Seasonal Adjustment in R

Table of contents

Part I

Welcome

This is the website for the work-in-progress edition of *Seasonal Adjustment in R*, an online Book by James Livsey and Christoph Sax.

About the book

This book will teach you how to do seasonal adjustment in R, using X13-ARIMA-SEATS.

Specifically, the audience will be both R users who want to learn about seasonal adjustment as well as seasonal adjustment practitioners, who are interested in using R. The book will be tailored to the practical applications of seasonal adjustment within R. It presents background material and references for the theoretically minded reader. The main focus, however, is on concrete problems and examples.

We will showcase methods through detailed examples with associated code. This presentation allows the academic level to be quite broad; understood by undergraduates all the way through advanced Ph.D. students.

Key features of the book

- Each chapter include a concrete practical problem and shows how X-13 can be used to address it
- Teach-by-example format
- Continuous connection of X-13ARIMA-SEATS input with R input and vice-versa
- Fundamental theoretical material is referenced throughout (mainly as an option)
- For each example given the book will give answers, code and provide data

1 Introduction

> **Note**
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is complete enough as an education tool and can be used in a course. It needs additional polishing. **It is part of the course materials intended for Dec 21, 2022**.

This book will teach you how to do seasonal adjustment in R using X-13ARIMA-SEATS. The audience will be both R users who want to learn about seasonal adjustment and seasonal adjustment practitioners who are interested in using R. The book will be tailored to the practical applications of seasonal adjustment within R.

Seasonal Adjustment

Many time series exhibit a regular seasonal pattern over the year. US unemployment, for example, is usually higher from January to March and again in June and July. Similarly, retail sales tend to peak during the Christmas season. This seasonal behavior is regular and predictable. The goal of seasonal adjustment is to estimate and remove the seasonal component from a time series.

Why do we want to do this? Seasonal data is usually hard to interpret. For example, if we want to learn from the US unemployment rate if the economy is moving out of a recession during certain months, we want the labor market data to be free from seasonal effects.

X-13ARIMA-SEATS

Fundamentally, seasonal adjustment decomposes a time series into a **trend**, a **seasonal**, and an **irregular** component. Seasonal adjustment is then the act of removing the seasonal estimate from the observed series. There are many ways to perform this decomposition. This book focuses on a particular one, X-13ARIMA-SEATS (X-13, for short), the seasonal adjustment software developed by the United States Census Bureau. X-13 offers an elaborate toolkit to perform seasonal adjustment. The software allows users to control all aspects of the modeling process or alternatively, to use automated methods to make all modeling choices. In the text we will try to present material with this in mind. Throughout we offer suggestions about when built-in automatic method are sufficient (and sometimes even preferred) and when an analyst can get the most "bang for the buck'' to control modeling options themselves.

R

This book will teach you how to use X-13 in R through the *seasonal* package, which offers access to all features of X-13 with a usually much simpler syntax. It should be noted that the seasonal package is not a re-coding of the X-13 software. Instead it is a translation from R to the X-13 software. This translation is done under the hood and practitioners need not concern themselves with this inner working of the package. However, we make this point such that users understand that conceptually anything that can be done in the native X-13ARIMA-SEATS software, either from the command line or HTML version, can be done in the seasonal R package. Also this means if additional clarification or information about seasonal adjustment is desired, the X-13 manual or research papers can be consulted. Any example or methods found via the X-13 documentation can be easily translated to the R seasonal package. In fact, all examples from the X-13 manual can be seen run in the R seasonal package at http://www.seasonal.website/examples.html. The required X-13 binaries are provided by the *x13binary* package and automatically included in *seasonal*. The next chapter

provides a minimal example to get you started in less than five minutes.

Target audience

We write this book for two primary audiences: The first focus is on current practitioners of seasonal adjustment who are interested in learning how to implement in R. This audience includes researchers from statistical agencies who want to use features of R to evaluate the properties of their seasonal adjustments.

The second focus is on current R users who want to learn seasonal adjustment. We are able to leverage the reader's knowledge of R to make learning seasonal adjustment easier. We will feature exciting applications outside official statistics, such as the seasonal adjustment of business data.

The book tries to be as practical as possible. It usually starts with a practical problem and shows how to solve it in a cookbook style. Formal derivations are usually avoided. Each chapter ends with a case study that discusses a real-life example of the topic.

History of X-13

In official statistics, seasonal adjustment has a long tradition. The US Census Bureau developed the original X-11 software in the 1960s, Statistics Canada (Dagum 1980) continued the development afterward. The following software packages by the US Census Bureau were called X-12-ARIMA (Findley et al. 1998) and X-13ARIMA-SEATS (or X-13, for short) (Monsell 2007). X-11 is still used as a name for filter-based seasonal adjustment methods within X-13. Meanwhile, TRAMO-SEATS, developed by the Bank of Spain (Caporello, Maravall, and Sánchez 2001), offers an alternative model-based approach to seasonal adjustment.

In its most recent version, X-13 offers these two seasonal adjustment methods in a single command-line tool written in Fortran. The National Bank of Belgium has created an alternative

Dagum, Estela Bee. 1980. *The x-11-ARIMA Seasonal Adjustment Method*. Statistics Canada, Seasonal Adjustment; Time Series Staff.

Findley, David F, Brian C Monsell, William R Bell, Mark C Otto, and Bor-Chung Chen. 1998. "New Capabilities and Methods of the x-12-ARIMA Seasonal-Adjustment Program." *Journal of Business & Economic Statistics* 16 (2): 127–52.

Monsell, B. 2007. "The x-13A-s Seasonal Adjustment Program." In *Proceedings of the 2007 Federal Committee on Statistical Methodology Research Conference*. http://www.fcsm.gov/07papers/Monsell.II-B.pdf.

Caporello, Gianluca, Agustin Maravall, and Fernando J Sánchez. 2001. "Program TSW Reference Manual." 0112. Banco de España Madrid. https://ideas.repec.org/p/bde/wpaper/0112.html.

Java-based implementation called JDemetra+ (National Bank of Belgium, Deutsche Bundesbank, Eurostat 2017), also widely deployed by statistical agencies.

National Bank of Belgium, Deutsche Bundesbank, Eurostat. 2017. *JDemetra+: Econometric Software for Seasonal Adjustment and Other Time Series Methods*. Eurostat. https://ec.europa.eu/eurostat/cros/content/download.

1.1 The seasonalbook package

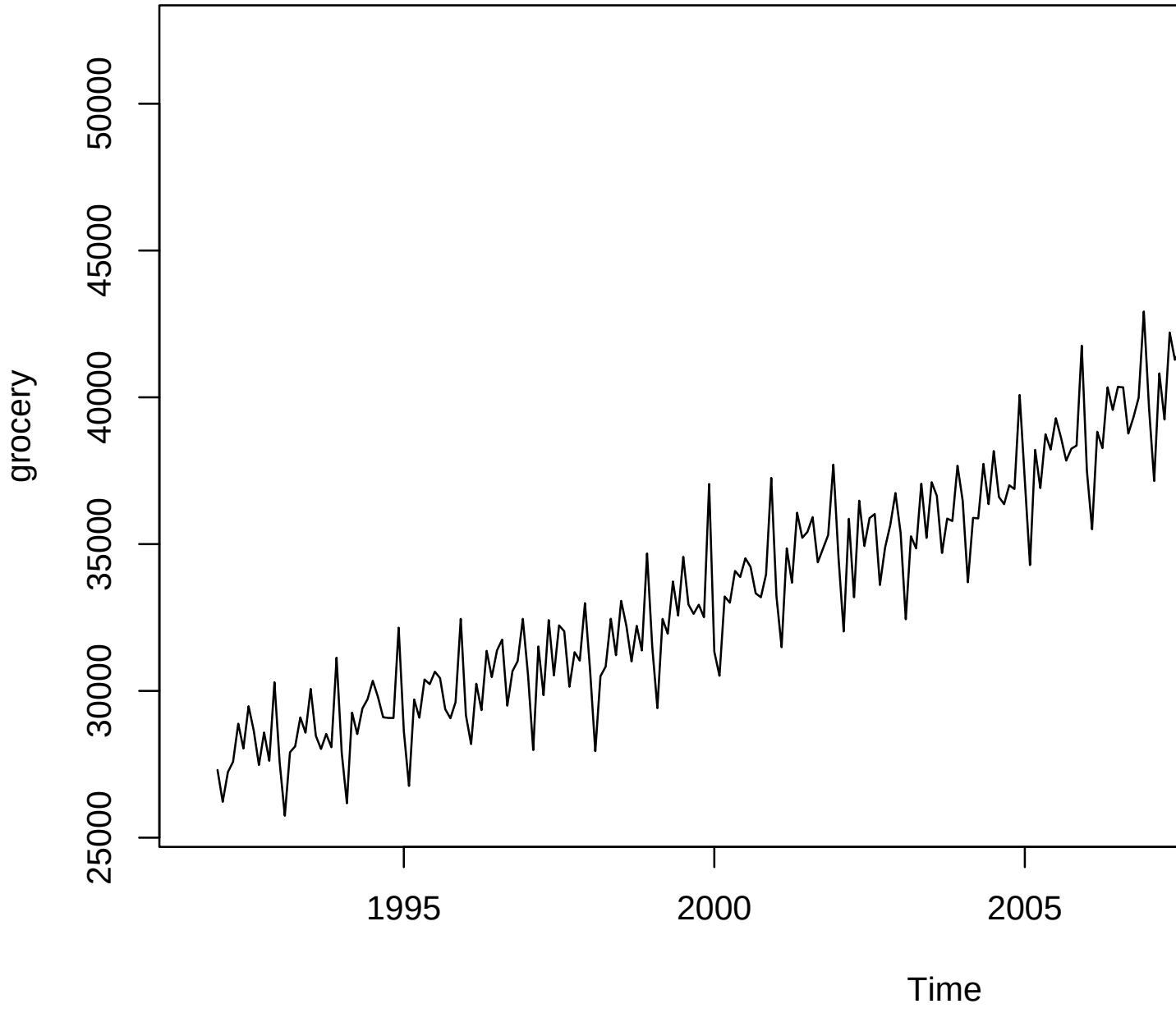
An R package that supplements "Seasonal Adjustment in R", and contains all data and examples.

To install:

```
remotes::install_github("christophsax/seasonalbook")
```

Example series:

```
library(seasonalbook)
plot(grocery)
```


grocery
25000
30000
35000
40000
45000
50000
1995
2000
2005
Time

Acknowledgements

We are indebted to the United States Census Bureau for X-13ARIMA-SEATS and support for research around the software. Help and support by Brian Monsell are especially acknowledged.

seasonal was originally developed for use at the Swiss State Secretariat of Economic Affairs. It has been dramatically improved thanks to suggestions and support from Matthias Bannert, Freya Beamish, Vidur Dhanda, Alain Galli, Ronald Indergand, Preetha Kalambaden, Stefan Leist, James Livsey, Pinaki Mukherjee, Bruno Parnisari and many others. The related work on the x13binary package facilitated (automated) deployment thanks to the R package system, CRAN, and GitHub for the x13prebuilt repository.

References

2 Getting started

> **Note**
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is complete enough as an education tool and can be used in a course. It needs additional polishing. **It is part of the course materials intended for Dec 21, 2022**.

2.1 Installation

If you use R, installing X-13ARIMA-SEATS from CRAN is as easy as installing any other R package (Sax and Eddelbuettel 2018):

```
install.packages("seasonal")
```

Sax, Christoph, and Dirk Eddelbuettel. 2018. "Seasonal Adjustment by X-13ARIMA-SEATS in R." *Journal of Statistical Software* 87 (11): 1–17. https://doi.org/10.18637/jss.v087.i11.

2.2 A minimal example

Once the package is installed, you can load it in the usual way:

```
library(seasonal)
```

The `seas()` function provides the core functionality of the package. By default, `seas` calls the automatic procedures of X-13 to perform a seasonal adjustment that works well in most circumstances:

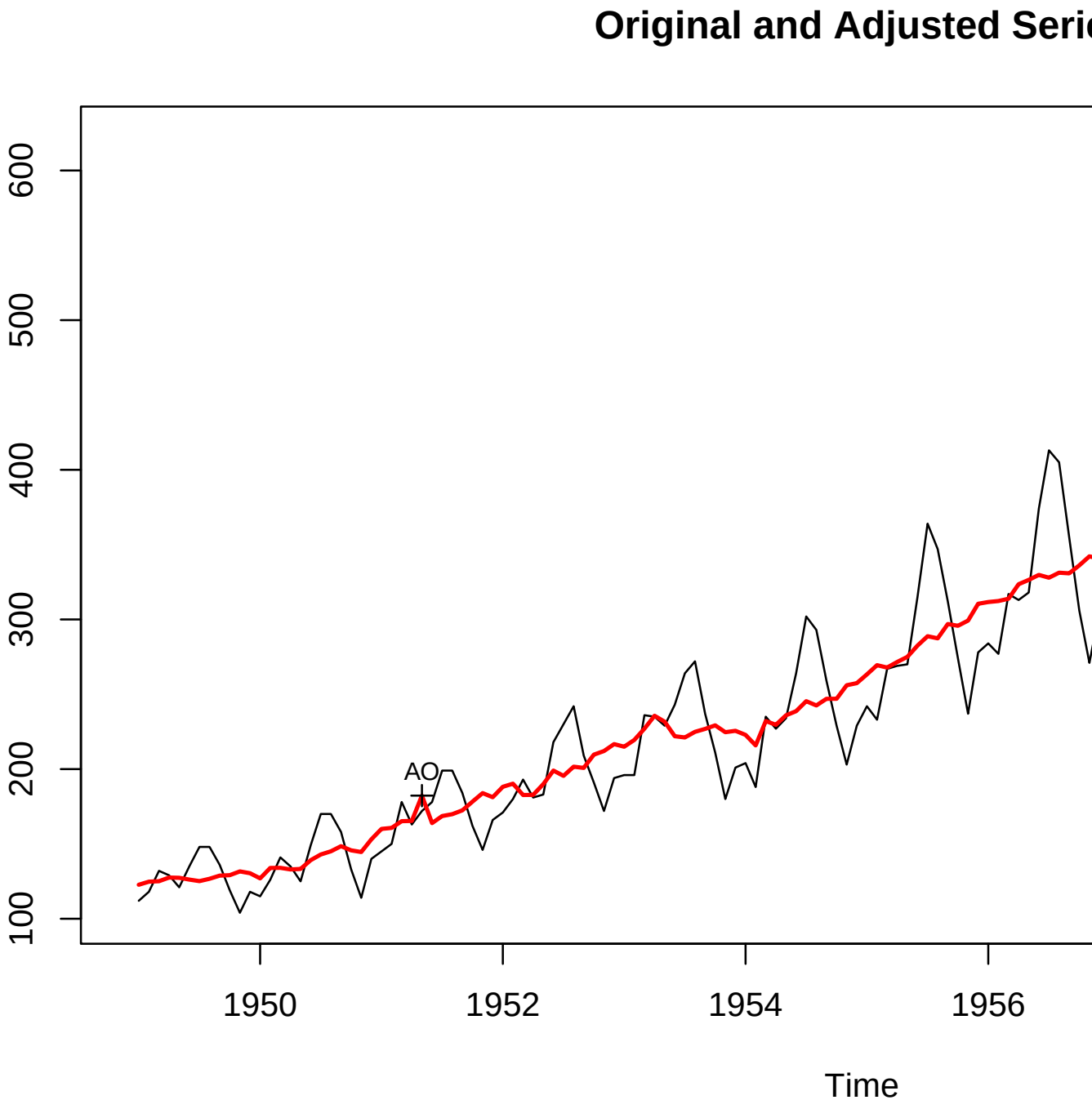
```
seas(AirPassengers)
#>
#> Call:
#> seas(x = AirPassengers)
#>
#> Coefficients:
#>           Weekday          Easter[1]         AO1951.May  MA-Nonseasonal-01
#>          -0.00295            0.01777            0.10016            0.11562
#>    MA-Seasonal-12
#>           0.49736
```

The first argument of seas is a time series of class `ts`. `ts` objects are frequently used in base R and are useful to store monthly, quarterly, or annual data. We restrict our attention to monthly and quarterly series. This is done for two reasons; first is these are the main frequencies handled by X-13, second, this sampling frequency make conceptual understanding of statistical methods, such as linear filters, easier to grasp. The `AirPassengers` example series is included in base R and shows monthly totals of international airline passengers from 1949 to 1960. `seas()` returns a `seas` object that contains the necessary information on the adjustment performed on this time series; we can assign it to a variable:

```
m <- seas(AirPassengers)
```

There are several functions and methods for `"seas"` objects. The final function returns the adjusted series. The plot method shows a plot with the unadjusted and the adjusted series.

```
plot(m)
```


Original and Adjusted Seri
600
500
400
300
200
100
AO
1950
1952
1954
1956
Time

As you can see, the adjusted series is much less volatile than the original one because the seasonal component was removed from the original series. But the adjusted series is not entirely smooth. This is because it still contains the irregular component.

This constitutes a crucial point about seasonal adjustment: It only removes regular, predictable movements, not irregular ones. In the adjusted series, we can see a decrease in airline passengers in 1953 and between 1957 and 1958. These decreases were difficult to discover in the original series.

The summary method displays an overview of the model, very similar to the one produced by other R classes (eg `lm` or `numeric`):

```
summary(m)
#>
#> Call:
#> seas(x = AirPassengers)
#>
#> Coefficients:
#>                      Estimate Std. Error z value Pr(>|z|)
#> Weekday           -0.0029497  0.0005232  -5.638 1.72e-08 ***
#> Easter[1]          0.0177674  0.0071580   2.482   0.0131 *
#> AO1951.May         0.1001558  0.0204387   4.900 9.57e-07 ***
#> MA-Nonseasonal-01  0.1156204  0.0858588   1.347   0.1781
#> MA-Seasonal-12     0.4973600  0.0774677   6.420 1.36e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> SEATS adj.  ARIMA: (0 1 1)(0 1 1)  Obs.: 144  Transform: log
#> AICc: 947.3, BIC: 963.9  QS (no seasonality in final):    0
#> Box-Ljung (no autocorr.): 26.65   Shapiro (normality): 0.9908
```

The summary gives an overview of the adjustment model and provides diagnostics. This book will help you do understand it in more detail. The following section discusses some of the elements and relates them to the chapters in this book.

2.3 Where to go from here

`seas(AirPassengers)` produces a good seasonal adjustment of the airline passengers time series. If you are very new to seasonal adjustment, the automated routines of X-13 and seasonal produce an adjustment that works well in most circumstances.

The command `seas(AirPassengers)` has invoked a large number of *specs* of X-13. *spec* is X-13 slang for a module within the software. X-13 is built on top of twenty specs that perform various subtasks of seasonal adjustment. Some specs are required most of the time (e.g., `regression`), while others are optional (e.g., `seats`) or purely technical (e.g., `spans` shortens the time series in use). Chapter 3 discusses the available specs in more detail.

This book teaches you how to use and fine-tune the individual specs and deal with concrete data problems.

2.3.1 Fundamentals

Specifically, the command `seas(unemp)` has invoked the following fundamental specs – they are involved in most adjustments and are covered in the **first part** of the book:

Transform A decision on initial transformation was made. The automated procedures concluded that a log transformation was made and a multiplicative seasonal adjustment model, rather than an additive model, was estimated. Chapter 4 discusses the choices. Since `transform` is a relatively simple spec, it is a good starting point to familiarize yourself with the spec idea.

Regression An automated model search concluded that `AirPassengers` is best modeled by an (0 1 1)(0 1 1) ARIMA model. Chapter 5 explains what that means and how such a model structure is determined and estimated.

SEATS / X11 Seasonal decomposition is performed by SEATS. SEATS is one of the two options for decomposing a series and is discussed in more detail in Chapter 6. The alternative, X11, is discussed in Chapter 7.

2.3.2 Data issues

The command `seas(AirPassengers)` has also dealt with various data issues, which are covered in the **second part** of the book:

Holiday Significant Easter effects have been found in `AirPassengers` and were removed from the adjusted series. Moving holidays like Easter or Chinese New Year are vital in seasonal adjustment since they may significantly impact the behavior of many time series. For `AirPassengers`, the number of passengers is higher in months with Easter. Moving holiday effects will be discussed in Chapter 8.

Weekday Not every month has the same number of weekdays. Since many activities (such as air traveling) differ between weekends and weekdays, this constitutes another predictable component. In `AirPassengers`, there are fewer passengers on a weekday than during a weekend, and the automated procedures decided to remove the effect. These effects are discussed in Chapter 9.

Outliers Certain data points may be well out of the ordinary. These *outliers* are a problem for the modeling and adjustment process. An automated procedure scanned the series for outliers and found an additive outlier on May 1951. This outlier is shown in the plot above, too. Outliers are discussed in Chapter 10.

Seasonal Breaks The seasonal pattern in `AirPassengers` looks relatively stable. Some time series, however, show abrupt changes in the seasonal pattern. Chapter 11 discusses them and shows how to deal with seasonal breaks.

2.3.3 Additional issues

The **third part** of the book deals with additional issues:

Presence of seasonality While the presence of seasonality in `AirPassengers` is prominent, this is not always the case. If a series has no seasonal pattern, there is no need for a seasonal adjustment. If it is adjusted anyway, the process adds noise to the series and should be avoided. Chapter 12 shows how seasonality can be detected and how to decide whether an adjustment should be made or not.

Annual constraining Usually, a seasonal adjustment may affect the annual values of a time series. In part, this is by design. The number of weekdays may differ between years, so the adjusted annual values may be different too. In part, this may be an artifact of the adjustment process. X-13 offers tools to enforce the annual values of the adjusted series to be the same as the original one. Chapter 13 shows how to constrain annual value and whether it is a good idea.

Indirect vs. direct adjustment Often, a seasonal adjustment may be performed on individual series or on an aggregate of multiple series. X-13 offers tools that let you compare these two possibilities. Chapter 14 discusses the options and helps you to decide which one is better.

2.3.4 Quality assessment

Adjusting a series with the automated procedure is straightforward. But is the resulting series a reasonable adjustment? The **fourth part** helps you to decide between competing seasonal adjustment models.

Quality measures In the lower part, the summary of the adjustment model shows various quality measures: The AICc and BIC information criterion and the QS, the Box-Ljung, and the Shapiro statistic. None of them shows any significance (indicated by one or several stars), which is a good sign. Various quality measures and their interpretation is shown in Chapter 15.

Revisions When comparing seasonal adjustment models, the stability of the model and the series is often an important consideration. One does not want to get a different series with a new data point. X-13 offers tools to analyze revisions. Chapter 16 discusses them and helps you to decide which model to pick.

2.4 References

Part II

Basics

> **Note**
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is complete enough as an education tool and can be used in a course. It needs additional polishing.

This section gets readers familiar with X-13ARIMA-SEATS. It begins by explaining the history and pedagogy of the software (Chapter 3). This leads directly into discussing the principal elements of X-13ARIMA-SEATS.

3 The fundamentals of X-13

> **Note**
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is complete enough as an education tool and can be used in a course. It needs additional polishing. **It is part of the course materials intended for Dec 21, 2022**.

X-13ARIMA-SEATS is organized by *specs*. Specs are software modules that deal with a particular task needed in the seasonal adjustment process. For example, the `transform` spec deals with the initial transformations of the time series. This spec will take care of if a series should be adjusted in logarithms.

3.1 Main specs

Some specs, like the `transform` and the `regression` spec, are used in most seasonal adjustment processes. Others fulfill a particular function and are used only occasionally. For example, the `history` spec allows an analysis of the revision history and is only called for diagnostical purposes. Other specs are mutually exclusive. You can choose `x11` or `seats` to decompose a time series, but not both. Table 3.1 lists the main specs of X-13 and describes what they do.

Table 3.1: Important specs that are used in most seasonal adjustment models

Spec name	What it does	Chapter
`estimate`	Estimates the regARIMA model specified by the `regression` and `arima` specs.	Chapter 5
`arima`	Specifies the ARIMA part of the regARIMA model.	Chapter 5
`regression`	Specification for including regression variables in a regARIMA model.	Chapter 5, Chapter 8, Chapter 9
`automdl`	Specifies the ARIMA part of the regARIMA model using an automatic procedure.	Chapter 5
`outlier`	Specification to perform automatic detection of additive (point) outliers.	Chapter 10
`seats`	Invoke the production of model-based signal extraction using SEATS. Default in the R seasonal package.	Chapter 6
`x11`	An optional spec for invoking seasonal adjustment by the X-11 methodology.	Chapter 7
`forecast`	Specification to forecast and/or backcast the time series given in the series spec using the estimated model.	Chapter 5

3.2 Interactions between specs

X-13 specs interact with each other. For example, once a series is transformed, it is usually passed to the `regression` and `arima` specs, which estimate a regARIMA model. To come up with a good model, it uses the `automdl` spec to determine a good ARIMA model automatically. To correct outlier values,

it collaborates with the `outlier` spec. Once the series is modeled, it is decomposed either by the `seats` or the `x11` spec. Figure 3.1 shows the interaction between the main specs in a typical seasonal adjustment run.

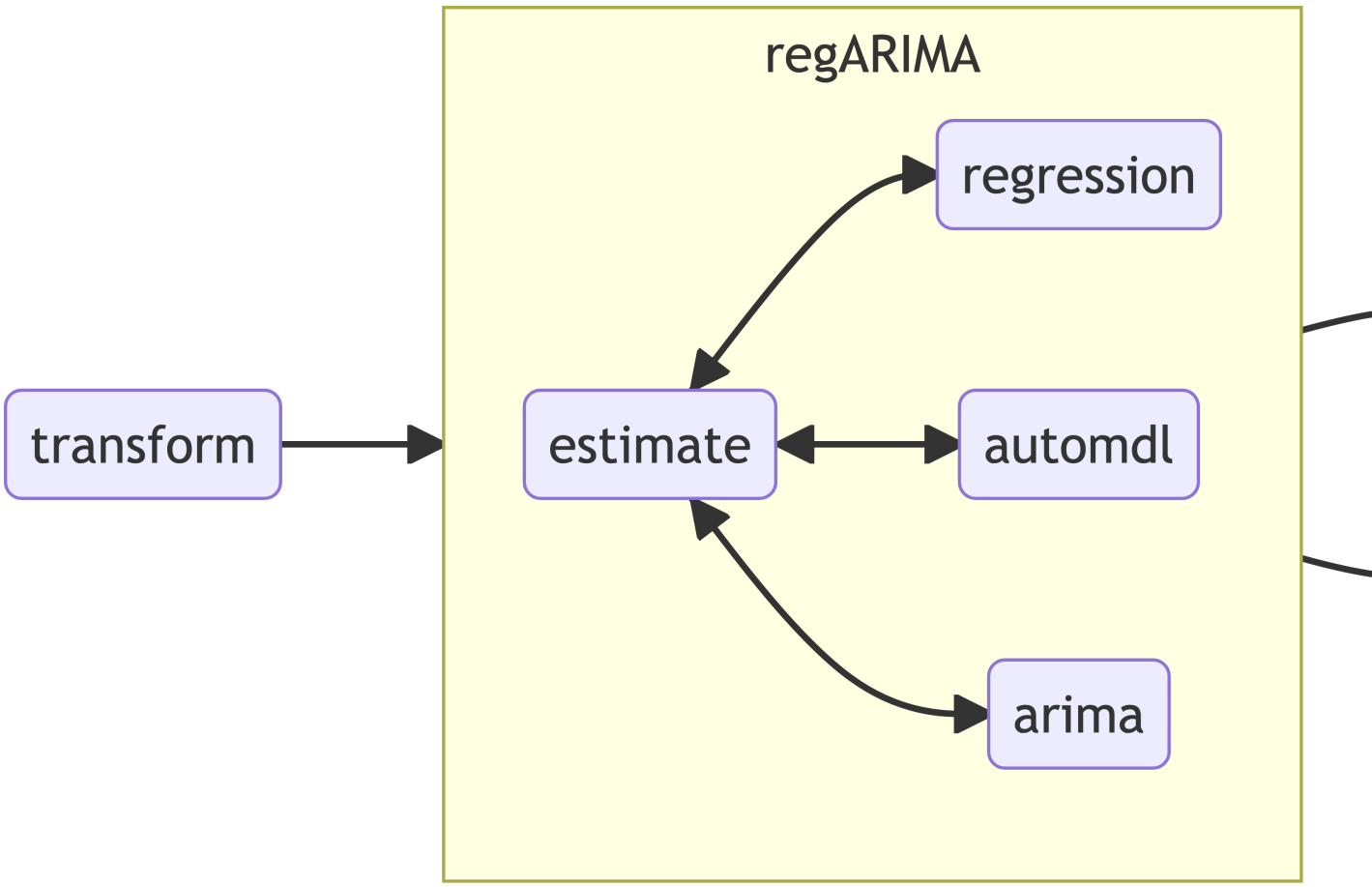


Figure 3.1: Interactions between X-13 specs.

3.3 Specs Arguments

Within specs, there are *arguments*. Spec arguments guide the behavior of the spec. For example, the `function` argument in the `transform` spec can be set to `"auto"`, `"none"`, `"log"`, `"sqrt"`, `"inverse"` or `"logistic"`. The default is set to `"auto"`, which causes an automated model evaluation between `"log"` and `"none"`. There are many other arguments, and the X-13 Manual (US Census Bureau 2017) is the canonical reference. This book will list and explain the frequently used arguments while skipping some of the more exotic ones.

US Census Bureau. 2017. *X-13ARIMA-SEATS Reference Manual*. Version 1.1. Washington, DC, USA: Time Series Research Staff, Center for Statistical Research; Methodology, US Census Bureau. http://www.census.gov/ts/x13as/docX13ASHTML.pdf.

3.4 Addressing specs from R

In the R package seasonal, spec argument combinations can be directly fed to the `seas()` function. For example, to turn off the log transformation in the `AirPassengers` example from Section 2.2, we can specify the following:

```
m_no_log <- seas(AirPassengers, transform.function = "none")
summary(m_no_log)
#>
#> Call:
#> seas(x = AirPassengers, transform.function = "none")
#>
#> Coefficients:
#>                   Estimate Std. Error z value Pr(>|z|)
#> Constant          30.62077    4.60956   6.643 3.08e-11 ***
#> Leap Year         11.32104    3.43088   3.300 0.000968 ***
#> Weekday           -0.90361    0.17787  -5.080 3.77e-07 ***
#> Easter[1]          6.89372    1.80972   3.809 0.000139 ***
#> AR-Nonseasonal-01  0.81929    0.04903  16.709  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> SEATS adj.  ARIMA: (1 0 0)(0 1 0)  Obs.: 144  Transform: none
#> AICc: 993.4, BIC:  1010  QS (no seasonality in final):    0
#> Box-Ljung (no autocorr.):  29.2   Shapiro (normality): 0.984
```

As you can see from the lower part of the summary, transform is now set equal to none. Note that the change in the `transform` argument has also affected the other specs. The ARIMA model is different now, and a leap-year adjustment is performed. We will discuss the working of the transform spec in more detail in the next chapter.

3.5 Less frequently used specs

While the main specs appear in most seasonal adjustment processes, other specs are less often used. Some of them have a diagnostic purpose. The `spectrum` spec, for example, draws and analyses the spectrum of a time series, similar to the R base function `spectrum()`. Other specs are more elaborate. For example, the `history` spec produces a sequence of runs from a sequence of truncated versions of the time series and allows the analysis of potential revisions. The `slidingspans` spec models various parts of the time series and has a similar purpose as `history`. All diagnostics specs are listed in Table 3.3.

Table 3.2: Diagnostic specs

Spec name	What it does	Chapter
`history`	Requesting a sequence of runs from a sequence of truncated versions of the time series for the purpose of creating historical records.	Chapter 16
`slidingspans`	Providing sliding spans stability analysis.	Chapter 16
`identify`	Produce tables and line printer plots of sample ACFs and PACFs.	
`spectrum`	Provides a choice between two spectrum diagnostics to detect seasonality or trading day effects.	
`check`	Produce statistics for diagnostic checking of residuals from the estimated model.	

The `force` and the `composite` spec are special-purpose specs. The former enforces the yearly totals of the seasonally adjusted series to be equal to those of the original series. The latter

allows a comparison of indirect and direct seasonal adjustments. Table 3.3 gives an overview of the special-purpose specs.

Table 3.3: Special purpose specs

Spec name	What it does	
`force`	Allow users to force yearly totals of the seasonally adjusted series to equal those of the original series for convenience.	Chapter 13
`composite`	Obtaining both indirect and direct adjustments of a composite series.	Chapter 14

Finally, a few specs are not covered in this book. Some of them are vintage specs that were important in earlier versions of X-13 but were superseded by other specs. It is generally recommended to use `regression` instead of `x11regression` and `automdl` instead of `pickmdl`. Other specs have a purely technical purpose. For example, the `series` spec provides X-13 with the data, starting date, and frequency. In R, this is handled by seasonal and will not be covered.

Table 3.4: Vintage and technical specs that won't be covered in this book)

Spec name	What it does
`x11regression`	Alternative to `regression`. Can only be used with X11.
`pickmdl`	Alternative to `automdl`. Can only be used with X11.
`series`	Provides X-13 with the data, the starting date and the frequency. In R, this is handled by seasonal and will not be covered.
`metadata`	Specification that allows users to insert metadata into the diagnostic summary file. In R, this is handled by seasonal and will not be covered.

3.6 Main user choices

While we will cover each spec in more detail, this section provides a few examples of frequent user choices. As we saw in the previous chapter, by default, seasonal uses defaults that work well in many circumstances. The following is a non-exhaustive list of deviations from the defaults. The default options of `seas()` are listed as explicit arguments and are discussed in the arguments section of the help page of

3.6.1 Using X11

While `seas()` calls SEATS by default, X11 is often easier to use. To perform a seasonal adjustment on `AirPassengers` with X11, we need to activate the `x11` spec.

```
m_x11 <- seas(AirPassengers, x11 = "")
```

An empty string `""` tells `seas()` to use the spec without an argument. Alternatively, you can also use an empty list, `list()`. If more than one mutually exclusive spec is included in `seas()`, specs are overwritten according to the priority rules shown in Table 3.5

Table 3.5: If more than one mutually exclusive spec is included, specs are overwritten according to priority rules.

Procedure	Priority rules
Model selection	1. `arima` 2. `pickmdl` 3. `automdl`
Adjustment procedure	1. `x11` 2. `seats`

This is why the default SEATS procedure in the introductory example was overwritten by the specification of `x11 = ""`.

3.6.2 Turning off auto modeling

By default, the `automdl` spec finds a good ARIMA model. By specifying the `model` argument of the `arima` spec, the automated modeling is deactivated. Instead of the automatically chosen (0 1 1)0 1 1) ARIMA model, the following estimates an (1 1 0)1 1 0) model.

```
m_arima <- seas(AirPassengers, arima.model = c(1, 1, 0, 1, 1, 0))
summary(m_arima)
#>
#> Call:
#> seas(x = AirPassengers, arima.model = c(1, 1, 0, 1, 1, 0))
#>
#> Coefficients:
#>                     Estimate Std. Error z value Pr(>|z|)
#> Weekday           -0.0029124  0.0004794  -6.076 1.23e-09 ***
#> Easter[1]          0.0167907  0.0067080   2.503   0.0123 *
#> AO1951.May         0.0950587  0.0194363   4.891 1.00e-06 ***
#> AR-Nonseasonal-01 -0.1078564  0.0871940  -1.237   0.2161
#> AR-Seasonal-12    -0.4588948  0.0790634  -5.804 6.47e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> SEATS adj.  ARIMA: (1 1 0)(1 1 0)  Obs.: 144  Transform: log
#> AICc: 951.6, BIC: 968.1  QS (no seasonality in final):    0
#> Box-Ljung (no autocorr.): 35.88 . Shapiro (normality): 0.9937
```

3.6.3 Turning off AIC testing and Outlier detection

By default, `seas()` evaluates the presence of weekday and Easter effects and checks for outliers in the data. Both can be turned off:

```
m_no_auto <- seas(AirPassengers, regression.aictest = NULL, outlier = NULL)
summary(m_no_auto)
#>
#> Call:
#> seas(x = AirPassengers, regression.aictest = NULL, outlier = NULL)
```

```
#>
#> Coefficients:
#>                   Estimate Std. Error z value Pr(>|z|)
#> MA-Nonseasonal-01  0.40181    0.07887   5.095  3.5e-07 ***
#> MA-Seasonal-12     0.55695    0.07626   7.304  2.8e-13 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> SEATS adj.  ARIMA: (0 1 1)(0 1 1)  Obs.: 144  Transform: log
#> AICc: 987.4, BIC: 995.8  QS (no seasonality in final):    0
#> Box-Ljung (no autocorr.): 28.04   Shapiro (normality): 0.9886
#> Messages generated by X-13:
#> Warnings:
#> - At least one visually significant trading day peak has been
#>   found in one or more of the estimated spectra.
```

In practice, many spec argument combinations can be extracted via the `static()` functions, which will be demonstrated in the next chapter. Alternatively, the seasonalview package offers a graphical user interface that allows you to click various spec argument combinations.

```
m <- seas(AirPassengers)
view(m)
```

3.7 References

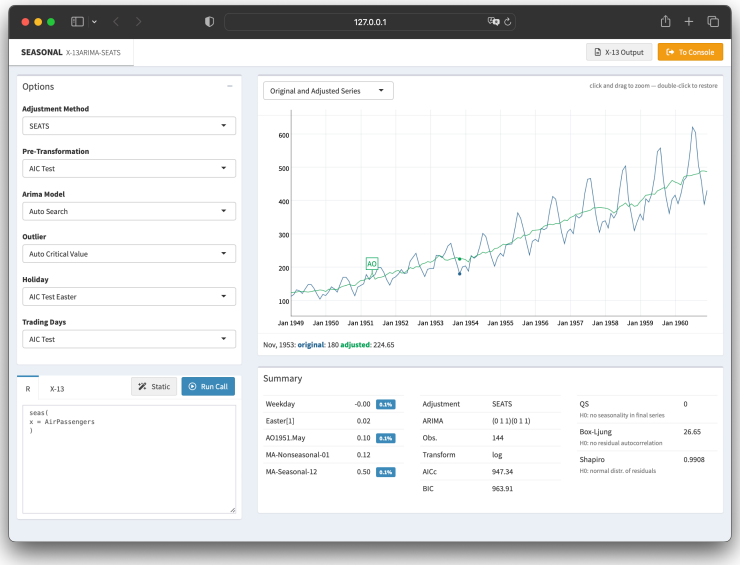


Figure 3.2: Manipulating spec argument combinations in the seasonalview graphical user interface

4 Transform

> **Note**
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is complete enough as an education tool and can be used in a course. It needs additional polishing. **It is part of the course materials intended for Dec 21, 2022**.

One of the first choices to make when modeling a time series is whether or not it needs to be transformed before modeling. Two types of transformation types typically occur within X-13, *prior modifications* and *transformations*.

4.1 Prior modifications and transformations

The first is a prior modification. A prior modification scales each observation for known fixed effects. These effects can be well-known and established, such as the length of a period or leap-year effects, or they can be more subjective such as a modification for a workers' strike. We will see how this comes into the regARIMA model in Chapter 5. We can think of prior-modification factors as events or corrections made to your data that are fixed throughout the adjustment process. These prior modification factors can also be permanent (default) or temporary. Permanent modifications are excluded from the final seasonal adjustment. Temporary modifications are removed while calculating seasonal factors but added back to the seasonally adjusted series.

The second type is a nonlinear transformation applied to the observations. This is typically a choice between logarithmic

transform and no transformation but for modeling can be any power of the Box-Cox transformation.

```
m_none <- seas(AirPassengers, transform.function = "none")
m_log  <- seas(AirPassengers, transform.function = "log")
```

4.2 Additive and multiplicative adjustment

As you remember from @sec-introduction, Seasonal adjustment decomposes a time series into a **trend**, a **seasonal**, and an **irregular** component. Algebraically, the fundamental identity of seasonal adjustment looks like this:

$$Y_t = T_t + S_t + I_t. \tag{4.1}$$

We seek the decompose our observed series Y_t into a trend T_t, a seasonal S_t, and an irregular I_t component. The formulation above is *additive*, i.e., the trend, the seasonal, and the irregular component sum up to the observed series. The goal of seasonal adjustment is to subtract the seasonal component:

$$A_t = Y_t - S_t.$$

For example, an observed value of 100 with a seasonal factor of 3.2 would result in a seasonally adjusted value of 100 - 3.2 = 96.8.

Alternatively, the decomposition can be *multiplicative*:

$$Y_t = T_t \cdot S_t \cdot I_t \tag{4.2}$$

I.e., the observed series is the product of the trend and the seasonal and irregular components. Since these are factors, the goal of seasonal adjustment is to remove seasonality by dividing by the seasonal factor.

$$A_t = \frac{Y_t}{S_t}.$$

For example, an observed value of 100 with a seasonal factor of 1.08 would result in a seasonally adjusted value of 100 / 1.08 = 92.59259. Hence for multiplicative models, values of $S_t > 1$ decrease the observed value, and $S_t < 1$ increase it. Keep this in mind when viewing seasonal factors such as in the function `monthplot(m)`.

For a multiplicative adjustment, it is sufficient to take logarithms of the initial series and re-transform the results after the decomposition. The `transform` spec takes care of this.

4.3 Model choice

X-13 has a built-in statistical test to decide between log and no transformation. The choice is made by comparing the AICc[1] value of an ARIMA (0 1 1)(0 1 1) model fit, or a user-specified ARIMA model, to the log-transformed series and the original series.

[1] With small sample sizes, a standard AIC test may select models with too many parameters. AICc tackles this problem by correcting for sample size.

For most practical purposes, this is an effective choice and can be left to the program to decide. If your series has negative values, it can not be log-transformed, and automatic selection is performed. Other restrictions on the allowed transformations exist, but these situations are rare. We can look at the results of the transformation tests by looking at specific statistics. The `udg()` function provides access to a large number of diagnostical statistics.The `qs()` function and the `AIC()`, `BIC()` and `logLik()` methods are wrappers that use `udg()` to access some specific diagnostical statistics.

```
m <- seas(AirPassengers)
udg(m, c("aictest.trans.aicc.nolog", "aictest.trans.aicc.log"))
#> aictest.trans.aicc.nolog   aictest.trans.aicc.log
#>                1021.1919                 987.3845
```

We see the AICc for log transformation is lower and hence selected. We saw this in **?@sec-getting-started**, were the summary of the seasonal object, `summary(m)`, has told us that `Transform: log`. The same infor transformation can also be found in many other places such as the HTML output with

`out(m)` or the udg with argument name `aictrans` such as `udg(m, "aictrans")`.

> **X-13 HTML Output**
>
> X-13ARIMA-SEATAS has a built-in HTML output that offers an extensive summary of a seasonal adjustment process. In R, this can be accessed using the `out()` function. E.g.,
>
> ```
> out(m)
> ```

The choice between `log` and `none` changes the type of seasonal decomposition that will occur and, hence, your interpretation of the seasonal factors. With no transformation, X-13 will perform an additive seasonal adjustment specified in Equation 4.1. If log transformation is selected, X-13 will perform a multiplicative adjustment as specified in Equation 4.2.

4.4 Transform options

The `transform` spec controls these options. Some primary options within this spec are

Spec option	Use	Example values	default
`transform.function`	Transform function	`none`, `log`, `auto`	`none`
`transform.aicdiff`	adjust tolerance of AIC test for log transform	`0.0`, `3.0`, `-4.5`	`-2.0`
`xtrans`	Prior adjustment factor	`seas(m, xtrans = y)`	`NULL`

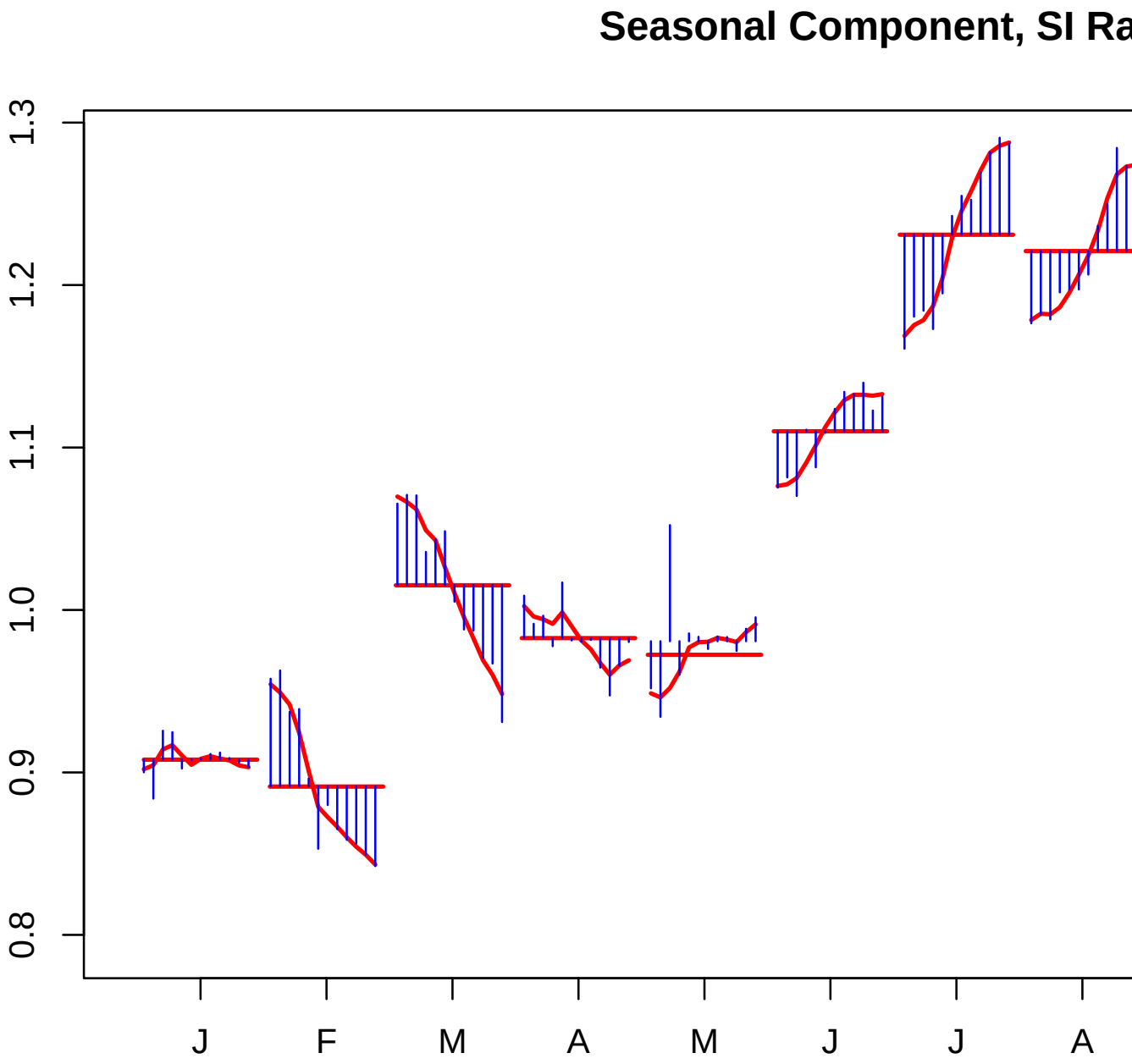
4.5 Case Study: `AirPassengers`

An increasing variance, also known as *heteroskadasticity* is one sign of requiring a logarithmic transform. We have already seen that this is present in `AirPassengers`. Let's verify that the automatic transformation identifies this.

```
transformfunction(m)
#> [1] "log"
```

This is also a good place to get our first look at the seasonal factors. The `monthplot()` method offers a convenient way to look at these:

```
monthplot(m)
```


Seasonal Component, SI Ra
1.3
1.2
1.1
1.0
0.9
0.8
J
F
M
A
M
J
J
A

Like the R base `monthplot()` function that can be applied on any time series (also on quarterly time series!), this groups time series data by months. If you look at the January (`J`), entry, the blue bars show the evolution of the detrended data from 1949 to 1960. The red bar shows the average seasonal factor over these years. The smooth red lines show the seasonal factors as estimated by the model.

As you can see from the plot, there are more passengers during the summer months and fewer in the winter. The seasonal factors change over time. The summer peak becomes more pronounced in later years, while the local peak in February and March disappears over time.

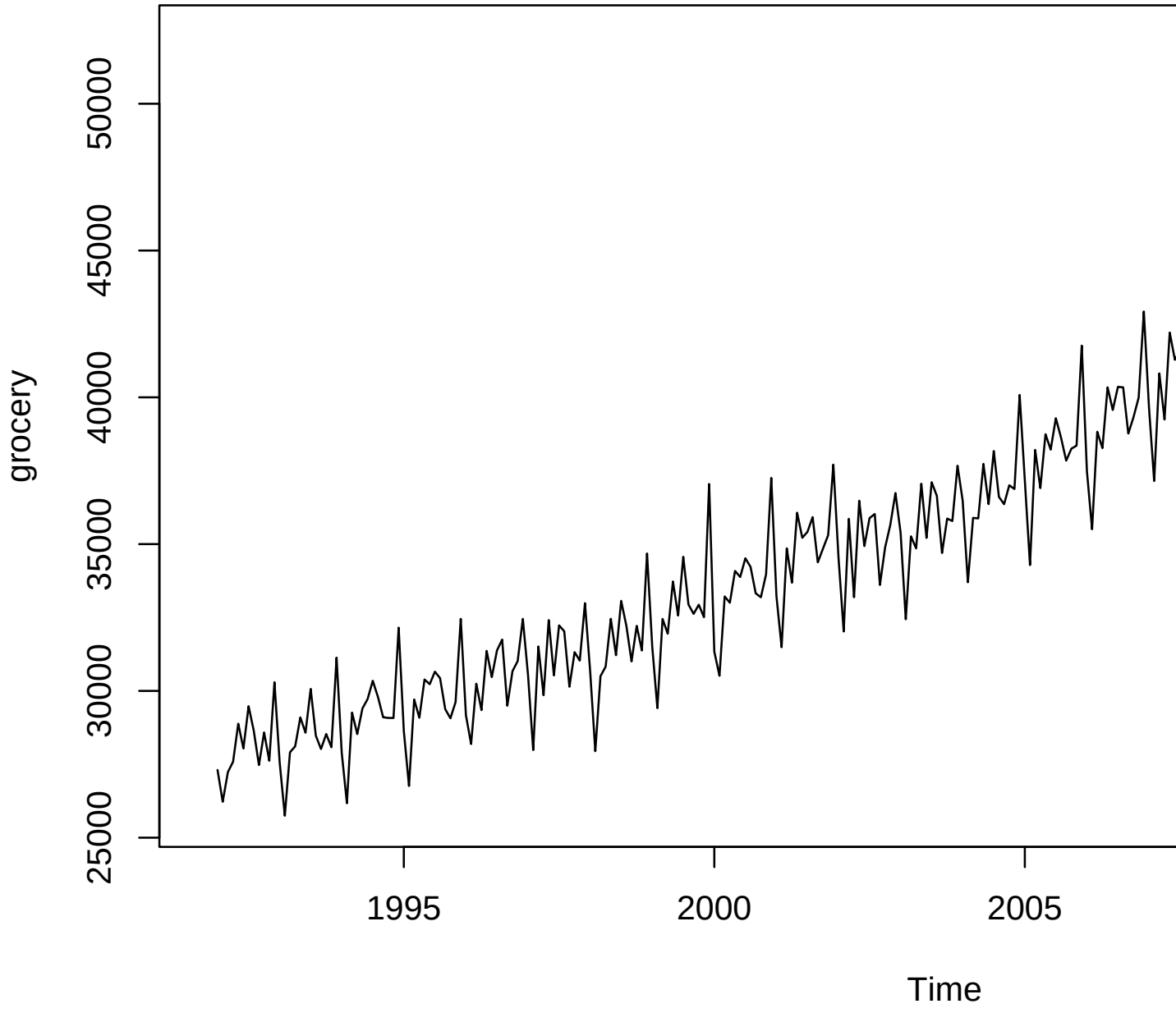
If you want to extract the seasonal factor directly into R, you can use the `series()` function:

```
series(m, "seats.seasonal")
```

4.6 Case Study: More difficult decision

Consider the situation where you are trying to decide on transform choices for monthly retail grocery store data. The series `grocery` is part of the *seasonalbook* package.

```
library(seasonalbook)
plot(grocery)
```


grocery
25000
30000
35000
40000
45000
50000
1995
2000
2005
Time

Visual inspection of the series shows no immediate reason to think we need to perform a logarithmic transform. There is possible seasonal heteroskadasity which could be mitigated by taking logs. Perform an X-11 adjustment with all the defaults of *seasonal*

```
m <- seas(grocery, x11 = "")
udg(m, c("aictest.trans.aicc.nolog", "aictest.trans.aicc.log"))
#> aictest.trans.aicc.nolog   aictest.trans.aicc.log
#>                 4202.960                 4201.042
```

This is interesting since the AICc for no transformation is lower than the AICc for log transform.

```
transformfunction(m)
#> [1] "log"
```

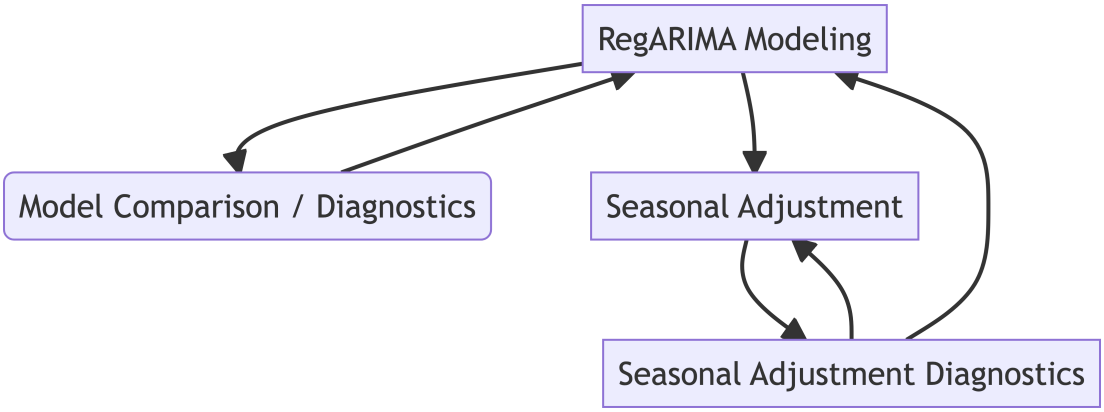
The default value for transform.aicdiff is -2, meaning the program slightly prefers log transform, and the difference between the AICc values must exceed 2. In this situation, the difference between the AICc values is -1.917597. Suppose you were to change this option to `transform.aicdiff = 0`, then the program selects no transform.

```
m2 <- seas(grocery, x11 = "", transform.aicdiff = 2)
transformfunction(m2)
#> [1] "none"
```

5 regARIMA Model

> **Note**
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter should be readable but needs polishing. **It is part of the course materials intended for Dec 21, 2022**.

An important part of the X13 procedure is regARIMA modeling. It sits above seasonal adjustment at the top of the flow chart describing progress towards an adequate adjustment. As with all statistical modeling, the process to find the best model is iterative in nature. This is represented as a loop at the left of our flow chart. An analyst can use statistical tools and subject matter expertise to fine tune and create an overall regARIMA model that is satisfactory to be used and proceed to the seasonal adjustment step of our work flow.



Depending on if you ultimately choose to perform a SEATS or X-11 seasonal adjustment, your usage of a regARIMA can dif-

fer. Both seasonal adjustment methods will use the regARIMA model to forecast the series in order to apply symmetric filters and extract components. (but I thought SEATS was model-based? What do you mean SEATS uses filters? More on this in Chapter XXX). SEATS will use the ARIMA model to derive components for the trend, seasonal and irregular terms. The regARIMA model is also the place to include exogenous information or regressors into your seasonal adjustments. Regressors such as holiday effects, additive outliers, levels shifts, or any user-defined effect can be included in your regARIMA model. This can be thought of as solely a utility to forecast extend your series or a detection method to do inference. In practice it is useful to use regARIMA modeling to answer questions such as, "Does my series have trading day effects?" or "Does this outlier effect my results?". Another reason for regARIMA modeling is to include exogenous regression variables in our analysis. These include, but are not limited to, moving holiday effects, trading day, outliers and level-shifts.

5.1 SARIMA model

As the name implies, there are two components that one needs to understand when fitting a regARIMA model; namely regression and ARIMA. Furthermore, the ARIMA part is made up by a differencing order and the stochastic ARMA portion. In this chapter, we try to break these three components down to the most fundamental ingredients without an overly technical exposition. Essentially, providing readers with enough information about each topic to understand the rest of this book and go off and perform satisfactory seasonal adjustment. The interested reader is encouraged to find material devoted to each of these components separately to more fully understand them.

ARIMA is an acronym describing the three parts of the modeling paradigm. AR = autoregressive, I = integrated (differenced), and MA = moving average. The prefix auto or "self", explains the AR portion perfectly. We model the current observation with lagged values from the past. This is illustrated with the classic autoregressive model of order 1:

$$Y_t = \phi Y_{t-1} + a_t$$

where Y_t is the observed time series, ϕ is a coefficient to be estimated and $\{a_t\}$ is an uncorrelated sequence of errors similar to that of standard linear regression. This model is notated AR(1). If instead of a single lag we used p lags, the model would be and AR(p) and have structure:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + a_t$$

where now we have p coefficients $\phi_1, \phi_2, \ldots, \phi_p$ to be estimated.

The moving average part of ARIMA model is similar in notation and reflects the number of lagged values of the error sequence should be included. For example, an MA(1) model with coefficient parameter θ is:

$$Y_t = a_t + \theta a_{t-1}$$

.

Note that instead of doing self-regression we include past values of the unobserved errors in the model at time t. If instead of a single lag we wanted q lags of the past error terms, we would have an MA(q) model:

$$Y_t = a_t + \theta_1 a_{t-1} + \theta_2 a_{t-2} + \cdots + \theta_q a_{t-q}$$

.

When we combine these two ideas we can model Y_t with p lagged values of itself together with q lagged values of the unobserved errors. Together it makes an ARMA(p, q) model, one of the fundamental ingredients to the regARIMA model. For a practitioner the automatic modeling done in X13 is often sufficient to find an appropriate value for p and q and hence a well fitting ARMA model. If it more important from a seasonal adjustment perspective to correctly specify the differencing and regression variables in your overall regARIMA model.

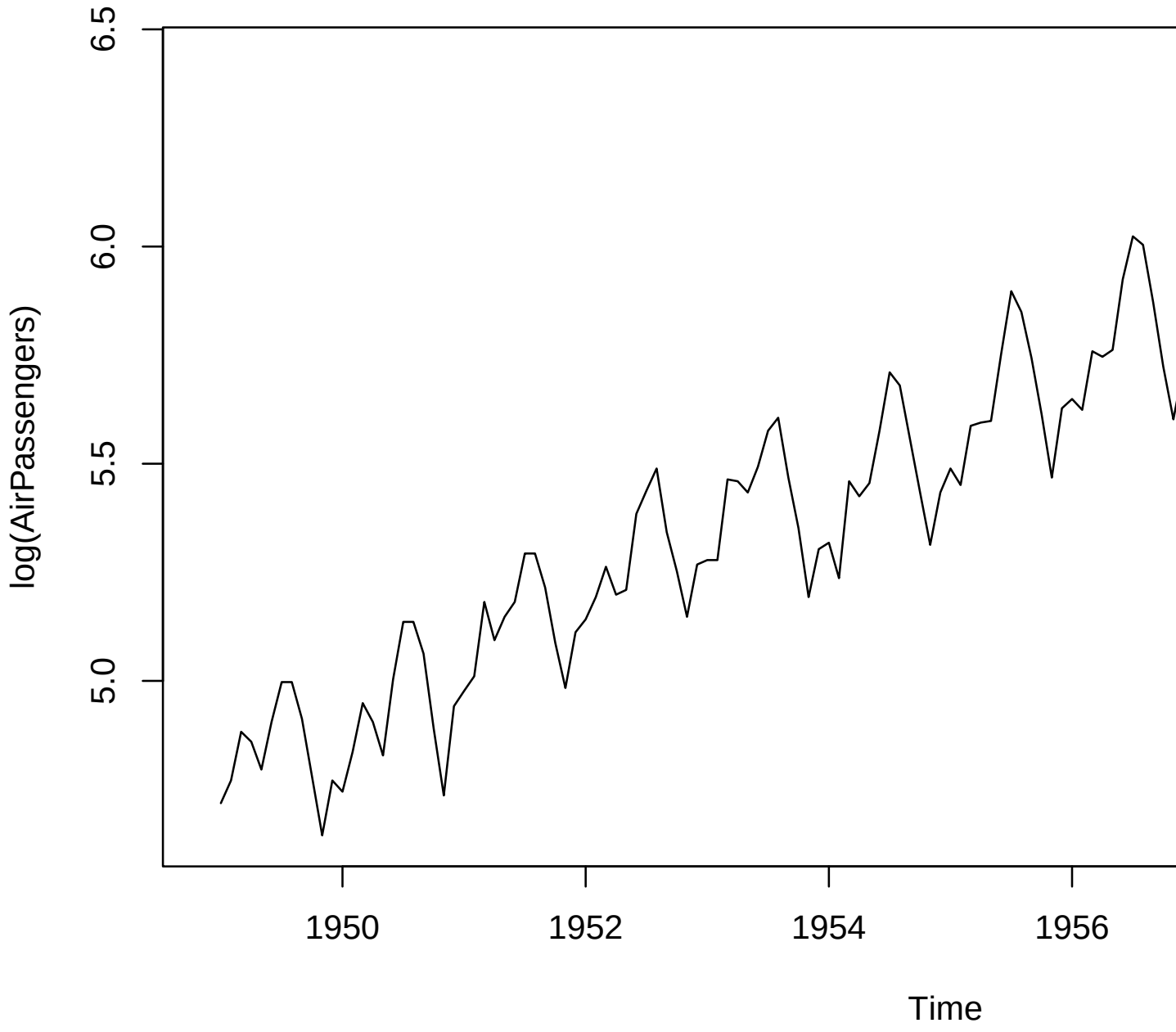
5.2 Differencing

ARMA models work best for stationary time series. This means the mean does not depend on time (such as increasing trend) or have a correlation structure that changes. Many techniques could be used to take a non-stationary time series and transform it to stationarity, one ubiquitous method is differencing. There is a famous results that states if you difference your series k times it will remove a polynomial trend of degree k. Essentially, if you observe a time series with a linear trend then first differencing will remove the trend. If a time series has quadratic trend (polynomial of order 2) then differencing twice will remove that trend. A similar phenomenon can happen at seasonal lags and often a time series will also require seasonal differencing to reduce it to stationary. The order of differencing, also called the intgreation order, for the non-seaosnal and seasonal parts of our model are usually notated as d and D respectively. When we bring the integration order together with the stocastic model specification we have the notation

$$\text{SARIMA} \quad \underbrace{(p, d, q)}_{\text{non-seasonal}} \quad \underbrace{(P, D, Q)}_{\text{seasonal}}.$$

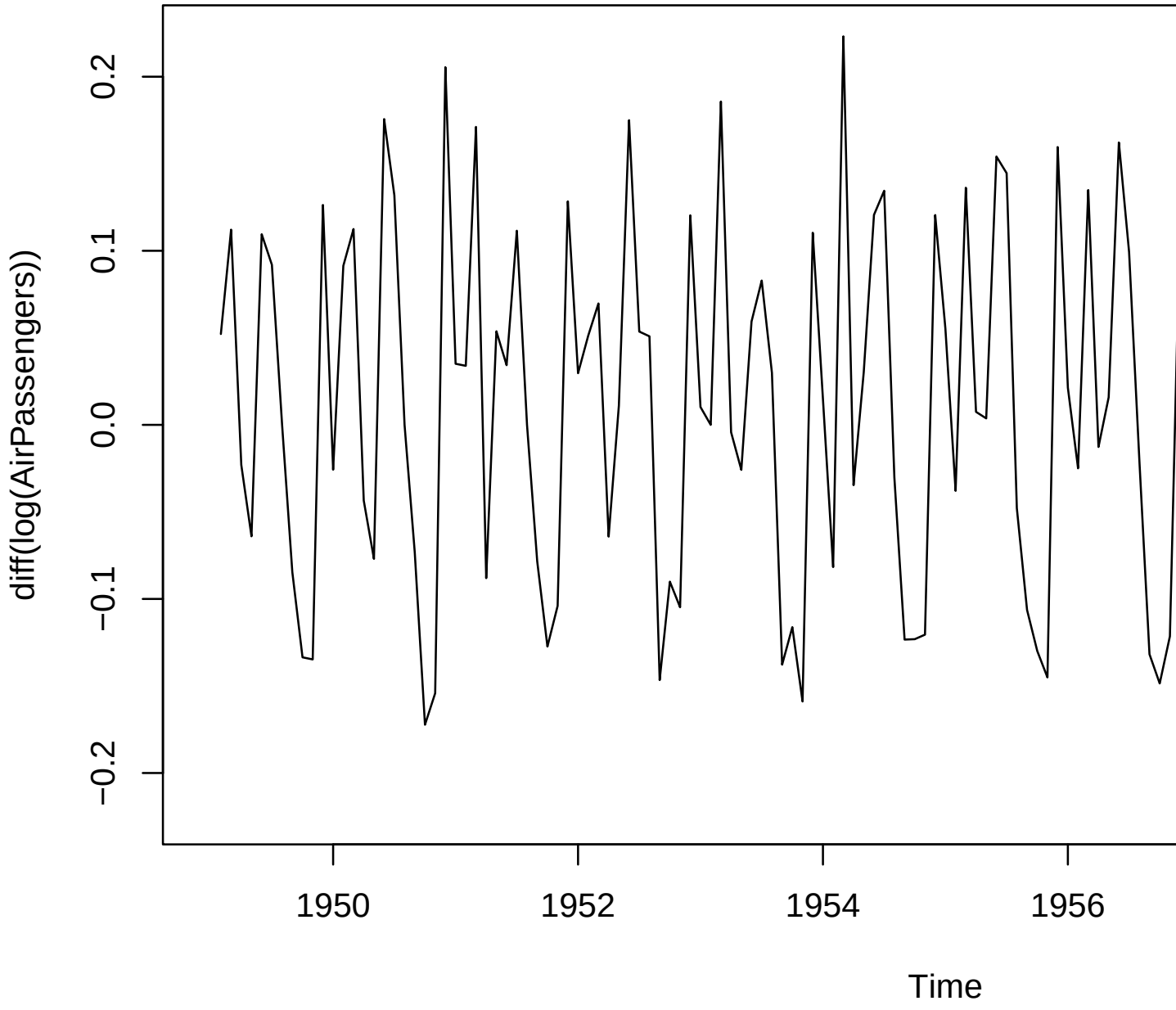
This can be seen easily with an example. Consider the log transformed `AirPassengers` series.

```
plot(log(AirPassengers))
```


log(AirPassengers)
6.5
6.0
5.5
5.0
1950
1952
1954
1956
Time

We see a clear increasing trend and seasonal pattern. Let's call the observed series X_t. We can difference the series to make $Y_t = \Delta X_t = X_t - X_{t-1}$. A plot of Y_t looks like

```
plot(diff(log(AirPassengers)))
```

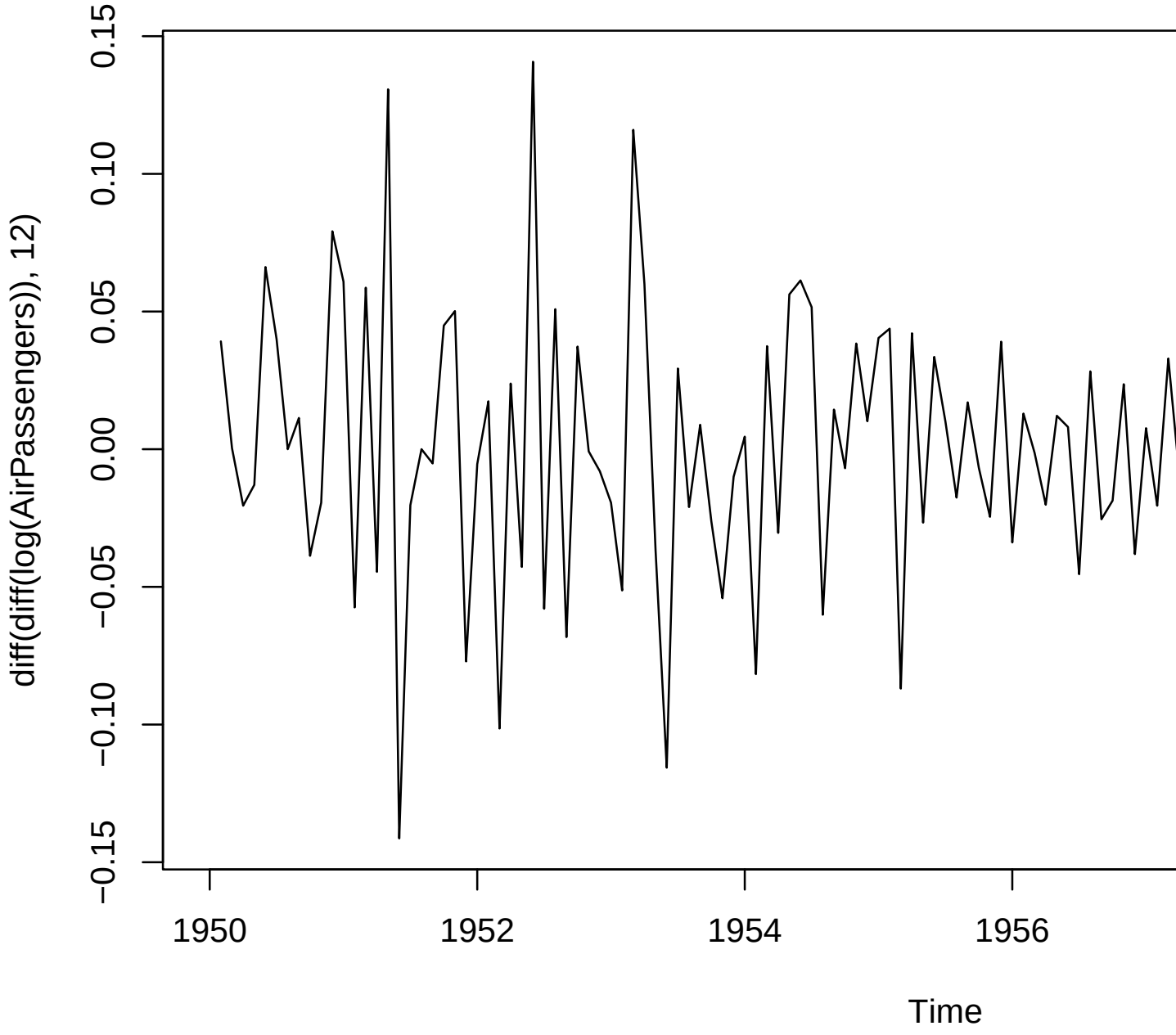

diff(log(AirPassengers))
0.2
0.1
0.0
−0.1
−0.2
1950
1952
1954
1956
Time

The trend has been removed however some seasonal trend (strong seasonal patterns) still exist. Apply seasonal differencing to the already first differenced series Y_t:

$$Z_t = Y_t - Y_{t-12}$$

A plot of Z_t:

```
plot(diff(diff(log(AirPassengers)), 12))
```


diff(diff(log(AirPassengers)), 12)
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
1950
1952
1954
1956
Time

Here we can see that both the original linear trend and seasonal pattern are removed and what is left is a stationary process that can adequately be modeled with an SARMA(0, 1)(0, 1) model. When you bring in the integration (differencing) order of one for the seasonal and non-seasonal components, we are left with the model named after this exact time series! The so called "airline model' ' is the SARIMA(0, 1, 1)(0, 1, 1) and the terminology came to popularity via Box and Jenkins, "Time Series Analysis, Forecasting and Control" textbook.

5.3 Fitting SARIMA (optional)

Here we present a very oversimplied way to start to understand what values of p, P, q and Q you can investigate for your time series of interest. Recall that earlier it was mentioned that using automatic model identification is sufficient for most to get an adequate seasonal adjustment. Hence, this is simply for the interested reader to begin to gain additional intuition into the stochastic structures involved in their series and the types of structures the automatic modeling procedures look at. One of the main tools in a time series analyist tool box is the autocorrelation function (ACF). This is a function that returns the correlation between observations h time units apart throughout the entire sample. So for $h = 2$ this means looking at the correlation between the pairs $(X_1, X_3), (X_2, X_4), (X_3, X_5), ...$ Then a way to build a SARIMA model is to match the sample ACF and the theoretical ACF of a given model. The main point distinguishing an AR(p) and MA(q) is how their theoretical ACF behaves. An AR(p) will have ACF the has exponential decay as h increases. For example, an AR(1) ACF is

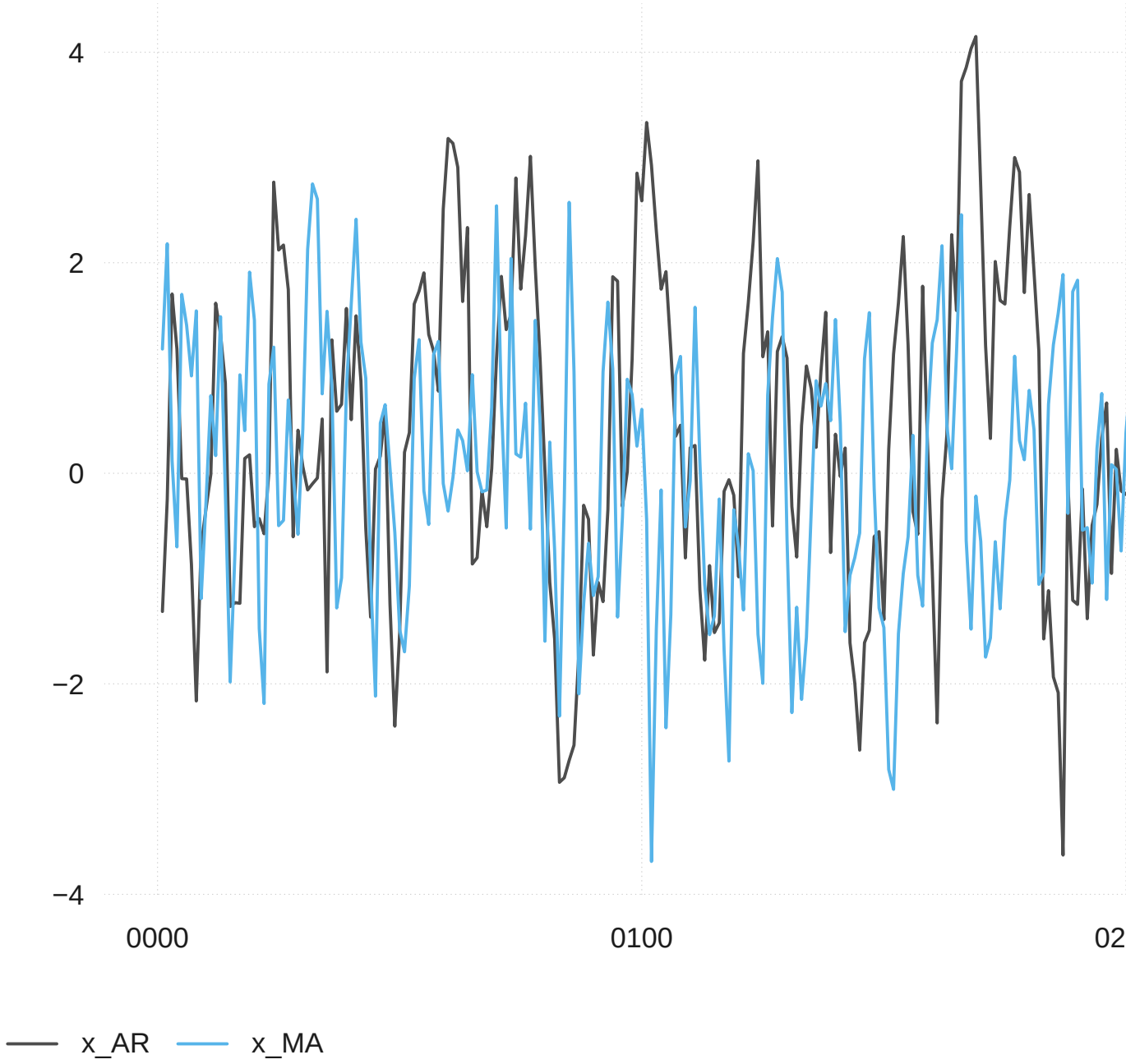
$$\rho(h) = \phi^h$$

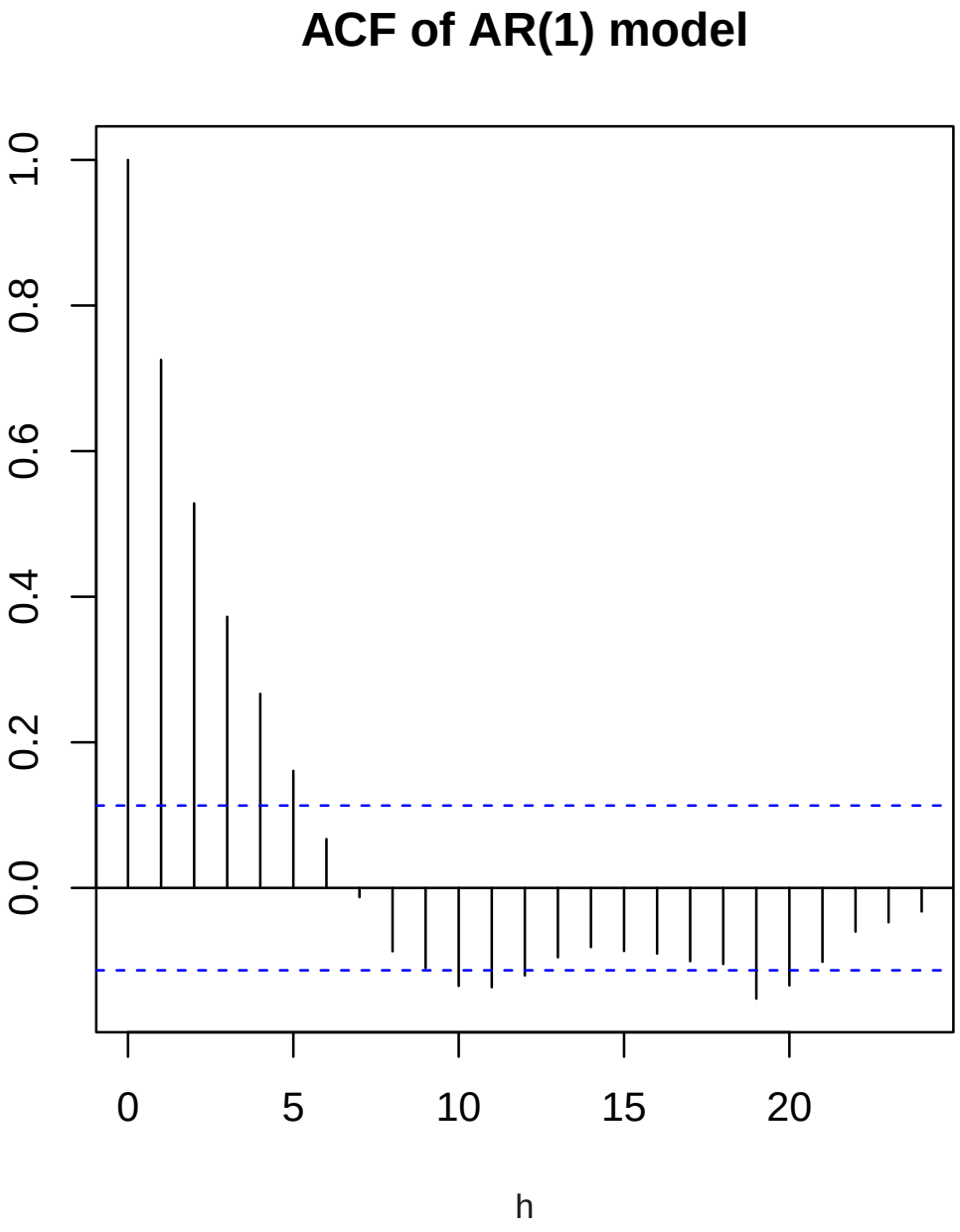
An MA(q) models ACF will be non-zero for the first q lags and then cutoff to zero thereafter. The ACF of an MA(1) is

$$\rho(h) = \begin{cases} 1 & h = 0 \\ \frac{\theta}{1+\theta^2} & h = 1 \\ 0 & \text{otherwise} \end{cases}$$

In practice of course the difference between decay and cut-off can be nebulous to detect but the interested reader is encouraged to explore the `arima.sim()` function the look at the sample ACF with the `acf()` function. As you increase the sample size it will converge to the theoretical ACF value and you can start to see the structures just discussed.

```
x_AR <- arima.sim(model = list(ar = .75), n = 300)
x_MA <- arima.sim(model = list(ma = .75), n = 300)
tsbox::ts_plot(cbind(x_AR, x_MA))
op <- par(mfrow = c(1, 2), mar = c(5, 2, 4, 2))
acf(x_AR, xlab = "h", main = ""); title("ACF of AR(1) model")
acf(x_MA, xlab = "h", main = ""); title("ACF of MA(1) model")
par <- op
```


4
2
0
−2
−4
0000
0100
02
x_AR
x_MA


ACF of AR(1) model
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
h

5.4 Regression

We have discussed SARIMA modeling (both the SARMA and differencing), now we see how exogenous regression variables come into play.

The regARIMA model takes the form

$$f\left(\frac{Y_t}{D_t}\right) = \beta' \mathbf{X}_t + Z_t.$$

Here Y_t is the observed time series. The function f represents a transformation, most commonly used is the log transform ie $f(x) = \log(x)$. D_t is any intervention that has taken place prior to any transformation or modeling. This intervention is usually subjective and customized for individual series on an as-needed basis. For example, if a soybean farmer strike occurred and the soybean export series suffered for its duration. This type of event might adversely affect the seasonal adjustment filters and automatic model identification routines and can be mediated as an initial step. If no transformation or intervention is needed the model form is:

$$Y_t = \underbrace{\beta' \mathbf{X}_t}_{\text{Regression}} + \underbrace{Z_t}_{\text{ARIMA}} .$$

The regression variables appear in the columns of the design matrix $\mathbf{X}_t$ and Z_t is an ARIMA process. This last assumption on Z_t is what distinguished a regARIMA model from more classic linear models and multiple linear regression where error terms are assumed uncorrelated.

In order to achieve a suitable seasonal adjustment it is important to get the regARIMA model correct. For most dataset the built in automatic modeling features of the X13 program will be suitable to detect a reasonable model. This can be used as a starting point for more rigorous regARIMA model development or used as the final regARIMA modeling choice for your seasonal adjustment needs. We evoke automatic model identification through the XXX spec. The default behavior of the R seasonal package is XXX which includes automatic model identification.

Automatic and manual model choice

As an aside, the general rule is to not use automatic modeling in production. This mean, if you are going to include seasonal adjustment as part of a large scale data processing that occurs regularly (say monthly), then it is not advisable to have automatic model identification run every month. Instead, an alternative process, is to run automodel once and then fix the model choice in the XXX spec file. This does not need to be done manually since the `static()` function from the seasonal package can do this for you.

Outlier Type	Automatic Detection Available?
Additive outliers (AO)	Yes (default)
Level shifts (LS)	Yes (default)
Temporary level shifts (TL)	Yes
Temporary changes (TC)	No
Ramps (RP, QI, QD)	No
Seasonal outliers (SO)	No

5.5 Case Study: AirPassengers

Consider the default seasonal adjustment:

```
library(seasonal)
m <- seasonal::seas(AirPassengers, x11 = "")
print(m$spc$automdl)
#> $print
#> [1] "bestfivemdl"
print(m$spc$arima)
#> NULL
```

Notice the value `NULL` indicates no ARIMA model is specified and the returned arguments for the automdl spec indicate it is active during the X13 run.

```
seasonal::udg(m, "automdl")
#>          automdl
#> "(0 1 1)(0 1 1)"
```

Indicates that automatic modeling identified the (0 1 1)(0 1 1) model as the best choice. If we want to hardcode this model for subsequent runs, and turn off automatic model identification, this can be done via

```
m_call <- seasonal::static(m)
#> seas(
#>   x = AirPassengers,
#>   x11 = "",
#>   regression.variables = c("td1coef", "easter[1]", "ao1951.May"),
#>   arima.model = "(0 1 1)(0 1 1)",
#>   regression.aictest = NULL,
#>   outlier = NULL,
#>   transform.function = "log"
#> )
m2 <- eval(m_call)
```

There are many options you can modify when searching for outliers in your series. Some of the most practical options to start your exploration are the *type*, *critical value* and *span* that you would like to search.

Here is an example of using span to limit the outlier search to the last few years of a series:

```
m_span <- seas(AirPassengers,
  outlier.types = c("ao", "ls", "tc"),
  outlier.critical = 4.0,
  outlier.span = "1958.jan, ")
summary(m_span)
#>
#> Call:
#> seas(x = AirPassengers, outlier.types = c("ao", "ls", "tc"),
#>     outlier.critical = 4, outlier.span = "1958.jan, ")
#>
#> Coefficients:
```

```
#>                    Estimate Std. Error z value Pr(>|z|)
#> Weekday           -0.002644   0.000604  -4.377 1.20e-05 ***
#> Easter[1]          0.021321   0.008395   2.540  0.01110 *
#> MA-Nonseasonal-01  0.235404   0.083756   2.811  0.00495 **
#> MA-Seasonal-12     0.543743   0.074644   7.284 3.23e-13 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> SEATS adj.  ARIMA: (0 1 1)(0 1 1)  Obs.: 144  Transform: log
#> AICc: 965.3, BIC: 979.2  QS (no seasonality in final):    0
#> Box-Ljung (no autocorr.): 28.26   Shapiro (normality): 0.9829 .
```

```
m_nospan <- seas(AirPassengers,
  outlier.types = c("ao", "ls", "tc"),
  outlier.critical = 4.0)
summary(m_nospan)
#>
#> Call:
#> seas(x = AirPassengers, outlier.types = c("ao", "ls", "tc"),
#>     outlier.critical = 4)
#>
#> Coefficients:
#>                     Estimate Std. Error z value Pr(>|z|)
#> Weekday           -0.0029497  0.0005232  -5.638 1.72e-08 ***
#> Easter[1]          0.0177674  0.0071580   2.482   0.0131 *
#> AO1951.May         0.1001558  0.0204387   4.900 9.57e-07 ***
#> MA-Nonseasonal-01  0.1156204  0.0858588   1.347   0.1781
#> MA-Seasonal-12     0.4973600  0.0774677   6.420 1.36e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> SEATS adj.  ARIMA: (0 1 1)(0 1 1)  Obs.: 144  Transform: log
#> AICc: 947.3, BIC: 963.9  QS (no seasonality in final):    0
#> Box-Ljung (no autocorr.): 26.65   Shapiro (normality): 0.9908
```

The default critical value is set based on the length of the outlier span. Notice the MA-Nonseasonal-01 value when comparing `m_span` with `m_nospan`. We see the choise of span, and ultimately the choise to include an outlier in your model can have a dramatic effect on the estimated regARIMA parameters.

6 SEATS

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.
> **It is part of the course materials intended for Jan 21, 2023**.

SEATS = "Signal Extraction in ARIMA Time Series' '

Both X-11 method and SEATS assume the observed time series is a function of latent (unobserved) components.

$$X_t = S_t + N_t$$

where S_t is a seasonal component and N_t is a non-seasonal component.

Usually the non-seasonal component is broken down further into trends, cycles, transitory, etc.

$$X_t = T_t + S_t + I_t$$

Ultimately, both X-11 and SEATS estimate the components T_t, S_t, I_t by passing moving average filters over the forecast extended series.

X-11 Filters:

- finite set of empirically developed moving average filters
- fixed filtering seen as easier to use (less statistical machinery)

SEATS filters:

- specifies stochastic models for unobserved components
- derives seasonal adjustment filters from these models
- infinite number of possible filter choices
- requires more statistical machinery

Sometimes SEATS includes a transitory component in its decomposition:

$$X_t = T_t + S_t + R_t + u_t$$

The transitory component captures short, erratic behavior that is not white noise, sometimes associated with awkward frequencies.

- The variation from the transitory component should not contaminate the trend or seasonal, and removing it allows SEATS to obtain smoother, more stable trends and seasonal components.
- In the final decomposition, the transitory and irregular components are usually combined.
- SEATS does not always estimate a transitory component

6.1 Quick refresher on ARIMA models and notation

The remaining of the SEATS section will heavily rely on the autoregressive and moving-average operators $\phi(B)$ and $\theta(B)$ where $BX_t = X_{t-1}$.

If X_t follows and ARIMA(p, d, q) model:

$$\phi(B)X_t = \theta(B)a_t$$

$$(1 - \phi_1 B - \cdots - \phi_p B^p)(1 - B)^d X_t = (1 + \theta_1 B + \cdots \theta_q B^q)a_t$$

For example, in an ARIMA(2,0,1) we are modeling X_t as:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + a_t + \theta_1 a_{t-1}$$

and all model information is contained in $\phi(B)$ and $\theta(B)$. Moreover, for any specified $\phi(B)$ and $\theta(B)$ that satisfy certain causality criteria there exists a unique Wold decomposition

$$\phi(B)X_t = \theta(B)a_t$$

$$X_t = \frac{\theta(B)}{\phi(B)}a_t = \Psi(B)a_t = \sum_{k=0}^{\infty} \psi_k a_{t-k}$$

6.2 SEATS Assumptions

- The linearized series can be represented by an ARIMA model which captures the stochastic structure of the series (Linearized series = series with regression effects removed)
- After differencing each with the ARIMA's differencing polynomial, the components are orthogonal (uncorrelated)

SEATs decomposes the autoregressive polynomial by its roots associating them with different latent components. For example, roots near seasonal frequencies are associated with the seasonal component and roots near zero are associated with the trend component.

$$\phi(B) = \phi_T(B) \cdot \phi_S(B) \cdot \phi_R(B).$$

Hence we have,

$$X_t = \frac{\theta(B)}{\phi(B)}a_t = \frac{\theta_T(B)}{\phi_T(B)}a_{T,t} + \frac{\theta_S(B)}{\phi_S(B)}a_{S,t} + \frac{\theta_R(B)}{\phi_R(B)}a_{R,t} + u_t$$

If the spectra of all components in non-negative the decomposition is admissible, SEATS finds admissible models for components

$$\phi_T(B)T_t = \theta_T(B)a_{T,t}$$

$$\phi_S(B)S_t = \theta_S(B)a_{S,t}$$
$$\phi_R(B)R_t = \theta_R(B)a_{R,t}$$

Problem: Infinite number of models that yield the same aggregate. The choices differ in how white noise is allocated among the components. *Solution:* Canonical Decomposition: SEATS uses the method of Pierce, Box-Hillmer, Tiao and Burman:

- Put all the white noise into the irregular components
- Maximize the variance of the irregular
- Minimizes the variance of the stationary transforms of the other components

This is called the *Canonical Decomposition.*

7 X11

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.
> **It is part of the course materials intended for Dec 21, 2022**.

One of the two main methods available in X-13ARIMA-SEATS to extract trend and seasonal components is the X-11 method. This is a nonparametric procedure that works by passing moving-average filters over the data to extract intended components.

In order to use symetric moving average filters at the end of the time series (current value), a regARIMA model is used to forecast extend the series. This RegARIMA model is where users can test for or specify outliers, trading day and moving holiday regressors in their adjustment. The forecast extended series is then used to filter.

Additionally, X-11 has a built in extreme value procedure included. This procedure identifies extremes and replaces. This results in a robust procedure that can automatically choose filters and identify extreme values without much user intervention. All that needs to be evoked beyond the default `seas()` call is to turn on the X11 spec option.

```
m <- seas(AirPassengers, x11 = "")
summary(m)
#>
#> Call:
#> seas(x = AirPassengers, x11 = "")
```

```
#>
#> Coefficients:
#>                      Estimate Std. Error z value Pr(>|z|)
#> Weekday           -0.0029497  0.0005232  -5.638 1.72e-08 ***
#> Easter[1]          0.0177674  0.0071580   2.482   0.0131 *
#> AO1951.May         0.1001558  0.0204387   4.900 9.57e-07 ***
#> MA-Nonseasonal-01  0.1156204  0.0858588   1.347   0.1781
#> MA-Seasonal-12     0.4973600  0.0774677   6.420 1.36e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> X11 adj.  ARIMA: (0 1 1)(0 1 1)  Obs.: 144  Transform: log
#> AICc: 947.3, BIC: 963.9  QS (no seasonality in final):    0
#> Box-Ljung (no autocorr.): 26.65   Shapiro (normality): 0.9908
#> Messages generated by X-13:
#> Warnings:
#> - Visually significant seasonal and trading day peaks have
#>   been found in one or more of the estimated spectra.
```

Before further discussion about the details of the X-11 process, let us see what happened during this modeling run…

When using the x11 spec you can change the length of the filter used for the trend and seasonal components with the `trendma` and `seasonalma` arguments respectively. Additionally, `sigmalim` will control the amount of extreme value adjustment that is done during the seasonal adjustment.

7.1 Additive and multiplicative (again)

The X-13ARIMA-SEATS development was highly motivated to study economic time series. As such, the default seasonal adjustment mode is multiplicative due to most seasonal economic time series displaying seasonal fluctuations that increase and decrease along with the level of the time series.

If your series does not have this feature then additive adjustment might be more appropriate. This can be changed in the `mode` argument of the x11 spec. For example, `seas(x,`

`x11.mode = 'add')` will perform an additive x11 run. There exist two other models for decomposition, pseudo-additive and log additive. These are less common than additive and multiplicative models and are not the focus of this text. If your series has some extremely small values in certain months (quarters) then pseudo-additive models could be worth further investigation. It has been observed that when multiplicative seasonal adjustment produces more extreme values in conjunction with small seasonal factors then pseudo-additive adjustment should be explored. NEED TO SHOW USERS HOW TO VIEW THEIR EXTREME VALUES - D8.B TABLE DESIGNATIONS NEXT TO OBSERVATIONS. A good reference on the subject is Baxter (1994).

7.2 Filter length

The X11 spec also allows users to control the length of the trend and seasonal moving average filters used during the adjustment. Generally speaking, longer filters imply a more stable seasonal component and shorter filters a more changing seasonal pattern. Of course, a longer filter will use more data for the caluclation of components at each time point. This is an important observation and understanding it might help a user decide on a short or long filter. Since longer filters use more data there tend to be smaller revisions when a new data point is added. However, there will be revisions to data values further back.

A shorter filter is just the opposite, they tend to produce smaller revisions but they do not extend as far back into the series. If a filter is not choosen by the user then automatic filter selection is used. To understand the length of a filter let's look at the (finite) number of choice available in during an x11 adjustment. Table 7.1 shows the different filters available for the seasonal component and the trend component.

Table 7.1: Filters available in X11

Value	Description
`s3x1`	3×1 moving average
`s3x3`	3×3 moving average

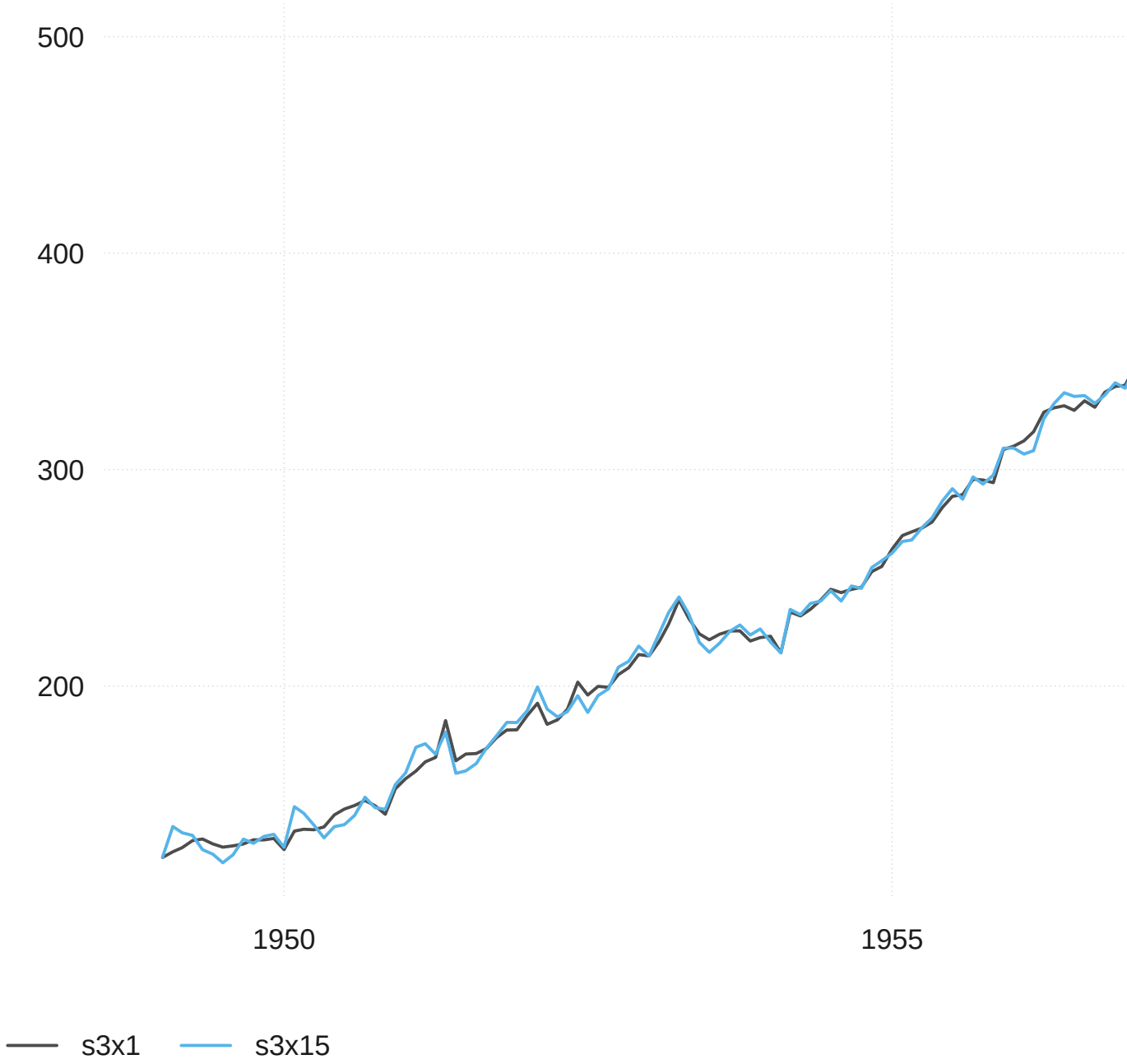
Value	Description
`s3x5`	3×5 moving average
`s3x9`	3×9 moving average
`s3x15`	3×15 moving average
`stable`	Stable seasonal filter. A single seasonal factor for each calendar month or quarter is generated by calculating the simple average of all the values for each month or quarter (taken after detrending and outlier adjustment).
`x11default`	A 3×3 moving average is used to calculate the initial seasonal factors in each iteration, and a 3×5 moving average to calculate the final seasonal factors.

7.3 Extrem value

> ⚠ Related concept: Outlier
>
> FIXME: explain how this is related to transform.

7.4 Case study

```
tsbox::ts_plot(
  s3x1 = predict(seas(AirPassengers, x11.seasonalma = "s3x1")),
  s3x15 = predict(seas(AirPassengers, x11.seasonalma = "s3x15"))
)
```


500
400
300
200
1950
1955
s3x1
s3x15

Part III

Data Problems

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

In part III we look at more in-depth at practical issues with seasonal adjustment. The focus is on concrete solutions to each situation presented. Each subsection will prominently feature a case study dedicated to each problem.

8 Irregular holidays

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.
> **It is part of the course materials intended for Jan 21, 2023**.

- Why should we adjust for holiday effects
- Easter adjustment
- User defined adjustments (Chinese New Year, Diwali)
- Case Study: How to adjust for Ramadan (which is connected with some additional challenges)

9 Trading days

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

- Why should we adjust for trading day effects
- Seven or two coefficient trading day
- Using country specific calendars
- Case Study: Movie tickets (or another series with very clear trading day effects)

10 Outliers

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

- Why care about outliers?
- Additive outliers, level shifts, temporary changes

11 Seasonal breaks

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

- Why to care about seasonal breaks?
- Detection of seasonal breaks
- Correction for seasonal breaks

Part IV

Other Issues

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

Part IV investigates more holistic issues that practitioners face. The main focus is to give classical methodology to answer their problems. Since these types of issues can be highly specialized, we concentrate on known solutions to the topics.

12 Presence of seasonality

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

Should a series be seasonally adjusted at all?

X-13AS removes seasonality from series, even if a series is not seasonal from the beginneing. If a series is not seasonal, the resulting series may be bad.

Fortunatly, X-13 contains a few tests that help users to decide if a series is is seasnal or not.

Before appying X-13AS ist may be necessary to descide if the series is seasonal.

12.0.1 Avalabele Tests

X13 offers has several formal checks:

- qs test
- ids
- m7

The *ids* test is closely connected to *m7*, but the *QS* test is quite different. Which tests are preferrable, and how should a user descide if the tests are not aligned?

12.0.2 ids test

http://www.ons.gov.uk/ons/guide-method/method-quality/general-methodology/time-series-analysis/guide-to-seasonal-adjustment.pdf

> From ONS 18.2 A general criterion for existence of seasonality
>
> Empirical research showed that the most appropriate test for seasonality is the "Combined test for the presence of identifiable seasonality", given after table D8 of the output. In particular, one of the following statements will always appear:
>
> 1. IDENTIFIABLE SEASONALITY PRESENT
> 2. IDENTIFIABLE SEASONALITY PROBABLY NOT PRESEN
> 3. IDENTIFIABLE SEASONALITY NOT PRESENT
>
> It is recommended that a series is adjusted in the first two cases and not adjusted in the last one. However there are two cases where one might need to deviate from this practice:

This is the ids test shown below

12.0.3 Case Study

13 Annual constraining

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

- Should the annual values be restrained?
- How to use the force spec

14 Indirect vs direct adjustment

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

- Should the subcomponents of a series be adjusted separately?

Part V

Quality assessment

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

This section focuses on diagnostic tools for seasonal adjustment. This will be written as a stand-alone section as well as a continuance of prior sections. The idea here is that many readers may be interested in checking the quality of their adjustments but not need help performing it.

15 Quality measures

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

- What is a good seasonal adjustment?
- M statistics
- Other statistics available in X13

16 Revisions

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

- How to measure revisions?
- Should a model be re-estimated each period?
- How to use the slidingspan and history spec

Part VI

The future of seasonal adjustment

> **!** Important
>
> You are reading an early draft of *Seasonal Adjustment in R*. This chapter is currently a dumping ground for ideas, and we don't recommend reading it.

This short section outlines the future projects in the seasonal adjustment field. Daily or multiple seasonal adjustment plays a major role here. Ideally, examples of how to solve these problems are given.

- Daily adjustment
- Multivariate seasonal adjustment
- Other methods

Status of the book

Table 16.1: Current status of sections

section name	status	due date
A minimal example	course_complete	2022-12-21
Overview of the software	course_complete	2022-12-21
Transform	course_complete	2022-12-21
X11	drafting	2022-12-21
regARIMA Model	polishing	2022-12-21
Holidays	drafting	2023-01-21
SEATS	drafting	2023-01-21