**Files**

A comma-separated values (CSV) file is a delimited text file that generally uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by the delimiter. CSV is a common data exchange format that is widely supported by consumer, business, and scientific applications. R makes it easy to export and import data in CSV format.

**Local Files**

Export data to a csv file

data("mtcars") # load the mtcars dataset

write.csv(mtcars, file = 'mtcars.csv') # export to file

Import data from a csv file

x <- read.csv('mtcars.csv') # read file

head(x) # print data

## X mpg cyl disp hp drat wt qsec vs am gear carb

## 1 Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4

## 2 Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4

## 3 Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1

## 4 Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1

## 5 Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2

## 6 Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

**Remote Files**

Some data providers offer data in csv format on their website. The STOXX website, a financial index provider, is one of these.

# read.csv is very flexible. For the full list of arguments type ?read.csv

x <- read.csv('<https://www.stoxx.com/document/Indices/Current/HistoricalData/h_3msx5e.txt>', sep = ';')

head(x)

## Date Symbol Indexvalue X

## 1 17.02.2020 SX5E 3853.27 NA

## 2 18.02.2020 SX5E 3836.54 NA

## 3 19.02.2020 SX5E 3865.18 NA

## 4 20.02.2020 SX5E 3822.98 NA

## 5 21.02.2020 SX5E 3800.38 NA

## 6 24.02.2020 SX5E 3647.98 NA

rownames(x) <- as.Date(x[,1], format = '%d.%m.%Y') # assign rownames

x[,c(1,ncol(x))] <- NULL # drop the first and last column

head(x) # print data

## Symbol Indexvalue

## 2020-02-17 SX5E 3853.27

## 2020-02-18 SX5E 3836.54

## 2020-02-19 SX5E 3865.18

## 2020-02-20 SX5E 3822.98

## 2020-02-21 SX5E 3800.38

## 2020-02-24 SX5E 3647.98

**R Packages**

**The ‘quantmod’ Package**

The quantmod package provides a very suitable function for downloading **financial data** from the web. This function is called getSymbols. The function works with a variety of sources.

# install the package

install.packages('quantmod')

# load the package

require(quantmod)

For stocks and shares, the yahoo source is used.

# retrieve Facebook quotes

x <- getSymbols(Symbols = 'FB', src = 'yahoo', auto.assign = FALSE)

tail(x)

## FB.Open FB.High FB.Low FB.Close FB.Volume FB.Adjusted

## 2020-05-08 212.24 213.21 210.85 212.35 12524000 212.35

## 2020-05-11 210.89 215.00 210.37 213.18 12893100 213.18

## 2020-05-12 213.29 215.28 210.00 210.10 14704600 210.10

## 2020-05-13 209.43 210.78 202.11 205.10 20684600 205.10

## 2020-05-14 202.56 206.93 200.69 206.81 17178900 206.81

## 2020-05-15 205.27 211.34 204.12 210.88 19375200 210.88

For currencies and metals, the oanda source is used. Symbols are the instruments’ ISO codes separated by /. ISO codes can be found [here](http://www.oanda.com/help/currency-iso-code).

# retrieve the historical euro/dollar exchange rate

x <- getSymbols(Symbols = 'EUR/USD', src = 'oanda', auto.assign = FALSE)

tail(x)

## EUR.USD

## 2020-05-10 1.083770

## 2020-05-11 1.082472

## 2020-05-12 1.083412

## 2020-05-13 1.084142

## 2020-05-14 1.080206

## 2020-05-15 1.081265

For economics series, the FRED source is used. Symbols can be found [here](https://research.stlouisfed.org/fred2/).

# retrieve the historical Gross Domestic Product for Japan

x <- getSymbols(Symbols = 'JPNNGDP', src = 'FRED', auto.assign = FALSE)

tail(x)

## JPNNGDP

## 2018-07-01 545545.2

## 2018-10-01 546737.7

## 2019-01-01 552687.8

## 2019-04-01 555954.0

## 2019-07-01 558237.1

## 2019-10-01 549920.9

**RESTful APIs**

An Application Program Interface (API) is basically a messenger that takes a request, tells a system what you want to do and then returns the response back to you. A RESTful API is an API that uses HTTP requests to GET, PUT, POST and DELETE data. The httr R package is a useful tool for working with HTTP. Each API has its very specific usage and documentation.

# install the package

install.packages('httr')

# load the package

require(httr)

**CRAN downloads**

The API of the CRAN downloads database.

**Example**. Which was the most downloaded package of the last month?

baseurl <- '<https://cranlogs.r-pkg.org/>' # API base url. See documentation

endpoint <- 'top/' # API endpoint. See documentation

period <- 'last-month/' # API parameter. See documentation

count <- 1 # API parameter. See documentation

url <- paste0(baseurl, endpoint, period, count) # build full url

x <- GET(url) # retrieve url

data <- content(x) # extract data

data # print data

## $start

## [1] "2020-04-15T00:00:00.000Z"

##

## $end

## [1] "2020-05-14T00:00:00.000Z"

##

## $downloads

## $downloads[[1]]

## $downloads[[1]]$package

## [1] "magrittr"

##

## $downloads[[1]]$downloads

## [1] "3889492"

The most downloaded package between **2020-04-15** and **2020-05-14** was **magrittr** with a total of **3889492** downloads.

**KuCoin API**

The API of KuCoin, cryptocurrency exchange. Documentation available [here](https://docs.kucoin.com/)

**Example**. Retrieve and plot Bitcoin price every minute in the last 24 hours.

# set GMT timezone. See documentation

Sys.setenv(TZ='GMT')

# API base url. See documentation

baseurl <- '<https://api.kucoin.com>'

# API endpoint. See documentation

endpoint <- '/api/v1/market/candles'

# today and yesterday in seconds

today <- as.integer(as.numeric(Sys.time()))

yesterday <- today - 24\*60\*60

# API parameters. See documentation

param <- c(symbol = 'BTC-USDT', type = '1min', startAt = yesterday, endAt = today)

# build full url. See documentation

url <- paste0(baseurl, endpoint, '?', paste(names(param), param, sep = '=', collapse = '&'))

# retrieve url

x <- GET(url)

# extract data

x <- content(x)

data <- x$data

# formatting

data <- sapply(1:length(data), function(i) {

# extract single candle

candle <- as.numeric(data[[i]])

# formatting. See documentation

return( c(time = candle[1], open = candle[2], close = candle[3], high = candle[4], low = candle[5]) )

})

# convert to xts

datetime <- as.POSIXct(data[1,], origin = '1970-01-01')

data <- xts(t(data[-1,]), [order.by](http://order.by) = datetime)

# plot closing values

plot(data$close, main = 'Bitcoin price in dollars')

**Web Scraping**

Web scraping is a technique for converting the data present in unstructured format (HTML tags) over the web to the structured format which can easily be accessed and used. The rvest package is a useful tool to scrape information from web pages.

# install the package

install.packages('rvest')

# load the package

require(rvest)

**Example**. Write a function to retrieve articles from [Google Scholar](https://scholar.google.com/) given a generic query string q.

getArticles <- function(q){

# build url

url <- paste0('<https://scholar.google.com/scholar?hl=en&q=>', q)

# sanitize url

url <- URLencode(url)

# get results

res <- read\_html(url) %>% # get url

html\_nodes('div.gs\_ri h3 a') %>% # select titles by css selector

html\_text() # extract text

# return results

return(res)

}

# retrieve articles about web scraping in r

getArticles('web scraping in r')

## [1] "Automated data collection with R: A practical guide to web scraping and text mining"

## [2] "Web Scraping With R"

## [3] "RCrawler: An R package for parallel web crawling and scraping"

## [4] "Web scraping with Python: Collecting more data from the modern web"

## [5] "Web scraping and Naïve Bayes classification for job search engine"

## [6] "Web scraping with Python"

## [7] "A primer on theory-driven web scraping: Automatic extraction of big data from the Internet for use in psychological research."

## [8] "The use of web-scraping software in searching for grey literature"

## [9] "R in Action"

## [10] "Web scraping techniques to collect data on consumer electronics and airfares for Italian HICP compilation"