Portfolio Theory: Assignment 1

Appendix B: R Code

Nesan Naidoo : NDXNES005

2025-09-21

PART II : Backtest Performance of the Tangency Portfolio

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).

Experiment 1 : In-Sample and Out-Of-Sample Sharpe Ratios

1.1 Libraries (Gebbie, 2025d)

```r
knitr::opts_chunk$set(
  warning = FALSE,
  message = FALSE,
  fig.width  = 7,
  fig.height = 5
)

suppressPackageStartupMessages({
  library(openxlsx)
  library(timeSeries)
  library(xts)
  library(zoo)
  library(matrixStats)
  library(quadprog)
  library(knitr)
  library(dplyr)
  library(ggplot2)
  library(tidyr)
})
```

1.2 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
```

```
# 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]
```

```r
  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
```

1.3 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
```

```
#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)

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

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

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

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

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

```
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"
```

1.4 Arithmetic Returns (Gebbie, 2025c)

```
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
```

```
# 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))

#  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
```

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

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

```
#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"
```

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

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

1.5 Tangency Portfolio (specifications: fully invested,no short-selling)(Gebbie, 2025c, 2025d)

```
tan.port <- function(mu, Sigma, rf=0){
  mu <- as.numeric(mu)
  Sigma <- as.matrix(Sigma)
  valid_idx <- which(!is.na(mu) & rowSums(is.na(Sigma)) == 0 & colSums(is.na(Sigma)) ==
  mu <- mu[valid_idx]
  Sigma <- Sigma[valid_idx, valid_idx]
  n <- length(mu)
  if(n == 0) stop("No valid assets to optimize. Check mu and Sigma.")
  # positive definite covariance
  Sigma <- Sigma + diag(1e-6, n)
  #maximize Sharpe ratio
  Dmat <- 2 * Sigma
  dvec <- rep(0, n)
  # Constraints which are sum(w) = 1 and w >= 0
  Amat <- cbind(rep(1, n), diag(n))
  bvec <- c(1, rep(0, n))
  meq  <- 1
  sol <- solve.QP(Dmat, dvec, Amat, bvec, meq)
  w <- sol$solution
  w[w < 1e-8] <- 0
  w <- w / sum(w)
  port_mean <- sum(w * mu)
  port_var  <- as.numeric(t(w) %*% Sigma %*% w)
  sharpe    <- (port_mean - rf) / sqrt(port_var)

  list(weights = w, mean = port_mean, var = port_var, sharpe = sharpe)
}
```

1.6 In-Sample and Out-of-Sample Split

```
tot.months <- nrow(rets_opt)
train.r <- 0.7
train.m <- floor(tot.months * train.r)
test.m  <- tot.months - train.m
# indices
train_idx <- 1:train.m
tst.idx  <- (train.m+1):tot.months
#returns
train_rets <- rets_opt[train_idx, ]
tst.rets  <- rets_opt[tst.idx, ]
# fill missing data using locf
train_rets <- na.locf(train_rets, na.rm=FALSE)
train_rets <- na.locf(train_rets, fromLast=TRUE)
tst.rets  <- na.locf(tst.rets, na.rm=FALSE)
```

```r
tst.rets  <- na.locf(tst.rets, fromLast=TRUE)
#only assets with valid data
train_rets <- train_rets[, colSums(!is.na(train_rets)) > 0, drop=FALSE]
tst.rets  <- tst.rets[, colnames(train_rets), drop=FALSE]
# Risk-free rates
rf_train <- if(!is.null(rets_cash)) mean(rets_cash[train_idx,], na.rm=TRUE) else 0
rf_test  <- if(!is.null(rets_cash)) mean(rets_cash[tst.idx,], na.rm=TRUE) else 0
# Tangency portfolio
mu_train <- colMeans(train_rets, na.rm=TRUE)
Sigma_train <- cov(train_rets, use="complete.obs")

tang <- tan.port(mu=mu_train, Sigma=Sigma_train, rf=rf_train)
w_hat <- tang$weights
if(is.null(w_hat)) stop("Tangency portfolio weights are NULL. Check your data!")
```

1.7 In-Sample & Out-of-Sample Portfolio Stats

```r
#Portfolio Returns with exact monthly rf
# Portfolio returns
port_train <- as.numeric(train_rets %*% w_hat)
port_test  <- as.numeric(tst.rets %*% w_hat)

# monthly risk-free series
rf_train_series <- if(!is.null(rets_cash)) as.numeric(rets_cash[train_idx, ]) else rep(
rf_tst.series  <- if(!is.null(rets_cash)) as.numeric(rets_cash[tst.idx, ]) else rep(0, 

summary_df <- data.frame(
  Period = c("In-Sample", "Out-of-Sample"),
  Mean = c(mean(port_train), mean(port_test)),
  Variance = c(var(port_train), var(port_test)),
  Sharpe = c(mean(port_train - rf_train_series, na.rm=TRUE)/sd(port_train - rf_train_ser
             mean(port_test  - rf_tst.series,  na.rm=TRUE)/sd(port_test  - rf_tst.series
)

knitr::kable(summary_df, digits=6, caption="In-Sample vs Out-of-Sample Portfolio Statist
```

Table 1: In-Sample vs Out-of-Sample Portfolio Statistics (Exact rf)

Period	Mean	Variance	Sharpe
In-Sample	0.004628	0.000316	-0.064860
Out-of-Sample	0.006642	0.000415	0.082794

1.8 Buy-and-Hold portfolio weights table

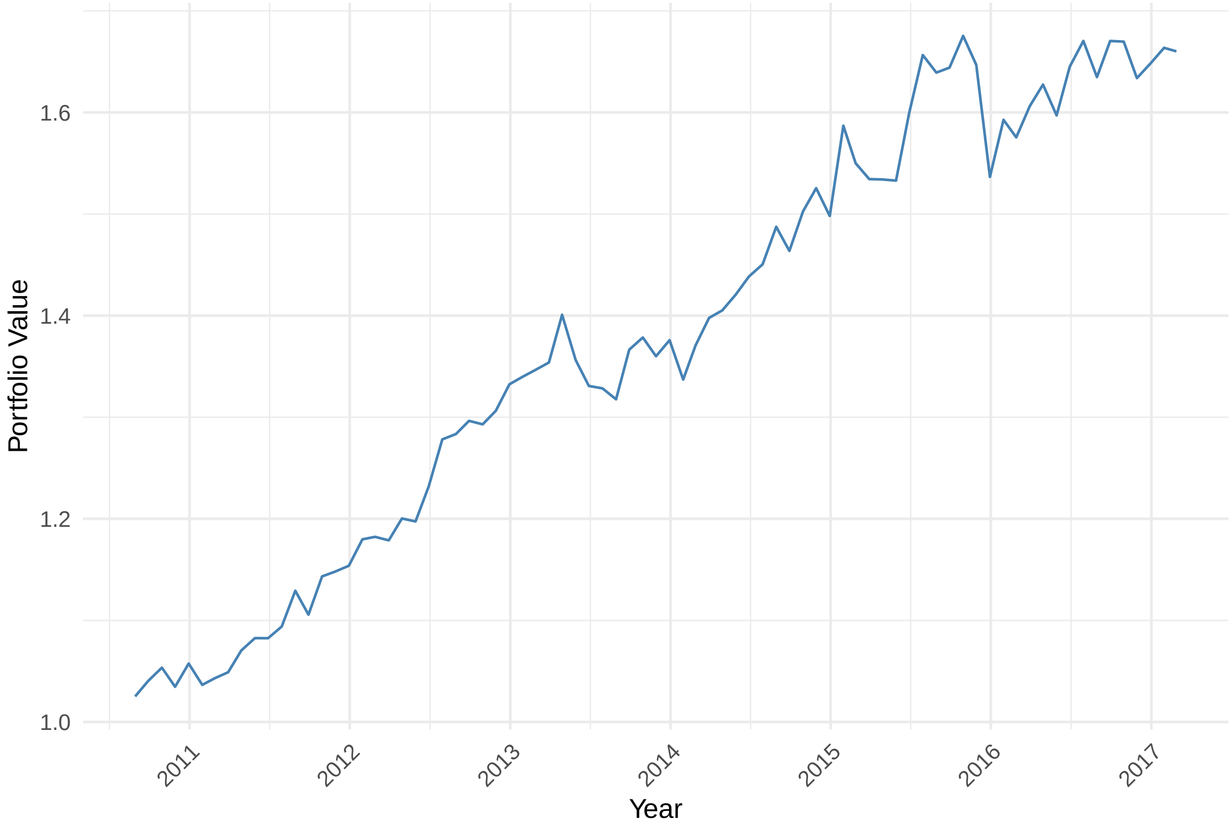
```r
# Only the assets that were used in optimization
assets_used <- colnames(train_rets)  #
weights_df <- data.frame(
  Asset  = assets_used,
  Weight = w_hat
)
knitr::kable(weights_df, digits=6, caption="Tangency Portfolio Weights (Buy-and-Hold)")
```

Table 2: Tangency Portfolio Weights (Buy-and-Hold)

Asset	Weight
ALBI	0.924960
J500	0.046595
J510	0.012100
J520	0.000000
J530	0.016344
J540	0.000000
J550	0.000000
J560	0.000000
J580	0.000000
J590	0.000000
ALSI	0.000000

1.9 Cumulative wealth -Buy and Hold

```r
tst.dates <- as.Date(index(rets_xts[tst.idx, ]))
# cumulative wealth
port_cum <- cumprod(1 + port_test)
plot_df <- data.frame(Date = tst.dates, Cumulative_Wealth = port_cum)

#Buy-and-Hold Portfolio Cumulative Wealth (OOS)
ggplot(plot_df, aes(x = Date, y = Cumulative_Wealth)) +
  geom_line(color = "steelblue") +
  scale_x_date(date_labels = "%Y", date_breaks = "1 year") +  # monthly breaks
  labs(title = "",
       x = "Year", y = "Portfolio Value") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



1.10 Final cumulative return

```r
final_ret <- tail(port_cum, 1)
tang_ret  <- prod(1 + tst.rets %*% w_hat) - 1
BH_summary <- data.frame(
  Window = 1,
  BH_Return = final_ret,
  Tangency_Expected = tang_ret
)
knitr::kable(BH_summary, digits=4, caption="Buy-and-Hold Cumulative Return vs Tangency E
```

Table 3: Buy-and-Hold Cumulative Return vs Tangency Expected Return

Window	BH_Return	Tangency_Expected
1	1.6601	0.6601

Experiment 2 : Out-Of-Sample Backtesting using a Rolling Window

2.1 Libraries (Gebbie, 2025d)

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

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
```

```
# 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), ]
```

```r
# 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)

# 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))

#  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
```

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

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

```
#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"
```

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

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

2.5 Tangency Portfolio (Gebbie, 2025c, 2025d)

```
tan.port <- function(mu, Sigma, rf=0, targets=seq(0.001,0.05,length.out=200)){
  n <- length(mu)
  Dmat <- 2*(Sigma + diag(1e-8,n))  # ensure PD covariance
  best <- list(sharpe=-Inf)

  for(tgt in targets){
    gamma <- as.numeric(mu - rf)
    Amat <- cbind(gamma, -gamma, rep(1,n), -rep(1,n), diag(n))
    bvec <- c(tgt, -tgt, 1, -1, rep(0,n))
    sol <- try(solve.QP(Dmat, dvec=rep(0,n), Amat=as.matrix(Amat), bvec=bvec, meq=0), si
    if(inherits(sol,"try-error")) next
    w <- sol$solution
    w[w<1e-8] <- 0
    if(sum(w)<=0) next
    w <- w/sum(w)
    port_mean <- sum(w*mu)
    port_var  <- as.numeric(t(w) %*% Sigma %*% w)
    if(port_var<=0) next
    sr <- (port_mean - rf)/sqrt(port_var)
    if(sr > best$sharpe) best <- list(weights=w, target=tgt, sharpe=sr, mean=port_mean,
  }
  return(best)
}
```

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

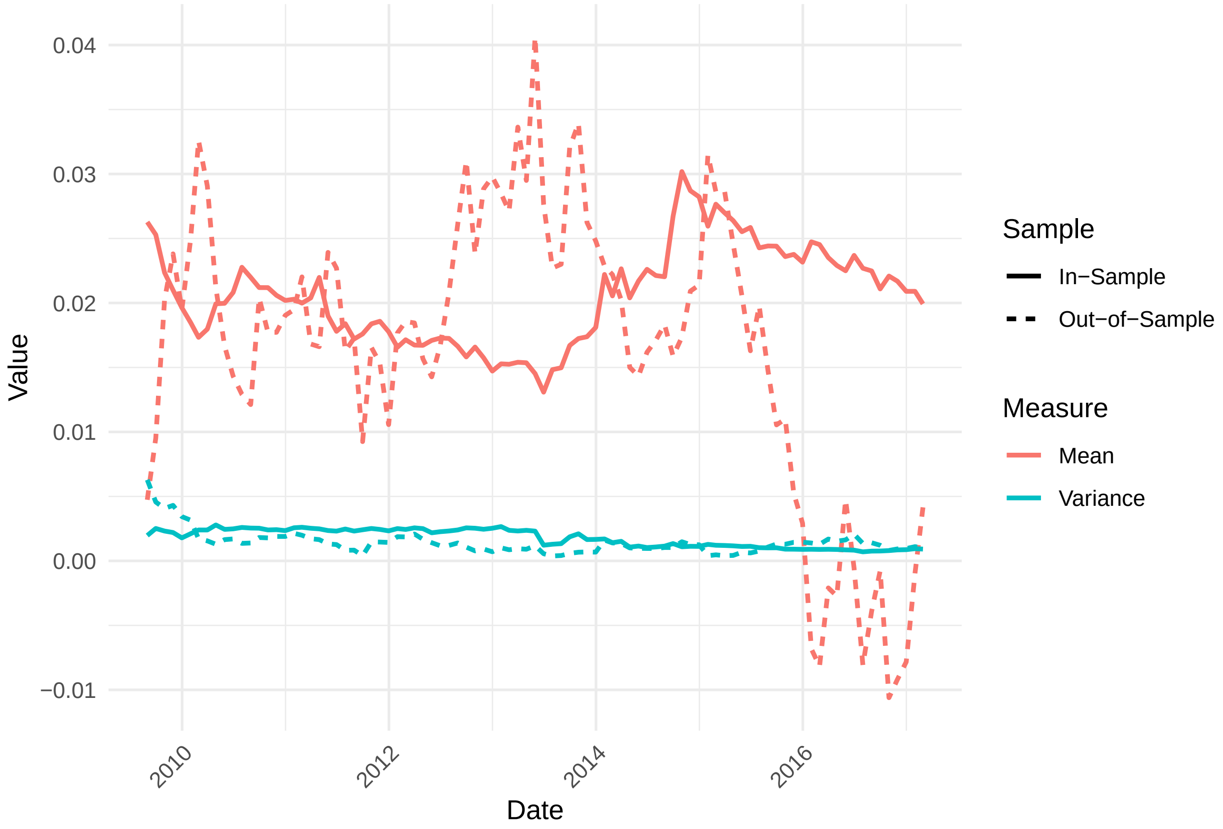
```
train.m <- 60 # 5 year period
test.m  <- 12 # 1 year period
roll_step <- 1 #1 month increments
n_obs <- nrow(rets_opt)
start_idxs <- seq(1, n_obs - train.m - test.m + 1, by=roll_step)
results <- list()

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]

  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")

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

  # Skip invalid windows
  if(any(!is.finite(mu_train)) || !is.finite(rf_train)) next
  upper_targ <- max(0.06, max(mu_train, na.rm=TRUE) - rf_train)
  if(!is.finite(upper_targ) || upper_targ <= 0) next
  targ_grid <- seq(0.0005, upper_targ, length.out=300)
  tang <- tan.port(mu=mu_train, Sigma=Sigma_train, rf=rf_train, targets=targ_grid)
  if(is.null(tang$weights)) next

  w_hat <- tang$weights
  port_train <- as.numeric(as.matrix(train_rets) %*% w_hat)
  port_test  <- as.numeric(as.matrix(tst.rets) %*% w_hat)


  results[[length(results)+1]] <- list(
    train_period = paste(index(train_rets)[1], index(train_rets)[nrow(train_rets)], sep=
    tst.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)),
    weights = w_hat,
    assets  = colnames(rets_opt)
  )
}

summary_df <- do.call(rbind, lapply(results, function(x) data.frame(
  train=x$train_period, test=x$tst.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
)))
knitr::kable(
  head(summary_df, 15),
  digits = 4,
  caption = "In-sample vs Out-of-sample Portfolio Statistics"
)
```

Table 4: In-sample vs Out-of-sample Portfolio Statistics

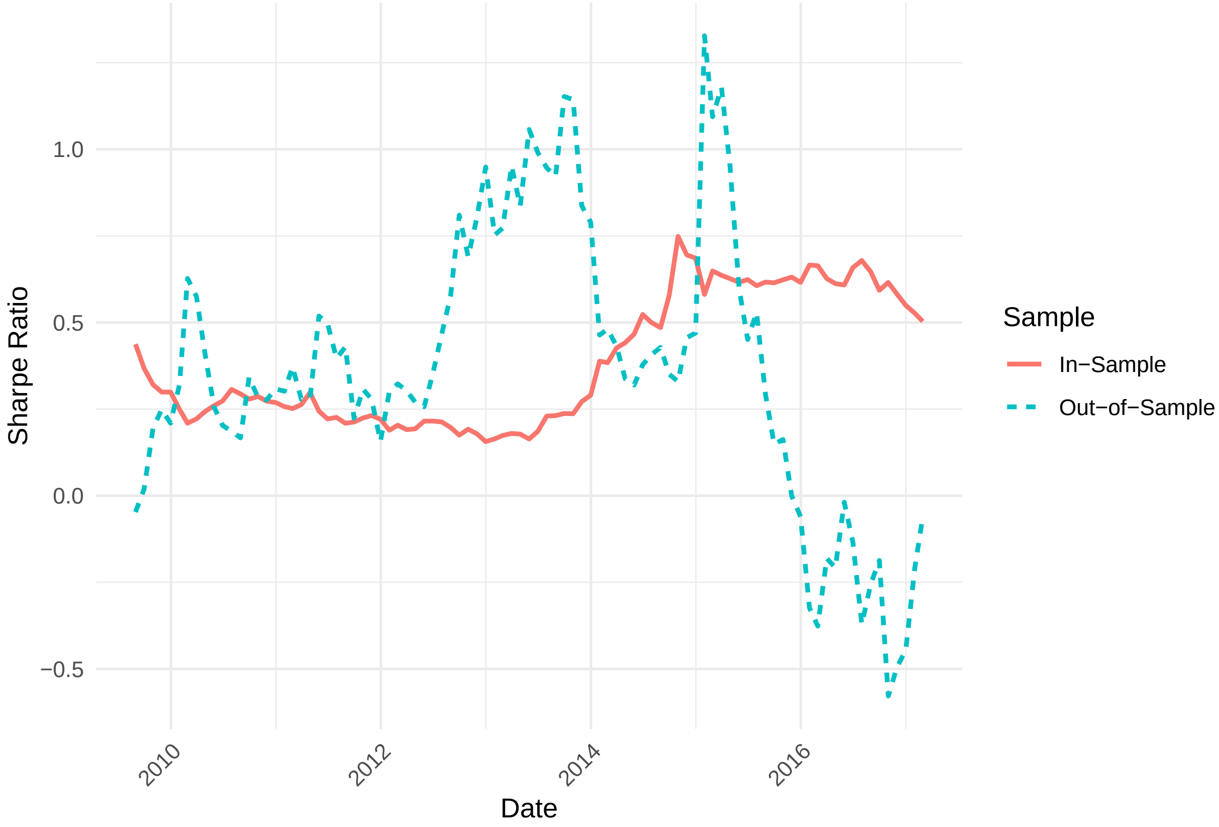
train	test	mu_IS	var_IS	SR_IS	mu_OOS	var_OOS	SR_OOS
2003-09-30 / 2008-08-31	2008-09-30 / 2009-08-31	0.0263	0.0020	0.4373	0.0047	0.0063	-0.0473
2003-10-31 / 2008-09-30	2008-10-31 / 2009-09-30	0.0253	0.0025	0.3671	0.0095	0.0046	0.0195
2003-11-30 / 2008-10-31	2008-11-30 / 2009-10-31	0.0224	0.0023	0.3205	0.0203	0.0041	0.1956
2003-12-31 / 2008-11-30	2008-12-31 / 2009-11-30	0.0210	0.0022	0.2992	0.0238	0.0043	0.2486
2004-01-31 / 2008-12-31	2009-01-31 / 2009-12-31	0.0196	0.0018	0.2992	0.0195	0.0034	0.2093
2004-02-29 / 2009-01-31	2009-02-28 / 2010-01-31	0.0185	0.0021	0.2492	0.0249	0.0032	0.3206
2004-03-31 / 2009-02-28	2009-03-31 / 2010-02-28	0.0173	0.0024	0.2093	0.0326	0.0017	0.6269
2004-04-30 / 2009-03-31	2009-04-30 / 2010-03-31	0.0180	0.0024	0.2216	0.0291	0.0016	0.5750
2004-05-31 / 2009-04-30	2009-05-31 / 2010-04-30	0.0199	0.0028	0.2431	0.0212	0.0013	0.4085
2004-06-30 / 2009-05-31	2009-06-30 / 2010-05-31	0.0200	0.0024	0.2597	0.0167	0.0017	0.2565
2004-07-31 / 2009-06-30	2009-07-31 / 2010-06-30	0.0208	0.0025	0.2735	0.0143	0.0017	0.2026
2004-08-31 / 2009-07-31	2009-08-31 / 2010-07-31	0.0228	0.0026	0.3068	0.0129	0.0014	0.1862
2004-09-30 / 2009-08-31	2009-09-30 / 2010-08-31	0.0220	0.0025	0.2939	0.0121	0.0014	0.1669
2004-10-31 / 2009-09-30	2009-10-31 / 2010-09-30	0.0212	0.0025	0.2787	0.0204	0.0018	0.3413
2004-11-30 / 2009-10-31	2009-11-30 / 2010-10-31	0.0212	0.0024	0.2864	0.0177	0.0018	0.2836

Plotting In Sample vs Out Of Sample Statistics

```r
p.dates <- as.Date(sapply(summary_df$test, function(x) {
  tail(strsplit(x, " / ")[[1]], 1)
}), format = "%Y-%m-%d")
summary_df$Date <- p.dates

p.long <- summary_df |>
  select(Date, mu_IS, var_IS, SR_IS, mu_OOS, var_OOS, SR_OOS) |>
  pivot_longer(-Date, names_to = "Metric", values_to = "Value") |>
  mutate(
    Type   = ifelse(grepl("_IS", Metric), "In-Sample", "Out-of-Sample"),
    Metric = gsub("_(IS|OOS)", "", Metric)
  ) |>
  mutate(
    Metric = case_when(
      Metric == "mu"  ~ "Mean",
      Metric == "var" ~ "Variance",TRUE ~ Metric
    ))

# Mean and Variance In-Sample vs Out-of-Sample Mean & Variance
p1<- p.long |>
  filter(Metric %in% c("Mean", "Variance")) |>
  ggplot(aes(x = Date, y = Value, color = Metric, linetype = Type)) +
  geom_line(linewidth = 0.9) +
  labs(
    title = "",
    x = "Date", y = "Value",
    color = "Measure", linetype = "Sample"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Sharpe Ratios In-Sample vs Out-of-Sample Sharpe Ratios
p2 <- p.long |>
  filter(Metric == "SR") |>
  ggplot(aes(x = Date, y = Value, color = Type, linetype = Type)) +
  geom_line(linewidth = 0.9) +
  labs(
    title = "",
    x = "Date", y = "Sharpe Ratio",
    color = "Sample", linetype = "Sample"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

p1
```



```
p2
```

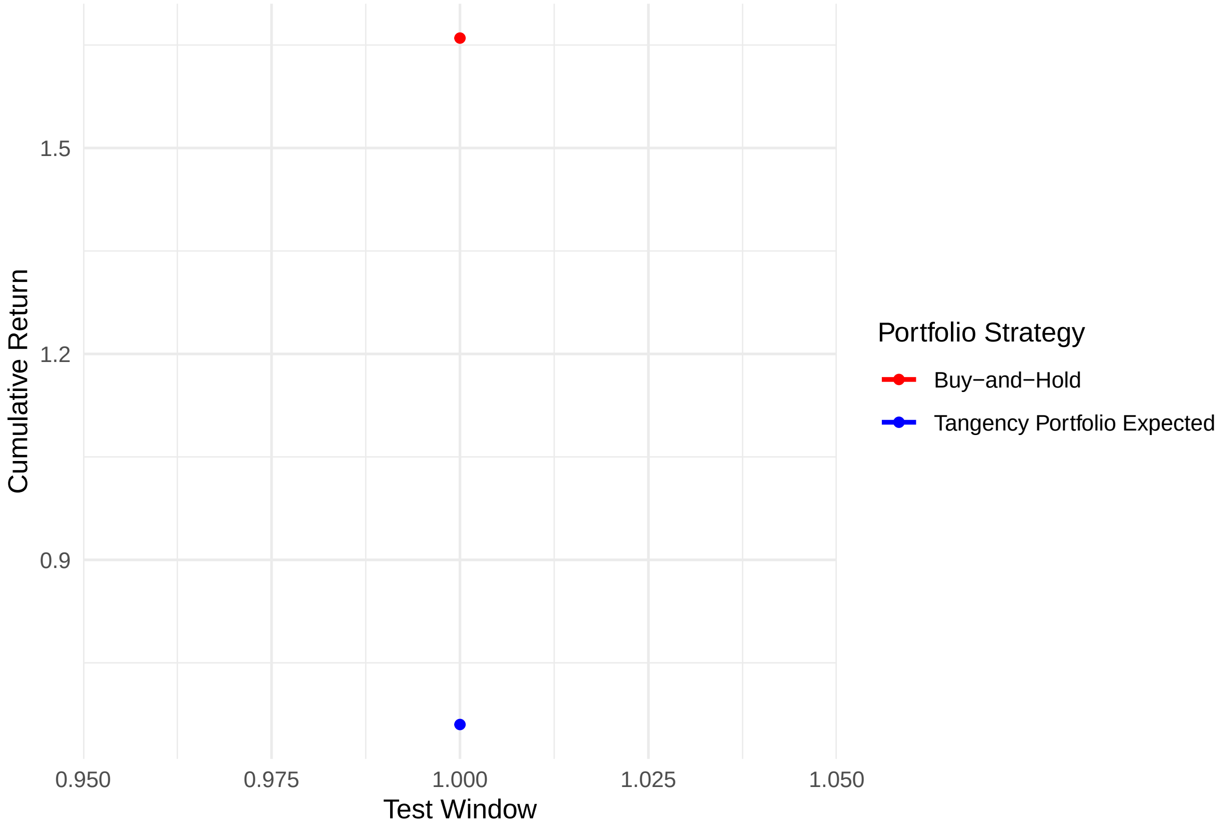


2.7 Buy and Hold Portfolio Simulation

```
# This section calculates portfolio returns, update weights due to price changes, reno

BH_results <- list()

for(idx in seq_along(results)) {

  # test period dates from results
  tst.period <- results[[idx]]$tst.period
  tst.start <- as.Date(substr(tst.period,1,10))
  tst.end   <- as.Date(substr(tst.period,14,23))
 tst.rets <- rets_opt[as.Date(index(rets_opt)) >= tst.start &
                        as.Date(index(rets_opt)) <= tst.end, ,drop=FALSE]

  w0 <- results[[idx]]$weights
  n_assets <- ncol(tst.rets)
  n_obs <- nrow(tst.rets)

  # if tst.rets is empty
  if(n_obs == 0) next

  Wts <- matrix(0, nrow=n_obs, ncol=n_assets)
  portRet <- numeric(n_obs)
  portPrc <- numeric(n_obs)
  portPrc[1] <- 1
  Wts[1,] <- w0

  for(t in 1:n_obs){
    portRet[t] <- sum(Wts[t,] * as.numeric(tst.rets[t,]))
    portPrc[t] <- ifelse(t==1, 1*(1+portRet[t]), portPrc[t-1]*(1+portRet[t]))

    if(t < n_obs){
      Wts[t+1,] <- Wts[t,] * (1 + as.numeric(tst.rets[t,]))
      Wts[t+1,] <- Wts[t+1,] / sum(Wts[t+1,])
    }
  }

  BH_results[[idx]] <- list(
    weights=Wts,
    asset_names=colnames(tst.rets),
    portPrc=portPrc,
    portRet=portRet,
    dates=index(tst.rets)
  )
}
```

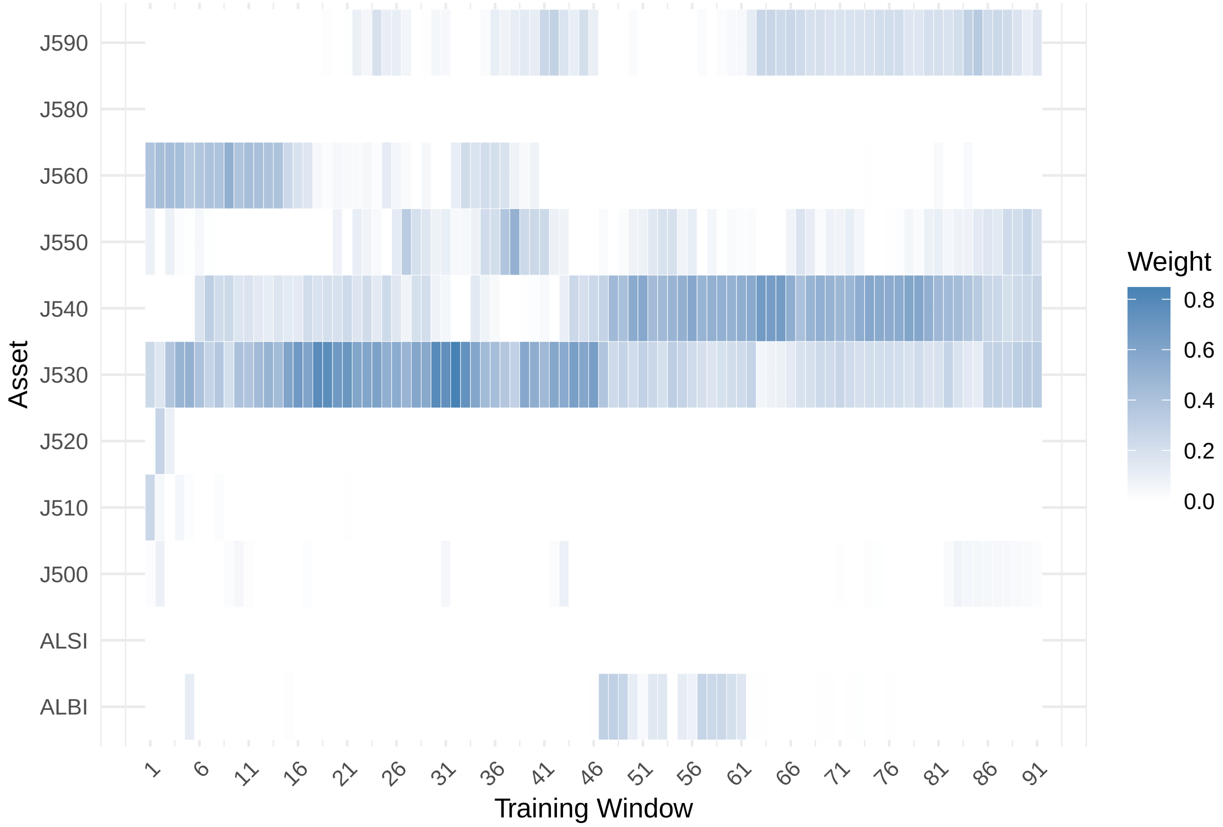
2.8 Final cumulative return

```
plot_df <- pivot_longer(BH_summary, cols = c(BH_Return, Tangency_Expected),names_to = "S

ggplot(plot_df, aes(x = Window, y = Value, color = Strategy)) +
  geom_line(linewidth = 0.9) +
  geom_point(size = 1.5) +
  labs(
    title = "",
    x = "Test Window", y = "Cumulative Return",
    color = "Portfolio Strategy"
  ) +
  scale_color_manual(
    values = c("BH_Return" = "red", "Tangency_Expected" = "blue"),
    labels = c("BH_Return" = "Buy-and-Hold", "Tangency_Expected" = "Tangency Portfolio E
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 0, hjust = 0.5))
```



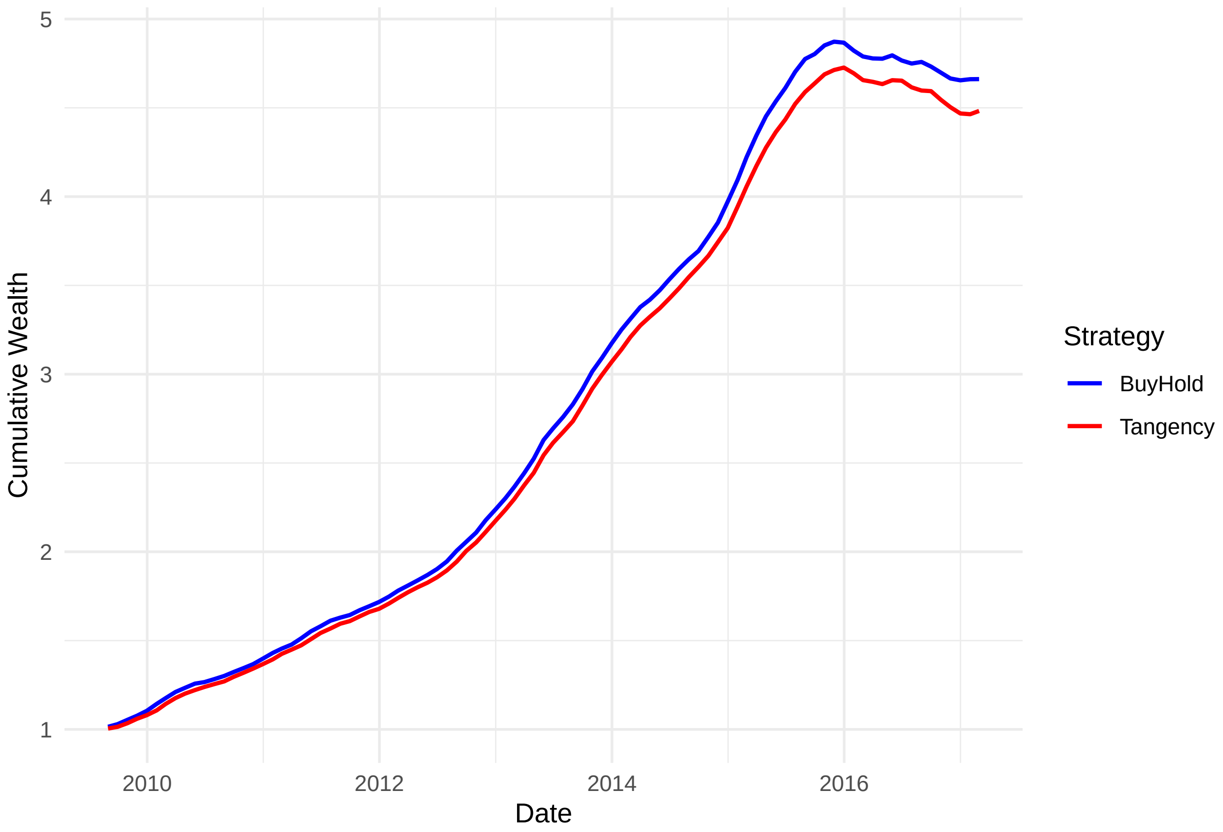
2.9 Tangency Portfolio Weights using heatmap

```r
weights_df <- do.call(rbind, lapply(seq_along(results), function(i) {
  n_assets <- length(results[[i]]$weights)
  data.frame(
    Window = i, # numeric window index
    Asset  = results[[i]]$assets, #asset names
    Weight = results[[i]]$weights,#corresponding weights
    stringsAsFactors = FALSE
  )
}))

 # Tangency Portfolio Weights Evolution
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="Training Window", y="Asset", fill="Weight"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```



2.10 Cumulative Out-of-Sample Performance vs Buy Hold

```
#out-of-sample returns
OOS_ret <- sapply(results, function(x) {
  val <- x$mu_OOS
  if(is.null(val) || !is.finite(val)) return(NA)
  as.numeric(val)
})
#remove NAs
OOS_ret <- OOS_ret[!is.na(OOS_ret)]
OOS_ret <- as.numeric(OOS_ret)
#Calculate cumulative wealth
cum_OOS <- cumprod(1 + OOS_ret)
#  Buy-and-Hold
BH_ret <- sapply(BH_results, function(x) {
  if(is.null(x$portRet)) return(NA)
  mean(as.numeric(x$portRet), na.rm=TRUE)
})
BH_ret <- BH_ret[!is.na(BH_ret)]
BH_ret <- as.numeric(BH_ret)
cum_BH <- cumprod(1 + BH_ret)

## keeps results with valid numeric data
val.res <- results[sapply(results, function(x) !is.null(x$mu_OOS) && is.finite(x$mu_OOS

#end-of-test-period dates as character
p.dates_char <- sapply(val.res, function(x) {
  tail(strsplit(x$tst.period, " / ")[[1]], 1)
})

p.dates <- as.Date(unlist(p.dates_char), format="%Y-%m-%d")
plot_df <- data.frame(
  Date     = p.dates,
  Tangency = cum_OOS,
  BuyHold  = cum_BH[1:length(cum_OOS)]
)

df.long <- pivot_longer(plot_df, cols=c("Tangency","BuyHold"),
                             names_to="Strategy", values_to="Cumulative_Wealth")

# Cumulative Out-of-Sample Performance
ggplot(df.long, aes(x=Date, y=Cumulative_Wealth, color=Strategy)) +
  geom_line(linewidth=0.8) +
  labs(title="",
       x="Date", y="Cumulative Wealth", color="Strategy") +
  scale_color_manual(values=c("blue","red")) +
  theme_minimal()
```


Cumulative Wealth
5
4
3
2
1
2010
2012
2014
2016
Date
Strategy
BuyHold
Tangency

References

Bailey, D. H., & López de Prado, M. (2014). The deflated sharpe ratio:correcting for selection bias, BacktestOverfitting, and non-normality. *The Journal of Portfolio Management*, *40*, 94–107. https://doi.org/10.3905/jpm.2014.40.5.094

Bailey, D., Borwein, J., López de Prado, M., & Zhu, Q. J. (2016). The probability of backtest overfitting. *The Journal of Computational Finance*. https://doi.org/10.21314/jcf.2016.322

Embrechts, P., Klüppelberg, C., & Mikosch, T. (1997). *Modelling extremal events for insurance and finance*. Springer.

Fisher, R. A., & Tippett, L. H. C. (1928). *Limiting forms of the frequency distribution of the largest or smallest member of a sample* (Vol. 24, pp. 180–190). Cambridge University Press.

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.

Gnedenko, B. (2006). *On limit theorems for a random number of random variables* (pp. 167–176). Springer.

Lee, W. (2000). *Theory and methodology of tactical asset allocation*. John Wiley & Sons.

Liu, Y., Rekkas, M., & Wong, A. (2012). Inference for the sharpe ratio using a likelihood-based approach. *Journal of Probability and Statistics*, *2012*, 1–24. https://doi.org/10.1155/2012/878561

Lo, A. W. (2002). The statistics of sharpe ratios. *Financial Analysts Journal*, *58*, 36–52. https://doi.org/10.2469/faj.v58.n4.2453

Resnick, S. I. (2008). *Extreme values, regular variation and point processes*. Springer, Cop.