Course Outline: Time Series Data in R

Harrison Brown

2022-05-23

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.

Course Outline: Manipulating Time Series Data in R

This course will introduce learners to working with time series data in R. Learners will explore how to store and format data in date and time objects as well as how to manipulate time series datasets through subsetting, indexing, and extraction. Examples of time series data across a variety of fields in business and science should be discussed. The course will cover summarization, frequency, missing data, resampling, and comparison techniques as well as window functions for both rolling and expanding windows.

Packages Used:

- `base` and `stats` (default libraries, but I wanted to name them explicitly)
- `zoo`
- `lubridate` (so far, only for the `lubridate::year()` function (Lesson 3.3))

Functions are written in `code text` below each **Learning Objective**

Chapter 1: Introduction to Time Series Data

Lesson 1.1: *What is Time Series Data*

- **Learning Objective:** Learner will be able to understand the foundations of time series data: rather than just analyzing a variable at different points in time, time series analysis studies *how* that variable changes with time.

Lesson 1.2: *Interpreting a Time Series*

- **Learning Objective:** Learner will be able to interpret a time series graph, understanding the x- and y-axes and identifying trends and periods at an introductory level.

Lesson 1.3: *Temporal data classes in R*

- **Learning Objective:** Learner will be introduced to different formats for temporal data in R, such as the `Date`, `numeric`, and `character` classes:
- e.g.: 2022-01-30, `19022`, and "2022-01-30" share the same information, but in different formats
- **Learning Objective:** Learners will be able to check classes of data stored as vectors or as columns in a dataframe or tibble; as some formats appear identical, it's important to understand the class of the data you're working with:
  - `class()`

Lesson 1.4: *Converting between data classes*

- **Learning Objective:** Learners will be able to convert between classes in R, such as converting a `character` vector/column to a `Date` vector/column:
  - `as.Date()`
  - `as.numeric()`
  - `as.character()`

Chapter 2: Time Series objects in R

Lesson 2.1: *How does R store Time Series Data?*

- **Learning Objective:** Learners will be introduced to `ts` objects in R, and how they differ from objects like vectors or data frames in how they store data, and in how that data is displayed by R:
  - `print()` (specifically, `print.ts()`)
  - `plot()` (specifically, `plot.ts()`)
- **Learning Objective:** Learners will be able to retrieve the temporal attributes (start and end points, as well as frequency) of a time series object:
  - `start()`
  - `end()`
  - `frequency()`

Lesson 2.2: *Create a Time Series object in Base R*

- **Learning Objective:** Learners will convert a vector of observations with a known start time and frequency (e.g., monthly data starting in the year 2004) into a `ts` object:
  - `ts()`
  - `as.ts()`

Lesson 2.3: *Using the Zoo Package to store time series data*

- **Learning Objective:** Learners will be introduced to the `zoo` object from the `zoo` package, and why is it different from base `ts`:
  - Zoo can use irregular time intervals, more robust, etc.
- **Learning Objective:** Learners will be able to convert and coerce time series objects with the `zoo` package:
  - `zoo::zoo()`
  - `zoo::as.zoo()`

Lesson 2.4: *Using Zoo to extract time and data vectors*

- **Learning Objective:** Learners can extract "core data" and time data from a `ts` or `zoo` object:
  - `time()`
  - `zoo::coredata()`

Chapter 3: Subsetting, Extracting, and Resampling

Lesson 3.1: *Subsetting a window of observations*

- **Learning Objective:** Learners will be able to extract a window of observations between a set of given points in time:
  - `window()`
  - `as.Date()`
  - `zoo::as.yearmon()`
- **Learning Objective:** Learners will use the `'['` operator with `as.Date()` to extract an observation from a specific time:
  - `'['`
  - `as.Date()`
  - `zoo::as.yearmon()`

Lesson 3.2: *Retrieving observations by index*

- **Learning Objective:** Learners will use the standard R functions to extract one or more observations by numerical index:
  - `'['`
  - `head()`
  - `tail()`
  - e.g.: `data[1:20]` retrieves observations 1 through 20, as does `head(data, n = 20)`

Lesson 3.3: *Resampling observations*

- **Learning Objective:** Learner will be able to re-sample observations to any interval of time (yearly, monthly, quarterly, etc.):
  - `aggregate()` (specifically, `aggregate.zoo()` for `zoo` objects)
  - `lubridate::year()`
  - `zoo::yearqtr()`
  - `zoo::yearmon()`
  - e.g.: `aggregate(data, by = lubridate::year, FUN = sum)` finds sums of observations within each year.

Lesson 3.4: *Imputing Missing Values*

- **Learning Objective:** Learners will use the `zoo` package to impute missing values with either linear interpolation or cubic spline interpolation:
  - `zoo::na.approx()` and `zoo::na.spline()`, respectively

Chapter 4: Rolling and Expanding Windows

Lesson 4.1: *What are windows?*

- **Learning Objective:** Learners will understand the utility of rolling and expanding windows: finding moving averages, cumulative sums, etc.

Lesson 4.2: *Calculating a Rolling Window*

- **Learning Objective:** Learners will be able to perform a rolling window operation on a time series, creating a moving average (or moving sum) of an arbitrary length:
  - `zoo::rollapply()`
  - `zoo::rollapplyr()` (convenience wrapper for `zoo::rollapply(align = "right")`)
  - e.g.: `zoo::rollapplyr(daily_data, FUN = mean, width = 7)` to create a 7-day rolling average from `daily_data`

Lesson 4.3: *Calculating an Expanding Window*

- **Learning Objective:** Learners will be able to create an expanding window – a rolling window where the "start" is fixed and the "end" moves:
  - `cumsum()`
  - `seq_along()`
  - `cumsum(data) / seq_along(data)` gives a rolling mean, which exists in `dplyr::cummean()` but not base R.

Lesson 4.4: *Plotting windows alongside Data*

- **Learning Objective:** Learners will be able to plot the rolling/expanding window alongside the original data, in order to visually assess how these operations affect the data:
  - `plot()`
  - `lines()`

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 importing the data, converting to a time series object (with `zoo`), imputing missing values, and plotting the resulting data. Additionally, you will create an aggregate of the data, as well as a rolling window average of the data.

5.1 Importing the Data

```
# The following libraries are included for you

library(zoo)
# 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 `zoo` object. It contains missing values,
# so we'll need to impute those!

# This will be hidden from the users, of course.
missing_co2 <- readRDS("data/missing.Rds")
```

5.2 Visualizing the Data

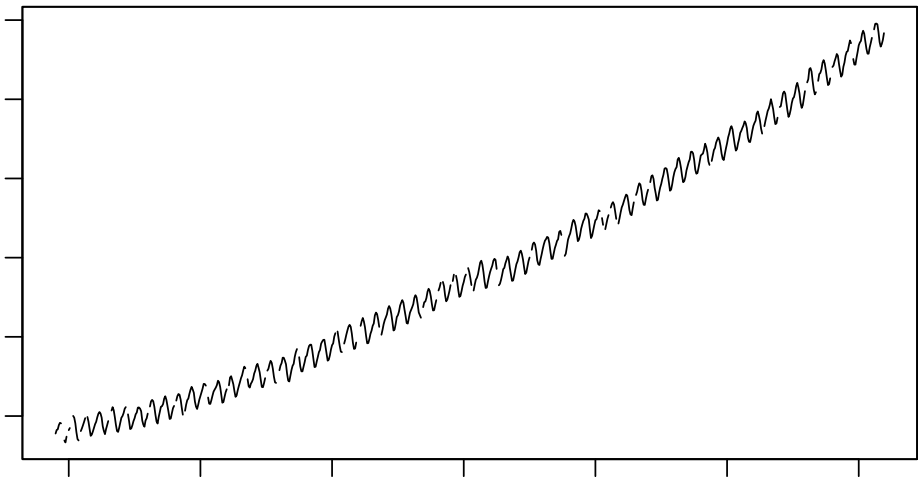
Perform basic data exploration by:

1. Printing the first 20 observations with `head()`, and,
2. Creating a plot of the data with `plot()`

```
head(missing_co2, n = 20)
```

```
## Jan 1959 Feb 1959 Mar 1959 Apr 1959 May 1959 Jun 1959 Jul 1959 Aug 1959
##   315.58   316.48   316.65   317.72   318.29   318.15       NA       NA
## Sep 1959 Oct 1959 Nov 1959 Dec 1959 Jan 1960 Feb 1960 Mar 1960 Apr 1960
##   313.84   313.33   314.81       NA   316.43   316.98       NA       NA
## May 1960 Jun 1960 Jul 1960 Aug 1960
##   320.04   319.59   318.18   315.90
```

```
plot(missing_co2)
```



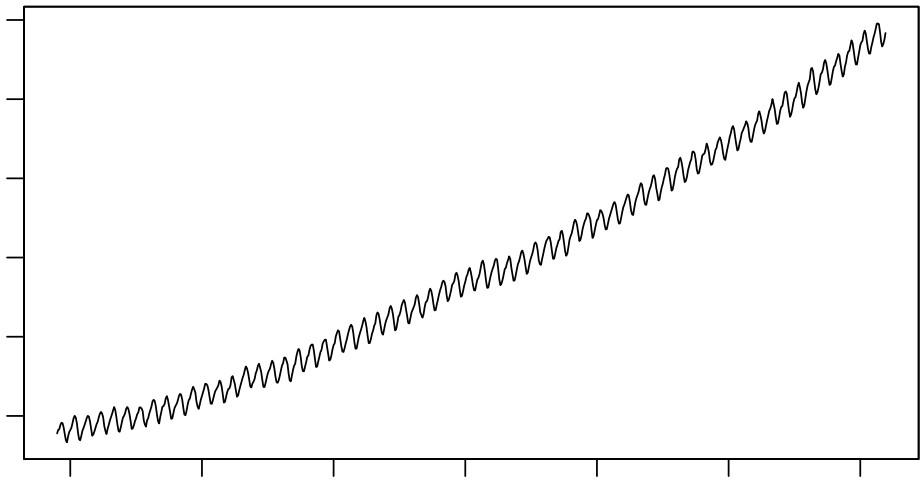
Notice that there are "holes" in the data? this suggests that we've got NA values, which is apparent when we view the first few observations with `head()`.

5.3 Imputing the Missing Values

Impute the missing values with a *cubic spline* interpolation, then plot the results

```
filled_co2 <- na.spline(missing_co2)
```

```
plot(filled_co2)
```



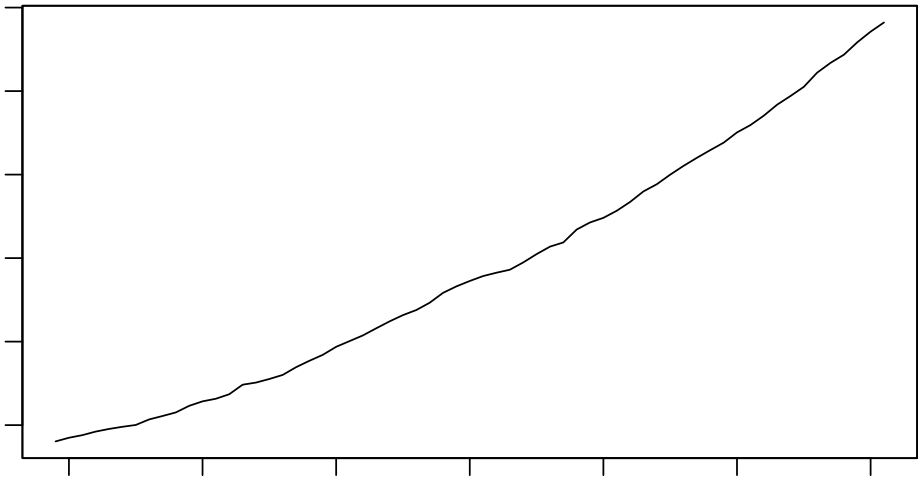
5.4 Yearly Aggregate

Using `aggregate()`, create a yearly mean of the data, then plot the data

*Hint: use `lubridate::year()`

```
yearly_co2 <- aggregate(filled_co2,
  by = lubridate::year,
  FUN = mean
)


plot(yearly_co2)
```



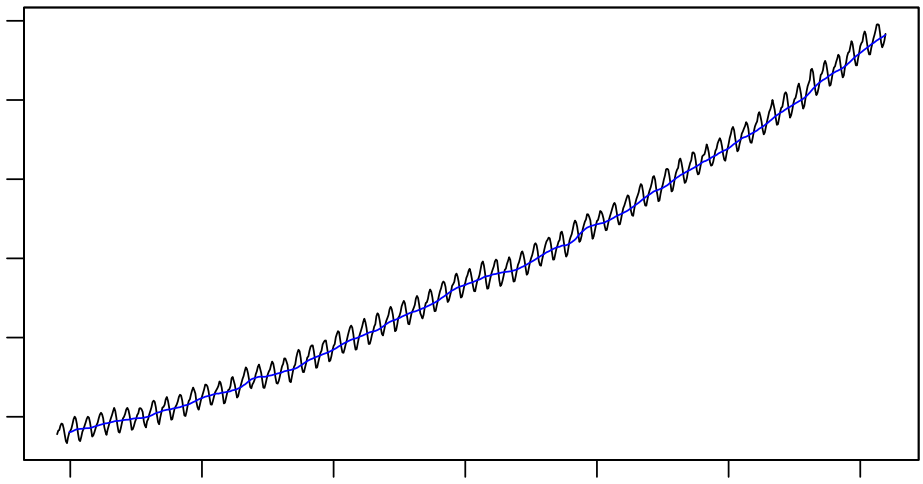
5.5 Rolling Window

Calculate a 12-month rolling window average of the data, then overlay the results (in blue) on top of the original data. Label the x-axis as "Time" and the y-axis as "CO2 Concentration", and give your graph the title "Monthly CO2 Concentration at Mauna Loa Observatory"

```
roll <- rollapplyr(filled_co2,
  FUN = mean,
  width = 12
)

plot(filled_co2,
     xlab = "Time",
     ylab = "CO2 Concentration",
     main = "CO2 Concentration at Mauna Loa Observatory")

lines(roll, col = "blue")
```



5.6 Given Code

The following code is given to the learners at the beginning of the exercise:

```
# Question 1: Explore the Data
___(missing_co2, ___ = ___)

___(missing_co2)
```

```
# Question 2: Impute Missing Values

filled_co2 <- ___(___)

___(filled_co2)
```

```
# Question 3: Find Yearly Mean Aggregate
yearly_co2 = aggregate(___, by = ___, FUN = ___)

___(___)
```

```
# Question 4: Calculate a Rolling Window
roll <- rollapplyr(filled_co2, FUN = ___, width = 12)

plot(___,
     xlab = ___,
     ylab = ___,
     main = "CO2 Concentration at Mauna Loa Observatory")

lines(___, col = ___)
```