Course Outline: Time Series Data in R

Harrison Brown

2022-05-16

Contents

Welcome

Welcome to the course outline for *Time Series Data in R*! This course offers methods and workflows for analyzing and interpreting time series data, an overview of when, why, and how to use time series data, and various utilities and packages in R that are beneficial to analysts.

By the end of this course, students will have the skills to:

- Interpret and understand time series plots
- Import ts data to create and manipulate `ts` objects from the `stats` package
- Understand why time series data is fundamentally different than non-ts data.
- Analyze time series data with plots
- ?Intro to Wavelet analysis?

Chapter 1

Introduction to time series data

1.1 Lesson: What is Time Series Data

- Learning Objective: Learner will be able to understand why and how TS-data differs from non-temporal data
- LO: What kinds of inferences and results can be obtained from TS-data
- LO: Converting to and from time-based data formats, such as `numeric`, `Date`, and `POSIXct` classes
  - Functions: `as.Date()`, `lubridate::`, etc.

1.2 Lesson: How to Interpret Time Series Data

- LO: Learner will understand how to interpret attributes of a basic time series plot
- LO: "Signal and Noise" in the context of TS data
- Introduction to Stationarity: Most real-world data are not stationary and require additional steps to work with

1.3 Lesson: Components of Time Series Data

Chapter 2

Creating and Manipulating Time Series

2.1 `ts` Class

2.2 Missing Values

```
d <- c('2001-01-01', '2001-01-02', '2001-01-04', '2001-01-05')
d <- as.Date(d)
date_range <- seq(min(d), max(d), by = 1)
date_range[!date_range %in% d]
```

```
## [1] "2001-01-03"
```

Chapter 3

Rolling and Expanding Windows

3.1 Rolling Window

- Moving lower and upper bound

3.1.1 Data

3.1.2 Calculating a Rolling Window

Chapter 4

Introduction to Forecasting in R

4.1 Methods for Forecasting

4.1.1 Exponential Smoothing

Chapter 5

Capstone Exercise

The final exercise for this course involves performing a time series analysis on real-world data: Carbon Dioxide concentration at the Mauna Loa Observatory, from early 1959 to Present. You'll go through the process of imputing missing values, testing for stationarity, decomposing the time series, and adjusting for seasonality. The goal for this exercise is a plot showing the seasonal-adjusted time series, which shows the "overall trend" in the data over the last few decades.

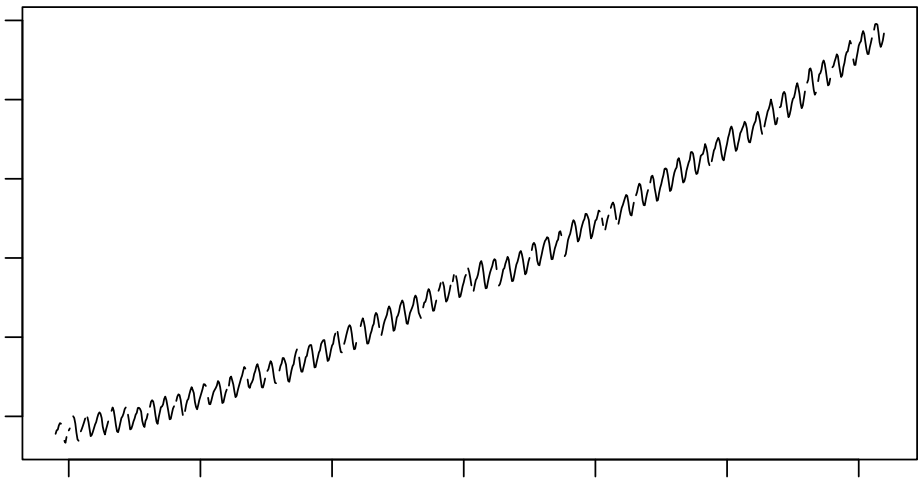
5.1 Importing the Data

```
# The following libraries are included for you

library(tidyverse)
library(zoo)
library(forecast)
library(tseries)
# Sample data from the Mauna Loa Observatory
# https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_mm_mlo.csv

# Data is already pre-processed as a `ts` object. It contains missing values, so
# we'll need to impute those!
co2 <- readRDS("data/missing.Rds")
```

5.2 Visual Checks

1. Plot your co2 data and see if the data is stationary, non-stationary (time-dependent), and seasonal or non-seasonal. Use the Augmented Dickey-Fuller test to determine stationarity.

```
plot.ts(co2)
```



```
adf.test(co2)
```

```
## Error in adf.test(co2): NAs in x
```

Looks like we have some missing values! Let's try and impute those to fill in the gaps:

2. Impute missing values with the `na.approx()` function from `zoo`:

```
co2 <- co2 %>%
  zoo::na.approx()

adf.test(co2)
```

```
## Warning in adf.test(co2): p-value greater than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  co2
## Dickey-Fuller = -0.12404, Lag order = 9, p-value = 0.99
## alternative hypothesis: stationary
```

Based on the statistic and p-value, the data is non-stationary.

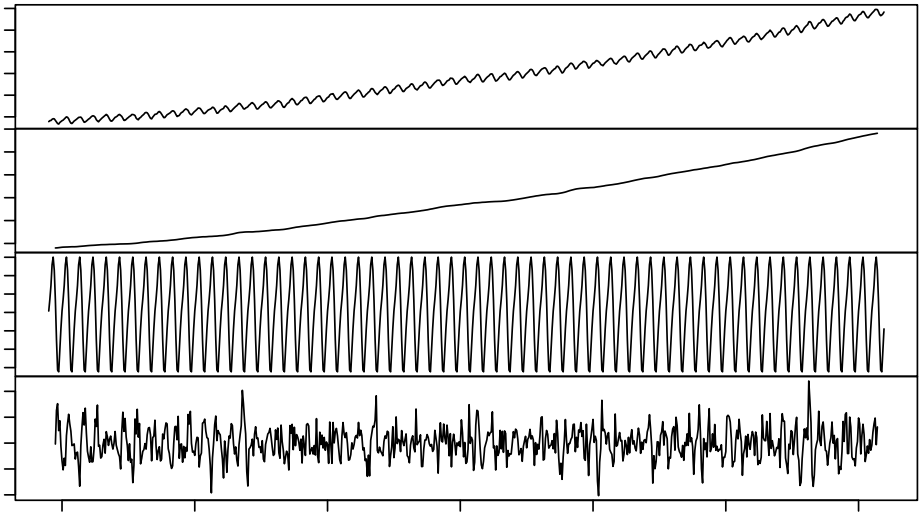
5.3 Decomposing the Time Series

We're interested in the seasonal, remainder, and specifically, trend components. We want to know what the data looks like when adjusted for seasonality. To do

this, we need to decompose our time series into its ETS components. Then, we can remove the seasonal component.

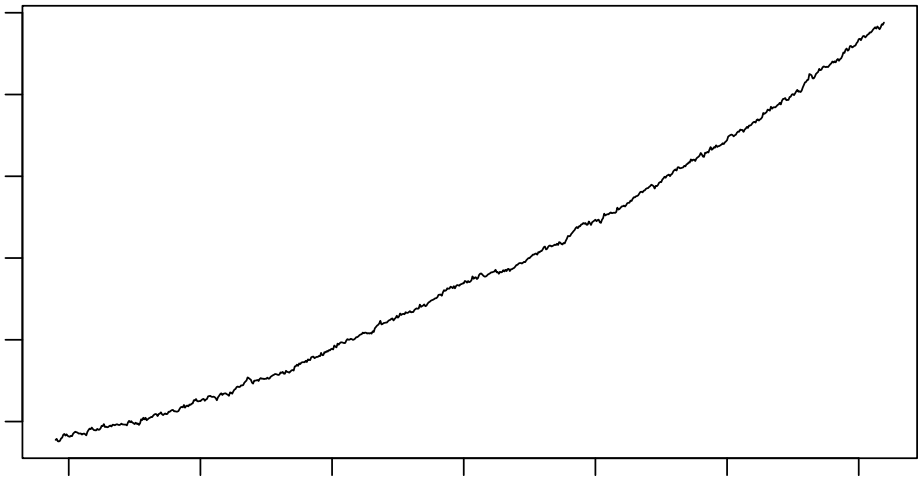
3. Decompose the time series and plot the resulting `decomposed.ts` object:

```
co2_decomp <- co2 %>%
  decompose()

plot(co2_decomp)
```



4. Then, adjust for seasonality, and plot the results. Label the y-axis as "CO2 Concentration", and title the plot "Seasonally-adjusted CO2 Concentration":

```
co2_decomp %>%
  seasadj() %>%

  # If we want to include a rolling window, it's really easy to do with
  # zoo::rollapplyr()

  #rollapplyr(FUN = mean, width = 12) %>%
  plot(ylab = "CO2 Concentration", main = "Seasonally-adjusted CO2 Concentration")
```



Voila! We now have the overall trend, plus remainder component, of CO2 concentrations. By removing the seasonality of data, we can better assess the year-to-year differences in our data, and can reduce the effect of intra- and inter-year cycles present in the data.

5.4 Example Code

This is the code given in the exercise itself

Step 1:

```
plot(___)

___.test(co2)
```

Step 2:

```
co2_impute <- co2 %>%
  __.___()

adf.test(co2_impute)
```

Step 3:

```
co2_decomp <- co2_impute %>%
  ___()

plot(___)
```

Step 4:

```
co2_decomp %>%
  ____() %>%
```

```
  plot(ylab = "___", main = "___")
```