HW-4 Tutorial: Backtesting Momentum Strategy using S&P 500 Energy Stocks

Team 3: Mukil Senthilkumar, Yifei Wang, Shubham Chorage

April 1, 2025

1 Learning Outcomes

After completing this tutorial, you will be able to:

- Implement a momentum trading strategy.
- Load and filter financial data for stocks in the Energy sector.
- Generate momentum-based trading signals using a rolling-window approach.
- Execute daily trades with both long and short positions based on signal strength.
- Apply position sizing techniques to allocate capital effectively across trades.
- Incorporate transaction costs into the backtesting framework.
- Backtest and analyze the performance of a trading strategy over different time periods and market conditions.
- Evaluate strategy effectiveness using financial metrics, including cumulative returns, Sharpe ratio, and maximum drawdown.

2 Homework Requirements

Our backtesting strategy must meet the following criteria:

a) Implement a trend-following, momentum, or arbitrage strategy. We use a momentum-based strategy.

b) Trade on daily basis, executing orders at the open and allowing positions to carry over.

c) Allow both long and short positions based on momentum signals.

d) Incorporate estimated transaction costs, including a fixed cost of $1 per trade and a variable cost of $0.005 per share.

e) Determine position sizing using signal strength, ensuring capital is allocated effectively across stocks.

f) Use a starting capital of $100,000 for investments.

g) Backtest using data from January 2023 to December 2024.

h) Evaluate performance using key financial metrics from Homework 3.

i) Conduct additional backtests on another industrial sector and on the energy sector for the period January 2020 to December 2021 to analyze strategy robustness.

3 Trading Strategy

- Load and filter financial data for stocks in the Energy sector.
- Generate momentum-based trade signals using historical price data.
- Simulate trades based on the generated signals, considering long and short positions.
- Position Sizing: Allocate capital proportionally based on the squared momentum signal strength, with a maximum trade limit of $5,000 per stock.
- Risk Management: Limit exposure per trade using a fixed capital allocation strategy, ensuring no single trade exceeds a predefined portion of available equity.
- Account for transaction costs, including a fixed $1 fee per trade and a variable cost of $0.005 per share.
- Track and update portfolio equity and daily returns throughout the backtest period.
- Evaluate strategy performance using key financial metrics.

4 Data Preparation and Stock Filtering

In this section, we set up our R environment, load the necessary libraries, and prepare our stock data. We ensure that our data is clean and contains sufficient historical observations.

Listing 1: Data Preparation and Stock Filtering

```r
# Load required libraries
library(tidyverse)
library(lubridate)
library(furrr)
library(scales)
library(ggplot2)
library(rstudioapi)

# Use 8 workers for multiprocessing
plan(multisession, workers = 8)

# Set working directory based on the current file path
current_path <- rstudioapi::getActiveDocumentContext()$path
setwd(dirname(current_path))
rm(list = ls())
options(scipen = 999)

# Set backtest period
backtest_start <- as_date("2023-01-01")
backtest_end   <- as_date("2024-12-31")

energy_stocks <- read_csv("stocks.csv") %>%
  filter(sector == "Energy" & date >= backtest_start - 40 & date <=
     backtest_end) %>%
  arrange(ticker, date) %>%
  select(ticker, date, open, close, sector)

# Exclude problematic tickers.
excluded_tickers <- c("IR", "AXON", "BLDR")
energy_stocks <- energy_stocks %>% filter(!ticker %in% excluded_tickers)

# Industrials sector
industrials_stocks <- read_csv("stocks.csv") %>%
  filter(sector == "Industrials" & date >= backtest_start - 40 & date <=
     backtest_end) %>%
  arrange(ticker, date) %>%
  select(ticker, date, open, close, sector) %>%
  filter(!ticker %in% excluded_tickers)

# Information Technology sector
tech_stocks <- read_csv("stocks.csv") %>%
  filter(sector == "Information␣Technology" & date >= backtest_start - 40
     & date <= backtest_end) %>%
41   arrange(ticker, date) %>%
42   select(ticker, date, open, close, sector) %>%
43   filter(!ticker %in% excluded_tickers)
```

Explanation:

- **Lines 1–7:** We load essential libraries. `tidyverse` provides powerful data manipulation tools, `forecast` lets us perform time-series forecasting with ETS models, `rstudioapi` is used for dynamically setting our working directory, `furrr` allows for parallel processing, and `ggplot2` is used to plot graphs.
- **Line 10:** We set up parallel processing with 8 workers, which helps speed up repetitive tasks.
- **Lines 13–16:** The working directory is set based on the active document's location; we clear the workspace to avoid variable conflicts and adjust number formatting for readability.
- **Lines 19–20:** Here we define our backtesting period from January 1, 2023, to December 31, 2024.
- **Lines 22–43:** We load the stock data from a CSV file, filter to include only stocks in the Energy, Industrials and Information Technology sector, arrange the data by ticker and date, and select only the necessary columns. We include 40 days before the start date to cover the 20-day rolling window.

5 Part 1: Generating Momentum Signals

This function generates momentum signals for each stock in the dataset using a rolling window approach. The momentum is calculated as the percentage change in the stock's closing price over a specified window.

If the momentum exceeds a given threshold (default of 15%), a long position signal is generated, while if the momentum is less than the negative threshold, a short position signal is generated.

The function also calculates the buy and sell prices based on these signals, with buying occurring at the open price for long trades and selling at the close price, and selling at the open price for short trades with covering at the close price.

Listing 2: Generate Trade Signals Based on Momentum

```
gen_momentum_signals <- function(stocks, window = 20, threshold = 0.15) {
  stocks <- stocks %>%
```

```r
3      group_by(ticker) %>%
4      arrange(date) %>%
5      mutate(momentum = (close / lag(close, window)) - 1,
6             long = if_else(momentum > threshold, TRUE, FALSE),
7             short = if_else(momentum < -threshold, TRUE, FALSE),
8             trade = long | short,
9             # For long trades: buy at open, sell at close;
10            # for short trades: sell at open, cover at close.
11            buy_price = if_else(long, open, if_else(short, close, NA_real_)
                 ),
12            sell_price = if_else(long, close, if_else(short, open, NA_real_
                 ))) %>%
13     ungroup()
14   return(stocks)
15 }
```

Explanation:

- **Lines 1–2:** The function `gen_momentum_signals` is defined to take stock data and a window size (defaulted to 20) and a threshold (defaulted to 0.15) as input. It processes the data by grouping it by ticker symbol.
- **Lines 3–5:** The data is sorted by date for each ticker, and a new `momentum` column is calculated as the percentage change in closing price over the defined rolling window (20 days by default).
- **Lines 6–7:** The function generates trade signals: a `long` signal is set if the momentum exceeds the threshold, and a `short` signal is set if the momentum is less than the negative threshold.
- **Line 8:** The `trade` column is defined to indicate whether a stock is in a trade (either long or short).
- **Lines 11–12:** The `buy_price` is set for long trades as the open price and for short trades as the close price. Similarly, the `sell_price` is set for long trades as the close price and for short trades as the open price.
- **Line 13:** The data is ungrouped after performing the necessary calculations.

6 Part 2: Simulate Trades & Record Trade Log and Daily Returns

This function simulates trades based on momentum signals for each stock within the specified backtest period. It processes trades by calculating the trade amount, handling long and short positions, and accounting for transaction costs. The function updates equity, logs trade details, and calculates daily returns. Due to the function's size, it is divided into sub-sections for better clarity. All sub-sections are part of the same function.

6.1 Part 2A: Simulating Trades and Recording the Trade Log

The simulation involves filtering the stocks for each date that have a trade signal (either long or short), weighting the trade signals based on the absolute value of momentum, and calculating the trade amounts accordingly. The trade log is updated with each trade's details, including the type of trade, the buy and sell prices, the number of shares traded, and the transaction costs.

Listing 3: Simulating Trades and Recording Trade Log

```
1  simulate_trades <- function(stocks, backtest_start, backtest_end,
2                               initial_equity = 100000, max_trade = 5000) {
3    # Select only dates within the backtest period.
4    trade_dates <- sort(unique(stocks$date[stocks$date >= backtest_start &
        stocks$date <= backtest_end]))
5
6    # Set initial equity
7    equity <- initial_equity
8
9    daily_returns <- tibble(date = trade_dates, equity = NA_real_, daily_
        return = NA_real_)
10
11   trade_log <- tibble(date = as.Date(character()), ticker = character(),
12                       trade_type = character(), trade_amount = double(),
13                       buy_price = double(), sell_price = double(),
14                       shares = double(), tx_cost = double(),
15                       gross_revenue = double(), net_revenue = double(),
16                       trade_return = double())
```

Explanation:

- **Line 1–2:** The function simulate_trades takes in stock data, the backtest period start and end dates, the initial equity, and a maximum trade limit.

- **Lines 3-5:** The trade dates are filtered from the stock data based on the backtest period.
- **Lines 6-16:** The equity is initialized, and placeholders for daily returns and the trade log are created as empty tibbles.

6.2 Part 2B: Calculating Signal Strength and Trade Weights

Within each trade date, the function calculates the strength of the trading signals based on the momentum of each stock. Instead of using absolute momentum directly, the signal strength is weighted by the square of the absolute momentum to emphasize stronger signals. The total signal strength for the day is calculated, and the weights for each stock are normalized to sum to one.

The function proceeds to simulate trades for each stock with a trade signal, calculating the trade amount based on the weight of the signal. The trade amount is capped by the max_trade limit. If the stock is designated for a long trade, the stock is bought at the open price and sold at the close price, while for short trades, the stock is sold at the open price and covered at the close price.

Transaction costs are applied to each trade, which includes a fixed \$1 per leg cost and a variable \$0.005 per share cost. The net revenue from each trade is calculated by deducting the transaction costs from the gross revenue.

Listing 4: Simulating Trades and Recording Trade Log

```r
for(i in seq_along(trade_dates)) { # For each trade date
  current_date <- trade_dates[i]
  # Filter stocks with a trade signal for the current day.
  day_data <- stocks %>% filter(date == current_date, trade == TRUE)

  # Define Weight
  day_data <- day_data %>% mutate(signal_strength = if_else(trade, abs(momentum)^2, 0))
  total_signal <- sum(day_data$signal_strength, na.rm = TRUE)

  if(total_signal == 0) {
    daily_returns$equity[i] <- equity
    daily_returns$daily_return[i] <- 0
    next
  }

  day_data <- day_data %>% mutate(weight = signal_strength / total_signal)

  # Simulate trades for each stock with a signal.
  for(j in 1:nrow(day_data)) {
    stock_row <- day_data[j, ]
    trade_amount <- min(stock_row$weight * equity, max_trade)
    if(trade_amount <= 0) next

    if(stock_row$long) {
      trade_type <- "long"
      buy_price <- stock_row$open
      sell_price <- stock_row$close
      trade_return <- (sell_price - buy_price) / buy_price
    } else if(stock_row$short) {
      trade_type <- "short"
      # For short: sell at open and cover at close.
      buy_price <- stock_row$close   # cover price
      sell_price <- stock_row$open     # sale price
      trade_return <- (buy_price - sell_price) / buy_price
    } else {
      next
    }

    shares <- trade_amount / buy_price

    # Transaction cost: fixed $1 per leg plus $0.005 per share each time
    tx_cost <- 2 * (1 + (shares * 0.005))
    gross_revenue <- trade_amount * (1 + trade_return)
    net_revenue <- gross_revenue - tx_cost

    equity <- equity - trade_amount + net_revenue

    trade_log <- trade_log %>% add_row(date = current_date,
                                       ticker = stock_row$ticker,
                                       trade_type = trade_type,
                                       trade_amount = trade_amount,
                                       buy_price = buy_price,
                                       sell_price = sell_price,
                                       shares = shares,
                                       tx_cost = tx_cost,
                                       gross_revenue = gross_revenue,
                                       net_revenue = net_revenue,
                                       trade_return = trade_return)
  }
```

Explanation:

- **Lines 1–7:** For each trade date, the function filters the data to include only the stocks with trade signals (long or short). The `signal_strength` is calculated as the square of the absolute value of momentum to prioritize stronger signals.
- **Line 8:** The total signal strength for the day is summed across all stocks.
- **Line 10-14:** If there is no signal for the day, no trades are made, and the daily return is recorded as 0.
- **Line 16:** If signals exist, the weights for each stock are calculated by normalizing the signal strength.
- **Line 18–23:** For each stock with a trade signal, the trade amount is calculated as a proportion of equity, with a cap defined by `max_trade`.
- **Lines 24-37:** Depending on whether the trade is long or short, the appropriate buy and sell prices are used, and the trade return is calculated.
- **Lines 39:** No of shares are computed by dividing `trade_amount` by `buy_price` of each unit.
- **Lines 41–43:** Transaction costs are calculated based on the number of shares traded, and the net revenue from the trade is computed.
- **Line 45:** The equity is updated by subtracting the trade amount and adding the net revenue from the trade.
- **Line 47:** The trade log is updated with the details of the trade, including the trade type, buy/sell prices, shares, transaction costs, and revenues

6.3 Part 2C: Recording Daily Returns and Final Results

Finally, after simulating all trades for the day, the function records the equity value at the end of the day and calculates the daily return as the percentage change in equity from the previous day. This is repeated for each trade date in the backtest period.

At the end of the simulation, the function returns the final equity value, the percentage return relative to the initial equity, the daily returns, and the trade log.

Listing 5: Recording Daily Returns and Final Results

```
  daily_returns$equity[i] <- equity # continuation of for loop
  if(i == 1) {
    daily_returns$daily_return[i] <- 0
```

```r
4      } else {
5        daily_returns$daily_return[i] <- (equity - daily_returns$equity[i-1]) / daily_returns$equity[i-1]
6      }
7    } # end of for loop (i)
8
9    list(final_equity = equity,
10        ret_pct = (equity - initial_equity) / initial_equity * 100,
11        daily_returns = daily_returns,
12        trade_log = trade_log)
13
14 } # end of function
```

Explanation:

- **Line 1–7:** After processing all trades for a given day, the equity is recorded, and the daily return is calculated as the percentage change from the previous day's equity.
- **Line 9-14:** After all dates are processed, the function returns a list containing the final equity, the total return percentage, the daily returns, and the trade log.

7 Part 3: Evaluate Performance Metrics & Plot Graphs

In this section, we use the `eval_performance` function to assess how well our trading strategy performed. This function takes in our trade logs and daily equity values, then calculates key performance metrics such as cumulative return, Sharpe ratio, and maximum drawdown. It also separates long and short trades to evaluate win rates and average returns for each. To help us visualize our results, it generates two important plots: an equity curve showing portfolio growth over time, and a drawdown chart that highlights periods of decline. We then apply this evaluation across three different sectors—Energy, Information Technology, and Industrials—for the 2023–2024 period and finally compare that with Energy sector performance from 2020–2021. This allows us to understand not just how our strategy performs in different sectors, but also how it holds up across different market conditions.

7.1 Part 3A: Compute metrics and summarize trade performance

Listing 6: Evaluating Performance Metrics

```r
eval_performance <- function(trade_log, daily_returns, title_suffix = "") {
  # Ensure there are no missing values in daily_returns
```

```
3   daily_returns <- daily_returns %>% mutate(daily_return = replace_na(
       daily_return, 0))
4
5   # Compute cumulative return
6   cum_return <- prod(1 + daily_returns$daily_return, na.rm = TRUE) - 1
7
8   # Compute cumulative equity and drawdowns
9   cum_eq <- cumprod(1 + daily_returns$daily_return)
10  max_cum_eq <- cummax(cum_eq)
11  drawdowns <- cum_eq / max_cum_eq - 1
12
13  # Handle cases where drawdowns are NA
14  if (all(is.na(drawdowns))) {
15    max_drawdown <- NA
16  } else {
17    max_drawdown <- min(drawdowns, na.rm = TRUE) * 100
18  }
19
20  # Handle cases where standard deviation is NA or zero
21  daily_return_sd <- sd(daily_returns$daily_return, na.rm = TRUE)
22  if (is.na(daily_return_sd) || daily_return_sd == 0) {
23    sharpe_ratio <- NA
24  } else {
25    sharpe_ratio <- round((mean(daily_returns$daily_return, na.rm = TRUE)
         - (0.04/365)) / daily_return_sd, 4)
26  }
27
28  # Compute trade performance summary
29  long_trades <- trade_log %>% filter(trade_type == "long")
30  short_trades <- trade_log %>% filter(trade_type == "short")
```

Explanation:

- **Lines 3:** Replaces any missing daily returns with 0 to avoid errors in downstream calculations.
- **Line 6:** Computes the cumulative portfolio return across the backtest period.
- **Line 9-11:** Builds the cumulative equity curve by compounding daily returns, then calculates drawdowns as the relative decline from the historical equity peak.
- **Lines 14-18:** Computes the maximum drawdown in percentage terms. If all drawdowns are NA, it returns NA, otherwise it finds the largest negative value.

- **Lines 21-26:** Calculates the daily Sharpe Ratio, adjusting for a 4 percents annual risk-free rate. It safeguards against division by zero or missing standard deviation.
- **Lines 29-30:** Separates the trade log into long and short trades.

7.2 Part 3B: Display summary, plot results, return output.

Listing 7: Display summary, plot results, return output

```
performance_summary <- tibble(
    "Long␣Trades" = nrow(long_trades),
    "%␣Winning␣Long␣Trades" = if(nrow(long_trades) > 0) round(mean(long_trades$trade_return > 0, na.rm = TRUE) * 100, 2) else NA,
    "Avg␣Long␣Trade␣Return" = if(nrow(long_trades) > 0) round(mean(long_trades$trade_return, na.rm = TRUE), 4) else NA,
    "Short␣Trades" = nrow(short_trades),
    "%␣Winning␣Short␣Trades" = if(nrow(short_trades) > 0) round(mean(short_trades$trade_return > 0, na.rm = TRUE) * 100, 2) else NA,
    "Avg␣Short␣Trade␣Return" = if(nrow(short_trades) > 0) round(mean(short_trades$trade_return, na.rm = TRUE), 4) else NA,
    "Daily␣Sharpe␣Ratio" = sharpe_ratio,
    "Cumulative␣Portfolio␣Return" = round(cum_return, 4),
    "Max␣Drawdown␣(%)" = round(max_drawdown, 2),
    "Overall␣%␣Winning␣Trades" = if(nrow(trade_log) > 0) round(mean(trade_log$trade_return > 0, na.rm = TRUE) * 100, 2) else NA
  )

  cat("Performance␣Summary", title_suffix, ":\n")
  glimpse(performance_summary)
  cat("\nFinal␣Equity:", daily_returns$equity[nrow(daily_returns)], "\n")

  # Equity Curve Plot
  p1 <- ggplot(daily_returns, aes(x = date, y = equity)) +
    geom_line(color = "blue") +
    labs(title = paste("Equity␣Curve", title_suffix),
         x = "Date", y = "Equity␣($)") +
    theme_minimal()

  # Drawdown Plot
  dd_df <- tibble(date = daily_returns$date, drawdown = drawdowns)
  p2 <- ggplot(dd_df, aes(x = date, y = drawdown)) +
    geom_line(color = "red") +
    scale_y_continuous(labels = percent) +
    labs(title = paste("Drawdowns", title_suffix),
```

```r
31          x = "Date", y = "Drawdown␣(%)") +
32      theme_minimal()
33
34    print(p1)
35    print(p2)
36
37    return(performance_summary)
38  }
```

- **Lines 1–12:** Builds a performance summary table including
  - Number of trades by type,
  - Win rate and average return,
  - Overall return and risk-adjusted performance
- **Lines 14–16:** Displays the performance table and final portfolio value
- **Lines 19–23:** Plots the equity curve, showing how the portfolio grows over time.
- **Lines 26–37:** Plots the drawdown curve, highlighting periods of loss relative to the peak and then returns the `performance_summary`.

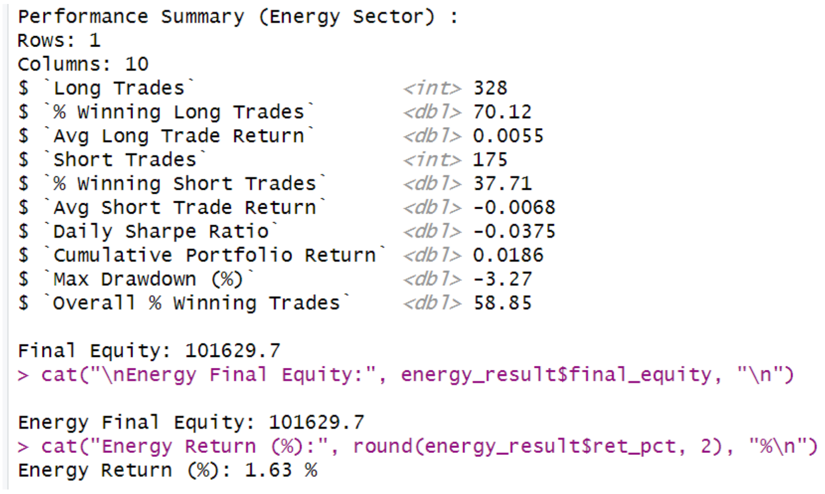
7.3 Part 3C: Running Backtests & Generating Plots

Listing 8: Running Backtests & Generating Plots

```r
# Energy sector
energy_signals <- gen_momentum_signals(energy_stocks, window = 20, threshold = 0.15)
energy_result <- simulate_trades(energy_signals, backtest_start, backtest_end,
                                 initial_equity = 100000, max_trade = 5000)
energy_performance <- eval_performance(energy_result$trade_log, energy_result$daily_returns,
                                       title_suffix = "(Energy␣Sector)")
cat("\nEnergy␣Final␣Equity:", energy_result$final_equity, "\n")
cat("Energy␣Return␣(%):", round(energy_result$ret_pct, 2), "%\n")

# Information Technology sector
tech_signals <- gen_momentum_signals(tech_stocks, window = 20, threshold = 0.15)
tech_result <- simulate_trades(tech_signals, backtest_start, backtest_end,
13                                  initial_equity = 100000, max_trade = 5000)
14 tech_performance <- eval_performance(tech_result$trade_log, tech_result$
       daily_returns,
15                                          title_suffix = "(Information␣
                                             Technology␣Sector)")
16 cat("\nInformation␣Technology␣Final␣Equity:", tech_result$final_equity, "\
       n")
17 cat("Information␣Technology␣Return␣(%):", round(tech_result$ret_pct, 2), "
       %\n")
18 
19 # Industrials sector
20 industrials_signals <- gen_momentum_signals(industrials_stocks, window =
       20, threshold = 0.15)
21 industrials_result <- simulate_trades(industrials_signals, backtest_start,
        backtest_end,
22                                          initial_equity = 100000, max_trade =
                                              5000)
23 industrials_performance <- eval_performance(industrials_result$trade_log,
       industrials_result$daily_returns,
24                                                 title_suffix = "(Industrials␣
                                                    Sector)")
25 
26 cat("\nIndustrials␣Final␣Equity:", industrials_result$final_equity, "\n")
27 cat("Industrials␣Return␣(%):", round(industrials_result$ret_pct, 2), "%\n"
       )
```

- **Lines 1–2:** Generate momentum signals for Energy sector stocks using a 20-day window and a threshold of 15
- **Lines 3–4:** Simulate trades for the Energy sector from the specified start to end dates, using $100,000 starting equity and limiting each trade to a maximum of $5,000.
- **Lines 5–6:** Evaluate Energy sector performance, including cumulative returns, Sharpe ratio, drawdowns, and generate related plots.



```
Performance Summary (Energy Sector) :
Rows: 1
Columns: 10
$ `Long Trades`                  <int> 328
$ `% Winning Long Trades`        <dbl> 70.12
$ `Avg Long Trade Return`        <dbl> 0.0055
$ `Short Trades`                 <int> 175
$ `% Winning Short Trades`       <dbl> 37.71
$ `Avg Short Trade Return`       <dbl> -0.0068
$ `Daily Sharpe Ratio`           <dbl> -0.0375
$ `Cumulative Portfolio Return`  <dbl> 0.0186
$ `Max Drawdown (%)`             <dbl> -3.27
$ `Overall % Winning Trades`     <dbl> 58.85

Final Equity: 101629.7
> cat("\nEnergy Final Equity:", energy_result$final_equity, "\n")

Energy Final Equity: 101629.7
> cat("Energy Return (%):", round(energy_result$ret_pct, 2), "%\n")
Energy Return (%): 1.63 %
```

Figure 1: performance table Energy Sector

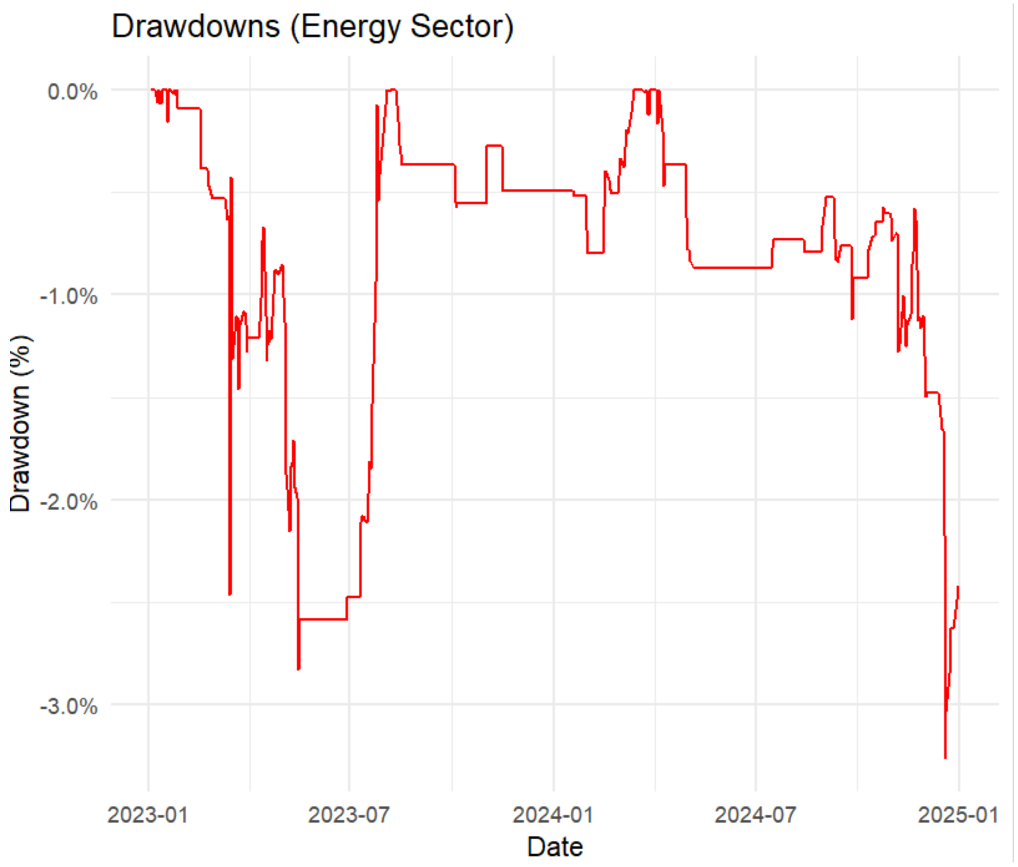


Figure 2: Drawdowns Energy Sector

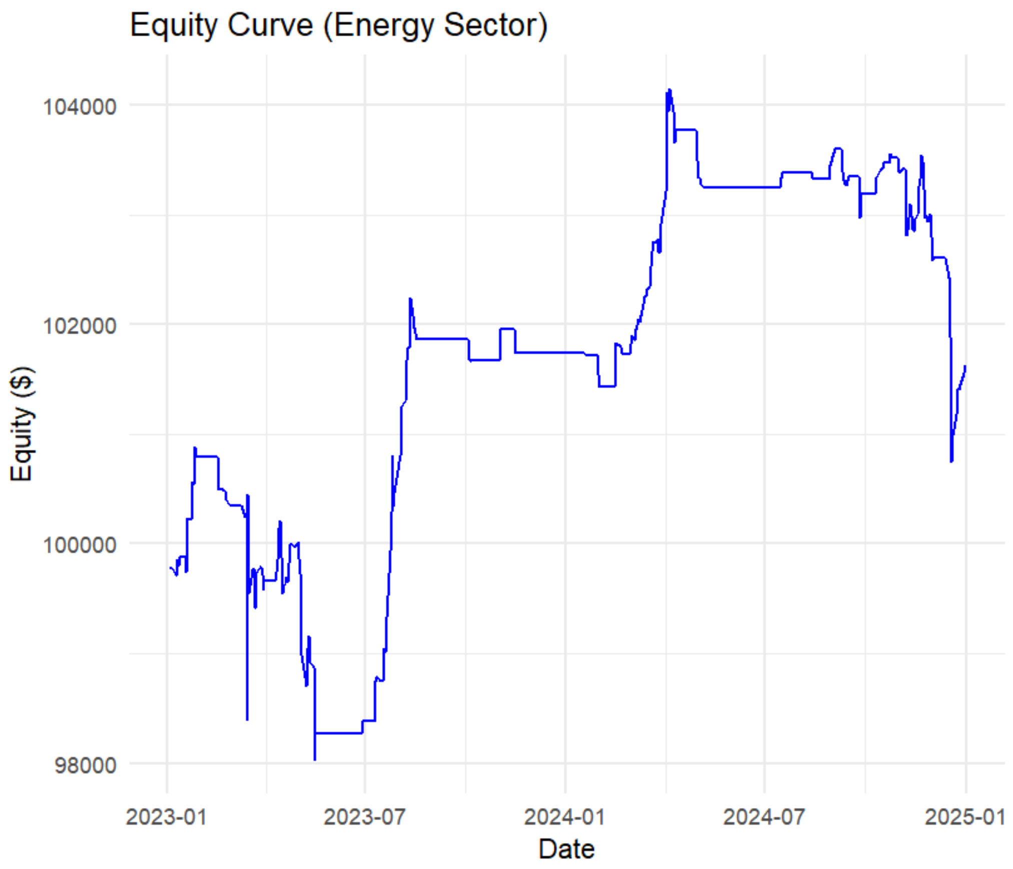
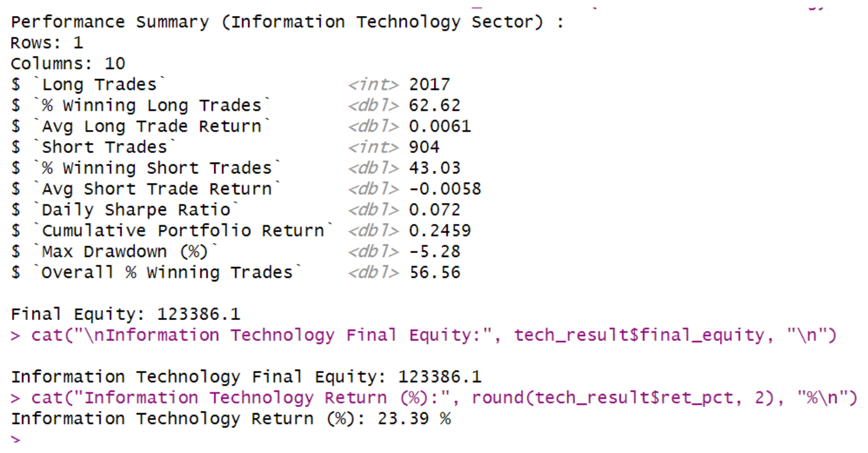


Figure 3: Equity Curve Energy Sector

- **Lines 7–8:** Print the final equity and total return percentage for the Energy sector.
- **Lines 11:** Generate momentum signals for Information Technology sector stocks
- **Lines 12-13:** Simulate trades for the Tech sector with the same portfolio rules.
- **Lines 14-15:** Evaluate performance and generate plots for the Tech sector.



```
Performance Summary (Information Technology Sector) :
Rows: 1
Columns: 10
$ `Long Trades`                <int> 2017
$ `% Winning Long Trades`      <dbl> 62.62
$ `Avg Long Trade Return`      <dbl> 0.0061
$ `Short Trades`               <int> 904
$ `% Winning Short Trades`     <dbl> 43.03
$ `Avg Short Trade Return`     <dbl> -0.0058
$ `Daily Sharpe Ratio`         <dbl> 0.072
$ `Cumulative Portfolio Return` <dbl> 0.2459
$ `Max Drawdown (%)`           <dbl> -5.28
$ `Overall % Winning Trades`   <dbl> 56.56

Final Equity: 123386.1
> cat("\nInformation Technology Final Equity:", tech_result$final_equity, "\n")

Information Technology Final Equity: 123386.1
> cat("Information Technology Return (%):", round(tech_result$ret_pct, 2), "%\n")
Information Technology Return (%): 23.39 %
>
```

Figure 4: performance table Information Tech Sector

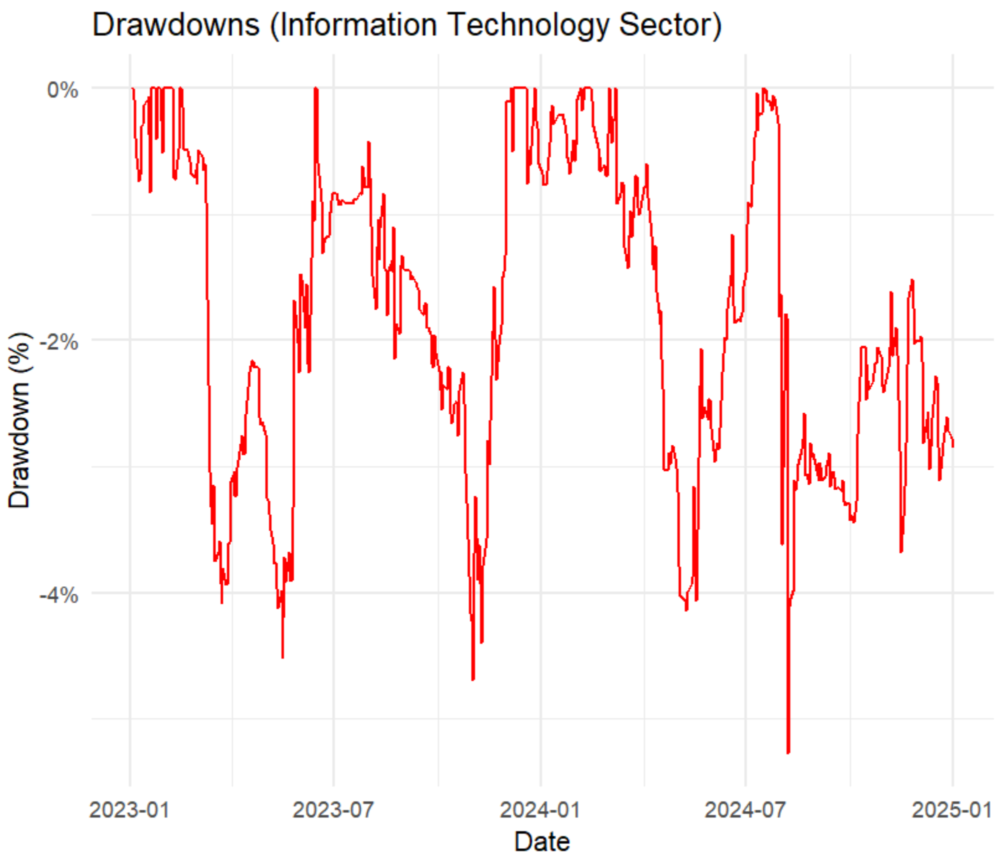


Figure 5: Drawdowns Information Technology Sector

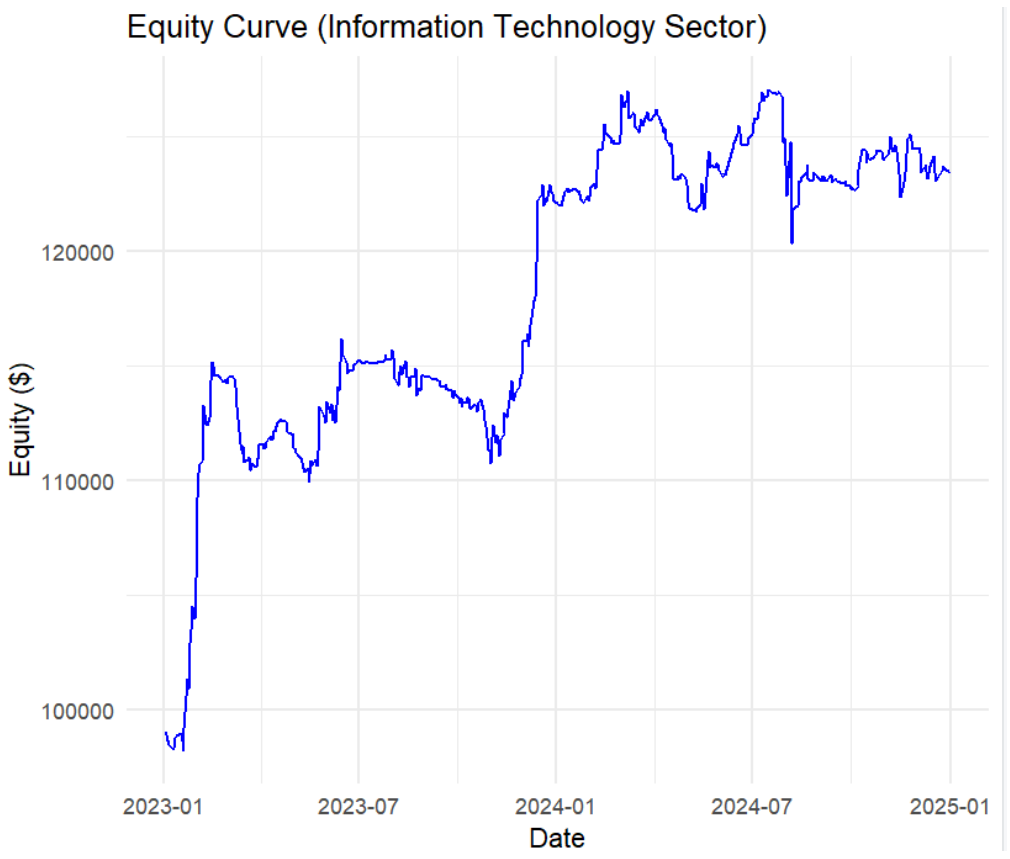
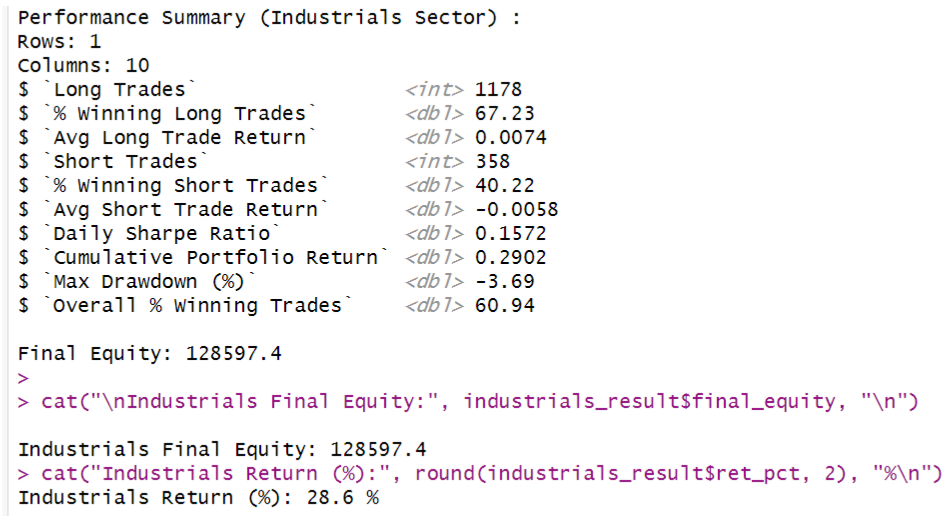


Figure 6: Equity Curve Information Technology Sector

- **Lines 16-17:** Print final equity and return for the Tech sector.
- **Lines 20:** Generate momentum signals for Industrials sector stocks.
- **Lines 21-22:** Run the trade simulation for Industrials
- **Lines 23-24:** Evaluate and visualize performance for Industrials.



```
Performance Summary (Industrials Sector) :
Rows: 1
Columns: 10
$ `Long Trades`                 <int> 1178
$ `% Winning Long Trades`       <dbl> 67.23
$ `Avg Long Trade Return`       <dbl> 0.0074
$ `Short Trades`                <int> 358
$ `% Winning Short Trades`      <dbl> 40.22
$ `Avg Short Trade Return`      <dbl> -0.0058
$ `Daily Sharpe Ratio`          <dbl> 0.1572
$ `Cumulative Portfolio Return` <dbl> 0.2902
$ `Max Drawdown (%)`            <dbl> -3.69
$ `Overall % Winning Trades`    <dbl> 60.94

Final Equity: 128597.4
>
> cat("\nIndustrials Final Equity:", industrials_result$final_equity, "\n")

Industrials Final Equity: 128597.4
> cat("Industrials Return (%):", round(industrials_result$ret_pct, 2), "%\n")
Industrials Return (%): 28.6 %
```

Figure 7: performance table Industrial Sector

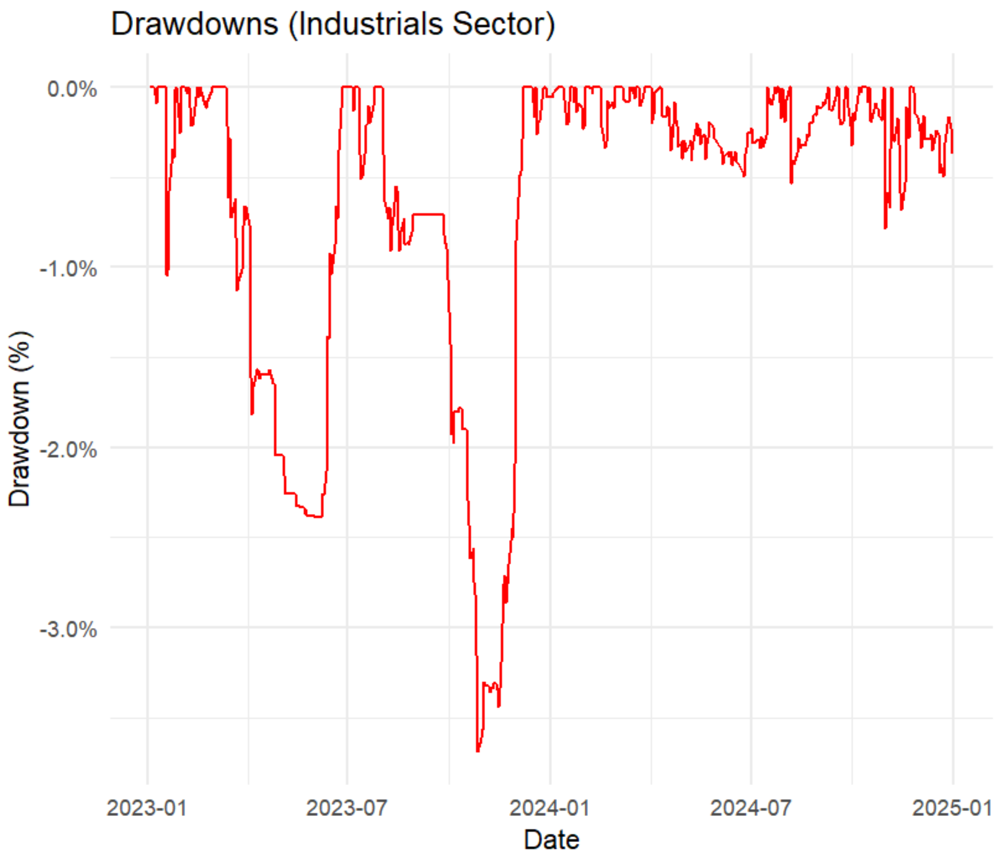


Figure 8: Drawdowns Industrial Sector

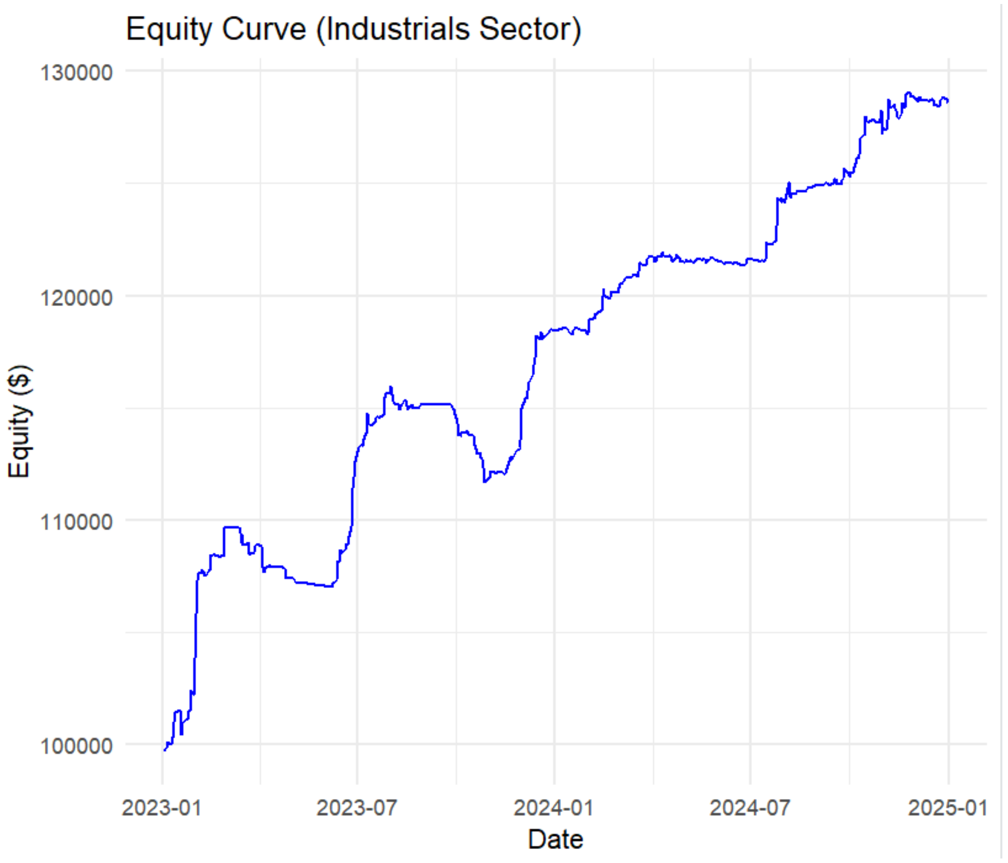


Figure 9: Equity Curve Industrial Sector

- **Lines 26-27:** Display the final equity and return for the Industrials sector.

7.4 Part 3D: compare performance with Energy from year 20-21

Listing 9: compare performance with Energy from year 20-21

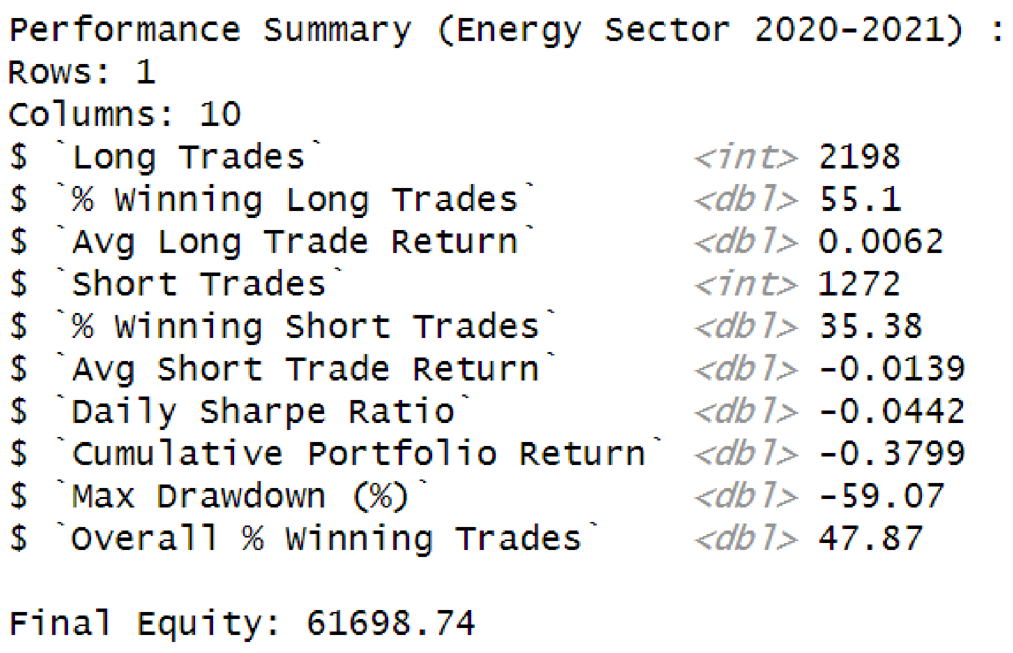
```r
# Energy sector
# Last part requires to compare performance with Energy from year 20-21
# Set new backtest period for Energy sector (2020-2021)

backtest_start_energy <- as_date("2020-01-01")
backtest_end_energy   <- as_date("2021-12-31")

energy_stocks_2020_2021 <- read_csv("stocks.csv") %>%
  filter(sector == "Energy" & date >= backtest_start_energy - 40 & date <=
      backtest_end_energy) %>%
  arrange(ticker, date) %>%
  select(ticker, date, open, close, sector) %>%
  filter(!ticker %in% excluded_tickers)
```

```
13
14 # Generate momentum signals for Energy sector (2020-2021) and simulate trades
15 energy_signals_2020_2021 <- gen_momentum_signals(energy_stocks_2020_2021, window = 20, threshold = 0.15)
16
17 energy_result_2020_2021 <- simulate_trades(energy_signals_2020_2021, backtest_start_energy, backtest_end_energy,
18                                               initial_equity = 100000, max_trade = 5000)
19
20 # Evaluate performance and generate plots for Energy sector (2020-2021)
21 energy_performance_2020_2021 <- eval_performance(energy_result_2020_2021$trade_log, energy_result_2020_2021$daily_returns,
22                                                       title_suffix = "(Energy␣Sector␣2020-2021)")
23
24 # Print final equity and returns for Energy sector (2020-2021)
25 cat("\nEnergy␣(2020-2021)␣Final␣Equity:", energy_result_2020_2021$final_equity, "\n")
26 cat("Energy␣(2020-2021)␣Return␣(%):", round(energy_result_2020_2021$ret_pct, 2), "%\n")
```

- **Lines 5-6:** Define the new backtest window from January 1, 2020 to December 31, 2021, for historical performance comparison.
- **Lines 8–12:** Load Energy sector stock data for the defined period, apply a 40-day buffer for signal calculation, sort by ticker and date, and remove problematic tickers.
- **Lines 15:** Apply the 20-day momentum strategy to the historical Energy sector data.
- **Lines 17-18:** Run trade simulations on the 2020–2021 Energy signals using the same capital and trade constraints.
- **Lines 21-22:** Evaluate historical performance and generate visual plots labeled for the 2020–2021 period energy.



```
Performance Summary (Energy Sector 2020-2021) :
Rows: 1
Columns: 10
$ `Long Trades`                  <int> 2198
$ `% Winning Long Trades`        <dbl> 55.1
$ `Avg Long Trade Return`        <dbl> 0.0062
$ `Short Trades`                 <int> 1272
$ `% Winning Short Trades`       <dbl> 35.38
$ `Avg Short Trade Return`       <dbl> -0.0139
$ `Daily Sharpe Ratio`           <dbl> -0.0442
$ `Cumulative Portfolio Return`  <dbl> -0.3799
$ `Max Drawdown (%)`             <dbl> -59.07
$ `Overall % Winning Trades`     <dbl> 47.87

Final Equity: 61698.74
```

Figure 10: performance table Energy Sector 2020

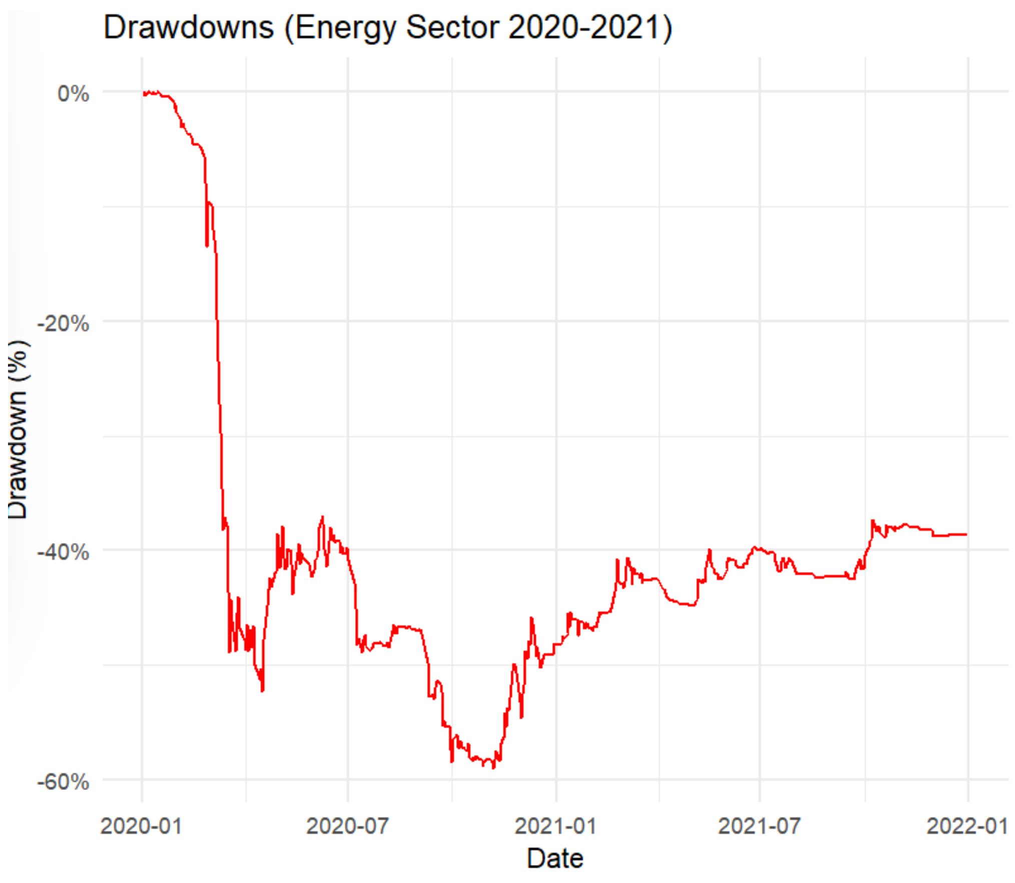


Figure 11: Drawdowns Energy 2020-2021

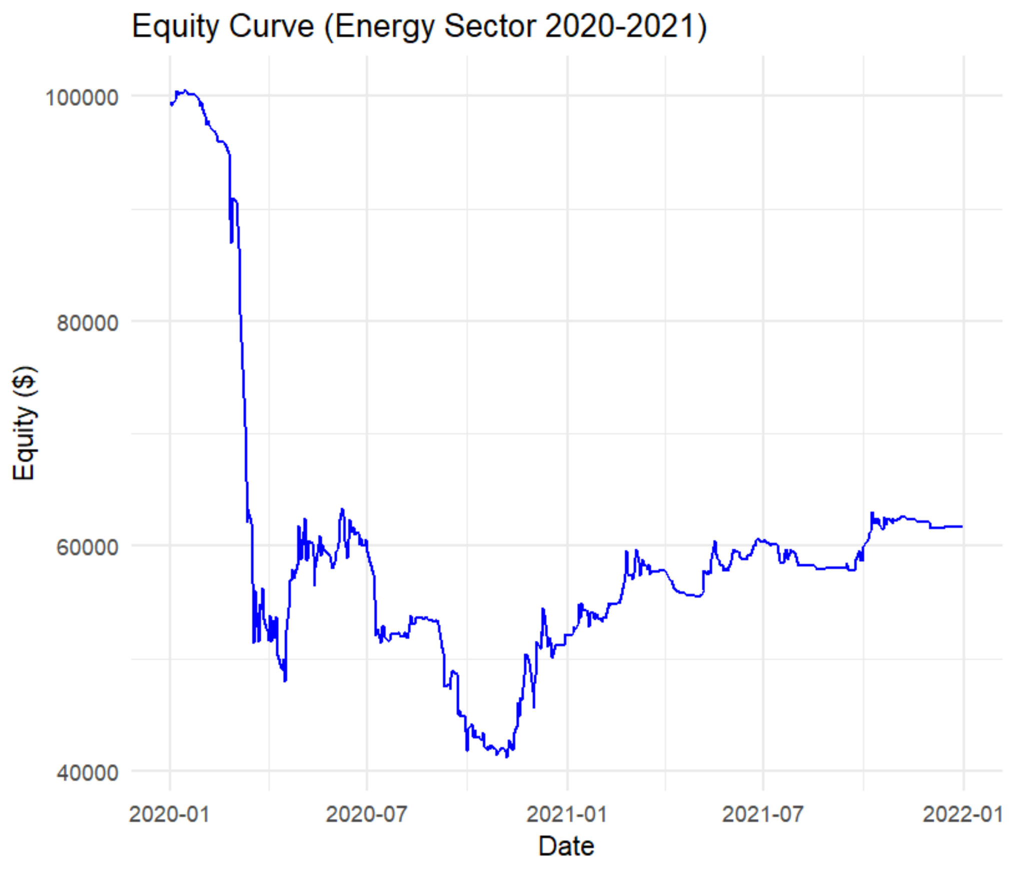


Figure 12: Equity Curve Energy 2020-2021

- **Lines 25-26:** Print final equity and total return percentage to compare with more recent results.

8 Conclusion

This final section summarizes the key steps, findings, and takeaways from our momentum-based trading strategy tutorial using S&P 500 sector stocks.

- **Step 1: Strategy Design** We implemented a momentum-based strategy that generates trading signals based on a 20-day return threshold of $\pm 15\%$. A stock generates a *long* signal if its return exceeds 15%, and a *short* signal if it falls below -15%.

- **Step 2: Trade Simulation** For each trading day from January 1, 2023 to December 31, 2024, we simulated trades based on the signal strength (squared momentum), allocated capital proportionally (with a $5,000 cap per trade), and updated portfolio equity while accounting for realistic transaction costs.

- **Step 3: Sector Backtesting** We applied this strategy to three sectors—**Energy**, **Information Technology**, and **Industrials**—using the same parameters and backtest window. For each sector, we generated trades, tracked equity, and evaluated performance metrics such as Sharpe ratio, max drawdown, and win rates.
- **Step 4: Historical Comparison (2020–2021)** To test robustness, we re-ran the strategy on Energy stocks during the earlier period of 2020–2021. This allowed us to assess how market conditions influence performance and to compare it with the 2023–2024 results.
- **Step 5: Performance Evaluation** We used a custom evaluation function that calculates cumulative return, daily Sharpe ratio, and trade-level win statistics. It also provides visual output including the equity curve and drawdown graph for each sector tested.
- **Step 6: Key Takeaways** The Energy sector showed strong performance during 2023–2024 but also revealed drawdowns under certain conditions. The 2020–2021 test displayed different characteristics, suggesting market regime plays a significant role in outcome. By comparing across sectors and periods, we confirmed that our strategy is flexible but sensitive to volatility and trend consistency.