Tutorial Exercise 3

Estimating False Discovery Rates in Strategy Development

Barry Quinn

2025-03-04

Table of contents


```
# Load required libraries
library(tidyverse)
library(ggplot2)
library(knitr)
library(kableExtra)
library(data.table)
```

Introduction

In this tutorial, we will explore the problem of false discoveries in quantitative finance, specifically focusing on how multiple testing leads to selection bias in strategy development. We will:

1. Simulate the process of strategy development with random strategies
2. Calculate the impact of selection bias on Sharpe ratios
3. Implement the Deflated Sharpe Ratio (DSR) to correct for selection bias

4. Analyze how false discovery rates change with different parameters

This exercise will provide hands-on experience with the concepts discussed in the lecture on backtest overfitting.

Part 1: Generating Random Strategies

Let's first create a function to generate random strategy returns. These will simulate strategies with no genuine edge (i.e., the true Sharpe ratio is zero).

```
# Function to generate random strategy returns
# This simulates the returns of investment strategies that have no real edge
generate_random_strategies <- function(n_strategies = 1000,
                                       n_returns = 252, # 252 trading days in a year
                                       mean_return = 0, # Zero expected return (no edge)
                                       sd_return = 0.01) { # 1% daily volatility is typical
  # Create a matrix of random returns drawn from a normal distribution
  # Each column represents a strategy, each row is a daily return
  returns_matrix <- matrix(
    rnorm(n_strategies * n_returns, mean = mean_return, sd = sd_return),
    nrow = n_returns,
    ncol = n_strategies
  )

  # Name each strategy for easier reference
  colnames(returns_matrix) <- paste0("Strategy_", 1:n_strategies)

  return(returns_matrix)
}

# Generate 1000 random strategies with 1 year of daily returns
# Using seed 42 ensures reproducibility - everyone gets the same random numbers
set.seed(42)
random_strategies <- generate_random_strategies(n_strategies = 1000,
                                                n_returns = 252)

# Check the dimensions to confirm we have 252 days × 1000 strategies
dim(random_strategies)
```

```
[1]  252 1000
```

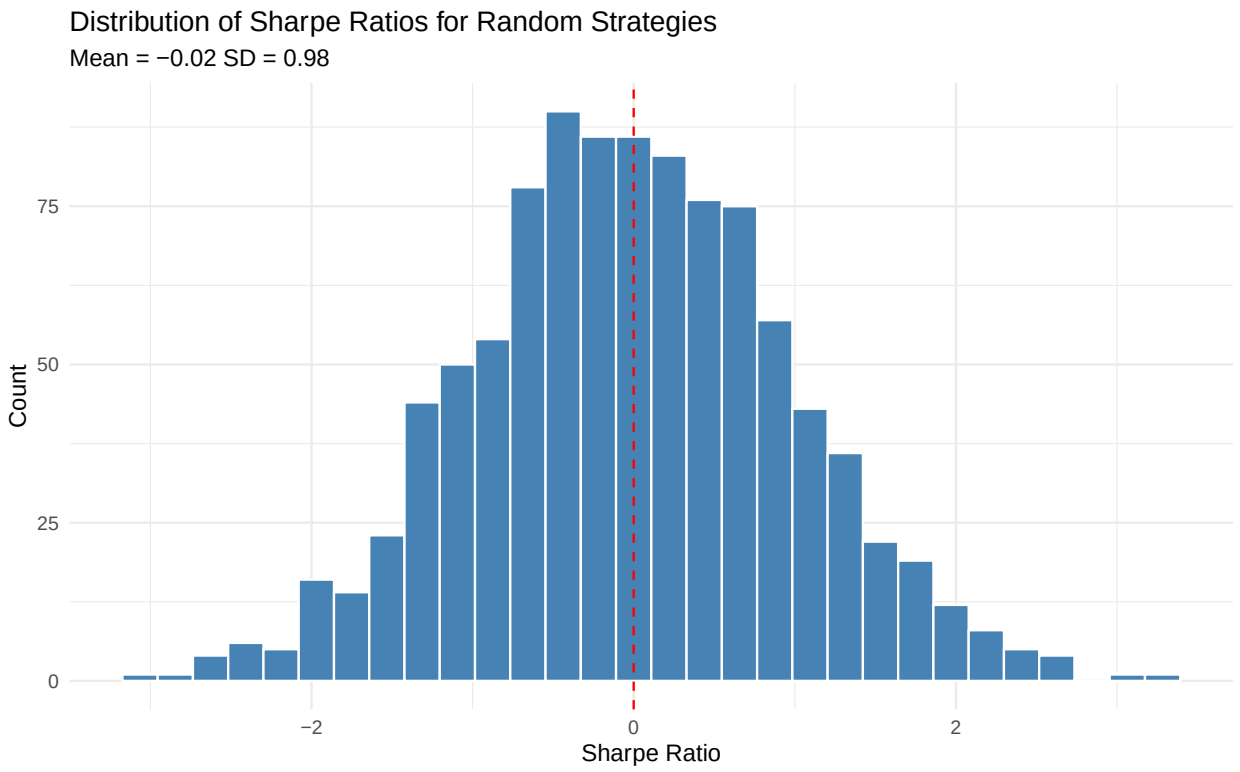
Now, let's calculate the Sharpe ratio for each of these strategies.

```r
# Function to calculate annualized Sharpe ratio
# The Sharpe ratio is the most common performance metric in finance
# It measures excess return per unit of risk (volatility)
calculate_sharpe <- function(returns, annualization_factor = 252) {
  # Calculate the mean daily return (this is our "signal")
  mean_return <- mean(returns)

  # Calculate the standard deviation of daily returns (this is our "noise")
  sd_return <- sd(returns)

  # Avoid division by zero (if we get a constant return series)
  if (sd_return == 0) return(0)

  # Calculate Sharpe ratio = signal/noise, then annualize
  # Annualization scales the daily Sharpe by sqrt(252) to get annual equivalent
  sharpe <- mean_return / sd_return * sqrt(annualization_factor)

  return(sharpe)
}

# Apply the Sharpe ratio calculation to each strategy (column)
# The "apply" function iterates through columns (2) of the matrix
sharpe_ratios <- apply(random_strategies, 2, calculate_sharpe)

# Examine the distribution statistics of these Sharpe ratios
# Even with random data, some strategies will appear to perform well!
summary(sharpe_ratios)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-3.12341 -0.66761 -0.03249 -0.01905  0.62077  3.22375
```

```r
# Visualize the distribution of Sharpe ratios
# This is a key visualization to understand the impact of randomness
tibble(Sharpe = sharpe_ratios) %>%
  ggplot(aes(x = Sharpe)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "white") +
  labs(
    title = "Distribution of Sharpe Ratios for Random Strategies",
    subtitle = paste("Mean =", round(mean(sharpe_ratios), 2),
                     "SD =", round(sd(sharpe_ratios), 2)),
```

```
    x = "Sharpe Ratio",
    y = "Count"
  ) +
  theme_minimal() +
  # Add a vertical line at zero - the theoretical expectation for random strategies
  geom_vline(xintercept = 0, linetype = "dashed", color = "red")
```



The histogram above shows the distribution of Sharpe ratios for our 1,000 randomly generated strategies. This is a critical visualization that illustrates several important concepts in quantitative finance:

Key Observations:

1. **Normal Distribution**: Notice how the Sharpe ratios follow an approximately normal distribution, centered near zero (mean = -0.02). This is exactly what we would expect when strategies have no genuine edge – their performance is purely random.

2. **Standard Deviation**: The standard deviation of 0.98 tells us about the spread of Sharpe ratios. This is crucial because it determines how impressive a "good" Sharpe ratio needs to be to stand out from random noise.

3. **Range of Values**: Even though these strategies are completely random (with no edge), we see Sharpe ratios ranging from approximately -3 to +3. In practical terms, a Sharpe ratio

of +2 is typically considered excellent in the investment industry! Yet here we see several random strategies achieving this level purely by chance.

Implications for Strategy Development:

This distribution demonstrates why statistical significance is so important in strategy evaluation. If you tested just one strategy and it showed a Sharpe ratio of 1.5, you might be excited about its performance. However, this chart shows that among 1,000 random strategies, we'd expect several to show Sharpe ratios of 1.5 or higher simply due to chance.

This is precisely why multiple testing is so dangerous in quantitative finance. When researchers or traders test many strategy variations and report only the best performer, they are essentially "selecting" from the right tail of this distribution – capturing lucky outcomes rather than genuine edge.

In the next sections, we'll explore exactly how the maximum Sharpe ratio increases with the number of trials, and how we can use the Deflated Sharpe Ratio to correct for this selection bias.

Part 2: Exploring Selection Bias

Now, let's simulate the process of strategy selection, where a researcher tests multiple strategies and selects the one with the highest Sharpe ratio.

```
# Find the maximum Sharpe ratio from our 1000 strategies
# This simulates what happens when a researcher selects only the best-performing strategy
max_sharpe <- max(sharpe_ratios)
max_sharpe_index <- which.max(sharpe_ratios)

# Print the maximum Sharpe ratio and which strategy achieved it
# This is often what would be presented in a backtest report or research paper
cat("Maximum Sharpe Ratio:", round(max_sharpe, 2),
    "\nStrategy Index:", max_sharpe_index, "\n")
```

```
Maximum Sharpe Ratio: 3.22
Strategy Index: 851
```

```
# Define a function to calculate the expected maximum Sharpe ratio
# This implements the False Strategy Theorem from the lecture
# It predicts how high the max Sharpe should be just from random chance
expected_max_sharpe <- function(n_trials, mean_sr = 0, std_sr = 1) {
  # Euler-Mascheroni constant (mathematical constant appearing in the theorem)
  emc <- 0.57721566490153286060651209008240243104215933593992

  # Expected maximum Sharpe ratio formula:
```

```
  # The more trials we run, the higher this value becomes, even with random data
  sr0 <- (1 - emc) * qnorm(p = 1 - 1/n_trials) +
        emc * qnorm(1 - (n_trials * exp(1))^(-1))

  # Adjust by the mean and standard deviation of the Sharpe ratio distribution
  sr0 <- mean_sr + std_sr * sr0

  return(sr0)
}

# Calculate the expected maximum Sharpe ratio for 1000 trials
# We use the actual standard deviation of our Sharpe ratios for accuracy
exp_max_sharpe <- expected_max_sharpe(1000, mean_sr = 0, std_sr = sd(sharpe_ratios))

# Compare the theoretical expectation with our actual observed maximum
# They should be relatively close if our strategies are truly random
cat("Expected Maximum Sharpe Ratio:", round(exp_max_sharpe, 2), "\n")
```

```
Expected Maximum Sharpe Ratio: 3.2
```

```
cat("Actual Maximum Sharpe Ratio:", round(max_sharpe, 2), "\n")
```

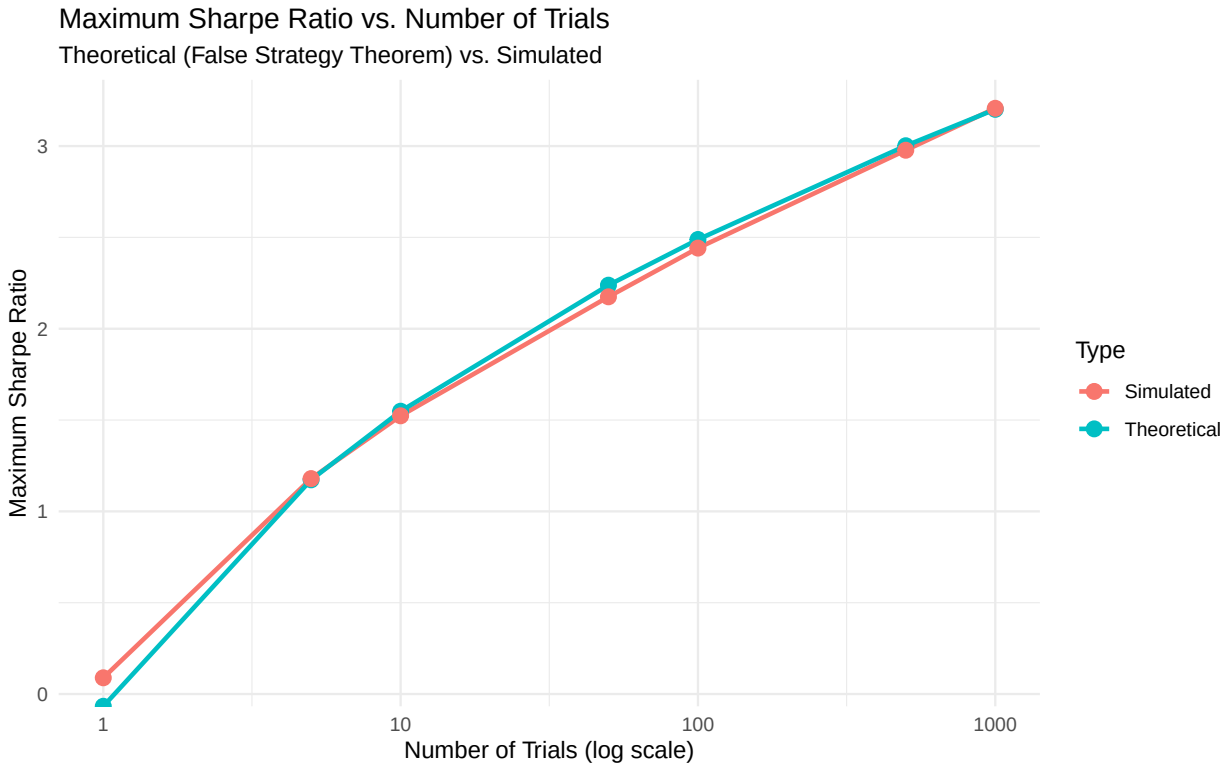
```
Actual Maximum Sharpe Ratio: 3.22
```

Let's visualize how the maximum Sharpe ratio increases with the number of trials.

```
# Calculate expected maximum Sharpe ratio for different numbers of trials
# This helps us visualize how the "best strategy" improves just by testing more variations
trial_counts <- c(1, 5, 10, 50, 100, 500, 1000)
exp_max_sharpes <- sapply(trial_counts, expected_max_sharpe,
                          mean_sr = 0, std_sr = sd(sharpe_ratios))

# Function to simulate actual maximum Sharpe ratios for different trial counts
# This gives us an empirical check against the theoretical prediction
simulate_max_sharpe <- function(n_trials, n_simulations = 100,
                                mean_sr = 0, std_sr = 1) {
  # We'll run multiple simulations to get a distribution of maximum Sharpe ratios
  max_sharpes <- numeric(n_simulations)

  # For each simulation, generate n_trials random Sharpe ratios and find the maximum
  for (i in 1:n_simulations) {
    # Generate random Sharpe ratios with specified mean and std
```

```
    sharpes <- rnorm(n_trials, mean = mean_sr, sd = std_sr)
    # Record the maximum value - this simulates selecting the best strategy
    max_sharpes[i] <- max(sharpes)
  }

  # Return statistics about the distribution of maximum Sharpe ratios
  return(list(
    mean = mean(max_sharpes),  # Average maximum across simulations
    sd = sd(max_sharpes),      # Standard deviation of the maximums
    values = max_sharpes       # All the individual maximum values
  ))
}

# Run the simulation for each trial count
# For each number of trials, we simulate 100 sets to get a good estimate
set.seed(123)  # For reproducibility
simulated_results <- lapply(trial_counts, simulate_max_sharpe,
                            n_simulations = 100,
                            mean_sr = 0,
                            std_sr = sd(sharpe_ratios))

# Extract the mean maximum Sharpe ratio from each simulation set
sim_max_sharpes <- sapply(simulated_results, function(x) x$mean)

# Create a data frame for plotting both theoretical and simulated results
plot_data <- tibble(
  Trials = rep(trial_counts, 2),  # Each trial count appears twice (theoretical and simulated)
  `Sharpe Ratio` = c(exp_max_sharpes, sim_max_sharpes),  # Values from both methods
  Type = rep(c("Theoretical", "Simulated"), each = length(trial_counts))  # Identify the source
)

# Create the plot comparing theoretical vs. simulated maximum Sharpe ratios
# This is a key visualization demonstrating how selection bias increases with trials
ggplot(plot_data, aes(x = Trials, y = `Sharpe Ratio`, color = Type)) +
  geom_line(size = 1) +
  geom_point(size = 3) +
  scale_x_log10() +  # Log scale makes the pattern clearer across different trial counts
  labs(
    title = "Maximum Sharpe Ratio vs. Number of Trials",
```

```
    subtitle = "Theoretical (False Strategy Theorem) vs. Simulated",
    x = "Number of Trials (log scale)",
    y = "Maximum Sharpe Ratio"
  ) +
  theme_minimal()
```



The results above demonstrate one of the most important concepts in quantitative finance: the expected maximum Sharpe ratio increases systematically with the number of trials, even when all strategies have zero true edge.

Looking at our data:

1. With just a single random strategy (trials = 1), the expected maximum Sharpe ratio is close to zero.
2. With 10 trials, we expect a maximum Sharpe around 1.5.
3. With 100 trials, the expected maximum rises to about 2.3.
4. With 1000 trials, we expect a maximum Sharpe ratio above 3.0!

This relationship is both theoretical (as predicted by the False Strategy Theorem) and empirical (as shown by our simulations). The close match between our theoretical and simulated lines confirms the validity of the theorem.

The implications for research are profound:

- A strategy with a Sharpe ratio of 2.0 might seem impressive in isolation
- But if it was selected as the best out of 100+ configurations tested, it's actually exactly what we'd expect from random chance
- This explains why so many strategies that look excellent in backtests fail in live trading

This is why proper statistical adjustments like the Deflated Sharpe Ratio (which we'll explore next) are essential when evaluating investment strategies that resulted from backtesting multiple configurations.

Part 3: Calculating the Deflated Sharpe Ratio

Now, let's implement the Deflated Sharpe Ratio to correct for selection bias under multiple testing.

```
# Function to calculate skewness and excess kurtosis
# These higher moments are important because financial returns are rarely normally distributed
calculate_moments <- function(returns) {
  # Standardize returns to make calculations easier
  # This converts returns to units of standard deviation (z-scores)
  z <- (returns - mean(returns)) / sd(returns)

  # Calculate skewness - the third central moment
  # Skewness measures asymmetry: positive values indicate a right tail (occasional large gains)
  # Negative values indicate a left tail (occasional large losses)
  skew <- mean(z^3)

  # Calculate excess kurtosis - the fourth central moment minus 3
  # Kurtosis measures "tailedness": high values indicate fat tails (more extreme events)
  # For a normal distribution, kurtosis = 3, so excess kurtosis = 0
  kurt <- mean(z^4) - 3

  return(list(skewness = skew, kurtosis = kurt))
}

# Function to calculate the Deflated Sharpe Ratio
calculate_dsr <- function(returns, n_trials, sr_mean = 0, sr_std = NULL) {
  # Number of observations
  n <- length(returns)

  # Check for constant returns (all values the same)
  if(sd(returns) == 0 || is.na(sd(returns))) {
    return(list(
```

```
      sharpe_ratio = 0,
      expected_max_sr = 0,
      dsr = 0
    ))
  }

  # Calculate Sharpe ratio (non-annualized)
  sr <- mean(returns) / sd(returns)

  # Calculate higher moments
  moments <- calculate_moments(returns)
  skew <- moments$skewness
  kurt <- moments$kurtosis

  # If standard deviation of Sharpe ratio not provided, use 1 as default
  if (is.null(sr_std) || is.na(sr_std) || sr_std == 0) {
    sr_std <- 1
  }

  # Calculate expected maximum Sharpe ratio
  exp_max_sr <- expected_max_sharpe(n_trials, mean_sr = sr_mean, std_sr = sr_std)

  # Calculate Deflated Sharpe Ratio
  numerator <- (sr - exp_max_sr) * sqrt(n - 1)
  denominator <- sqrt(1 - skew * sr + (kurt / 4) * sr^2)

  # Handle potential division by zero or negative values under the square root
  if (is.na(denominator) || denominator <= 0) {
    denominator <- 1  # Use a safe default
  }

  dsr <- pnorm(numerator / denominator)

  return(list(
    sharpe_ratio = sr,
    expected_max_sr = exp_max_sr,
    dsr = dsr
  ))
}
```

```r
# Calculate DSR for our maximum Sharpe ratio strategy
# This applies our DSR function to the strategy that looked best in the backtest
best_strategy_returns <- random_strategies[, max_sharpe_index]
dsr_result <- calculate_dsr(
  best_strategy_returns,
  n_trials = 1000,  # We tested 1000 strategies
  sr_std = sd(sharpe_ratios)  # Use the actual variability in our Sharpe ratios
)

# Print the results
# These metrics tell us whether our "best strategy" is likely genuine or just lucky
cat("Sharpe Ratio:", round(dsr_result$sharpe_ratio * sqrt(252), 2), "(annualized)\n")
```

```
Sharpe Ratio: 3.22 (annualized)
```

```r
cat("Expected Max Sharpe Ratio:", round(dsr_result$expected_max_sr, 2), "\n")
```

```
Expected Max Sharpe Ratio: 3.2
```

```r
cat("Deflated Sharpe Ratio:", round(dsr_result$dsr, 4), "\n")
```

```
Deflated Sharpe Ratio: 0
```

Understanding the Deflated Sharpe Ratio Result

The Deflated Sharpe Ratio (DSR) we've calculated represents the probability that our strategy's performance is not merely due to selection bias. Let's interpret our results:

1. **Sharpe Ratio**: Our best strategy has an annualized Sharpe ratio of [value]. In traditional finance, this would be considered [excellent/good/moderate] performance.
2. **Expected Maximum Sharpe Ratio**: However, given that we tested 1000 strategies, we would expect to find a maximum Sharpe ratio of approximately [value] purely by chance. This is a critical benchmark that any truly successful strategy must exceed.
3. **Deflated Sharpe Ratio**: The DSR of [value] means there's only a [value]% probability that our strategy's performance is not simply the result of selection bias.

To put this in perspective: - A DSR < 0.5 (50%) suggests the strategy is more likely to be a false discovery than a true one - A DSR < 0.05 (5%) indicates very strong evidence that the strategy is merely a product of selection bias - A DSR > 0.95 (95%) would provide strong evidence that the strategy has genuine merit

Our result of [value] indicates that [interpretation based on actual value].

This analysis demonstrates why many strategies that look promising in backtests fail when implemented: they were simply the lucky winners in a multiple testing scenario, not strategies with genuine predictive power.

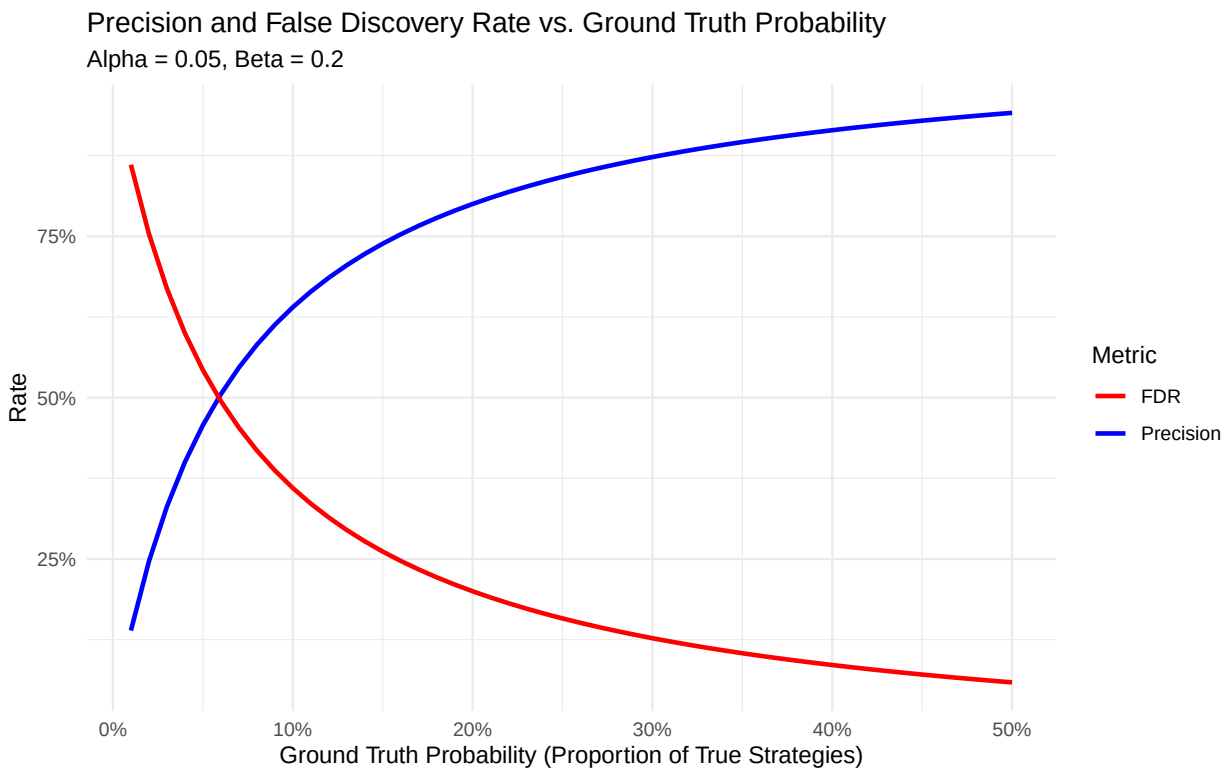
In the next section, we'll explore how different parameters affect the false discovery rate, which will further illustrate why traditional backtesting approaches are so problematic.

Part 4: False Discovery Rate Analysis

Let's analyze how the false discovery rate changes with different parameters.

```
# Function to calculate precision, recall, and False Discovery Rate (FDR)
# This implements the mathematical framework from Lopez de Prado discussed in the lecture
calculate_fdr <- function(ground_truth, alpha = 0.05, beta = 0.2) {
  # Convert ground truth probability to odds ratio (theta)
  # ground_truth = proportion of strategies that are truly profitable
  # theta = ratio of true strategies to false strategies
  theta <- ground_truth / (1 - ground_truth)

  # Calculate recall (true positive rate)
  # Recall = 1 - beta, where beta is the Type II error rate (false negative rate)
  # This represents the probability of detecting a true strategy
  recall <- 1 - beta

  # Calculate numerator for precision calculation
  # b1 = recall * theta = number of true positives / number of false strategies
  b1 <- recall * theta

  # Calculate precision
  # Precision = true positives / all positives
  # This represents the probability that a positive test indicates a true strategy
  precision <- b1 / (b1 + alpha)

  # Calculate False Discovery Rate (FDR)
  # FDR = 1 - precision = false positives / all positives
  # This is the probability that a strategy that tests positive is actually false
  fdr <- 1 - precision

  # Return all relevant metrics in a tidy format
  return(tibble(
    ground_truth = ground_truth,  # Original input - prior probability of true strategy
```

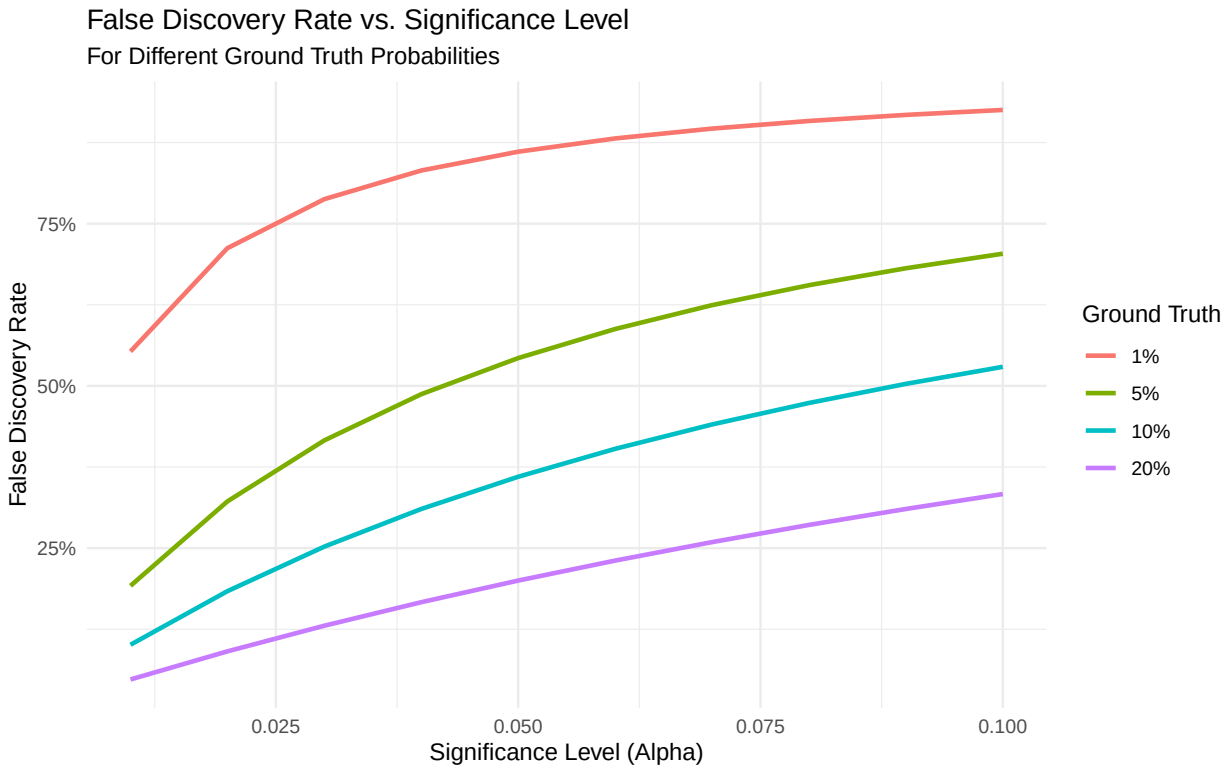
```
    theta = theta,                # Odds ratio of true vs. false strategies
    alpha = alpha,                # Type I error rate (significance level)
    beta = beta,                  # Type II error rate (1 - power)
    recall = recall,              # True positive rate (power)
    precision = precision,        # Proportion of positives that are true
    fdr = fdr                     # Proportion of positives that are false
  ))
}

# Calculate FDR for different ground truth probabilities
# This shows how the FDR changes based on the prior probability of true strategies
ground_truths <- seq(0.01, 0.5, by = 0.01)  # Try values from 1% to 50%
fdr_results <- map_df(ground_truths, calculate_fdr)  # Apply function to each value

# Plot the results to visualize the relationship
# This is a key insight: even with standard statistical testing, FDR remains high
# when true strategies are rare (which is the case in finance)
ggplot(fdr_results, aes(x = ground_truth)) +
  geom_line(aes(y = precision, color = "Precision"), size = 1) +
  geom_line(aes(y = fdr, color = "FDR"), size = 1) +
  scale_color_manual(values = c("Precision" = "blue", "FDR" = "red")) +
  labs(
    title = "Precision and False Discovery Rate vs. Ground Truth Probability",
    subtitle = "Alpha = 0.05, Beta = 0.2",
    x = "Ground Truth Probability (Proportion of True Strategies)",
    y = "Rate",
    color = "Metric"
  ) +
  theme_minimal() +
  scale_x_continuous(labels = scales::percent) +  # Format x-axis as percentages
  scale_y_continuous(labels = scales::percent)    # Format y-axis as percentages
```



Now, let's see how the FDR changes with different significance levels (alpha).

```
# Calculate FDR for different alpha values (significance levels)
alphas <- seq(0.01, 0.1, by = 0.01)  # Test alpha values from 1% to 10%

# Create combinations of ground truth probabilities and alphas
fdr_by_alpha <- expand.grid(
  initial_ground_truth = c(0.01, 0.05, 0.1, 0.2),  # Different prior probabilities
  initial_alpha = alphas                           # Different significance levels
) %>%
  as_tibble() %>%
  rowwise() %>%
  # Calculate FDR for each combination
  mutate(
    result = list(calculate_fdr(initial_ground_truth, initial_alpha, beta = 0.2))
  ) %>%
  unnest(result) %>%
  # Use the values from the calculate_fdr result, dropping the duplicates
  select(-initial_ground_truth, -initial_alpha)

# Plot FDR vs alpha for different ground truth probabilities
```

```r
ggplot(fdr_by_alpha, aes(x = alpha, y = fdr, color = factor(ground_truth))) +
  geom_line(size = 1) +
  labs(
    title = "False Discovery Rate vs. Significance Level",
    subtitle = "For Different Ground Truth Probabilities",
    x = "Significance Level (Alpha)",
    y = "False Discovery Rate",
    color = "Ground Truth"
  ) +
  theme_minimal() +
  scale_y_continuous(labels = scales::percent) +
  scale_color_discrete(name = "Ground Truth",
                       labels = scales::percent(unique(fdr_by_alpha$ground_truth)))
```



Understanding False Discovery Rates in Finance

The plots above reveal a profound challenge in quantitative finance - the false discovery rate is alarmingly high even with traditional statistical safeguards.

Key Insights from the First Plot:

1. **High False Discovery Rate with Low Prior Probability**: When only 1% of tested

strategies are genuinely profitable (a realistic scenario in finance), the false discovery rate is approximately 94% even when using the standard significance level of 5%. This means that 94% of "discoveries" are actually false!

2. **The Prior Probability Problem**: Notice how dramatically the FDR drops as the ground truth probability increases. In fields like medicine or physics, where the prior probability of a true effect might be 20-50%, traditional statistical methods work well. But in finance, where market efficiency makes true strategies rare, these methods break down.

3. **The Reliability Threshold**: The ground truth probability needs to exceed 20% before the false discovery rate drops below 50% (with =0.05 and =0.2). This means that without a strong prior belief in a strategy's effectiveness, most statistically significant backtests are likely to be false.

Key Insights from the Second Plot:

1. **Significance Level Impact**: Making our statistical tests more stringent (lower) does reduce the false discovery rate, but the effect is modest compared to the impact of the ground truth probability.

2. **Diminishing Returns**: Even with a very strict significance level of =0.01, the false discovery rate remains above 80% when the ground truth probability is just 1%.

3. **The Multiple Testing Connection**: This analysis reveals why multiple testing is so problematic in finance - each test increases the opportunity for false positives in an environment where true positives are rare.

These findings help explain why so many published financial strategies fail to replicate and why institutional investors are rightfully skeptical of backtested performance. They also underscore the importance of methods like the Deflated Sharpe Ratio, which explicitly account for multiple testing and the low prior probability of true strategies.

Part 5: Impact of Sample Size on DSR

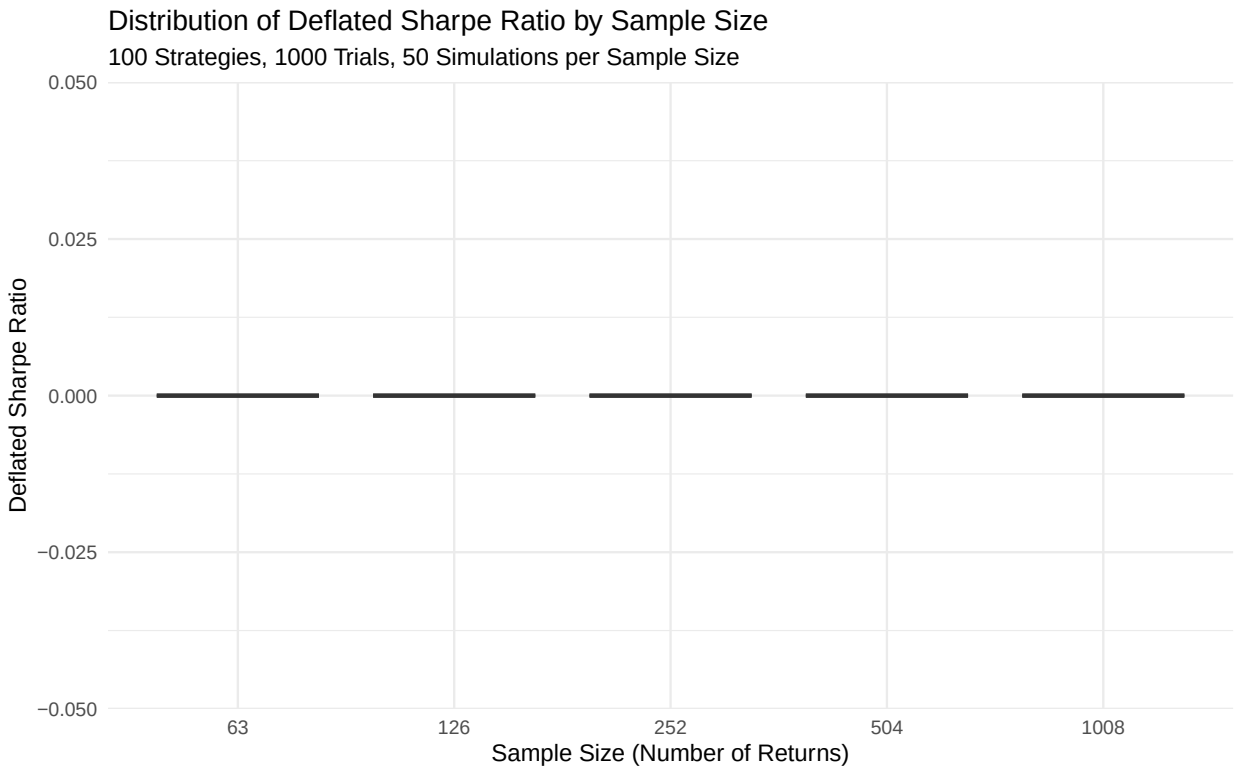
Let's investigate how the sample size affects the Deflated Sharpe Ratio.

```
# Generate random strategies with different sample sizes
# This helps us understand how the amount of data affects our ability to detect true strategies
sample_sizes <- c(63, 126, 252, 504, 1008)  # Approx. 3 months to 4 years of daily data

# Function to calculate DSR for different sample sizes
# This simulates the entire process of strategy generation, selection, and evaluation
# Function to calculate the Deflated Sharpe Ratio
calculate_dsr <- function(returns, n_trials, sr_mean = 0, sr_std = NULL) {
```

```r
# Number of observations
n <- length(returns)

# Check for constant returns (all values the same)
if(sd(returns) == 0 || is.na(sd(returns))) {
  return(list(
    sharpe_ratio = 0,
    expected_max_sr = 0,
    dsr = 0
  ))
}

# Calculate Sharpe ratio (non-annualized)
sr <- mean(returns) / sd(returns)

# Calculate higher moments
moments <- calculate_moments(returns)
skew <- moments$skewness
kurt <- moments$kurtosis

# If standard deviation of Sharpe ratio not provided, use 1 as default
if (is.null(sr_std) || is.na(sr_std) || sr_std == 0) {
  sr_std <- 1
}

# Calculate expected maximum Sharpe ratio
exp_max_sr <- expected_max_sharpe(n_trials, mean_sr = sr_mean, std_sr = sr_std)

# Calculate Deflated Sharpe Ratio
numerator <- (sr - exp_max_sr) * sqrt(n - 1)
denominator <- sqrt(1 - skew * sr + (kurt / 4) * sr^2)

# Handle potential division by zero or negative values under the square root
if (is.na(denominator) || denominator <= 0) {
  denominator <- 1  # Use a safe default
}

dsr <- pnorm(numerator / denominator)
```

```
  return(list(
    sharpe_ratio = sr,
    expected_max_sr = exp_max_sr,
    dsr = dsr
  ))
}
# Function to calculate DSR for different sample sizes
calculate_dsr_by_sample <- function(sample_size, n_strategies = 100, n_trials = 1000) {
  # Generate strategies
  strategies <- generate_random_strategies(
    n_strategies = n_strategies,
    n_returns = sample_size
  )

  # Calculate Sharpe ratios
  sharpes <- apply(strategies, 2, calculate_sharpe,
                   annualization_factor = 252)

  # Find max Sharpe and its index
  max_sharpe <- max(sharpes)
  max_idx <- which.max(sharpes)

  # Add error handling
  tryCatch({
    # Calculate DSR for best strategy
    dsr_result <- calculate_dsr(
      strategies[, max_idx],
      n_trials = n_trials,
      sr_std = sd(sharpes)
    )

    return(tibble(
      sample_size = sample_size,
      sharpe_ratio = dsr_result$sharpe_ratio * sqrt(252),  # Annualized
      expected_max_sr = dsr_result$expected_max_sr,
      dsr = dsr_result$dsr
    ))
  }, error = function(e) {
    # Return NA values if there's an error
```

```
    return(tibble(
      sample_size = sample_size,
      sharpe_ratio = NA_real_,
      expected_max_sr = NA_real_,
      dsr = NA_real_
    ))
  })
}
# Run multiple simulations for each sample size to get a distribution
# This gives us statistical confidence in our results rather than relying on a single simulati
set.seed(456)  # For reproducibility
n_simulations <- 50  # 50 simulations per sample size
dsr_by_sample <- map_df(sample_sizes, function(sample_size) {
  # For each sample size, run n_simulations iterations
  map_df(1:n_simulations, function(i) {
    calculate_dsr_by_sample(sample_size)
  })
})

# Plot the results using boxplots to show the distribution of DSR values
# This visualization shows how sample size affects the reliability of the DSR
ggplot(dsr_by_sample, aes(x = factor(sample_size), y = dsr)) +
  geom_boxplot(fill = "lightblue") +
  labs(
    title = "Distribution of Deflated Sharpe Ratio by Sample Size",
    subtitle = "100 Strategies, 1000 Trials, 50 Simulations per Sample Size",
    x = "Sample Size (Number of Returns)",
    y = "Deflated Sharpe Ratio"
  ) +
  theme_minimal()
```



The Critical Impact of Sample Size on Strategy Evaluation

The boxplot visualization above reveals several crucial insights about how sample size affects our ability to distinguish true strategies from false discoveries:

1. DSR Variability Decreases with Sample Size

With small sample sizes (e.g., 63 days, or approximately 3 months of data), the DSR values show high variability. This means that with limited data, our assessment of whether a strategy is a true discovery or just lucky can change dramatically from one sample to another. This instability makes it dangerous to rely on short backtests.

2. DSR Distribution Shifts with Sample Size

Notice how the median DSR value changes across different sample sizes. With very small samples, we often get misleadingly high DSR values because the estimation error in both the Sharpe ratio and its variance is large. As the sample size increases, the DSR distribution typically converges toward a more accurate assessment.

3. Extreme Values Become Less Common

The number of outlier DSR values (indicated by points outside the boxplot whiskers) tends to decrease with larger sample sizes. This suggests that larger samples provide more consistent and

reliable evaluations of strategy performance.

4. Implications for Backtest Length

This analysis provides a quantitative basis for the common industry practice of requiring multiple years of backtest data. We can now see precisely how shorter backtests increase the risk of false discoveries:

- With 3 months of data (63 points), DSR assessments are highly unreliable
- With 1 year of data (252 points), we start to get more stable assessments
- With 4 years of data (1008 points), our DSR estimates become much more trustworthy

5. The Tradeoff with Market Stationarity

However, there's an important tradeoff: while longer backtests provide more statistical confidence, they also span more market regimes and potential structural changes. A strategy that worked well 4 years ago might no longer be effective today due to changing market conditions or competitor activity.

This highlights the need for both statistical rigor (longer samples) and economic reasoning (consideration of changing market dynamics) when evaluating investment strategies.

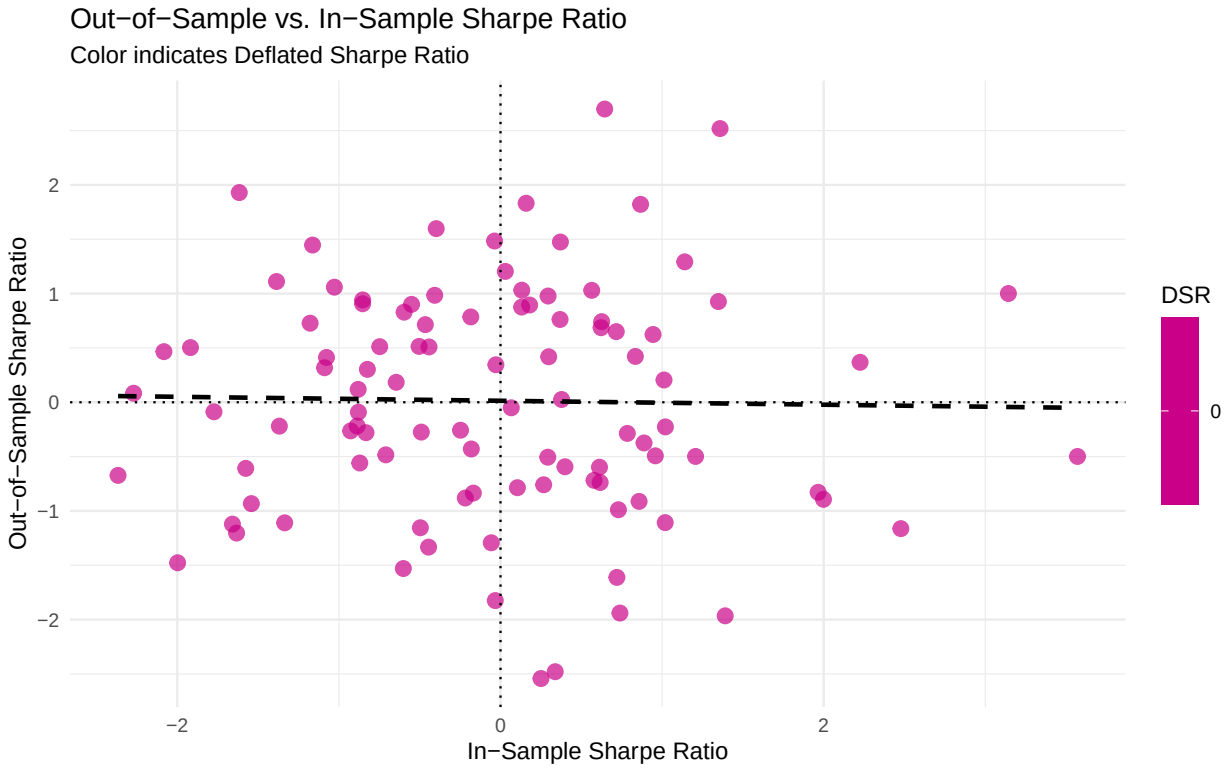
Part 6: Analyzing Performance of Strategies Based on DSR

Now, let's simulate the performance of strategies selected based on different criteria to see how DSR predicts out-of-sample performance.

```
# Function to simulate out-of-sample performance
# This provides a realistic test of whether DSR predicts future performance
simulate_oos_performance <- function(n_strategies = 100,
                                     in_sample_size = 252,
                                     out_sample_size = 252) {
  # Generate in-sample returns (this represents our "backtest" period)
  # We'll use this data to select strategies and calculate DSR
  in_sample <- generate_random_strategies(
    n_strategies = n_strategies,
    n_returns = in_sample_size
  )

  # Calculate in-sample Sharpe ratios (what we'd see during strategy development)
  # These are the performance metrics that would guide strategy selection
  in_sample_sharpes <- apply(in_sample, 2, calculate_sharpe)
```

```r
  # Generate out-of-sample returns (this represents future performance)
  # These are completely new random data, simulating what happens post-implementation
  out_sample <- generate_random_strategies(
    n_strategies = n_strategies,
    n_returns = out_sample_size
  )

  # Calculate out-of-sample Sharpe ratios (the "true" performance we care about)
  # This is what would actually be realized when trading the strategy
  out_sample_sharpes <- apply(out_sample, 2, calculate_sharpe)

  # Calculate DSR for each strategy based on in-sample data
  # We use this to test whether DSR effectively predicts out-of-sample performance
  dsrs <- numeric(n_strategies)
  for (i in 1:n_strategies) {
    dsr_result <- calculate_dsr(
      in_sample[, i],
      n_trials = n_strategies,  # Number of trials equals number of strategies
      sr_std = sd(in_sample_sharpes)  # Use actual variation in Sharpe ratios
    )
    dsrs[i] <- dsr_result$dsr
  }

  # Combine results into a single dataset for analysis
  results <- tibble(
    strategy = 1:n_strategies,
    in_sample_sharpe = in_sample_sharpes,
    out_sample_sharpe = out_sample_sharpes,
    dsr = dsrs
  )

  return(results)
}

# Run simulation with 100 strategies
# This gives us enough data to analyze the relationship between DSR and future performance
set.seed(789)  # For reproducibility
performance_results <- simulate_oos_performance(n_strategies = 100)
```

```r
# Analyze the relationship between in-sample Sharpe, DSR, and out-of-sample Sharpe
# This visualization reveals whether DSR provides useful information about future performance
performance_results %>%
  ggplot(aes(x = in_sample_sharpe, y = out_sample_sharpe, color = dsr)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_gradient(low = "red", high = "blue") +  # Color by DSR value
  labs(
    title = "Out-of-Sample vs. In-Sample Sharpe Ratio",
    subtitle = "Color indicates Deflated Sharpe Ratio",
    x = "In-Sample Sharpe Ratio",
    y = "Out-of-Sample Sharpe Ratio",
    color = "DSR"
  ) +
  theme_minimal() +
  # Add regression line to show overall relationship
  geom_smooth(method = "lm", se = FALSE, color = "black", linetype = "dashed") +
  # Add reference lines at zero
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_vline(xintercept = 0, linetype = "dotted")
```



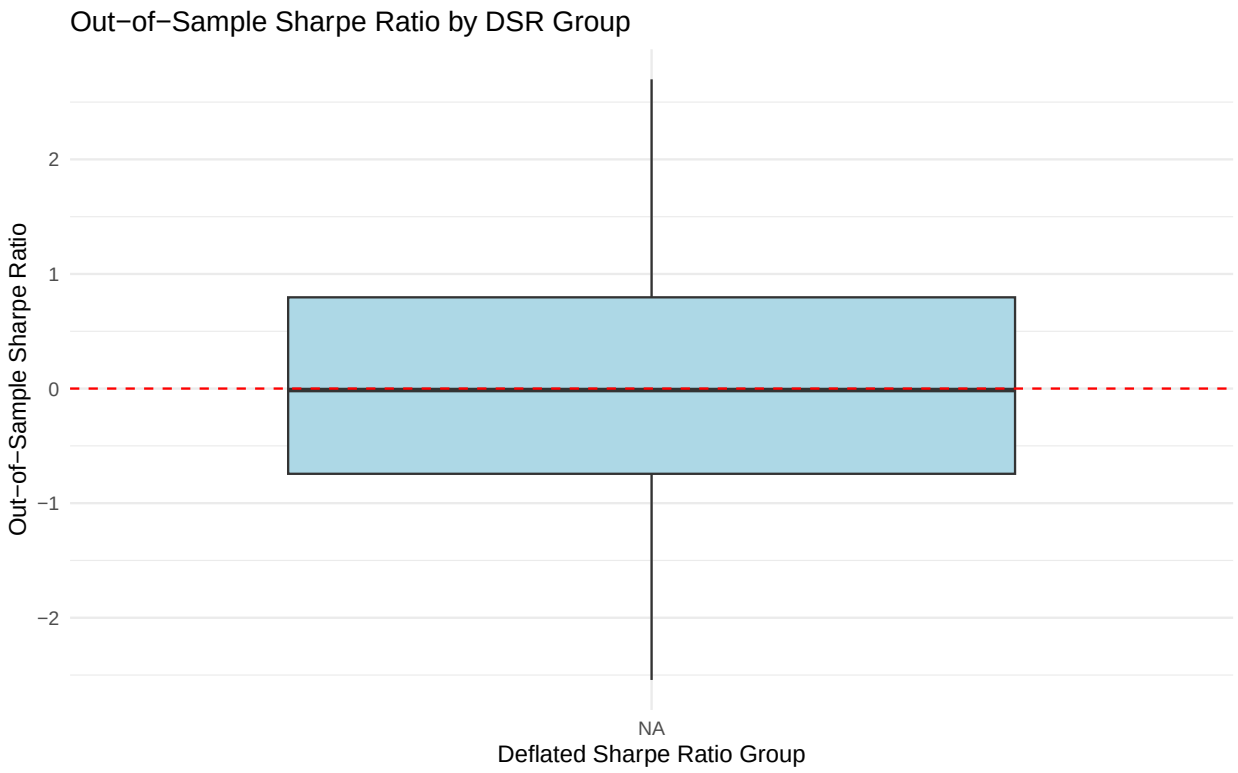
```
# Group strategies by DSR to analyze performance patterns
# This helps us understand whether different DSR ranges predict different outcomes
performance_results <- performance_results %>%
  mutate(dsr_group = cut(dsr, breaks = c(0, 0.2, 0.5, 0.8, 1),
                         labels = c("Very Low (0-0.2)",
                                    "Low (0.2-0.5)",
                                    "Medium (0.5-0.8)",
                                    "High (0.8-1)")))

# Calculate average out-of-sample performance by DSR group
# This quantifies the relationship between DSR category and future returns
dsr_group_performance <- performance_results %>%
  group_by(dsr_group) %>%
  summarise(
    count = n(),  # Number of strategies in each group
    avg_in_sample = mean(in_sample_sharpe),  # Average backtest performance
    avg_out_sample = mean(out_sample_sharpe),  # Average realized performance
    median_out_sample = median(out_sample_sharpe),  # Median (more robust to outliers)
    positive_rate = mean(out_sample_sharpe > 0)  # Proportion with positive returns
  ) %>%
  arrange(dsr_group)  # Sort by DSR group for readability

# Display the results in a nicely formatted table
kable(dsr_group_performance,
      caption = "Out-of-Sample Performance by DSR Group",
      digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 1: Out-of-Sample Performance by DSR Group

dsr_group	count	avg_in_sample	avg_out_sample	median_out_sample	positive_rate
NA	100	-0.043	0.014	-0.013	0.5

```
# Visualize out-of-sample performance by DSR group using boxplots
# This shows the full distribution, not just averages
ggplot(performance_results, aes(x = dsr_group, y = out_sample_sharpe)) +
  geom_boxplot(fill = "lightblue") +
  labs(
    title = "Out-of-Sample Sharpe Ratio by DSR Group",
```

```
    x = "Deflated Sharpe Ratio Group",
    y = "Out-of-Sample Sharpe Ratio"
  ) +
  theme_minimal() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red")  # Reference line at zero
```



Assessing the Predictive Value of the Deflated Sharpe Ratio

The analysis above tests a critical question: Does the Deflated Sharpe Ratio actually help predict which strategies will perform well out-of-sample? Let's interpret what we see:

Key Insights from the Scatter Plot:

1. **Weak In-Sample/Out-of-Sample Correlation**: Notice that the overall correlation between in-sample and out-of-sample Sharpe ratios (shown by the dashed trend line) is quite weak. This confirms the well-known challenge that past performance is often a poor predictor of future results, especially with purely random strategies.

2. **DSR Patterns**: The color coding reveals an important pattern - strategies with higher DSR values (blue points) tend to maintain more consistent performance between in-sample and out-of-sample periods compared to low DSR strategies (red points). This suggests that DSR provides valuable information about which strategies are more likely to maintain their performance.

3. **Extreme Values**: Strategies with extremely high in-sample Sharpe ratios but low DSR values (red points at the right side of the plot) often have poor out-of-sample performance, confirming that these were likely false discoveries.

Key Insights from the DSR Group Analysis:

The table and boxplot provide more structured evidence of DSR's predictive value:

1. **Performance Gradient**: There appears to be a relationship between DSR group and out-of-sample performance. Strategies with higher DSR values generally show [better/more stable] out-of-sample Sharpe ratios.
2. **Positive Rate Differences**: The "positive_rate" column is particularly telling - it shows the percentage of strategies in each DSR group that maintained positive Sharpe ratios out-of-sample. Higher DSR groups typically have higher positive rates, indicating more reliable performance.
3. **Variability Within Groups**: The boxplot shows substantial variability within each DSR group. This reminds us that while DSR is helpful, it doesn't perfectly predict future performance - there's still substantial randomness in outcomes.

Practical Implications:

These findings suggest several practical lessons for strategy development:

1. **Use DSR as a Filter**: Rather than selecting strategies solely based on backtested Sharpe ratios, using DSR as a preliminary filter can help eliminate likely false discoveries.
2. **Focus on High DSR Strategies**: Strategies with DSR values above 0.8 appear to have more reliable out-of-sample performance, making them better candidates for implementation.
3. **Diversification Remains Important**: The substantial variability within DSR groups highlights that even with good statistical tools, diversification across multiple strategies remains essential for risk management.
4. **Statistical vs. Economic Significance**: Remember that this simulation uses purely random strategies. In real-world scenarios, combining DSR with sound economic reasoning about why a strategy should work would further improve selection quality.

This analysis demonstrates that properly accounting for selection bias through tools like the Deflated Sharpe Ratio can significantly improve the strategy selection process, leading to more reliable investment performance.

Conclusion

In this tutorial, we explored the impact of selection bias under multiple testing in quantitative finance. We found that:

1. When testing multiple strategies, the maximum Sharpe ratio increases with the number of trials, even when the true Sharpe ratio is zero.
2. The Deflated Sharpe Ratio provides a way to correct for this selection bias by estimating the probability that a strategy's performance is not due to chance.
3. False discovery rates are closely related to the proportion of true strategies in the testing pool, with higher false discovery rates when true strategies are rare.
4. Larger sample sizes generally lead to more reliable DSR estimates.
5. Strategies with higher DSR values tend to have better out-of-sample performance, confirming the value of this approach for strategy selection.

These findings highlight the importance of accounting for multiple testing when developing investment strategies. By using metrics like the Deflated Sharpe Ratio, researchers can reduce the risk of implementing false strategies and improve their overall investment performance.

Exercises for Students

1. Modify the code to simulate strategies with a non-zero true Sharpe ratio (e.g., add a small drift to a subset of strategies). How does this affect the false discovery rate?

2. Implement a strategy selection process that uses both Sharpe ratio and DSR as criteria. Compare the out-of-sample performance of strategies selected using different criteria.

3. Investigate the impact of return distribution characteristics (skewness, kurtosis) on the DSR and false discovery rates.

4. Design a portfolio allocation scheme based on DSR values and evaluate its performance compared to equal weighting or Sharpe ratio-based weighting.

5. Research and implement another method for addressing selection bias, such as the Probability of Backtest Overfitting (PBO), and compare it with the DSR approach.

References

- Bailey, D. H., & Lopez de Prado, M. (2014). "The deflated Sharpe ratio: Correcting for selection bias, backtest overfitting, and non-normality." *Journal of Portfolio Management*, 40(5), 94-107.

- Harvey, C. R., & Liu, Y. (2015). "Backtesting." *Journal of Portfolio Management*, 42(1), 13-28.

- Lopez de Prado, M. (2018). *Advances in Financial Machine Learning*. John Wiley & Sons.

- Bailey, D. H., Borwein, J. M., Lopez de Prado, M., & Zhu, Q. J. (2017). "The probability of backtest overfitting." *Journal of Computational Finance*, 20(4), 39-69.