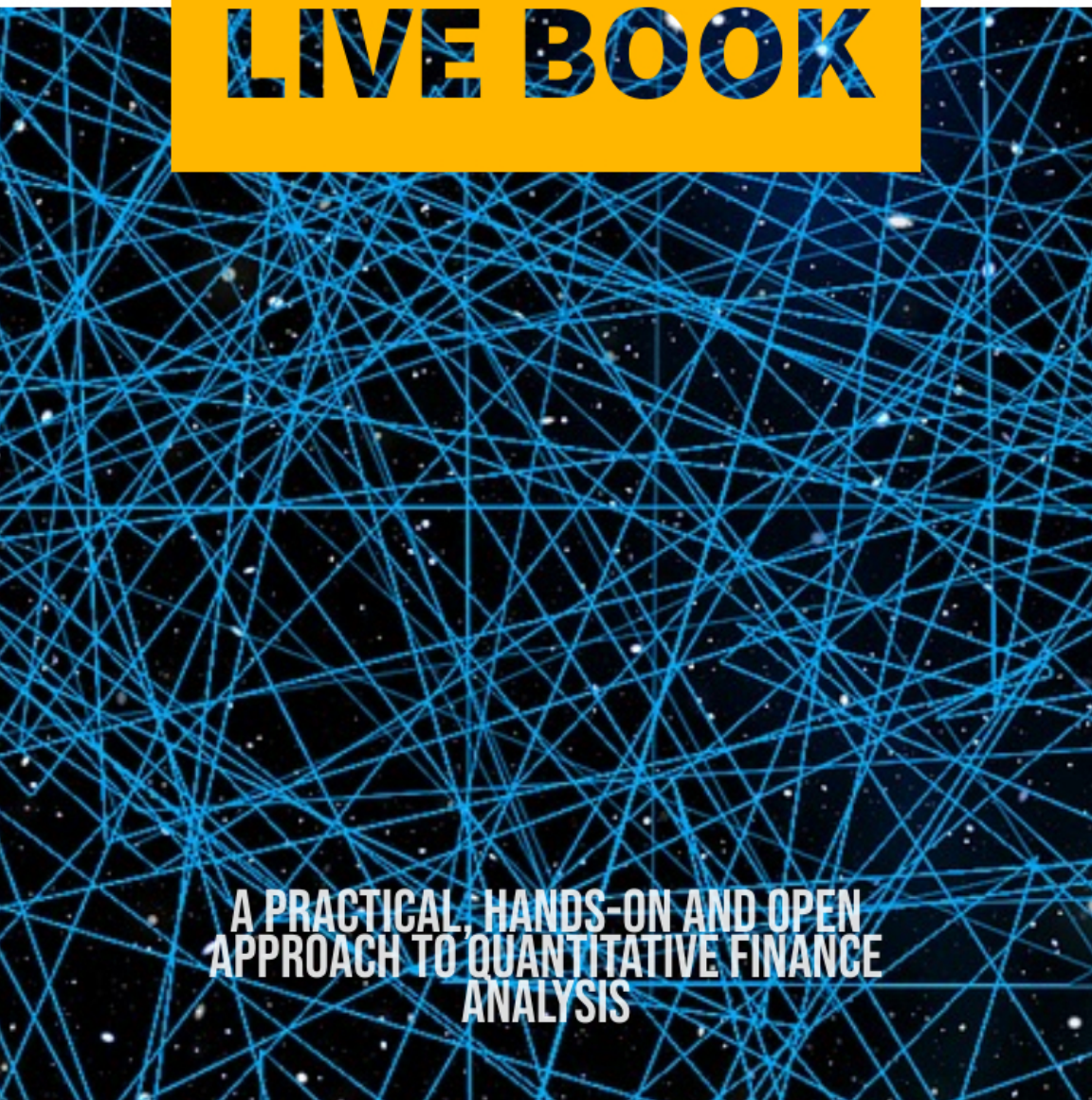
Thársis T. P. Souza

OPEN QUANT LIVE BOOK



A PRACTICAL, HANDS-ON AND OPEN APPROACH TO QUANTITATIVE FINANCE ANALYSIS

The Open Quant Live Book

Thársis T. P. Souza

2019-01-03

Contents

Preface

Working Contents

1. The Basics

- I/O
- Stylized Facts
- Correlation & Causation

2. Algo Trading

- Investment Process
- Backtesting
- Factor Investing
- Limit Order

3. Portfolio Optimization

- Modern Portfolio Theory
- Measuring Risk
- Linear Programming

4. Machine Learning

- Intro
- Agent-Based Models
- Binary Classifiers
- AutoML
- Hierarchical Risk Parity

5. Econophysics

- Entropy, Efficiency and Coupling
- Transfer Entropy, Information Transfer and Causality
- Financial Networks

6. Alternative Data

Contribute

The Book is Open[1] and we are looking for co-authors. Feel free to reach out[2] or simply create a pull request with your contribution on our Github project[3].

Book's information

First published at: openquant.netlify.com[4].

[1] https://github.com/souzatharsis/open-quant-live-book
[2] http://www.souzatharsis.com/
[3] https://github.com/souzatharsis/open-quant-live-book
[4] https://openquant.netlify.com/
[5] https://creativecommons.org/licenses/by-nc-sa/4.0/

Part I

The Basics

Chapter 1

I/O

In this Chapter, we will introduce basic input and output functions in R. You will learn how to read text and excel files as well as how to read large files. We will also show how to obtain free financial and economic data from sources such as Quandl, IEX and Alpha Vantage.

1.1 Reading and Writing

1.1.1 Text Files

The most basic and commonly used option to import data from text files in R is the use of the function `read.table` from the **r-base**. We can use this function to read text files with extensions such as `.txt` and `.csv`.

```
dat.table <- read.table(file = "<name of your file>.txt")
dat.csv <- read.csv(file = "<name of your file>.csv")
```

The package **readr** provides functions for reading text data into R that are much faster that the functions from the **r-base**. The `read_table`

function from the package **readr** provides a near-replacement for the `read.table` function.

```
library(readr)
dat.table <- readr::read_table2(file = "<name of your file>.txt")
dat.csv <- readr::read_csv(file = "<name of your file>.csv")
```

Another option to save data is to write it in `rds` format. Data stored in `rds` format has the advantage to keep the original data struture and type of the object saved. Also, `.rds` files are compressed and consume less space than files saved in `.csv` format. A data.frame object can be saved in `rds` format and then loaded back as follows:

```
write_rds(dat.frame, path = "<name of your file>.rds")
dat.frame <- read_rds(path = "<name of your file>.rds")
```

1.1.2 Excel Files

The package `readxl` has an ease to use interface to functions that load excel documents in R. The functions `read_xls` and `read_xlsx` can be used to read excel files as follows:

```
library(readxl)
readxl::read_xls(path = "<name of your file>.xls")
readxl::read_xlsx(path = "<name of your file>.xlsx")
```

The function `read_excel()` automatically detects the extension of the input file as follows:

```
readxl::read_excel("<name and extension of your file>", sheet = "<sh
```

In the `read_excel` function, the `sheet` argument can receive either the target sheet name or index number, where sheet indexing starts at 1.

The `readxl` has been oberving increased use compared to other comparable packages such as **gdata** and the **xlsx** due to its relative ease of use and performance. Also, the `readxl` do not have depency with external code libraries while the packages **gdata** and **xlsx** depend on `ActiveState PERL` and the `Java JDK`, respectively.

1.1.3 Large Files

1.2 Data Sources

In this section, we will provide examples of on how to obtain financial and economic data from public sources.

1.2.1 Alpha Vantage

Alpha Vantage offers free access to pricing data including:

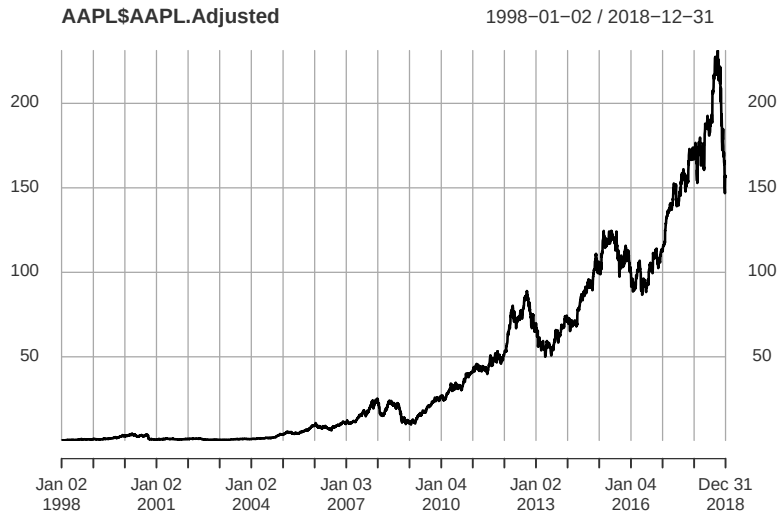
- Stock Time Series Data;
- Physical and Digital/Crypto Currencies (e.g., Bitcoin);
- Technical Indicators and
- Sector Performances.

The data are available in JSON and CSV format via REST APIs. The **quantmod** and the **alphavantager** R packages offer a lightweight R interface to the Alpha Vantage API. Daily stock prices can be obtained with the `quantmod::getSymbols` function as follows:

```
getSymbols(Symbols = "AAPL", src = "av", output.size = "full",
  adjusted = TRUE, api.key = "your API key")
```

The output data is stored in an object with the same name as the corresponding symbol, in this example `AAPL`. The output data looks like this

AAPL.Open	AAPL.High	AAPL.Low	AAPL.Close	AAPL.Volume	AAPL.Ad
13.6	16.2	13.5	16.2	6411700	
16.5	16.6	15.2	15.9	5820300	
15.9	20.0	14.8	18.9	16182800	
18.8	19.0	17.3	17.5	9300200	
17.4	18.6	16.9	18.2	6910900	
18.1	19.4	17.5	18.2	7915600	



We called the `quantmod::getSymbols` function with the following arguments:

- `Symbols='AAPL'` defines a character vector specifying the names of each symbol to be loaded, here specified by the symbol of the company Apple Inc.;
- `src="av"` specifies the sourcing method, here defined with the value corresponding to Alpha Vantage;
- `output.size="full"`specified length of the time series returned. The strings `compact` and `full` are accepted with the following specifications: `compact` returns only the latest 100 data points; `full` returns the full-length time series of up to 20 years of historical data;
- `adjusted=TRUE` defines a boolean variable to include a column of closing prices adjusted for dividends and splits;
- `api.key` specifies your Alpha Vantage API key.

1.2.2 IEX

The IEX Group operates the Investors Exchange (IEX), a stock exchange for U.S. equities that is built for investors and companies. IEX offers U.S. reference and market data including end-of-day and intraday pricing data. IEX offers an API with "a set of services designed for developers and engineers. It can be used to build high-quality apps and services". Data sourced from the IEX API is freely available for commercial subject to conditions[1] and the use of their API is subject to additional terms of use[2].

IEX lists the following github project as an unofficial API for R: https://github.com/imanuelcostigan/iex. We will provide examples on how to obtain intraday pricing data using this package. First, we will use the **devtools** to install the package directly from its github repository as follows:

```
library(devtools)
install_github("imanuelcostigan/iex")
```

The **iex** package provides 4 set of functions as follows:

- `last`: Provides IEX near real time last sale price, size and time. Last is ideal for developers that need a lightweight stock quote. IEX API real time API documentation[3].
- `market`: Provides exchange trade volume data in near real time. IEX market API documentation[4].
- `stats`: A set of functions that return trading statistics. IEX stats API documentation[5].
- `tops`: Provides IEX's aggregated bid and offer position in near real time for all securities on IEX's displayed limit order book. IEX API TOPS documentation[6].

For instance, the `last` function has the following arguments:

[1] https://iextrading.com/api-exhibit-a/
[2] https://iextrading.com/api-terms/
[3] https://iextrading.com/developer/docs/#last
[4] https://iextrading.com/developer/#market-market
[5] https://iextrading.com/developer/#stats
[6] https://iextrading.com/developer/#tops-tops

- `symbols`: A vector of tickers (case insensitive). Special characters will be escaped. A list of eligible symbols is published daily[7] by the IEX. When set to `NULL` (default) returns values for all symbols.
- `fields`: A vector of fields names to return (case sensitive). When set to `NULL` (default) returns values for all fields.
- `version`: The API version number, which is used to define the API URL.

We can obtain intraday stock price data with the `last` function as follows:

```
dat <- last(symbols = c("AAPL"), fields = c("symbol", "price",
  "size"))
```

The function returns an S3 object of class `iex_api` which has three accessible fields: `path` , `response` and `content`.

- The `path` contains the corresponding IEX API path:

```
dat$path
```

```
## [1] "tops/last"
```

- The `response` contains the unparsed IEX API response:

```
dat$response
```

```
## Response [https://api.iextrading.com/1.0/tops/last?symbols=AAPL&f
##   Date: 2019-01-04 04:49
##   Status: 200
##   Content-Type: application/json; charset=utf-8
##   Size: 45 B
```

- The `content` contains the parsed content from the API's response (usually being a list).

```
dat$content
```

```
## [[1]]
## [[1]]$symbol
```

[7] https://iextrading.com/trading/eligible-symbols/

```
## [1] "AAPL"
##
## [[1]]$price
## [1] 142
##
## [[1]]$size
## [1] 100
```

According to the developer, this package causes R to pause 0.2 seconds after executing an API call to avoid the user being throttled by the IEX API (which enforces a 5 request per second limit). Documentation about the other set of functions can be obtained at https://github.com/imanuelcostigan/iex/tree/master/man.

1.2.3 Quandl

Chapter 2

Stylized Facts

2.1 Introduction

2.2 Distribution of Returns

2.2.1 Fat Tails

A distribuição de retornos financeiros apresenta leptokurtose. A ocorrência de eventos extremos é mais provável comparado com uma distribuição normal, i.e., as caudas da distribuição empírica de retornos são mais "pesadas" comparadas com as caudas esperadas supondo uma distribuição normal de probabilidade.

2.2.2 Skewness

A distribuição empírica de retornos é distorcida para esquerda. Retornos negativos são mais prováveis que retornos positivos.

2.3 Volatility

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{2.1}$$

2.3.1 Time-invariance

A volatilidade de retornos financeiros não é constante ao longo do tempo.

2.3.2 Volatility Clustering

Eventos extremos são observados próximos um dos outros.

2.3.3 Correlation with Trading Volume

O volume de negociação de um ativo tem correlação significante com a volatilidade do mesmo.

2.4 Correlation

$$\rho = \frac{\sum_{t=1}^{T}(r_t - \hat{r}_t)(s_t - \hat{s}_t)}{\sqrt{\sum_{t=1}^{T}(r_t^\tau - \hat{r}_t^\tau)^2}\sqrt{\sum_{t=1}^{T}(s_t - \hat{s}_t)^2}}, \tag{2.2}$$

onde $\hat{r}_t$ e $\hat{s}_t$ são a média amostral de r_t e s_t, respectivamente.

2.4.1 Time-invariance

A correlação entre duas series temporais de retornos financeiros não é constante ao longo do tempo.

2.4.2 Auto-correlation

Retornos financeiros apresentam baixa autocorrelação (linear), exceto em escalas de tempo muito baixas, e.g., minutos, onde há presença de efeitos de microstructura. Por outro lado, a função de autocorrelação do valor absoluto de retornos financeiros decai lentamente com o tempo.

A correlação contemporânea é maior do que a correlação cruzada.

Chapter 3

Correlation & Causation

Part II

Algo Trading

Chapter 4

Limit Order

Part III

Portfolio Optimization

Part IV

Machine Learning

Part V

Econophysics

Chapter 5

Entropy

Chapter 6

Transfer Entropy

Chapter 7

Financial Networks