

# Advanced Empirical Finance: Topics and Data Science

---

Stefan Voigt

Spring 2024

University of Copenhagen and Danish Finance Institute

# Welcome!



## First: The fine print

- What is this course about?
- How is this course organized?
- Set-up R, Python, Quarto, and RStudio
- Tidy coding principles

# Course introduction

---

# What is empirical Finance? What is data science?

---

## Empirical Finance

- Broadly: Empirical Finance includes any empirical work in financial economics or financial econometrics
- Financial markets provide vast data to inform financial decisions: Can we predict investment risks? What determines asset prices? Are markets efficient?

## Data Science

- Plethora of buzzwords: big data, Machine learning, AI
- The less intriguing reality: How to extract knowledge from typically very large or unstructured data?
- This course: We incorporate skills from computer science, mathematics, statistics, and information visualization, projected on economics theory
- *You can (try to) do tech without statistics, but you cannot do it without coding*

**(How) can state-of-the-art methods improve financial decision making?**

# Topics of this course

---

- Optimal portfolio allocation (in theory and with real data)
- Backtesting of portfolio strategies
- Risk premia and factor selection
- Machine learning toolbox for financial forecasting
- Risk management and volatility estimation
- High-frequency econometrics

## **At the end of the course, you should**

- Understand the current state of research in empirical asset pricing matters
- Have your own ready-to-use state-of-the-art toolbox for empirical analysis
- Recognize the value of reproducible research (and know how to conduct a reproducible analysis)

# Objectives and tasks

---

## (Guided) coding assignments

- Plan, perform, and implement data-science applications from scratch
- Master and carry through relevant asset pricing models and solutions in new, unpredictable, and complex contexts
- In plain words: *Learn how to use tidy coding principles in R or Python for empirical projects by writing code and conducting your research!*

## The lecture is based on very recent academic papers

- We discuss and criticize multifactor asset pricing models, portfoliometrics, and high-frequency econometrics
- The reading list is available on Absalon
- Be prepared to discuss the literature during the lecture!

# This course within the KU curriculum

---

## **AEF aims to close the gap between some core finance courses at KU**

- Financial Decision Making
- Financial Econometrics A

## **Related courses to consider**

- Introduction to Programming and Numerical Analysis
- Various seminars in (applied) Finance
- Master thesis in Finance track (Do you consider writing your thesis on empirical Finance? Check my homepage and get in touch *early!*)

# Administration

---



# Team and communication

---

- Stefan (stefan.voigt@econ.ku.dk, [www.voigtstefan.me](http://www.voigtstefan.me))
- Weekly office hours - a chance to ask questions! (Thursdays, 1.30 - 2.30)
- Teaching assistant: Pedro ([pedro@di.ku.dk](mailto:pedro@di.ku.dk))
- Pedro moderates Absalon discussions, but it is **your responsibility** to actively engage in the exchange of ideas, code, and knowledge!
- We provide all exercises, slides, data, and other documents via Absalon
- **Remainder of the team:** all the peers around you - connect and help each other out!

## Lecture hall

- Lecture and Exercise class 2: CSS 35-3-12. Exercise Class 1: CSS 15-3-15
- I stream and record all lectures with Zoom (no guarantee that things always work, and the ultimate priority is the crowd on campus!)
- Join online for lectures, exercises, or my office hour on Zoom: 669 9264 2905 (password: 687063)

# This is how the course works

---

- **Most effort of this course should be spend on *doing* empirical work!**
- Lectures on Monday (even weeks) and every Thursday (see timetable): Methods, Theory, state-of-the-art literature
- Lecture plan in Absalon contains weekly coding assignments
- I provide detailed solutions - but you are *strongly recommended* to solve everything on your own
- If you alert me in due time, I will address some exercises or discuss the solutions
- The exercise classes are in the style of open office-hours. Pedro is there to help you - but it is your responsibility to ask if you have questions

The learning curve is very steep, but remember that you can reach out **anytime** in case something is unclear!

# Mandatory Assignments

---

- 7-day long coding assignments in which we transfer the theory into practice
- The assignments can be written individually or by **groups of a maximum of three students** (you can sign up on the Absalon discussion board if you search for group members)
- You are supposed to hand in one (!) .qmd (Quarto) script with commented source code and one (!) .pdf report in which you describe your methods and results (e.g., with figures, tables, equations, etc.)
- The source code has to create every figure, table, and number used in your report
- **Handed-in report: max. 7 pages for .pdf file in total** (font size 12pt, use this (click here) template)
- All parts must be answered in English
- Be aware of the usual rules for plagiarism and co-written assignments
- *What is the difficulty of mandatory assignments?* I assume that the weekly exercises do not pose any challenge for you anymore

# Mark the dates: Mandatory Assignments

---

1. mandatory assignment starts **March 1st**
  2. mandatory assignment starts **April 12th**
  3. mandatory assignment starts **May 3rd**
- The assignments (and data, eventually) will always be uploaded on Absalon
  - You will need to unlock the peergrade panel once in Absalon
  - Deadline: One week to work on your solutions
  - Peer feedback opens two days after the submission, and everybody will be asked to provide feedback for two peer assignments individually (subtle hint: this is *one of the most critical steps towards success* - reserve time to give helpful feedback and learn from the answers of your peers)
  - The peer feedback period is open for seven days
  - **Minimum requirement to get feedback:** Attempt to solve every task. Document your struggles if you do not finish an exercise. The code needs to run without any error or interruption so that your peers can focus on the content instead of fixing your bugs

# Peer feedback

---

- The mandatory assignments will be available for peer feedback on Absalon
- *Everybody* gets assigned 2 of the peer's submissions for each mandatory assignment (no group work)
- You have one week to provide feedback
- If you receive feedback you feel needs to be drafted more carefully, flag it. Feedback is the most important part of the process, and I will be strict in discarding useless comments

# Final Exam

---

- To qualify for the exam, you have to hand in a minimum of two out of the three mandatory assignments *and* provide useful written peer feedback for a minimum of two out of the three mandatory assignments to *two* other assignments
- Portfolio exam: **48 hours from XX to XX**
- The exam is a written assignment and consists of two parts: one selected mandatory assignment (you can use the peer feedback to improve your initial submission) and a set of entirely new exercises
- You will hand in a .pdf report and a .qmd file that replicates all results in your report

**Where do we start?**

---

## Your background? Why did you choose to enroll in this course?

---



<https://padlet.com/stefanvoigt2/8xyqpu91evwvndss>

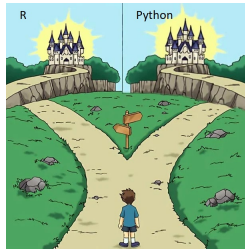
1. Bachelor's/Master's/Ph.D. Student?
2. Coding experience (General, R, Python, tidyverse)?
3. General interest in Finance? Data science? Skills for your future career?



# Introduction to R, Python, and Quarto

---

# This year: Language-agnostic coding



- Instead of focusing on sharpening your coding skills within Python or R, I want to make a case for tidy coding
- We discuss structures of clean, reproducible code that can be achieved using R *and* Python
- Irrespective of your language preferences, code can be readable, understandable, and intuitive as long as you follow standard ground rules

# Why Python and R?

---

I stick to R or Python (or Julia) because

1. coding languages should be open-source
2. you want an active community of users
3. coding languages should be well-established within the (finance) industry
4. we need communication tools to ensure reproducibility, high-quality visualization, and flexibility for data input and wrangling

Less of an issue

1. Computing speed (unless you want to work in HF or similar fields)

# What is R? Python? Quarto?

---

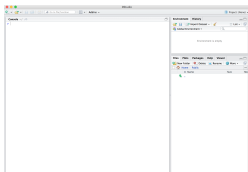
- R and Python are languages and environments for statistical computing
- Both are free to download and use / open-source (Users can expand the functionalities through add-ons called packages) / Data processing is easy / Data visualization tools are pervasive
- Python and R are nowadays the de-facto standard tools in Finance (+ you can run Python scripts from within R and vice versa)
- With Quarto you can embed code directly into the analysis. It is easy to share your (reproducible) R and Python output using Quarto

# Why RStudio? Setup for this course

---

- While *R* and *Python* run computations, *RStudio* is an integrated development environment (IDE) that provides an interface by adding many convenient features and tools
- You can use either Python or R and process your code with Quarto (there will be an asynchronous lecture on how to use Quarto)
- There are solutions to the exercises in both R and Python
- During peer feedback you may receive Python or R submissions - stick to tidy coding principles, and you will be able to understand what the code of your peers is doing even if you are not an expert in their chosen language
- Reversely, that means: Take tidy coding seriously to maximize your chance for valuable feedback
- We will all work in the same environment to ensure your peers can reproduce your results
- Strongly recommend using RStudio as an interface, but you can use other editors

# Getting started with RStudio

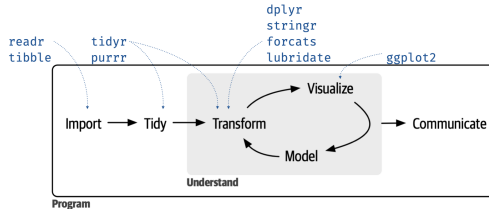


- In case you did not set your environment up yet, follow the Technical Prerequisite section in Absalon:
- Install R / Install RStudio / Install Quarto / clone the course repo

```
install.packages(reticulate)
reticulate::install_python(version="3.10.11", force = TRUE)
renv::restore() # fetch all R and Python packages required for this courses
```

- If you have never used R or RStudio, Chapters 1-3 of this book are an excellent starting point: **Basic introduction to R**
- Sign up for a free account at DataCamp (only with econ.ku.dk or alumni.ku.dk address)

# Data science workflow with the tidyverse and pandas / numpy



- The tidyverse is “a framework for managing data that aims at doing the cleaning and preparing steps much easier”
- We work almost exclusively with tidyverse packages: ggplot, dplyr, readr, ...
- Hadley Wickhams and Garret Grolemonds famous book *R for Data Science* explains everything we need
- During this semester, we cover almost every topic from their book
- Python equivalent: pandas, numpy, plotnine

# Ready for a first case study in R?

- Start by opening your project or getting the working directory right

```
setwd("path_to_your_course_folder")
```

- load the package tidyverse
- Import data: read\_csv(), read\_txt(), .., or download with tidyquant

```
library(tidyverse)
library(tidyquant)
prices_raw <- tq_get(c("AAPL", "MSFT"), get = "stock.prices", from = "2000-01-01")
prices_raw # print the object prices_raw in the console
```

```
# A tibble: 12,090 x 8
  symbol date      open high  low close  volume adjusted
  <chr>   <date>      <dbl> <dbl> <dbl> <dbl>      <dbl> <dbl>
1 AAPL   2000-01-03  0.936  1.00  0.908  0.999 535796800    0.847
2 AAPL   2000-01-04  0.967  0.988  0.903  0.915 512377600    0.776
3 AAPL   2000-01-05  0.926  0.987  0.920  0.929 778321600    0.787
4 AAPL   2000-01-06  0.948  0.955  0.848  0.848 767972800    0.719
# i 12,086 more rows
```

- Tidy/Transform, e.g. with rename() or mutate()

```
prices_tmp <- rename(prices_raw, traded_shares = volume) # Rename volume column name
prices_vol <- mutate(prices_tmp, volume_usd = traded_shares * close) # create a new column
```



# Give your code some air

## Syntax with the pipe `|>` (R) and `.` (Python)

- `verb(subject, complement)` is replaced by `subject |> verb(complement)` (R) or `subject.verb(complement)` (Python)
- No need to name unimportant variables
- Clear readability

```
prices <- prices_raw |>
  rename(traded_shares = volume) |>
  mutate(volume_usd = traded_shares * close / 1000000) # Volume in million USD
prices # Take a glimpse at your data
```

```
# A tibble: 12,090 x 9
  symbol date      open high  low close traded_shares adjusted volume_usd
  <chr>  <date>    <dbl> <dbl> <dbl> <dbl>      <dbl>      <dbl>      <dbl>
1 AAPL  2000-01-03 0.936 1.00  0.908 0.999    535796800    0.847    535.
2 AAPL  2000-01-04 0.967 0.988 0.903 0.915    512377600    0.776    469.
3 AAPL  2000-01-05 0.926 0.987 0.920 0.929    778321600    0.787    723.
4 AAPL  2000-01-06 0.948 0.955 0.848 0.848    767972800    0.719    651.
# i 12,086 more rows
```

- Do not get confused. Some sources in R use (outdated) `%>%` instead of `|>`!

# The same with Python

- Workhorse packages: pandas, numpy, and plotnine

```
import pandas as pd
import numpy as np
from plotnine import *
import yfinance as yf

prices = (yf.download(tickers=["AAPL", "MSFT"],
                       start="2000-01-01",
                       progress=False)

         .melt(ignore_index=False,
              var_name=["variable", "symbol"])
         .reset_index()
         .pivot(index=["Date", "symbol"],
               columns="variable",
               values="value")
         .reset_index()
         .rename(columns = {"Date": "date",
                           "Volume": "traded_shares",
                           "Close": "close",
                           "Adj Close": "adjusted"})
         .assign(volume_usd = lambda x: x["traded_shares"] * x["close"] / 1000000)
         )

prices
```

	variable	date	symbol	adjusted	...	Open	traded_shares	volume_usd
0		2000-01-03	AAPL	0.847207	...	0.936384	535796800.0	535.497815
1		2000-01-03	MSFT	36.132267	...	58.687500	53228400.0	3102.217688
2		2000-01-04	AAPL	0.775779	...	0.966518	512377600.0	468.917227
3		2000-01-04	MSFT	34.911720	...	56.781250	54119000.0	3047.576187
4		2000-01-05	AAPL	0.787131	...	0.926339	778321600.0	722.726855
...		...	...	...	...	...	...	...
12085		2024-01-09	MSFT	375.790009	...	372.010010	20830000.0	7827.705878

# Next steps

- Select or drop columns

```
prices |> select(symbol, date, adjusted) # Only select symbol, date, adjusted
prices |> select(-close) # Drop close price
```

- Working with date format

```
prices <- prices |> mutate(year = year(date))
```

- Joining different tables (too fast? Enjoy this illustration)

```
prices_sp500 <- tq_get("^GSPC", from = "2000-01-01") |> select(date, market = adjusted)
joint_prices <- left_join(prices, prices_sp500, join_by(date))
joint_prices |> select(symbol, date, adjusted, market)
```

```
# A tibble: 12,090 x 4
  symbol date      adjusted market
  <chr>  <date>      <dbl>  <dbl>
1 AAPL  2000-01-03    0.847  1455.
2 AAPL  2000-01-04    0.776  1399.
3 AAPL  2000-01-05    0.787  1402.
4 AAPL  2000-01-06    0.719  1403.
# i 12,086 more rows
```

# Basic exploratory data analysis

- count, group\_by and summarise solve *many* data science questions
- *How many daily observations per symbol?*

```
prices |> count(symbol)
```

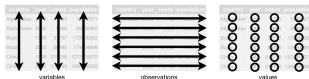
```
# A tibble: 2 x 2
  symbol      n
  <chr>   <int>
1 AAPL    6045
2 MSFT    6045
```

- *Which ticker had the highest average daily volume (in USD)?*

```
average_volume <- prices |>
  group_by(symbol) |>
  summarise(avg_volume = mean(volume_usd))
average_volume |> arrange(desc(avg_volume))
```

```
# A tibble: 2 x 2
  symbol avg_volume
  <chr>       <dbl>
1 AAPL      5425.
2 MSFT      2930.
```

# Tidy data: “grammar” of data science workflows



- Did you ever think about what makes data easy to work with?
- Clear data structure to ensure similar workflows for everyday tasks
- Tidy data is a standard way of mapping the meaning of a dataset to its structure
- In tidy data, each variable forms a column. Each observation forms a row. Each cell is a single measurement

```
avg_vol_ts <- prices |> group_by(year, symbol) |> summarise(avg_volume = mean(volume_usd)) |>
  pivot_wider(names_from = symbol, values_from = avg_volume)
avg_vol_ts # One year per row, two observations per year
```

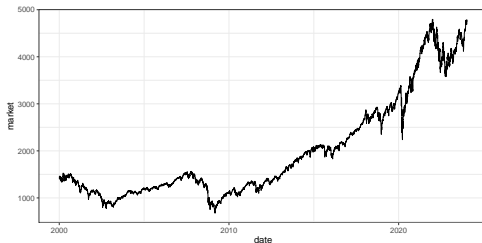
```
# A tibble: 25 x 3
# Groups:   year [25]
  year AAPL MSFT
<dbl> <dbl> <dbl>
1  2000 353. 2980.
2  2001 139. 2359.
3  2002 106. 2059.
4  2003 92.8 1689.
# i 21 more rows
```

```
# Back to the initial setup
avg_vol_ts |> pivot_longer(cols = -year, names_to = "symbol", values_to = "avg_volume")
```

# Visualization with the tidyverse: ggplot2

- ggplot2: Grammar of graphics differentiates between the data and the representation
- Data (data frame being plotted)
- Geometrics (geometric shape that represents the data: point, boxplot, histogram)
- Aesthetics (color, size, shape)

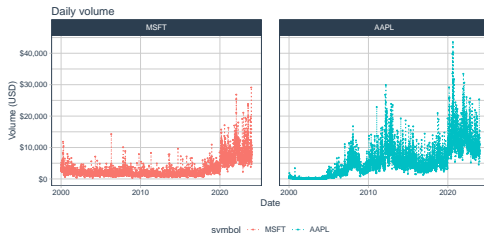
```
ggplot(data = prices_sp500) + aes(x = date, y = market) + geom_line()
```



```
prices_sp500 |> ggplot() + aes(x = date, y = market) + geom_point()  
prices_sp500 |> ggplot() + aes(x = date, y = market) + geom_point(color = "red")
```

# ggplot2 - (Many) other options

```
p <- prices |> ggplot() + aes(x = date, y = volume_usd, color = symbol) +  
  geom_point(size = 0.2) + geom_line(linetype = "dotted")  
  
library(scales) # for scale labeling  
p + labs(x = "Date", y = "Volume (USD)", title = "Daily volume") +  
  facet_wrap(~symbol, scales = "free_x") + theme_tq() +  
  scale_y_continuous(labels = dollar)
```



- Many more themes here: <https://ggplot2.tidyverse.org/reference/ggtheme.html>
- Virtually unlimited possibilities, see Cedric Scherer's blog

# Tidy coding principles in a nutshell

---

- Great code is very subjective but the aim should be to make it human-readable
- In this course I want you to think and reflect about *what* makes code useful?

## Core principles to make code readable

1. chaining ( `|>` or `.` )
2. intuitive naming conventions (`trading_volume_usd` instead of `tmp_Var_2`)
3. Tidy data principles with a clear data structure
4. Embed code directly into the analysis with Quarto

## Quarto?

- Suppose you write a seminar paper that requires you to analyze some data such that you can state some summary statistics in your document. **How would you do this, typically?**
- You can weave narrative text and code together to produce elegantly formatted output as documents, web pages, blog posts, books, and more
- No more copy-paste, no more manually rebuilding analyses from disparate components, and no more dread when the data is updated and you need to run an analysis



# Language-agnostic tidy coding principles

```
# R
returns <- prices |>
  filter(symbol == "AAPL") |>
  arrange(date) |>
  mutate(ret = adjusted / lag(adjusted) - 1) |>
  select(symbol, date, ret) |>
  drop_na(ret)
returns
```

```
# A tibble: 6,044 x 3
  symbol date      ret
  <chr> <date>      <dbl>
1 AAPL  2000-01-04 -0.0843
2 AAPL  2000-01-05  0.0146
3 AAPL  2000-01-06 -0.0865
4 AAPL  2000-01-07  0.0474
# i 6,040 more rows
```

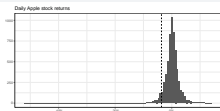
```
# Python
returns = (prices
  .query('symbol == "AAPL"')
  .sort_values("date")
  .assign(ret = lambda x: x["adjusted"].pct_change())
  .get(["symbol", "date", "ret"])
  .dropna())
returns.head(4)
```

	variable	symbol	date	ret
2		AAPL	2000-01-04	-0.084310
4		AAPL	2000-01-05	0.014633
6		AAPL	2000-01-06	-0.086538
8		AAPL	2000-01-07	0.047369

# Language-agnostic visualization with tidy coding principles

```
# R
quantile_05 <- quantile(returns$ret,
                        probs = 0.05)

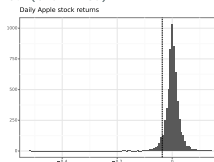
returns |>
  ggplot(aes(x = ret)) +
  geom_histogram(bins = 100) +
  geom_vline(aes(xintercept = quantile_05),
             linetype = "dashed") +
  labs(x = NULL, y = NULL,
       title = "Daily Apple stock returns"
  ) +
  scale_x_continuous(labels = percent)
```



```
# Python
quantile_05 = returns["ret"].quantile(0.05)

(ggplot(returns, aes(x="ret")) +
  geom_histogram(bins=100) +
  geom_vline(aes(xintercept=quantile_05),
             linetype="dashed") +
  labs(x="", y="",
       title="Daily Apple stock returns"
  )
)
```

<Figure Size: (640 x 480)>



# The most important slide: How to get help online and from peers

---

## Try to stick to the following roadmap if you encounter problems

1. If you do not know what a command, e.g., `cov()` is doing, type `help(cov)` in the console
2. Search online, e.g. *R cov* - ChatGPT will know
3. Use the discussion board in Absalon (you are also *very* welcome to provide answers to your questions after they have been solved)
4. Check out the tidyverse cheatsheets

## ChatGPT and Github Copilot

1. I **recommend** using ChatGPT and the like throughout the course for code-related questions
2. I expect that you invest the time you save in making clever use of AI to ensure that your code and assignments are of outstanding quality