http://mintegration.eu/

Strategy Replication - Evolutionary Optimization based on Financial Sentiment Data

---

Author:
Nick Kirk
nk@mintegration.eu

September 27, 2015

```
################################################################################
##                                                      _______                 ##
##                                                     |       \                ##
##   ____ _____  ____  _    _  ____ _____| |_  ___ |   []    )               ##
##  / ___)  _  )|  _ \| |  | |/ ___)  _  |  _)/ _ \|   ||   /                ##
## | |   ( (/ ( | | ) | |__| ( (___| ( | | |_| |_| |   ||   |                ##
## |_|    \____)| ||_/|___/|_|\____)__||_|\___)___/|___||___|                ##
##              | |                2015, Nick Kirk                           ##
##                                 http://mintegration.eu                    ##
## This work is licensed under the Creative Commons Attribution 4.0          ##
## International License. To view a copy of this license, visit              ##
## http://creativecommons.org/licenses/by/4.0/.                              ##
##                                                                           ##
## Original Source Paper:                                                    ##
## ----------------------                                                    ##
## Title:   Computing trading strategies based on financial sentiment data   ##
##          using evolutionary optimization                                  ##
## Author:  Ronald Hochreiter                                                ##
## Book:    Mendel 2015, Recent Advances in Soft Computing                   ##
##          (The series "Advances in Intelligent Systems and Computing",     ##
##          Volume 378 2015, Springer)                                       ##
##          http://arxiv.org/abs/1504.02972:                                 ##
##                                                                           ##
################################################################################

source("R/replicatoR_mendel_ronh_f.R")
source(file = "R/quandl.R")

# Load required libraries
library(xts)
library(PerformanceAnalytics)
library(quadprog)
library(xtable)

# ------------------------------------------------------------------------------
# Only call once - Download EOD prices from Quandl
# ------------------------------------------------------------------------------
if (! file.exists("data/djia_prices.rds")){
    prices_xts <- NULL
    download_EOD(input_file = "data/djia.csv",
                 output_file = "data/djia_prices",
                 start_date = "2010-01-01")
}

# Read the saved EOD prices
stocks_xts <- readRDS(file = "data/djia_prices.rds")

# Get the list of assets
(tickers <- unique(stocks_xts[, "TICKER"]))
```

```
##  [1] "AXP"  "BA"   "CAT"  "CSCO" "CVX"  "DD"   "DIS"  "GE"   "GS"   "HD"
## [11] "IBM"  "INTC" "JNJ"  "JPM"  "KO"   "MCD"  "MMM"  "MRK"  "MSFT" "NKE"
## [21] "PFE"  "PG"   "T"    "TRV"  "UNH"  "UTX"  "V"    "VZ"   "WMT"  "XOM"
```

```
# Calculate the log returns
calc_log_ret(tickers)

# Save each asset to the global env
save_assets(tickers)

# Form a timeseries of only the returns
returns <- NULL
for (i in 1:length(tickers)) {
```

```
    returns <- cbind(returns, get(tickers[i])[-1, "Log_Ret"])
}
colnames(returns) <- tickers
returns <- returns["2010/2014"]
print(returns[1:20, 1:4])
```

```
##                         AXP             BA          CAT          CSCO
## 2010-01-05 -0.0022018358  0.0322269570  0.011884690 -0.0044651992
## 2010-01-06  0.0160353290  0.0298833782  0.003033369 -0.0065306355
## 2010-01-07  0.0160194699  0.0396838429  0.004030232  0.0044943896
## 2010-01-08 -0.0007148815 -0.0096931292  0.011165852  0.0052856395
## 2010-01-11 -0.0115081587 -0.0119214280  0.060917039 -0.0028426415
## 2010-01-12  0.0131754212 -0.0072547721 -0.029914393 -0.0159872231
## 2010-01-13  0.0030889890  0.0120077107  0.001444971  0.0180185055
## 2010-01-14  0.0124957420  0.0065189279 -0.005631098  0.0125026835
## 2010-01-15 -0.0068179411 -0.0120936267 -0.023507449 -0.0222906899
## 2010-01-19  0.0133569648 -0.0027990469  0.013383099  0.0182746202
## 2010-01-20  0.0004654410 -0.0074472831 -0.019389123 -0.0178648681
## 2010-01-21 -0.0192629870 -0.0167508104 -0.049920321 -0.0177727348
## 2010-01-22 -0.0884787287 -0.0244519321 -0.046813228 -0.0430311329
## 2010-01-25 -0.0209486582  0.0001730852  0.016272413  0.0008703221
## 2010-01-26  0.0081697648 -0.0012122263  0.012794120 -0.0039224282
## 2010-01-27  0.0148498235  0.0705742462 -0.044109996  0.0108578700
## 2010-01-28 -0.0325915837  0.0101213813 -0.030011749 -0.0275909773
## 2010-01-29  0.0061260013 -0.0318312032  0.007300705 -0.0022227171
## 2010-02-01  0.0144987382  0.0179890378  0.013310712  0.0115045517
## 2010-02-02  0.0209770737 -0.0029216057  0.011456600  0.0126777654
```

```
# StockTwits data
stocktwits_daily_final_xts <- NULL
load_stocktwits_daily()

# Plot the returns
#plot.zoo(returns)
```

```
###############################################################################
# For each asset, optimize the trading rule-set
###############################################################################
results_evo <- c()
results_long <- c()
for(asset in 1:(length(tickers))) {

    ticker <- tickers[asset]
    sentiment <- stocktwits_daily_final_xts[stocktwits_daily_final_xts$SYMBOL %in%
                                                ticker, ]["2010/2013", c(2, 3, 5, 6, 7)]
    storage.mode(sentiment) <- "numeric"

    # Convert Sentiment data (0...1, relative values)
    sentiment_in <- cbind(sentiment$BULLISH_INTENSITY/4,
                          sentiment$BEARISH_INTENSITY/4,
                          sentiment$BULL_SCORED_MESSAGES/sentiment$TOTAL_SCANNED_MESSAGES,
                          sentiment$BEAR_SCORED_MESSAGES/sentiment$TOTAL_SCANNED_MESSAGES)

    names(sentiment_in) <- c("i_bull", "i_bear", "r_bull", "r_bear")
    returns_in <- returns[, ticker]["2010/2013"]
    population <- c()

    # Initialize the population of 100 genotypes
    for(i in 1:100) {
        genotype <- op_random()

        # Add the genotype to the population
        population <- rbind(population, genotype)
    }

    # The maximum number of generations (i.e. iterations)
    max_generations <- 100
    generations <- 1

    repeat {
        ###########################################################################
        # Each genotype encodes a trading rule-set.
        # Let's optimize the rule-set using the in-sample test data.
        ###########################################################################
        results_pop <- c()
        for(j in 1:nrow(population)) {
            perf <- round(evaluate_ruleset(sentiment_in,
                                           returns_in,
                                           population[j, ])[[1]], 3)
            results_pop <- rbind(results_pop, c(population[j, ], perf))
        }

        # Exit when the maximum number of generations is reached
        if(generations >= max_generations) break
        generations <- generations + 1

        ###########################################################################
        # For each generation, the new population is constructed as follows:
        #
        # i:   The 10 best chromosomes of the previous population (elitist selection)
        # ii:  20 of the binary flip mutation operators
        # iii: 20 of the random mutation operators
        # iv:  20 of the division mutation operators
        # v:   10 random chromosomes,
        #
        # such that the population size is 80.
        ###########################################################################
```

```
        # Order the previous population by the Sharpe ratio
        # (i.e. the fitness function)
        prev_population <- results_pop[order(results_pop[, ncol(results_pop)],
                                             decreasing = TRUE), ]

        # Step i
        population <- prev_population[1:10, 1:8]

        # Step ii, iii and iv
        for(i in 1:20) {
            population <- rbind(population, op_mutation_flip(prev_population))
            population <- rbind(population, op_mutation_random(prev_population))
            population <- rbind(population, op_mutation_divide(prev_population))
        }

        # Step v
        for(i in 1:10) {
            population <- rbind(population, op_random())
        }
    }

    # The new population has been formed.
    # Now order the genotypes by the Sharpe ratio and select the best.
    # (i.e. the best genotype in the current population is the optimal one)
    results_pop <- results_pop[order(results_pop[, ncol(results_pop)],
                                     decreasing = TRUE), ][1, ]
    print(results_pop)
    results_evo <- rbind(results_evo, results_pop)

    # Long-only buy-and-hold strategy
    perf_long <- round(evaluate_ruleset(sentiment_in, returns_in, NULL,
                                        evolutionary = FALSE)[[1]], 3)
    results_long <- rbind(results_long, perf_long)

} # end loop
rownames(results_evo) <- tickers
rownames(results_long) <- tickers

# Save the results
if (! file.exists("data/results_evo.rds")){
    print("saving ....")
    saveRDS(results_evo, file = "data/results_evo.rds")
    saveRDS(results_long, file = "data/results_long.rds")
}
```

```
##############################################################################
# 1) Tabulate the single stock "in-sample" results:
#    - Long-only buy-and-hold strategy v.s Optimal rule-based trading strategy.
##############################################################################

# Restart from here (to save time) once the results have been saved
results_evo <- readRDS(file = "data/results_evo.rds")
results_long <- readRDS(file = "data/results_long.rds")

results_insample <- cbind(results_long, results_evo[, 9:11])
colnames(results_insample) <- c("r_long", "vol_long", "sr_long", "r_evo", "vol_evo", "sr_evo")
```

```
print(xtable(results_insample,
             caption = 'Single stock in-sample results of the evolutionary optimization',
             digits = c(0, rep(3, 6))),
      include.rownames=TRUE,
      caption.placement = 'bottom', size = "normalsize")
```

	r_long	vol_long	sr_long	r_evo	vol_evo	sr_evo
AXP	1.052	0.015	0.047	1.462	0.013	0.065
BA	1.341	0.014	0.054	1.688	0.013	0.068
CAT	0.433	0.016	0.025	0.938	0.014	0.042
CSCO	-0.195	0.017	-0.002	0.985	0.010	0.058
CVX	0.660	0.012	0.041	0.789	0.012	0.047
DD	0.952	0.014	0.049	1.187	0.011	0.070
DIS	1.261	0.013	0.057	1.454	0.013	0.063
GE	0.837	0.013	0.042	1.067	0.013	0.051
GS	-0.105	0.016	0.003	0.546	0.010	0.036
HD	1.896	0.012	0.077	2.348	0.012	0.090
IBM	0.416	0.010	0.031	1.419	0.008	0.086
INTC	0.274	0.013	0.020	0.649	0.009	0.045
JNJ	0.566	0.008	0.052	0.581	0.008	0.053
JPM	0.227	0.017	0.017	0.679	0.016	0.032
KO	0.549	0.009	0.044	0.670	0.008	0.053
MCD	0.679	0.008	0.053	0.568	0.006	0.066
MMM	0.723	0.012	0.047	0.840	0.012	0.053
MRK	0.470	0.011	0.035	0.632	0.010	0.048
MSFT	0.209	0.012	0.017	0.482	0.011	0.032
NKE	1.263	0.014	0.054	1.725	0.011	0.077
PFE	0.755	0.011	0.047	0.893	0.011	0.054
PG	0.449	0.008	0.042	0.453	0.007	0.050
T	0.466	0.009	0.039	0.586	0.007	0.055
TRV	0.870	0.012	0.054	1.012	0.011	0.067
UNH	1.228	0.015	0.055	2.280	0.013	0.088
UTX	0.595	0.013	0.039	0.523	0.011	0.040
V	1.245	0.015	0.048	1.340	0.010	0.068
VZ	0.831	0.009	0.056	0.939	0.009	0.065
WMT	0.532	0.008	0.044	0.721	0.007	0.063
XOM	0.516	0.010	0.037	0.544	0.010	0.039

Table 1: Single stock in-sample results of the evolutionary optimization

```
################################################################################
# Tabulate the accuracy of the model
################################################################################

# in-sample
# --------------
results_pop <- calc_results_with_optimal_evo("2010/2013")
total_accuracy_in <- calc_accuracy("2010/2013", results_pop)
total_accuracy_in

##  [1] 0.5319149 0.5471125 0.5207700 0.5309017 0.5531915 0.5440729 0.5501520
##  [8] 0.5420466 0.5298886 0.5481256 0.5420466 0.5136778 0.5288754 0.5217832
## [15] 0.5420466 0.5177305 0.5592705 0.5278622 0.5157042 0.5248227 0.5207700
## [22] 0.5086120 0.5562310 0.5521783 0.5400203 0.5116515 0.5146910 0.5582573
## [29] 0.5471125 0.5197568

# out-of-sample
# --------------
results_pop <- calc_results_with_optimal_evo("2014")
total_accuracy_out <- calc_accuracy("2014", results_pop)
total_accuracy_out

##  [1] 0.4940239 0.5179283 0.5139442 0.5258964 0.5139442 0.5139442 0.5776892
##  [8] 0.4940239 0.4661355 0.5418327 0.4980080 0.5418327 0.5697211 0.5577689
## [15] 0.5577689 0.4900398 0.4701195 0.5378486 0.4980080 0.5298805 0.5258964
## [22] 0.5298805 0.4940239 0.5577689 0.5019920 0.4940239 0.4820717 0.4980080
## [29] 0.5179283 0.4940239
```

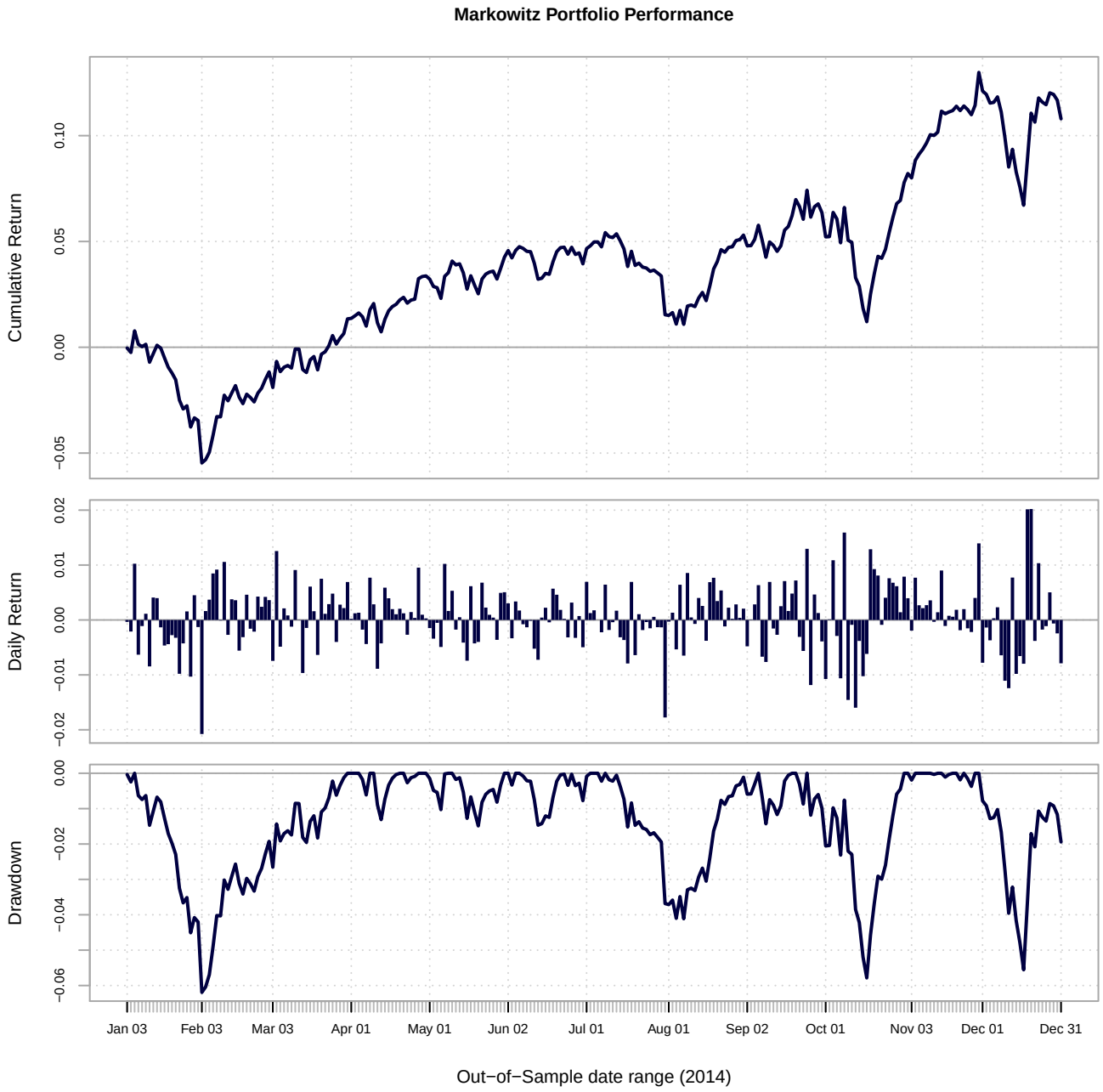
```
accuracy <- data.frame(mean(total_accuracy_in), mean(total_accuracy_out))
colnames(accuracy) <- c("In-Sample", "Out-of_Sample")
print(xtable(accuracy,
             caption = 'Evolutionary Model Accuracy'),
      include.rownames=FALSE,
      caption.placement = 'bottom', size = "normalsize")
```

In-Sample	Out-of_Sample
0.53	0.52

Table 2: Evolutionary Model Accuracy

```r
################################################################################
# Use the best (i.e. optimal) genotype in the out-of-sample data.
#
# Form an equally weighted portfolio out of the single investment evolutionary
# strategies. Each day, modify the weights so that the portfilio comprises only
# the assets that are in a long position.
#
# Compare with;
# 1)buy-and-hold optimal Markowitz portfolio
# 2)buy-and-hold 1-over-N portfolio
################################################################################

results_outofsample <- matrix(NA, nrow = 5, ncol = 3)
colnames(results_outofsample) <- c("Markowitz", "1-over-N", "Evolutionary")
rownames(results_outofsample) <- c("Semi Deviation", "Downside Deviation (Rf=0%)",
                                   "Maximum Drawdown", "Historical VaR (95%)",
                                   "Historical ES (95%)")

# Daily data (252 days per year)
T <- nrow(returns["2010/2013"]) / 4
N <- length(tickers)

# ------------------------
# 1) Markowitz portfolio
# ------------------------

# Asset mean returns
mu <- as.vector(apply(X = (returns["2010/2013"] * T), 2, mean))

# Target mean return (mean of the mean vectors)
mu_target <- mean(T * results_insample[, "sr_evo"] * results_insample[, "vol_evo"])

# Covariance matrix
Sigma <- cov(returns["2010/2013"])

# Solution using quadratic programming
A <- cbind(rep(1, N), mu, diag(N))
b <- c(1, mu_target, rep(0, N))
opt <- solve.QP(Sigma, rep(0, N), A, b, 2)
w <- matrix(opt$solution)
rownames(w) <- tickers
w <- round(w, 4) * 100


# Now use the optimal weights with the out-of-sample data
prtf_ret_simple <- Return.rebalancing(xts(returns["2014"],
                                          order.by = unique(index(returns["2014"])),
                                          format = "%Y/%m/%d"),
                                      c(w),
                                      geometric = TRUE)

results_outofsample <- set_risk_metrics(prtf_ret_simple[2:nrow(prtf_ret_simple), ],
                                        results_outofsample, 1)
```

```r
# Plot performance chart
charts.PerformanceSummary(prtf_ret_simple[2:nrow(prtf_ret_simple), ],
                          colorset = rich10equal,
                          main="Markowitz Portfolio Performance",
                          xlab = "Out-of-Sample date range (2014)")
```

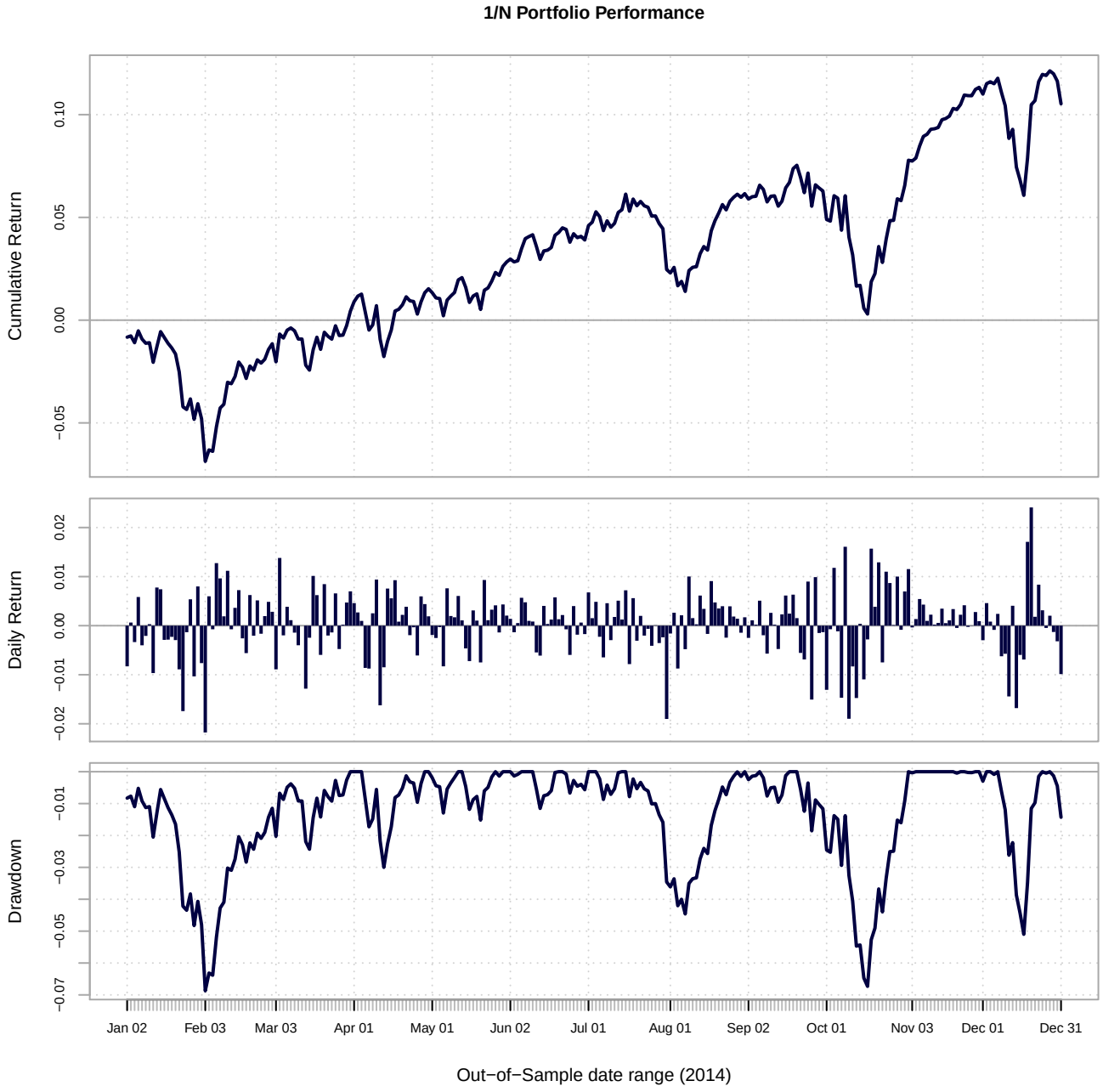



```
# Print the weights
w <- matrix(w)
rownames(w) <- tickers
w <- as.matrix(w[w > 0, ])
colnames(w) <- "Portfolio weight [%]"
print(xtable(t(w),
             caption = 'Optimal Markowitz portfolio using daily return data from 2010-2013'),
      include.rownames=TRUE, include.colnames=TRUE,
      caption.placement = 'bottom', size = "normalsize")
```

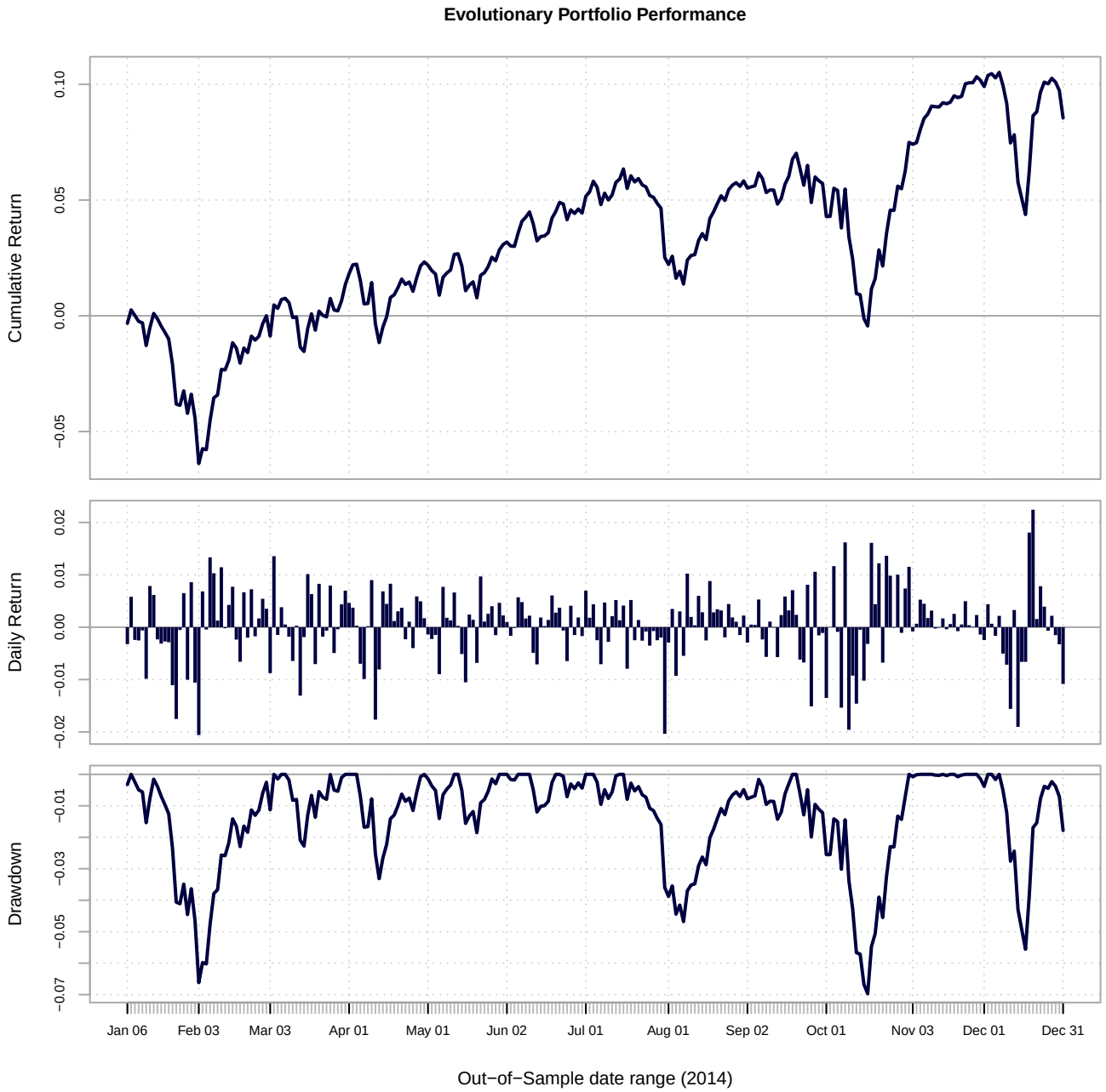
	HD	JNJ	MCD	NKE	PG	UNH	V	VZ	WMT
Portfolio weight [%]	9.62	18.54	23.68	0.46	11.11	3.48	1.18	14.33	17.61

Table 3: Optimal Markowitz portfolio using daily return data from 2010-2013

```r
# ------------------------
# 2) 1/N portfolio
# ------------------------
w <- c(rep(1/N, N))
prtf_ret_simple <- Return.rebalancing(xts(returns["2014"],
                                          order.by = unique(index(returns["2014"])),
                                          format = "%Y/%m/%d"),
                                      w,
                                      geometric = FALSE)

results_outofsample <- set_risk_metrics(prtf_ret_simple, results_outofsample, 2)

charts.PerformanceSummary(prtf_ret_simple, colorset = rich10equal,
                          main="1/N Portfolio Performance",
                          xlab = "Out-of-Sample date range (2014)")
```



```r
# ------------------------
# 3) Evolutionary
# ------------------------
results_pop <- calc_results_with_optimal_evo("2014")
row_sums <- apply(results_pop, 1, sum)
w <- results_pop / row_sums

prtf_ret_simple <- Return.rebalancing(xts(returns["2014"],
                                          order.by = unique(index(returns["2014"])),
                                          format = "%Y/%m/%d"),
                                      w,
                                      geometric = FALSE)

results_outofsample <- set_risk_metrics(prtf_ret_simple, results_outofsample, 3)

charts.PerformanceSummary(prtf_ret_simple, colorset = rich10equal,
                          main="Evolutionary Portfolio Performance",
                          xlab = "Out-of-Sample date range (2014)")
```

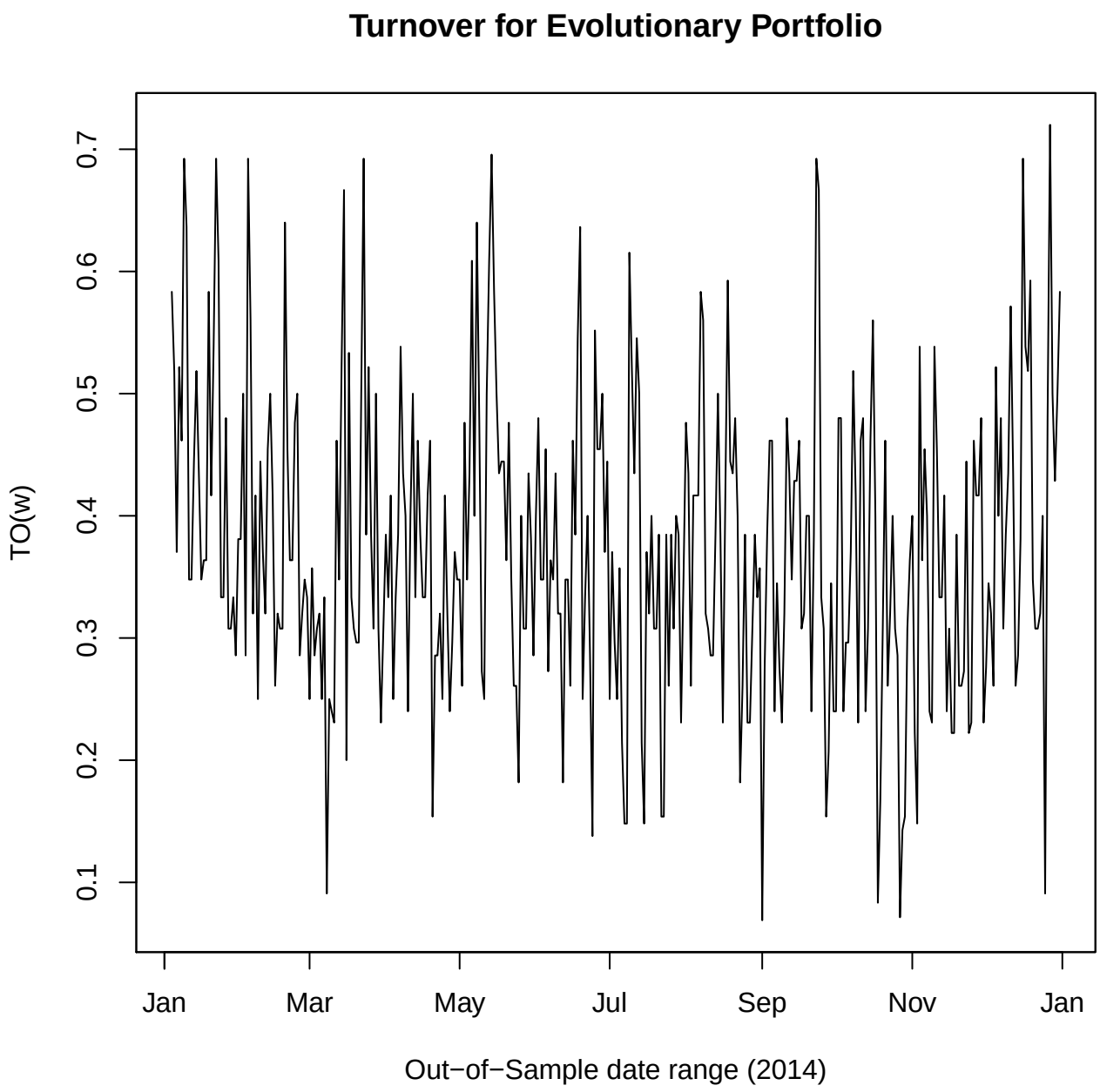


```
print(xtable(results_outofsample,
             caption = 'Selected risk metrics for the different out-of-sample tests',
             digits = c(0, 4, 4, 4)),
      include.rownames=TRUE,
      caption.placement = 'bottom', size = "normalsize")
```

	Markowitz	1-over-N	Evolutionary
Semi Deviation	0.0042	0.0049	0.0050
Downside Deviation (Rf=0%)	0.0040	0.0047	0.0049
Maximum Drawdown	0.0619	0.0687	0.0697
Historical VaR (95%)	-0.0097	-0.0106	-0.0110
Historical ES (95%)	-0.0128	-0.0158	-0.0164

Table 4: Selected risk metrics for the different out-of-sample tests

```
plot(TO(w), main = "Turnover for Evolutionary Portfolio",
     xlab = "Out-of-Sample date range (2014)")
```



Acknowledgements

I would like to thank Ronald Hochreiter for kindly responding to my emails and for writing a really neat paper.