Course Outline: Time Series Data in R

Harrison Brown

2022-06-05

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). I wanted to keep the package dependencies to just `zoo`, but `lubridate::year()` makes the conversion in Lesson 3.3 much easier for the learner).

Terms Defined:

- *Temporal Data*: Data that describe a measurement of a variable at a specific point in time, such as the chance of rain on June 20, 2009, or the price of a particular stock market at 2:00 p.m. on January 9, 1987.
- *Temporal Attributes*: The characteristics and attributes that define the sampling frequency (*monthly*, *weekly*, etc.) and the start- and end-points of *Temporal Data*.
- *Core Data*: Within R, *Core Data* is the "measured" data without any temporal information; i.e., the set of values that do not have any temporal attributes.
- *Imputing*: The statistical process of substituting missing values through a certain process, e.g. replacing missing values with 0 or the average of adjacent values.

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 core principal of time series data: time series analysis studies *how* a variable changes with time, rather than just analyzing the variable at different points in time.
- **Learning Objective:** Learners will be introduced to basic exploratory functions to examine and visualize time series objects in R:
  - `print()` (specifically, `print.ts()`)
  - `plot()` (specifically, `plot.ts()`)

Lesson 1.2: *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 coerce objects of different classes to the desired class, and check the classes of objects:
  - `class()`
  - `as.Date()`
  - `as.numeric()`
  - `as.character()`

Lesson 1.3: *Formatting `Dates` in R*

- **Learning Objective:** Learners will be introduced to formatting Dates in R by manipulating character and Date vectors, in order to both improve legibility of and standardize the format of temporal data.
  - `format()` (specifically, `format.Date()`)
    - e.g.: `format(as.Date("1987-04-12"), "%b %d, %Y")` gives `"May 12, 1987"`

Chapter 2: Time Series objects in R

0.0.1 Lesson 2.1: *Time Series Attributes*

- **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()`

    - The `window` function can usually coerce the `start` and `end` arguments correctly, but it's good practice to tell the function exactly the date and format we want to use.

Lesson 3.2: *Subsetting specific observations*

- **Learning Objective:** Learners will use the `'['` operator with `as.Date()` to extract an observation from a specific time or a specific index:
  - `'['`
    - `data[1:20]` retrieves observations 1 through 20; `data[12]` retrieves the 12^{th} observation, etc.
  - `as.Date()`
    - `data[as.Date("2020-03-01")]` retrieves the observation for March 1, 2020.
  - `zoo::as.yearmon()`
    - If data are stored in year-month format, `data[as.yearmon("Jul 2019")]` retrieves the observation for the month of July, 2019.

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
  - `zoo::fill()` can be used to fill all `NA` values with a given value, e.g. 0.

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.
- **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 4.2: *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 4.3: *zoo's roll functions*

- **Learning Objective:** Learners will be introduced to the other `roll*` functions within `zoo`, allowing for more compact and legible code when performing rolling window calculations:
  - `zoo::rollmean()`
  - `zoo::rollmedian()`
  - `zoo::rollsum()`
  - `zoo::rollmax()`

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

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.

1.1 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")
```

1.2 Visualizing the Data

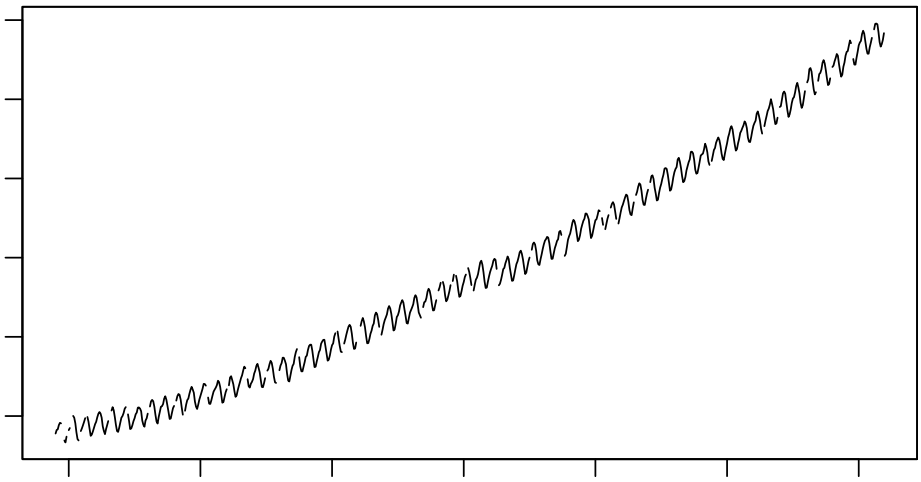
Perform basic data exploration by:

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)
```



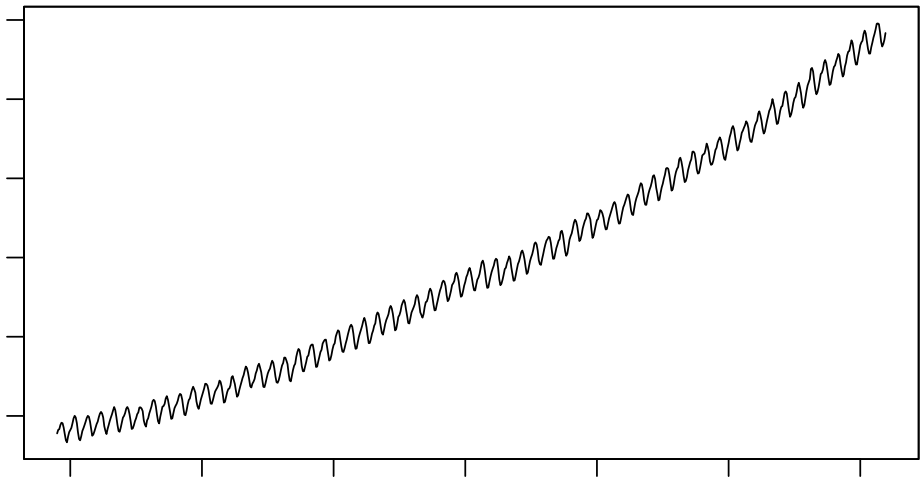
Notice that there are "holes" in the data? this suggests that we've got NA values, which is apparent when we view the first few observations with `head()`.

1.3 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)
```



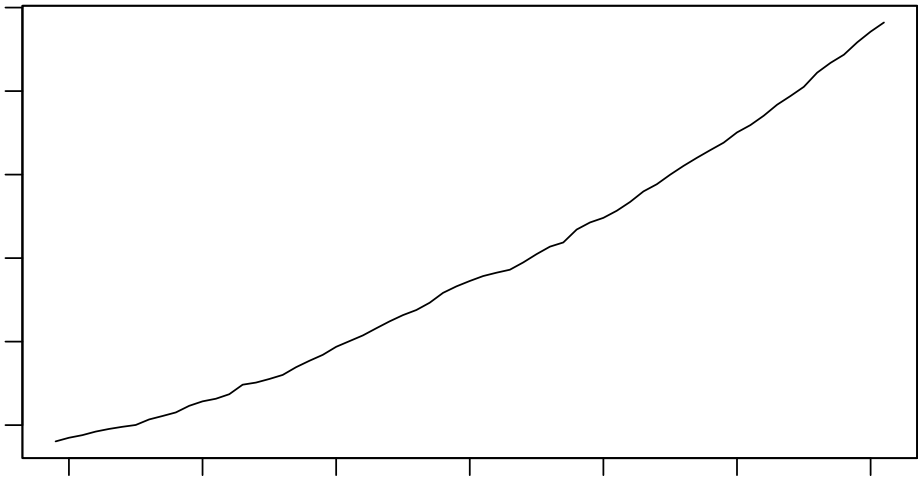
1.4 Yearly Aggregate

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)
```



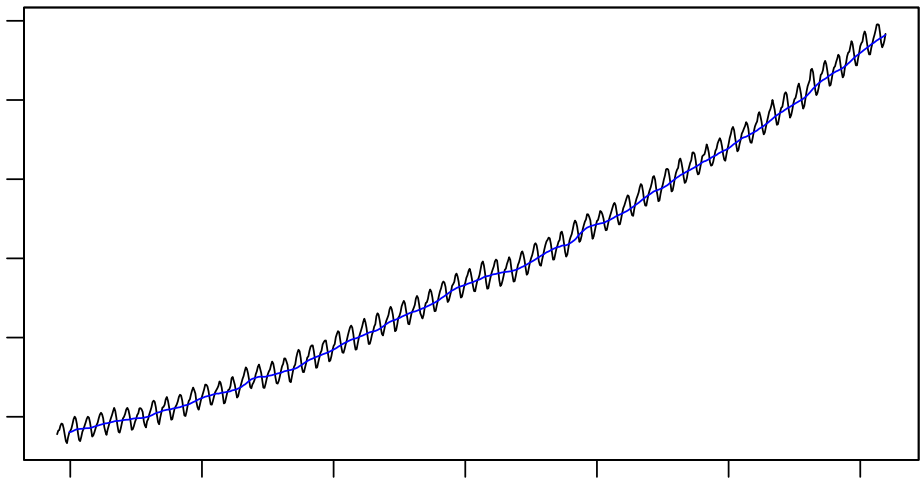
1.5 Rolling Window

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")
```



1.6 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 = ___)
```