Understanding Portfolio Selection - Pfaff

Stefan Glogger

13 Juli 2017

Code beschleunigen:

- Zwischenschritte nur 1x berechnen, speichern, dann mit gespeicherterm rechnen (e.g. marginal contribution of risk)
- möglichst Variablen lokal weitergeben (f bekommt Gewichtungsvektor, default ist Gleichgewichtung)
- allgemein: libraries da einbinden, wo erstmals benötigt
- mit echten Zahlen rechnen (statt mit x% und dann mit *100 und /100 rumwerfen)

R Preparation

Install and load Packages

```
# install.packages("FRAPO","mco")
library(FRAPO)
```

```
## Loading required package: cccp
## Loading required package: Rglpk
## Loading required package: slam
## Using the GLPK callable library version 4.47
## Loading required package: timeSeries
## Loading required package: timeDate
## Financial Risk Modelling and Portfolio Optimisation with R (version 0.4-1)
```

```
library(mco)
```

Data Preparation

Loading of Data

MultiAsset contains Month-end price data of stock and bond indices and gold.

```
data(MultiAsset)
Prices <- timeSeries(MultiAsset, charvec = rownames(MultiAsset))
NAssets <- ncol(Prices)
R <- returns(Prices, method = "discrete", percentage = TRUE)
```

Defining parameters

```r
# Return
TargetRpa <- 6 ## targeted return of 6 % p.a.
TargetR <- 100 * ((1 + TargetRpa / 100)^(1 / 12) - 1) # % per month

# Volatility
TargetVol <- 4 ## % p.a.

# Further parameters
l <- rep(1, 3) ## goal weighting, alle gleichgewichtet (lambda)
WeightedSum <- FALSE
mu <- colMeans(R)
S <- cov(R)
```

Note: Used as global parameters lateron.

Defining functions

Objective Function f.

- Input: Weights
- Calculation:
  - y[1]: return: -1 * weight for return in objective function (with λ_1) * portfolio return (weighted with x) / Target Return
  - y[2]: volatility: weight for volatility in objective function (with λ_2) * yearly volatility of portfolio / Target Volatility
  - y[3]: marginal contribution to portfolio risk (in terms of portfolio standard deviation) -> variance (as it is squared), takes dispersion into account
- Output: Either the weighted sum directly or the three (weighted) components

Idea: Normalization of objective function components by dividing by target value

```r
f <- function(x){
  y <- numeric(3)
  y[1] <- -1.0 * l[1] * drop(crossprod(x, mu)) / TargetR
  y[2] <- l[2] * drop(sqrt(t(x) %*% S %*% x)) * sqrt(12) / TargetVol

  #die MRC-Funktion berechnet die Marginal contributions zum Portfolio Risk, x sind die weights
  ## vgl. "Portfolio Analysis"
  y[3] <- l[3] * sum((mrc(x, S) / 100)^2)
   if(WeightedSum){
    return(sum(y))
     } else {
     return(y)
      }
  }
```

Restriction function g Input: Weights Idea: Weights should sum up to 1

```r
g <- function(x){
  c(1.01 - sum(x), sum(x) - 0.99)
}
```

Calculation

Optimization by nsga2

```
?nsga2
```

```
ans <- nsga2(f, NAssets, 3, lower.bounds = rep(0, NAssets), upper.bounds = rep(1, NAssets), constraints
```

so constraints are evaluated as ≥ 0.

Preparation of Results

? Das ist dann so nicht sinnvoll: wenn nsga2 nur pareto effiziente ausgibt, muss ich nicht nur die pareto-optimalen zurückgeben (bzw. die Funktion nsga2 müsste gar nicht die Information pareto.optimal zurückgeben, da ja eh TRUE)

```
mco <- data.frame(ans$value[ans$pareto.optimal, ])

mco[, 1] <- ((1 + (-1.0 * mco[, 1] * TargetR) / 100)^12- 1.0) * 100
colnames(mco) <- c("Return", "Risk", "Diversification")
```

1. Spalte von mco enthält Funktionswert von f_1(x), also normierten Return (monatlich). jetzt multiplizieren mit Normierungsfaktor -> monatlicher Return, dann hochskalieren auf Jahr
2. Spalte müsste auch noch entnormiert werden (*TargetVol)
3. Spalte unsicher, ob monatliches Ergebnis rauskommt

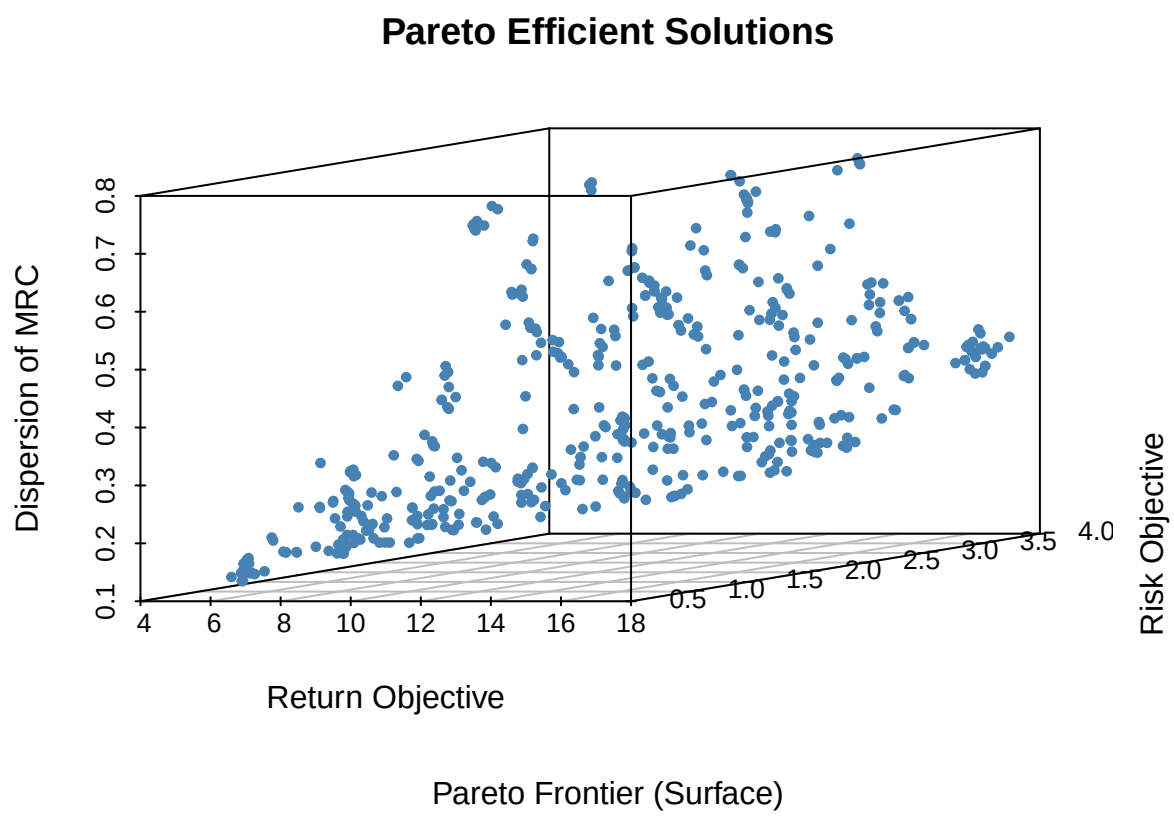
Visualization of Results

Scatterplot

Needed Packages

```
# install.packages("scatterplot3d")
library(scatterplot3d)
```

```
scatterplot3d(mco,
              main = "Pareto Efficient Solutions",
              sub = "Pareto Frontier (Surface)",
              xlab = "Return Objective",
              ylab = "Risk Objective",
              zlab = "Dispersion of MRC",
              angle = 15,
              highlight.3d = FALSE,
              box = TRUE,
              color = "steelblue",
              pch = 19, type = "p",
              cex.symbols = 0.6)
```



Plot 2

```
# install.packages("akima") # Interpolation of Irregularly and Regularly Spaced Data
# install.packages("fields") # Tools for Spatial Data
library(akima)
library(fields)
```

```
## Loading required package: spam
## Loading required package: dotCall64
## Loading required package: grid
## Spam version 2.1-1 (2017-07-02) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
##
## Attaching package: 'spam'
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
## Loading required package: maps
```

```r
# install.packages("fPortfolio")
### install.packages("fAssets")
# install.packages("Rdonlp2", repos="http://R-Forge.R-project.org")
# install.packages("PerformanceAnalytics")
# install.packages("ggtern")
library(fPortfolio) ### needs package fAssets, ecodist, Rsymphony
```

```
## Loading required package: fBasics
##
## Rmetrics Package fBasics
## Analysing Markets and calculating Basic Statistics
## Copyright (C) 2005-2014 Rmetrics Association Zurich
## Educational Software for Financial Engineering and Computational Science
## Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY.
## https://www.rmetrics.org --- Mail to: info@rmetrics.org
## Loading required package: fAssets
##
## Rmetrics Package fAssets
## Analysing and Modeling Financial Assets
## Copyright (C) 2005-2014 Rmetrics Association Zurich
## Educational Software for Financial Engineering and Computational Science
## Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY.
## https://www.rmetrics.org --- Mail to: info@rmetrics.org
##
## Rmetrics Package fPortfolio
## Portfolio Optimization
## Copyright (C) 2005-2014 Rmetrics Association Zurich
## Educational Software for Financial Engineering and Computational Science
## Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY.
## https://www.rmetrics.org --- Mail to: info@rmetrics.org
```

```r
library(Rdonlp2)
```

```
##
## Attaching package: 'Rdonlp2'
## The following object is masked from 'package:fPortfolio':
##
##     donlp2NLP
```

```r
library(ggtern)
```

```
## Loading required package: ggplot2
```

```
## --
## Consider donating at: http://ggtern.com
## Even small amounts (say $10-50) are very much appreciated!
## Remember to cite, run citation(package = 'ggtern') for further info.
## --

##
## Attaching package: 'ggtern'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, aes, annotate, calc_element, ggplot, ggplot_build,
##     ggplot_gtable, ggplotGrob, ggsave, layer_data, theme,
##     theme_bw, theme_classic, theme_dark, theme_gray, theme_light,
##     theme_linedraw, theme_minimal, theme_void
```

Further calculation

interpolate (to draw smoothly)

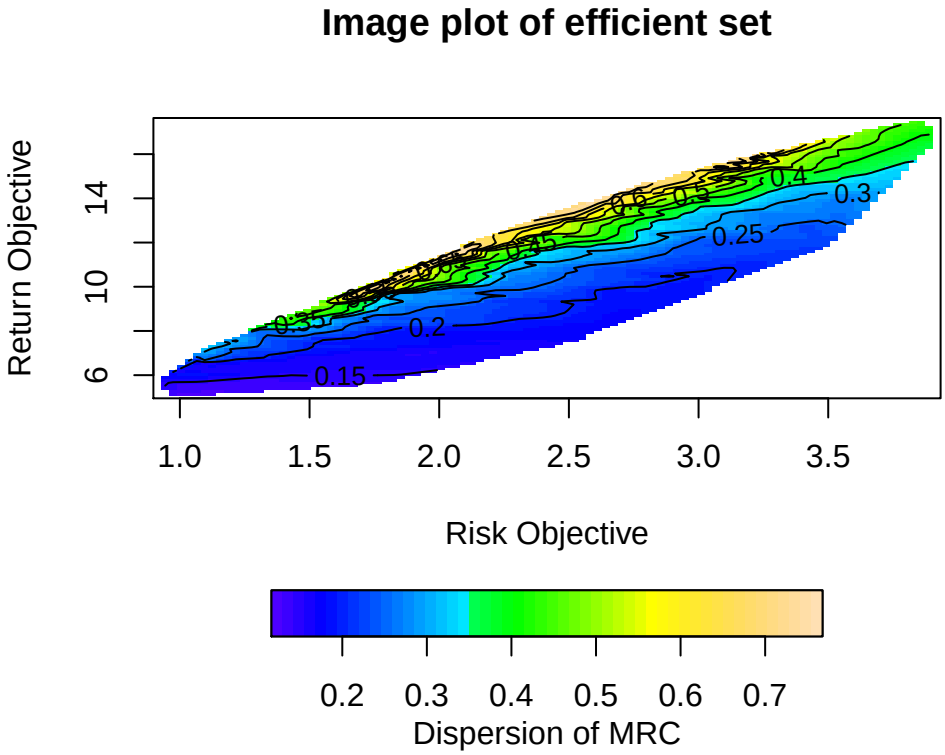
```
s <- interp(mco[, 2], mco[, 1], mco[, 3],xo = seq(min(mco[, 2]), max(mco[, 2]), length = 100),
            yo = seq(min(mco[, 1]), max(mco[, 1]), length = 100),
            duplicate = "mean"
            )
```

Plot Image

```
par(mar = c(5, 6, 5, 6))

image.plot(s, nlevel = 50,
           main = "Image plot of efficient set",
           legend.lab = "Dispersion of MRC",
           xlab = "Risk Objective",
           ylab = "Return Objective",
           legend.mar = 4,
           horizontal = TRUE,
           legend.shrink = 0.7,
           col = topo.colors(50))

contour(s, add = TRUE, nlevels = 20, labcex = 0.8)
```



verstehe noch nicht genau, was ich hier sehe. Vermute: 3 Informationen 2-dimensional dargestellt. Farbe als dritte Information. Sehe, welche Ausprägungen efficient portfolios haben

Plot 3

Further calculations

Erstelle Gewichtung von Portfolio. mögliche Aufteilung von 1 auf drei verschiedene Kriterien, wobei jedem Kriterium mindestens Gewicht 0.05 gegeben wird. -> Gewichtung für die Zielkriterien wird vorgegeben (f_1, f_2, f_3) gespeichert in wobj

Programmierstil: letztes Kriterium kann (hier) schon mal Gewicht von 0 bekommen

```
grid <- expand.grid(x = seq(0.05, 0.95, by = 0.05),
                    y = seq(0.05, 0.95, by = 0.05))
grid <- grid[which(rowSums(grid) <= 1.0), ]
wobj <- as.matrix(cbind(grid, 1 - rowSums(grid)),
                  nrow = nrow(grid), ncol = 3)
#Wobj ist eine Art Gitterverfahren und W ist die Lösung des Optimierers, wie viel jedes Asset zum Ziel
```

W hat in Zeilen alle möglichen Gewichtungen der drei Zielfunktionen und in Spalten die verschiedenen verwendeten Assets (wie viel dem Asset in dem optimalen Portfolio zugeteilt wird)

verwende donlp2NLP, um optimale Portfoliogewichtung unter gegebener Gewichtung der drei Zielkriterien zu finden * start: gleichverteilte Gewichte * objective: die Funktion f mit ihren drei Zielkriterien, ACHTUNG: WeightedSum ist auf TRUE gesetzt, damit schaut f nach einem Vektor l, der die Gewichtung der drei Zielfunktionen enthält, diese wird in jedem Durchlauf der for-Schleife angepasst (miserabler Programmierstil)

* par.lower: jeder Parameter (Gewicht für Assets) ist mindestens 0 -> kein short-sell erlaubt * ineqA, ineqA.lower, ineqA.upper: die Gewichte der Assets sollen zu 1 aufsummieren

:D In package fPortfolio wird function donlp2NLP erneut definiert -> masked die Version vom Originalpackage -> muss Zielfunktion mit "objective =" übergeben, statt mit "fun =" bzw. wenn Code am Stück durchläuft, dann doch mit "fun = f"

```
W <- matrix(NA, nrow = nrow(wobj), ncol = NAssets)
WeightedSum <- TRUE
IneqA <- matrix(1, nrow = 1, ncol = NAssets)
ew <- rep(1 / NAssets, NAssets) # starte mit gleichverteilter Gewichtung
library(fPortfolio) ## for donlp2NLP ### sollte schon eingebunden sein
for(i in 1:nrow(wobj)){
  l <- c(wobj[i, ])
  W[i, ] <- donlp2NLP(start = ew, fun = f,
                      par.lower = rep(0, NAssets),
                      ineqA = IneqA, ineqA.lower = 1.0,
                      ineqA.upper = 1.0)$solution
}
```

calculate expected shortfall for the different portfolios (found above and stored in W) anscheinend spezielles Format benötigt, dass ES() rechnen kann

Programmierstil: Funktion vorher definieren, dann apply mit der Funktion

```
#ES ist aus PerformanceAnalytics
library(PerformanceAnalytics)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following object is masked from 'package:timeSeries':
##
##     time<-
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```
Es95Mod <- apply(W, 1, function(x){
  r <- timeSeries(R %*% x / 100, time(R))
  -100 * ES(r)
})
```

Plot

Daten abspeichern (Gewichtung der Zielkriterien zusammen mit expected shortfall von gefundenem, optimalem Portfolio)

```
terndat <- data.frame(cbind(wobj, Es95Mod))
colnames(terndat) <- c("x", "y", "z", "value")
```

Bereite ternary-Plot vor.

```
#Theme for ternary plot
terntheme <- function(){
  list(theme_rgbg(), theme(legend.position = c(0, 1),legend.justification = c(0, 1),
                           plot.margin=unit(c(0, 2,0, 2), "cm")))
}
```

Und jetzt der ternary plot

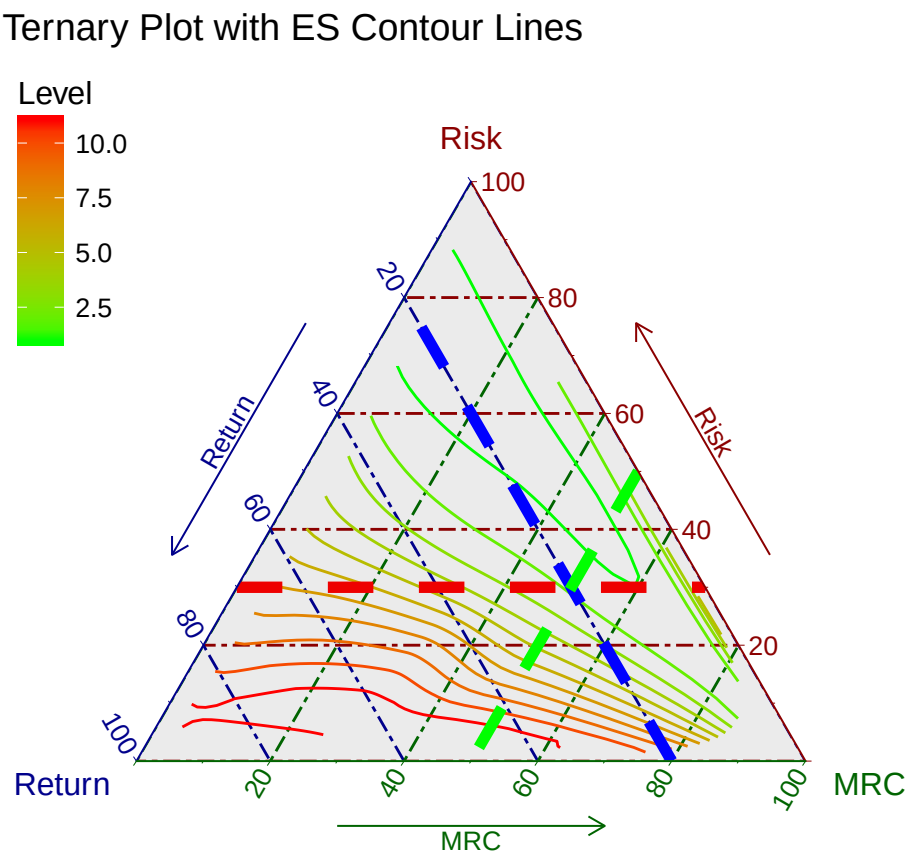
alte Version

```
ggtern(terndat, aes(x = x, y = y, z = z, value = value)) +
  geom_interpolate_tern(aes(value = value, color = ..level..), binwidth = 1.0) +
  terntheme() +
  theme_hidegrid_minor() +
  theme_showgrid_major() +
  Lline(0.2, color = "blue", linetype = 2) + ## x
  Tline(0.3, colour = "red2", linetype = 2) + ## y
  Rline(0.5, color = "brown", linetype = 2) + ## z
  scale_color_gradient(low = "green", high = "red") +
  labs(x = "Return", y = "Risk", z = "MRC",
         title = "Ternary Plot with ES Contour Lines",
         color = "Level")
```

meine Version

es kommen Warnmeldungen, dass eine Version veraltet ist, aber das liegt an ggtern

```
ggtern(terndat, aes(x, y, z, value)) +
  geom_interpolate_tern(aes(value = value, color = ..level..), binwidth = 1.0) +
  terntheme() +
  theme_hidegrid_minor() +
  theme_showgrid_major() +
  Lline(Lintercept =  0.2, colour = "blue", linetype = 2, lwd=2) + ## x
  Tline(Tintercept = 0.3, colour = "red2", linetype = 2, lwd=2) + ## y
  Rline(Rintercept = 0.5, color = "green", linetype = 2, lwd=2) + ## z
  scale_color_gradient(low = "green", high = "red") +
  labs(x = "Return", y = "Risk", z = "MRC",
         title = "Ternary Plot with ES Contour Lines",
         color = "Level")
```



keine Ahnung, warum er return auf 0.2 (blue), risk auf 0.3 (red2), MRC auf 0.5 setzt.

```
terndat[terndat$x==0.2 & terndat$y==0.3,]
```

```
##      x   y   z    value
## 99 0.2 0.3 0.5 1.945196
```

Der minimale value wird erreicht bei

```
terndat[which(terndat$value == min(terndat$value)),]
```

```
##       x   y z   value
## 253 0.3 0.7 0 1.35998
```

Backtest

Passe die Zeiten an ep enthält nur noch die hinteren zeiten (erste 59 werden rausgelöscht)

```
library(cccp) ## for ERC portolio
## backtest, extending window
ep <- time(R)[-c(1:59)]
bs <- length(ep)
sp <- rep(start(R), bs)
```

Initialisiere Objecte verschiedene Matrizen mit Zeilen (Anzahl Zeitpunkte), Spalten (verschiedene Assets)

```
## initialising object
Wmco <- matrix(NA, nrow = bs, ncol = NAssets)
Wmsr <- Wmdp <- Wgmv <- Werc <- Wmco
```

Goal weighting

```
l <- c(0.2, 0.1, 0.7) ## goal weighting
```

berechne die ganzen Daten zu jedem Zeitpunkt die optimalen Portfolios Wmco unser optimales Portfolio (mit den Gewichten wie oben) Werc riskp parity solution (long only portfolio with budget constraint) Wgmv global minimum variance portfolio (long only) Wmdp most diversified portfolio (long only) Wmsr tangencyPortfolio (portfolio with the highest return/risk ratio on the efficient frontier)

```
for(i in 1:bs){
    rdat <- window(R, start = sp[i], end = ep[i])
    mu <- colMeans(rdat)
    S <- cov(rdat)

    Wmco[i, ] <- donlp2NLP(start = ew, fun = f,
                           par.lower = rep(0, NAssets), ineqA = IneqA,
                           ineqA.lower = 1.0, ineqA.upper = 1.0)$solution
    # manchmal "fun= ", manchmal "objective=" needed

    ans <- tangencyPortfolio(rdat)
    Wmsr[i, ] <- getWeights(ans)

    ans <- PMD(rdat)
    Wmdp[i, ] <- FRAPO::Weights(ans) / 100

    ans <- PGMV(rdat)
    Wgmv[i, ] <- FRAPO::Weights(ans) / 100

    ans <- rp(ew, S, ew, optctrl = ctrl(trace = FALSE))
    Werc[i, ] <- c(getx(ans))
    }

W <- list("MCO" = Wmco, "MSR" = Wmsr, "MDP" = Wmdp,
    "GMV" = Wgmv, "ERC" = Werc)
```

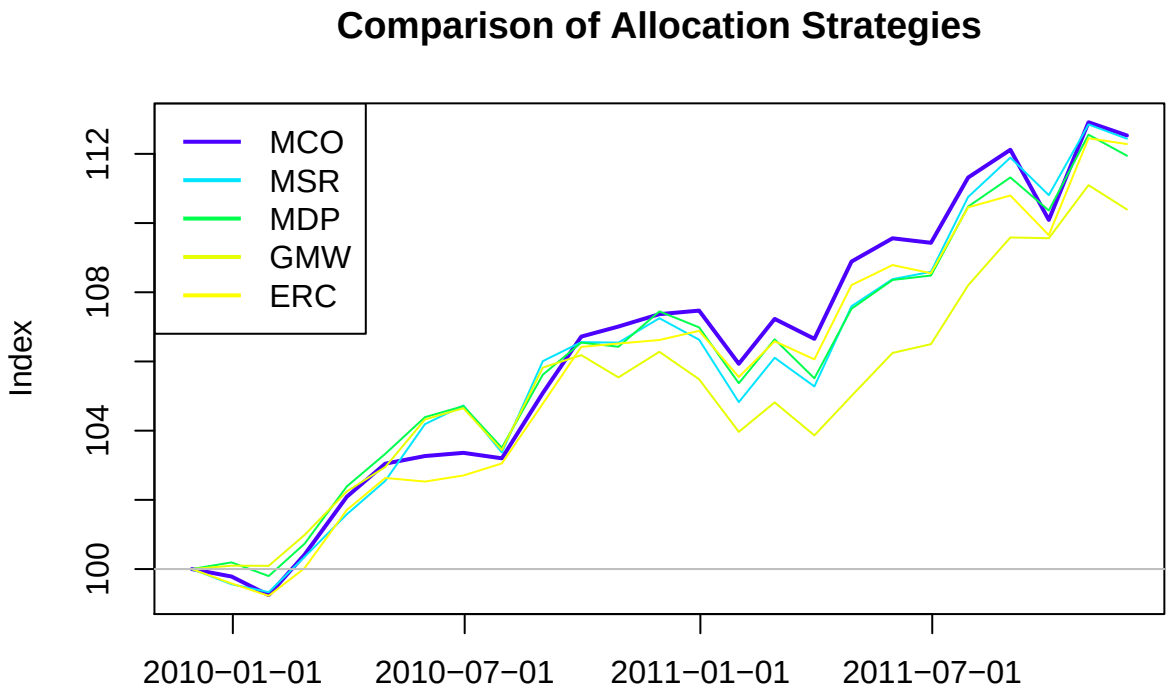
Nehme jedes Listenelement von W (jede Portfoliokategorie) dazu die timeSeries (je fixes Portfolio) wTSL1 ist immer 1 hinten dran berechne die Portfolioreturns zu jedem Zeitpunkt berechne wie sich Portfolio über Zeit entwickelt (cumprod = kumuliertes Produkt)

```
E <- lapply(W, function(x){
    wTs <- timeSeries(x, charvec = ep)
    wTsL1 <- lag(wTs, 1)
    RetFac <- 1 + rowSums(R[ep, ] * wTsL1) / 100.0
    RetFac[1] <- 100
    timeSeries(cumprod(RetFac), charvec = ep)
})
```

Plot

```
cols <- topo.colors(6)
plot(E[[1]], lwd = 2,
    ylab = "Index", xlab = "", col = cols[1],
    main = "Comparison of Allocation Strategies")
```

```
lines(E[[2]], col = cols[2])
lines(E[[3]], col = cols[3])
lines(E[[4]], col = cols[4])
lines(E[[5]], col = cols[5])
legend("topleft", legend = c("MCO", "MSR", "MDP", "GMW", "ERC"), col = cols, lty = 1, lwd = 2)
abline(h = 100, col = "gray")
```



da funktioniert unsere Strategie am besten (aber wie zu den Gewichten gekommen???)

RstratTs is same as Rstrat but other format (xts = extensible time-series object) Bench hat 0 überall

S1 nimmt dann die RstratTs und berechnet die annualisierten returns, annualized std.dev and annualized sharpe S2 ist Value at Risk (wird mit - multipliziert) ??? aber warum mit -100???

```
Rstrat <- matrix(unlist(lapply(E, Return.calculate)), ncol = 5)
RstratTs <- na.omit(xts(Rstrat, order.by = as.Date(ep)))
Bench <- xts(rep(0, nrow(RstratTs)), order.by = as.Date(ep)[-1])

S1 <- as.matrix(table.AnnualizedReturns(RstratTs, Rf = Bench, scale = 12))
S2 <- VaR(RstratTs)

ans <- rbind(S1, -100 * S2)
colnames(ans) <- c("MCO", "MSR", "MDP", "GMV", "ERC")
rownames(ans) <- c("Return (p.a.)", "StdDev. Risk (p.a.)", "Sharpe Ratio", "VaR (p.a.)")
round(ans, 3)
```

```
##                       MCO   MSR   MDP   GMV   ERC
## Return (p.a.)       0.061 0.060 0.058 0.051 0.060
```

```
## StdDev. Risk (p.a.) 0.038 0.039 0.037 0.034 0.034
## Sharpe Ratio        1.605 1.532 1.585 1.513 1.746
## VaR (p.a.)          1.301 1.380 1.308 1.211 1.017
```