Scraping of Exchange Rates

Ömer Çelikörs

2023-01-01

Contents

1. Package Description

2. Functions Used in Package

2.1. Package Install

2.2. Convert from String to Date Object and Check Date

2.3. Find Differences of Two Dates

2.4. Check Length of Date Range

2.5. Scrape Target Url

2.6. Get Exchange Rates Between Two Dates

2.7. Get Exchange Rates on a Single Date

3. Test Package

1. Package Description

- This package aims to take **exchanges rates of Turkish Central Bank** from site. Package can be expanded in new versions. **Exchanges rates of new countries** can be added with **country parameters**.
- Package includes **two main functions**. One of them is called **"get_exchange_rates_with_single_date(date)"** and other function name is **"get_exchange_rates_with_date_range(start_date, end_date)"**.
- Main functions return exchanges rates of related date as a **data frame**.
- All date formats have to be format of **"DD.MM.YYYY"**.
- Date range must be **between 0 and 30**.
- If status code of response equals to **404**, then it means that **there is no data related to requested date**. Also, data frame does not contain date of 404 but it can be found on output of code execution.
- When any error exposes **during request**, package catches error and **gives report to client**. In addition, while scraping site if any error is occured on **only one of the requests**, this request is **skipped** and **passed another** request. As a result, code execution is **not stopped**

but error text is **shown to client**.

- Delay between requests are **1 seconds**. This numeric value is implemented as hard-coded. If parameter of changing of delay is added, this situation might lead to be sent request with a little delay by client. It is aimed not to create dense server traffic.

2. Functions Used in Package

2.1. Package Install

Function checks related packages **whether they are installed or not**. If a package is not installed, function installs pacakage or if a package is already installed then function imports pacakage:

```
install_packages_if_necessary <- function(packages) {

  # Install packages not yet installed.
  installed_packages <- packages %in% rownames(installed.packages())
  if (any(installed_packages == FALSE)) {
    install.packages(packages[!installed_packages])
  }

  # Packages loading.
  invisible(lapply(packages, library, character.only = TRUE))
}
```

2.2. Convert from String to Date Object and Check Date

String which is inserted by client is converted to date object. Obtained data is checked by looking its format. If it is not validated, an error is thrown and code execution is stopped:

```
convert_to_date <- function(date) {
  date <- as.Date(date, "%d.%m.%Y")
  return (date)
}
```

```
is_date_check <- function(date, date_name) {
  if (is.na(date) ){
    str1 <- date_name
    str2 <- " is not valid."
    validation_text <- paste(str1,str2,sep="")
    stop(validation_text)
  } else {
    str1 <- date_name
    str2 <- " is valid."
    validation_text <- paste(str1,str2,sep="")
    print(validation_text)
  }
}
```

2.3. Find Differences of Two Dates

When "get_exchange_rates_with_date_range" method is used, this function is called:

```
find_diff_of_dates <- function(start_date, end_date) {
  days_diff <- end_date - start_date
  return (days_diff)
}
```

2.4. Check Length of Date Range

Date range has to be between 0 and 30. Otherwise, it leads to occur validation error:

```
days_diff_check <- function(days_diff) {
  if (days_diff > 30 | days_diff < 0){
    validation_text <- "Validation error is occured. Day differences must
                        not be greater than 30 or lower than 0."
    stop(validation_text)
  } else {
    validation_text <- "Date range is valid."
    print(validation_text)
  }
}
```

2.5. Scrape Target Url

Function provides request to target endpoint. If any error is obtained, it throws error text. If any 404 http error is taken, this means that there is no data on site:

```
scrape_target_url <- function(date) {

  install_packages_if_necessary(c("curl", "xml2", "XML"))

  day <- format(date, format = "%d")
  month <- format(date, format = "%m")
  year <- format(date, format = "%Y")

  year_month <- paste(year, month, sep="")
  day_month_year <- paste(day, month, year, sep="")

  target_url <- sprintf("https://www.tcmb.gov.tr/kurlar/%s/%s.xml", year_month,
                        day_month_year)

  tryCatch(
    {

      # Read the xml file.
      exchange_rates_data= read_xml(target_url,
                                    options = c("NOBLANKS","NOWARNING"))
      Sys.sleep(1)
```

```
      # Parse the exchange_rate_data into R structure representing XML tree.
      exchange_rates_xml <- xmlParse(exchange_rates_data)

      # Convert the parsed XML to a dataframe.
      exchange_rates <- xmlToDataFrame(nodes=getNodeSet(exchange_rates_xml,
                                                        "//Currency"))
      exchange_rates$Date <- format(date, "%d.%m.%Y")

      return (exchange_rates)

    }, warning = function(w) {

      print(paste("Warning text:", w, "Date:", date, sep=" "))

    }, error = function(e) {

      is_include <- grepl("404", e, fixed = TRUE)

      if (!is_include){
        exchange_rates <- NULL
        print(paste("Error text:", e, "Date:", format(date, "%d.%m.%Y"),
                    sep=" "))
        return (exchange_rates)
      } else {
        return (404)
      }

    }
  )

}
```

2.6. Get Exchange Rates Between Two Dates

This is main function. It gets two parameters as start_date and end_date. Function calls helper functions. As a result, function returns all exchange rates between two dates as a data frame structure:

```
get_exchange_rates_with_date_range <- function(start_date, end_date) {
  start_date <- convert_to_date(start_date)
  end_date <- convert_to_date(end_date)

  is_date_check(start_date, "Start date")
  is_date_check(end_date, "End date")

  days_diff <- find_diff_of_dates(start_date, end_date)
  days_diff_check(days_diff)
```

```
  result_exchange_rates <- data.frame()
  for (added_day in 0:days_diff) {
    exchange_rates <- scrape_target_url(start_date + added_day)

    if (is.null(exchange_rates)){
      print(paste("There is an error on date",
                  format(start_date + added_day, "%d.%m.%Y"), sep=" "))
      next
    } else if (is.numeric(exchange_rates)) {

      print(paste("There is no data on date",
                  format(start_date + added_day, "%d.%m.%Y"), sep=" "))
      next

    }
    result_exchange_rates <- rbind(result_exchange_rates,
                                   exchange_rates)
  }

  return (result_exchange_rates)
}
```

2.7. Get Exchange Rates on a Single Date

This is also main function. Function uses related helper functions and reaches exchange rates only for a single date. At the end, function returns data frame structure:

```
get_exchange_rates_with_single_date <- function(date) {
  date <- convert_to_date(date)
  is_date_check(date, "Date")
  exchange_rates <- scrape_target_url(date)
  if (is.null(exchange_rates)){

    print(paste("There is an error on date", format(date, "%d.%m.%Y"), sep=" "))

  } else if (is.numeric(exchange_rates)) {

    print(paste("There is no data on date", format(date, "%d.%m.%Y"), sep=" "))

  } else {

    return (exchange_rates)

  }
}
```

3. Test Package

Tests of main functions:

```
# As a date
# normal date
# result1 <- get_exchange_rates_with_single_date("05.12.2022")
# date of holiday
# result1 <- get_exchange_rates_with_single_date("04.12.2022")
# invalidated date format
# result1 <- get_exchange_rates_with_single_date("05/12/2022")

# As a date range
# normal date range
# result2 <- get_exchange_rates_with_date_range("27.12.2022", "29.12.2022")
# bigger than 30 days
# result2 <- get_exchange_rates_with_date_range("27.10.2022", "29.12.2022")
# lower than 0 day
# result2 <- get_exchange_rates_with_date_range("27.12.2022", "29.10.2022")
# invalidated date format
# result2 <- get_exchange_rates_with_date_range("27-12-2022", "29.10.2022")
```