Accessing and Managing Financial Data with Tidy Finance

useR! Conference

Christoph Scheuch

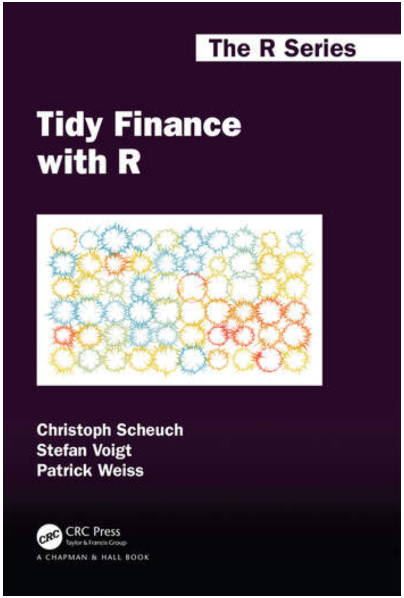
2024-07-09

A few years ago ...

Two PhD students in Vienna

- Hardly any public code or data
- Hard to reproduce papers
- 80% of time spent preparing data

Since then ...

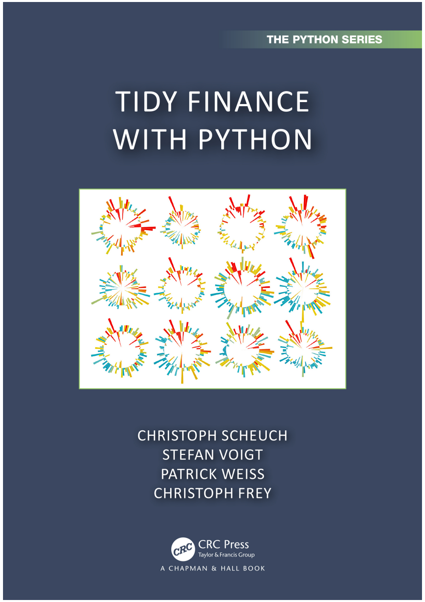


The R Series

Tidy Finance with R

Christoph Scheuch
Stefan Voigt
Patrick Weiss

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK



THE PYTHON SERIES

TIDY FINANCE WITH PYTHON

CHRISTOPH SCHEUCH
STEFAN VOIGT
PATRICK WEISS
CHRISTOPH FREY

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

What is Tidy Finance?

A **transparent**, **open-source** approach to research in financial economics, featuring **multiple programming languages**

The `tidyfinance` packages is a simple way to:

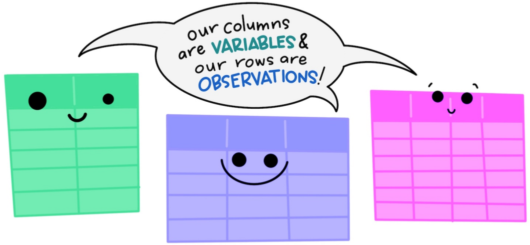
- Load our approach into R
- Use helper functions to download & process data
- Easily compile multiple data sources

Why tidy?

1. Write code that is **easy to read** for humans
2. **Compose simple functions** to solve complex problems
3. **Embrace functional programming** for reproducible results
4. **Reuse data structures** across applications

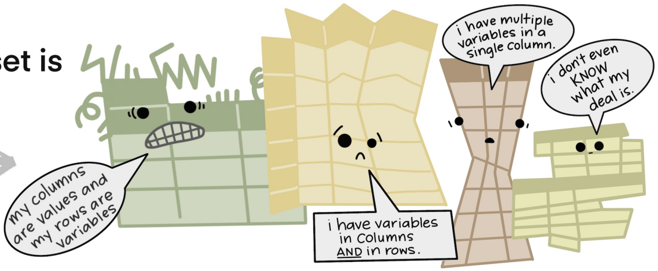
Recap: what is *tidy data*?

The standard structure of tidy data means that "tidy datasets are all alike..."



"...but every messy dataset is messy in its own way."

—HADLEY WICKHAM


my columns are values and my rows are variables
i have variables in columns AND in rows.
i have multiple variables in a single column.
i don't even KNOW what my deal is.

A consistent interface to financial data

```r
library(tidyfinance)

download_data(
  type = "factors_ff3_monthly",
  start_date = "2023-01-01",
  end_date = "2023-12-31"
) |>
  print(n = 5)
```

A consistent interface to financial data

```r
library(tidyfinance)

download_data(
  type = "factors_ff3_monthly",
  start_date = "2023-01-01",
  end_date = "2023-12-31"
) |>
  print(n = 5)
```

```
# A tibble: 12 x 5
  date       risk_free mkt_excess     smb     hml
  <date>         <dbl>      <dbl>   <dbl>   <dbl>
1 2023-01-01    0.0035     0.0665  0.05   -0.0402
2 2023-02-01    0.0034    -0.0258  0.0117 -0.0081
3 2023-03-01    0.0036     0.0251 -0.0551 -0.0886
4 2023-04-01    0.0035     0.0061 -0.0335 -0.0004
```

Deep dive: download raw data

```r
raw_data <- download_french_data(dataset)
raw_data <- raw_data$subsets$data[[1]]
```

Deep dive: parse dates

```r
raw_data <- download_french_data(dataset)
raw_data <- raw_data$subsets$data[[1]]

if (grepl("monthly", type)) {
  processed_data <- raw_data |>
    mutate(date = floor_date(ymd(paste0(date, "01")), "month"))
} else {
  processed_data <- raw_data |>
    mutate(date = ymd(date))
}
```

Deep dive: transform numeric columns

```r
raw_data <- download_french_data(dataset)
raw_data <- raw_data$subsets$data[[1]]

if (grepl("monthly", type)) {
  processed_data <- raw_data |>
    mutate(date = floor_date(ymd(paste0(date, "01")), "month"))
} else {
  processed_data <- raw_data |> mutate(date = ymd(date))
}

processed_data <- processed_data |>
  mutate(across(-date, ~na_if(.,-99.99)),
         across(-date, ~ . / 100))
```

Deep dive: rename columns

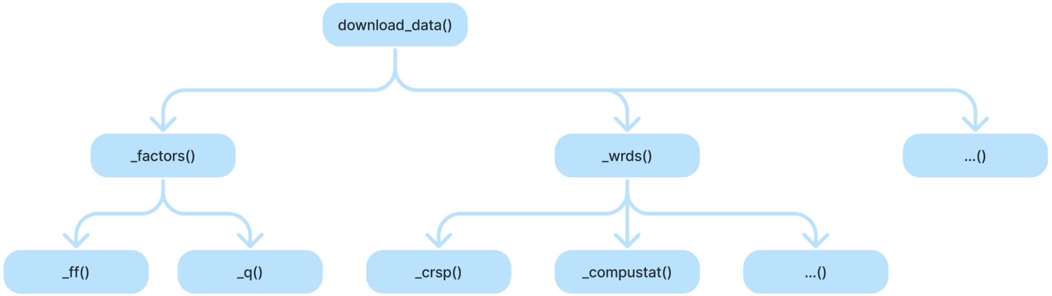
```r
raw_data <- download_french_data(dataset)
raw_data <- raw_data$subsets$data[[1]]

if (grepl("monthly", type)) {
  processed_data <- raw_data |>
    mutate(date = floor_date(ymd(paste0(date, "01")), "month"))
} else {
  processed_data <- raw_data |> mutate(date = ymd(date))
}

processed_data <- processed_data |>
  mutate(across(-date, ~na_if(.,-99.99)),
         across(-date, ~ . / 100)) |>
  rename_with(tolower)

processed_data |>
  rename_with(tolower) |>
```

List of supported data sources

Currently 32 data sets supported from these domains:

- Fama-French factors
- Q factors
- Goyal-Welch macroeconomic predictors
- Wharton Research Database Service (WRDS)

Easy to extend supported sources



Example: load packages

```r
library(tidyfinance)
library(dplyr)
```

Example: load stock returns

```r
library(tidyfinance)
library(dplyr)

crsp <- download_data(
  "wrds_crsp_monthly", "2023-01-01", "2023-12-31"
)
```

Example: load factors

```r
library(tidyfinance)
library(dplyr)

crsp <- download_data(
  "wrds_crsp_monthly", "2023-01-01", "2023-12-31"
)

factors <- download_data(
  "factors_ff3_monthly", "2023-01-01", "2023-12-31"
)
```

Example: join data

```r
library(tidyfinance)
library(dplyr)

crsp <- download_data(
  "wrds_crsp_monthly", "2023-01-01", "2023-12-31"
)

factors <- download_data(
  "factors_ff3_monthly", "2023-01-01", "2023-12-31"
)

stock_returns <- crsp |>
  left_join(factors, join_by(month == date))
```

Example: winsorize column

```r
library(tidyfinance)
library(dplyr)

crsp <- download_data(
  "wrds_crsp_monthly", "2023-01-01", "2023-12-31"
)

factors <- download_data(
  "factors_ff3_monthly", "2023-01-01", "2023-12-31"
)

stock_returns <- crsp |>
  left_join(factors, join_by(month == date))

stock_returns <- stock_returns |>
  mutate(mktcap_winsorized = winsorize(mktcap, 0.05))
```

Example: summary statistics

```r
data |>
  create_summary_statistics(mktcap, mktcap_winsorized)

```

Example applications: summary statistics

```r
data |>
  create_summary_statistics(mktcap, mktcap_winsorized)

```

```
# A tibble: 2 x 7
  variable               n   mean     sd   min   q50       max
  <chr>              <int>  <dbl>  <dbl> <dbl> <dbl>     <dbl>
1 mktcap             45552 10287. 73406. 0.307  507. 3071345.
2 mktcap_winsorized  45552  4452.  9355. 8.84   507.   37332.
```

Example: assign portfolios

```r
data |>
  group_by(date) |>
  mutate(
    portfolio = assign_portfolio(
      pick(everything()), "mktcap_winsorized", n_portfolios = 10
    )
  )

```

Example: assign portfolios

```r
data |>
  group_by(date) |>
  mutate(
    portfolio = assign_portfolio(
      pick(everything()), "mktcap_winsorized", n_portfolios = 10
    )
  )

```

```
# A tibble: 45,552 x 4
  date       permno mktcap_winsorized portfolio
  <date>      <int>             <dbl>     <int>
1 2023-02-28  12591            37332.        10
2 2023-02-28  12592               98.9        3
3 2023-02-28  12615              383.         5
4 2023-02-28  12622            37332.        10
5 2023-02-28  12623             8577.         9
# i 45,547
```

Example: assign portfolios

```r
data |>
  group_by(date) |>
  mutate(
    portfolio = assign_portfolio(
      pick(everything()), "mktcap_winsorized", n_portfolios = 10
    )
  )

```

Currently working on `calculate_portfolio_returns()`

- Calculate value-weighted and equal-weighted returns for different sorting methods

Example: estimate model

```r
data |>
  estimate_model("ret_excess ~ mkt_excess + smb + hml")

```

Example: estimate model

```r
data |>
  estimate_model("ret_excess ~ mkt_excess + smb + hml")

```

```
  mkt_excess       smb       hml
1  0.9657947 0.9403686 0.2689844
```

Example: estimate model

```r
data |>
  estimate_model("ret_excess ~ mkt_excess + smb + hml")

```

Curently working on `roll_capm_estimation()`:

- Estimate betas for different lookbacks and multiple factors

Tidy approach to financial data

- Check out open source content at **tidy-finance.org**
- Get in touch for **teaching materials**
- Submit issues to **extend supported types**
- Follow me for news: linkedin.com/in/christophscheuch