Date Handling in R

Dr. Praveena Musunuru

December 15, 2024

Overview

Working with Dates in R: `as.Date`

- Converts strings to date objects.
- Format specifiers for date conversion:
  - %Y: Year with century (e.g., 2024).
  - %y: Year without century (e.g., 24 for 2024).
  - %m: Month as a decimal number (01-12).
  - %d: Day of the month (01-31).
  - %b: Abbreviated month name (e.g., Jan).
  - %B: Full month name (e.g., January).
  - %a: Abbreviated weekday name (e.g., Mon).
  - %A: Full weekday name (e.g., Monday).
- Example: `as.Date("4/14/2018", format="%m/%d/%Y")`

Time-Based Data: `POSIXct`

- Handles date-time objects with precision down to seconds.
- Stores data as seconds since the Unix epoch (1970-01-01 00:00:00 UTC).
- Supports time zones and date-time calculations.
- Example:
- time = as.POSIXct("1985-05-05 13:45:00", format="%Y-%m-%d %H:%M:%S")
  print(time)
- Comparison with `as.Date`:
  - `as.Date()` handles only dates (day-level precision).
  - `POSIXct` includes both date and time.

What is Time Series?

- Sequence of ordered data points measured at specific points in time.
- Examples: Stock prices, weather data, and sales over time.

Time-Series Data Types

- Regularly spaced data: Daily, Monthly, Weekly.
- Irregularly spaced data: Raw trades data.

`ts` Objects

- Represents regularly spaced time-series data (e.g., monthly, quarterly).
- Requires specification of start time and frequency.
- Seamlessly integrates with R's built-in time-series functions.
- Example:
  - Monthly data from Jan 2023 to Dec 2023
  - data = c(100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210)
  - my_ts = ts(data, start = c(2023, 1), frequency = 12)

xts Objects

- Provides a matrix-like structure for time-series data.
- Indexed by date or date-time objects.
- Enables powerful time-based subsetting and aggregation.
- Facilitates conversion of irregular data to regular time series.
- Example:

```
library(xts)
data <- rnorm(10)
dates <- seq(as.Date("2016-01-01"), length=10, by="days")
my_xts <- xts(data, order.by=dates)
```

Why Use `xts`?

- Time-Based Indexing:
  - Easily subset or filter data by date ranges.
  - Example: `my_xts["2023-01-01/2023-03-31"]`
- Integration with Other Packages:
  - Foundation for libraries like `quantmod` and `PerformanceAnalytics`.
- Regularization:
  - Converts irregular timestamps into regular intervals.
- Matrix-Like Structure:
  - Supports matrix-like operations for time-series data.
- Aggregation and Transformation:
  - Aggregate data by periods (e.g., monthly averages).
  - Example: `apply.monthly(my_xts, colMeans)`

zoo Package

- Flexible time-series structure.
- Allows irregular and regular time-series.
- Example:

```
library(zoo)
data <- rnorm(10)
dates <- seq(as.Date("2016-01-01"), length=10, by="days")
my_zoo <- zoo(data, dates)
```

Subsetting xts Objects

- Filter by date range.
- Example:

```
tmp_2002 <- my_xts["2002"]
jan_to_mar <- my_xts["2002-01-01/2002-03-22"]
```

Extracting Specific Entries

- Example operations:

```
last_week <- last(my_xts, "1 week")
first_three_days <- my_xts[1:3]
```

NIFTY

- Loaded the Nifty return data into `nifty_ret`
- Created a `Date` vector and removed the Date column.
- Converted the data into an `xts` object using the `xts()` function.
- Checked structure of the `nifty_xts` object.
- Displayed the first 10 entries of the data and identified the last entry.
- Filtered data for the year 2004
- Selected data from October 1, 2005 to Dec 31, 2005.
- Retrieved the last week's data and the last 2 days excluding the first 2 entries.

Breakdown of Nifty Data Selection Expression

We start with the expression:

- **Step 1:** `first(tmp_xts, '1 year')`:
  - Selects the first year of data from `tmp_xts`.
  - Data from the start of the dataset until one year later.
- **Step 2:** `last(first(tmp_xts, '1 year'), '1 quarter')`:
  - Selects the last quarter (3 months) of the first year.
- **Step 3:** `last(last(first(tmp_xts, '1 year'), '1 quarter'), '2 month')`:
  - From the 3 months selected in the previous step, selects the last 2 months.
- **Step 4:** `first(last(last(first(tmp_xts, '1 year'), '1 quarter'), '2 month'), '2 week')`:
  - From the 2-month period, selects the first 2 weeks.
- **Step 5:** `last(first(last(last(first(tmp_xts, '1 year'), '1 quarter'), '2 month'), '2 week'), '1 day')`:
  - From the first 2 weeks selected, keeps only the last day.