Introduction to FixerR

Guanchen Zhang, Shun Chi, Xiaomeng (Sophia) Wang, Xin (Alex) Guo

2018-04-15

About the Package

`FixerR` is an R wrapper package for the Fixer API:

> Powered by 15+ exchange rate data sources, the Fixer API is capable of delivering real-time exchange rate data for 170 world currencies. The API comes with multiple endpoints, each serving a different use case. Endpoint functionalities include getting the latest exchange rate data for all or a specific set of currencies, converting amounts from one currency to another, retrieving Time-Series data for one or multiple currencies and querying the API for daily fluctuation data.

This R package implements the APIs offered in the free tier, including queries for:

- the current exchange rates based on a reference currency
- historical exchange rates on a given date
- historical exchange rates within a given date range

This R package provides users the options to define the target and base currencies and the date range. Users are encouraged to sign up for your own API key. To use your own API key, add an `access_key` argument to each function call. We include a default API key in the package only for unit-testing purposes.

This R package is intended for users who need both real-time and historical exchange rates. Due to the pricing limit, this package does not implement the paid services from Fixer. However, we provide a function that works similar to the time series endpoint from the paid services.

Package Install

`FixerR` is available on GitHub:

```
devtools::install_github('UBC-MDS/FixerR', build_vignettes = TRUE)
```

Note: to build the vignettes, you need to run the following R code to setup your API key on your local machine *before installing the package*. Otherwise, set `build_vignettes = FALSE` or ignore the argument.

```
Sys.setenv("ACCESS_KEY" = "YOUR_KEY")
```

After the install, run the following to view the vignettes.

```
library(FixerR)
browseVignettes('FixerR')
```

Load the Package

```
library(FixerR)
```

Currency Symbols

The three-letter abbreviation used in the `Fixer` API can be queried from `/symbols` endpoint in the API. These symbols will be used to define the target and base currencies as function arguments. We include in the

package the table that includes all the symbols for you to look up. You can view the table from the `symbols` data set after loading the package.

	symbol	currency
8	AUD	Australian Dollar
28	CAD	Canadian Dollar
33	CNY	Chinese Yuan
47	EUR	Euro
50	GBP	British Pound Sterling
74	JPY	Japanese Yen
150	USD	United States Dollar

Access Key

For each function call, please add your fixer API key to the argument `access_key`.

Symbol Arguments

Each function requires two main arguments: `symbol` and `base_symbol`. `symbol` refers to the 3-letter abbreviation of the target currency (converted to) default to `CAD`. `base_symbol` refers to the base currency default to `USD`.

Current Exchange Rates

`get_current_rate(symbol, base_symbol, access_key)` implements the `/latest` endpoint in the fixer API which returns the *latest* exchange rates off the base currency. The output is a number representing the exchange rate, e.g. 0.79 if 1 CAD = 0.79 USD.

Historical Exchange Rates

`get_historical_rate(date, symbol, base_symbol, access_key)` implements the historical rates endpoint (e.g. `/YYYY-MM-DD`) which returns the exchange rates on the specified date. An additional argument, `date`, is required. `date` refers to a date string in the format `YYYY-MM-DD`, e.g. 2018-04-13. The output is a number representing the exchange rate, e.g. 0.79 if 1 CAD = 0.79 USD.

Period Rates

`get_period_rate(start_date, end_date, symbol, base_symbol, access_key)` implements the same endpoint as `get_historical_rate` does but loops over the time interval between `start_date` and `end_date`. The output is a data frame containing a date and rate column. The data frame can be used for time series analysis.

Note: to avoid hitting the rate limit cap, the function is designed to work with a maximum of 30 days. If you need data from a larger interval, please use this function iteratively and add time delay between calls if necessary.

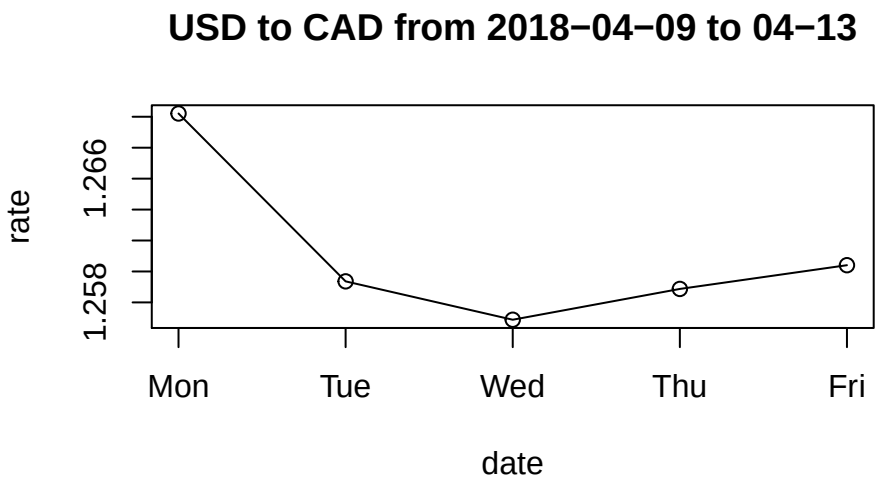
Example

```
# get the exchange rate between CAD and USD
get_current_rate(symbol = 'CAD', base_symbol = 'USD', access_key = access_key)
#> [1] 1.260502
```

```
# get the exchange rate for CAD on 2018-04-13
get_historical_rate(date='2018-03-10',
                    symbol = 'CAD',
                    base_symbol = 'USD',
                    access_key = access_key)
#> [1] 1.281204
```

```
# get the exchange raes for CAD in the past 5 days
df <- get_period_rate(start_date = '2018-04-09',
                      end_date = '2018-04-13',
                      symbol = 'CAD',
                      base_symbol = 'USD',
                      access_key = access_key)
```

```
# plot the data
plot(df, type='o', main='USD to CAD from 2018-04-09 to 04-13')
```



For more data, call the `get_period_rate` function iteratively:

```
# Get exchange rates for a month
days <- list(batch1 = c(1,15), batch2 = c(16,30))

long_data <- data.frame()
for (d in days) {
  df <- get_period_rate(start_date = paste0('2018-04-', d[1]),
                        end_date = paste0('2018-04-', d[2]),
                        symbol = 'CAD',
                        base_symbol = 'USD',
                        access_key = access_key)
  long_data <- rbind(long_data, df)

  Sys.sleep(5)  # wait 5 seconds
}
```