Appendix B : Code

Coding for this section was completed using RStudio 2024.09.0+375 ("Cranberry Hibiscus" Release) and was based on R and MATLAB code provided by Professor Tim Gebbie(STA4028Z).

2.1 Libraries (Gebbie, 2025d)

```
# load required libraries
suppressPackageStartupMessages({
library(openxlsx)
library(timeSeries)
library(xts)
library(zoo)
library(matrixStats)
library(quadprog)
library(knitr)
library(dplyr)
library(ggplot2)
library(tidyr)
library(PerformanceAnalytics)
})
```

2.2 Load data and preprocessing (Gebbie, 2025d)

```
# reading in all 4 sheets into a list
dfS <- list()
for (i in 1:4) {
  dfS[[i]] <- read.xlsx("_raw_data/PT-TAA-JSE-Daily-1994-2017.xlsx", sheet = i, detectDa
  cat("Sheet", i, "loaded with dimensions:", dim(dfS[[i]]), "\n")
}
```

```
## Sheet 1 loaded with dimensions: 8439 2
## Sheet 2 loaded with dimensions: 8405 4
## Sheet 3 loaded with dimensions: 8439 28
## Sheet 4 loaded with dimensions: 8439 20
```

```r
# define entities and which assets to keep
Entities <- c('X1','STEFI','ALBI','J203','J500', sprintf("J5%d", seq(10,90,by=10)))
Items    <- c('Date','TRI','Stefi')

#cleaning each sheet
for (i in 1:4) {
  tI0 <- sapply(colnames(dfS[[i]]), function(x) any(grepl(paste(Entities, collapse="|")
  tI1 <- sapply(dfS[[i]][2,], function(x) any(grepl(paste(Items, collapse="|"), x)))
  tI  <- tI0 & tI1

  # remove header rows
  dfS[[i]] <- dfS[[i]][-c(1,2), tI]
  names(dfS[[i]])[1] <- "Date"

  newColNames <- strsplit(colnames(dfS[[i]]), ":")
  for(m in 2:length(newColNames)) names(dfS[[i]])[m] <- newColNames[[m]][1]

  cat("Sheet", i, "columns after cleaning:", colnames(dfS[[i]]), "\n")
}
```

```
## Sheet 1 columns after cleaning: Date ALBI
## Sheet 2 columns after cleaning: Date RATESTEFI
## Sheet 3 columns after cleaning: Date J500 J510 J520 J530 J540 J550 J560 J580 J590
## Sheet 4 columns after cleaning: Date J203
```

```r
# fixing ALBI column
dfS[[1]][,2] <- as.numeric(dfS[[1]][,2])
dfS[[1]] <- dfS[[1]][!is.na(dfS[[1]][,2]), ]#removes rows where ALBI is NA
```

2.3 Merge into single timeSeries object (Gebbie, 2025d)

```r
# converts first sheet to timeSeries
tsTAA <- timeSeries(dfS[[1]][, 2:ncol(dfS[[1]])], as.Date(dfS[[1]][,1]))
cat("Initial tsTAA dimensions:", dim(tsTAA), "\n")
```

```
## Initial tsTAA dimensions: 4324 1
```

```r
# merges remaining sheets
for (i in 2:4) {
  tsTmp <- timeSeries(dfS[[i]][, 2:ncol(dfS[[i]])], as.Date(dfS[[i]][,1]))
  tsTAA <- cbind(tsTAA, tsTmp)
  cat("After merging sheet", i, "dimensions:", dim(tsTAA), "\n")
}
```

```
## After merging sheet 2 dimensions: 8437 2
## After merging sheet 3 dimensions: 8437 11
## After merging sheet 4 dimensions: 8437 12
```

```r
# renaming indices for clarity
setFinCenter(tsTAA) <- "Johannesburg"
names(tsTAA)[grep("TS.1.1", names(tsTAA))] <- "ALBI"
names(tsTAA)[grep("TS.1.2", names(tsTAA))] <- "STEFI"
names(tsTAA)[grep("TS.1", names(tsTAA))] <- "ALSI"

cat("Columns after renaming:", colnames(tsTAA), "\n")
```

```
## Columns after renaming: ALBI STEFI J500 J510 J520 J530 J540 J550 J560 J580 J590 ALSI
```

```r
#all numeric columns are numeric
for (j in 1:ncol(tsTAA)) {
  tsTAA[, j] <- as.numeric(tsTAA[, j])
}
#remove rows with all NAs
tsTAA <- tsTAA[rowSums(is.na(tsTAA)) < ncol(tsTAA), ]

# Using timeSeries daily2monthly and ensure tsTAA is valid
tsTAA_monthly <- tryCatch(
  daily2monthly(tsTAA),
  error = function(e) {
    stop("Error in daily2monthly: tsTAA might contain non-timeSeries columns or non-nume
  }
)

#  monthly price index
tsIdx  <- index2wealth(tsTAA_monthly)
```

```r
# geometric monthly returns
tsGRet <- diff(log(tsIdx))

cat("tsTAA_monthly dimensions:", dim(tsTAA_monthly), "\n")
```

```
## tsTAA_monthly dimensions: 261 12
```

```r
cat("tsGRet dimensions:", dim(tsGRet), "\n")
```

```
## tsGRet dimensions: 261 12
```

```r
cat("Columns in tsGRet:\n"); print(colnames(tsGRet))
```

```
## Columns in tsGRet:

##  [1] "ALBI"  "STEFI" "J500"  "J510"  "J520"  "J530"  "J540"  "J550"  "J560"
## [10] "J580"  "J590"  "ALSI"
```

2.4 Arithmetic Returns (Gebbie, 2025c)

```r
setFinCenter(tsTAA) <- "Africa/Johannesburg"
summary(dfS[[1]][,2])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   173.7   256.9   343.9   357.5   442.0   545.9
```

```r
# Checks that tsTAA is a proper 'timeSeries' object
tsTAA_monthly <- tryCatch(
  daily2monthly(tsTAA),
  error = function(e) {
    message("Error in daily2monthly(): converting tsTAA to xts first")
    xts_obj <- as.xts(tsTAA)
    apply.monthly(xts_obj, colMeans, na.rm=TRUE)
  }
)




#geometric returns
tsGRet <- diff(log(tsTAA_monthly))
```

```r
#  fill missing data using LOCF
tsGRet_filled <- na.locf(as.xts(tsGRet), na.rm = FALSE)
summary(tsGRet_filled[,"ALBI"])
```

```
##      Index                                 ALBI
##  Min.   :1995-06-30 00:00:00.00   Min.   :-0.06908
##  1st Qu.:2000-11-30 00:00:00.00   1st Qu.:-0.00232
##  Median :2006-04-30 00:00:00.00   Median : 0.00362
##  Mean   :2006-04-30 22:31:43.45   Mean   : 0.00701
##  3rd Qu.:2011-09-30 00:00:00.00   3rd Qu.: 0.01581
##  Max.   :2017-02-28 00:00:00.00   Max.   : 0.16900
##                                   NA's   :99
```

```r
any(!is.na(tsGRet_filled[,"ALBI"]))
```

```
## [1] TRUE
```

```r
#checking for columns that are all NA
cols_allNA <- colSums(!is.na(tsGRet_filled)) == 0
tsGRet_filled <- tsGRet_filled[, !cols_allNA]

# converting to arithmetic returns
simple_mat <- exp(as.matrix(tsGRet_filled)) - 1
rets_xts <- xts(simple_mat, order.by = index(tsGRet_filled))
colnames(rets_xts) <- colnames(tsGRet_filled)

# Excludes cash asset
cash_idx <- grep("STEFI", colnames(rets_xts), ignore.case = TRUE)
cash_name <- ifelse(length(cash_idx) > 0, colnames(rets_xts)[cash_idx[1]], NA)

rets_opt <- if(!is.na(cash_name)) rets_xts[, -cash_idx, drop=FALSE] else rets_xts
rets_cash <- if(!is.na(cash_name)) rets_xts[, cash_idx, drop=FALSE] else NULL

cat("Assets used for optimisation:\n"); print(colnames(rets_opt))
```

```
## Assets used for optimisation:

##  [1] "ALBI" "J500" "J510" "J520" "J530" "J540" "J550" "J560" "J580" "J590"
```

```
## [11] "ALSI"
```

```r
if(!is.na(cash_name)) cat("Cash excluded from optimisation:", cash_name, "\n")
```

```
## Cash excluded from optimisation: STEFI
```

2.5 Black-Litterman function (Gebbie, 2025c, 2025d)

```r
blacklittermann <- function(Pi, Sigma, P, Q, Omega, tau=0.05, gamma=1, constrain=TRUE){
  n <- length(Pi)
  Sigma_inv <- solve(Sigma)
  Omega_inv <- solve(Omega)
  mu_post <- solve(Sigma_inv*tau + t(P) %*% Omega_inv %*% P) %*%
             (Sigma_inv %*% (tau * Pi) + t(P) %*% Omega_inv %*% Q)
  Sigma_post <- Sigma + solve(Sigma_inv*tau + t(P) %*% Omega_inv %*% P)
  w <- solve(gamma*Sigma_post) %*% mu_post
  if(constrain){
    w[w<0] <- 0
    w <- w / sum(w)
  }
  w <- as.numeric(w)
  names(w) <- names(Pi)
  list(weights=w, mu_post=mu_post, Sigma_post=Sigma_post)
}
```

2.5 a. Test Case

```r
Pi <- c(0.25, 0.10, 0.05)
Sigma <- matrix(c(0.09, 0.024, -0.006,
                  0.024, 0.01, 0.0003,
                  -0.006, 0.0003, 0.0025), nrow=3, byrow=TRUE)
P <- matrix(c(1, -1, 0,
              0, 1, -1), nrow=2, byrow=TRUE)
Q <- matrix(c(0.2, -0.05), nrow=2)
Omega <- matrix(c(0.3, 0,
                  0, 0.55), nrow=2)
bl_test <- blacklittermann(Pi, Sigma, P, Q, Omega, tau=0.05, gamma=1, constrain=TRUE)
bl_test$weights
```

```
## [1] 0.3063286 0.0000000 0.6936714
```

```r
sum(bl_test$weights)
```

```
## [1] 1
```

2.6 Rolling Window Experiment (Gebbie, 2025c, 2025d)

```r
### Rolling-window Black-Litterman Backtest
train.m <- 60  # 5 yr training period
test.m  <- 12  # 1 yr test period
roll_step <- 1 # 1-mthincrements
n_obs <- nrow(rets_opt)
start_idxs <- seq(1, n_obs - train.m - test.m + 1, by=roll_step)
results <- list()
prev_w <- rep(0, ncol(rets_opt)) # initialising previous weights for turnover

# ALSI column for beta calc
benchmark <- tsGRet_filled[, "ALSI", drop=FALSE]

for(i in seq_along(start_idxs)){
  s <- start_idxs[i]
  train_idx <- s:(s+train.m-1)
  tst.idx   <- (s+train.m):(s+train.m+test.m-1)

  train_rets <- rets_opt[train_idx, , drop=FALSE]
  tst_rets   <- rets_opt[tst.idx, , drop=FALSE]
  bench_train <- benchmark[train_idx, , drop=FALSE]
  bench_test  <- benchmark[tst.idx, , drop=FALSE]

  # invalid windows
  if(any(!is.finite(as.matrix(train_rets))) || any(!is.finite(as.matrix(tst_rets)))) ne

  mu_train    <- colMeans(train_rets, na.rm=TRUE)
  Sigma_train <- cov(as.matrix(train_rets), use="complete.obs")

  # Risk-free
  rf_train <- if(!is.null(rets_cash)) mean(rets_cash[train_idx, ], na.rm=TRUE) else 0
```

```r
  rf_test  <- if(!is.null(rets_cash)) mean(rets_cash[tst.idx, ], na.rm=TRUE) else 0

  # Black-Litterman views
  P <- matrix(0, nrow=1, ncol=ncol(train_rets))
  P[1, 1:3] <- 1/3        # simple view
  Q <- matrix(0.01, nrow=1) # expected 1% return
  Omega <- matrix(0.0001, nrow=1) # view uncertainty

  bl_res <- blacklittermann(Pi=mu_train, Sigma=Sigma_train, P=P, Q=Q, Omega=Omega, tau=(

  # Constraint weights: no short, fully invested
  w_hat <- bl_res$weights
  w_hat[w_hat < 0] <- 0
  w_hat <- w_hat / sum(w_hat)

  turnover <- sum(abs(w_hat - prev_w))
  prev_w <- w_hat

  # Portfolio returns
  port_train <- as.numeric(as.matrix(train_rets) %*% w_hat)
  port_test  <- as.numeric(as.matrix(tst_rets) %*% w_hat)

  # Tracking Error relative to ALSI
port_diff <- port_test - as.numeric(bench_test)
t.err <- sd(port_diff, na.rm = TRUE)

  # Jensen Alpha & Beta relative to ALSI
xcess.p.train  <- port_train  - rf_train
xcess.b.train <- as.numeric(bench_train) - rf_train
CAPM_train <- lm(xcess.p.train ~ xcess.b.train)
alpha_IS <- coef(CAPM_train)[1]
beta_IS  <- coef(CAPM_train)[2]

# OOS excess returns
excess_port_test  <- port_test  - rf_test
excess_bench_test <- as.numeric(bench_test) - rf_test
```

```r
CAPM_test <- lm(excess_port_test ~ excess_bench_test)
alpha_OOS <- coef(CAPM_test)[1]
beta_OOS  <- coef(CAPM_test)[2]

  # Cumulative equity curve
eq_curve_test <- cumprod(1 + port_test)

  # Results
  results[[length(results)+1]] <- list(
  train_period = paste(index(train_rets)[1], index(train_rets)[nrow(train_rets)], sep="
  test_period  = paste(index(tst_rets)[1], index(tst_rets)[nrow(tst_rets)], sep=" / "),
  mu_IS = mean(port_train, na.rm=TRUE),
  var_IS = var(port_train, na.rm=TRUE),
  SR_IS = (mean(port_train, na.rm=TRUE) - rf_train)/sqrt(var(port_train, na.rm=TRUE)),
  mu_OOS = mean(port_test, na.rm=TRUE),
  var_OOS = var(port_test, na.rm=TRUE),
  SR_OOS = (mean(port_test, na.rm=TRUE) - rf_test)/sqrt(var(port_test, na.rm=TRUE)),
  alpha_IS = alpha_IS,
  beta_IS = beta_IS,
  alpha_OOS = alpha_OOS,
  beta_OOS = beta_OOS,
  turnover = turnover,
  t.err = t.err,
  eq_curve_test = eq_curve_test,
  weights = w_hat,
  assets  = colnames(rets_opt)
)

}

# makes it easier to plot
summary_df <- do.call(rbind, lapply(results, function(x) data.frame(
  train=x$train_period, test=x$test_period,
  mu_IS=x$mu_IS, var_IS=x$var_IS, SR_IS=x$SR_IS,
  mu_OOS=x$mu_OOS, var_OOS=x$var_OOS, SR_OOS=x$SR_OOS,
```

```r
  alpha_IS=x$alpha_IS, beta_IS=x$beta_IS,
  alpha_OOS=x$alpha_OOS, beta_OOS=x$beta_OOS,
  turnover=x$turnover,
  t.err = x$t.err
)))


knitr::kable(head(summary_df,10),
  digits = 4,
  caption = "In-sample vs Out-of-sample BL Portfolio Statistics"
)
```

Table 1: In-sample vs Out-of-sample BL Portfolio Statistics

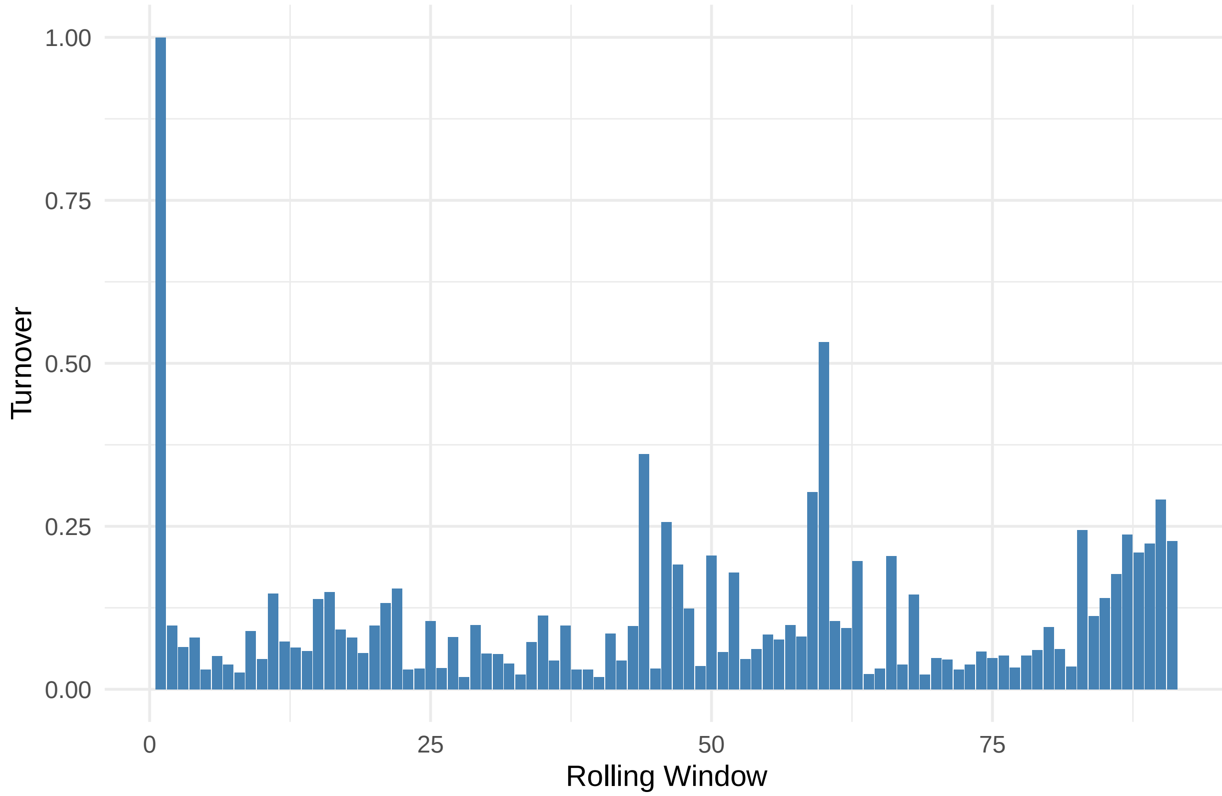
	train	test	mu_IS	var_IS	SR_IS	mu_OOS	var_OOS	SR_OOS	alpha_IS	beta_IS	alpha_OOS	beta_OOS	turnover	t.err
(Intercept)	2003-09-30 / 2008-08-31	2008-09-30 / 2009-08-31	0.0186	0.0013	0.3252	-0.0048	0.0048	-0.1914	0.0014	0.7444	-0.0020	0.7920	1.0000	0.0242
(Intercept)1	2003-10-31 / 2008-09-30	2008-10-31 / 2009-09-30	0.0167	0.0015	0.2488	0.0033	0.0036	-0.0830	0.0009	0.7493	-0.0034	0.7763	0.0979	0.0236
(Intercept)2	2003-11-30 / 2008-10-31	2008-11-30 / 2009-10-31	0.0145	0.0017	0.1838	0.0153	0.0026	0.1474	0.0013	0.7612	-0.0030	0.7796	0.0649	0.0209
(Intercept)3	2003-12-31 / 2008-11-30	2008-12-31 / 2009-11-30	0.0148	0.0016	0.1939	0.0144	0.0026	0.1349	0.0016	0.7484	-0.0045	0.7784	0.0791	0.0209
(Intercept)4	2004-01-31 / 2008-12-31	2009-01-31 / 2009-12-31	0.0142	0.0016	0.1818	0.0142	0.0026	0.1387	0.0017	0.7373	-0.0053	0.7731	0.0300	0.0206

	train	test	mu_IS	var_IS	SR_IS	mu_OOS	var_OOS	SR_OOS	alpha_IS	beta_IS	alpha_OOS	beta_OOS	turnover	te_err
(Intercept)	2004-02-29 / 2009-01-31	2009-02-28 / 2010-01-31	0.0133	0.0016	0.1579	0.0142	0.0025	0.1450	0.0019	0.7363	-0.0059	0.7802	0.0512	0.0194
(Intercept)	2004-03-31 / 2009-02-28	2009-03-31 / 2010-02-28	0.0122	0.0017	0.1249	0.0204	0.0018	0.3230	0.0021	0.7345	-0.0087	0.8546	0.0378	0.0139
(Intercept)	2004-04-30 / 2009-03-31	2009-04-30 / 2010-03-31	0.0136	0.0018	0.1545	0.0192	0.0017	0.3069	0.0021	0.7311	-0.0089	0.9028	0.0261	0.0123
(Intercept)	2004-05-31 / 2009-04-30	2009-05-31 / 2010-04-30	0.0136	0.0017	0.1550	0.0194	0.0016	0.3212	0.0016	0.7239	-0.0064	0.8534	0.0890	0.0126
(Intercept)	2004-06-30 / 2009-05-31	2009-06-30 / 2010-05-31	0.0152	0.0018	0.1885	0.0084	0.0013	0.0604	0.0020	0.7285	-0.0058	0.7720	0.0467	0.0136

2.7 Portfolio Turnover

```r
turnover_df <- data.frame(
  Window = 1:length(results),
  Turnover = sapply(results, function(x) x$turnover)
)

# Portfolio Turnover per Rolling Window
ggplot(turnover_df, aes(x = Window, y = Turnover)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "",
       x = "Rolling Window", y = "Turnover") +
  theme_minimal()
```



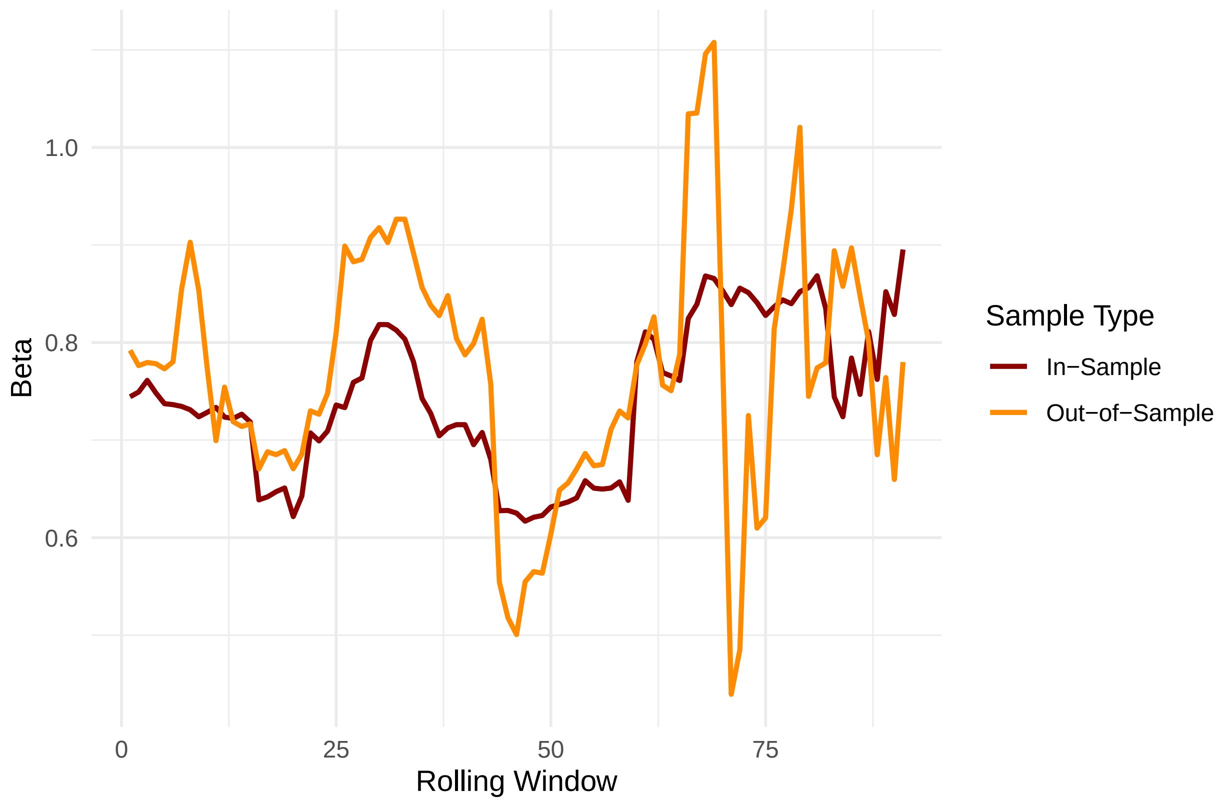
2.8 Jensens alpha and beta

```r
# in-sample and out-of-sample Alpha and Beta
ab_df <- data.frame(
  Window = 1:length(results),
  alpha_IS = sapply(results, function(x) x$alpha_IS),
  alpha_OOS = sapply(results, function(x) x$alpha_OOS),
  beta_IS = sapply(results, function(x) x$beta_IS),
  beta_OOS = sapply(results, function(x) x$beta_OOS)
)

# In-Sample vs. Out-of-Sample Jensen's Alpha
ggplot(ab_df, aes(x = Window)) +
  geom_line(aes(y = alpha_IS, color = "In-Sample"), linewidth = 0.9) +
  geom_line(aes(y = alpha_OOS, color = "Out-of-Sample"), linewidth = 0.9) +
  labs(title = "",
       x = "Rolling Window", y = "Alpha", color = "Sample Type") +
```

```r
  theme_minimal() +
  scale_color_manual(values = c("In-Sample" = "darkgreen", "Out-of-Sample" = "darkblue"
```


```r
# In-Sample vs. Out-of-Sample Portfolio Beta
ggplot(ab_df, aes(x = Window)) +
  geom_line(aes(y = beta_IS, color = "In-Sample"), linewidth = 0.9) +
  geom_line(aes(y = beta_OOS, color = "Out-of-Sample"), linewidth = 0.9) +
  labs(title = "",
       x = "Rolling Window", y = "Beta", color = "Sample Type") +
  theme_minimal() +
  scale_color_manual(values = c("In-Sample" = "darkred", "Out-of-Sample" = "darkorange"
```



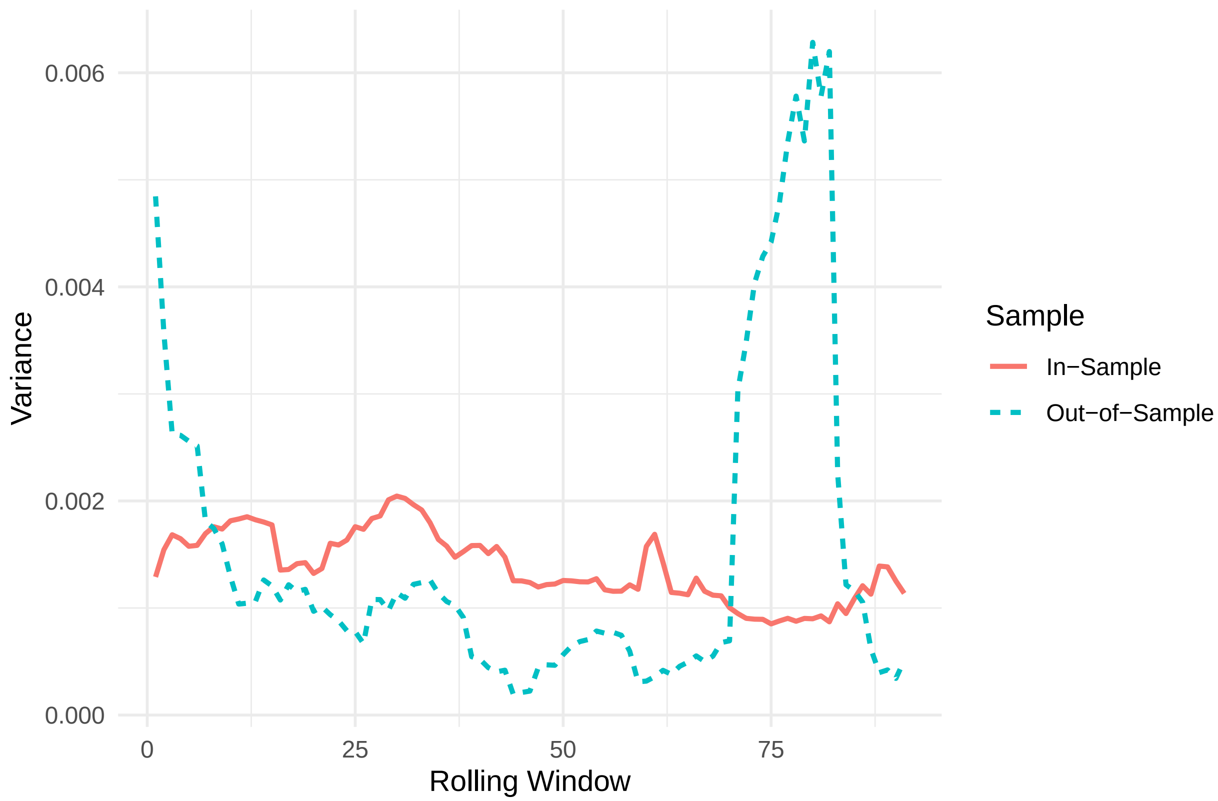
2.9 Plotting In Sample vs Out Of Sample Statistics

```
library(tidyr)
summary_df <- do.call(rbind, lapply(seq_along(results), function(i) {
  x <- results[[i]]
  data.frame(
    Window = i,
    train = x$train_period,
    test = x$test_period,
    mu_IS = x$mu_IS, var_IS = x$var_IS, SR_IS = x$SR_IS,
    mu_OOS = x$mu_OOS, var_OOS = x$var_OOS, SR_OOS = x$SR_OOS,
    alpha_IS = x$alpha_IS, beta_IS = x$beta_IS,
    alpha_OOS = x$alpha_OOS, beta_OOS = x$beta_OOS,
    turnover = x$turnover,
    t.err = x$t.err
  )
}))
```

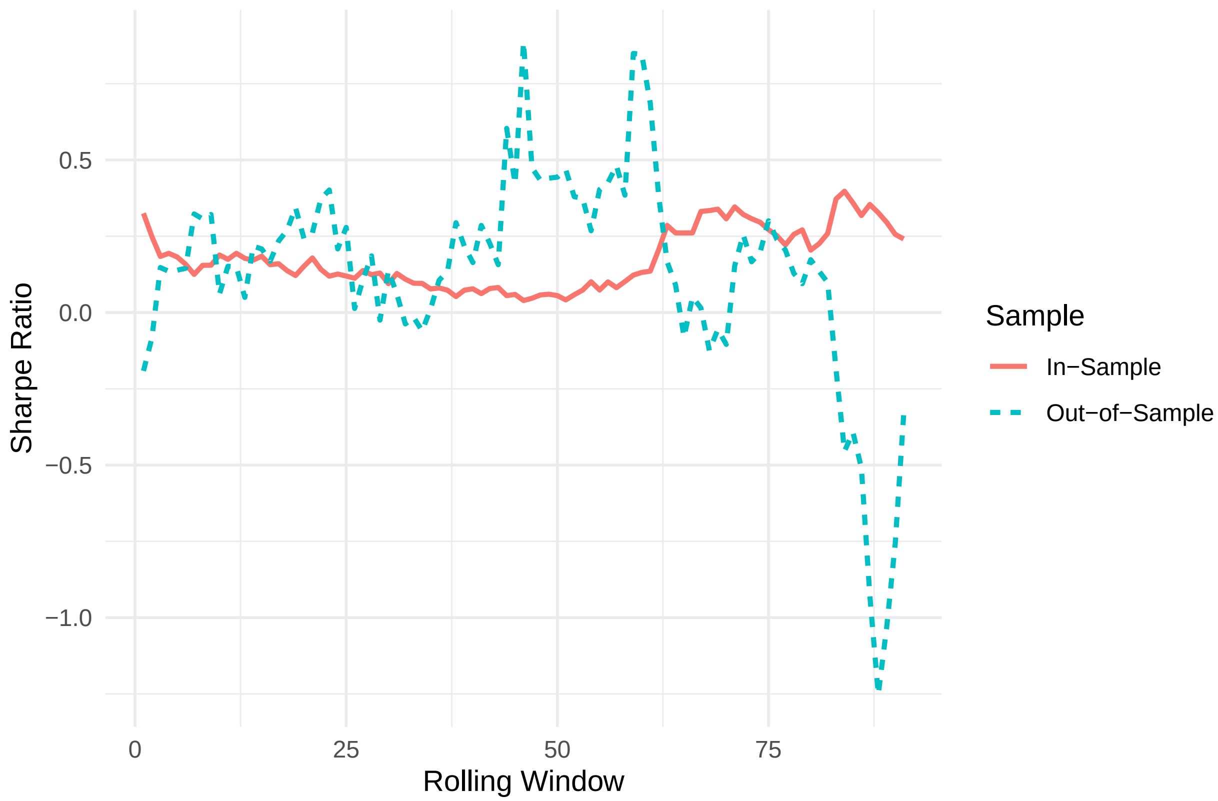
```r
sum_l <- summary_df %>%
  pivot_longer(
    cols = c(mu_IS, var_IS, SR_IS, mu_OOS, var_OOS, SR_OOS),
    names_to = "Metric",
    values_to = "Value"
  ) %>%
  mutate(
    Sample = ifelse(grepl("_IS", Metric), "In-Sample", "Out-of-Sample"),
    Metric = gsub("_(IS|OOS)", "", Metric),
    Metric = case_when(
      Metric == "mu" ~ "Mean",
      Metric == "var" ~ "Variance",
      Metric == "SR" ~ "Sharpe",
      TRUE ~ Metric
    )
  )


# Mean
ggplot(subset(sum_l, Metric == "Mean"),
       aes(x = Window, y = Value, color = Sample, linetype = Sample)) +
  geom_line(linewidth = 0.9) +
  labs(
    title = "",
    x = "Rolling Window",
    y = "Mean Return"
  ) +
  theme_minimal()
```


```r
# Variance
ggplot(subset(sum_l, Metric == "Variance"),
       aes(x = Window, y = Value, color = Sample, linetype = Sample)) +
  geom_line(linewidth = 0.9) +
  labs(
    title = "",
    x = "Rolling Window",
    y = "Variance"
  ) +
  theme_minimal()
```




```r
# Sharpe Ratio
ggplot(subset(sum_l, Metric == "Sharpe"),
       aes(x = Window, y = Value, color = Sample, linetype = Sample)) +
  geom_line(linewidth = 0.9) +
  labs(
    title = "",
    x = "Rolling Window",
    y = "Sharpe Ratio"
  ) +
  theme_minimal()
```

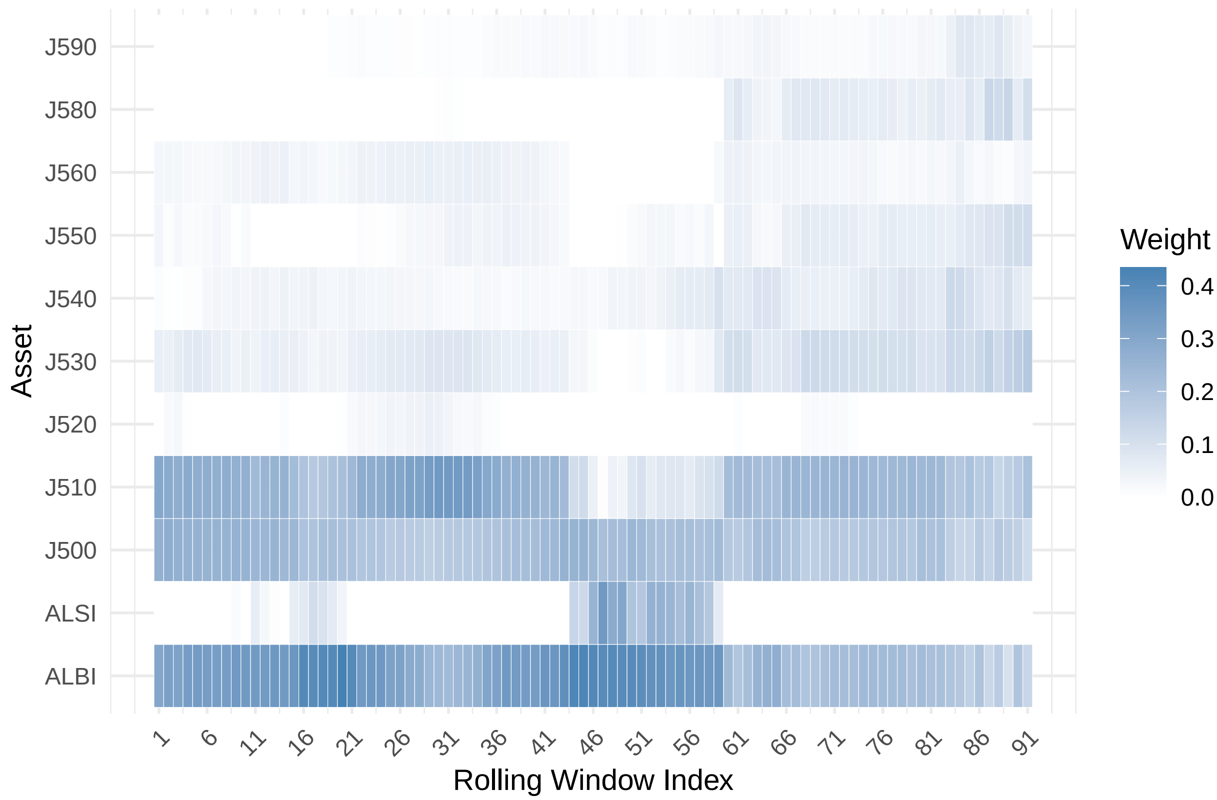


2.10 Portfolio Weights

```
# Combine all rolling window weights into a single data frame
weights_df <- do.call(rbind, lapply(seq_along(results), function(i) {
  data.frame(
    Window = i, #index
    Asset  = results[[i]]$assets,# asset names
    Weight = results[[i]]$weights,#  weights
    stringsAsFactors = FALSE
  )
}))

# Evolution of Black-Litterman Portfolio Weights
ggplot(weights_df, aes(x = Window, y = Asset, fill = Weight)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  scale_x_continuous(
```

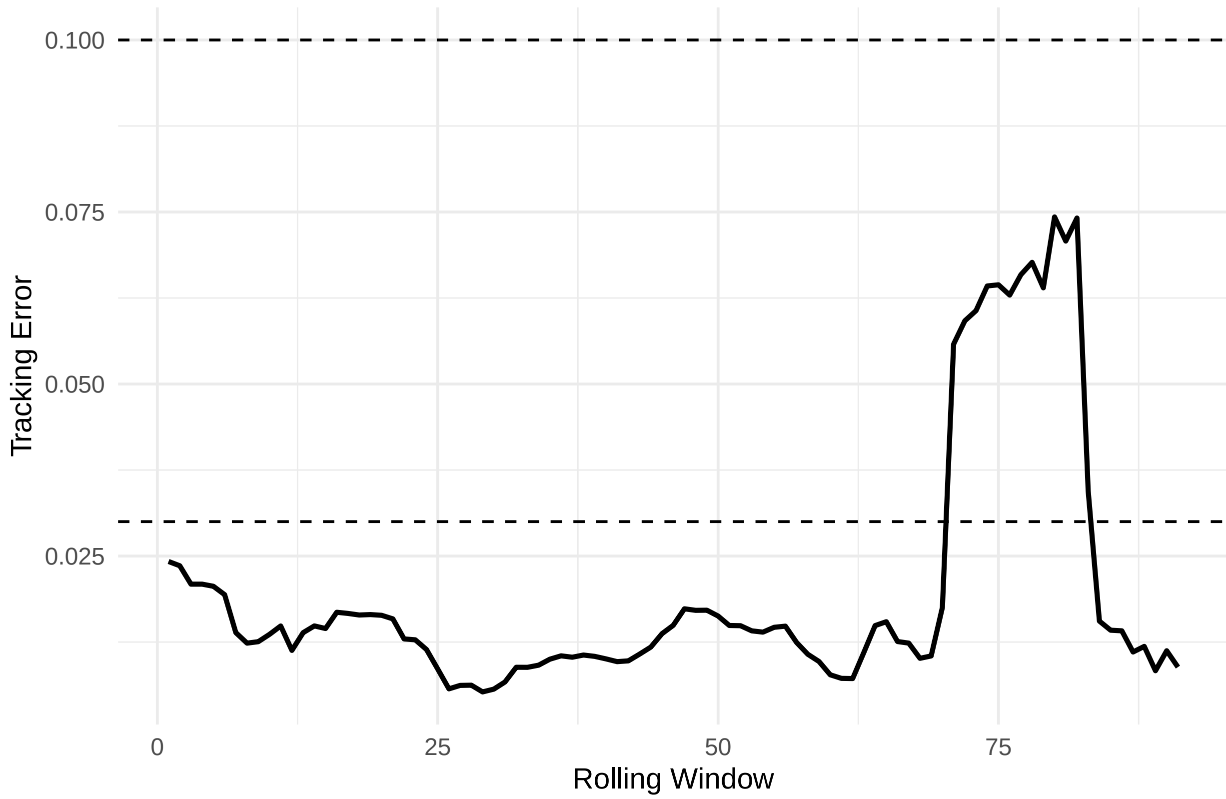
```
    breaks = seq(min(weights_df$Window), max(weights_df$Window), by = 5)
  ) +
  labs(
    title = "",
    x = "Rolling Window Index",
    y = "Asset",
    fill = "Weight"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



2.11 Tracking Error

```
### Tracking Error Plot
t.err_df <- data.frame(
  Window = 1:length(results),
  TrackingError = sapply(results, function(x) x$t.err)
```

```r
)
# Rolling Window Tracking Error vs ALSI
ggplot(t.err_df, aes(x = Window, y = TrackingError)) +
  geom_line(linewidth = 0.9) +
  geom_hline(yintercept = 0.03, linetype="dashed") +
  geom_hline(yintercept = 0.10, linetype="dashed") +
  labs(title = "",
       x = "Rolling Window", y = "Tracking Error") +
  theme_minimal()
```



References

Gebbie, T. (2025a). *PortfolioTheory-backtest-001.r*. Unpublished teaching material.

Gebbie, T. (2025b). *PortfolioTheoryLecture001.mlx*. Unpublished teaching material.

Gebbie, T. (2025c). *PortfolioTheoryLecture003.pdf*. Unpublished teaching material.

Gebbie, T. (2025d). *PortfolioTheory-PrepareData-000.r*. Unpublished teaching material.