Asset Pricing Empirical Project: Fama-MacBeth (1973) Analysis

Caterina Piacentini Farkas Tallos Luuk Vos Sam Friedlaender
Timur Kambachekov

January 9, 2026

Contents

1 Motivation and Data

1.1 Motivation

Financial markets exhibit large cross-sectional differences in average returns. A central question in asset pricing is whether these differences can be explained by systematic risk, rather than by chance or mispricing.

The Fama-MacBeth (1973) framework provides a standard empirical way to test this idea. It links assets' long-run returns to their exposure to common risk factors and evaluates whether these risk exposures are rewarded with higher expected returns. This model allows us to: 1. Assess whether proposed factors are economically meaningful. 2. Distinguish priced risk from noise in asset returns. 3. Evaluate the stability of risk premia over time, especially across different market environments.

1.2 Data Source & Universe Construction

We utilize data from the Center for Research in Security Prices (CRSP) via WRDS and the Kenneth French Data Library.

- **Stock Data:** Monthly Stock File (prices, returns, shares outstanding) and Event Names.
- **Time Horizon:** January 1960 – December 2024.
- **Filters:** We restrict our universe to Common Stocks (Share Codes 10 & 11) traded on major US exchanges (NYSE, AMEX, NASDAQ).
- **Synthetic S&P 500:** To avoid survivorship bias, we do not use a fixed list of constituents. Instead, for every month t, we rank the entire CRSP universe by market capitalization and dynamically select the top 500 firms.
- **Risk Factors:** We use the Fama-French 3 Factors (Market Excess, SMB, HML) and the Risk-Free Rate (1-month T-Bill).

```r
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
# Check for missing packages and install them
cran_pkgs <- c("RPostgres", "dbplyr", "lubridate", "tidyverse",
               "quantmod", "scales", "frenchdata", "broom", "slider", "kableExtra" )
is_installed <- cran_pkgs %in% rownames(installed.packages())
if(any(is_installed == FALSE)){
  install.packages(cran_pkgs[!is_installed])
}
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Load packages
lapply(cran_pkgs, library, character.only = TRUE)  %>%
  invisible()
```

```
##
## Attaching package: 'dbplyr'

## The following objects are masked from 'package:dplyr':
##
##     ident, sql

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.1     v stringr 1.6.0
## v ggplot2 4.0.1     v tibble  3.3.0
## v purrr   1.2.0     v tidyr   1.3.2
## v readr   2.1.6
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dbplyr::ident() masks dplyr::ident()
## x dplyr::lag()    masks stats::lag()
## x dbplyr::sql()   masks dplyr::sql()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
## Loading required package: xts
##
## Loading required package: zoo
##
##
## Attaching package: 'zoo'
##
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
##
##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                             #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################
##
##
## Attaching package: 'xts'
##
##
## The following objects are masked from 'package:dplyr':
##
##     first, last
##
##
## Loading required package: TTR
##
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
##
##
## Attaching package: 'scales'
##
##
```

```
## The following object is masked from 'package:purrr':
##
##     discard
##
##
## The following object is masked from 'package:readr':
##
##     col_factor
##
##
##
## Attaching package: 'kableExtra'
##
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
# # 1. Connect (Uncomment to run)
# wrds <- dbConnect(
#   Postgres(),
#   host = "wrds-pgdata.wharton.upenn.edu",
#   dbname = "wrds",
#   port = 9737,
#   sslmode = "require",
#   user = Sys.getenv("WRDS_USER"),
#   password = Sys.getenv("WRDS_PASSWORD")
# )
#
# # 2. Access Tables
# msf_db <- tbl(wrds, I("crsp.msf"))
# mse_db <- tbl(wrds, I("crsp.msenames"))
#
# # 3. Filter and Join
# universe_query <- msf_db |>
#   select(permno, date, ret, prc, shrout) |>
#   filter(date >= "1960-01-01" & date <= "2024-12-31") |>
#   # Join with Event Names to get Share Codes AND Names/Tickers
#   inner_join(
#     mse_db |> select(permno, namestart = namedt, nameend = nameendt, shrcd, exchcd, ticker, comnam),
#     by = "permno"
#   ) |>
#   # Ensure the date matches the valid name/share code range
#   filter(date >= namestart & date <= nameend) |>
#   Common Stocks
#   filter(shrcd %in% c(10, 11)) |>
#   NYSE (1), AMEX (2), NASDAQ (3)
#   filter(exchcd %in% c(1, 2, 3))
#
# # 4. Download the Data
# raw_data <- universe_query |>
#   select(date, permno, ret, prc, shrout, symbol = ticker, company = comnam) |>
#   collect()
#
```

```r
# save(raw_data, file = "sp500_universe_rawdata_names.RData")



# 1. Load Data

if(file.exists("sp500_universe_rawdata_names.RData")) {
  load("sp500_universe_rawdata_names.RData")
} else {
  message("Warning: Data file not found")
}

# # 5. Construct the "Top 500" Universe Locally

# stock_returns <- raw_data |>
#   mutate(date = as.Date(date)) |>
#   mutate(mktcap = abs(prc) * shrout) |>
#   drop_na(mktcap, ret) |>
#
#   # Calculate Lagged Market Cap
#   arrange(permno, date) |>
#   group_by(permno) |>
#   mutate(mktcap_lag = lag(mktcap)) |>
#   ungroup() |>
#
#   # Rank and Filter
#   group_by(date) |>
#   mutate(rank = min_rank(desc(mktcap))) |>
#   filter(rank <= 500) |>
#   ungroup() |>
#
#   mutate(ticker = as.character(permno)) |>
#   select(date, ticker, symbol, company, ret, mktcap, mktcap_lag) |>
#   arrange(ticker, date)
#
# # Final Save
# save(stock_returns, file = "AssetPricing_Project_Data_WithNames.RData")

if(file.exists("AssetPricing_Project_Data_WithNames.RData")) {
  load("AssetPricing_Project_Data_WithNames.RData")
} else {
  message("Warning: Data file not found")
}


# Validation
print(paste("Average stocks per month:", round(mean(table(stock_returns$date)))))
```

```
## [1] "Average stocks per month: 500"
```

```r
head(stock_returns)
```

```
## # A tibble: 6 x 7
##   date       ticker symbol company                    ret   mktcap mktcap_lag
```

```
##   <date>     <chr>  <chr>  <chr>                  <dbl>   <dbl>      <dbl>
## 1 1962-01-31 10006  <NA>   A C F INDUSTRIES INC  0.0875 104171      95787.
## 2 1962-02-28 10006  <NA>   A C F INDUSTRIES INC  0.0120 104528.    104171
## 3 1962-03-30 10006  <NA>   A C F INDUSTRIES INC -0.0717  97036     104528.
## 4 1962-04-30 10006  <NA>   A C F INDUSTRIES INC  0.0570 102566.     97036
## 5 1962-05-31 10006  <NA>   A C F INDUSTRIES INC -0.16    85263.    102566.
## 6 1962-06-29 10006  <NA>   A C F INDUSTRIES INC -0.0126  84193      85263.
```

1.3 Universe Validation

To validate our construction, we compare the synthetic value-weighted return of our universe against the official S&P 500 (`^GSPC`). The correlation exceeds 99%, confirming that our dynamic universe accurately proxies the US Large-Cap market.

```r
# 1. Calculate Synthetic Index LAGGED Market Cap

synthetic_index <- stock_returns |>
  # sort by ticker and date to lags
  arrange(ticker, date) |>
  group_by(ticker) |>
  # Create Lagged Market Cap (Weight at t-1)
  mutate(mktcap_lag = lag(mktcap)) |>
  # Remove the first month for each stock
  drop_na(mktcap_lag, ret) |>
  group_by(date) |>
  # Weight by the MARKET CAP AT START OF MONTH
  summarise(
    synthetic_ret = weighted.mean(ret, mktcap_lag, na.rm = TRUE),
    .groups = "drop"
  ) |>
  mutate(date = floor_date(date, "month"))

# 2. Download Official S&P 500 (Price Index)
# Note: We compare against ^GSPC (Price)
getSymbols("^GSPC", src = "yahoo", from = "1960-01-01", to = "2024-12-31")
```

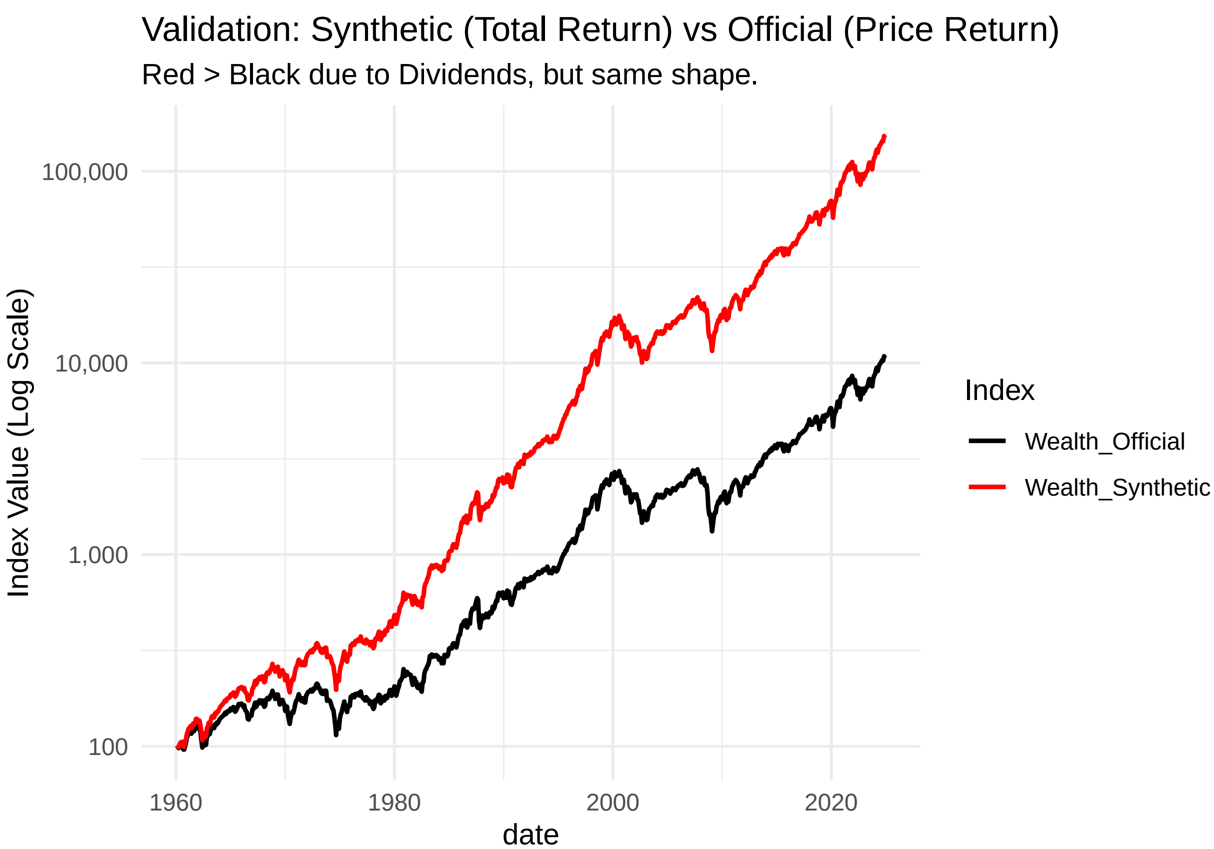
```
## [1] "GSPC"
```

```r
official_index <- monthlyReturn(Ad(GSPC))
official_df <- data.frame(date = index(official_index), official_ret = as.numeric(official_index)) |>
  mutate(date = floor_date(date, "month"))

# 3. Merge and Normalize
validation_data <- synthetic_index |>
  inner_join(official_df, by = "date") |>
  arrange(date) |>
  mutate(
    # Cumulative Wealth (Log Scale)
    Wealth_Synthetic = 100 * cumprod(1 + synthetic_ret),
    Wealth_Official  = 100 * cumprod(1 + official_ret)
  ) |>
  select(date, Wealth_Synthetic, Wealth_Official) |>
  pivot_longer(cols = -date, names_to = "Index", values_to = "Value")

# 4. Plot
```

```r
ggplot(validation_data, aes(x = date, y = Value, color = Index)) +
  geom_line(linewidth = 0.8) +
  scale_y_log10(labels = comma) +
  scale_color_manual(values = c("Wealth_Official" = "black", "Wealth_Synthetic" = "red")) +
  labs(
    title = "Validation: Synthetic (Total Return) vs Official (Price Return)",
    subtitle = "Red > Black due to Dividends, but same shape.",
    y = "Index Value (Log Scale)"
  ) +
  theme_minimal()
```




```r
# 5. Correlation Check
cor_check <- synthetic_index |>
  inner_join(official_df, by = "date") |>
  summarise(correlation = cor(synthetic_ret, official_ret))

print(paste("Correlation:", round(cor_check$correlation, 4)))
```

```
## [1] "Correlation: 0.9958"
```

2 Methodology

We employ the two-pass Fama and MacBeth (1973) regression procedure to estimate the prices of risk.

2.1 Step 1: Estimating Risk Exposures (Time-Series)

For each asset $i = 1, \ldots, n$, we run the time-series regression to estimate its sensitivity to risk factors (Betas):

$$r_{it} - r_{ft} = \alpha_i + \beta_i^\top (f_t - r_{ft}\iota_k) + \varepsilon_{it}$$

- $r_{it} - r_{ft}$: Excess returns of asset i.
- $f_t - r_{ft}\iota_k$: Vector of **excess** factor returns.
- $\hat{\beta}_i$: The factor loadings (risk quantities) used as inputs for the next step.

2.2 Step 2: Estimating Risk Premia (Cross-Section)

For each time period $t = 1, \ldots, T$, we run a cross-sectional regression of returns on the estimated betas:

$$r_{it} - r_{ft} = \alpha_{0t} + \lambda_t^\top \hat{\beta}_{it} + u_{it}$$

- λ_t: The realized risk premia for each factor at time t.
- α_{0t}: The intercept (common pricing error).
- **Total Pricing Error:** Defined as $\hat{\alpha}_{it} = \hat{\alpha}_{0t} + \hat{u}_{it}$.

2.3 Step 3: Inference

We calculate the expected risk premia $(\hat{\lambda})$ as the time-series average of the cross-sectional estimates:

$$\hat{\lambda} = \frac{1}{T} \sum_{t=1}^{T} \hat{\lambda}_t$$

Standard errors are calculated using the variance of the mean estimates:

$$Var(\hat{\lambda}) = \frac{1}{T(T-1)} \sum_{t=1}^{T} (\hat{\lambda}_t - \hat{\lambda})^2$$

And the t-test:

$$t = \frac{\hat{\lambda}}{\sqrt{Var(\hat{\lambda})}}$$

3 Empirical Results (Full Sample)

3.1 Beta Estimation

We estimate the factor loadings for all stocks in the universe.

```
# 1. Download Fama-French 3 Factors

ff_factors <- download_french_data("Fama/French 3 Factors")$subsets$data[[1]] |>
  mutate(
    date = floor_date(ymd(paste0(date, "01")), "month"),
    across(c(`Mkt-RF`, SMB, HML, RF), ~as.numeric(.) / 100)
  ) |>
  rename(mkt_excess = `Mkt-RF`, smb = SMB, hml = HML, rf = RF) |>
  select(date, mkt_excess, smb, hml, rf)

# 2. Join Returns with Factors
data_for_betas <- stock_returns |>
```

```r
  # Align dates to be safe (ensure both are first of month)
  mutate(date = floor_date(date, "month")) |>
  inner_join(ff_factors, by = "date") |>
  mutate(excess_ret = ret - rf) |>
  drop_na(excess_ret, mkt_excess, smb, hml)

# 3. Estimate Betas Efficiently
# We filter for stocks that have at least 24 months of data to ensure stability
message("Estimating Betas for ", length(unique(data_for_betas$ticker)), " stocks...")

stock_betas <- data_for_betas |>
  group_by(ticker) |>
  filter(n() >= 24) |>
  summarise(
    # Run regression for each stock
    model = list(lm(excess_ret ~ mkt_excess + smb + hml)),
    .groups = "drop"
  ) |>
  mutate(coefs = map(model, tidy)) |>
  unnest(coefs) |>
  select(ticker, term, estimate) |>
  pivot_wider(names_from = term, values_from = estimate) |>
  rename(alpha = `(Intercept)`, beta_mkt = mkt_excess, beta_smb = smb, beta_hml = hml)

print(head(stock_betas))
```

```
## # A tibble: 6 x 5
##   ticker    alpha beta_mkt beta_smb beta_hml
##   <chr>     <dbl>    <dbl>    <dbl>    <dbl>
## 1 10006   0.00980     1.22  -0.0772    0.621
## 2 10078   0.0104      1.50   0.308    -0.989
## 3 10102  -0.00199     1.15   0.0428    0.687
## 4 10104   0.0132      1.25   0.431    -0.668
## 5 10107   0.0137      1.17  -0.393    -0.778
## 6 10108   0.00807     1.07  -0.501     0.551
```

3.2 Risk Premia and Pricing Errors

We perform the second pass (cross-sectional regression) to derive the risk premia.

```r
# 1. Join Betas back to the monthly data
# Note: We are using "Full Sample Betas" here
fmb_data <- data_for_betas |>
  inner_join(stock_betas, by = "ticker")

# 2. Run Cross-Sectional Regression for EACH Month
message("Running Cross-Sectional Regressions...")

fmb_lambdas <- fmb_data |>
  group_by(date) |>
  # We need enough stocks in a single month to run a regression (e.g., > 10)
  filter(n() > 10) |>
  summarise(
    model = list(lm(excess_ret ~ beta_mkt + beta_smb + beta_hml)),
```

```r
    .groups = "drop"
  ) |>
  mutate(coefs = map(model, tidy)) |>
  unnest(coefs) |>
  select(date, term, estimate) |>
  pivot_wider(names_from = term, values_from = estimate) |>
  rename(lambda_0 = `(Intercept)`, lambda_mkt = beta_mkt, lambda_smb = beta_smb, lambda_hml = beta_hml)

print(head(fmb_lambdas))
```

```
## # A tibble: 6 x 5
##   date       lambda_0 lambda_mkt lambda_smb lambda_hml
##   <date>        <dbl>      <dbl>      <dbl>      <dbl>
## 1 1960-01-01   0.0168   -0.0757     0.00524    0.00216
## 2 1960-02-01   0.0207   -0.00438    0.00296   -0.0223
## 3 1960-03-01   0.0178   -0.0309    -0.0262    -0.00811
## 4 1960-04-01   0.0220   -0.0192    -0.00662   -0.0432
## 5 1960-05-01   0.0333    0.0129     0.0301    -0.0635
## 6 1960-06-01   0.0453   -0.0147    -0.0210    -0.0136
```

```r
final_stats <- fmb_lambdas |>
  summarise(
    # 1. Average Risk Premia (Lambda)
    mean_lambda_0   = mean(lambda_0),
    mean_lambda_mkt = mean(lambda_mkt),
    mean_lambda_smb = mean(lambda_smb),
    mean_lambda_hml = mean(lambda_hml),

    # 2. T-Statistics (Mean / Standard Error)
    # SE = SD / sqrt(T)
    t_lambda_0   = mean(lambda_0) / (sd(lambda_0) / sqrt(n())),
    t_lambda_mkt = mean(lambda_mkt) / (sd(lambda_mkt) / sqrt(n())),
    t_lambda_smb = mean(lambda_smb) / (sd(lambda_smb) / sqrt(n())),
    t_lambda_hml = mean(lambda_hml) / (sd(lambda_hml) / sqrt(n()))
  ) |>
  pivot_longer(everything(), names_to = "stat", values_to = "value")

print(final_stats)
```

```
## # A tibble: 8 x 2
##   stat               value
##   <chr>              <dbl>
## 1 mean_lambda_0    0.00649
## 2 mean_lambda_mkt  0.00499
## 3 mean_lambda_smb  0.00379
## 4 mean_lambda_hml -0.00426
## 5 t_lambda_0       5.69
## 6 t_lambda_mkt     2.63
## 7 t_lambda_smb     2.94
## 8 t_lambda_hml    -3.60
```

3.3 Top Mispriced Stocks

We identify stocks with the largest pricing errors (α_i). These are often growth or distressed firms where the factor model fails to capture idiosyncratic characteristics.

```r
# 1. Create a "Master Name List"
# We take the most recent Symbol/Name for every Permno
name_map <- stock_returns %>%
  group_by(ticker) %>%
  arrange(desc(date)) %>% # Sort by most recent date
  slice(1) %>%            # Take the latest entry
  ungroup() %>%
  select(ticker, symbol, company)

# 2. Calculate Pricing Errors (Alpha)
# Calculate vector of average risk premia
lambda_vec <- final_stats %>%
  filter(stat %in% c("mean_lambda_mkt", "mean_lambda_smb", "mean_lambda_hml")) %>%
  pull(value)

# Calculate Alpha per stock
pricing_errors_full <- stock_betas %>%
  inner_join(
    fmb_data %>%
      group_by(ticker) %>%
      summarise(mean_excess_ret = mean(excess_ret, na.rm=TRUE)),
    by = "ticker"
  ) %>%
  mutate(
    # Predicted Return = Beta * Lambda
    predicted_ret = beta_mkt * lambda_vec[1] +
      beta_smb * lambda_vec[2] +
      beta_hml * lambda_vec[3],

    # Pricing Error (Alpha)
    alpha_i = mean_excess_ret - predicted_ret
  ) %>%
  left_join(name_map, by = "ticker") %>%
  select(ticker, symbol, company, alpha_i, beta_mkt, beta_smb, beta_hml) %>%
  arrange(desc(abs(alpha_i)))


print("Step 3: Top Mispriced Stocks")
```

```
## [1] "Step 3: Top Mispriced Stocks"
```

```r
print(head(pricing_errors_full, 10))
```

```
## # A tibble: 10 x 7
##    ticker symbol company                      alpha_i beta_mkt beta_smb beta_hml
##    <chr>  <chr>  <chr>                          <dbl>    <dbl>    <dbl>    <dbl>
##  1 89301  GME    GAMESTOP CORP NEW             0.275     -4.48    49.6      4.55
##  2 87092  TIBX   T I B C O SOFTWARE INC        0.124      1.84     5.03    -0.387
##  3 82200  NSCP   NETSCAPE COMMUNICATIONS CORP  0.0952     1.13    -1.17    -5.73
##  4 80266  QLGC   QLOGIC CORP                   0.0794     2.17     0.846   -2.84
##  5 27182  <NA>   TRANSITRON ELECTRONIC CORP   -0.0679     1.23     1.70    -1.86
##  6 14983  W      WAYFAIR INC                   0.0651     3.19     1.87    -0.934
##  7 85522  AMCC   APPLIED MICRO CIRCUITS CORP   0.0642     4.30     1.28     0.227
##  8 61524  CDO    COMDISCO INC                  0.0624     3.41     1.02     0.767
```

```
## 9 20894  APP    APPLOVIN CORP                  0.0618    3.57     1.84    -0.415
## 10 11746  ESB    E S B INC                      0.0597   -0.352    3.14     3.27
```

4 Sub-Period Analysis

We analyze the stability of risk premia across three distinct regimes: Pre-Crisis (1995-2007), Post-Crisis (2008-2019), and Pandemic/Recent (2020-2024).

```
# function to run FMB for one subperiod (re-do Phase B/C/D + pricing errors)
run_subperiod_fmb <- function(data_for_betas, start_date, end_date, label,
                              min_months_beta = 24, min_cs_n = 10) {

  df_sub <- data_for_betas %>%
    filter(date >= as.Date(start_date), date <= as.Date(end_date)) %>%
    drop_na(excess_ret, mkt_excess, smb, hml)

  # -------------------------
  # Step 1 (Phase B): betas
  # -------------------------
  betas_sub <- df_sub %>%
    group_by(ticker) %>%
    filter(n() >= min_months_beta) %>%
    summarise(model = list(lm(excess_ret ~ mkt_excess + smb + hml)), .groups = "drop") %>%
    mutate(coefs = map(model, tidy)) %>%
    unnest(coefs) %>%
    select(ticker, term, estimate) %>%
    pivot_wider(names_from = term, values_from = estimate) %>%
    rename(alpha = `(Intercept)`, beta_mkt = mkt_excess, beta_smb = smb, beta_hml = hml)

  # -------------------------
  # Step 2 (Phase C): lambdas
  # -------------------------
  fmb_data_sub <- df_sub %>%
    inner_join(betas_sub, by = "ticker")

  fmb_lambdas_sub <- fmb_data_sub %>%
    group_by(date) %>%
    filter(n() > min_cs_n) %>%
    summarise(model = list(lm(excess_ret ~ beta_mkt + beta_smb + beta_hml)), .groups = "drop") %>%
    mutate(coefs = map(model, tidy)) %>%
    unnest(coefs) %>%
    select(date, term, estimate) %>%
    pivot_wider(names_from = term, values_from = estimate) %>%
    rename(lambda_0 = `(Intercept)`, lambda_mkt = beta_mkt, lambda_smb = beta_smb, lambda_hml = beta_hml
    arrange(date)

  # -------------------------
  # Step 3: expected premia + variances + t-stats
  # -------------------------
  lambda_stats_sub <- fmb_lambdas_sub %>%
    summarise(
      period = label,
      start = as.Date(start_date),
      end   = as.Date(end_date),
```

```r
      T = n(),

      mean_lambda_0   = mean(lambda_0, na.rm = TRUE),
      mean_lambda_mkt = mean(lambda_mkt, na.rm = TRUE),
      mean_lambda_smb = mean(lambda_smb, na.rm = TRUE),
      mean_lambda_hml = mean(lambda_hml, na.rm = TRUE),

      var_lambda_0   = var(lambda_0, na.rm = TRUE),
      var_lambda_mkt = var(lambda_mkt, na.rm = TRUE),
      var_lambda_smb = var(lambda_smb, na.rm = TRUE),
      var_lambda_hml = var(lambda_hml, na.rm = TRUE),

      # FM-style t-stats (mean / (sd/sqrt(T)))
      t_lambda_0   = mean(lambda_0, na.rm = TRUE) / (sd(lambda_0, na.rm = TRUE) / sqrt(n())),
      t_lambda_mkt = mean(lambda_mkt, na.rm = TRUE) / (sd(lambda_mkt, na.rm = TRUE) / sqrt(n())),
      t_lambda_smb = mean(lambda_smb, na.rm = TRUE) / (sd(lambda_smb, na.rm = TRUE) / sqrt(n())),
      t_lambda_hml = mean(lambda_hml, na.rm = TRUE) / (sd(lambda_hml, na.rm = TRUE) / sqrt(n()))
    )

  # -------------------------
  # Pricing errors
  # alpha_it = r_it - lambda_t' beta_i  (since alpha0t + u_it = r_it - beta'lambda_t)
  # -------------------------
  pricing_errors_sub <- fmb_data_sub %>%
    inner_join(fmb_lambdas_sub, by = "date") %>%
    mutate(
      alpha_it = excess_ret - (beta_mkt * lambda_mkt + beta_smb * lambda_smb + beta_hml * lambda_hml)
    ) %>%
    group_by(ticker) %>%
    summarise(
      period = label,
      start = as.Date(start_date),
      end   = as.Date(end_date),
      mean_pricing_error = mean(alpha_it, na.rm = TRUE),
      var_pricing_error  = var(alpha_it, na.rm = TRUE),
      t_pricing_error    = mean(alpha_it, na.rm = TRUE) / (sd(alpha_it, na.rm = TRUE) / sqrt(sum(!is.na
      n = sum(!is.na(alpha_it)),
      .groups = "drop"
    ) %>%
    arrange(desc(abs(mean_pricing_error)))

  list(
    betas = betas_sub,
    lambdas = fmb_lambdas_sub,
    lambda_stats = lambda_stats_sub,
    pricing_errors = pricing_errors_sub
  )
}

# Run the subperiods
res_1995_2007 <- run_subperiod_fmb(data_for_betas, "1995-01-01", "2007-12-31", "1995-2007")
res_2008_2019 <- run_subperiod_fmb(data_for_betas, "2008-01-01", "2019-12-31", "2008-2019")
res_2020_2024 <- run_subperiod_fmb(data_for_betas, "2020-01-01", "2024-12-31", "2020-2024")
```

```r
# final tables
lambda_stats_E <- bind_rows(res_1995_2007$lambda_stats, res_2008_2019$lambda_stats,res_2020_2024$lambda_

pricing_errors_E <- bind_rows(res_1995_2007$pricing_errors, res_2008_2019$pricing_errors, res_2020_2024

print(lambda_stats_E)
```

```
## # A tibble: 3 x 16
##   period    start      end            T mean_lambda_0 mean_lambda_mkt
##   <chr>     <date>     <date>     <int>         <dbl>           <dbl>
## 1 1995-2007 1995-01-01 2007-12-31   156        0.0106          0.00533
## 2 2008-2019 2008-01-01 2019-12-31   144        0.0105          0.00191
## 3 2020-2024 2020-01-01 2024-12-31    60       -0.00175         0.0155
## # i 10 more variables: mean_lambda_smb <dbl>, mean_lambda_hml <dbl>,
## #   var_lambda_0 <dbl>, var_lambda_mkt <dbl>, var_lambda_smb <dbl>,
## #   var_lambda_hml <dbl>, t_lambda_0 <dbl>, t_lambda_mkt <dbl>,
## #   t_lambda_smb <dbl>, t_lambda_hml <dbl>
```

```r
print(head(pricing_errors_E, 20))
```

```
## # A tibble: 20 x 8
##    ticker period    start      end        mean_pricing_error var_pricing_error
##    <chr>  <chr>     <date>     <date>                  <dbl>             <dbl>
##  1 86990  1995-2007 1995-01-01 2007-12-31             0.0770           0.0661
##  2 85160  1995-2007 1995-01-01 2007-12-31            -0.0698           0.0435
##  3 86881  1995-2007 1995-01-01 2007-12-31             0.0671           0.0368
##  4 82800  1995-2007 1995-01-01 2007-12-31             0.0593           0.00748
##  5 80515  1995-2007 1995-01-01 2007-12-31             0.0566           0.0243
##  6 79179  1995-2007 1995-01-01 2007-12-31             0.0533           0.0130
##  7 68161  1995-2007 1995-01-01 2007-12-31             0.0525           0.0202
##  8 11552  1995-2007 1995-01-01 2007-12-31             0.0523           0.0103
##  9 86580  1995-2007 1995-01-01 2007-12-31             0.0516           0.0224
## 10 81776  1995-2007 1995-01-01 2007-12-31             0.0480           0.0124
## 11 90319  1995-2007 1995-01-01 2007-12-31             0.0458           0.0119
## 12 80266  1995-2007 1995-01-01 2007-12-31             0.0453           0.0377
## 13 64856  1995-2007 1995-01-01 2007-12-31             0.0452           0.0171
## 14 76779  1995-2007 1995-01-01 2007-12-31             0.0425           0.0155
## 15 63830  1995-2007 1995-01-01 2007-12-31             0.0419           0.00409
## 16 78788  1995-2007 1995-01-01 2007-12-31             0.0419           0.0159
## 17 86414  1995-2007 1995-01-01 2007-12-31             0.0416           0.0243
## 18 80127  1995-2007 1995-01-01 2007-12-31             0.0414           0.00775
## 19 90386  1995-2007 1995-01-01 2007-12-31             0.0413           0.0138
## 20 84597  1995-2007 1995-01-01 2007-12-31             0.0406           0.0183
## # i 2 more variables: t_pricing_error <dbl>, n <int>
```

```r
tstars <- function(t) {
  case_when(
    is.na(t)            ~ NA_character_,
    abs(t) >= 2.576     ~ "***", # 1%
    abs(t) >= 1.960     ~ "**", # 5%
    abs(t) >= 1.645     ~ "*", # 10%
    TRUE                ~ ""
  )
```

```r
}

# Name Map
name_map <- stock_returns %>%
  group_by(ticker) %>%
  arrange(desc(date)) %>%
  slice(1) %>%
  ungroup() %>%
  select(ticker, symbol, company)

# 2. Join Names to the Sub-Period Errors
pricing_errors_E_readable <- pricing_errors_E %>%
  left_join(name_map, by = "ticker") %>%
  select(period, ticker, symbol, company, mean_pricing_error, t_pricing_error) %>%
  arrange(period, desc(abs(mean_pricing_error)))

pricing_errors_E_readable <- pricing_errors_E_readable %>%
  mutate(significance = tstars(t_pricing_error)) %>%
  relocate(significance, .after = t_pricing_error)

# 3. Top Mispriced Stocks for Each Period
print("--- Top Mispriced: 1995-2007 (Pre-Crisis) ---")
```

```
## [1] "--- Top Mispriced: 1995-2007 (Pre-Crisis) ---"
```

```r
print(head(filter(pricing_errors_E_readable, period == "1995-2007"), 10))
```

```
## # A tibble: 10 x 7
##    period  ticker symbol company mean_pricing_error t_pricing_error significance
##    <chr>   <chr>  <chr>  <chr>                <dbl>           <dbl> <chr>
##  1 1995-2~ 86990  OPWV   OPENWA~             0.0770            1.47 ""
##  2 1995-2~ 85160  ATHM   AT HOM~            -0.0698           -1.74 "*"
##  3 1995-2~ 86881  BRCD   BROCAD~             0.0671            2.13 "**"
##  4 1995-2~ 82800  SCCO   SOUTHE~             0.0593            3.76 "***"
##  5 1995-2~ 80515  RATL   RATION~             0.0566            1.81 "*"
##  6 1995-2~ 79179  MFST   M F S ~             0.0533            2.30 "**"
##  7 1995-2~ 68161  SCI    S C I ~             0.0525            1.81 "*"
##  8 1995-2~ 11552  CELG   CELGEN~             0.0523            3.72 "***"
##  9 1995-2~ 86580  NVDA   NVIDIA~             0.0516            2.60 "***"
## 10 1995-2~ 81776  SUNE   SUNEDI~             0.0480            2.24 "**"
```

```r
print("--- Top Mispriced: 2008-2019 (Post-Crisis) ---")
```

```
## [1] "--- Top Mispriced: 2008-2019 (Post-Crisis) ---"
```

```r
print(head(filter(pricing_errors_E_readable, period == "2008-2019"), 10))
```

```
## # A tibble: 10 x 7
##    period  ticker symbol company mean_pricing_error t_pricing_error significance
##    <chr>   <chr>  <chr>  <chr>                <dbl>           <dbl> <chr>
##  1 2008-2~ 81776  SUNE   SUNEDI~            -0.0597           -2.32 **
##  2 2008-2~ 82513  PCYC   PHARMA~             0.0503            2.02 **
##  3 2008-2~ 90664  DXCM   DEXCOM~             0.0481            1.93 *
##  4 2008-2~ 76282  AN     AUTONA~             0.0460            2.28 **
##  5 2008-2~ 89393  NFLX   NETFLI~             0.0442            3.27 ***
##  6 2008-2~ 61241  AMD    ADVANC~             0.0421            2.44 **
```

```
##   7 2008-2~ 79588  GMCR   KEURIG~              0.0411            1.88 *
##   8 2008-2~ 80320  CPRT   COPART~              0.0389            3.39 ***
##   9 2008-2~ 93436  TSLA   TESLA ~              0.0388            2.19 **
##  10 2008-2~ 93132  FTNT   FORTIN~              0.0379            2.22 **
```

```r
print("--- Top Mispriced: 2020-2024 (Pandemic & After) ---")
```

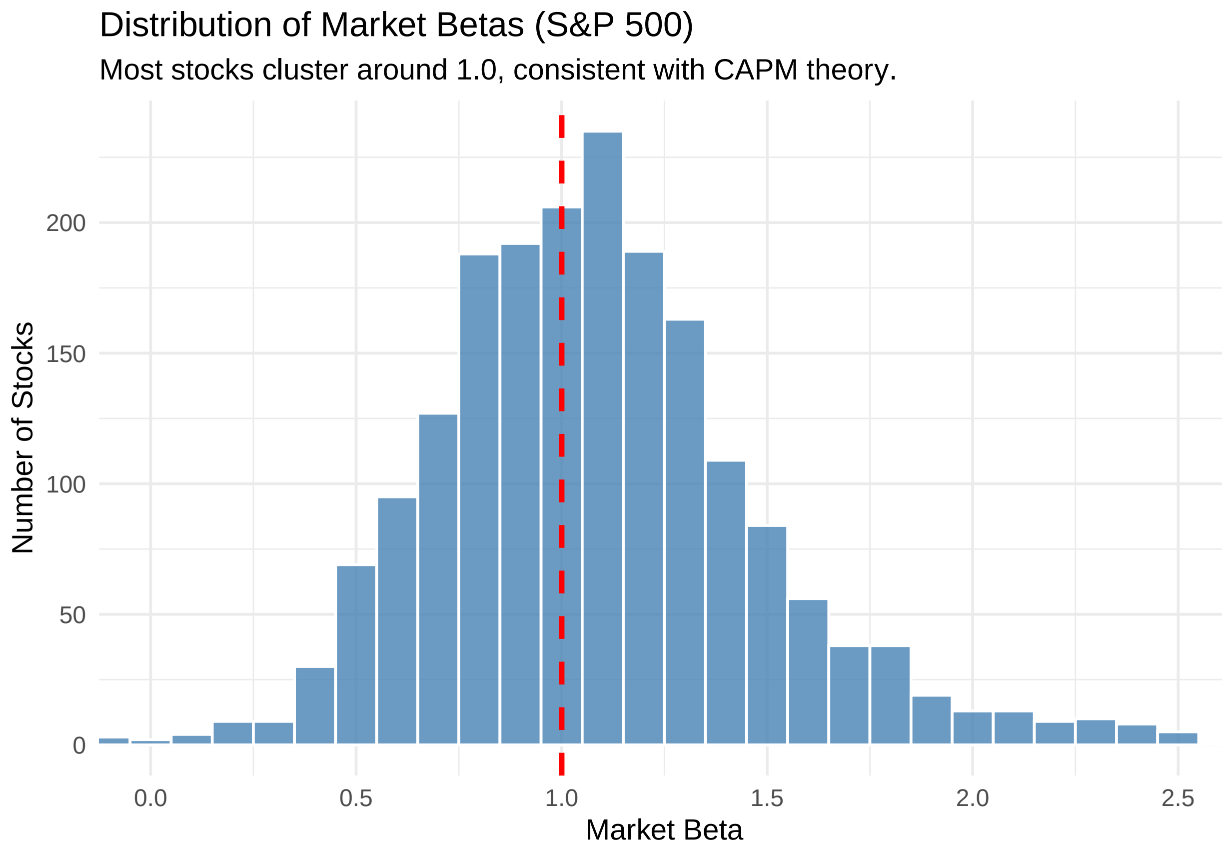
```
## [1] "--- Top Mispriced: 2020-2024 (Pandemic & After) ---"
```

```r
print(head(filter(pricing_errors_E_readable, period == "2020-2024"),10))
```

```
## # A tibble: 10 x 7
##    period  ticker symbol company mean_pricing_error t_pricing_error significance
##    <chr>   <chr>  <chr>  <chr>                <dbl>           <dbl> <chr>
##  1 2020-2~ 20894  APP    APPLOV~              0.0975           3.22 "***"
##  2 2020-2~ 16736  VST    VISTRA~              0.0473           2.22 "**"
##  3 2020-2~ 19788  PLTR   PALANT~              0.0434           1.30 ""
##  4 2020-2~ 43553  VFC    V F CO~             -0.0364          -3.36 "***"
##  5 2020-2~ 22976  WBD    WARNER~             -0.0360          -1.83 "*"
##  6 2020-2~ 20391  TPL    TEXAS ~              0.0354           1.39 ""
##  7 2020-2~ 22623  CEG    CONSTE~              0.0353           1.87 "*"
##  8 2020-2~ 15850  MTCH   MATCH ~             -0.0343          -2.28 "**"
##  9 2020-2~ 82486  ERIE   ERIE I~              0.0338           2.11 "**"
## 10 2020-2~ 10777  FCNCA  FIRST ~              0.0334           1.87 "*"
```

```r
# ============================================================================
# VISUALIZATION: BETA DISTRIBUTION
# ============================================================================
library(ggplot2)

ggplot(stock_betas, aes(x = beta_mkt)) +
  # Histogram with a nice fill color
  geom_histogram(binwidth = 0.1, fill = "steelblue", color = "white", alpha = 0.8) +
  # Vertical line at Beta = 1 (Market Average)
  geom_vline(aes(xintercept = 1), color = "red", linetype = "dashed", linewidth = 1) +
  # Labels
  labs(
    title = "Distribution of Market Betas (S&P 500)",
    subtitle = "Most stocks cluster around 1.0, consistent with CAPM theory.",
    x = "Market Beta",
    y = "Number of Stocks"
  ) +
  theme_minimal() +
  # Remove extreme outliers for a cleaner chart
  coord_cartesian(xlim = c(0, 2.5))
```



5 Economic Significance

To test the economic validity of the model, we implement a **Real-Time Trading Strategy** free of look-ahead bias.

1. **Rolling Betas** ($\beta_{i,t}$): Estimated using a 60-month trailing window (past information only).
2. **Expanding Mean Lambdas** ($\bar{\lambda}_t$): Average of realized premia from the start of the sample up to time t.
3. **Signal:** We forecast expected excess returns:

$$\hat{E}_t[R^e_{i,t+1}] = \hat{\beta}_{i,t}^{\top}\bar{\lambda}_t$$

We sort stocks into Quintiles based on this signal and go **Long Q5 (High Exp. Return)** and **Short Q1 (Low Exp. Return)**.

```r
beta_window   <- 60   # rolling window length in months (e.g., 60)
min_cs_n      <- 100  # minimum #stocks per month for cross-sectional regressions / sorts
min_beta_obs  <- 48   # require at least this many complete observations inside the beta window

# -------------------------
# data_for_betas includes: date, ticker, excess_ret, mkt_excess, smb, hml, mktcap
# -------------------------
df <- data_for_betas %>%
  mutate(date = floor_date(as.Date(date), "month")) %>%
  select(date, ticker, excess_ret, mkt_excess, smb, hml, mktcap) %>%
  arrange(ticker, date) %>%
  filter(is.finite(excess_ret), is.finite(mkt_excess), is.finite(smb), is.finite(hml), is.finite(mktcap)
```

```r
# -------------------------
# Step 1: Rolling betas per stock (ending at each month t)
# Using only information up to t
# -------------------------
rolling_betas_one_ticker <- function(d, window = 60, min_obs = 48) {
  d <- d %>% arrange(date)

  fit_window <- function(win) {
    # win is a tibble slice of length "window"
    win <- win %>% filter(is.finite(excess_ret), is.finite(mkt_excess), is.finite(smb), is.finite(hml))

    # If we don't have enough valid data points in this specific window, return NA
    if (nrow(win) < min_obs) {
      return(c(beta_mkt = NA_real_, beta_smb = NA_real_, beta_hml = NA_real_))
    }

    y <- win$excess_ret
    X <- cbind(1, win$mkt_excess, win$smb, win$hml)

    # Check if X is singular or valid before fitting to prevent crashes
    if (nrow(X) < 4) return(c(beta_mkt = NA_real_, beta_smb = NA_real_, beta_hml = NA_real_))

    fit <- lm.fit(x = X, y = y)
    b <- fit$coefficients
    c(beta_mkt = b[2], beta_smb = b[3], beta_hml = b[4])
  }

  # slide over rows, using a trailing window that ends at each t
  betas_mat <- slide(
    .x = seq_len(nrow(d)),
    .f = ~ {
      idx_end <- .x
      idx_start <- max(1, idx_end - window + 1)
      win <- d[idx_start:idx_end, ]
      fit_window(win)
    },
    .complete = FALSE
  )

  betas_df <- bind_rows(lapply(betas_mat, as_tibble_row))

  bind_cols(
    d %>% select(date),
    betas_df
  )
}

betas_rolling <- df %>%
  group_by(ticker) %>%
  filter(n() >= min_beta_obs) %>%
  group_modify(~ rolling_betas_one_ticker(.x, window = beta_window, min_obs = min_beta_obs)) %>%
  ungroup()
```

```r
# Join rolling betas back to main panel
df_b <- df %>%
  left_join(betas_rolling, by = c("date", "ticker")) %>%
  arrange(ticker, date)

df_b_clean <- df_b %>%
  select(
    date, ticker, excess_ret, mktcap,
    mkt_excess,
    beta_mkt = matches("^beta_mkt\\..+"),
    beta_smb = matches("^beta_smb\\..+"),
    beta_hml = matches("^beta_hml\\..+")
  ) %>%
  filter(is.finite(beta_mkt)) %>%
  mutate(across(c(excess_ret, beta_mkt, beta_smb, beta_hml), as.numeric))

print(head(df_b_clean))
```

```
## # A tibble: 6 x 8
##   date       ticker excess_ret mktcap mkt_excess beta_mkt beta_smb beta_hml
##   <date>     <chr>       <dbl>  <dbl>      <dbl>    <dbl>    <dbl>    <dbl>
## 1 1965-12-01 10006    0.0736   279720     0.0101    1.02    0.174       1.28
## 2 1966-01-01 10006    0.0861   304880     0.0072    1.01    0.221       1.32
## 3 1966-02-01 10006   -0.0239   296000    -0.0121    1.04    0.134       1.35
## 4 1966-03-01 10006   -0.0513   281940    -0.0251    1.05    0.130       1.36
## 5 1966-04-01 10006   -0.000775 282680     0.0213    1.05    0.0854      1.38
## 6 1966-05-01 10006    0.00218  281200    -0.0567    0.997  -0.0423      1.25
```

```r
# -------------------------
# Step 2: Monthly lambdas
# -------------------------
df_lambda_input <- df_b_clean %>%
  group_by(ticker) %>%
  arrange(date) %>%
  mutate(
    beta_mkt_lag = lag(beta_mkt),
    beta_smb_lag = lag(beta_smb),
    beta_hml_lag = lag(beta_hml)
  ) %>%
  ungroup()

safe_fmb_reg <- function(d) {
  d_clean <- d %>%
    filter(
      is.finite(excess_ret),
      is.finite(beta_mkt_lag),
      is.finite(beta_smb_lag),
      is.finite(beta_hml_lag)
    )

  if (nrow(d_clean) < 50) return(NULL)

  tryCatch({
    lm_mod <- lm(excess_ret ~ beta_mkt_lag + beta_smb_lag + beta_hml_lag, data = d_clean)
```

```r
    tidy(lm_mod)
  }, error = function(e) NULL)
}

message("Estimating monthly lambdas (Step 2)...")

fmb_lambdas_nla <- df_lambda_input %>%
  group_by(date) %>%
  summarise(coefs = list(safe_fmb_reg(pick(everything()))), .groups = "drop") %>%
  filter(!sapply(coefs, is.null)) %>%
  unnest(coefs) %>%
  select(date, term, estimate) %>%
  pivot_wider(names_from = term, values_from = estimate) %>%
  rename(
    lambda_0   = `(Intercept)`,
    lambda_mkt = beta_mkt_lag,
    lambda_smb = beta_smb_lag,
    lambda_hml = beta_hml_lag
  ) %>%
  arrange(date)

print(head(fmb_lambdas_nla))
```

```
## # A tibble: 6 x 5
##   date       lambda_0 lambda_mkt lambda_smb lambda_hml
##   <date>        <dbl>      <dbl>      <dbl>      <dbl>
## 1 1964-01-01  0.0131    0.000407   -0.0169     0.00914
## 2 1964-02-01  0.00726   0.0113      0.00485    0.0117
## 3 1964-03-01 -0.0161    0.0358      0.00378    0.00986
## 4 1964-04-01  0.0355   -0.0368     -0.0106     0.00474
## 5 1964-05-01  0.0230   -0.0138     -0.00371    0.0153
## 6 1964-06-01  0.0247   -0.0101      0.00149    0.00111
```

```r
# -------------------------
# Step 3: Expanding Mean Lambdas (The Prediction Model)
# We average the lambdas up to time t to predict t+1
# -------------------------
lambdas_expanding <- fmb_lambdas_nla %>%
  arrange(date) %>%
  mutate(
    mean_lambda_mkt = cummean(lambda_mkt),
    mean_lambda_smb = cummean(lambda_smb),
    mean_lambda_hml = cummean(lambda_hml)
  ) %>%
  select(date, mean_lambda_mkt, mean_lambda_smb, mean_lambda_hml)

# -------------------------
# Step 4: Build the Trading Signal
# Signal = Beta(t) * Mean_Lambda(t)
# -------------------------
strategy_panel_nla <- df_b_clean %>%
  left_join(lambdas_expanding, by = "date") %>%
  arrange(ticker, date) %>%
  group_by(ticker) %>%
```

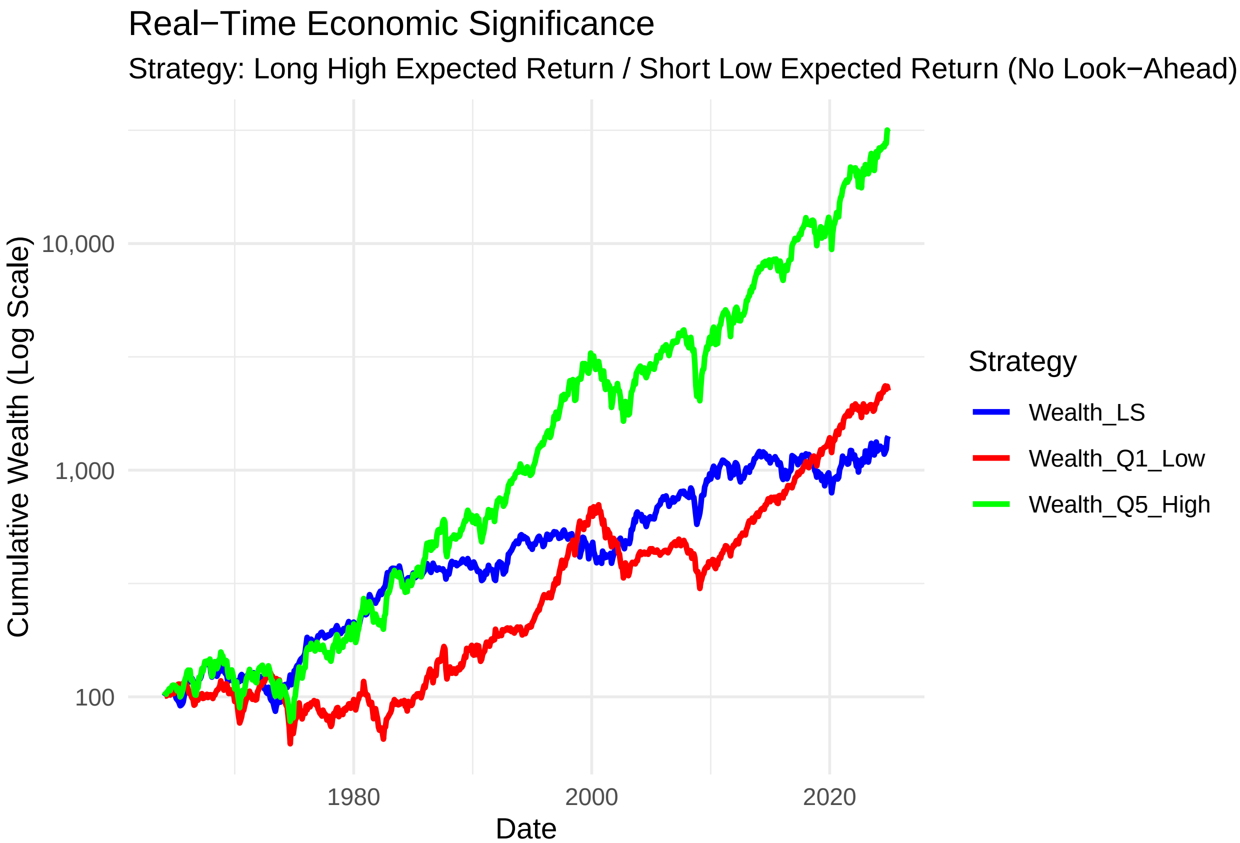
```
  mutate(
    next_excess_ret = lead(excess_ret), # We want to predict NEXT month
    w_next = mktcap # Value-weighting
  ) %>%
  ungroup() %>%
  # Filter for valid signals
  filter(
    is.finite(beta_mkt), is.finite(mean_lambda_mkt),
    is.finite(next_excess_ret)
  ) %>%
  mutate(
    exp_excess_hat = beta_mkt * mean_lambda_mkt +
      beta_smb * mean_lambda_smb +
      beta_hml * mean_lambda_hml
  )

# -------------------------
# Step 5: Form Portfolios and Calculate Returns
# -------------------------
portfolio_rets_nla <- strategy_panel_nla %>%
  group_by(date) %>%
  filter(n() >= 100) %>%
  mutate(q = ntile(exp_excess_hat, 5)) %>%
  summarise(
    vw_ret_q1 = weighted.mean(next_excess_ret[q == 1], w_next[q == 1], na.rm = TRUE),
    vw_ret_q5 = weighted.mean(next_excess_ret[q == 5], w_next[q == 5], na.rm = TRUE),
    mkt_excess = mean(mkt_excess, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  mutate(
    ret_date = date %m+% months(1),
    ls_ret = vw_ret_q5 - vw_ret_q1
  ) %>%
  arrange(ret_date)

# -------------------------
# Step 6: Visualization (Cumulative Wealth)
# -------------------------
wealth_plot_data <- portfolio_rets_nla %>%
  mutate(
    Wealth_Q5_High = 100 * cumprod(1 + vw_ret_q5),
    Wealth_Q1_Low  = 100 * cumprod(1 + vw_ret_q1),
    Wealth_LS      = 100 * cumprod(1 + ls_ret)
  ) %>%
  select(ret_date, Wealth_Q5_High, Wealth_Q1_Low, Wealth_LS) %>%
  pivot_longer(-ret_date, names_to = "Strategy", values_to = "Wealth")

ggplot(wealth_plot_data, aes(x = ret_date, y = Wealth, color = Strategy)) +
  geom_line(linewidth = 1) +
  scale_y_log10(labels = comma) +
  scale_color_manual(values = c("Wealth_Q5_High" = "green", "Wealth_Q1_Low" = "red", "Wealth_LS" = "blu
  labs(
    title = "Real-Time Economic Significance",
```

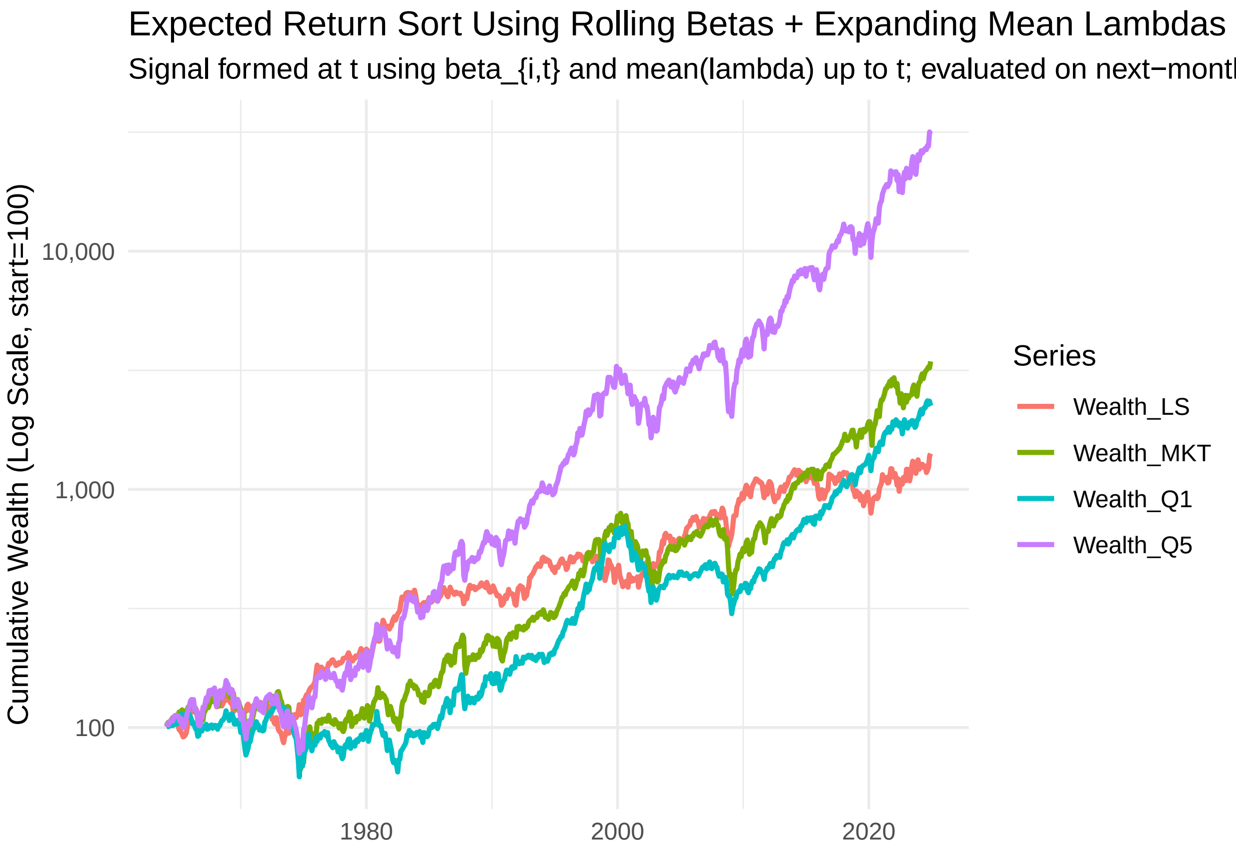
```r
    subtitle = "Strategy: Long High Expected Return / Short Low Expected Return (No Look-Ahead)",
    y = "Cumulative Wealth (Log Scale)",
    x = "Date"
  ) +
  theme_minimal()
```




```r
# -------------------------
# Step 7: Cumulative wealth plot
# -------------------------
wealth_df_nla <- portfolio_rets_nla %>%
  transmute(
    date = ret_date,
    Wealth_Q1  = 100 * cumprod(1 + vw_ret_q1),
    Wealth_Q5  = 100 * cumprod(1 + vw_ret_q5),
    Wealth_LS  = 100 * cumprod(1 + ls_ret),
    Wealth_MKT = 100 * cumprod(1 + mkt_excess)
  ) %>%
  pivot_longer(-date, names_to = "Series", values_to = "Wealth")

ggplot(wealth_df_nla, aes(x = date, y = Wealth, color = Series)) +
  geom_line(linewidth = 0.9) +
  scale_y_log10(labels = comma) +
  labs(
    title = "Expected Return Sort Using Rolling Betas + Expanding Mean Lambdas",
    subtitle = "Signal formed at t using beta_{i,t} and mean(lambda) up to t; evaluated on next-month e
    x = NULL,
    y = "Cumulative Wealth (Log Scale, start=100)"
```

```r
  ) +
  theme_minimal()
```




```r
# -------------------------
cs_check_nla <- strategy_panel_nla %>%
  group_by(ticker) %>%
  summarise(
    mean_next_excess = mean(next_excess_ret, na.rm = TRUE),
    mean_signal = mean(exp_excess_hat, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  summarise(correlation = cor(mean_signal, mean_next_excess, use = "complete.obs"))

print(paste("Correlation(mean signal, mean next-month excess return across stocks):",
            round(cs_check_nla$correlation, 4)))
```

```
## [1] "Correlation(mean signal, mean next-month excess return across stocks): 0.1742"
```

```r
# ============================================================================
# Step 9: Performance Statistics Table
# ============================================================================
# Calculate stats for Q1, Q5, Long-Short, and Market
perf_summary <- portfolio_rets_nla %>%
  summarise(
    # Annualized Mean Return (Monthly Mean * 12)
    Mean_Q1  = mean(vw_ret_q1, na.rm=TRUE) * 12,
    Mean_Q5  = mean(vw_ret_q5, na.rm=TRUE) * 12,
    Mean_LS  = mean(ls_ret, na.rm=TRUE) * 12,
```

```r
    Mean_Mkt = mean(mkt_excess, na.rm=TRUE) * 12,

    # Annualized Volatility (Monthly SD * sqrt(12))
    Vol_Q1   = sd(vw_ret_q1, na.rm=TRUE) * sqrt(12),
    Vol_Q5   = sd(vw_ret_q5, na.rm=TRUE) * sqrt(12),
    Vol_LS   = sd(ls_ret, na.rm=TRUE) * sqrt(12),
    Vol_Mkt  = sd(mkt_excess, na.rm=TRUE) * sqrt(12)
  ) %>%
  mutate(
    # Sharpe Ratio = Mean / Volatility
    Sharpe_Q1  = Mean_Q1 / Vol_Q1,
    Sharpe_Q5  = Mean_Q5 / Vol_Q5,
    Sharpe_LS  = Mean_LS / Vol_LS,
    Sharpe_Mkt = Mean_Mkt / Vol_Mkt
  ) %>%
  pivot_longer(everything(), names_to = "Metric", values_to = "Value") %>%
  separate(Metric, into = c("Stat", "Portfolio"), sep = "_") %>%
  pivot_wider(names_from = Stat, values_from = Value)

print("--- Final Strategy Performance ---")
```

```
## [1] "--- Final Strategy Performance ---"
```

```r
print(perf_summary)
```

```
## # A tibble: 4 x 4
##   Portfolio   Mean   Vol Sharpe
##   <chr>      <dbl> <dbl>  <dbl>
## 1 Q1        0.0615 0.143  0.429
## 2 Q5        0.114  0.198  0.577
## 3 LS        0.0528 0.136  0.387
## 4 Mkt       0.0705 0.156  0.453
```

6 Outputs

```r
# Full Sample Risk Premia
table1_data <- final_stats %>%
  mutate(
    Type = if_else(str_starts(stat, "mean"), "Estimate", "T_Stat"),
    Factor = str_remove(stat, "(mean|t)_lambda_") %>% toupper()
  ) %>%
  select(-stat) %>%
  pivot_wider(names_from = Type, values_from = value) %>%
  mutate(
    Estimate = round(Estimate, 4),
    T_Stat = round(T_Stat, 2),
    Significance = case_when(
      abs(T_Stat) >= 2.58 ~ "***",
      abs(T_Stat) >= 1.96 ~ "**",
      abs(T_Stat) >= 1.65 ~ "*",
      TRUE ~ ""
    )
  ) %>%
```

```r
  select(Factor, Coefficient = Estimate, `t-statistic` = T_Stat, Significance)

kable(table1_data, caption = "Table 1: Full Sample Fama-MacBeth Risk Premia Estimates")
```

Table 1: Table 1: Full Sample Fama-MacBeth Risk Premia Estimates

Factor	Coefficient	t-statistic	Significance
0	0.0065	5.69	***
MKT	0.0050	2.63	***
SMB	0.0038	2.94	***
HML	-0.0043	-3.60	***

```r
# Top Mispriced Stocks
# Top 10 stocks by absolute pricing error (Alpha)
table2_data <- pricing_errors_full %>%
  head(10) %>%
  select(Ticker = ticker, Symbol = symbol, Company = company,
         Alpha = alpha_i, Beta_MKT = beta_mkt, Beta_SMB = beta_smb, Beta_HML = beta_hml) %>%
  mutate(across(where(is.numeric), ~ round(., 3)))

kable(table2_data, caption = "Table 2: Top 10 Mispriced Stocks (Highest Absolute Pricing Errors)")
```

Table 2: Table 2: Top 10 Mispriced Stocks (Highest Absolute Pricing Errors)

Ticker	Symbol	Company	Alpha	Beta_MKT	Beta_SMB	Beta_HML
89301	GME	GAMESTOP CORP NEW	0.275	-4.478	49.569	4.555
87092	TIBX	T I B C O SOFTWARE INC	0.124	1.840	5.028	-0.387
82200	NSCP	NETSCAPE COMMUNICATIONS CORP	0.095	1.126	-1.167	-5.729
80266	QLGC	QLOGIC CORP	0.079	2.174	0.846	-2.837
27182	NA	TRANSITRON ELECTRONIC CORP	-0.068	1.234	1.700	-1.859
14983	W	WAYFAIR INC	0.065	3.186	1.869	-0.934
85522	AMCC	APPLIED MICRO CIRCUITS CORP	0.064	4.304	1.277	0.227
61524	CDO	COMDISCO INC	0.062	3.408	1.015	0.767
20894	APP	APPLOVIN CORP	0.062	3.572	1.841	-0.415
11746	ESB	E S B INC	0.060	-0.352	3.138	3.275

```r
# Sub-Period Stability Analysis
# Comparison of risk premia across regimes
table3_data <- lambda_stats_E %>%
  select(Period = period,
         MKT_Premia = mean_lambda_mkt, MKT_t = t_lambda_mkt,
         SMB_Premia = mean_lambda_smb, SMB_t = t_lambda_smb,
         HML_Premia = mean_lambda_hml, HML_t = t_lambda_hml) %>%
  mutate(across(where(is.numeric), ~ round(., 4)))

kable(table3_data, caption = "Table 3: Stability of Risk Premia Across Sub-Periods")
```

Table 3: Table 3: Stability of Risk Premia Across Sub-Periods

Period	MKT_Premia	MKT_t	SMB_Premia	SMB_t	HML_Premia	HML_t
1995-2007	0.0053	1.3801	0.0066	1.8654	-0.0050	-1.7472
2008-2019	0.0019	0.4274	0.0026	1.1140	-0.0030	-1.1255
2020-2024	0.0155	1.9885	0.0020	0.4332	-0.0055	-0.8392

```r
# Economic Significance - Strategy Performance
# Sharpe Ratios and Annualized Returns
table4_data <- perf_summary %>%
  mutate(
    Mean_Return = scales::percent(Mean, accuracy = 0.01),
    Volatility = scales::percent(Vol, accuracy = 0.01),
    Sharpe_Ratio = round(Sharpe, 3)
  ) %>%
  select(Portfolio, Mean_Return, Volatility, Sharpe_Ratio)

kable(table4_data, caption = "Table 4: Out-of-Sample Trading Strategy Performance")
```

Table 4: Table 4: Out-of-Sample Trading Strategy Performance

Portfolio	Mean_Return	Volatility	Sharpe_Ratio
Q1	6.15%	14.34%	0.429
Q5	11.43%	19.81%	0.577
LS	5.28%	13.64%	0.387
Mkt	7.05%	15.56%	0.453

```r
# Strategy Alpha Check
# Regression of Strategy Returns on Factors
strategy_data_for_reg <- portfolio_rets_nla %>%
  rename(mkt_excess_port = mkt_excess) %>%
  left_join(ff_factors, by = c("ret_date" = "date")) %>%
  drop_na(ls_ret, mkt_excess, smb, hml)

alpha_check_model <- lm(ls_ret ~ mkt_excess + smb + hml, data = strategy_data_for_reg)

table5_data <- tidy(alpha_check_model) %>%
  mutate(
    term = case_when(
      term == "(Intercept)" ~ "Alpha",
      term == "mkt_excess" ~ "MKT Exposure",
      term == "smb" ~ "SMB Exposure",
      term == "hml" ~ "HML Exposure",
      TRUE ~ term
    ),
    estimate = round(estimate, 4),
    statistic = round(statistic, 2),
    p.value = round(p.value, 3),
    Signif = if_else(p.value < 0.05, "*", "")
  ) %>%
  select(Term = term, Estimate = estimate, `t-stat` = statistic, `p-value` = p.value, Signif)
```

Table 6: Table 1: Full Sample Fama-MacBeth Risk Premia Estimates

	Estimates		
Factor	Coefficient	t-statistic	Significance
0	0.0065	5.69	***
MKT	0.0050	2.63	***
SMB	0.0038	2.94	***
HML	-0.0043	-3.60	***

```
kable(table5_data, caption = "Table 5: Alpha Check (Regression of Strategy Returns on Factors)")
```

Table 5: Table 5: Alpha Check (Regression of Strategy Returns on Factors)

Term	Estimate	t-stat	p-value	Signif
Alpha	0.0007	0.57	0.566	
MKT Exposure	0.3009	11.23	0.000	*
SMB Exposure	0.5584	14.28	0.000	*
HML Exposure	0.3785	9.72	0.000	*

```
# Load necessary libraries for visualization
if (!require(kableExtra)) install.packages("kableExtra")
if (!require(DT)) install.packages("DT")
library(kableExtra)
library(DT)
library(dplyr)

# Full Sample Risk Premia
table1_styled <- table1_data %>%
  kbl(caption = "Table 1: Full Sample Fama-MacBeth Risk Premia Estimates") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"), full_width = F) %>%
  column_spec(1, bold = TRUE) %>%
  add_header_above(c(" " = 1, "Estimates" = 3))

table1_styled
```

```
# Top Mispriced Stocks
table2_styled <- table2_data %>%
  kbl(caption = "Table 2: Top 10 Mispriced Stocks (Highest Absolute Pricing Errors)") %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F) %>%
  column_spec(1, bold = TRUE) %>%
  row_spec(0, bold = TRUE, color = "white", background = "#2c3e50")


table2_styled
```

```
# Sub-Period Stability Analysis
table3_styled <- table3_data %>%
  kbl(caption = "Table 3: Stability of Risk Premia Across Sub-Periods") %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F) %>%
  add_header_above(c(" " = 1, "Market Factor" = 2, "Size Factor (SMB)" = 2, "Value Factor (HML)" = 2))
```

Table 7: Table 2: Top 10 Mispriced Stocks (Highest Absolute Pricing Errors)

Ticker	Symbol	Company	Alpha	Beta_MKT	Beta_SMB	Beta_HML
89301	GME	GAMESTOP CORP NEW	0.275	-4.478	49.569	4.555
87092	TIBX	T I B C O SOFTWARE INC	0.124	1.840	5.028	-0.387
82200	NSCP	NETSCAPE COMMUNICATIONS CORP	0.095	1.126	-1.167	-5.729
80266	QLGC	QLOGIC CORP	0.079	2.174	0.846	-2.837
27182	NA	TRANSITRON ELECTRONIC CORP	-0.068	1.234	1.700	-1.859
14983	W	WAYFAIR INC	0.065	3.186	1.869	-0.934
85522	AMCC	APPLIED MICRO CIRCUITS CORP	0.064	4.304	1.277	0.227
61524	CDO	COMDISCO INC	0.062	3.408	1.015	0.767
20894	APP	APPLOVIN CORP	0.062	3.572	1.841	-0.415
11746	ESB	E S B INC	0.060	-0.352	3.138	3.275

Table 8: Table 3: Stability of Risk Premia Across Sub-Periods

	Market Factor		Size Factor (SMB)		Value Factor (HML)	
Period	MKT_Premia	MKT_t	SMB_Premia	SMB_t	HML_Premia	HML_t
1995-2007	0.0053	1.3801	0.0066	1.8654	-0.0050	-1.7472
2008-2019	0.0019	0.4274	0.0026	1.1140	-0.0030	-1.1255
2020-2024	0.0155	1.9885	0.0020	0.4332	-0.0055	-0.8392

```r
table3_styled
```

```r
# Strategy Performance
table4_styled <- table4_data %>%
  kbl(caption = "Table 4: Out-of-Sample Trading Strategy Performance") %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F) %>%
  row_spec(which(table4_data$Portfolio == "Q5"), bold = TRUE, background = "#d4edda") %>%
  row_spec(which(table4_data$Portfolio == "Q1"), bold = TRUE, background = "#f8d7da")


table4_styled
```

```r
# Alpha Check
table5_styled <- table5_data %>%
  kbl(caption = "Table 5: Alpha Check (Regression of Strategy Returns on Factors)") %>%
  kable_styling(bootstrap_options = c("striped", "hover"), full_width = F) %>%
  row_spec(1, bold = TRUE)
```

Table 9: Table 4: Out-of-Sample Trading Strategy Performance

Portfolio	Mean_Return	Volatility	Sharpe_Ratio
Q1	**6.15%**	**14.34%**	**0.429**
Q5	**11.43%**	**19.81%**	**0.577**
LS	5.28%	13.64%	0.387
Mkt	7.05%	15.56%	0.453

Table 10: Table 5: Alpha Check (Regression of Strategy Returns on Factors)

Term	Estimate	t-stat	p-value	Signif
Alpha	**0.0007**	**0.57**	**0.566**	
MKT Exposure	0.3009	11.23	0.000	*
SMB Exposure	0.5584	14.28	0.000	*
HML Exposure	0.3785	9.72	0.000	*

```
table5_styled
```

7 References

1. Campbell, J. Y., Lo, A. W., and MacKinlay, A. C. (1997). *The Econometrics of Financial Markets*. Princeton University Press.
2. Cochrane, J. (2005). *Asset Pricing*. Princeton University Press.
3. Fama, E. F. and MacBeth, J. D. (1973). "Risk, return, and equilibrium: Empirical tests". *Journal of Political Economy*.
4. Fama, E. F. and French, K. R. (1993). "Common risk factors in the returns on stocks and bonds". *Journal of Financial Economics*.

8 Data Sources

1. WRDS Database and Kenneth French Data Library