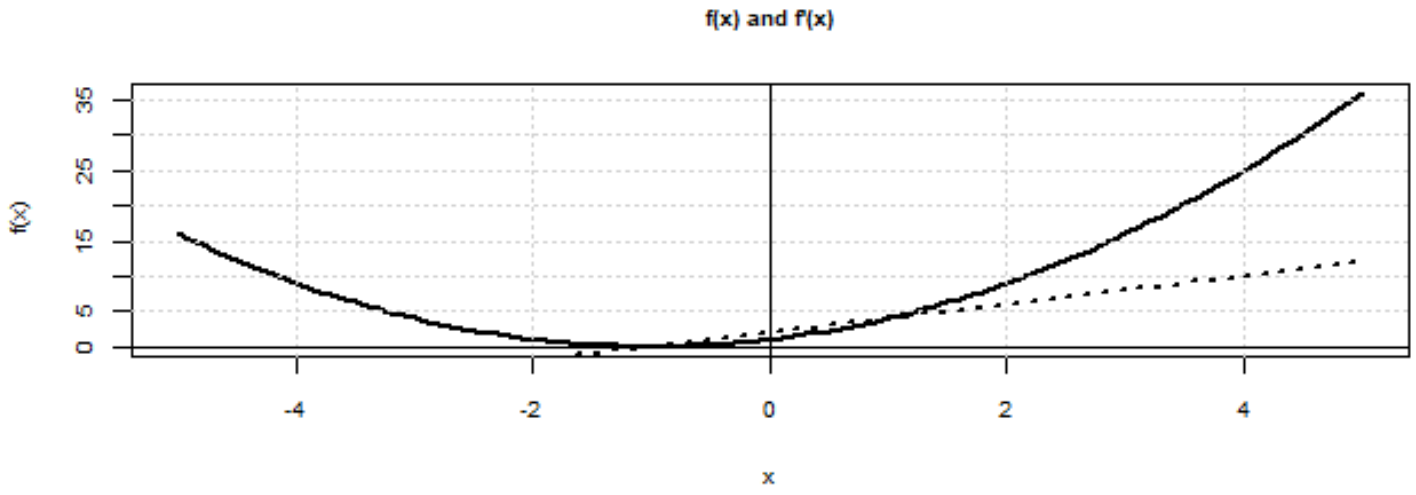
Optimization

```
# Create the function
f <- function(x) {
  return ((1 + x)^2)
}

# Create derivative
fp <- function(x) {
  return(2 * (1 + x))
}

par(mfrow = c(1, 1))
# Plot function and derivative
x <- seq(-5, 5, 0.1)
plot(x, f(x), type = 'l', lwd = 2,
     main = "f(x) and f'(x)",
     cex.main = 0.8,
     cex.lab = 0.8,
     cex.axis = 0.8)
grid()
lines(x, fp(x), lty = 3, lwd = 2)
abline(h = 0)
abline(v = 0)
```



Newton's Method

```
f <- function(x) {
    return(x ^ 2 - 4 * x + 1)
}

uniroot(f, c(-8, 1))
```

```
$root
[1] 0.2679509

$f.root
[1] -6.068318e-06

$iter
[1] 10

$init.it
[1] NA

$estim.prec
[1] 6.103516e-05
```

```
uniroot(f, c(-1, 2))
```

```
$root
[1] 0.2679363

$f.root
[1] 4.455014e-05

$iter
[1] 6

$init.it
[1] NA

$estim.prec
[1] 6.103516e-05
```

```
# Newton's method with first order approximation

newton <- function(f, tol = 1E-12, x0 = 1, N = 20) {
  # N = total number of iterations
  # x0 = initial guess
  # tol = abs(xn+1 - xn)
```

```
  # f = function to be evaluated for a root

  h <- 0.001
  i <- 1; x1 <- x0
  p <- numeric(N)
  while( i <= N ) {
    df_dx <- (f(x0 + h) - f(x0)) / h
    x1 <- (x0 - (f(x0) / df_dx))
    p[i] <- x1
    i <- i + 1
    if( abs(x1 - x0 < tol)) {
      break
    }
    x0 <- x1
  }
  return(p[1:(i-1)])
}

newton(f, x0 = -10)
```

```
[1] -4.1247552 -1.3070553 -0.1069219  0.2346811  0.2676451  0.2679493  0.2679492
```

```
newton(f, x0 = 10)
```

```
[1] 6.187738
```

```
newton(f, x0 = 0.25)
```

```
[1] 0.2678622 0.2679492 0.2679492
```

Symbolic Math

```
# Create an expression
e <- expression(sin(x))

D(e, "x")
```

```
cos(x)
```

```
f_expr <- expression(x ^ 2 + 4 * x - 1)

eval(f_expr, list(x = 2))
```

```
[1] 11
```

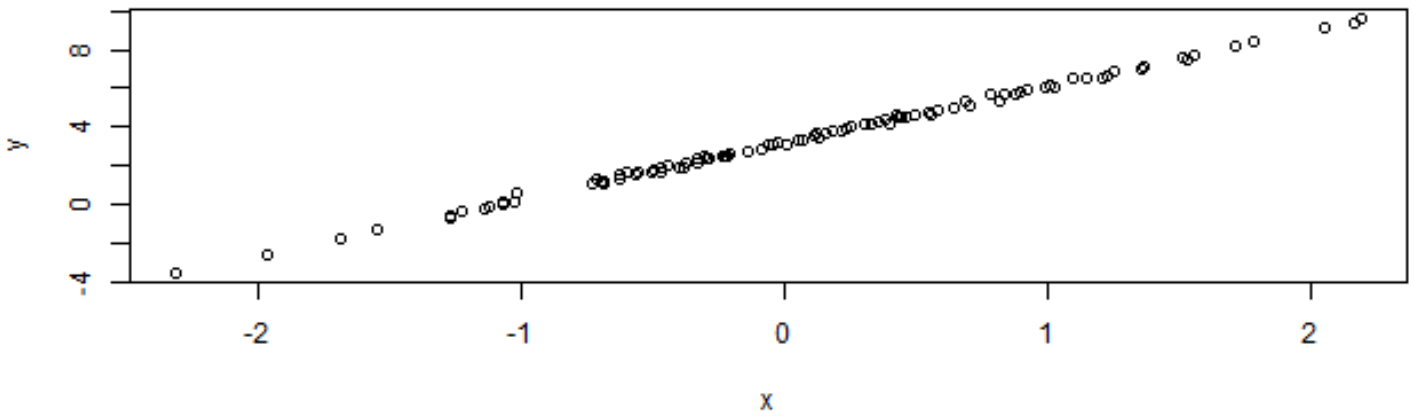
```
newton_alternate <- function(f, tol = 1E-12, x0 = 1, N = 20) {
  # N = total number of iterations
```

```
  # x0 = initial guess
  # tol = abs(xn+1 - xn)
  # f = expression to be evaluated for a root

  # Compute the symbolic derivative
  df_dx = D(f, "x")

  i <- 1; x1 <- x0
  p <- numeric(N)

  while (i <= N) {
    x1 <- (x0 - eval(f, list(x = x0)) /
      eval(df_dx, list(x = x0)))
    p[i] <- x1
    i <- i + 1
    if (abs(x1 - x0) < tol) {
      break
    }
    x0 <- x1
  }
  return(p[1:(i-1)])
}

newton_alternate(f_expr, x0 = 10)
```

```
[1] 4.2083333 1.5068512 0.4663159 0.2468156 0.2360937 0.2360680 0.2360680
[8] 0.2360680
```

Brute Force

```
# Create a set of random ponits x
set.seed(123)

x <- rnorm(100, 0, 1)

# Make y a function of x
y <- 3.2 + 2.9 * x + rnorm(100, 0, 0.1)

plot(x, y)
```




```
objective_function <- function(y, x, a, b) {
  value <- sum((y - (a * x + b)) ^ 2)
  return(value)
}

# Create a range of a and b values and loop through all of them

a <- seq(-10, 10, 0.25)
b <- seq(-10, 10, 0.25)

output <- list()
z <- 1
for(i in 1:length(a)) {
  for(j in 1:length(b)) {
    output[[z]] <- c(objective_function(y, x, a[i], b[j]),
                     a[i], b[j])
    z <- z + 1
  }
}

# Create a matridx  out of the list and find the minimum value
mat <- do.call(rbind, output)
colnames(mat) <- c("obj", "a", "b")

smallest <- which(mat[, "obj"] == min(mat[, "obj"]))

mat[smallest, ]
```

```
     obj        a        b
```

```
2.324319 3.000000 3.250000
```

```
a <- seq(-5, 5, 0.1)
b <- seq(-5, 5, 0.1)

output <- list()
z <- 1
for(i in 1:length(a)) {
  for(j in 1:length(b)) {
    output[[z]] <- c(objective_function(y, x, a[i], b[j]),
                     a[i], b[j])
    z <- z + 1
  }
}

# Create a matridx  out of the list and find the minimum value
mat <- do.call(rbind, output)
colnames(mat) <- c("obj", "a", "b")

smallest <- which(mat[, "obj"] == min(mat[, "obj"]))

mat[smallest, ]
```

```
      obj         a         b
0.9372788 2.9000000 3.2000000
```

R Optimization

```
args(optim)
```

```
function (par, fn, gr = NULL, ..., method = c("Nelder-Mead",
    "BFGS", "CG", "L-BFGS-B", "SANN", "Brent"), lower = -Inf,
    upper = Inf, control = list(), hessian = FALSE)
NULL
```

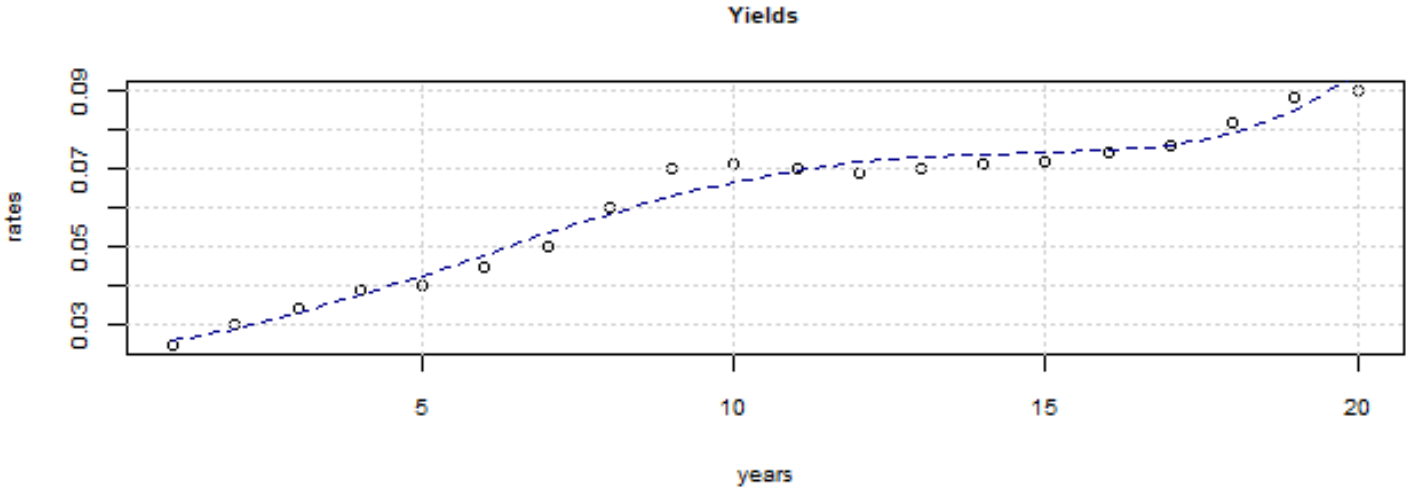
Curve-fitting

```
# Create fictitious yields
rates <- c(0.025, 0.03, 0.034, 0.039, 0.04,
           0.045, 0.05, 0.06, 0.07, 0.071,
           0.07, 0.069, 0.07, 0.071, 0.072,
           0.074, 0.076, 0.082, 0.088, 0.09)
maturities <- 1:20
```

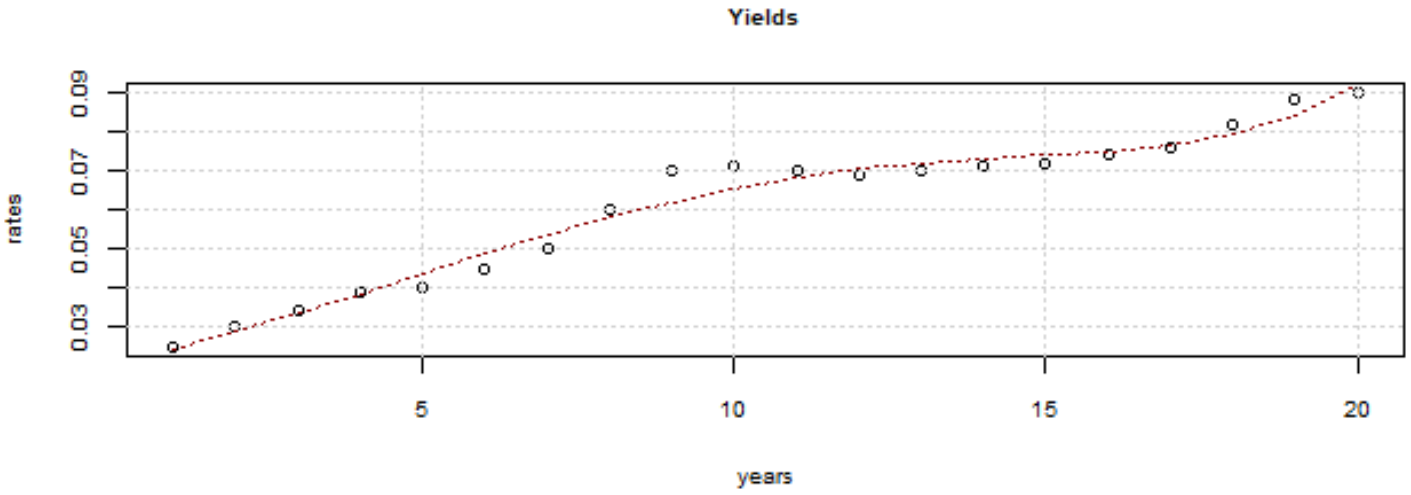
```
plot(maturities, rates,
     xlab = "years",
     main = "Yields",
     cex.main = 0.8,
     cex.lab = 0.8,
     cex.axis = 0.8)
grid()

poly_5 <- function(x, p) {
  f <- p[1] + p[2] * x + p[3] * x^2 +
    p[4] * x^3 + p[5] * x^4 + p[6] * x^5
  return(f)
}

obj_5 <- function(x, y, p) {
  error <- ( y - poly_5(x, p))^2
  return(sum(error))
}

# Fit the paramters. Assume 0 for all inital values
out_5 <- optim(obj_5, par = c(0, 0, 0, 0, 0, 0),
               x = maturities, y = rates)

out_5
```

```
$par
[1]  2.430124e-02  1.313895e-03  5.522933e-04  7.568574e-07 -4.211948e-06
[6]  1.533096e-07

$value
[1] 0.0001731166

$counts
function gradient
     501       NA

$convergence
[1] 1

$message
NULL
```

```
lines(poly_5(maturities, out_5$par), lwd = 1.5, lty = 2, col = "darkblue")
```

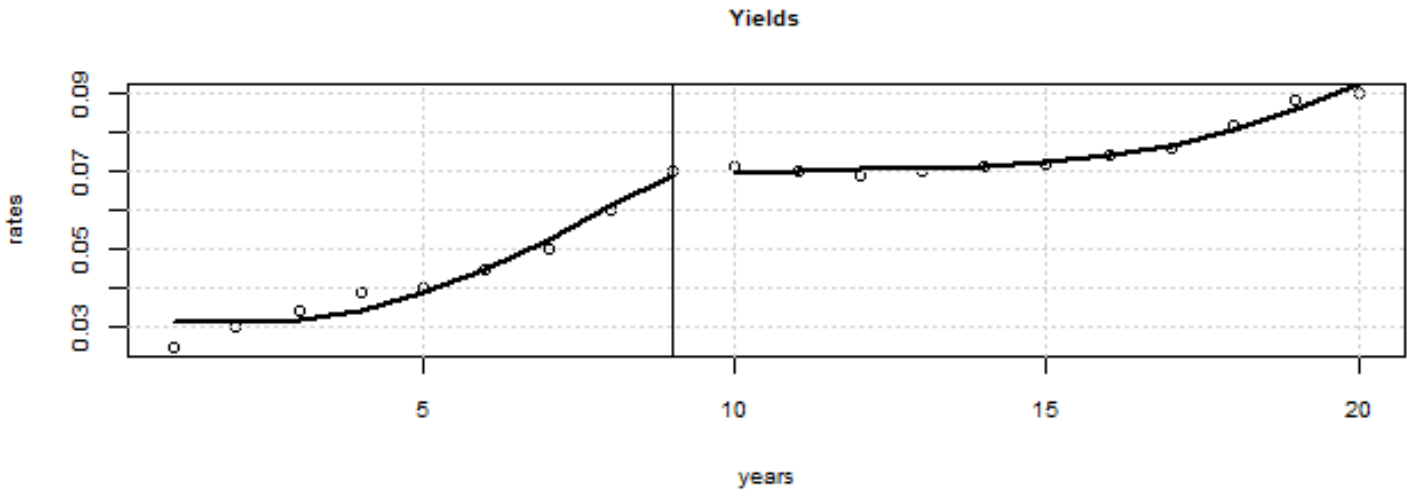



```
plot(maturities, rates,
     xlab = "years",
     main = "Yields",
     cex.main = 0.8,
     cex.lab = 0.8,
     cex.axis = 0.8)
grid()

poly_7 <- function(x, p) {
  f <- p[1] + p[2] * x +
    p[3] * x^2 + p[4] * x^3 +
    p[5] * x^4 + p[6] * x^5 +
    p[7] * x^7
  return(f)
}

obj_7 <- function(x, y, p) {
  error <- (y - poly_7(x, p)) ^ 2
  return(sum(error))
}

# Fit the parameters. Assume 0 for all initial values.
out_7 <- optim(obj_7, par = c(0, 0, 0, 0, 0, 0, 0),
               x = maturities, y = rates)
lines(poly_7(maturities, out_7$par), lwd = 1.5, lty = 3, col = "darkred")
```

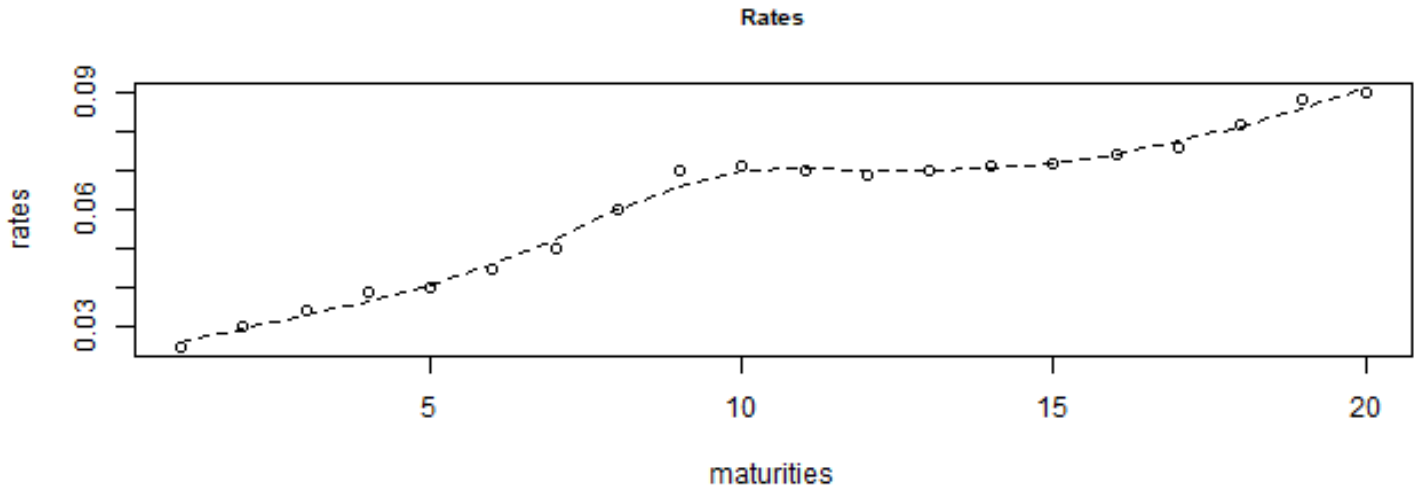



```
# Specify two polynomials to be used for fitting purposes.
poly_5 <- function(x, a) {
  f <- a[1] + a[2] * x + a[3] * x^2 +
    a[4] * x^3 + a[5] * x^4 +
    a[6] * x^5
  return(f)
}

poly_3 <- function(x, offset, intercept, b) {
  f <- intercept + b[1] * (x - offset) +
    b[2] * (x - offset)^2 +
    b[3] * (x - offset)^3
}

obj_3_5 <- function(x, y, offset, p) {

  # All points are at infinity initially
  fit <- rep(Inf, length(x))
  ind_5 <- x <= offset
  ind_3 <- x > offset

  fit[ind_5] <- poly_5(x[ind_5], p[1:6])
  fit[ind_3] <- poly_3(x[ind_3], offset,
    poly_5(offset, p[1:6]), p[7:9])

  error <- (y - fit) ^ 2
  return(sum(error))
}
```

```
# Fit the parameters.  Assume 0 for all initial values
offset <- 9
out_3_5 <- optim(obj_3_5, par = rep(0, 9),
  x = maturities, y = rates, offset = offset)

plot(maturities, rates, xlab = "years",
  main = "Yields",
  cex.main = 0.8,
  cex.lab = 0.8,
  cex.axis = 0.8)
grid()
lines(poly_5(maturities[maturities <= offset],
  out_3_5$par[1:6]), lwd = 2)
lines(c(rep(NA, offset),
  poly_3(maturities[maturities > offset], offset,
  poly_5(offset, out_3_5$par[1:6]),
  out_3_5$par[7:9])), lwd = 2)
abline(v = offset)
```




```
# Fit loess to the data
obj <- loess(rates ~ maturities, span = 0.5)

# Plot the data and the fit
plot(maturities, rates, main = "Rates", cex.main = 0.8)
lines(predict(obj), lty = 2)
```




```r
# Drawdown function
compute_drawdown <- function(x, returns_default = TRUE,
  geometric = TRUE) {
  # x = Vector of raw pnl or returns
  # If returns_default = FALSE, the geometric
  # argument is ignored and the pnl is used.
  # Output = the maximum drawdown
  if(returns_default) {
    # Cumulative return calculation
    if(geometric) {
      cumulative_return <- cumprod(1 + x)
    } else {
      cumulative_return <- 1 + cumsum(x)
    }
    max_cumulative_return <- cummax(c(1, cumulative_return))[-1]
    drawdown <- -(cumulative_return / max_cumulative_return - 1)
  } else {
    # PnL vector is used
    cumulative_pnl <- c(0, cumsum(x))
    drawdown <- cummax(cumulative_pnl) - cumulative_pnl
    drawdown <- drawdown[-1]
  }
  # Drawdown vector for either pnl or returns
  return(drawdown)
}

obj_max_drawdown <- function(w, r_matrix, small_weight) {
  # w is the weight of every stock
  # r_matrix is the returns matrix of all stocks
```

```
  # Portfolio return
  portfolio_return <- r_matrix %*% w

  # Max drawdown
  drawdown_penalty <- max(compute_drawdown(portfolio_return))

  # Create penalty component for sum of weights
  weight_penalty <- 100 * (1 - sum(w)) ^ 2

  # Create a penalty component for negative weights
  negative_penalty <- -sum(w[w < 0])

  # Create penalty component for small weights
  small_weight_penalty <- 100 * sum(w[w < small_weight])

  # Objective function to minimize
  obj <- drawdown_penalty + weight_penalty +
    negative_penalty + small_weight_penalty
  return(obj)
}

# Calculate a returns matrix for multiple stocks
symbol_names <- c("AXP", "BA", "CAT", "CVX",
  "DD", "DIS", "GE", "HD", "IBM",
  "INTC", "KO", "MMM", "MRK",
  "PG", "T", "UTX", "VZ")

getSymbols(symbol_names, from = "2010-12-31", to = "2014-12-31")
```

```
'getSymbols' currently uses auto.assign=TRUE by default, but will
use auto.assign=FALSE in 0.5-0. You will still be able to use
'loadSymbols' to automatically load data. getOption("getSymbols.env")
and getOption("getSymbols.auto.assign") will still be checked for
alternate defaults.

This message is shown once per session and may be disabled by setting
options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
```

```
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
pausing 1 second between requests for more than 5 symbols
```

```
 [1] "AXP"  "BA"   "CAT"  "CVX"  "DD"   "DIS"  "GE"   "HD"   "IBM"  "INTC"
[11] "KO"   "MMM"  "MRK"  "PG"   "T"    "UTX"  "VZ"
```

```
# Load these prices into memory
price_matrix <- NULL
for(name in symbol_names) {
  # Extract the adjusted close price vector
  price_matrix <- cbind(price_matrix, get(name)[, 6])
}

colnames(price_matrix) <- symbol_names

# Compute returns
returns_matrix <- apply(price_matrix, 2, function(x) diff(log(x)))

# Specify a small weight below which the allocation should be 0%
small_weight_value <- 0.02

# Specify lower and upper bounds for the weights
lower <- rep(0, ncol(returns_matrix))
upper <- rep(1, ncol(returns_matrix))

optim_result <- DEoptim(obj_max_drawdown, lower, upper,
  control = list(NP = 400, itermax = 300, F = 0.25, CR = 0.75),
  returns_matrix, small_weight_value)
```

```
Iteration: 1 bestvalit: 1007.555978 bestmemit:    0.049364    0.072632    0.331117    0.017377
Iteration: 2 bestvalit: 1007.555978 bestmemit:    0.049364    0.072632    0.331117    0.017377
Iteration: 3 bestvalit: 977.235934 bestmemit:    0.049364    0.072632    0.331117    0.017377
Iteration: 4 bestvalit: 977.235934 bestmemit:    0.049364    0.072632    0.331117    0.017377
Iteration: 5 bestvalit: 977.235934 bestmemit:    0.049364    0.072632    0.331117    0.017377
Iteration: 6 bestvalit: 967.236083 bestmemit:    0.049364    0.072632    0.331117    0.017377
Iteration: 7 bestvalit: 920.540270 bestmemit:    0.049364    0.072632    0.331117    0.017377
Iteration: 8 bestvalit: 920.540270 bestmemit:    0.049364    0.072632    0.331117    0.017377
Iteration: 9 bestvalit: 914.160062 bestmemit:    0.049364    0.072632    0.331117    0.017377
Iteration: 10 bestvalit: 865.127033 bestmemit:    0.192919    0.350608    0.110167    0.253413
Iteration: 11 bestvalit: 846.878038 bestmemit:    0.192919    0.350608    0.110167    0.253413
Iteration: 12 bestvalit: 749.461572 bestmemit:    0.192919    0.350608    0.110167    0.253413
```

```
Iteration: 13 bestvalit: 705.115055 bestmemit:    0.192919    0.350608    0.110167    0.253413
Iteration: 14 bestvalit: 705.115055 bestmemit:    0.192919    0.350608    0.110167    0.253413
Iteration: 15 bestvalit: 705.115055 bestmemit:    0.192919    0.350608    0.110167    0.253413
Iteration: 16 bestvalit: 705.115055 bestmemit:    0.192919    0.350608    0.110167    0.253413
Iteration: 17 bestvalit: 705.115055 bestmemit:    0.192919    0.350608    0.110167    0.253413
Iteration: 18 bestvalit: 650.526641 bestmemit:    0.192919    0.376964    0.062290    0.228816
Iteration: 19 bestvalit: 650.526641 bestmemit:    0.192919    0.376964    0.062290    0.228816
Iteration: 20 bestvalit: 629.878602 bestmemit:    0.247533    0.262076    0.056284    0.118840
Iteration: 21 bestvalit: 621.495957 bestmemit:    0.283592    0.134343    0.071410    0.045854
Iteration: 22 bestvalit: 578.101421 bestmemit:    0.345019    0.380309    0.159426    0.099805
Iteration: 23 bestvalit: 543.621953 bestmemit:    0.192919    0.376964    0.000738    0.228816
Iteration: 24 bestvalit: 520.473477 bestmemit:    0.192919    0.270117    0.057458    0.228816
Iteration: 25 bestvalit: 514.048004 bestmemit:    0.345019    0.380309    0.159426    0.099805
Iteration: 26 bestvalit: 514.048004 bestmemit:    0.345019    0.380309    0.159426    0.099805
Iteration: 27 bestvalit: 514.048004 bestmemit:    0.345019    0.380309    0.159426    0.099805
Iteration: 28 bestvalit: 462.755776 bestmemit:    0.258154    0.237097    0.274182    0.353164
Iteration: 29 bestvalit: 445.636738 bestmemit:    0.244059    0.226129    0.015148    0.069122
Iteration: 30 bestvalit: 424.491459 bestmemit:    0.368875    0.271234    0.146875    0.162155
Iteration: 31 bestvalit: 385.787142 bestmemit:    0.258154    0.314877    0.246640    0.175446
Iteration: 32 bestvalit: 362.863866 bestmemit:    0.405948    0.238053    0.108594    0.280348
Iteration: 33 bestvalit: 345.441675 bestmemit:    0.125508    0.255596    0.067563    0.197917
Iteration: 34 bestvalit: 275.362501 bestmemit:    0.328747    0.183639    0.108594    0.280348
Iteration: 35 bestvalit: 275.362501 bestmemit:    0.328747    0.183639    0.108594    0.280348
Iteration: 36 bestvalit: 275.362501 bestmemit:    0.328747    0.183639    0.108594    0.280348
Iteration: 37 bestvalit: 274.955498 bestmemit:    0.328747    0.183639    0.108594    0.280348
Iteration: 38 bestvalit: 244.397948 bestmemit:    0.227003    0.091115    0.193014    0.204493
Iteration: 39 bestvalit: 244.397948 bestmemit:    0.227003    0.091115    0.193014    0.204493
Iteration: 40 bestvalit: 213.604990 bestmemit:    0.183308    0.032627    0.193014    0.204493
Iteration: 41 bestvalit: 202.724836 bestmemit:    0.183308    0.032627    0.193014    0.204493
Iteration: 42 bestvalit: 202.455485 bestmemit:    0.183308    0.032627    0.193014    0.204493
Iteration: 43 bestvalit: 174.251209 bestmemit:    0.086601    0.053898    0.177028    0.317305
Iteration: 44 bestvalit: 172.355236 bestmemit:    0.086601    0.053898    0.177028    0.317305
Iteration: 45 bestvalit: 149.816221 bestmemit:    0.086601    0.053898    0.177028    0.317305
Iteration: 46 bestvalit: 136.310761 bestmemit:    0.086601    0.053898    0.177028    0.317305
Iteration: 47 bestvalit: 136.310761 bestmemit:    0.086601    0.053898    0.177028    0.317305
Iteration: 48 bestvalit: 136.310761 bestmemit:    0.086601    0.053898    0.177028    0.317305
Iteration: 49 bestvalit: 136.310761 bestmemit:    0.086601    0.053898    0.177028    0.317305
Iteration: 50 bestvalit: 135.535671 bestmemit:    0.086601    0.053898    0.177028    0.317305
Iteration: 51 bestvalit: 122.201291 bestmemit:    0.047002    0.053898    0.177028    0.317305
Iteration: 52 bestvalit: 105.110603 bestmemit:    0.035052    0.101880    0.126616    0.239715
Iteration: 53 bestvalit: 99.097719 bestmemit:    0.009671    0.112835    0.149656    0.261549
Iteration: 54 bestvalit: 99.097719 bestmemit:    0.009671    0.112835    0.149656    0.261549
Iteration: 55 bestvalit: 94.494036 bestmemit:    0.009671    0.089295    0.149656    0.261549
Iteration: 56 bestvalit: 94.494036 bestmemit:    0.009671    0.089295    0.149656    0.261549
Iteration: 57 bestvalit: 89.970486 bestmemit:    0.009671    0.089295    0.149656    0.261549
```

```
Iteration: 58 bestvalit: 89.970486 bestmemit:    0.009671    0.089295    0.149656    0.261549
Iteration: 59 bestvalit: 76.357258 bestmemit:    0.034723    0.116886    0.174321    0.271535
Iteration: 60 bestvalit: 75.932985 bestmemit:    0.058253    0.088467    0.154155    0.230291
Iteration: 61 bestvalit: 74.822250 bestmemit:    0.058253    0.088467    0.154155    0.230291
Iteration: 62 bestvalit: 68.874269 bestmemit:    0.034723    0.116886    0.174321    0.271535
Iteration: 63 bestvalit: 66.069185 bestmemit:    0.058253    0.088467    0.154155    0.230291
Iteration: 64 bestvalit: 60.317773 bestmemit:    0.179219    0.092570    0.103734    0.192268
Iteration: 65 bestvalit: 55.211888 bestmemit:    0.179219    0.092570    0.103734    0.192268
Iteration: 66 bestvalit: 54.176235 bestmemit:    0.061634    0.121516    0.143536    0.190512
Iteration: 67 bestvalit: 50.626440 bestmemit:    0.061634    0.121516    0.143536    0.190512
Iteration: 68 bestvalit: 39.666711 bestmemit:    0.061634    0.121516    0.143536    0.178047
Iteration: 69 bestvalit: 37.813224 bestmemit:    0.082659    0.242125    0.046473    0.008860
Iteration: 70 bestvalit: 31.809850 bestmemit:    0.145411    0.091830    0.111033    0.137647
Iteration: 71 bestvalit: 29.642137 bestmemit:    0.145411    0.091830    0.111033    0.137647
Iteration: 72 bestvalit: 29.642137 bestmemit:    0.145411    0.091830    0.111033    0.137647
Iteration: 73 bestvalit: 24.937960 bestmemit:    0.145411    0.091830    0.111033    0.137647
Iteration: 74 bestvalit: 24.641393 bestmemit:    0.124495    0.074443    0.109657    0.128895
Iteration: 75 bestvalit: 24.641393 bestmemit:    0.124495    0.074443    0.109657    0.128895
Iteration: 76 bestvalit: 21.352671 bestmemit:    0.124495    0.074443    0.123563    0.123897
Iteration: 77 bestvalit: 21.352671 bestmemit:    0.124495    0.074443    0.123563    0.123897
Iteration: 78 bestvalit: 21.352671 bestmemit:    0.124495    0.074443    0.123563    0.123897
Iteration: 79 bestvalit: 21.352671 bestmemit:    0.124495    0.074443    0.123563    0.123897
Iteration: 80 bestvalit: 21.352671 bestmemit:    0.124495    0.074443    0.123563    0.123897
Iteration: 81 bestvalit: 18.896688 bestmemit:    0.082659    0.242125    0.046473    0.008860
Iteration: 82 bestvalit: 16.311111 bestmemit:    0.124495    0.074443    0.115184    0.098525
Iteration: 83 bestvalit: 16.311111 bestmemit:    0.124495    0.074443    0.115184    0.098525
Iteration: 84 bestvalit: 16.201430 bestmemit:    0.124495    0.074443    0.115184    0.098525
Iteration: 85 bestvalit: 14.662095 bestmemit:    0.090094    0.065424    0.096383    0.110919
Iteration: 86 bestvalit: 14.662095 bestmemit:    0.090094    0.065424    0.096383    0.110919
Iteration: 87 bestvalit: 10.322132 bestmemit:    0.073564    0.093780    0.113179    0.095526
Iteration: 88 bestvalit: 9.020106 bestmemit:    0.073564    0.093780    0.113179    0.095526
Iteration: 89 bestvalit: 8.258545 bestmemit:    0.073564    0.093780    0.113179    0.095526
Iteration: 90 bestvalit: 7.453538 bestmemit:    0.073564    0.089337    0.108825    0.077529
Iteration: 91 bestvalit: 7.353135 bestmemit:    0.069904    0.089337    0.108825    0.077529
Iteration: 92 bestvalit: 7.172975 bestmemit:    0.069904    0.089337    0.108825    0.077529
Iteration: 93 bestvalit: 7.172975 bestmemit:    0.069904    0.089337    0.108825    0.077529
Iteration: 94 bestvalit: 6.459164 bestmemit:    0.082342    0.109040    0.116559    0.143140
Iteration: 95 bestvalit: 5.273129 bestmemit:    0.082342    0.109040    0.116559    0.143140
Iteration: 96 bestvalit: 4.179123 bestmemit:    0.081202    0.122667    0.122781    0.135294
Iteration: 97 bestvalit: 4.179123 bestmemit:    0.081202    0.122667    0.122781    0.135294
Iteration: 98 bestvalit: 3.940177 bestmemit:    0.081202    0.122667    0.122781    0.135294
Iteration: 99 bestvalit: 2.407084 bestmemit:    0.100055    0.079066    0.100288    0.078997
Iteration: 100 bestvalit: 2.407084 bestmemit:    0.100055    0.079066    0.100288    0.078997
Iteration: 101 bestvalit: 2.024285 bestmemit:    0.100055    0.079066    0.100288    0.078997
Iteration: 102 bestvalit: 2.024285 bestmemit:    0.100055    0.079066    0.100288    0.078997
```

```
Iteration: 103 bestvalit: 1.449727 bestmemit:    0.100055    0.079066    0.100288    0.068639
Iteration: 104 bestvalit: 1.443780 bestmemit:    0.100055    0.079066    0.100288    0.068639
Iteration: 105 bestvalit: 1.384013 bestmemit:    0.076892    0.090643    0.097815    0.101456
Iteration: 106 bestvalit: 1.384013 bestmemit:    0.076892    0.090643    0.097815    0.101456
Iteration: 107 bestvalit: 1.134773 bestmemit:    0.076892    0.090643    0.097815    0.101456
Iteration: 108 bestvalit: 1.134773 bestmemit:    0.076892    0.090643    0.097815    0.101456
Iteration: 109 bestvalit: 1.104434 bestmemit:    0.076892    0.090643    0.097815    0.101946
Iteration: 110 bestvalit: 1.004727 bestmemit:    0.071789    0.071852    0.106540    0.087329
Iteration: 111 bestvalit: 0.850381 bestmemit:    0.085949    0.074523    0.112448    0.082881
Iteration: 112 bestvalit: 0.850381 bestmemit:    0.085949    0.074523    0.112448    0.082881
Iteration: 113 bestvalit: 0.731297 bestmemit:    0.068853    0.080248    0.094666    0.101946
Iteration: 114 bestvalit: 0.690656 bestmemit:    0.068853    0.080248    0.094666    0.101946
Iteration: 115 bestvalit: 0.399341 bestmemit:    0.074774    0.061281    0.093123    0.081290
Iteration: 116 bestvalit: 0.365192 bestmemit:    0.075111    0.062020    0.088048    0.073505
Iteration: 117 bestvalit: 0.365192 bestmemit:    0.075111    0.062020    0.088048    0.073505
Iteration: 118 bestvalit: 0.333407 bestmemit:    0.075111    0.062020    0.088048    0.073505
Iteration: 119 bestvalit: 0.264786 bestmemit:    0.066703    0.075399    0.083340    0.074334
Iteration: 120 bestvalit: 0.264786 bestmemit:    0.066703    0.075399    0.083340    0.074334
Iteration: 121 bestvalit: 0.226891 bestmemit:    0.066703    0.075399    0.083340    0.074334
Iteration: 122 bestvalit: 0.226891 bestmemit:    0.066703    0.075399    0.083340    0.074334
Iteration: 123 bestvalit: 0.226891 bestmemit:    0.066703    0.075399    0.083340    0.074334
Iteration: 124 bestvalit: 0.219955 bestmemit:    0.081217    0.058277    0.098744    0.072789
Iteration: 125 bestvalit: 0.219955 bestmemit:    0.081217    0.058277    0.098744    0.072789
Iteration: 126 bestvalit: 0.219955 bestmemit:    0.081217    0.058277    0.098744    0.072789
Iteration: 127 bestvalit: 0.213170 bestmemit:    0.081217    0.058277    0.098744    0.072789
Iteration: 128 bestvalit: 0.212147 bestmemit:    0.081217    0.052361    0.098744    0.072789
Iteration: 129 bestvalit: 0.212147 bestmemit:    0.081217    0.052361    0.098744    0.072789
Iteration: 130 bestvalit: 0.212147 bestmemit:    0.081217    0.052361    0.098744    0.072789
Iteration: 131 bestvalit: 0.211633 bestmemit:    0.081217    0.052361    0.098744    0.072789
Iteration: 132 bestvalit: 0.211447 bestmemit:    0.081217    0.052361    0.098744    0.072789
Iteration: 133 bestvalit: 0.211447 bestmemit:    0.081217    0.052361    0.098744    0.072789
Iteration: 134 bestvalit: 0.211447 bestmemit:    0.081217    0.052361    0.098744    0.072789
Iteration: 135 bestvalit: 0.211447 bestmemit:    0.081217    0.052361    0.098744    0.072789
Iteration: 136 bestvalit: 0.211447 bestmemit:    0.081217    0.052361    0.098744    0.072789
Iteration: 137 bestvalit: 0.211108 bestmemit:    0.071507    0.052510    0.097875    0.077082
Iteration: 138 bestvalit: 0.209455 bestmemit:    0.082982    0.042813    0.096147    0.079438
Iteration: 139 bestvalit: 0.208581 bestmemit:    0.073851    0.052530    0.093913    0.076342
Iteration: 140 bestvalit: 0.208531 bestmemit:    0.073851    0.052370    0.093913    0.076342
Iteration: 141 bestvalit: 0.208531 bestmemit:    0.073851    0.052370    0.093913    0.076342
Iteration: 142 bestvalit: 0.208137 bestmemit:    0.082360    0.051261    0.096600    0.072763
Iteration: 143 bestvalit: 0.207475 bestmemit:    0.082360    0.051261    0.096600    0.072763
Iteration: 144 bestvalit: 0.207356 bestmemit:    0.082360    0.051261    0.096600    0.072763
Iteration: 145 bestvalit: 0.205482 bestmemit:    0.080151    0.059574    0.091676    0.072060
Iteration: 146 bestvalit: 0.205482 bestmemit:    0.080151    0.059574    0.091676    0.072060
Iteration: 147 bestvalit: 0.205050 bestmemit:    0.077346    0.059479    0.091895    0.076504
```

```
Iteration: 148 bestvalit: 0.203717 bestmemit:    0.080151    0.059574    0.090625    0.074220
Iteration: 149 bestvalit: 0.203717 bestmemit:    0.080151    0.059574    0.090625    0.074220
Iteration: 150 bestvalit: 0.203139 bestmemit:    0.075629    0.058976    0.089977    0.069138
Iteration: 151 bestvalit: 0.202374 bestmemit:    0.075604    0.057366    0.089105    0.073385
Iteration: 152 bestvalit: 0.202374 bestmemit:    0.075604    0.057366    0.089105    0.073385
Iteration: 153 bestvalit: 0.201429 bestmemit:    0.079216    0.051944    0.089236    0.075846
Iteration: 154 bestvalit: 0.201429 bestmemit:    0.079216    0.051944    0.089236    0.075846
Iteration: 155 bestvalit: 0.200226 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 156 bestvalit: 0.200226 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 157 bestvalit: 0.200226 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 158 bestvalit: 0.200226 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 159 bestvalit: 0.200185 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 160 bestvalit: 0.199569 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 161 bestvalit: 0.199569 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 162 bestvalit: 0.199389 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 163 bestvalit: 0.199358 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 164 bestvalit: 0.199358 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 165 bestvalit: 0.199358 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 166 bestvalit: 0.199332 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 167 bestvalit: 0.199332 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 168 bestvalit: 0.199332 bestmemit:    0.077103    0.050728    0.091793    0.078587
Iteration: 169 bestvalit: 0.198308 bestmemit:    0.078580    0.053412    0.090098    0.079460
Iteration: 170 bestvalit: 0.198308 bestmemit:    0.078580    0.053412    0.090098    0.079460
Iteration: 171 bestvalit: 0.197858 bestmemit:    0.078068    0.053412    0.090098    0.079460
Iteration: 172 bestvalit: 0.197858 bestmemit:    0.078068    0.053412    0.090098    0.079460
Iteration: 173 bestvalit: 0.197834 bestmemit:    0.078678    0.051929    0.090579    0.081844
Iteration: 174 bestvalit: 0.197834 bestmemit:    0.078678    0.051929    0.090579    0.081844
Iteration: 175 bestvalit: 0.196713 bestmemit:    0.078678    0.051929    0.090579    0.081844
Iteration: 176 bestvalit: 0.196713 bestmemit:    0.078678    0.051929    0.090579    0.081844
Iteration: 177 bestvalit: 0.196713 bestmemit:    0.078678    0.051929    0.090579    0.081844
Iteration: 178 bestvalit: 0.196713 bestmemit:    0.078678    0.051929    0.090579    0.081844
Iteration: 179 bestvalit: 0.196713 bestmemit:    0.078678    0.051929    0.090579    0.081844
Iteration: 180 bestvalit: 0.196713 bestmemit:    0.078678    0.051929    0.090579    0.081844
Iteration: 181 bestvalit: 0.196713 bestmemit:    0.078678    0.051929    0.090579    0.081844
Iteration: 182 bestvalit: 0.196161 bestmemit:    0.080318    0.055575    0.089687    0.078663
Iteration: 183 bestvalit: 0.196161 bestmemit:    0.080318    0.055575    0.089687    0.078663
Iteration: 184 bestvalit: 0.196161 bestmemit:    0.080318    0.055575    0.089687    0.078663
Iteration: 185 bestvalit: 0.195679 bestmemit:    0.081488    0.049854    0.087843    0.080520
Iteration: 186 bestvalit: 0.195331 bestmemit:    0.081488    0.049854    0.087843    0.080520
Iteration: 187 bestvalit: 0.195330 bestmemit:    0.081488    0.049854    0.087843    0.080520
Iteration: 188 bestvalit: 0.195330 bestmemit:    0.081488    0.049854    0.087843    0.080520
Iteration: 189 bestvalit: 0.195330 bestmemit:    0.081488    0.049854    0.087843    0.080520
Iteration: 190 bestvalit: 0.195141 bestmemit:    0.081042    0.050225    0.090846    0.081356
Iteration: 191 bestvalit: 0.195056 bestmemit:    0.080567    0.049854    0.087843    0.080520
Iteration: 192 bestvalit: 0.195056 bestmemit:    0.080567    0.049854    0.087843    0.080520
```

```
Iteration: 193 bestvalit: 0.195056 bestmemit:    0.080567    0.049854    0.087843    0.080520
Iteration: 194 bestvalit: 0.194891 bestmemit:    0.080567    0.049854    0.085867    0.081114
Iteration: 195 bestvalit: 0.194891 bestmemit:    0.080567    0.049854    0.085867    0.081114
Iteration: 196 bestvalit: 0.194891 bestmemit:    0.080567    0.049854    0.085867    0.081114
Iteration: 197 bestvalit: 0.194891 bestmemit:    0.080567    0.049854    0.085867    0.081114
Iteration: 198 bestvalit: 0.194891 bestmemit:    0.080567    0.049854    0.085867    0.081114
Iteration: 199 bestvalit: 0.194891 bestmemit:    0.080567    0.049854    0.085867    0.081114
Iteration: 200 bestvalit: 0.194891 bestmemit:    0.080567    0.049854    0.085867    0.081114
Iteration: 201 bestvalit: 0.194100 bestmemit:    0.079903    0.057727    0.086214    0.080404
Iteration: 202 bestvalit: 0.194100 bestmemit:    0.079903    0.057727    0.086214    0.080404
Iteration: 203 bestvalit: 0.194100 bestmemit:    0.079903    0.057727    0.086214    0.080404
Iteration: 204 bestvalit: 0.194100 bestmemit:    0.079903    0.057727    0.086214    0.080404
Iteration: 205 bestvalit: 0.194100 bestmemit:    0.079903    0.057727    0.086214    0.080404
Iteration: 206 bestvalit: 0.194100 bestmemit:    0.079903    0.057727    0.086214    0.080404
Iteration: 207 bestvalit: 0.194100 bestmemit:    0.079903    0.057727    0.086214    0.080404
Iteration: 208 bestvalit: 0.194100 bestmemit:    0.079903    0.057727    0.086214    0.080404
Iteration: 209 bestvalit: 0.194009 bestmemit:    0.079903    0.057727    0.086214    0.080404
Iteration: 210 bestvalit: 0.193792 bestmemit:    0.079819    0.057845    0.087073    0.079512
Iteration: 211 bestvalit: 0.192815 bestmemit:    0.073419    0.055464    0.089351    0.080706
Iteration: 212 bestvalit: 0.192815 bestmemit:    0.073419    0.055464    0.089351    0.080706
Iteration: 213 bestvalit: 0.192815 bestmemit:    0.073419    0.055464    0.089351    0.080706
Iteration: 214 bestvalit: 0.192815 bestmemit:    0.073419    0.055464    0.089351    0.080706
Iteration: 215 bestvalit: 0.192797 bestmemit:    0.073419    0.055464    0.089351    0.080706
Iteration: 216 bestvalit: 0.192797 bestmemit:    0.073419    0.055464    0.089351    0.080706
Iteration: 217 bestvalit: 0.192669 bestmemit:    0.073419    0.055464    0.089351    0.080706
Iteration: 218 bestvalit: 0.192025 bestmemit:    0.076158    0.048437    0.088312    0.073768
Iteration: 219 bestvalit: 0.192025 bestmemit:    0.076158    0.048437    0.088312    0.073768
Iteration: 220 bestvalit: 0.192025 bestmemit:    0.076158    0.048437    0.088312    0.073768
Iteration: 221 bestvalit: 0.192025 bestmemit:    0.076158    0.048437    0.088312    0.073768
Iteration: 222 bestvalit: 0.191851 bestmemit:    0.076158    0.047506    0.088156    0.074541
Iteration: 223 bestvalit: 0.191605 bestmemit:    0.076158    0.047506    0.087957    0.075149
Iteration: 224 bestvalit: 0.191551 bestmemit:    0.076158    0.047506    0.087957    0.075149
Iteration: 225 bestvalit: 0.191551 bestmemit:    0.076158    0.047506    0.087957    0.075149
Iteration: 226 bestvalit: 0.191551 bestmemit:    0.076158    0.047506    0.087957    0.075149
Iteration: 227 bestvalit: 0.191551 bestmemit:    0.076158    0.047506    0.087957    0.075149
Iteration: 228 bestvalit: 0.191551 bestmemit:    0.076158    0.047506    0.087957    0.075149
Iteration: 229 bestvalit: 0.191551 bestmemit:    0.076158    0.047506    0.087957    0.075149
Iteration: 230 bestvalit: 0.191551 bestmemit:    0.076158    0.047506    0.087957    0.075149
Iteration: 231 bestvalit: 0.191551 bestmemit:    0.076158    0.047506    0.087957    0.075149
Iteration: 232 bestvalit: 0.191455 bestmemit:    0.075459    0.044492    0.088193    0.076617
Iteration: 233 bestvalit: 0.190313 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 234 bestvalit: 0.190313 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 235 bestvalit: 0.190313 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 236 bestvalit: 0.190313 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 237 bestvalit: 0.190313 bestmemit:    0.073977    0.040140    0.088193    0.076405
```

```
Iteration: 238 bestvalit: 0.190313 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 239 bestvalit: 0.190209 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 240 bestvalit: 0.190209 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 241 bestvalit: 0.190209 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 242 bestvalit: 0.190209 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 243 bestvalit: 0.190209 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 244 bestvalit: 0.190209 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 245 bestvalit: 0.190208 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 246 bestvalit: 0.190208 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 247 bestvalit: 0.189971 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 248 bestvalit: 0.189971 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 249 bestvalit: 0.189971 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 250 bestvalit: 0.189925 bestmemit:    0.073977    0.040140    0.088193    0.076405
Iteration: 251 bestvalit: 0.189837 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 252 bestvalit: 0.189701 bestmemit:    0.077352    0.047823    0.089816    0.078870
Iteration: 253 bestvalit: 0.189547 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 254 bestvalit: 0.189547 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 255 bestvalit: 0.189547 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 256 bestvalit: 0.189470 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 257 bestvalit: 0.189470 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 258 bestvalit: 0.189470 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 259 bestvalit: 0.189470 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 260 bestvalit: 0.189470 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 261 bestvalit: 0.189470 bestmemit:    0.073128    0.040859    0.088140    0.076405
Iteration: 262 bestvalit: 0.189075 bestmemit:    0.072986    0.044398    0.087539    0.079250
Iteration: 263 bestvalit: 0.189075 bestmemit:    0.072986    0.044398    0.087539    0.079250
Iteration: 264 bestvalit: 0.189075 bestmemit:    0.072986    0.044398    0.087539    0.079250
Iteration: 265 bestvalit: 0.188832 bestmemit:    0.075417    0.043638    0.087031    0.075720
Iteration: 266 bestvalit: 0.188832 bestmemit:    0.075417    0.043638    0.087031    0.075720
Iteration: 267 bestvalit: 0.188765 bestmemit:    0.075417    0.043638    0.087031    0.075720
Iteration: 268 bestvalit: 0.188442 bestmemit:    0.076098    0.044344    0.084216    0.079900
Iteration: 269 bestvalit: 0.188442 bestmemit:    0.076098    0.044344    0.084216    0.079900
Iteration: 270 bestvalit: 0.188442 bestmemit:    0.076098    0.044344    0.084216    0.079900
Iteration: 271 bestvalit: 0.188301 bestmemit:    0.076098    0.044344    0.084216    0.079900
Iteration: 272 bestvalit: 0.188301 bestmemit:    0.076098    0.044344    0.084216    0.079900
Iteration: 273 bestvalit: 0.187996 bestmemit:    0.074459    0.039840    0.088140    0.077321
Iteration: 274 bestvalit: 0.187910 bestmemit:    0.074459    0.039840    0.088140    0.077321
Iteration: 275 bestvalit: 0.187885 bestmemit:    0.074459    0.039840    0.088140    0.077321
Iteration: 276 bestvalit: 0.187885 bestmemit:    0.074459    0.039840    0.088140    0.077321
Iteration: 277 bestvalit: 0.187885 bestmemit:    0.074459    0.039840    0.088140    0.077321
Iteration: 278 bestvalit: 0.187870 bestmemit:    0.074459    0.039840    0.088140    0.077321
Iteration: 279 bestvalit: 0.187870 bestmemit:    0.074459    0.039840    0.088140    0.077321
Iteration: 280 bestvalit: 0.187870 bestmemit:    0.074459    0.039840    0.088140    0.077321
Iteration: 281 bestvalit: 0.187870 bestmemit:    0.074459    0.039840    0.088140    0.077321
Iteration: 282 bestvalit: 0.187347 bestmemit:    0.073221    0.041160    0.086549    0.076631
```

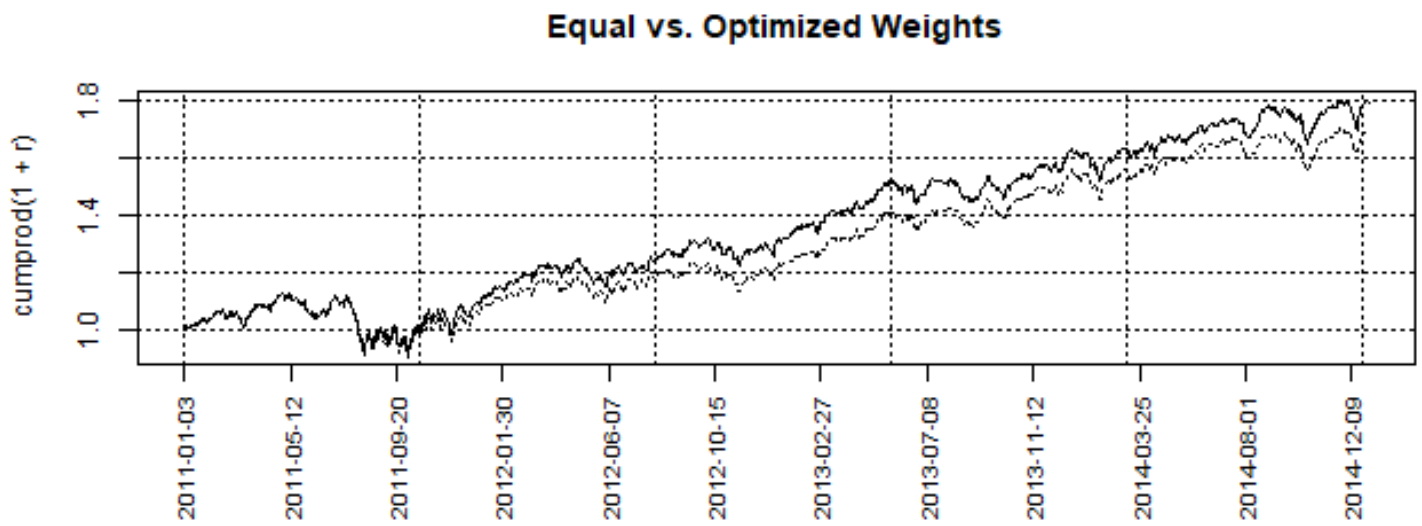
```
Iteration: 283 bestvalit: 0.187347 bestmemit:    0.073221    0.041160    0.086549    0.076631
Iteration: 284 bestvalit: 0.187310 bestmemit:    0.073221    0.041160    0.086549    0.076631
Iteration: 285 bestvalit: 0.187238 bestmemit:    0.076065    0.041688    0.086512    0.078046
Iteration: 286 bestvalit: 0.187205 bestmemit:    0.073221    0.041160    0.086549    0.076631
Iteration: 287 bestvalit: 0.187181 bestmemit:    0.073221    0.041160    0.086549    0.076631
Iteration: 288 bestvalit: 0.187181 bestmemit:    0.073221    0.041160    0.086549    0.076631
Iteration: 289 bestvalit: 0.187181 bestmemit:    0.073221    0.041160    0.086549    0.076631
Iteration: 290 bestvalit: 0.187181 bestmemit:    0.073221    0.041160    0.086549    0.076631
Iteration: 291 bestvalit: 0.187181 bestmemit:    0.073221    0.041160    0.086549    0.076631
Iteration: 292 bestvalit: 0.186885 bestmemit:    0.073049    0.038779    0.086075    0.077693
Iteration: 293 bestvalit: 0.186806 bestmemit:    0.073049    0.038779    0.086075    0.077693
Iteration: 294 bestvalit: 0.186760 bestmemit:    0.076659    0.038176    0.084435    0.076555
Iteration: 295 bestvalit: 0.186760 bestmemit:    0.076659    0.038176    0.084435    0.076555
Iteration: 296 bestvalit: 0.186760 bestmemit:    0.076659    0.038176    0.084435    0.076555
Iteration: 297 bestvalit: 0.186760 bestmemit:    0.076659    0.038176    0.084435    0.076555
Iteration: 298 bestvalit: 0.186760 bestmemit:    0.076659    0.038176    0.084435    0.076555
Iteration: 299 bestvalit: 0.186636 bestmemit:    0.074332    0.037670    0.085463    0.079272
Iteration: 300 bestvalit: 0.186636 bestmemit:    0.074332    0.037670    0.085463    0.079272
```

```
weights <- optim_result$optim$bestmem

sum(weights)
```

```
[1] 0.9983104
```

```
## 0.9978

weights <- weights / sum(weights)

# Equally weighted portfolio
equal_weights <- rep(1 / 17, 17)
equal_portfolio <- returns_matrix %*% equal_weights
equal_portfolio_cumprod <- cumprod(1 + equal_portfolio)

# Optimal max drawdown portfolio
optimized_portfolio <- returns_matrix %*% weights
drawdown_portfolio_cumprod <- cumprod(1 + optimized_portfolio)

main_title <- "Equal vs. Optimized Weights"
plot(drawdown_portfolio_cumprod, type = 'l', xaxt = 'n',
  main = main_title, xlab = "", ylab = "cumprod(1 + r)")
lines(equal_portfolio_cumprod, lty = 3)
grid(col = 'black')

# Set x-axis labels
label_location <- seq(1, length(drawdown_portfolio_cumprod),
```

```
  by = 90)
labels <- rownames(returns_matrix)[label_location]
axis(side = 1, at = label_location, labels = labels,
  las = 2, cex.axis= 0.8)
```




```
# Equal weighted
max(compute_drawdown(equal_portfolio))
```

```
[1] 0.1972771
```

```
## [1] 0.597

# Optimized for the smallest max drawdown
max(compute_drawdown(optimized_portfolio))
```

```
[1] 0.1866453
```

```
## [1] 0.515
```