FRE7241 Algorithmic Portfolio Management

Lecture#3, Fall 2023

Jerzy Pawlowski *jp3900@nyu.edu*

NYU Tandon School of Engineering

September 19, 2023

NYU | TANDON SCHOOL OF ENGINEERING

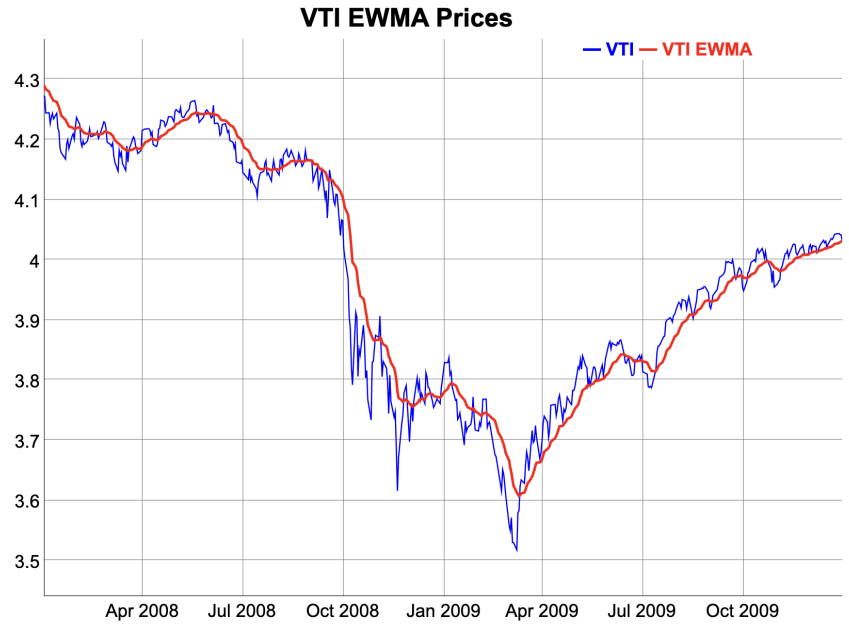
EWMA Price Technical Indicator

The *Exponentially Weighted Moving Average Price* (*EWMA*) is defined as the weighted average of prices over a rolling interval:

$$p_i^{EWMA} = (1 - \lambda) \sum_{j=0}^{\infty} \lambda^j p_{i-j}$$

Where the decay parameter λ determines the rate of decay of the *EWMA* weights, with smaller values of λ producing faster decay, giving more weight to recent prices, and vice versa.

The function `HighFreq::roll_wsum()` calculates the convolution of a time series with a vector of weights.




```
> # Extract the log VTI prices
> ohlc <- log(rutils::etfenv$VTI)
> closep <- quantmod::Cl(ohlc)
> colnames(closep) <- "VTI"
> nrows <- NROW(closep)
> # Calculate the EWMA weights
> look_back <- 111
> lambda <- 0.9
> weightv <- lambda^(0:look_back)
> weightv <- weightv/sum(weightv)
> # Calculate the EWMA prices as a convolution
> ewmacpp <- HighFreq::roll_sumw(closep, weightv=weightv)
> pricev <- cbind(closep, ewmacpp)
> colnames(pricev) <- c("VTI", "VTI EWMA")
```

```
> # Dygraphs plot with custom line colors
> colnamev <- colnames(pricev)
> dygraphs::dygraph(pricev["2008/2009"], main="VTI EWMA Prices") %>%
+   dySeries(name=colnamev[1], strokeWidth=1, col="blue") %>%
+   dySeries(name=colnamev[2], strokeWidth=2, col="red") %>%
+   dyLegend(show="always", width=200)
> # Standard plot of  EWMA prices with custom line colors
> x11(width=6, height=5)
> plot_theme <- chart_theme()
> colorv <- c("blue", "red")
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(pricev["2009"], theme=plot_theme,
+        lwd=2, name="VTI EWMA Prices")
> legend("topleft", legend=colnames(pricev), y.intersp=0.5,
+  inset=0.1, bg="white", lty=1, lwd=6, cex=0.8,
+  col=plot_theme$col$line.col, bty="n")
```

Recursive EWMA Price Indicator

The *EWMA* prices can be calculated recursively as follows:

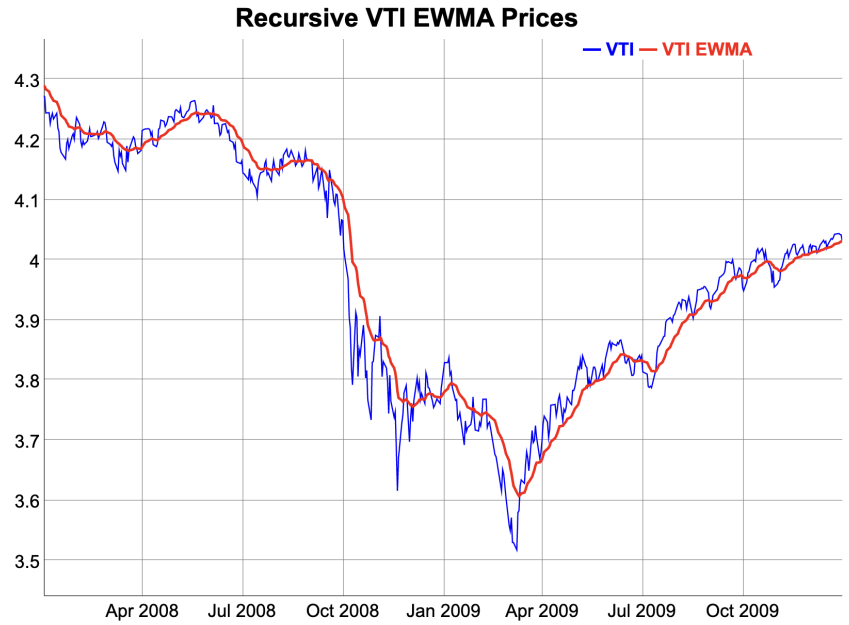
$$p_i^{EWMA} = (1 - \lambda)p_i + \lambda p_{i-1}^{EWMA}$$

Where the decay parameter λ determines the rate of decay of the *EWMA* weights, with smaller values of λ producing faster decay, giving more weight to recent prices, and vice versa.

The recursive *EWMA* prices are slightly different from those calculated as a convolution, because the convolution uses a fixed look-back interval.

The compiled C++ function `stats:::C_rfilter()` calculates the exponentially weighted moving average prices recursively.

The function `HighFreq::run_mean()` calculates the exponentially weighted moving average prices recursively.




```
> # Calculate the EWMA prices recursively using C++ code
> ewmar <- .Call(stats:::C_rfilter, closep, lambda, c(as.numeric(cl
> # Or R code
> # ewmar <- filter(closep, filter=lambda, init=as.numeric(closep[1
> ewmar <- (1-lambda)*ewmar
> # Calculate the EWMA prices recursively using RcppArmadillo
> ewmacpp <- HighFreq::run_mean(closep, lambda=lambda)
> all.equal(drop(ewmacpp), ewmar)
> # Compare the speed of C++ code with RcppArmadillo
> library(microbenchmark)
> summary(microbenchmark(
+   Rcpp=HighFreq::run_mean(closep, lambda=lambda),
+   rfilter=.Call(stats:::C_rfilter, closep, lambda, c(as.numeric(c
```

```
> # Dygraphs plot with custom line colors
> pricev <- cbind(closep, ewmacpp)
> colnames(pricev) <- c("VTI", "VTI EWMA")
> colnamev <- colnames(pricev)
> dygraphs::dygraph(pricev["2008/2009"], main="Recursive VTI EWMA Pr
+   dySeries(name=colnamev[1], strokeWidth=1, col="blue") %>%
+   dySeries(name=colnamev[2], strokeWidth=2, col="red") %>%
+   dyLegend(show="always", width=200)
> # Standard plot of  EWMA prices with custom line colors
> plot_theme <- chart_theme()
> colorv <- c("blue", "red")
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(pricev["2009"], theme=plot_theme,
+        lwd=2, name="VTI EWMA Prices")
> legend("topleft", legend=colnames(pricev), y.intersp=0.5,
+  inset=0.1, bg="white", lty=1, lwd=6, cex=0.8,
+  col=plot_theme$col$line.col, bty="n")
```

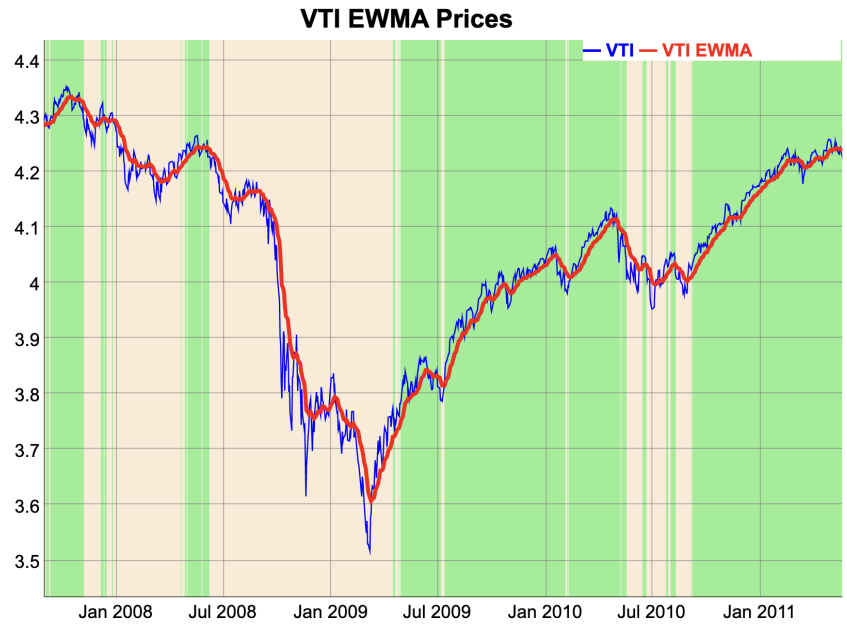
The EWMA Crossover Strategy

The trend following *EWMA Crossover* strategy switches its stock position depending if the current price is above or below the *EWMA*.

If the stock price is above the *EWMA* price, then the strategy switches to long \$1 dollar of stock, and if it is below, to short \$1 dollar of stock.

The strategy holds the same position until the *EWMA* crosses over the current price (either from above or below), and then it switches its position.

The strategy is therefore always either long \$1 dollar of stock or short \$1 dollar of stock.




```
> # Calculate the EWMA prices recursively using C++ code
> lambda <- 0.984
> ewmacpp <- HighFreq::run_mean(closep, lambda=lambda)
> # Calculate the positions, either: -1, 0, or 1
> indic <- sign(closep - ewmacpp)
> posv <- rutils::lagit(indic, lagg=1)
> # Create colors for background shading
> crossd <- (rutils::diffit(posv) != 0)
> shadev <- posv[crossd]
> crossd <- c(zoo::index(shadev), end(posv))
> shadev <- ifelse(drop(zoo::coredata(shadev)) == 1, "lightgreen",
> # Create dygraph object without plotting it
> dyplot <- dygraphs::dygraph(pricev, main="VTI EWMA Prices") %>%
+   dySeries(name=colnamev[1], strokeWidth=1, col="blue") %>%
+   dySeries(name=colnamev[2], strokeWidth=3, col="red") %>%
+   dyLegend(show="always", width=200)
```

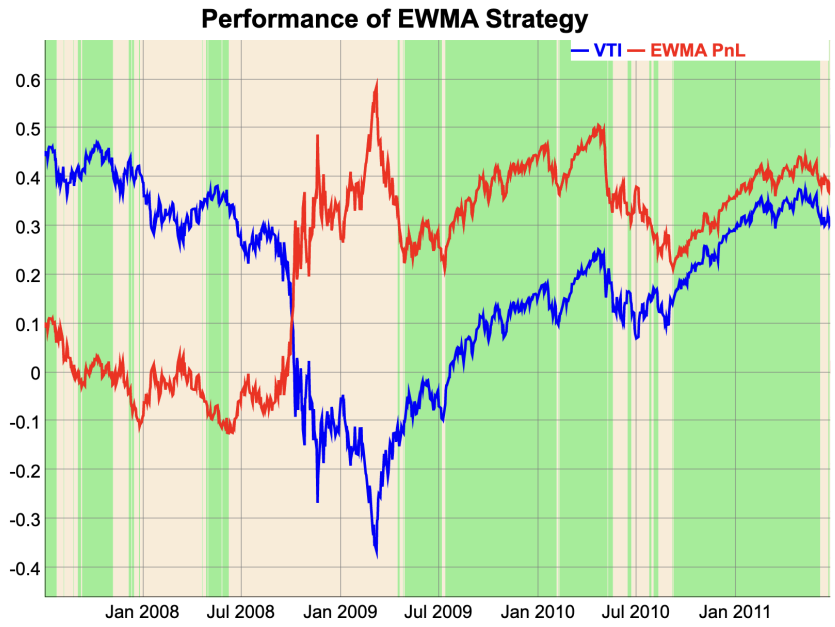
```
> # Add shading to dygraph object
> for (i in 1:NROW(shadev)) {
+   dyplot <- dyplot %>% dyShading(from=crossd[i], to=crossd[i+1], c
+ }  # end for
> # Plot the dygraph object
> dyplot
> # Standard plot of EWMA prices with position shading
> quantmod::chart_Series(pricev, theme=plot_theme,
+        lwd=2, name="VTI EWMA Prices")
> add_TA(posv > 0, on=-1, col="lightgreen", border="lightgreen")
> add_TA(posv < 0, on=-1, col="lightgrey", border="lightgrey")
> legend("topleft", legend=colnames(pricev),
+  inset=0.1, bg="white", lty=1, lwd=6, y.intersp=0.5,
+  col=plot_theme$col$line.col, bty="n")
```

EWMA Crossover Strategy Performance

The crossover strategy trades at the *Close* price on the same day that prices cross the *EWMA*, which may be difficult in practice.

The crossover strategy performance is worse than the underlying asset (*VTI*), but it has a negative correlation to it, which is very valuable when building a portfolio.

```
> # Calculate the daily profits and losses of EWMA strategy
> retp <- rutils::diffit(closep)  # VTI returns
> pnls <- retp*posv
> colnames(pnls) <- "EWMA"
> wealthv <- cbind(retp, pnls)
> colnames(wealthv) <- c("VTI", "EWMA PnL")
> # Annualized Sharpe ratio of EWMA strategy
> sqrt(252)*sapply(wealthv, function (x) mean(x)/sd(x))
> # The crossover strategy has a negative correlation to VTI
> cor(wealthv)[1, 2]
> # Plot dygraph of EWMA strategy wealth
> # Create dygraph object without plotting it
> colorv <- c("blue", "red")
> dyplot <- dygraphs::dygraph(cumsum(wealthv), main="Performance o
+   dyOptions(colors=colorv, strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
> # Add shading to dygraph object
> for (i in 1:NROW(shadev)) {
+   dyplot <- dyplot %>%
+     dyShading(from=crossd[i], to=crossd[i+1], color=shadev[i])
+ }  # end for
> # Plot the dygraph object
> dyplot
```



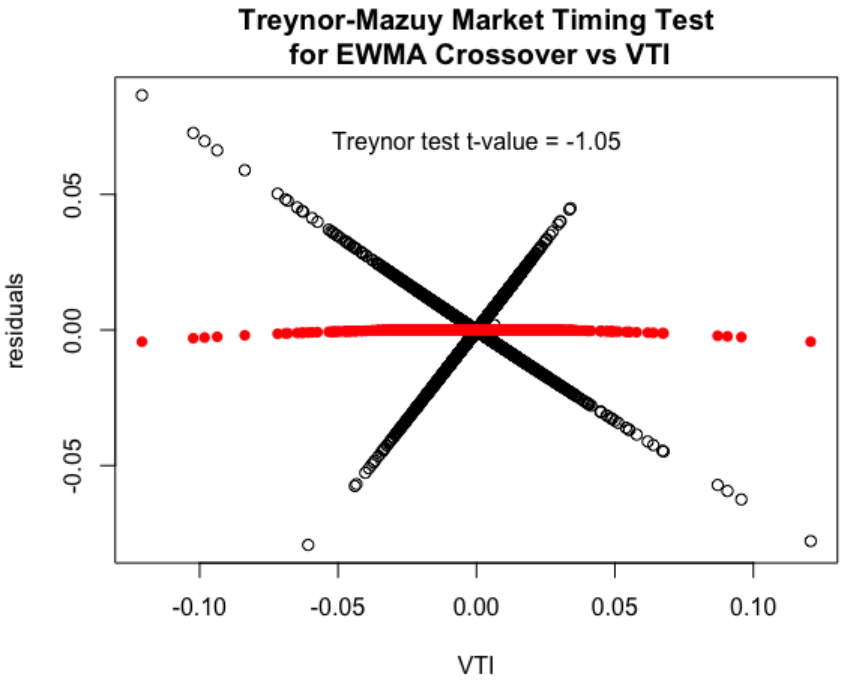

```
> # Standard plot of EWMA strategy wealth
> x11(width=6, height=5)
> plot_theme <- chart_theme()
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(cumsum(wealthv), theme=plot_theme,
+        name="Performance of EWMA Strategy")
> add_TA(posv > 0, on=-1, col="lightgreen", border="lightgreen")
> add_TA(posv < 0, on=-1, col="lightgrey", border="lightgrey")
> legend("top", legend=colnames(wealthv), y.intersp=0.5,
+  inset=0.05, bg="white", lty=1, lwd=6,
+  col=plot_theme$col$line.col, bty="n")
```

EWMA Crossover Strategy Market Timing Skill

The EWMA crossover strategy shorts the market during significant selloffs, but otherwise doesn't display market timing skill.

The t-value of the *Treynor-Mazuy* test is negative, but not statistically significant.

```
> # Test EWMA crossover market timing of VTI using Treynor-Mazuy tes
> desm <- cbind(pnls, retp, retp^2)
> desm <- na.omit(desm)
> colnames(desm) <- c("EWMA", "VTI", "treynor")
> regmod <- lm(EWMA ~ VTI + treynor, data=desm)
> summary(regmod)
> # Plot residual scatterplot
> resids <- (desm$EWMA - regmod$coeff["VTI"]*retp)
> resids <- regmod$residuals
> plot.default(x=retp, y=resids, xlab="VTI", ylab="residuals")
> title(main="Treynor-Mazuy Market Timing Test\n for EWMA Crossover
> # Plot fitted (predicted) response values
> coefreg <- summary(regmod)$coeff
> fitv <- regmod$fitted.values - coefreg["VTI", "Estimate"]*retp
> tvalue <- round(coefreg["treynor", "t value"], 2)
> points.default(x=retp, y=fitv, pch=16, col="red")
> text(x=0.0, y=0.8*max(resids), paste("Treynor test t-value =", tvalue))
```

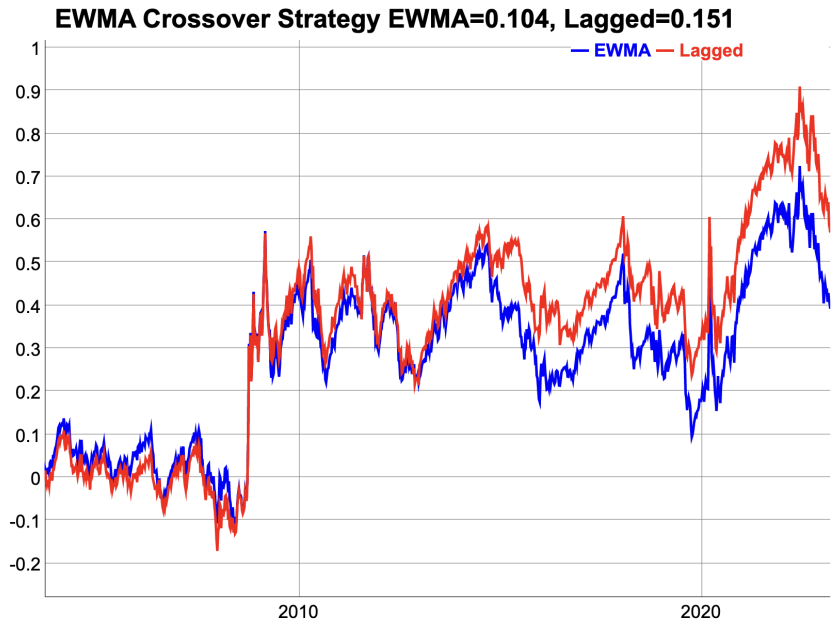


EWMA Crossover Strategy With Lag

The crossover strategy suffers losses when prices are range-bound without a trend, because whenever it switches position the prices soon change direction. (This is called a *"whipsaw"*.)

To prevent whipsaws and over-trading, the crossover strategy may choose to delay switching positions until the indicator repeats the same value for several periods.

There's a tradeoff between switching positions too early and risking a whipsaw, and waiting too long and missing an emerging trend.




```
> # Determine trade dates right after EWMA has crossed prices
> indic <- sign(closep - ewmacpp)
> # Calculate the positions from lagged indicator
> lagg <- 2
> indic <- HighFreq::roll_sum(indic, lagg)
> # Calculate the positions, either: -1, 0, or 1
> posv <- rep(NA_integer_, nrows)
> posv[1] <- 0
> posv <- ifelse(indic == lagg, 1, posv)
> posv <- ifelse(indic == (-lagg), -1, posv)
> posv <- zoo::na.locf(posv, na.rm=FALSE)
> posv <- xts::xts(posv, order.by=zoo::index(closep))
> # Lag the positions to trade in next period
> posv <- rutils::lagit(posv, lagg=1)
> # Calculate the PnLs of lagged strategy
> pnlslag <- retp*posv
> colnames(pnlslag) <- "Lagged Strategy"
```

```
> wealthv <- cbind(pnls, pnlslag)
> colnames(wealthv) <- c("EWMA", "Lagged")
> # Annualized Sharpe ratios of EWMA strategies
> sharper <- sqrt(252)*sapply(wealthv, function (x) mean(x)/sd(x))
> # Plot both strategies
> endd <- rutils::calc_endpoints(wealthv, interval="weeks")
> dygraphs::dygraph(cumsum(wealthv)[endd], main=paste("EWMA Crossove
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

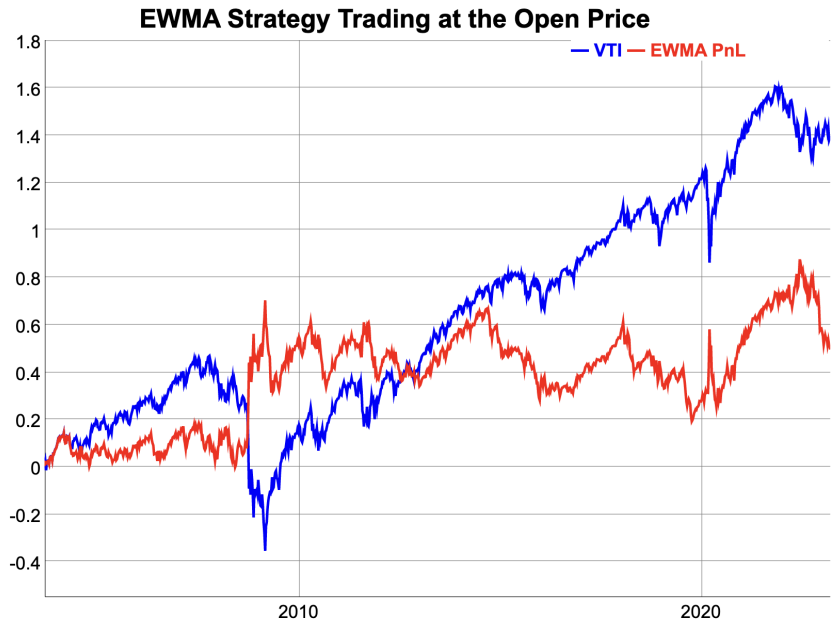
EWMA Strategy Trading at the Open Price

In practice it may not be possible to trade immediately at the *Close* price on the same day that prices cross the *EWMA*.

Then the strategy may trade at the *Open* price on the next day.

The Profit and Loss (*PnL*) on a trade date is the sum of the realized *PnL* from closing the old position, plus the unrealized *PnL* after opening the new position.

```
> # Calculate the positions, either: -1, 0, or 1
> indic <- sign(closep - ewmacpp)
> posv <- rutils::lagit(indic, lagg=1)
> # Calculate the daily pnl for days without trades
> pnls <- retp*posv
> # Determine trade dates right after EWMA has crossed prices
> crossd <- which(rutils::diffit(posv) != 0)
> # Calculate the realized pnl for days with trades
> openp <- quantmod::Op(ohlc)
> closelag <- rutils::lagit(closep)
> poslag <- rutils::lagit(posv)
> pnls[crossd] <- poslag[crossd]*(openp[crossd] - closelag[crossd])
> # Calculate the unrealized pnl for days with trades
> pnls[crossd] <- pnls[crossd] +
+   posv[crossd]*(closep[crossd] - openp[crossd])
> # Calculate the wealth
> wealthv <- cbind(retp, pnls)
> colnames(wealthv) <- c("VTI", "EWMA PnL")
> # Annualized Sharpe ratio of EWMA strategy
> sqrt(252)*sapply(wealthv, function (x) mean(x)/sd(x))
> # The crossover strategy has a negative correlation to VTI
> cor(wealthv)[1, 2]
```




```
> # Plot dygraph of EWMA strategy wealth
> endd <- rutils::calc_endpoints(wealthv, interval="weeks")
> dygraphs::dygraph(cumsum(wealthv)[endd], main="EWMA Strategy Trad
+   dyOptions(colors=colorv, strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
> # Standard plot of EWMA strategy wealth
> quantmod::chart_Series(cumsum(wealthv)[endd], theme=plot_theme,
+        name="EWMA Strategy Trading at the Open Price")
> legend("top", legend=colnames(wealthv),
+  inset=0.05, bg="white", lty=1, lwd=6,
+  col=plot_theme$col$line.col, bty="n")
```

EWMA Crossover Strategy With Transaction Costs

The *bid-ask spread* is the percentage difference between the *ask* (offer) minus the *bid* prices, divided by the *mid* price.

The bid-ask spread for many liquid ETFs is about 1 basis point. For example the *XLK ETF*

Let n_t be the number of shares of the stock owned at time t, and let p_t be their price.

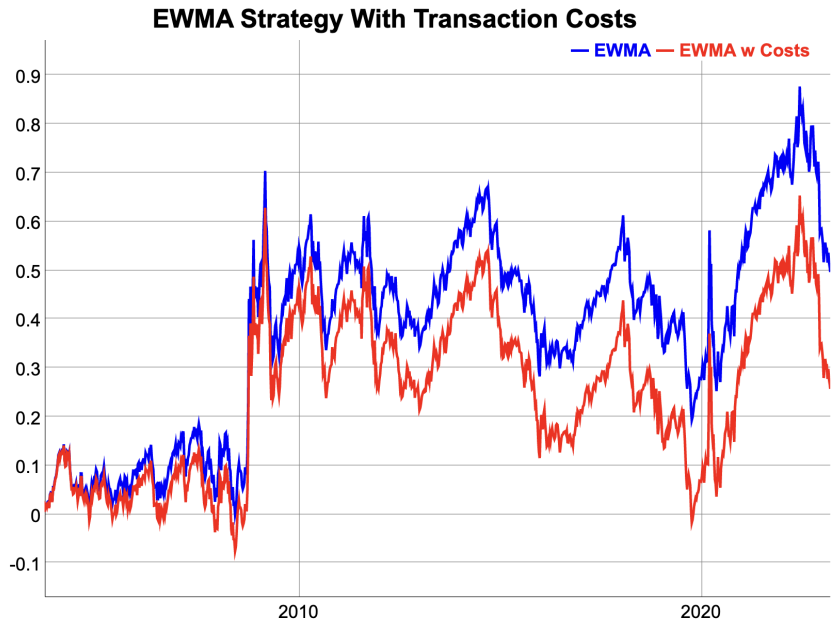
Then the traded dollar amount of the stock is equal to the change in the number of shares times the stock price: $\Delta n_t p_t$.

The the *transaction costs* c^r due to the *bid-ask spread* are equal to half the *bid-ask spread* δ times the absolute value of the traded dollar amount of the stock:

$$c^r = \frac{\delta}{2} |\Delta n_t| \, p_t$$

If d_t is the dollar amount of the stock owned at time t then the *transaction costs* c^r are equal to:

$$c^r = \frac{\delta}{2} |\Delta d_t|$$




```
> # bidask equal to 1 bp for liquid ETFs
> bidask <- 0.001
> # Calculate the transaction costs
> costs <- 0.5*bidask*abs(poslag - posv)
> # Plot strategy with transaction costs
> wealthv <- cbind(pnls, pnls - costs)
> colnames(wealthv) <- c("EWMA", "EWMA w Costs")
> colorv <- c("blue", "red")
> dygraphs::dygraph(cumsum(wealthv)[endd], main="EWMA Strategy With
+   dyOptions(colors=colorv, strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

Simulation Function for EWMA Crossover Strategy

The *EWMA* strategy can be simulated by a single function, which allows the analysis of its performance depending on its parameters.

The function `sim_ewma()` performs a simulation of the *EWMA* strategy, given an *OHLC* time series of pricev, and a decay parameter λ.

The function `sim_ewma()` returns the *EWMA* strategy positions and returns, in a two-column *xts* time series.

```
> sim_ewma <- function(ohlc, lambda=0.9, look_back=333, bidask=0.001
+                trend=1, lagg=1) {
+   closep <- quantmod::Cl(ohlc)
+   retp <- rutils::diffit(closep)
+   nrows <- NROW(ohlc)
+   # Calculate the EWMA prices
+   ewmacpp <- HighFreq::run_mean(closep, lambda=lambda)
+   # Calculate the indicator
+   indic <- trend*sign(closep - ewmacpp)
+   if (lagg > 1) {
+     indic <- HighFreq::roll_sum(indic, lagg)
+     indic[1:lagg] <- 0
+   }  # end if
+   # Calculate the positions, either: -1, 0, or 1
+   posv <- rep(NA_integer_, nrows)
+   posv[1] <- 0
+   posv <- ifelse(indic == lagg, 1, posv)
+   posv <- ifelse(indic == (-lagg), -1, posv)
+   posv <- zoo::na.locf(posv, na.rm=FALSE)
+   posv <- xts::xts(posv, order.by=zoo::index(closep))
+   # Lag the positions to trade on next day
+   posv <- rutils::lagit(posv, lagg=1)
+   # Calculate the PnLs of strategy
+   pnls <- retp*posv
+   costs <- 0.5*bidask*abs(rutils::diffit(posv))
+   pnls <- (pnls - costs)
+   # Calculate the strategy returns
+   pnls <- cbind(posv, pnls)
+   colnames(pnls) <- c("positions", "pnls")
+   pnls
+ }  # end sim_ewma
```

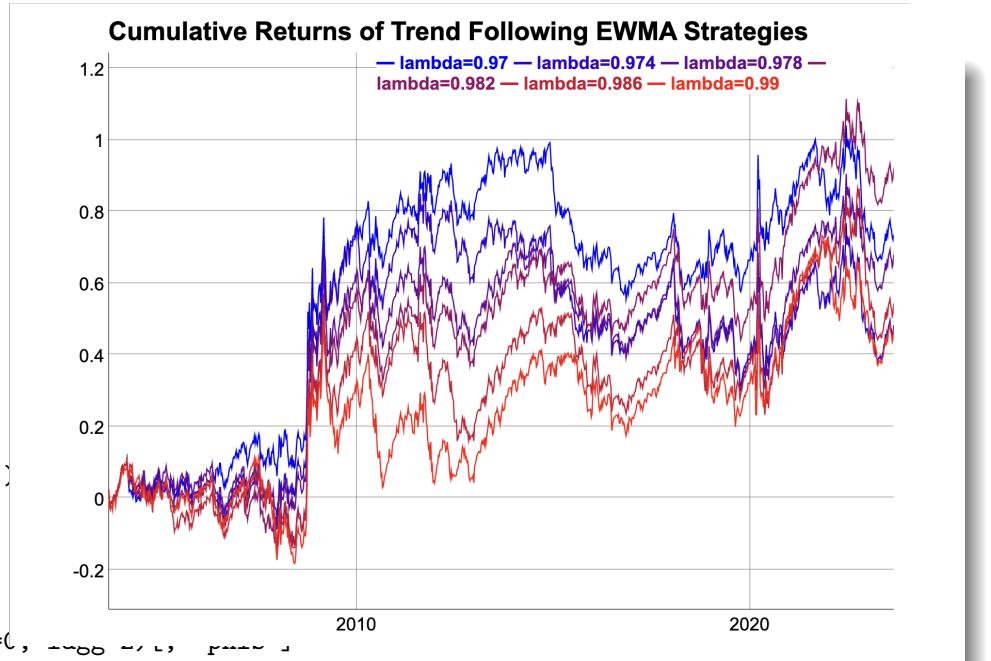
Simulating Multiple Trend Following EWMA Strategies

Multiple *EWMA* strategies can be simulated by calling the function `sim_ewma()` in a loop over a vector of λ parameters.

But `sim_ewma()` returns an *xts* time series, and `sapply()` cannot merge *xts* time series together.

So instead the loop is performed using `lapply()` which returns a list of *xts*, and the list is merged into a single *xts* using the functions `do.call()` and `cbind()`.

```
> source("/Users/jerzy/Develop/lecture_slides/scripts/ewma_model.R")
> lambdav <- seq(from=0.97, to=0.99, by=0.004)
> # Perform lapply() loop over lambdav
> pnltrend <- lapply(lambdav, function(lambda) {
+   # Simulate EWMA strategy and Calculate the returns
+   sim_ewma(ohlc=ohlc, lambda=lambda, look_back=look_back, bidask=0,
+ })  # end lapply
> pnltrend <- do.call(cbind, pnltrend)
> colnames(pnltrend) <- paste0("lambda=", lambdav)
```




```
> # Plot dygraph of multiple EWMA strategies
> colorv <- colorRampPalette(c("blue", "red"))(NCOL(pnltrend))
> endd <- rutils::calc_endpoints(pnltrend, interval="weeks")
> dygraphs::dygraph(cumsum(pnltrend)[endd], main="Cumulative Returns
+   dyOptions(colors=colorv, strokeWidth=1) %>%
+   dyLegend(show="always", width=400)
> # Plot EWMA strategies with custom line colors
> plot_theme <- chart_theme()
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(cumsum(pnltrend), theme=plot_theme,
+   name="Cumulative Returns of EWMA Strategies")
> legend("topleft", legend=colnames(pnltrend), inset=0.1,
+   bg="white", cex=0.8, lwd=rep(6, NCOL(pnltrend)),
+   col=plot_theme$col$line.col, bty="n")
```

Simulating EWMA Strategies Using Parallel Computing

Simulating *EWMA* strategies naturally lends itself to parallel computing, since the simulations are independent from each other.

The function `parLapply()` is similar to `lapply()`, and performs loops under *Windows* using parallel computing on several CPU cores.

The resulting list of time series can then be collapsed into a single *xts* series using the functions `rutils::do_call()` and `cbind()`.

```
> # Initialize compute cluster under Windows
> library(parallel)
> ncores <- detectCores() - 1  # Number of cores
> cluster <- makeCluster(detectCores()-1)
> clusterExport(cluster,
+   varlist=c("ohlc", "look_back", "sim_ewma"))
> # Perform parallel loop over lambdav under Windows
> pnltrend <- parLapply(cluster, lambdav, function(lambda) {
+   library(quantmod)
+   # Simulate EWMA strategy and Calculate the returns
+   sim_ewma(ohlc=ohlc, lambda=lambda, look_back=look_back)[, "pnls"
+ })  # end parLapply
> stretduster(cluster)  # Stop R processes over cluster under Windo
> # Perform parallel loop over lambdav under Mac-OSX or Linux
> pnltrend <- mclapply(lambdav, function(lambda) {
+   library(quantmod)
+   # Simulate EWMA strategy and Calculate the returns
+   sim_ewma(ohlc=ohlc, lambda=lambda, look_back=look_back)[, "pnls"
+ }, mc.cores=ncores)  # end mclapply
> pnltrend <- do.call(cbind, pnltrend)
> colnames(pnltrend) <- paste0("lambda=", lambdav)
```

Optimal Decay Parameter of Trend Following EWMA Strategies

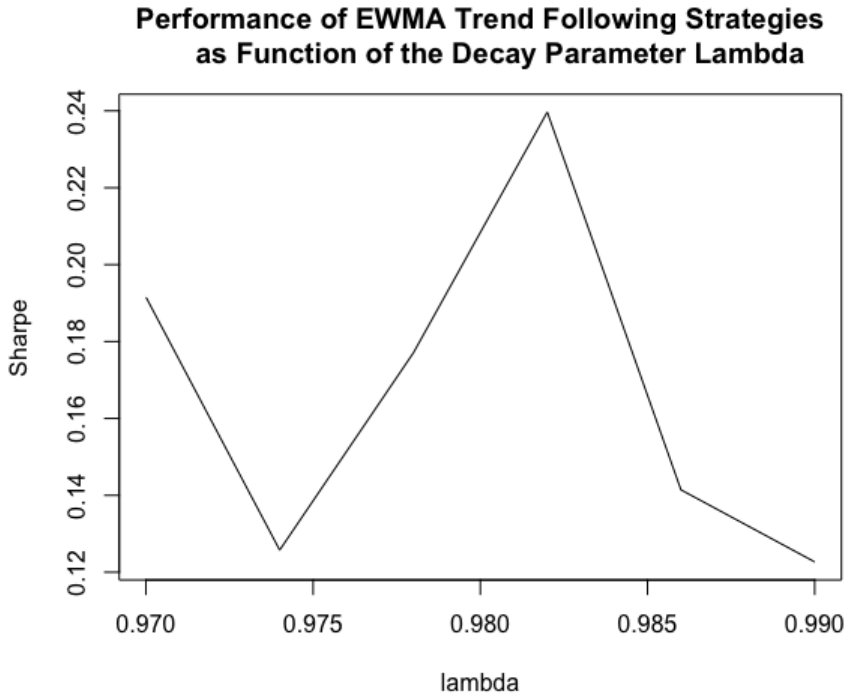
The performance of trend following *EWMA* strategies depends on the λ decay parameter, with smaller λ parameters performing better than larger ones.

The optimal λ parameter applies significant weight to returns $8 - 12$ months in the past, which is consistent with research on trend following strategies.

The *Sharpe ratios* of *EWMA* strategies with different λ parameters can be calculated by performing an `sapply()` loop over the *columns* of returns.

`sapply()` treats the columns of *xts* time series as list elements, and loops over the columns.

Performing loops in `R` over the *columns* of returns is acceptable, but `R` loops over the *rows* of returns should be avoided.

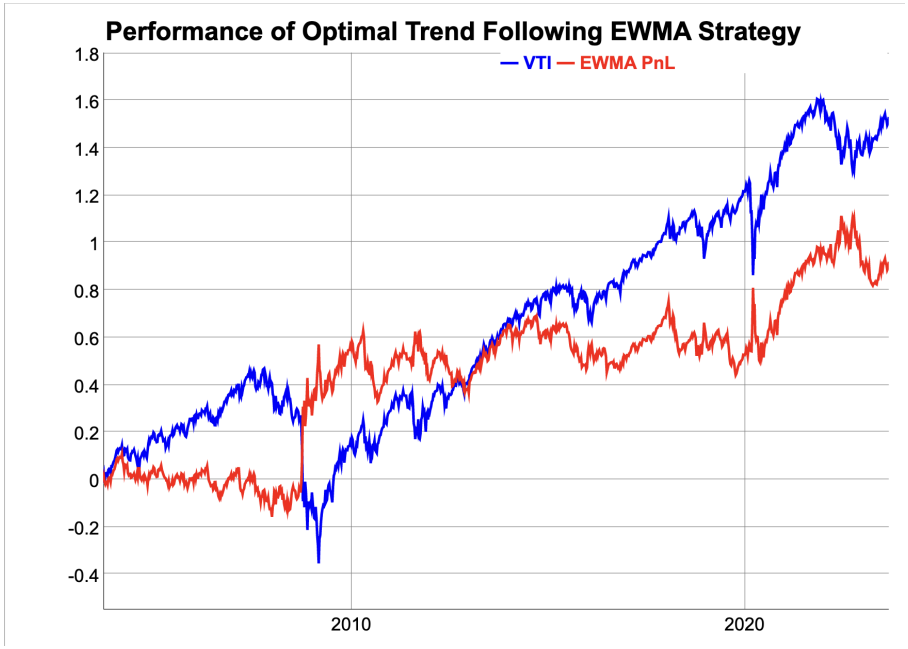



```
> # Calculate the annualized Sharpe ratios of strategy returns
> sharpetrend <- sqrt(252)*sapply(pnltrend, function(xtsv) {
+   mean(xtsv)/sd(xtsv)
+ })  # end sapply
> # Plot Sharpe ratios
> dev.new(width=6, height=5, noRStudioGD=TRUE)
> plot(x=lambdav, y=sharpetrend, t="l",
+      xlab="lambda", ylab="Sharpe",
+      main="Performance of EWMA Trend Following Strategies
+      as Function of the Decay Parameter Lambda")
```

Optimal Trend Following EWMA Strategy

The best performing trend following *EWMA* strategy has a relatively small λ parameter, corresponding to slower weight decay (giving more weight to past pricev), and producing less frequent trading.

```
> # Calculate the optimal lambda
> lambda <- lambdav[which.max(sharpetrend)]
> # Simulate best performing strategy
> ewmatrend <- sim_ewma(ohlc=ohlc, lambda=lambda, bidask=0, lagg=2)
> posv <- ewmatrend[, "positions"]
> trendopt <- ewmatrend[, "pnls"]
> wealthv <- cbind(retp, trendopt)
> colnames(wealthv) <- c("VTI", "EWMA PnL")
> # Calculate the Sharpe and Sortino ratios
> sqrt(252)*sapply(wealthv, function(x)
+   c(Sharpe=mean(x)/sd(x), Sortino=mean(x)/sd(x[x<0])))
> cor(wealthv)[1, 2]
> # Plot dygraph of EWMA strategy wealth
> dygraphs::dygraph(cumsum(wealthv)[endd], main="Performance of Opt
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```



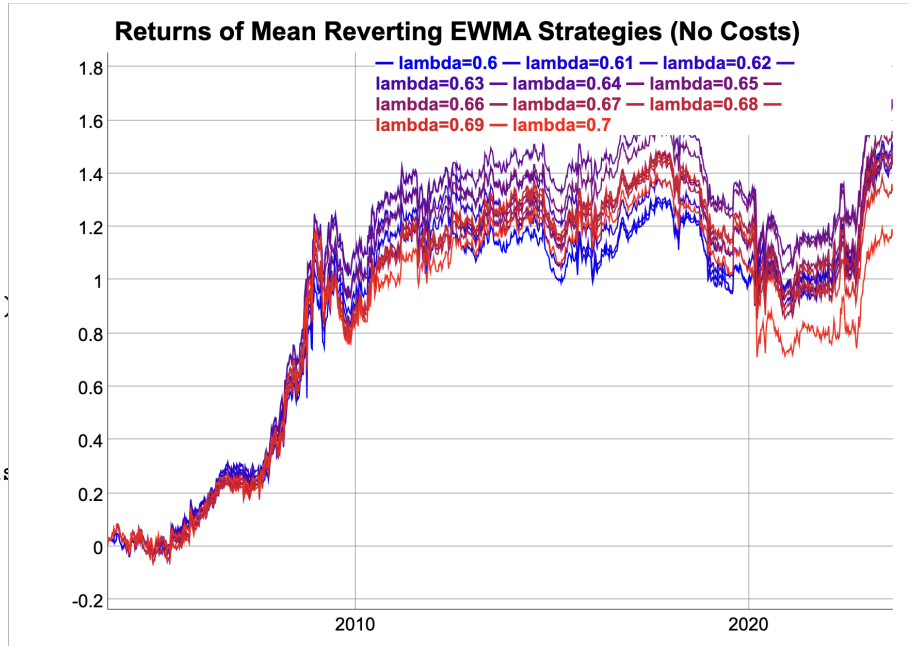

```
> # Plot EWMA PnL with position shading
> # Standard plot of EWMA strategy wealth
> x11(width=6, height=5)
> plot_theme <- chart_theme()
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(cumsum(wealthv), theme=plot_theme,
+        name="Performance of EWMA Strategy")
> add_TA(posv > 0, on=-1, col="lightgreen", border="lightgreen")
> add_TA(posv < 0, on=-1, col="lightgrey", border="lightgrey")
> legend("top", legend=colnames(wealthv),
+  inset=0.05, bg="white", lty=1, lwd=6,
+  col=plot_theme$col$line.col, bty="n")
```

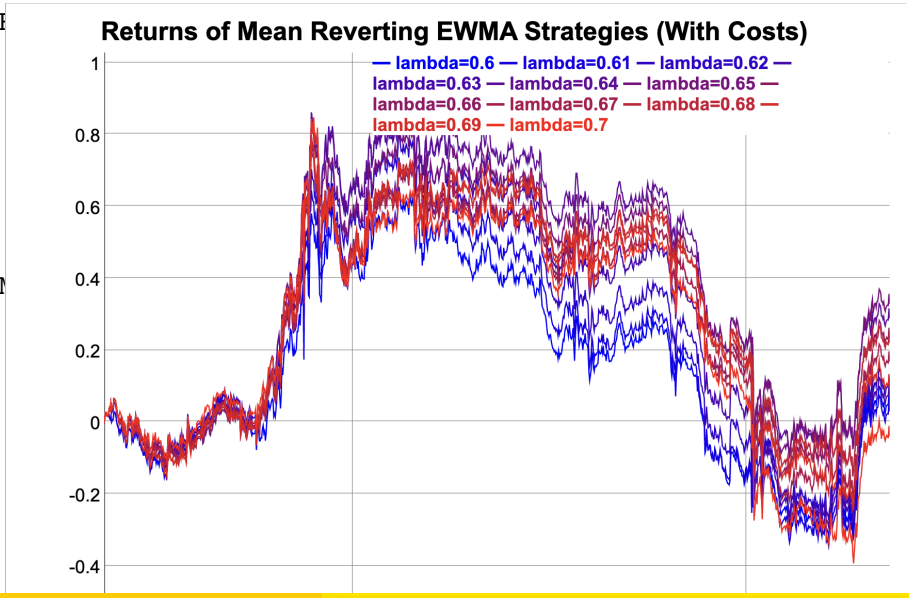
Mean Reverting EWMA Crossover Strategies

Mean reverting EWMA crossover strategies can be simulated using function `sim_ewma()` with argument `trend=(-1)`.

The profitability of mean reverting strategies can be significantly improved by using limit orders, to reduce transaction costs.

```
> source("/Users/jerzy/Develop/lecture_slides/scripts/ewma_model.R")
> lambdav <- seq(0.6, 0.7, 0.01)
> # Perform lapply() loop over lambdav
> pnlrevert <- lapply(lambdav, function(lambda) {
+   # Simulate EWMA strategy and Calculate the returns
+   sim_ewma(ohlc=ohlc, lambda=lambda, bidask=0, trend=(-1))[, "pnls
+ })  # end lapply
> pnlrevert <- do.call(cbind, pnlrevert)
> colnames(pnlrevert) <- paste0("lambda=", lambdav)
> # Plot dygraph of mean reverting EWMA strategies
> colorv <- colorRampPalette(c("blue", "red"))(NROW(lambdav))
> dygraphs::dygraph(cumsum(pnlrevert)[endd], main="Returns of Mean R
+   dyOptions(colors=colorv, strokeWidth=1) %>%
+   dyLegend(show="always", width=400)
> # Plot EWMA strategies with custom line colors
> x11(width=6, height=5)
> plot_theme <- chart_theme()
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(pnlrevert,
+   theme=plot_theme, name="Cumulative Returns of Mean Reverting EWM
> legend("topleft", legend=colnames(pnlrevert),
+   inset=0.1, bg="white", cex=0.8, lwd=6,
+   col=plot_theme$col$line.col, bty="n")
```





Performance of Mean Reverting EWMA Strategies

The *Sharpe ratios* of *EWMA* strategies with different λ parameters can be calculated by performing an `sapply()` loop over the *columns* of returns.

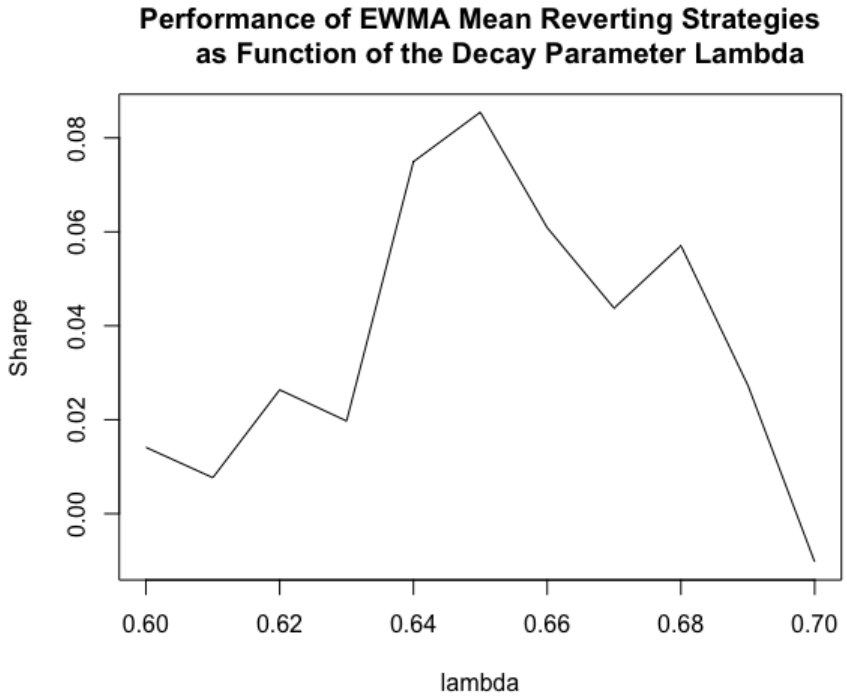
`sapply()` treats the columns of *xts* time series as list elements, and loops over the columns.

Performing loops in `R` over the *columns* of returns is acceptable, but `R` loops over the *rows* of returns should be avoided.

The performance of mean reverting *EWMA* strategies depends on the λ parameter, with performance decreasing for very small or very large λ parameters.

For too large λ parameters, the trading frequency is too high, causing high transaction costs.

For too small λ parameters, the trading frequency is too low, causing the strategy to miss profitable trades.



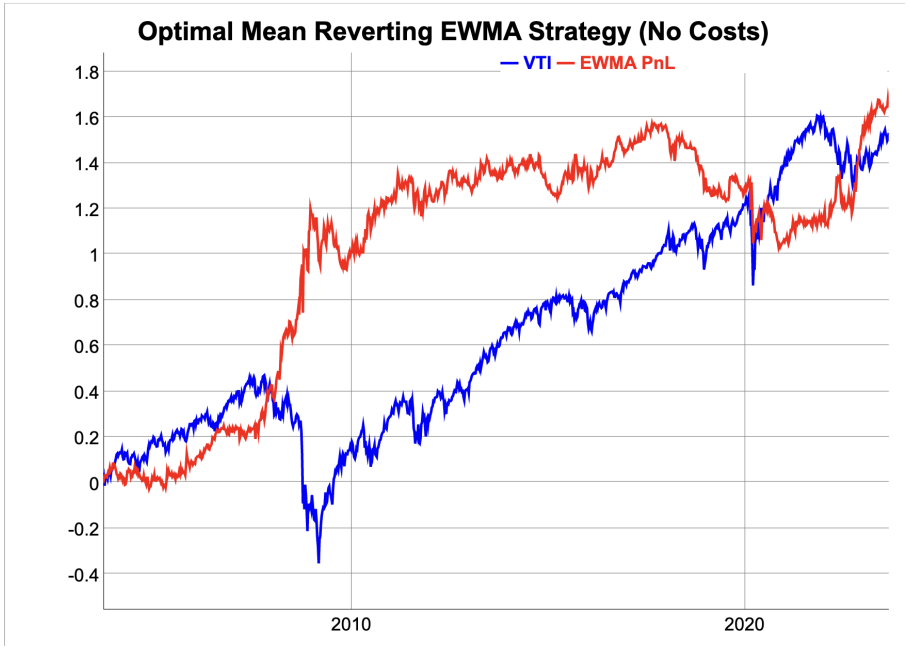

```
> # Calculate the Sharpe ratios of strategy returns
> sharperevert <- sqrt(252)*sapply(pnlrevert, function(xtsv) {
+   mean(xtsv)/sd(xtsv)
+ })  # end sapply
> plot(x=lambdav, y=sharperevert, t="l",
+      xlab="lambda", ylab="Sharpe",
+      main="Performance of EWMA Mean Reverting Strategies
+      as Function of the Decay Parameter Lambda")
```

Optimal Mean Reverting EWMA Strategy

Reverting the direction of the trend following *EWMA* strategy creates a mean reverting strategy.

The best performing mean reverting *EWMA* strategy has a relatively large λ parameter, corresponding to faster weight decay (giving more weight to recent prices), and producing more frequent trading.

But a too large λ parameter also causes very high trading frequency, and high transaction costs.




```
> # Calculate the optimal lambda
> lambda <- lambdav[which.max(sharperevert)]
> # Simulate best performing strategy
> ewmarevert <- sim_ewma(ohlc=ohlc, bidask=0.0,
+   lambda=lambda, trend=(-1))
> posv <- ewmarevert[, "positions"]
> revertopt <- ewmarevert[, "pnls"]
> wealthv <- cbind(retp, revertopt)
> colnames(wealthv) <- c("VTI", "EWMA PnL")
> # Plot dygraph of EWMA strategy wealth
> colorv <- c("blue", "red")
> dygraphs::dygraph(cumsum(wealthv)[endd], main="Optimal Mean Reve
+   dyOptions(colors=colorv, strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

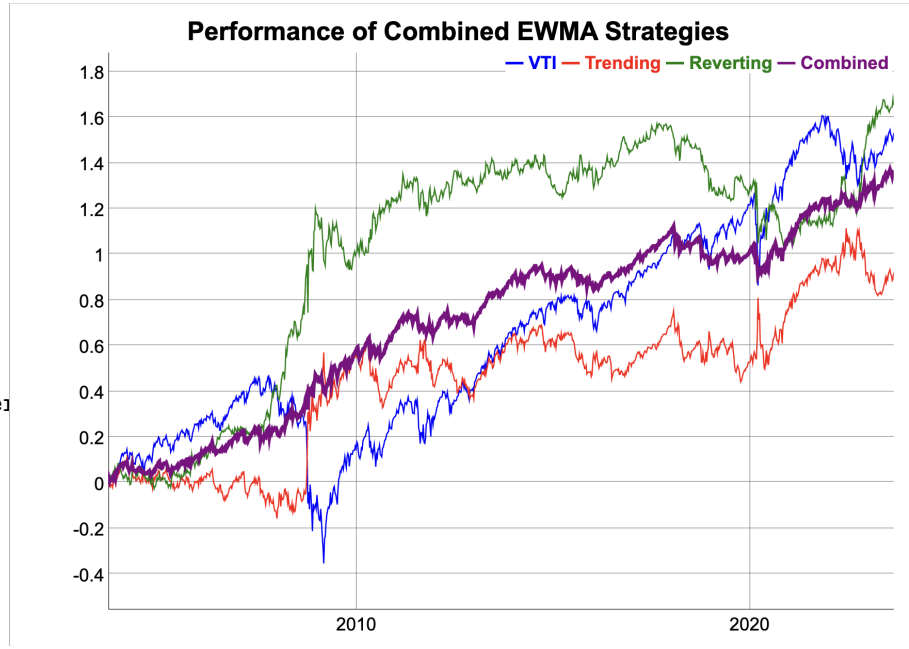
```
> # Standard plot of EWMA strategy wealth
> x11(width=6, height=5)
> plot_theme <- chart_theme()
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(cumsum(wealthv), theme=plot_theme,
+        name="Optimal Mean Reverting EWMA Strategy")
> add_TA(posv > 0, on=-1, col="lightgreen", border="lightgreen")
> add_TA(posv < 0, on=-1, col="lightgrey", border="lightgrey")
> legend("top", legend=colnames(wealthv),
+  inset=0.05, bg="white", lty=1, lwd=6,
+  col=plot_theme$col$line.col, bty="n")
```

Combining Trend Following and Mean Reverting Strategies

The returns of trend following and mean reverting strategies are usually negatively correlated to each other, so combining them can achieve significant diversification of risk.

The main advantage of EWMA crossover strategies is that they provide positive returns and a diversification of risk with respect to static stock portfolios.

```
> # Calculate the correlation between trend following and mean rever
> trendopt <- ewmatrend[, "pnls"]
> colnames(trendopt) <- "trend"
> revertopt <- ewmarevert[, "pnls"]
> colnames(revertopt) <- "revert"
> cor(cbind(retp, trendopt, revertopt))
> # Calculate the combined strategy
> combstrat <- (retp + trendopt + revertopt)/3
> colnames(combstrat) <- "combined"
> # Calculate the annualized Sharpe ratio of strategy returns
> retc <- cbind(retp, trendopt, revertopt, combstrat)
> colnames(retc) <- c("VTI", "Trending", "Reverting", "Combined")
> sqrt(252)*sapply(retc, function(xtsv) mean(xtsv)/sd(xtsv))
```



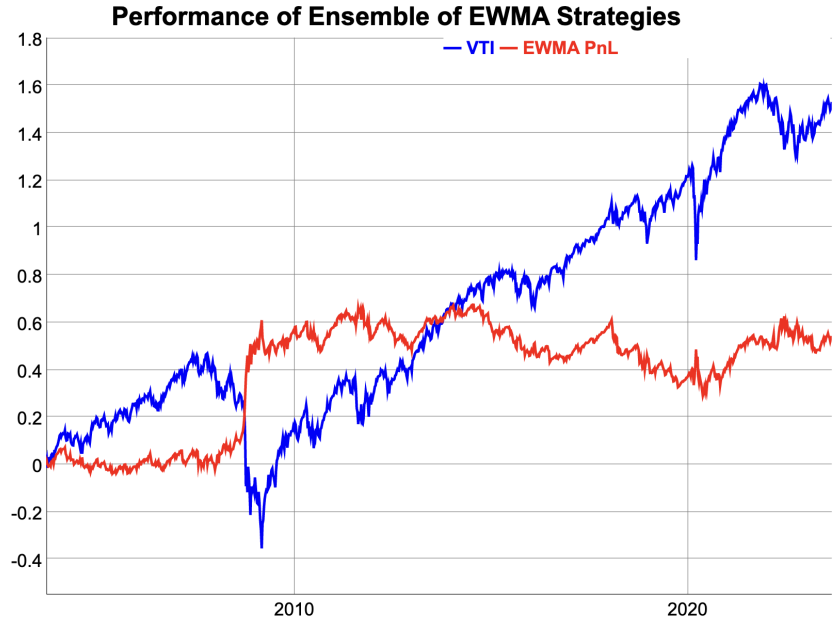

```
> # Plot dygraph of EWMA strategy wealth
> colorv <- c("blue", "red", "green", "purple")
> dygraphs::dygraph(cumsum(retc)[endd], main="Performance of Combine
+   dyOptions(colors=colorv, strokeWidth=1) %>%
+   dySeries(name="Combined", label="Combined", strokeWidth=3) %>%
+   dyLegend(show="always", width=200)
> # Standard plot of EWMA strategy wealth
> plot_theme <- chart_theme()
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(pnls, theme=plot_theme,
+        name="Performance of Combined EWMA Strategies")
> legend("topleft", legend=colnames(pnls),
+  inset=0.05, bg="white", lty=1, lwd=6,
+  col=plot_theme$col$line.col, bty="n")
```

Ensemble of EWMA Strategies

Instead of selecting the best performing *EWMA* strategy, one can choose a weighted average of strategies (ensemble), which corresponds to allocating positions according to the weights.

The weights can be chosen to be proportional to the Sharpe ratios of the *EWMA* strategies.

```
> # Calculate the weights proportional to Sharpe ratios
> weightv <- c(sharpetrend, sharperevert)
> weightv[weightv < 0] <- 0
> weightv <- weightv/sum(weightv)
> retc <- cbind(pnltrend, pnlrevert)
> retc <- retc %*% weightv
> retc <- cbind(retp, retc)
> colnames(retc) <- c("VTI", "EWMA PnL")
> # Plot dygraph of EWMA strategy wealth
> colorv <- c("blue", "red")
> dygraphs::dygraph(cumsum(retc)[endd], main="Performance of Ensemble of EWMA Strategies") %>%
+   dyOptions(colors=colorv, strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
> # Standard plot of EWMA strategy wealth
> plot_theme <- chart_theme()
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(cumsum(retc), theme=plot_theme,
+        name="Performance of Ensemble of EWMA Strategies")
> legend("topleft", legend=colnames(pnls),
+  inset=0.05, bg="white", lty=1, lwd=6,
+  col=plot_theme$col$line.col, bty="n")
```

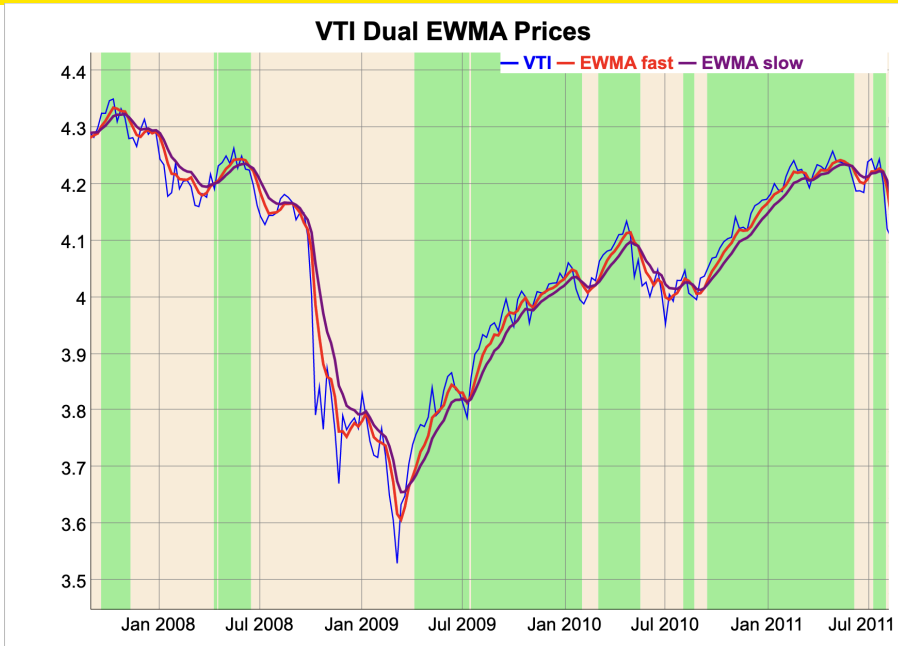


Simulating the Dual EWMA Crossover Strategy

In the *Dual EWMA Crossover* strategy, the stock position depends on the difference between two moving averages.

The stock position flips when the fast moving *EWMA* crosses the slow moving *EWMA*.

```
> # Calculate the fast and slow EWMAs
> lambdaf <- 0.89
> lambdas <- 0.95
> # Calculate the EWMA prices
> ewmaf <- HighFreq::run_mean(closep, lambda=lambdaf)
> ewmas <- HighFreq::run_mean(closep, lambda=lambdas)
> # Calculate the EWMA prices
> pricev <- cbind(closep, ewmaf, ewmas)
> colnames(pricev) <- c("VTI", "EWMA fast", "EWMA slow")
> # Calculate the positions, either: -1, 0, or 1
> indic <- sign(ewmaf - ewmas)
> lagg <- 2
> indic <- HighFreq::roll_sum(indic, lagg)
> posv <- rep(NA_integer_, nrows)
> posv[1] <- 0
> posv <- ifelse(indic == lagg, 1, posv)
> posv <- ifelse(indic == (-lagg), -1, posv)
> posv <- zoo::na.locf(posv, na.rm=FALSE)
> posv <- xts::xts(posv, order.by=zoo::index(closep))
> posv <- rutils::lagit(posv, lagg=1)
```



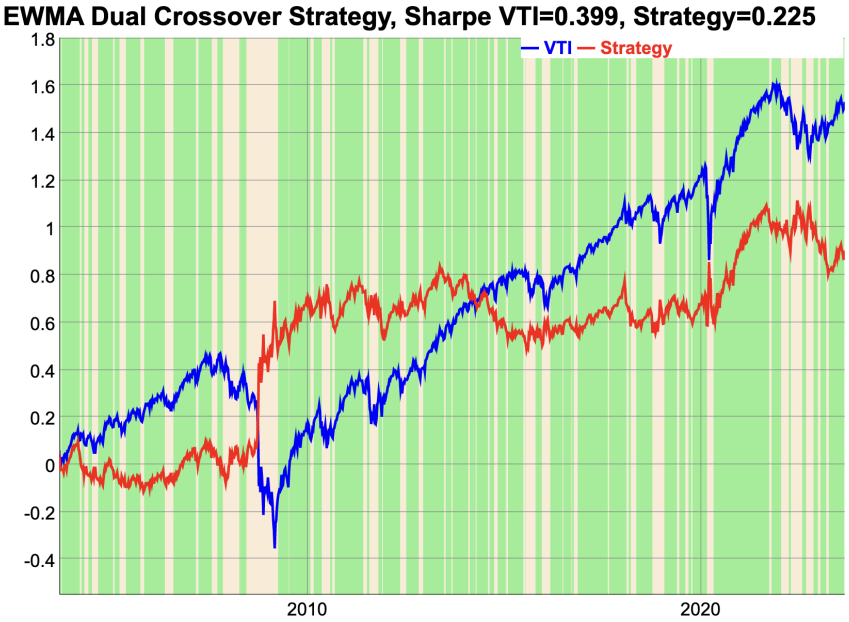

```
> # Create colors for background shading
> crossd <- (rutils::diffit(posv) != 0)
> shadev <- posv[crossd]
> crossd <- c(zoo::index(shadev), end(posv))
> shadev <- ifelse(drop(zoo::coredata(shadev)) == 1, "lightgreen", "
> # Plot dygraph
> colnamev <- colnames(pricev)
> dyplot <- dygraphs::dygraph(pricev[endd], main="VTI Dual EWMA Pric
+   dySeries(name=colnamev[1], strokeWidth=1, col="blue") %>%
+   dySeries(name=colnamev[2], strokeWidth=2, col="red") %>%
+   dySeries(name=colnamev[3], strokeWidth=2, col="purple") %>%
+   dyLegend(show="always", width=200)
> for (i in 1:NROW(shadev)) {
+   dyplot <- dyplot %>% dyShading(from=crossd[i], to=crossd[i+1], c
+ }  # end for
> dyplot
```

Dual EWMA Crossover Strategy Performance

The crossover strategy suffers losses when prices are range-bound without a trend, because whenever it switches position the prices soon change direction. (This is called a *"whipsaw"*.)

The crossover strategy performance is worse than the underlying asset (*VTI*), but it has a negative correlation to it, which is very valuable when building a portfolio.

```
> # Calculate the daily profits and losses of strategy
> pnls <- retp*posv
> colnames(pnls) <- "Strategy"
> wealthv <- cbind(retp, pnls)
> # Annualized Sharpe ratio of Dual EWMA strategy
> sharper <- sqrt(252)*sapply(wealthv, function (x) mean(x)/sd(x))
> # The crossover strategy has a negative correlation to VTI
> cor(wealthv)[1, 2]
```




```
> # Plot Dual EWMA strategy
> dyplot <- dygraphs::dygraph(cumsum(wealthv)[endd], main=paste("EWM
+   dyOptions(colors=c("blue", "red"), strokeWidth=2)
> # Add shading to dygraph object
> for (i in 1:NROW(shadev)) {
+   dyplot <- dyplot %>% dyShading(from=crossd[i], to=crossd[i+1], c
+ }  # end for
> # Plot the dygraph object
> dyplot
```

Simulation Function for the Dual EWMA Crossover Strategy

The *Dual EWMA* strategy can be simulated by a single function, which allows the analysis of its performance depending on its parameters.

The function `sim_ewma2()` performs a simulation of the *Dual EWMA* strategy, given an *OHLC* time series of pricev, and two decay parameters λ_1 and λ_2.

The function `sim_ewma2()` returns the *EWMA* strategy positions and returns, in a two-column *xts* time series.

```
> sim_ewma2 <- function(ohlc, lambdaf=0.1, lambdas=0.01,
+                       bidask=0.001, trend=1, lagg=1) {
+   if (lambdaf >= lambdas) return(NA)
+   closep <- quantmod::Cl(ohlc)
+   retp <- rutils::diffit(closep)
+   nrows <- NROW(ohlc)
+   # Calculate the EWMA prices
+   ewmaf <- HighFreq::run_mean(closep, lambda=lambdaf)
+   ewmas <- HighFreq::run_mean(closep, lambda=lambdas)
+   # Calculate the positions, either: -1, 0, or 1
+   indic <- sign(ewmaf - ewmas)
+   if (lagg > 1) {
+     indic <- HighFreq::roll_sum(indic, lagg)
+     indic[1:lagg] <- 0
+   }  # end if
+   posv <- rep(NA_integer_, nrows)
+   posv[1] <- 0
+   posv <- ifelse(indic == lagg, 1, posv)
+   posv <- ifelse(indic == (-lagg), -1, posv)
+   posv <- zoo::na.locf(posv, na.rm=FALSE)
+   posv <- xts::xts(posv, order.by=zoo::index(closep))
+   # Lag the positions to trade on next day
+   posv <- rutils::lagit(posv, lagg=1)
+   # Calculate the PnLs of strategy
+   pnls <- retp*posv
+   costs <- 0.5*bidask*abs(rutils::diffit(posv))
+   pnls <- (pnls - costs)
+   # Calculate the strategy returns
+   pnls <- cbind(posv, pnls)
+   colnames(pnls) <- c("positions", "pnls")
+   pnls
+ }  # end sim_ewma2
```

Dual EWMA Strategy Performance Matrix

Multiple *Dual EWMA* strategies can be simulated by calling the function `sim_ewma2()` in two loops over the vectors of λ parameters.

The function `outer()` calculates the values of a function over a grid spanned by two variables, and returns a matrix of function values.

The function `Vectorize()` performs an `apply()` loop over the arguments of a function, and returns a vectorized version of the function.

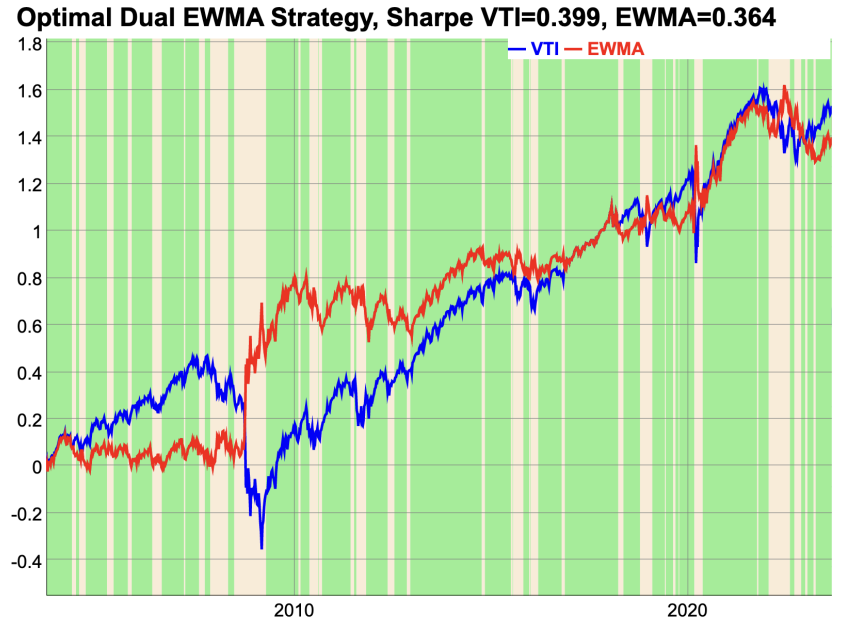
```
> source("/Users/jerzy/Develop/lecture_slides/scripts/ewma_model.R")
> lambdafv <- seq(from=0.85, to=0.99, by=0.01)
> lambdasv <- seq(from=0.85, to=0.99, by=0.01)
> # Calculate the Sharpe ratio of dual EWMA strategy
> calc_sharpe <- function(ohlc, lambdaf, lambdas, bidask, trend, lag
+   if (lambdaf >= lambdas) return(NA)
+   pnls <- sim_ewma2(ohlc=ohlc, lambdaf=lambdaf, lambdas=lambdas,
+     bidask=bidask, trend=trend, lagg=lagg)[, "pnls"]
+   sqrt(252)*mean(pnls)/sd(pnls)
+ }  # end calc_sharpe
> # Vectorize calc_sharpe with respect to lambdaf and lambdas
> calc_sharpe <- Vectorize(FUN=calc_sharpe,
+   vectorize.args=c("lambdaf", "lambdas"))
> # Calculate the matrix of PnLs
> sharpem <- outer(lambdafv, lambdasv, FUN=calc_sharpe, ohlc=ohlc,
+            bidask=0.0, trend=1, lagg=2)
> # Or perform two sapply() loops over lambda vectors
> sharpem <- sapply(lambdasv, function(lambdas) {
+   sapply(lambdafv, function(lambdaf) {
+     if (lambdaf >= lambdas) return(NA)
+     calc_sharpe(ohlc=ohlc, lambdaf=lambdaf, lambdas=lambdas,
+           bidask=0.0, trend=1, lagg=2)
+   })  # end sapply
+ })  # end sapply
> colnames(sharpem) <- lambdasv
> rownames(sharpem) <- lambdafv
```

Optimal Dual EWMA Strategy

The best *Dual EWMA* strategy performs better than the best *single EWMA* strategy, because it has an extra parameter that can be adjusted to improve in-sample performance.

But this doesn't guarantee better out-of-sample performance.

```
> # Calculate the PnLs for the optimal strategy
> whichv <- which(sharpem == max(sharpem, na.rm=TRUE), arr.ind=TRUE)
> lambdaf <- lambdafv[whichv[1]]
> lambdas <- lambdasv[whichv[2]]
> ewma_opt <- sim_ewma2(ohlc=ohlc, lambdaf=lambdaf, lambdas=lambdas,
+   bidask=0.0, trend=1, lagg=2)
> pnls <- ewma_opt[, "pnls"]
> wealthv <- cbind(retp, pnls)
> colnames(wealthv)[2] <- "EWMA"
> # Calculate the Sharpe and Sortino ratios
> sqrt(252)*sapply(wealthv, function(x)
+   c(Sharpe=mean(x)/sd(x), Sortino=mean(x)/sd(x[x<0])))
> # Annualized Sharpe ratio of Dual EWMA strategy
> sharper <- sqrt(252)*sapply(wealthv, function (x) mean(x)/sd(x))
> # The crossover strategy has a negative correlation to VTI
> cor(wealthv)[1, 2]
```

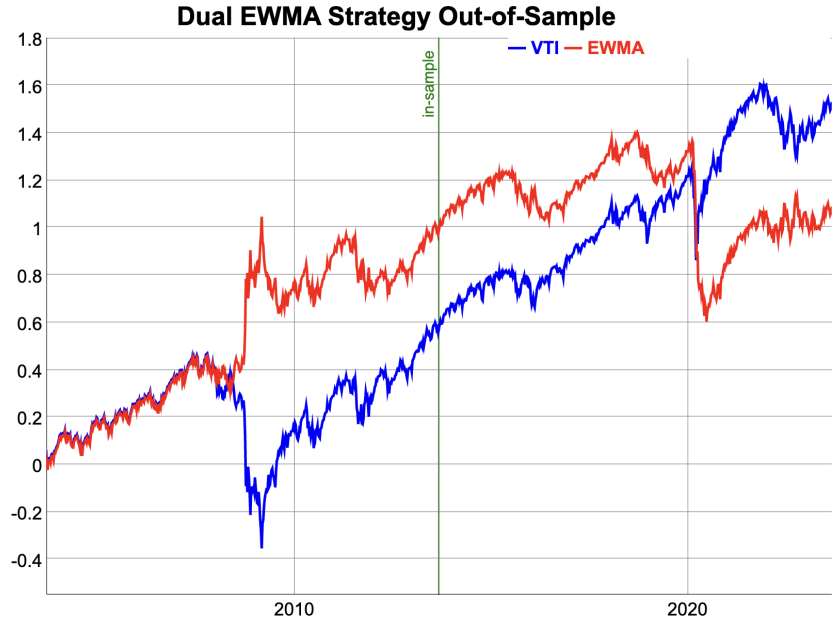



```
> # Create colors for background shading
> posv <- ewma_opt[, "positions"]
> crossd <- (rutils::diffit(posv) != 0)
> shadev <- posv[crossd]
> crossd <- c(zoo::index(shadev), end(posv))
> shadev <- ifelse(drop(zoo::coredata(shadev)) == 1, "lightgreen",
> # Plot Optimal Dual EWMA strategy
> dyplot <- dygraphs::dygraph(cumsum(wealthv), main=paste("Optimal
+   dyOptions(colors=c("blue", "red"), strokeWidth=2)
> # Add shading to dygraph object
> for (i in 1:NROW(shadev)) {
+   dyplot <- dyplot %>% dyShading(from=crossd[i], to=crossd[i+1],
+ }  # end for
> # Plot the dygraph object
> dyplot
```

Performance of Dual Crossover Strategy *Out-of-Sample*

In-sample, the best *Dual EWMA* strategy performs better than *VTI*, because it has two parameters that can be adjusted to improve performance.

But out-of-sample, the best *Dual EWMA* strategy performs worse than *VTI*, because it's been *overfitted* in-sample.




```
> # Define in-sample and out-of-sample intervals
> insample <- 1:(nrows %/% 2)
> outsample <- (nrows %/% 2 + 1):nrows
> # Calculate the matrix of PnLs
> sharpem <- outer(lambdafv, lambdasv,
+            FUN=calc_sharpe, ohlc=ohlc[insample, ],
+            bidask=0.0, trend=1, lagg=2)
> colnames(sharpem) <- lambdasv
> rownames(sharpem) <- lambdafv
> # Calculate the PnLs for the optimal strategy
> whichv <- which(sharpem == max(sharpem, na.rm=TRUE), arr.ind=TRU
> lambdaf <- lambdafv[whichv[1]]
> lambdas <- lambdasv[whichv[2]]
> pnls <- sim_ewma2(ohlc=ohlc, lambdaf=lambdaf, lambdas=lambdas,
+             bidask=0.0, trend=1, lagg=2)[, "pnls"]
> wealthv <- cbind(retp, pnls)
> colnames(wealthv)[2] <- "EWMA"
> # Calculate the Sharpe and Sortino ratios in-sample and out-of-sample
> sqrt(252)*sapply(wealthv[insample, ], function(x)
+   c(Sharpe=mean(x)/sd(x), Sortino=mean(x)/sd(x[x<0])))
> sqrt(252)*sapply(wealthv[outsample, ], function(x)
+   c(Sharpe=mean(x)/sd(x), Sortino=mean(x)/sd(x[x<0])))
```

```
> # Dygraphs plot with custom line colors
> endd <- rutils::calc_endpoints(wealthv, interval="weeks")
> dygraphs::dygraph(cumsum(wealthv)[endd], main="Dual EWMA Strategy
+   dyEvent(zoo::index(wealthv[last(insample)]), label="in-sample",
+   dyOptions(colors=c("blue", "red"), strokeWidth=2)
```

Volume-Weighted Average Price Indicator

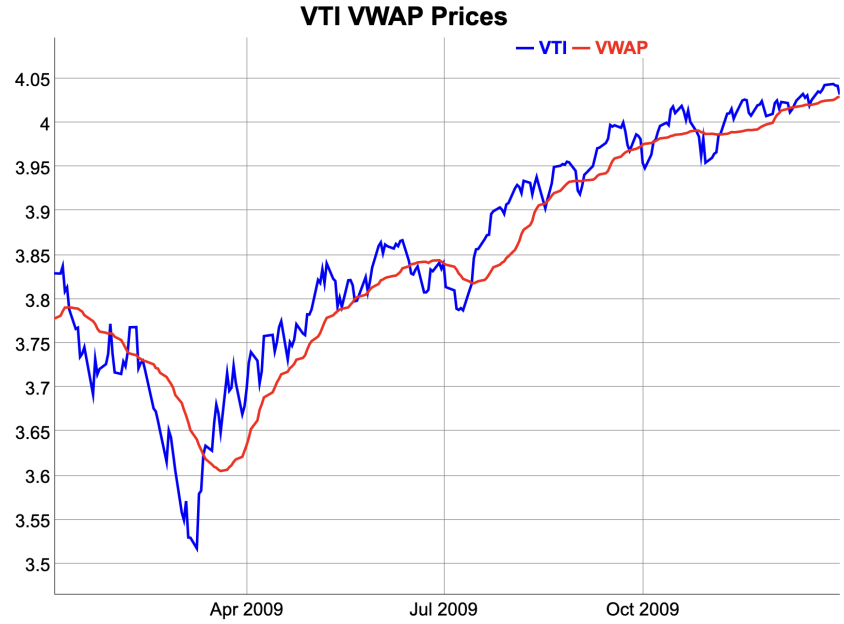
The Volume-Weighted Average Price (*VWAP*) is defined as the sum of prices multiplied by trading volumes, divided by the sum of volumes:

$$p_t^{VWAP} = \frac{\sum_{j=0}^{n} v_{t-j} p_{t-j}}{\sum_{j=0}^{n} v_{t-j}}$$

The *VWAP* applies more weight to prices with higher trading volumes, which allows it to react more quickly to recent market volatility.

The drawback of the *VWAP* indicator is that it applies large weights to prices far in the past.

The *VWAP* is often used as a technical indicator in trend following strategies.




```
> # Calculate the log OHLC prices and volumes
> ohlc <- rutils::etfenv$VTI
> closep <- log(quantmod::Cl(ohlc))
> colnames(closep) <- "VTI"
> volum <- quantmod::Vo(ohlc)
> colnames(volum) <- "Volume"
> nrows <- NROW(closep)
> # Calculate the VWAP prices
> look_back <- 21
> vwap <- HighFreq::roll_sum(closep*volum, look_back)
> volumr <- HighFreq::roll_sum(volum, look_back)
> vwap <- vwap/volumr
> colnames(vwap) <- "VWAP"
> pricev <- cbind(closep, vwap)
```

```
> # Dygraphs plot with custom line colors
> colorv <- c("blue", "red")
> dygraphs::dygraph(pricev["2009"], main="VTI VWAP Prices") %>%
+   dyOptions(colors=colorv, strokeWidth=2)
> # Plot VWAP prices with custom line colors
> x11(width=6, height=5)
> plot_theme <- chart_theme()
> plot_theme$col$line.col <- colors
> quantmod::chart_Series(pricev["2009"], theme=plot_theme,
+        lwd=2, name="VTI VWAP Prices")
> legend("bottomright", legend=colnames(pricev),
+  inset=0.1, bg="white", lty=1, lwd=6, cex=0.8,
+  col=plot_theme$col$line.col, bty="n")
```

Recursive VWAP Price Indicator

The *VWAP* prices p^{VWAP} can also be calculated as the ratio of the volume weighted prices μ^{pv} divided by the mean trading volumes μ^{v}:

$$p^{VWAP} = \frac{\mu^{pv}}{\mu^{v}}$$

The volume weighted prices μ^{pv} and the mean trading volumes μ^{v} are both calculated recursively:

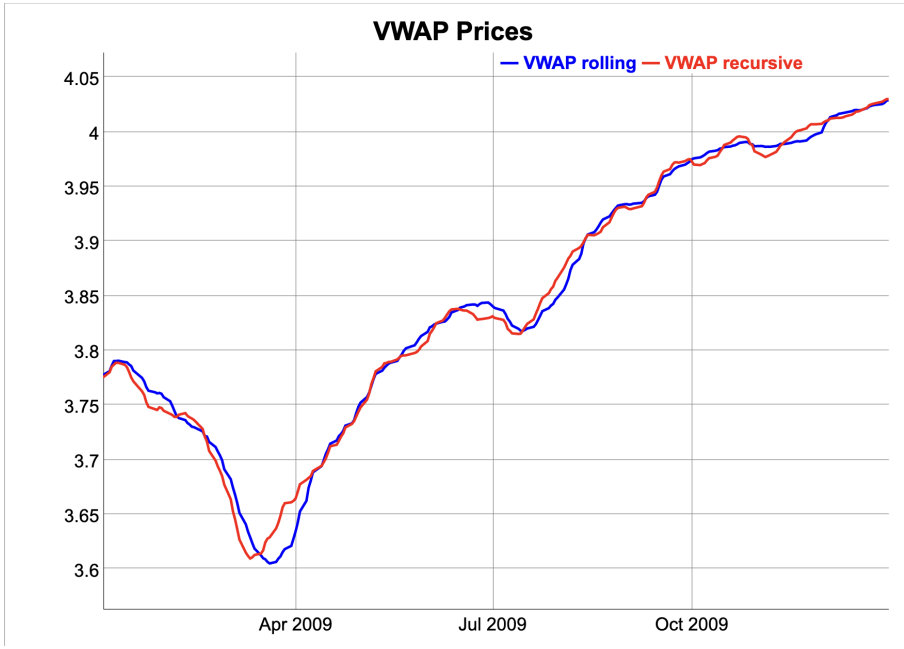
$$\mu^{v}_{t} = \lambda\mu^{v}_{t-1} + (1-\lambda)v_t$$
$$\mu^{pv}_{t} = \lambda\mu^{pv}_{t-1} + (1-\lambda)v_t p_t$$

The recursive *VWAP* prices are slightly different from those calculated as a convolution, because the convolution uses a fixed look-back interval.

The advantage of the recursive *VWAP* indicator is that it gradually "forgets" about large trading volumes far in the past.

The compiled C++ function `stats:::C_rfilter()` calculates the trailing weighted values recursively.

The function `HighFreq::run_mean()` also calculates the trailing weighted values recursively.




```
> # Calculate the VWAP prices recursively using C++ code
> lambda <- 0.9
> volumer <- .Call(stats:::C_rfilter, volum, lambda, c(as.numeric(vo
> pricer <- .Call(stats:::C_rfilter, volum*closep, lambda, c(as.nume
> vwapr <- pricer/volumer
> # Calculate the VWAP prices recursively using RcppArmadillo
> vwapcpp <- HighFreq::run_mean(closep, lambda=lambda, weightv=volum
> all.equal(vwapr, drop(vwapcpp))
> # Dygraphs plot the VWAP prices
> pricev <- xts(cbind(vwap, vwapr), zoo::index(ohlc))
> colnames(pricev) <- c("VWAP rolling", "VWAP recursive")
> dygraphs::dygraph(pricev["2009"], main="VWAP Prices") %>%
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

Simulating the VWAP Crossover Strategy

In the trend following *VWAP Crossover* strategy, the stock position switches depending if the current price is above or below the *VWAP*.

If the current price crosses above the *VWAP*, then the strategy switches its stock position to a fixed unit of long risk, and if it crosses below, to a fixed unit of short risk.

To prevent whipsaws and over-trading, the crossover strategy delays switching positions until the indicator repeats the same value for several periods.


```
> # Calculate the VWAP prices recursively using RcppArmadillo
> lambda <- 0.99
> vwapcpp <- HighFreq::run_mean(closep, lambda=lambda, weightv=volu
> # Calculate the positions from lagged indicator
> indic <- sign(closep - vwapcpp)
> lagg <- 2
> indic <- HighFreq::roll_sum(indic, lagg)
> # Calculate the positions, either: -1, 0, or 1
> posv <- rep(NA_integer_, nrows)
> posv[1] <- 0
> posv <- ifelse(indic == lagg, 1, posv)
> posv <- ifelse(indic == (-lagg), -1, posv)
> posv <- zoo::na.locf(posv, na.rm=FALSE)
> posv <- xts::xts(posv, order.by=zoo::index(closep))
> # Lag the positions to trade in next period
> posv <- rutils::lagit(posv, lagg=1)
> # Calculate the PnLs of VWAP strategy
> retp <- rutils::diffit(closep)  # VTI returns
> pnls <- retp*posv
> colnames(pnls) <- "VWAP"
> wealthv <- cbind(retp, pnls)
> colnamev <- colnames(wealthv)
```

```
> # Annualized Sharpe ratios of VTI and VWAP strategy
> sharper <- sqrt(252)*sapply(wealthv, function (x) mean(x)/sd(x))
> # Create colors for background shading
> crossd <- (rutils::diffit(posv) != 0)
> shadev <- posv[crossd]
> crossd <- c(zoo::index(shadev), end(posv))
> shadev <- ifelse(drop(zoo::coredata(shadev)) == 1, "lightgreen", "
> # Plot dygraph of VWAP strategy
> # Create dygraph object without plotting it
> dyplot <- dygraphs::dygraph(cumsum(wealthv), main=paste("VWAP Cros
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
> # Add shading to dygraph object
> for (i in 1:NROW(shadev)) {
+   dyplot <- dyplot %>% dyShading(from=crossd[i], to=crossd[i+1], c
+ }  # end for
> # Plot the dygraph object
> dyplot
```

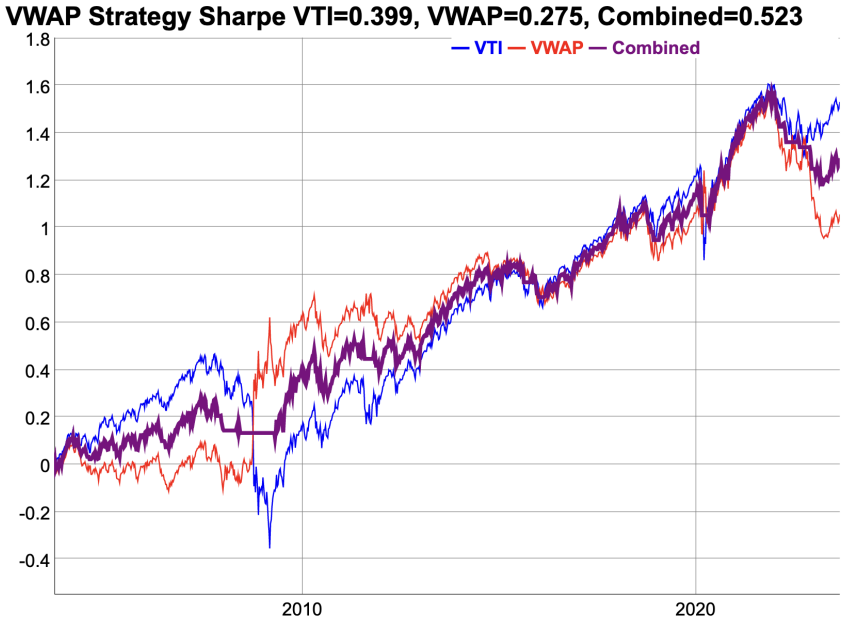
Combining VWAP Crossover Strategy with Stocks

Even though the *VWAP* strategy doesn't perform as well as a static buy-and-hold strategy, it can provide risk reduction when combined with it.

This is because the *VWAP* strategy has a negative correlation with respect to the underlying asset.

In addition, the *VWAP* strategy performs well in periods of extreme market selloffs, so it can provide a hedge for a static buy-and-hold strategy.

The *VWAP* strategy serves as a dynamic put option in periods of extreme market selloffs.




```
> # Calculate the correlation of VWAP strategy with VTI
> cor(retp, pnls)
> # Combine VWAP strategy with VTI
> wealthv <- cbind(retp, pnls, 0.5*(retp+pnls))
> colnames(wealthv) <- c("VTI", "VWAP", "Combined")
> sharper <- sqrt(252)*sapply(wealthv, function (x) mean(x)/sd(x))
```

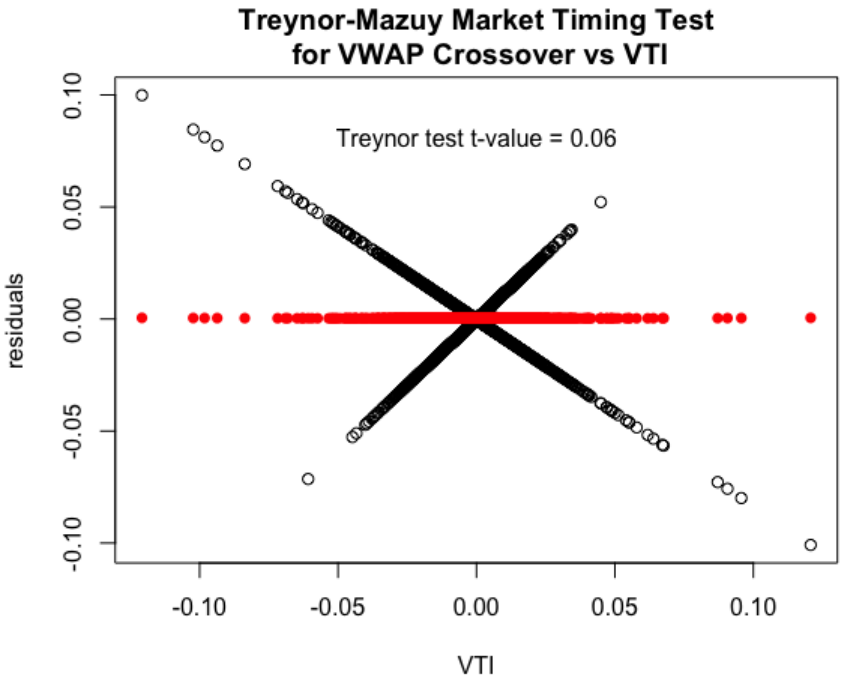
```
> # Plot dygraph of VWAP strategy combined with VTI
> colorv <- c("blue", "red", "purple")
> dygraphs::dygraph(cumsum(wealthv)[endd],
+   paste("VWAP Strategy Sharpe", paste(paste(names(sharper), round(
+   dyOptions(colors=colorv, strokeWidth=1) %>%
+   dySeries(name="Combined", label="Combined", strokeWidth=3) %>%
+   dyLegend(show="always", width=200)
```

VWAP Crossover Strategy Market Timing Skill

The VWAP crossover strategy shorts the market during significant selloffs, but otherwise doesn't display market timing skill.

The t-value of the *Treynor-Mazuy* test is negative, but not statistically significant.

```
> # Test VWAP crossover market timing of VTI using Treynor-Mazuy tes
> desm <- cbind(pnls, retp, retp^2)
> desm <- na.omit(desm)
> colnames(desm) <- c("VWAP", "VTI", "treynor")
> regmod <- lm(VWAP ~ VTI + treynor, data=desm)
> summary(regmod)
> # Plot residual scatterplot
> resids <- (desm$VWAP - regmod$coeff["VTI"]*retp)
> resids <- regmod$residuals
> # x11(width=6, height=6)
> plot.default(x=retp, y=resids, xlab="VTI", ylab="residuals")
> title(main="Treynor-Mazuy Market Timing Test\n for VWAP Crossover
> # Plot fitted (predicted) response values
> coefreg <- summary(regmod)$coeff
> fitv <- regmod$fitted.values - coefreg["VTI", "Estimate"]*retp
> tvalue <- round(coefreg["treynor", "t value"], 2)
> points.default(x=retp, y=fitv, pch=16, col="red")
> text(x=0.0, y=0.8*max(resids), paste("Treynor test t-value =", tvalue))
```



Simulation Function for VWAP Crossover Strategy

The *VWAP* strategy can be simulated by a single function, which allows the analysis of its performance depending on its parameters.

The function `sim_vwap()` performs a simulation of the *VWAP* strategy, given an *OHLC* time series of pricev, and the length of the look-back interval (`look_back`).

The function `sim_vwap()` returns the *VWAP* strategy positions and returns, in a two-column *xts* time series.

```
> sim_vwap <- function(ohlc, lambda=0.9, bidask=0.001, trend=1, lagg
+   closep <- log(quantmod::Cl(ohlc))
+   volum <- quantmod::Vo(ohlc)
+   retp <- rutils::diffit(closep)
+   nrows <- NROW(ohlc)
+   # Calculate the VWAP prices
+   vwap <- HighFreq::run_mean(closep, lambda=lambda, weightv=volum)
+   # Calculate the indicator
+   indic <- trend*sign(closep - vwap)
+   if (lagg > 1) {
+     indic <- HighFreq::roll_sum(indic, lagg)
+     indic[1:lagg] <- 0
+   }  # end if
+   # Calculate the positions, either: -1, 0, or 1
+   posv <- rep(NA_integer_, nrows)
+   posv[1] <- 0
+   posv <- ifelse(indic == lagg, 1, posv)
+   posv <- ifelse(indic == (-lagg), -1, posv)
+   posv <- zoo::na.locf(posv, na.rm=FALSE)
+   posv <- xts::xts(posv, order.by=zoo::index(closep))
+   # Lag the positions to trade on next day
+   posv <- rutils::lagit(posv, lagg=1)
+   # Calculate the PnLs of strategy
+   pnls <- retp*posv
+   costs <- 0.5*bidask*abs(rutils::diffit(posv))
+   pnls <- (pnls - costs)
+   # Calculate the strategy returns
+   pnls <- cbind(posv, pnls)
+   colnames(pnls) <- c("positions", "pnls")
+   pnls
+ }  # end sim_vwap
```

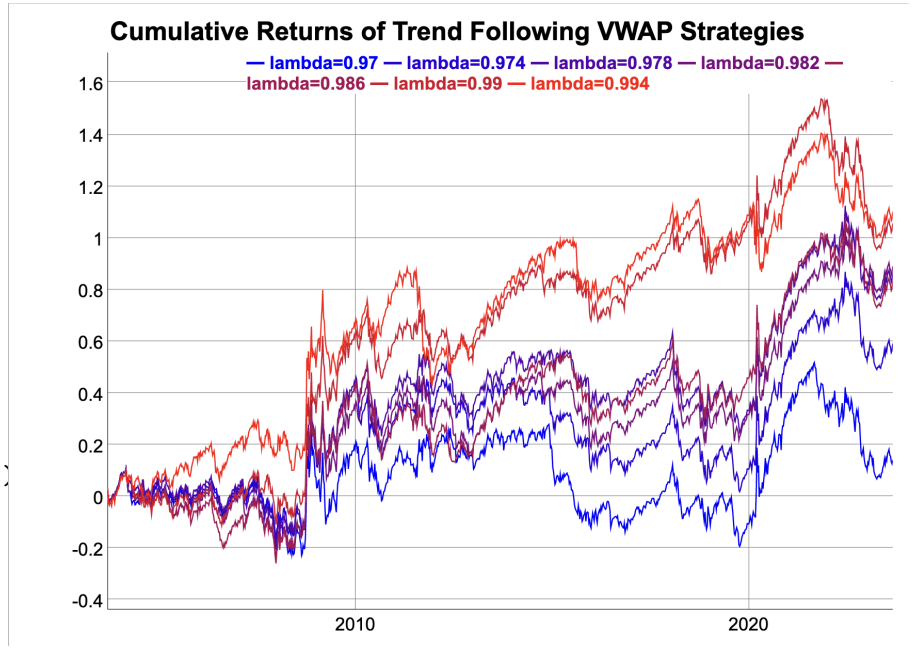
Simulating Multiple Trend Following VWAP Strategies

Multiple *VWAP* strategies can be simulated by calling the function `sim_vwap()` in a loop over a vector of λ parameters.

But `sim_vwap()` returns an *xts* time series, and `sapply()` cannot merge *xts* time series together.

So instead the loop is performed using `lapply()` which returns a list of *xts*, and the list is merged into a single *xts* using the functions `do.call()` and `cbind()`.

```
> source("/Users/jerzy/Develop/lecture_slides/scripts/ewma_model.R")
> lambdav <- seq(from=0.97, to=0.995, by=0.004)
> # Perform lapply() loop over lambdav
> pnls <- lapply(lambdav, function(lambda) {
+   # Simulate VWAP strategy and Calculate the returns
+   sim_vwap(ohlc=ohlc, lambda=lambda, bidask=0, lagg=2)[, "pnls"]
+ })  # end lapply
> pnls <- do.call(cbind, pnls)
> colnames(pnls) <- paste0("lambda=", lambdav)
```




```
> # Plot dygraph of multiple VWAP strategies
> colorv <- colorRampPalette(c("blue", "red"))(NCOL(pnls))
> dygraphs::dygraph(cumsum(pnls)[endd], main="Cumulative Returns of
+   dyOptions(colors=colorv, strokeWidth=1) %>%
+   dyLegend(show="always", width=500)
> # Plot VWAP strategies with custom line colors
> x11(width=6, height=5)
> plot_theme <- chart_theme()
> plot_theme$col$line.col <- colorv
> quantmod::chart_Series(cumsum(pnls), theme=plot_theme,
+   name="Cumulative Returns of VWAP Strategies")
> legend("topleft", legend=colnames(pnls), inset=0.1,
+   bg="white", cex=0.8, lwd=rep(6, NCOL(pnls)),
+   col=plot_theme$col$line.col, bty="n")
```

Bollinger Bands

The *Bollinger Bands* improve the moving average feature by adding information about the volatility.

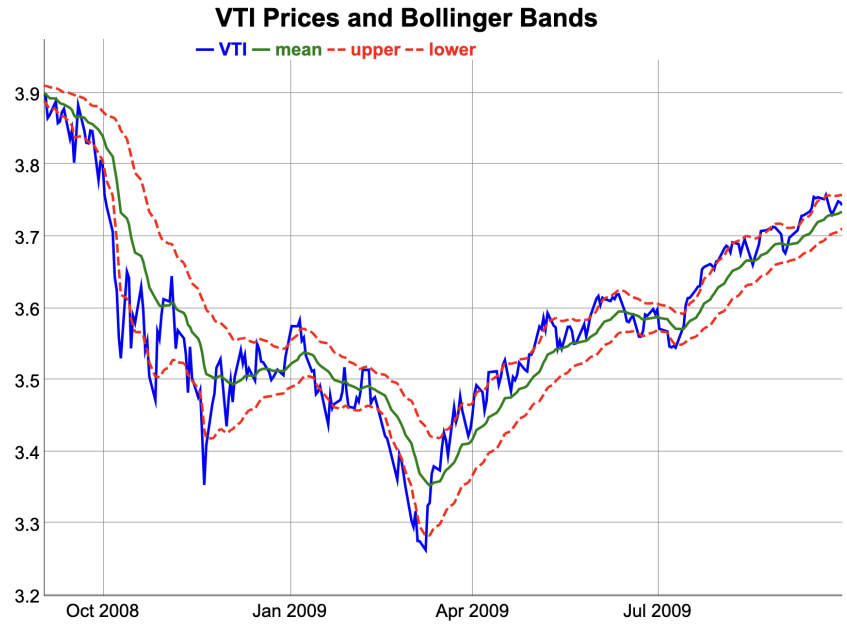
The Bollinger Bands are three time series, with the middle band equal to the trailing mean prices, the upper band equal to the mean prices plus the trailing standard deviation, and the lower band equal to the mean prices minus the standard deviation.

The Bollinger Bands are often used to indicate that prices are cheap if they are below the lower band, and rich (expensive) if they are above the upper band.

The decay parameter λ determines the rate of decay of the weights, with smaller values of λ producing faster decay, giving more weight to recent prices, and vice versa.

The functions `HighFreq::run_mean()` and `HighFreq::run_var()` calculate the trailing mean and variance by recursively updating the past estimates with the new values, using the weight decay factor λ.

```
> # Extract the log VTI prices
> pricev <- log(na.omit(rutils::etfenv$prices$VTI))
> nrows <- NROW(pricev)
> # Calculate the trailing mean prices
> lambda <- 0.9
> meanv <- HighFreq::run_mean(pricev, lambda=lambda)
> # Calculate the trailing volatilities
> volat <- HighFreq::run_var(pricev, lambda=lambda)
> volat <- sqrt(volat)
```




```
> # Dygraphs plot of Bollinger bands
> priceb <- cbind(pricev, meanv, meanv+volat, meanv-volat)
> colnames(priceb)[2:4] <- c("mean", "upper", "lower")
> colnamev <- colnames(priceb)
> dygraphs::dygraph(priceb["2008-09/2009-09"], main="VTI Prices and
+   dySeries(name=colnamev[1], strokeWidth=2, col="blue") %>%
+   dySeries(name=colnamev[2], strokeWidth=2, col="green") %>%
+   dySeries(name=colnamev[3], strokeWidth=2, strokePattern="dashed"
+   dySeries(name=colnamev[4], strokeWidth=2, strokePattern="dashed"
+   dyLegend(show="always", width=200)
```

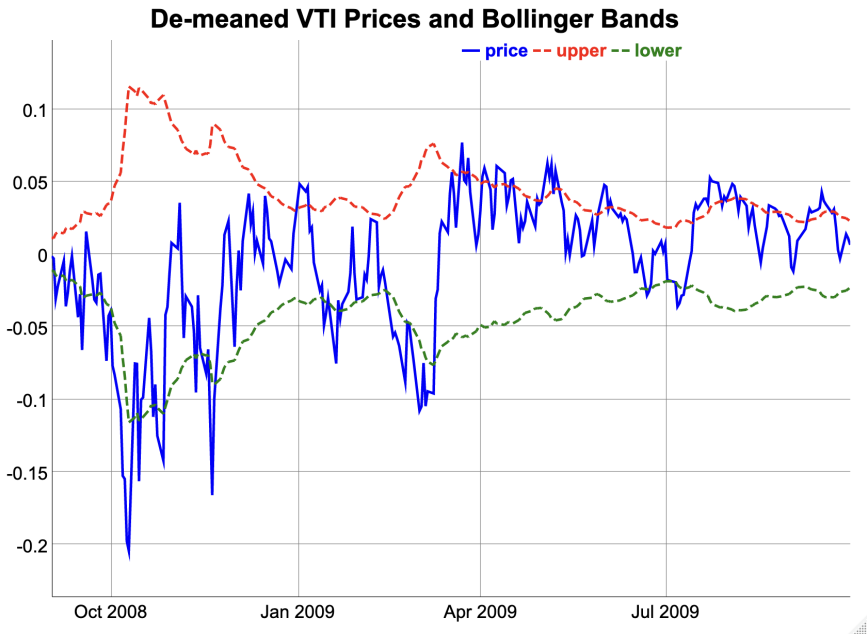
Bollinger Bands With Centered Prices

Centering (de-meaning) the prices provides a better view of the *Bollinger Bands*.

The centered prices tend to be range-bound between the *Bollinger Bands*.

When the centered price is below the lower band it's considered cheap, and if it's above the upper band it's considered rich (expensive).

When the centered price is close to zero it's considered fair (neutral).




```
> # Center the prices
> pricem <- pricev - meanv
> # Dygraphs plot of Bollinger bands
> priceb <- cbind(pricem, volat, -volat)
> colnames(priceb) <- c("price", "upper", "lower")
> colnamev <- colnames(priceb)
> dygraphs::dygraph(priceb["2008-09/2009-09"],
+   main="Centered VTI Prices and Bollinger Bands") %>%
+   dySeries(name=colnamev[1], strokeWidth=2, col="blue") %>%
+   dySeries(name=colnamev[2], strokeWidth=2, strokePattern="dashed"
+   dySeries(name=colnamev[3], strokeWidth=2, strokePattern="dashed"
+   dyLegend(show="always", width=200)
```

The Bollinger Band Strategy

The *Bollinger Band* strategy switches to long \$1 dollar of the stock when prices are cheap (below the lower band), and sells short -\$1 dollar of stock when prices are rich (expensive - above the upper band). It goes flat \$0 dollar of stock (unwinds) if the stock reaches a fair (mean) price.

The strategy is therefore always either long \$1 dollar of stock, or short -\$1 dollar of stock, or flat \$0 dollar of stock.

The upper and lower Bollinger bands can be chosen to be a multiple of the standard deviations above and below the mean prices.

The Bollinger strategy is a *mean reverting* (contrarian) strategy because it bets on prices reverting to their mean value.

The Bollinger strategy is a type of *statistical arbitrage* strategy.

Statistical arbitrage strategies try to exploit short-term anomalies in prices, when prices diverge from their equilibrium values and then revert back to them.

```
> # Calculate the trailing mean prices and volatilities
> lambda <- 0.1
> meanv <- HighFreq::run_mean(pricev, lambda=lambda)
> volat <- HighFreq::run_var(pricev, lambda=lambda)
> volat <- sqrt(volat)
> # Prepare the simulation parameters
> pricen <- as.numeric(pricev) # Numeric price
> pricem <- pricen - meanv # Centered price
> threshv <- volat
> posv <- integer(nrows) # Stock positions
> posv[1] <- 0 # Initial position
> # Calculate the positions from Bollinger bands
> for (it in 2:nrows) {
+   if (pricem[it-1] > threshv[it-1]) {
+     # Enter short
+     posv[it] <- (-1)
+   } else if (pricem[it-1] < (-threshv[it-1])) {
+     # Enter long
+     posv[it] <- 1
+   } else if ((posv[it-1] < 0) && (pricem[it-1] < 0)) {
+     # Unwind short
+     posv[it] <- 0
+   } else if ((posv[it-1] > 0) && (pricem[it-1] > 0)) {
+     # Unwind long
+     posv[it] <- 0
+   } else {
+     # Do nothing
+     posv[it] <- posv[it-1]
+   }  # end if
+ }  # end for
> # Calculate the number of trades
> ntrades <- sum(abs(rutils::diffit(posv)) > 0)
> # Calculate the pnls
> retp <- rutils::diffit(pricev)
> pnls <- retp*posv
```

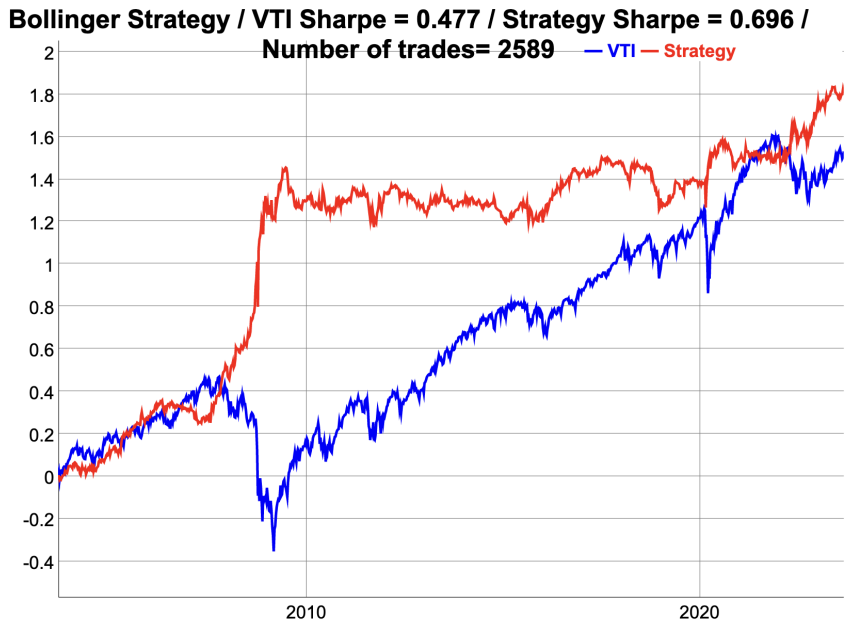
Bollinger Strategy Performance

The *Bollinger Band* strategy has two parameters: the weight decay factor λ and the standard deviation multiple n.

The best strategy parameters can be found using backtest simulation, but it risks overfitting the parameters to the in-sample data, and poor performance out-of-sample.

The *Bollinger Band* strategy has performed well for *VTI* with $\lambda = 0.1$, but it hasn't performed well for most other stocks.

The *Bollinger Band* strategy had its best performance for *VTI* prior to the financial crisis of 2008-2009.



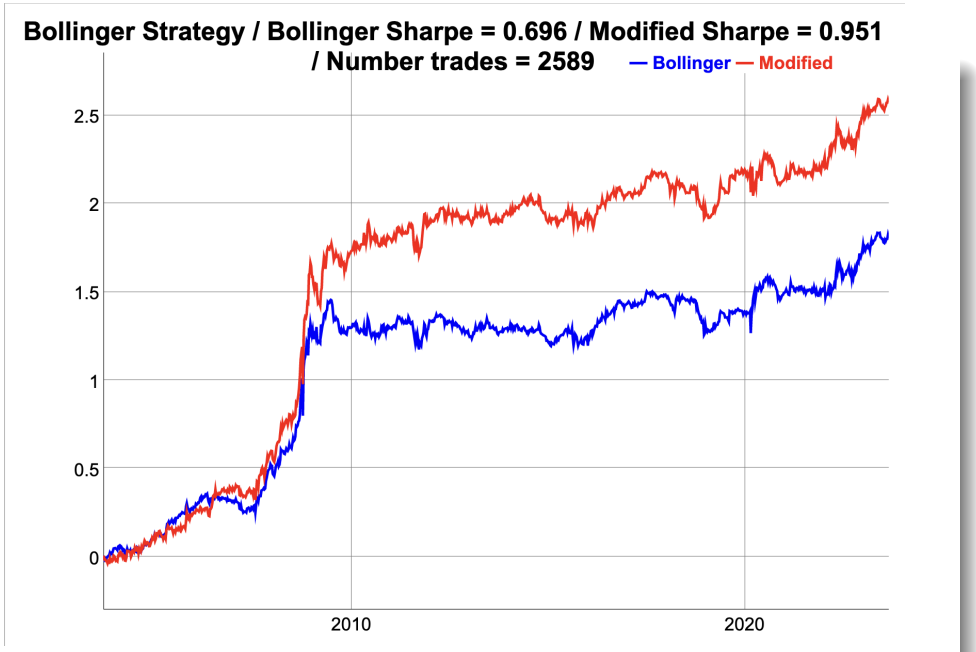

```
> # Calculate the Sharpe ratios
> wealthv <- cbind(retp, pnls)
> colnames(wealthv) <- c("VTI", "Strategy")
> sharper <- sqrt(252)*sapply(wealthv, function(x) mean(x)/sd(x[x<0]
> sharper <- round(sharper, 3)
> # Dygraphs plot of Bollinger strategy
> colnamev <- colnames(wealthv)
> captiont <- paste("Bollinger Strategy", "/ \n",
+   paste0(paste(colnamev[1:2], "Sharpe =", sharper), collapse=" / "
+   "Number trades =", ntrades)
> endd <- rutils::calc_endpoints(wealthv, interval="weeks")
> dygraphs::dygraph(cumsum(wealthv)[endd], main=captiont) %>%
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

The Modified Bollinger Strategy

Unwinding the position when the stock reaches a fair (mean) price doesn't necessarily produce better performance.

In the modified Bollinger Strategy the positions are held until the price reaches the opposite extreme, so that the strategy is always either long \$1 dollar of stock or short \$1 dollar of stock.

```
> # Simulate the modified Bollinger strategy
> posv <- integer(nrows) # Stock positions
> posv[1] <- 0 # Initial position
> for (it in 2:nrows) {
+   if (pricem[it-1] > threshv[it-1]) {
+     # Enter short
+     posv[it] <- (-1)
+   } else if (pricem[it-1] < (-threshv[it-1])) {
+     # Enter long
+     posv[it] <- 1
+   } else {
+     # Do nothing
+     posv[it] <- posv[it-1]
+   }  # end if
+ }  # end for
> # Calculate the PnLs
> pnls2 <- retp*posv
```




```
> # Calculate the Sharpe ratios
> wealthv <- cbind(pnls, pnls2)
> colnames(wealthv) <- c("Bollinger", "Modified")
> sharper <- sqrt(252)*sapply(wealthv, function(x) mean(x)/sd(x[x<0]
> sharper <- round(sharper, 3)
> # Dygraphs plot of Bollinger strategy
> colnamev <- colnames(wealthv)
> captiont <- paste("Bollinger Strategy", "/ \n",
+   paste0(paste(colnamev[1:2], "Sharpe =", sharper), collapse=" / "
+   "Number trades =", ntrades)
> endd <- rutils::calc_endpoints(wealthv, interval="weeks")
> dygraphs::dygraph(cumsum(wealthv)[endd], main=captiont) %>%
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

Fast Bollinger Strategy Simulation

The Bollinger Strategy is path-dependent so simulating it requires performing a loop, which can be slow in `R`.

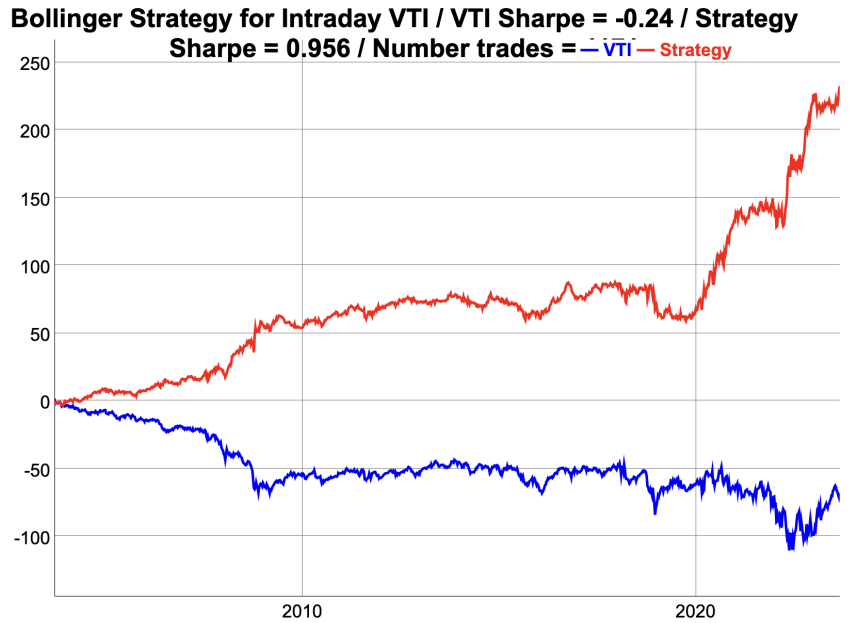
The modified Bollinger Strategy can be simulated quickly using the compiled `C++` functions `ifelse()` and `zoo::na.locf()`.

```
> # Simulate the modified Bollinger strategy quickly
> posf <- rep(NA_integer_, nrows)
> posf[1] <- 0
> posf <- ifelse(pricem > threshv, -1, posf)
> posf <- ifelse(pricem < -threshv, 1, posf)
> posf <- zoo::na.locf(posf)
> # Lag the positions to trade in the next period
> posf <- rutils::lagit(posf, lagg=1)
> # Compare the positions
> all.equal(posv, posf)
```

Bollinger Strategy For Intraday VTI Returns

The Bollinger Strategy has performed well for intraday returns of the *VTI* ETF, because intraday prices exhibit significant mean-reversion.

This simulation doesn't account for transaction costs, which would likely erase all profits if market orders were used for trade executions. But the strategy could be profitable if limit orders were used for trade executions.



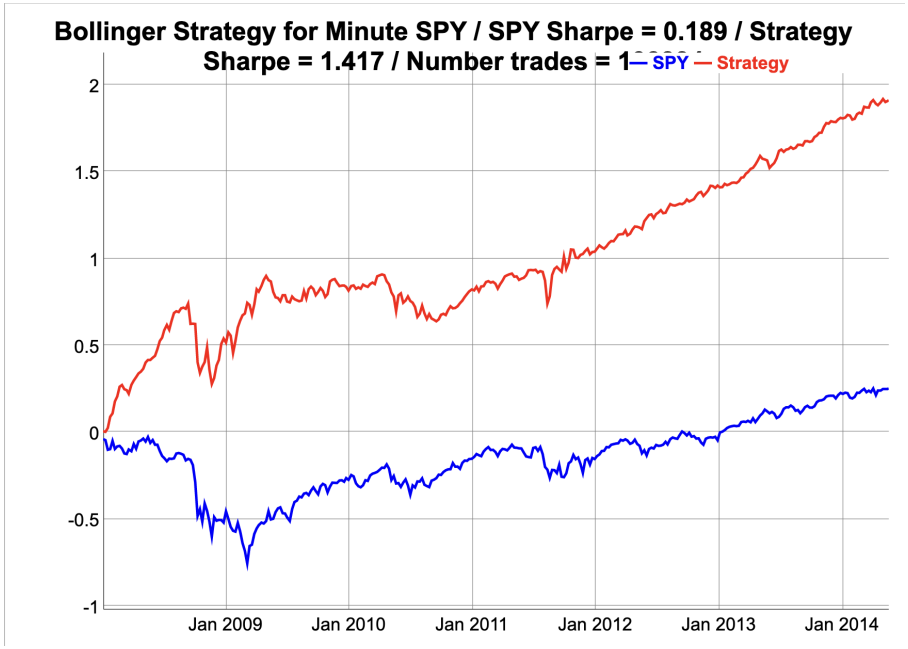

```
> # Calculate the intraday open-to-close VTI returns
> ohlc <- rutils::etfenv$VTI
> openp <- quantmod::Op(ohlc)
> highp <- quantmod::Hi(ohlc)
> lowp <- quantmod::Lo(ohlc)
> closep <- quantmod::Cl(ohlc)
> retp <- (closep - openp)
> # Calculate the cumulative VTI returns
> pricev <- cumsum(retp)
> nrows <- NROW(pricev)
> lambda <- 0.1
> meanv <- HighFreq::run_mean(pricev, lambda=lambda)
> volat <- HighFreq::run_var(pricev, lambda=lambda)
> volat <- sqrt(volat)
> # Calculate the positions from Bollinger bands
> threshv <- volat
> pricem <- zoo::coredata(pricev - meanv)
> posv <- rep(NA_integer_, nrows)
> posv[1] <- 0
> posv <- ifelse(pricem > threshv, -1, posv)
> posv <- ifelse(pricem < -threshv, 1, posv)
> posv <- zoo::na.locf(posv)
> # Lag the positions to trade in the next period
> posv <- rutils::lagit(posv, lagg=1)
> # Calculate the number of trades and the PnLs
> ntrades <- sum(abs(rutils::diffit(posv)) > 0)
> pnls <- retp*posv
```

```
> # Calculate the Sharpe ratios
> wealthv <- cbind(retp, pnls)
> colnames(wealthv) <- c("VTI", "Strategy")
> nyears <- as.numeric(end(pricev)-start(pricev))/365
> sharper <- sqrt(nrows/nyears)*sapply(wealthv, function(x) mean(x)/
> sharper <- round(sharper, 3)
> # Dygraphs plot of Bollinger strategy
> colnamev <- colnames(wealthv)
> captiont <- paste("Bollinger Strategy for Intraday VTI", "/ \n",
+   paste0(paste(colnamev[1:2], "Sharpe =", sharper), collapse=" / "
+   "Number trades =", ntrades)
> endd <- rutils::calc_endpoints(wealthv, interval="weeks")
> dygraphs::dygraph(cumsum(wealthv)[endd], main=captiont) %>%
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

Bollinger Strategy For Intraday SPY Returns

The Bollinger Strategy has performed well for 1-minute prices of the *SPY* ETF, because intraday prices exhibit significant mean-reversion.

This simulation doesn't account for transaction costs, which would likely erase all profits if market orders were used for trade executions. But the strategy could be profitable if limit orders were used for trade executions.




```
> # Calculate the trailing mean prices and volatilities of SPY
> pricev <- log(quantmod::Cl(HighFreq::SPY))
> nrows <- NROW(pricev)
> lambda <- 0.1
> meanv <- HighFreq::run_mean(pricev, lambda=lambda)
> volat <- HighFreq::run_var(pricev, lambda=lambda)
> volat <- sqrt(volat)
> # Calculate the positions from Bollinger bands
> threshv <- volat
> pricem <- zoo::coredata(pricev - meanv)
> posv <- rep(NA_integer_, nrows)
> posv[1] <- 0
> posv <- ifelse(pricem > threshv, -1, posv)
> posv <- ifelse(pricem < -threshv, 1, posv)
> posv <- zoo::na.locf(posv)
> # Lag the positions to trade in the next period
> posv <- rutils::lagit(posv, lagg=1)
> # Calculate the number of trades and the PnLs
> ntrades <- sum(abs(rutils::diffit(posv)) > 0)
> retp <- rutils::diffit(pricev)
> pnls <- retp*posv
```

```
> # Calculate the Sharpe ratios
> wealthv <- cbind(retp, pnls)
> colnames(wealthv) <- c("SPY", "Strategy")
> nyears <- as.numeric(end(pricev)-start(pricev))/365
> sharper <- sqrt(nrows/nyears)*sapply(wealthv, function(x) mean(x)/
> sharper <- round(sharper, 3)
> # Dygraphs plot of Bollinger strategy
> colnamev <- colnames(wealthv)
> captiont <- paste("Bollinger Strategy for Minute SPY", "/ \n",
+   paste0(paste(colnamev[1:2], "Sharpe =", sharper), collapse=" / "
+   "Number trades =", ntrades)
> endd <- rutils::calc_endpoints(wealthv, interval="weeks")
> dygraphs::dygraph(cumsum(wealthv)[endd], main=captiont) %>%
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

The Hampel Filter Bands

The Bollinger bands can be improved by using nonparametric measures of location (*median*) and dispersion (*MAD*).

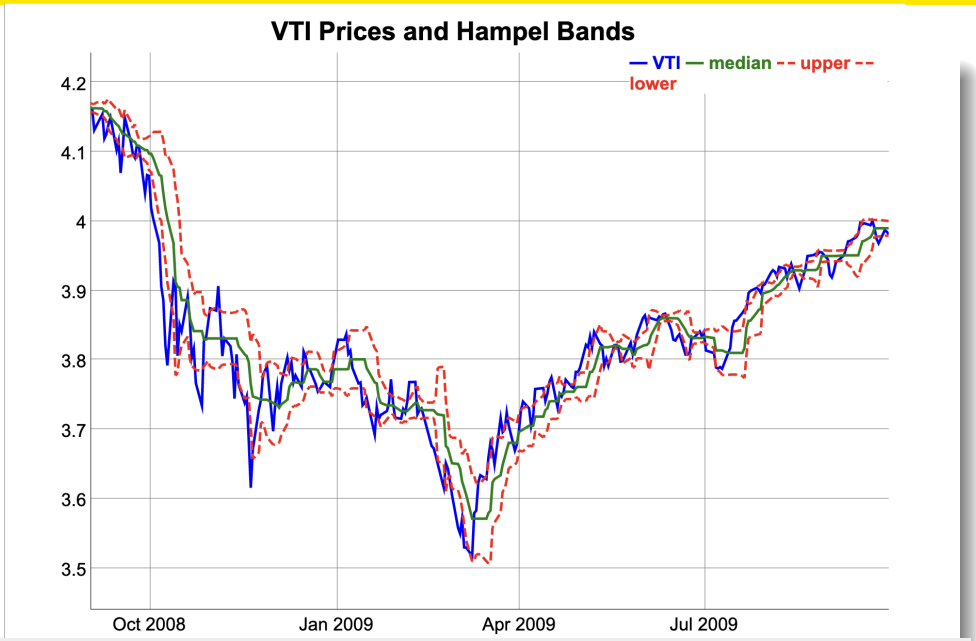
The *Median Absolute Deviation* (*MAD*) is a nonparametric measure of dispersion (variability):

$$\text{MAD} = \text{median}(\text{abs}(p_t - \text{median}(\mathbf{p})))$$

The *Hampel z-score* is equal to the deviation from the median divided by the *MAD*:

$$z_i = \frac{p_t - \text{median}(\mathbf{p})}{\text{MAD}}$$

A time series of *z-scores* over past data can be calculated using a trailing look-back window.




```
> # Extract time series of VTI log prices
> pricev <- log(na.omit(rutils::etfenv$prices$VTI))
> nrows <- NROW(pricev)
> # Define look-back window
> look_back <- 11
> # Calculate time series of trailing medians
> medianv <- HighFreq::roll_mean(pricev, look_back, method="nonpara
> # medianv <- TTR::runMedian(pricev, n=look_back)
> # Calculate time series of MAD
> madv <- HighFreq::roll_var(pricev, look_back=look_back, method="
> # madv <- TTR::runMAD(pricev, n=look_back)
> # Calculate time series of z-scores
> zscores <- ifelse(madv > 0, (pricev - medianv)/madv, 0)
> zscores[1:look_back, ] <- 0
> tail(zscores, look_back)
> range(zscores)
```

```
> # Plot histogram of z-scores
> histp <- hist(zscores, col="lightgrey",
+   xlab="z-scores", breaks=50, xlim=c(-4, 4),
+   ylab="frequency", freq=FALSE, main="Hampel Z-Scores histogram")
> lines(density(zscores, adjust=1.5), lwd=3, col="blue")
> # Dygraphs plot of Hampel bands
> priceb <- cbind(pricev, medianv, medianv+madv, medianv-madv)
> colnames(priceb)[2:4] <- c("median", "upper", "lower")
> colnamev <- colnames(priceb)
> dygraphs::dygraph(priceb["2008-09/2009-09"], main="VTI Prices and
+   dySeries(name=colnamev[1], strokeWidth=2, col="blue") %>%
+   dySeries(name=colnamev[2], strokeWidth=2, col="green") %>%
+   dySeries(name=colnamev[3], strokeWidth=2, strokePattern="dashed"
+   dySeries(name=colnamev[4], strokeWidth=2, strokePattern="dashed"
+   dyLegend(show="always", width=200)
```

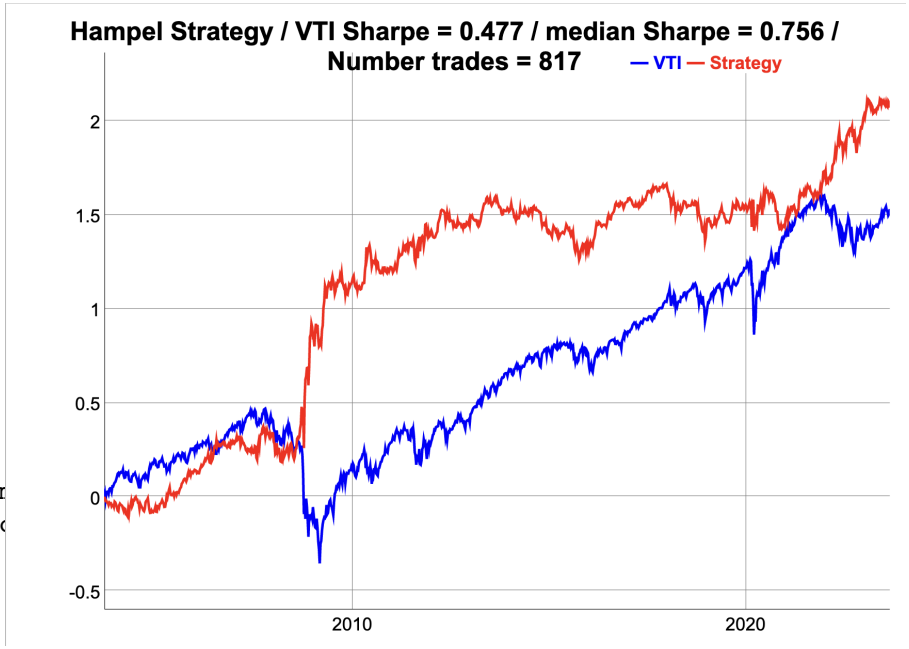
Hampel Filter Strategy

The Hampel filter strategy is a contrarian strategy that uses Hampel z-scores to establish long and short positions.

The Hampel strategy has two meta-parameters: the look-back interval and the threshold level.

The best choice of the meta-parameters can be determined through simulation.

```
> # Calculate the time series of trailing medians and MAD
> look_back <- 3
> medianv <- HighFreq::roll_mean(pricev, look_back, method="nonparam
> madv <- HighFreq::roll_var(pricev, look_back=look_back, method="no
> # Calculate the time series of z-scores
> zscores <- ifelse(madv > 0, (pricev - medianv)/madv, 0)
> zscores[1:look_back, ] <- 0
> range(zscores)
> # Calculate the positions
> threshv <- 1
> posv <- rep(NA_integer_, nrows)
> posv[1] <- 0
> posv[zscores > threshv] <- (-1)
> posv[zscores < -threshv] <- 1
> posv <- zoo::na.locf(posv)
> posv <- rutils::lagit(posv)
> # Calculate the number of trades and the PnLs
> ntrades <- sum(abs(rutils::diffit(posv)) > 0)
> retp <- rutils::diffit(pricev)
> pnls <- retp*posv
```

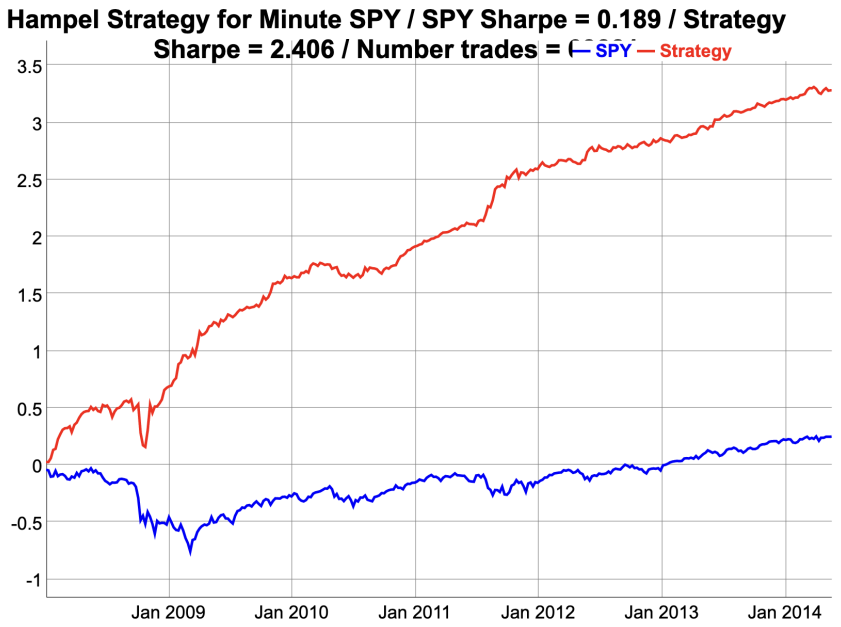



```
> # Calculate the Sharpe ratios
> wealthv <- cbind(retp, pnls)
> colnames(wealthv) <- c("VTI", "Strategy")
> sharper <- sqrt(252)*sapply(wealthv, function(x) mean(x)/sd(x[x<0]
> sharper <- round(sharper, 3)
> # Dygraphs plot of Hampel strategy
> captiont <- paste("Hampel Strategy", "/ \n",
+   paste0(paste(colnamev[1:2], "Sharpe =", sharper), collapse=" / "
+   "Number trades =", ntrades)
> endd <- rutils::calc_endpoints(wealthv, interval="weeks")
> colnamev <- colnames(wealthv)
> dygraphs::dygraph(cumsum(wealthv)[endd], main=captiont) %>%
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

Hampel Filter Strategy For Intraday SPY Returns

The Hampel Filter strategy has performed well for 1-minute prices of the *SPY* ETF, because intraday prices exhibit significant mean-reversion.

This simulation doesn't account for transaction costs, which would likely erase all profits if market orders were used for trade executions. But the strategy could be profitable if limit orders were used for trade executions.




```
> # Calculate the trailing mean prices and volatilities of SPY
> pricev <- log(quantmod::Cl(HighFreq::SPY))
> nrows <- NROW(pricev)
> # Calculate the price medians and MAD
> look_back <- 3
> medianv <- HighFreq::roll_mean(pricev, look_back, method="nonparar
> madv <- HighFreq::roll_var(pricev, look_back=look_back, method="no
> # Calculate the time series of z-scores
> zscores <- ifelse(madv > 0, (pricev - medianv)/madv, 0)
> zscores[1:look_back, ] <- 0
> # Calculate the positions
> threshv <- 1
> posv <- rep(NA_integer_, nrows)
> posv[1] <- 0
> posv[zscores < -threshv] <- 1
> posv[zscores > threshv] <- (-1)
> posv <- zoo::na.locf(posv)
> posv <- rutils::lagit(posv)
> # Calculate the number of trades and the PnLs
> ntrades <- sum(abs(rutils::diffit(posv)) > 0)
> retp <- rutils::diffit(pricev)
> pnls <- retp*posv
```

```
> # Calculate the Sharpe ratios
> wealthv <- cbind(retp, pnls)
> colnames(wealthv) <- c("SPY", "Strategy")
> nyears <- as.numeric(end(pricev)-start(pricev))/365
> sharper <- sqrt(nrows/nyears)*sapply(wealthv, function(x) mean(x)/
> sharper <- round(sharper, 3)
> # Dygraphs plot of Hampel strategy
> colnamev <- colnames(wealthv)
> captiont <- paste("Hampel Strategy for Minute SPY", "/ \n",
+   paste0(paste(colnamev[1:2], "Sharpe =", sharper), collapse=" / "
+   "Number trades =", ntrades)
> endd <- rutils::calc_endpoints(wealthv, interval="weeks")
> dygraphs::dygraph(cumsum(wealthv)[endd], main=captiont) %>%
+   dyOptions(colors=c("blue", "red"), strokeWidth=2) %>%
+   dyLegend(show="always", width=200)
```

Homework Assignment

Required

- Study all the lecture slides in *FRE7241_Lecture_3.pdf*, and run all the code in *FRE7241_Lecture_3.R*