Time Series Lab - Score Edition Manual

Modelling time series in five steps

Rutger Lit

Time
Series
Lab

Preface

Time Series Lab - Score Edition is a software program to analyze, model, and forecast time series. The software allows users to specify a wide range of dynamic components and probability distributions to extract the maximum amount of signal from the time series data. More information can be found on `https://timeserieslab.com`. The software is developed by R. Lit (Nlitn) in cooperation with Prof. S.J. Koopman and Prof. A.C. Harvey. *Time Series Lab - Score Edition* should be cited as:

Lit, R., S.J. Koopman, and A.C. Harvey (2019-2020), *Time Series Lab - Score Edition*: `https://timeserieslab.com`

Credits: Icons - Flaticon

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

Bugs: found 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 - Score Edition* or inquiries about customized version of the program, please send an email to `info@timeserieslab.com`.

Contents

Preface ii

1 Getting started 1
1.1 Installing and starting *TSL-SE* 2
1.2 Selecting models 2
1.3 A five step modelling procedure 4

2 Main menu and model output 5
2.1 Main menu 5
2.2 Menu bar 5

3 Step 1: Loading and preparing data 7
3.1 Database 7
3.2 Data selection 7
3.3 Data transformation 9
3.4 Graphical inspection of the data 10

4 Step 2: Model setup 12
4.1 Distribution 12
4.2 Select components for *Location* 13
4.3 Select components for *Scale* 15
4.4 Model specification 16
4.5 Advanced settings 16
4.5.1 Score settings 16
4.5.2 Advanced settings location / scale 16

5 Step 3: Estimation 18
5.1 Edit and fix parameter values 18
5.2 Estimation options 19
5.3 Additional output 20
5.4 Text output for Nile data series 20

6 Step 4: Graphical output 22
6.1 Selecting plot components 22
6.2 Additional functionality 24
6.2.1 Clear all 24
6.2.2 Add subplot 24
6.2.3 Output tests 24
6.2.4 Save all components 24

7 Step 5: Forecasting 25
7.1 Forecast settings 25
7.2 Plot area 25
7.3 Out-of-sample model fit 26

8 Case study 29
8.1 Estimating a dynamic Gompertz model 29
8.1.1 Including a daily seasonal 32
8.1.2 Creating subplots 34

Appendices 35

A Dynamic models 36

B Score-driven models 39

C Submodels of score-driven models 41
C.1 The ARMA model 41
C.2 The GARCH model 42

Bibliography 43

Chapter 1

Getting started

If you're interested in time series analysis and forecasting, you are at the right place. The *Time Series Lab - Score Edition* (*TSL-SE*) software package makes time series analysis available to anyone with a basic knowledge of statistics. The program is written in such a way that results can be obtained quickly. However, many advanced options are available for the time series experts among us.

Although not strictly required, we advise you to read Appendix A – C 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 models (which are constant over time). The algorithms of *TSL-SE* are based on the score-driven methodology, see Creal et al. (2013) and Harvey (2013). Appendix B discusses the mathematical framework of score-driven models. Knowledge of the methodology is not required to use *TSL-SE* but is provided to the interested reader. Appendix C shows that well-known models like ARMA and GARCH models are submodels of score-driven models.

There are a few key things to know about *TSL-SE* before you start. First, *TSL-SE* operates using a number of different steps (1 – 5). 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 will warn 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 functionalities.

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

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

TSL - SE 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*, *Seasonal component*, and *Explanatory variables*. We will see examples of *time-varying components* throughout this manual.

1.1 Installing and starting *TSL - SE*

You can download the *TSL - SE* software for free from `https://timeserieslab.com`. Currently only the Windows platform is supported. *TSL - SE* can be started by double-clicking the icon on the desktop or by clicking the Windows **Start** button and selecting *TSL - SE* from the list of installed programs.

1.2 Selecting models

After starting the software, you see the screen as depicted in Figure 1.1. The *Model category* menu shows several pre-specified models that come with *TSL - SE*. The pre-specified models help the user to set up their model and get results quickly.

Important: Selecting the *Score-driven models* option in the *Model category* menu allows the user to do **all** model settings manually and uncover the full potential of score-driven models. The pre-specified models skip certain parts of the program.

The above information is also communicated to you via the orange info button on the front page of the program (see also Figure 1.1). It tells you that:

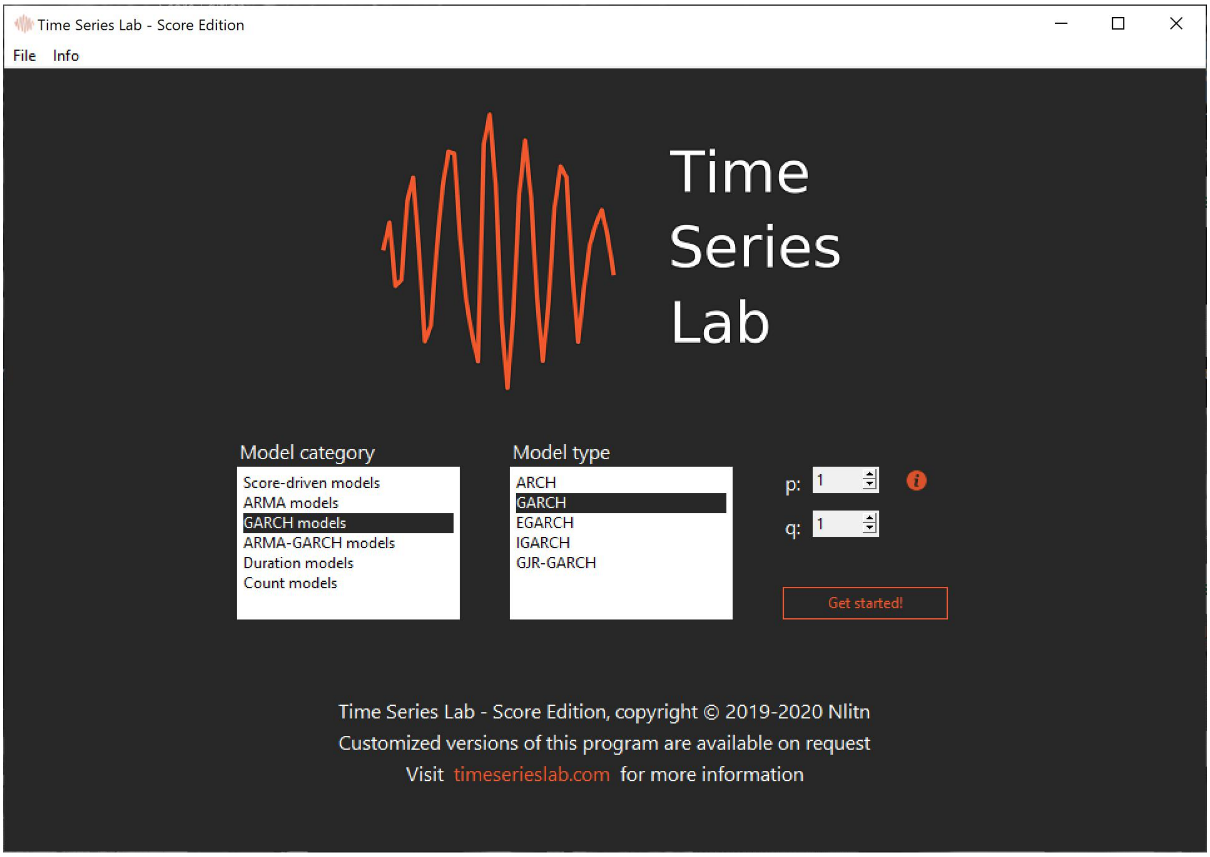
> *TSL - SE* allows you to analyze and forecast a wide range of linear and non-linear time series models. Score-driven models are so versatile that well-known models like ARMA and GARCH models, are subclasses of score-driven models.
>
> In the 'Model category' and 'Model type' section, you find pre-specified models that are submodels of score-driven models. If you want to use the full potential of score-driven models and specify the model components yourself, please select the 'Score-driven models' option in the 'Model category' list.
>
> After selections are made, press the 'Get started' button and you will be taken to the next step of the modelling process.
>
> You can always return to this page by clicking File > Front page in the top left corner.

It should be emphasized that whichever choice you make, your model of choice is always a score-driven model.

The pre-specified *Model categories* are *ARMA*, *GARCH*, *ARMA-GARCH*, *Duration*, and

Figure 1.1

Front page of *TSL - SE* with pre-specified GARCH model



p: 1

q: 1

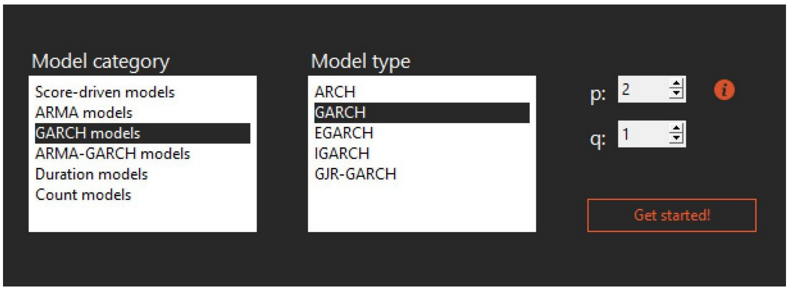
Time Series Lab - Score Edition, copyright © 2019-2020 Nlitn
Customized versions of this program are available on request
Visit timeserieslab.com for more information

Count models with in each category several *Model types*. For example, if we are interested in time-varying volatility in stock returns and we wish to model our data with a *GARCH(2,1)* model, we choose *GARCH models* from the *Model category* menu, *GARCH* from the *Model type* menu, and set $p = 2, q = 1$ in the spin boxes, see Figure 1.2 for a screen shot of the selections.

Figure 1.2

***Time Series Lab - Score Edition* - GARCH(2,1) selection**

Time Series Lab - Score Edition front page with pre-specified model menus and GARCH(2,1) volatility model selected.



p: 2

q: 1

The *Get started* button takes us to the next step of the modelling process. The selections made determine which step will be next. For example, if no data set is loaded yet, clicking the *Get started* button leads us to the *load data page*, see also Chapter 3. We can always return to the *front page* by clicking *File* > *Front page* in the menu bar at the top of the page.

1.3 A five step modelling procedure

The modelling process in *TSL - SE* consists of a five step procedure: Load data, Model setup, Estimation, Graphics, and Forecasting. Green and red arrow buttons let you go back and forth in the program one step at a time. In some cases, the next step cannot be displayed because the program is missing information, for example the selection of the dependent variable in step one. Steps four and five can only be reached after the model is estimated. If a pre-specified model is selected on the *front page*, the program skips step two because all model settings are already specified by the program. However, if needed, step two can always be reached by going to the *main page* and selecting step two from there, see also Chapter 2.

Chapter 2

Main menu and model output

2.1 Main menu

🏠 The main page of *TSL - SE* consists of five buttons from which you can go directly to the modelling step of your choice. Vice versa, from each of the five modelling steps, the user can always go to the main page by clicking the home button as shown here on the left. The main page with example modelling output is shown in Figure 2.1. The right area of the main page is dedicated to text output that will be printed during the modelling process. The text output area works as a basic text editor in which you can type and remove text. Press Ctrl+z or Ctrl+y to undo or redo the changes you made in the editor. Right-mouse clicking the text area opens a menu with additional options. During estimation of the model, the program returns to the main page where intermediate estimation results will be shown. The output of the text editor can be saved by clicking *File* > *Save text output* from the *menu bar* at the top of the program, see Section 2.2 for more details. The *Save text output* option can also be found under the right-mouse click menu.

2.2 Menu bar

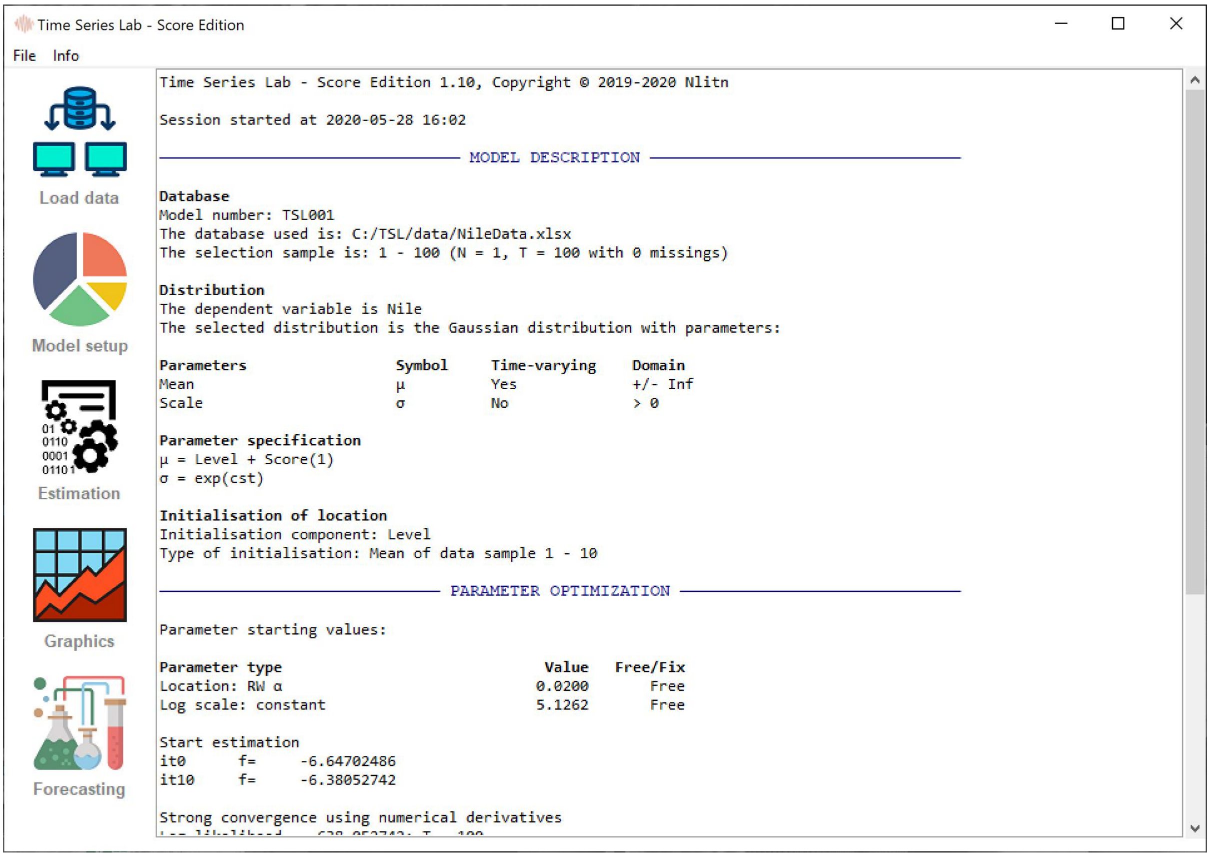
The majority of the program options can be found in the five modelling steps. Some functionalities are however in the menu bar at the top of the page. We briefly discuss the menu bar options.

File menu

Front page: from anywhere in the program, the user can always go back to the front page, where the pre-specified models can be selected.
Main menu: from anywhere in the program, the user can always go back to the main menu.
Load data: shortcut to open the load data window from anywhere in the program.
Save text output: the text output as printed on the main menu output page can be saved in .txt format via this option.

Figure 2.1

Mainpage of *TSL - SE* with buttons to the five modelling steps



```
Time Series Lab - Score Edition 1.10, Copyright © 2019-2020 Nlitn

Session started at 2020-05-28 16:02

──────────────────────────────────── MODEL DESCRIPTION ────────────────────────────────────

Database
Model number: TSL001
The database used is: C:/TSL/data/NileData.xlsx
The selection sample is: 1 - 100 (N = 1, T = 100 with 0 missings)

Distribution
The dependent variable is Nile
The selected distribution is the Gaussian distribution with parameters:

Parameters                  Symbol      Time-varying     Domain
Mean                        μ           Yes              +/- Inf
Scale                       σ           No               > 0

Parameter specification
μ = Level + Score(1)
σ = exp(cst)

Initialisation of location
Initialisation component: Level
Type of initialisation: Mean of data sample 1 - 10

────────────────────────────────── PARAMETER OPTIMIZATION ─────────────────────────────────

Parameter starting values:

Parameter type                              Value    Free/Fix
Location: RW α                             0.0200        Free
Log scale: constant                        5.1262        Free

Start estimation
it0       f=      -6.64702486
it10      f=      -6.38052742

Strong convergence using numerical derivatives
```

Save components: after a model is successfully estimated, select this option to save all extracted time series signal into an Excel or .csv file for further processing.
Exit: exits the program, unsaved work is lost.

Info menu

Check for updates: the program checks whether updates are available on start up. Users can do this manually via this option. If an update is available, the program asks whether it should be updated to the newest version.
Changelog: overview of *TSL - SE* version history with all (semi-)important changes to the program. This can be anything from bug fixes to new features.
About Time Series Lab: some background info on *TSL - SE*.
Feedback: ways to get in contact with the Time Series Lab team.
Documentation: the document you are currently reading can be opened via *TSL - SE*.

Chapter 3

Step 1: Loading and preparing data

The first step in any time series analysis is the inspection and preparation of data. From the main page, clicking the button on the left brings you to the data inspection and preparation page as displayed in Figure 3.1. The Nile data set[1] that comes bundled with the installation file of *TSL - SE* is used as illustration.

3.1 Database

The data set is loaded and selected from the file system by pressing the *Load data* button (below the *Main menu* button).

Important: The data set should be in column format with headers. The format of the data should be *.xls(x), or *.csv, *.txt with comma's as field separation. The program 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 are displayed in the *Database* section at the top left of the page. Clicking on a header name plots the data in the plot area at the bottom of the page. As shown in 3.1, the Nile data is currently highlighted and (automatically) plotted.

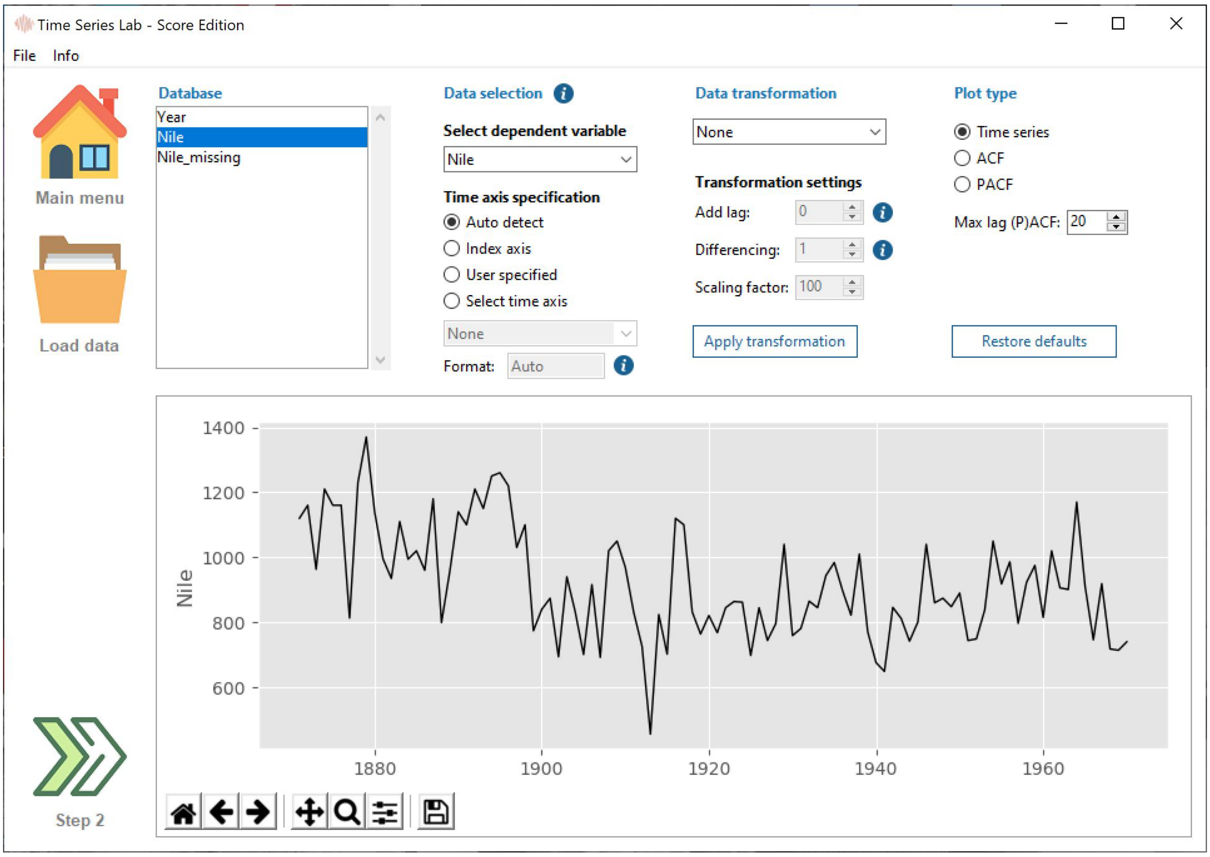
3.2 Data selection

The highlighted variable Nile also appears in the *Select dependent variable* pull 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, analyze, and forecast. The

[1] The Nile data set consists of a series of readings of the annual flow volume of the river Nile at the city of Aswan from 1871 to 1970.

Figure 3.1

Data inspection and preparation page



Database
Year
Nile
Nile_missing

Data selection

Select dependent variable: Nile

Time axis specification
● Auto detect
○ Index axis
○ User specified
○ Select time axis

None

Format: Auto

Data transformation: None

Transformation settings
Add lag: 0
Differencing: 1
Scaling factor: 100

Apply transformation

Plot type
● Time series
○ ACF
○ PACF

Max lag (P)ACF: 20

Restore defaults

dependent variable needs to be specified because without it, the software cannot estimate a model. 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 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 (technical) text behind the info button tells us:

> Specify date format codes according to the 1989 C-standard convention, e.g.
>
> 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.

3.3 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* pull 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* on it and selecting the *Delete from database* option. For the transformations: *Lag operator*, *Difference*, and *Scaling*, the program needs extra user input. Depending on the selection, spin boxes under *Transformation settings* become operable.

Add lag:
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. Note the text behind the information buttons that says:

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

Differencing:
A non-stationary time series[2] can be made stationary by differencing. For example, if y_t

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

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 previous observation from the current observation. *TSL-SE* accommodates for this procedure since differencing is common practice among time series researchers. However, the methodology of *TSL-SE* allows the user to explicitly model non-stationary time series and differencing is not strictly necessary. Note the text behind the information buttons 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 or some small values, e.g. < 0.001, could potentially lead to numerical instabilities. This can be solved by scaling the time series to more manageable numbers.

3.4 Graphical inspection of the data

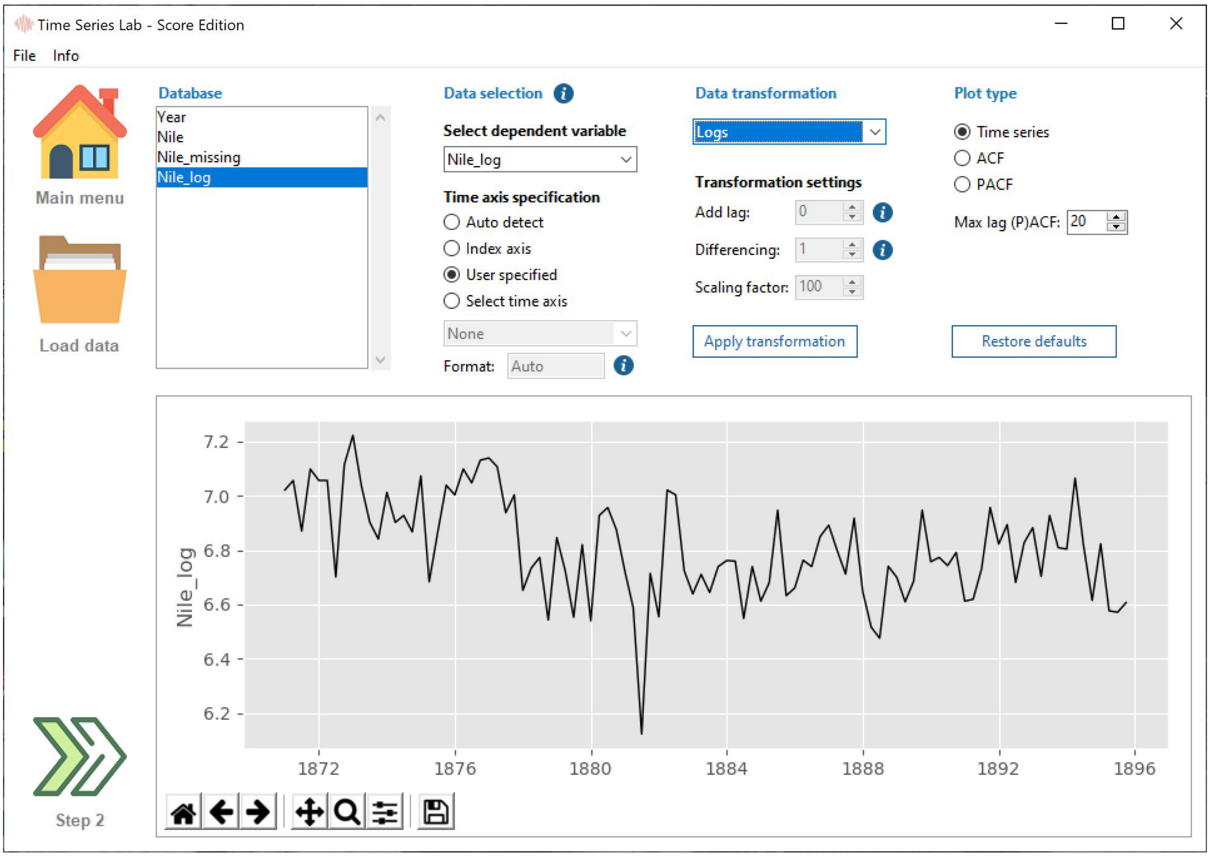
Plot type
Different types of time series plots can be activated by selecting one of the three options: *Time series*, *ACF*, or *PACF*. The *Time series* option plots the selected time series. ACF and PACF plots are advanced time series features and the explanation and ideas behind them are out of the scope of this manual.

Plots
The plot area at the bottom of the window has more option than shown here. For example, clicking the *right mouse* button on the plot area opens a menu with additional options. The title of the plot, and the axes titles can be specified. Furthermore, characteristics of the selected time series can be plotted in the top right corner of the graph. Finally, the buttons in the bottom left corner of the plot area add additional functionality such as zooming in and saving of the figure to a drive.

It is time to go to step 2 of the modelling process. Please press the *Step 2* button in the left bottom corner of your screen to go to the *Model setup* page.

Figure 3.2

Data inspection and preparation page



Time Series Lab - Score Edition

Database: Year, Nile, Nile_missing, Nile_log (selected)

Data selection

Select dependent variable: Nile_log

Time axis specification

○ Auto detect
○ Index axis
● User specified
○ Select time axis

None

Format: Auto

Data transformation: Logs

Transformation settings

Add lag: 0

Differencing: 1

Scaling factor: 100

Plot type

● Time series
○ ACF
○ PACF

Max lag (P)ACF: 20

Chapter 4

Step 2: Model setup

Step 2 of the software looks like Figure 4.1 and can be reached via the *Main menu*, or via *Step 1* or *Step 3*. *Step 2* is the heart of the program and it is here where important modelling decision are made. The selections on this page are based on what we want to model and what our data characteristics are.

Important: Always ask yourself, *"Which parameter needs to be time-varying?"* Is it the *mean* (location) of the distribution or the *variance* (scale). In *TSL - SE* it is even possible to have both parameters time-varying.

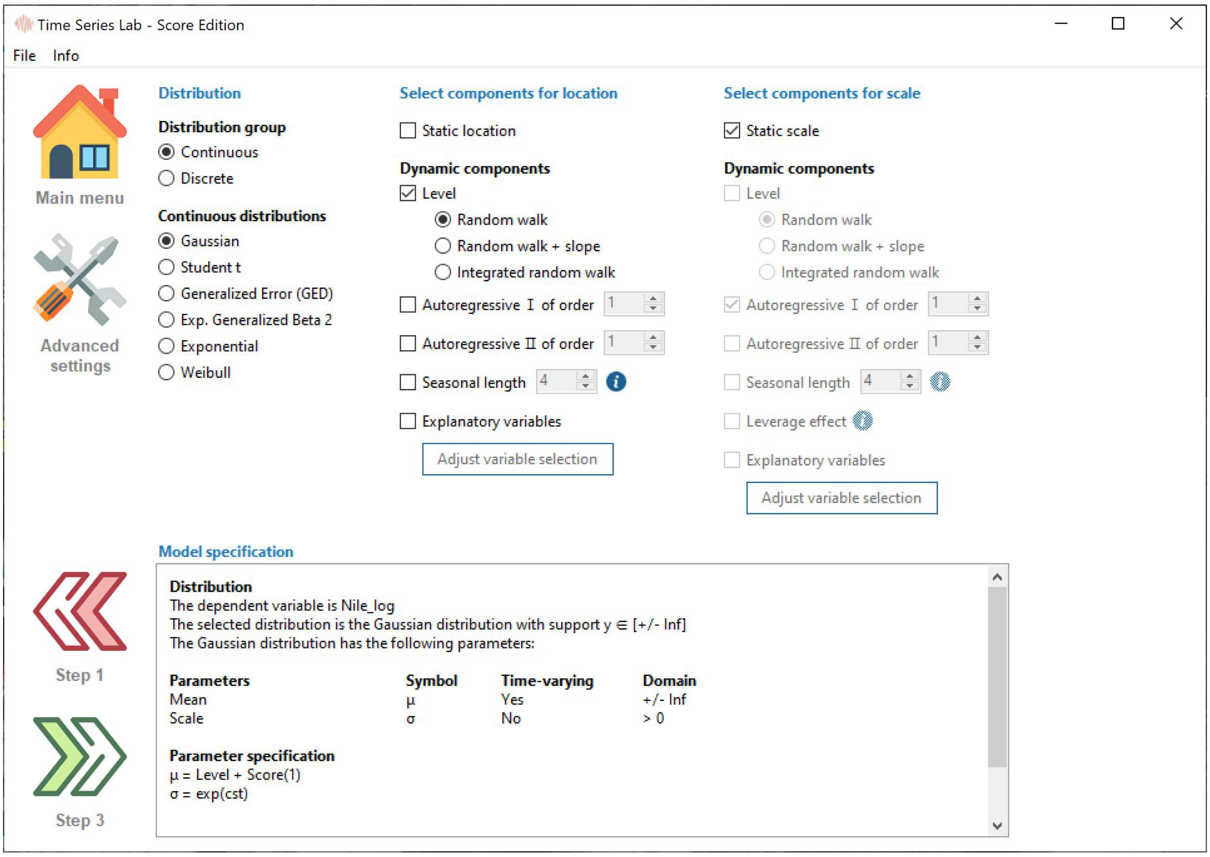
4.1 Distribution

TSL - SE divides the probability distributions in two *Distribution groups*; *Continuous* distributions and *Discrete* distributions. The choice of distribution depends strongly on the characteristics of the time series data. The *Discrete* distributions are the most specific of the two groups because they can only be applied to discrete data, in contrast to *Continuous* distributions who can handle both continuous and discrete data. Discrete data can only take certain values and are often integers (whole numbers) but categorical data could also be regarded as discrete. Examples of discrete data are the number of goals scored by a football team or the number of earthquakes in a certain region. Continuous data are not restricted to certain values, and can occupy any value over a continuous range. Examples of continuous data are stock returns or the lap times of an F1 car.

From the choice of distributions, the *Gaussian* is the most well-known but it is not the only one. And here lies the power of *TSL - SE*: the program can handle so many different time series because several probability distributions which each there own unique specifications are part of *TSL - SE*. This makes the program extremely versatile and almost any time series can be analyzed with the software.

We continue with the *Nile data* that we loaded in *Step 1* and select the *Gaussian* distribution for our data, see Figure 4.1.

Figure 4.1
Probability distributions and model components


Time Series Lab - Score Edition
File Info
Main menu
Advanced settings
Step 1
Step 3
Adjust variable selection
Adjust variable selection

Distribution

Distribution group
● Continuous
○ Discrete

Continuous distributions
● Gaussian
○ Student t
○ Generalized Error (GED)
○ Exp. Generalized Beta 2
○ Exponential
○ Weibull

Select components for location

□ Static location

Dynamic components
☑ Level
● Random walk
○ Random walk + slope
○ Integrated random walk
□ Autoregressive I of order 1
□ Autoregressive II of order 1
□ Seasonal length 4
□ Explanatory variables

Select components for scale

☑ Static scale

Dynamic components
□ Level
● Random walk
○ Random walk + slope
○ Integrated random walk
☑ Autoregressive I of order 1
□ Autoregressive II of order 1
□ Seasonal length 4
□ Leverage effect
□ Explanatory variables

Model specification

Distribution
The dependent variable is Nile_log
The selected distribution is the Gaussian distribution with support y ∈ [+/- Inf]
The Gaussian distribution has the following parameters:

Parameters	Symbol	Time-varying	Domain
Mean	μ	Yes	+/- Inf
Scale	σ	No	> 0

Parameter specification
μ = Level + Score(1)
σ = exp(cst)

4.2 Select components for *Location*

The *Nile* data set is an example of data with a time-varying *Location*. We are not so much interested in the spread of the *Nile* data around the *Location* but we want to know how the level of the *Nile* behaves over time, for example, to forecast the level of the *Nile* in the next year. For a symmetric distribution like the *Gaussian* distribution, the *Location* is equal to the *Mean*. This is however not valid for all probability distributions. Additionally to *Location*, *TSL - SE* also reports the *Mean* of the distribution in its text and graphical output.

Since we are interested in a time-varying location we untick the check box for *Static location* so that all dynamic components become available. Notice how the dynamic components are *greyed out* if we would tick the check box for *Static location*. For the *Nile data* illustration, our selection for the *Location* parameter should like Figure 4.1

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 - SE 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 - SE* falls in the class of *filters* since it filters the time series from the *noise* to obtain *signal*. It is the *signal* what we are interested in because it tells us something about 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. In the score-driven methodology as explained in Appendix B, the scaled score of the predictive density is the driver. 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 score updating function.

We discuss each dynamic component and its characteristics.

Level

The *Level* component is a non-stationary component. In its simplest form it's a *Random walk* and can be extended with a drift parameter for direction. The *Integrated random walk* is a special type of *Random walk + drift* component and often gives a smooth(er) pattern over time. Although the *Random walk* is mathematically very simple, it proves very useful in many cases.

Autoregressive I and II

The *Autoregressive* component is a stationary component. Its order can be specified up to lags of $p = 31$. *TSL - SE* always restricts the autoregressive parameters in such a way that the autoregressive process stays within the stationary region.

Seasonal

The *Seasonal* component is a non-stationary component. Its seasonal length can be specified up to $s = 52$. The s seasonal components sum up to zero for identification. This is enforced by estimating the first $s-1$ which, together with the zero sum restriction, identifies the last one (more in this in *Step 3* of Section 5). The number of seasons of the *Seasonal* component depends on the data. Number of seasons should not be taken literally (although for quarterly data it would be correct). It refers to the number of periods before the seasonal process repeats itself. The info button clarifies more and tells us:

> Examples of seasonal specifications are:
> Monthly data, s = 12.
> Quarterly data, s = 4.
> Daily data, when modelling the weekly pattern, s = 7.

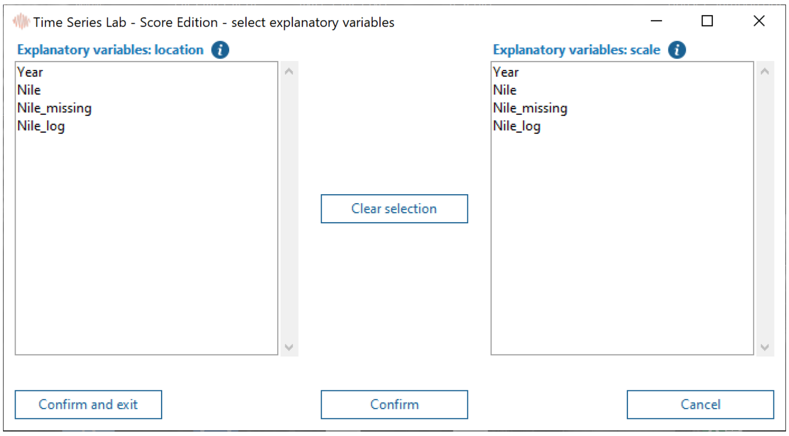
Explanatory variables

Ticking the *Explanatory variables* box opens a new window where *Explanatory variables* can be selected, see Figure 4.2. Notice that the *Location* and *Scale* component can have different *Explanatory variables*. All variables, including the newly created variable Nile_log from *Step 1* (Section 3.1) are present and can be selected as *Explanatory variables*. The variable Nile_log is used as illustration and should not be included in the model.

Figure 4.2

TSL* - *SE*- selection of *Explanatory variables

Window where *Explanatory variables* can be selected for *Location* and *Scale* separately. All variables that were loaded and newly transformed under *Step 1* can be selected via the menu's.


Time Series Lab - Score Edition - select explanatory variables
Explanatory variables: location
Year
Nile
Nile_missing
Nile_log
Clear selection
Explanatory variables: scale
Year
Nile
Nile_missing
Nile_log
Confirm and exit
Confirm
Cancel

4.3 Select components for *Scale*

A time-varying scale is useful for time series models with time-varying volatility. Typical examples are *GARCH* models, see also Appendix C. For the description of the model components for *Scale* see the *Location* section above. *Scale* has one extra component compared to *Location* and that is the *Leverage component*. The *Leverage* effect is present in some time series and captures asymmetric volatility. Time-varying volatility does not play a big role in the *Nile* data set so we tick the check box for *Static scale* so that all dynamic components become *greyed out*. For the *Nile data* illustration, our selection for the *Scale* parameter should like Figure 4.1

4.4 Model specification

At the bottom of our *Step 2* screen, we find a summary of all our model decisions, see also Figure 4.1. Our component selections and settings are directly translated to this model summary. The number of probability distribution parameters differ per distribution. For example the *Poisson* distribution only has one parameter (*Location* or *Intensity*) while the *Exp. Generalized Beta 2* distribution has four (*Location*, *Scale*, and two shape parameters). Only *Location* and *Scale* parameter can be selected as time-varying.

4.5 Advanced settings

The *Advanced settings* button as shown here on the left gives us access to the advanced model settings.

4.5.1 Score settings

For model stability reasons, Inverse Fisher scaling is the best option for the majority of models, see Appendix B for more information about score scaling. Multiple lags of the score updating function can be included in the model. The number of score lags can be set individually for *Location* and *Scale*. For the majority of time series models, lag 1 will be sufficient. ARCH models can be replicated by ticking the *Set* $\alpha = \phi$ check box, see Appendix C for more information.

4.5.2 Advanced settings location / scale

Dynamic components need to be initialized, i.e. they need to start from some specified value. If more than one dynamic component is selected, a choice need to be made which component should be initialized to avoid identification issues. The component that is initialized is free to start from a value other than zero, based on the model setting *Unconditional mean*, *Estimate*, or *Mean of data sample*.

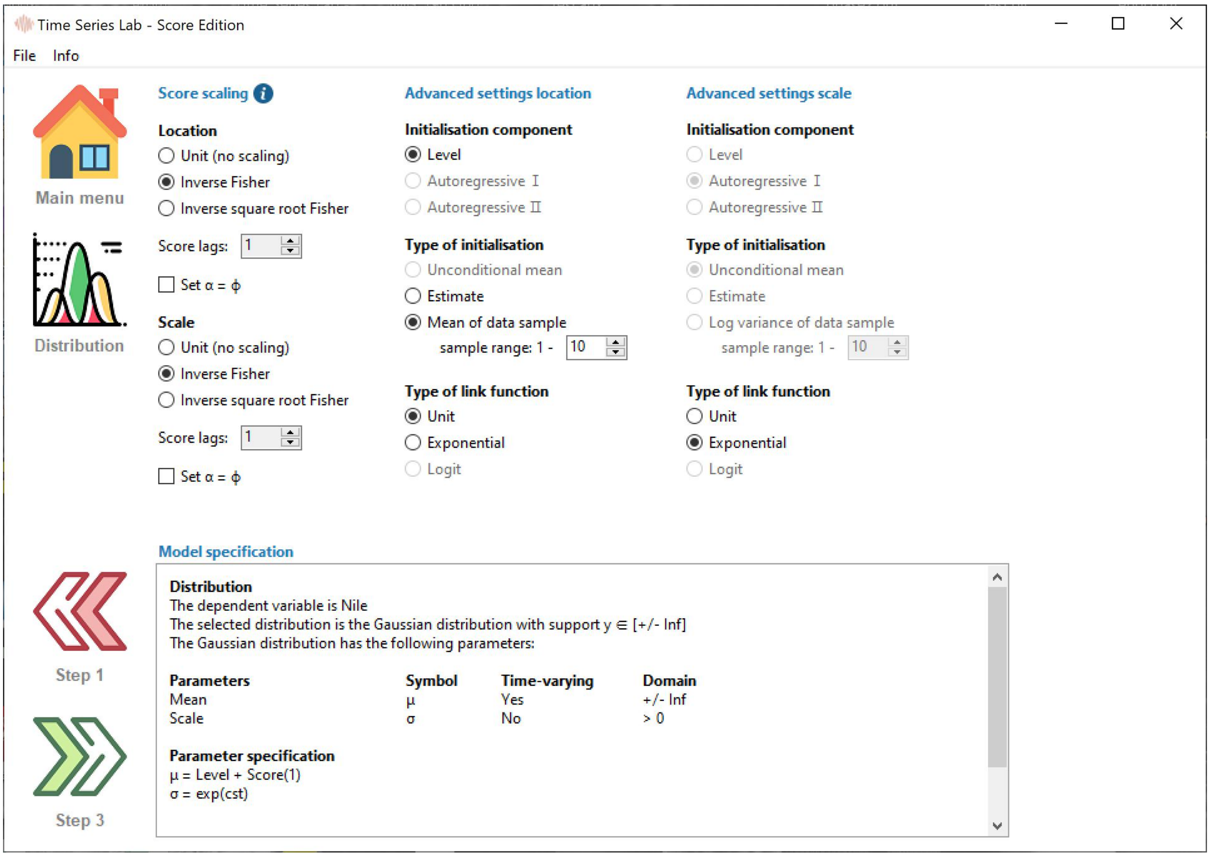
The *Unconditional mean* option can only be selected for the stationary *Autoregressive* components I and II for the simple reason that non-stationary components do not have an *Unconditional mean*. The initialization can be estimated as well meaning the first element of the dynamic component will be part of the *hyper parameter* vector that will be estimated in *Step 3*, see Section 5. The last initialization method is to determine the initialization non-parametrically from the data. For our *Nile* data example this translates to taking the average of the first 10 observations of the time series. The sample range can be set to a different number as well. A sample range of 1 means that the initialization component starts from the first observation of the time series. All non-initialization components start from zero to avoid identification issues.

The link function creates, as the name suggest, a link between the signal and the model components. A *unit link* means a one-to-one relationship between the signal and the model components. The *exponential* link function is often used to ensure positivity. For example the *intensity* of a *Poisson* distribution cannot be negative so modelling the *intensity* $\lambda_t = \exp(\alpha_t)$ with α_t being the signal at time t ensures that λ_t will never be negative. Another example is the variance (or standard error) of a distribution which cannot be negative and is therefore often modelled in combination with the *exponential* link function. For our *Nile* data illustration, we use the model settings as shown in Figure 4.3.

It is time to go to *Step 3* of the modelling process. Please press the *Step 3* button in the left bottom corner of your screen to go to the *Estimation* page.

Figure 4.3

Probability distributions and model components



Score scaling

Location

○ Unit (no scaling)
● Inverse Fisher
○ Inverse square root Fisher

Score lags: 1

□ Set α = ϕ

Scale

○ Unit (no scaling)
● Inverse Fisher
○ Inverse square root Fisher

Score lags: 1

□ Set α = ϕ

Advanced settings location

Initialisation component

● Level
○ Autoregressive I
○ Autoregressive II

Type of initialisation

○ Unconditional mean
○ Estimate
● Mean of data sample
sample range: 1 - 10

Type of link function

● Unit
○ Exponential
○ Logit

Advanced settings scale

Initialisation component

○ Level
● Autoregressive I
○ Autoregressive II

Type of initialisation

● Unconditional mean
○ Estimate
○ Log variance of data sample
sample range: 1 - 10

Type of link function

○ Unit
● Exponential
○ Logit

Model specification

Distribution
The dependent variable is Nile
The selected distribution is the Gaussian distribution with support y ∈ [+/- Inf]
The Gaussian distribution has the following parameters:

Parameters	**Symbol**	**Time-varying**	**Domain**
Mean	μ	Yes	+/- Inf
Scale	σ	No	> 0

Parameter specification
μ = Level + Score(1)
σ = exp(cst)

Chapter 5

Step 3: Estimation

The *estimation* page of *TSL - SE* looks like Figure 5.2 and can be reached via the *Main menu*, or via *Step 2* or *Step 4*. On this page we specify estimation settings based on the choices we made in *Step 2*.

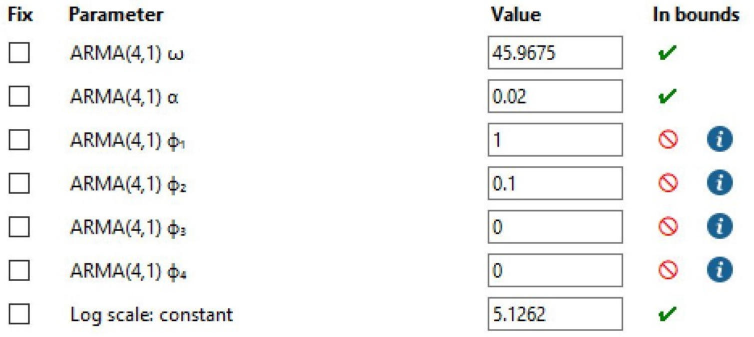
5.1 Edit and fix parameter values

After we have left *Step 2*, the software collected all the *hyper parameters* that belong to our probability distribution and model component choices and summarized them on the *estimation* page. Before we click the green *Estimate* button below the *Main menu* button to estimate the model, we have the option to fix parameters to a certain value. We do this by ticking the *Fix* check box and by specifying the value that we want to fix the parameter to in the *Value* entry boxes. If a *Fix* check box is not ticked, the corresponding value in the *Value* entry box is taken as starting value for the optimization algorithm. 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 5.1.

Figure 5.1

***TSL - SE* warning for a non-stationary *ARMA(4,1)* process**

An *ARMA(4,1)* model is specified in *Step 2* of *TSL - SE*. The program warns the user via the *In bounds* column that the (user) specified values lead to an *ARMA* process that is non-stationary.

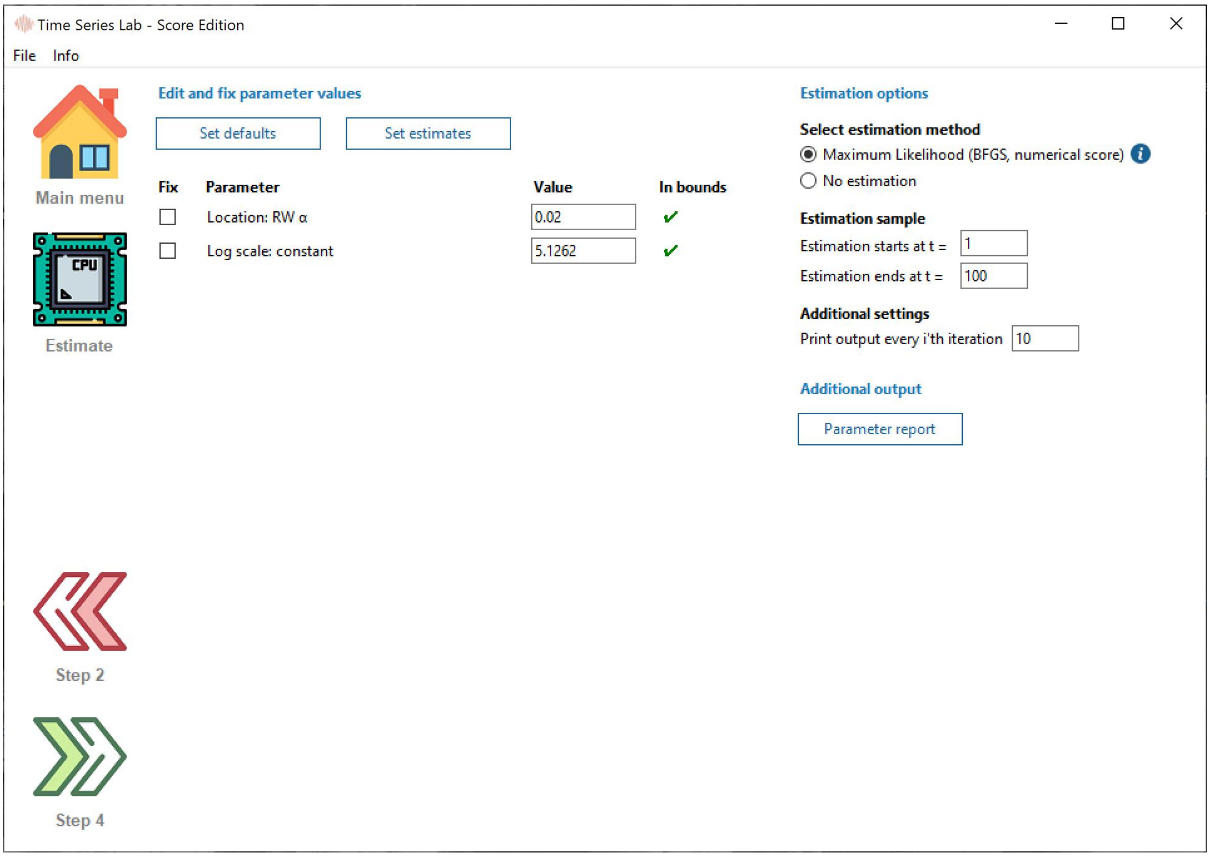


Fix	Parameter	Value	In bounds
□	ARMA(4,1) ω	45.9675	✓
□	ARMA(4,1) α	0.02	✓
□	ARMA(4,1) ϕ_1	1	⊘ ℹ
□	ARMA(4,1) ϕ_2	0.1	⊘ ℹ
□	ARMA(4,1) ϕ_3	0	⊘ ℹ
□	ARMA(4,1) ϕ_4	0	⊘ ℹ
□	Log scale: constant	5.1262	✓

Based on our *Nile* data model selections in *Step 2*, *TSL - SE* only needs to estimate two *hyper parameters*. The first one is κ that influences the behavior of the updating of the *Random walk* score-driven component and the second parameter is the constant scale (or variance) of the Gaussian distribution, see also Figure 5.2.

Figure 5.2

Estimation page of *TSL - SE*



Edit and fix parameter values

Fix	Parameter	Value	In bounds
□	Location: RW α	0.02	✓
□	Log scale: constant	5.1262	✓

Estimation options

Select estimation method

● Maximum Likelihood (BFGS, numerical score)

○ No estimation

Estimation sample

Estimation starts at t = 1

Estimation ends at t = 100

Additional settings

Print output every i'th iteration 10

Additional output

5.2 Estimation options

The *hyper parameters* can be estimated by *Maximum Likelihood* with the *BFGS* algorithm or *No estimation* can be performed so that text and graphical output will be based on the provided starting values. Maximizing the likelihood for the *Nile* data set 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 more narrow sample by setting the *Estimation starts at t* and *Estimation end at t* entry boxes.

5.3 Additional output

After the successful estimation of a model, a (hyper) parameter report can be generated. Clicking the *Parameter report* button brings us to the *Main 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. As a rule of thumb, the generation of a parameter report takes at least the amount of time it takes to maximize the likelihood.

For our *Nile* data illustration, we use the model settings as shown in Figure 5.2. It is time to go to *Step 4* of the modelling process. Please click the green *Estimate* button below the *Main menu* button to start the estimation process. During estimation we will see the Main page with (intermediate) optimization results and once the optimization is finished we will be taken to the *Graphical* output of *Step 4*. During and after the estimation of the Nile model we see the text output in the next section.

5.4 Text output for Nile data series

Time Series Lab - Score Edition 1.20, Copyright © 2019-2020 Nlitn

Session started at 2020-05-28 14:32

———————————— MODEL DESCRIPTION ————————————

Database
Model number: TSL001
The database used is: C:/TSL/data/NileData.xlsx
The selection sample is: 1 - 100 (N = 1, T = 100 with 0 missings)

Distribution
The dependent variable is Nile

```
The selected distribution is the Gaussian distribution with parameters:

Parameters   Symbol   Time-varying   Domain
Mean         μ        Yes            +/- Inf
Scale        σ        No             > 0

Parameter specification
μ = Level + Score(1)
σ = exp(cst)

Initialisation of location
Initialisation component: Level
Type of initialisation: Mean of data sample 1 - 10

——————————————— PARAMETER OPTIMIZATION ———————————————

Parameter starting values:

Parameter type        Value    Free/Fix
Location: RW κ        0.0200       Free
Log scale: constant   5.1262       Free

Start estimation
it0    f=   -6.64702486
it10   f=   -6.38052742

Strong convergence using numerical derivatives
Log-likelihood = -638.052742; T = 100

Optimized parameter values:

Parameter type        Value    Free/Fix
Location: RW κ        0.2483       Free
Log scale: constant   4.9616       Free

Estimation process completed in 0.0419 seconds

——————————————— STATE INFORMATION ———————————————

Component location        Initial     Time T
Mean                    1132.6000   825.7410
Random walk             1132.6000   825.7410

Component scale           Initial     Time T
Standard deviation       142.8205   142.8205
```

Chapter 6

Step 4: Graphical output

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 - SE* after the estimation process has finished. We can manually reach this page by clicking the button on the left, located on the main page, **after** successful estimation. The default graphical output for the *Nile* data illustration is displayed in Figure 6.1.

6.1 Selecting plot components

Plotting a component 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.

A *Composite signal*, α_t, is defined as the sum of components. In Intermezzo 1 this would be the sum of the level, seasonal, and explanatory variables component, i.e. $\alpha_t = \mu_t + \gamma_t + X_t\beta$. Note that a composite signal is the sum of model components **before** it enters the *link function* so if we model, for example, time-varying volatility as

$$\sigma_t = \exp(\mu_t + X_t\beta),$$

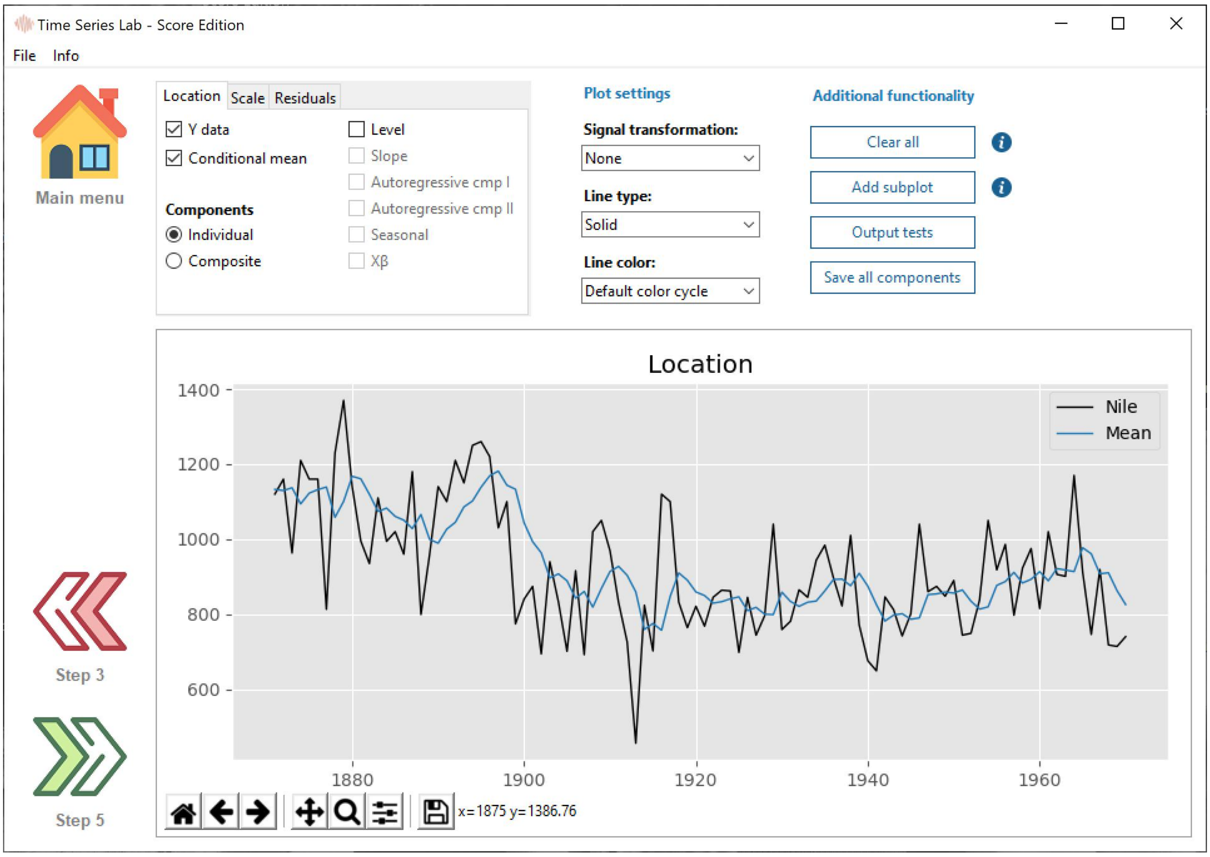
the composite signal would be $\mu_t + X_t\beta$. In our *Nile* data illustration we have the unique situation that the *mean*, *composite signal*, and *level* are exactly equal due to the *unit* link function and the *level* being the only component in the model. Despite our simple *Random walk* model, which is often a very 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 $t-1$.

Pearson residuals are the standardized versions of the *Residuals*. For the *Nile* data illustration, the (time-varying) mean is subtracted from the *Residuals* and the remainder is divided by the (constant) standard deviation of the *Gaussian* distribution.

The *Autocorrelation function* (ACF) plot of the *Pearson residuals* are shown in Figure 6.2 which shows that all dynamics in the time series are nicely captured by our model, although

Figure 6.1

Default graphical output for Nile data and Random Walk



Time Series Lab - Score Edition

File Info

Location | Scale | Residuals

☑ Y data

☑ Conditional mean

□ Level

□ Slope

□ Autoregressive cmp I

□ Autoregressive cmp II

□ Seasonal

□ Xβ

Components

● Individual

○ Composite

Plot settings

Signal transformation: None

Line type: Solid

Line color: Default color cycle

Additional functionality

Clear all

Add subplot

Output tests

Save all components

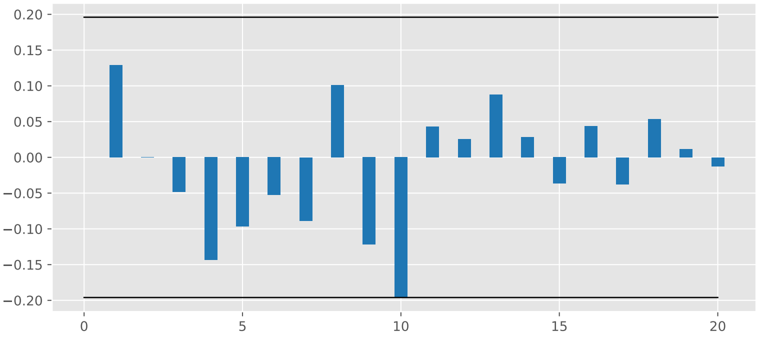
x=1875 y=1386.76

one spike at lag 10 is close to the confidence interval.

Figure 6.2

Autocorrelation plot of Nile data Pearson residuals

Autocorrelation plot of Nile data Pearson residuals. The score-driven *Random walk* model with time-varying mean and constant variance nicely captures the dynamics in the time series.



6.2 Additional functionality

6.2.1 Clear all

The *Clear all* button is rigorous and clears everything from the graph including all subplots. To have more refined control over the (sub)plots, right-mouse click on a subplot for more options. Section 6.2.2 discusses subplots in more detail.

6.2.2 Add subplot

The graph area of *TSL-SE* can consist of a maximum of nine subplots. Subplots can be convenient to graphically summarize model results in one single plot. To add a subplot to the graph, click the *Add subplot* button. Notice that an *empty* subplot is added to the existing graph which correspond 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 box appears shortly around the subplot as a sign that the subplot is active.

Location and *scale* components can be represented in one subplot, just activate the subplot of your choice and switch to the *location* or *scale* tab and tick the check boxes you need. The info button to the right of the *Add subplot* button summarizes these findings and tells us:

> 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.

6.2.3 Output tests

The residual Autocorrelation is graphically representation in the ACF plot. Autocorrelation can also be tested by clicking the *Output tests* button. *Durbin-Watson* and *Ljung-Box* tests are performed and output is printed to the Main page. Summary statistics of *Residuals*, *Pearson Residuals*, and *Score* are printed as well.

6.2.4 Save all components

All model components can be save to disc for further processing or archival purposes. After clicking the *Save all components* button, a window opens that allows you to save the components to an *.xls(x) file or a comma separated *.csv file.

Chapter 7

Step 5: Forecasting

The *forecasting* page of *TSL-SE* with the *Nile* data as illustration looks like Figure 7.1. The page can be reached via the *Main menu* or *Step 4*. Forecasting a time series if 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.

7.1 Forecast settings

The spin box located under *forecast settings*, currently has a reading of 10 and determines the number of time periods in the forecast window. The other spin box (with reading currently 99) determines how many time points the program needs to plot before the first forecast period. Both spin boxes are interactive with the plot in the sense that changing them by clicking the up and down buttons immediately affect the plot window. Numbers can also be entered manually in the spin boxes followed by pressing the *Enter key* on the keyboard.

The current forecast selection can be cleared by clicking the *Clear selection* button and forecasts can be saved in *.xls(x) or *.csv format by clicking the *Save all forecasts* button.

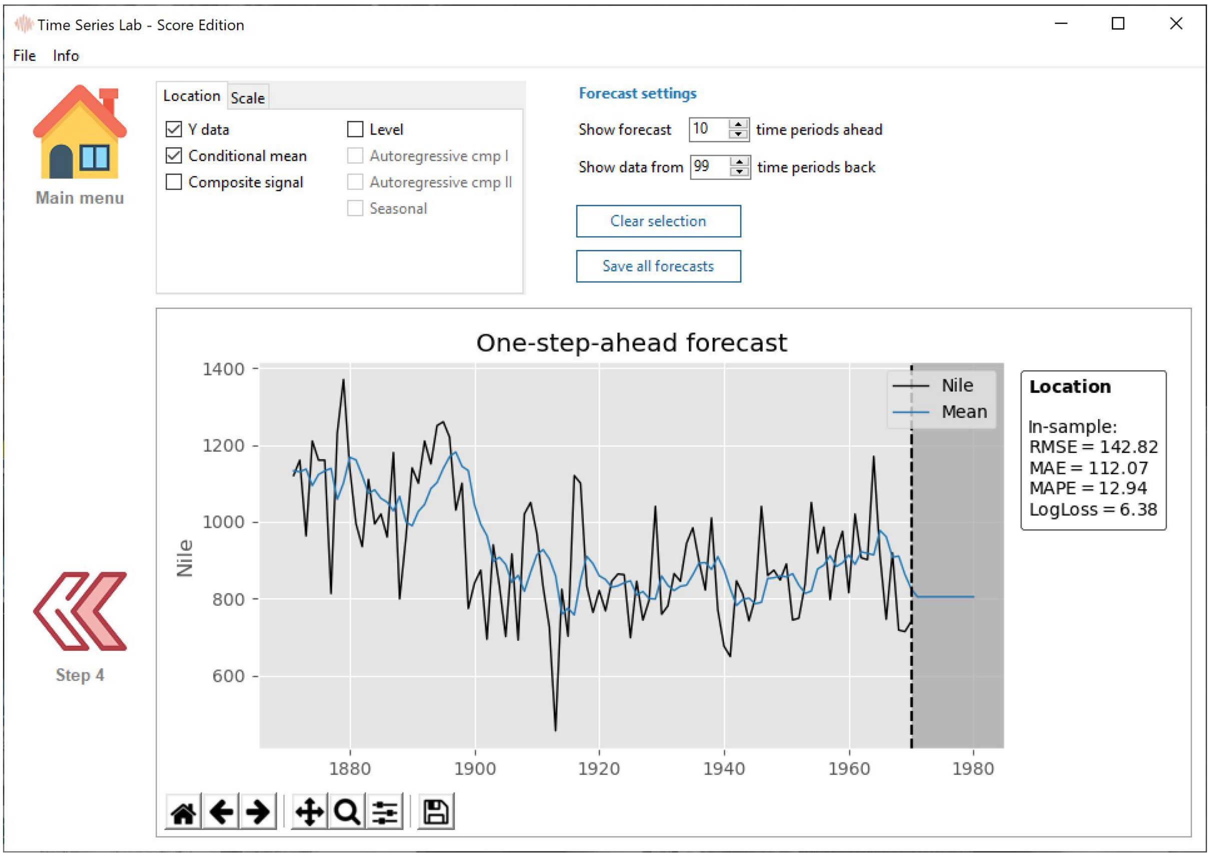
7.2 Plot area

Our *Nile* data forecasts are show in Figure 7.1 for 10 time periods in the future. We see that the forecast is simply a straight line and no dynamics are present in the forecast. This is the result of our model choices. We select a time-varying *location* composed of a score-driven *Random walk* component but the forecast of a *Random walk* is just the value of the last time period. Since we are *out-of-sample* with our forecasts (no observations are present), no score-updating takes place and the result is a straight line into the future.

This is also the reason why the text box on the right of the graph only shows in-sample loss functions because no observations are present after 1970 so losses cannot be calculated. Later in this section we will see an example where we do have *out-of-sample* observations. The calculated loss functions are Root Mean Squared Error *RMSE*, Mean Absolute Error *MAE*, and Mean Absolute Percentage Error *MAPE*. The last one is not available for all data and distribution combinations. The fourth value in the text box is the *LogLoss* which is defined as minus the average Log-likelihood value. Since many more loss functions are possible, forecasts can be saved with the *Save all forecasts* button to allow the user to apply tailor-made forecast themselves.

Figure 7.1

Estimation page of *TSL - SE*



☑ Y data
☑ Conditional mean
☐ Composite signal
☐ Level
☐ Autoregressive cmp I
☐ Autoregressive cmp II
☐ Seasonal

Show forecast 10 time periods ahead

Show data from 99 time periods back

Location

In-sample:
RMSE = 142.82
MAE = 112.07
MAPE = 12.94
LogLoss = 6.38

7.3 Out-of-sample model fit

To give an idea of the additional options in *TSL - SE* if *out-of-sample* observations were present we go back two steps in the modelling process (to step 3). On the *estimation* page we change the *Estimation ends at* value from 100 to 90. We click the *Estimate* button and after the estimation is finished we go to step 5, *forecasting*. Our screen now looks like the one in Figure 7.2. Three things stand out.

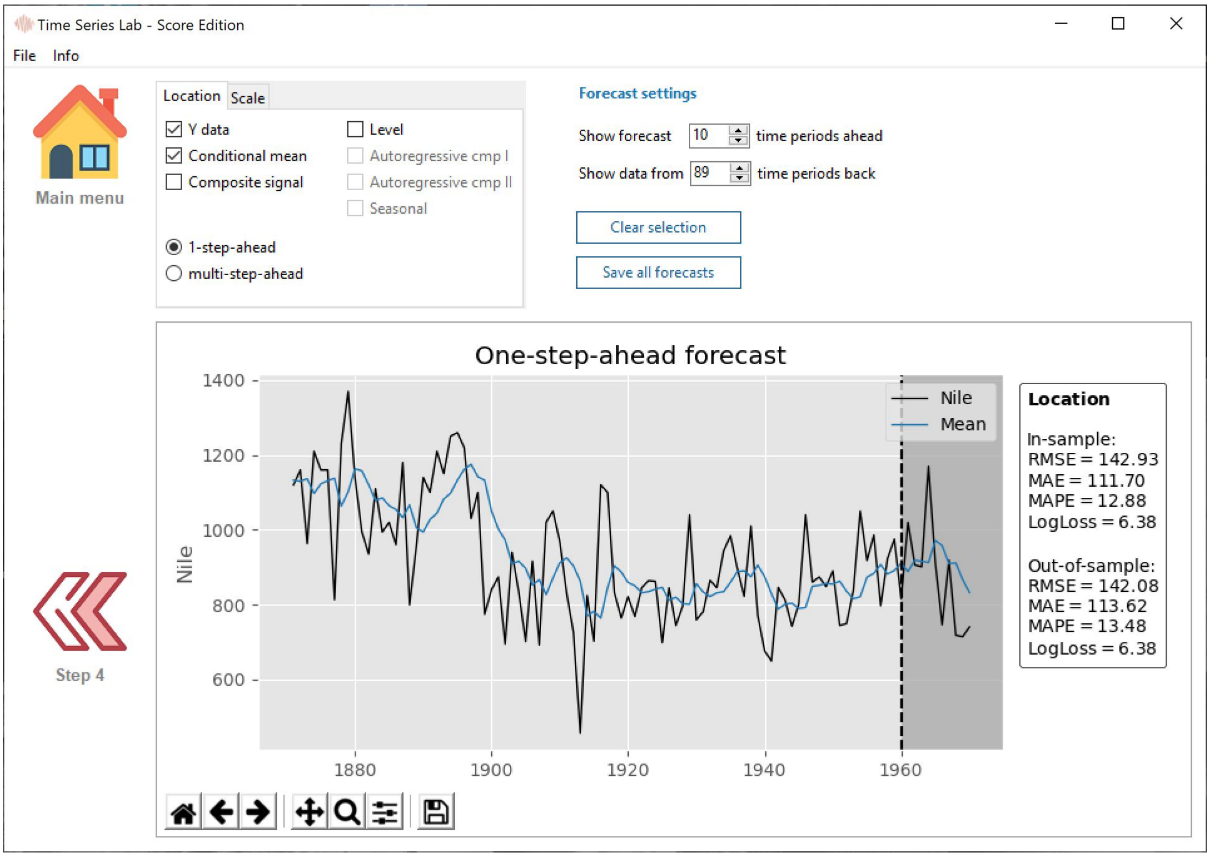
First, radio buttons appear in the top left box of the program window. We can choose between *1-step-ahead* forecasting and *multiple-step-ahead*. The difference is if score-driven updating is taken into account or not. If we have an observation at time t we can calculate the score and update our signal for time $t+1$ accordingly. If we make *multiple-step-ahead* forecasts we do not take the information from the observations into account and our forecasts are again a straight line just as in Figure 7.1.

Two, with *1-step-ahead* forecasting, dynamics are present in our forecast because after we have made our forecast we are able to update our *signal* based on the information up to time t.

Three, with the presence of observations we can now calculate *out-of-sample* forecast errors. The text box on the right communicates *out-of-sample* losses via the same loss functions as used to asses *in-sample* model fit. Comparing the *out-of-sample* losses for *1-step-ahead* and *multiple-step-ahead* forecasting shows that the loss is smaller for *1-step-ahead* forecasting which is not surprising since *1-step-ahead* forecasting takes more information into account, see also Figure 7.3 and Figure 7.4.

Figure 7.2

Estimation page of *TSL - SE*



☑ Y data
☑ Conditional mean
□ Composite signal
□ Level
□ Autoregressive cmp I
□ Autoregressive cmp II
□ Seasonal
● 1-step-ahead
○ multi-step-ahead

Show forecast 10 time periods ahead
Show data from 89 time periods back

Location

In-sample:
RMSE = 142.93
MAE = 111.70
MAPE = 12.88
LogLoss = 6.38

Out-of-sample:
RMSE = 142.08
MAE = 113.62
MAPE = 13.48
LogLoss = 6.38

Figure 7.3

One-step-ahead forecast for Nile data

One-step-ahead forecast for Nile data. The model is a *Random walk* score-driven model and the forecast sample is 10 year ranging from 1961 to 1970.

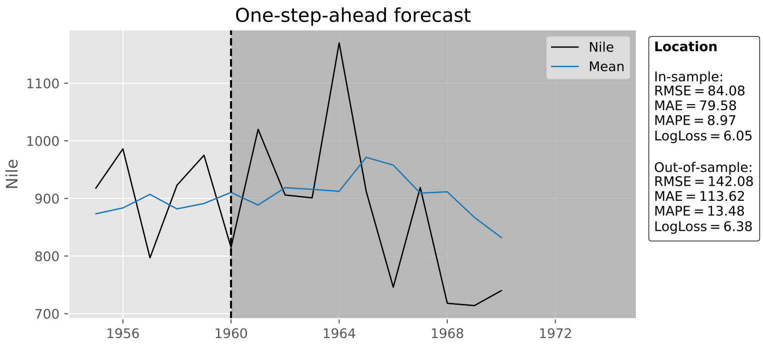
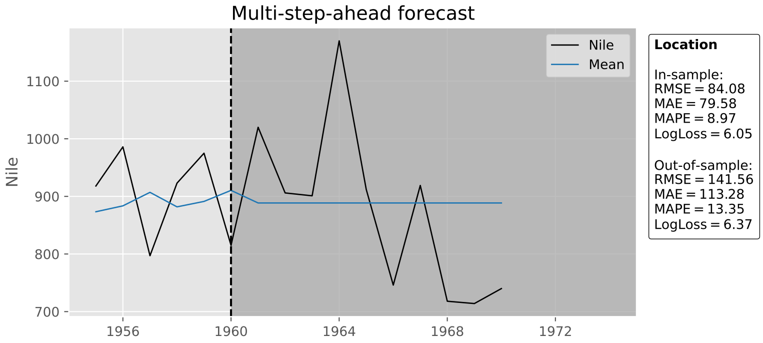


Figure 7.4

Multi-step-ahead forecast for Nile data

Multi-step-ahead forecast for Nile data. The model is a *Random walk* score-driven model and the forecast sample is 10 year ranging from 1961 to 1970.



Chapter 8

Case study

8.1 Estimating a dynamic Gompertz model

This case study is based on the work by Harvey and Kattuman (2020) and Harvey and Lit (2020) and discusses the modelling of growth curves with an example in epidemiology and concerns coronavirus. We fully replicate the modelling results and demonstrate the versatility of *TSL - SE*. First a summary of the model

Intermezzo 2: dynamic growth curves model

The generalized logistic class of growth curves contains the logistic and Gompertz as special cases. They lead to a model in which the increase, $y_{t,}$ at time t depends on the cumulative total Y_t. Specifically,

$$\ln y_t = \rho \ln Y_{t-1} + \delta_t + \varepsilon_t, \qquad \rho \geq 1, \qquad t = 1, \ldots, T, \tag{8.1}$$

where $y_t = Y_t - Y_{t-1}$, δ_t is a trend component, and ε_t is a serially independent Gaussian disturbance with mean zero and constant variance, σ^2_ε, that is $\varepsilon_t \sim NID(0, \sigma^2_\varepsilon)$.
When y_t is small, it may be necessary to adopt a discrete distribution, particularly if some observations are zero. A good choice is the negative binomial which, when parameterized in terms of a time-varying mean, ξ_t, and a fixed positive shape parameter, υ, has probability mass function (PMF)

$$p(y_t) = \frac{\Gamma(\upsilon + y_t)}{y_t!\,\Gamma(\upsilon)}\, \xi_t^{y_t} (\upsilon + \xi_t)^{-y_t} (1 + \xi_t/\upsilon)^{-\upsilon}, \qquad y_t = 0, 1, 2, \ldots.$$

An exponential link function ensures that ξ_t remains positive and at the same time yields an equation similar to (8.1):

$$\ln \xi_t = \rho \ln Y_{t-1} + \delta_t, \quad t = 2, ..., T, \tag{8.2}$$

The model in Intermezzo 2 with ρ set to one is the dynamic Gompertz model and can be fully replicated in *TSL-SE*. We discuss all *TSL-SE* model settings step by step. The data used in this case study is from March 11th 2020 up to, and including, May 6th was obtained from the ECDC website. The data comes bundled with the *TSL-SE* installer and the data file is called *GermanyCovid.xlsx* and is plotted in Figure A.1.

Front page

Select the *Score-driven models* option in the *Model category* menu and click *Get started*.

Step 1

Load the *GermanyCovid.xlsx* file located in the *data* folder of *TSL-SE*. Select the variable *DGerDeath* in the *Database* field or from the pull down menu under *Select dependent variable*.

Step 2

Choose *Discrete* for the distribution group and *Negative Binomial* for the distribution. Make sure *Static intensity* is un-ticked. Tick the *Level* check box and select the *Random walk + slope* component. Furthermore tick the *Explanatory variables* check box and select the *LGerDeaths_1* explanatory variable for intensity. This variable is $\ln Y_{t-1}$ in Intermezzo 2. Make sure that the *Autoregressive processes* and the *Seasonal* are un-ticked.

Click the *Advanced settings* button and select the *Estimate* option under *Type of initialisation*. Choose the *Exponential* link function.

Step 3

Fix the *Log intensity:* β*_LGerDeaths_1* parameter to 1.0 by ticking the *Fix* check box in front of it and typing 1.0 in the *Value* field.
Click the green *Estimate* button.

Output

We should now see the graph in Figure A.2 on our screen and the following text output on the *main page*.

```
Time Series Lab - Score Edition 1.20, Copyright © 2019-2020 Nlitn

Session started at 2020-05-29 08:56

———————————————————— MODEL DESCRIPTION ————————————————————

Database
Model number: TSL001
The database used is: C:/TSL/data/GermanyCovid.xlsx
The selection sample is: 1 - 57 (N = 1, T = 57 with 0 missings)

Distribution
```

The dependent variable is DGerDeath
The selected distribution is the Negative Binomial distribution with parameters:

Parameters	**Symbol**	**Time-varying**	**Domain**
Mean	λ	Yes	> 0
Dispersion	r	No	> 0

Parameter specification
$\lambda = \exp(\text{Level} + \text{X}\beta + \text{Score}(1))$
r = constant

Explanatory variables
Explanatory variable for location is: LGerDeaths_1

Initialisation of intensity
Initialisation component: Level
Type of initialisation: Estimate

——————————— PARAMETER OPTIMIZATION ———————————

Parameter starting values:

Parameter type	**Value**	**Free/Fix**
Log intensity: RW κ	0.0200	Free
Log intensity: slope κ	0.0200	Free
Log intensity: init	4.8098	Free
Log intensity: init slope	0.0000	Free
Log intensity: β_LGerDeaths_1	1.0000	Fixed
Dispersion	5.0000	Free

```
Start estimation
it0    f=   -16.35430024
it10   f=    -5.43577140
it20   f=    -4.98910688
it30   f=    -4.91104728
it40   f=    -4.90077214
it50   f=    -4.89394127
it60   f=    -4.89272556
it70   f=    -4.89271867
it80   f=    -4.89271867
it82   f=    -4.89271867
```

Strong convergence using numerical derivatives
Log-likelihood = -278.884964; T = 57

Optimized parameter values:

Parameter type	**Value**	**Free/Fix**
Log intensity: RW κ	2.5372e-51	Free
Log intensity: slope κ	2.2026e-07	Free
Log intensity: init	-0.2443	Free
Log intensity: init slope	-0.0687	Free
Log intensity: β_LGerDeaths_1	1.0000	Fixed
Dispersion	5.7105	Free

Estimation process completed in 1.0981 seconds

——————————— STATE INFORMATION ———————————

Component intensity	**Initial**	**Time T**
Mean	1.5665	114.2042
Composite signal	0.4488	4.7380
Random walk	-0.2443	-4.0912
Slope	-0.0687	-0.0687
Xβ	0.6931	8.8292

8.1.1 Including a daily seasonal

To include the daily (seasonal) effect in the model, we take the following steps:

Step 2

Leave everything the same except tick the *Seasonal* check box and select 7 as the seasonal length.

Step 3

Make sure the *Log intensity: β_LGerDeaths_1* parameter is still fixed to 1.0 and click the green *Estimate* button.

Output

We should now see the graph in Figure A.3 on our screen and the following text output on the *main page*.

Time Series Lab - Score Edition 1.20, Copyright © 2019-2020 Nlitn

Session started at 2020-05-29 08:56

———————— MODEL DESCRIPTION ————————

Database
Model number: TSL002
The database used is: C:/TSL/data/GermanyCovid.xlsx
The selection sample is: 1 - 57 (N = 1, T = 57 with 0 missings)

Distribution
The dependent variable is DGerDeath
The selected distribution is the Negative Binomial distribution with parameters:

Parameters	**Symbol**	**Time-varying**	**Domain**
Mean	λ	Yes	> 0
Dispersion	r	No	> 0

Parameter specification
λ = exp(Level + Seasonal(7) + Xβ + Score(1))
r = constant

Explanatory variables
Explanatory variable for location is: LGerDeaths_1

Initialisation of intensity
Initialisation component: Level

```
Type of initialisation: Estimate

―――――――――――――――――― PARAMETER OPTIMIZATION ――――――――――――――――――

Parameter starting values:

Parameter type                         Value   Free/Fix
Log intensity: RW κ               1.2000e-07       Free
Log intensity: slope κ            7.0000e-08       Free
Log intensity: init                  -0.2444       Free
Log intensity: init slope            -0.0687       Free
Log intensity: seasonal κ             0.0200       Free
Log intensity: init seasonal 1        0.0000       Free
Log intensity: init seasonal 2        0.0000       Free
Log intensity: init seasonal 3        0.0000       Free
Log intensity: init seasonal 4        0.0000       Free
Log intensity: init seasonal 5        0.0000       Free
Log intensity: init seasonal 6        0.0000       Free
Log intensity: β_LGerDeaths_1         1.0000      Fixed
Dispersion                            5.7101       Free

Start estimation
it0    f=   -4.88405823
it10   f=   -4.72022123
it20   f=   -4.70145346
it30   f=   -4.70103513

Strong convergence using numerical derivatives
Log-likelihood = -267.959002; T = 57

Optimized parameter values:

Parameter type                         Value   Free/Fix
Log intensity: RW κ               1.2000e-07       Free
Log intensity: slope κ            6.9995e-08       Free
Log intensity: init                  -0.2255       Free
Log intensity: init slope            -0.0700       Free
Log intensity: seasonal κ         3.9944e-07       Free
Log intensity: init seasonal 1        0.1983       Free
Log intensity: init seasonal 2        0.1365       Free
Log intensity: init seasonal 3        0.3596       Free
Log intensity: init seasonal 4       -0.1110       Free
Log intensity: init seasonal 5       -0.1566       Free
Log intensity: init seasonal 6       -0.4516       Free
Log intensity: β_LGerDeaths_1         1.0000      Fixed
Dispersion                           13.2568       Free

Estimation process completed in 0.8404 seconds

―――――――――――――――――― STATE INFORMATION ――――――――――――――――――

Component intensity    Initial     Time T
Mean                    1.9464   132.1427
Composite signal        0.6660     4.8839
Random walk            -0.2255    -4.1437
Slope                  -0.0700    -0.0700
Seasonal                0.1983     0.1983
Xβ                      0.6931     8.8292
```

8.1.2 Creating subplots

All the extracted components of our dynamic Gompertz model can be visualized in one graph. Take the following steps:

Step 4
Click the *Clear all* button. Go to the first tab called *Intensity* and set the radio button to *individual* components. Tick the *Level* check box. Click the *Add subplot* button and tick the *Seasonal* check box. Click the *Add subplot* button and tick the $X\beta$ check box.

Go to the second tab called *Residuals*. Click the *Add subplot* button and tick the *Residuals* check box.

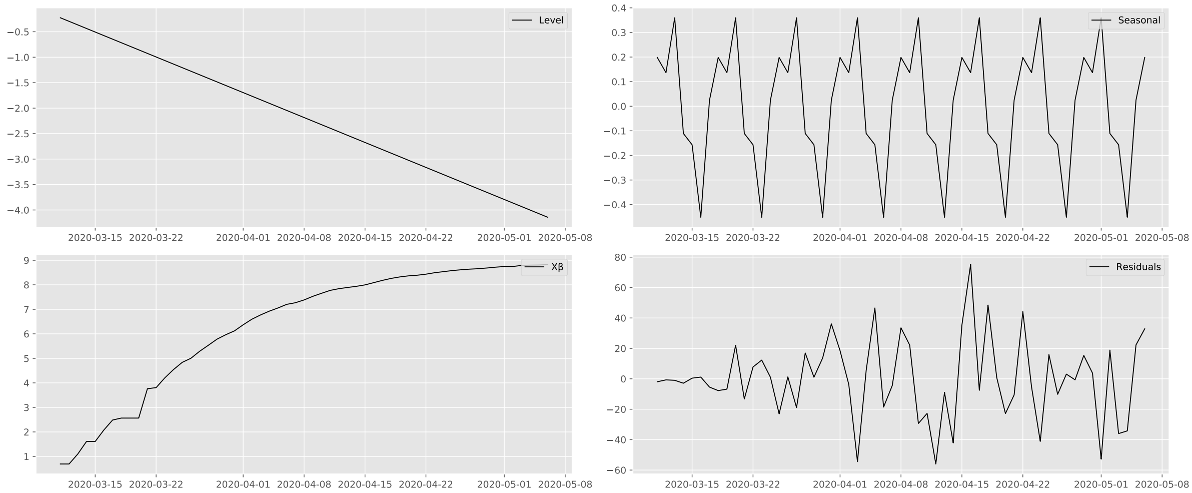
You should now see the 2 × 2 subplot in Figure 8.1. How do these components relate to our time series y_t? Our time series, called DGerDeath, can be fully reconstructed by summing up the time-varying components in the following way

$$DGerDeath = \text{exp}(Level + Seasonal + X\beta) + Residuals$$

You can verify this by saving the components and manually check the above equation.

Figure 8.1

All extracted components of the dynamic Gompertz model



Appendices

Appendix A

Dynamic models

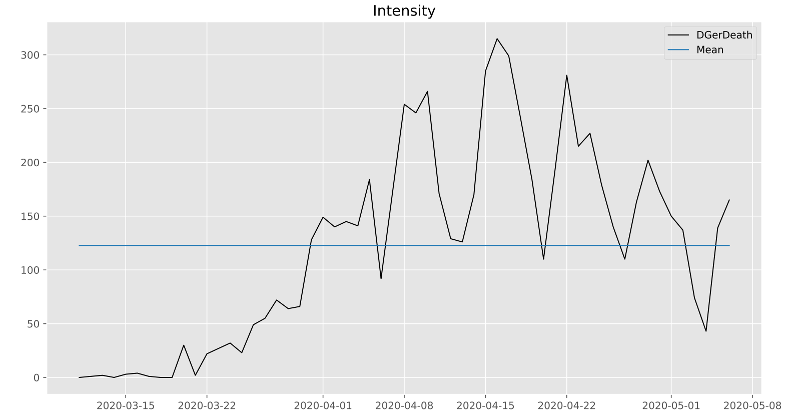
Why do we need dynamic models? Short answer, the world is dynamic. The more we can capture dynamics, the better we understand the world's processes and the better we can predict them. Many processes exhibit some form of dynamic structure. The list of examples is endless and contains almost every academic 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 deaths in Germany from coronavirus from March 11th 2020 up to, and including, May 6th, see Figure A.1.

Figure A.1

Number of deaths in Germany from coronavirus

Number of deaths in Germany from coronavirus from March 11th 2020 up to, and including, May 6th. The estimated static mean of the Negative Binomial distribution is displayed as well. Log Likelihood value: -329.27.



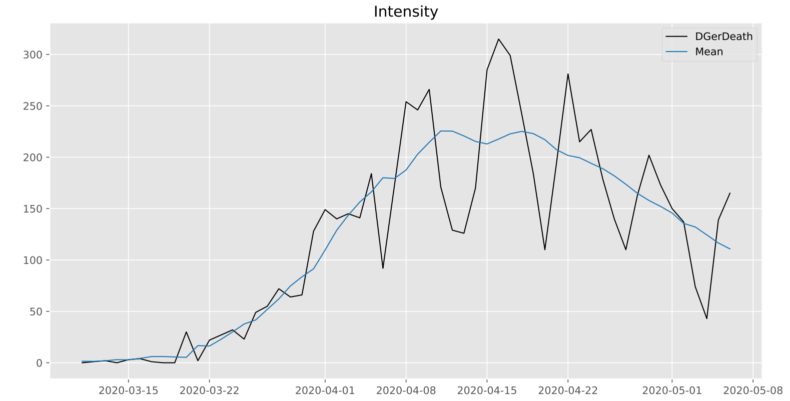
The series is analysed by Harvey and Kattuman (2020) and Harvey and Lit (2020) with *TSL*-*SE*. Figure A.1 shows the static mean of the Negative Binomial distribution and we can clearly see that a static mean would give a model fit that can be easily improved on. In the beginning of the sample the mean is much to high and during the worst period the static mean is far below the actual number of deaths. Needless to say, we could not use a static model to make accurate forecasts for this series. To capture the in-sample model fit in a number, we use the Log Likelihood value which for the Negative Binomial model and a static mean, applied to these series, is -329.27.

Now consider the situation if we would make the mean time-varying by allowing it to have some smooth pattern over time, we refer to the case study in Chapter 8 and Harvey and Lit (2020) for model details. The dynamic mean clearly follows the data much better and as a result our Log Likelihood value increases (strongly) to -278.83.

Figure A.2

Number of deaths from coronavirus and time-varying mean

Number of deaths in Germany from coronavirus and the dynamic mean of the Negative Binomial distribution. Log Likelihood value: -278.83.

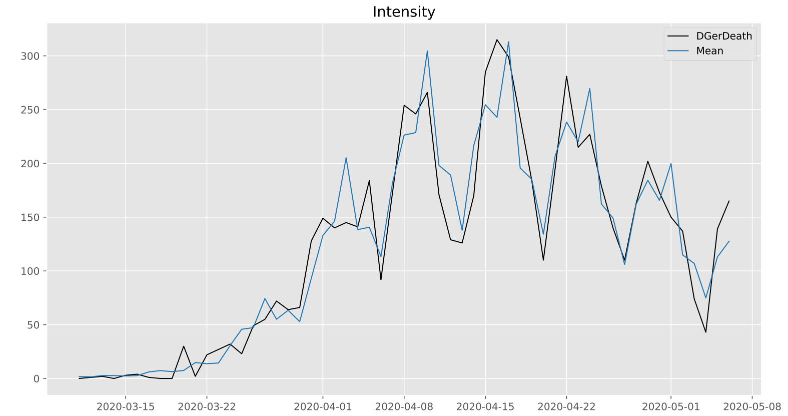


As it turns out, model fit can be further improved by taking into account the daily effect of the time series. The Log Likelihood value increases to -267.78 and the excellent model fit is displayed in Figure A.3. The figures in this section can all be replicated with *TSL*-*SE*, see the case study in Chapter 8 for model details.

Figure A.3

Number of deaths from coronavirus, dynamic mean with daily effect

Number of deaths in Germany from coronavirus and the dynamic mean of the Negative Binomial distribution. The dynamic mean includes a daily seasonal component. Log Likelihood value: -267.78.



Appendix B

Score-driven models

Consider a parametric model for an observed time series $y = (y_1', \ldots, y_n')'$ that is formulated conditionally on a latent $m \times 1$ time-varying parameter vector α_t, for time index $t = 1, \ldots, n$. 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

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

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 $y = (y_1', \ldots, y_n')'$. 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 C

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.

C.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 $t = 1, \ldots, T$ and where NID means Normally Independently Distributed. If we apply the score-driven methodology as discussed in Appendix B 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.

C.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 $t = 1, \dots, T$ 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

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.

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.

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.

Harvey, A. and R. Lit (2020). Coronavirus and the score-driven negative binomial distribution. *Time Series Lab - Article Series* (3).

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 P. Kattuman (2020). Time series models based on growth curves with applications to forecasting coronavirus. Discussion paper, mimeo.

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.

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.

Time Series Lab - Article Series

The *Time Series Lab - Article Series* started in 2020 with Professor S.J. Koopman, Professor A.C. Harvey, and Dr. R.Lit as joint editors. The *Time Series Lab - Article Series* are dedicated to research performed with *Time Series Lab* software. The scope of the series includes the analysis and forecasting of a wide range of time series in fields like economics, finance, sports, climatology, biology, and health science. The following papers appeared in the *Time Series Lab - Article Series*:

003 A.C. Harvey and R. Lit, Coronavirus and the Score-driven Negative Binomial Distribution

002 R. Lit and S.J Koopman, Forecasting the 2020 edition of the Boat Race

001 R. Lit, Forecasting the VIX in the midst of COVID-19