

## 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

### 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()`
  - `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. By adding temporal attributes to the data, such as start time, frequency, and end time, learners will be able to properly manipulate data with the methods outlined in the remainder of the course.
  - `ts()`
  - `as.ts()`
  - e.g.: assume data is a numeric vector of observations; `ts(data, start = zoo::as.yearmon("Jan 2004"), frequency = 12)`

### 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()`
    - `window(data, start = "2020-01-01", end = "2020-12-31")` retrieves observations between (inclusive) Jan 1, 2020 and Dec 31, 2020.
  - `as.Date()`
    - 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<sup>th</sup> 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()`