.NET for systematic strategies

A brief recap on systematic strategies

Mnacho Echenim

Grenoble INP-Ensimag

2021-2022

On systematic strategies

Definition

A systematic strategy can be executed without any human intervention

In other words, this is a portfolio management strategy in which an algorithm

- Computes the moments when the portfolio is updated
- Computes the composition of the portfolio

Input : Initial value V_{t_0} of the portfolio (how much to invest)
Input : Market data for the underlying assets
Output: Portfolio values

```
1  begin
2      UpdateCompo(t_0) // nb: there can be additional parameters
3      foreach date t > t_0 do
4          V_t := UpdatePortfolioValue(t)
5          if RebalancingTime(t) then
6              UpdateCompo(t)
7          end
8      end
9      return (V_{t_1}, ..., V_{t_N})
10 end
```

Algorithm 1: Overview

On systematic strategies (2)

Constraint to satisfy:

- The portfolio must be **self-financing**
  - The portfolio value immediately before and after an update must be the same
- It is forbidden to use data ahead of time
  - For example, `UpdateCompo`(t) cannot use any data from time $t + \delta$, where $\delta > 0$

What distinguishes systematic strategies

- When does `RebalancingTime` return **true**?
- How does `UpdateCompo` compute a new composition?

Example 1: uniform strategy

Principle: Construct a portfolio with n assets and every Monday, invest the same amount of cash in each asset

```
RebalancingTime(t)
begin
    return t.Day == Monday
end
```

```
UpdateCompo(t, V_t)
begin
    for i = 1, ..., n do
        q_i ⟵ V_t / (n · S_i^t)
    end
    return (q_1, ..., q_n)
end
```

Example 2: min vol with target return

$$\texttt{OptWeights}(\Sigma, \rho, \tau) \;=\; \min_{\omega \in [0,1]^n} \left(\omega^{\mathrm{T}} \cdot \Sigma \cdot \omega\right) \quad \text{s.t.} \quad \omega^{\mathrm{T}} \rho \geq \tau$$

```
1 UpdateCompo(t, V_t, Σ_t, ρ_t, τ_t)
2 begin
3     ω ⟵ OptWeights(Σ_t, ρ_t, τ_t)
4     for i = 1, ..., n do
5         q_i ⟵ ω_i · V_t / S_i^t
6     end
7     return (q_1, ..., q_n)
8 end
```

Question

- V_t is already computed in the main loop
- How are Σ_t, ρ_t and τ_t computed?

Focus on the computation of Σ_t

How is Σ_t provided?

- From a black box
  - The user uses additional information to provide Σ_t
  - Example: the data was simulated and the user knows the simulation parameters
- It is estimated from **past** market data
  - The user provides an estimation window W
  - **Fixed estimation window**: Σ_t is constant, and estimated once and for all on market data between $t_0 - W$ and t_0
  - **Sliding estimation window**: Σ_t is estimated on market data between $t - W$ and t

Nb

- An estimation window has a size at least 1 (the current data)
- There is no relationship between rebalancing times and the size of an estimation window!

General organization

- Starts on Sept. 2nd, ends on Sept. 8th (computer rooms E200 & E201)
  - Oral defense on Sept. 9th
  - **Warning:** file sharing does not work with Windows
  - Git repo: `gitlab.ensimag.fr` (if necessary, install and use the github client to access files)
- 4 students per group
  - The groups should be created on `teide`
  - Grades may be different for the members of a same group
- Source code and final report should be uploaded to `teide` **the previous evening** and the report should contain:
  - The software architecture of the tool
  - A list of the functionalities that were implemented
  - A presentation of the tests that were carried out for the validation phase

Goal of the project

Tool to develop

A Windows WPF application that permits to perform forward and backtests on a hedging portfolio

- Quick recap: given an option O with maturity T and initial price p, the hedging portfolio for O
  - Invests in the underlying assets of the option and in the risk-free rate
  - Has an initial value $V_0 = p$
  - At time t_i contains a quantity $\delta^j_{t_i}$ of the jth risky asset
  - Has a final value $V_T = \mathrm{payoff}(O, \texttt{mkt})$
- Options to hedge:
  - Vanilla call: $(S_T - K)_+$
  - Average basket: $\left(\sum \omega_i S^i_T - K\right)_+$

Forwardtest or backtest?

In general:

- A forwardtest consists in running the algorithm on simulated data
- A backtest consists in running the algorithm on historical data

Code organization

The code should not depend on the data source that is used

What you will learn to use from the .NET framework

- Developing user interfaces with WPF
  Cons:
  - A bit outdated
  - Only runs on Windows operating systems

  Pros:
  - Simple technology to learn the MVVM design pattern
  - Should be close to the next generation of .NET user interfaces (.NET MAUI)
- Handling dependencies (*Nuget*)
- Interfacing native and managed code (*P/Invoke*)
- Working with persistent data (*LINQ*)
- Creating and running unit tests (*NUnit*, . . .)

What is provided

Available market data

- A dozen shares from the CAC40 index
- 5 years of clean market data
- All data is available from Ensimag, or through a VPN
  - Server: `ingefin.ensimag.fr`
  - Database (readonly): `DotNetDB`
  - Identification: *etudiant/edn!2015*

Pricing library

- FinancialProducts
  - Share, VanillaCall, BasketOption
- Utilities.MarketDataFeed
  - ShareValue, DataFeed, SimulatedDataFeedProvider, SemiHistoricDataFeedProvider
  - RiskFreeRateProvider
- Computations
  - Pricer, PricingResults

Wall Risk Engine

- Library for risk analysis and risk management
- Developed in `C` (unmanaged code)
- Main function to use
  - WREmodelingCovariance
  - To be invoked on the log-returns of the considered assets

Final recommendations

- Always have a clear definitions of the functionalities you are about to code
- Use an incremental approach to your development
  - Each increment must have a clear outcome
- Make sure you spend enough time and effort on code refactoring
  - Your code must be readable and maintainable at all times