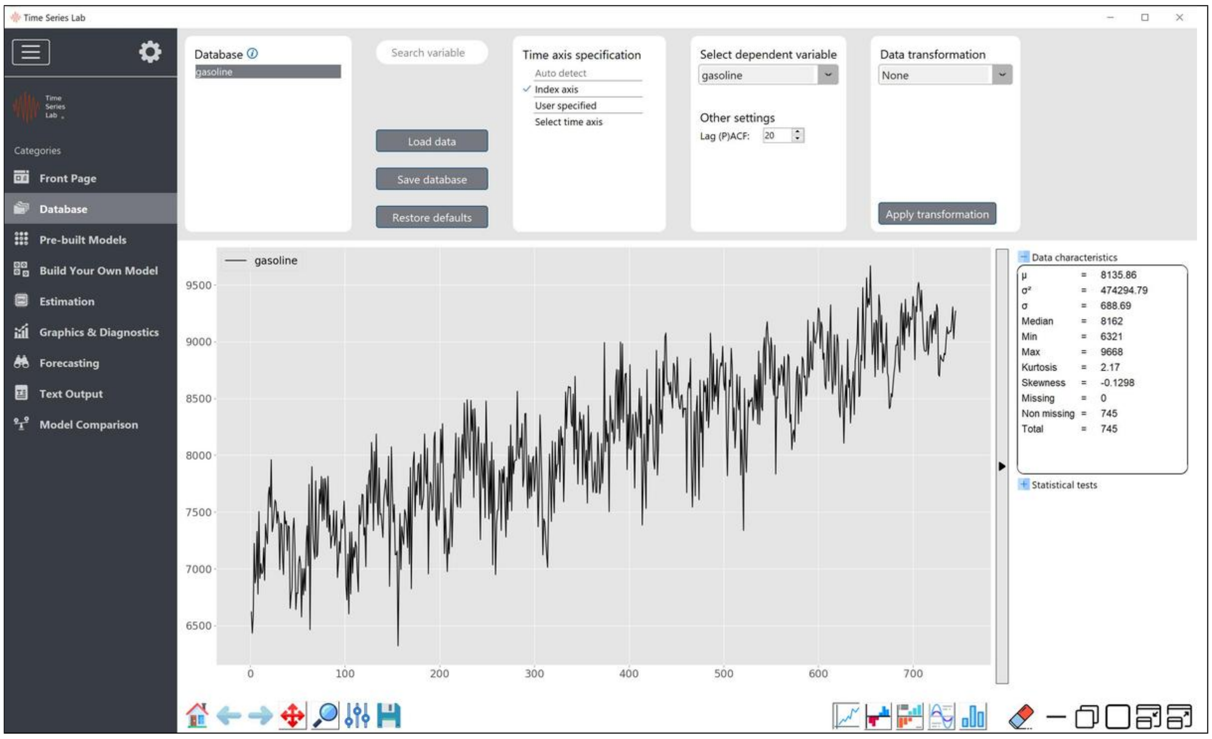
12.2 Gasoline consumption

The data for this case study is weekly data on US finished motor gasoline product supplied (in thousands of barrels per day) from February 1991 to May 2005. It is part of the R package fpp2 and available from the EIA website. It is also bundled with the installation file of TSL. The dataset is used in the *TBATS* paper of De Livera et al. (2011). Furthermore, the dataset is analysed by R.J. Hyndman on his blog.

In Figure 12.8, the gasoline dataset is loaded into TSL and plotted. The upward trend and seasonality pattern is clearly visible in the data. The Data characteristics area shows $T = 745$ observations.

Information: On the *Database* page, you can copy the contents of the *Data characteristics* pane to the clipboard by right-mouse clicking the area and selecting Copy contents or by selecting the text and clicking Ctrl-c.

Figure 12.8
TSL Database page with Gasoline dataset loaded



Database
gasoline

Time axis specification
Auto detect
✓ Index axis
User specified
Select time axis

Select dependent variable
gasoline

Other settings
Lag (P)ACF: 20

Data transformation
None

Data characteristics

μ	=	8135.86
σ²	=	474294.79
σ	=	688.69
Median	=	8162
Min	=	6321
Max	=	9668
Kurtosis	=	2.17
Skewness	=	-0.1298
Missing	=	0
Non missing	=	745
Total	=	745

12.2.1 Local Linear Trend model

The seasonal pattern of this time series is important but for illustrative purposes, we start our analysis without a seasonal component and select the *Local Linear Trend model* which is a model with a trend component but no seasonal component. Select the *Local Linear Trend model* model on the *Pre-built models* page. Alternatively, you can go to the *Build your own model page* and select a time-varying level and a time-varying slope.

Our time series consist of a total of 745 observations (February 1991 to May 2005). For this case study we select the first 484 observations as Training sample and leaving 261 observations as Validation sample. Drag (and/or click) the sample bar on the *Pre-built models* page to set a Training sample of size 484, or alternatively, set the *start* and *end* of the Training sample to 1 and 484 on the *Estimation* page.

Click the *Process Dashboard* button on the *Pre-built models* page or the *Estimate* button on the *Estimation* page. TSL estimates the model and if you go to the *Text output* page you see a *green* colored message informing us that:

All selected models and series were estimated successfully

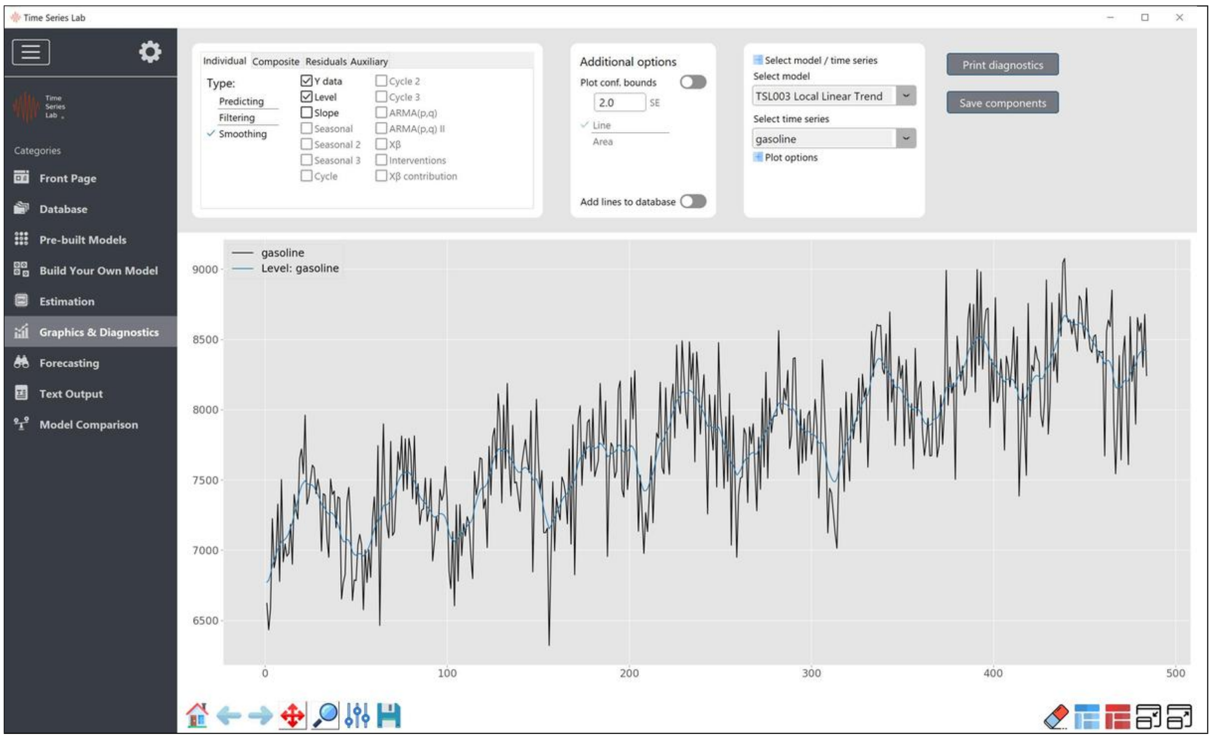
Furthermore, at the bottom of the Text output page we find the *Model fit*. For the current model this is:

```
Variable: gasoline
Model: TSL003 Local Linear Trend
                                               TSL003
Log likelihood                              -3485.355
Akaike Information Criterion (AIC)           6978.710
Bias corrected AIC (AICc)                    6978.794
Bayesian Information Criterion (BIC)         6995.439
in-sample MSE                              1.1177e+05
... RMSE                                      334.317
... MAE                                       268.147
... MAPE                                        3.480
Sample size                                       484
Effective sample size                             482
* based on one-step-ahead forecast errors
```

We report these numbers here to show the improvement of adding a seasonal component later. The graphical output of the current model is shown in Figure 12.9. With a smoothed level through the data, the seasonal pattern is even better visible. The triangular pattern in the level will appear later as well when we plot the forecasting performance of the model.

Let's assess the forecast performance of the model by going to the *Model comparison* page. This page can be viewed by clicking the *Model comparison* button in the button bar on the left of your screen. Note that this button is only visible when a Validation sample is specified. Click on the green *Start loss calculation* button in the top right of the window. Under *User defined models*, a new check-button appears which you should tick. The resulting TSL screen is shown in Figure 12.10. The pyramid shaped loss line in Figure 12.10 can be explained by the fact that a *forecast* from the Local Linear Trend model is a *straight line*

Figure 12.9
Graph page of TSL with Gasoline dataset



that is upward sloping for our data set. The forecasts do not take into account the seasonal pattern of the data so when the data is at the highest or lowest point in the seasonal cycle, the loss is the highest. Let's verify this. Go to the *Forecasting* page and select *multi-step-ahead* in the top left corner. Navigate to *Plot options* and Show forecast 150 periods ahead. The resulting window should look the one presented in Figure 12.11.

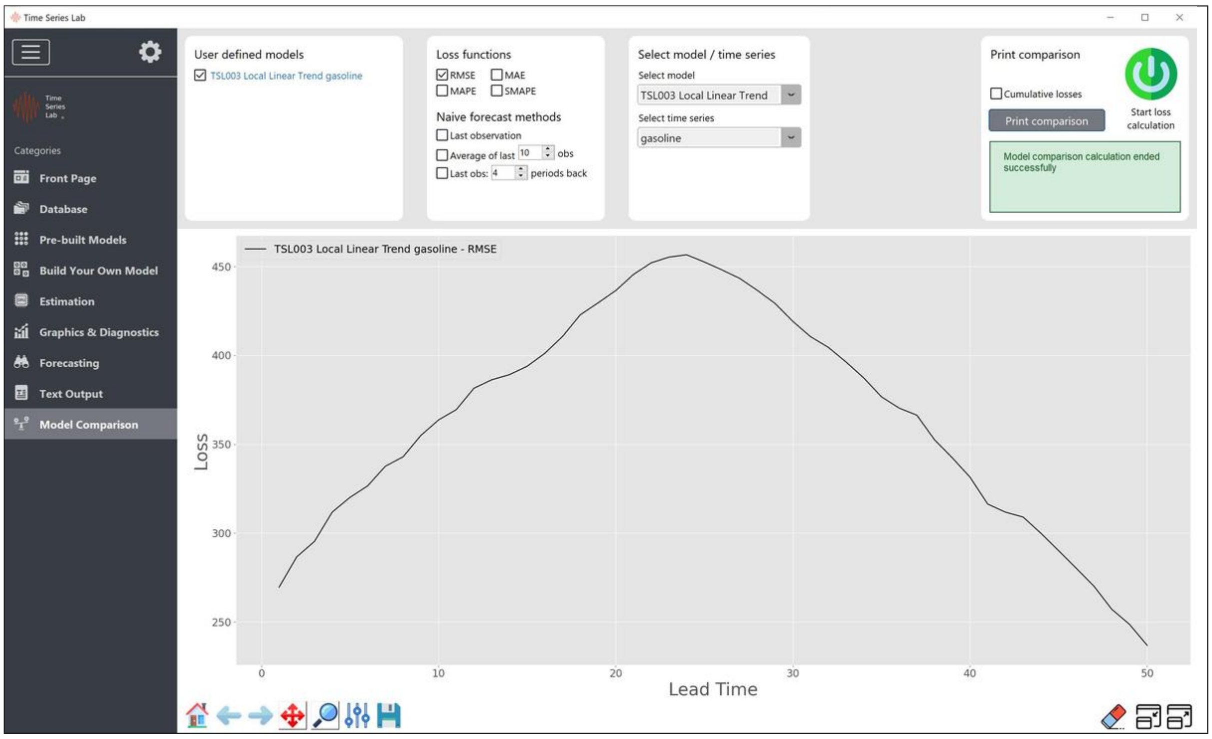
12.2.2 Basic Structural Time Series model

It is time to introduce a seasonal component. We go to the *Build your own model* page and select a time-varying level, a time-varying slope, and a time-varying seasonal. The resulting model is called the *Basic Structural Time Series model* by Harvey (1990). Set the Seasonal period length to $365.25/7 \approx 52.179$ (weekly data taking leap years into account) and a Number of factors equal to 22.

Information: Seasonal period length is the number of time points after which the seasonal repeats. This can be a fractional number. For example, with daily data, specify a period of 365.25 for a seasonal that repeats each year, taking leap years into account. Number of factors specifies the seasonal flexibility. Note that a higher number is not always better and parsimonious models often perform better in forecasting.

Figure 12.10

RMSE loss Local Linear Trend model and Gasoline dataset



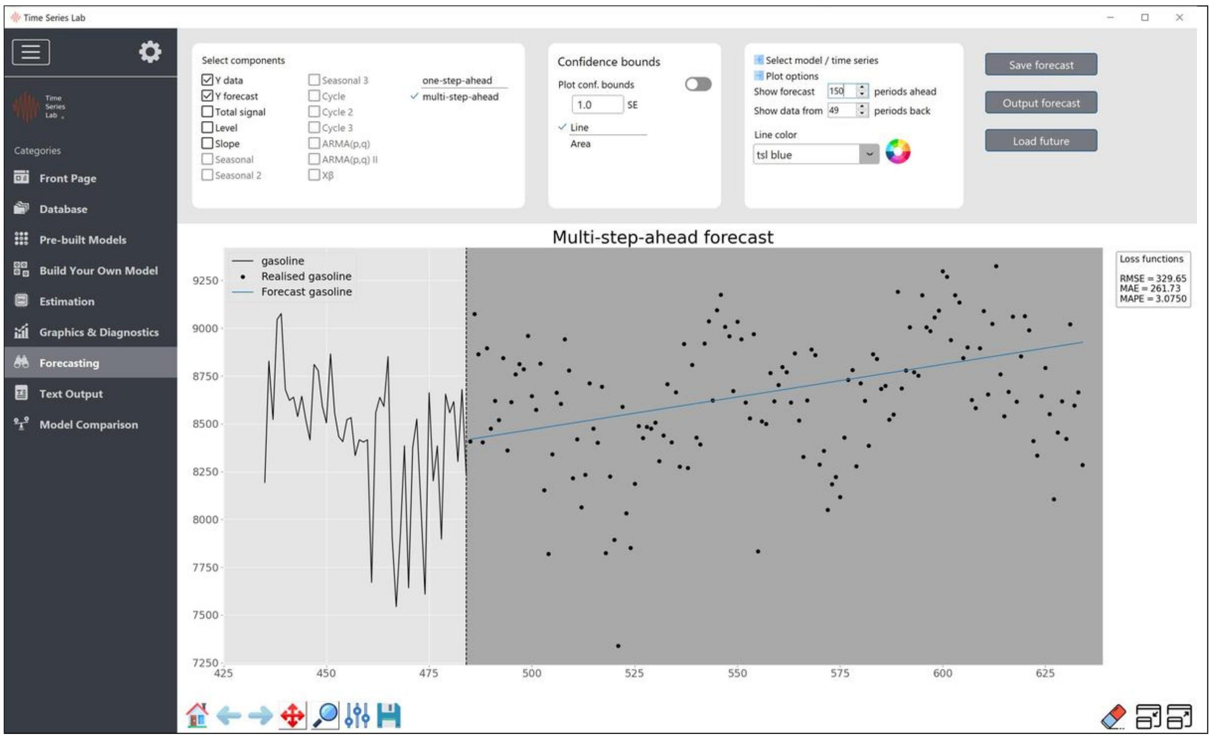
RMSE loss for the Local Linear Trend model and the Gasoline dataset for a forecast horizon of h=50.

The *Build your own model* page should look like the one in Figure 12.12. Estimate the model and go to the *Text output* page. We see that the model fit is improved by adding the seasonal component.

```
Variable: gasoline
Model: TSL004
                                                TSL004
Log likelihood                               -3218.400
Akaike Information Criterion (AIC)            6532.799
Bias corrected AIC (AICc)                     6543.613
Bayesian Information Criterion (BIC)          6733.539
in-sample MSE                               1.0083e+05
... RMSE                                       317.539
... MAE                                        252.502
... MAPE                                         3.244
Sample size                                        484
Effective sample size                              430
* based on one-step-ahead forecast errors
```

Figure 12.11

Forecasts for h=150 time points ahead



Plot conf. bounds 1.0 SE

Show forecast 150 periods ahead

Show data from 49 periods back

Line color: tsl blue

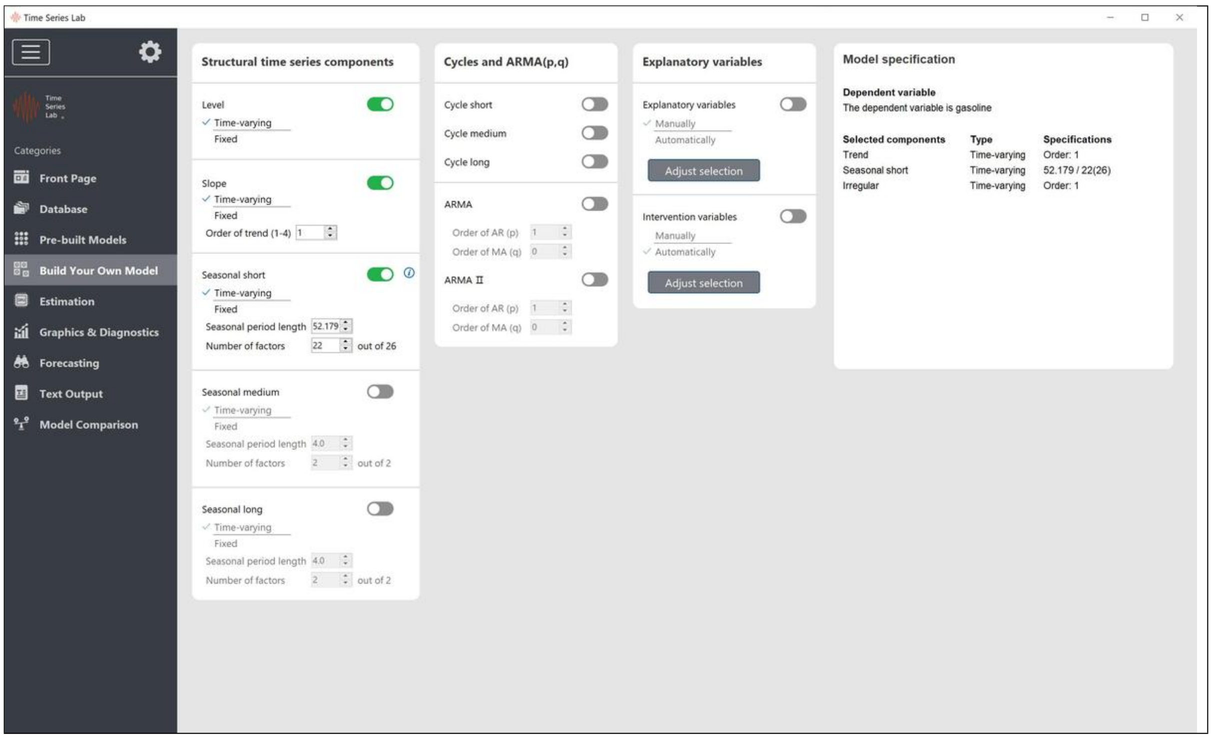
RMSE = 329.65
MAE = 261.73
MAPE = 3.0750

A large improvement in forecasting performance, compared to the Local Linear Trend model, can be seen if we start (and plot) the loss calculation on the *Model comparison* page. This shows how important it can be to model the seasonal pattern of a time series. We will see an example of multiple seasonal patterns in a time series which makes the correct handling even more important. Plotting both RMSE losses leads to Figure 12.13. Next, go back to the Build your own model page and lower the number of factors of the seasonal component. Parsimonious models often perform better in forecasting. A better performing number of factors is 7 although other values might be even better for forecasting. A model comparison between 22 and 7 factors is made in Figure 12.14. The loss corresponding to the model with 7 factors is lower than the one that is presented in Figure 2 of De Livera et al. (2011) which is obtained with the TBATS package.

12.2.3 Further exploration

- Estimate the model with a Level, Slope, and Seasonal with period length 52. Verify that by not taking the leap year into account (52.0 instead of 52.179), forecasts become worse.
- Forecasts can further be improved by adding explanatory variables. In TSL you can do this with the click of a couple of buttons on the Model setup page. Let us know which variables you have used to boost the forecast precision for the gasoline dataset.

Figure 12.12
Component selection page



Structural time series components

Level ✓ Time-varying / Fixed

Slope ✓ Time-varying / Fixed
Order of trend (1-4) 1

Seasonal short ✓ Time-varying / Fixed
Seasonal period length 52.179
Number of factors 22 out of 26

Seasonal medium ✓ Time-varying / Fixed
Seasonal period length 4.0
Number of factors 2 out of 2

Seasonal long ✓ Time-varying / Fixed
Seasonal period length 4.0
Number of factors 2 out of 2

Cycles and ARMA(p,q)

Cycle short
Cycle medium
Cycle long

ARMA
Order of AR (p) 1
Order of MA (q) 0

ARMA II
Order of AR (p) 1
Order of MA (q) 0

Explanatory variables

Explanatory variables ✓ Manually / Automatically

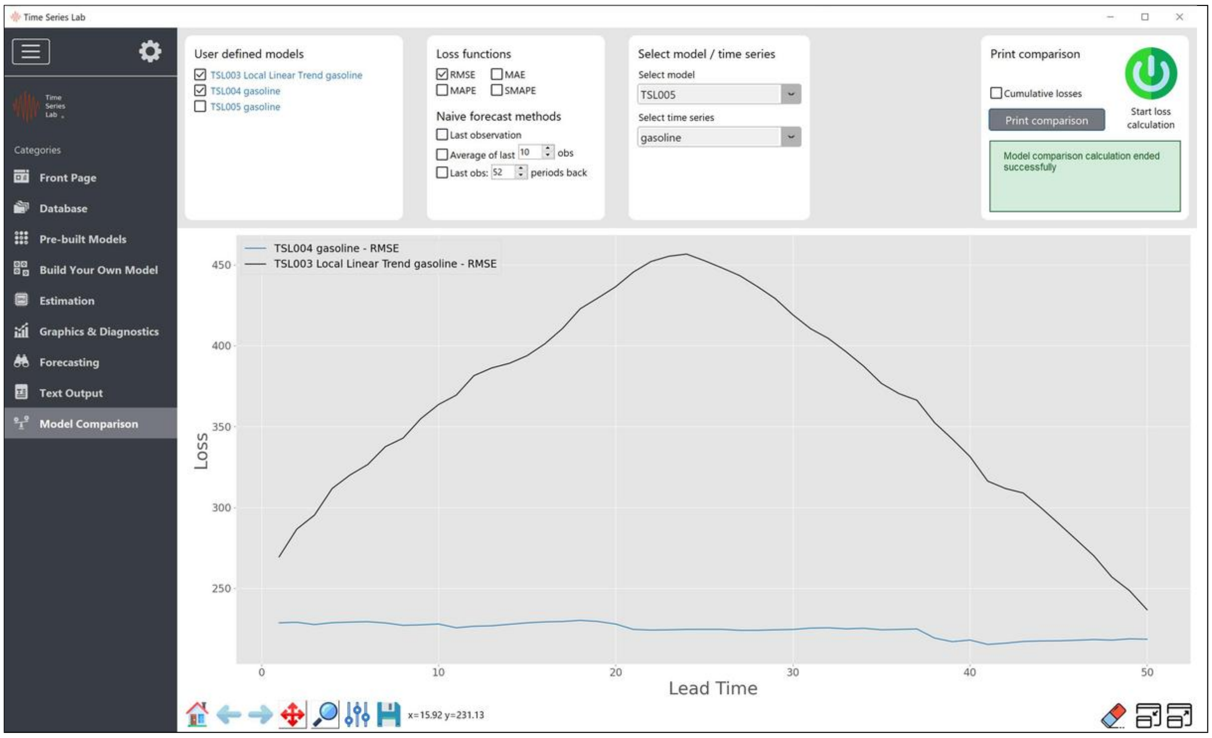
Intervention variables Manually / ✓ Automatically

Model specification

Dependent variable
The dependent variable is gasoline

Selected components	**Type**	**Specifications**
Trend	Time-varying	Order: 1
Seasonal short	Time-varying	52.179 / 22(26)
Irregular	Time-varying	Order: 1

Figure 12.13
Forecast performance of LLT and Basic Structural model



User defined models
☑ TSL003 Local Linear Trend gasoline
☑ TSL004 gasoline
☐ TSL005 gasoline

Loss functions
☑ RMSE ☐ MAE
☐ MAPE ☐ SMAPE

Naive forecast methods
☐ Last observation
☐ Average of last 10 obs
☐ Last obs: 52 periods back

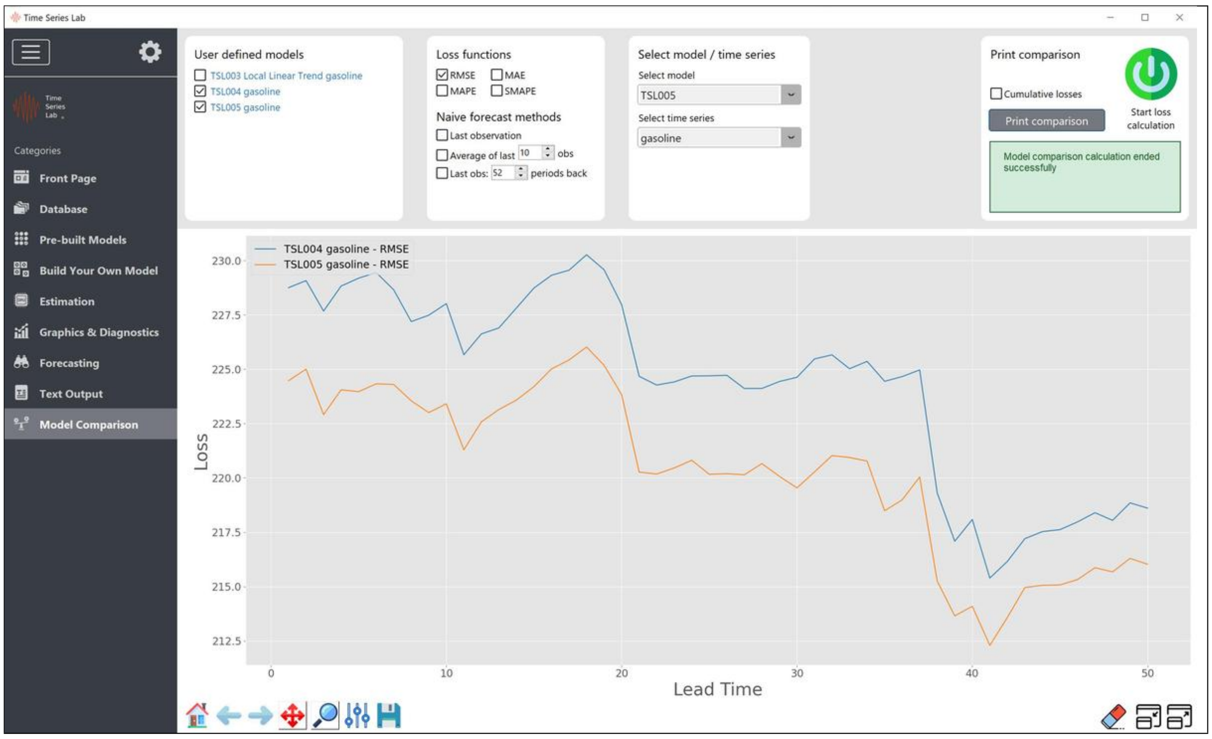
Select model / time series
Select model: TSL005
Select time series: gasoline

Print comparison
☐ Cumulative losses

Model comparison calculation ended successfully

Figure 12.14

Forecast performance Basic Structural model



User defined models
- ☐ TSL003 Local Linear Trend gasoline
- ☑ TSL004 gasoline
- ☑ TSL005 gasoline

Loss functions
- ☑ RMSE ☐ MAE
- ☐ MAPE ☐ SMAPE

Naive forecast methods
- ☐ Last observation
- ☐ Average of last 10 obs
- ☐ Last obs: 52 periods back

Select model: TSL005

Select time series: gasoline

☐ Cumulative losses

Model comparison calculation ended successfully

12.3 UK GAS consumption

The Energy data set that comes bundled with TSL has quarterly data on UK energy consumption. The data set is good for illustrative purposes since the data exhibits time-varying seasonality and the presence of strong outliers.

Load the Energy data set and select variable *ofuGASl* where the trailing l means that logarithms were taken from the original data. The trend is upwards with some increase in the rate of growth in the years following the introduction of cheaper natural gas from the North Sea at the end of the 1960s.

12.3.1 Energy consumption *without* intervention variables

Select a time-varying level, time-varying slope, and time-varying seasonal on the *Build your own model* page. Since the data is quarterly data, we have a seasonal period of $s = 4$. Go to the Estimation page and click the Estimate button. After TSL is finished with the Estimation, go to the *Text output* page where you find (partially) the following output:

```
--------------------------------- PARAMETER SUMMARY ---------------------------------

Variance of disturbances:

Variance type                    Value        q-ratio
Level variance              3.7108e-07     2.2775e-04
Slope variance              7.4626e-06         0.0046
Seasonal variance           8.3846e-04         0.5146
Irregular variance              0.0016         1.0000


Seasonal short properties:

Period                           Value        Std.Err         t-stat           Prob
1                               0.6082         0.0804          7.562     1.7114e-11
2                              -0.0884         0.0766         -1.153         0.2514
3                              -0.6695         0.0658        -10.167         0.0000
4                               0.1497         0.0573          2.611         0.0104

                                 Value                                         Prob
Seasonal chi2 test               79.15                                   4.6816e-17

--------------------------------- MODEL FIT ---------------------------------

Model: TSL001
variable: ofuGASl

                                            TSL001
Log likelihood                             83.1412
```

```
Akaike Information Criterion (AIC)          -148.2824
Bias corrected AIC (AICc)                   -146.4456
Bayesian Information Criterion (BIC)        -124.1432
in-sample MSE                                  0.0108
... RMSE                                       0.1039
... MAE                                        0.0696
... MAPE                                       1.2493
Sample size                                       108
Effective sample size                             103
* based on one-step-ahead forecast errors
```

Under *Seasonal short properties* we see the average value of the seasonal periods. The seasonal component is a component that evolves around zero. The periods of the seasonal tell us that gas consumption is on average higher in Q1 and Q4 (> 0) and lower in Q2 and Q3 (< 0) which is not surprising due to temperature effects. An important statistic is the *Seasonal chi2 test* which is the combined effect of the seasonal component. The individual components are not **all** statistically different from zero (p-value of Q2 is 0.2514) but the total effect of the seasonal component is strongly significant with a p-value of 4.6816e-17.

Next, go to the *Graph page* and construct a figure with four subplots (use add subplot button bottom right) with the following content:

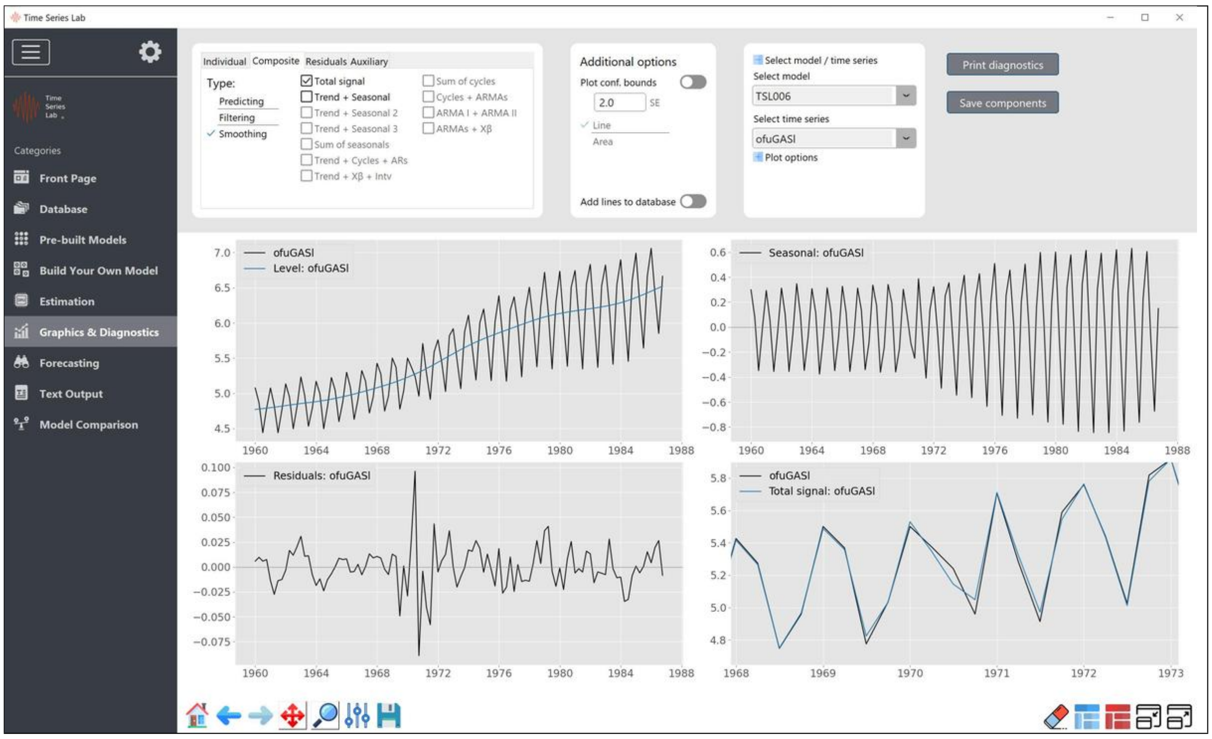
- Top left: y data and smoothed level
- Top right: smoothed seasonal
- Bottom left: smoothed residuals
- Bottom right: y data and Total signal, zoomed in on the period $1968 - 1973$.

The figure should look like the one in Figure 12.15. The first graph shows the trend; there is a strong increase in gas usage with the introduction of natural gas from the North sea in the early 1970s. The seasonality is shown in the graph in the top right corner in terms of its multiplicative effect on the trend. The greater dispersion in the seasonal pattern over time is due to a higher proportion of gas being used for heating as usage increased in the 1970s. The final graph shows the seasonally adjusted series produced by fitting the structural time series model. Furthermore, two spikes are present in the residuals around 1970 as can be seen in the bottom left panel. The *Total signal* (bottom right) tracks the data accurately except for a discrepancy in 1970. Let's investigate the residuals further. On the Graph page, press the *Print diagnostics* button and select *Outlier and break diagnostics* and click Continue. TSL prints the following on the Text output page.

```
Diagnostic output for:
model: TSL001
variable: ofuGASl


Outlier and break diagnostics
```

Figure 12.15

Basic Structural model for energy consumption



The figure shows the result of fitting a model with time-varing level, slope, and seasonal $s = 4$. The top left panel show the data and the smoothed level. The top right panel shows the smoothed seasonal. The bottom left panel shows the smoothed residuals. The bottom right panel shows the data and the total signal, zoomed in on the period $1968 - 1973$.

```
Values larger than 3.00 for Irregular residual:

Period                             Value          Prob
1970-07-01                         4.287    1.9773e-05
1970-10-01                        -3.972    6.4463e-05

No values larger than 3.00 for Level residual:

No values larger than 3.00 for Slope residual:
```

Outliers are identified from large Irregular residuals and structural breaks from Level residuals. We see that the outlier periods are identified as 1970-07-01 (Q3) and 1970-10-01 (Q4).

12.3.2 Energy consumption *with* intervention variables

We use the same model as in the last section and add *Intervention variables* to the model, i.e. switch on *Intervention variables* on the Build your own model page and select the Automatically option. Estimate the model.

```
--------------------------------- PARAMETER SUMMARY ---------------------------------

Variance of disturbances:

Variance type                   Value        q-ratio
Level variance             2.5186e-04         0.4572
Slope variance             5.2830e-06         0.0096
Seasonal variance          5.5081e-04         1.0000
Irregular variance         8.6844e-05         0.1577


Seasonal short properties:

Period                          Value        Std.Err         t-stat           Prob
1                              0.6116         0.0616          9.926         0.0000
2                             -0.0862         0.0574         -1.500         0.1367
3                             -0.6579         0.0479        -13.739         0.0000
4                              0.1325         0.0411          3.223         0.0017

                                Value                                         Prob
Seasonal chi2 test              197.0                                   1.8447e-42


Intervention coefficients:

Beta                            Value        Std.Err         t-stat           Prob
beta_outlier_1970-07-01        0.4024         0.0530          7.585     1.6711e-11
beta_outlier_1970-10-01       -0.3372         0.0530         -6.357     6.0066e-09

--------------------------------- MODEL FIT ---------------------------------

Model: TSL002
variable: ofuGASl

                                            TSL002
Log likelihood                            108.4862
Akaike Information Criterion (AIC)       -194.9724
Bias corrected AIC (AICc)                -192.2224
Bayesian Information Criterion (BIC)     -165.4690
in-sample MSE                               0.0095
... RMSE                                    0.0976
... MAE                                     0.0659
... MAPE                                    1.1722
Sample size                                    108
Effective sample size                          101
* based on one-step-ahead forecast errors
```

We see that TSL correctly finds the outliers that were presented at the end of the last section and that the corresponding probabilities are closed to zero meaning the outliers are strongly significant. There is another way of assessing the contribution of the added outliers and that is by performing a *likelihood ratio test* (LR test) between the model with and without the outliers. In a LR test you compare two log-likelihood values and test if the difference is statistically significant based on a number of degrees-of-freedom. The null hypothesis of the LR test is:

H_0 : the smaller model provides as good a fit for the data as the larger model

with the test statistic of the LR test given by

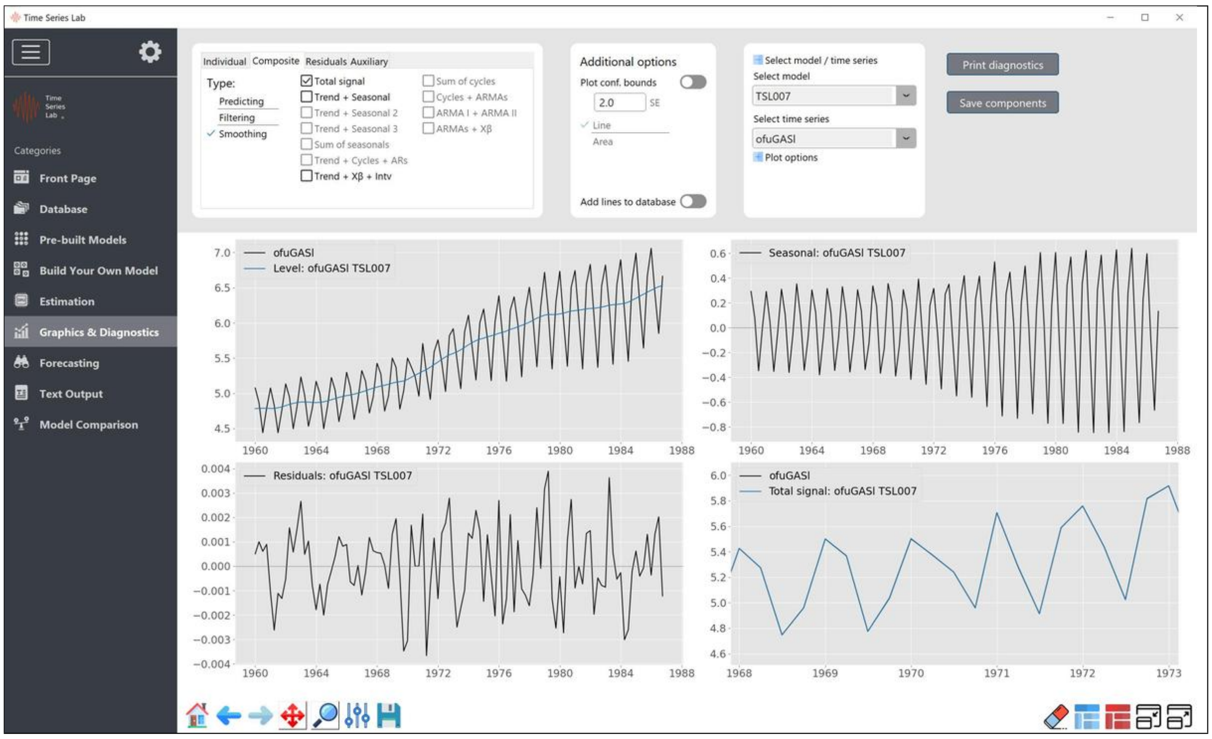
$$-2\left[\ell(\theta_0) - \ell(\theta_1)\right]$$

where $\ell(\theta_i)$ is the log-likelihood of model i. The LR statistic for our two likelihoods is 50.68 and the statistic is *Chi-square* distributed with degrees of freedom equal to the difference in the number of parameters for the two models, which is 2 in our case (2 extra outlier variables). The probability belonging to our LR statistic is 9.8416e-12 so we strongly reject our null hypothesis.

A similar graph like Figure 12.15, with the outliers added, is presented in Figure 12.16. Compared to Figure 12.15, we see some important differences. One, the spikes in the residuals are gone. Two, the discrepancy in 1970 in the Total signal is completely gone.

Figure 12.16

Basic Structural model + Interventions for energy consumption



Type: Smoothing

☑ Total signal

Plot conf. bounds: 2.0 SE

Select model: TSL007

Select time series: ofuGASI

The figure shows the result of fitting a model with time-varing level, slope, seasonal $s = 4$, and intervention variables. The top left panel show the data and the smoothed level + interventions. The top right panel shows the smoothed seasonal. The bottom left panel shows the smoothed residuals. The bottom right panel shows the data and the total signal, zoomed in on the period $1968 - 1973$.

12.4 El Nino

El Niño is a well-known phenomenon in climate science and is characterized by higher than average sea surface temperatures in the central and eastern equatorial Pacific Ocean. It has a substantial impact on the climate in many parts of the world. Hence, it has been given much coverage in the popular media, and it is the subject of extensive research in the scientific world. El Niño typically causes changes in weather patterns related to temperature, pressure and rainfall. Thus, a warm event may not only have a negative impact on local economies, but can also have negative consequences for public health, as in some regions these changes increase substantially the risk of water-borne and/or vector-borne diseases. Given its huge impact particularly on some developing countries bordering the Pacific Ocean, it is self-evident that a timely forecast of the next El Niño event is important. Much scientific research has been devoted to the development of forecasting methods for El Niño. The oscillation is characterized by an irregular period of between 2 and 7 years. Currently, forecasts are issued regularly for up to three seasons in advance, but the long term of more than one year ahead forecasts remain a real challenge. At the same time, one of the two main theories about the physics underlying El Niño implies that it may be a self-sustaining climatic fluctuation that is quasi-periodic, with several dominant peaks in its spectrum, the main one being at about every 4-5 years and a secondary at about 2 years. This suggests that it may be predictable at lead times of several years, see also Li et al. (2020).

In this Case study we show that we can make accurate forecasts with TSL. We build up the model in several steps and show increases in training sample model fit and validation sample forecast performance in each step.

12.4.1 Loading and inspecting the data

Load and select the *elnino.csv* data set from the file system by pressing the Load data button or by clicking *File* ▶ *Load data*.

Our series of interest is the *EN3.4* time series. This is a time series of monthly temperature values which is referred to as the Niño3.4 time series and which is the area-averaged sea surface temperature in the region (5° N - 5° S, 170° W - 120° W). In this area the El Niño events are identified, see also the discussion in Bamston et al. (1997). The National Centers for Environmental Information (NOAA) defines an El Niño or La Niña event as a phenomenon in the equatorial Pacific Ocean characterised by a five consecutive 3-month running mean of sea surface temperature (SST) anomalies in the Niño 3.4 region that is above (below) the threshold of $+$0.5°C (-0.5°C)[1].

In our empirical study, the Niño3.4 time series, denoted by y_t, is the variable of interest. The variable is observed from January 1982 to the end of 2015 with 34 years of data and 407 monthly observations. For this period, observations for 24 predictor variables are available which consist of physical measures of zonal wind stress and sea temperatures at different

[1]Details can be found on the website of NOAA, `https://www.ncdc.noaa.gov/`

depths in the ocean and at different locations. Petrova et al. (2017) give a detailed account of the selection of these variables. For graphs, acronyms and references to data sources for all time series, we refer to Appendix B of Li et al. (2020).

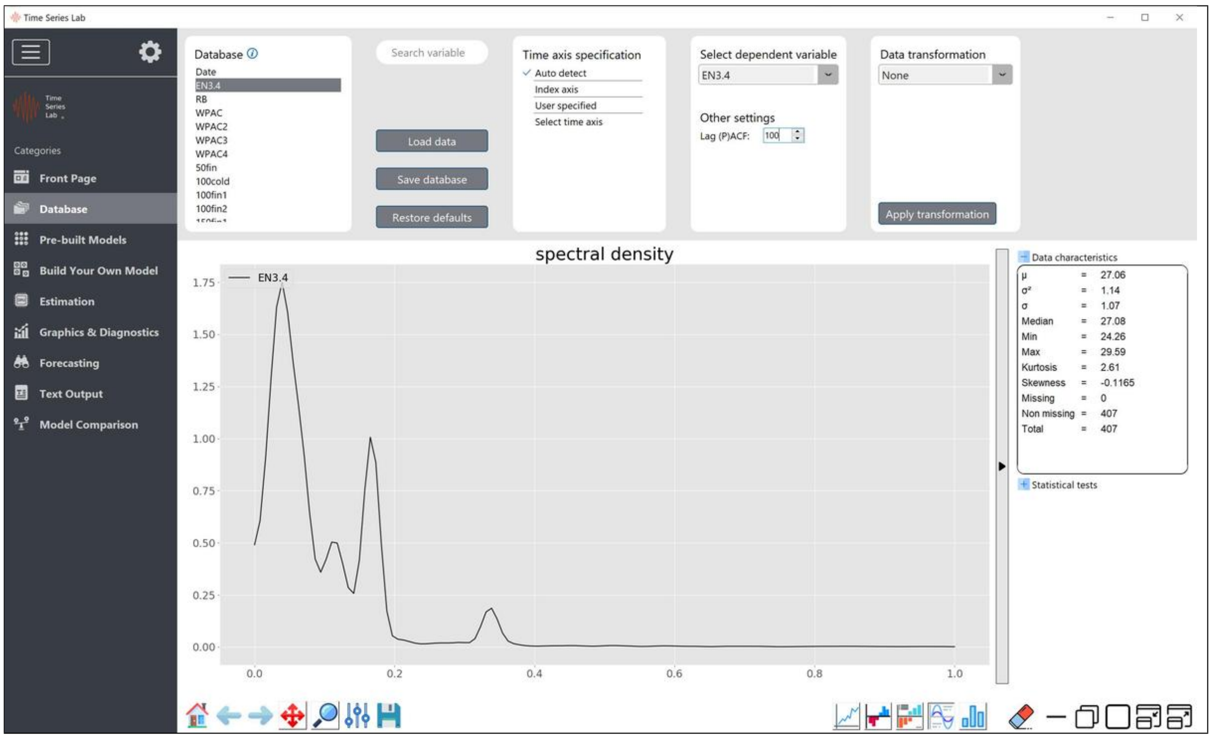
Click the vertical arrow bar on the right of the screen to see additional information about the selected time series. The Statistical tests panel shows the result of the Augmented Dickey-Fuller test and KPSS test. The ADF test (with intercept) strongly implies that the null hypothesis of a unit root is rejected. Based on the KPSS test, we cannot reject the null hypothesis of trend stationarity which suggests that the Niño3.4 time series is generated from a stationary process around a fixed mean.

12.4.1.1 Periodicity in the time series

The spectral density is a very useful plot to get an idea about cyclical patterns in the time series. Click on the *spectral density* button in the bottom right corner of the graph and change the number of lags to 100 in the spinbox under *other settings*. The screen should like to the one in Figure 12.17. From the sample spectrum, we can identify four peaks which correspond

Figure 12.17

Spectral density of the EN3.4 time series



Time axis specification: ✓ Auto detect

Select dependent variable: EN3.4

Other settings: Lag (P)ACF: 100

Data transformation: None

Data characteristics

```
μ           =  27.06
σ²          =  1.14
σ           =  1.07
Median      =  27.08
Min         =  24.26
Max         =  29.59
Kurtosis    =  2.61
Skewness    =  -0.1165
Missing     =  0
Non missing =  407
Total       =  407
```

to periods of approximately 6, 12, 18, and 51 months. The 6 and 12 month periods correspond to the monthly seasonality and the 18 and 51 month periods will be modelled with cycles.

12.4.2 Model: level + slope + seasonal

We build a model with a time-varying level, time-varying slope, and time-varying seasonal. Select these components on the Build your own model page. The cycles come at a later stage. After selecting the components, go to the Estimation page and change the end of the sample to 324 (27 years) which leaves 83 months as Validation sample. Click the Estimate button and after TSL is done estimating, go to the Text output page where we see:

```
---------------------------------- PARAMETER SUMMARY ----------------------------------

Variance of disturbances:

Variance type                      Value        q-ratio
Level variance                    0.0872         1.0000
Slope variance                    0.0000         0.0000
Seasonal variance                 0.0000         0.0000
Irregular variance            8.4884e-05     9.7383e-04


State vector at period 2008-12-01:

Component                          Value        Std.Err         t-stat           Prob
Level                            26.2030         0.0576       454.9773         0.0000
Slope                            -0.0032         0.0164        -0.1922         0.8477

---------------------------------- MODEL FIT ----------------------------------

Model: TSL006
variable: EN3.4

                                          TSL006
Log likelihood                          -95.6977
Akaike Information Criterion (AIC)      221.3954
Bias corrected AIC (AICc)               222.9539
Bayesian Information Criterion (BIC)    278.1066
in-sample MSE                             0.1042
... RMSE                                  0.3229
... MAE                                   0.2526
... MAPE                                  0.9364
Sample size                                  324
Effective sample size                        311
* based on one-step-ahead forecast errors
```

The *Variance of disturbances* show something interesting. We have two of the four variances estimated at zero. A value of zero for a variance indicates that the corresponding component is deterministic. If this is the case, a standard regression type significance test can be carried out on the corresponding component in the state. If it is not significantly different from zero, it may be possible to simplify the model by eliminating that particular component. Since the Slope component has a variance of zero and the *t-stat* for the Slope component at time $2008 - 12 - 01$ has a value of -0.1923 with corresponding *p-value* of 0.8477, we can safely remove the Slope component from the model.

De-select the Slope component on the Build your own model page and re-estimate the model. Go to the Text output page where we see:

```
--------------------------------- PARAMETER SUMMARY ---------------------------------

Variance of disturbances:

Variance type                    Value       q-ratio
Level variance                  0.0869        1.0000
Seasonal variance               0.0000        0.0000
Irregular variance          8.5250e-05    9.8151e-04


Seasonal short properties:

Period                           Value       Std.Err        t-stat          Prob
1                              -0.5053        0.0568        -8.901        0.0000
2                              -0.3412        0.0567        -6.018    4.9541e-09
3                               0.1496        0.0566         2.642        0.0087
4                               0.7176        0.0566        12.680        0.0000
5                               0.8035        0.0566        14.205        0.0000
6                               0.6069        0.0565        10.732        0.0000
7                               0.2026        0.0565         3.583    3.9341e-04
8                              -0.1848        0.0566        -3.267        0.0012
9                              -0.2843        0.0566        -5.023    8.5673e-07
10                             -0.3200        0.0566        -5.650    3.6164e-08
11                             -0.3806        0.0567        -6.712    9.0447e-11
12                             -0.4640        0.0568        -8.172    7.5495e-15

                                 Value                                      Prob
Seasonal chi2 test               302.4                                2.7131e-58


State vector at period 2008-12-01:

Component                        Value       Std.Err        t-stat          Prob
Level                            26.20        0.0575         455.7             0

--------------------------------- MODEL FIT ---------------------------------

Model: TSL007
variable: EN3.4

                                                        TSL007
```

```
Log likelihood                                -92.5260
Akaike Information Criterion (AIC)            213.0520
Bias corrected AIC (AICc)                     214.4112
Bayesian Information Criterion (BIC)          265.9824
in-sample MSE                                   0.0990
... RMSE                                        0.3146
... MAE                                         0.2471
... MAPE                                        0.9165
Sample size                                        324
Effective sample size                              312
* based on one-step-ahead forecast errors
```

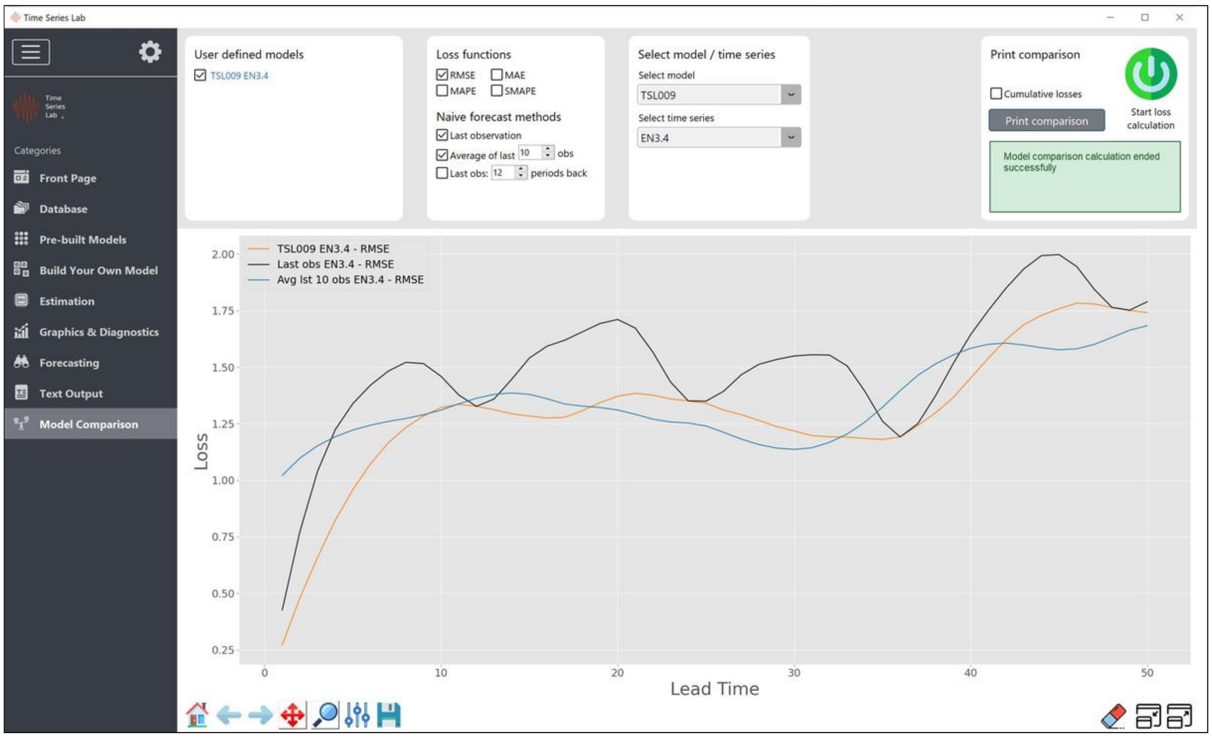
Since the variance of the seasonal component is zero as well, we have a deterministic seasonal which is a special case of the stochastic seasonal. Alternatively, the fixed seasonal can be incorporated within X_t as it is usually done in regression models. If the seasonal component is deterministic, either because it is specified to be 'fixed' at the outset or its disturbance variance is estimated to be zero, a joint test of significance can be carried out on the s - 1 seasonal effects. The test is essentially the same as a test for the joint significance of a set of explanatory variables in regression. Under the null hypothesis of no seasonality, the large sample distribution of the test statistic, denoted by *Seasonal chi2 test* in the output, is χ^2_{s-1} distributed. The *Prob* value is the probability of a χ^2_{s-1} variable exceeding the value of the test statistic. In the case of a stochastic seasonal, the joint seasonal test is also produced although a formal joint test of significance of the seasonal effect is inappropriate. However, the seasonal pattern is persistent throughout the series and when the seasonal pattern changes relatively slowly, which is usually the case, the test statistic can provide a useful guide to the relative importance of the seasonal. The formal definition of the test statistic is

$$a'P_1a$$

where a contains the estimates of the s - 1 seasonal effects at time T and P is the corresponding variance matrix. From the text output we see that we have a strongly significant seasonal pattern with a p-value of close to zero.

Finally, we have the *Model fit* which we will use to compare this model to other models. The result above are for the Training sample but we need Validation results as well to be able to thoroughly compare models with each other. Go to the Model comparison page and click the Start loss calculation button in the top right corner of the screen. We refer to Chapter 10 for more information on loss calculations. After the loss calculations are made, an entry appears under *user defined models* in the top left corner of the screen. Tick the box and also tick the *Last observation* and *Average of last 10 observations* boxes to add two benchmark models to the graph. The resulting graph should like like the one presented in Figure 12.18. We can see that compared to the (simple) benchmark models, our model is performing better for 1 to 18-step-ahead forecast but after that our model is not better anymore. For forecast horizons of 19 and higher our model does not perform better than a very simple benchmark

Figure 12.18
Forecast losses for different lags



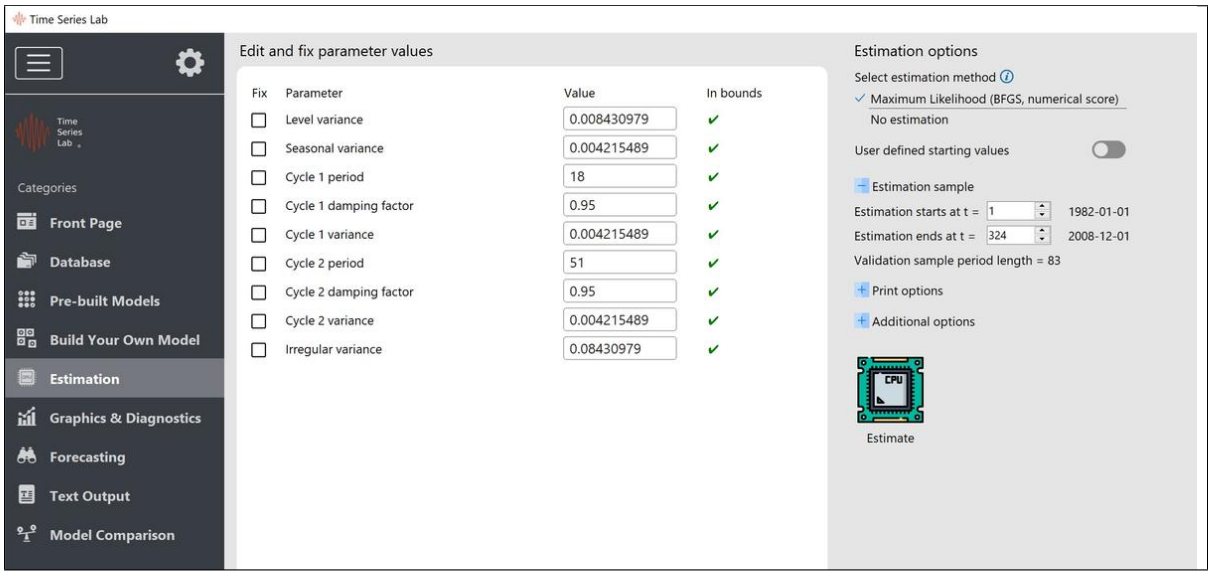
model. You should always compare your model to simple benchmark models and sometimes you come to the conclusion that you cannot do better. But of course we do not end this Case study here because we can improve further.

12.4.3 Model: level + seasonal + cycle1 + cycle2

We continue the modelling process by adding two cycles to our existing model on the Build your own model page. Next, go to the Estimation page. Under the header *Edit and fix parameter values*, all parameters that need can be estimated are summarized. For specific model specifications we can fix parameters at certain values. If *User defined starting values* is switched off (which is the default), an algorithm determines the starting values before optimization. Note that the current values in the column *Value* are not the algorithmic determined starting values. That process starts after the green Estimate button has been clicked. For the majority of the cases, it is best to let TSL determine the starting values. An exception to the above is the *period length* of the cycle which can be set as starting value by the user regardless if *User defined starting values* is switched on or off. Since we target a period of 18 months and a period of around 51 months (information obtained from the spectral density), we set the starting value of the cycle 1 period to 18 and the cycle 2 period to 51. Note that we always have cycle period 1 $<$ cycle period 2 $<$ cycle period 3. The resulting screen should look like the one in Figure 12.19. Click the Estimate button and go to the Text output page. Among other output, we have:

Figure 12.19

Estimation page of TSL



Edit and fix parameter values

Fix	Parameter	Value	In bounds
☐	Level variance	0.008430979	✓
☐	Seasonal variance	0.004215489	✓
☐	Cycle 1 period	18	✓
☐	Cycle 1 damping factor	0.95	✓
☐	Cycle 1 variance	0.004215489	✓
☐	Cycle 2 period	51	✓
☐	Cycle 2 damping factor	0.95	✓
☐	Cycle 2 variance	0.004215489	✓
☐	Irregular variance	0.08430979	✓

Estimation options

Select estimation method

✓ Maximum Likelihood (BFGS, numerical score)

No estimation

User defined starting values

Estimation sample

Estimation starts at t = 1 1982-01-01

Estimation ends at t = 324 2008-12-01

Validation sample period length = 83

Print options

Additional options

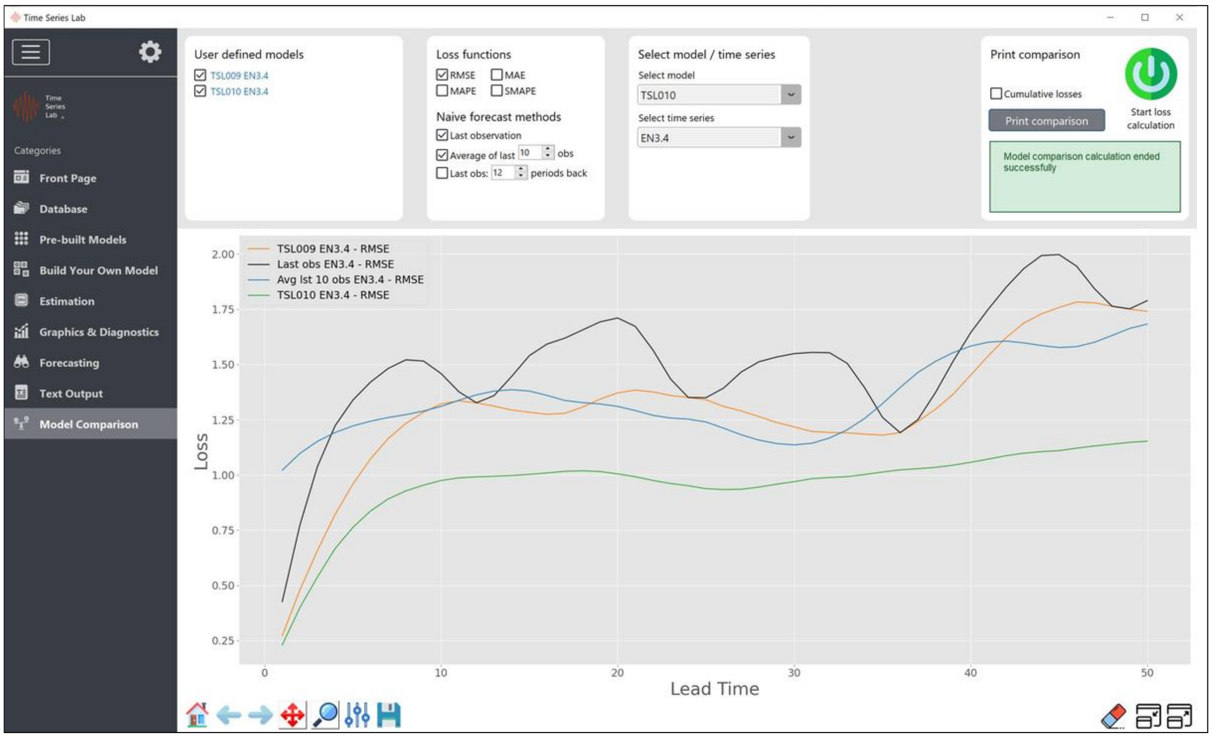
```
--------------------------------- PARAMETER SUMMARY ---------------------------------

Cycle properties:

Parameter type                  Cycle 1        Cycle 2
Variance                         0.1708         0.7423
Period                          17.8175        48.1535
Frequency                        0.3526         0.1305
Damping factor                   0.9622         0.9721
Amplitude                        0.5931         0.9511

---------------------------- TRAINING SAMPLE MODEL FIT -----------------------------

Variable: EN3.4
Model: TSL008
                                              TSL008
Log likelihood                              -53.8648
Akaike Information Criterion (AIC)          147.7296
Bias corrected AIC (AICc)                   150.5019
Bayesian Information Criterion (BIC)        223.3445
in-sample MSE                                 0.0775
... RMSE                                      0.2783
... MAE                                       0.2146
... MAPE                                      0.7985
Sample size                                      324
Effective sample size                            312
* based on one-step-ahead forecast errors
```

We see that the periods of the cycles are estimated around 18 months (1.5 years) and 48 months (4 years) which nicely corresponds with the information from the spectral density. The model is also improved on training sample fit since we have a higher log likelihood and lower in-sample losses.

Go to the Model comparison page and add the loss of the latest model to the graph. You should see a loss line that is below all other loss lines meaning the latest model performs better for all forecast horizons, see also Figure 12.20.

Figure 12.20

Forecast losses for different lags


User defined models: ☑ TSL009 EN3.4 ☑ TSL010 EN3.4

Loss functions: ☑ RMSE ☐ MAE ☐ MAPE ☐ SMAPE

Naive forecast methods: ☑ Last observation ☑ Average of last 10 obs ☐ Last obs: 12 periods back

Select model: TSL010

Select time series: EN3.4

☐ Cumulative losses

Model comparison calculation ended successfully

12.4.4 Further exploration

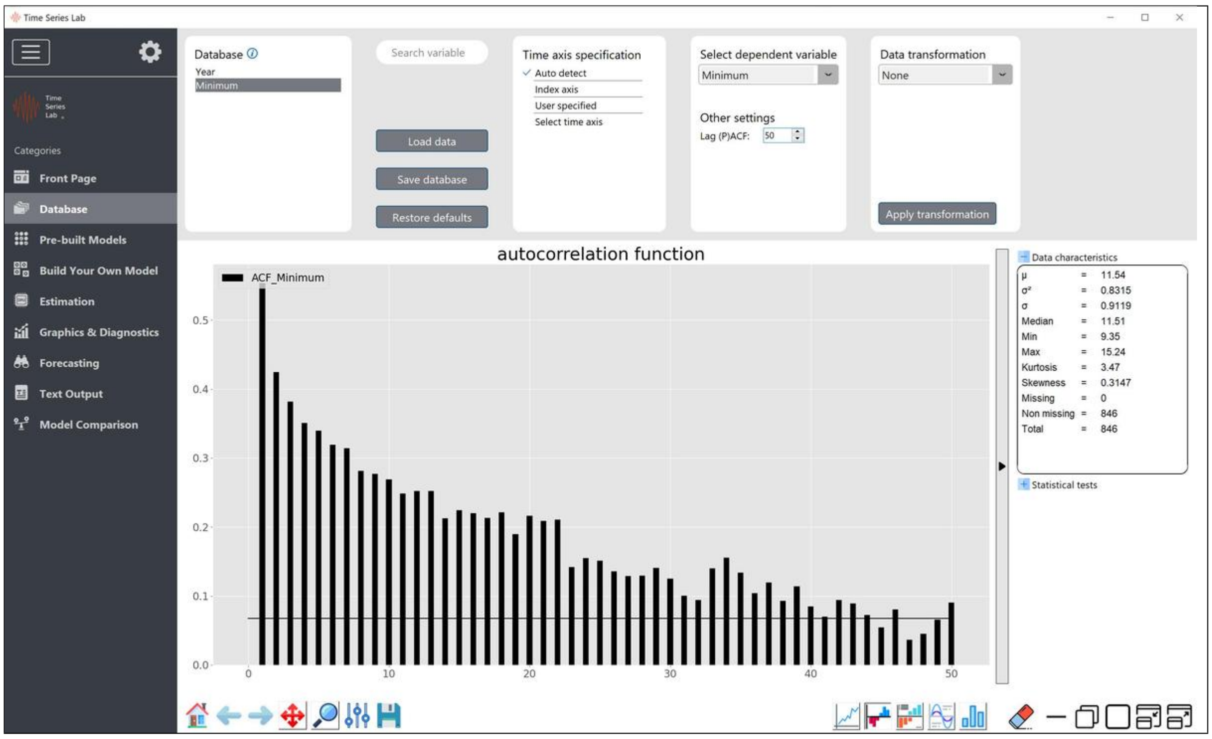
- Verify that the ACF of the standardized residuals from the model with time-varying level, slope, and seasonal (no cycles) has a cyclical pattern indicating that there is signal left to explain.

12.5 Long memory

We continue the case studies with a record of the lowest annual water levels on the Nile river during 622-1467 measured at the island of Roda, near Cairo, Egypt. The series is also available till 1918 but has periods of many missing values which is not the topic of this case study. For missing value analysis see for example Case study 12.1.4.

The Nile Minimum dataset is part of any TSL installer file and can be found in the data folder located in the install folder of TSL. When we inspect the autocorrelation function of the time series on the Database page, we find that the ACF displays a classic long memory pattern. Even after increasing the number of lags to 50, we still find significant lags, see also Figure 12.21

Figure 12.21
Autocorrelation function Nile Minimum



Database: Year, Minimum

Time axis specification: ✓ Auto detect

Select dependent variable: Minimum

Other settings: Lag (P)ACF: 50

Data transformation: None

Data characteristics

```
μ           =  11.54
σ²          =  0.8315
σ           =  0.9119
Median      =  11.51
Min         =  9.35
Max         =  15.24
Kurtosis    =  3.47
Skewness    =  0.3147
Missing     =  0
Non missing =  846
Total       =  846
```

In this case study we will demonstrate several ways of modelling this dataset. We begin with the score-driven models which show interesting results, especially if we deviate from the Normal distribution.

12.5.1 Score-driven models

The power of score-driven models lies in the ability of score-driven models to deviate from the Normal distribution for the irregular component of the model. A distribution like the Student t distribution, for example, is much less susceptible to outliers in the data. Furthermore, the score-driven models allow us to choose an arbitrary number of AR orders (p) and score lags (q). But how to choose p and q? It turns out that the algorithm of Hyndman and Khandakar (2008) to find optimum values for p and q works for score-driven models as well, see also Section 4.2.1. This does not come as a surprise since ARMA models are subsets of score-driven models.

We navigate to the *Pre-built models* page of TSL and select only the model *DCS-g* in the score-driven column. We then tick the *Auto detect optimum p, q* and select an 100%/0% ratio for Training and Validation sample. Press the *Process dashboard* button in the bottom right corner. TSL starts working and comes up with an optimum of $p = 2, q = 2$ with a constant included.

```
------------------------------ PARAMETER OPTIMIZATION ------------------------------

Model: DCS-g
The dependent variable Y is: Minimum
The selection sample is: 0622-01-01 - 1467-01-01 (N = 1, T = 846 with 0 missings)

Lower AIC found with value 2041.4941
Model specs: p = 0, q = 1, constant included

Lower AIC found with value 1876.851
Model specs: p = 1, q = 2, constant included

Lower AIC found with value 1876.6624
Model specs: p = 2, q = 2, constant included

----------------------------- TRAINING SAMPLE MODEL FIT ----------------------------

Model: TSL001 DCS-g(2,2)
variable: Minimum

                                              TSL001
Log likelihood                             -932.3312
Akaike Information Criterion (AIC)          1876.6624
Bias corrected AIC (AICc)                   1876.7626
Bayesian Information Criterion (BIC)        1905.1056
in-sample MSE                                  0.5313
... RMSE                                       0.7289
... MAE                                        0.5403
... MAPE                                       4.6854
Sample size                                       846
Effective sample size                             844
* based on one-step-ahead forecast errors
```

Continuing the modelling process we select only the *DCS-t* model in the score-driven column. We then tick the *Auto detect optimum p, q* box and press the *Process dashboard* button in the bottom right corner. After TSL is done finding the optimum number p, q we have the results

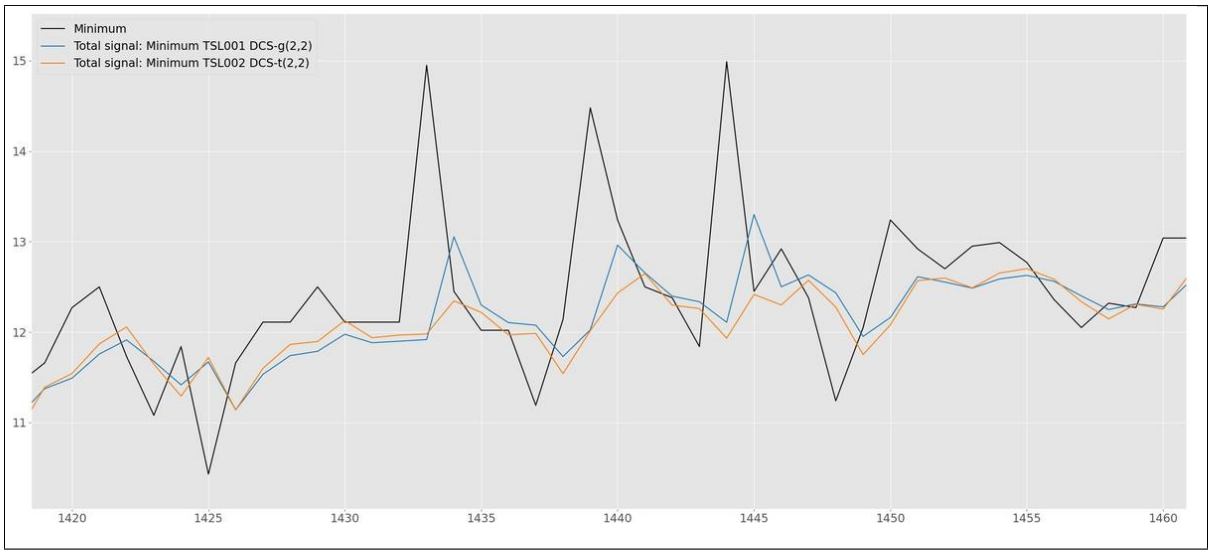
```
------------------------------ PARAMETER OPTIMIZATION -----------------------------

Model: DCS-t
The dependent variable Y is: Minimum
The selection sample is: 0622-01-01 - 1467-01-01 (N = 1, T = 846 with 0 missings)

Lower AIC found with value 2005.2239
Model specs: p = 0, q = 1, constant included

Lower AIC found with value 1801.8766
Model specs: p = 4, q = 2, constant included

Lower AIC found with value 1799.8932
Model specs: p = 3, q = 2, constant included

Lower AIC found with value 1798.3775
Model specs: p = 2, q = 2, constant included

--------------------------------- MODEL FIT ---------------------------------

Model: TSL002 DCS-t(2,2)
variable: Minimum

                                                TSL002
Log likelihood                               -892.1888
Akaike Information Criterion (AIC)           1798.3775
Bias corrected AIC (AICc)                    1798.5112
Bayesian Information Criterion (BIC)         1831.5612
in-sample MSE                                   0.5242
... RMSE                                        0.7240
... MAE                                         0.5315
... MAPE                                        4.5938
Sample size                                        846
Effective sample size                              844
* based on one-step-ahead forecast errors
```

The improvement in model fit is large. The likelihood improved 40 likelihood points and the AIC of the DCS-t(2,2) model is almost 100 points lower (better). In-sample measures MSE, RMSE, MAE, and MAPE are all better as well, albeit less dramatic. The student t model had one extra model parameter that needs to be estimated. This is the degrees of freedom and it is estimated at 4.8061 which shows that the tails of the distribution are much thicker than that of the Normal distribution meaning that the probability of extreme events becomes larger. Note that for degrees of freedom going to infinity, the DCS-t(p, q) model reverts to the DCS-g(p, q) model. In practice, the degrees of freedom do not need to go all the way to infinity. Degrees of freedom being estimated > 100 already closely resembles the DCS-g(p, q) model. The effect of thicker tails can be see in Figure 12.22 as well. This figure shows the extracted signal of the DCS-t(2, 2) and DCS-t(2, 2) model. We see that the

DCS-t(2, 2) reacts less strongly to the outliers in the data

Figure 12.22

Extracted signal of the DCS-t(2, 2) and DCS-t(2, 2) model



12.5.2 Two component model

Another way to model a long memory process is by using two components, one persistent component and one (less persistent) stationary component. We go to the Build your own model page of TSL and select a time-varying level (Random Walk). Do not select a slope component but do select an ARMA(1,0). Go to the Estimation page and click the Estimate button. The result is a model with a log likelihood value roughly similar to the DCS-g(2, 2) model and an extracted signal that shows a comparable pattern as well, see Figure 12.23.

We can also specifically deal with outliers and possible structural breaks in the data. Go to the Build your own model page and add Automatically find Intervention variables to the model. Go to the Estimation page and click the Estimate button. The result is an even better model fit than the one from the DCS-t(2, 2) model. We have:

```
Intervention coefficients:

Beta                              Value       Std.Err         t-stat          Prob
beta_outlier_0627-01-01           2.244        0.5456          4.112    4.3092e-05
beta_outlier_0646-01-01           2.772        0.5448          5.089    4.4709e-07
beta_outlier_0656-01-01           1.694        0.5448          3.110        0.0019
beta_outlier_0660-01-01           1.681        0.5448          3.085        0.0021
beta_outlier_0691-01-01          -2.202        0.5447         -4.043    5.7813e-05
beta_outlier_0713-01-01          -1.738        0.5447         -3.191        0.0015
beta_outlier_0719-01-01           1.750        0.5447          3.212        0.0014
beta_outlier_0809-01-01           3.502        0.5447          6.428    2.1893e-10
beta_outlier_0878-01-01           2.747        0.5447          5.043    5.6515e-07
```

```
beta_outlier_0962-01-01            1.889       0.5447       3.468    5.5135e-04
beta_outlier_0981-01-01           -1.650       0.5447      -3.029        0.0025
beta_outlier_1060-01-01            1.781       0.5447       3.269        0.0011
beta_outlier_1067-01-01            1.782       0.5447       3.271        0.0011
beta_outlier_1100-01-01            1.922       0.5447       3.528    4.4246e-04
beta_outlier_1292-01-01           -1.946       0.5447      -3.573    3.7329e-04
beta_outlier_1357-01-01            3.548       0.5447       6.513    1.2805e-10
beta_outlier_1409-01-01            1.812       0.5471       3.312    9.6656e-04
beta_outlier_1433-01-01            2.707       0.5448       4.968    8.2434e-07
beta_outlier_1439-01-01            1.897       0.5448       3.482    5.2474e-04
beta_outlier_1444-01-01            2.809       0.5448       5.156    3.1567e-07
beta_break_1397-01-01             -1.815       0.4063      -4.466    9.0860e-06
beta_break_1411-01-01              1.608       0.4073       3.947    8.5933e-05

--------------------------------- MODEL FIT ---------------------------------

Model: TSL004
variable: Minimum

                                                 TSL004
Log likelihood                                -779.5080
Akaike Information Criterion (AIC)            1613.0160
Bias corrected AIC (AICc)                     1614.8644
Bayesian Information Criterion (BIC)          1741.0100
in-sample MSE                                    0.5107
... RMSE                                         0.7146
... MAE                                          0.5235
... MAPE                                         4.5169
Sample size                                         846
Effective sample size                               823
* based on one-step-ahead forecast errors
```

TSL finds 20 outliers and 2 structural breaks, roughly 1 interventions every 38 year. The extracted signal of the two component model with interventions is roughly comparable to the DCS-t(2, 2) models as can be seen in Figure 12.23.

Figure 12.23

Extracted signal DCS and two component models

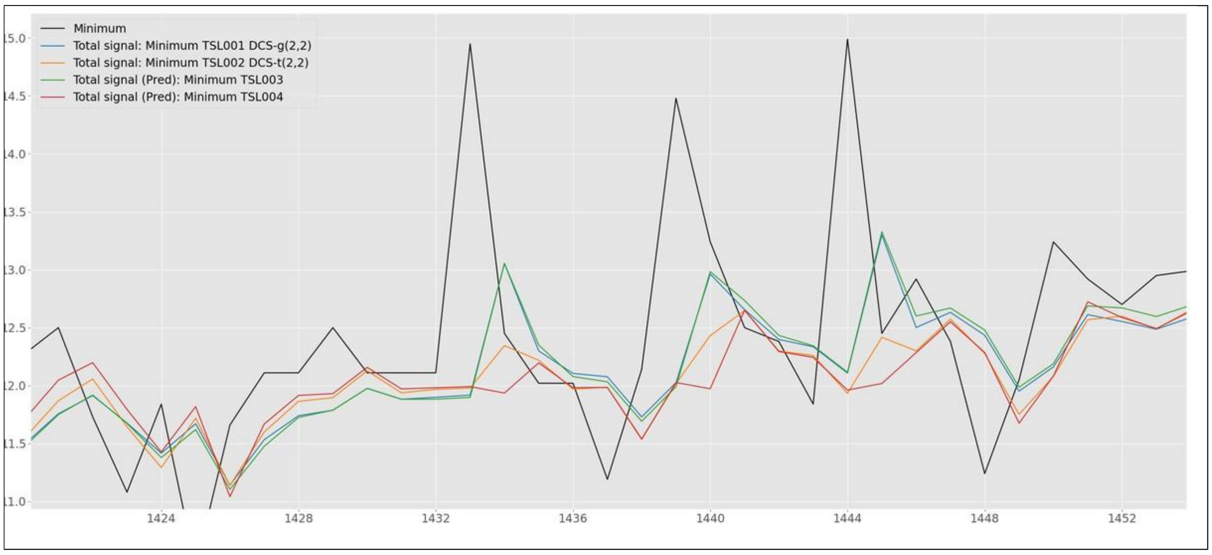


Figure 12.24

Structural break

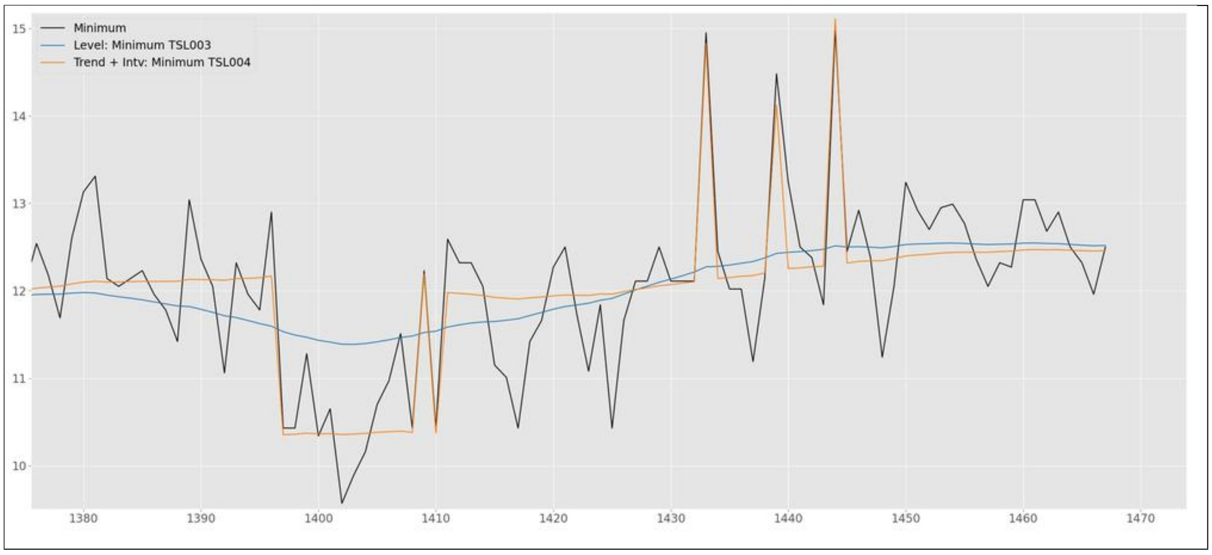
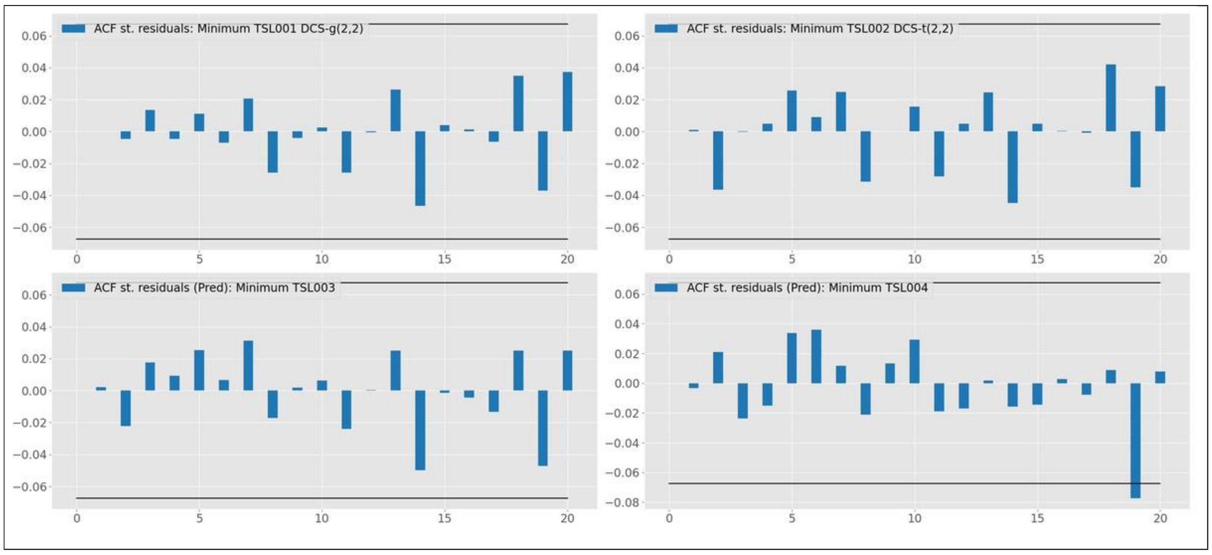


Figure 12.25

Autocorrelation functions



12.6 US meat production

This case study is on US meat production. Load the dataset *us_meat_production.csv* from the data folder located in the install folder of TSL. Select and plot the *Pork* time series. We see monthly production of pork in the United States in millions of pounds. At first glance, this dataset should not give us too many problems with a clear upward trend and a monthly seasonal. However, this time series turns out to hide some very interesting dynamics which we will uncover in this case study.

12.6.1 Basic structural time series model

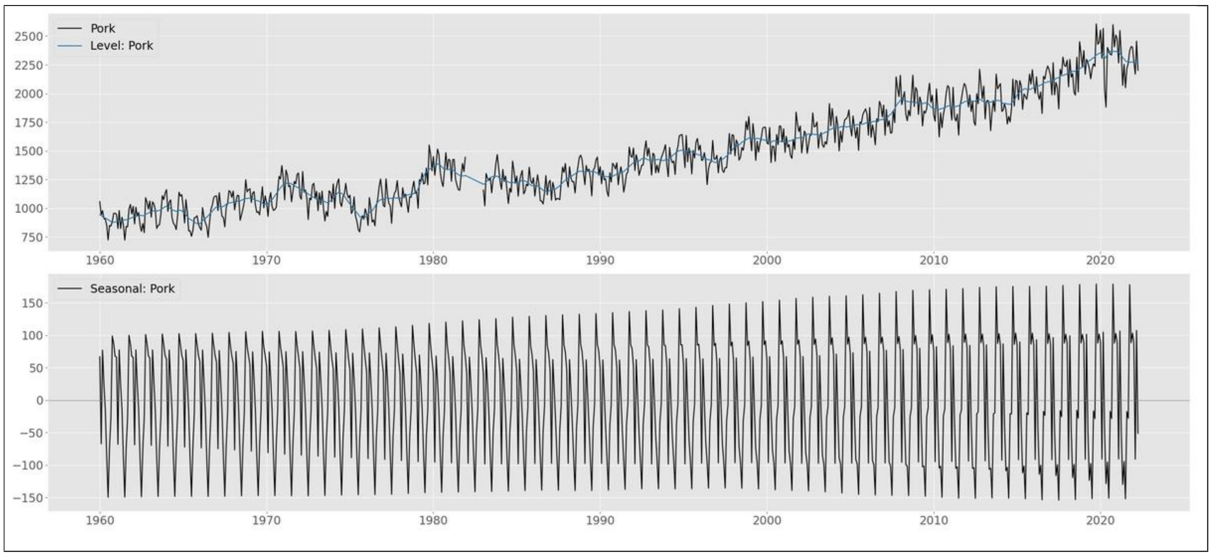
We begin with the Basic structural time series model consisting of a time-varying level, slope, and monthly seasonal (s=12). Select these components on the Build your own model page and Estimate the model by clicking the Estimate button on the Estimation page. The model fit is:

```
--------------------------------- MODEL FIT ---------------------------------

Model: TSL001
variable: Pork

                                                   TSL001
Log likelihood                                  -4187.358
Akaike Information Criterion (AIC)               8406.715
Bias corrected AIC (AICc)                        8407.472
Bayesian Information Criterion (BIC)             8480.335
in-sample MSE                                    6005.796
... RMSE                                           77.497
... MAE                                            60.766
... MAPE                                            4.357
Sample size                                           748
Effective sample size                                 735
* based on one-step-ahead forecast errors
```

The graphical output is presented in Figure 12.26. We see a time-varying seasonal pattern where the difference between months increase over time. Without investigating further we could accept this model but the ACF of the *predicted standardized residuals* (top right panel of Figure 12.27) shows something interesting and worrying at the same time. The ACF shows strong signs of residual autocorrelation and it is periodic with a period of 3, often followed by a negative correlation spike. The spectral density (bottom right panel of Figure 12.27) confirms the periodicity with a peak at $2.0/0.7 \approx 2.86$. What is going on here? Apparently there is strong periodicity left in the residuals.
It could be that the periodic behaviour has something to do with reporting rules. A period of ≈ 2.86 means at the end of a quarter. Could it be that numbers need to be reported

Figure 12.26

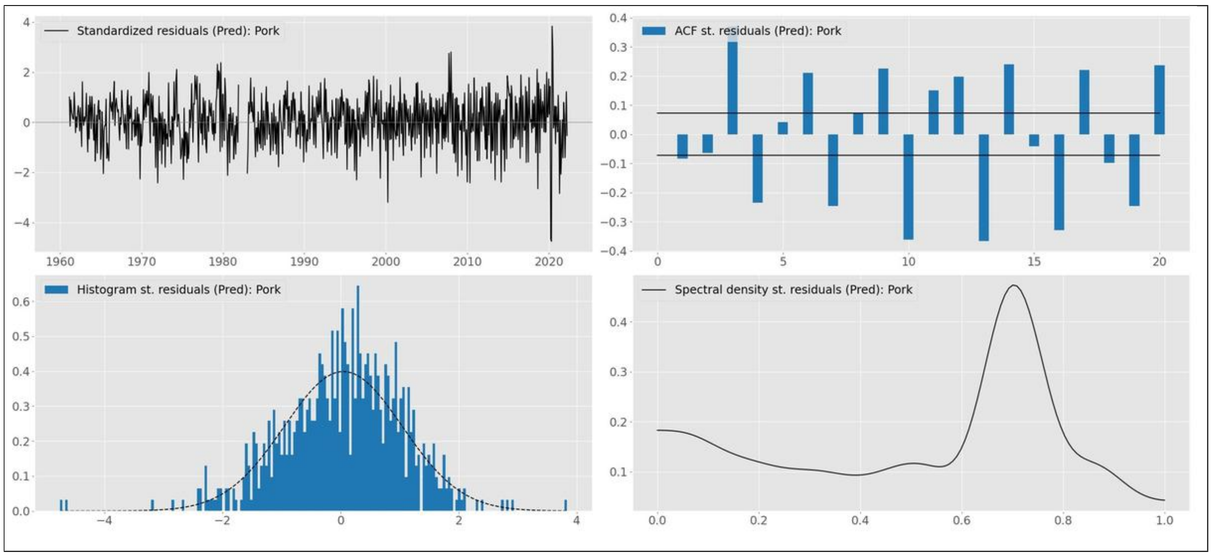
Extracted Level and Seasonal



at the end of each quarter and numbers of the first and second month within a quarter are sometimes reported in the third month?

Figure 12.27

Graphical diagnostics of standardized residuals



12.6.2 Adding a cycle component

If we add a cycle to our model, the model fit strongly improves. The output of TSL is:

```
---------------------------- PARAMETER SUMMARY ----------------------------
```

```
Cycle properties:

Parameter type                   Cycle 1
Variance                       1368.1797
Period                            2.8716
Frequency                         2.1880
Damping factor                    0.9999
Amplitude                        63.0507

--------------------------------- MODEL FIT ---------------------------------

Model: TSL002
variable: Pork

                                                  TSL002
Log likelihood                                 -4031.384
Akaike Information Criterion (AIC)              8100.767
Bias corrected AIC (AICc)                       8101.829
Bayesian Information Criterion (BIC)            8188.190
in-sample MSE                                   3915.803
... RMSE                                          62.576
... MAE                                           47.334
... MAPE                                           3.440
Sample size                                          748
Effective sample size                                735
* based on one-step-ahead forecast errors
```

Notice that the period of the cycle is estimated at 2.87 which is close to the information contained in the spectral density with a period of $2.0/0.7 \approx 2.86$. The graphical output is shown in Figure 12.28. Again the series shows something interesting. The Extracted cycle itself shows periodic behaviour with a cycle within the cycle.

12.6.3 Quarterly data

Another way of dealing with the large spikes in the autocorrelation plot is by aggregating the data. The series can be converted from a monthly series into a quarterly one by summing up the data every three months. Currently this cannot be done in TSL itself but let us know if this is a feature you want to see!
For now we used Excel to aggregate the data. The output of modelling the resulting quarterly series with a time-varing level, slope, and seasonal (s=4) is presented in Figure 12.29. The figure shows us that the ACF of the predicted residuals does not show significant autocorrelation.

Figure 12.28
Extracted Level, Seasonal, and Cycle

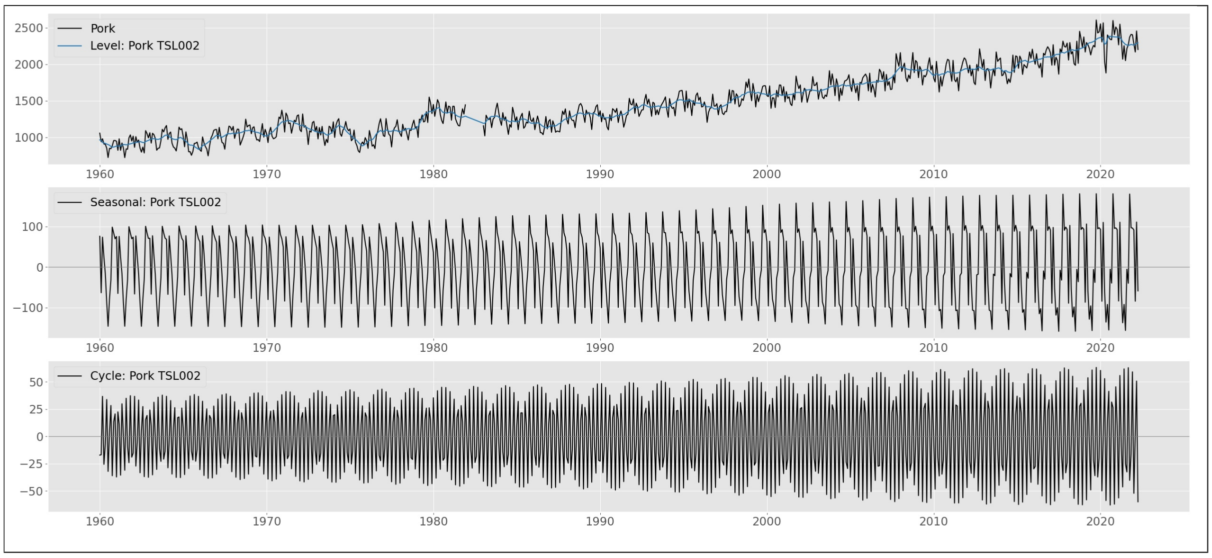
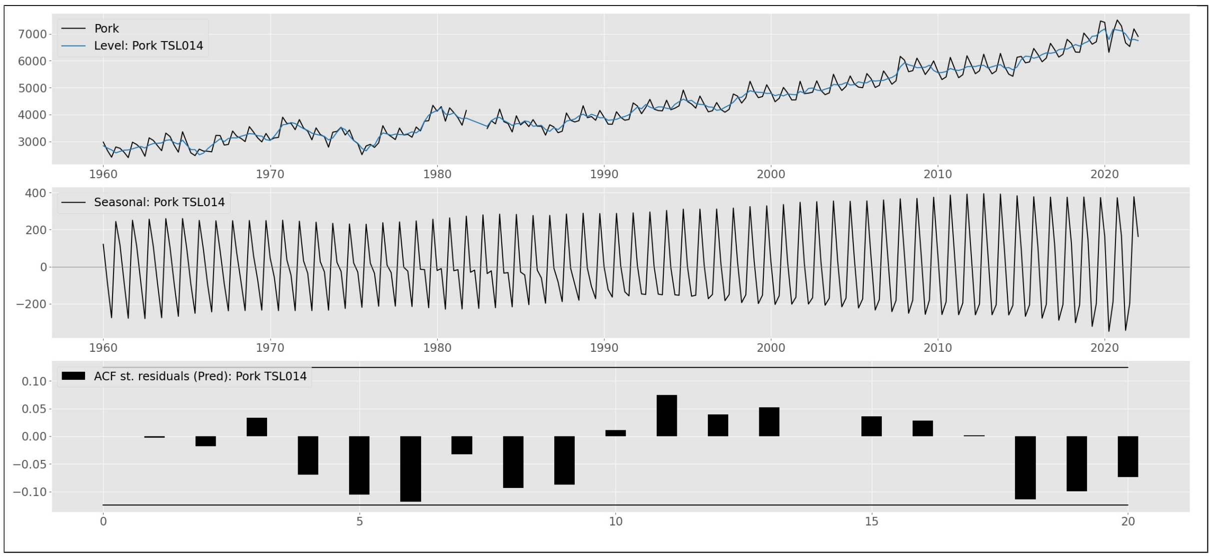


Figure 12.29
Extracted Level and Seasonal for quarterly series



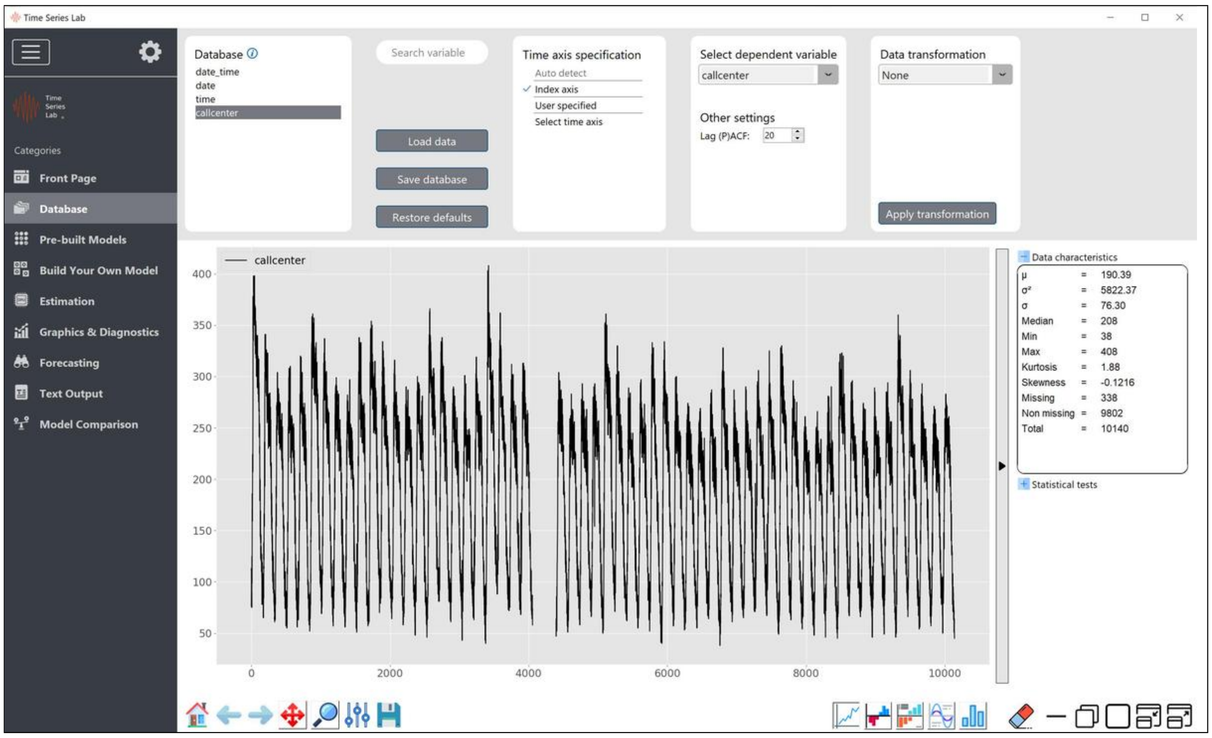
12.6.4 Further exploration

- Verify that in the model with the cycle, if the cycle period is fixed at exactly 3 months, the model fit is much worse compared to the model with the estimated cycle period of 2.87.
- Verify that the ACF of the standardized residuals of the model with the cycle is much better compared to the model without a cycle but that we still see correlation spikes at some lags.
- Show that the model fit of the model with the cycle can be further improved by letting TSL search for outliers and structural breaks.

12.7 Call center

In this case study we analyse call center data to illustrate how to use TSL to model complex seasonal patterns. The series is used in the second application of the *TBATS* paper of De Livera et al. (2011) and it can be downloaded from here. The call center time series consists of 10,140 observations on call arrivals per 5-minute interval between 7:00 AM and 9:00 PM on weekdays. The series contains a daily seasonal pattern with period 169 (number of 5-minute intervals between 7:00 AM and 9:00 PM) and a weekly seasonal pattern with period 169 $\times$ 5 weekdays $=$ 845. Just as in De Livera et al. (2011), we use 7605 observations (9 weeks) for our training sample which leaves 2535 observations (3 weeks) to analyse forecasting performance. Note that in contrast to the data as shown in figure 1b and 5 of De Livera et al. (2011), the data set we downloaded has two days of missing values (04/04/2003 and 07/04/2003), see Figure 12.30. On further inspection of figure 5 of De Livera et al. (2011), we see that the corresponding days in their figure have a more smooth pattern than the rest of the data so they might have used some fill-in values for the missing values. In TSL there is no need for this. Missing values are part of time series analysis, see also Section 12.1.4.

Figure 12.30
Call center data set with two days of missing values



Select dependent variable: callcenter

Lag (P)ACF: 20

Data transformation: None

Data characteristics

```
μ            =   190.39
σ²           =   5822.37
σ            =   76.30
Median       =   208
Min          =   38
Max          =   408
Kurtosis     =   1.88
Skewness     =   -0.1216
Missing      =   338
Non missing  =   9802
Total        =   10140
```

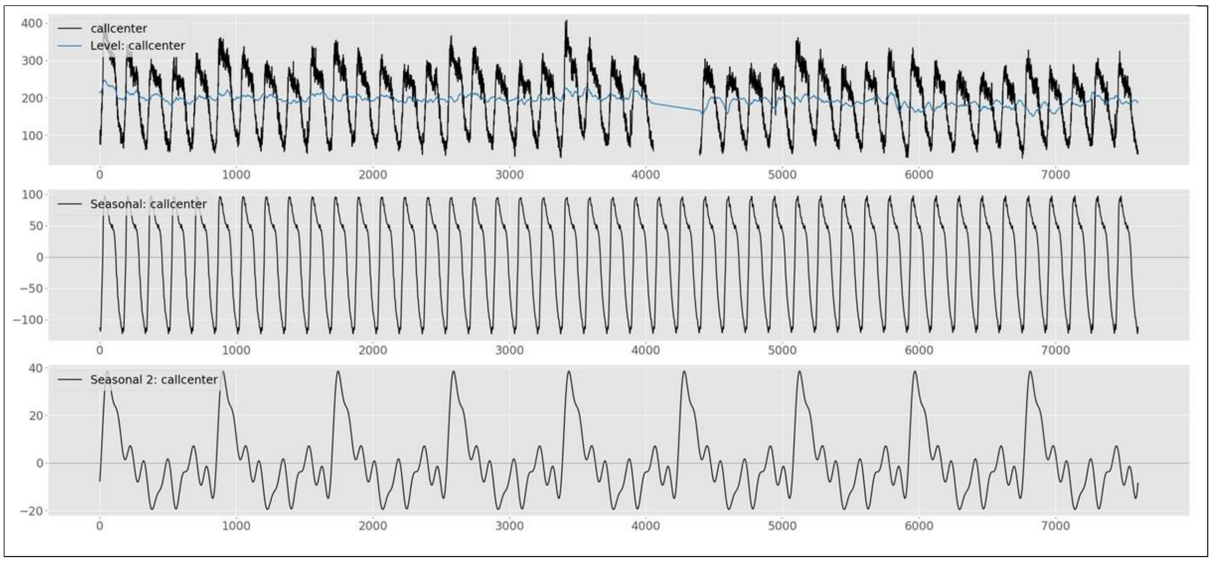
12.7.1 Building the model

We select a time-varying level, time-varying seasonal 1 with a period of 169 and 20 factors, and a time-varying seasonal 2 with a period of 845 and 10 factors.

Important: The number of factors, in combination with the time series length, strongly influences estimation times and higher numbers seldom lead to better forecasts, Therefore, it is strongly advised to not choose the number of factors too high and it is almost never needed to go beyond 40.

Select a training sample of 7605 on the Estimation page and estimate the model. After TSL is done estimating, the graph page shows us Figure 12.31: From Figure 12.31 we see that

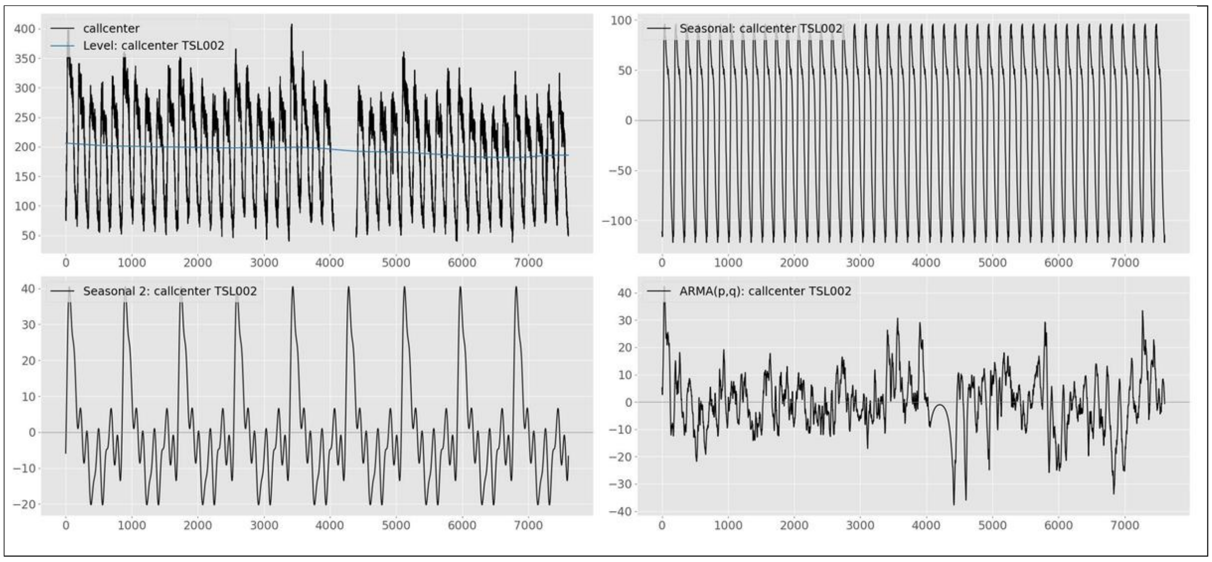
Figure 12.31
Extracted level and two seasonals



TSL has no problem with the missing data. The Kalman Filter / Smoother algorithm nicely interpolates all the selected components. Furthermore, we see from the top panel that the Level is not smooth and it looks like the level itself picks up some dynamics. This is confirmed by the ACF of the predicted residuals which shows significant first order autocorrelation among other lags. Before we fix this, let's first make some forecasts to compare this model to the following models. Go to the *model comparison* page and click the *start loss calculation* button in the top right corner.

Our next modelling step is to add an ARMA(1,0) process to the existing model to do something about the first order autocorrelation that is still present in the residuals. Select the ARIMA(1,0) component on the Build your own model page, leave everything else the same, and estimate the model. The result should be like Figure 12.32. We learn from Figure 12.32 that the level is much smoother. If we look at the ACF of the predicted residuals we see that

Figure 12.32
Extracted level and two seasonals + ARMA(1,0) errors



the first order autocorrelation is still present. Again, let's make some forecasts to compare this model to the first and the following models. Go to the *model comparison* page and click the *start loss calculation* button in the top right corner.

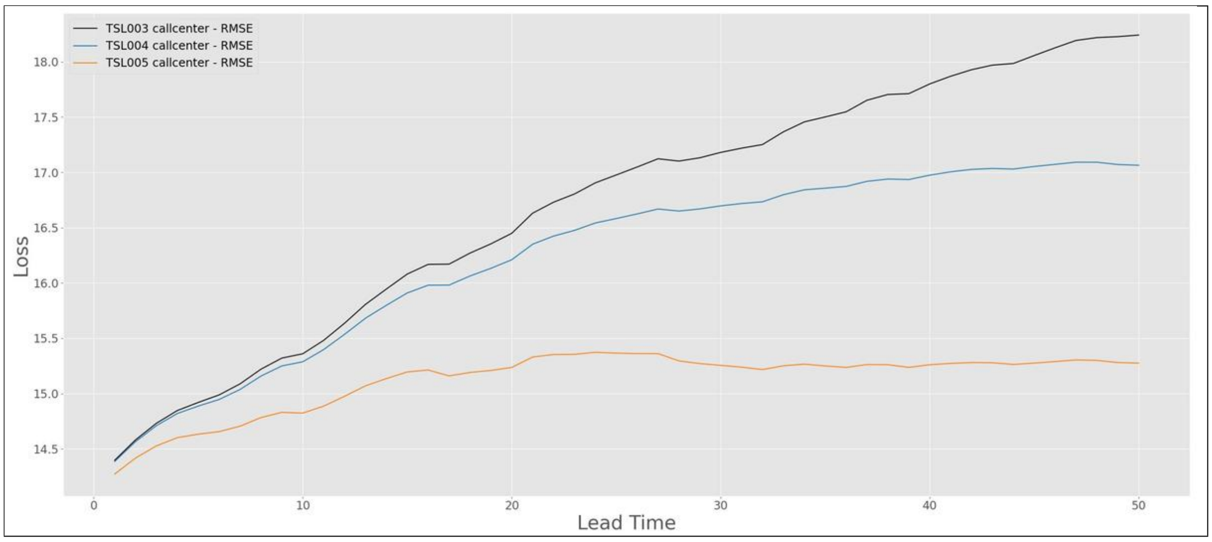
12.7.2 Seasonal variance extension

The default is to have one variance per seasonal component for all seasonal factors. In some situations, this is somewhat restrictive and estimating additional seasonal variances can improve model fit and forecast performance. However, there are an extremely large number of factor combinations and machine learning needs to assist here since we cannot try all combinations. TSL has a machine learning method that determines which seasonal factor gets its own variance parameter. To see this in action, go the top menu bar and click on *Advanced settings* and switch on *Seasonal 1 variance extension*. Go to the Estimation page and make sure no parameters are set to fixed. Click on Estimate and wait till the algorithm is finished. This takes some time with an extensive model like this. After the estimation is completed, go to the Model comparison page and start the loss calculation for this model as well. After that is completed you should see three check boxes in the top left corner. When all are checked the resulting figure should look like the one in Figure 12.33. The lowest loss line belongs to the last estimated model and is at least as good as the loss obtained from the TBATS package as presented in De Livera et al. (2011).

Keep in mind that the analysis performed by TSL in this case study is based on the call center time series *with* missing values.

Figure 12.33

Model comparison based on forecast performance



12.8 Regression with ARMA errors

TSL makes extensive use of models in state space form due to the many advantages this brings. Once a model has been put in a state space form, the way is opened for the application of a number of important algorithms. At the centre of these is the Kalman filter. The Kalman filter is a recursive procedure for computing the optimal estimator of the state vector at time t, based on the information available at time t, see also Harvey (1990).

The dynamics of state space models come from stochastic components, see also Appendix B. If the components are deterministic (meaning the error terms of the state components have variance zero and therefore disappear from the equation), the Kalman filter would be equivalent to the ordinary least square (OLS) recursions. This means that if you would only include explanatory variables in TSL and no time-varying components you would be estimating a standard regression model as given by

$$y_i = \alpha + X_i\beta + \varepsilon_i, \qquad i = 1, \dots, n. \tag{12.1}$$

The estimates of β, denoted by $\hat{\beta}$, are quickly found with the equation

$$\hat{\beta} = (X'X)^{-1}X'y \tag{12.2}$$

and you normally would not use the Kalman filter to find $\hat{\beta}$. However, the results from the Kalman filter should be exactly the same as the $\hat{\beta}$ from above and it is illustrative to see the results from the static regression model in TSL. These results will later be extended with ARMA(p,q) errors.

12.8.1 Regression model in TSL

Load the El Nino dataset which can be found in the data folder located in the install folder of TSL. Select the EN3.4 series from the loaded data set. Go to the Build you own model page and switch-on the *Explanatory variables* and select all variables except the *Date* variable from the pop-up window. If you need to include a constant in the regression model, you can add a column of ones to the dataset but more convenient is just to add a *fixed* Level component to the model. A *fixed* Level component is in this scenario exactly the same as the constant α in the standard regression model of (12.1). Go to the Estimation page and estimate the model. The result should be:

```
Regression coefficients:

Beta                        Value       Std.Err        t-stat          Prob
beta_RB                    0.1591        0.0510        3.1229        0.0019
beta_WPAC                 -0.1293        0.1998       -0.6470        0.5180
beta_WPAC2                 0.9271        0.2026        4.5751    6.4450e-06
beta_WPAC3                -0.1793        0.1903       -0.9419        0.3468
```

```
beta_WPAC4                 -0.3629        0.1686       -2.1525        0.0320
beta_50fin                  0.5637        0.2164        2.6046        0.0096
beta_100cold               -0.0055        0.0256       -0.2166        0.8286
beta_100fin1               -0.1267        0.0915       -1.3848        0.1669
beta_100fin2               -0.4103        0.0824       -4.9782    9.7295e-07
beta_150fin1               -0.2586        0.1310       -1.9739        0.0491
beta_150fin2                0.1432        0.0693        2.0656        0.0395
beta_200fin1                0.2533        0.1456        1.7404        0.0826
beta_200fin2               -0.0347        0.1005       -0.3454        0.7300
beta_250fin1                0.2143        0.1494        1.4346        0.1522
beta_250fin2               -0.0651        0.1389       -0.4688        0.6395
beta_300fin1               -0.1427        0.2210       -0.6459        0.5188
beta_300fin2               -0.1130        0.2308       -0.4894        0.6248
beta_400fin1               -0.3206        0.2428       -1.3203        0.1875
beta_400fin2                0.1122        0.2679        0.4186        0.6758
beta_500fin1               -0.6584        0.2839       -2.3193        0.0209
beta_500fin2               -0.3397        0.2786       -1.2194        0.2235
beta_wnd160.200_0.10      -33.6202        2.8739      -11.6985        0.0000
beta_wnd180.220_-4.4       86.0116        5.3721       16.0107        0.0000
beta_wnd180.210_-10.0     -16.4366        4.5228       -3.6341    3.1704e-04


State vector at period 2015-11-01:

Component                    Value       Std.Err        t-stat          Prob
Level                        28.02         3.808         7.358    1.1546e-12
```

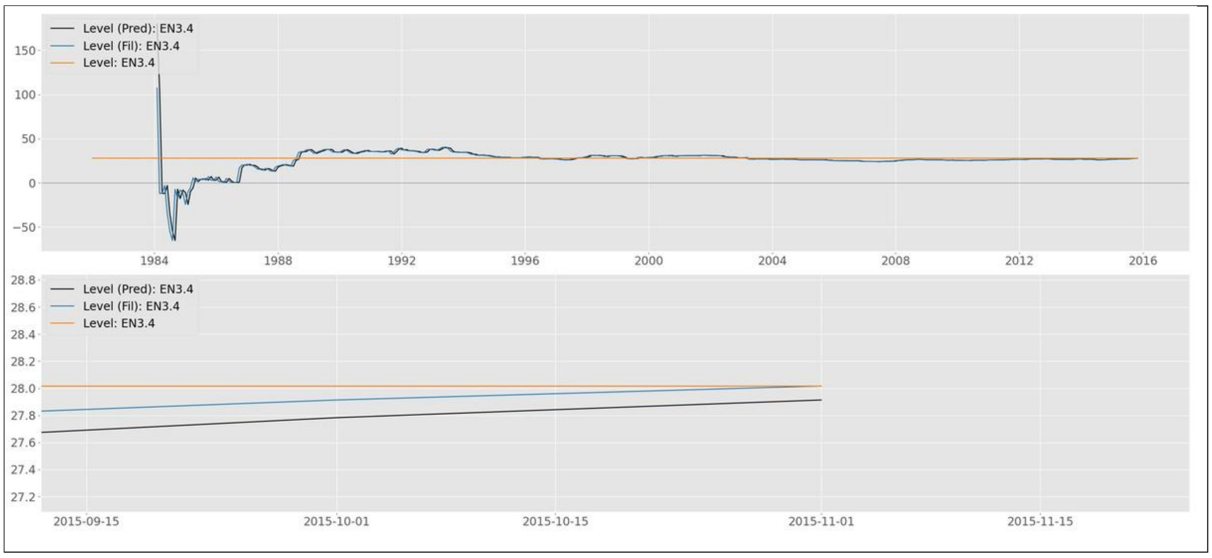
which is exactly equal to the OLS estimate $\hat{\beta}$ as we would calculate it from 12.2.

But what does *Predicting*, *Filtering*, and *Smoothing* mean in the case of the regression model we just estimated? Remember that, being in time point t, *Predicting* uses the data up to time $t-1$, *Filtering* the data up to time t, and *Smoothing* uses all the data. If we would plot the fixed level (constant α) for *Predicting*, *Filtering*, and *Smoothing* we see that Smoothing gives a straight line while *Predicting*, *Filtering* build the level up over time to the end of the data set, see also Figure 12.34. With the above logic, the estimates for *Filtering* and *Smoothing* should be the same at time $t = T$ when all data is used. If we look at the bottom panel of Figure 12.34 we see that this is indeed the case.

If you would like to end up with a set of only significant variables, based on a user-specified t-stat bound, you can select the *Automatically* option for the explanatory variables on the Build your own model page. Estimating the model leads to the following estimates.

```
Regression coefficients:

Beta                         Value       Std.Err        t-stat          Prob
beta_RB                     0.1692        0.0474         3.570    4.0098e-04
beta_WPAC2                  0.9083        0.1476         6.153    1.8695e-09
beta_WPAC4                 -0.5950        0.1118        -5.322    1.7235e-07
```

Figure 12.34
Predicted, Filtered, and Smoothed constant in regression model



```
beta_50fin                    0.4916       0.1719        2.860       0.0045
beta_100fin1                 -0.1781       0.0533       -3.344   9.0407e-04
beta_100fin2                 -0.3515       0.0643       -5.466   8.1630e-08
beta_150fin2                  0.0894       0.0320        2.795       0.0054
beta_500fin1                 -0.9462       0.2136       -4.430   1.2228e-05
beta_wnd160.200_0.10        -35.4201       2.2560      -15.701       0.0000
beta_wnd180.220_-4.4         92.1050       4.6275       19.904       0.0000
beta_wnd180.210_-10.0       -20.3677       3.8036       -5.355   1.4566e-07


State vector at period 2015-11-01:

Component                      Value      Std.Err       t-stat         Prob
Level                          24.19        2.799        8.640   2.2204e-16
```

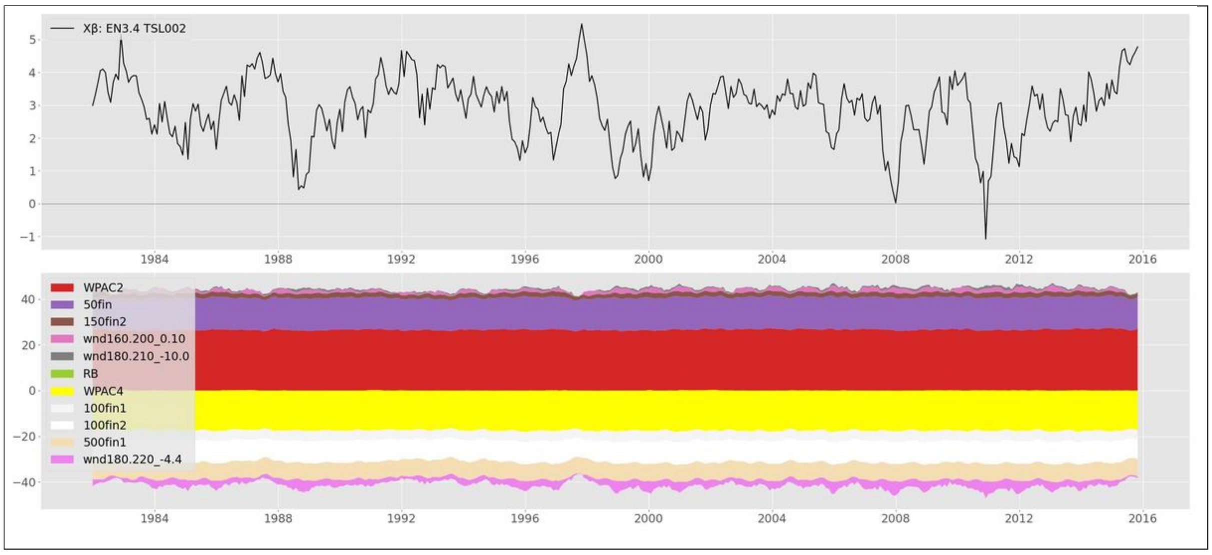
from which we can see that all variables are significant with an absolute t-stat of at least 2.795.

Figure 12.35 shows the contribution of all X's combined $(X\hat{\beta})$ in the top panel and the individual contributions of the X's in a sandgraph in the bottom panel.

12.8.2 Regression model with ARMA(p,q) errors

The ACF plot of the predicted residuals shows that there is first and second lag autocorrelation left in the residuals. We can combat this by introducing ARMA(p,q) errors in the model. Select an additional ARMA(2,1) model from the Build your own model page, select all variables except the *Date* variable, set Explanatory variables to automatic and Estimate the model. The output is:

Figure 12.35
Contribution of all significant X's in a Sandgraph



```
Variance of disturbances:

Variance type                      Value        q-ratio
Level variance                    0.0000              0
ARMA variance                     0.0769              1


ARMA properties:

Parameter type                     Value
Unconditional variance            1.0645
AR2 phi1                          1.6105
AR2 phi2                         -0.7024
MA1 theta1                       -0.1509


Regression coefficients:

Beta                             Value         Std.Err          t-stat           Prob
beta_WPAC3                      0.6416          0.1025           6.258     1.0117e-09
beta_WPAC4                     -0.2615          0.0889          -2.941         0.0035
beta_150fin2                   -0.1771          0.0412          -4.304     2.1204e-05
beta_200fin1                    0.1417          0.0423           3.345     9.0050e-04
beta_250fin2                    0.3072          0.0981           3.133         0.0019
beta_wnd160.200_0.10           -5.5813          1.4499          -3.849     1.3797e-04
beta_wnd180.220_-4.4           11.9443          2.6044           4.586     6.0621e-06
beta_wnd180.210_-10.0          -6.6402          2.0543          -3.232         0.0013
```

```
State vector at period 2015-11-01:

Component                          Value        Std.Err        t-stat          Prob
Level                             14.058         1.5445         9.102             0
ARMA(p,q)                          1.990         0.1953        10.189             0
```

The ACF show no residual correlation in the first lags but the 12th lag has a large spike. This is due to the missing of a seasonal component. Additional measures can be lagged explanatory variables or a monthly seasonal component.

12.9 Smooth trend

The data for this case study is the HadCRUT5 annual data on global sea and land temperature. HadCRUT is a dataset of monthly instrumental temperature records formed by combining the sea surface temperature records compiled by the Hadley Centre of the UK Met Office and the land surface air temperature records compiled by the Climatic Research Unit (CRU) of the University of East Anglia. The time series are presented as temperature anomalies (deg C) relative to 1961-1990. The data can be found here.

The data can also be found in the data folder located in the install folder of TSL under the name *global_temp.csv*. Loading and plotting the data in TSL shows an upward trend in the data starting in the 1980's so we clearly need a slope component in our model. Furthermore, the autocorrelation function shows clear signs of long memory.

12.9.1 Integrated Random Walk

Select a time-varying level and time-varying slope on the Build your own model page. Go to the Estimation page and set the end of the Training sample to 141 (1990-01-01). Estimate the model and when TSL is done, go to the *Forecast* page. Under *Plot options* set the forecast to 32 periods ahead and select *multi-step-ahead* forecast. Verify that the *multi-step-ahead* forecasts are bad. So how to improve?

Go to the Build your own model page and select a *fixed* level and time-varying slope. The corresponding model is called an *Integrated Random Walk* model and the result is a model with a much smoother trend. Estimate the model and go to the *Forecast* page. Verify that the *multi-step-ahead* forecasts are already much better and all, except two, data points lie within $\pm$ 1 standard error. Going back to the *Graph* page and plotting the ACF for the *Predicted* residuals reveals first-lag autocorrelation.

Add an ARMA(1,0) component to our latest model and estimate the model. The result is an increase in log likelihood, no significant autocorrelation in the *Predicted* residuals and an even better forecasting performance.

The training sample results should like Figure 12.36, The test sample results should like Figure 12.37, and the Forecasting performance of all three models that we estimated are compared in Figure 12.38.

Figure 12.36
Temperature data with Integrated Random Walk

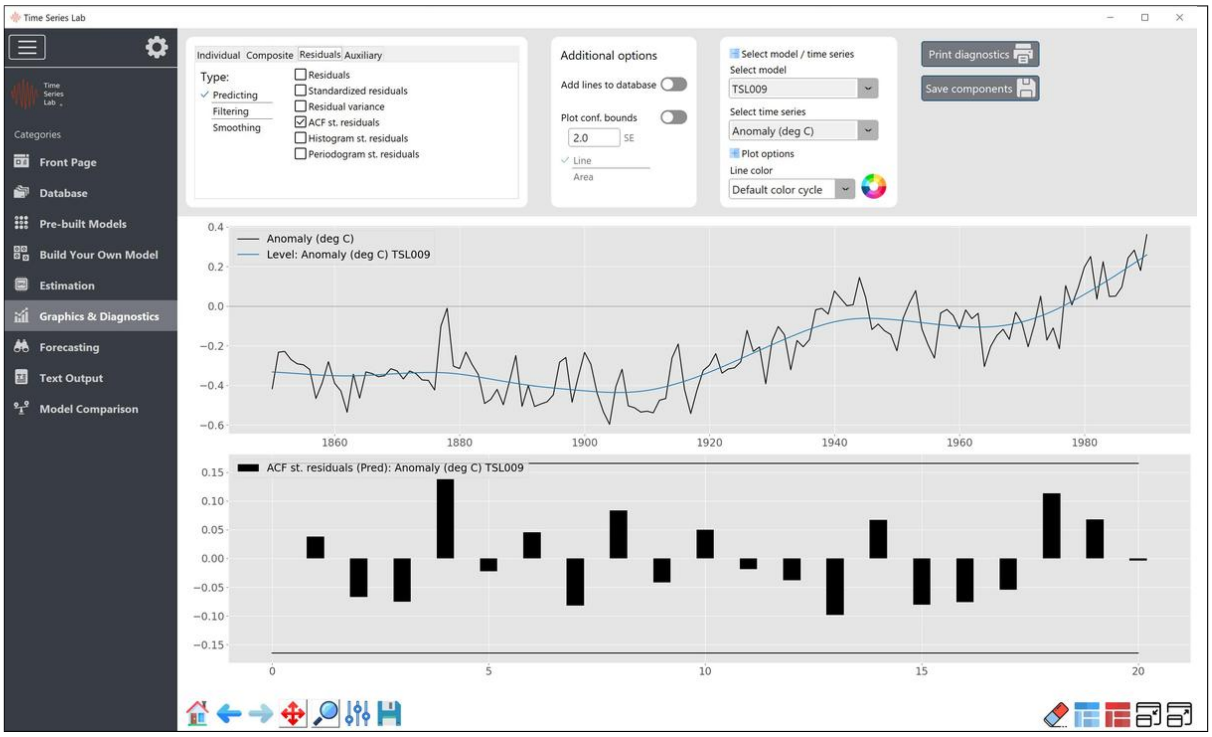


Figure 12.37
32-step-ahead forecast for Temperature data

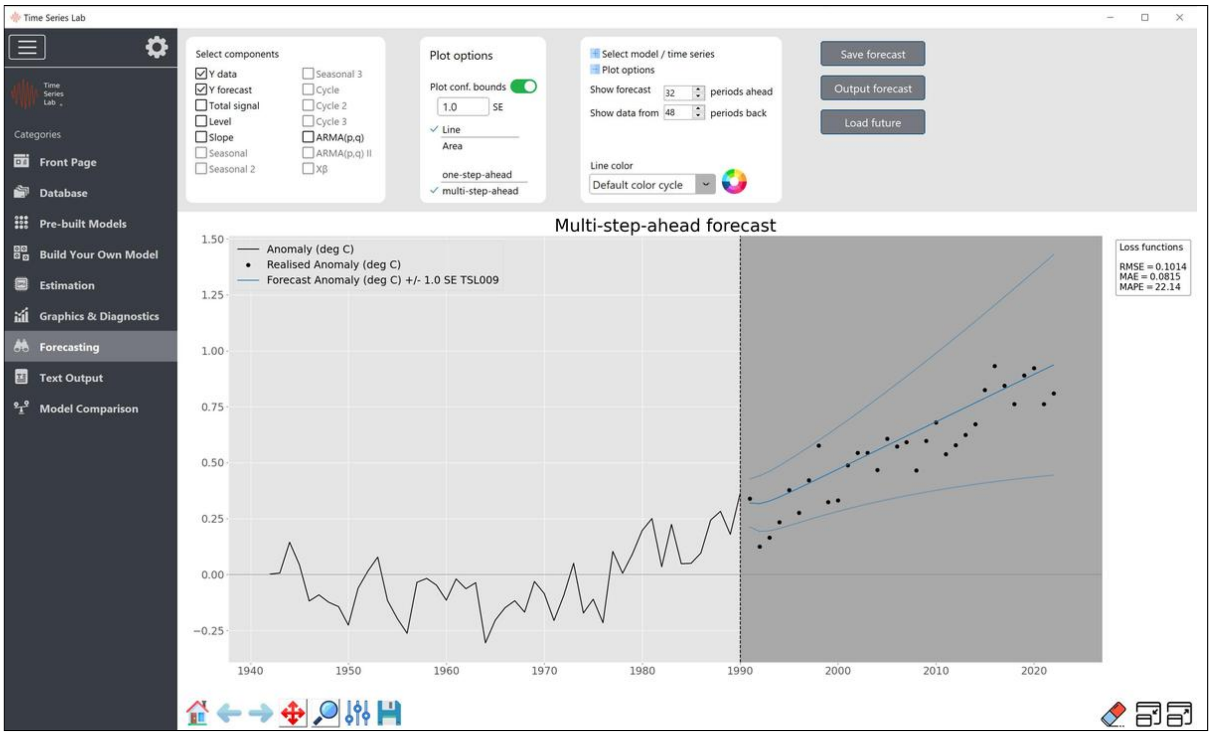
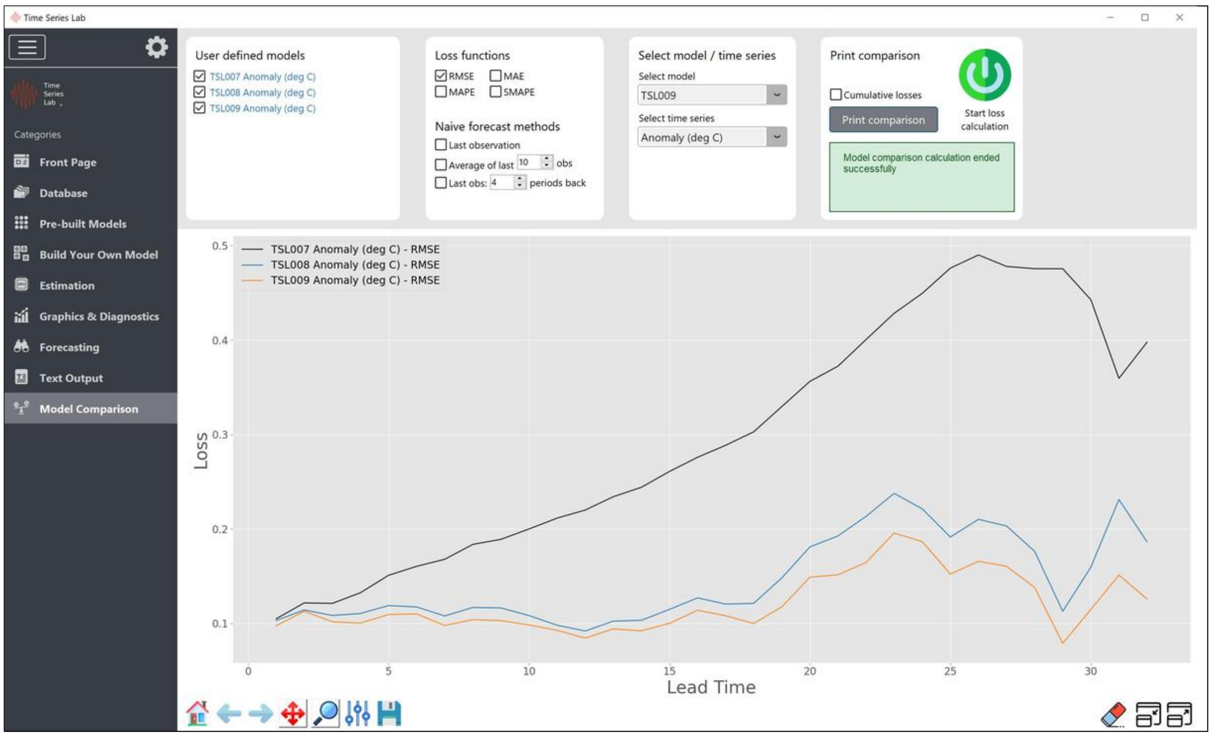


Figure 12.38

Forecasting performance for three models



User defined models
☑ TSL007 Anomaly (deg C)
☑ TSL008 Anomaly (deg C)
☑ TSL009 Anomaly (deg C)

Loss functions
☑ RMSE ☐ MAE
☐ MAPE ☐ SMAPE

Naive forecast methods
☐ Last observation
☐ Average of last 10 obs
☐ Last obs: 4 periods back

Select model: TSL009
Select time series: Anomaly (deg C)

☐ Cumulative losses

Model comparison calculation ended successfully

TSL007 Anomaly (deg C) - RMSE
TSL008 Anomaly (deg C) - RMSE
TSL009 Anomaly (deg C) - RMSE

12.10 Electricity consumption

This case study is on hourly electricity consumption in megawatts. The data can be found here or in the TSL data folder under the name *hourly_elec.csv*. The time series starts at 01/01/2005 00:00:00 and ends at 31/12/2017 23:00:00. The time series is challenging in several ways. First, the series is very long with 113,952 observations. Second, the series has multiple seasonal patterns with an intraday hourly pattern with a period of 24, a day-of-the-week pattern with a period of $24 \times 7 = 168$, and an annual seasonal pattern with a period of $24 \times 365.25 = 8766$ (taking leap years into account). We use a training sample consisting of 105,192 observations that ranges from 01/01/2005 00:00:00 to 31/12/2016 23:00:00. The validation sample is the last year of the data set and consists of 8760 observations. With long time series like these, in combination with several dynamic components, estimation times can become relatively long. The goal of this case study is to find a good model and estimate the model parameters. Once these parameters are estimated and the forecasting performance of the model is satisfactory, in general the model parameters do not need to be estimated again for a while and forecasts are obtained quickly. The reason that model parameters do not need to be estimated again is because in most cases, adding data to an already long time series does not change parameter estimates by much.

12.10.1 Local level model

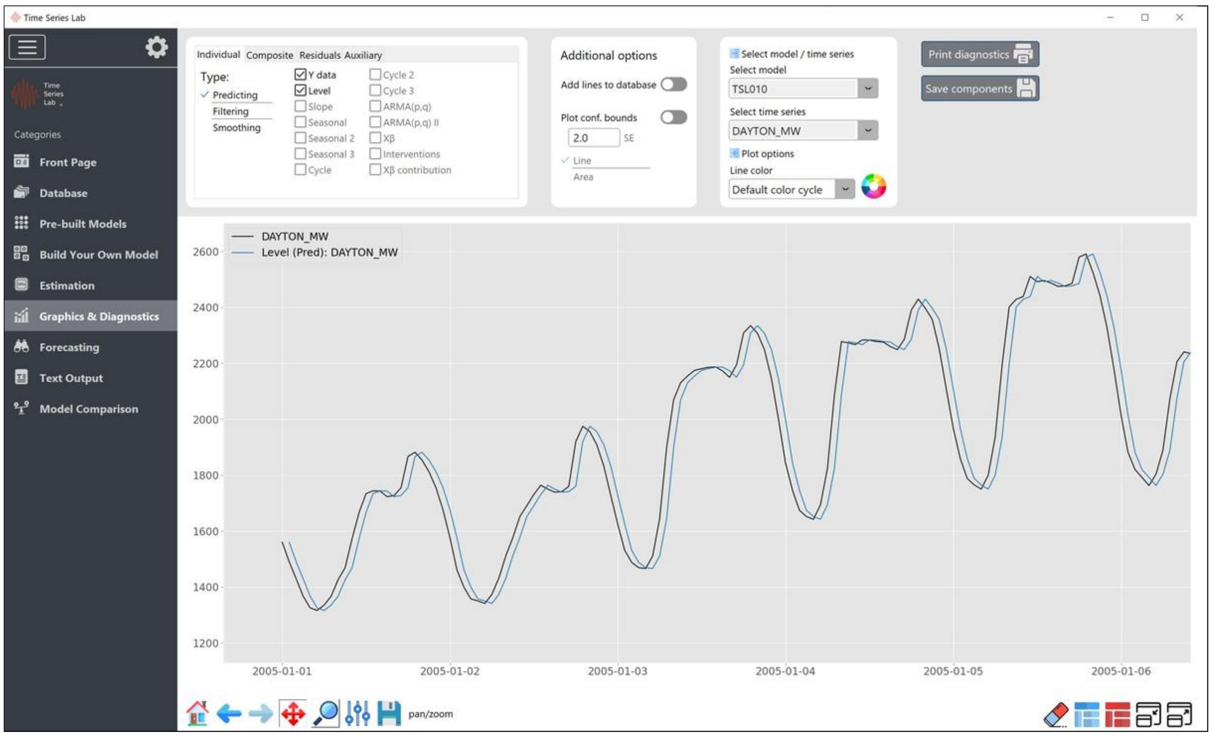
To set a benchmark, we model the time series with a Local Level model. Select a time-varying level on the Build your own model page, set the end of the training sample to 105,192 on the Estimation page and Estimate the model. TSL shows us the following estimates for the variances:

```
Variance of disturbances:

Variance type                     Value        q-ratio
Level variance                 7641.628         1.0000
Irregular variance                7.548     9.8772e-04
```

We see that the variance of the Level is very high compared to the variance of the Irregular. This means that the specified Level is not informative for the data, something not surprising since we know that several other important dynamics are not yet accounted for. No clear signal is found in the data and the best forecast for the next time period is just the last observation. The result of such a model is that the *Predicted* Level dutifully follows the data but the predictive power of the model is low, something we will see later. Zooming in on the beginning of the training sample we see the *Predicted level* following the data and lagging by one time point in Figure 12.39

Start a loss calculation on the Model comparison page to set a forecasting performance benchmark.

Figure 12.39
Path of Local Level model



12.10.2 Daily seasonal with a period of 24

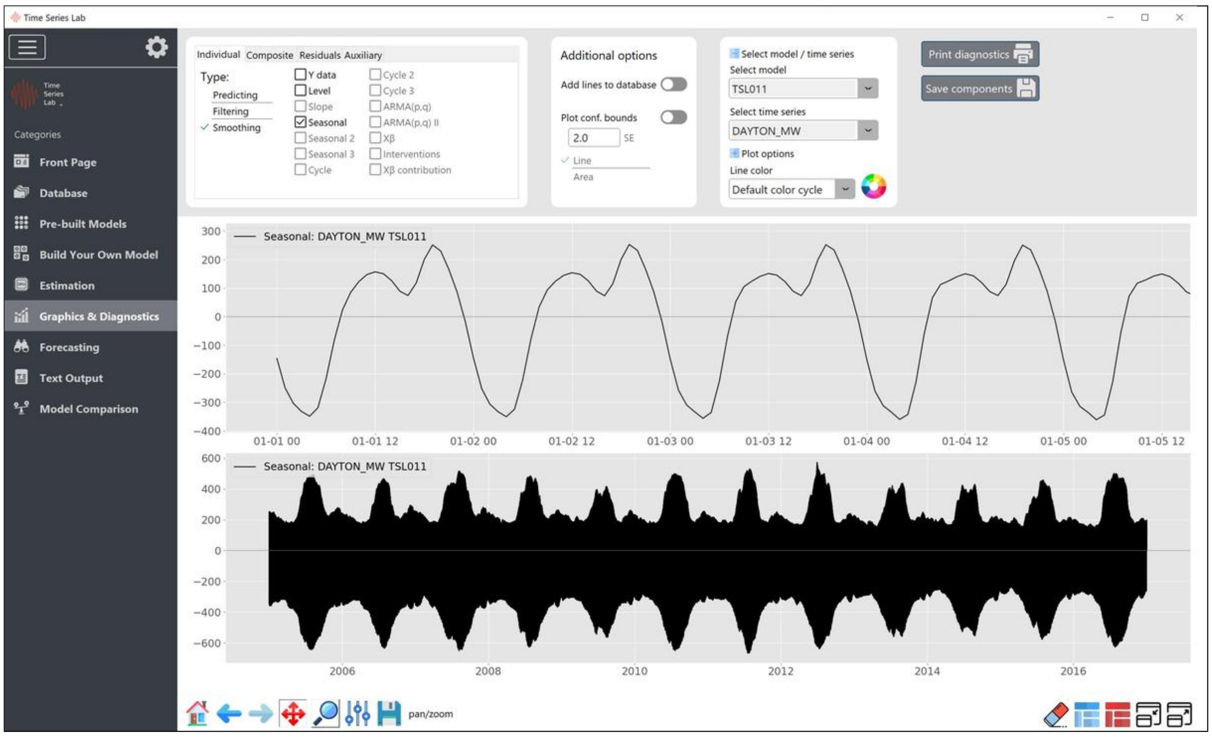
Our next task is to model the intraday pattern of the electricity consumption with a seasonal component with a period of 24. Take 6 factors. After TSL is done estimating the model, we find our first seasonal pattern in the top panel of Figure 12.40. After midnight, we see a decrease in electricity consumption which bottoms out around 4 am. From that point on, the day slowly starts and with that electricity consumption increases. It peaks at noon, goes down a bit and increases further to the daily peak around 7pm after which it declines till the next day starts. Interestingly, the differences between daily high and low values is larger during the summer as we can see from the bottom panel of Figure 12.40.

Start a loss calculation on the Model comparison page to see the effect on forecasting performance from adding the first seasonal.

12.10.3 Weekday seasonal with a period of 168

We continue building our model by adding the second seasonal, the day-of-the-week seasonal, with a period of $24 \times 7 = 168$. Take 6 factors. After TSL is done estimating the model, we find our second seasonal pattern in Figure 12.41. The top panel is the extracted intraday pattern that we saw in Figure 12.40. The mid panel is our extracted day-of-the-week pattern and we see that electricity consumption is on average lower on Saturday and Sunday compared

Figure 12.40
Intraday seasonal pattern



to the weekdays. The bottom panel shows the sum of the two seasonal patterns.

Start a loss calculation on the Model comparison page to see the effect on forecasting performance from adding the second seasonal. The loss calculation becomes a time-consuming process since we have to make many h-step-ahead forecasts, see also Section 10.1.

12.10.4 Annual seasonal with a period of 8766

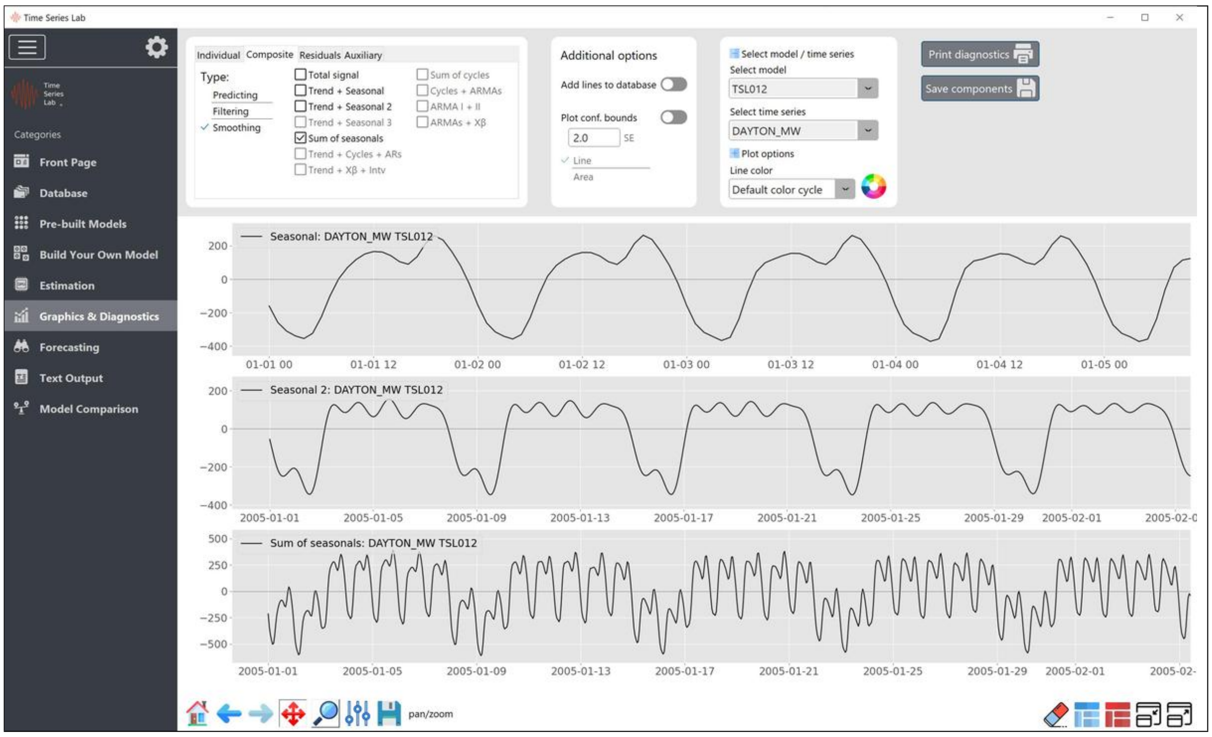
The third and last seasonal component, models the annual seasonal pattern to take the differences in electricity consumption per month of the year into account. This seasonal behaves gradually over time since one periods spans 8766 observations. Take 6 factors. To have the third seasonal not interfere with the level, we can adjust the model in several ways. For example, limiting the variance of the level and the 3rd seasonal component by restricting them to a certain value. A probably more elegant way is adding an ARMA(1,0) component to the model which will effectively serve as AR(1) errors. The result is a smooth behaving level and 3rd seasonal component.

Our final text output is:

```
Variance of disturbances:


Variance type                        Value        q-ratio
Level variance                      0.3892     3.6432e-04
```

Figure 12.41

Intraday, day-of-the-week, and sum of both seasonal pattern

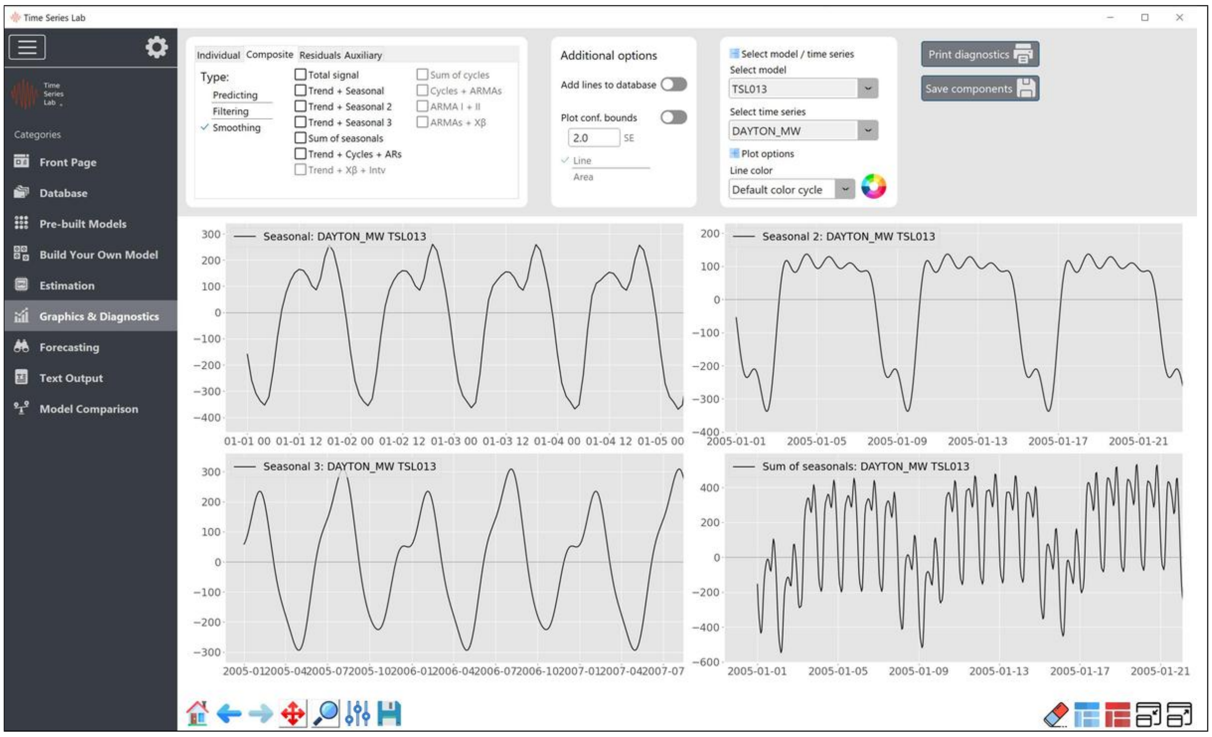


```
Seasonal variance                 0.8066     7.5514e-04
Seasonal variance 2           9.2342e-04     8.6447e-07
Seasonal variance 3               0.0000         0.0000
ARMA variance                  1068.1885         1.0000
Irregular variance                0.0000         0.0000

                                   Value                              Prob
Seasonal chi2 test                 600.6                        8.0100e-121

                                   Value                              Prob
Seasonal chi2 test                1950.5                                 0

                                   Value                              Prob
Seasonal chi2 test                 500.1                         2.0618e-99


ARMA properties:

Parameter type                     Value
Unconditional variance        41547.5465
AR1 phi1                          0.9871
```

with the three seasonal components all extremely significant with p-values effectively zero.

Figure 12.42 shows the extracted intraday, day-of-the-week and annual patterns. The bottom right panel is the sum of the three seasonal components. The botom left panel shows the annual seasonal component. it shows that electricity consumption has two peaks during the year. The first one is around february and the second peaks during the summer months.

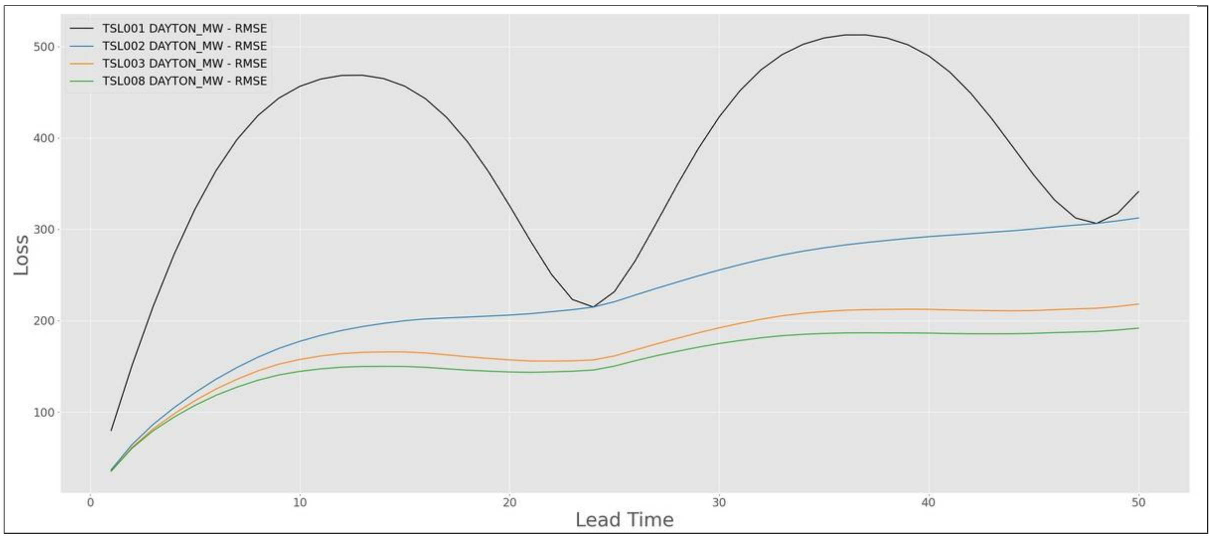
Figure 12.42
Intraday, day-of-the-week, annual pattern, and sum of seasonals


Select model: TSL013
Select time series: DAYTON_MW
Plot conf. bounds: 2.0 SE
Line color: Default color cycle

It is time to compare the forecasting performance of our models. Start a loss calculation on the Model comparison page to see the effect on forecasting performance from adding the third seasonal. Plotting the losses of the four models in shown in Figure 12.43. We see that in each modelling step, the loss is lower. It shows that taking into account the different seasonal patterns makes a lot of difference in forecasting.

Figure 12.43

Forecast losses for the seasonal models



Appendices

Appendix A

Dynamic models

Why do we need dynamic models? Short answer, many real-world processess are dynamic / time-varying. The more we can capture dynamics, the better we understand the processes and the better we can predict them. Many processes exhibit some form of dynamic structure. The list of examples is endless and contains almost every (academic) field. For example, *finance* where the volatility of stock price returns is not constant over time. In *Economics*, where the sale of clothing items exhibit strong seasonality due to summer and winter but also daily seasonal patterns because Saturday will be, in general, a more busy day than Monday, the trajectory of a rocket in *Engineering*, The El Niño effect due to change in water temperature in *Climatology*, the number of oak processionary caterpillars throughout the year in *Biology*, to name a diverse few. If we would be interested in saying anything meaningful about the examples above we need to deal with time-varyingness in some sort of way.

We illustrate the strength of dynamic models with figures. The data is the number of cases of Dengue (logged values) in a region of Venezuela from 2001 to 2017, see Figure A.1.

Figure A.1

Number of cases of Dengue in a region of Venezuela.

Number of cases of Dengue in a region of Venezuela from 2001 to 2017. The estimated static mean is displayed as well.

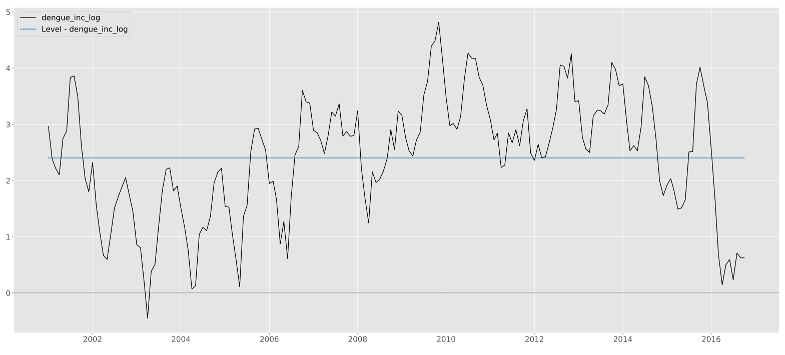


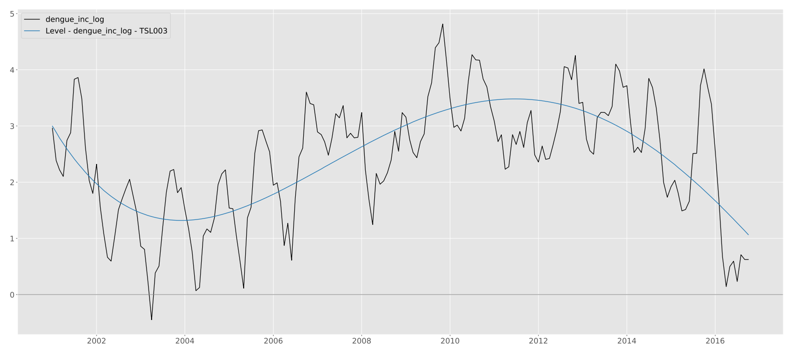
Figure A.1 shows the static mean and we can clearly see that a static mean would give a model fit that can be easily improved on. In the beginning of the sample, the mean is much too high and during the worst period the static mean is far below the actual number of cases. Needless to say, we could not use a static model to make accurate forecasts for this series.

Now consider the situation if we would make the mean time-varying by allowing it to have some smooth pattern over time. The dynamic mean clearly follows the data much better.

Figure A.2

Cases of Dengue and time-varying mean

Number of cases of Dengue in a region of Venezuela from 2001 to 2017 with time-varying mean.

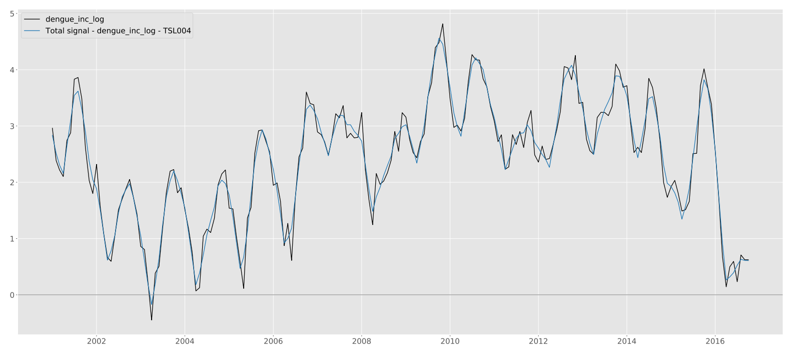


As it turns out, model fit can be further improved by taking into account the monthly effect of the time series. The improved model fit is displayed in Figure A.3. The figures in this section can all be replicated with TSL.

Figure A.3

Number of cases of Dengue with dynamic mean with monthly effect

Number of cases of Dengue in a region of Venezuela from 2001 to 2017 and dynamic mean. The dynamic mean includes a monthly seasonal component.



Appendix B

State Space models

Consider a parametric model for an observed time series $y = (y_1', \dots, y_n')'$ that is formulated conditionally on a latent $m \times 1$ time-varying parameter vector α_t, for time index $t = 1, \dots, n$. We are interested in the statistical behavior of the state vector, α_t, given a subset of the data, i.e. the data up to time $t-1$ (forecasting), the data up to time t (filtering) or the whole data set (smoothing). One possible framework for such an analysis is the state space model, the general form of which is given by

$$y_t|\alpha_t \sim p(y_t|\alpha_t;\psi), \qquad \alpha_{t+1} \sim p(\alpha_{t+1}|\alpha_t;\psi), \qquad \alpha_1 \sim p(\alpha_1;\psi), \tag{B.1}$$

where $p(y_t|\alpha_t;\psi)$ is the observation density, $p(\alpha_{t+1}|\alpha_t;\psi)$ is the state transition density with initial density $p(\alpha_1;\psi)$ and ψ is a static parameter vector.

Minimum mean square error (MMSE) estimates of α_t and MMSE forecasts for y_t can be obtained by the Kalman filter and related smoother methods if the following three conditions are met: (i) the state transition density $p(\alpha_{t+1}|\alpha_t;\psi)$ for α_t is linear and Gaussian, (ii) the relation between y_t and α_t in $p(y_t|\alpha_t;\psi)$ is linear and (iii) the observation y_t is, conditional on α_t, normally distributed. In other words, $p(y_t|\alpha_t;\psi)$, $p(\alpha_{t+1}|\alpha_t;\psi)$ and $p(\alpha_1;\psi)$ are Gaussian and the observation and transition relations are linear. If all three conditions are satisfied, the state space model of (B.1) reduces to the linear Gaussian state space model,

$$\begin{aligned} y_t &= Z\alpha_t + \varepsilon_t, & \varepsilon_t &\sim N(0, H_t), & & \\ \alpha_{t+1} &= T\alpha_t + \eta_t, & \eta_t &\sim N(0, Q_t), & \alpha_1 &\sim p(a_1, P_1), \end{aligned} \tag{B.2}$$

for $t = 1, \dots, n$, see for example Durbin and Koopman (2012, Part I). The violation of at least one of the three properties means that the state space model becomes nonlinear and/or non-Gaussian for which we have to rely on other methods to obtain optimal estimates.

In TSL, we work with Linear Gaussian State Space models and the principle of maximum likelihood estimation (MLE). The main motivation to use MLE are the well established and well documented properties of MLE.

Appendix C

Score-driven models

Consider a parametric model for an observed time series $y = (y_1', \dots, y_n')'$ that is formulated conditionally on a latent $m \times 1$ time-varying parameter vector α_t, for time index $t = 1, \dots, n$. We are interested in the statistical behavior of α_t given a subset of the data, i.e. the data up to time $t-1$. One possible framework for such an analysis is the class of score-driven models in which the latent time-varying parameter vector α_t is updated over time using an autoregressive updating function based on the score of the conditional observation probability density function, see Creal et al. (2013) and Harvey (2013). The updating function for α_t is given by

$$\alpha_{t+1} = \omega + \sum_{i=1}^{p} A_i s_{t-i+1} + \sum_{j=1}^{q} B_j \alpha_{t-j+1},$$

where ω is a vector of constants, A and B are fixed coefficient matrices and s_t is the scaled score function which is the driving force behind the updating equation. The unknown coefficients ω, A and B depend on the static parameter vector ψ. The definition of s_t is

$$s_t = S_t \cdot \nabla_t, \qquad \nabla_t = \frac{\partial \log p(y_t | \alpha_t, \mathcal{F}_{t-1}; \psi)}{\partial \alpha_t}, \qquad t = 1, \dots, n,$$

where ∇_t is the score vector of the (predictive) density $p(y_t | \alpha_t, \mathcal{F}_{t-1}; \psi)$ of the observed time series $y = (y_1', \dots, y_n')'$. The information set $\mathcal{F}_{t-1}$ usually consists of lagged variables of α_t and y_t but can contain exogenous variables as well. To introduce further flexibility in the model, the score vector ∇_t can be scaled by a matrix S_t. Common choices for S_t are unit scaling, the inverse of the Fisher information matrix, or the square root of the Fisher inverse information matrix. The latter has the advantage of giving s_t a unit variance since the Fisher information matrix is the variance matrix of the score vector. In this framework and given past information, the time-varying parameter vector α_t is perfectly predictable one-step-ahead.

The score-driven model has three main advantages: (i) the 'filtered' estimates of the time-varying parameter are optimal in a Kullback-Leibler sense;(ii) since the score-driven models are observation driven, their likelihood is known in closed-form; and (iii) the forecasting performance of these models is comparable to their parameter-driven counterparts, see Koopman et al. (2016). The second point emphasizes that static parameters can be estimated in a straightforward way using maximum likelihood methods.

Appendix D

Submodels of score-driven models

Score-driven models encompass several other econometric models, among several well-known like ARMA models and the GARCH model of Engle (1982). Furthermore the ACD model of Engle and Russell (1998), the autoregressive conditional multinomial (ACM) model of Russell and Engle (2005), the GARMA models of Benjamin et al. (2003), and the Poisson count models discussed by Davis et al. (2005). We now show mathematically how ARMA and GARCH models are submodels of score-driven models.

D.1 The ARMA model

Consider the time-varying mean model

$$y_t = \alpha_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathrm{NID}(0, \sigma^2),$$

for $t = 1, \dots, T$ and where NID means Normally Independently Distributed. If we apply the score-driven methodology as discussed in Appendix C and we take $p = q = 1$ we have,

$$\alpha_{t+1} = \omega + \beta\alpha_t + \kappa s_t, \qquad s_t = S_t \cdot \nabla,$$

where

$$\nabla_t = \frac{\partial \ell_t}{\partial \alpha_t}, \qquad S_t = -E_{t-1}\left[\frac{\partial^2 \ell_t}{\partial \alpha_t \partial \alpha_t}\right]^{-1},$$

with

$$\ell_t = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log \sigma^2 - \frac{1}{2\sigma^2}(y_t - \alpha_t)^2.$$

We obtain

$$\nabla_t = \frac{1}{\sigma^2}(y_t - \alpha_t), \qquad S_t = \sigma^2,$$

and $s_t = y_t - \alpha_t$ which is the prediction error. This means that the score updating becomes

$$\alpha_{t+1} = \omega + \beta\alpha_t + \kappa(y_t - \alpha_t),$$

and if we now replace $\alpha_t = y_t - \varepsilon_t$, we have

$$y_{t+1} = \omega + \beta y_t + \varepsilon_{t+1} + (\kappa - \beta)\varepsilon_t,$$

and hence score updating implies the ARMA(1,1) model for y_t

$$y_t = \omega + \phi y_{t-1} + \varepsilon_t + \theta \varepsilon_{t-1},$$

where $\phi \equiv \beta$ and $\theta = \kappa - \beta$. Furthermore, if we set $\kappa = \beta$, we obtain the AR(1) model and if we set $\beta = 0$ we obtain the MA(1) model. The above is valid for higher lag orders p, q as well which means that the score-driven framework encompasses the ARMA(p,q) model.

D.2 The GARCH model

The strong results of the above section holds, with a couple of small changes, for the time-varying variance model as well. Consider the time-varying variance model

$$y_t = \mu + \varepsilon_t, \qquad \varepsilon_t \sim \mathrm{NID}(0, \alpha_t),$$

for $t = 1, \ldots, T$ and where NID means Normally Independently Distributed. After setting $\mu = 0$ we have the predictive logdensity

$$\ell_t = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log \alpha_t - \frac{y_t^2}{2\alpha_t}.$$

We obtain

$$\nabla_t = \frac{1}{2\alpha_t^2} y_t^2 - \frac{1}{2\alpha_t} = \frac{1}{2\alpha_t^2}(y_t^2 - \alpha_t).$$

Furthermore we have $S_t = 2\alpha_t^2$ and we obtain $s_t = y_t^2 - \alpha_t$. This means that the score updating becomes

$$\alpha_{t+1} = \omega + \beta \alpha_t + \kappa(y_t^2 - \alpha_t),$$

and hence score updating implies the GARCH(1,1) model

$$\alpha_{t+1} = \omega + \phi \alpha_t + \kappa^* y_t^2,$$

where $\phi = \beta - \kappa$ and $\kappa^* \equiv \kappa$. Furthermore, if we set $\kappa = \beta$, we obtain the ARCH(1) model. The above is valid for higher lag orders of p, q as well which means that the score-driven framework encompasses the GARCH(p,q) model.

It should be emphasized that a score-driven time-varying variance model with Student t distributed errors is not equal to a GARCH-t model.

Bibliography

Bamston, A. G., M. Chelliah, and S. B. Goldenberg (1997). Documentation of a highly enso-related sst region in the equatorial pacific: Research note. *Atmosphere-ocean 35*(3), 367–383.

Bates, J. M. and C. W. Granger (1969). The combination of forecasts. *Journal of the Operational Research Society 20*(4), 451–468.

Benjamin, M. A., R. A. Rigby, and D. M. Stasinopoulos (2003). Generalized autoregressive moving average models. *Journal of the American Statistical association 98*(461), 214–223.

Box, G. E., G. M. Jenkins, G. C. Reinsel, and G. M. Ljung (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Brown, R. G. (1959). *Statistical forecasting for inventory control*. McGraw/Hill.

Creal, D. D., S. J. Koopman, and A. Lucas (2013). Generalized autoregressive score models with applications. *Journal of Applied Econometrics 28(5)*, 777–795.

Davis, R. A., W. T. Dunsmuir, and S. B. Streett (2005). Maximum likelihood estimation for an observation driven model for poisson counts. *Methodology and Computing in Applied Probability 7*(2), 149–159.

De Livera, A. M., R. J. Hyndman, and R. D. Snyder (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American statistical association 106*(496), 1513–1527.

Durbin, J. and S. J. Koopman (2012). *Time Series Analysis by State Space Methods* (2nd ed.). Oxford: Oxford University Press.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica 50(4)*, 987–1007.

Engle, R. F. and J. R. Russell (1998). Autoregressive conditional duration: a new model for irregularly spaced transaction data. *Econometrica 66(5)*, 1127–1162.

Granger, C. W. and R. Ramanathan (1984). Improved methods of combining forecasts. *Journal of forecasting 3*(2), 197–204.

Hansen, B. E. (2008). Least-squares forecast averaging. *Journal of Econometrics 146*(2), 342–350.

Harvey, A. C. (1990). *Forecasting, structural time series models and the Kalman filter*. Cambridge university press.

Harvey, A. C. (2013). *Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series*, Volume 52. Cambridge: Cambridge University Press.

Harvey, A. C. and S. J. Koopman (1992). Diagnostic checking of unobserved-components time series models. *Journal of Business & Economic Statistics 10*(4), 377–389.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting 20*(1), 5–10.

Hyndman, R. J. and Y. Khandakar (2008). Automatic time series forecasting: the forecast package for r. *Journal of statistical software 27*, 1–22.

Koopman, S. J., A. Lucas, and M. Scharth (2016). Predicting time-varying parameters with parameter-driven and observation-driven models. *Review of Economics and Statistics 98(1)*, 97–110.

Li, M., S. J. Koopman, R. Lit, and D. Petrova (2020). Long-term forecasting of el niño events via dynamic factor simulations. *Journal of Econometrics 214*(1), 46–66.

Petrova, D., S. J. Koopman, J. Ballester, and X. Rodó (2017). Improving the long-lead predictability of el niño using a novel forecasting scheme based on a dynamic components model. *Climate Dynamics 48*(3), 1249–1276.

Russell, J. R. and R. F. Engle (2005). A discrete-state continuous-time model of financial transactions prices and times: The autoregressive conditional multinomial–autoregressive conditional duration model. *Journal of Business & Economic Statistics 23*(2), 166–180.

Timmermann, A. (2006). Forecast combinations. *Handbook of economic forecasting 1*, 135–196.

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management science 6*(3), 324–342.