Low-Volatility Cycles Analysis - Example Walkthrough

Advanced Investment Strategies LVC

2025-10-19

Contents

Introduction

This notebook demonstrates the complete workflow for replicating the Garcia-Feijóo et al. (2015) paper on Low-Volatility Cycles.

Paper Summary

Title: Low-Volatility Cycles: The Influence of Valuation and Momentum on Low-Volatility Portfolios

Main Finding: The low-volatility anomaly is driven by cyclical variations in book-to-price ratios and momentum effects.

Setup

Load Packages

```
# List of required packages
packages <- c(
  # Workflow and pipeline
  "targets",

  # Data wrangling and visualization
  "tidyverse",
  "lubridate",
  "zoo",
  "broom",
  "ggplot2",
  "knitr",

  # Database access and I/O
  "DBI",
  "RSQLite",
  "dbplyr",
  "readr"
)

# Install any missing packages
installed <- packages %in% rownames(installed.packages())
if (any(!installed)) {
  install.packages(packages[!installed])
}

# Load all required packages
lapply(packages, library, character.only = TRUE)
```

```
## [[1]]
## [1] "targets"   "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [7] "methods"   "base"
##
## [[2]]
##  [1] "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
##  [7] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "targets"   "stats"
## [13] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
```

```
## [[3]]
##  [1] "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
##  [7] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "targets"   "stats"
## [13] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[4]]
##  [1] "zoo"       "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"
##  [7] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse" "targets"
## [13] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [19] "base"
##
## [[5]]
##  [1] "broom"     "zoo"       "lubridate" "forcats"   "stringr"   "dplyr"
##  [7] "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
## [13] "targets"   "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [19] "methods"   "base"
##
## [[6]]
##  [1] "broom"     "zoo"       "lubridate" "forcats"   "stringr"   "dplyr"
##  [7] "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
## [13] "targets"   "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [19] "methods"   "base"
##
## [[7]]
##  [1] "knitr"     "broom"     "zoo"       "lubridate" "forcats"   "stringr"
##  [7] "dplyr"     "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"
## [13] "tidyverse" "targets"   "stats"     "graphics"  "grDevices" "utils"
## [19] "datasets"  "methods"   "base"
##
## [[8]]
##  [1] "DBI"       "knitr"     "broom"     "zoo"       "lubridate" "forcats"
##  [7] "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"     "tibble"
## [13] "ggplot2"   "tidyverse" "targets"   "stats"     "graphics"  "grDevices"
## [19] "utils"     "datasets"  "methods"   "base"
##
## [[9]]
##  [1] "RSQLite"   "DBI"       "knitr"     "broom"     "zoo"       "lubridate"
##  [7] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
## [13] "tibble"    "ggplot2"   "tidyverse" "targets"   "stats"     "graphics"
## [19] "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[10]]
##  [1] "dbplyr"    "RSQLite"   "DBI"       "knitr"     "broom"     "zoo"
##  [7] "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
## [13] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "targets"   "stats"
## [19] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[11]]
##  [1] "dbplyr"    "RSQLite"   "DBI"       "knitr"     "broom"     "zoo"
##  [7] "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
## [13] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "targets"   "stats"
## [19] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
```

```r
# Source all custom functions (used by {targets})
tar_source()
```

Configuration

```r
# Analysis parameters
N_PORTFOLIOS <- 5
BETA_WINDOW <- 60
MIN_OBS <- 24
```

Data Loading and Cleaning

Load Raw Data

```r
# Load CRSP data
crsp_raw <- load_crsp_data("data/crsp_monthly.csv")

# Load Compustat data
compustat_raw <- load_compustat_data("data/compustat_annual.csv")

# Load market returns
market_returns <- load_market_returns("data/market_returns.csv")

# Load Fama-French factors
ff_factors <- load_ff_factors("data/ff_factors.csv")
```

Clean and Merge Data

```r
# Clean CRSP
crsp_clean <- clean_crsp_data(crsp_raw)

# --- Price screen (abs(prc) in [2, 1000]) + NYSE large-cap breakpoint (monthly) ---
crsp_sz <- crsp_clean %>%
  # price screen (CRSP uses negative sign convention → use abs)
  filter(!is.na(prc), abs(prc) >= 2, abs(prc) <= 1000) %>%
  mutate(
    mktcap = abs(prc) * shrout * 1000  # USD
  ) %>%
  # guards
  filter(!is.na(mktcap), mktcap > 0, !is.na(exchcd), !is.na(date), !is.na(permno))

# NYSE 67th-percentile size cutoff per month
nyse_cut <- crsp_sz %>%
  filter(exchcd == 1L) %>%
  group_by(date) %>%
  summarize(cut67 = quantile(mktcap, probs = 2/3, na.rm = TRUE), .groups = "drop")

# Keep large-cap universe (apply NYSE cutoff to all exchanges)
crsp_largecap <- crsp_sz %>%
  inner_join(nyse_cut, by = "date") %>%
  filter(mktcap >= cut67)
```

```r
# Clean Compustat
compustat_clean <- clean_compustat_data(compustat_raw)

# Make Compustat unique at (permno, year)
compustat_unique <- compustat_clean %>%
  dplyr::mutate(year = if (!"year" %in% names(.)) lubridate::year(datadate) else year) %>%
  dplyr::arrange(permno, year, dplyr::desc(datadate), dplyr::desc(at), dplyr::desc(be)) %>%
  dplyr::group_by(permno, year) %>%
  dplyr::slice_head(n = 1) %>%
  dplyr::ungroup() %>%
  dplyr::select(permno, datadate, year, be_usd, at_usd)  # << keep *_usd


# Merge datasets
merged_data <- merge_crsp_compustat(crsp_largecap, compustat_unique)

# Preview
head(merged_data) %>% kable()
```

permno	date	ret	prc	shrout	shrcd	exchcd	mktcap	cut67	year	month	comp_year	be_usd	at_usd	bm	bp
10006	1969-10-01	0.113889	-50.125	5641	10	1	282755	12770	2997	5	1969	22800	70484	9410	0.000806	0.000806
10006	1969-11-01	0.002465	-49.625	5641	10	1	279934	6226	2016	7	1969	22800	70484	9410	0.000814	0.000814
10006	1969-12-01	-0.012595	-49.000	5641	10	1	276409	0026	4830	6	1969	22800	70484	9410	0.000824	0.000824
10006	1970-01-01	-0.053571	-46.375	5641	10	1	261601	3746	9229	1	1969	22800	70484	9410	0.000871	0.000871
10006	1970-02-01	0.051274	-48.125	5641	10	1	271473	1254	8108	2	1969	22800	70484	9410	0.000839	0.000839
10006	1970-03-01	0.023377	-49.250	5641	10	1	277819	2562	8016	3	1969	22800	70484	9410	0.000820	0.000820

Portfolio Formation

Calculate Rolling Betas (with parallel computation)

```r
# Packages
req_pkgs <- c("future", "future.apply")
new <- req_pkgs[!req_pkgs %in% rownames(installed.packages())]
if (length(new)) install.packages(new, quiet = TRUE)
lapply(req_pkgs, library, character.only = TRUE)
```

```
## [[1]]
##  [1] "future"    "dbplyr"    "RSQLite"   "DBI"       "knitr"     "broom"
```

```
##  [7] "zoo"       "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"
## [13] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse" "targets"
## [19] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [25] "base"
##
## [[2]]
##  [1] "future.apply" "future"       "dbplyr"       "RSQLite"      "DBI"
##  [6] "knitr"        "broom"        "zoo"          "lubridate"    "forcats"
## [11] "stringr"      "dplyr"        "purrr"        "readr"        "tidyr"
## [16] "tibble"       "ggplot2"      "tidyverse"    "targets"      "stats"
## [21] "graphics"     "grDevices"    "utils"        "datasets"     "methods"
## [26] "base"
```

```
# --- Detect and cap cores ---
n_cores_total <- parallel::detectCores(logical = TRUE)
n_cores <- min( max(1, n_cores_total - 1), 32 )  # cap at 32 workers to avoid connection overflow

cat("Detected", n_cores_total, "cores → using", n_cores, "workers\n")
```

```
## Detected 128 cores → using 32 workers
```

```
# Optional: limit BLAS threads per worker
if ("data.table" %in% .packages()) data.table::setDTthreads(1)
if ("RhpcBLASctl" %in% rownames(installed.packages())) {
  library(RhpcBLASctl)
  blas_set_num_threads(1)
  omp_set_num_threads(1)
}

# Set up plan
plan(multisession, workers = n_cores)

# Split data by stock
id_col <- "permno"
stopifnot(id_col %in% names(merged_data))
by_stock <- split(merged_data, merged_data[[id_col]], drop = TRUE)

# Compute rolling betas in parallel
beta_list <- future.apply::future_lapply(
  by_stock,
  function(df) calculate_rolling_betas(df, market_returns, window = BETA_WINDOW),
  future.seed = TRUE
)

stock_betas <- dplyr::bind_rows(beta_list)

plan(sequential)  # restore default

summary(stock_betas$beta) |>
  tibble::enframe(name = "stat", value = "beta") |>
  knitr::kable()
```

stat	beta
Min.	-0.4819
1st Qu.	0.6787

stat	beta
Median	0.9457
Mean	0.9613
3rd Qu.	1.2082
Max.	4.3486

Form Beta-Sorted Portfolios

```
# Keep valid betas
betas_ok <- stock_betas %>%
  dplyr::filter(!is.na(beta))

# Assign stocks to portfolios by within-month beta quantiles
beta_portfolios <- betas_ok %>%
  dplyr::group_by(date) %>%
  dplyr::mutate(portfolio = dplyr::ntile(beta, N_PORTFOLIOS)) %>%
  dplyr::ungroup()

# Portfolio distribution check
table(beta_portfolios$portfolio) %>% knitr::kable()
```

Var1	Freq
1	26615
2	26448
3	26310
4	26169
5	26031

```
# --- Basic distribution checks ------------------------------------------------
cat("\nNumber of stocks per beta portfolio:\n")
```

```
##
## Number of stocks per beta portfolio:
```

```
table(beta_portfolios$portfolio)
```

```
##
##     1     2     3     4     5
## 26615 26448 26310 26169 26031
```

```
cat("\nSample dates range:\n")
```

```
##
## Sample dates range:
```

```
range(beta_portfolios$date)
```

```
## [1] "1964-06-01" "2023-06-01"
```

```
cat("\nObservations per month (should be stable over time):\n")
```

```
##
## Observations per month (should be stable over time):
```

```r
beta_portfolios %>%
  dplyr::count(date) %>%
  summary(n)
```

```
##       date                  n        
##  Min.   :1964-06-01   Min.   :  1.0  
##  1st Qu.:1979-03-01   1st Qu.: 66.0  
##  Median :1993-12-01   Median :176.0  
##  Mean   :1993-11-30   Mean   :185.6  
##  3rd Qu.:2008-09-01   3rd Qu.:306.0  
##  Max.   :2023-06-01   Max.   :364.0
```

```r
# --- Average beta by portfolio ------------------------------------------------
beta_summary <- beta_portfolios %>%
  dplyr::group_by(portfolio) %>%
  dplyr::summarise(
    n_obs = dplyr::n(),
    mean_beta = mean(beta, na.rm = TRUE),
    sd_beta   = sd(beta, na.rm = TRUE),
    min_beta  = min(beta, na.rm = TRUE),
    max_beta  = max(beta, na.rm = TRUE),
    .groups = "drop"
  )

knitr::kable(beta_summary, caption = "Average Beta by Portfolio")
```

Table 4: Average Beta by Portfolio

portfolio	n_obs	mean_beta	sd_beta	min_beta	max_beta
1	26615	0.4482313	0.1835825	-0.4819263	1.050566
2	26448	0.7361554	0.1328851	0.1845468	1.131857
3	26310	0.9427337	0.1200000	0.4343826	1.397822
4	26169	1.1491460	0.1127868	0.6436831	2.038192
5	26031	1.5443872	0.3067701	0.8334124	4.348606

```r
# --- Merge with CRSP large-cap data to inspect market cap & returns -----------
beta_panel <- beta_portfolios %>%
  dplyr::select(permno, date, portfolio) %>%
  dplyr::inner_join(
    crsp_largecap %>% dplyr::select(permno, date, ret, mktcap),
    by = c("permno", "date")
  )

# 3) Average market cap and mean raw return per portfolio
capret_summary <- beta_panel %>%
  dplyr::group_by(portfolio) %>%
  dplyr::summarise(
    avg_mktcap_mil = mean(mktcap / 1e6, na.rm = TRUE),   # in billions
    mean_ret = mean(ret, na.rm = TRUE),
    sd_ret   = sd(ret, na.rm = TRUE),
    n_obs    = dplyr::n(),
    .groups  = "drop"
  )
```

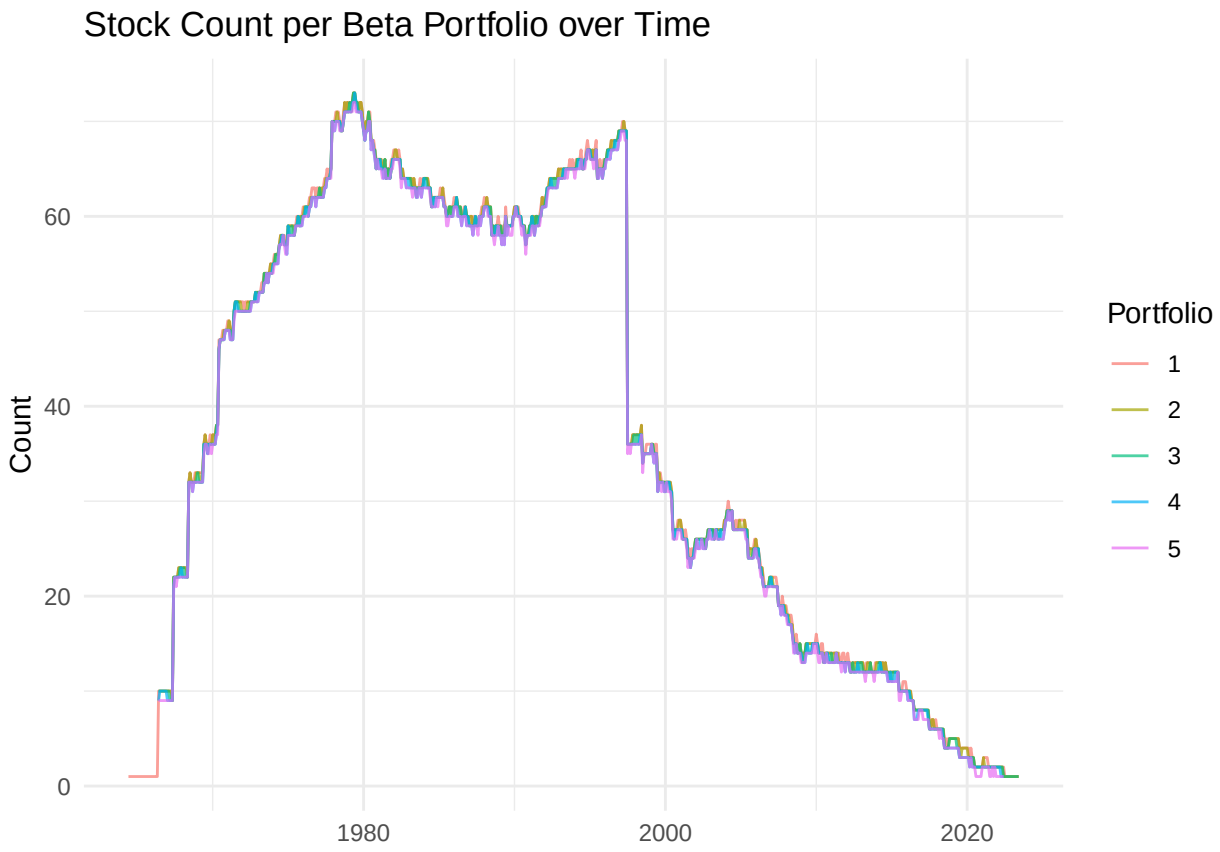
```
knitr::kable(capret_summary, caption = "Average Market Cap and Return by Portfolio")
```

Table 5: Average Market Cap and Return by Portfolio

portfolio	avg_mktcap_mil	mean_ret	sd_ret	n_obs
1	6060983	0.0116217	0.0619109	26615
2	7319066	0.0122614	0.0716316	26448
3	6720545	0.0135257	0.0807443	26310
4	6163045	0.0130761	0.0878929	26169
5	5556038	0.0149799	0.1081366	26031

```
# --- Optional: Time-series consistency check ---------------------------------
portfolio_counts <- beta_portfolios %>%
  dplyr::group_by(date, portfolio) %>%
  dplyr::summarise(n = dplyr::n(), .groups = "drop")

ggplot2::ggplot(portfolio_counts,
                ggplot2::aes(x = date, y = n, color = as.factor(portfolio))) +
  ggplot2::geom_line(alpha = 0.7) +
  ggplot2::labs(title = "Stock Count per Beta Portfolio over Time",
                x = NULL, y = "Count",
                color = "Portfolio") +
  ggplot2::theme_minimal()
```



Portfolio Analysis

Book-to-Price Spreads

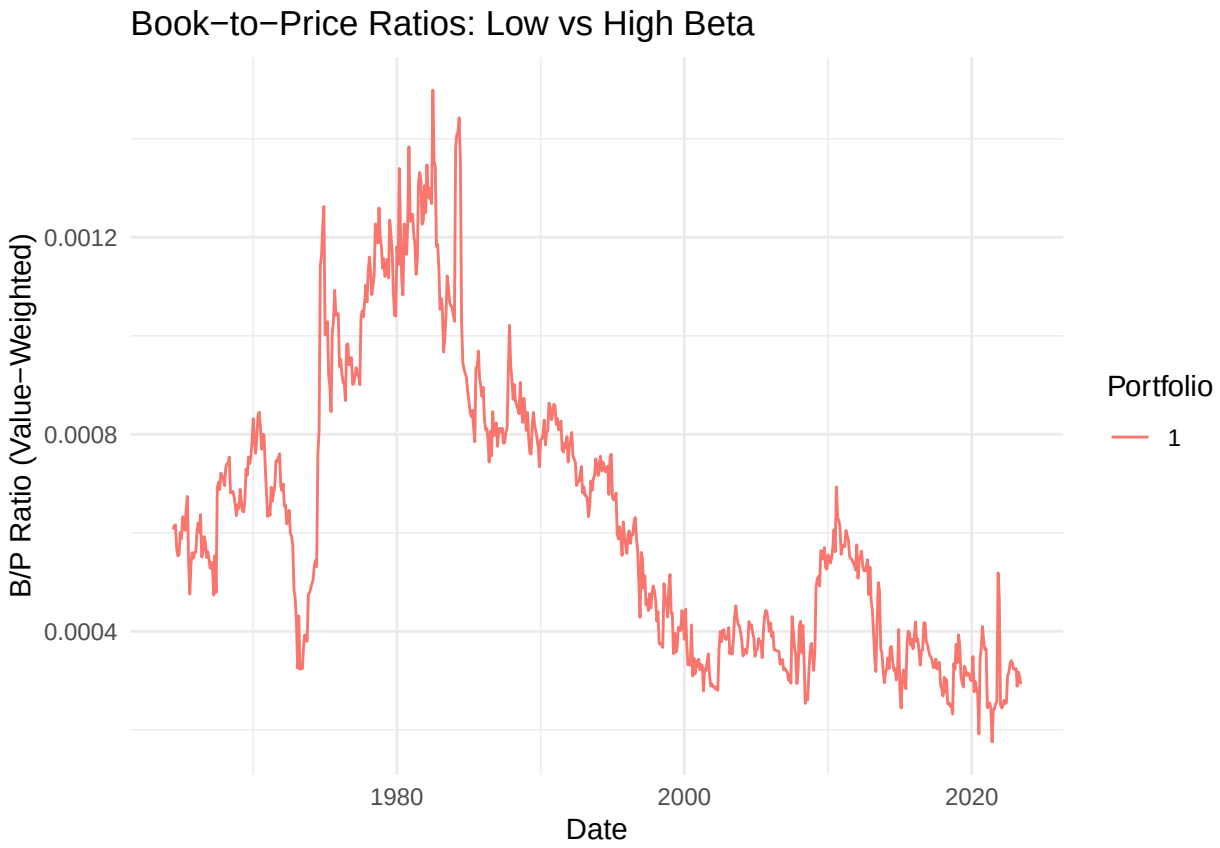
```
# Calculate B/P spreads

# join portfolios to accounting+CRSP panel to get bp and mktcap
beta_panel <- beta_portfolios %>%
  dplyr::select(permno, date, portfolio) %>%
  dplyr::inner_join(
    merged_data %>% dplyr::select(permno, date, bp, mktcap, ret),
    by = c("permno","date")
  ) %>%
  dplyr::arrange(permno, date) %>%
  dplyr::group_by(permno) %>%
  dplyr::mutate(w_lag = dplyr::lag(mktcap)) %>%
  dplyr::ungroup()



bp_spreads <- calculate_bp_spreads(beta_portfolios)

# Plot B/P spreads over time
bp_spreads %>%
  filter(portfolio %in% c(1, 10)) %>%
  ggplot(aes(x = date, y = bp_vw, color = as.factor(portfolio))) +
  geom_line() +
  labs(
    title = "Book-to-Price Ratios: Low vs High Beta",
    x = "Date",
    y = "B/P Ratio (Value-Weighted)",
    color = "Portfolio"
  ) +
  theme_minimal()
```



Portfolio Returns

```
# Calculate returns
portfolio_returns <- calculate_portfolio_returns(beta_portfolios)

# Summary by portfolio
portfolio_returns %>%
  group_by(portfolio) %>%
  summarise(
    mean_ret_ew = mean(ret_ew, na.rm = TRUE) * 12 * 100,
    sd_ret_ew = sd(ret_ew, na.rm = TRUE) * sqrt(12) * 100,
    sharpe_ew = mean_ret_ew / sd_ret_ew
  ) %>%
  kable(digits = 2)
```

portfolio	mean_ret_ew	sd_ret_ew	sharpe_ew
1	12.11	14.00	0.86
2	13.53	15.66	0.86
3	15.77	18.82	0.84
4	13.56	19.46	0.70
5	17.91	25.08	0.71

Regression Analysis

CAPM Regressions

```
# Run CAPM regressions
capm_results <- run_capm_regressions(portfolio_returns, market_returns)

# Display results
capm_results %>%
  select(portfolio, alpha_ew, alpha_t_ew, beta_ew, r2_ew) %>%
  kable(digits = 3)
```

portfolio	alpha_ew	alpha_t_ew	beta_ew	r2_ew
1	0.005	4.641	0.576	0.423
2	0.006	5.943	0.789	0.643
3	0.008	5.562	0.896	0.572
4	0.006	4.824	1.040	0.717
5	0.009	5.608	1.310	0.684
0	-0.007	NA	-0.688	NA

Fama-French 4-Factor

```
# Run FF4 regressions
ff4_results <- run_ff4_regressions(portfolio_returns, ff_factors)

# Display results
ff4_results %>%
  select(portfolio, alpha_ew, alpha_t_ew, beta_mkt_ew,
         beta_smb_ew, beta_hml_ew, beta_umd_ew) %>%
  kable(digits = 3)
```

portfolio	alpha_ew	alpha_t_ew	beta_mkt_ew	beta_smb_ew	beta_hml_ew	beta_umd_ew
1	0.002	1.848	0.671	-0.187	0.320	0.035
2	0.003	2.549	0.857	-0.166	0.179	0.011
3	0.005	3.351	0.924	-0.085	0.169	-0.127
4	0.002	1.432	1.071	-0.005	0.260	-0.130
5	0.005	3.335	1.291	0.046	0.174	-0.267
0	-0.007	NA	-0.572	-0.226	0.179	0.302

Results Visualization

Alpha Comparison

```
# Compare CAPM and FF4 alphas
alpha_comparison <- capm_results %>%
  select(portfolio, capm_alpha = alpha_ew) %>%
  left_join(
    ff4_results %>% select(portfolio, ff4_alpha = alpha_ew),
    by = "portfolio"
  ) %>%
```

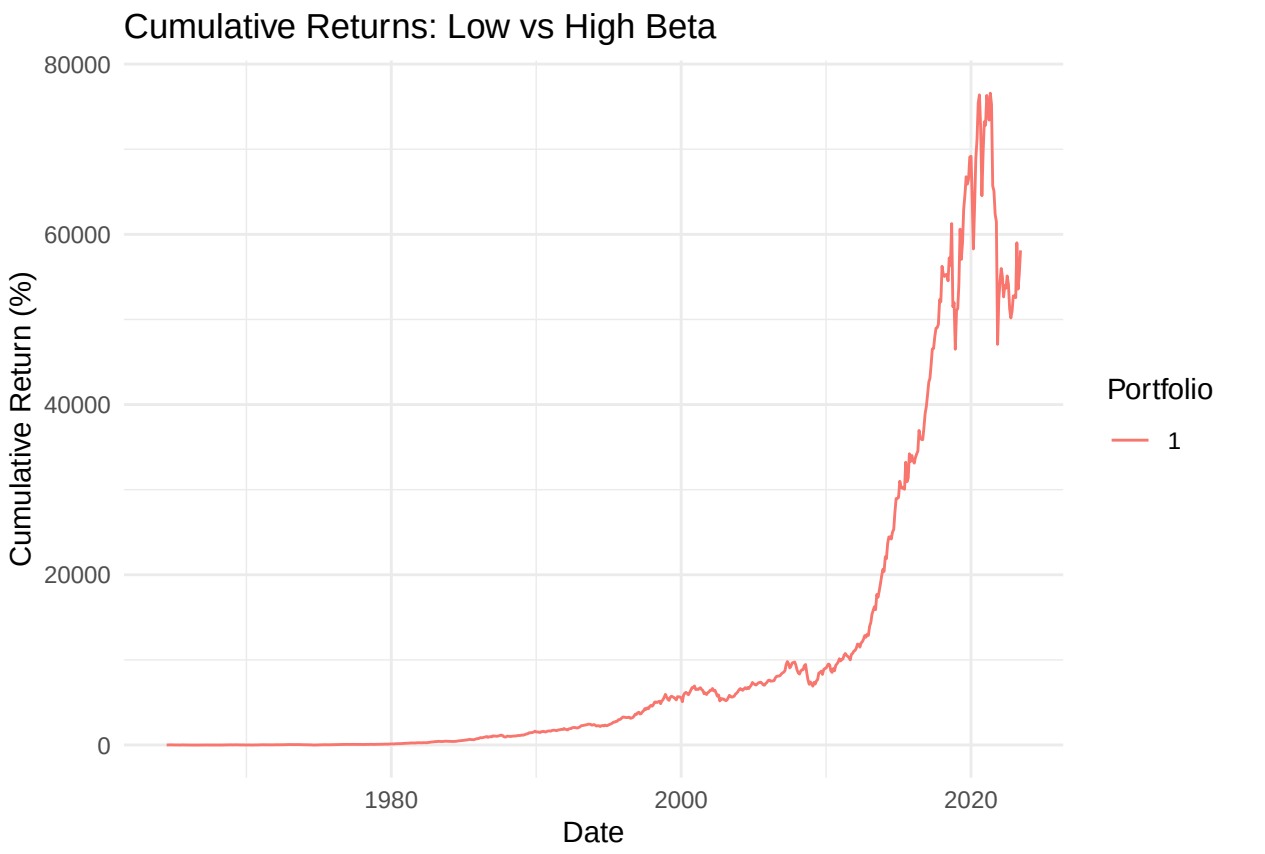
```
  filter(portfolio != 0) %>%
  pivot_longer(cols = c(capm_alpha, ff4_alpha),
               names_to = "model",
               values_to = "alpha")

ggplot(alpha_comparison, aes(x = portfolio, y = alpha * 100, fill = model)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(
    title = "Monthly Alphas by Portfolio",
    x = "Beta Portfolio (1=Low, 10=High)",
    y = "Alpha (% per month)",
    fill = "Model"
  ) +
  theme_minimal()
```

Cumulative Returns

```
# Calculate and plot cumulative returns
portfolio_returns %>%
  filter(portfolio %in% c(1, 10)) %>%
  group_by(portfolio) %>%
  arrange(date) %>%
  mutate(cum_ret = cumprod(1 + ret_vw) - 1) %>%
  ggplot(aes(x = date, y = cum_ret * 100, color = as.factor(portfolio))) +
  geom_line() +
```

```
  labs(
    title = "Cumulative Returns: Low vs High Beta",
    x = "Date",
    y = "Cumulative Return (%)",
    color = "Portfolio"
  ) +
  theme_minimal()
```



Using Targets Workflow

Instead of running code chunks individually, use the targets pipeline:

```
# Visualize pipeline
tar_visnetwork()

# Run full pipeline
tar_make()

# Load specific results
tar_load(table3)
tar_load(capm_results)

# View loaded data
print(table3)
```

Interpreting Results

Key Findings

1. **Low-Beta Anomaly**: Portfolio 1 (low beta) tends to outperform Portfolio 10 (high beta)
2. **Book-to-Price Effect**: Low-beta stocks often have higher B/P ratios
3. **Alpha Patterns**: After controlling for FF4 factors, alphas may diminish

Robustness Checks

Consider testing: - Different portfolio formation frequencies - Alternative beta estimation windows - Subperiod analysis - Size and liquidity filters

Conclusion

This analysis replicates the key findings of Garcia-Feijóo et al. (2015), showing how valuation and momentum influence low-volatility portfolio returns.

References

Garcia-Feijóo, L., Kochard, L., Sullivan, R. N., & Wang, P. (2015). Low-Volatility Cycles: The Influence of Valuation and Momentum on Low-Volatility Portfolios. *Financial Analysts Journal*, 71(3), 47-60.

Session Info

```
sessionInfo()
```

```
## R version 4.4.1 (2024-06-14)
## Platform: x86_64-pc-linux-gnu
## Running under: Ubuntu 22.04.5 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so;  LAPACK version 3.10.0
##
## Random number generation:
##  RNG:     L'Ecuyer-CMRG
##  Normal:  Inversion
##  Sample:  Rejection
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Etc/UTC
## tzcode source: system (glibc)
```

```
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] RhpcBLASctl_0.23-42 future.apply_1.11.3 future_1.34.0
##  [4] dbplyr_2.5.1        RSQLite_2.3.9       DBI_1.2.3
##  [7] knitr_1.50          broom_1.0.10        zoo_1.8-14
## [10] lubridate_1.9.4     forcats_1.0.0       stringr_1.5.1
## [13] dplyr_1.1.4         purrr_1.1.0         readr_2.1.5
## [16] tidyr_1.3.1         tibble_3.2.1        ggplot2_4.0.0
## [19] tidyverse_2.0.0     targets_1.11.4
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.6       xfun_0.51          processx_3.8.6     lattice_0.22-6
##  [5] callr_3.7.6        tzdb_0.5.0         vctrs_0.6.5        tools_4.4.1
##  [9] ps_1.9.0           generics_0.1.4     base64url_1.4      parallel_4.4.1
## [13] blob_1.2.4         pkgconfig_2.0.3    data.table_1.17.8  secretbase_1.0.5
## [17] RColorBrewer_1.1-3 S7_0.2.0           lifecycle_1.0.4    compiler_4.4.1
## [21] farver_2.1.2       tinytex_0.57       codetools_0.2-20   htmltools_0.5.8.1
## [25] yaml_2.3.10        pillar_1.10.2      cachem_1.1.0       parallelly_1.43.0
## [29] tidyselect_1.2.1   digest_0.6.37      stringi_1.8.7      listenv_0.9.1
## [33] labeling_0.4.3     fastmap_1.2.0      grid_4.4.1         cli_3.6.5
## [37] magrittr_2.0.3     dichromat_2.0-0.1  withr_3.0.2        prettyunits_1.2.0
## [41] scales_1.4.0       backports_1.5.0    bit64_4.6.0-1      timechange_0.3.0
## [45] rmarkdown_2.30     globals_0.16.3     igraph_2.1.4       bit_4.6.0
## [49] hms_1.1.3          memoise_2.0.1      evaluate_1.0.3     rlang_1.1.6
## [53] glue_1.8.0         rstudioapi_0.17.1  R6_2.6.1
```