Advanced Empirical Finance: Topics and Data Science

Stefan Voigt

Spring 2024

University of Copenhagen and Danish Finance Institute

Asset prices and returns

Asset prices and returns

Topics

- Stock market returns
- Diversification and risk management

General framework and notation

- We consider N assets, time series of prices $P_{t,i}$ of asset i at time t
- Net returns are $r_{t+1,i} := (P_{t+1,i} - P_{t,i})/P_{t,i}$
- Gross returns are $R_{t+1,i} := P_{t+1,i}/P_{t,i} = 1 + r_{t+1,i}$
- I refer to log returns as $\tilde{r}_{t+1,i} := \log(P_{t+1,i}) - \log(P_{t,i})$
- Dividends: modify the definitions, e.g., $\tilde{r}_{t+1} = \log(P_{t+1,i} + D_{t+1,i}) - \log(P_{t,i})$

```r
library(tidyverse)
library(tidyquant)
prices_sp500 <- tq_get("^GSPC", from = "2000-01-01") |> select(date, adjusted)
prices_sp500 <- prices_sp500 |> mutate(net_return = (adjusted - lag(adjusted)) / lag(adjusted),
                                        gross_return = adjusted / lag(adjusted),
                                        log_return = log(adjusted) - log(lag(adjusted)))
head(prices_sp500)
```

```
# A tibble: 6 x 5
  date       adjusted net_return gross_return log_return
  <date>        <dbl>      <dbl>        <dbl>      <dbl>
1 2000-01-03    1455.  NA               NA       NA
2 2000-01-04    1399.  -0.0383           0.962  -0.0391
3 2000-01-05    1402.   0.00192          1.00     0.00192
4 2000-01-06    1403.   0.000956         1.00     0.000955
# i 2 more rows
```

```r
prices_sp500 <- prices_sp500 |> drop_na() # Remove NAs
```

Distributional properties of returns

Moments of the return distribution

- Define $\mu = E(r_{t+1})$ and $\Sigma = \text{Cov}(r_{t+1}) = E((r_{t+1} - \mu)(r_{t+1} - \mu)')$
- **Later:** $\mu_t := E(r_{t+1}|F_t)$ and $\Sigma_t = \text{Cov}(r_{t+1}|F_t)$ where F_t denotes the available information at t

Sample moments

- Suppose you have T observations of the $(N \times 1)$ vector, $r_1, \ldots, r_t, \ldots, r_T$
- The sample counterparts $\hat{\mu}$ and $\hat{\Sigma}$ are

$$\hat{\mu} = \frac{1}{T}\sum_{t=1}^{T} r_t \text{ and } \hat{\Sigma} = \frac{1}{T-1}\sum_{t=1}^{T}\left((r_t - \hat{\mu})(r_t - \hat{\mu})'\right)$$

```r
returns <- prices_sp500 |> pull(net_return)
c(sum(returns)/length(returns), mean(returns)) # (daily) average
```

```
[1] 0.00028 0.00028
```

```r
c(sum((returns - mean(returns))^2)/(length(returns) - 1), var(returns)) # (daily) variance
```

```
[1] 0.000152 0.000152
```

Distributional properties of returns

- Estimate the sample variance-covariance matrix $\hat{\Sigma}$

```
prices <- tq_get(c("META", "AAPL", "MSFT"), from = "2000-01-01")
price_matrix <- prices |>
  group_by(symbol) |> # Net returns per symbol
  transmute(date, return = (adjusted - lag(adjusted)) / lag(adjusted)) |>
  pivot_wider(names_from = symbol, values_from = return) |>
  drop_na() |> select(-date)
(sigma <- price_matrix |> cov())
```

```
         META     AAPL     MSFT
META 0.000650 0.000197 0.000199
AAPL 0.000197 0.000319 0.000182
MSFT 0.000199 0.000182 0.000279
```

- Volatility σ is the standard deviation (the square root of the variance σ^2)
- For daily data, the annualized volatility and mean are $\approx \sqrt{250}\hat{\sigma}$ and $\approx 250\hat{\mu}$

```
bind_rows(100 * sqrt(250 * diag(sigma)), # Annualized volatility (in percent),
          100 * 250 * colMeans(price_matrix)) # Annualized return (in percent))
```

```
# A tibble: 2 x 3
   META  AAPL  MSFT
  <dbl> <dbl> <dbl>
1  40.3  28.2  26.4
2  29.4  24.9  27.8
```

Optimal portfolio allocation

Optimal (static) portfolio choice

- **Aim:** Choose $(N \times 1)$ vector ω such that $\sum_{t=1}^{T} \omega_i = \iota' \omega = 1$ where ι is an $(N \times 1)$ vector of ones
- Portfolio returns $r_t^{pf} = \omega' r_t$
- Common properties of utility functions (concave)

$$U'(r_t) > 0 \text{ and } U(E(r_t)) > E(U(r_t))$$

- Preference for higher expected return and lower volatility $\sigma^{pf} = \sqrt{\text{Var}\left(r_t^{pf}\right)}$

$$E(r_t^{pf}) = \omega' \mu \qquad \begin{aligned}(\sigma^{pf})^2 &= E\left(\omega'(r_t - \mu)(r_t - \mu)'\omega\right) \\ &= E\left(tr\left(\omega'(r_t - \mu)(r_t - \mu)'\omega\right)\right) \\ &= tr\left(\omega' E\left((r_t - \mu)(r_t - \mu)\right)' \omega\right) = \omega' \Sigma \omega\end{aligned}$$

Minimum variance portfolio

- The minimum variance portfolio weights are given by the solution to

$$\omega_{\text{mvp}} = \arg\min \omega'\Sigma\omega \text{ s.t. } \iota'\omega = 1$$

- The Lagrangian reads

$$L(\omega) = \omega'\Sigma\omega - \lambda(\omega'\iota - 1)$$

- Analytic solution by solving the first-order conditions of the Lagrangian equation

$$\frac{\partial L(\omega)}{\partial\omega} = 0 \Leftrightarrow 2\Sigma\omega = \lambda\iota \Rightarrow \omega = \frac{\lambda}{2}\Sigma^{-1}\iota$$

- Constraint delivers: $1 = \iota'\omega = \frac{\lambda}{2}\iota'\Sigma^{-1}\iota \Rightarrow \lambda = \frac{2}{\iota'\Sigma^{-1}\iota}$

$$\Rightarrow \omega_{\text{mvp}} = \frac{\Sigma^{-1}\iota}{\iota'\Sigma^{-1}\iota}$$

- Variance of the minimum variance portfolio return is $\omega_{\text{mvp}}'\Sigma\omega_{\text{mvp}} = \frac{1}{\iota'\Sigma^{-1}\iota}$

Computing the minimum variance portfolio

- Suppose you know the variance-covariance matrix, e.g.,

$$\Sigma = \begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix} \text{ with } \Sigma^{-1} = \frac{1}{8}\begin{pmatrix} 4 & -2 \\ -2 & 3 \end{pmatrix}$$

- Then you can compute the minimum variance portfolio weights as follows

```
# R
Sigma <- matrix(c(3, 2, 2, 4), ncol = 2) # Define 2 x 2 matrix Sigma
Sigma_inv <- solve(Sigma) # Invert the matrix
w <- Sigma_inv %*% rep(1, 2) # %*% is matrix multiplication in R
w <- w / sum(w)

t(w) # t() transposes a vector/matrix
```

```
      [,1]  [,2]
[1,] 0.667 0.333
```

```
# Python
import numpy as np

Sigma = np.array([[3, 2],
                  [2, 4]]) # Define 2 x 2 matrix Sigma
Sigma_inv = np.linalg.inv(Sigma) # Invert the matrix
w = Sigma_inv @ np.ones(2)

w/w.sum()
```

```
array([0.66666667, 0.33333333])
```

- The minimum variance portfolio weights are thus $\omega_{mvp} = \frac{1}{3}\begin{pmatrix} 2 & 1 \end{pmatrix}$
- Which practical issues are important here?

The efficient portfolio

- Consider an investor who aims to achieve minimum variance *given a desired expected return* $\bar{\mu}$

$$\omega_{\text{eff}}(\bar{\mu}) = \arg\min \omega'\Sigma\omega \text{ s.t. } \iota'\omega = 1 \text{ and } \omega'\mu \geq \bar{\mu}$$

- The Lagrangian reads

$$L(\omega) = \omega'\Sigma\omega - \lambda(\omega'\iota - 1) - \tilde{\lambda}(\omega'\mu - \bar{\mu})$$

- Solve the first-order conditions

$$2\Sigma\omega = \lambda\iota + \tilde{\lambda}\mu$$

$$\omega = \frac{\lambda}{2}\Sigma^{-1}\iota + \frac{\tilde{\lambda}}{2}\Sigma^{-1}\mu$$

The efficient portfolio

- The two constraints ($\omega'\iota = 1$ and $\omega'\mu \geq \bar{\mu}$) imply

$$1 = \iota'\omega = \frac{\lambda}{2}\underbrace{\iota'\Sigma^{-1}\iota}_{C} + \frac{\tilde{\lambda}}{2}\underbrace{\iota'\Sigma^{-1}\mu}_{D} \Rightarrow \lambda = \frac{2 - \tilde{\lambda}D}{C}$$

$$\begin{aligned}\bar{\mu} = \mu'\omega &= \frac{\lambda}{2}\underbrace{\mu'\Sigma^{-1}\iota}_{D} + \frac{\tilde{\lambda}}{2}\underbrace{\mu'\Sigma^{-1}\mu}_{E} = \frac{1}{2}\left(\frac{2 - \tilde{\lambda}D}{C}\right)D + \frac{\tilde{\lambda}}{2}E \\ &= \frac{D}{C} + \frac{\tilde{\lambda}}{2}\left(E - \frac{D^2}{C}\right)\end{aligned}$$

$$\Rightarrow \tilde{\lambda} = 2\frac{\bar{\mu} - D/C}{E - D^2/C}$$

- As a result, the efficient portfolio weight takes the form (for $\bar{\mu} \geq D/C = \mu'\omega_{\text{mvp}}$)

$$\omega_{\text{eff}}(\bar{\mu}) = \omega_{\text{mvp}} + \frac{\tilde{\lambda}}{2}\left(\Sigma^{-1}\mu - \frac{D}{C}\Sigma^{-1}\iota\right)$$

The efficient portfolio

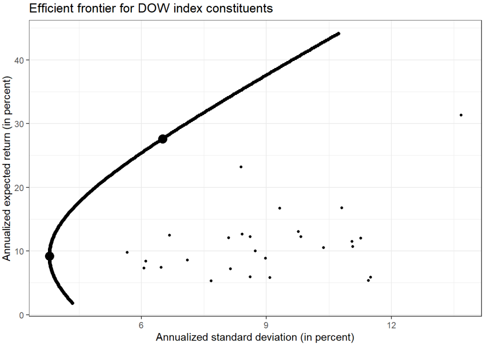
- Note that
$$\iota'\left(\Sigma^{-1}\mu - \frac{D}{C}\Sigma^{-1}\iota\right) = D - D = 0 \text{ so } \iota'\omega_{\text{eff}} = \iota'\omega_{\text{mvp}} = 1$$
and
$$\mu'\omega_{\text{eff}} = \frac{D}{C} + \bar{\mu} - \frac{D}{C} = \bar{\mu}$$
- The efficient portfolio allocates wealth in the minimum variance portfolio ω_{mvp} and a levered (self-financing) portfolio to increase the expected return

Two mutual fund theorem

- Assume you computed $\omega_{\text{eff}}(\bar{\mu})$ and $\omega_{\text{eff}}(\tilde{\mu})$ for $\bar{\mu} > \tilde{\mu} \geq D/C$, then any linear combination with $c \in \mathbb{R}_+$ can be represented as
$$\omega^* = c\omega_{\text{eff}}(\bar{\mu}) + (1-c)\omega_{\text{eff}}(\tilde{\mu}) = \omega_{\text{mvp}} + \frac{\lambda^*}{2}\left(\Sigma^{-1}\mu - \frac{D}{C}\Sigma^{-1}\iota\right)$$
with $\lambda^* = 2\frac{c\bar{\mu}+(1-c)\tilde{\mu}-D/C}{E-D^2/C}$.
- As a result, any portfolio of two efficient portfolios is also efficient

The efficient frontier (coding exercise)

- Download historical price data for *all* stocks that are part of the Dow Jones index and obtain estimates $\hat{\mu}$ and $\hat{\Sigma}$.
- Compute ω_{mvp} and arbitrary $\omega_{\text{eff}}(\bar{\mu})$
- Then, apply the two-fund theorem to characterize the *entire* efficient frontier



The efficient frontier with a risk-free rate

- Assume there is a $N+1$-th asset which offers a risk-free rate $r_f > 0$ and has zero volatility
- The investor allocates fractions x of wealth to the risky assets and the remainder $(1-\iota' x)$ to the risk-free asset with portfolio return

$$r_t^{\text{pf}} = x' r_t + (1-\iota' x) r_f = r_f + x'(r_t - r_f \iota)$$

- The mean-variance problem can be expressed as

$$\min x'\Sigma x \text{ s.t. } x'(\mu - r_f) \geq \bar{\mu}$$

- The solution is much simpler than before:

$$x^* = \frac{\bar{\mu}}{(\mu - r_f)'\Sigma^{-1}(\mu - r_f)}\Sigma^{-1}(\mu - r_f)$$

- The resulting portfolio of investments in risky assets, $\frac{x^*}{\iota' x^*}$ is the **efficient tangent portfolio**

$$\omega_{\text{tgc}} = \frac{\Sigma^{-1}(\mu - r_f)}{\iota'\Sigma^{-1}(\mu - r_f)}$$

- Why does $\bar{\mu}$ not show up in ω_{tgc}?

Empirical problems (Discussion)

Let's move from theory to practice:

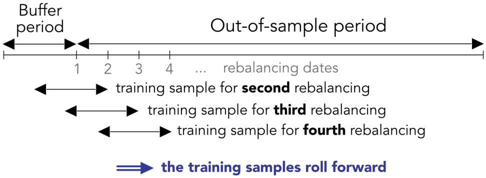
1. Which decisions do you as a portfolio manager have to take?
2. What issues may occur and lead to deviations from the theoretically optimal portfolio?
3. How do you evaluate your choices?

Portfolio backtesting

- Portfolio backtesting is often perceived as a quest to find the best strategy or at least a solidly profitable one (downside: data snooping, **p**-hacking)
- Out-of-sample test to analyze the (hypothetical) performance of a strategy

Procedure

1. Fix investment horizon N, sample period T and estimation window size h
2. Out-of-sample: Never use information an investor would not have at the time of decision
3. For each period, recompute $\hat{\Sigma}$ and $\hat{\mu}$ and reallocate wealth
4. At the end of each period store the portfolio performance (e.g. return)
5. Compare different strategies by evaluating the average out-of-sample performance



Evaluation metrics

- Typical metrics are the out-of-sample portfolio return (eventually risk-adjusted)

$$\hat{E}(r^{pf}) = \frac{1}{T-h-1} \sum_{t=h+1}^{T} r_t^{pf}$$

  Note that $r_t^{pf} = \omega_t' r_{t+1}$ where ω_t denotes the weights based on information available up to time t
- Portfolio volatility $\sqrt{\hat{\sigma}^2 \left(r^{pf}\right)}$
- Later we will also consider transaction costs which depend on rebalancing from ω_{t-1} to ω_t

Rolling windows in R and Python

- You will need `for` loops to mimic sliding through time
- `for` loops provide a way to tell, “Do *this* for every value of *that*.” In R and Python syntax, this looks like this:

```r
# R
for (value in c("My", "first", "for", "loop")) {
  print(value)
}
```

```
[1] "My"
[1] "first"
[1] "for"
[1] "loop"
```

```python
# Python
for value in ["My", "first", "for", "loop"]:
    print(value)
```

```
My
first
for
loop
```

- Task: Develop pseudo-code for portfolio backtesting

Functions in R and Python

- Writing a function has three big advantages over using copy-and-paste:

1. Functions with an evocative name make your code easier to understand
2. As requirements change, you only need to update the code in one place
3. You eliminate the chance of making incidental mistakes when you copy and paste

- Keep in mind: functions are for computers but also *for humans* to make clear code
- Use names that explain what the function is doing and make use of comments `#` for readers (including future-you)

```r
# R
compute_sum <- function(x, y){
  return(x+y)
}
```

```python
# Python
def compute_sum(x, y):
    return x + y
```

3 key steps to creating a new function

- Pick a meaningful name for the function
- List the inputs to the function inside `function(...)`
- Place code in the body of the function, a `{ }` block that immediately follows `function(...)`

```
compute_efficient_portfolio <- function(sigma, mu, mu_bar = 0.30 / 250){
  iota <- rep(1, ncol(sigma))
  sigma_inv <- solve(sigma)
  w_mvp <- sigma_inv %*% iota
  w_mvp <- w_mvp / sum(w_mvp)
  C <- as.numeric(t(iota)%*%sigma_inv%*%iota)
  D <- as.numeric(t(iota)%*%sigma_inv%*%mu)
  E <- as.numeric(t(mu)%*%sigma_inv%*%mu)
  lambda_tilde <- as.numeric(2*(mu_bar -D/C)/(E-D^2/C))
  weff <- w_mvp + lambda_tilde / 2 * (sigma_inv%*%mu - D/C*sigma_inv%*%iota)
  return(t(weff))
}
compute_efficient_portfolio(sigma = sigma, mu = colMeans(price_matrix))
```

```
      META   AAPL MSFT
[1,] 0.432 -0.525 1.09
```

```
compute_efficient_portfolio(sigma, colMeans(price_matrix), mu_bar = 0.25 / 250)
```

```
       META AAPL  MSFT
[1,] -0.121 0.91 0.211
```

Parameter uncertainty

- Consider a quadratic utility function with risk aversion γ and certainty equivalent

$$CE(\omega) = \omega'\mu - \frac{\gamma}{2}\omega'\Sigma\omega$$

- Maximum expected utility portfolio (under the constraint $\iota'\omega = 1$) is equivalent to the framework above (minimize volatility for a given level of return $\bar{\mu}$) (the proof is an exercise: show that there is a bijective mapping from γ to $\bar{\mu}$)
- Optimal investment: efficient portfolio $\omega_\gamma(\mu, \Sigma)$
- Econometrician can only invest in $\omega_\gamma(\hat{\mu}, \hat{\Sigma})$
- As a result: Inefficient wealth allocation

How much does parameter uncertainty matter?

- Certainty equivalent loss is defined as

$$CEL = CE(\omega_\gamma(\mu, \Sigma)) - E\left(CE(\omega_\gamma(\hat{\mu}, \hat{\Sigma}))\right) \geq 0$$

- Cho (2007) shows that the loss can be approximated by

$$CEL \approx \frac{\gamma}{2} \times tr\left(\text{cov}(\omega_\gamma(\hat{\mu}, \hat{\Sigma}))\hat{\Sigma}\right)$$

- Depends on the risk-aversion γ, the variance-covariance matrix of $\omega_{\text{eff}}(\hat{\mu}, \hat{\Sigma})$ and the return covariance
- Kan and Zhou (2007) and DeMiguel (2009) consider different portfolio weights that reduce the certainty equivalent loss
- It can be shown that the naive portfolio $\frac{1}{N}\iota$ can be optimal if estimation (and model) uncertainty is huge

How imposing restrictions helps (in theory)

- Consider the Kuhn-Tucker conditions for the minimum variance portfolio with short-selling constraints

$$\Sigma\omega - \lambda = \lambda^0 \iota \text{ where } \lambda_i \geq 0 \text{ and } \lambda_i = 0 \text{ if } \omega_i > 0$$

- Let $\omega_{\text{no short}}$ denote the solution to the problem above
- Set $\tilde{\Sigma} = \Sigma - (\lambda\iota' + \iota\lambda')$. Then, $\omega_{\text{no short}}$ is the (unconstrained) minimum variance portfolio for $\tilde{\Sigma}$ because:

$$\begin{aligned}\tilde{\Sigma}\omega_{\text{no short}} &= \Sigma\omega_{\text{no short}} - \lambda \underbrace{\iota'\omega_{\text{no short}}}_{=1} + \iota \underbrace{\lambda'\omega_{\text{no short}}}_{=0} \\ &= \Sigma\omega_{\text{no short}} - \lambda = \lambda^0 \iota\end{aligned}$$

- What is so special about $\tilde{\Sigma}$ instead of Σ? Consider the (unconstrained) first-order condition again. If stock i's marginal contribution to the portfolio variance is large, the weights ω_i are reduced (become negative)

$$\Sigma\omega = \lambda_0 \iota \Leftrightarrow \sum_{j=1}^{N} \omega_j \Sigma_{i,j} = \lambda_0 \forall i$$

- With $\tilde{\Sigma}$, assets which would imply negative weights are treated as if the variance is reduced by $2\lambda_i$ and covariance with asset j is reduced by $\lambda_i + \lambda_j$

How imposing restrictions helps (in R)

- Two commonly used approaches are naive diversification $\omega_{\text{naive}} = \frac{1}{N}\iota$ and short-sell constrained portfolios

$$\omega_{\text{no short}} = \arg\max \omega'\mu - \frac{\gamma}{2}\omega'\Sigma\omega \quad \text{s.t. } \omega'\iota = 1 \text{ and } \omega_i \geq 0 \quad \forall i$$

- No closed-form solution exists to compute $\omega_{\text{no short}}$
- Function `solve.QP` from package `quadprog` delivers the numerical solution to quadratic programming problems of the form

$$\min(-\mu'\omega + 1/2\omega'\Sigma\omega) \text{ s.t. } A'\omega >= b_0$$

- The function takes argument `meq` for the number of equality constraints
- A for $\omega_{\text{no short}}$ is $(N + 1 \times N)$ and of the form (`meq` = 1)

$$A = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ & & & \\ 0 & 0 & \dots & 1 \end{pmatrix}' \qquad b_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \\ 0 \end{pmatrix}$$

- For more complex optimization routines in R, this link (optimization task view) helps

How imposing restrictions helps (in R)

- The code snippet below shows how to implement quadratic programming

```r
# install.packages("quadprog")
N <- ncol(sigma)
A <- t(rbind(1,                    # vector of ones
             diag(N)))             # identity matrix
cbind(t(A), c(1, rep(0, N)))
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    1    0    0    0
[3,]    0    1    0    0
[4,]    0    0    1    0
```

```r
solution <- quadprog::solve.QP(Dmat = 2 * sigma, # 2 resembles gamma
                               dvec = colMeans(price_matrix),
                               Amat = A,
                               bvec = c(1, rep(0, N)),
                               meq = 1)
names(solution) # Output is a list of relevant values. You can access list elements with $
```

```
[1] "solution"                "value"                   "unconstrained.solution"
[4] "iterations"              "Lagrangian"              "iact"
```

```r
solution$solution
```

```
[1] 0.1946 0.0917 0.7137
```

Empirical evidence for the CAPM

Main implication: Capital asset pricing model

- **If** all investors share beliefs about Σ and μ **and** can borrow and invest without limits at r_f, **everybody** invests a fraction of her wealth in ω_{tgc} and in the risk-free rate (two mutual fund theorem)
- The tangency portfolio is the market portfolio ω_{m} and the individual weights of asset i are just

$$\omega_{\text{tgc, i}} = \omega_{\text{m, i}} = \frac{P_i SCO_i}{\sum_{j=1}^{N} P_j SCO_j}$$

  where SCO_j is the number of shares outstanding of a stock i.
- To align the two results, the capital asset pricing model imposes constraints for the expected return of stock i

$$E(r_i) - r_f = \underbrace{\frac{Cov(r_i, r_m)}{\text{Var}(r_m)}}_{\beta_i} E(r_m - r_f)$$

- Expected returns are entirely determined by the price of risk (market risk premium) and the comovement of asset i with the market, β_i.

The US Stock market

- Large parts of the academic literature focus on US stock markets
- Stocks are listed on US exchanges (NYSE, AMEX, NASDAQ, and some smaller ones)
- Extensive data on prices and trading activity is provided by the Center for Research in Security Prices (CRSP), maintained by the University of Chicago, Booth School of Business
- Full sample ranges from December 1925 and is continuously updated
- A large part of the following computations is based on the textbook *Empirical Asset Pricing: The Cross Section of Stock Returns* (available via library from reading list in Absalon)

How to get the data

- Note that in the textbook, we explain how to get CRSP data from the WRDS interface. This does not work on the KU campus!
- Consult Absalon for a detailed description of the CRSP sample. As KU students you have direct access to the data. Familiarize yourself with CRSP.

Composition of the CRSP sample (Chapter 7, Bali, Murray, Engle)

- Familiarize yourself with the cleaning steps in the exercise on the CRSP sample
- Monthly processed (!) data is available in `tidy_finance_r.sqlite` and `tidy_finance_python.sqlite` file in Absalon
- Only contains US stocks (`shrcd%in%c(10,11)`)
- Variable `exchcd` determines listing exchanges, `siccd` lists industry

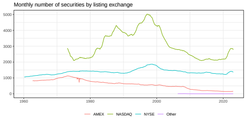
```r
library(RSQLite)
tidy_finance <- dbConnect(SQLite(), "../data/tidy_finance_r.sqlite", extended_types = TRUE) # Connect to sql
crsp_monthly <- tbl(tidy_finance, "crsp_monthly") |> collect()
```

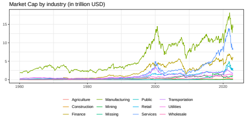
```python
import pandas as pd
import sqlite3

tidy_finance_python = sqlite3.connect(
  database="../data/tidy_finance_python.sqlite"
)

crsp_monthly = (pd.read_sql_query(
  sql=("SELECT permno, month, industry, ret_excess "
       "FROM crsp_monthly"),
  con=tidy_finance_python,
  parse_dates={"month"})
  .dropna()
)
```

Industry classification

- We adjust the market capitalization values for inflation using the Consumer Price Index (CPI, All Urban Consumer series) from the Bureau of Labor Statistics website.

```
crsp_monthly |> count(exchange, date) |>
  ggplot(aes(x = date, y = n, color = exchange)) + geom_line() +
  labs(x = NULL, y = NULL, color = NULL, title = "Monthly number of securities by listing exchange")
crsp_monthly |>
  left_join(tbl(tidy_finance, "cpi_monthly") |> collect(), join_by(month)) |>
  group_by(month, industry) |>
  summarize(mktcap = sum(mktcap / cpi) / 1000000) |> ungroup() |>
  ggplot(aes(x = month, y = mktcap, color = industry)) + geom_line() +
  labs(x = NULL, y = NULL, color = NULL, title = "Market Cap by industry (in trillion USD)")
```





Compute returns in the CRSP sample

- Monthly stock returns are found in the `RET` field in CRSP
- Problems occur for delistings which require (tedious) adjustments
- Bali, Murray, Engle explain these adjustments (see code snippet below)

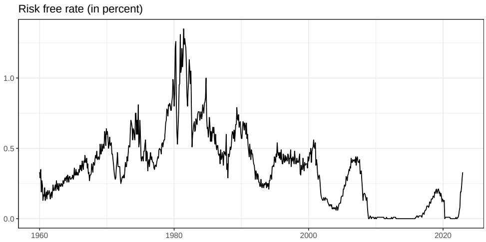
```
raw_crsp_monthly |> # Process Raw file (sqlite contains processed data already)
  mutate(ret_adj = case_when(
    is.na(dlstcd) ~ ret,
    !is.na(dlstcd) & !is.na(dlret) ~ dlret,
    dlstcd %in% c(500, 520, 580, 584) | (dlstcd >= 551 & dlstcd <= 574) ~ -0.30,
    dlstcd == 100 ~ ret,
    TRUE ~ -1)) |>
  select(-c(dlret, dlstcd))
```

- *Note*: The `tidy_finance_*.sqlite` data already contains the processed CRSP file with excess and raw returns. For the sake of simplicity, I removed the delisting information (consult the exercises for more details)

Risk-free rate

- Excess return for a stock is the difference between the stock return and the return on the risk-free security over the same period
- Monthly risk-free security return data from Ken French's data library

```
factors_ff_monthly <- tbl(tidy_finance, "factors_ff3_monthly") |> collect()
factors_ff_monthly |> ggplot(aes(x = month, y = 100 * rf)) +
  geom_line() +  labs(x = NULL, y = NULL, title = "Risk free rate (in percent)")
```



Excess returns

- We mainly work with adjusted *excess* returns
- The table below illustrates the time-series averages of cross-sectional means from the entire CRSP sample

```
crsp_monthly |>
  group_by(month) |>
  summarise(across(ret_excess,
                   list(mean = mean, sd = sd, min = min,
                        q25 = ~quantile(., 0.25),
                        median = median,
                        q75 = ~quantile(., 0.75), max = max),
                   .names = "{.fn} return")) |>
  summarise(across(-month, mean)) |>
  kableExtra::kable()
```

mean return	sd return	min return	q25 return	median return	q75 return	max return
0.008	0.154	-0.695	-0.064	-0.003	0.063	2.77

The market factor

- The market risk premia is the market excess return $z_t = r_{m,t} - r_{f,t}$
- We proxy for the market as the value-weighted portfolio of all US-based common stocks in CRSP
- This aggregation is already provided by Kenneth French on his homepage
- Sharpe ratio is computed as $\frac{\hat{\mu}_z}{\hat{\sigma}_z}$

```
factors_ff_monthly |>
    mutate(mkt_excess = 100 * mkt_excess) |>
    summarise(across(mkt_excess, list(mean = ~ 12 * mean(.),
                                     sd = ~sqrt(12) * sd(.),
                                     Sharpe = ~sqrt(12) * mean(.)/sd(.)),
                     .names = "{.fn} (annualized)")) |>
  kableExtra::kable()
```

mean (annualized)	sd (annualized)	Sharpe (annualized)
6.55	15.5	0.422

Estimating the CAPM

- The CAPM of Sharpe (1964), Lintner (1965), and Mossin (1966) originates the literature on asset pricing models
- The CAPM is an equilibrium model in a single-period economy
- Asset risk is the covariance of its return with the market portfolio return
- The higher the co-movement the less desirable the asset is, hence, the asset price is lower and the expected return is higher
- Risk premium, or the market price of risk, is the expected value of the excess market return
- The market price of risk, common to all assets, is set in equilibrium by the risk aversion of investors

Regression specification of stock *i*

$$r_{i,t} - r_{f,t} = \alpha_i + \underbrace{\frac{Cov(r_i, r_m)}{\mathrm{Var}(r_m)}}_{\beta_i}(r_{m,t} - r_{f,t}) + \varepsilon_{i,t}$$

- For the econometric analysis of the model, we assume that innovations $\varepsilon_{i,t}$ are independently and identically distributed (IID) through time and jointly multivariate normal
- Expected returns are entirely determined by the price of risk (market risk premium) and the co-movement of asset *i* with the market, β_i
- To determine whether the portfolio/asset generates abnormal returns relative to the CAPM risk model, we evaluate whether the fitted intercept coefficient α_i, which serves as the estimate of the average abnormal return per period, is statistically distinguishable from zero
- Discuss: What are the questionable assumptions behind the baseline regression framework?

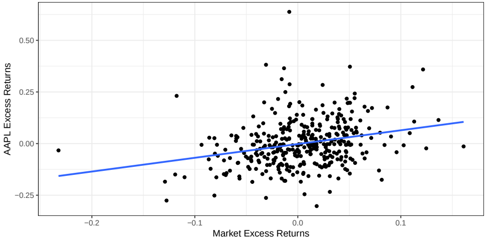
CAPM - Example

- Simple linear regression (OLS) either via `solve(t(X)%*%X)%*%t(X)%*%Y` or `lm()`

```
apple_monthly <- crsp_monthly |> filter(permno == 19764) |> # 19764 = APPLE
  left_join(factors_ff_monthly)
capm_regression <- lm(ret_excess ~ mkt_excess, data = apple_monthly) # linear model
capm_regression |> broom::tidy() |> knitr::kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-0.002	0.006	-0.372	0.71
mkt_excess	0.666	0.129	5.178	0.00

```
apple_monthly |> ggplot(aes(x = mkt_excess, y = ret_excess)) + geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(x = "Market Excess Returns", y = "AAPL Excess Returns")
```



Computing beta for the CRSP universe

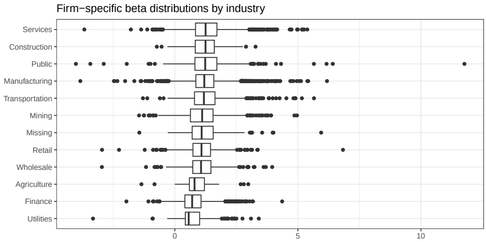
- Market beta for month *t* is estimated with data from prior to and including *t*, e.g. 5 years of monthly data
- Rolling-window regressions are straightforward from a methodological perspective but tricky to implement
- Exercises: conduct rolling window regression of beta for the entire CRSP universe

```r
roll_capm_estimation <- function(data, months, min_obs) {
  data <- data |>
    arrange(month)

  betas <- slide_period_vec(
    .x = data,
    .i = data$month,
    .period = "month",
    .f = ~ estimate_capm(., min_obs),
    .before = months - 1,
    .complete = FALSE
  )

  return(tibble(
    month = unique(data$month),
    beta = betas
  ))
}
```

```python
def roll_capm_estimation(data, window_size, min_obs):

    data = data.sort_values("month")

    result = (RollingOLS.from_formula(
      formula="ret_excess ~ mkt_excess",
      data=data,
      window=window_size,
      min_nobs=min_obs
      )
      .fit()
      .params["mkt_excess"]
    )

    result.index = data.index

    return result
```

CAPM beta in the industry-cross section

```
beta <- tbl(tidy_finance, "beta") |> collect() |>
  inner_join(crsp_monthly, join_by(month, permno)) |> drop_na(beta_monthly)
beta  |> group_by(industry, permno) |> summarise(beta = mean(beta_monthly)) |>
   ggplot(aes(x = reorder(industry, beta, FUN = median), y = beta)) +
  geom_boxplot() + coord_flip() +
  labs(x = NULL, y = NULL, title = "Firm-specific beta distributions by industry"
  )
```



Portfolio sorts with rolling windows for CRSP

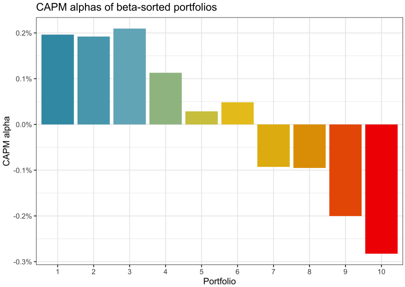
- Discuss: What does the CAPM imply?
- Univariate portfolio analysis (nonparametric technique):

1. Calculate breakpoints (at $t-1$) to divide the sample into portfolios
2. Calculate the average return $r_{p,t}$ within each portfolio for each period t
3. Examine variation in these average values of $r_{p,t}$ across the different portfolios

```
beta_portfolios <- beta |> # A stylized example (note time convention in exercise!)
  group_by(month) |>
  mutate(breakpoint = median(beta_monthly),
         portfolio = case_when(beta_monthly <= breakpoint ~ "low",
                               beta_monthly > breakpoint ~ "high")) |>
  group_by(month, portfolio) |>
  summarize(ret = weighted.mean(ret_excess, mktcap_lag), .groups = "drop")

beta_portfolios |> # Positive alpha?
  pivot_wider(names_from = portfolio, values_from = ret) |>
  mutate(long_short = high - low) |>
  left_join(factors_ff_monthly) |>
  lm(long_short ~ mkt_excess, data = _) |> broom::tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept) -0.00371  0.000953     -3.89 1.11e-  4
2 mkt_excess   0.725    0.0210       34.5  8.17e-154
```

Bad news for CAPM?



- Figure shows decile portfolio sorts based on *lag* beta
- Portfolio 10 corresponds to the highest beta decile
- Each bar corresponds to the CAPM alpha of the value-weighted portfolio performance

How to use (CAPM) factor structure for portfolio optimization

- Suppose the CAPM holds: implied factor structure for expected excess returns is

$$\mu = E(r) = r_f + \beta E(r_m - r_f)$$

$$\Sigma = \sigma_m^2 \beta\beta' + \Sigma_\varepsilon$$

- If Σ_ε is a diagonal matrix, estimation of Σ requires only $2N + 1$ instead of $N(N-1)/2$ parameters
- Practical recipe for portfolio optimization with general factor structure:

1. Estimate β_i for each asset
2. Estimate market risk premium $E(r_m - r_f)$
3. (Univariate) estimation of the elements of Σ_ε based on residualized returns

$$\hat{\varepsilon}_{i,t} = r_{t,i} - r_f - \hat{\beta}_i(r_{m,t} - r_f)$$

4. Replace sample estimates $\hat{\mu}$ and $\hat{\Sigma}$ with the theoretically implied values computed above to choose

$$\omega = \arg\max \hat{\mu}'\omega - \frac{\gamma}{2}\omega'\hat{\Sigma}\omega$$

Testing the CAPM

- Suppose we want to test if the CAPM holds jointly for N assets
- The CAPM implies that all elements of the $(N \times 1)$ vector α are zero in the joint regression framework

$$Z_t = \alpha + \beta Z_{m,t} + \varepsilon_t$$

  where β is the $(N \times 1)$ vector of market betas (if α is zero, then the market portfolio is the tangency portfolio) and $Z_{i,t}$ denotes excess returns

- Standard ordinary least squares (OLS) estimation delivers

$$\hat{\alpha} = \hat{\mu} - \hat{\beta}\hat{\mu}_m$$

  with

$$\hat{\beta} = \frac{\sum(Z_t - \hat{\mu})(Z_{m,t} - \hat{\mu}_m)}{\sum(Z_{m,t} - \hat{\mu}_m)^2} \text{ and } \hat{\mu}_i = \frac{1}{T}\sum_{t=1}^{T} Z_{i,t}$$

- The Wald-test statistic of the null hypothesis $H_0 : \alpha = 0$ is $J = \hat{\alpha}' \left(Var(\hat{\alpha})\right)^{-1} \hat{\alpha}$

Testing the CAPM

- MacKinlay (1987) and Gibbons, Ross, and Shanken (1989) developed the finite-sample distribution of J which yields

$$J = \frac{T - N - 1}{N}\left(1 + \frac{\hat{\mu}_m^2}{\hat{\sigma}_m^2}\right)^{-1} \hat{\alpha}'\hat{\Sigma}^{-1}\hat{\alpha}$$

  where $\hat{\Sigma} = Cov(\hat{\varepsilon})$ and $\hat{\sigma}$ is the standard deviation of the market excess returns.
- Under the null hypothesis, J is unconditionally distributed central F with N degrees of freedom in the numerator and $(T - N - 1)$ degrees of freedom in the denominator

Exercises - compute the test statistics for a sample of portfolio returns

1. Compute the MLE estimates and the test statistic `J <-`
2. Evaluate the p-value `pf(J, T, T - N - 1)` with appropriate degrees of freedom

- For more information, see Chapter 4 of *The Econometrics of Financial Markets*

Fama Mac-Beth regressions

- Instead of focusing on the mean-variance efficiency of the market portfolio, the CAPM also implies a linear relationship between expected returns and market betas which completely explains the cross-section of expected returns
- Portfolio sorts already revealed that this may not be the case
- These implications can also be tested using a cross-sectional regression methodology (Fama and MacBeth, 1973)
- Basic idea: For each cross-section of returns (e.g., each month), project asset returns on factor exposures or characteristics that resemble exposure to a risk factor and then aggregate the estimates in the time dimension
- E.g., first, for each month t, estimate

$$Z_t = \gamma_{0,t}\iota + \gamma_{1,t}\hat{\beta} + \eta_t$$

- Then, we analyse time series of $\hat{\gamma}_{0,t}$ and $\hat{\gamma}_{1,t}$.
- CAPM implies that $E(\gamma_{0,t}) = 0$ (no mispricing) and $E(\gamma_{1,t}) > 0$ (positive market premium)
- In most applications we use $\hat{\beta} \Rightarrow$ errors-in-variables problem (Shanken, 1992)

Unobservability of the market portfolio

- Gibbons, Ross, and Shankens test focuses on the mean-variance efficiency of the market portfolio
- Most tests use a value- or equal-weighted basket of NYSE and AMEX stocks as the market proxy, whereas theoretically, the market portfolio contains all assets
- Roll (1977) emphasizes that tests of the CAPM only reject the mean-variance efficiency of the proxy and that the model might not be rejected if the return on the true market portfolio were used

Implications of the overwhelming evidence against the CAPM?

- Replace CAPM with multifactor models with several sources of risk
- Maybe the evidence against the CAPM is overstated because of mismeasurement of the market portfolio, improper neglect of conditioning information, data-snooping, or sample-selection bias
- What if no risk-based model can explain the anomalies of the stock market behavior (behavioral finance)?

Theoretical drawbacks of the CAPM

- The CAPM assumes that the average investor cares only about the performance of the investment portfolio but wealth could emerge from other sources and higher-order risks could play a role
- The CAPM assumes a static one-period model. In Merton's (1973) ICAPM, the demand for risky assets is attributed not only to the mean-variance component, as in the CAPM, but also to hedging against unfavorable shifts in the investment opportunity set
- Empirically, the poor performance of the single factor CAPM motivated a search for multifactor models