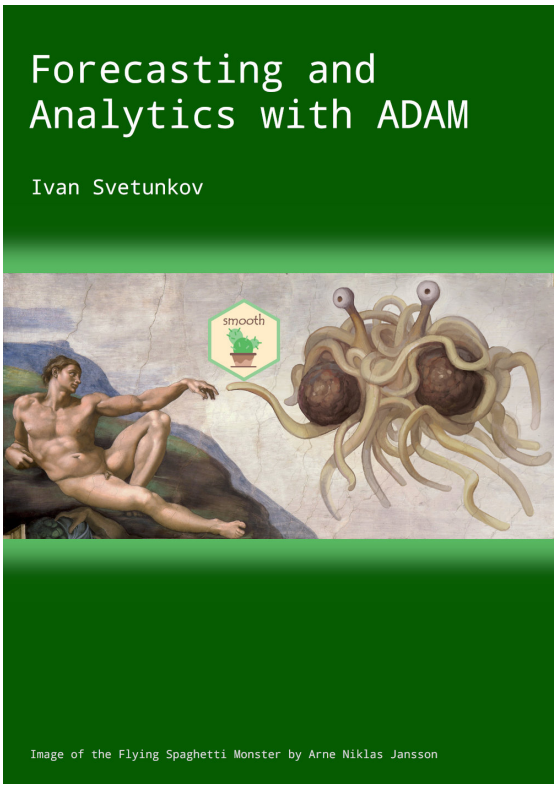
Forecasting and Analytics with ADAM

Ivan Svetunkov

2022-01-13

Contents

Preface



What is ADAM?

ADAM stands for "Augmented Dynamic Adaptive Model". The term "adaptive model" means that the parameters of the model change over time according to some assumed process. The word "dynamic" reflects the idea that the model has time series related components (ETS, ARIMA). Finally, the word "augmented" is included because ADAM is the model that supports additional features not included in the conventional ETS / ARIMA. ADAM is a unified framework for constructing ETS / ARIMA / regression, based on more advanced statistical

instruments. For example, classical ARIMA is built on the assumption of normality of the error term, but ADAM lifts this assumption and allows using other distributions as well (e.g. Generalised Normal, Inverse Gaussian etc). Another example, typically the conventional models are estimated either via the maximisation of the likelihood function or using basic losses like MSE or MAE, but ADAM has a broader spectrum of losses and allows using custom ones. There is much more, and we will discuss different aspects of ADAM in detail later in this textbook. The ADAM model includes:

1. ETS;
2. ARIMA;
3. Regression;
4. TVP regression;
5. Combination of (1), (2) and either (3), or (4);
6. Automatic selection / combination of states for ETS;
7. Automatic orders selection for ARIMA;
8. Variables selection for regression part;
9. Normal and non-normal distributions;
10. Automatic selection of most suitable distributions;
11. Multiple seasonality;
12. Occurrence part of the model to handle zeroes in data (intermittent demand);
13. Handling uncertainty of estimates of parameters.

All these extensions are needed to solve specific real life problems, so we will have examples and case studies later in the book to see how all of this can be used. The `adam()` function from `smooth` package implements ADAM and supports the following features:

1. Model diagnostics using plot() and other methods;
2. Confidence intervals for parameters of models;
3. Automatic outliers detection;
4. Handling missing data;
5. Fine-tuning of persistence vector (smoothing parameters);
6. Fine-tuning of initial values of the state vector (e.g. level / trend / seasonality);
7. Two initialisation options (optimal / backcasting);
8. Advanced and custom loss functions;
9. Provided ETS, ARMA and regression parameters;
10. Fine-tuning of the optimiser (selection of optimisation algorithm and convergence criteria);

This textbook uses two packages from R, namely `greybox`, which focuses on forecasting using regression models, and `smooth`, which implements Single Source of Error (SSOE) state space models for time series analysis and forecasting. The textbook focuses on explaining how ADAM ("ADAM is Dynamic Adaptive Model" - recursive acronym), one of the `smooth` functions (introduced in v3.0.0) works, also showing how it can be used in practice with examples from R.

If you want to run examples from the textbook, two packages are needed (**??**):

```
install.packages("greybox")
install.packages("smooth")
```

Some explanations of functions from the packages are given in my blog: Package greybox for R, Package smooth for R.

An important thing to note is that this textbook **does not use `tidyverse` packages**. I like base R, and, to be honest, I am sure that `tidyverse` packages are great, but I have never needed them in my research. So, I will not use pipeline operators, `tibble` or `tsibble` objects and `ggplot2`. I assume throughout the textbook that you can do all those nice tricks on your own if you want to.

You can use the following to cite the online version of this book:

- Svetunkov, I. (2021) Forecasting and Analytics with ADAM: Lancaster, UK. openforecast.org/adam. Accessed on [current date].

If you use LaTeX, the following can be used instead:

```
@MISC{SvetunkovAdam,
    title = {Forecasting and Analytics with ADAM},
    author = {Ivan Svetunkov},
    howpublished = {OpenForecast},
    note = {(version: [current date])},
    url = {https://openforecast.org/adam/},
    year = {2021}
}
```

License

Acknowledgments

I would like to thank Tobias Schmidt for his help in refining earlier parts of the textbook and correcting grammatical mistakes.

Chapter 1

Introduction

I started writing this book in 2020 during the COVID-19 pandemic, having figured out that it had been more than ten years since publication of the fundamental textbook of **?**. Their original textbook discussed ETS (Error-Trend-Seasonality) model in the Single Source of Error (SSOE) form and that the topic has not been updated substantially since then. If you are interested in learning more about exponential smoothing, then this is a must-read material on the topic.

However, there has been some progress in the area since 2008, and I have developed some models and functions based on SSOE, making the framework more flexible and general. Given that publication of all aspects of these models in peer-reviewed journals would be very time consuming, I have decided to summarise all progress in this book, showing what happens inside the models and how to use the functions in different cases, so that there is a source to refer to.

Many parts of this textbook rely on such topics as a model, scales of information, model uncertainty, likelihood, information criteria and model building. All these topics are discussed in detail in the online textbook of **?**. You are recommended to familiarise yourself with them before moving to ADAM's more advanced modelling topics.

In this chapter, we explain what forecasting is, how it is different from planning and analytics and the main forecasting principles one should follow in order not to fail in trying to predict the future.

1.1 Forecasting, planning and analytics

While there are many definitions of what forecast is, I like the following one, proposed by Sergey Svetunkov (**?**): **Forecast is a scientifically justified assertion about possible states of an object in future**. This definition

does not have the word "probability" in it, because in some cases, forecasts do not rely on rigorous statistical methods and theory of probabilities. For example, the Delphi method allows obtaining judgmental forecasts, typically focusing on what to expect, not on the probability side. An essential word in the definition is "**scientific**". If a prediction is made based on coffee grounds, then it is not a forecast. Judgmental predictions, on the other hand, can be considered as forecasts if a person has a reason behind them. If they do not, this should be called "guesses", not forecasts. Finally, the word "future" is important as well, as it shows the focus of the discipline: without the future, there is no forecasting, only overfitting. As for the definition of **forecasting**, it is a process of producing forecasts – as simple as that.

Forecasting is a vital activity carried by many companies, some of which do it unconsciously or label it as "demand planning" or "predictive analytics". However, there is a difference between the terms "forecasting" and "planning". The latter relies on the former and implies the company's actions to adjust its decisions. For example, if we forecast that the sales will go down, a company should make some marketing decisions to increase the demand on the product. The first part relates to forecasting, while the second relates to planning. If a company does not like a forecast, it should change something in its activities, not in the forecast itself. It is important not to confuse these terms in practice.

Another crucial thing to keep in mind is that any forecasting activity should be done to inform decisions. Forecasting for the sake of forecasting is pointless. Yes, we can forecast the overall number of hospitalisations due to SARS-CoV-2 virus in the world for the next decade, but what decisions can be made based on that? If there are some decisions, then this exercise is worthwhile. If not, then this is just a waste of time.

Example. *Retailers typically need to order some amount of milk that they will sell over the next week. They do not know how much they will sell, so they usually order, hoping to satisfy, let us say, 95% of demand. This situation tells us that the forecasts need to be made a week ahead, they should be cumulative (considering the overall demand during a week before the following order) and that they should focus on an upper bound of a 95% prediction interval. Producing only point forecasts would not be helpful in this situation.*

Related to this is the question of forecasts accuracy. In reality, accurate forecasts do not always translate to good decisions. This is because many different aspects of reality need to be taken into account, and forecasting focuses only on one of them. Capturing the variability of demand correctly is sometimes more useful than producing very accurate point forecasts – this is because many decisions are based on distributions of values rather than on point forecasts. The classical example of this situation is inventory management, where the ordering decisions are made based on quantiles of distribution to form safety stock. Furthermore, the orders are typically done in pallets, so it is not important whether the expected demand is 99 or 95 units if a pallet includes 100 units of a product.

This means that whenever we produce forecasts, we need to consider how they will be used and by whom.

In some cases, accurate forecasts might be wasted if people make decisions differently and/or do not trust what they see. For example, a demand planner might decide that a straight line is not a good point forecast and would start changing the values, introducing noise. This might happen due to a lack of experience, expertise or trust in models, and this means that it is crucial to understand who will use the forecasts and how.

Finally, in practice, not everything can be solved with forecasting. In some cases, companies can make decisions based on other reasons. For example, promotional decisions can be dictated by the existing stock of the product that needs to be moved out. In another case, if the holding costs for a product are low, then there is no need to spend time forecasting the demand on it – a company can implement a simple replenishment policy, ordering, when the stock reaches some threshold. And in times of crisis, some decisions are dictated by the company's financial situation, not by forecasts: you do not need to predict demand on products that are sold out of prestige if they are not profitable, and a company needs to cut costs.

Summarising all the above it makes sense to determine what decisions will be made based on forecast, by whom and how. There is no need to waste time and effort on improving the forecasting accuracy if the process in the company is flawed and forecasts are then ignored, not needed or amended inadequately.

As for analytics, this is a relatively new term, which implies a set of activities based on analysis, forecasting and optimisation to support informed managerial decisions. The term is broad and relies on many research areas, including forecasting, simulations, optimisation etc. In this textbook, we will focus on the forecasting side, occasionally discussing how to analyse the existing processes (thus touching the analytics part) and how various models could help make good practical decisions.

1.2 Forecasting principles

If you have decided that you need to forecast something, it makes sense to keep several important forecasting principles in mind.

First, as discussed earlier, you need to understand why the forecast is required, how it will be used and by whom. Answers to these questions will guide you in deciding what technique to use, how specifically to forecast and what should be reported. For example, if a client does not know machine learning, it might be unwise to use Neural Networks for forecasting – the client will not trust the technique and thus will not trust the forecasts, switching to simpler methods. If the final decision is to order a number of units, it would be more reasonable to produce cumulative forecasts over the lead time (time between the order

and product delivery) and form safety stock based on the model and assumed distribution.

When you understand what to forecast and how the second principle comes into play: select the relevant error measure. You need to decide how to measure the accuracy of forecasting methods, keeping in mind that accuracy needs to be as close to the final decision as possible. For example, if you need to decide the number of nurses for a specific day in the A&E department based on the patients' attendance, then it would be more reasonable to compare models in terms of their quantile performance (see Section @ref(uncertainty)) rather than expectation or median. Thus, it would be more appropriate to calculate pinball loss instead of MAE or RMSE (see details in Section @ref(forecastsEvaluation)).

Third, you should always test your models on a sample of data not seen by them. Train your model on one part of a sample (*train set* or *in-sample*) and test it on another one (*test set* or *holdout sample*). This way, you can have some guarantees that the model will not overfit the data and that it will be reasonable when you need to produce a final forecast. Yes, there are cases when you do not have enough data to do that. All you can do in these situations is use simpler, robust models (for example, damped trend exponential smoothing by **?**; and **?**; or Theta by **?**) and to use judgment in deciding whether the final forecasts are reasonable or not. But in all the other cases, you should test the model on the data it is unaware of. The recommended approach, in this case, is rolling origin, discussed in more detail in Section @ref(rollingOrigin).

Fourth, the forecast horizon should be aligned with specific decisions in practice. If you need predictions for a week ahead, there is no need to produce forecasts for the next 52 weeks. On the one hand, this is costly and excessive; on the other hand, the accuracy measurement will not align with the company's needs. The related issue is the test set (or holdout) size selection. There is no unique guideline for this, but it should not be shorter than the forecasting horizon.

Fifth, the time series aggregation level should be as close to the specific decisions as possible. There is no need to produce forecasts on an hourly level for the next week (168 hours ahead) if the decision is based on the order of a product for the whole week. We would not need such a granularity of data for the decision; aggregating the actual values to the weekly level will do the trick. Still, we would waste a lot of time making complicated models work on an hourly level.

Sixth, you need to have benchmark models. Always compare forecasts from your favourite approach with those from Naïve, global average and/or regression – depending on what you deal with specifically. If your fancy Neural Network performs worse than Naïve, it does not bring value and should not be used in practice. Comparing one Neural Network with another is also not a good idea because Simple Exponential Smoothing (see Section @ref(SES)), being a much simpler model, might beat both networks, and you would never find out about that. If possible, also compare forecasts from the proposed approach with forecasts of other well-established benchmarks, such as ETS (**?**), ARIMA

(**?**) and Theta (**?**).

Finally, when comparing forecasts from different models, you might end up with several very similar performing approaches. If the difference between them is not significant, then the general recommendation is to select the faster and simpler one. This is because simpler models are more difficult to break, and those that work faster are more attractive in practice due to reduced energy consumption (save the planet and stop global warming! **?**).

These principles do not guarantee that you will end up with the most accurate forecasts, but at least you will not end up with unreasonable ones.

1.3 Types of forecasts

Depending on circumstances, we might require different types of forecasts with different characteristics. It is essential to understand what your model produces to measure its performance correctly (see Section @ref(errorMeasures)) and make correct decisions in practice. Several things are typically produced for forecasting purposes. We start with the most popular one.

1.3.1 Point forecasts

The classical and most often produced thing is the point forecast, which corresponds to some trajectory from a model. This, however, might align with different types of statistics depending on the model and its assumptions. In the case of a pure additive model (such as linear regression), the point forecasts correspond to the conditional expectation (**mean**) from the model. The conventional interpretation of this value is that it shows what to expect on average if the situation would repeat itself many times (e.g. if we have the day with similar conditions, then the average temperature will be 10 degrees Celsius). In the case of time series, this interpretation is difficult to digest, given that time does not repeat itself, but this is the best we can have. I will discuss the technicalities of producing conditional expectations from ADAM in Section @ref(ADAMForecastingExpectation).

Another type of point forecast is the (conditional) geometric expectation (**geometric mean**). It typically arises, when the model is applied to the data in logarithms and the final forecast is only exponentiated. This becomes apparent from the following definition of geometric mean:

$$\check{y} = \sqrt[T]{\prod_{t=1}^{T} y_t} = \exp\left(\frac{1}{T}\sum_{t=1}^{T}\log(y_t)\right), (\#eq:GeoMean) \tag{1.1}$$

where y_t is the actual value, and T is the sample size. To use the geometric mean, we need to assume that the actual values can only be positive. Otherwise, the root in c might produce imaginary units (for example, taking a square root out of a negative number) or be equal to zero (if one of the values is zero).

In general, the arithmetic and geometric means are related via the following inequality:

$$\check{y} \leq \mu, (\#eq : GeoAndArithMeans) \tag{1.2}$$

where $\check{y}$ is the geometric mean and μ is the arithmetic one. Although geometric mean makes sense in many contexts, it is more difficult to explain than the arithmetic one to decision makers.

Finally, sometimes **medians** are used in place of point forecasts. In this case, the point forecast splits the sample into two halves and shows the level below which 50% of observations will lie in the future.

Note, the specific type of point forecast will differ from the model used in construction. For example, in the case of the pure additive model, assuming some symmetric distribution (e.g. Normal one), the arithmetic mean, geometric mean and median will coincide. In this case, there is nothing to choose from. On the other hand, a model constructed in logarithms will assume an asymmetric distribution for the original data, leading to the following relation between the means and the median (in case of positively skewed distribution):

$$\check{y} \leq \tilde{y} \leq \mu, (\#eq : GeoAndArithMeansAndMedian) \tag{1.3}$$

where $\tilde{y}$ is the median of distribution.

1.3.2 Quantiles and prediction intervals

As some forecasters say, all point forecasts are wrong. They will never correspond to the actual values because they only capture the model's mean (or median) performance, as discussed in the previous subsection. Everything that is not included in the point forecast can be considered as the uncertainty of demand. For example, we never will be able to say precisely how many cups of coffee we will sell following Monday, but we can at least capture the main tendencies and the uncertainty around our point forecast.

Figure @ref(fig:adamExampleNormal) shows an example with a well-behaved demand, for which the best point forecast is the straight line. To capture the uncertainty of demand, we can construct the prediction interval, which will tell which bound the demand will lie in $1-\alpha$ per cent of cases. The interval in Figure @ref(fig:adamExampleNormal) has the width of 95% ($\alpha = 0.05$) and shows that if the situation is repeated many times, the actual demand will be between 78.83 and 119.99. Capturing the uncertainty correctly is important because real-life decisions need to be made based on the full information, not only on the point forecasts.

We will discuss how to produce prediction intervals in more detail in Section @ref(ADAMForecastingPI). For a more detailed discussion on the concepts of prediction and confidence intervals, see Chapter 5 of **?**.

Another way to capture the uncertainty (related to the prediction interval) is via specific quantiles of distribution. The prediction interval typically has two

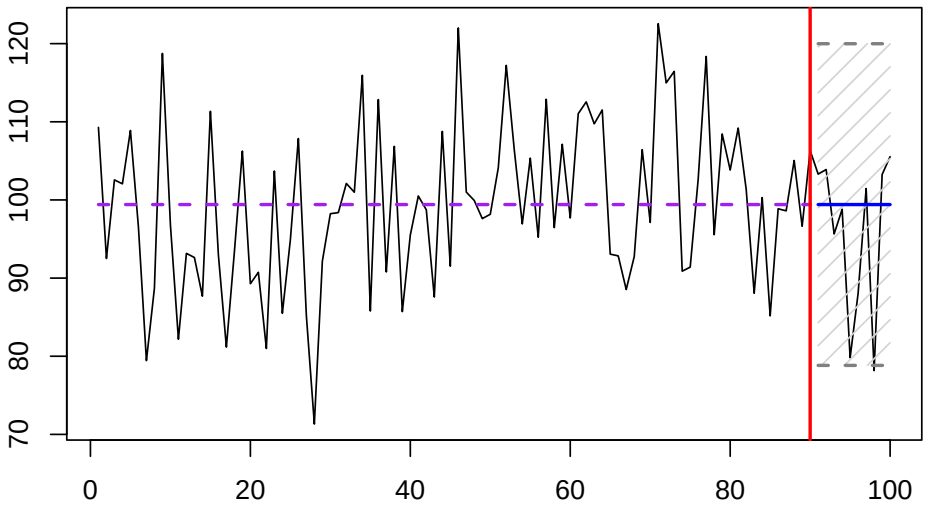


Figure 1.1: (#fig:adamExampleNormal)An example of a well behaved data, point forecast and a 95% prediction interval.

sides, leaving $\frac{\alpha}{2}$ values on the left and the same on the right, outside the bounds of the interval. Instead of producing the interval, in some cases, we might need just a specific quantile, essentially creating the one-sided prediction interval (see Section @ref(forecastingADAMOtherOneSided) for technicalities). The bound in this case will show the particular value below which the pre-selected percentage of cases would lie. This becomes especially useful in such contexts as safety stock calculation (because we are not interested in knowing the lower bound, we want products to satisfy some proportion of demand).

1.3.3 Forecast horizon

Finally, an important aspect in forecasting is the horizon, for which we need to produce forecasts. Depending on the context, we might need:

1. Only a specific value h steps ahead, e.g., the temperature following Monday.
2. All values from 1 to h steps ahead, e.g. how many patients we will have each day next week.
3. Cumulative values for the period from 1 to h steps ahead, e.g. what the cumulative demand over the lead time (the time between the order and product delivery) will be (see discussion in Section @ref(forecastingADAMOtherCumulative)).

It is essential to understand how decisions are made in practice and align them with the forecast horizon. In combination with the point forecasts and prediction intervals discussed above, this will give us an understanding of what to produce

from the model and how. For example, in the case of safety stock calculation, it would be more reasonable to produce quantile of the cumulative over the lead time demand than to produce point forecasts from the model.

1.4 Models, methods and typical assumptions

While we do not aim to fully cover the topic of models, methods, and typical assumptions of statistical models, we need to make several important definitions to clarify what we will discuss in this textbook. For a more detailed discussion, see Chapters 1 and 12 of **?**.

Cambridge dictionary (**?**) defines **method** as a particular way of doing something. So, the method does not necessarily explain how the structure appears or how the error term interacts with it; it only describes how a value is produced. In our context, the forecasting method would be a formula that generates point forecasts based on some parameters and available data. It would not explain how what underlies the data.

Statistical model on the other hand, is a 'mathematical representation of a real phenomenon with a complete specification of distribution and parameters' (**?**). It explains what happens inside the data, reveals the structure and shows how the error term interacts with the structure.

While discussing statistical models, we should also define **true model**. It is "the idealistic statistical model that is correctly specified (has all the necessary components in the correct form), and applied to the data in population" (**?**). Some statisticians also use the term **Data Generating Process** (DGP) when discussing the true model. Still, we need to distinguish between the two terms, as DGP implies that the data is somehow generated using a mathematical formula. In real life, the data is not generated from any function; it comes from a measurement of a complex process, influenced by many factors (e.g. behaviour of a group of customers based on their individual preferences and mental states). The DGP is useful when we want to conduct experiments on simulated data in a controlled environment, but it is not helpful when applying models to the data. Finally, the true model is an abstract notion because it is never known or reachable. But it is still a useful one, as it allows us to see what would happen when we know the model and, more importantly, what would happen when the model we use is wrong.

The related to this definition is the **estimated** or **applied model**, which is the statistical model that is applied to the available sample of data. This model will almost always be wrong because even if we know the specification of the true model for some mysterious reason, we would still need to estimate it on our data. In this case, the estimates of parameters would differ from those in the population, and thus the model will still be wrong.

Mathematically, in the simplest case the true model can be written as:

$$y_t = \mu_{y,t} + \epsilon_t, (\#eq : TrueModel) \tag{1.4}$$

where y_t is the actual value, $\mu_{y,t}$ is the structure and ϵ_t is the true noise. If we manage to capture the structure correctly, the model applied to the sample of data would be written as:

$$y_t = \hat{\mu}_{y,t} + e_t, (\#eq : AppliedModel) \tag{1.5}$$

where $\hat{\mu}_{y,t}$ is the estimate of the structure $\mu_{y,t}$ and e_t is the estimate of the noise ϵ_t (also known as "**residuals**"). If the structure is captured correctly, there would still be a difference between @ref(eq:TrueModel) and @ref(eq:AppliedModel) because the latter is estimated on the data. However, if the sample size increases and we use an adequate estimation procedure, then due to Central Limit Theorem (see Chapter 4 of **?**), the distance between the two models will decrease and asymptotically (with the increase of sample size) e_t would converge to ϵ_t. This does not happen automatically, and some assumptions should hold for this to happen.

1.4.1 Assumptions of statistical models

Very roughly, the typical assumptions of statistical models can be split into the following categories (**?**):

1. Model is correctly specified:

a. We have not omitted important variables in the model (underfitting the data);
b. We do not have redundant variables in the model (overfitting the data);
c. The necessary transformations of the variables are applied;
d. We do not have outliers in the model;

2. Residuals are independent and identically distributed (i.i.d.):

a. There is no autocorrelation in the residuals;
b. The residuals are homoscedastic;
c. The expectation of residuals is zero, no matter what;
d. The variable follows the assumed distribution;
e. More, generally speaking, the distribution of residuals does not change over time;

3. The explanatory variables are not correlated with anything but the response variable:

a. No multicollinearity;
b. No endogeneity.

Many of these assumptions come to the idea that we have correctly captured the structure, meaning that we have not omitted any essential variables, we have

not included the redundant ones, and we transformed all the variables correctly (e.g. took logarithms, where needed). If all these assumptions hold, then we would expect the applied model to converge to the true one with the increase of the sample size. If some of them do not hold, then the point forecasts from our model might be biased, or we might end up producing wider (or narrower) prediction intervals than needed.

These assumptions with their implications on an example of multiple regression are discussed in detail in Chapter 12 of **?**. The diagnostics of dynamic models based on these assumptions is discussed in Chapter @ref(diagnostics).

Chapter 2

Forecasts evaluation

As discussed in Section @ref(forecastingPlanningAnalytics), forecasts should serve a specific purpose. They should not be made "just because" but help make decisions. The decision then dictates the kind of forecast that should be made – its form and its time horizon(s). It also dictates how the forecast should be evaluated – a forecast only being as good as the quality of the decisions it enables.

When you understand how your system works and what sort of forecasts you should produce, you can start an evaluation process, measuring the performance of different forecasting models/methods and selecting the most appropriate for your data. There are various ways to measure and compare their performance.

This chapter discusses the most common approaches, focusing on evaluating point forecasts, then moving towards prediction intervals and quantile forecasts. After that, we discuss how to choose the appropriate error measure and, finally, ensure that the model performs consistently on the available data via rolling origin evaluation and statistical tests.

2.1 Measuring accuracy of point forecasts

We start with a setting in which we are interested in point forecasts only. In this case, we typically begin by splitting the available data into train and test sets, applying the models under consideration to the former, and producing forecasts on the latter, not showing that part to the models. This is called the "fixed origin" approach: we fix the point in time from which to produce forecasts, produce them, calculate some error measure and compare the models.

Different error measures can be used in this case. Which measure to use depends on the specific need. Here we briefly discuss the most important measures and refer to (**???**) for the gory details.

The majority of point forecast measures relies on the following two popular metrics:

Root Mean Squared Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{h}\sum_{j=1}^{h}\left(y_{t+j} - \hat{y}_{t+j}\right)^2}, (\#eq:RMSE) \tag{2.1}$$

and **Mean Absolute Error** (MAE):

$$\text{MAE} = \frac{1}{h}\sum_{j=1}^{h}\left|y_{t+j} - \hat{y}_{t+j}\right|, (\#eq:MAE) \tag{2.2}$$

where y_{t+j} is the actual value j steps ahead from the holdout, $\hat{y}_{t+j}$ is the j steps ahead point forecast, and h is the forecast horizon. As you see, these error measures aggregate the performance of competing forecasting methods across the forecasting horizon, averaging out the specific performances on each j. If this information needs to be retained, the summation can be dropped to obtain "SE" and "AE" values.

It is well-known (see, for example, **?**) that the **mean value of distribution minimises RMSE**, and **the median value minimises MAE**. So, when selecting between the two, you should consider this property. It also implies, for example, that MAE-based error measures should not be used for the evaluation of models on intermittent demand because zero forecast will minimise MAE, when the sample contains more than 50% of zeroes (see, for example, **?**).

Another error measure that has been used in some cases is Root Mean Squared Logarithmic Error (RMSLE, see discussion in **?**):

$$\text{RMSLE} = \exp\left(\sqrt{\frac{1}{h}\sum_{j=1}^{h}\left(\log y_{t+j} - \log \hat{y}_{t+j}\right)^2}\right).(\#eq:RMSLE) \tag{2.3}$$

It assumes that the actual values and the forecasts are positive and is **minimised by geometric mean**. I have added the exponentiation in the formula @ref(eq:RMSLE), which is sometimes omitted, bringing the metric to the original scale to have the same units as the actual values y_t.

The main difference in the three measures arises when the data we deal with is not symmetric – in that case, the arithmetic, geometric means, and median will be different. Thus, the error measures might recommend different approaches depending on what specifically is produced as a point forecast from the model (see discussion in Section @ref(typesOfForecastsPoint)).

2.1.1 An example in R

In order to see how the error measures work, we consider the following example based on a couple of forecasting functions from `smooth` package for R (**?**, and **?**) and measures from `greybox`:

```
y <- rnorm(100,100,10)
model1 <- es(y,h=10,holdout=TRUE)
model2 <- ces(y,h=10,holdout=TRUE)
# RMSE
setNames(sqrt(c(MSE(model1$holdout, model1$forecast),
                MSE(model2$holdout, model2$forecast))),
         c("ETS","CES"))
# MAE
setNames(c(MAE(model1$holdout, model1$forecast),
           MAE(model2$holdout, model2$forecast)),
         c("ETS","CES"))
# RMSLE
setNames(exp(sqrt(c(MSE(log(model1$holdout), log(model1$forecast)),
                    MSE(log(model2$holdout), log(model2$forecast))))),
         c("ETS","CES"))
```

```
##      ETS      CES
## 9.492744 9.494683
```

```
##      ETS      CES
## 7.678865 7.678846
```

```
##      ETS      CES
## 1.095623 1.095626
```

Given that the distribution of the original data is symmetric, all three error measures should generally recommend the same model. But also, given that the data we generated for the example is stationary, the two models will produce very similar forecasts. The values above demonstrate the latter point – the accuracy between the two models is roughly the same. Note that we have evaluated the same point forecasts from the models using different error measures, which would be wrong if the distribution of the data was skewed. In our case, the model relies on normal distribution so that the point forecast would coincide with arithmetic mean, geometric mean and median.

2.1.2 Aggregating error measures

The main advantage of the error measures discussed in the previous subsection is that they are straightforward and have a clear interpretation: they reflect the "average" distances between the point forecasts and the observed values. They are perfect for the work with only one time series. However, they are not suitable when a set of time series is under consideration, and a forecasting method needs to be selected across them. This is because they are scale-dependent and contain specific units: if you measure sales of apples in units, then MAE, RMSE and RMSLE (defined in equation @ref(eq:RMSLE)) will show the error in units as well. And, as we know, you should not add up apples with oranges – the result might not make sense.

To tackle this issue, different error scaling techniques have been proposed, resulting in a zoo of error measures:

1. MAPE – Mean Absolute Percentage Error:

$$\text{MAPE} = \frac{1}{h}\sum_{j=1}^{h}\frac{|y_{t+j} - \hat{y}_{t+j}|}{y_{t+j}}, (\#eq : MAPE) \tag{2.4}$$

2. MASE – Mean Absolute Scaled Error (**?**):

$$\text{MASE} = \frac{1}{h}\sum_{j=1}^{h}\frac{|y_{t+j} - \hat{y}_{t+j}|}{\bar{\Delta}_y}, (\#eq : MASE) \tag{2.5}$$

   where $\bar{\Delta}_y = \frac{1}{t-1}\sum_{j=2}^{t}|\Delta y_j|$ is the mean absolute value of the first differences $\Delta y_j = y_j - y_{j-1}$ of the in-sample data;
3. rMAE – Relative Mean Absolute Error (**?**):

$$\text{rMAE} = \frac{\text{MAE}_a}{\text{MAE}_b}, (\#eq : rMAE) \tag{2.6}$$

   where MAE_a is the mean absolute error of the model under consideration and MAE_b is the MAE of the benchmark model;
4. sMAE – scaled Mean Absolute Error (**?**):

$$\text{sMAE} = \frac{\text{MAE}}{\bar{y}}, (\#eq : sMAE) \tag{2.7}$$

   where $\bar{y}$ is the mean of the in-sample data.
5. and others.

There is no "best" error measure. All have advantages and disadvantages, but some are more suitable in some circumstances than others. For example:

1. MAPE is scale sensitive (if the actual values are measured in thousands of units, the resulting error will be much lower than in the case of hundreds of units) and cannot be estimated on data with zeroes. Furthermore, this error measure is biased, preferring when models underforecast the data (see, for example, **?**) and is not minimised by either mean or median, but by an unknown quantity. Accidentally, in the case of Log-Normal distribution, it is minimised by the mode (see discussion in **?**). Despite all the limitations, MAPE has a simple interpretation as it shows the percentage error (as the name suggests);
2. MASE avoids the disadvantages of MAPE but does so at the cost of a simple interpretation. This is because of the division by the first differences of the data (some interpret this as an in-sample one-step-ahead Naïve forecast, which does not simplify the interpretation);
3. rMAE avoids the disadvantages of MAPE, has a simple interpretation (it shows by how much one model is better than the other), but fails, when

either MAE_a or MAE_b for a specific time series is equal to zero. In practice, this happens more often than desired and can be considered a severe error measure limitation. Furthermore, the increase of rMAE (for example, with the increase of sample size) might mean that either the method A is performing better than before or that the method B is performing worse than before – it is not possible to tell the difference unless the denominator in the formula @ref(eq:rMAE) is fixed;

4. sMAE avoids the disadvantages of MAPE has an interpretation close to it but breaks down when the data has a trend.

When comparing different forecasting methods, it might make sense to calculate several error measures for comparison. The choice of metric might depend on the specific needs of the forecaster. Here are a few rules of thumb, however:

- If you want a robust measure that works consistently, but you do not care about the interpretation, then go with MASE.
- If you want an interpretation, go with rMAE or sMAE. Just keep in mind that if you decide to use rMAE or any other relative measure, you might get attacked by its creator, Andrey Davydenko, who might blame you for stealing his creation, even if you put a reference to his work.
- If the data does not exhibit trends (stationary), you can use sMAE.
- You should typically avoid MAPE and other percentage error measures because the actual values highly influence them in the holdout.

Furthermore, similarly to the measures above, there have been proposed RMSE-based scaled and relative error metrics, which would measure the performance of methods in terms of means rather than medians. Here is a brief list of some of them:

1. RMSSE – Root Mean Squared Scaled Error (**?**):

$$\text{RMSSE} = \sqrt{\frac{1}{h}\sum_{j=1}^{h}\frac{(y_{t+j} - \hat{y}_{t+j})^2}{\bar{\Delta}_y^2}}; (\#eq : RMSSE) \tag{2.8}$$

2. rRMSE – Relative Root Mean Squared Error (**?**):

$$\text{rRMSE} = \frac{\text{RMSE}_a}{\text{RMSE}_b}; (\#eq : rRMSE) \tag{2.9}$$

3. sRMSE – scaled Root Mean Squared Error (**?**):

$$\text{sRMSE} = \frac{\text{RMSE}}{\bar{y}}.(\#eq : sRMSE) \tag{2.10}$$

Similarly, RMSSLE, rRMSLE and sRMSLE can be proposed, using the same principles as in @ref(eq:RMSSE), @ref(eq:rRMSE) and @ref(eq:sRMSE) to assess performance of models in terms of geometric means across time series.

Finally, when aggregating the performance of forecasting methods across several time series, sometimes it makes sense to look at the distribution of errors

– this way, you will know which of the methods fails seriously and which does a consistently good job. If an aggregate measure is needed, then **use mean and median of the chosen metric**. The mean might be non-finite for some error measures, especially when a method performs exceptionally poorly on a time series (an outlier). Still, it will give you information about the average performance of the method and might flag extreme cases. The median at the same time is robust to outliers and is always calculable, no matter what the distribution of the error term is. Furthermore, comparing mean and median might provide additional information about the tail of distribution without reverting to histograms or the calculation of quantiles. **?** argues for the use of geometric mean for relative and scaled measures. Still, as discussed earlier, it might become equal to zero or infinity if the data contains outliers (e.g. two cases, when one of the methods produced a perfect forecast, or the benchmark in rMAE produced a perfect forecast). At the same time, if the distribution of errors in logarithms is symmetric (which is the main argument of **?**), then geometric mean will coincide with median, so there is no point in calculating the geometric mean at all.

2.1.3 Demonstration in R

In R, there is a variety of functions that calculate the error measures discussed above, including the `accuracy()` function from `forecast` package and `measures()` from `greybox`. Here is an example of how the measures can be calculated based on a couple of forecasting functions from `smooth` package for R and a set of generated time series:

```
# Apply a model to a test data to get names of error measures
y <- rnorm(100,100,10)
test <- es(y,h=10,holdout=TRUE)
# Define number of iterations
nsim <- 100
# Create an array for nsim time series, 2 models and a set of error measures
errorMeasures <- array(NA, c(nsim,2,length(test$accuracy)),
                       dimnames=list(NULL,c("ETS","CES"),
                                     names(test$accuracy)))
# Start a loop for nsim iterations
for(i in 1:nsim){
  # Generate a time series
  y <- rnorm(100,100,10)
  # Apply ETS
  testModel1 <- es(y,"ANN",h=10,holdout=TRUE)
  errorMeasures[i,1,] <- measures(testModel1$holdout, testModel1$forecast,
                                  actuals(testModel1))
  # Apply CES
  testModel2 <- ces(y,h=10,holdout=TRUE)
  errorMeasures[i,2,] <- measures(testModel2$holdout, testModel2$forecast,
```

```
                                    actuals(testModel2))
}
```

The default benchmark method for relative measures above is Naïve. To see how the distribution of error measures would look like, we can produce violinplots via the `vioplot()` function from the `vioplot` package. We will focus on rRMSE measure (see Figure @ref(fig:errorMeasuresrRMSEDistLog)).

```
vioplot::vioplot(errorMeasures[,,"rRMSE"])
```

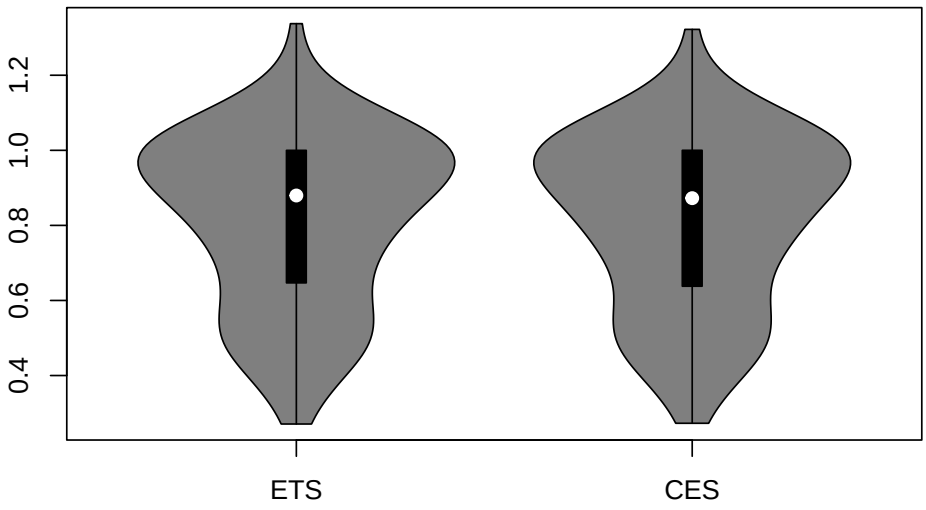


Figure 2.1: (#fig:errorMeasuresrRMSEDist)Distribution of rRMSE on the original scale.

The distributions in Figure @ref(fig:errorMeasuresrRMSEDistLog) look similar, and it is hard to tell which one performs better. Besides, they do not look symmetric, so we will take logarithms to see if this fixes the issue with the skewness (Figure @ref(fig:errorMeasuresrRMSEDistLog)).

```
vioplot::vioplot(log(errorMeasures[,,"rRMSE"]))
```

Figure @ref(fig:errorMeasuresrRMSEDistLog) demonstrates that the distribution in logarithms is skewed, so the geometric mean in this case would not be suitable and might provide a misleading information. So, we calculate mean and median rRMSE to check the overall performance of the two models:

```
# Calculate mean rRMSE
apply(errorMeasures[,,"rRMSE"],2,mean)
```

```
##       ETS       CES
## 0.8163452 0.8135303
```

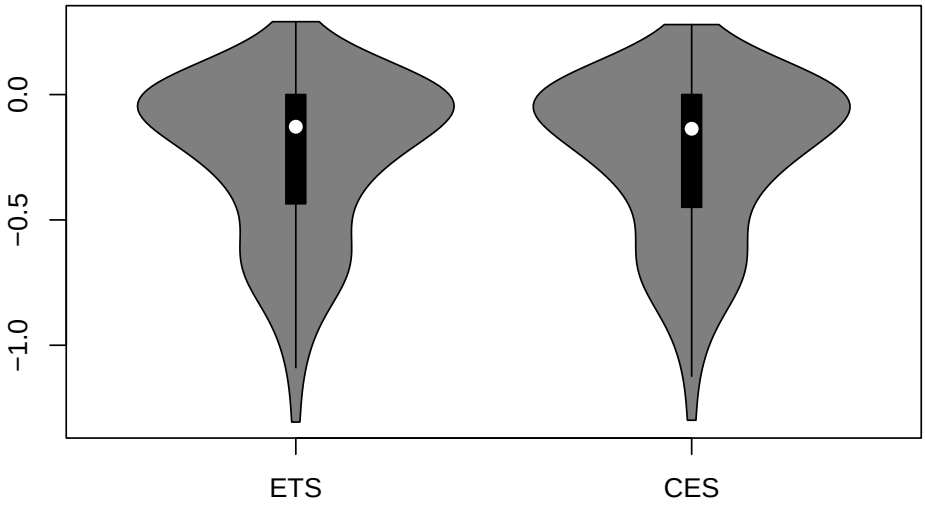


Figure 2.2: (#fig:errorMeasuresrRMSEDistLog)Distribution of rRMSE on the log scale.

```
# Calculate median rRMSE
apply(errorMeasures[,,"rRMSE"],2,median)
```

```
##       ETS       CES
## 0.8796325 0.8725286
```

Based on the values above, we cannot make any solid conclusion about the performance of the two models: in terms of both mean and median rRMSE, CES is doing slightly better, but the difference between the two models is not substantial, so we can probably choose the one that is easier to work with.

2.2 Measuring uncertainty

As discussed in Section @ref(typesOfForecastsInterval), point forecasts are not sufficient for adequate decision making – prediction intervals and quantiles are needed to capture the uncertainty of demand around the point forecast. As with point forecasts, multiple measures can be used to evaluate them. There are several useful measures for the evaluation of intervals. We start with the simplest of them, coverage.

1. **Coverage** shows the percentage of observations lying inside the interval:

$$\text{coverage} = \frac{1}{h}\sum_{j=1}^{h} \left(\mathbb{1}(y_{t+j} < l_{t+j}) \times \mathbb{1}(y_{t+j} > u_{t+j}) \right), (\#eq:coverage) \tag{2.11}$$

where l_{t+j} is the lower bound and u_{t+j} is the upper bound of the interval and $\mathbb{1}(\cdot)$ is the indicator function, returning one, when the condition is true and zero otherwise. Ideally, the coverage should be equal to the confidence level of the interval, but in reality, this can only be observed asymptotically (with the increase of the sample size), as the sample size increases due to the inheritted randomness of any sample estimates of parameters;

2. **Range** shows the width of the prediction interval:

$$\text{range} = \frac{1}{h}\sum_{j=1}^{h}(u_{t+j} - l_{t+j}); (\#eq : range) \tag{2.12}$$

If the range of interval from one model is lower than the range of the other one, then the uncertainty about the future values is lower for the first one. However, the narrower interval might not include as many actual values in the holdout sample, leading to lower coverage. So, there is a natural trade-off between the two measures.

3. **Mean Interval Score** (**?**) combines the properties of the previous two measures:

$$\begin{aligned}\text{MIS} =& \frac{1}{h}\sum_{j=1}^{h}\Bigg((u_{t+j} - l_{t+j}) + \frac{2}{\alpha}(l_{t+j} - y_{t+j})\mathbb{1}(y_{t+j} < l_{t+j}) + \\ & \frac{2}{\alpha}(y_{t+j} - u_{t+j})\mathbb{1}(y_{t+j} > u_{t+j})\Bigg),\end{aligned} \quad (\#eq : MIS) \tag{2.13}$$

where α is the significance level. If the actual values lie outside of the interval, they get penalised with a ratio of $\frac{2}{\alpha}$, proportional to the distance from the interval bound. At the same time the width of the interval positively influences the value of the measure: the wider the interval, the higher the score. The ideal model with MIS = 0 should have all the actual values in the holdout lying on the bounds of the interval and $u_{t+j} = l_{t+j}$, implying that the bounds coincide with each other and that there is no uncertainty about the future (which is not possible in real life).

4. **Pinball Score** (**?**) measures the accuracy of models in terms of specific quantiles (this is usually applied to different quantiles produced from the model, not just to the lower and upper bounds of 95% interval):

$$\text{PS} = (1-\alpha)\sum_{y_{t+j}<q_{t+j},j=1,\dots,h}|y_{t+j}-q_{t+j}| + \alpha\sum_{y_{t+j}\geq q_{t+j},j=1,\dots,h}|y_{t+j}-q_{t+j}|, (\#eq : pinball) \tag{2.14}$$

where q_{t+j} is the value of the specific quantile of the distribution. PS shows how well we capture the specific quantile in the data. The lower the value of pinball is, the closer the bound is to the specific quantile of the holdout distribution. If the PS is equal to zero, then we have done the perfect job in hitting that specific quantile. The main issue with PS is that it is very difficult to assess the quantiles correctly on small

samples. For example, in order to get a better idea of how the 0.975 quantile performs, we would need to have at least 40 observations, so that 39 of them would be expected to lie below this bound $\left(\frac{39}{40} = 0.975\right)$. In fact, quantiles are not always uniquely defined (see, for example, **?**), which makes the measurement difficult.

Similar to the pinball function, it is possible to propose the expectile-based score, but while it has good statistical properties (**?**), it is more difficult to interpret.

Range, MIS and PS are unit-dependent. To aggregate them over several time series, they need to be scaled either via division by either the in-sample mean or in-sample mean absolute differences to obtain the scaled counterparts of the measures or via division by the values from the benchmark model to get the relative one. The idea here would be similar to what we discussed for MAE and RMSE in Section @ref(errorMeasures).

If you are interested in the model's overall performance, then MIS provides this information. However, it does not show what happens specifically inside and is difficult to interpret. Coverage and range are easier to interpret but only give information about the specific prediction interval. They typically must be traded off against each other (i.e. one can either cover more or have a narrower interval). Academics prefer pinball for uncertainty assessment, as it shows more detailed information about the predictive distribution from each model. However, while it is easier to interpret than MIS, it is still not as straightforward as coverage and range. So, the selection of the measure depends on your specific situation and the understanding of statistics by decision-makers.

2.2.1 Example in R

Continuing the example from Section @ref(errorMeasures), we could produce prediction intervals from the two models and compare them using MIS and pinball:

```
model1Forecast <- forecast(model1,h=10,interval="p",level=0.95)
model2Forecast <- forecast(model2,h=10,interval="p",level=0.95)

# Mean Interval Score
setNames(c(MIS(model1$holdout, model1Forecast$lower,
               model1Forecast$upper, 0.95),
           MIS(model2$holdout, model2Forecast$lower,
               model2Forecast$upper, 0.95)),
         c("Model 1", "Model 2"))
```

```
##  Model 1  Model 2
## 39.68038 46.91441
```

```
# Pinball for the upper bound
setNames(c(pinball(model1$holdout, model1Forecast$upper, 0.975),
```

```
          pinball(model2$holdout, model2Forecast$upper, 0.975)),
        c("Model 1", "Model 2"))
```

```
##  Model 1  Model 2
## 5.402511 6.703046
```

```
# Pinball for the lower bound
setNames(c(pinball(model1$holdout, model1Forecast$lower, 0.025),
          pinball(model2$holdout, model2Forecast$lower, 0.025)),
        c("Model 1", "Model 2"))
```

```
##  Model 1  Model 2
## 4.517584 5.025557
```

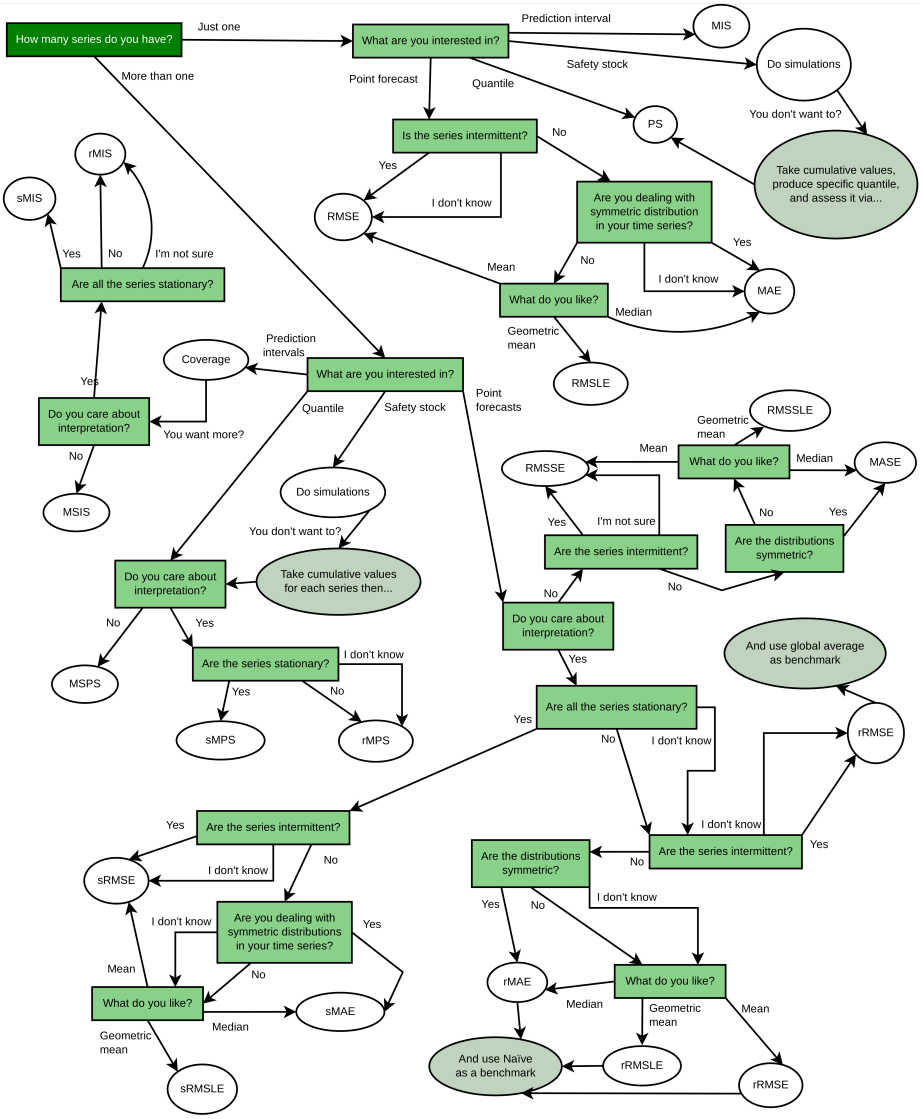
```
# Coverage
setNames(c(mean(model1$holdout > model1Forecast$lower &
                model1$holdout < model1Forecast$upper),
           mean(model2$holdout > model2Forecast$lower &
                model2$holdout < model2Forecast$upper)),
        c("Model 1", "Model 2"))
```

```
## Model 1 Model 2
##     0.9     0.9
```

The values above imply that the first model (ETS) performed better than the second one in terms of MIS and pinball loss (the interval was narrower). However, these measures do not tell much in terms of the performance of models when only applied to one time series. To see more solid results, we need to apply models to a set of time series, produce prediction intervals, calculate measures and then look at their aggregate performance, e.g. via mean / median or quantiles. The loop and the analysis would be similar to the one discussed in Section @ref(#errorMeasuresExampleBig), so we do not repeat it here.

2.3 How to choose appropriate error measure

While, in general, the selection of error measures should be dictated by the specific problem at hand, some guidelines might be helpful in the process. I have summarised them in the flowchart in Figure @ref(fig:errorMeasuresFlowChart).



The flowchart does not provide excessive options and simplifies the possible process. It does not discuss the quantile and interval measures in detail, as there are many options for them in this direction, and the idea of the flowchart is to list the most important ones. The aim of this is to provide a guideline for selection based on:

1. Number of time series under consideration. If there are several of them and you need to aggregate the error measure, you need to use either scaled or relative ones. In the case of just one time series, you do not need to scale the error measure;
2. What specifically you want to measure: point forecasts, quantiles, prediction interval or something else;
3. Whether the interpretability of the error measure is essential or not. If not, then scaled measures similar to **?** can be used. If yes, then the choice is between relative and scaled using mean measures;
4. Whether the data is stationary or not. If it is, then it is safe to use scaled measures similar to **?** because the division by in-sample mean would be meaningful. Otherwise, you should either use **?** scaling or relative measures;
5. Whether the data is intermittent or not. If it is and you are interested in point forecasts, then you should use RMSE based measures – other measures might recommend zero forecast as the best one;
6. Symmetry of distribution of demand. If it is symmetric (which does not happen very often), then the median will coincide with the mean and geometric mean, and it would not be important whether to use RMSE-, MAE- or RMSLE- based measure. In that case, just use a MAE-based one;
7. What you need (denoted as "What do you like?" in the flowchart). If you are interested in mean performance, then use RMSE based measures. The median minimises MAE, and the geometric mean minimises RMSLE. This relates to the discussion in Section @ref(typesOfForecasts).

The point forecast related error measures have been discussed in Section @ref(errorMeasures), while the interval and quantile ones – in Section @ref(uncertainty).

You can also download this flowchart in pdf format via this link.

2.4 Rolling origin

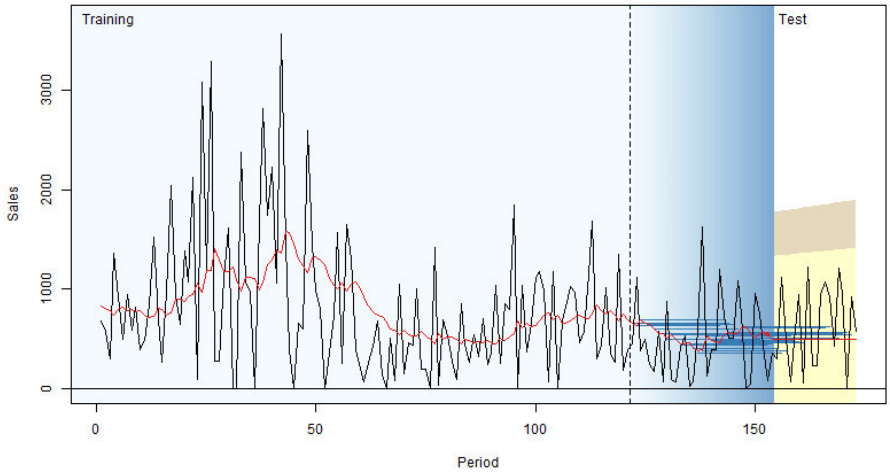
Remark. *The text in this section is based on the vignette for the greybox package, written by the author of this textbook.*

When there is a need to select the most appropriate forecasting model or method for the data, the forecaster usually splits the sample into two parts: in-sample (aka "training set") and holdout sample (aka out-sample or "test set"). The model is estimated on the in-sample, and its forecasting performance is evaluated using some error measure on the holdout sample.

Using this procedure only once is known as "fixed origin" evaluation. However, this might give a misleading impression of the accuracy of forecasting methods. If, for example, the time series contains outliers or level shifts, a poor model

might perform better in fixed origin evaluation than a more appropriate one. Besides, a good performance might happen by chance. So it makes sense to have a more robust evaluation technique. An alternative procedure known as "rolling origin" evaluation is one such technique.

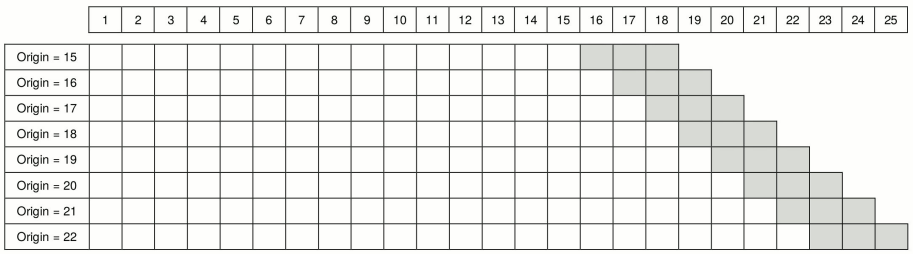
In rolling origin evaluation, the forecasting origin is repeatedly moved forward, and forecasts are produced from each origin (**?**). This technique allows obtaining several forecast errors for time series, which gives a better understanding of how the models perform. This can be considered a time series analogue to cross-validation techniques (**?**). Here is a simple graphical representation, courtesy of Nikos Kourentzes.



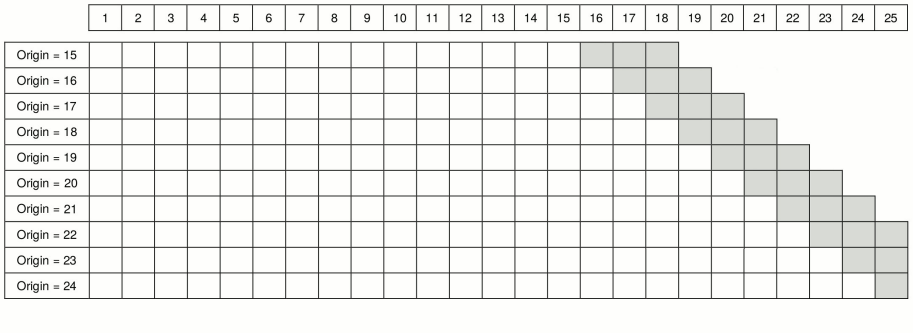
There are different options of how this can be done.

2.4.1 Principles of Rolling origin

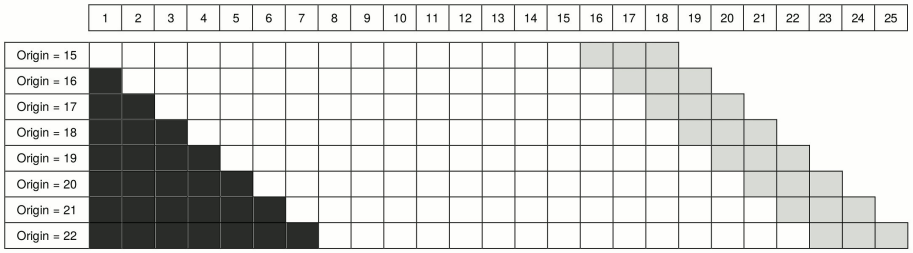
Figure @ref(fig:ROProcessCO) (**?**) illustrates the basic idea behind rolling origin. White cells correspond to the in-sample data, while the light grey cells correspond to the three-steps-ahead forecasts. The time series in the figure has 25 observations, and forecasts are produced for eight origins starting from observation 15. The model is estimated on the first in-sample set, and forecasts are created for the holdout. Next, another observation is added to the end of the in-sample set, the test set is advanced, and the procedure is repeated. The process stops when there is no more data left. This is a rolling origin with a **constant holdout** sample size. As a result of this procedure, eight one to three steps ahead forecasts are produced. Based on them, we can calculate the preferred error measures and choose the best performing model (see Section @ref(errorMeasuresAggregate)).



Another option for producing forecasts via rolling origin would be to continue with rolling origin even when the test sample is smaller than the forecast horizon, as shown in Figure @ref(fig:ROProcessNoCO). In this case, the procedure continues until origin 22, when the last complete set of three-steps-ahead forecasts can be produced but continues with a decreasing forecasting horizon. So the two-steps-ahead forecast is produced from origin 23, and only a one-step-ahead forecast is produced from origin 24. As a result, we obtain ten one-step-ahead forecasts, nine two-steps-ahead forecasts and eight three-steps-ahead forecasts. This is a rolling origin with a **non-constant holdout** sample size, which can be helpful with small samples when we don't have enough observations.



Finally, in both cases above, we had the **increasing in-sample** size. However, we might need a **constant in-sample** for some research purposes. Figure @ref(fig:ROProcessCOCI) demonstrates such a setup. In this case, in each iteration, we add an observation to the end of the in-sample series and remove one from the beginning (dark grey cells).



2.4.2 Rolling origin in R

The function `ro()` from the `greybox` package (written by Yves Sagaert and Ivan Svetunkov in 2016 on the way to the International Symposium on Forecasting) implements the rolling origin evaluation for any function you like with a predefined `call` and returns the desired `value`. It heavily relies on the two variables: `call` and `value` – so it is pretty important to understand how to formulate them to get the desired results. `ro()` is a very flexible function, but as a result, it is not very simple. In this subsection, we will see how it works on a couple of examples.

We start with a simple example, generating a series from normal distribution:

```
y <- rnorm(100,100,10)
```

We use an ARIMA(0,1,1) model implemented in the `stats` package (this model is discussed in Section @ref(ARIMA)):

```
ourCall <- "predict(arima(x=data,order=c(0,1,1)),n.ahead=h)"
```

The call that we specify includes two important elements: `data` and `h`. `data` specifies where the in-sample values are located in the function that we want to use, and **it needs to be called "data"** in the call; `h` will tell our function, where the forecasting horizon is specified in the selected function. Note that in this example we use `arima(x=data,order=c(0,1,1))`, which produces a desired ARIMA(0,1,1) model and then we use `predict(..., n.ahead=h)`, which produces an h steps ahead forecast from that model.

Having the call, we also need to specify what the function should return. This can be the conditional mean (point forecasts), prediction intervals, the parameters of a model, or, in fact, anything that the model returns (e.g. name of the fitted model and its likelihood). However, there are some differences in what `ro()` returns depending on what the function returns. If it is a vector, then `ro()` will produce a matrix (with values for each origin in columns). If it is a matrix, then an array is returned. Finally, if it is a list, then a list of lists is returned.

In order not to overcomplicate things, we start from collecting the conditional mean from the `predict()` function:

```
ourValue <- c("pred")
```

NOTE: If you do not specify the value to return, the function will try to return everything, but it might fail, especially if many values are returned. So, to be on the safe side, **always provide the `value`, when possible**.

Now that we have specified `ourCall` and `ourValue`, we can produce forecasts from the model using rolling origin. Let's say that we want three-steps-ahead forecasts and eight origins with the default values of all the other parameters:

```
returnedValues1 <- ro(y, h=3, origins=8,
                      call=ourCall, value=ourValue)
```

The same can be achieved using the following loop:

```
obs <- 100
roh <- 8
h <- 3
data <- y
returnedValues1 <- setNames(vector("list",3),
                            c("actuals","holdout","pred"))
returnedValues1$actuals <- y
returnedValues1$holdout <- returnedValues1$pred <- matrix(NA,h,roh,
                                  dimnames=list(paste0("h",1:h),
                                                paste0("origin",1:roh)))
for(i in 1:roh){
  testModel <- arima(x=data[1:(obs-roh+i-h)],order=c(0,1,1))
  returnedValues1$holdout[,i] <- data[-c(1:(obs-roh+i-h))]
  returnedValues1$pred[,i] <- predict(testModel, n.ahead=h)$pred
}
```

The function returns a list with all the values that we asked for plus the actual values from the holdout sample. We can calculate some basic error measure based on those values, for example, scaled Absolute Error (?):

```
apply(abs(returnedValues1$holdout - returnedValues1$pred),
      1, mean, na.rm=TRUE) /
  mean(returnedValues1$actuals)
```

```
##         h1         h2         h3
## 0.08814781 0.09038073 0.08028730
```

In this example, we use the `apply()` function to distinguish between the different forecasting horizons and have an idea of how the model performs for each of them. These numbers do not tell us much on their own, but if we compare the performance of this model with another one, we could infer if one model is more appropriate for the data than the other one. For example, applying ARIMA(1,1,2) to the same data, we will get:

```
ourCall <- "predict(arima(x=data,order=c(1,1,2)),n.ahead=h)"
returnedValues2 <- ro(y, h=3, origins=8,
                      call=ourCall, value=ourValue)
apply(abs(returnedValues2$holdout - returnedValues2$pred),
      1, mean, na.rm=TRUE) /
  mean(returnedValues2$actuals)
```

```
##         h1         h2         h3
## 0.08403870 0.09061090 0.07949088
```

Comparing these errors with the ones from the previous model, we can conclude which of the approaches is more suitable for the data.

We can also plot the forecasts from the rolling origin, which shows how the models behave:

```
par(mfcol=c(2,1), mar=c(4,4,1,1))
plot(returnedValues1)
plot(returnedValues2)
```

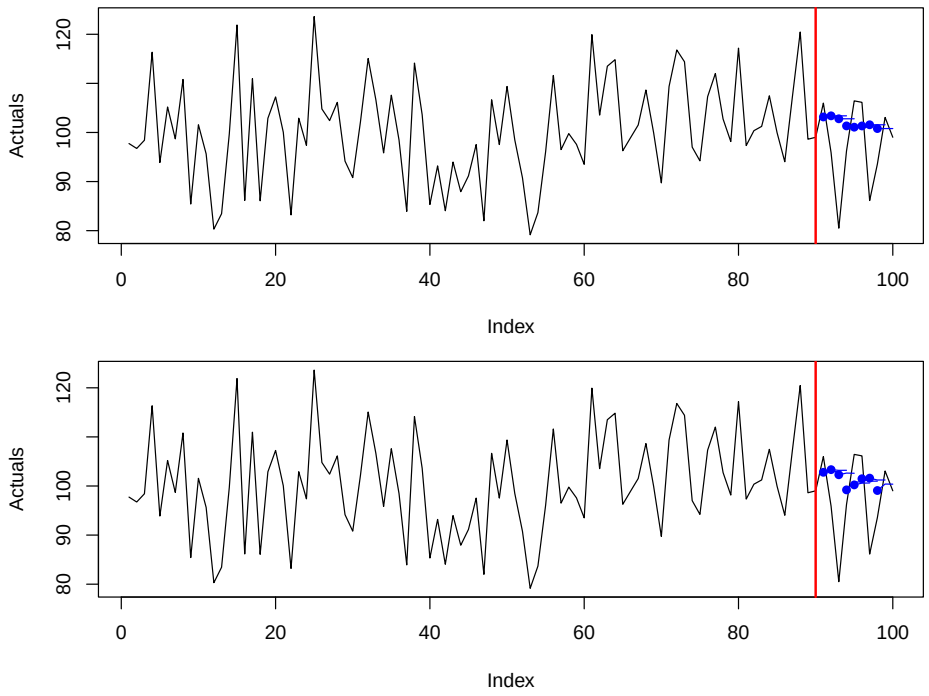


Figure 2.3: (#fig:roExample01)Rolling origin performance of two forecasting methods

In Figure @ref(fig:roExample01), the forecasts from different origins are close to each other. This is because the data is stationary, and both models produce flat lines as forecasts.

The rolling origin function from the `greybox` package also allows working with explanatory variables and returning prediction intervals if needed. Some further examples are discussed in the vignette of the package: `vignette("ro","greybox")`.

Practically speaking, if we have a set of forecasts from different models we can analyse the distribution of error measures and come to conclusions about

performance of models. Here is an example with analysis of performance for $h = 1$ based on absolute errors:

```
aeValuesh1 <- cbind(abs(returnedValues1$holdout -
                        returnedValues1$pred)[1,],
                    abs(returnedValues1$holdout -
                        returnedValues2$pred)[1,])
colnames(aeValuesh1) <- c("ARIMA(0,1,1)","ARIMA(1,1,2)")
boxplot(aeValuesh1)
points(apply(aeValuesh1,2,mean),pch=16,col="red")
```

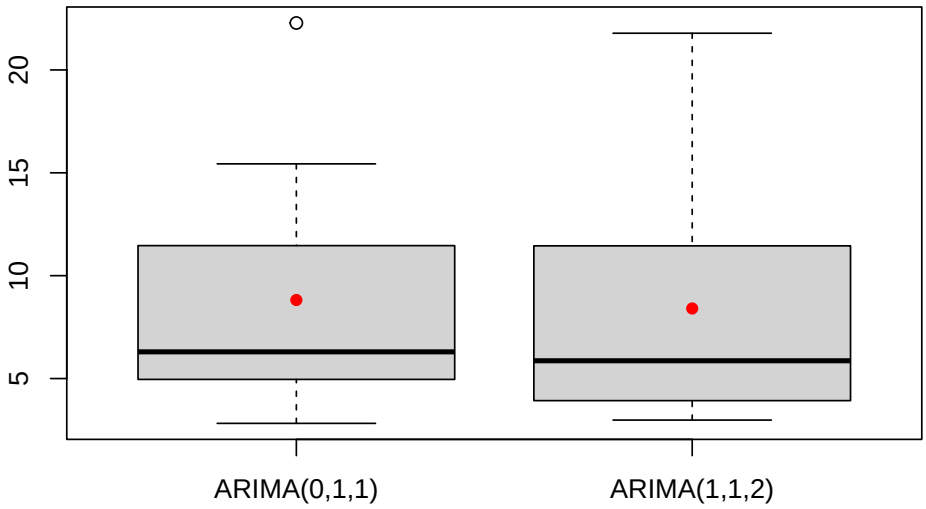


Figure 2.4: (#fig:roExample02)Boxplots of error measures of two methods.

The boxplots in Figure @ref(fig:roExample02) can be interpreted as any other boxplots applied to random variables (see, for example, discussion in Section 2.2 of **?**).

2.5 Statistical comparison of forecasts

After applying several competing models to the data and obtaining a distribution of error terms, we might find that some performed very similarly. In this case, there might be a question, whether the difference is significant and which of the forecasting models we should select. Consider the following artificial example, where we have four competing models and measure their performance in terms of RMSSE:

```
smallCompetition <- matrix(NA, 100, 4,
                           dimnames=list(NULL,
```

```
                                              paste0("Method",c(1:4))))
smallCompetition[,1] <- rnorm(100,1,0.35)
smallCompetition[,2] <- rnorm(100,1.2,0.2)
smallCompetition[,3] <- runif(100,0.5,1.5)
smallCompetition[,4] <- rlnorm(100,0,0.3)
```

We can check the mean and median error measures in this example in order to see, how the methods perform overall:

```
overalResults <- matrix(c(colMeans(smallCompetition),
                          apply(smallCompetition, 2, median)),
                        4, 2, dimnames=list(colnames(smallCompetition),
                                            c("Mean","Median")))
round(overalResults,5)
```

```
##             Mean  Median
## Method1 1.04148 0.97523
## Method2 1.23750 1.23572
## Method3 1.00213 1.02843
## Method4 0.97789 0.92591
```

In this artificial example, it looks like the most accurate method in terms of mean and median RMSSE is Method 4, and the least accurate one is Method 2. However, the difference in terms of accuracy between methods 1, 3 and 4 does not look substantial. So, should we conclude that Method 4 is the best? Let's first look at the distribution of errors using `vioplot()` function from `vioplot` package (Figure @ref(fig:smallCompetitionBoxplot)).

```
vioplot::vioplot(smallCompetition)
points(colMeans(smallCompetition), col="red", pch=16)
```

The violin plots in Figure @ref(fig:smallCompetitionBoxplot) show that the distribution of errors for Method 2 is shifted higher than the distributions of other methods. It also looks like Method 2 is working more consistently, meaning that the variability of the errors is lower (the size of the box on the graph). It is difficult to tell whether Method 1 is better than Methods 3 and 4 or not – their boxes intersect and roughly look similar, with Method 4 having a slightly shorter box and Method 3 having the box slightly lower positioned.

This is all the basics of descriptive statistics, which allows concluding that in general, Methods 1, 3 and 4 do a better job than Method 2. This is also reflected in the mean and median error measures discussed above. So, what should we conclude?

We should not make hasty decisions, and we should remember that we are dealing with a sample of data (100 time series), so inevitably, the performance of methods will change if we try them on different data sets. If we had a population of all the time series in the world, we could run our methods and

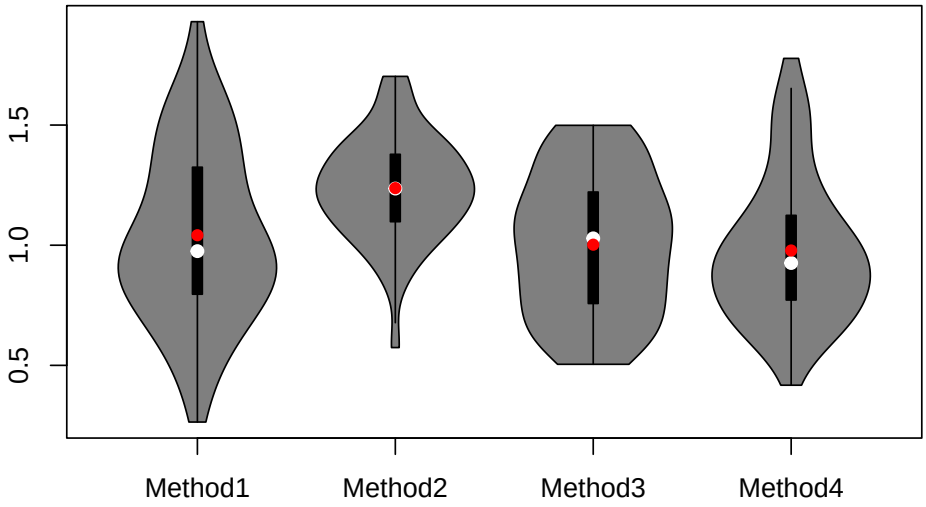


Figure 2.5: (#fig:smallCompetitionBoxplot)Boxplot of RMSE for the artificial example

make a more solid conclusion about their performances. But here, we deal with a sample. So it might make sense to see whether the difference in the performance of methods is significant. How should we do that?

First, we can **compare means** of distributions of errors using a parametric statistical test. We can try F-test (**?**), which will tell us whether the mean performance of methods is similar or not. Unfortunately, this will not tell us how the methods compare. But t-test (**?**) could be used to do that instead for pairwise comparison. One could also use a regression model with dummy variables for methods, giving us parameters and their confidence intervals (based on t-statistics), telling us how the means of methods compare. However, F-test, t-test and t-statistics from regression rely on strong assumptions related to the distribution of the means of error measures (normality). If we had a large sample (e.g. a thousand of series) and well-behaved distribution, we could try it, hoping that the central limit theorem would work and might get something relatively meaningful. However, on 100 observations, this still could be an issue, especially given that the distribution of error measures is typically asymmetric (the estimate of mean might be biased, which leads to many issues).

Second, we could **compare medians** of distributions of errors. They are robust to outliers, so their estimates should not be too biased in case of skewed distributions on smaller samples. To have a general understanding of performance (is everything the same or is there at least one method that performs differently), we could try the Friedman test (**?**), which could be considered a nonparametric alternative of F-test. This should work in our case but won't tell us how specif-

ically the methods compare. We could try the Wilcoxon signed-ranks test (**?**), which could be considered a nonparametric counterpart of the t-test. However, it only applies to two variables, while we want to compare four.

Luckily, there is the Nemenyi test (**?**), which is equivalent to the MCB test (**??**). What the test does, is it ranks the performance of methods for each time series and then takes the mean of those ranks and produces confidence bounds for those means. The means of ranks correspond to medians, so by using this test, we compare medians of errors of different methods. If the confidence bounds for different methods intersect, we can conclude that the medians are not different from a statistical point of view. Otherwise, we can see which of the methods has a higher rank and which has the lower one. There are different ways to present the test results, and there are several R functions that implement it, including `nemenyi()` from the `tsutils` package. However, we will use the function `rmcb()` from the `greybox`, which has more flexible plotting capabilities, supporting all the default parameters for the `plot()` method.

```
smallCompetitionTest <- rmcb(smallCompetition, plottype="none")
smallCompetitionTest
plot(smallCompetitionTest, "mcb", main="")
```

```
## Regression for Multiple Comparison with the Best
## The significance level is 5%
## The number of observations is 100, the number of methods is 4
## Significance test p-value: 0
```

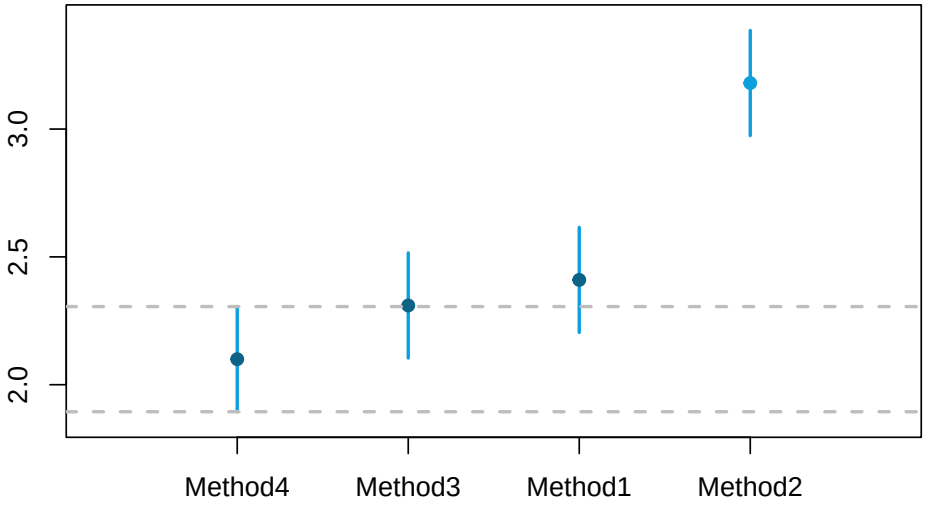


Figure 2.6: (#fig:mcbForCompetition)MCB test results for small competition.

Figure @ref(fig:mcbForCompetition) shows that Methods 1, 3 and 4 are not statistically different – their intervals intersect, so we cannot tell the difference

between them, even though the mean rank of Method 4 is lower than for the other methods. Method 2, on the other hand, is significantly worse than the other methods: it has the highest mean rank of all, and its interval does not intersect intervals of other methods.

Note that while this is a good way of presenting the results, all the MCB test does is a comparison of mean ranks. It does not tell much about the distribution of errors and neglects the distances between values (i.e. 0.1 is lower than 0.11, so the first method has a lower rank, which is precisely the same result as with comparing 0.1 and 100). This happens because by doing the test, we move from a numerical scale to the ordinal one (see Section 1.2 of **?**). Finally, like any other statistical test, it will get its power when the sample increases. We know that the null hypothesis "variables are equal to each other" in reality is always wrong (see Section 5.3 of **?**), so the increase of sample size will lead at some point to the correct conclusion: methods are statistically different. Here is a demonstration of this assertion:

```
largeCompetition <-
  matrix(NA, 100000, 4,
         dimnames=list(NULL, paste0("Method",c(1:4))))
# Generate data
largeCompetition[,1] <- rnorm(100000,1,0.35)
largeCompetition[,2] <- rnorm(100000,1.2,0.2)
largeCompetition[,3] <- runif(100000,0.5,1.5)
largeCompetition[,4] <- rlnorm(100000,0,0.3)
# Run the test
largeCompetitionTest <- rmcb(largeCompetition, plottype="none")
plot(largeCompetitionTest, "mcb", main="")
```

In the plot in Figure @ref(fig:mcbForCompetitionLarge), Method 4 has become significantly worse than Methods 1 and 3 in terms of mean ranks (note that it was winning in the small competition). The difference between Methods 1 and 3 is still not significant, but it would become if we continue increasing the sample size. This example tells us that we need to be careful when selecting the best method, as this might change under different circumstances. At least we knew from the start that Method 2 was not suitable.

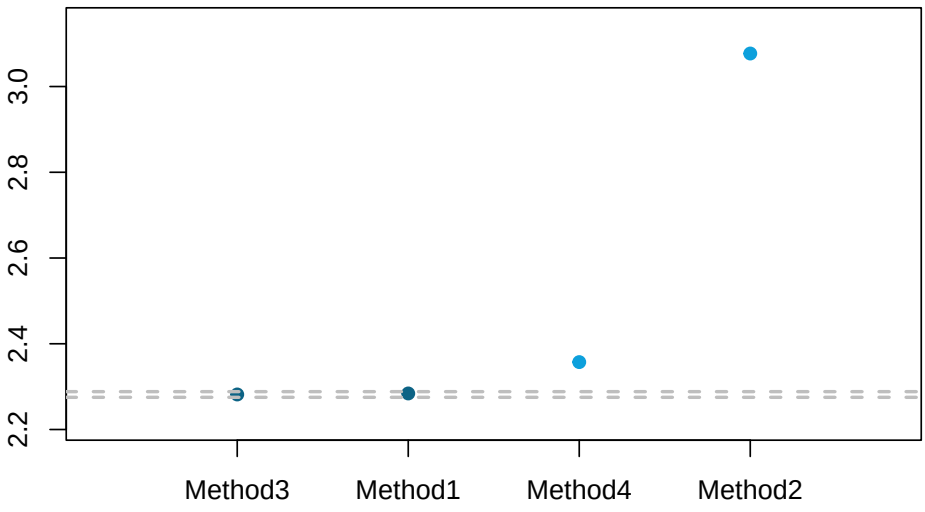


Figure 2.7: (#fig:mcbForCompetitionLarge)MCB test results for large competition.

Chapter 3

From time series components to ETS

Before we turn to the state space framework, ETS, ARIMA and other models, we need to discuss time series decomposition and the ETS taxonomy. These topics lie at the heart of ETS models and are essential for understanding further material.

In this chapter, we start with a discussion of time series components, then move to the idea of decomposing time series into distinct components and then to the conventional ETS taxonomy, as formulated by **?**, demonstrating its connection with the previous topics.

3.1 Time series components

The main idea behind many forecasting techniques is that any time series can contain several unobservable components, such as:

1. **Level** of the series – the average value for a specific time period,
2. **Growth** of the series – the average increase or decrease of the value over a period of time,
3. **Seasonality** – a pattern that repeats itself with a fixed periodicity.
4. **Error** – unexplainable white noise.

The level is the fundamental component that is present in any time series. In the simplest form (without variability), when plotted on its own without other components, it will look like a straight line, shown, for example, in Figure @ref(fig:levelExample).

```
level <- rep(100,40)
plot(ts(level, frequency=4),
```

```
     type="l", xlab="Time", ylab="Sales", ylim=c(80,160))
```

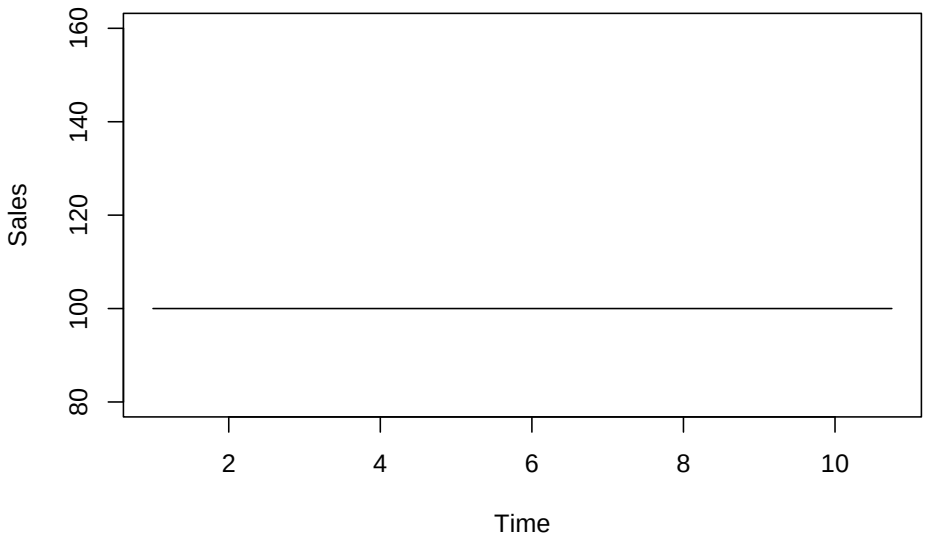


Figure 3.1: (#fig:levelExample)Level of time series without any variability.

If the time series exhibits growth, the level will change depending on the observation. For example, if the growth is positive and constant, we can update the level in Figure @ref(fig:levelExample) to have a straight line with a non-zero slope as shown in Figure @ref(fig:trendExample).

```
growth <- c(1:40)
plot(ts(level+growth, frequency=4),
     type="l", xlab="Time", ylab="Sales", ylim=c(80,160))
```

The seasonal pattern will introduce some similarities from one period to another. This pattern does not have to literally be seasonal, like beer sales being higher in Summer than in Winter (season of the year). Any pattern with a fixed periodicity works: the number of hospital visitors is higher on Mondays than on Saturdays or Sundays because people tend to stay at home over the weekend. This can be considered as the day of week seasonality. Furthermore, if we deal with hourly data, sales are higher during the daytime than at night (hour of the day seasonality). Adding a deterministic seasonal component to the example above will result in fluctuations around the straight line, as shown in Figure @ref(fig:seasonalExample).

```
seasonal <- rep(c(10,15,-20,-5),10)
plot(ts(level+growth+seasonal, frequency=4),
     type="l", xlab="Time", ylab="Sales", ylim=c(80,160))
```

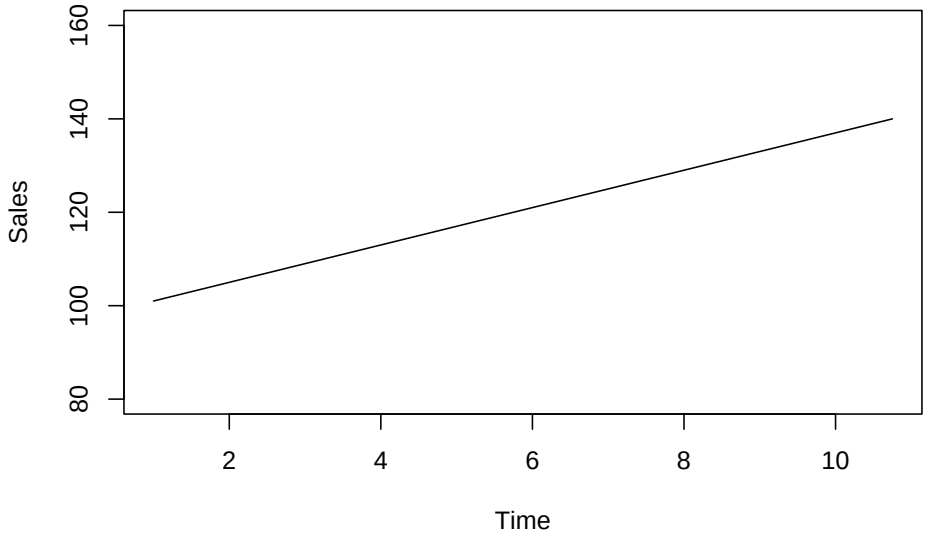


Figure 3.2: (#fig:trendExample)Time series with a positive trend and no variability.

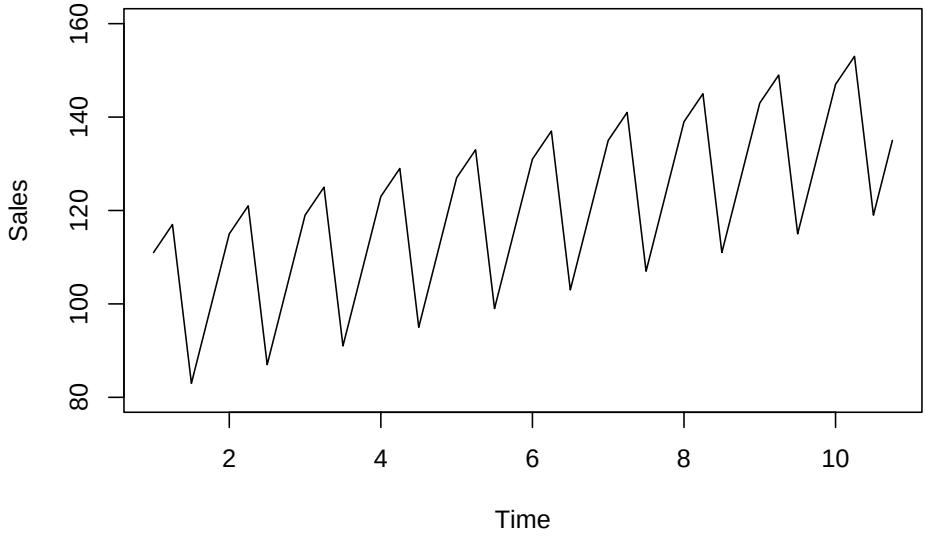


Figure 3.3: (#fig:seasonalExample)Time series with a positive trend, seasonal pattern and no variability.

Finally, we can introduce the random error to the plots above to have a more realistic time series as shown in Figure @ref(fig:allExample).

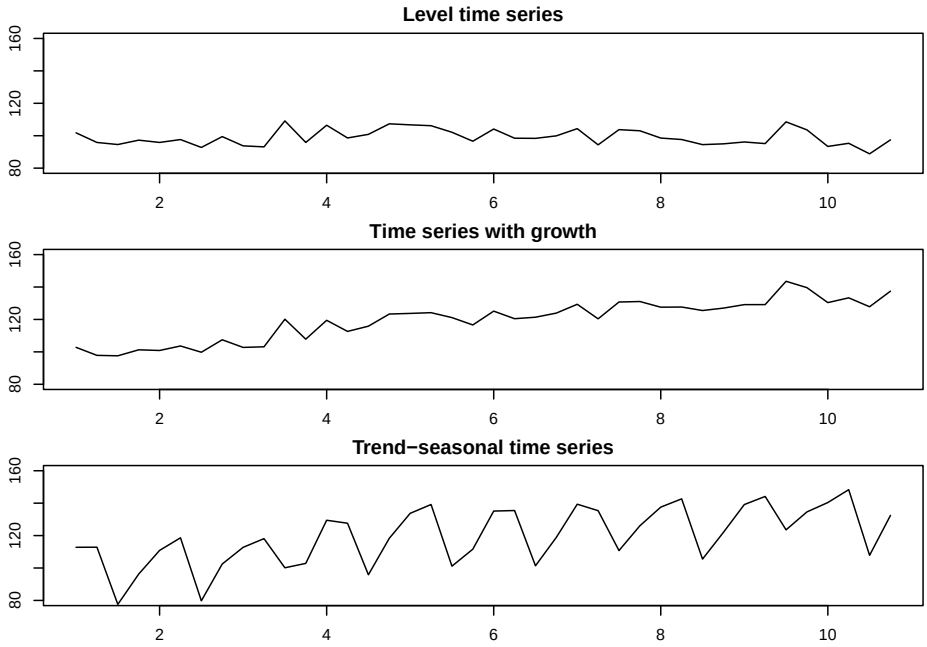


Figure 3.4: (#fig:allExample)Time series with random errors.

Figure @ref(fig:allExample) shows artificial time series with the above components. The level, growth, and seasonal components in those plots are **deterministic**, they are fixed and do not evolve over time (growth is positive and equal to 1 from year to year). However, in real life, typically, these components will have more complex dynamics, changing over time and thus demonstrating their **stochastic** nature. For example, in the case of stochastic seasonality, the seasonal shape might change, and instead of having peaks in sales in January, the data would exhibit peaks in May due to the change in consumers' behaviour.

Note that each textbook and paper might use slightly different names to refer to the abovementioned components. For example, in classical decomposition (**?**), it is assumed that (1) and (2) jointly represent a "trend" component so that a model will contain error, trend and seasonality. This decomposition has modifications, which include cyclical component(s).

When it comes to ETS, the growth component (2) is called "trend", so the model consists of the four components: level, trend, seasonal and the error term. We will use the ETS formulation in this textbook. According to this formulation, the components can interact in one of two ways: additively or multiplicatively.

The pure additive model, in this case, can be summarised as:

$$y_t = l_{t-1} + b_{t-1} + s_{t-m} + \epsilon_t, (\#eq : PureAdditive) \tag{3.1}$$

where l_{t-1} is the level, b_{t-1} is the trend, s_{t-m} is the seasonal component with periodicity m (e.g. 12 for months of year data, implying that something is repeated every 12 months) – all these components are produced on the previous observations and are used on the current one. Finally, ϵ_t is the error term, which follows some distribution and has zero mean. The pure additive models were plotted in Figure @ref(fig:allExample). Similarly, the pure multiplicative model is:

$$y_t = l_{t-1} b_{t-1} s_{t-m} \varepsilon_t, (\#eq : PureMultiplicative) \tag{3.2}$$

where ε_t is the error term with a mean of one. The interpretation of the model @ref(eq:PureAdditive) is that the different components add up to each other, so, for example, the sales increase over time by the value b_{t-1}, each January they typically change by the amount s_{t-m}, and that there is still some randomness in the model. The pure additive models can be applied to data with positive, negative and zero values. In the case of the multiplicative model @ref(eq:PureMultiplicative), the interpretation is different, showing by how many times the sales change over time and from one season to another. The sales, in this case, will change every January by $(s_{t-m} - 1)\%$ from the baseline. The model @ref(eq: PureMultiplicative) only works on strictly positive data (data with purely negative values are also possible but rare in practice).

It is also possible to define mixed models in which, for example, the trend is additive but the other components are multiplicative:

$$y_t = (l_{t-1} + b_{t-1}) s_{t-m} \varepsilon_t (\#eq : MixedAdditiveTrend) \tag{3.3}$$

These models work well in practice when the data has large values far from zero. In other cases, however, they might break and produce strange results (e.g. negative values on positive data), so the conventional decomposition techniques only consider the pure models.

3.2 Classical Seasonal Decomposition

3.2.1 How to do?

One of the classical textbook methods for decomposing the time series into unobservable components is "Classical Seasonal Decomposition" (**?**). It assumes either a pure additive or pure multiplicative model, is done using centred moving averages and is focused on approximation, not forecasting. The idea of the method can be summarised in the following steps:

1. Decide which of the models to use based on the type of seasonality in the data: additive @ref(eq:PureAdditive) or multiplicative @ref(eq:PureMultiplicative)

2. Smooth the data using a centred moving average (CMA) of order equal to the periodicity of the data m. If m is the an number then the formula is:

$$d_t = \frac{1}{m} \sum_{i=-(m-1)/2}^{(m-1)/2} y_{t+i}, (\#eq : CMAOdd) \tag{3.4}$$

which means that, for example, the value on Thursday is the average of values from Monday to Sunday. If m is an even number then a different weighting scheme is typically used, involving the inclusion of additional an value:

$$d_t = \frac{1}{m} \left(\frac{1}{2} \left(y_{t+(m-1)/2} + y_{t-(m-1)/2} \right) + \sum_{i=-(m-2)/2}^{(m-2)/2} y_{t+i} \right), (\#eq : CMAEven) \tag{3.5}$$

which means that we use half of the December of the previous year and half of the December of the current year to calculate the centred moving average in June. The values d_t are placed in the middle of the window going through the series (e.g. on Thursday, the average will contain values from Monday to Sunday).

The resulting series is deseasonalised. When we average, e.g. sales in a year, we automatically remove the potential seasonality, which can be observed each month individually. A drawback of using CMA is that we inevitably lose $\frac{m}{2}$ observations at the beginning and the end of the series.

In R, the `ma()` function from the `forecast` package implements CMA.

3. De-trend the data:

- For the additive decomposition this is done using: $y'_t = y_t - d_t$;
- For the multiplicative decomposition, it is: $y'_t = \frac{y_t}{d_t}$;

4. If the data is seasonal, the average value for each period is calculated based on the de-trended series. e.g. we produce average seasonal indices for each January, February, etc. This will give us the set of seasonal indices s_t;
5. Calculate the residuals based on what you assume in the model:

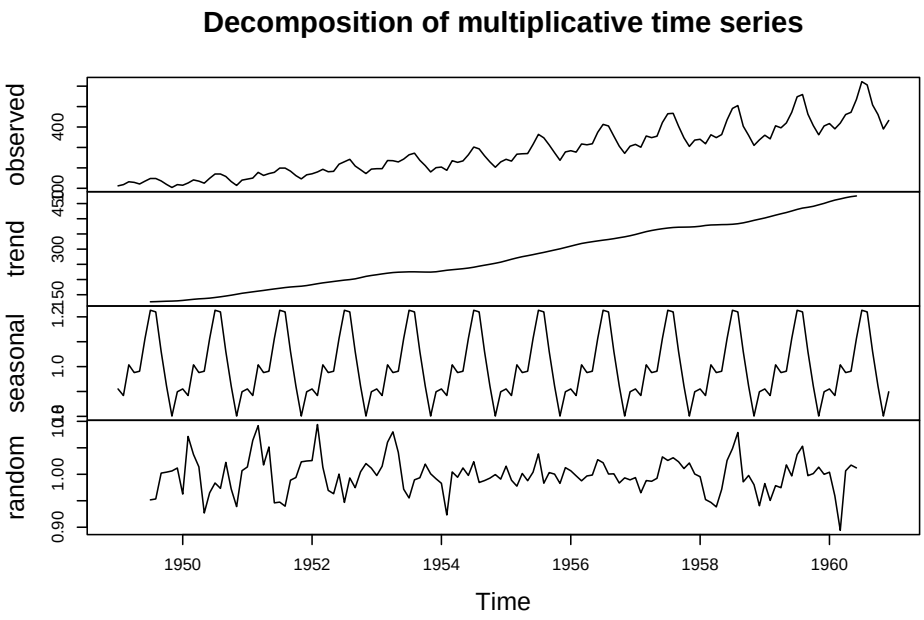
- additive seasonality: $e_t = y_t - d_t - s_t$;
- multiplicative seasonality: $e_t = \frac{y_t}{d_t s_t}$;
- no seasonality: $e_t = y'_t$.

Note that the functions in R typically allow selecting between additive and multiplicative seasonality. There is no option for "none", and so even if the data is not seasonal, you will nonetheless get values for s_t in the output. Also, notice that the classical decomposition assumes that there is a deseasonalised series d_t but does not make any further split of this variable into level l_t and trend b_t.

3.2.2 A couple of examples

An example of the classical decomposition in R is the `decompose()` function from `stats` package. Here is an example with pure multiplicative model and `AirPassengers` data:

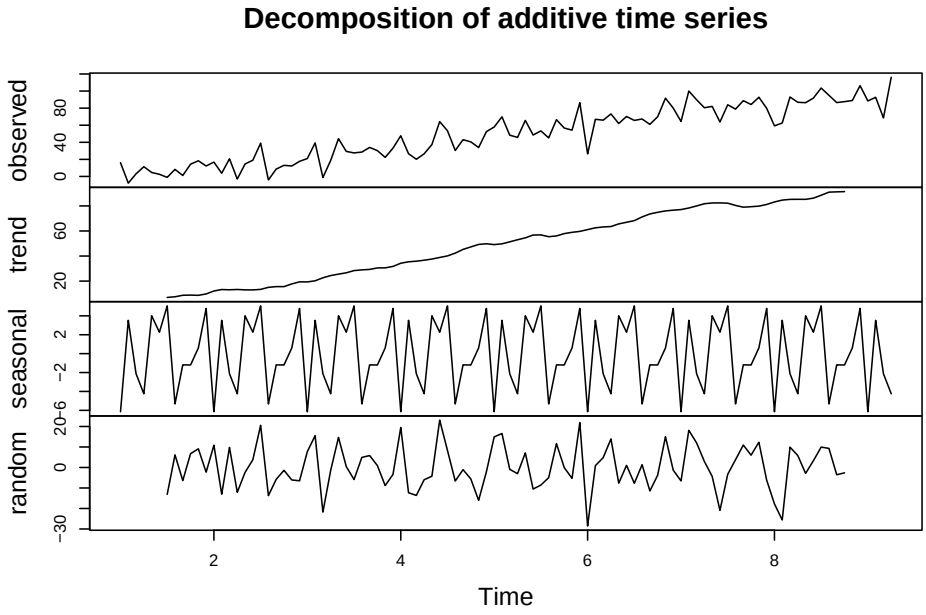
```
ourDecomposition <- decompose(AirPassengers,
                              type="multiplicative")
plot(ourDecomposition)
```



We can see that the function has smoothed the original series and produced the seasonal indices. Note that the trend component has gaps at the beginning and the end. This is because the method relies on CMA (see above). Note also that the error term still contains some seasonal elements, which is a downside of such a simple decomposition procedure. However, the lack of precision in this method is compensated by the simplicity and speed of calculation. Note again that the trend component in `decompose()` function is in fact $d_t = l_t + b_t$.

Here is an example of decomposition of the **non-seasonal data** (we assume pure additive model in this example):

```
y <- ts(c(1:100)+rnorm(100,0,10),frequency=12)
ourDecomposition <- decompose(y, type="additive")
plot(ourDecomposition)
```



As you can see, the original data is not seasonal, but the decomposition assumes that it is and proceeds with the default approach returning a seasonal component. You get what you ask for.

3.2.3 Other techniques

There are other techniques that decompose series into error, trend and seasonal components but make different assumptions about each component. The general procedure, however, always remains the same: (1) smooth the original series, (2) extract the seasonal components, (3) smooth them out. The methods differ in the smoother they use (LOESS, e.g., uses a bisquare function instead of CMA), and in some cases, multiple rounds of smoothing are performed to make sure that the components are split correctly.

There are many functions in R that implement seasonal decomposition. Here is a small selection:

- `decomp()` from the `tsutils` package does classical decomposition and fills in the tail and head of the smoothed trend with forecasts from exponential smoothing;
- `stl()` from the `stats` package uses a different approach – seasonal decomposition via LOESS. It is an iterative algorithm that smoothes the states and allows them to evolve over time. So, for example, the seasonal component in STL can change;
- `mstl()` from the `forecast` package does the STL for data with several seasonalities;
- `msdecompose()` from the `smooth` package does a classical decomposition

for multiple seasonal series.

3.2.4 "Why bother?"

"Why to decompose?" you may wonder at this point. Understanding the idea behind decompositions and how to perform them helps understand ETS, which relies on it. From a practical point of view, it can be helpful if you want to see if there is a trend in the data and whether the residuals contain outliers or not. It will *not* show you if the data is seasonal as the seasonality is *assumed* in the decomposition (I stress this because many students think otherwise). Additionally, when seasonality cannot be added to the model under consideration decomposing the series, predicting the trend and then reseasonalising can be a viable solution. Finally, the values from the decomposition can be used as starting points for the estimation of components in ETS or other dynamic models relying on the error-trend-seasonality.

3.3 Simple forecasting methods

Now that we understand that time series might contain different components and that there are approaches for their decomposition, we can introduce several simple forecasting methods that can be used in practice, at least as benchmarks. Their usage aligns with the idea of forecasting principles discussed in Section @ref(forecastingPrinciples).

3.3.1 Naïve

Naïve is one of the simplest forecasting methods. According to each, the one-step-ahead forecast is equal to the most recent actual value:

$$\hat{y}_t = y_{t-1}.(\#eq : Naive) \tag{3.6}$$

Using this approach might sound naïve indeed, but there are cases where it is very hard to outperform the method. Consider an example with temperature forecasting. If we want to know what the temperature outside will be in 5 minutes, then Naïve would be typically very accurate: the temperature in 5 minutes will be the same as it is right now. The statistical model underlying Naïve is called "Random Walk" and is written as:

$$y_t = y_{t-1} + \epsilon_t.(\#eq : RandomWalk) \tag{3.7}$$

The variability of ϵ_t will impact the speed of change of the data: the higher it is, the more rapid the values will change. Random Walk and Naïve can be represented in Figure @ref(fig:naiveExample). In the example below, we use a simple moving average (discussed later in Section @ref(SMA)) of order 1 to generate the data from Random Walk and then produce forecasts using Naïve.

```
y <- sim.sma(1, 120)
testModel <- sma(y$data, 1,
                 h=10, holdout=TRUE)
plot(testModel, 7, main="")
```

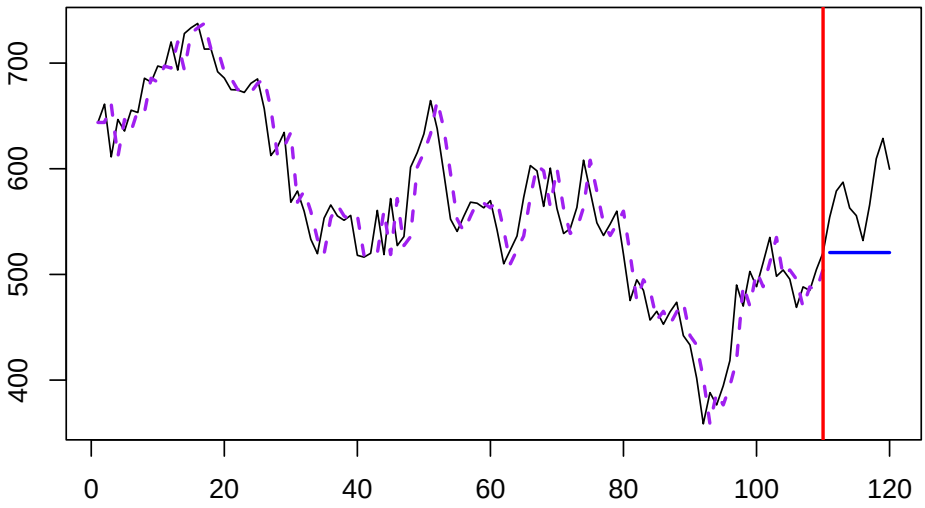


Figure 3.5: (#fig:naiveExample)A Random Walk example.

As shown from the plot in Figure @ref(fig:naiveExample), Naïve lags behind the actual time series by one observation because of how it is constructed via equation @ref(eq:Naive). The point forecast corresponds to the straight line parallel to the x-axis. Given that the data was generated from Random Walk, the point forecast shown in Figure @ref(fig:naiveExample) is the best possible forecast for the time series, even though it exhibits rapid changes in the level.

Note that if the time series exhibits level shifts or other types of unexpected changes in dynamics, Naïve will update rapidly and reach the new level instantaneously. However, because it only has a memory of one (last) observation, it will not filter out the noise in the data but rather copy it into the future. So, it has limited usefulness in practice. However, being the simplest possible forecasting method, it is considered one of the basic forecasting benchmarks. If your model cannot beat it, it is not worth using.

3.3.2 Global Mean

While Naïve considered only one observation (the most recent one), global mean (aka "global average") relies on all the observations in the data:

$$\hat{y}_t = \bar{y} = \frac{1}{T}\sum_{t=1}^{T} y_t, (\#eq : GlobalMean) \tag{3.8}$$

where T is the sample size. The model underlying this forecasting method is called "global level" and is written as:

$$y_t = \mu + \epsilon_t, (\#eq : GlobalLevel) \tag{3.9}$$

so that the $\bar{y}$ is an estimate of the fixed expectation μ. Graphically, this is represented with a straight line going through the series as shown in Figure @ref(fig:globalMeanExample).

```
y <- rnorm(120, 100, 10)
testModel <- es(y, "ANN", persistence=0,
                h=10, holdout=TRUE)
plot(testModel, 7, main="")
```

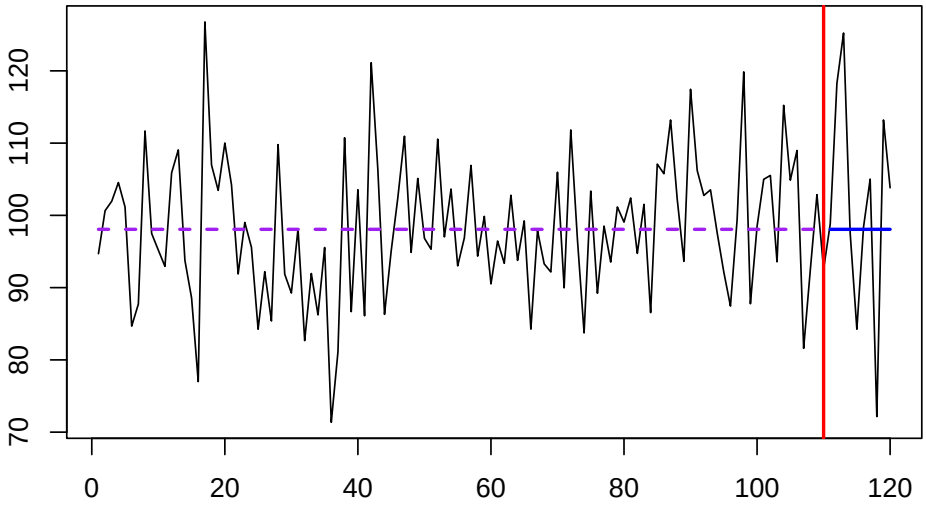


Figure 3.6: (#fig:globalMeanExample)A global level example.

The series shown in Figure @ref(fig:globalMeanExample) is generated from the global level model, and the point forecast corresponds to the forecast from the global mean method. Note that the method assumes that the weights between the in-sample observation are equal, i.e. the first observation has precisely the exact weight of $\frac{1}{T}$ as the last one. Suppose the series exhibits some changes in level over time. In that case, the global mean will not be suitable because it

will produce the averaged out forecast, considering values for parts before and after the change.

3.3.3 Simple Moving Average

Naïve and Global Mean can be considered as opposite points in the spectrum of possible level time series (although there are series beyond Naïve, see for example ARIMA(0,1,1) with $\theta_1 > 0$, discussed in Section @ref(ARIMA)). The series between them exhibit slow changes in level and can be modelled using different forecasting approaches. One of those is the Simple Moving Average (SMA), which uses the mechanism of the mean for a small part of the time series. It relies on the formula:

$$\hat{y}_t = \frac{1}{m}\sum_{j=1}^{m} y_{t-j}, (\#eq:SMA) \tag{3.10}$$

which implies going through time series with something like a "window" of m observations and using their average for forecasting. The order m determines the length of the memory of the method: if it is equal to 1, then @ref(eq:SMA) turns into Naïve, while in the case of $m = T$ it transforms into Global Mean. The order m is typically decided by a forecaster, keeping in mind that the lower m corresponds to the shorter memory method, while the higher one corresponds to the longer one.

? have shown that SMA has an underlying non-stationary AR(m) model with $\phi_j = \frac{1}{m}$ for all $j = 1, \dots, m$. While the conventional approach to forecasting from SMA is to produce the straight line, equal to the last obtained observation, **?** demonstrate that, in general, the point forecast does not correspond to the straight line.

```
y <- sim.sma(10,120)
par(mfcol=c(2,2), mar=c(2,2,2,1))
# SMA(1)
testModel <- sma(y, order=1,
                 h=10, holdout=TRUE)
plot(testModel, 7, main=testModel$model)
# SMA(10)
testModel <- sma(y, order=10,
                 h=10, holdout=TRUE)
plot(testModel, 7, main=testModel$model)
# SMA(20)
testModel <- sma(y, order=20,
                 h=10, holdout=TRUE)
plot(testModel, 7, main=testModel$model)
# SMA(110)
testModel <- sma(y, order=110,
                 h=10, holdout=TRUE)
```

```
plot(testModel, 7, main=testModel$model)
```

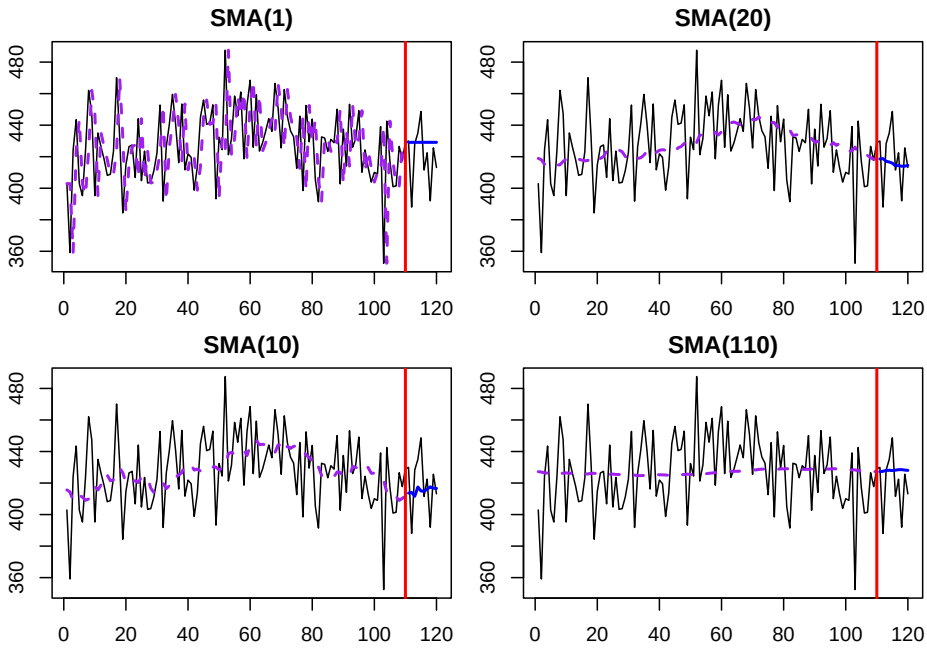


Figure 3.7: (#fig:SMAExample)Examples of SMA time series and several SMA models of different orders applied to it.

Figure @ref(fig:SMAExample) demonstrates the time series generated from SMA(10) and several SMA models applied to it. We can see that the higher orders of SMA lead to smoother fitted lines and calmer point forecasts. On the other hand, the SMA of a very high order, such as SMA(110), does not follow the changes in time series efficiently, ignoring the potential changes in level. Given the difficulty with selecting the order m, **?** proposed using information criteria for the order selection of SMA in practice.

Finally, an attentive reader has already spotted that the formula for SMA corresponds to the procedure of CMA of an odd order from equation @ref(eq:CMAOdd). They are similar, but they have a different purpose: CMA is needed to smooth out the series, the calculated values are inserted in the middle of the average, while SMA is used for forecasting, and the point forecasts are inserted at the end of the average.

3.3.4 Random Walk with drift

So far, we have discussed the methods used for level time series. But as mentioned in Section @ref(tsComponents), there are other components in the time

series. In the case of the series with a trend, Naïve, Global Mean and SMA will be inappropriate because they would be missing the essential component. The simplest model that can be used in this case is called "Random Walk with drift", which is formulated as:

$$y_t = y_{t-1} + a_0 + \epsilon_t, (\#eq : RandomWalkWithDrift) \tag{3.11}$$

where a_0 is a constant term, the introduction of which leads to increasing or decreasing trajectories, depending on the value of a_0. The point forecast from this model is calculated as:

$$\hat{y}_{t+h} = y_t + a_0 h, (\#eq : RandomWalkWithDriftForecast) \tag{3.12}$$

implying that the forecast from the model is a straight line with the slope parameter a_0. Figure @ref(fig:RWDriftExample) shows how the data generated from Random Walk with drift and $a_0 = 10$ looks like. This model is discussed in Section @ref(ARIMA).

```
y <- sim.ssarima(orders=list(i=1), lags=1, obs=120,
                 constant=10)
testModel <- msarima(y, orders=list(i=1), constant=TRUE,
                     h=10, holdout=TRUE)
plot(testModel, 7, main="")
```

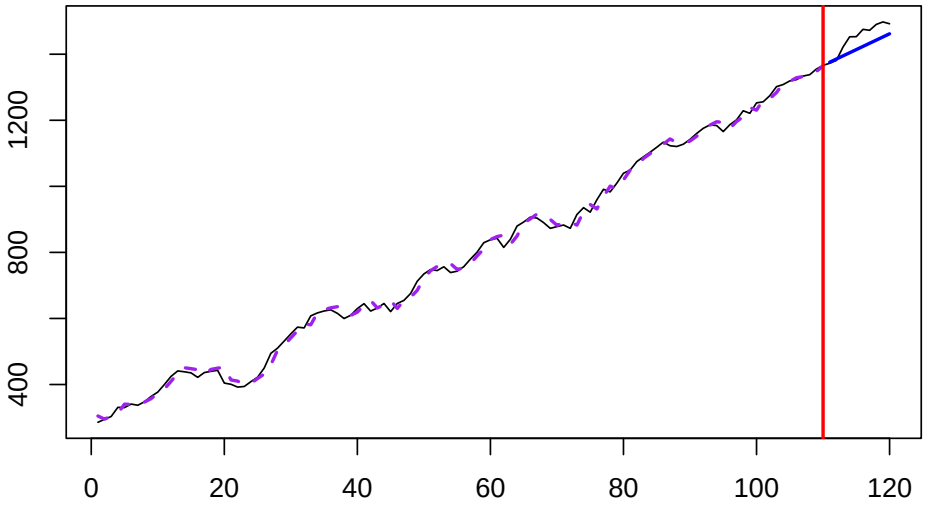


Figure 3.8: (#fig:RWDriftExample)Random Walk with drift data and the model applied to that data.

The data in Figure @ref(fig:RWDriftExample) demonstrates a positive trend (because $a_0 > 0$) and randomness from one observation to another. The model is helpful as a benchmark and a special case for several other models because it is simple and requires the estimation of only one parameter.

3.3.5 Seasonal Naïve

Finally, in the case of seasonal data, there is a simple forecasting method that can be considered as a good benchmark in many situations. Similar to Naïve, Seasonal Naïve relies only on one observation, but instead of taking the most recent value, it uses the value from the same period a season ago. For example, for producing a forecast for January 1984, we would use January 1983. Mathematically this is written as:

$$\hat{y}_t = y_{t-m}, (\#eq : NaiveSeasonal) \tag{3.13}$$

where m is the seasonal frequency. This method has an underlying model, Seasonal Random Walk:

$$\hat{y}_t = y_{t-m} + \epsilon_t . (\#eq : RWSeasonal) \tag{3.14}$$

Similar to Naïve, the higher variability of the error term ϵ_t in @ref(eq:RWSeasonal) is, the faster the data exhibit changes. Seasonal Naïve does not require estimation of any parameters and thus is considered one of the popular benchmarks in seasonal data. Figure @ref(fig:NaiveSeasonalExample) demonstrates how the data generated from seasonal Random Walk looks and how the point forecast from the seasonal Naïve applied to this data performs.

```
y <- sim.ssarima(orders=list(i=1), lags=4,
                 obs=120, sd=50)
testModel <- msarima(y, orders=list(i=1), lags=4,
                     h=10, holdout=TRUE)
plot(testModel, 7, main="")
```

Similarly to the previous methods, if other approaches cannot outperform seasonal Naïve, it is not worth spending time on those approaches.

3.4 ETS taxonomy

Building on the idea of time series components (from Section @ref(tsComponents)), we can move to the ETS taxonomy. ETS stands for "Error-Trend-Seasonality" and defines how specifically the components interact with each other. Based on the type of error, trend and seasonality, **?** proposed a taxonomy, which was then developed further by **?** and refined by **?**. According to this taxonomy, error, trend and seasonality can be:

1. Error: "Additive" (A), or "Multiplicative" (M);
2. Trend: "None" (N), or "Additive" (A), or "Additive damped" (Ad), or "Multiplicative" (M), or "Multiplicative damped" (Md);
3. Seasonality: "None" (N), or "Additive" (A), or "Multiplicative" (M).

According to this taxonomy, the model @ref(eq:PureAdditive) is denoted as ETS(A,A,A) while the model @ref(eq:PureMultiplicative) is denoted as ETS(M,M,M), and @ref(eq:MixedAdditiveTrend) is ETS(M,A,M).

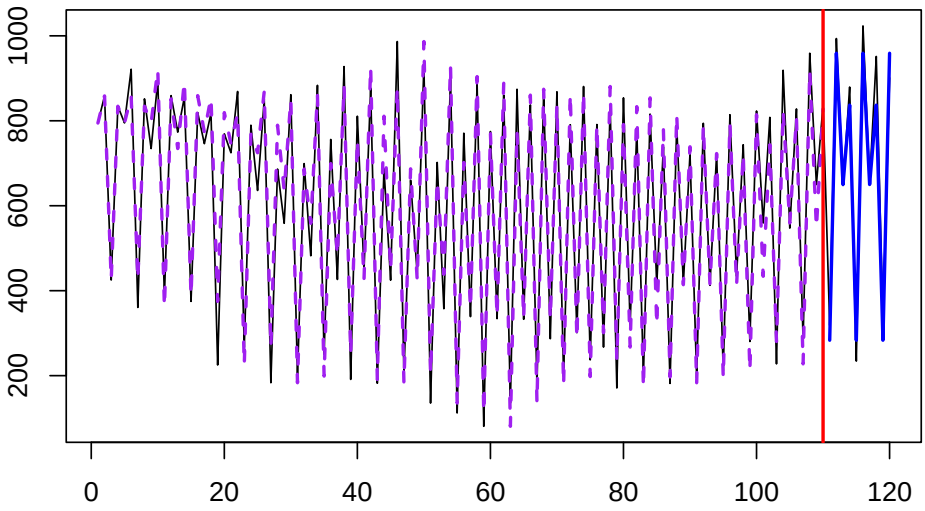


Figure 3.9: (#fig:NaiveSeasonalExample)Seasonal Random Walk and Seasonal Naïve.

The components in the ETS taxonomy have clear interpretations. Furthermore, ETS supports 30 models with different types of error, trend and seasonality. Figure @ref(fig:ETSTaxonomyAdditive) shows examples of different time series with deterministic (they do not change over time) level, trend, seasonality and with the additive error term.

Things to note from the plots in Figure @ref(fig:ETSTaxonomyAdditive):

1. When seasonality is multiplicative, its amplitude increases with the increase of the level of the data, while with additive seasonality, the amplitude is constant. Compare, for example, ETS(A,A,A) with ETS(A,A,M): for the former, the distance between the highest and the lowest points in the first year is roughly the same as in the last year. In the case of ETS(A,A,M) the distance increases with the increase in the level of series.
2. When the trend is multiplicative, data exhibits exponential growth/decay result.
3. The damped trend models slow down both additive and multiplicative trends.
4. It is practically impossible to distinguish additive and multiplicative seasonality if the level of series does not change because the amplitude of seasonality will be constant in both cases (compare ETS(A,N,A) and ETS(A,N,M)).

Figure @ref(fig:ETSTaxonomyMultiplicative) demonstrates a similar plot for the multiplicative error models.

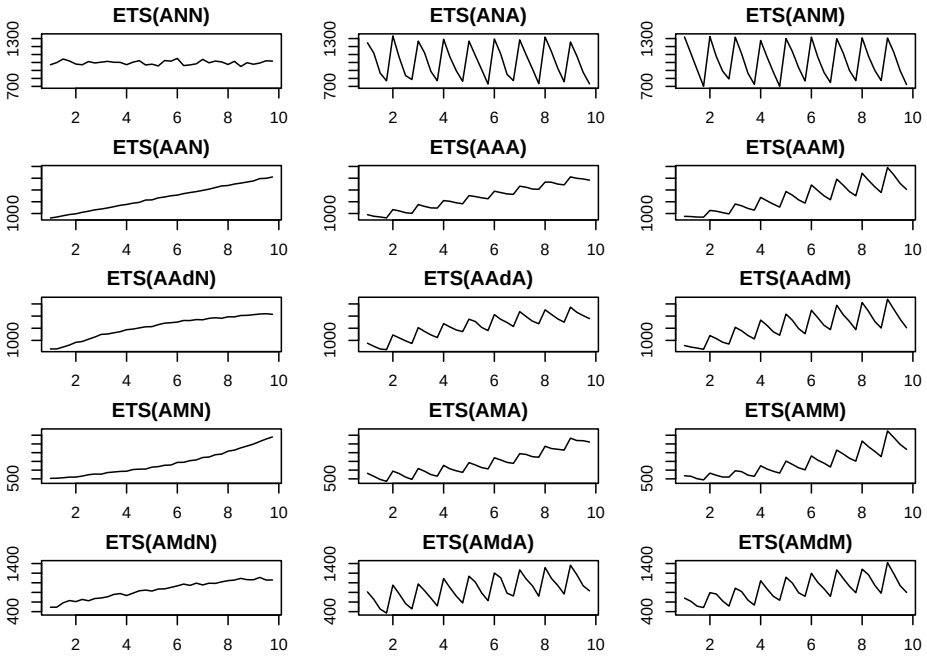


Figure 3.10: (#fig:ETSTaxonomyAdditive)Time series corresponding to the additive error ETS models

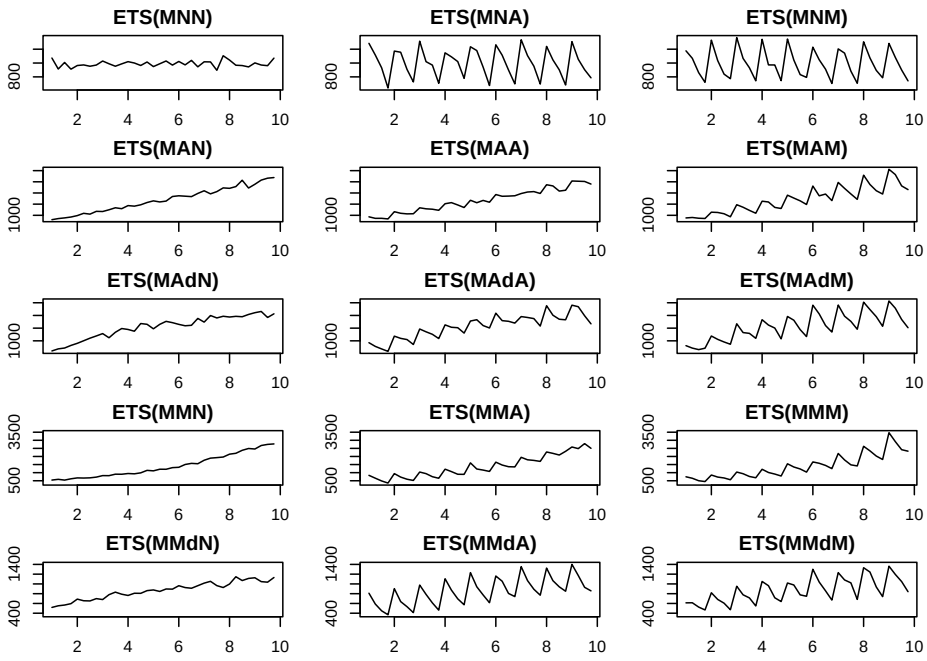


Figure 3.11: (#fig:ETSTaxonomyMultiplicative)Time series corresponding to the multiplicative error ETS models

The plots in Figure @ref(fig:ETSTaxonomyMultiplicative) show roughly the same idea as the additive case, the main difference being that the variance of the error increases with the increase of the level of the data – this becomes clearer on ETS(M,A,N) and ETS(M,M,N) data. This property is called heteroscedasticity in statistics, and **?** argue that the main benefit of the multiplicative error models is in capturing this feature.

We will discuss the most important ETS family members in the following chapters. Not all the models in this taxonomy are sensible, and some are typically ignored entirely. Although ADAM implements the entire taxonomy, we will discuss potential issues and what to expect from them.

3.5 Mathematical models in the ETS taxonomy

I hope that it becomes more apparent to the reader how the ETS framework is built upon the idea of time series decomposition (from Section @ref(tsComponents)). By introducing different components, defining their types, and adding the equations for their update, we can construct models that would work better on the time series. The equations discussed in Section @ref(tsComponents) represent so-called "measurement" or "observation" equations of the ETS models. But we should also consider the potential change in components over time. The "transition" or "state" equation is supposed to reflect this change: they explain how the level, trend or seasonal components evolve.

As discussed in Section @ref(ETSTaxonomy), given different types of components and their interactions, we end up with 30 models in the taxonomy. Tables @ref(tab:ETSAdditiveError) and @ref(tab:ETSMultiplicativeError) summarise mathematically all 30 ETS models shown graphically on Figures @ref(fig:ETSTaxonomyAdditive) and @ref(fig:ETSTaxonomyMultiplicative), presenting formulae for measurement and transition equations.

From a statistical point of view, formulae in Tables @ref(tab:ETSAdditiveError) and @ref(tab:ETSMultiplicativeError) correspond to the "true models" (see Section 1.2 of **?**), they explain the models underlying potential data, but when it comes to their construction and estimation, the ϵ_t is substituted by the estimated e_t (which is calculated differently depending on the error type), and time series components and smoothing parameters are also replaced by their estimated analogues (e.g. $\hat{\alpha}$ instead of α). However, if the values of these models' parameters were known, it would be possible to produce point forecasts and conditional h steps ahead expectations from these models.

Table @ref(tab:ETSModelsForecasts) summarises:

- Conditional one step ahead expectation $\mu_{y,t} = \mu_{y,t|t-1}$;
- Multiple steps ahead point forecast $\hat{y}_{t+h}$;
- Conditional multiple steps ahead expectation $\mu_{y,t+h|t}$;

Table 3.1: (#tab:ETSAdditiveError)Additive error ETS models

	Nonseasonal	Additive seasonality
No trend	$y_t = l_{t-1} + \epsilon_t$ $l_t = l_{t-1} + \alpha\epsilon_t$	$y_t = l_{t-1} + s_{t-m} + \epsilon_t$ $l_t = l_{t-1} + \alpha\epsilon_t$ $s_t = s_{t-m} + \gamma\epsilon_t$
Additive trend	$y_t = l_{t-1} + b_{t-1} + \epsilon_t$ $l_t = l_{t-1} + b_{t-1} + \alpha\epsilon_t$ $b_t = b_{t-1} + \beta\epsilon_t$	$y_t = l_{t-1} + b_{t-1} + s_{t-m} + \epsilon$ $l_t = l_{t-1} + b_{t-1} + \alpha\epsilon_t$ $b_t = b_{t-1} + \beta\epsilon_t$ $s_t = s_{t-m} + \gamma\epsilon_t$
Additive damped trend	$y_t = l_{t-1} + \phi b_{t-1} + \epsilon_t$ $l_t = l_{t-1} + \phi b_{t-1} + \alpha\epsilon_t$ $b_t = \phi b_{t-1} + \beta\epsilon_t$	$y_t = l_{t-1} + \phi b_{t-1} + s_{t-m} +$ $l_t = l_{t-1} + \phi b_{t-1} + \alpha\epsilon_t$ $b_t = \phi b_{t-1} + \beta\epsilon_t$ $s_t = s_{t-m} + \gamma\epsilon_t$
Multiplicative trend	$y_t = l_{t-1}b_{t-1} + \epsilon_t$ $l_t = l_{t-1}b_{t-1} + \alpha\epsilon_t$ $b_t = b_{t-1} + \beta\frac{\epsilon_t}{l_{t-1}}$	$y_t = l_{t-1}b_{t-1} + s_{t-m} + \epsilon_t$ $l_t = l_{t-1}b_{t-1} + \alpha\epsilon_t$ $b_t = b_{t-1} + \beta\frac{\epsilon_t}{l_{t-1}}$ $s_t = s_{t-m} + \gamma\epsilon_t$
Multiplicative damped trend	$y_t = l_{t-1}b_{t-1}^{\phi} + \epsilon_t$ $l_t = l_{t-1}b_{t-1}^{\phi} + \alpha\epsilon_t$ $b_t = b_{t-1}^{\phi} + \beta\frac{\epsilon_t}{l_{t-1}}$	$y_t = l_{t-1}b_{t-1}^{\phi} + s_{t-m} + \epsilon_t$ $l_t = l_{t-1}b_{t-1}^{\phi} + \alpha\epsilon_t$ $b_t = b_{t-1}^{\phi} + \beta\frac{\epsilon_t}{l_{t-1}}$ $s_t = s_{t-m} + \gamma\epsilon_t$

Table 3.2: (#tab:ETSMultiplicativeError)Multiplicative error ETS models

	Nonseasonal	Additive seasonality
No trend	$y_t = l_{t-1}(1+\epsilon_t)$ $l_t = l_{t-1}(1+\alpha\epsilon_t)$	$y_t = (l_{t-1}+s_{t-m})(1+\epsilon_t)$ $l_t = l_{t-1}+\alpha\mu_{y,t}\epsilon_t$ $s_t = s_{t-m}+\gamma\mu_{y,t}\epsilon_t$
Additive trend	$y_t = (l_{t-1}+b_{t-1})(1+\epsilon_t)$ $l_t = (l_{t-1}+b_{t-1})(1+\alpha\epsilon_t)$ $b_t = b_{t-1}+\beta\mu_{y,t}\epsilon_t$	$y_t = (l_{t-1}+b_{t-1}+s_{t-m})(1+\epsilon_t)$ $l_t = l_{t-1}+b_{t-1}+\alpha\mu_{y,t}\epsilon_t$ $b_t = b_{t-1}+\beta\mu_{y,t}\epsilon_t$ $s_t = s_{t-m}+\gamma\mu_{y,t}\epsilon_t$
Additive damped trend	$y_t = (l_{t-1}+\phi b_{t-1})(1+\epsilon_t)$ $l_t = (l_{t-1}+\phi b_{t-1})(1+\alpha\epsilon_t)$ $b_t = \phi b_{t-1}+\beta\mu_{y,t}\epsilon_t$	$y_t = (l_{t-1}+\phi b_{t-1}+s_{t-m})(1+\epsilon_t)$ $l_t = l_{t-1}+\phi b_{t-1}+\alpha\mu_{y,t}\epsilon_t$ $b_t = \phi b_{t-1}+\beta\mu_{y,t}\epsilon_t$ $s_t = s_{t-m}+\gamma\mu_{y,t}\epsilon_t$
Multiplicative trend	$y_t = l_{t-1}b_{t-1}(1+\epsilon_t)$ $l_t = l_{t-1}b_{t-1}(1+\alpha\epsilon_t)$ $b_t = b_{t-1}(1+\beta\epsilon_t)$	$y_t = (l_{t-1}b_{t-1}+s_{t-m})(1+\epsilon_t)$ $l_t = l_{t-1}b_{t-1}+\alpha\mu_{y,t}\epsilon_t$ $b_t = b_{t-1}+\beta\frac{\mu_{y,t}}{l_{t-1}}\epsilon_t$ $s_t = s_{t-m}+\gamma\mu_{y,t}\epsilon_t$
Multiplicative damped trend	$y_t = l_{t-1}b_{t-1}^{\phi}(1+\epsilon_t)$ $l_t = l_{t-1}b_{t-1}^{\phi}(1+\alpha\epsilon_t)$ $b_t = b_{t-1}^{\phi}(1+\beta\epsilon_t)$	$y_t = (l_{t-1}b_{t-1}^{\phi}+s_{t-m})(1+\epsilon_t)$ $l_t = l_{t-1}b_{t-1}^{\phi}+\alpha\mu_{y,t}\epsilon_t$ $b_t = b_{t-1}^{\phi}+\beta\frac{\mu_{y,t}}{l_{t-1}}\epsilon_t$ $s_t = s_{t-m}+\gamma\mu_{y,t}\epsilon_t$

In the case of the additive error models, the point forecasts correspond to the expectations only when the expectation of the error term is zero, i.e. $\mathrm{E}(\epsilon_t) = 0$. In contrast, in the case of the multiplicative one, the condition is changed to $\mathrm{E}(1 + \epsilon_t) = 1$.

Remark. ***Not all the point forecasts of ETS models correspond to conditional expectations***. *This issue applies to the models with multiplicative trend and/or multiplicative seasonality. This is because the ETS model assumes that different states are correlated (they have the same source of error), and as a result, multiple steps ahead values (when h>1) of states introduce products of error terms. So, the conditional expectations in these cases might not have analytical forms ("n.c.f." in Table @ref(tab:ETSModelsForecasts) stands for "no closed-form"), and when working with these models, simulations might be required. This does not apply to the one-step-ahead forecasts, for which all the classical formulae work.*

The multiplicative error models have the same one step ahead expectations and point forecasts as the additive error ones. However, due to the multiplication by the error term, the multiple steps ahead conditional expectations between the two types of models might differ, specifically for the multiplicative trend and multiplicative seasonal models. These values do not have closed forms and can only be obtained via simulations.

Although there are 30 potential ETS models, not all of them are stable. So, Rob Hyndman has reduced the pool of models under consideration in the `ets()` function of `forecast` package to the following 19: ANN, AAN, AAdN, ANA, AAA, AAdA, MNN, MAN, MAdN, MNA, MAA, MAdA, MNM, MAM, MAdM, MMN, MMdN, MMM, MMdM. In addition, the multiplicative trend models are unstable in data with outliers, so they are switched off in the `ets()` function by default, which reduces the pool of models further to the first 15.

Table 3.3: (#tab:ETSModelsForecasts)Point forecasts and expectations of ETS models. n.c.f. stands for "No Closed Form".

	Nonseasonal	Additive seasonality
No trend	$\mu_{y,t} = l_{t-1}$ $\hat{y}_{t+h} = l_t$ $\mu_{y,t+h\|t} = \hat{y}_{t+h}$	$\mu_{y,t} = l_{t-1} + s_{t-m}$ $\hat{y}_{t+h} = l_t + s_{t+h-m\lceil \frac{h}{m} \rceil}$ $\mu_{y,t+h\|t} = \hat{y}_{t+h}$
Additive trend	$\mu_{y,t} = l_{t-1} + b_{t-1}$ $\hat{y}_{t+h} = l_t + hb_t$ $\mu_{y,t+h\|t} = \hat{y}_{t+h}$	$\mu_{y,t} = l_{t-1} + b_{t-1} + s_{t-m}$ $\hat{y}_{t+h} = l_t + hb_{t-1} + s_{t+h-m\lceil \frac{h}{m} \rceil}$ $\mu_{y,t+h\|t} = \hat{y}_{t+h}$
Additive damped trend	$\mu_{y,t} = l_{t-1} + \phi b_{t-1}$ $\hat{y}_{t+h} = l_t + \sum_{j=1}^{h} \phi^j b_t$ $\mu_{y,t+h\|t} = \hat{y}_{t+h}$	$\mu_{y,t} = l_{t-1} + \phi b_{t-1} + s_{t-m}$ $\hat{y}_{t+h} = l_t + \sum_{j=1}^{h} \phi^j b_{t-1} + s_{t+h-m\lceil \frac{h}{m} \rceil}$ $\mu_{y,t+h\|t} = \hat{y}_{t+h}$
Multiplicative trend	$\mu_{y,t} = l_{t-1} b_{t-1}$ $\hat{y}_{t+h} = l_t b_t^h$ $\mu_{y,t+h\|t}$ – n.c.f. for $h > 1$	$\mu_{y,t} = l_{t-1} b_{t-1} + s_{t-m}$ $\hat{y}_{t+h} = l_t b_{t-1}^h + s_{t+h-m\lceil \frac{h}{m} \rceil}$ $\mu_{y,t+h\|t}$ – n.c.f. for $h > 1$
Multiplicative damped trend	$\mu_{y,t} = l_{t-1} b_{t-1}^{\phi}$ $\hat{y}_{t+h} = l_t b_t^{\sum_{j=1}^{h} \phi^j}$ $\mu_{y,t+h\|t}$ – n.c.f. for $h > 1$	$\mu_{y,t} = l_{t-1} b_{t-1}^{\phi} + s_{t-m}$ $\hat{y}_{t+h} = l_t b_{t-1}^{\sum_{j=1}^{h} \phi^j} + s_{t+h-m\lceil \frac{h}{m} \rceil}$ $\mu_{y,t+h\|t}$ – n.c.f. for $h > 1$

Chapter 4

Conventional Exponential Smoothing

Now that we know how time series can be decomposed into components, we can discuss the exponential smoothing methods, focusing on the most popular ones. We do not detail how the methods were originally derived and how to work with them. Instead, we focus on their connection with ETS and the main ideas behind the conventional ETS.

The reader interested in the history of exponential smoothing, how it was developed, and what papers contributed to the field can refer to the reviews of **?** and **?**. They summarise all the progress in exponential smoothing up until 1985 and until 2006 respectively.

4.1 Simple Exponential Smoothing

We start our discussion of exponential smoothing with the original Simple Exponential Smoothing (SES) forecasting method, which was formulated by (**?**):

$$\hat{y}_{t+1} = \hat{\alpha} y_t + (1 - \hat{\alpha})\hat{y}_t, (\#eq : BrownMethod) \tag{4.1}$$

where $\hat{\alpha}$ is the smoothing parameter defined by analyst and which is typically restricted with (0, 1) region (this region is arbitrary, and we will see in Section @ref(ETSParametersBounds) what is the correct one). This is one of the simplest forecasting methods. The smoothing parameter is typically interpreted as a weight between the actual value and the predicted one-step-ahead. If the smoothing parameter is close to zero, then more weight is given to the previous fitted value $\hat{y}_t$ and the new information is neglected. If $\hat{\alpha} = 0$, then the method becomes equivalent to the Global Mean method, discussed in Section @ref(GlobalMean). When it is close to one, then mainly the actual value y_t

is taken into account. If $\hat{\alpha} = 1$, then the method transforms into Naïve, discussed in Section @ref(Naive). By changing the smoothing parameter value, the forecaster can decide how to approximate the data and filter out the noise.

Also, notice that this is a recursive method, meaning that there needs to be some starting point $\hat{y}_1$ to apply @ref(eq:BrownMethod) to the existing data. Different initialisation and estimation methods for SES have been discussed in the literature. Still, the state of the art one is to estimate $\hat{\alpha}$ and $\hat{y}_1$ together by minimising some loss function (**?**). Typically MSE (see Section @ref(errorMeasures)) is used as one, minimising the squares of one step ahead forecast errors.

4.1.1 Examples of application

Here is an example of how this method works on different time series. We start with generating a stationary series and using `es()` function from `smooth` package. Although it implements the ETS model, we will see later the connection between SES and ETS(A,N,N). We start with the stationary time series and $\hat{\alpha} = 0$:

```
y <- rnorm(100,100,10)
ourModel <- es(y, model="ANN", h=10, persistence=0)
plot(ourModel, 7, main="")
```

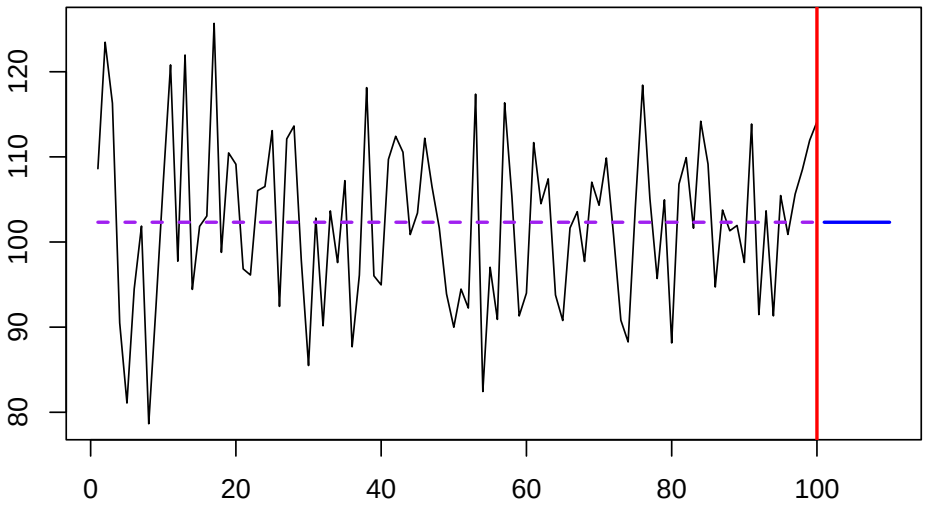


Figure 4.1: (#fig:SESExample1)An example with a time series and SES forecast. $\hat{\alpha} = 0$

As we see from Figure @ref(fig:SESExample1), the SES works well in this case, capturing the deterministic level of the series and filtering out the noise. In this case, it works like a global average applied to the data. As mentioned before, the method is flexible, so if we have a level shift in the data and in-

crease the smoothing parameter, it will adapt and get to the new level. Figure @ref(fig:SESExample2) shows an example with a level shift in the data.

```
y <- c(rnorm(50,100,10),rnorm(50,130,10))
ourModel <- es(y, model="ANN", h=10, persistence=0.1)
plot(ourModel, 7, main="")
```

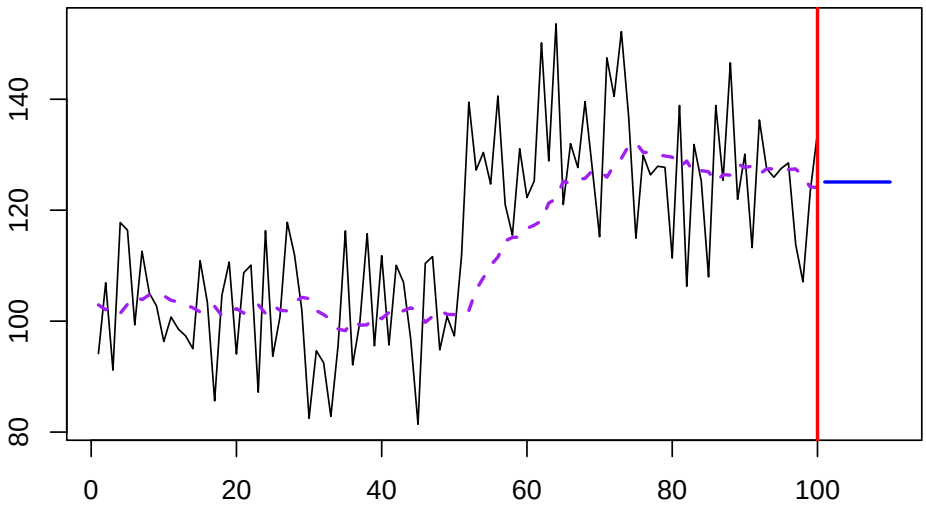


Figure 4.2: (#fig:SESExample2)An example with a time series and SES forecast. $\hat{\alpha} = 0.1$

With $\hat{\alpha} = 0.1$, SES manages to get to the new level, but now the method starts adapting to noise a little bit – it follows the peaks and troughs and repeats them with a lag, but with a much smaller magnitude (see Figure @ref(fig:SESExample2)). Increasing the smoothing parameter, the model will react to the changes much faster, at the cost of responding more to noise. This is shown in Figure @ref(fig:SESExamples) with different smoothing parameters values.

If we set $\hat{\alpha} = 1$, we will end up with Naive forecasting method (see Section @ref(Naive)), which is not appropriate for our example (see Figure @ref(fig:SESExampleNaive)).

So, when working with SES, we need to make sure that the reasonable smoothing parameter is selected. This can be done automatically via minimising the MSE (see Figure @ref(fig:SESExample3)):

```
ourModel <- es(y, model="ANN", h=10, loss="MSE")
plot(ourModel, 7, main=paste0("SES with alpha=",
                              round(ourModel$persistence,3)))
```

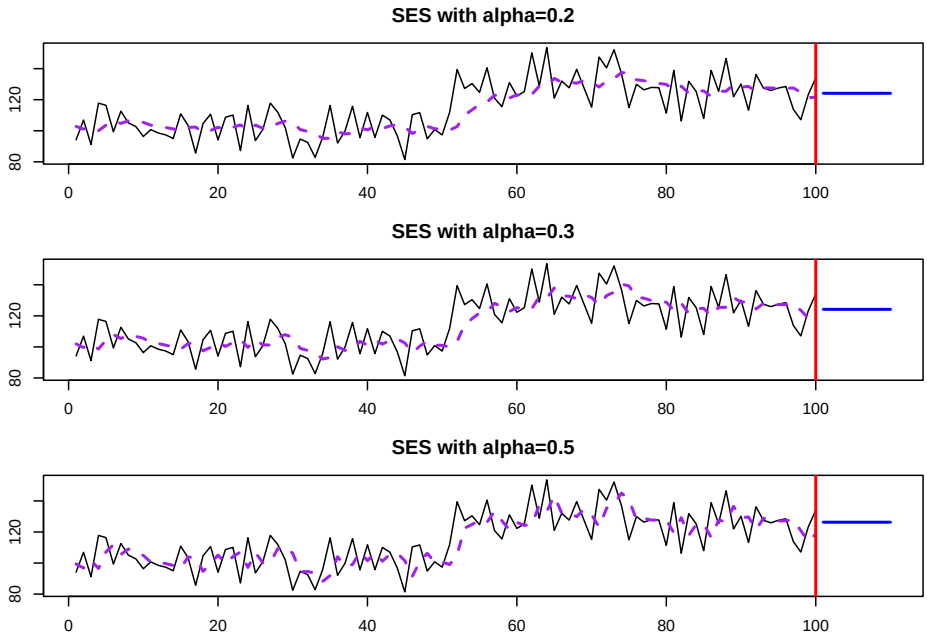


Figure 4.3: (#fig:SESExamples)SES with different smoothing parameters applied to the same data.

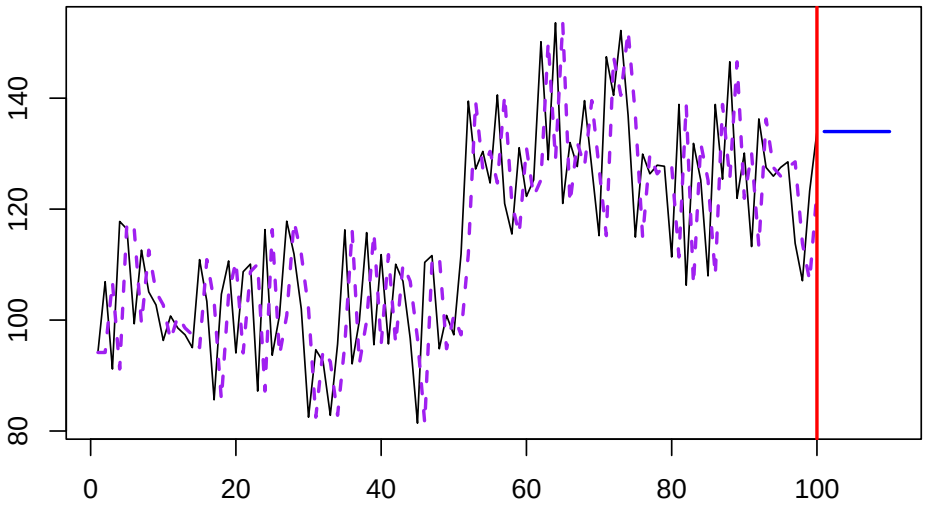


Figure 4.4: (#fig:SESExampleNaive)SES with $\hat{\alpha} = 1$.

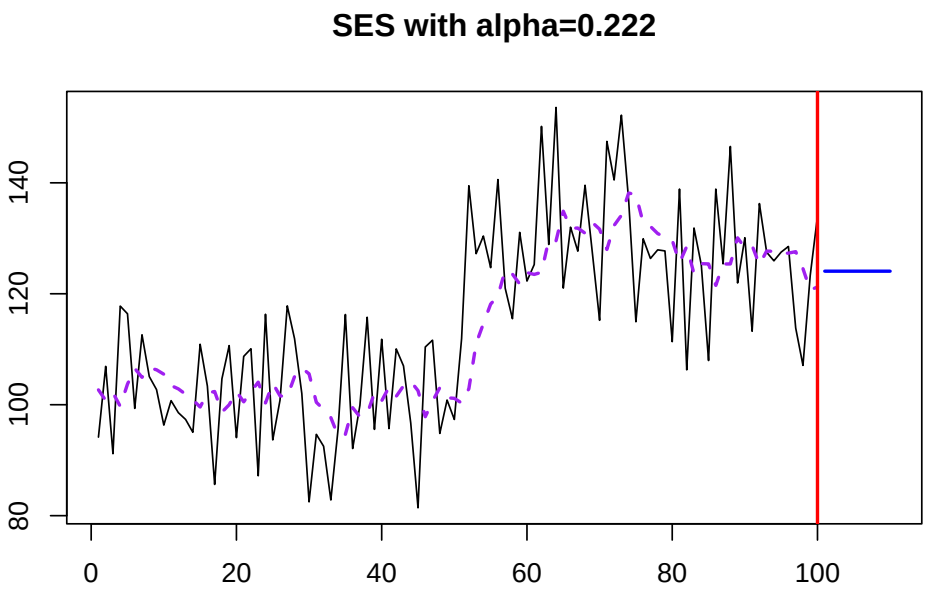


Figure 4.5: (#fig:SESExample3)SES with optimal smoothing parameter.

This approach won't guarantee that we will get the most appropriate $\hat{\alpha}$. Still, it has been shown in the literature that the optimisation of smoothing parameters on average leads to improvements in terms of forecasting accuracy (see, for example, **?**).

4.1.2 Why "exponential"?

Now, **why is it called "exponential"**? Because the same method can be represented in a different form, if we substitute $\hat{y}_t$ in right hand side of @ref(eq:BrownMethod) by the formula for the previous step:

$$\begin{aligned} \hat{y}_t =& \hat{\alpha} y_{t-1} + (1-\hat{\alpha})\hat{y}_{t-1}, \\ \hat{y}_{t+1} =& \hat{\alpha} y_t + (1-\hat{\alpha})\hat{y}_t = \\ & \hat{\alpha} y_t + (1-\hat{\alpha})\left(\hat{\alpha} y_{t-1} + (1-\hat{\alpha})\hat{y}_{t-1}\right). \end{aligned} \quad (\#eq:BrownMethodExponential1) \quad (4.2)$$

By repeating this procedure for each $\hat{y}_{t-1}$, $\hat{y}_{t-2}$ etc, we will obtain a different form of the method:

$$\hat{y}_{t+1} = \hat{\alpha} y_t + \hat{\alpha}(1-\hat{\alpha}) y_{t-1} + \hat{\alpha}(1-\hat{\alpha})^2 y_{t-2} + \dots + \hat{\alpha}(1-\hat{\alpha})^{t-1} y_1 + (1-\hat{\alpha})^t \hat{y}_1 \quad (\#eq:BrownMethodExponential2) \quad (4.3)$$

or equivalently:

$$\hat{y}_{t+1} = \hat{\alpha} \sum_{j=0}^{t-1} (1-\hat{\alpha})^j y_{t-j} + (1-\hat{\alpha})^t \hat{y}_1. \quad (\#eq:BrownMethodExponential3) \quad (4.4)$$

In the form @ref(eq:BrownMethodExponential3), each actual observation has a weight infront of it. For the most recent observation it is equal to $\hat{\alpha}$, for the previous one it is $\hat{\alpha}(1-\hat{\alpha})$, then $\hat{\alpha}(1-\hat{\alpha})^2$ etc. These form the geometric series or an exponential curve. Figure @ref(fig:BrownExponentialExample) shows an example with $\hat{\alpha} = 0.25$ for a sample of 30 observations:

```
plot(0.25*(1-0.25)^c(0:30), type="b",
     xlab="Time lags", ylab="Weights")
```

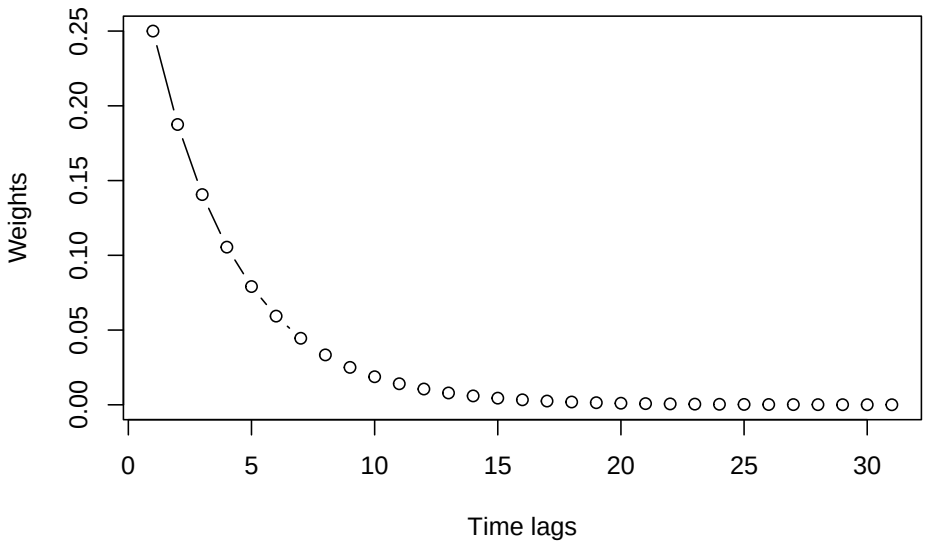


Figure 4.6: (#fig:BrownExponentialExample)Example of weights distribution for $\hat{\alpha} = 0.25$

This explains the name "exponential". The term "smoothing" comes from the idea that the parameter $\hat{\alpha}$ should be selected so that the method smooths the original time series and does not react to noise.

4.1.3 Error correction form of SES

Finally, an alternative form of SES is known as error correction form and involves some simple permutations, taking that $e_t = y_t - \hat{y}_t$ is the one step ahead forecast error, formula @ref(eq:BrownMethod) can be written as:

$$\hat{y}_{t+1} = \hat{y}_t + \hat{\alpha} e_t . (\#eq : SESErrorCorrection) \tag{4.5}$$

In this form, the smoothing parameter $\hat{\alpha}$ has a different meaning: it regulates how much the model reacts to the forecast error. In this interpretation, it no longer needs to be restricted with (0, 1) region, but we would still typically want it to be closer to zero to filter out the noise, not to adapt to it.

As you see, SES is a straightforward method. It is easy to explain to practitioners, and it is very easy to implement in practice. However, this is just a forecasting method (see Section @ref(modelsMethods)), so it provides a way of generating point forecasts but does not explain where the error comes from and how to create prediction intervals.

4.2 SES and ETS

4.2.1 ETS(A,N,N)

There have been several tries to develop statistical models underlying SES, and we know now that it has underlying ARIMA(0,1,1), local level MSOE (Multiple Source of Error, **?**) and SSOE (Single Source of Error, **?**) models. According to **?**, the ETS(A,N,N) model also underlies the SES method. It can be formulated in the following way, as discussed in Section @ref(ETSTaxonomyMaths):

$$\begin{matrix} y_t = l_{t-1} + \epsilon_t \\ l_t = l_{t-1} + \alpha \epsilon_t \end{matrix}, (\#eq:ETSANN) \tag{4.6}$$

where, as we know from Section @ref(tsComponents), l_t is the level of the data, ϵ_t is the error term and α is the smoothing parameter. Note that we use α without the "hat" symbol, which implies that there is a "true" value of the parameter (which could be obtained if we had all the data in the world or just knew it for some reason). It is easy to show that ETS(A,N,N) underlies SES. In order to see this, we need to move towards estimation phase and use $\hat{l}_{t-1} = l_{t-1}$ and $\hat{\alpha}$ instead of α and e_t as estimate of ϵ_t:

$$\begin{matrix} y_t = \hat{l}_{t-1} + e_t \\ \hat{l}_t = \hat{l}_{t-1} + \hat{\alpha} e_t \end{matrix}, (\#eq:ETSANNEstimation) \tag{4.7}$$

Inserting the second equation in the first one, we get:

$$y_t = \hat{l}_{t-2} + \hat{\alpha} e_{t-1} + e_t. (\#eq:ETSANNEstimation01) \tag{4.8}$$

The one step ahead forecast from ETS(A,N,N) is $\hat{y}_t = \hat{l}_{t-1}$ (see Section @ref(ETSTaxonomyMaths)), while the actual value can be represented as $y_t = \hat{y}_t + e_t$, leading to:

$$\hat{y}_t + e_t = \hat{y}_{t-1} + \hat{\alpha} e_{t-1} + e_t. (\#eq:ETSANNEstimation2) \tag{4.9}$$

Cancelling out e_t and shifting everything by one step ahead, we obtain the error correction form @ref(eq:SESErrorCorrection) of SES. But now, the main benefit of having the model @ref(eq:ETSANN) instead of just the method @ref(eq:SESErrorCorrection) is in having a flexible framework, which allows adding other components, selecting the most appropriate ones, consistently estimating parameters (see Section 4.3 of **?**), producing prediction intervals etc.

In order to see the data that corresponds to the ETS(A,N,N) we can use `sim.es()` function from `smooth` package. Here are several examples with different smoothing parameters values:

```
# list with generated data
y <- vector("list",6)
# Parameters for DGP
initial <- 1000
meanValue <- 0
sdValue <- 20
alphas <- c(0.1,0.3,0.5,0.75,1,1.5)
# Go through all alphas and generate respective data
for(i in 1:length(alphas)){
  y[[i]] <- sim.es("ANN", 120, 1, 12, persistence=alphas[i],
                   initial=initial, mean=meanValue, sd=sdValue)
}
```

The generated data can be plotted the following way:

```
par(mfrow=c(3,2), mar=c(2,2,2,1))
for(i in 1:6){
  plot(y[[i]], main=paste0("alpha=",y[[i]]$persistence),
       ylim=initial+c(-500,500))
}
```

This simple simulation shows that the smoothing parameter in ETS(A,N,N) controls the variability in the data (Figure @ref(fig:DGPetsANNExample)): the higher α is, the higher variability is and less predictable the data becomes. With the higher values of α, the level changes faster, leading to increased uncertainty about the future values of the level in the data.

When it comes to the application of this model to the data, the conditional h steps ahead mean corresponds to the point forecast and is equal to the last observed level:

$$\mu_{y,t+h|t} = \hat{y}_{t+h} = l_t, (\#eq : ETSANNForecast) \tag{4.10}$$

this holds because it is assumed (see Section @ref(assumptions)) that $\mathrm{E}(\epsilon_t) = 0$, which implies that the conditional h steps ahead expectation of the level in the model is (from the second equation in @ref(eq:ETSANN)):

$$\mathrm{E}(l_{t+h}|t) = l_t + \mathrm{E}(\alpha \sum_{j=1}^{h-1} \epsilon_{t+j}|t) = l_t. (\#eq : ETSANNForecasthStepsAhead) \tag{4.11}$$

Here is an example of a forecast from ETS(A,N,N) with automatic parameter estimation using `es()` function from `smooth` package:

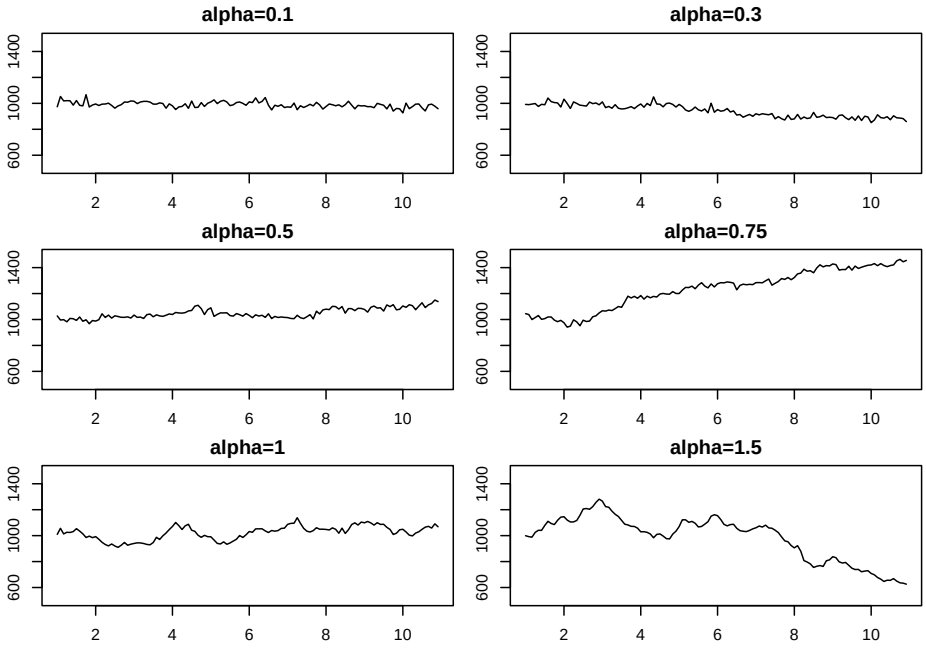


Figure 4.7: (#fig:DGPetsANNExample)Local level data corresponding to ETS(A,N,N) model with different smoothing parameters.

```
# Generate the data
y <- sim.es("ANN", 120, 1, 12, persistence=0.3, initial=1000)
# Apply ETS(A,N,N) model
es(y$data, "ANN", h=12, interval=TRUE, holdout=TRUE, silent=FALSE)
```

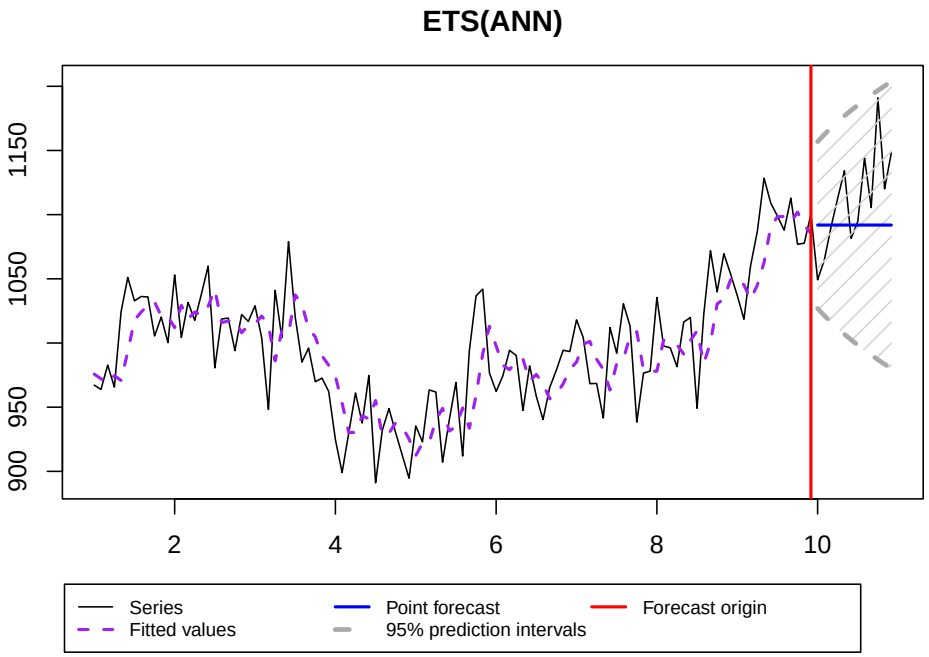


Figure 4.8: (#fig:ETSANNExample)An example of ETS(A,N,N) applied to the data generated from the same model.

```
## Time elapsed: 0.03 seconds
## Model estimated: ETS(ANN)
## Persistence vector g:
##  alpha
## 0.4224
## Initial values were optimised.
##
## Loss function type: likelihood; Loss function value: 528.5871
## Error standard deviation: 32.7686
## Sample size: 108
## Number of estimated parameters: 3
## Number of degrees of freedom: 105
## Information criteria:
##      AIC     AICc      BIC     BICc
## 1063.174 1063.405 1071.220 1071.761
```

```
##
## 95% parametric prediction interval was constructed
## 100% of values are in the prediction interval
## Forecast errors:
## MPE: 1.6%; sCE: 23.4%; Asymmetry: 57.3%; MAPE: 2.9%
## MASE: 1.128; sMAE: 3.3%; sMSE: 0.2%; rMAE: 1.061; rRMSE: 1.081
```

As we see from Figure @ref(fig:ETSANNExample), the true smoothing parameter is 0.3, but the estimated one is not exactly 0.3, which is expected because we deal with an in-sample estimation. Also, notice that with such a smoothing parameter, the prediction interval widens with the increase of the forecast horizon. If the smoothing parameter were lower, the bounds would not increase, but this might not reflect the uncertainty about the level correctly. Here is an example with $\alpha = 0.01$ on the same data (Figure @ref(fig:ETSANNExamplealpha0.1))

```
ourModel <- es(y$data, "ANN", h=12, interval=TRUE,
               holdout=TRUE, silent=FALSE, persistence=0.01)
```

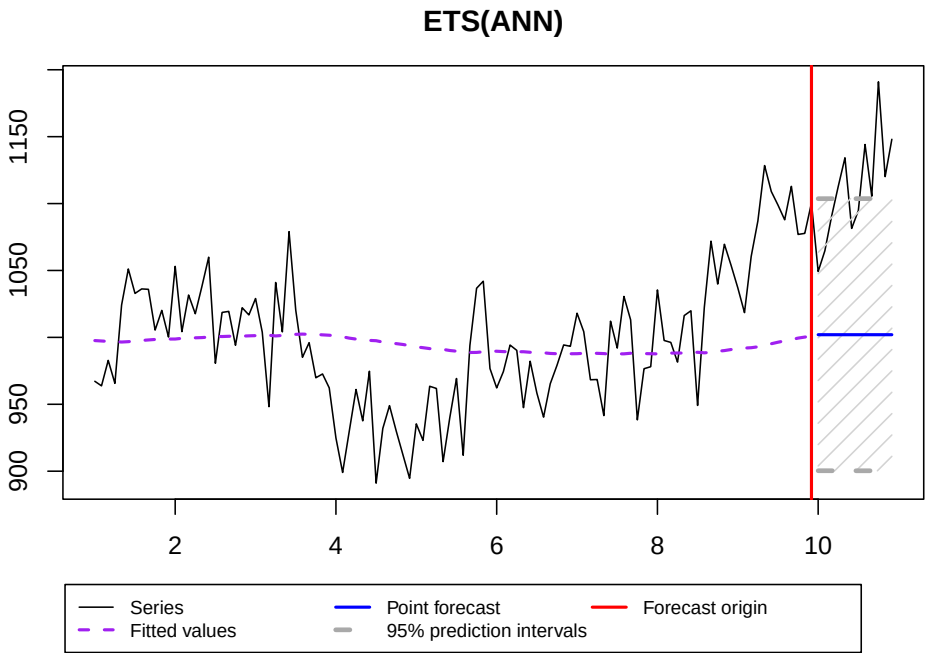


Figure 4.9: (#fig:ETSANNExamplealpha0.1)ETS(A,N,N) with $\hat{\alpha} = 0.01$ applied to the data generated from the same model with $\alpha = 0.3$.

Figure @ref(fig:ETSANNExamplealpha0.1) shows that the prediction interval does not expand, but at the same time is wider than needed, and the forecast is biased – the model does not keep up to the fast-changing time series. So,

it is essential to correctly estimate the smoothing parameters not only to approximate the data but also to produce a less biased point forecast and a more appropriate prediction interval.

4.2.2 ETS(M,N,N)

? demonstrate that there is another ETS model, underlying SES. It is the model with multiplicative error, which is formulated in the following way, as mentioned in Chapter @ref(ETSTaxonomyMaths):

$$\begin{matrix} y_t = l_{t-1}(1+\epsilon_t) \\ l_t = l_{t-1}(1+\alpha\epsilon_t) \end{matrix}, (\#eq:ETSMNN) \tag{4.12}$$

where $(1+\epsilon_t)$ corresponds to the ε_t discussed in Section @ref(tsComponents). In order to see the connection of this model with SES, we need to revert to the estimation of the model on the data again:

$$\begin{matrix} y_t = \hat{l}_{t-1}(1+e_t) \\ \hat{l}_t = \hat{l}_{t-1}(1+\hat{\alpha}e_t) \end{matrix}.(\#eq:ETSMNNEstimation) \tag{4.13}$$

where one step ahead forecast is (Section @ref(ETSTaxonomyMaths)) $\hat{y}_t = \hat{l}_{t-1}$ and $e_t = \frac{y_t - \hat{y}_t}{\hat{y}_t}$. Substituting these values in second equation of @ref(eq:ETSMNNEstimation) we obtain:

$$\hat{y}_{t+1} = \hat{y}_t \left(1 + \hat{\alpha}\frac{y_t - \hat{y}_t}{\hat{y}_t}\right) (\#eq:ETSMNNEstimation2) \tag{4.14}$$

Finally, opening the brackets, we get the SES in the form similar to @ref(eq:SESErrorCorrection):

$$\hat{y}_{t+1} = \hat{y}_t + \hat{\alpha}(y_t - \hat{y}_t).(\#eq:ETSMNNEstimation4) \tag{4.15}$$

This example again demonstrates the difference between a forecasting method and a model. When we use SES, we ignore the distributional assumptions, which restricts the usage of the method. When we use a model, we assume a specific structure, which on the one hand, makes it more restrictive, but on the other hand, gives it additional features. The main ones in the case of ETS(M,N,N) in comparison with ETS(A,N,N) are:

1. The variance of the actual values in ETS(M,N,N) increases with the increase of the level l_t. This allows modelling heteroscedasticity situation in the data;
2. If $(1+\epsilon_t)$ is always positive, then the ETS(M,N,N) model will always produce only positive forecasts (both point and interval). This makes this model applicable in principle to the data with low levels.

An alternative to @ref(eq:ETSMNN) would be the model @ref(eq:ETSANN) applied to the data in logarithms (assuming that the data we work with is always positive), implying that:

$$\begin{matrix} \log y_t = l_{t-1} + \epsilon_t \\ l_t = l_{t-1} + \alpha \epsilon_t \end{matrix} .(\#eq:ETSANNLogs) \tag{4.16}$$

However, to produce forecasts from @ref(eq:ETSANNLogs), exponentiation is needed, making the application of the model more difficult than needed. The ETS(M,N,N), on the other hand, does not rely on exponentiation, making it safe in cases when the model produces very high values (e.g. `exp(1000)` returns infinity in R).

Finally, the conditional h steps ahead mean of ETS(M,N,N) corresponds to the point forecast and is equal to the last observed level, but only if $E(1 + \epsilon_t) = 1$:

$$\mu_{y,t+h|t} = \hat{y}_{t+h} = l_t .(\#eq:ETSMNNForecast) \tag{4.17}$$

And here is an example with the ETS(M,N,N) data (Figure @ref(fig:ETSMNNExample)):

```
y <- sim.es("MNN", 120, 1, 12, persistence=0.3, initial=1000)
ourModel <- es(y$data, "MNN", h=12, holdout=TRUE,
               interval=TRUE, silent=FALSE)
```

```
ourModel
```

```
## Time elapsed: 0.02 seconds
## Model estimated: ETS(MNN)
## Persistence vector g:
##  alpha
## 0.2871
## Initial values were optimised.
##
## Loss function type: likelihood; Loss function value: 632.4832
## Error standard deviation: 0.095
## Sample size: 108
## Number of estimated parameters: 3
## Number of degrees of freedom: 105
## Information criteria:
##      AIC     AICc      BIC     BICc
## 1270.966 1271.197 1279.013 1279.553
##
## 95% parametric prediction interval was constructed
## 100% of values are in the prediction interval
## Forecast errors:
## MPE: -3.8%; sCE: -44.3%; Asymmetry: -34.3%; MAPE: 7.6%
## MASE: 0.927; sMAE: 8.7%; sMSE: 1.1%; rMAE: 0.942; rRMSE: 1.043
```

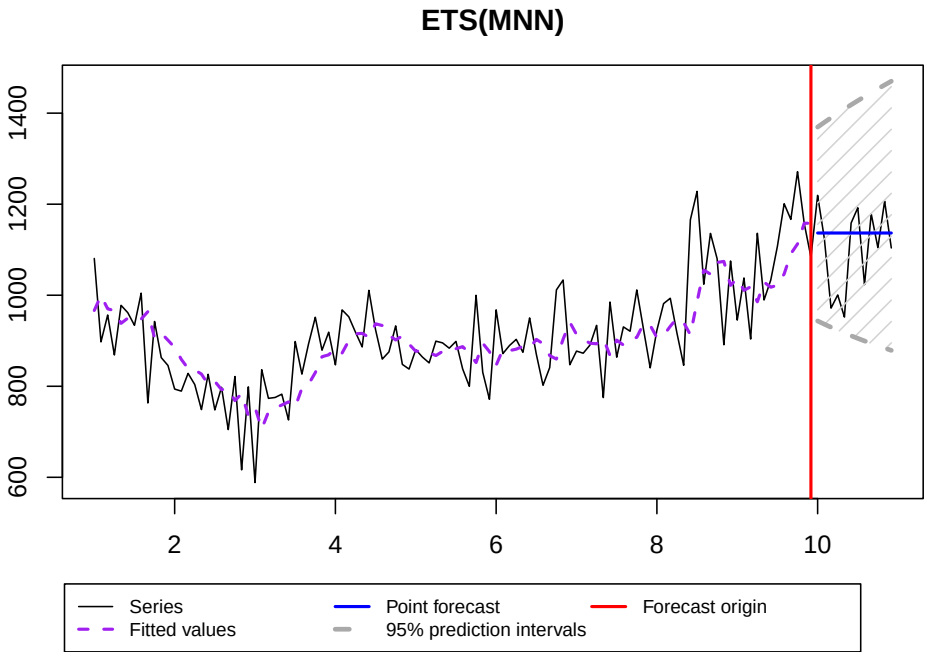


Figure 4.10: (#fig:ETSMNNExample)ETS(M,N,N) model applied to the data generated from the same model.

Conceptually, the data in Figure @ref(fig:ETSMNNExample) looks very similar to the one from ETS(A,N,N) (Figure @ref(fig:ETSANNExample)), but demonstrating the changing variance of the error term with the change of the level.

4.3 Several examples of ETS and related exponential smoothing methods

There are other exponential smoothing methods, which include more components, as discussed in Section @ref(tsComponents). This includes but is not limited to: Holt's (**?**, originally proposed in 1957), Holt-Winter's (**?**), multiplicative trend (**?**), Damped trend (originally proposed by **?**, and then picked up by **?**), Damped trend Holt-Winters (**?**) and damped multiplicative trend (**?**) methods. We will not discuss them here one by one, as we will not use them further in this textbook. Instead, we will focus on the ETS models underlying them.

We already understand that there can be different components in time series and that they can interact either in an additive or a multiplicative way, which gives us the taxonomy above, discussed in Section @ref(ETSTaxonomy). This section considers several examples of ETS models and their relations to the conventional exponential smoothing methods.

4.3.1 ETS(A,A,N)

This is also sometimes known as local trend model and is formulated similar to ETS(A,N,N), but with addition of the trend equation. It underlies Holt's method (**?**):

$$\begin{aligned} y_t &= l_{t-1} + b_{t-1} + \epsilon_t \\ l_t &= l_{t-1} + b_{t-1} + \alpha\epsilon_t, (\#eq:ETSAAN) \\ b_t &= b_{t-1} + \beta\epsilon_t \end{aligned} \tag{4.18}$$

where β is the smoothing parameter for the trend component. It has a similar idea as ETS(A,N,N): the states evolve, and the speed of their change depends on the values of α and β. The trend is not deterministic in this model: both the intercept and the slope change over time. The higher the smoothing parameters are, the more uncertain the level and the slope will be; thus, the higher the uncertainty about the future values is.

Here is an example of the data that corresponds to the ETS(A,A,N) model:

```
y <- sim.es("AAN", 120, 1, 12, persistence=c(0.3,0.1),
            initial=c(1000,20), mean=0, sd=20)
plot(y)
```

The series in Figure @ref(fig:ETSAANExample) demonstrates a trend that changes over time. If we needed to produce forecasts for this data, we would

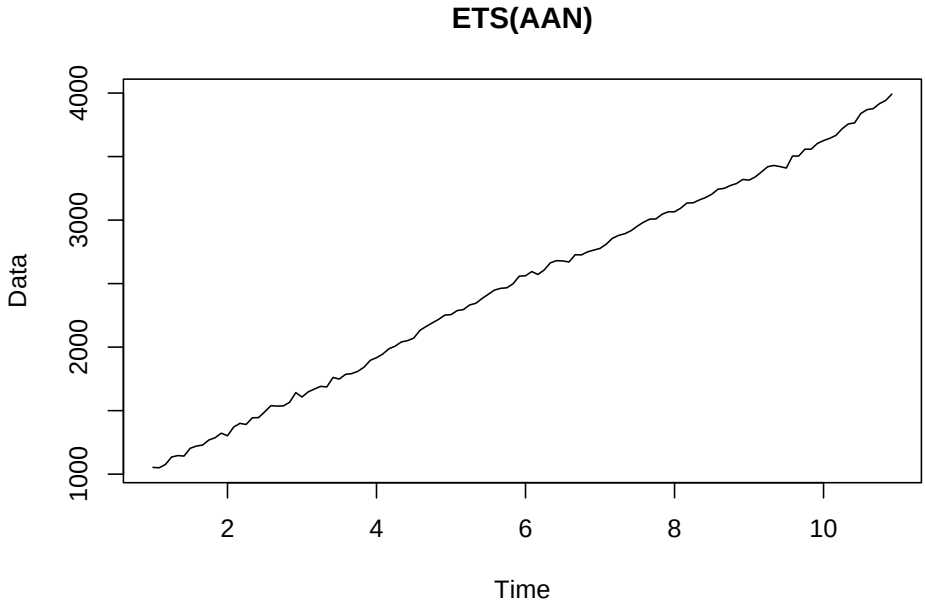


Figure 4.11: (#fig:ETSAANExample)Data generated from ETS(A,A,N) model.

capture the dynamics of the trend component and then use the last values for the several steps ahead prediction.

The point forecast h steps ahead from this model is a straight line with a slope b_t (as shown in Table @ref(tab:ETSAdditiveError) from Section @ref(ETSTaxonomyMaths)):

$$\mu_{y,t+h|t} = \hat{y}_{t+h} = l_t + hb_t . (\#eq : ETSAANForecast) \tag{4.19}$$

This becomes apparent if one takes the conditional expectations $\mathrm{E}(l_{t+h}|t)$ and $\mathrm{E}(b_{t+h}|t)$ in the second and third equations of @ref(eq:ETSAAN) and then inserts them in the measurement equation. Graphically it will look as shown in Figure @ref(fig:ETSAANExampleForecast):

```
esModel <- es(y, h=10, silent=FALSE)
```

If you want to experiment with the model and see how its parameters influence the fit and forecast, you can use the following R code:

```
esModel <- es(y$data, h=10, silent=FALSE, persistence=c(0.2,0.1))
```

where `persistence` is the vector of smoothing parameters (first $\hat{\alpha}$, then $\hat{\beta}$). By changing their values, we will make the model less/more responsive to the changes in the data.

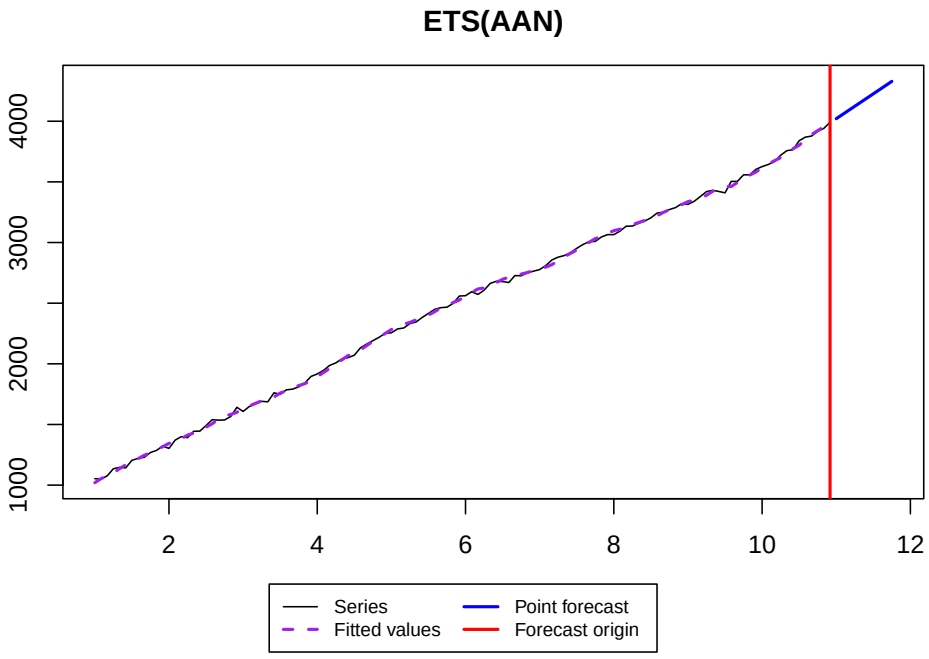


Figure 4.12: (#fig:ETSAANExampleForecast)ETS(A,A,N) and a point forecast produced from it.

4.3.2 ETS(A,Ad,N)

This is the model that underlies Damped trend method (**?**):

$$\begin{aligned} y_t &= l_{t-1} + \phi b_{t-1} + \epsilon_t \\ l_t &= l_{t-1} + \phi b_{t-1} + \alpha \epsilon_t, (\#eq:ETSAAdN) \\ b_t &= \phi b_{t-1} + \beta \epsilon_t \end{aligned} \tag{4.20}$$

where ϕ is the dampening parameter, typically lying between 0 and 1. If it is equal to zero, the model reduces to ETS(A,N,N), @ref(eq:ETSANN). If it is equal to one, it becomes equivalent to ETS(A,A,N), @ref(eq:ETSAAN). The dampening parameter slows down the trend, making it non-linear. An example of data that corresponds to ETS(A,Ad,N) is provided in Figure @ref(fig:ETSAAdNExample).

```
y <- sim.es("AAdN", 120, 1, 12, persistence=c(0.3,0.1),
            initial=c(1000,20), phi=0.95, mean=0, sd=20)
plot(y)
```

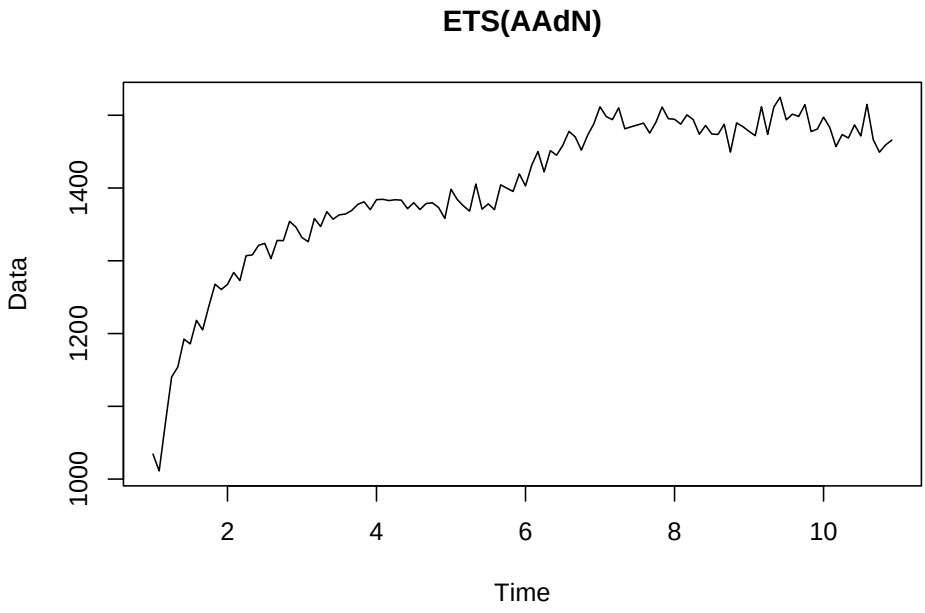


Figure 4.13: (#fig:ETSAAdNExample)An example of ETS(A,Ad,N) data.

Visually it is typically challenging to distinguish ETS(A,A,N) from ETS(A,Ad,N) data. So, some other model selection techniques are recommended (see Section @ref(ETSSelection)).

The point forecast from this model is a bit more complicated (see Section

@ref(ETSTaxonomyMaths)):

$$\mu_{y,t+h|t} = \hat{y}_{t+h} = l_t + \sum_{j=1}^{h} \phi^j b_t . (\#eq : ETSAAdNForecast) \tag{4.21}$$

It corresponds to the slowing down trajectory, as shown in Figure @ref(fig:ETSAAdNExampleForecast).

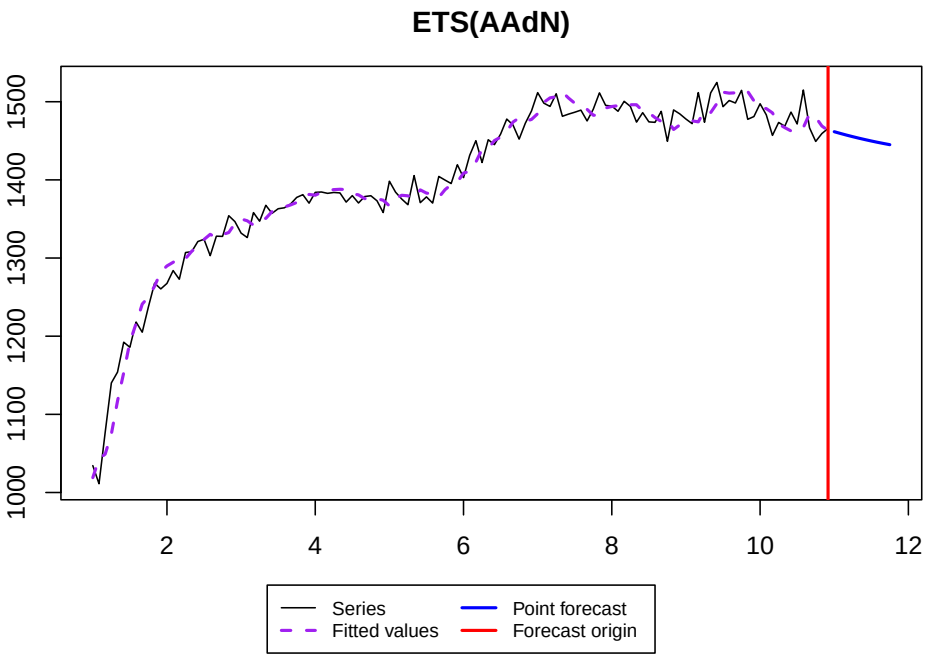


Figure 4.14: (#fig:ETSAAdNExampleForecast)A point forecast from ETS(A,Ad,N).

As can be seen in Figure @ref(fig:ETSAAdNExampleForecast), the forecast trajectory from the ETS(A,Ad,N) has a slowing down element in it. This is because of the $\phi = 0.95$ in our example.

4.3.3 ETS(A,A,M)

Finaly, this is an exotic model with additive error and trend, but multiplicative seasonality. Still, we list it here, because it underlies the Holt-Winters method

(**?**):

$$\begin{aligned} y_t &= (l_{t-1} + b_{t-1})s_{t-m} + \epsilon_t \\ l_t &= l_{t-1} + b_{t-1} + \alpha\frac{\epsilon_t}{s_{t-m}} \\ b_t &= b_{t-1} + \beta\frac{\epsilon_t}{s_{t-m}} \\ s_t &= s_{t-m} + \gamma\frac{\epsilon_t}{l_{t-1} + b_{t-1}} \end{aligned} \quad , (\#eq : ETSAAM) \tag{4.22}$$

where s_t is the seasonal component and γ is its smoothing parameter. This is one of the potentially unstable models, which due to the mix of components, might produce unreasonable forecasts because the seasonal component might become negative, while in the multiplicative model, it should always be positive. Still, it might work on the strictly positive high-level data. Figure @ref(fig:ETSAAMExample) shows how the data for this model can look like.

```
y <- sim.es("AAM", 120, 1, 4, persistence=c(0.3,0.05,0.2),
            initial=c(1000,20), initialSeason=c(0.9,1.1,0.8,1.2),
            mean=0, sd=20)
plot(y)
```

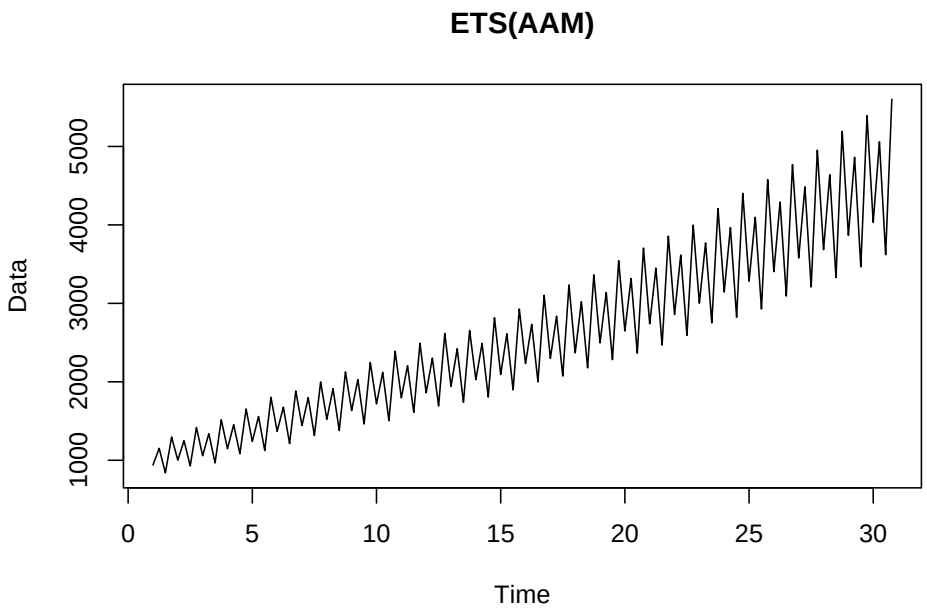


Figure 4.15: (#fig:ETSAAMExample)An example of ETS(A,A,M) data.

The data in Figure @ref(fig:ETSAAMExample) exhibits an additive trend with increasing seasonal amplitude, which are the two characteristics of the model.

Finally, the point forecast from this model build upon the ETS(A,A,N), introducing seasonal component:

$$\hat{y}_{t+h} = (l_t + hb_t)s_{t+h-m\lceil \frac{h}{m} \rceil}, (\#eq : ETSAAAForecast) \tag{4.23}$$

where $\lceil \frac{h}{m} \rceil$ is the rounded up value of the fraction in the brackets. The point forecast from this model is shown in Figure @ref(fig:ETSAAMExampleForecast).

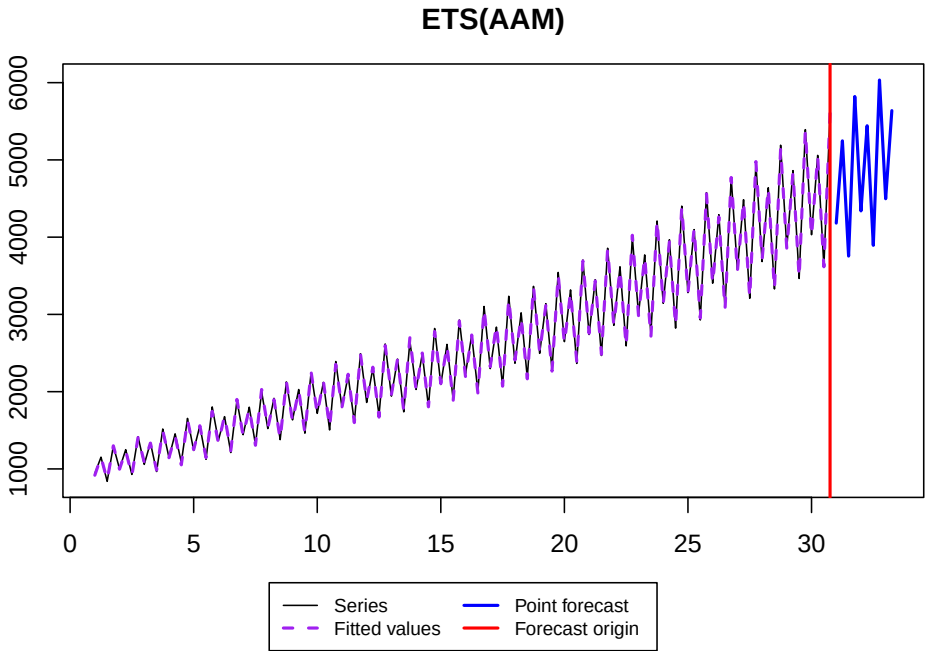


Figure 4.16: (#fig:ETSAAMExampleForecast)A point forecast from ETS(A,A,M).

Remark. *The point forecasts produced from this model do not correspond to the conditional expectations. This was discussed in Section @ref(ETSTaxonomyMaths).*

? argue that in ETS models, the error term should be aligned with the seasonal component because it is difficult to motivate why the amplitude of seasonality increases with the increase of level, while the variability of the error term stays the same. So, they recommend using ETS(M,A,M) instead of ETS(A,A,M) if you deal with positive high volume data. This is a reasonable recommendation, but keep in mind that both models might break if you deal with low volume data.

4.4 ETS assumptions, estimation and selection

Several assumptions need to hold for the conventional ETS models to work properly. Some of them have already been discussed in Section @ref(assumptions), and we will not discuss them here again. What is important in our context is that the conventional ETS assumes that the error term ϵ_t follows the normal distribution with zero mean and variance σ^2. The normal distribution is defined for positive, negative and zero values. This is not a big deal for additive models, which assume that the actual value can be anything. And it is not an issue for the multiplicative models when we deal with high-level positive data (e.g. thousands of units): the variance of the error term will be small enough for the ϵ_t not to become less than minus one. However, if the level of the data is low, then the variance of the error term can be large enough for the normally distributed error to cover negative values, less than minus one. This implies that the error term $1+\epsilon_t$ can become negative, and the model will break. This is a potential flaw in the conventional ETS model with the multiplicative error term. So, what the standard multiplicative error ETS model assumes, in fact, is that **the data we work with is strictly positive and has high-level values**.

Based on the assumption of normality of error term, the ETS model can be estimated via the maximisation of likelihood (see Chapter 13 of **?**), which is equivalent to the minimisation of the mean squared one step ahead forecast error e_t. Note that in order to apply the ETS models to the data, we also need to know the initial values of components, $\hat{l}_0, \hat{b}_0, \hat{s}_{-m+2}, \hat{s}_{-m+3}, \dots, \hat{s}_0$. The conventional approach is to estimate these values together with the smoothing parameters during likelihood maximisation. As a result, the optimisation might involve a large number of parameters. In addition, the variance of the error term is considered as an additional parameter in the maximum likelihood estimation, so the number of parameters for different models is (here "*" stands for any type):

1. ETS(*,N,N) – 3 parameters: $\hat{l}_0$, $\hat{\alpha}$ and $\hat{\sigma}^2$;
2. ETS(*,*,N) – 5 parameters: $\hat{l}_0$, $\hat{b}_0$, $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\sigma}^2$;
3. ETS(*,*d,N) – 6 parameters: $\hat{l}_0$, $\hat{b}_0$, $\hat{\alpha}$, $\hat{\beta}$, $\hat{\phi}$ and $\hat{\sigma}^2$;
4. ETS(*,N,*) – 4+m-1 parameters: $\hat{l}_0$, $\hat{s}_{-m+2}, \hat{s}_{-m+3}, \dots, \hat{s}_0$, $\hat{\alpha}$, $\hat{\gamma}$ and $\hat{\sigma}^2$;
5. ETS(*,*,*) – 6+m-1 parameters: $\hat{l}_0$, $\hat{b}_0$, $\hat{s}_{-m+2}, \hat{s}_{-m+3}, \dots, \hat{s}_0$, $\hat{\alpha}$, $\hat{\beta}$, $\hat{\gamma}$ and $\hat{\sigma}^2$;
6. ETS(*,*d,*) – 7+m-1 parameters: $\hat{l}_0$, $\hat{b}_0$, $\hat{s}_{-m+2}, \hat{s}_{-m+3}, \dots, \hat{s}_0$, $\hat{\alpha}$, $\hat{\beta}$, $\hat{\gamma}$, $\hat{\phi}$ and $\hat{\sigma}^2$.

Note that in the case of seasonal models, we typically make sure that the initial seasonal indices are normalised, so we only need to estimate $m-1$ of them, the last one is calculated based on the linear combination of the others. For example, for the additive seasonality, it is equal to $-\sum_{j=1}^{m-1} s_j$ because the sum of all the indices should be equal to zero.

When it comes to selecting the most appropriate model, the conventional approach involves the application of all models to the data and then selecting the most appropriate of them based on an information criterion (see Section 13.4 of **?**). This was first proposed by **?**. In the case of the conventional ETS model, this relies on the likelihood value of normal distribution used in the estimation of the model.

Finally, the assumption of normality is used to generate prediction intervals from the model. There are typically two ways of doing that:

1. Calculating the variance of multiple steps ahead forecast error and then using it for the intervals construction (see Chapter 6 of **?**);
2. Generating thousands of possible paths for the components of the series and the actual values and then taking the necessary quantiles for the prediction intervals;

Typically, (1) is applied for pure additive models, where the closed forms for the variances are known, and the assumption of normality holds for several steps ahead. In some special cases of mixed models, approximations for variances work on short horizons (see Section 6.4 of **?**). But in all the other cases, (2) should be used, despite being typically slower than (1) and producing bounds that differ slightly from run to run due to randomness.

4.5 State space form of ETS

One of the main advantages of the ETS model is its state space form, which gives it the flexibility. **?** use the following general formulation of the model with the first equation called "measurement equation" and the second one "transition equation":

$$\begin{aligned} y_t &= w(\mathbf{v}_{t-1}) + r(\mathbf{v}_{t-1})\epsilon_t \\ \mathbf{v}_t &= f(\mathbf{v}_{t-1}) + g(\mathbf{v}_{t-1})\epsilon_t \end{aligned}, (\#eq : ETSConventionalStateSpace) \tag{4.24}$$

where $\mathbf{v}_t$ is the state vector, containing the components of series (level, trend and seasonal), $w(\cdot)$ is the measurement, $r(\cdot)$ is the error, $f(\cdot)$ is the transition and $g(\cdot)$ is the persistence functions. Depending on the types of components these functions can have different values:

1. Depending on the types of trend and seasonality $w(\mathbf{v}_{t-1})$ will be equal either to the addition or multiplication of components. The special cases were presented in tables @ref(tab:ETSAdditiveError) and @ref(tab:ETSMultiplicativeError) in the Section @ref(ETSTaxonomyMaths). For example, in case of ETS(M,M,M) it is: $w(\mathbf{v}_{t-1}) = l_{t-1}b_{t-1}s_{t-m}$;
2. If the error is additive, then $r(\mathbf{v}_{t-1}) = 1$, otherwise (in case of multiplicative error) it is $r(\mathbf{v}_{t-1}) = w(\mathbf{v}_{t-1})$. For example, for ETS(M,M,M) it will be $r(\mathbf{v}_{t-1}) = l_{t-1}b_{t-1}s_{t-m}$;

3. The transition function $f(\cdot)$ will produce values depending on the types of trend and seasonality and will correspond to the first parts in the Tables @ref(tab:ETSAdditiveError) and @ref(tab:ETSMultiplicativeError) of the transition equations (dropping the error term). This function records how components interact with each other and how they change from one observation to another (thus the term "transition"). An example is the ETS(M,M,M) model, for which the transition function will produce three values: $l_{t-1}b_{t-1}$, b_{t-1} and s_{t-m} respectively for the level, trend and seasonal components. So, if we drop the persistence function $g(\cdot)$ and the error term ϵ_t for a moment, the second equation in @ref(eq:ETSConventionalStateSpace) will be:

$$\begin{aligned} l_t &= l_{t-1}b_{t-1} \\ b_t &= b_{t-1} \quad , (\#eq:ETSMMMTransitionFunction) \\ s_t &= s_{t-m} \end{aligned} \tag{4.25}$$

4. Finally, the persistence function will differ from one model to another, but in some special cases it can either be: $g(\mathbf{v}_{t-1}) = \mathbf{g}$ if all components are additive, or $g(\mathbf{v}_{t-1}) = f(\mathbf{v}_{t-1})\mathbf{g}$ if they are all multiplicative. $\mathbf{g}$ is the vector of smoothing parameters, called in the ETS context the "persistence vector". An example of persistence function is the ETS(M,M,M) model, for which it is: $l_{t-1}b_{t-1}\alpha$, $b_{t-1}\beta$ and $s_{t-m}\gamma$ respectively for the level, trend and seasonal components. Uniting this with the transition function @ref(eq:ETSMMMTransitionFunction) we get the equation from the table @ref(tab:ETSMultiplicativeError):

$$\begin{aligned} l_t &= l_{t-1}b_{t-1} + l_{t-1}b_{t-1}\alpha\epsilon_t \\ b_t &= b_{t-1} + b_{t-1}\beta\epsilon_t \quad , (\#eq:ETSMMMTransitionEquation01) \\ s_t &= s_{t-m} + s_{t-m}\gamma\epsilon_t \end{aligned} \tag{4.26}$$

which can be simplified to:

$$\begin{aligned} l_t &= l_{t-1}b_{t-1}(1+\alpha\epsilon_t) \\ b_t &= b_{t-1}(1+\beta\epsilon_t) \quad .(\#eq:ETSMMMTransitionEquation) \\ s_t &= s_{t-m}(1+\gamma\epsilon_t) \end{aligned} \tag{4.27}$$

Some of mixed models have more complicated persistence function values. For example, for ETS(A,A,M) it is:

$$g(\mathbf{v}_{t-1}) = \begin{pmatrix} \alpha \frac{1}{s_{t-m}} \\ \beta \frac{1}{s_{t-m}} \\ \gamma \frac{1}{l_{t-1}+b_{t-1}} \end{pmatrix}, \tag{4.28}$$

which results in the state space model discussed in subsection @ref(ETSAAMModel).

The compact form @ref(eq:ETSConventionalStateSpace) is thus convenient, it underlies all the 30 ETS models discussed in the Sections @ref(ETSTaxonomy) and @ref(ETSTaxonomyMaths). Unfortunately, they cannot be used directly for deriving conditional values, so they are needed just for the general understanding of ETS and can be used in coding.

4.5.1 Pure additive state space model

The more useful state space model in ETS framework is the pure additive one, which, based on the discussion above, is formulated as:

$$\begin{aligned} y_t &= \mathbf{w}' \mathbf{v}_{t-1} + \epsilon_t \\ \mathbf{v}_t &= \mathbf{F}\mathbf{v}_{t-1} + \mathbf{g}\epsilon_t \end{aligned}, (\#eq:ETSConventionalStateSpaceAdditive) \tag{4.29}$$

where $\mathbf{w}$ is the measurement vector, showing how the components form the structure, $\mathbf{F}$ is the transition matrix, showing how components interact with each other and change over time (e.g. level is equal to the previous level plus trend) and $\mathbf{g}$ is the persistence vector, containing smoothing parameters. The conditional expectation and variance can be derived based on @ref(eq:ETSConventionalStateSpaceAdditive), together with bounds on the smoothing parameters for any model that can be formulated in this way. And, as mentioned above, any pure additive ETS model can be written in the form @ref(eq:ETSConventionalStateSpaceAdditive), which means that all of them have relatively simple analytical formulae for the statistics mentioned above. For example, the h steps ahead conditional expectation and variance of the model @ref(eq:ETSConventionalStateSpaceAdditive) are (**?**, Chapter 6):

$$\begin{aligned} \mu_{y,t+h} = \mathrm{E}(y_{t+h}|t) &= \mathbf{w}' \mathbf{F}^{h-1} \mathbf{v}_t \\ \sigma_h^2 = \mathrm{V}(y_{t+h}|t) &= \left(\mathbf{w}' \mathbf{F}^{j-1} \mathbf{g}\mathbf{g}' \mathbf{F}' \mathbf{w} + 1\right) \sigma^2 \end{aligned}, (\#eq:ETSConventionalStateSpaceAdditiveExpectation) \tag{4.30}$$

where σ^2 is the variance of the error term. Formulae @ref(eq:ETSConventionalStateSpaceAdditiveExpectation) can be used for generation of respective moments from any pure additive ETS model. The conditional expectation can also be used for some mixed models as an approximation for the true conditional mean.

4.6 Parameters bounds

While many practitioners and academics accept that the smoothing parameters of ETS models should lie between zero and one, this is not entirely true for the models. There are, in fact, several possible restrictions on smoothing parameters, and it is worth discussing them separately:

1. **Classical or conventional** bounds are $\alpha, \beta, \gamma \in (0, 1)$. The idea behind them originates from the simple exponential smoothing method (Section @ref(SES)), where it is logical to restrict the bounds with this region because then the smoothing parameters regulate what weight the actual

value y_t will have and what weight will be assigned to the predicted one $\hat{y}_t$. **?** showed that this condition is sometimes too loose and, in other cases, is too restrictive to some ETS models. **?** was one of the first to show that the bounds are wider than this region for many exponential smoothing methods. Still, the conventional restriction is the most often used in practice, just because it is nice to work with.

2. **Usual or traditional** bounds are those that satisfy the set of the following equations:

$$\begin{aligned} &\alpha \in [0, 1) \\ &\beta \in [0, \alpha) \qquad , (\#eq : ETSUsualBounds) \\ &\gamma \in [0, 1-\alpha) \end{aligned} \tag{4.31}$$

This set of restrictions guarantees that the weights decline over time exponentially (see Section @ref(whyExponential)), and the ETS models have the property of "averaging" the values over time. In the lower boundary condition, the model's components become deterministic, and we can say that they are calculated as the global averages of the values over time.

3. **Admissible** bounds, satisfying stability condition. The idea here is that the most recent observation should have a higher weight than the older ones, which is regulated via the smoothing parameters. However, in this case, we do not impose the restriction of exponential decay of weights on the models, so they can oscillate or decay harmonically as long as their absolute values decrease over time. The condition is more complicated mathematically than the previous two. It will be discussed later in the textbook for the pure additive models (see Section @ref(ADAMETSPureAdditive)), but here are several examples for bounds, satisfying this condition (from Chapter 10 of **?**):

- ETS(A,N,N): $\alpha \in (0, 2)$;
- ETS(A,A,N): $\alpha \in (0, 2); \beta \in (0, 4 - 2\alpha)$;
- ETS(A,N,A): $\alpha \in \left(\frac{-2}{m-1}, 2 - \gamma\right); \gamma \in (\max(-m\alpha, 0), 2 - \alpha)$;

As you see, the admissible bounds are much wider than the conventional and usual ones. In fact, smoothing parameters can become either negative or greater than one in some cases for some models, which is hard to interpret but might indicate that the data is difficult to predict. Furthermore, the admissible bounds correspond to the restrictions of the parameters for ARIMA models, underlying some of pure additive ETS models. In a way, they are more natural for the ETS models than the other two because they follow the formulation and arise naturally. However, their usage in practice has been met with mixed success, with only a handful of papers using them instead of (1) or (2) (e.g. **?**; mention that they appear in some cases and **?**, use them in their model).

The admissible bounds are calculated based on the discount matrix in the R code, which will be discussed in the context of pure additive ADAM ETS models in the chapter @ref(ADAMETSPureAdditive).

Chapter 5

Pure additive ADAM ETS

Now that we are familiar with the conventional ETS, we can move to the discussion of ADAM implementation, which has several important differences from the classical one. We start the discussion with the pure additive models, which are much easier to work with than other ETS models. This chapter focuses on technical details of the model, discussing general formulation in algebraic form, then moving to recursive relations, which are needed to understand how to produce forecasts from the model and how to estimate it correctly (i.e. impose restrictions on the parameters). Finally, we discuss the distributional assumptions for ADAM ETS, introducing not only the Normal distribution but also showing how to use Laplace, S, Generalised Normal, Log-Normal, Gamma and Inverse Gaussian distributions in the context.

5.1 Model formulation

The pure additive case is interesting, because this is the group of models that have closed forms for both conditional mean and variance. In order to understand how we can get to the general model form, we consider an example of ETS(A,A,A) model, which, as discussed in Section @ref(ETSTaxonomyMaths), is formulated as:

$$\begin{aligned} y_t &= l_{t-1} + b_{t-1} + s_{t-m} + \epsilon_t \\ l_t &= l_{t-1} + b_{t-1} + \alpha\epsilon_t \\ b_t &= b_{t-1} + \beta\epsilon_t \\ s_t &= s_{t-m} + \gamma\epsilon_t \end{aligned} . (\#eq:ETSADAMAAA) \tag{5.1}$$

The same model can be represented in the matrix form based on @ref(eq:ETSADAMAAA):

$$\begin{aligned} y_t &= \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} l_{t-1} \\ b_{t-1} \\ s_{t-m} \end{pmatrix} + \epsilon_t \\ \begin{pmatrix} l_t \\ b_t \\ s_t \end{pmatrix} &= \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} l_{t-1} \\ b_{t-1} \\ s_{t-m} \end{pmatrix} + \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \epsilon_t \end{aligned} , (\#eq:ETSADAMAAAMatrixForm) \tag{5.2}$$

I use spaces in equation @ref(eq:ETSADAMAAA) to show how the matrix form is related to the general one. The positions of l_{t-1}, b_{t-1} and s_{t-m} correspond to the non-zero values in the matrices in @ref(eq:ETSADAMAAAMatrixForm). Now we can give names to each matrix and vector, for example:

$$\begin{aligned} \mathbf{w} &= \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{F} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{g} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}, \\ \mathbf{v}_t &= \begin{pmatrix} l_t \\ b_t \\ s_t \end{pmatrix}, \mathbf{v}_{t-\mathbf{l}} = \begin{pmatrix} l_{t-1} \\ b_{t-1} \\ s_{t-m} \end{pmatrix}, \quad \mathbf{l} = \begin{pmatrix} 1 \\ 1 \\ m \end{pmatrix} \end{aligned} , (\#eq:ETSADAMAAAMatrices) \tag{5.3}$$

Substituting @ref(eq:ETSADAMAAAMatrices) into @ref(eq:ETSADAMAAAMatrixForm), we get the general pure additive ADAM ETS model:

$$\begin{aligned} y_t =& \mathbf{w}' \mathbf{v}_{t-\mathbf{l}} + \epsilon_t \\ \mathbf{v}_t =& \mathbf{F} \mathbf{v}_{t-\mathbf{l}} + \mathbf{g} \epsilon_t \end{aligned}, (\#eq:ETSADAMStateSpacePureAdditive) \tag{5.4}$$

where $\mathbf{w}$ is the measurement vector, $\mathbf{F}$ is the transition matrix, $\mathbf{g}$ is the persistence vector and $\mathbf{v}_{t-\mathbf{l}}$ is the vector of lagged components and $\mathbf{l}$ is the vector of lags. The important thing to note is that the ADAM model is based on the model discussed in Section @ref(ETSConventionalModelAdditive). It is formulated using lags of components rather than their transition over time. This comes to the elements of the vector $\mathbf{l}$. Just for the comparison, the conventional ETS(A,A,A), formulated according to @ref(eq:ETSConventionalStateSpace) would have the following transition matrix (instead of @ref(eq:ETSADAMAAAMatrices)):

$$\mathbf{F} = \begin{pmatrix} 1 & 1 & \mathbf{0}'_{m-1} & 0 \\ 0 & 1 & \mathbf{0}'_{m-1} & 0 \\ 0 & 0 & \mathbf{0}'_{m-1} & 1 \\ \mathbf{0}_{m-1} & \mathbf{0}_{m-1} & \mathbf{I}_{m-1} & \mathbf{0}_{m-1} \end{pmatrix}, (\#eq:ETSADAMAAAMatricesTransition) \tag{5.5}$$

where $\mathbf{I}_{m-1}$ is the identity matrix of the size $(m-1) \times (m-1)$ and $\mathbf{0}_{m-1}$ is the vector of zeroes of size $m-1$. The main benefit of using the vector of lags $\mathbf{l}$ instead of the conventional mechanism in the transition equation is in the reduction of dimensions of matrices (the transition matrix contains 3×3 elements

in case of ETS(A,A,A) instead of $(2 + m) \times (2 + m)$ as for the conventional ETS model). The model @ref(eq:ETSADAMStateSpacePureAdditive) is more parsimonious than the conventional one and simplifies some of the calculations, making it realistic, for example, to apply models to data with large frequency m (e.g. 24, 48, 52, 365). The main disadvantage of this approach is in the complications arising in the derivation of conditional expectation and variance, which still have closed forms, but are more cumbersome. They are discussed later in this chapter in Section @ref(pureAdditiveExpectationAndVariance).

5.2 Recursive relation

A useful thing that can be derived from the pure additive model @ref(eq:ETSADAMStateSpacePureAdditive) is the recursive value, which can be used for further inference.

First, when we produce forecast for h steps ahead, it is important to understand what the actual value h steps ahead might be, given all the information we have on the observation t:

$$\begin{aligned} y_{t+h} &= \mathbf{w}' \mathbf{v}_{t+h-\mathbf{l}} + \epsilon_{t+h} \\ \mathbf{v}_{t+h} &= \mathbf{F} \mathbf{v}_{t+h-\mathbf{l}} + \mathbf{g} \epsilon_{t+h} \end{aligned}, (\#eq:ETSADAMStateSpacePureAdditiveRecursion01) \tag{5.6}$$

where $\mathbf{v}_{t+h-\mathbf{l}}$ is the vector of previous states, given the lagged values $\mathbf{l}$. In order to obtain the recursion, we need to split the measurement and persisitence vectors together with the transition matrix into parts for the same lags of components, leading to the following transition equation in @ref(eq:ETSADAMStateSpacePureAdditiveRecursion01):

$$\begin{aligned} y_{t+h} &= (\mathbf{w}'_{m_1} + \mathbf{w}'_{m_2} + \dots + \mathbf{w}'_{m_d}) \mathbf{v}_{t-hl} + \epsilon_{t+h} \\ \mathbf{v}_{t+h} &= (\mathbf{F}_{m_1} + \mathbf{F}_{m_2} + \dots + \mathbf{F}_{m_d}) \mathbf{v}_{t-hl} + (\mathbf{g}_{m_1} + \mathbf{g}_{m_2} + \dots \mathbf{g}_{m_d}) \epsilon_{t+h} \end{aligned}, (\#eq:ETSADAMStateSpacePureAdditiveRec \tag{5.7}$$

where $m_1, m_2, \dots, m_d$ are the distinct seasonal frequencies. So, for example, in case of ETS(A,A,A) model on quarterly data (periodicity is equal to four), $m_1 = 1$, $m_2 = 4$, leading to $\mathbf{F}_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ and $\mathbf{F}_4 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, where the split of the transition matrix is done column-wise. This split of matrices and vectors into distinct sub matrices and subvectors is needed in order to get the correct recursion and obtain the correct conditional expectation and variance.

By substituting the values in the transition equation of @ref(eq:ETSADAMStateSpacePureAdditiveRecursion02)

with their previous values until we reach t, we get:

$$\begin{aligned}\mathbf{v}_{t-hl} =&\mathbf{F}_{m_1}^{\lceil \frac{h}{m_1} \rceil - 1} \mathbf{v}_t + \sum_{j=1}^{\lceil \frac{h}{m_1} \rceil - 1} \mathbf{F}_{m_1}^{j-1} \mathbf{g}_{m_1} \epsilon_{t+m_1 \lceil \frac{h}{m_1} \rceil - j} + \\ &\mathbf{F}_{m_2}^{\lceil \frac{h}{m_2} \rceil - 1} \mathbf{v}_t + \sum_{j=1}^{\lceil \frac{h}{m_2} \rceil - 1} \mathbf{F}_{m_2}^{j-1} \mathbf{g}_{m_2} \epsilon_{t+m_2 \lceil \frac{h}{m_2} \rceil - j} + \\ &... \\ &\mathbf{F}_{m_d}^{\lceil \frac{h}{m_d} \rceil - 1} \mathbf{v}_t + \sum_{j=1}^{\lceil \frac{h}{m_d} \rceil - 1} \mathbf{F}_{m_d}^{j-1} \mathbf{g}_{m_d} \epsilon_{t+m_d \lceil \frac{h}{m_d} \rceil - j}\end{aligned} \tag{5.8}$$

Inserting @ref(eq:ETSADAMStateSpacePureAdditiveRecursion03) in the measurement equation of @ref(eq:ETSADAMStateSpacePureAdditiveRecursion02), we get:

$$\begin{aligned}y_{t+h} =&\mathbf{w}'_{m_1} \mathbf{F}_{m_1}^{\lceil \frac{h}{m_1} \rceil - 1} \mathbf{v}_t + \mathbf{w}'_{m_1} \sum_{j=1}^{\lceil \frac{h}{m_1} \rceil - 1} \mathbf{F}_{m_1}^{j-1} \mathbf{g}_{m_1} \epsilon_{t+m_1 \lceil \frac{h}{m_1} \rceil - j} + \\ &\mathbf{w}'_{m_2} \mathbf{F}_{m_2}^{\lceil \frac{h}{m_2} \rceil - 1} \mathbf{v}_t + \mathbf{w}'_{m_2} \sum_{j=1}^{\lceil \frac{h}{m_2} \rceil - 1} \mathbf{F}_{m_2}^{j-1} \mathbf{g}_{m_2} \epsilon_{t+m_2 \lceil \frac{h}{m_2} \rceil - j} + \\ &\cdots + \\ &\mathbf{w}'_{m_d} \mathbf{F}_{m_d}^{\lceil \frac{h}{m_d} \rceil - 1} \mathbf{v}_t + \mathbf{w}'_{m_d} \sum_{j=1}^{\lceil \frac{h}{m_d} \rceil - 1} \mathbf{F}_{m_d}^{j-1} \mathbf{g}_{m_d} \epsilon_{t+m_d \lceil \frac{h}{m_d} \rceil - j} + \\ &\epsilon_{t+h}\end{aligned} \tag{5.9}$$

Substituting the specific values of $m_1, m_2, \dots, m_d$ in @ref(eq:ETSADAMStateSpacePureAdditiveRecursion will simplify the equation and make it easier to understand. For example, for ETS(A,N,N), $m_1 = 1$ and all the other frequencies are equal to zero, so the recursion @ref(eq:ETSADAMStateSpacePureAdditiveRecursion04) simplifies to:

$$y_{t+h} = \mathbf{w}'_1 \mathbf{F}_1^{h-1} \mathbf{v}_t + \mathbf{w}'_1 \sum_{j=1}^{h-1} \mathbf{F}_1^{j-1} \mathbf{g}_1 \epsilon_{t+h-j} + \epsilon_{t+h}, \tag{5.10}$$

which is the recursion obtained by **?**, page 103.

5.3 Conditional expectation and variance

Now, why is the recursion @ref(eq:ETSADAMStateSpacePureAdditiveRecursion04) important? This is because we can take the expectation and variance of

@ref(eq:ETSADAMStateSpacePureAdditiveRecursion04) conditional on the values of the state vector $\mathbf{v}_t$ on the observation t (assuming that the error term is homoscedastic, uncorrelated and has the expectation of zero) in order to get:

$$\begin{aligned} \mu_{y,t+h} &= \mathrm{E}(y_{t+h}|t) = \sum_{i=1}^{d} \left(\mathbf{w}'_{m_i} \mathbf{F}_{m_i}^{\lceil \frac{h}{m_i} \rceil - 1} \right) \mathbf{v}_t \\ \sigma_h^2 &= \mathrm{V}(y_{t+h}|t) = \left(\sum_{i=1}^{d} \left(\mathbf{w}'_{m_i} \sum_{j=1}^{\lceil \frac{h}{m_i} \rceil - 1} \mathbf{F}_{m_i}^{j-1} \mathbf{g}_{m_i} \mathbf{g}'_{m_i} (\mathbf{F}'_{m_i})^{j-1} \mathbf{w}_{m_i} \right) + 1 \right) \sigma^2 \end{aligned}, (\#eq:ETSADAMStateSpacePureAd \tag{5.11}$$

where σ^2 is the variance of the error term. These two formulae are cumbersome, but they give the analytical solutions to the two moments. Having obtained both of them, we can construct prediction intervals, assuming, for example, that the error term follows normal distribution (see Section @ref(ADAMForecastingPI) for details):

$$y_{t+h} \in \left(\mathrm{E}(y_{t+h}|t) + z_{\frac{\alpha}{2}} \sqrt{\mathrm{V}(y_{t+h}|t)}, \mathrm{E}(y_{t+h}|t) + z_{1-\frac{\alpha}{2}} \sqrt{\mathrm{V}(y_{t+h}|t)} \right), (\#eq:ETSADAMStateSpacePureAdditivePre \tag{5.12}$$

where $z_{\frac{\alpha}{2}}$ is quantile of standardised normal distribution for the level α. When it comes to other distributions (see Section @ref(ADAMETSAdditiveDistributions)), in order to get the conditional h steps ahead scale parameter, we can first calculate the variance using @ref(eq:ETSADAMStateSpacePureAdditiveRecursionMeanAndVariance) and then using the relation between the scale and the variance for the specific distribution (see discussion in Chapter 3 of **?**) to get the necessary value.

5.3.1 Example with ETS(A,N,N)

For example, for the ETS(A,N,N) model, discussed above, we get:

$$\begin{aligned} \mathrm{E}(y_{t+h}|t) =& \mathbf{w}'_1 \mathbf{F}_1^{h-1} \mathbf{v}_t \\ \mathrm{V}(y_{t+h}|t) =& \left(\mathbf{w}'_1 \sum_{j=1}^{h-1} \mathbf{F}_1^{j-1} \mathbf{g}_1 \mathbf{g}'_1 (\mathbf{F}'_1)^{j-1} \mathbf{w}_1 + 1 \right) \sigma^2 \end{aligned}, (\#eq:ETSADAMStateSpaceANNRecursionMeanAndVarian \tag{5.13}$$

or by substituting $\mathbf{F} = 1$, $\mathbf{w} = 1$, $\mathbf{g} = \alpha$ and $\mathbf{v}_t = l_t$:

$$\begin{aligned} \mu_{y,t+h} =& l_t \\ \sigma_h^2 =& \left((h-1)\alpha^2 + 1 \right) \sigma^2 \end{aligned}, (\#eq:ETSADAMStateSpaceANNRecursionMeanAndVariance) \tag{5.14}$$

which is the same conditional expectation and variance as in the Section @ref(ETSTaxonomyMaths) and in the **?** textbook.

5.4 Stability and forecastability conditions

Another important aspect of the pure additive model @ref(eq:ETSADAMStateSpacePureAdditive) is the restriction on the smoothing parameters. This is related to the stability and forecastability conditions of the model. The **stability** implies that the weights for observations decay over time (Section @ref(whyExponential)), guaranteeing that the newer ones will have higher weights than the older ones. If this condition holds, the model behaves "steadily", forgetting the past values eventually. The **forecastability** does not guarantee that the weights decay, but it guarantees that the initial value of the state vector will have a constant impact on forecasts, i.e. will not increase in weight with the increase of the forecast horizon. An example of the non-stable, but forecastable model is ETS(A,N,N) with $\alpha = 0$. In this case, it reverts to the global level model (Section @ref(GlobalMean)), where the initial value impacts the final forecast in the same way as it does for the first observation.

In order to obtain both conditions, we need to use a reduced form of ETS by inserting the measurement equation in the transition equation via $\epsilon_t = y_t - \mathbf{w}'\mathbf{v}_{t-\mathbf{l}}$:

$$\begin{aligned}\mathbf{v}_t =& \mathbf{F}\mathbf{v}_{t-\mathbf{l}} + \mathbf{g}\left(y_t - \mathbf{w}'\mathbf{v}_{t-\mathbf{l}}\right) \\ =& \left(\mathbf{F} - \mathbf{g}\mathbf{w}'\right)\mathbf{v}_{t-\mathbf{l}} + \mathbf{g}y_t\end{aligned} .(\#eq:ETSADAMStateSpacePureAdditiveBackRecursion01) \tag{5.15}$$

The matrix $\mathbf{D} = \mathbf{F} - \mathbf{g}\mathbf{w}'$ is called the discount matrix and it shows how the weights diminish over time. It is the main part of the model that determines, whether the model will be stable / forecastable or not.

5.4.1 Example with ETS(A,N,N)

In order to better understand what we plan to discuss in this section, consider an example of ETS(A,N,N) model, for which $\mathbf{F} = 1$, $\mathbf{w} = 1$, $\mathbf{g} = \alpha$, $\mathbf{v}_t = l_t$ and $\mathbf{l} = 1$. Inserting these values in @ref(eq:ETSADAMStateSpacePureAdditiveBackRecursion01), we get:

$$l_t = (1 - \alpha)\, l_{t-1} + \alpha y_t, \quad .(\#eq:ETSADAMStateSpaceANNBackRecursion01) \tag{5.16}$$

which corresponds to the formula of Simple Exponential Smoothing from Section @ref(SES). The discount matrix, in this case, is $\mathbf{D} = 1 - \alpha$. If we now substitute the values for the level on the right-hand side of the equation @ref(eq:ETSADAMStateSpaceANNBackRecursion01) by the previous values of the level, we will obtain the recursion that we have already discussed in Section @ref(whyExponential), but now in terms of the "true" components and parameters:

$$l_t = \alpha \sum_{j=0}^{t-1} (1-\alpha)^j y_{t-j} + (1-\alpha)^t l_0, \quad .(\#eq:ETSADAMStateSpaceANNBackRecursion02) \tag{5.17}$$

The *stability* condition for ETS(A,N,N) is that the discount value $1 - \alpha$ is less than or equal to one by absolute value. This way, the weights will decay in time because of the exponentiation in @ref(eq:ETSADAMStateSpaceANNBackRecursion02) to the power of j. This condition is satisfied when $\alpha \in (0, 2)$, which is admissible bound discussed in Section @ref(ETSParametersBounds).

As for the *forecastability* condition, in this case it implies that $\lim_{t \to \infty} (1 - \alpha)^t = \text{const}$, which means that the effect of the initial state on future values stays the same. This is achievable, for example, when $\alpha = 0$, but is violated, when $\alpha < 0$ or $\alpha \geq 2$. So, the bounds for the smoothing parameters in the ETS(A,N,N) model, guaranteeing the forecastability of the model (i.e. making it useful), are:

$$\alpha \in [0, 2). (\#eq : ETSADAMStateSpaceANNBounds) \tag{5.18}$$

5.4.2 Comming back to the general case

In the general case, the logic is the same as with ETS(A,N,N), but it implies the usage of linear algebra. Due to our lagged formulation, the recursion becomes more complicated:

$$\begin{aligned}
\mathbf{v}_t = &\mathbf{D}_{m_1}^{\lceil \frac{t}{m_1} \rceil} \mathbf{v}_0 + \sum_{j=0}^{\lceil \frac{t}{m_1} \rceil - 1} \mathbf{D}_{m_1}^{j} y_{t-jm_1} + \\
&\mathbf{D}_{m_2}^{\lceil \frac{t}{m_2} \rceil} \mathbf{v}_0 + \sum_{j=0}^{\lceil \frac{t}{m_2} \rceil - 1} \mathbf{D}_{m_2}^{j} y_{t-jm_2} + \\
&\cdots + \\
&\mathbf{D}_{m_d}^{\lceil \frac{t}{m_d} \rceil} \mathbf{v}_0 + \sum_{j=0}^{\lceil \frac{t}{m_d} \rceil - 1} \mathbf{D}_{m_d}^{j} y_{t-jm_d}
\end{aligned}, (\#eq : ETSADAMStateSpacePureAdditiveRecursion05) \tag{5.19}$$

where $\mathbf{D}_{m_i} = \mathbf{F}_{m_i} - \mathbf{g}_{m_i} \mathbf{w}'_{m_i}$ is the discount matrix for each lagged part of the model. The stability condition in this case is that the absolute values of all the non-zero eigenvalues of the discount matrices $\mathbf{D}_{m_i}$ are lower than one. This condition can be checked at the model construction stage, ensuring that the selected parameters guarantee the stability of the model. As for the forecastability, the idea is that the initial value of the state vector should not have an increasing impact on the last observed value, which is obtained by inserting @ref(eq:ETSADAMStateSpacePureAdditiveRecursion05) in the mea-

surement equation:

$$\begin{aligned} y_t =& \mathbf{w}'_{m_1} \mathbf{D}_{m_1}^{\lceil \frac{t-1}{m_1} \rceil} \mathbf{v}_0 + \mathbf{w}'_{m_1} \sum_{j=0}^{\lceil \frac{t-1}{m_1} \rceil - 1} \mathbf{D}_{m_1}^{j} y_{t-1-jm_1} + \\ & \mathbf{w}'_{m_2} \mathbf{D}_{m_2}^{\lceil \frac{t-1}{m_2} \rceil} \mathbf{v}_0 + \mathbf{w}'_{m_2} \sum_{j=0}^{\lceil \frac{t-1}{m_2} \rceil - 1} \mathbf{D}_{m_2}^{j} y_{t-1-jm_2} +, (\#eq : ETSADAMStateSpacePureAdditiveRecursion \\ & \cdots + \\ & \mathbf{w}'_{m_d} \mathbf{D}_{m_d}^{\lceil \frac{t-1}{m_d} \rceil} \mathbf{v}_0 + \mathbf{w}'_{m_d} \sum_{j=0}^{\lceil \frac{t-1}{m_d} \rceil - 1} \mathbf{D}_{m_d}^{j} y_{t-1-jm_d} \end{aligned} \tag{5.20}$$

and analysing the impact of $\mathbf{v}_0$ on the actual value y_t. In our case **forecastability** condition implies that:

$$\lim_{t \to \infty} \left(\mathbf{w}'_{m_i} \mathbf{D}_{m_i}^{\lceil \frac{t-1}{m_i} \rceil} \mathbf{v}_0 \right) = \text{const for all } i = 1, \dots, d. (\#eq : ETSADAMStateSpacePureAdditiveRecursion \tag{5.21}$$

5.5 Distributional assumptions in pure additive ETS

While the conventional ETS assumes that the error term follows Normal distribution, ADAM ETS proposes some flexibility, implementing the following options for the error term distribution in the additive error models:

1. Normal: $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$;
2. Laplace: $\epsilon_t \sim \mathcal{L}aplace(0, s)$;
3. S: $\epsilon_t \sim \mathcal{S}(0, s)$;
4. Generalised Normal: $\epsilon_t \sim \mathcal{GN}(0, s, \beta)$.

The conditional moments and stability / forecastability conditions do not change for the model with these new assumptions. The main element that changes is the scale and the width of prediction intervals. Given that scales of these distributions are linearly related to the variance, one can calculate the conditional variance as discussed in Section @ref(pureAdditiveExpectationAndVariance) and then use it in order to obtain the respective scales. Having the scales it becomes straightforward to calculate the needed quantiles for the prediction intervals. Here are the formulae for the scales of distributions mentioned above:

1. Normal: scale is σ_h^2;
2. Laplace: $s_h = \sigma_h \sqrt{\frac{1}{2}}$;
3. S: $s_h = \sqrt{\sigma_h} \sqrt[4]{\frac{1}{120}}$;

4. Generalised Normal: $s_h = \sigma_h \sqrt{\frac{\Gamma(1/\beta)}{\Gamma(3/\beta)}}$.

The estimation of pure additive ETS models can be done via the maximisation of the likelihood of the assumed distribution (see Chapter 13 of **?**), which in some cases coincide with the popular losses (e.g. Normal and MSE, or Laplace and MAE).

In addition, the following more exotic options for the additive error models are available in ADAM ETS:

1. Log Normal: $\left(1 + \frac{\epsilon_t}{\mu_{y,t}}\right) \sim \log\mathcal{N}\left(-\frac{\sigma^2}{2}, \sigma^2\right)$. Here $\mu_{y,t} = \mathbf{w}'\mathbf{v}_{t-\mathbf{l}}$ is one step ahead point forecast, σ^2 is the variance of the error term in logarithms and the $-\frac{\sigma^2}{2}$ appears due to the restriction $\mathrm{E}(\epsilon_t) = 0$.
2. Inverse Gaussian: $\left(1 + \frac{\epsilon_t}{\mu_{y,t}}\right) \sim \mathcal{IG}(1, s)$;
3. Gamma: $\left(1 + \frac{\epsilon_t}{\mu_{y,t}}\right) \sim \Gamma(s^{-1}, s)$;

The possibility of application of these distributions arises from the reformulation of the original pure additive model @ref(eq:ETSADAMStateSpacePureAdditive) into:

$$\begin{aligned} y_t =& \mathbf{w}'\mathbf{v}_{t-\mathbf{l}}\left(1 + \frac{\epsilon_t}{\mathbf{w}'\mathbf{v}_{t-\mathbf{l}}}\right). (\#eq:ETSADAMStateSpacePureAdditiveReformulated) \\ \mathbf{v}_t =& \mathbf{F}\mathbf{v}_{t-\mathbf{l}} + \mathbf{g}\epsilon_t \end{aligned} \tag{5.22}$$

The connection between the two formulations becomes apparent when opening the brackets in the measurement equation of @ref(eq:ETSADAMStateSpacePureAdditiveReformulated). Note that in this case, the model assumes that the data is strictly positive, and while it might be possible to fit the model on the data with negative values, the calculation of the scale and the likelihood might become impossible. Using alternative losses (e.g. MSE) is a possible solution in this case.

5.6 Examples of application

5.6.1 Non-seasonal data

To see how the pure additive ADAM ETS works, we will try it out using the `adam()` function from the `smooth` package for R on Box-Jenkins sales data. We start with plotting the data:

```
plot(BJsales)
```

The series in Figure @ref(fig:BJsalesDataPlot) seem to exhibit trend, so we will apply ETS(A,A,N) model:

```
adamModel <- adam(BJsales, "AAN")
adamModel
```

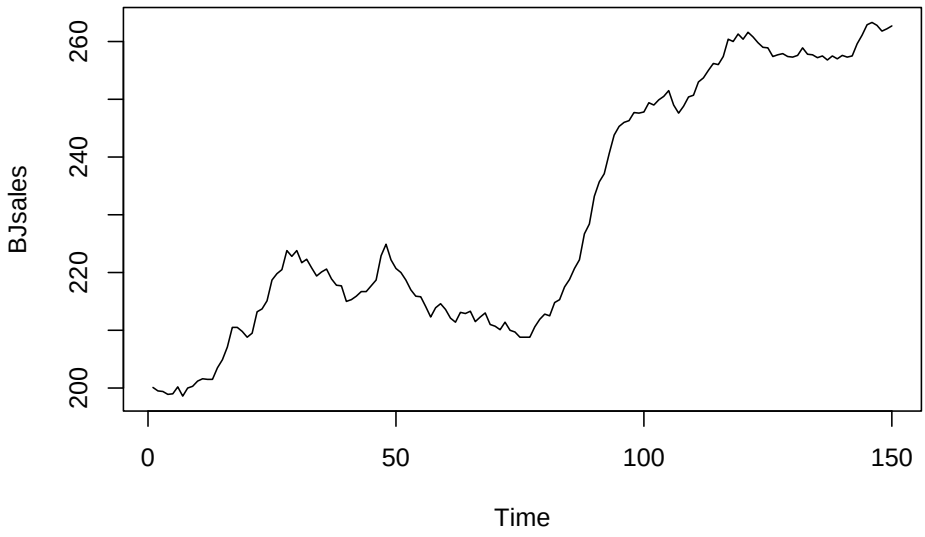


Figure 5.1: (#fig:BJsalesDataPlot)Box-Jenkins sales data.

```
## Time elapsed: 0.02 seconds
## Model estimated using adam() function: ETS(AAN)
## Distribution assumed in the model: Normal
## Loss function type: likelihood; Loss function value: 258.8098
## Persistence vector g:
##  alpha   beta
## 1.0000 0.2438
##
## Sample size: 150
## Number of estimated parameters: 5
## Number of degrees of freedom: 145
## Information criteria:
##      AIC     AICc      BIC     BICc
## 527.6196 528.0362 542.6728 543.7166
```

The model's output summarises which specific model was constructed, assuming what distribution, how it was estimated, and the values of smoothing parameters. It also reports the sample size, the number of parameters, degrees of freedom and produces information criteria (see Section 13.4 of **?**). We can compare this model with the ETS(A,N,N) to see which of them performs better in terms of information criteria (e.g. in terms of AICc):

```
adam(BJsales, "ANN")
```

```
## Time elapsed: 0.01 seconds
## Model estimated using adam() function: ETS(ANN)
```

```
## Distribution assumed in the model: Normal
## Loss function type: likelihood; Loss function value: 273.2898
## Persistence vector g:
## alpha
##     1
##
## Sample size: 150
## Number of estimated parameters: 3
## Number of degrees of freedom: 147
## Information criteria:
##      AIC     AICc      BIC     BICc
## 552.5795 552.7439 561.6114 562.0233
```

In this situation the AICc for ETS(A,N,N) is higher than for ETS(A,A,N), so we should use the latter for forecasting purposes. We can produce point forecasts and prediction interval (in this example we will construct 90% and 95% ones) and plot them (Figure @ref(fig:BJsalesAANForecast)):

```
plot(forecast(adamModel, h=10,
              interval="prediction", level=c(0.9,0.95)))
```

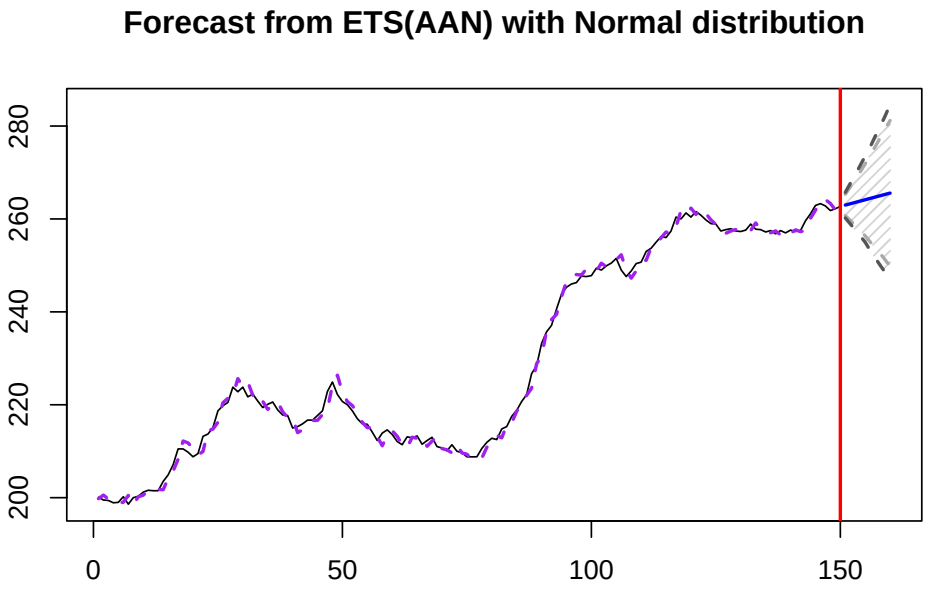


Figure 5.2: (#fig:BJsalesAANForecast)Forecast for Box-Jenkins sales data from ETS(A,A,N) model.

Notice that the smoothing parameters of ETS(A,A,N) are very high, with $\alpha = 1$. This might mean that the maximum likelihood is achieved in the *admissible* bounds. We can try it out and see what happens:

```
adamModel <- adam(BJsales, "AAN", bounds="admissible")
adamModel
```

```
## Time elapsed: 0.04 seconds
## Model estimated using adam() function: ETS(AAN)
## Distribution assumed in the model: Normal
## Loss function type: likelihood; Loss function value: 258.5358
## Persistence vector g:
##  alpha   beta
## 1.0541 0.2185
##
## Sample size: 150
## Number of estimated parameters: 5
## Number of degrees of freedom: 145
## Information criteria:
##      AIC     AICc      BIC     BICc
## 527.0716 527.4883 542.1248 543.1687
```

Both smoothing parameters are now higher, which implies that the uncertainty about the future values of states is higher as well, which is then reflected in the slightly wider prediction interval (Figure @ref(fig:BJsalesAANForecastAdmissible)):

```
plot(forecast(adamModel, h=10,
              interval="prediction", level=c(0.9,0.95)))
```

Although the values of smoothing parameters are larger than one, the model is still stable. In order to see that, we can calculate the discount matrix $\mathbf{D}$ using the objects returned by the function:

```
discountMatrix <- adamModel$transition - adamModel$persistence %*%
  adamModel$measurement[nobs(adamModel),,drop=FALSE]
eigen(discountMatrix)$values
```

```
## [1]  0.79538429 -0.06800887
```

Notice that the absolute values of both eigenvalues in the matrix are less than one, which means that the newer observations have higher weights than the older ones and that the absolute values of weights decrease over time, making the model stable.

If we want to test ADAM ETS with another distribution, it can be done using the respective parameter (here we use Generalised Normal, estimating the shape together with the other parameters):

```
adamModel <- adam(BJsales, "AAN", distribution="dgnorm")
print(adamModel, digits=3)
```

```
## Time elapsed: 0.03 seconds
## Model estimated using adam() function: ETS(AAN)
```

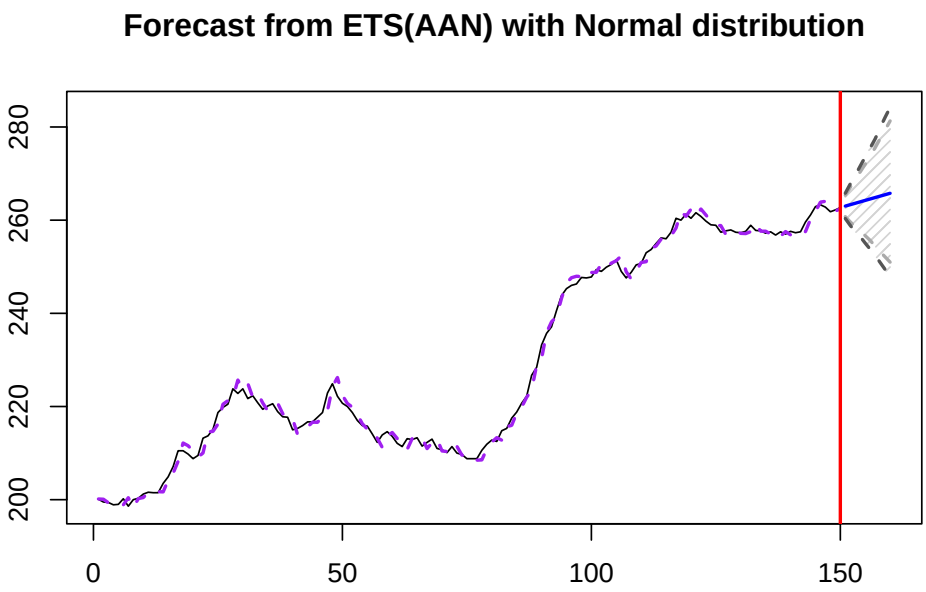


Figure 5.3: (#fig:BJsalesAANForecastAdmissible)Forecast for Box-Jenkins sales data from ETS(A,A,N) model with admissible bounds.

```
## Distribution assumed in the model: Generalised Normal with shape=1.741
## Loss function type: likelihood; Loss function value: 258.456
## Persistence vector g:
## alpha  beta
## 1.000 0.217
##
## Sample size: 150
## Number of estimated parameters: 6
## Number of degrees of freedom: 144
## Information criteria:
##     AIC    AICc     BIC    BICc
## 528.913 529.500 546.977 548.448
```

Similar to the previous cases, we can plot the forecasts from the model:

```
plot(forecast(adamModel, h=10, interval="prediction"))
```

The prediction interval in this case is slightly wider than in the previous one, because $\mathcal{GN}$ distribution with $\beta = 1.74$ has fatter tails than the normal distribution (Figure @ref(fig:BJsalesAANForecastGN)).

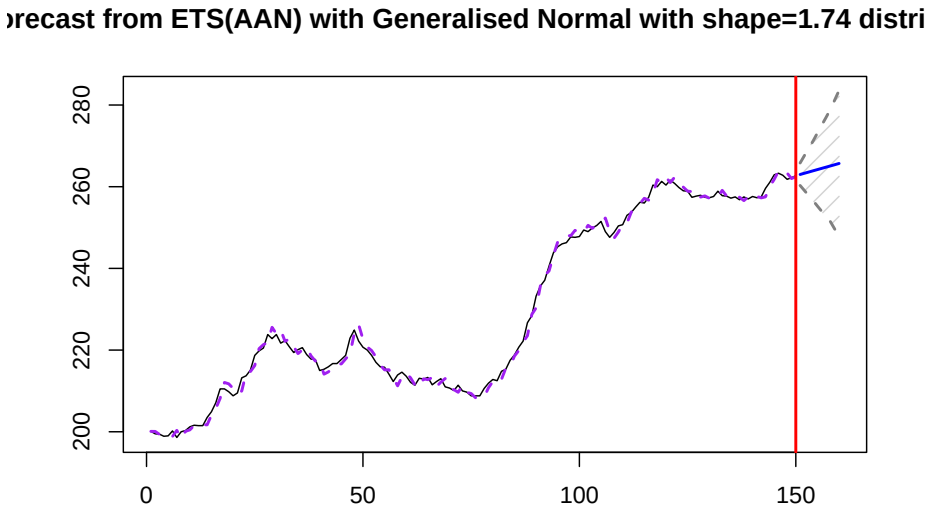


Figure 5.4: (#fig:BJsalesAANForecastGN)Forecast for Box-Jenkins sales data from ETS(A,A,N) model with Generalised Normal distribution.

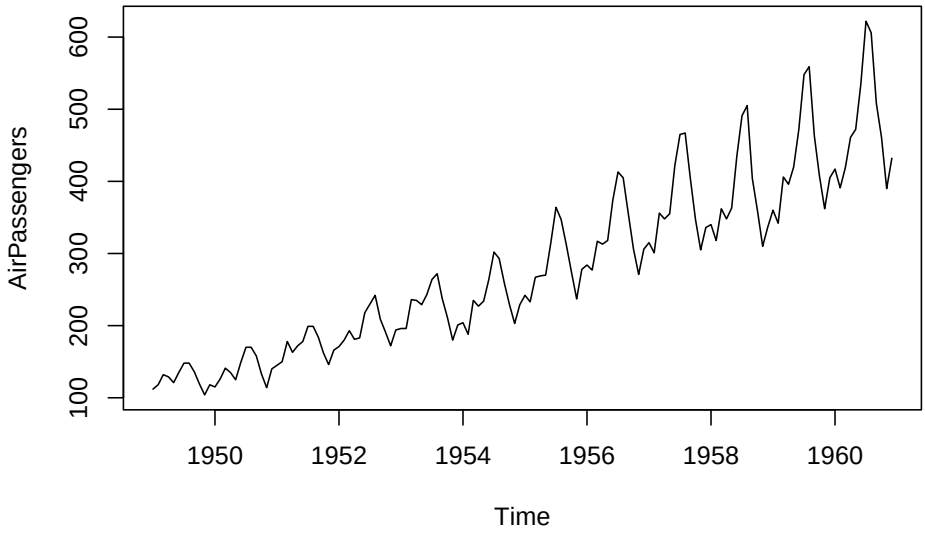


Figure 5.5: (#fig:AirPassengersPlot)Air passengers data from Box-Jenkins textbook.

5.6.2 Seasonal data

Now we will check what happens in the case of seasonal data. We use `AirPassengers` data, plotted in Figure @ref(fig:AirPassengersPlot), which has multiplicative seasonality. But for demonstration purposes, we will see what happens when we use the wrong model with additive seasonality. We will withhold 12 observations to look closer at the performance of the ETS(A,A,A) model in this case:

```
adamModel <- adam(AirPassengers, "AAA", lags=12,
                  h=12, holdout=TRUE)
```

Remark. *In this specific case, the `lags` parameter is not necessary because the function will automatically get the frequency from the `ts` object. If we were to provide a vector of values instead of the `ts` object, we would need to specify the correct lag. Note that `1` (lag for level and trend) is unnecessary; the function will always use it anyway.*

Remark. *In some cases, the optimiser might converge to the local minimum, so if you find the results unsatisfactory, it might make sense to reestimate the model tuning the parameters of the optimiser (see Section @ref(ADAMInitialisation) for details). Here is an example (we increase the number of iterations in the optimisation and set new starting values for the smoothing parameters):*

```
adamModel$B[1:3] <- c(0.2,0.1,0.3)
adamModel <- adam(AirPassengers, "AAA", lags=12,
                  h=12, holdout=TRUE,
                  B=adamModel$B, maxeval=1000)
adamModel
```

```
## Time elapsed: 0.19 seconds
## Model estimated using adam() function: ETS(AAA)
## Distribution assumed in the model: Normal
## Loss function type: likelihood; Loss function value: 513.0026
## Persistence vector g:
##  alpha   beta  gamma
## 0.1928 0.0000 0.8072
##
## Sample size: 132
## Number of estimated parameters: 17
## Number of degrees of freedom: 115
## Information criteria:
##      AIC     AICc      BIC     BICc
## 1060.005 1065.374 1109.013 1122.119
##
## Forecast errors:
## ME: 6.216; MAE: 14.162; RMSE: 17.75
```

```
## sCE: 28.418%; Asymmetry: 51.9%; sMAE: 5.395%; sMSE: 0.457%
## MASE: 0.588; RMSSE: 0.567; rMAE: 0.186; rRMSE: 0.172
```

Notice that because we fit the seasonal additive model to the data with multiplicative seasonality, the smoothing parameter γ has become large – the seasonal component needs to be updated to keep up with the changing seasonal profile. In addition, because we use the `holdout` parameter, the function also reports the error measures for the point forecasts on that part of the data. This can be useful when comparing the performance of several models on a time series. Here is how the forecast from ETS(A,A,A) looks on this data:

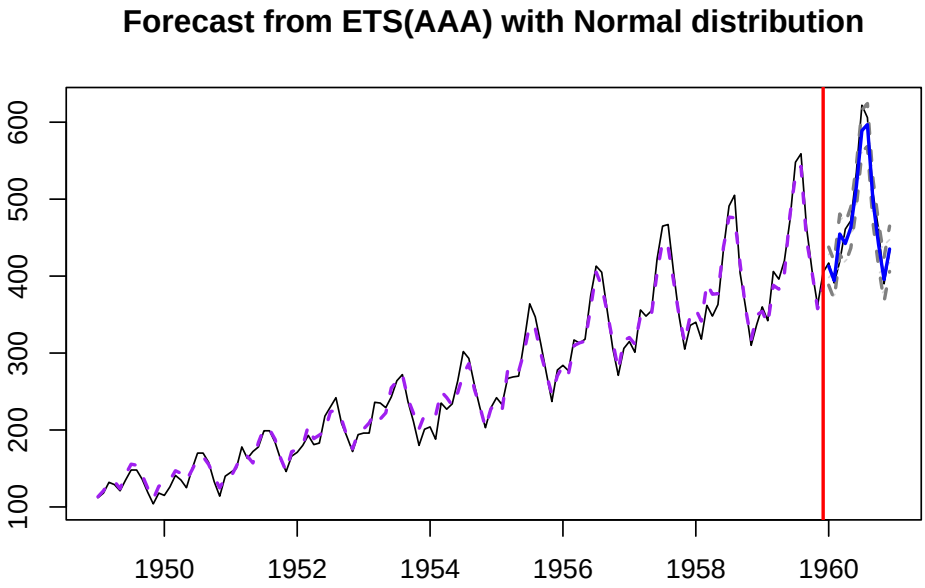


Figure 5.6: (#fig:AirPassengersAAAForecast)Forecast for air passengers data using ETS(A,A,A) model.

Figure @ref(fig:AirPassengersAAAForecast) demonstrates that while the fit to the data is far from perfect, due to a pure coincidence, the point forecast from this model is decent.

In order to see how the ADAM ETS decomposes the data into components, we can plot it via the `plot()` method with `which` parameter:

We can see on the graph in Figure @ref(fig:AirPassengersAAADecomposition) that the residuals still contain some seasonality, so there is room for improvement. This probably happened because the data exhibits multiplicative seasonality rather than the additive one. For now, we do not aim to fix this issue.

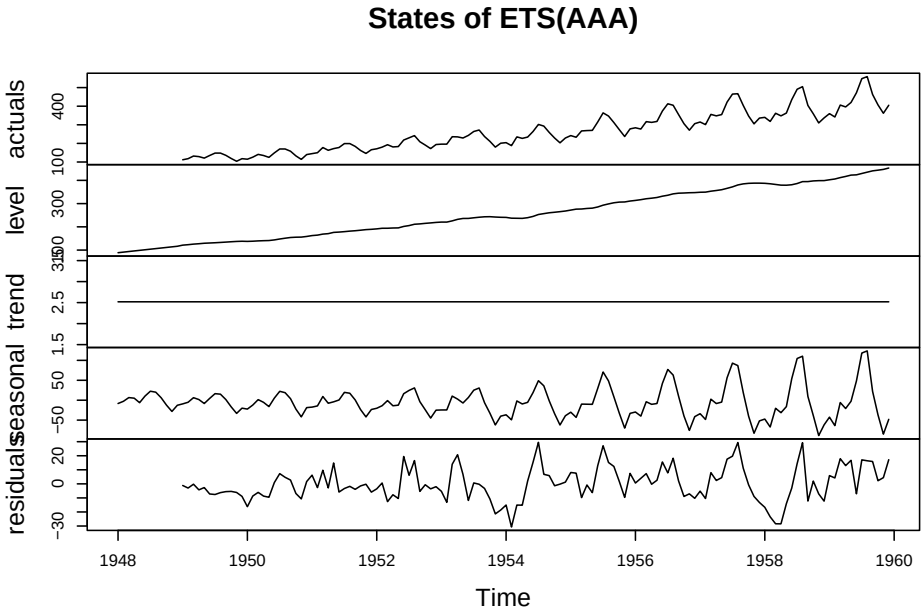


Figure 5.7: (#fig:AirPassengersAAADecomposition)Decomposition of air passengers data using ETS(A,A,A) model.

Chapter 6

Pure multiplicative ADAM ETS

There is a reason why we discuss pure multiplicative ADAM ETS models separately: they are suitable for the positive data, especially when the level is low, yet they do not rely on prior data transformations (such as taking logarithms or applying a power transform). However, the models discussed in this chapter are not easy to work with – they typically do not have closed forms for the conditional h steps ahead mean and variance and do not have well-defined parameters space. Furthermore, they make more sense in conjunction with positive-valued distributions, although they also work with the normal one. All these aspects are discussed in this chapter.

6.1 Model formulation

The pure multiplicative ETS implemented in ADAM framework can be formulated using logarithms, similar to how the pure additive ADAM ETS is formulated in @ref(eq:ETSADAMStateSpacePureAdditive):

$$\begin{aligned} \log y_t =& \mathbf{w}' \log(\mathbf{v}_{t-\mathbf{l}}) + \log(1 + \epsilon_t) \\ \log \mathbf{v}_t =& \mathbf{F} \log \mathbf{v}_{t-\mathbf{l}} + \log(\mathbf{1}_k + \mathbf{g}\epsilon_t) \end{aligned}, (\#eq:ETSADAMStateSpacePureMultiplicative) \tag{6.1}$$

where $\mathbf{1}_k$ is the vector of ones, containing k elements (number of components in the model), log is the natural logarithm, applied element-wise to the vectors, and all the other values have been discussed in the previous sections. An example of a pure multiplicative model is ETS(M,M,M), for which we have the following

values:

$$\mathbf{w} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{F} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{g} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}, \\ \mathbf{v}_t = \begin{pmatrix} l_t \\ b_t \\ s_t \end{pmatrix}, \mathbf{l} = \begin{pmatrix} 1 \\ 1 \\ m \end{pmatrix}, \qquad \mathbf{1}_k = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} .(\#eq:ETSADAMMMMMatrices) \tag{6.2}$$

By inserting these values in the equation @ref(eq:ETSADAMStateSpacePureMultiplicative), we obtain model in logarithms:

$$\begin{aligned} \log y_t &= \log l_{t-1} + \log b_{t-1} + \log s_{t-m} + \log (1 + \epsilon_t) \\ \log l_t &= \log l_{t-1} + \log b_{t-1} + \log(1 + \alpha\epsilon_t) \\ \log b_t &= \log b_{t-1} + \log(1 + \beta\epsilon_t) \\ \log s_t &= \log s_{t-m} + \log(1 + \gamma\epsilon_t) \end{aligned} , (\#eq:ETSADAMMMMMLogs) \tag{6.3}$$

which after exponentiation becomes equal to the one, discussed in Section @ref(ETSTaxonomyMaths):

$$\begin{aligned} y_t &= l_{t-1} b_{t-1} s_{t-m} (1 + \epsilon_t) \\ l_t &= l_{t-1} b_{t-1} (1 + \alpha\epsilon_t) \\ b_t &= b_{t-1} (1 + \beta\epsilon_t) \\ s_t &= s_{t-m} (1 + \gamma\epsilon_t) \end{aligned} .(\#eq:ETSADAMMMMM) \tag{6.4}$$

An interesting observation is that the model @ref(eq:ETSADAMMMMMLogs) will produce values similar to the model ETS(A,A,A) applied to the data in logarithms, when the values of smoothing parameters are close to zero. This becomes apparent, when recalling the limit:

$$\lim_{x \to 0} \log(1 + x) = x .(\#eq:limitOf1x) \tag{6.5}$$

Based on that, the model will become close to the following one in cases of small values of smoothing parameters:

$$\begin{aligned} \log y_t &= \log l_{t-1} + \log b_{t-1} + \log s_{t-m} + \epsilon_t \\ \log l_t &= \log l_{t-1} + \log b_{t-1} + \alpha\epsilon_t \\ \log b_t &= \log b_{t-1} + \beta\epsilon_t \\ \log s_t &= \log s_{t-m} + \gamma\epsilon_t \end{aligned} , (\#eq:ETSADAMMMMMLogsEquivalent) \tag{6.6}$$

which is the ETS(A,A,A) applied to the data in the logarithms. In many cases, the smoothing parameters will be small enough for the limit @ref(eq:limitOf1x) to hold, so the two models will produce similar forecasts. The main benefit of @ref(eq:ETSADAMMMMMLogsEquivalent) is that it has closed forms for the conditional mean and variance, so

the model @ref(eq:ETSADAMMMMLogsEquivalent) can be used instead of @ref(eq:ETSADAMMMMLogs) when the smoothing parameters are close to zero, and the variance of the error term is small to get conditional moments and quantiles of distribution. However, the form @ref(eq:ETSADAMMMMLogsEquivalent) does not permit mixed components – it only supports the multiplicative ones, making it detached from the other ETS models.

6.2 Recursive relation

Similarly to how it was done for the pure additive model in Section @ref(adamETSPureAdditiveRecursive), we can show what the recursive relation will look like for the pure multiplicative one (the logic here is the same, the main difference is in working with logarithms instead of the original values):

$$\begin{aligned}\log y_{t+h} =& \mathbf{w}'_{m_1}\mathbf{F}_{m_1}^{\lceil \frac{h}{m_1} \rceil -1} \log \mathbf{v}_t + \mathbf{w}'_{m_1} \sum_{j=1}^{\lceil \frac{h}{m_1} \rceil -1} \mathbf{F}_{m_1}^{j-1} \log\left(\mathbf{1}_k + \mathbf{g}_{m_1}\epsilon_{t+m_1\lceil \frac{h}{m_1} \rceil -j}\right) + \\ & \mathbf{w}'_{m_2}\mathbf{F}_{m_2}^{\lceil \frac{h}{m_2} \rceil -1} \log \mathbf{v}_t + \mathbf{w}'_{m_2} \sum_{j=1}^{\lceil \frac{h}{m_2} \rceil -1} \mathbf{F}_{m_2}^{j-1} \log\left(\mathbf{1}_k + \mathbf{g}_{m_2}\epsilon_{t+m_2\lceil \frac{h}{m_2} \rceil -j}\right) + \\ & ... \\ & \mathbf{w}'_{m_d}\mathbf{F}_{m_d}^{\lceil \frac{h}{m_d} \rceil -1} \log \mathbf{v}_t + \mathbf{w}'_{m_d} \sum_{j=1}^{\lceil \frac{h}{m_d} \rceil -1} \mathbf{F}_{m_d}^{j-1} \log\left(\mathbf{1}_k + \mathbf{g}_{m_d}\epsilon_{t+m_d\lceil \frac{h}{m_d} \rceil -j}\right) + \\ & \log\left(1 + \epsilon_{t+h}\right)\end{aligned} .(\#eq:ETSADAMStateSpacePureM \tag{6.7}$$

In order to see how this recursion works, we can take the example of ETS(M,N,N), for which $m_1 = 1$ and all the other frequencies are equal to zero:

$$y_{t+h} = \exp\left(\mathbf{w}'_1\mathbf{F}_1^{h-1}\log \mathbf{v}_t + \mathbf{w}'_1\sum_{j=1}^{h-1}\mathbf{F}_1^{j-1}\log\left(\mathbf{1}_k + \mathbf{g}_1\epsilon_{t+h-j}\right) + \log\left(1+\epsilon_{t+h}\right)\right), (\#eq:ETSMNNADAMStateSpa \tag{6.8}$$

or after inserting $\mathbf{w}_1 = 1$, $\mathbf{F}_1 = 1$, $\mathbf{v}_t = l_t$, $\mathbf{g}_1 = \alpha$ and $\mathbf{1}_k = 1$:

$$y_{t+h} = l_t \prod_{j=1}^{h-1}\left(1+\alpha\epsilon_{t+h-j}\right)\left(1+\epsilon_{t+h}\right) .(\#eq:ETSMNNADAMStateSpacePureMultiplicativeRecursion02) \tag{6.9}$$

This recursion is useful to understand how the states evolve, and in the case of ETS(M,N,N), it allows obtaining the conditional expectation and variance. But in general, for models with trend and/or seasonality, it cannot be used to calculate moments, like the one for the pure additive ADAM ETS. This is discussed in Section @ref(pureMultiplicativeExpectationAndVariance)).

6.3 The problem with moments in pure multiplicative ETS

The recursion @ref(eq:ETSADAMStateSpacePureMultiplicativeRecursion) obtained in the previous subsection shows how the previous values influence the logarithms of states. While it is possible to calculate the expectation of the logarithm of the variable y_{t+h}, in general, this does not allow deriving the expectation of the variable in the original scale. This is because of the convolution of terms $\log(\mathbf{1}_k + \mathbf{g}_{m_i}\epsilon_{t+j})$ for different j. To better understand this issue, we consider this persistence part of the equation for the ETS(M,N,N) model:

$$\log(1+\alpha\epsilon_t) = \log(1-\alpha+\alpha(1+\epsilon_t)). (\#eq:ETSMNNADAMPersistenceIssue) \tag{6.10}$$

Whatever we assume about the distribution of the variable $(1 + \epsilon_t)$, the distribution of @ref(eq:ETSMNNADAMPersistenceIssue) will be more complicated. For example, if we assume that $(1 + \epsilon_t) \sim \log\mathcal{N}(0, \sigma^2)$, then the distribution of @ref(eq:ETSMNNADAMPersistenceIssue) is something like exp three-parameter log normal distribution (**?**). The convolution of @ref(eq:ETSMNNADAMPersistenceIssue) for different t does not follow a known distribution, so it is not possible to calculate the conditional expectation and variance based on @ref(eq:ETSADAMStateSpacePureMultiplicativeRecursion). Similar issues arrise if we assume any other distribution. The problem is worsened in case of multiplicative trend and / or multiplicative seasonality models, because then the recursion @ref(eq:ETSADAMStateSpacePureMultiplicativeRecursion) contains several errors on the same observation (e.g. $\log(1 + \alpha\epsilon_t)$ and $\log(1 + \beta\epsilon_t)$).

The only way to derive the conditional expectation and variance for the pure multiplicative models is to use the formulae from tables @ref(tab:ETSAdditiveError) and @ref(tab:ETSMultiplicativeError) in Section @ref(ETSTaxonomyMaths) and manually derive the values in the original scale. This works well only for the ETS(M,N,N) model, for which it is possible to take conditional expectation and variance of the recursion @ref(eq:ETSMNNADAMStateSpacePureMultiplicativeRecursion02) to obtain:

$$\begin{aligned} \mu_{y,t+h} = \mathrm{E}(y_{t+h}|t) =& l_t \\ \mathrm{V}(y_{t+h}|t) =& l_t^2 \left(\left(1 + \alpha^2\sigma^2\right)^{h-1} (1 + \sigma^2) - 1 \right), \end{aligned} (\#eq:ETSMNNADAMConditionalValues) \tag{6.11}$$

where σ^2 is the variance of the error term. For the other models, the conditional moments do not have a general closed forms because of the product of $\log(1 + \alpha\epsilon_t)$, $\log(1 + \beta\epsilon_t)$ and $\log(1 + \gamma\epsilon_t)$. It is still possible to derive the moments for special cases of h, but this is a tedious process. In order to see that, we

demonstrate here how the recursion looks for ETS(M,Md,M) model:

$$y_{t+h} = l_{t+h-1} b_{t+h-1}^{\phi} s_{t+h-m} \left(1 + \epsilon_{t+h}\right) =$$
$$l_t b_t^{\sum_{j=1}^{h} \phi^j} s_{t+h-m\lceil \frac{h}{m} \rceil} \prod_{j=1}^{h-1} \left(\left(1 + \alpha\epsilon_{t+j}\right) \prod_{i=1}^{j} (1 + \beta\epsilon_{t+i})^{\phi^{j-i}} \right) \prod_{j=1}^{\lceil \frac{h}{m} \rceil} \left(1 + \gamma\epsilon_{t+j}\right) \left(1 + \epsilon_{t+h}\right), (\#eq : ETSMMdMADAMRecu \tag{6.12}$$

The conditional expectation of the recursion @ref(eq:ETSMMdMADAMRecursion) does not have a simple form, because of the difficulties in calculating the expectation of $(1 + \alpha\epsilon_{t+j})(1 + \beta\epsilon_{t+i})^{\phi^{j-i}}(1 + \gamma\epsilon_{t+j})$. In a simple example of $h = 2$ and $m > h$ the conditional expectation based on @ref(eq:ETSMMdMADAMRecursion) can be simplified to:

$$\mu_{y,t+2} = l_t b_t^{\phi+\phi^2} \left(1 + \alpha\beta\sigma^2\right), (\#eq : ETSMMdMADAMRecursionHorizon2) \tag{6.13}$$

introducing the second moment, the variance of the error term σ^2. The case of $h = 3$ implies the appearance of the third moment, the $h = 4$ – the fourth etc. This is why there are no closed forms for the conditional moments for the pure multiplicative models with trend and/or seasonality. In some special cases, when smoothing parameters and the variance of error term are all low, it is possible to use approximate formulae for some of the multiplicative models. These are discussed in Chapter 6 of **?**. In a special case, when all smoothing parameters are equal to zero or when $h = 1$, the conditional expectation will coincide with the point forecast from the Tables @ref(tab:ETSAdditiveError) and @ref(tab:ETSMultiplicativeError) in Section @ref(ETSTaxonomyMaths). But in general, the best thing that can be done in this case is the simulation of possible paths (using the formulae from the tables mentioned above) and then the calculation of mean and variance based on them. Finally, it can be shown for pure multiplicative models that:

$$\hat{y}_{t+h} \leq \check{y}_{t+h} \leq \mu_{y,t+h}, (\#eq : ETSADAMpointValueInequality) \tag{6.14}$$

where $\mu_{y,t+h}$ is the conditional h steps ahead expectation, $\check{y}_{t+h}$ is the conditional h steps ahead geometric expectation (expectation in logarithms) and $\hat{y}_{t+h}$ is the point forecast (**?**).

6.4 Smoothing parameters bounds

Similar to the pure additive ADAM ETS, it is possible to have different restrictions on smoothing parameters for pure multiplicative models. However, in this case, the classical and the usual restrictions become more reasonable from the model's point of view. In contrast, the derivation of admissible bounds becomes a challenging task. Consider the ETS(M,N,N) model, for which the level is updated using the following relation:

$$l_t = l_{t-1}(1+\alpha\epsilon_t) = l_{t-1}(1-\alpha+\alpha(1+\epsilon_t)).(\#eq : ETSMNNADAMLevelUpdate) \tag{6.15}$$

As discussed previously, the main benefit of pure multiplicative models is in dealing with positive data. So, it is reasonable to assume that $(1 + \epsilon_t) > 0$, which implies that the actual values will always be positive and that each model component should also be positive. This means that $\alpha(1 + \epsilon_t) > 0$, which implies that $(1 - \alpha + \alpha(1 + \epsilon_t)) > 1 - \alpha$ or equivalently based on @ref(eq:ETSMNNADAMPersistenceIssue) $(1+\alpha\epsilon_t) > 1-\alpha$ should always hold. In order for the model to make sense, the condition $(1 + \alpha\epsilon_t) > 0$ should hold as well, ensuring that the level is always positive. Connecting the two inequalities, this can be achieved when $1 - \alpha \geq 0$, meaning that $\alpha \leq 1$. Furthermore, for the level to be positive irrespective of the specific error on observation t, the smoothing parameter should be non-negative. So, in general, the bounds $[0, 1]$ guarantee that the model ETS(M,N,N) will produce positive values only. The two special cases $\alpha = 0$ and $\alpha = 1$ make sense because the level in @ref(eq:ETSMNNADAMLevelUpdate) will be positive in both of them, implying that for the former, the model becomes equivalent to the global level, while for the latter the model is equivalent to Random Walk. Using similar logic, it can be shown that the **classical restriction** $\alpha, \beta, \gamma \in [0, 1]$ guarantees that the model will always produce positive values.

The more restrictive condition of the **usual bounds**, discussed in Section @ref(ETSParametersBounds), makes sense as well, although it might be more restrictive than needed. But it has a different idea: guaranteeing that the model exhibits averaging properties.

Finally, the **admissible bounds** might still make sense for the pure multiplicative models, but the condition for parameters bounds becomes more complicated and implies that the distribution of the error term becomes trimmed from below to satisfy the classical restrictions discussed above. Very crudely, the conventional restriction from pure additive models can be used to approximate the proper admissible bounds, given the limit @ref(eq:limitOf1x), but this should be used with care, given the discussion above.

From the practical point of view, the pure multiplicative models typically have low smoothing parameters, close to zero, because they rely on multiplication of components rather than on addition, so even the classical restriction might seem broad in many situations.

6.5 Distributional assumptions in pure multiplicative ETS

The conventional assumption for the error term in ETS is that $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, which guarantees that the conditional expectation of the model will be equal to the point forecasts when the trend and seasonal components are not multiplicative. In general, ETS works well in many cases with this assumption, mainly when the data is strictly positive, and the level of series is high (e.g. thousands of units). However, this assumption might become unhelpful when dealing with

lower-level data because the models may start generating non-positive values, which contradicts the idea of pure multiplicative ETS models. **?** studied the ETS models with multiplicative error and suggested that applying ETS on data in logarithms is a better approach than just using ETS(M,Y,Y) models (here "Y" stands for non-additive components). However, this approach sidesteps the ETS taxonomy, creating a new group of models. An alternative (also discussed in **?**) is to assume that the error term $1+\epsilon_t$ follows some distribution for positive data. The authors mentioned log Normal, truncated and Gamma distributions but never explored them further.

? discussed several options for the distribution of $1 + \epsilon_t$ in ETS, including log Normal, Gamma and Inverse Gaussian. Other distributions for positive data can be applied as well, but their usage might become complicated, because they need to meet condition $\mathrm{E}(1 + \epsilon_t) = 1$ in order for the expectation to coincide with the point forecasts for models with non-multiplicative trend and seasonality. For example, if the error term follows log Normal distribution, then this restriction implies that the location of the distribution should be non-zero: $1+\epsilon_t \sim \log\mathcal{N}\left(-\frac{\sigma^2}{2}, \sigma^2\right)$. Using this principle the following distributions can be used for ADAM ETS:

1. Inverse Gaussian: $(1 + \epsilon_t) \sim \mathcal{IG}(1, s)$;
2. Gamma: $(1 + \epsilon_t) \sim \Gamma(s^{-1}, s)$;
3. Log Normal: $(1 + \epsilon_t) \sim \log\mathcal{N}\left(-\frac{\sigma^2}{2}, \sigma^2\right)$.

The MLE of s in $\mathcal{IG}$ is straightforward and is:

$$\hat{s} = \frac{1}{T}\sum_{t=1}^{T}\frac{e_t^2}{1+e_t}, (\#eq : ETSMultiplicativeErrorMLESigmaIG) \tag{6.16}$$

where e_t is the estimate of the error term ϵ_t. However, when it comes to the MLE of scale parameter for the log Normal distribution with the aforementioned restrictions, it is more complicated and is (**?**):

$$\hat{\sigma}^2 = 2\left(1 - \sqrt{1 - \frac{1}{T}\sum_{t=1}^{T}\log^2(1+e_t)}\right).(\#eq : ETSMultiplicativeErrorMLESigmaLogN) \tag{6.17}$$

Finally, MLE of s in Γ does not have a closed form. Luckily, method of moments can be used to obtain its value (**?**):

$$\hat{s} = \frac{1}{T}\sum_{t=1}^{T} e_t^2.(\#eq : ETSMultiplicativeErrorMLESigmaGamma) \tag{6.18}$$

Even if we deal with strictly positive high level data, it is not necessary to limit the distribution with Normal only. The following distributions can be applied as well:

1. Normal: $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, implying that $y_t = \mu_t(1 + \epsilon_t) \sim \mathcal{N}(\mu_t, \mu_t^2\sigma^2)$;

2. Laplace: $\epsilon_t \sim \mathcal{L}aplace(0, s)$, meaning that $y_t = \mu_t(1 + \epsilon_t) \sim \mathcal{L}aplace(\mu_t, \mu_t s)$;
3. S: $\epsilon_t \sim \mathcal{S}(0, s)$, so that $y_t = \mu_t(1 + \epsilon_t) \sim \mathcal{S}(\mu_t, \sqrt{\mu_t}s)$;
4. Generalised Normal: $\epsilon_t \sim \mathcal{GN}(0, s, \beta)$ and $y_t = \mu_t(1 + \epsilon_t) \sim \mathcal{GN}(\mu_t, \mu_t^{\beta} s)$;

Note that the MLE of scale parameters for these distributions will be calculated differently than in the case of pure additive models. For example, for the normal distribution it is:

$$\hat{\sigma}^2 = \frac{1}{T}\sum_{t=1}^{T} \frac{y_t - \hat{\mu}_t}{\hat{\mu}_t}, (\#eq : ETSMultiplicativeErrorMLESigmaNormal) \tag{6.19}$$

where the main difference from the additive error case arises from the measurement equation of the multiplicative error models:

$$y_t = \mu_t(1 + \epsilon_t), (\#eq : ETSMultiplicativeErrorMeasurement) \tag{6.20}$$

implying that

$$e_t = \frac{y_t - \hat{\mu}_t}{\hat{\mu}_t}.(\#eq : ETSMultiplicativeErrorFormula) \tag{6.21}$$

The estimates of scale can then be used in the next phase, when parameters are optimised via the maximisation of respective log-likelihood function. The maximum likelihood approach is in case of ADAM models is discussed in detail in Section @ref(ADAMETSEstimationLikelihood).

The distributional assumptions impact both the estimation of models and the prediction intervals. In the case of asymmetric distributions (such as log Normal, Gamma and Inverse Gaussian), the intervals will typically be asymmetric, with the upper bound being further away from the point forecast than the lower one. Furthermore, even with the comparable estimates of scales of distributions, Inverse Gaussian distribution will typically produce wider bounds than Log-Normal and Gamma. The width of intervals relates to the kurtosis of distributions, which is discussed in Chapter 3 of **?**.

6.6 Examples of application

6.6.1 Non-seasonal data

We continue our examples with the same Box-Jenkins sales data by fitting the ETS(M,M,N) model, but this time with a holdout of 10 observations:

```
adamModel <- adam(BJsales, "MMN", h=10, holdout=TRUE)
adamModel
```

```
## Time elapsed: 0.03 seconds
## Model estimated using adam() function: ETS(MMN)
```

```
## Distribution assumed in the model: Gamma
## Loss function type: likelihood; Loss function value: 245.3759
## Persistence vector g:
##  alpha   beta
## 1.0000 0.2412
##
## Sample size: 140
## Number of estimated parameters: 5
## Number of degrees of freedom: 135
## Information criteria:
##      AIC     AICc      BIC     BICc
## 500.7518 501.1996 515.4600 516.5664
##
## Forecast errors:
## ME: 3.217; MAE: 3.33; RMSE: 3.784
## sCE: 14.124%; Asymmetry: 91.6%; sMAE: 1.462%; sMSE: 0.028%
## MASE: 2.817; RMSSE: 2.482; rMAE: 0.925; rRMSE: 0.921
```

The output above is similar to the one we discussed in Section @ref(ADAMETSPureAdditiveExamples), so we can compare the two models using various criteria and select the most appropriate. Even though the default distribution for the multiplicative error models in ADAM is Γ, we can compare this model with the ETS(A,A,N) via information criteria. For example, here are the AICc for the two models:

```
# ETS(M,M,N)
AICc(adamModel)
```

```
## [1] 501.1996
```

```
# ETS(A,A,N)
AICc(adam(BJsales, "AAN", h=10, holdout=TRUE))
```

```
## [1] 497.2624
```

The comparison is fair because both models were estimated via likelihood, and both likelihoods are formulated correctly, without omitting any terms (e.g. the `ets()` function from the `forecast` package omits the $-\frac{T}{2}\log\left(2\pi e \frac{1}{T}\right)$ for convenience, which makes it incomparable with other models). In this example, the pure additive model is more suitable for the data than the pure multiplicative one.

Figure @ref(fig:BJSalesadamETSMMN) shows how the model fits the data and what forecast it produces. Note that the function produces the **point forecast** in this case, which is not equivalent to the conditional expectation! The point forecast undershoots the actual values in the holdout.

If we want to produce the forecasts (conditional expectation and prediction interval) from the model, we can do it, using the same command as in Section @ref(ADAMETSPureAdditiveExamples):

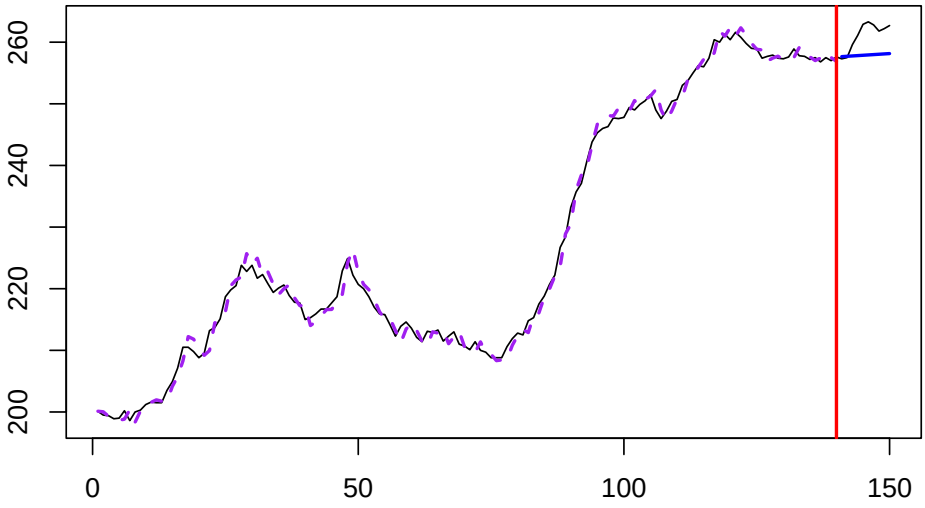


Figure 6.1: (#fig:BJSalesadamETSMMN)Model fit for Box-Jenkins Sales data from ETS(M,M,N).

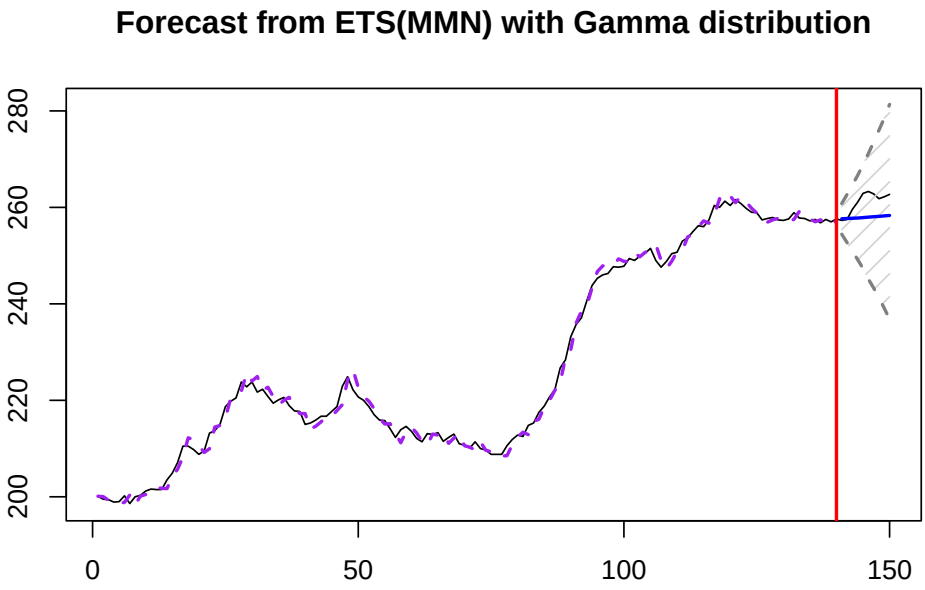


Figure 6.2: (#fig:BJSalesadamETSMMNForecast)Forecast for Box-Jenkins Sales data from ETS(M,M,N).

Note that, when we ask for "prediction" interval, the `forecast()` function will automatically decide what to use based on the estimated model: in case of pure additive one, it will use analytical solutions, while in the other cases, it will use simulations (see Section @ref(ADAMForecastingPI)). The point forecast obtained from the forecast function corresponds to the conditional expectation and is calculated based on the simulations. This also means that it will differ slightly from one run of the function to another (reflecting the uncertainty in the error term). Still, the difference, in general, should be negligible for a large number of simulation paths.

The forecast with prediction interval is shown in Figure @ref(fig:BJSalesadamETSMMNForecast). The conditional expectation is not very different from the point forecast in this example. This is because the variance of the error term is close to zero, thus bringing the two close to each other:

```
sigma(adamModel)^2
```

```
## [1] 3.928668e-05
```

We can also compare the performance of ETS(M,M,N) with Γ distribution and the conventional ETS(M,M,N), assuming normality:

```
adamModelNormal <- adam(BJsales, "MMN", h=10, holdout=TRUE,
                        distribution="dnorm")
adamModelNormal
```

```
## Time elapsed: 0.02 seconds
## Model estimated using adam() function: ETS(MMN)
## Distribution assumed in the model: Normal
## Loss function type: likelihood; Loss function value: 245.3872
## Persistence vector g:
## alpha  beta
## 1.000 0.241
##
## Sample size: 140
## Number of estimated parameters: 5
## Number of degrees of freedom: 135
## Information criteria:
##      AIC     AICc      BIC     BICc
## 500.7745 501.2222 515.4827 516.5890
##
## Forecast errors:
## ME: 3.217; MAE: 3.33; RMSE: 3.785
## sCE: 14.126%; Asymmetry: 91.6%; sMAE: 1.462%; sMSE: 0.028%
## MASE: 2.817; RMSSE: 2.483; rMAE: 0.925; rRMSE: 0.921
```

In this specific example, the two distributions produce very similar results with almost indistinguishable estimates of parameters.

6.6.2 Seasonal data

The `AirPassengers` data used in Section @ref(ADAMETSPureAdditiveExamples) has (as we discussed) multiplicative seasonality. So, the ETS(M,M,M) model might be more suitable than the pure additive one that we used previously:

```
adamModel <- adam(AirPassengers, "MMM", h=12, holdout=TRUE)
adamForecast <- forecast(adamModel, h=12, interval="prediction")
adamModel
```

```
## Time elapsed: 0.1 seconds
## Model estimated using adam() function: ETS(MMM)
## Distribution assumed in the model: Gamma
## Loss function type: likelihood; Loss function value: 468.5176
## Persistence vector g:
##  alpha   beta  gamma
## 0.7684 0.0206 0.0000
##
## Sample size: 132
## Number of estimated parameters: 17
## Number of degrees of freedom: 115
## Information criteria:
##       AIC      AICc       BIC      BICc
##  971.0351  976.4036 1020.0428 1033.1492
##
## Forecast errors:
## ME: -5.617; MAE: 15.496; RMSE: 21.938
## sCE: -25.677%; Asymmetry: -23.1%; sMAE: 5.903%; sMSE: 0.698%
## MASE: 0.643; RMSSE: 0.7; rMAE: 0.204; rRMSE: 0.213
```

Notice that the smoothing parameter $\gamma = 0$, which implies that we deal with the data with deterministic multiplicative seasonality. Comparing the information criteria (e.g. AICc) with the ETS(A,A,A) (discussed in Section @ref(ADAMETSPureAdditiveExamplesETSAAA)), the pure multiplicative model does a better job at fitting the data than the additive one:

```
adamModelAdditive <- adam(AirPassengers, "AAA", lags=12, h=12, holdout=TRUE)
AICc(adamModelAdditive)
```

```
## [1] 1130.756
```

The conditional expectation and prediction interval from this model are mroe adequate as well (Figure @ref(fig:AirPassengersMMMForecast)):

If we want to calculate the error measures based on the conditional expectation, we can use the `measures()` function from `greybox` package the following way:

```
measures(adamModel$holdout,adamForecast$mean,actuals(adamModel))
```

```
##           ME          MAE          MSE          MPE         MAPE
```

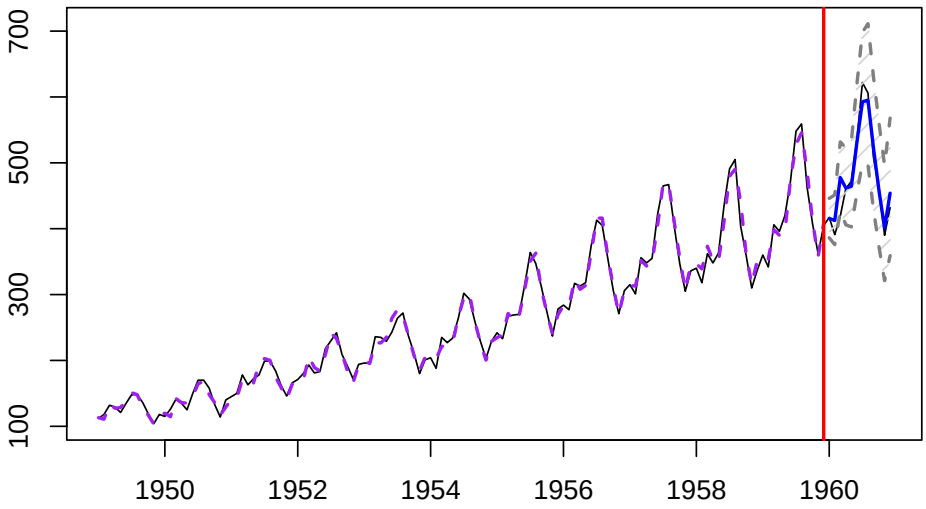


Figure 6.3: (#fig:AirPassengersMMMForecast)Forecast for air passengers data using ETS(M,M,M) model.

```
##  -5.906419957  15.435651011 480.764388422  -0.016920200   0.033700674
##           sCE          sMAE          sMSE          MASE         RMSSE
##  -0.270015562   0.058804177   0.006977482   0.640909757   0.699803714
##          rMAE          rRMSE          rAME     asymmetry          sPIS
##   0.203100671   0.212925597   0.082994191  -0.250913526   2.216357048
```

These can be compared with the measures from the ETS(A,A,A) model:

```
measures(adamModel$holdout,adamModelAdditive$forecast,actuals(adamModel))
```

```
##           ME          MAE          MSE          MPE         MAPE
##   28.36910729  37.11462699 2442.64739763   0.04881281   0.07053896
##           sCE          sMAE          sMSE          MASE         RMSSE
##    1.29691091   0.14139314   0.03545090   1.54105107   1.57739510
##          rMAE          rRMSE          rAME     asymmetry          sPIS
##    0.48835036   0.47994571   0.39862914   0.69383499  -7.31044007
```

Comparing, for example, MSE from the two models, we can conclude that the pure multiplicative model is more accurate than the pure additive one.

We can also produce the plot of the time series decomposition according to ETS(M,M,M) (see Figure @ref(fig:AirPassengersMMMDecomposition)).

The plot in Figure @ref(fig:AirPassengersMMMDecomposition) shows that the residuals are more random for the pure multiplicative model than for the ETS(A,A,A), but there still might be some structure left. The autocorrelation and partial autocorrelation functions (discussed in Section @ref(BJApproach))

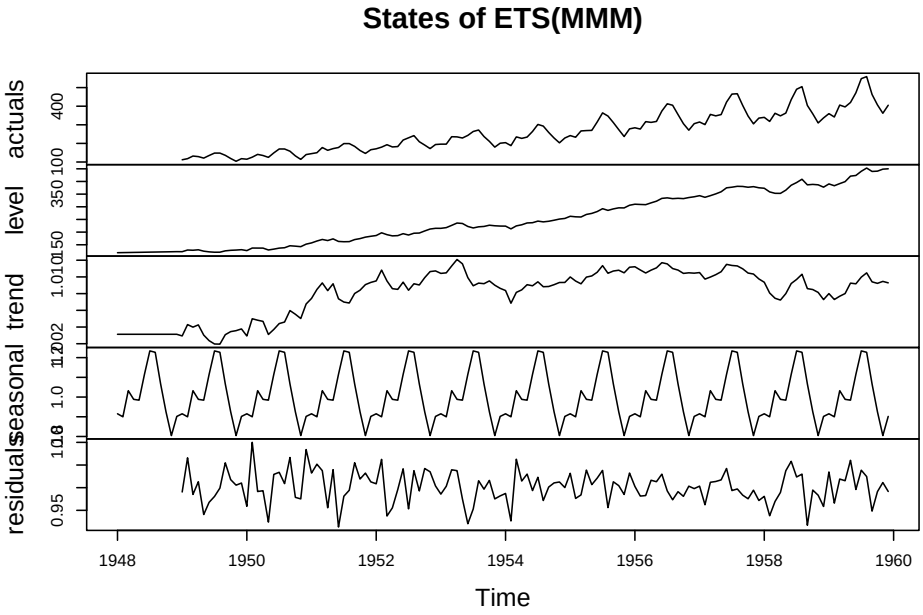


Figure 6.4: (#fig:AirPassengersMMMDecomposition)Decomposition of air passengers data using ETS(M,M,M) model.

might help in understanding this better:

```
par(mfcol=c(2,1), mar=c(2,4,2,1))
plot(adamModel,10:11)
```

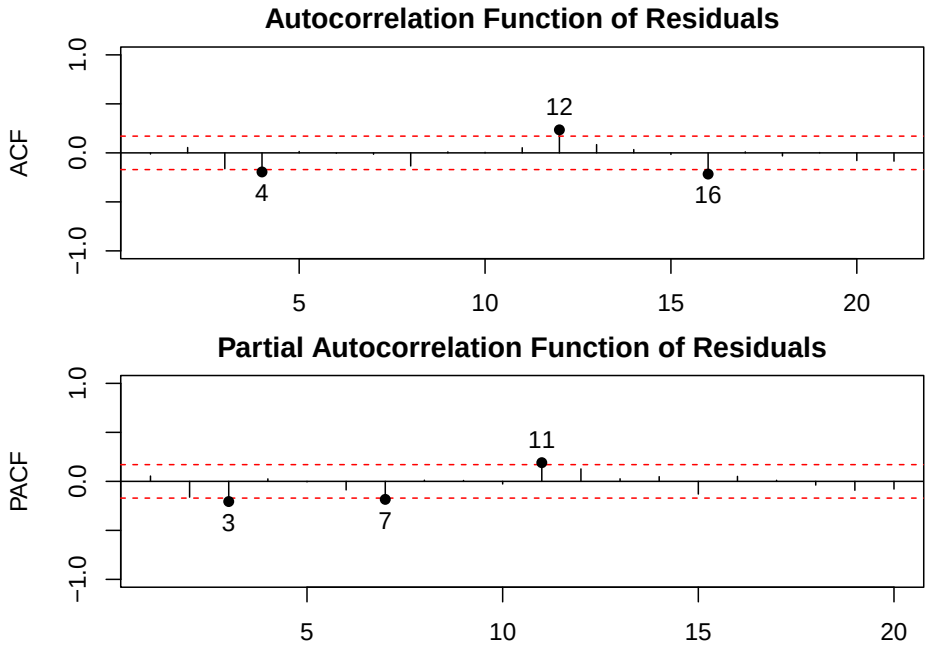


Figure 6.5: (#fig:AirPassengersMMMACFPACF)ACF and PACF of residuals of ETS(M,M,M) model.

The plot in Figure @ref(fig:AirPassengersMMMDecomposition) shows that there is still some correlation left in the residuals, which could be either due to pure randomness or imperfect estimation of the model. Tuning the parameters of the optimiser or selecting a different model might solve the problem.

Chapter 7

General ADAM ETS model

Now that we have discussed two special cases of ADAM ETS models (namely pure additive from Chapter @ref(ADAMETSPureAdditive) and pure multiplicative from Chapter @ref(ADAMETSPureMultiplicative) ADAM ETS), we can move to the discussion of the general model and special cases of it, including some of the mixed models. We will consider two important groups of mixed ADAM ETS models: with non-multiplicative and with multiplicative trends. They have very different properties that are worth mentioning. Finally, we will discuss the question of the normalisation of seasonal indices. This topic was studied in the literature back in the 80s when we still did not have a proper ETS model, but it has not been thoroughly discussed since then. In Section @ref(ADAMETSSeasonalNormalisation), we take a more critical view towards the topic.

7.1 Model formulation

Based on the discussion in previous chapters, we can summarise the general ADAM ETS model. It is built upon the conventional model discussed in Section @ref(ETSConventionalModel) but has several significant differences, the most important of which is that it is formulated using lags of components rather than the transition of them over time (this was discussed in Chapter @ref(ADAMETSPureAdditive) for the pure additive model). The general ADAM ETS model is formulated in the following way:

$$\begin{aligned} y_t =& w(\mathbf{v}_{t-\mathbf{l}}) + r(\mathbf{v}_{t-\mathbf{l}})\epsilon_t \\ \mathbf{v}_t =& f(\mathbf{v}_{t-\mathbf{l}}) + g(\mathbf{v}_{t-\mathbf{l}})\epsilon_t \end{aligned}, (\#eq:ETSADAMStateSpace) \tag{7.1}$$

where $\mathbf{v}_{t-\mathbf{l}}$ is the vector of lagged components and $\mathbf{l}$ is the vector of lags, while all the other functions correspond to the ones used in @ref(eq:ETSConventionalStateSpace). This model form is mainly useful

for the formulation, rather than for the derivations, as discussed in Section @ref(ETSConventionalModel). Not only it encompasses any pure models, it also allows formulating any of the mixed ones. For example, the ETS(M,A,M) will have the following values:

$$
\begin{aligned}
w(\mathbf{v}_{t-\mathbf{l}}) = (l_{t-1} + b_{t-1})s_{t-m},\ r(\mathbf{v}_{t-\mathbf{l}}) = w(\mathbf{v}_{t-\mathbf{l}}),\\
f(\mathbf{v}_{t-\mathbf{l}}) = \begin{pmatrix} l_{t-1} + b_{t-1} \\ b_{t-1} \\ s_{t-m} \end{pmatrix},\ g(\mathbf{v}_{t-\mathbf{l}}) = \begin{pmatrix} \alpha(l_{t-1} + b_{t-1}) \\ \beta(l_{t-1} + b_{t-1}) \\ \gamma s_{t-m} \end{pmatrix},\\
\mathbf{v}_t = \begin{pmatrix} l_t \\ b_t \\ s_t \end{pmatrix},\ \mathbf{l} = \begin{pmatrix} 1 \\ 1 \\ m \end{pmatrix},\\
\mathbf{v}_{t-\mathbf{l}} = \begin{pmatrix} l_{t-1} \\ b_{t-1} \\ s_{t-m} \end{pmatrix}
\end{aligned}
.
$$

By inserting these values in @ref(eq:ETSADAMStateSpace) we will get the classical ETS(M,A,M) model, mentioned in Section @ref(ETSTaxonomyMaths):

$$
\begin{aligned}
y_t =&(l_{t-1} + b_{t-1})s_{t-m}(1 + \epsilon_t)\\
l_t =&(l_{t-1} + b_{t-1})(1 + \alpha\epsilon_t)\\
b_t =&b_{t-1} + (l_{t-1} + b_{t-1})\beta\epsilon_t\\
s_t =&s_{t-m}(1 + \gamma\epsilon_t)
\end{aligned}
.(\#eq:ETSADAMMAM) \tag{7.2}
$$

The model @ref(eq:ETSADAMStateSpace) with different values for the functions is the basis of `adam()` function from `smooth` package.

7.2 Mixed ADAM ETS models

? proposed five classes of ETS models, based on the types of their components:

1. ANN; AAN; AAdN; ANA; AAA; AAdA;
2. MNN; MAN; MAdN; MNA; MAA; MAdA;
3. MNM; MAM; MAdM;
4. MMN; MMdN; MMM; MMdM;
5. ANM; AAM; AAdM; MMA; MMdA; AMN; AMdN; AMA; AMdA; AMM; AMdM

The idea behind this split is to distinguish the models by their complexity and the availability of analytical expressions for conditional moments. Class 1 models were discussed in Chapter @ref(ADAMETSPureAdditive). They have analytical expressions for conditional mean and variance; they can be applied to any data; they have simple formulae for prediction intervals.

? demonstrate that models from Class 2 have closed forms for conditional expectation and variance, with the former corresponding to the point forecasts.

However, the conditional distribution from these models is not Gaussian, so there are no formulae for the prediction intervals from these models. Yes, in some cases, Normal distribution might be used as a satisfactory approximation for the real one, but simulations should generally be preferred.

Class 3 models suffer from similar issues, but the situation worsens: there are no analytical solutions for the conditional mean and variance, and there are only approximations to these statistics.

Class 4 models were discussed in Chapter @ref(ADAMETSPureMultiplicative). They do not have analytical expressions for the moments, their conditional h steps ahead distributions represent a complex convolution of products of the basic ones, but they are appropriate for the positive data and become more valuable when the level of series is low, as already discussed in Chapter @ref(ADAMETSPureMultiplicative).

Finally, Class 5 models might have infinite variances, specifically on long horizons and when the data has low values. Indeed, when the level in one of these models becomes close to zero, there is an increased chance of breaking the model due to the appearance of negative values. Consider an example of the ETS(A,A,M) model, which might have a negative trend, leading to negative values, which are then multiplied by the positive seasonal indices. This would lead to unreasonable values of states in the model. That is why in practice, these models should only be used when the level of the series is high. Furthermore, some Class 5 models are very difficult to estimate and are very sensitive to the smoothing parameter values.

The `ets()` function from the `forecast` package supports only Classes 1 – 4 for the reasons explained above.

To be fair, any mixed model can potentially break when the level of the series is close to zero. For example, ETS(M,A,N) can have a negative trend, which might lead to the negative level and, as a result, to the multiplication of pure positive error term by the negative components. Estimating such a model on real data becomes a non-trivial task.

In addition, as discussed above, simulations are typically needed to get prediction intervals for models of Classes 2 – 5 and conditional mean and variance for models for Classes 4 – 5. All of this, in my opinion, means that the more useful classification of ETS models is the following (this classification was first proposed by **?**):

A. Pure additive models (Chapter @ref(ADAMETSPureAdditive)): ANN; AAN; AAdN; ANA; AAA; AAdA; B. Pure multiplicative models (Chapter @ref(ADAMETSPureMultiplicative)): MNN; MMN; MMdN; MNM; MMM; MMdM; C. Mixed models with non-multiplicative trend (Section @ref(ADAMETSMixedModelsGroup3)): MAN; MAdN; MNA; MAA; MAdA; MAM; MAdM; ANM; AAM; AAdM; D. Mixed models with multiplicative trend (Section @ref(ADAMETSMixedModelsGroup4): MMA; MMdA; AMN;

AMdN; AMA; AMdA; AMM; AMdM;

The main idea behind the split to (C) and (D) is that the multiplicative trend makes it almost impossible to derive the formulae for the conditional moments of the distribution. So this class of models can be considered as the most challenging one.

`adam()` function supports all 30 ETS models, but you should keep in mind the limitations of some of them discussed in this section.

7.3 Mixed models with non-multiplicative trend

There are two subclasses in this class of models: one with a non-multiplicative seasonal component (MAN, MAdN, MNA, MAA, MAdA) and another one with the multiplicative one (MAM; MAdM; ANM; AAM; AAdM). The conditional mean for the former models coincides with the point forecasts, while the conditional variance can be calculated using the following recursive formula (**?**, page 84):

$$\text{V}(y_{t+h}|t) = (1+\sigma^2)\xi_h - \mu^2_{t+h|t}$$
$$\text{where } \xi_1 = \mu^2_{t+1|t} \text{ and } \xi_h = \mu^2_{t+h|t} + \sigma^2 \sum_{j=1}^{h-1} c_j^2 \xi_{h-j}, (\#eq:ETSADAMMixedModels31Variance) \tag{7.3}$$

where σ^2 is the variance of the error term. As for the second subgroup, the conditional mean corresponds to the point forecasts, when the forecast horizon is less than or equal to the seasonal frequency of the component (i.e. $h \leq m$), and there is a cumbersome formula for calculating the conditional mean to some of models in this subgroup for the $h > m$. When it comes to the conditional variance, there exists a formula for some of models in the second subgroup, but they are cumbersome as well. The interested reader is directed to **?**, page 85.

When it comes to the parameters bounds for the models in this group, the first subgroup of models should have similar bounds to the ones for the respective additive error (Section @ref(stabilityConditionAdditiveError)) models because they both underlie the same exponential smoothing methods, but with additional restrictions, coming from the multiplicative error (Section @ref(stabilityConditionMultiplicativeError)).

1. The *traditional bounds* (aka "usual") should work fine for these models for the same reasons they work for the pure additive ones, although they might be too restrictive in some cases;
2. The *admissible bounds* for smoothing parameters for the models in this group might be too wide and violate the condition of $(1+\alpha\epsilon_t) > 0, (1+\beta\epsilon_t) > 0, (1+\gamma\epsilon_t) > 0$, which is important in order not to break the models.

The second subgroup is more challenging in terms of parameters bounds because of the multiplication of states by the seasonal components.

7.4 Mixed models with multiplicative trend

This is the most challenging class of models (MMA; MMdA; AMN; AMdN; AMA; AMdA; AMM; AMdM). These do not have analytical formulae for conditional moments, and they are very sensitive to smoothing parameter values and may lead to explosive forecasting trajectories. So, to obtain conditional expectation, variance and prediction interval from the models of these classes, simulations should be used.

One of the issues encountered when using these models is the explosive trajectories because of the multiplicative trend. As a result, when these models are estimated, it makes sense to set the initial value of the trend to 1 or a lower value so that the optimiser does not encounter difficulties in the calculations.

Furthermore, the combination of components for the models in this class makes even less sense than the combination for Class C (discussed in Section **??**ADAMETSMixedModelsGroup3)). For example, the multiplicative trend assumes either explosive growth or decay, as shown in Figure @ref(fig:plotsOfExponent).

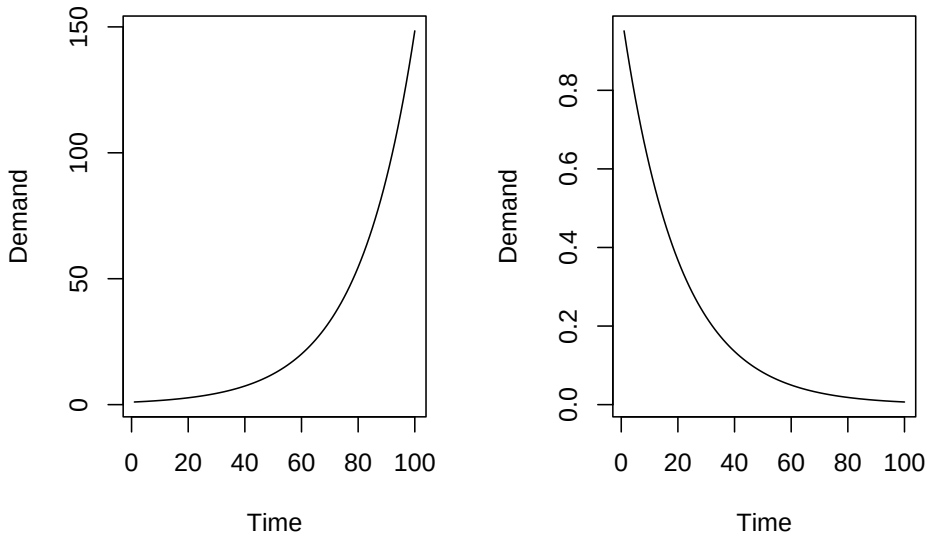


Figure 7.1: (#fig:plotsOfExponent)Plots of exponential increase and exponential decline.

However, assuming that either seasonal component, the error term, or both will have precisely the same impact on the final value irrespective of the point

of time is unrealistic for this situation. The more reasonable would be for the amplitude of seasonality to decrease together with the exponential decay of the trend and for the variance of the error term to do the same. But this means that we are talking about the pure multiplicative models (Chapter @ref(ADAMETSPureMultiplicative)), not the mixed ones. There is only one situation where such mixed models could make sense: when the speed of change of the exponential trajectory is close to zero, meaning that the level of the series does not change rapidly and when the volume of the data is high. In this case, the mixed models might perform well and even produce more accurate forecasts than the models from the other classes.

When it comes to the bounds of the parameters, this is a mystery for the mixed models of this class. This is because the recursive relations are complicated, and calculating the discount matrix or anything like that becomes challenging. The usual bounds should still be okay, but keep in mind that these mixed models are typically not very stable. So from my experience, the smoothing parameters should be as low as possible, assuming that the initial values guarantee a working model (not breaking at some of the observations).

7.5 Normalisation of seasonal indices in ETS models

One of the ideas arising from time series decomposition (Section @ref(ClassicalDecomposition)), inherited by the conventional ETS, is the renormalisation of seasonal indices. It comes to one of the two principles, depending on the type of seasonality:

1. If the model has additive seasonality, then the seasonal indices should add up to zero in a specific period of time, e.g. monthly indices should add up to 0 over the yearly period;
2. If the model has multiplicative seasonality, then the geometric mean of seasonal indices over a specific period should be equal to one.

Condition (2) in the conventional ETS is substituted by "the arithmetic mean of multiplicative indices should be equal to one", which does not have reasonable grounds behind it. If we deal with the multiplicative effect, the geometric mean should be used, not the arithmetic one. Otherwise, we introduce bias in the model by multiplying components by indices. While the normalisation is a natural element of the time series decomposition and works fine for the initial seasonal indices, renormalising the seasonal indices over time might not be natural for the ETS.

? discuss different mechanisms for the renormalisation of seasonal indices, which, as the authors claim, are needed to make the principles (1) and (2) hold from period to period in the data. However, I argue that this is an unnatural idea for the ETS models. The indices should only be normalised during the initialisation of the model (at the moment $t = 0$), and that they should vary independently

for the rest of the sample. The rationale for this comes from the model itself. To illustrate it, I will use ETS(A,N,A), but the idea can be easily applied to any other ETS model with any types of components and any number of seasonal frequencies. Just a reminder, this model is formulated as:

$$\begin{aligned} y_t =& l_{t-1} + s_{t-m} + \epsilon_t \\ l_t =& l_{t-1} + \alpha\epsilon_t \\ s_t =& s_{t-m} + \gamma\epsilon_t \end{aligned} .(\#eq:ETSANAExampleNormalisation) \tag{7.4}$$

Let's assume that this is the "true model" (as discussed in Section @ref(modelsMethods)) for whatever data we have for whatever reason. In this case, the set of equations @ref(eq:ETSANAExampleNormalisation) tells us that the seasonal indices change over time, depending on the smoothing parameter γ values and each specific value of ϵ_t, which is assumed to be i.i.d. All seasonal indices s_{t+i} in a particular period (e.g. monthly indices in a year) can be written down explicitly based on @ref(eq:ETSANAExampleNormalisation):

$$\begin{aligned} s_{t+1} =& s_{t+1-m} + \gamma\epsilon_{t+1} \\ s_{t+2} =& s_{t+2-m} + \gamma\epsilon_{t+2} \\ \vdots& \\ s_{t+m} =& s_t + \gamma\epsilon_{t+m} \end{aligned} .(\#eq:ETSANAExampleNormalisationIndices01) \tag{7.5}$$

If this is how the data is "generated" and the seasonality evolves over time, then there is only one possibility, for the indices $s_{t+1}, s_{t+2}, \dots, s_{t+m}$ to add up to zero:

$$s_{t+1}+s_{t+2}+\dots+s_{t+m} = 0 (\#eq:ETSANAExampleNormalisationIndices02a) \tag{7.6}$$

or after inserting in @ref(eq:ETSANAExampleNormalisationIndices01):

$$s_{t+1-m}+s_{t+2-m}+\dots+s_t+\gamma\left(\epsilon_{t+1}+\epsilon_{t+2}+\dots+\epsilon_{t+m}\right) = 0 (\#eq:ETSANAExampleNormalisationIndices02b) \tag{7.7}$$

meaning that:

a. the previous indices $s_{t+1-m}, s_{t+2-m}, \dots, s_t$ add up to zero **and**
b. either $\gamma = 0$,
c. or the sum of error terms $\epsilon_{t+1}, \epsilon_{t+2}, \dots, \epsilon_{t+m}$ is zero.

Note that we do not consider the situation $s_{t+1-m}+\dots+s_t = -\gamma\left(\epsilon_{t+1}+\dots+\epsilon_{t+m}\right)$ as it does not make sense. The condition (a) can be considered reasonable if the previous indices are normalised. (b) means that the seasonal indices do not evolve over time. However, (c) implies that the error term is not independent, because $\epsilon_{t+m} = -\epsilon_{t+1} - \epsilon_{t+2} - \dots - \epsilon_{t+m-1}$, which violates one of the basic assumptions of the model from Section @ref(assumptions), meaning that @ref(eq:ETSANAExampleNormalisationIndices01) cannot be considered as the "true" model any more, as it omits some important elements. Thus renormalisation is unnatural for ETS from the "true" model point of view.

Alternatively each seasonal index could be updated on each observation t (to make sure that the indices are renormalised). In this situation we have:

$$\begin{aligned} & s_t = s_{t-m} + \gamma \epsilon_t \\ & s_{t-m+1} + s_{t-m+2} + \dots + s_{t-1} + s_t = 0 \end{aligned}',$$

which can be rewritten as $s_{t-m} + \gamma \epsilon_t = -s_{t+1-m} - s_{t+2-m} - \dots - s_{t-1}$, meaning that:

$$s_{t-m} + s_{t+1-m} + s_{t+2-m} + \dots + s_{t-1} = -\gamma \epsilon_t,$$

But due to the renormalisation, the sum on the left-hand side should be equal to zero, implying that either $\gamma = 0$ or $\epsilon_t = 0$. While the former might hold in some cases (deterministic seasonality), the latter cannot hold for all $t = 1, \dots, T$ and violates the model's assumptions. The renormalisation is thus impossible without changing the structure of the model. **?** acknowledge that and propose in Chapter 8 some modifications for the seasonal ETS model (i.e. introducing new models), which we do not aim to discuss in this Chapter.

The discussion in this section demonstrates that the renormalisation of seasonal indices is unnatural for the ETS model and should not be used. This does not imply that the initial seasonal indices (corresponding to the observation $t = 0$) cannot be normalised. On the contrary, this is the desired approach as it reduces the number of estimated parameters in the model.

Chapter 8

Conventional ARIMA

Another important dynamic element in ADAM is the ARIMA model (developed originally by **?**). ARIMA stands for "AutoRegressive Integrated Moving Average", although the name does not tell much on its own and needs additional explanation, which will be provided in this chapter.

The main idea of the model is that the data might have dynamic relations over time, where the new values depend on the values on the previous observations. This becomes more obvious in the case of engineering systems and modelling physical processes. For example, **?** use an example of a series of CO_2 output of a furnace when the input gas rate changes. In this case, the elements of the ARIMA process are natural, as the CO_2 cannot just drop to zero when the gas is switched off – it will leave the furnace, in reducing quantity over time (i.e. leaving $\phi_1 \times 100\%$ of CO_2 in the next minute, where ϕ_1 is a parameter in the model).

Another example where AR processes are natural is the temperature in the room, measured with 5 minutes intervals. In this case, the temperature at 5:30 pm will depend on the one at 5:25 pm (if the temperature outside the room is lower, then the one in the room will go down slightly due to the loss of heat). So, in these examples, the ARIMA model can be considered a "true model" (see discussion in Section @ref(modelsMethods)). Unfortunately, when it comes to time series from the social or business domains, it becomes very difficult to motivate ARIMA usage from the modelling point of view. For example, the demand on products does not reproduce itself and in real life does not depend on the demand on previous observations unless we are talking about repetitive purchases by the same group of consumers. So, if we construct ARIMA for such a process, we turn a blind eye to the fact that the observed time series relations in the data are most probably spurious. At best, in this case, ARIMA can be considered a very crude approximation of a complex process (demand is typically influenced by price changes, consumer behaviour, and promotional activities).

Thus, whenever we work with ARIMA models in social or business domains, we should keep in mind that they are wrong even from the philosophical point of view. Nevertheless, they still can be useful, which is why we discuss them in this chapter. We focus our discussion on forecasting with ARIMA. A reader interested in time series analysis is directed to **?**.

This chapter will discuss the main theoretical properties of ARIMA processes (i.e. what would happen if the data indeed followed the specified model), moving to more practical aspects in the next chapter. We start the discussion with the non-seasonal ARIMA models, explaining how the forecasts from those models would look like, then move to the seasonal and multi-seasonal ARIMA, then discuss the classical Box-Jenkins approach for ARIMA order selection and its limitations. Finally, we explain the connection between ARIMA and ETS models.

8.1 Introduction to ARIMA

ARIMA contains several elements:

1. AR(p) – the AutoRegressive part, showing how the variable is impacted by its values on the previous observations. It contains p lags. For example, the quantity of the liquid in a vessel with an opened tap on some observation will depend on the quantity on the previous steps;
2. I(d) – the number of differences d taken in the model (I stands for "Integrated"). Working with differences rather than with the original data means that we deal with changes and rates of changes, not with just values. Technically, differences are needed to make data stationary (i.e. with fixed expectation and variance, although there are different definitions of the term *stationarity*, see below);
3. MA(q) – the Moving Average component, explaining how the previous white noise impacts the variable. It contains q lags. Once again, in technical systems, the idea that random error can affect the value has a relatively simple explanation. For example, when the liquid drips out of a vessel, we might not be able to predict the air fluctuations, which would impact the flow and could be perceived as elements of random noise. This randomness might, in turn, influence the quantity of liquid in a vessel on the following observation, thus introducing the MA elements in the model.

I intentionally do not provide ARIMA examples from the demand forecasting area, as these are much more difficult to motivate and explain than the examples from the more technical areas.

Before continuing our discussion, we should define the term **stationarity**. There are two definitions in the literature: one refers to "strict stationarity", while the other refers to the "weak stationarity":

- Time series is said to be **weak stationary** when its conditional expecta-

tion and variance are constant, and the variance is finite on all observations;
- Time series is said to be **strong stationary** when its unconditional joint probability distribution does not change over time. This automatically implies that all its moments are constant (i.e. the process is also weak stationary).

The stationarity is essential in the ARIMA context and plays an important role in regression analysis. If the series is not stationary, it might be challenging to estimate its moments correctly using conventional methods. In some cases, it might be impossible to get the correct parameters (e.g. there is an infinite combination of parameters that would produce the minimum of the selected loss function). To avoid this issue, the series is transformed to ensure that the moments are finite and constant (e.g. take logarithms or do Box-Cox transform, take differences or detrend the series). After that, the model becomes easier to identify. In contrast with ARIMA, the ETS models are almost always non-stationary and do not require the series to be stationary. We will see the connection between the two approaches later in this chapter.

8.1.1 AR(p)

We start with a simple AR(1) model, which is written as:

$$y_t = \phi_1 y_{t-1} + \epsilon_t, (\#eq:ARIMA100Example) \tag{8.1}$$

where ϕ_1 is the parameter of the model. This formula tells us that the value on the previous observation is carried out to the new one in the proportion of ϕ_1. Typically, the parameter ϕ_1 is restricted with the region (-1, 1) to make the model stationary, but very often in real life, ϕ_1 lies in the (0, 1) region. If the parameter is equal to 1, the model becomes equivalent to Random Walk (Section @ref(Naive)).

The forecast trajectory (conditional expectation several steps ahead) of this model would typically correspond to the exponentially declining curve. Here is a simple example in R of a very basic forecast from AR(1) with $\phi_1 = 0.9$ (see Figure @ref(fig:AR1Trajectory)):

```
y <- vector("numeric", 20)
y[1] <- 100
phi <- 0.9
for(i in 2:length(y)){
    y[i] <- phi * y[i-1]
}
plot(y, type="l", xlab="horizon", ylab="Forecast")
```

If, for some reason, we get $\phi_1 > 1$, then the trajectory corresponds to an exponential increase, becoming explosive, implying non-stationary behaviour. The model, in this case, becomes very difficult to work with, even if the parameter

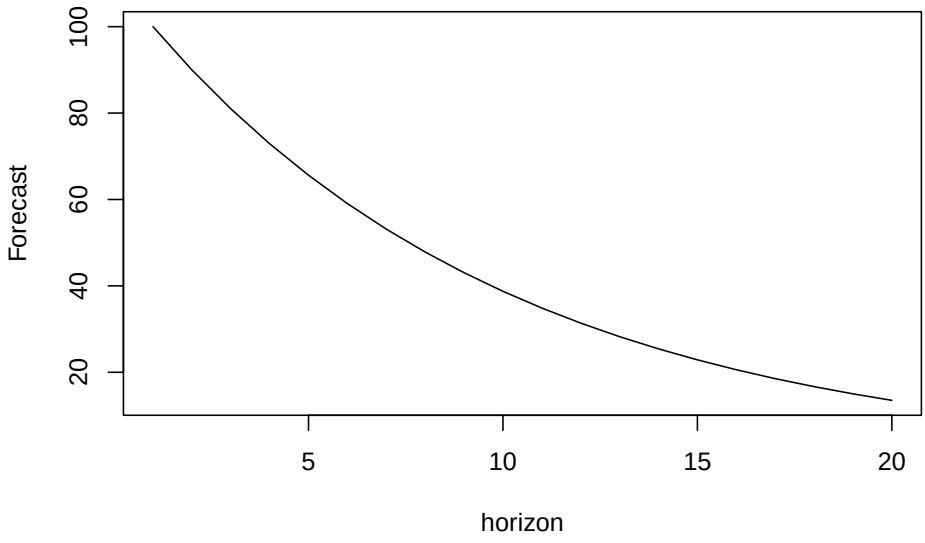


Figure 8.1: (#fig:AR1Trajectory)Forecast trajectory for AR(1) with $\phi_1 = 0.9$.

is close to one. So it is typically advised to restrict the parameter with the stationarity region (we will discuss this in more detail later in this chapter).

In general, it is possible to imagine the situation, when the value at the moment of time t would depend on several previous values, so the model AR(p) would be needed:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t, (\#eq:ARIMAp00Example) \quad (8.2)$$

where ϕ_i is the parameter for the i-th lag of the model. So, the model assumes that the data on the recent observation is influenced by the p previous observations. The more lags we introduce in the model, the more complicated the forecasting trajectory becomes, potentially demonstrating harmonic behaviour. Here is an example of a point forecast from AR(3) model $y_t = 0.9y_{t-1} - 0.7y_{t-2} + 0.6y_{t-3} + \epsilon_t$ (Figure @ref(fig:AR3Trajectory)):

```
y <- vector("numeric", 30)
y[1:3] <- c(100, 75, 30)
phi <- c(0.9,-0.7,0.6)
for(i in 4:30){
    y[i] <- phi[1] * y[i-1] + phi[2] * y[i-2] + phi[3] * y[i-3]
}
plot(y, type="l", xlab="horizon", ylab="Forecast")
```

No matter what the forecast trajectory of the AR model is, it will asymptotically converge to zero as long as the model is stationary.

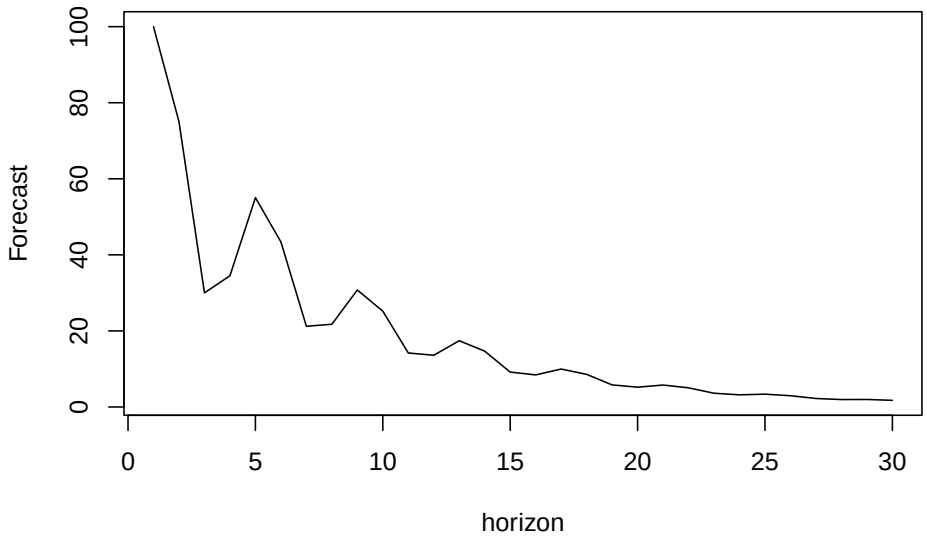


Figure 8.2: (#fig:AR3Trajectory)Forecast trajectory for an AR(3) model.

8.1.2 MA(q)

Before discussing the "Moving Averages" model, we should acknowledge that the name is quite misleading and that the model has *nothing to do* with Centred Moving Averages used in time series decomposition (Section @ref(ClassicalDecomposition)) or Simple Moving Averages (discussed in Section @ref(SMA)). The idea of the simplest MA(1) model can be summarised in the following mathematical way:

$$y_t = \theta_1 \epsilon_{t-1} + \epsilon_t, (\#eq:ARIMA001Example) \tag{8.3}$$

where θ_1 is the parameter of the model, typically lying between (-1, 1), showing what part of the error is carried out to the next observation. Because of the conventional assumption that the error term has a zero mean ($\mathrm{E}(\epsilon_t) = 0$), the forecast trajectory of this model is just a straight line coinciding with zero starting from the $h = 2$. But for the one step ahead forecast we have:

$$\mathrm{E}(y_{t+1}|t) = \theta_1 \mathrm{E}(\epsilon_t|t) + \mathrm{E}(\epsilon_{t+1}|t) = \theta_1 \epsilon_t. (\#eq:ARIMA001ExampleForecast) \tag{8.4}$$

Starting from $h = 2$ there are no observable error terms ϵ_t, so all the values past that are equal to zero:

$$\mathrm{E}(y_{t+2}|t) = \theta_1 \mathrm{E}(\epsilon_{t+1}|t) + \mathrm{E}(\epsilon_{t+2}|t) = 0. (\#eq:ARIMA001ExampleForecastInSample) \tag{8.5}$$

So, the forecast trajectory for MA(1) model converges to zero, when $h > 1$.

More generally, MA(q) model is written as:

$$y_t = \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q} + \epsilon_t, (\#eq : ARIMA00qExample) \quad (8.6)$$

where θ_i is the parameters for the i-th lag of the error term, typically restricted with the so-called invertibility region (discussed in the next section). In this case, the model implies that the recent observation is influenced by several errors on previous observations (your mistakes in the past will haunt you in the future). The more lags we introduce, the more complicated the model becomes. As for the forecast trajectory, it will reach zero when $h > q$.

8.1.3 ARMA(p,q)

Connecting the models @ref(eq:ARIMAp00Example) and @ref(eq:ARIMA00qExample), we get the more complicated model, ARMA(p,q):

$$y_t = \phi_1 y_{t-1}+\phi_2 y_{t-2}+\cdots+\phi_p y_{t-p}+\theta_1\epsilon_{t-1}+\theta_2\epsilon_{t-2}+\cdots+\theta_q\epsilon_{t-q}+\epsilon_t, (\#eq : ARIMAp0q) \quad (8.7)$$

which has the properties of the two models discussed above. The forecast trajectory from this model will have a combination of trajectories for AR and MA for $h \leq q$ and then will correspond to AR(p) for $h > q$.

To simplify the work with ARMA models, the equation @ref(eq:ARIMAp0q) is typically rewritten by moving all terms with y_t to the left-hand side:

$$y_t-\phi_1 y_{t-1}-\phi_2 y_{t-2}-\cdots-\phi_p y_{t-p} = \theta_1\epsilon_{t-1}+\theta_2\epsilon_{t-2}+\cdots+\theta_q\epsilon_{t-q}+\epsilon_t.(\#eq : ARIMAp0qLeft) \quad (8.8)$$

Furthermore, in order to make this even more compact, the backshift operator B can be introduced. It just shows by how much the subscript of the variable is shifted back in time:

$$y_t B^i = y_{t-i}.(\#eq : backshiftOperator) \quad (8.9)$$

Using @ref(eq:backshiftOperator), the ARMA model can be written as:

$$y_t(1-\phi_1 B-\phi_2 B^2-\cdots-\phi_p B^p) = \epsilon_t(1+\theta_1 B+\theta_2 B^2+\cdots+\theta_q B^q).(\#eq : ARIMAp0qCompacter) \quad (8.10)$$

Finally, we can also introduce the AR and MA polynomial functions to make the model even more compact:

$$\begin{aligned} \varphi^p(B) &= 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p \\ \vartheta^q(B) &= 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q. \end{aligned} (\#eq : ARMAPolynomials) \quad (8.11)$$

Inserting the functions @ref(eq:ARMAPolynomials) in @ref(eq:ARIMAp0qCompacter) leads to the compact presentation of ARMA model:

$$y_t\varphi^p(B) = \epsilon_t\vartheta^q(B).(\#eq : ARIMAp0qCompact) \quad (8.12)$$

The model @ref(eq:ARIMAp0qCompact) can be considered a compact form of @ref(eq:ARIMAp0q). It is more difficult to understand and interpret but easier

to work with from a mathematical point of view. In addition, this form permits introducing additional elements, which will be discussed later in this chapter.

Coming back to the ARMA model @ref(eq:ARIMAp0q), as discussed earlier, it assumes convergence of forecast trajectory to zero, the speed of which is regulated via the parameters. This implies that the data has the mean of zero, and ARMA should be applied to somehow pre-processed data so that it is stationary and varies around zero. This means that if you work with non-stationary and/or with non-zero mean data, the pure AR/MA or ARMA will be inappropriate – some prior transformations are in order.

8.1.4 ARMA with constant

One of the simpler ways to deal with the issue with zero forecasts is to introduce the constant (or intercept) in ARMA:

$$y_t = a_0+\phi_1 y_{t-1}+\phi_2 y_{t-2}+\cdots+\phi_p y_{t-p}+\theta_1\epsilon_{t-1}+\theta_2\epsilon_{t-2}+\cdots+\theta_p\epsilon_{t-p}+\epsilon_t (\#eq:ARIMAp0qExample) \tag{8.13}$$

or

$$y_t\varphi^p(B) = a_0 + \epsilon_t\vartheta^q(B), (\#eq:ARIMAp0qCompactConstant) \tag{8.14}$$

where a_0 is the constant parameter, which in this case also works as the unconditional mean of the series. The forecast trajectory in this case would converge to a_0 instead of zero, but with some minor distinctions from the ARMA without constant. For example, in case of ARMA(1,1) with constant we will have:

$$y_t = a_0 + \phi_1 y_{t-1} + \theta_1\epsilon_{t-1} + \epsilon_t. (\#eq:ARIMA101ConstExample01) \tag{8.15}$$

The conditional expectation of y_{t+h} for $h = 1$ and $h = 2$ can be written as (based on the discussions in previous sections):

$$\begin{matrix}\mathrm{E}(y_{t+1}|t) = a_0 + \phi_1 y_t + \theta_1\epsilon_t \\ \mathrm{E}(y_{t+2}|t) = a_0 + \phi_1\mathrm{E}(y_{t+1}|t) = a_0 + \phi_1 a_0 + \phi_1^2 y_t + \phi_1\theta_1\epsilon_t\end{matrix}, (\#eq:ARIMA101ConstExampleForecasth1) \tag{8.16}$$

or in general for the horizon h:

$$\mathrm{E}(y_{t+h}|t) = \sum_{j=1}^{h} a_0\phi_1^{j-1}+\phi_1^h y_t+\phi_1^{h-1}\theta_1\epsilon_t. (\#eq:ARIMA101ConstExampleForecast) \tag{8.17}$$

So, the forecast trajectory from this model dampens out, similar to the ETS(A,Ad,N) model (Subsection @ref(ETSAAdN)), and the rate of dampening is regulated by the value of ϕ_1. The following simple example demonstrates this point (see Figure @ref(fig:ARMAConstantTrajectory); I drop the MA(1) part because it does not change the shape of the curve):

```
y <- vector("numeric", 20)
y[1] <- 100
phi <- 0.9
for(i in 2:length(y)){
    y[i] <- 100 + phi * y[i-1]
}
plot(y, type="l", xlab="horizon", ylab="Forecast")
```

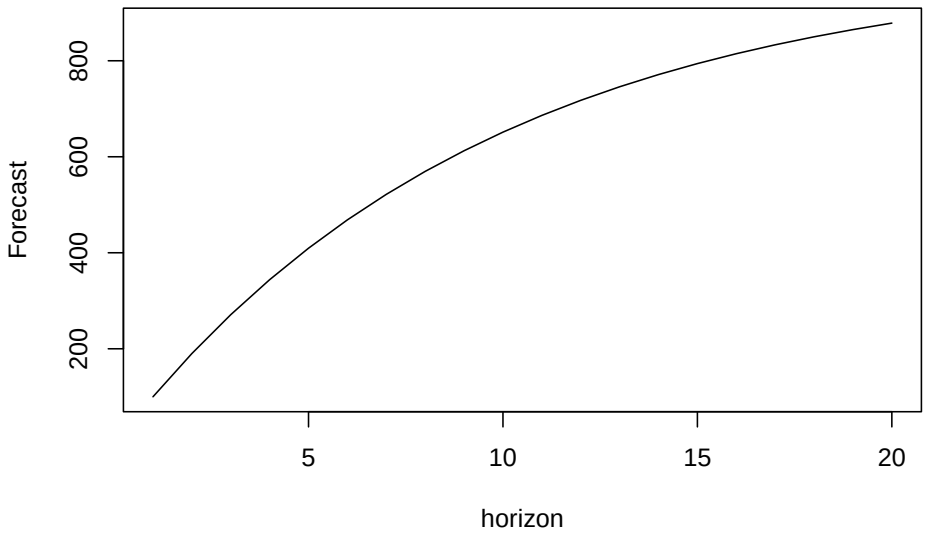


Figure 8.3: (#fig:ARMAConstantTrajectory)Forecast trajectory for an ARMA(1,1) model with constant.

The more complicated ARMA(p,q) models with p>1 will have more complex trajectories with potential harmonics, but the idea of dampening in AR(p) part of the model stays.

Finally, as alternative to adding a_0, each actual value of y_t can be centred via $y'_t = y_t - \bar{y}$, making sure that the mean of y'_t is zero and ARMA can be applied to the y'_t data instead of y_t. However, this approach introduces additional steps, but the result on stationary data is typically the same as adding the constant.

8.1.5 I(d)

Based on the previous discussion, we can conclude that ARMA cannot be applied to non-stationary data. So, if we deal with one, we need to make it stationary somehow. The conventional way of doing that is by taking differences in the data. The logic behind this is straightforward: if the data is not stationary, the mean somehow changes over time. This can be, for example, due

to a trend in the data. In this case, we should be talking about the change of variable y_t rather than the variable itself. So we should work on the following data instead:

$$\Delta y_t = y_t - y_{t-1} = y_t(1-B), (\#eq:ARIMADifferencesFirst) \tag{8.18}$$

if the differences have constant mean. The simplest model with differences is I(1), which is also known as the **Random walk** (see Section @ref(Naive)):

$$\Delta y_t = \epsilon_t, (\#eq:ARIMARandomWalk) \tag{8.19}$$

which can be reformulated in a simpler, more interpretable form by inserting @ref(eq:ARIMADifferencesFirst) in @ref(eq:ARIMARandomWalk) and regrouping elements:

$$y_t = y_{t-1} + \epsilon_t. (\#eq:ARIMARandomWalk02) \tag{8.20}$$

The model @ref(eq:ARIMARandomWalk02) can also be perceived as AR(1) with $\phi_1 = 1$. This is a non-stationary model, meaning that the unconditional mean of y_t is not constant. The forecast from this model corresponds to the Naïve method (see Section @ref(Naive)) with a straight line equal to the last observed actual value (again, assuming that $\mathrm{E}(\epsilon_t) = 0$ and that other basic assumptions from Section @ref(assumptions) hold):

$$\mathrm{E}(y_{t+h}|t) = \mathrm{E}(y_{t+h-1}|t) + \mathrm{E}(\epsilon_{t+h}|t) = y_t. (\#eq:ARIMARandomWalkForecast) \tag{8.21}$$

Another simple model that relies on differences of the data is called **Random Walk with drift** (which was discussed in Section @ref(RWWithDrift)) and is formulated by adding constant a_0 to the right hand side of equation @ref(eq:ARIMARandomWalk):

$$\Delta y_t = a_0 + \epsilon_t. (\#eq:ARIMARandomWalkWithDrift) \tag{8.22}$$

This model has some similarities with the global level model, which is formulated via the actual value rather than differences (see Section @ref(GlobalMean)):

$$y_t = a_0 + \epsilon_t. (\#eq:ARIMAGlobalMean) \tag{8.23}$$

Using a similar regrouping as with the Random Walk, we can obtain a simpler form of @ref(eq:ARIMARandomWalkWithDrift):

$$y_t = a_0 + y_{t-1} + \epsilon_t. (\#eq:ARIMARandomWalkWithDrift02) \tag{8.24}$$

which is, again, equivalent to AR(1) model with $\phi_1 = 1$, but this time with a constant. The term "drift" appears because a_0 acts as an additional element, showing the tendency in the data: if it is positive, the model will exhibit a positive trend; if it is negative, the trend will be negative. This can be seen for the conditional mean, for example, for the case of $h = 2$:

$$\mathrm{E}(y_{t+2}|t) = \mathrm{E}(a_0) + \mathrm{E}(y_{t+1}|t) + \mathrm{E}(\epsilon_{t+2}|t) = a_0 + \mathrm{E}(a_0 + y_t + \epsilon_t|t) = 2a_0 + y_t, (\#eq:ARIMARandomWalkWithDriftFo \tag{8.25}$$

or in general for the horizon h:

$$\mathrm{E}(y_{t+h}|t) = ha_0 + y_t.(\#eq : ARIMARandomWalkWithDriftForecasth) \tag{8.26}$$

In a manner similar to @ref(eq:ARIMADifferencesFirst), we can also introduce second differences of the data (differences of differences) if we suspect that the change of variable over time is not stationary, which would be written as:

$$\Delta^2 y_t = \Delta y_t - \Delta y_{t-1} = y_t - y_{t-1} - y_{t-1} + y_{t-2}, (\#eq : ARIMADifferencesSecond) \tag{8.27}$$

which can also be written using the backshift operator:

$$\Delta^2 y_t = y_t(1 - 2B + B^2) = y_t(1-B)^2.(\#eq : ARIMADifferencesSecondBackshift) \tag{8.28}$$

In fact, we can introduce higher level differences if we want (but typically we should not) based on the idea of @ref(eq:ARIMADifferencesSecondBackshift):

$$\Delta^d = (1-B)^d.(\#eq : ARIMADifferences) \tag{8.29}$$

Using @ref(eq:ARIMADifferences), the I(d) model is formulated as:

$$\Delta^d y_t = \epsilon_t.(\#eq : ARIMA0d0) \tag{8.30}$$

8.1.6 ARIMA(p,d,q)

Finally, having made the data stationary via the differences, we can introduce ARMA elements @ref(eq:ARIMAp0qCompact) to it which would be done on the differenced data, instead of the original y_t:

$$y_t \Delta^d(B) \varphi^p(B) = \epsilon_t \vartheta^q(B), (\#eq : ARIMApdqCompact) \tag{8.31}$$

or in a more general form @ref(eq:ARIMAp0qCompacter) with @ref(eq:ARIMADifferencesSecondBackshi

$$y_t(1-B)^d(1 - \phi_1 B - \cdots - \phi_p B^p) = \epsilon_t(1 + \theta_1 B + \cdots + \theta_q B^q), (\#eq : ARIMApdq) \tag{8.32}$$

which is ARIMA(p,d,q) model. This model allows producing trends with some values of differences and also inherits the trajectories from both AR(p) and MA(q). This implies that the point forecasts from the model can exhibit complicated trajectories, depending on the values of the model's parameters.

The model @ref(eq:ARIMApdq) is difficult to interpret in a general form, but opening the brackets and moving all elements but y_t to the right-hand side helps understand each specific model.

8.1.7 Parameters bounds

ARMA models have two conditions that need to be satisfied for them to be useful and to work appropriately:

1. Stationarity,
2. Invertibility.

Condition (1) has already been discussed in Section @ref(ARIMAIntro). It is imposed on the model's AR parameters, ensuring that the forecast trajectories do not exhibit explosive behaviour (in terms of both mean and variance). (2) is equivalent to the stability condition in ETS (Section @ref(stabilityConditionAdditiveError)) and refers to the MA parameters: it guarantees that the old observations do not have an increasing impact on the recent ones. The term "invertibility" comes from the idea that any MA process can be represented as an infinite AR process via the inversion of the parameters. For example, MA(1) model, which is written as:

$$y_t = \epsilon_t + \theta_1 \epsilon_{t-1} = \epsilon_t (1 + \theta_1 B), (\#eq:ARIMA100Example01) \tag{8.33}$$

can be rewritten as:

$$y_t (1 + \theta_1 B)^{-1} = \epsilon_t, (\#eq:ARIMA100Example02) \tag{8.34}$$

or in a slightly easier to digest form (based on @ref(eq:ARIMA100Example01) and the idea that $\epsilon_t = y_t - \theta_1 \epsilon_{t-1}$, implying that $\epsilon_{t-1} = y_{t-1} - \theta_1 \epsilon_{t-2}$):

$$y_t = \theta_1 y_{t-1} - \theta_1^2 \epsilon_{t-2} + \epsilon_t = \theta_1 y_{t-1} - \theta_1^2 y_{t-2} + \theta_1^3 \epsilon_{t-2} + \epsilon_t = \sum_{j=1}^{\infty} -1^{j-1} \theta_1^j y_{t-j} + \epsilon_t. (\#eq:ARIMA100Example03) \tag{8.35}$$

The recursion in @ref(eq:ARIMA100Example03) shows that the recent actual value y_t depends on the previous infinite number of values of y_{t-j} for $j = \{1, \dots, \infty\}$. The parameter θ_1, in this case, is exponentiated and leads to the distribution of weights in this infinite series exponentially (reminds SES from Section @ref(SES), doesn't it?). The *invertibility* condition makes sure that those weights decline over time with the increase of j so that the older observations do not have an increasing impact on the most recent y_t.

There are different ways how to check both conditions, the conventional one is by calculating the roots of the polynomial equations:

$$\begin{matrix} \varphi^p(B) = 0 \text{ for AR} \\ \vartheta^q(B) = 0 \text{ for MA} \end{matrix}, (\#eq:ARIMApdqConditionsCompact) \tag{8.36}$$

or expanding the functions in @ref(eq:ARIMApdqConditionsCompact) and substituting B with a variable x (for convenience):

$$\begin{matrix} 1 - \phi_1 x - \phi_2 x^2 - \dots - \phi_p x^p = 0 \text{ for AR} \\ 1 + \theta_1 x + \theta_2 x^2 + \dots + \theta_q x^q = 0 \text{ for MA} \end{matrix}. (\#eq:ARIMApdqConditions) \tag{8.37}$$

Solving the first equation for x in @ref(eq:ARIMApdqConditions), we get p roots (some of them might be complex numbers). For the model to be stationary, all the roots must be greater than one by absolute value. Similarly, if all the roots

of the second equation in @ref(eq:ARIMApdqConditions) are greater than one by absolute value, then the model is invertible (aka stable).

Calculating roots of polynomials is a difficult task, so there are simpler special cases for both conditions that guarantee that the more complicated ones are satisfied:

$$\begin{aligned} 0 < \sum_{j=1}^{p} \phi_j < 1 \\ .(\#eq:ARIMApdqConditionsSpecial) \\ 0 < \sum_{j=1}^{q} \theta_j < 1 \end{aligned} \tag{8.38}$$

But note that the condition @ref(eq:ARIMApdqConditionsSpecial) is rather restrictive and not generally applicable for all ARIMA models. It can be used to skip the check of the more complicated condition @ref(eq:ARIMApdqConditions) if it is satisfied by a set of estimated parameters.

Finally, in a special case with AR(p) model with $0 < \sum_{j=1}^{p} \phi_j < 1$ and $\sum_{j=1}^{p} \phi_j = 1$, we end up with the moving weighted average, which is a non-stationary model. This becomes apparent from the connection between Simple Moving Average and AR processes (**?**).

8.2 Seasonal ARIMA

8.2.1 Single seasonal ARIMA

When it comes to the actual data, we typically have relations between consecutive observations and between observations happening with some fixed seasonal lags. In the ETS framework, these relations are taken care of via seasonal indices, repeating every m observations. In the ARIMA framework, this is done via introducing lags in the model elements. For example, seasonal AR(P)_m with lag m can be written similar to AR(p), but with some minor modifications:

$$y_t = \phi_{m,1} y_{t-m} + \cdots + \phi_{m,P} y_{t-Pm} + \varepsilon_t, (\#eq:SARIMAP00Example) \tag{8.39}$$

where $\phi_{m,i}$ is the parameter for the lagged im actual value in the model, and ε_t is the error term of the seasonal AR model. We use the underscore "m" to show that the parameters here refer to the seasonal part of the model. The idea of the model @ref(eq:SARIMAP00Example) on the example of monthly data is that the current observation is influenced by a similar value, the same month a year ago, then the same month two years ago etc. This is hard to justify from the theoretical point of view, but this model allows capturing complex relations in the data.

Similarly to seasonal AR(P), we can have seasonal MA(Q)_m:

$$y_t = \theta_{m,1} \varepsilon_{t-m} + \cdots + \theta_{m,Q} \varepsilon_{t-Qm} + \varepsilon_t, (\#eq:SARIMA00QExample) \tag{8.40}$$

where $\theta_{m,i}$ is the parameter for the lagged error term in the model. This model is even more difficult to justify than the MA(q) because it is difficult to explain how the white noise the same month last year can impact the actual value this year. Still, this is a useful instrument for forecasting purposes.

Finally, we have the seasonal differences, I(D)_m, which are easier to present using the backshift operator:

$$y_t(1-B^m)^D = \varepsilon_t.(\#eq:SARIMA0D0Example) \tag{8.41}$$

The seasonal differences allow dealing with the seasonality that changes its amplitude in the data, i.e. model the multiplicative seasonality via ARIMA by making the seasonality itself stationary.

A special case of I(D) model is I(1)_m, which is a seasonal **Random Walk**, underlying Seasonal Naïve method from Section @ref(NaiveSeasonal):

$$y_t(1-B^m) = \varepsilon_t, (\#eq:SARIMA010Example1) \tag{8.42}$$

or equivalently:

$$y_t = y_{t-m} + \varepsilon_t.(\#eq:SARIMA010Example2) \tag{8.43}$$

Combining @ref(eq:SARIMAP00Example), @ref(eq:SARIMA00QExample) and @ref(eq:SARIMA0D0Example) we get pure seasonal ARIMA(P,D,Q)_m model in the compact notation, similar to the one we had for ARIMA(p,d,q):

$$y_t(1-B^m)^D(1-\phi_{m,1}B^m-\cdots-\phi_{m,P}B^{Pm}) = \varepsilon_t(1+\theta_{m,1}B^m+\cdots+\theta_{m,Q}B^{Qm}), (\#eq:SARIMAPDQ) \tag{8.44}$$

or if we introduce the polynomial functions for seasonal AR and MA and use notation similar to @ref(eq:ARIMADifferencesSecondBackshift):

$$y_t\Delta^D(B^m)\varphi^P(B^m) = \varepsilon_t\vartheta^Q(B^m), (\#eq:SARIMAPDQCompact) \tag{8.45}$$

where

$$\begin{aligned}
&\Delta^D(B^m) = (1-B^m)^D \\
&\varphi^P(B^m) = 1-\phi_{m,1}B^m-\cdots-\phi_{m,P}B^{Pm} (\#eq:SARIMAPolynomials) \\
&\vartheta^Q(B^m) = 1+\theta_{m,1}B^m+\cdots+\theta_{m,Q}B^{Qm}.
\end{aligned} \tag{8.46}$$

Now that we have taken care of the seasonal part of the model, we should not forget that there is a non-seasonal one. If it exists in the data, then ε_t would not be just a white noise, but could be modelled using a non-seasonal ARIMA(p,d,q):

$$\varepsilon_t\Delta^d(B)\varphi^p(B) = \epsilon_t\vartheta^q(B), (\#eq:ARIMApdqForError) \tag{8.47}$$

implying that:

$$\varepsilon_t = \epsilon_t\frac{\vartheta^q(B)}{\Delta^d(B)\varphi^p(B)}.(\#eq:ARIMApdqForErrorRewritten) \tag{8.48}$$

Inserting @ref(eq:ARIMApdqForErrorRewritten) into @ref(eq:SARIMAPDQCompact), we get the final SARIMA(p,d,q)(P,D,Q)$_m$ model in the compact form after regrouping the polynomials:

$$y_t\Delta^D(B^m)\varphi^P(B^m)\Delta^d(B)\varphi^p(B) = \epsilon_t\vartheta^Q(B^m)\vartheta^q(B).(\#eq:SARIMApdqPDQCompact) \tag{8.49}$$

The equation @ref(eq:SARIMApdqPDQCompact) does not tell us much about what happens in the model. It just shows how different elements interact with each other in it. To understand, what SARIMA means, we need to take an example and see what impacts the current actual value. For example, here is what we will have in the case of SARIMA(1,0,1)(1,0,1)$_4$ (i.e. applied to stationary quarterly data):

$$y_t\Delta^0(B^4)\varphi^1(B^4)\Delta^0(B)\varphi^1(B) = \epsilon_t\vartheta^1(B^4)\vartheta^1(B).(\#eq:SARIMA101101Example01) \tag{8.50}$$

Inserting the values of polynomials @ref(eq:SARIMAPolynomials), @ref(eq:ARIMADifferences) and @ref(eq:ARMAPolynomials) in @ref(eq:SARIMA101101Example01), we get:

$$y_t(1-\phi_{4,1}B^4)(1-\phi_1B) = \epsilon_t(1+\theta_{4,1}B^4)(1+\theta_1B),(\#eq:SARIMA101101Example02) \tag{8.51}$$

which is slightly easier to understand but still does not explain how past values impact the present. So, we open the brackets and move all the elements except for y_t to the right-hand side of the equation to get:

$$y_t = \phi_1y_{t-1}+\phi_{4,1}y_{t-4}-\phi_1\phi_{4,1}y_{t-5}+\theta_1\epsilon_{t-1}+\theta_{4,1}\epsilon_{t-4}+\theta_1\theta_{4,1}\epsilon_{t-5}+\epsilon_t.(\#eq:SARIMA101101Example03) \tag{8.52}$$

So, now we see that SARIMA(1,0,1)(1,0,1)$_4$ implies that the present values is impacted by the value in the previous quarter ($\phi_1y_{t-1} + \theta_1\epsilon_{t-1}$), the value last year ($\phi_{4,1}y_{t-4} + \theta_{4,1}\epsilon_{t-4}$) on the same quarter and the value from last year on the previous quarter ($-\phi_1\phi_{4,1}y_{t-5} + \theta_1\theta_{4,1}\epsilon_{t-5}$), which introduces a much more complicated interaction than any ETS model can. However, this complexity is obtained with a minimum number of parameters: we have three lagged actual values and three lagged error terms, but we only have four parameters to estimate, not six. The more complicated SARIMA models would have even more complicated interactions, making it more challenging to interpret, but all of that comes with a benefit of having a parsimonious model with just $p + q + P + Q$ parameters to estimate.

When it comes to forecasting from such model as SARIMA(1,0,1)(1,0,1)$_4$, the trajectories would have elements of the classical ARMA model, discussed in Section @ref(ARMA), converging to zero as long as there is no constant and the model is stationary. The main difference would be in having the seasonal element. Here is an R example of a prediction for such a model for $h > m + 1$ (see Figure @ref(fig:SARIMATrajectory); MA part is dropped because the expectation of the error term is assumed to be equal to zero):

```
y <- vector("numeric", 100)
y[1:5] <- c(97,87,85,94,95)
phi <- c(0.6,0.8)
for(i in 6:length(y)){
    y[i] <- phi[1] * y[i-1] + phi[2] * y[i-4] -
      phi[1] * phi[2] * y[i-5]
}
plot(y, type="l", xlab="horizon", ylab="Forecast")
```

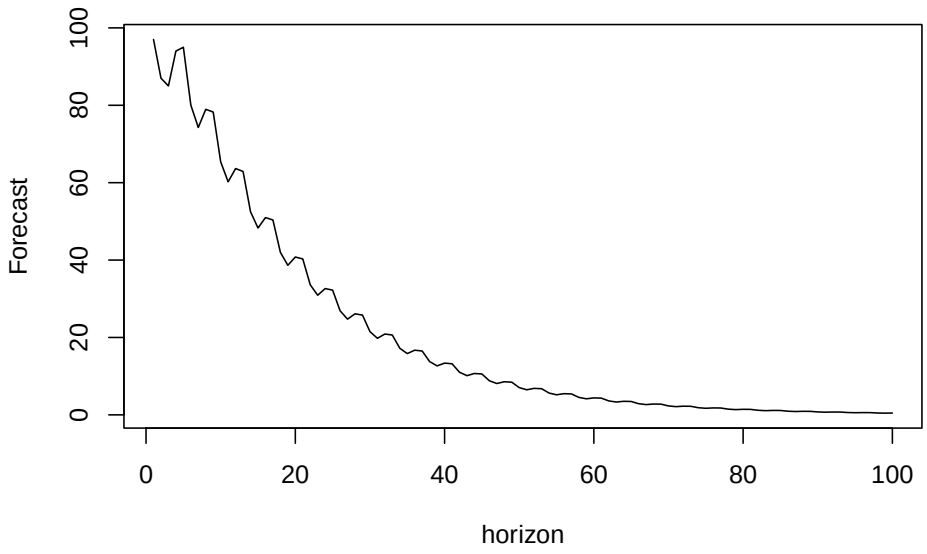


Figure 8.4: (#fig:SARIMATrajectory)Forecast trajectory for a SARIMA$(1,0,1)(1,0,1)_4$.

As we see from Figure @ref(fig:SARIMATrajectory), the values converge to zero due to $0 < \phi_1 < 1$ and the seasonality disappears because $0 < \phi_{4,1} < 1$ as well. So, this is the forecast implied by the SARIMA without differences. If the differences are introduced, then the model would produce non-stationary and seasonally non-stationary trajectories.

8.2.2 SARIMA with constant

In addition, it is possible to add the constant term to the SARIMA model, and it will have a more complex effect on the forecast trajectory, depending on the order of the model. In case of zero differences, the effect will be similar to ARMA (Section @ref(ARMAConstant)), introducing the dampening trajectory. Here is an example (see Figure @ref(fig:SARIMAConstantTrajectory)):

```
y <- vector("numeric", 100)
y[1:5] <- c(97,87,85,94,95)
phi <- c(0.6,0.8)
for(i in 6:length(y)){
    y[i] <- phi[1] * y[i-1] + phi[2] * y[i-4] -
      phi[1] * phi[2] * y[i-5] + 8
}
plot(y, type="l", xlab="horizon", ylab="Forecast")
```

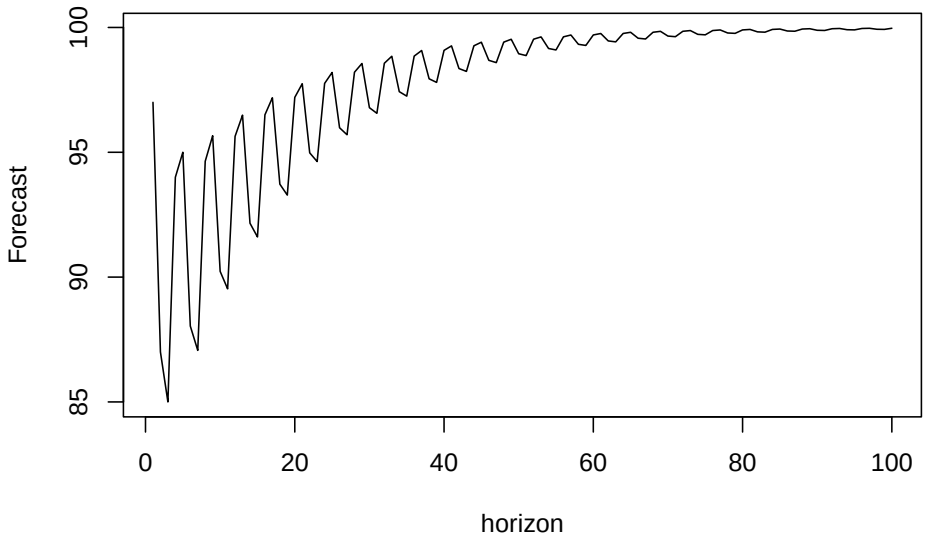


Figure 8.5: (#fig:SARIMAConstantTrajectory)Forecast trajectory for a SARIMA model with constant.

In case of the model with the differences, the constant would have a two-fold effect: working as a drift for the non-seasonal part and increasing the amplitude of seasonality for the seasonal one. Here is an example from SARIMA$(1,0,0)(1,1,0)_4$ with constant:

$$y_t(1-\phi_{4,1}B^4)(1-\phi_1 B)(1-B^4) = \epsilon_t + a_0, (\#eq:SARIMA101110Example01) \tag{8.53}$$

which can be reformulated as (after opening brackets and moving elements to the right-hand side):

$$y_t = \phi_1 y_{t-1} + (1+\phi_{4,1})y_{t-4} - (1+\phi_{4,1})\phi_1 y_{t-5} - \phi_{4,1}y_{t-8} + \phi_1\phi_{4,1}y_{t-9} + a_0 + \epsilon_t. (\#eq:SARIMA101110Exam$$

(8.54)

This formula can then be used to see, what the trajectory from such model will be:

```
y <- vector("numeric", 100)
y[1:9] <- c(96,87,85,94,97,88,86,95,98)
phi <- c(0.6,0.8)
for(i in 10:length(y)){
    y[i] <- phi[1] * y[i-1] + (1+phi[2]) * y[i-4] -
      (1+ phi[2]) *phi[1] * y[i-5] - phi[2] * y[i-8] +
      phi[1] * phi[2] * y[i-9] + 0.1
}
plot(y, type="l", xlab="horizon", ylab="Forecast")
```

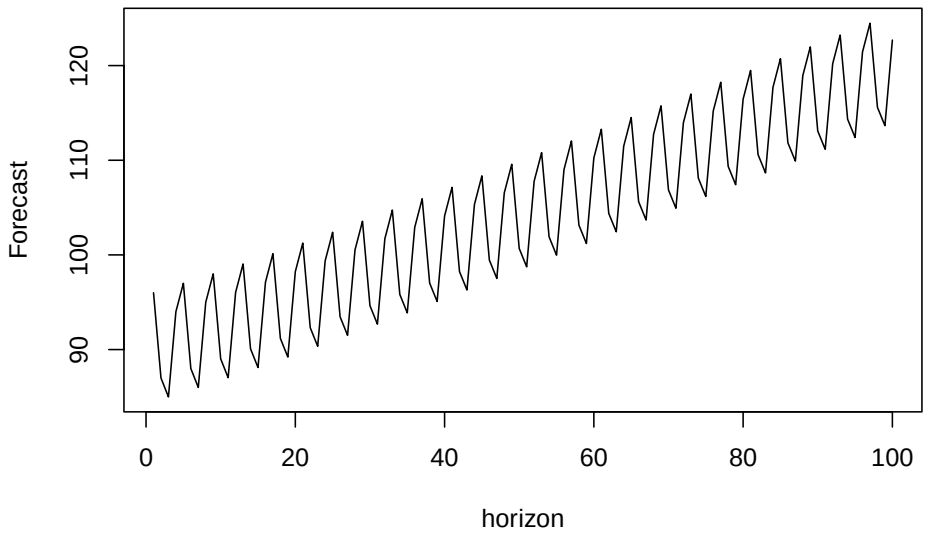


Figure 8.6: (#fig:SARIMADiffConstantTrajectory)Forecast trajectory for a SARIMA model with differences and a constant.

As we see from Figure @ref(fig:SARIMADiffConstantTrajectory), the trajectory exhibits a drift, coming from the non-seasonal part of the model and a stable seasonality (the amplitude of which does not converge to zero anymore). More complex behaviours for the future trajectories can be obtained with higher orders of seasonal and non-seasonal parts of the SARIMA model.

8.2.3 Multiple seasonal ARIMA

Using the same approach as the conventional SARIMA, we can introduce more terms (similar to how **?**, did it) with several seasonal frequencies. For example, we can have an hour of the day, a day of the week and a week of year frequencies in the data. Given that we work with the hourly data in this situation, we should introduce three seasonal ARIMA elements with seasonalities $m_1 = 24$, $m_2 = 24 \times 7$ and $m_3 = 24 \times 7 \times 365$. In this example, we would have AR, I

and MA polynomials for each seasonal part of the model, introducing a triple seasonal ARIMA, which is not even easy to formulate in the compact form. This type of model with multiple seasonal components can be called "Multiple Seasonal ARIMA", MSARIMA, which in general can be written as:

$$y_t \Delta^{D_n}(B^{m_n})\varphi^{P_n}(B^{m_n}) ... \Delta^{D_0}(B^{m_0})\varphi^{P_0}(B^{m_0}) = \epsilon_t \vartheta^{Q_n}(B^{m_n}) ... \vartheta^{Q_0}(B^{m_0}), (\#eq : MSARIMACompac \tag{8.55}$$

where n is the number of seasonal cycles, and $D_0 = d$, $P_0 = p$, $Q_0 = q$ and $m_0 = 1$ for convenience. The slightly more compact and even less comprehensible form of @ref(eq:MSARIMACompact) is:

$$y_t \prod_{j=0}^{n} \Delta^{D_j}(B^{m_j})\varphi^{P_j}(B^{m_j}) = \epsilon_t \prod_{j=0}^{n} \vartheta^{Q_j}(B^{m_j}), (\#eq : MSARIMACompactFinal) \tag{8.56}$$

Conceptually, the model @ref(eq:MSARIMACompactFinal) is neat, as it captures all the complex relations in the data, but it is not easy to understand and work with, not to mention the potential estimation and order selection problems. To understand what the forecast from such a model can be, we would need to take a special case, multiply the polynomials and move all the past elements on the right-hand side, leaving only y_t on the left-hand side one, as we did with SARIMA example above. It is worth noting that the `msarima()` function from the `smooth` package implements the model @ref(eq:MSARIMACompactFinal), although not in this form, but in the state space form, discussed in Chapter @ref(ADAMARIMA).

8.2.4 Parameters bounds for MSARIMA

When it comes to parameters bounds of SARIMA, the logic stays similar to the process discussed for the case of non-seasonal model in Section @ref(ARIMABounds), with the only difference being that instead of analysing the polynomials of a specific part of a model, we need to consider the product of all polynomials. So, the *stationarity* condition for the MSARIMA is for all the roots of the following polynomial to be greater than one by absolute value (lie outside the unit circle):

$$\prod_{j=0}^{n} \varphi^{P_j}(B^{m_j}) = 0, (\#eq : MSARIMABoundsStationarity) \tag{8.57}$$

while the invertibility condition is for all the roots of the following polynomial to lie outside the unit circle:

$$\prod_{j=0}^{n} \vartheta^{Q_j}(B^{m_j}) = 0. (\#eq : MSARIMABoundsInvertibility) \tag{8.58}$$

Both of these conditions are difficult to check, especially for high frequencies m_j: the polynomial equation of order n has n complex roots, so if

you fit a multiple seasonal ARIMA on hourly data, where the maximum frequency is $24 \times 7 \times 365 = 61,320$, then the equation will have at least 61,320 roots (this number will increase if there are lower frequencies or non-seasonal orders of the model). Finding all of them is not a trivial task even for modern computers (for example, the `polyroot()` function from the `base` package cannot handle this). So, when considering ARIMA on high-frequency data with high seasonal frequency values, it might make sense to find other ways of checking the stationarity and stability conditions. The `msarima()` and the `adam()` functions in the `smooth` package use the state space form of ARIMA (discussed in Chapter @ref(StateSpaceARIMA)) and rely on slightly different principles of checking the same conditions. They do that more efficiently than in the case of the conventional approach of finding the roots of polynomials @ref(eq:MSARIMABoundsStationarity) and @ref(eq:MSARIMABoundsInvertibility).

8.3 Box-Jenkins approach

Now that we are more or less familiar with the idea of ARIMA models, we can move to practicalities. As it might become apparent from the previous sections, one of the issues with the model is the identification of orders p, d, q, P_j, D_j, Q_j etc. Back in the 20th century, when computers were slow, this was a challenging task, so George Box and Gwilym Jenkins (**?**) developed a methodology for identifying and estimating ARIMA models. While there are more efficient ways of order selection for ARIMA nowadays, some of their principles are still used in time series analysis and in forecasting. We briefly outline the idea in this section, not purporting to give a detailed explanation of the approach.

8.3.1 Identifying stationarity

Before doing any time series analysis, we need to make the data stationary, which is done via the differences in the context of ARIMA (Section @ref(Differences)). But before doing anything, we need to understand whether the data is stationary or not in the first place: over-differencing typically is harmful to the model and would lead to misspecification issues. At the same time, in the case of under-differencing, it might not be possible to identify the model correctly.

There are different ways of understanding whether the data is stationary or not. The simplest of them is just looking at the data: in some cases, it becomes evident that the mean of the data changes or that there is a trend in the data, so the conclusion would be relatively easy to make. If it is not stationary, then taking differences and analysing the differenced data again would be the next step to ensure that the second differences are not needed.

The more formal approach would be to conduct statistical tests, such as ADF (the `adf.test()` from the `tseries` package) or KPSS (the `kpss.test()` from the `tseries` package). Note that they test different hypotheses:

1. In the case of ADF, it is:

- H_0: the data is **not** stationary;
- H_1: the data is stationary;

2. In the case of KPSS:

- H_0: the data is stationary;
- H_1: the data is **not** stationary;

I do not plan to discuss how the tests are conducted and what they imply. It should suffice to say that ADF is based on estimating parameters of the AR model and then testing the hypothesis for those parameters, while KPSS includes the component of Random Walk in a model (with potential trend) and checks whether the variance of that component is zero or not. Both tests have their advantages and disadvantages and sometimes might contradict each other. No matter what test you choose, do not forget what testing a statistical hypothesis means (see Section 5.3 of **?**): if you fail to reject H_0, it does not mean that it is true.

Note that even if you select the test-based approach, the procedure should still be iterative: test the hypothesis, take differences if needed, test the hypothesis again etc. This way, we can determine the order of differences I(d).

When you work with seasonal data, the situation becomes more complicated. Yes, you can probably spot seasonality by visualising the data, but it is not easy to conclude whether the seasonal differences are needed. In this case, the Canova-Hansen test (the `ch.test()` in the `uroot` package) can be used to formally test the hypothesis similar to the one in the KPSS test, but about the seasonal differences.

Only after making sure that the data is stationary we can move to the identification of AR and MA orders.

8.3.2 Autocorrelation function (ACF)

In the core of the Box-Jenkins approach, lies the idea of autocorrelation and partial autocorrelation functions. **Autocorrelation** is the correlation (see Section 6.3 of **?**) of a variable with itself from a different period of time. Here is an example of autocorrelation coefficient for lag 1:

$$\rho(1) = \frac{\sigma_{y_t,y_{t-1}}}{\sigma_{y_t}\sigma_{y_{t-1}}} = \frac{\sigma_{y_t,y_{t-1}}}{\sigma^2_{y_t}}, (\#eq : autoCorrelation) \tag{8.59}$$

where $\rho(1)$ is the "true" autocorrelation coefficient, $\sigma_{y_t,y_{t-1}}$ is the covariance between y_t and y_{t-1}, while σ_{y_t} and $\sigma_{y_{t-1}}$ are the "true" standard deviations of y_t and y_{t-1}. Note that $\sigma_{y_t} = \sigma_{y_{t-1}}$, because we are talking about one and the same variable, thus the simpler formula on the right-hand side of @ref(eq:autoCorrelation). The formula @ref(eq:autoCorrelation) corresponds to the classical correlation coefficient, so this interpretation is the same as for

the classical one: the value of $\rho(1)$ shows the closeness of the lagged relation to linear. If it is close to one, then this means that variable has a strong linear relation with itself on the previous observation. It obviously does not tell you anything about the causality, just shows a technical relation between variables, even if it is spurious in real life.

Using the formula @ref(eq:autoCorrelation), we can calculate the autocorrelation coefficients for other lags as well, just substituting y_{t-1} with y_{t-2}, y_{t-3}, ..., $y_{t-\tau}$ etc. In a way, $\rho(\tau)$ can be considered as a function of a lag τ, which is called the "Autocorrelation function" (ACF). If we know the order of the ARIMA process we deal with, we can plot the values of ACF on the y-axis by changing the τ on the x-axis. **?** discuss different theoretical functions for several special cases of ARIMA, which we do not plan to repeat here fully. But, for example, they show that if you deal with AR(1) process, then the $\rho(1) = \phi_1$, $\rho(2) = \phi_1^2$ etc. This can be seen on the example of $\rho(1)$ by calculating the covariance for AR(1):

$$\sigma_{y_t,y_{t-1}} = \text{cov}(y_t, y_{t-1}) = \text{cov}(\phi_1 y_{t-1} + \epsilon_t, y_{t-1}) = \text{cov}(\phi_1 y_{t-1}, y_{t-1}) = \phi_1 \sigma^2_{y_t}, (\#eq : autoCovarianceAR1) \tag{8.60}$$

which when inserted in @ref(eq:autoCorrelation) leads to $\rho(1) = \phi_1$. The ACF for AR(1) with a positive ϕ_1 will have the shape shown in Figure @ref(fig:ACFExampleAR1) (on the example of $\phi_1 = 0.9$).

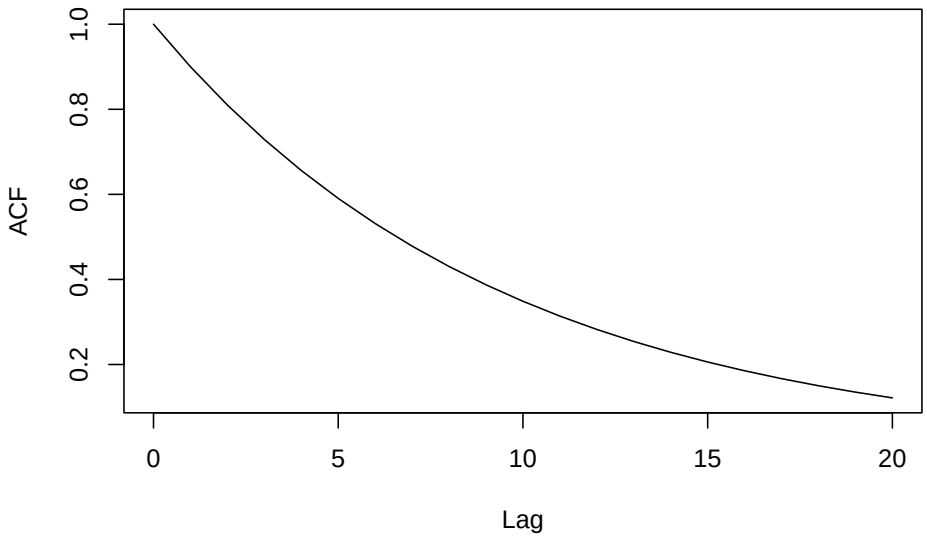


Figure 8.7: (#fig:ACFExampleAR1)ACF for AR(1) model.

Note that $\rho(0) = 1$ just because the value is correlated with itself, so lag 0 is typically dropped as not being useful. The declining shape of the ACF tells us that if y_t is correlated with y_{t-1}, then the correlation between y_{t-1} and y_{t-2}

will be exactly the same, also implying that y_t is somehow correlated with y_{t-2}, even if there is no true correlation between them. It is difficult to say anything for the AR process based on ACF exactly because of this temporal relation of the variable with itself.

On the other hand, ACF can be used to judge the order of MA(q) process. For example, if we consider MA(1) (Section @ref(MA)), then the $\rho(1)$ will depend on the following covariance:

$$\sigma_{y_t,y_{t-1}} = \text{cov}(y_t, y_{t-1}) = \text{cov}(\theta_1\epsilon_{t-1}+\epsilon_t, \theta_1\epsilon_{t-2}+\epsilon_{t-1}) = \text{cov}(\theta_1\epsilon_{t-1}, \epsilon_{t-1}) = \theta_1\sigma^2, (\#eq : autoCovariance \quad (8.61)$$

where σ^2 is the variance of the error term, which in case of MA(1) is equal to $\sigma^2_{y_t}$, because $\text{E}(y_t) = 0$. However, the covariance between the higher lags will be equal to zero for the pure MA(1) (given that the usual assumptions from Section @ref(assumptions) hold). **?** showed that for the moving averages, ACF tells more about the order of the model than for the autoregressive one: *ACF will drop rapidly right after the specific lag q for the MA(q) process.*

When it comes to seasonal models, in the case of seasonal AR(P), ACF will decrease exponentially from season to season (e.g. you would see a decrease on lags 4, 8, 12 etc. for SAR(1) and $m = 4$), while in case of seasonal MA(Q), ACF would drop abruptly, starting from the lag $(Q + 1)m$ (so, the subsequent seasonal lag from the one that the process has, e.g. on lag 8, if we deal with SMA(1) with $m = 4$).

8.3.3 Partial autocorrelation function (PACF)

The other instrument useful for time series analysis with respect to ARIMA is called "partial ACF". The idea of this follows from ACF directly. As we have spotted, if the process we deal with follows AR(1), then $\rho(2) = \phi_1^2$ just because of the temporal relation. In order to get rid of this temporal effect, when calculating the correlation between y_t and y_{t-2} we could remove the correlation $\rho(1)$ in order to get the clean effect of y_{t-2} on y_t. This type of correlation is called "partial", denoting it as $\varrho(\tau)$. There are different ways how to do that. One of the simplest is to estimate the following regression model:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_\tau y_{t-\tau} + \epsilon_t, (\#eq : PACFRegression) \quad (8.62)$$

where a_i is the parameter for the i-th lag of the model. In this regression, all the relations between y_t and $y_{t-\tau}$ are captured separately, so the last parameter a_τ is clean of all the temporal effects discussed above. We then can use the value $\varrho(\tau) = a_\tau$ as the coefficient, showing this relation. In order to obtain the PACF, we would need to construct and estimate regressions @ref(eq:PACFRegression) for each lag $\tau = \{1, 2, \dots, p\}$ and get the respective parameters a_1, a_2, ..., a_p, which would correspond to $\varrho(1)$, $\varrho(2)$, ..., $\varrho(p)$.

Just to show what this implies, we consider calculating PACF for AR(1) process. In this case, the true model is:

$$y_t = \phi_1 y_{t-1} + \epsilon_t.$$

For the first lag we estimate exactly the same model, so that $\varrho(1) = \phi_1$. For the second lag we estimate the model:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \epsilon_t.$$

But we know that for AR(1), $a_2 = 0$, so when estimated in population, this would result in $\varrho(2) = 0$ (in case of a sample, this would be a value very close to zero). If we continue with other lags, we will come to the same conclusion: for all lags $\tau > 1$ for the AR(1), we will have $\varrho(\tau) = 0$. This is one of the properties of PACF: *if we deal with AR(p) process, then PACF drops rapidly to zero right after the lag p.*

When it comes to MA(q) process, PACF behaves differently. In order to understand how it would behave, we take an example of MA(1) model, which is formulated as:

$$y_t = \theta_1 \epsilon_{t-1} + \epsilon_t.$$

As it was discussed in Section @ref(ARIMABounds), MA process can be also represented as an infinite AR (see @ref(eq:ARIMA100Example03) for derivation):

$$y_t = \sum_{j=1}^{\infty} -1^{j-1} \theta_1^j y_{t-j} + \epsilon_t.$$

If we construct and estimate the regression @ref(eq:PACFRegression) for any lag τ for such process we will get $\varrho(\tau) = -1^{\tau-1}\theta_1^{\tau}$. This would correspond to the exponentially decreasing curve (if the parameter θ_1 is positive, this will be an alternating series of values), similar to the one we have seen for the AR(1) and ACF. More generally, PACF will decline exponentially for MA(q) process, starting from the $\varrho(q) = \theta_q$.

When it comes to seasonal ARIMA models, the behaviour of PACF would resemble one of the non-seasonal ones, but with lags, multiple to the seasonality m. e.g., for the SAR(1) process with $m = 4$, the $\varrho(4) = \phi_{4,1}$, while $\varrho(8) = 0$.

8.3.4 Summary

Summarising the discussions in this section, we can conclude that:

1. For AR(p) process, ACF will decrease either exponentially or alternating (depending on the parameters' values), starting from the lag p;
2. For AR(p) process, PACF will drop abruptly right after the lag p;
3. For MA(q) process, ACF will drop abruptly right after the lag q;
4. For MA(q) process, PACF will decline either exponentially or alternating (based on the specific values of parameters), starting from the lag q.

These rules are tempting to use when determining the appropriate order of the ARMA model. The Box-Jenkins approach relies on that, considering that the seasonal orders should be identified first. The order selection process is iterative, analysing the ACF / PACF of residuals of tested models.