

---

# R Tools for Portfolio Optimization

**Guy Yollin**

Quantitative Research Analyst  
Rotella Capital Management  
Bellevue, Washington

# Background

- Rotella Capital Management
  - Quantitative Research Analyst
    - Systematic CTA hedge fund trading 80+ global futures and foreign exchange markets
- Insightful Corporation
  - Director of Financial Engineering
    - Developers of S-PLUS<sup>®</sup>, S+FinMetrics<sup>®</sup>, and S+NuOPT<sup>®</sup>
- J.E. Moody, LLC
  - Financial Engineer
    - Futures Trading, Risk Management, Business Development
- OGI School of Engineering at Oregon Health & Science University
  - Adjunct Instructor
    - Statistical Computing & Financial Time Series Analysis
- Electro Scientific Industries, Inc
  - Director of Engineering, Vision Products Division
    - Machine Vision and Pattern Recognition
- **Started Using R in 1999**

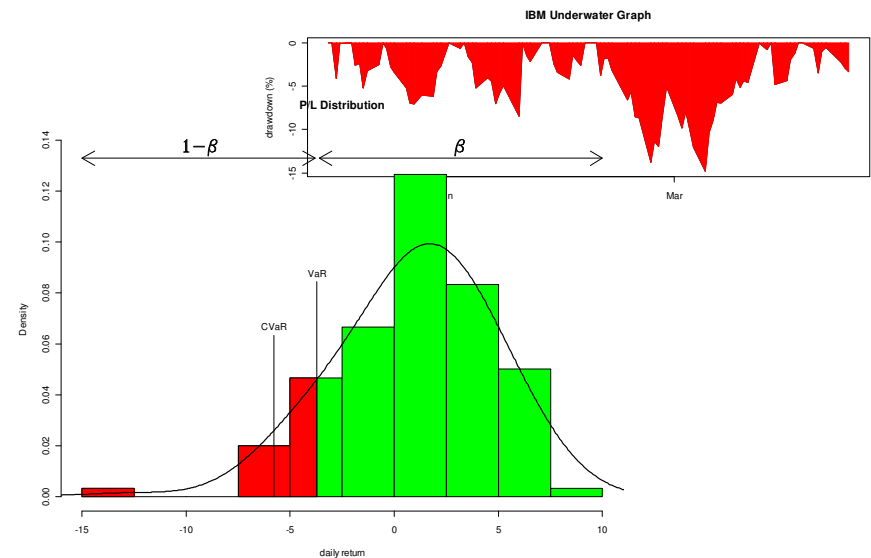
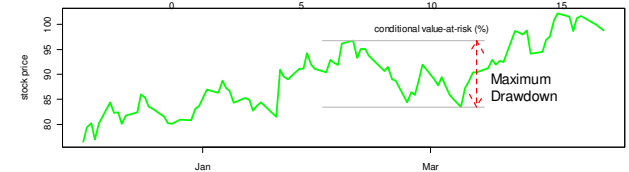
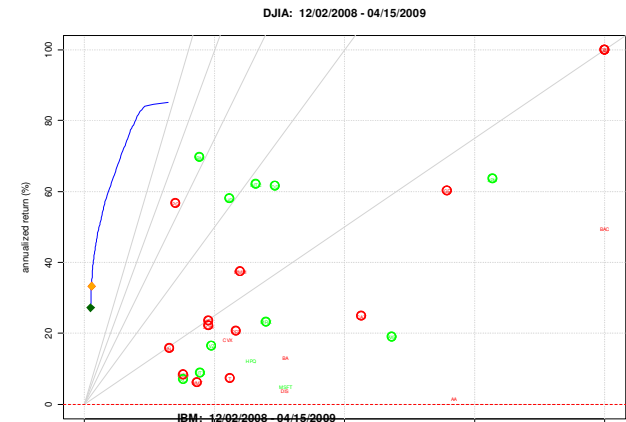
# Introduction

## R-SIG-FINANCE QUESTION:

Can I do < fill in the blank >  
portfolio optimization in R?

## ANSWER:

Yes! (98% confidence level)

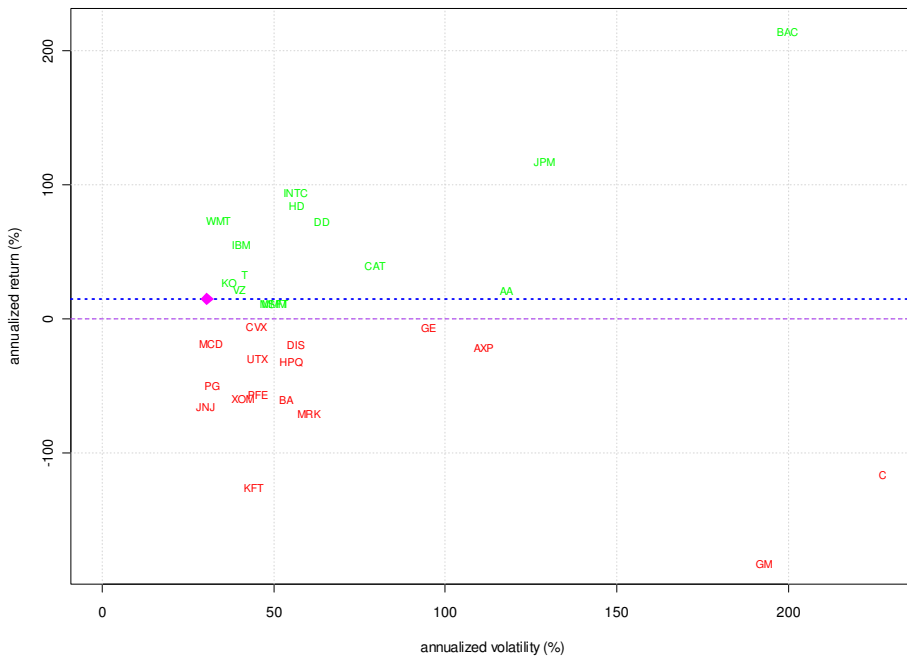


# Outline

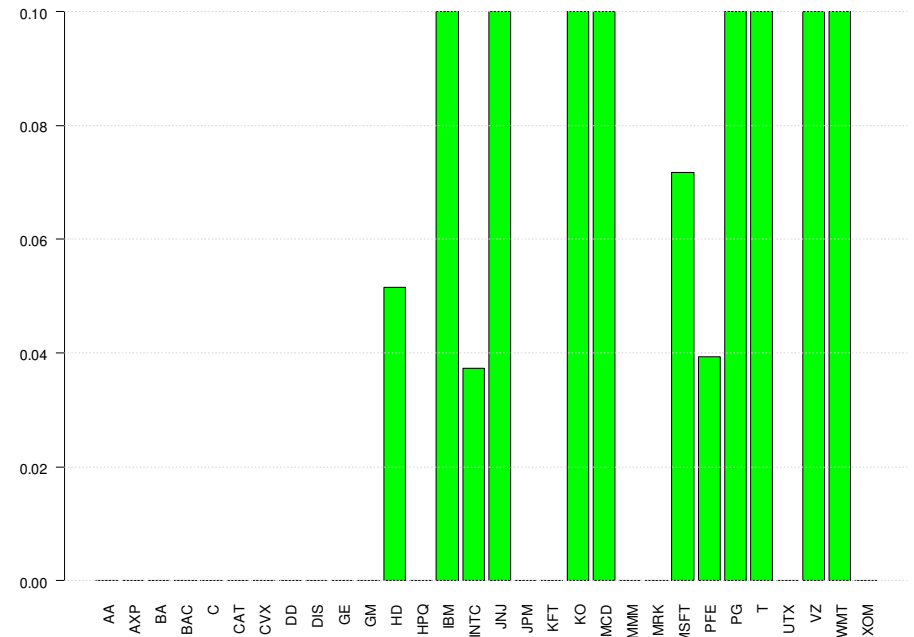
- Mean-Variance Portfolio Optimization
  - quadratic programming
    - tseries, quadprog
- Conditional Value-at-Risk Optimization
  - linear programming
    - Rglpk\_solve\_LP package
- General Nonlinear Optimization
  - Differential Evolution Algorithm
    - DEoptim package
      - Omega Optimization
      - Adding Constraints
      - Maximum Drawdown Optimization
      - R-Ratio Optimization
- Wrap-Up

# Efficient Portfolio Solution

DJIA Returns: 02/04/2009 - 04/03/2009



Portfolio Weights



# Mean-Variance Portfolio Optimization

- Function

- `portfolio.optim {tseries}`

- Description

- computer mean-variance efficient portfolio

- Usage

```
portfolio.optim(x, pm = mean(x), riskless = FALSE, shorts = FALSE,  
              rf = 0.0, reslow = NULL, reshigh = NULL, covmat = cov(x), ...)
```

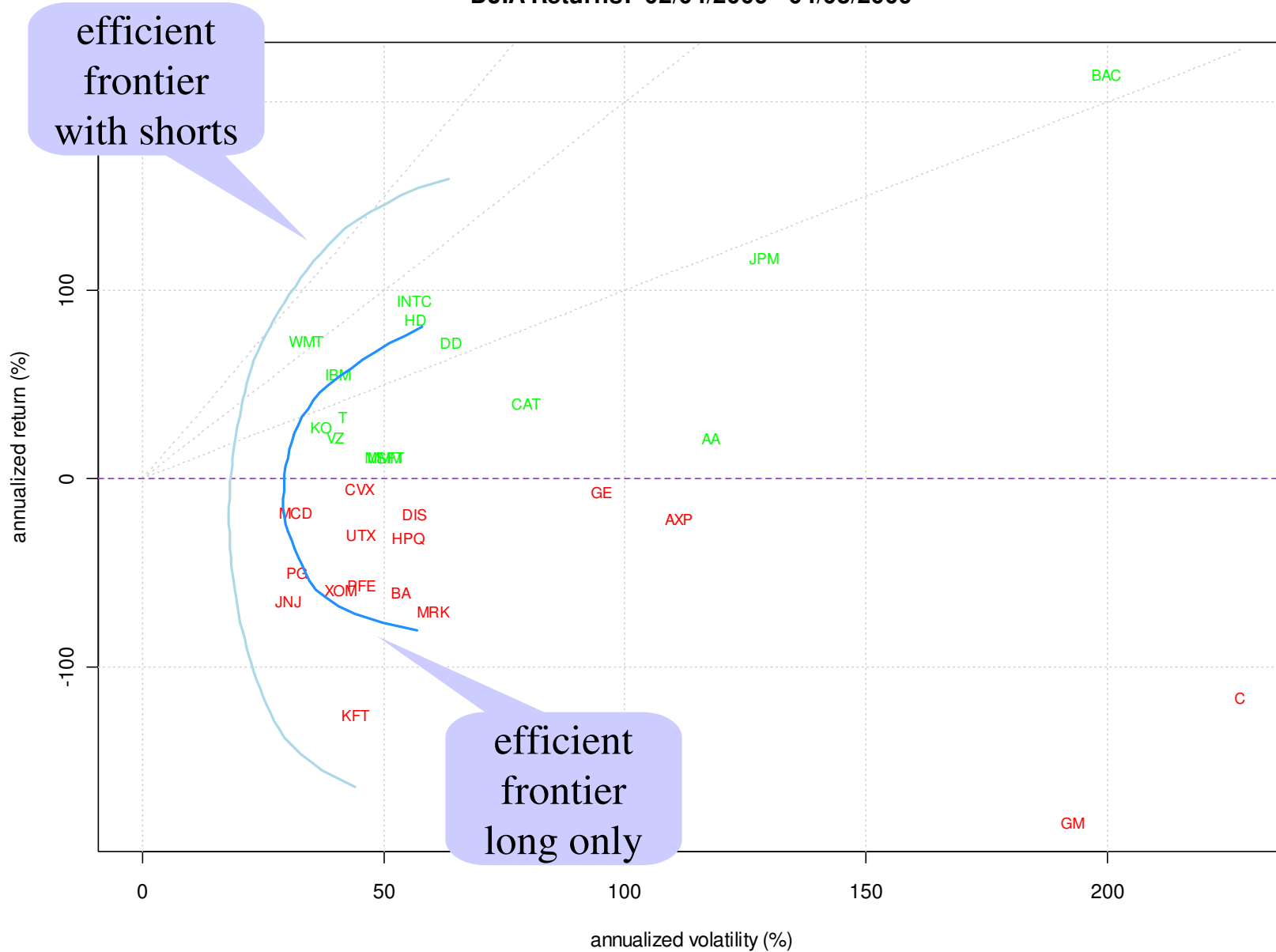
- Example

```
> averet = matrix(colMeans(r), nrow=1)  
> rcov = cov(r)  
> target.return = 15/250  
> port.sol = portfolio.optim(x = averet, pm = target.return,  
                           covmat = rcov, shorts = F, reslow = rep(0,30), reshigh = rep(0.1,30))  
> w = cleanWeights(port.sol$pw, syms)  
> w[w!=0]  
   HD  IBM INTC  JNJ   KO  MCD MSFT  PFE   PG    T   VZ  WMT  
0.05 0.10 0.04 0.10 0.10 0.10 0.07 0.04 0.10 0.10 0.10 0.10
```

1 - returns vector  
2 - covariance matrix  
3 - minimum return

# Efficient Frontier

DJIA Returns: 02/04/2009 - 04/03/2009



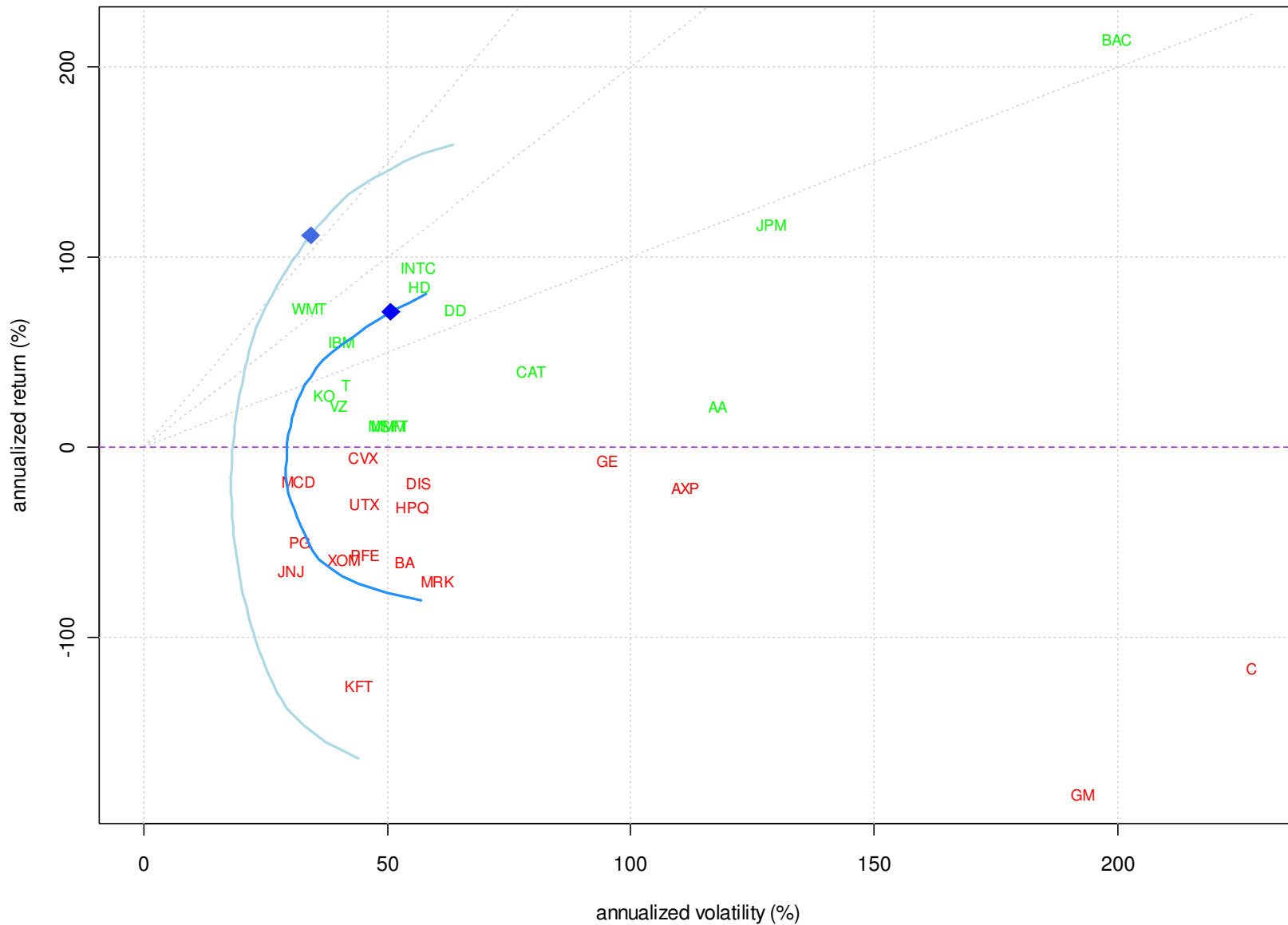
# Efficient Frontier Calculation

```
effFrontier = function (averet, rcov, nports = 20, shorts=T, wmax=1)
{
  mxret = max(abs(averet))
  mnret = -mxret
  n.assets = ncol(averet)
  reshigh = rep(wmax,n.assets)
  if( shorts )
  {
    reslow = rep(-wmax,n.assets)
  } else {
    reslow = rep(0,n.assets)
  }
  min.rets = seq(mnret, mxret, len = nports)
  vol = rep(NA, nports)
  ret = rep(NA, nports)
  for (k in 1:nports)
  {
    port.sol = NULL
    try(port.sol <- portfolio.optim(x=averet, pm=min.rets[k], covmat=rcov,
      reshigh=reshigh, reslow=reslow,shorts=shorts), silent=T)
    if ( !is.null(port.sol) )
    {
      vol[k] = sqrt(as.vector(port.sol$pw %*% rcov %*% port.sol$pw))
      ret[k] = averet %*% port.sol$pw
    }
  }
  return(list(vol = vol, ret = ret))
}
```

wrapped in try to  
handle unfeasible  
optimizations

# Maximum Sharpe Ratio

DJIA Returns: 02/04/2009 - 04/03/2009



# Maximum Sharpe Ratio

```
maxSharpe = function (averet, rcov, shorts=T, wmax = 1)
{
  optim.callback = function (param, averet, rcov, reshigh, reslow, shorts)
  {
    port.sol = NULL
    try(port.sol <- portfolio.optim(x=averet, pm=param, covmat=rcov,
    reshigh=reshigh, reslow=reslow, shorts=shorts), silent = T)
    if (is.null(port.sol)) {
      ratio = 10^9
    } else {
      m.return = averet %*% port.sol$pw
      m.risk = sqrt(as.vector(port.sol$pw %*% rcov %*% port.sol$pw))
      ratio = -m.return/m.risk
      assign("w", port.sol$pw, inherits=T)
    }
    return(ratio)
  }
}

ef = effFrontier(averet=averet, rcov=rcov, shorts=shorts, wmax=wmax, nports = 100)

n = ncol(averet)
reshigh = rep(wmax, n)
if( shorts ) {
  reslow = -reshigh
} else {
  reslow = rep(0, n)
}

max.sh = which.max(ef$ret/ef$vol)
w = rep(0, ncol(averet))
xmin = optimize(f=optim.callback, interval=c(ef$ret[max.sh-1], upper=ef$ret[max.sh+1]),
  averet=averet, rcov=rcov, reshigh=reshigh, reslow=reslow, shorts=shorts)
return(w)
}
```

callback  
function calls  
portfolio.optim()

use optimize() to  
find return level  
with maximum  
Sharpe ratio

# Solving Quadratic Programs

- Function
  - `solve.QP` {quadprog}
- Description
  - solve quadratic program

general quadratic program

$$\text{Minimize: } -d^T b + \frac{1}{2} b^T D b$$

$$\text{Subject to: } A^T b \geq b_0$$



mean-variance portfolio optimization

$$\text{Minimize: } w^T \Omega w$$

$$\text{Subject to: } \sum_i \bar{r}_i w_i = r_{min}$$

$$\sum_i w_i = 1$$

$$w_i^{min} \leq w_i \leq w_i^{max}$$

- Usage

`solve.QP(Dmat, dvec, Amat, bvec, meq=0, factorized=FALSE)`

# Extending portfolio.optim

- Modify portfolio.optim
  - Market neutral (weights sum to zero)

```
if (!is.null(reslow) & !is.null(reshigh)) {  
  a3 <- matrix(0, k, k)  
  diag(a3) <- 1  
  Amat <- t(rbind(a1, a2, a3, -a3))  
  b0 <- c(1, pm, reslow, -reshigh)  
} else {  
  Amat <- t(rbind(a1, a2))  
  b0 <- c(1, pm)  
}
```



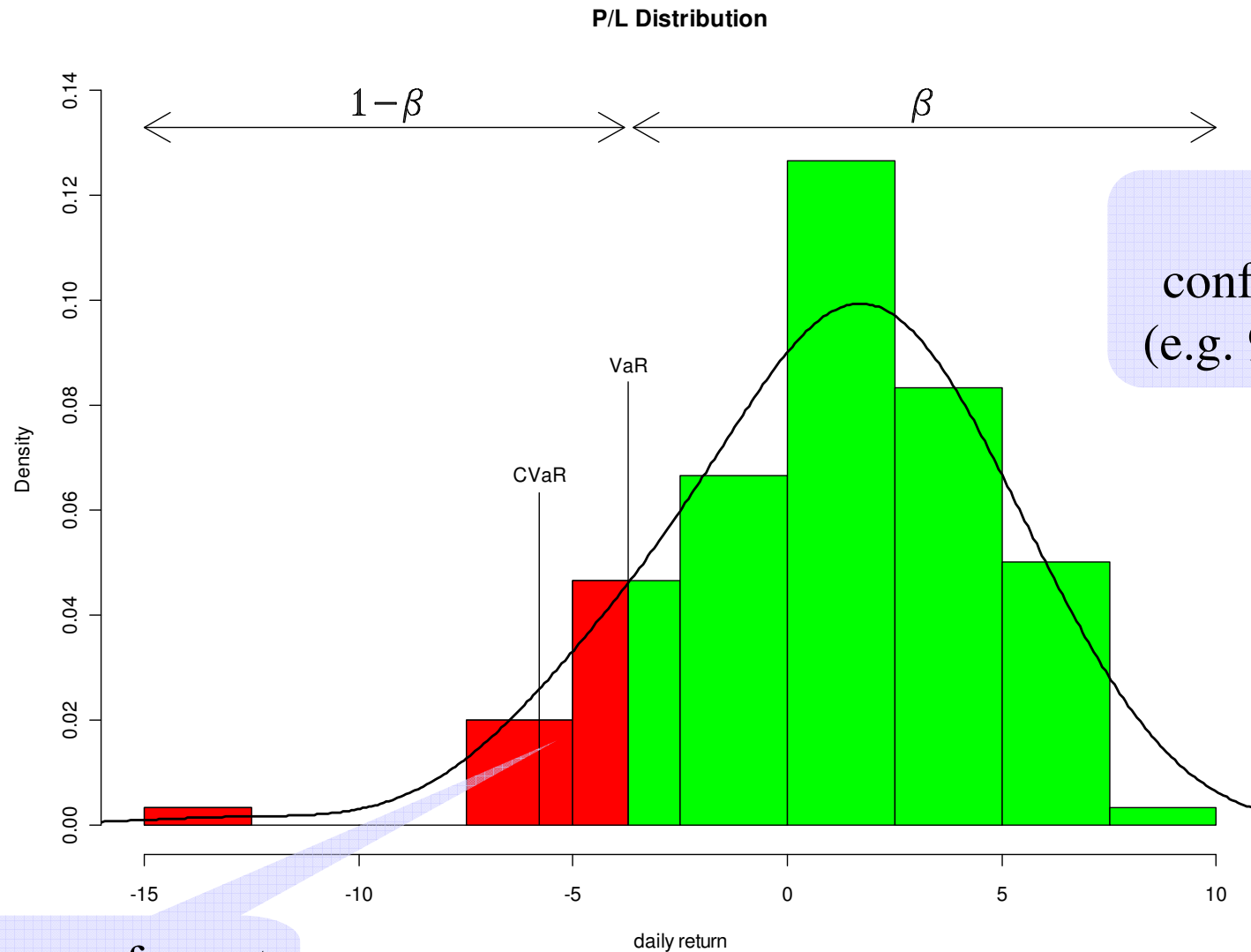
```
if (!is.null(reslow) & !is.null(reshigh)) {  
  a3 <- matrix(0, k, k)  
  diag(a3) <- 1  
  Amat <- t(rbind(a1, a2, a3, -a3))  
  b0 <- c(weight.sum, pm, reslow, -reshigh)  
} else {  
  Amat <- t(rbind(a1, a2))  
  b0 <- c(weight.sum, pm)  
}
```

- Call solve.QP directory
  - add group constraints
  - add linear transaction cost constraints
  - etc.

# Outline

- Mean-Variance Portfolio Optimization
  - quadratic programming
    - tseries, quadprog
- Conditional Value-at-Risk Optimization
  - linear programming
    - Rglpk\_solve\_LP package
- General Nonlinear Optimization
  - Differential Evolution Algorithm
    - DEoptim package
      - Omega Optimization
      - Adding Constraints
      - Maximum Drawdown Optimization
      - R-Ratio Optimization
- Wrap-Up

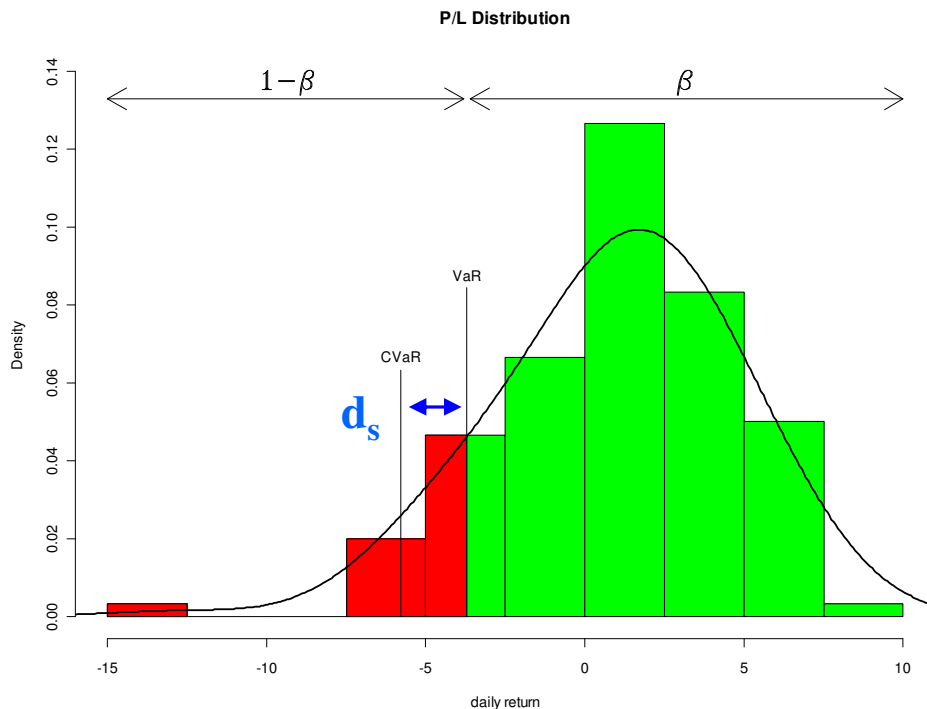
# Conditional Value-at-Risk



mean of worst  $1-\beta$  losses

# CVaR Optimization as a Linear Program

- CVaR



$$R_{CVAR}(w, \beta) = R_{VAR} + \underbrace{\frac{1}{S} \sum_{s=1}^S \max(R_{VAR} - w'R_s, 0)}_{\substack{\text{average excess loss over all scenarios} \\ \text{probability of excess loss} \\ \text{conditional excess loss} \\ \text{average loss if loss occurs}}}$$

- CVaR Optimization

Minimize:  $R_{VAR} + \frac{1}{S} \frac{1}{1-\beta} \sum_{s=1}^S d_s$

Subject to:  $d_s \geq R_{VAR} + w'R_s$

$$d_s \geq 0$$

$$w'R \geq R_{min}$$

$$\sum_i w_i = 1$$

see B. Sherer 2003

# Solving Linear Programs

- Function
  - `Rglpk_solve_LP` {Rglpk}
- Description
  - solves linear and MILP programs (via GNU Linear Programming Kit)

general linear program

$$\begin{array}{ll} \text{Minimize:} & c^T x \\ \text{Subject to:} & Ax \geq b_0 \end{array}$$



CVaR portfolio optimization

$$c^T = \left[ 0 \quad 0 \quad \dots \quad 0 \quad \frac{-1}{(1-\beta)S} \quad \frac{-1}{(1-\beta)S} \quad \dots \quad \frac{-1}{(1-\beta)S} \quad -1 \right]$$

$$x^T = [w_1 \quad w_2 \quad \dots \quad w_n \quad d_1 \quad d_2 \quad \dots \quad d_S \quad R_{VaR}]$$

$$A = \begin{bmatrix} 1 & 1 & \dots & 1 & 0 & \dots & 0 & 0 \\ \bar{r}_1 & \bar{r}_2 & \dots & \bar{r}_n & 0 & \dots & 0 & 0 \\ r_{11} & r_{12} & \dots & r_{1n} & 1 & 0 & \dots & 1 \\ r_{21} & r_{22} & \dots & r_{2n} & 0 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 1 \\ r_{s1} & r_{s2} & \dots & r_{sn} & 0 & \dots & 1 & 1 \end{bmatrix} \quad b_0 = \begin{bmatrix} 1 \\ r_{min} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- Usage

```
Rglpk_solve_LP(obj, mat, dir, rhs, types = NULL, max = FALSE,
               bounds = NULL, verbose = FALSE)
```

# CVaR Optimization

```
cvarOpt = function(rmat, alpha=0.05, rmin=0, wmin=0, wmax=1, weight.sum=1)
{
  require(Rglpk)
  n = ncol(rmat) # number of assets
  s = nrow(rmat) # number of scenarios i.e. periods
  averet = colMeans(rmat)

  # creat objective vector, constraint matrix, constraint rhs
  Amat = rbind(cbind(rbind(1, averet), matrix(data=0, nrow=2, ncol=s+1)),
              cbind(rmat, diag(s), 1))
  objL = c(rep(0, n), rep(-1/(alpha*s), s), -1)
  bvec = c(weight.sum, rmin, rep(0, s))

  # direction vector
  dir.vec = c("==", ">=", rep(">=", s))

  # bounds on weights
  bounds = list(lower = list(ind = 1:n, val = rep(wmin, n)),
               upper = list(ind = 1:n, val = rep(wmax, n)))

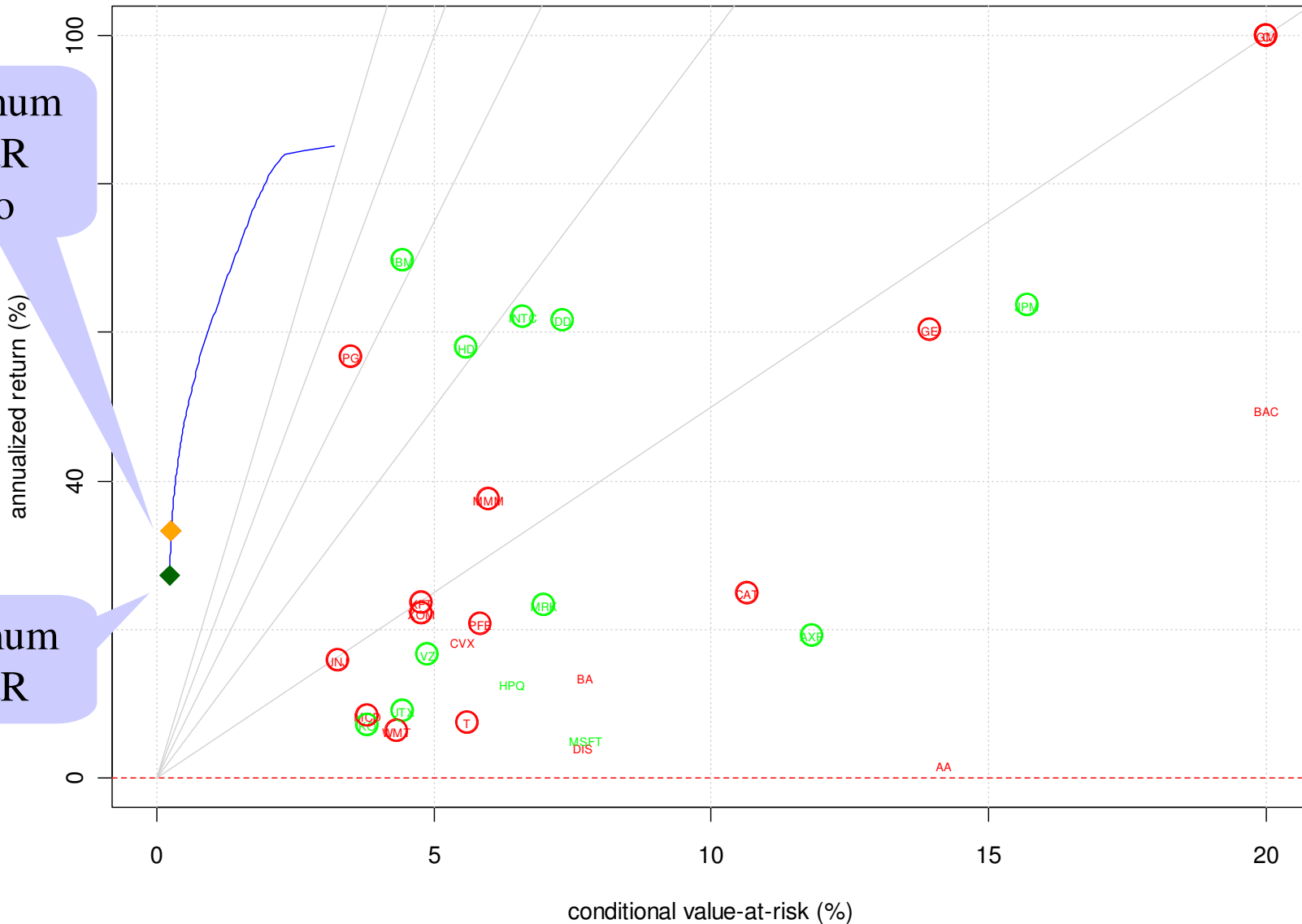
  res = Rglpk_solve_LP(obj=objL, mat=Amat, dir=dir.vec, rhs=bvec,
                      types=rep("C", length(objL)), max=T, bounds=bounds)

  w = as.numeric(res$solution[1:n])
  return(list(w=w, status=res$status))
}
```

supports general  
equality/inequality  
constraints

# CVaR Efficient Frontier

DJIA: 12/02/2008 - 04/15/2009



Note: some assets not displayed due to cropping

**R Tools for Portfolio Optimization**

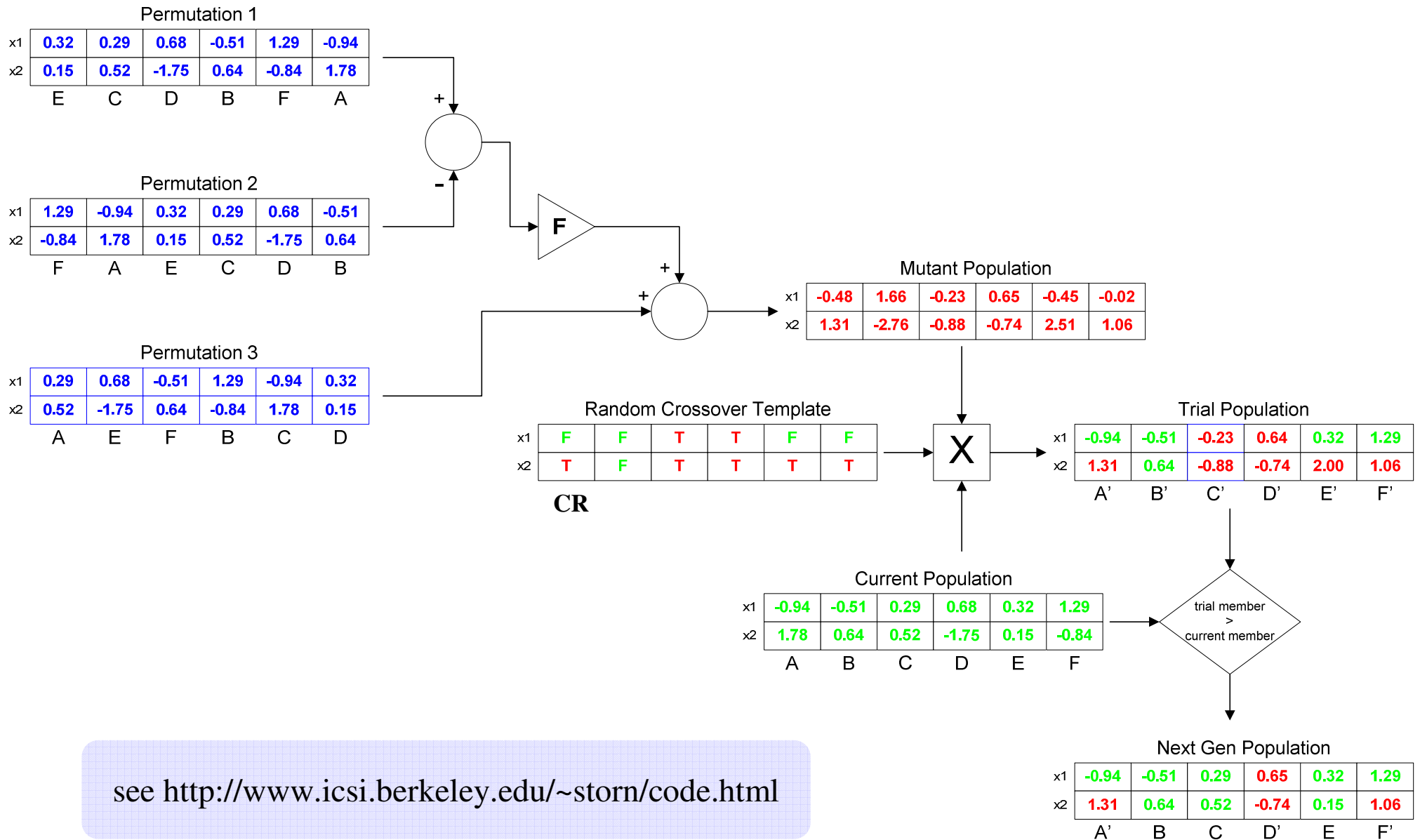
# Outline

- Mean-Variance Portfolio Optimization
  - quadratic programming
    - tseries, quadprog
- Conditional Value-at-Risk Optimization
  - linear programming
    - Rglpk\_solve\_LP package
- **General Nonlinear Optimization**
  - **Differential Evolution Algorithm**
    - **DEoptim package**
      - Omega Optimization
      - Adding Constraints
      - Maximum Drawdown Optimization
      - R-Ratio Optimization
- Wrap-Up

# Differential Evolution

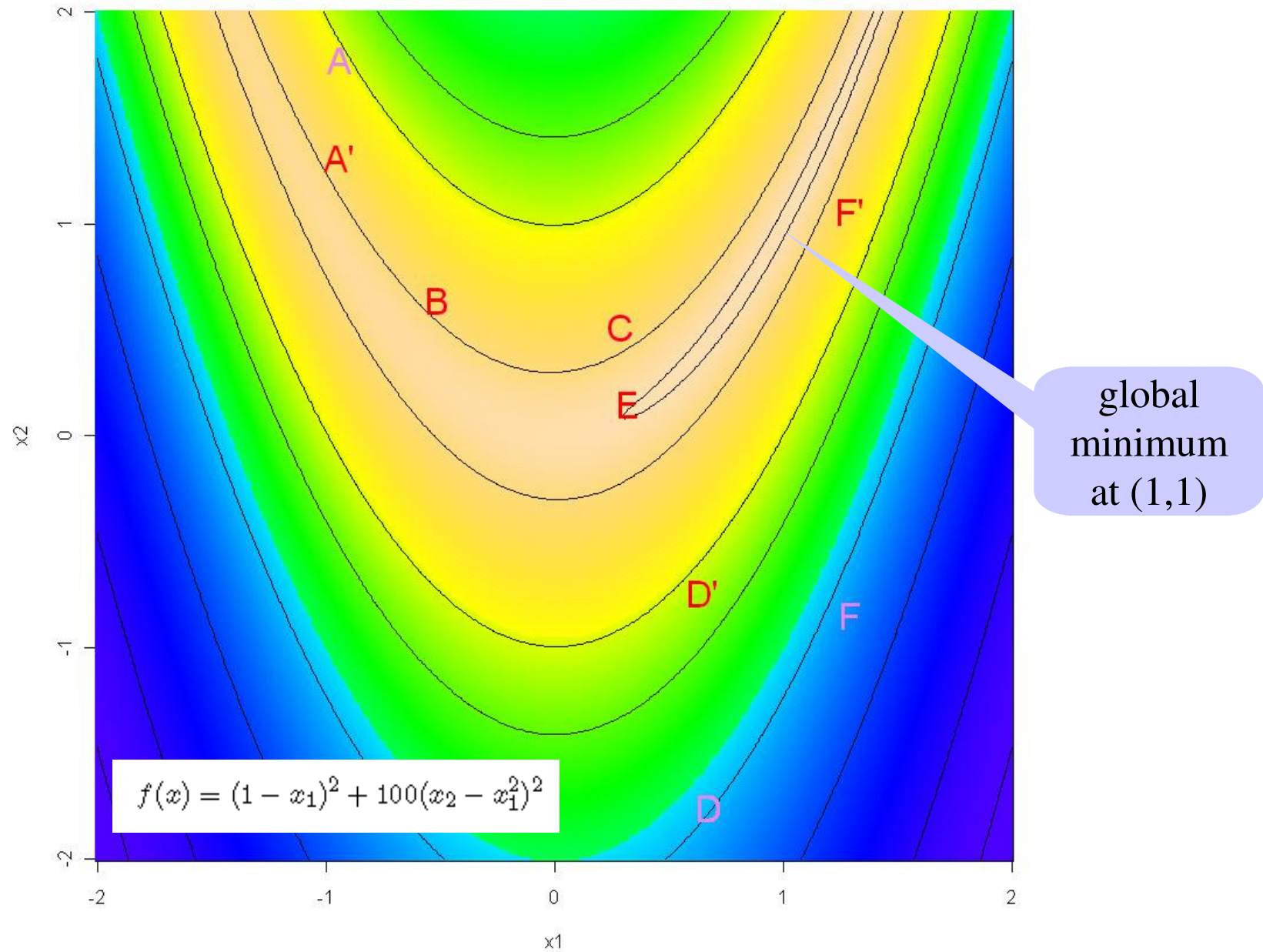
- DE is a very **simple** and yet very **powerful** population based stochastic function minimizer
- Ideal for global optimization of multidimensional multimodal functions (i.e. *really hard problems*)
- Developed in mid-1990 by Berkeley researchers Ken Price and Rainer Storm
- Implemented in R in the package DEoptim

# Differential Evolution Algorithm



see <http://www.icsi.berkeley.edu/~storn/code.html>

# DE Example



# Differential Evolution Function

- Function

- DEoptim {DEoptim}

- Description

- performs evolutionary optimization via differential evolution algorithm

- Usage

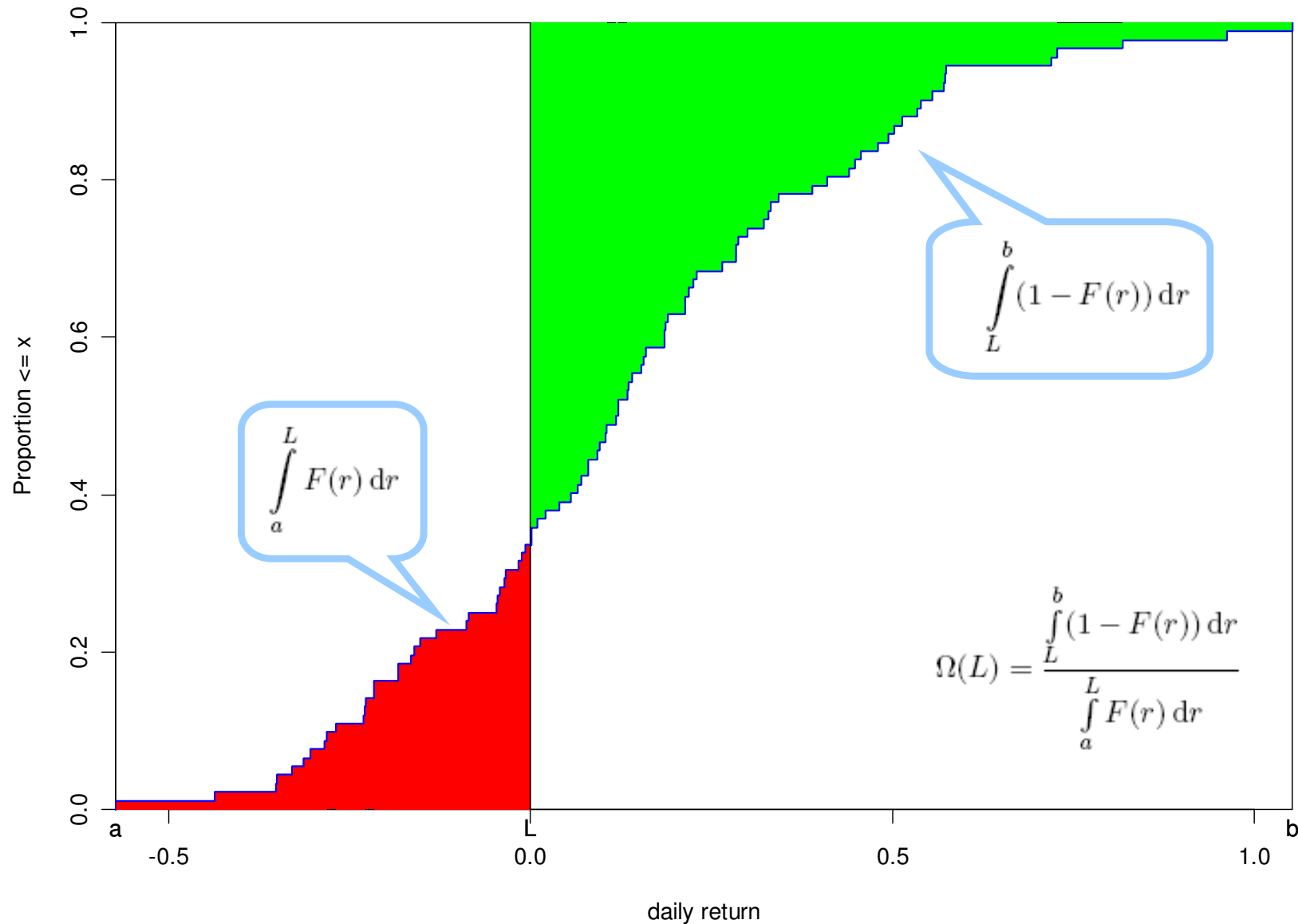
```
DEoptim(FUN, lower, upper, control = list(), ...)
```

- Example

```
> lower = c(-2,-2)
> upper = c(2,2)
> res = DEoptim(banana, lower, upper)
> res$optim
$bestmem
      par1      par2
0.9987438 0.9976079
$bestval
[1] 2.986743e-06
$nfeval
[1] 5050
$iter
[1] 100
```

# Omega Performance Measure

Portfolio Returns Cumulative Distribution Function



# Omega Performance Measure

- Omega Performance Measure:

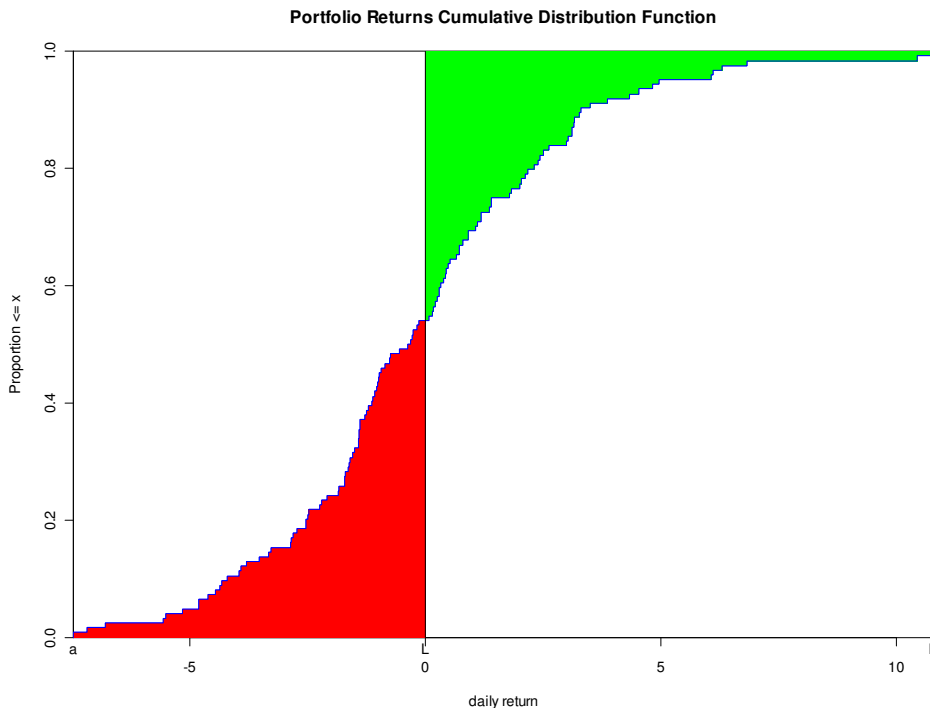
$$\Omega(L) = \frac{\int_a^b (1 - F(r)) dr}{\int_a^L F(r) dr}$$

- utilizes entire returns distribution
- ratio of call price to put price with strike at L:

$$\Omega(L) = \frac{C(L)}{P(L)}$$

- simple calculation:

$$\text{omega} = \text{mean}(\text{pmax}(r-L, 0)) / \text{mean}(\text{pmax}(L-r, 0))$$



See  
Keating & Shadwick 2002  
Kazemi et. al., 2003

# Omega Optimization

Maximize:  $\Omega(L)$   
Subject to:  $\sum_i |w_i| = 1$   
 $0 \leq w_i \leq w_i^{max}$

objective  
function

```
optOmega = function(x,ret,L)
{
  retu = ret %*% x
  obj = -Omega(retu,L=L,method="simple")
  weight.penalty = 100*(1-sum(x))^2
  return( obj + weight.penalty )
}
```

calls Omega() from  
PerformanceAnalytics

```
> lower = rep(0,n.assets)
> upper = rep(wmax,n.assets)
```

```
> res = DEoptim(optOmega,lower,upper,
  control=list(NP=2000,itermax=1000,F=0.2,CR=0.8),
  ret=coredata(r),L=L)
```

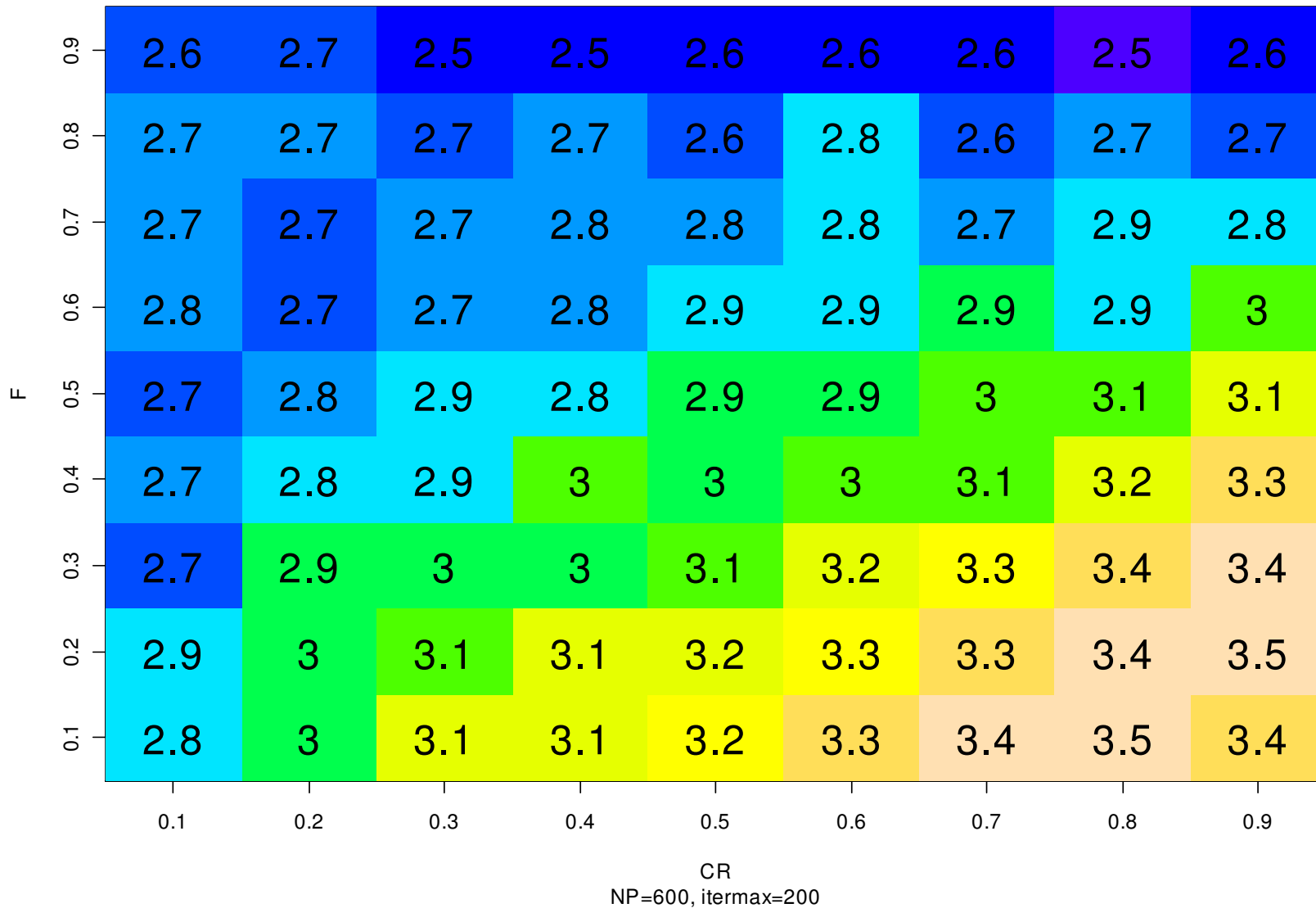
```
> w = cleanWeights(res$optim$bestmem,syms)
```

```
> w[w!=0]
```

AXP	BA	C	CAT	CVX	DD	DIS	GE	GM	HD	IBM	INTC	JNJ	KO	MCD	MMM
0.02	0.03	0.02	0.04	0.05	0.08	0.01	0.02	0.01	0.03	0.04	0.09	0.05	0.08	0.05	0.04
MRK	PG	T	UTX	VZ	WMT	XOM									
0.04	0.10	0.08	0.04	0.06	0.03	0.00									

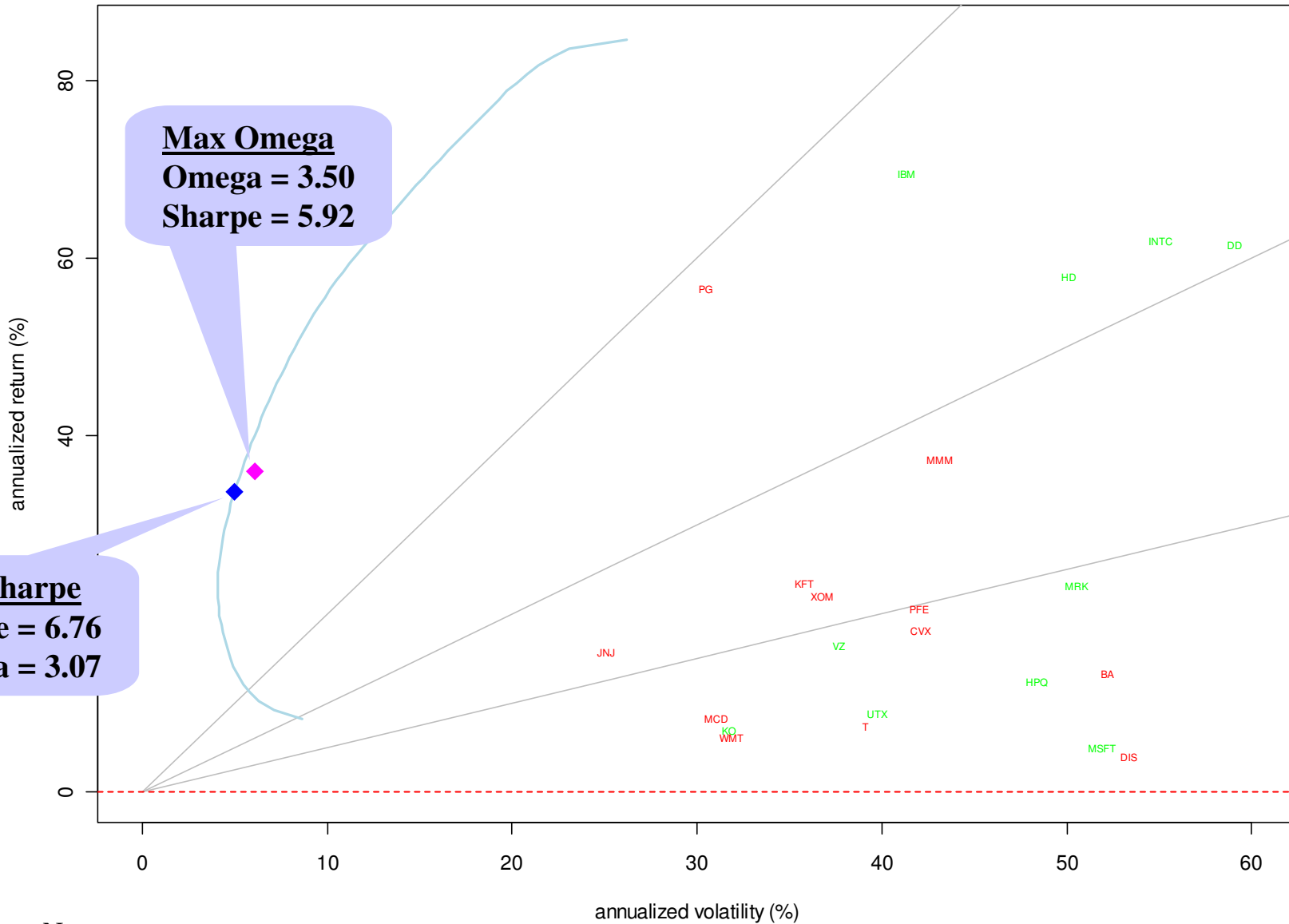
# Effect of DE Parameters on Optimization

Optimal Omega as a function of F and CR



# Max Omega versus Max Sharpe

DJIA: 12/02/2008 - 04/15/2009

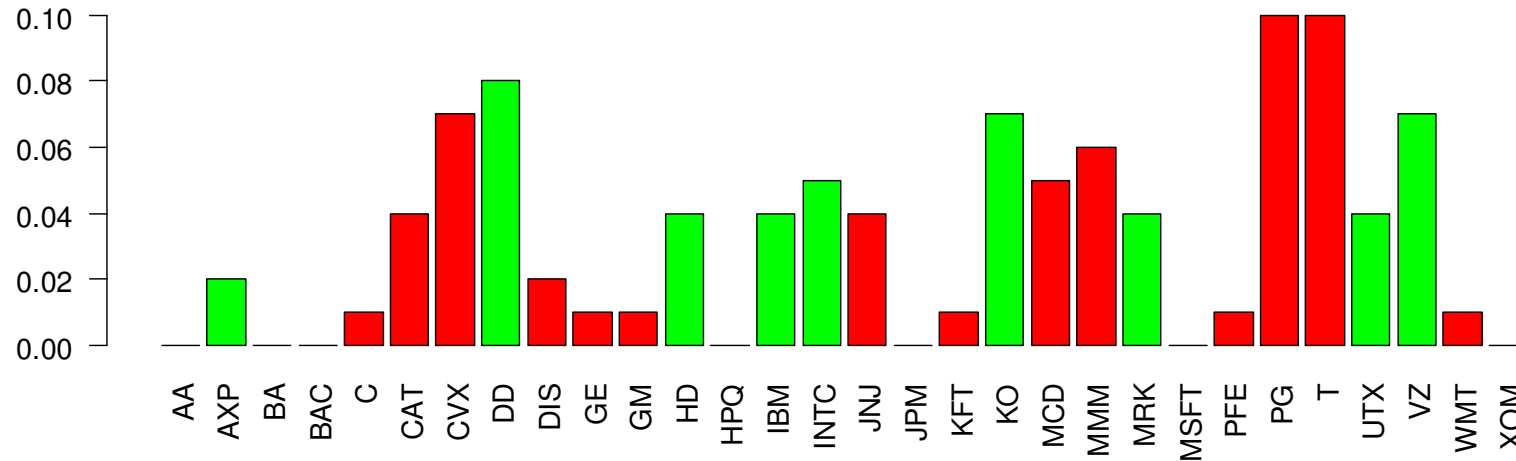


Note: some assets not displayed due to cropping

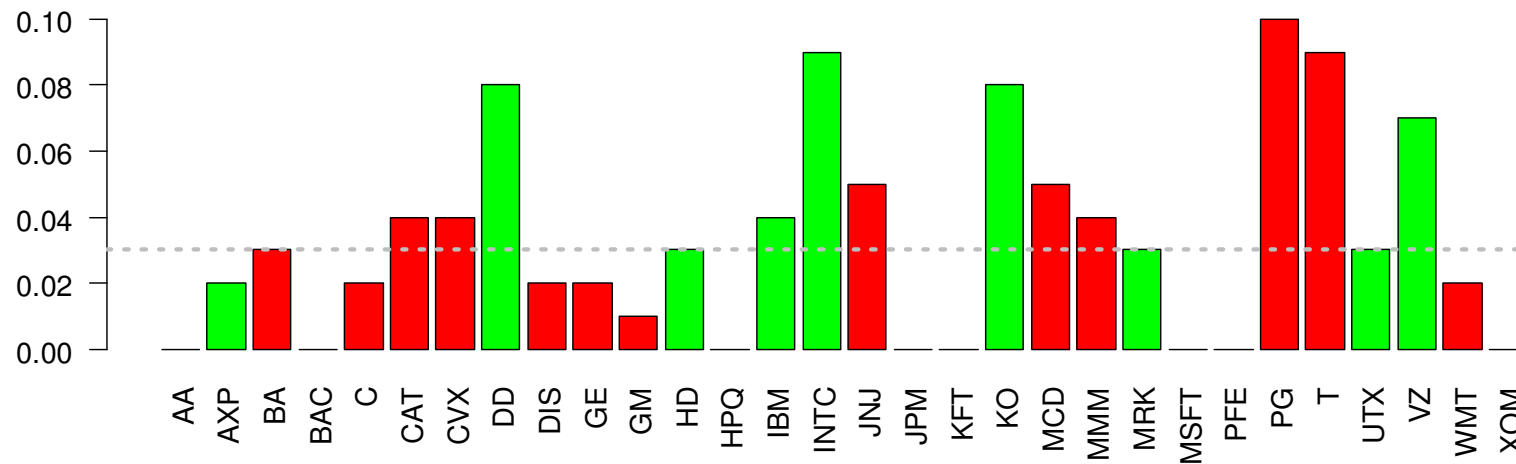
**R Tools for Portfolio Optimization**

# Weight Comparison

Weights of Max Sharpe Portfolio  
Omega = 3.07



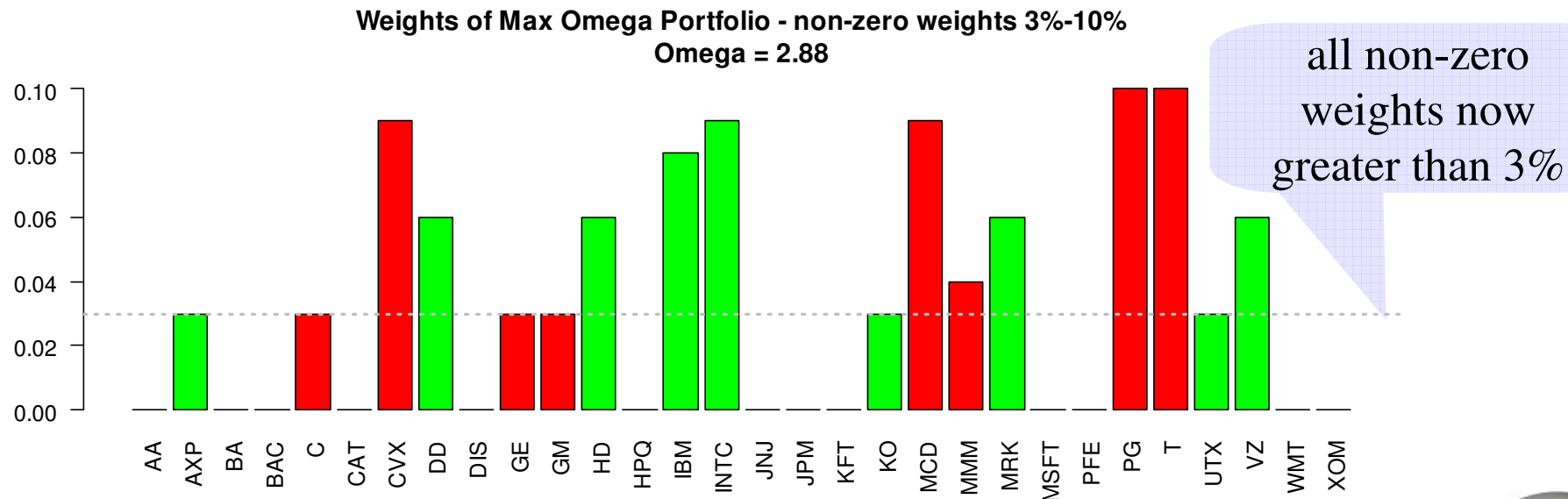
Weights of Max Omega Portfolio  
Omega = 3.5



# Optimization with Additional Constraints

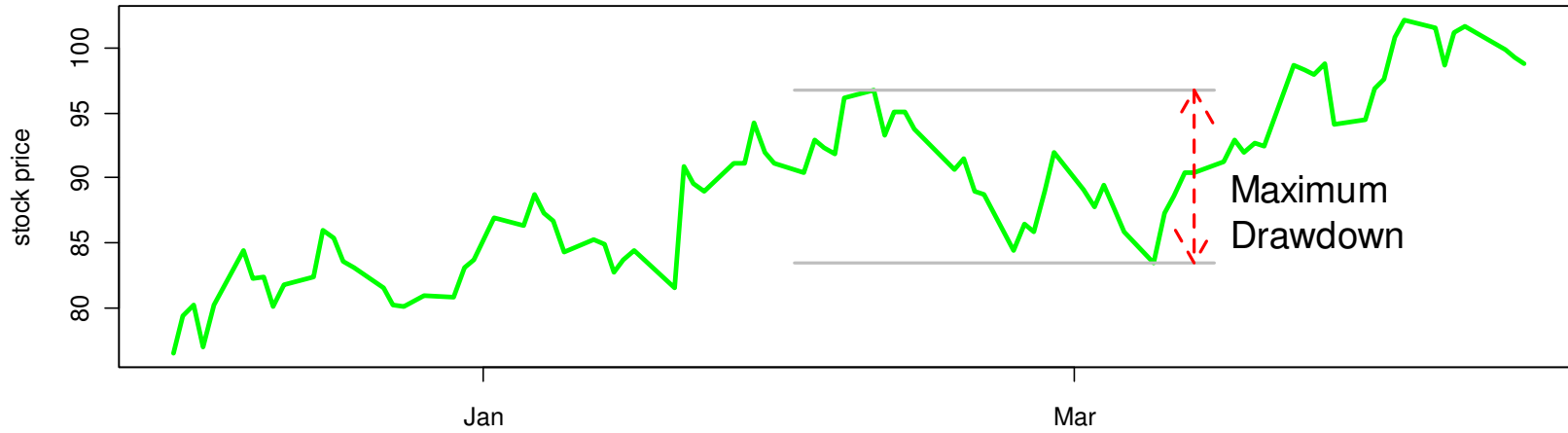
```
# max omega with non-zero weights between 3% & 10%
optOmega.gt3 = function(x,ret,L)
{
  retu = ret %*% x
  obj = -Omega(retu,L=L,method="simple")
  weight.penalty = 100*(1-sum(x))^2
  small.weight.penalty = 100*sum(x[x<0.03])
  return( obj + weight.penalty + small.weight.penalty )
}

res = DEoptim(optOmega.gt3,lower,upper,
  control=list(NP=2000,itermax=1000,F=0.2,CR=0.8),
  ret=codata(r),L=L)
```

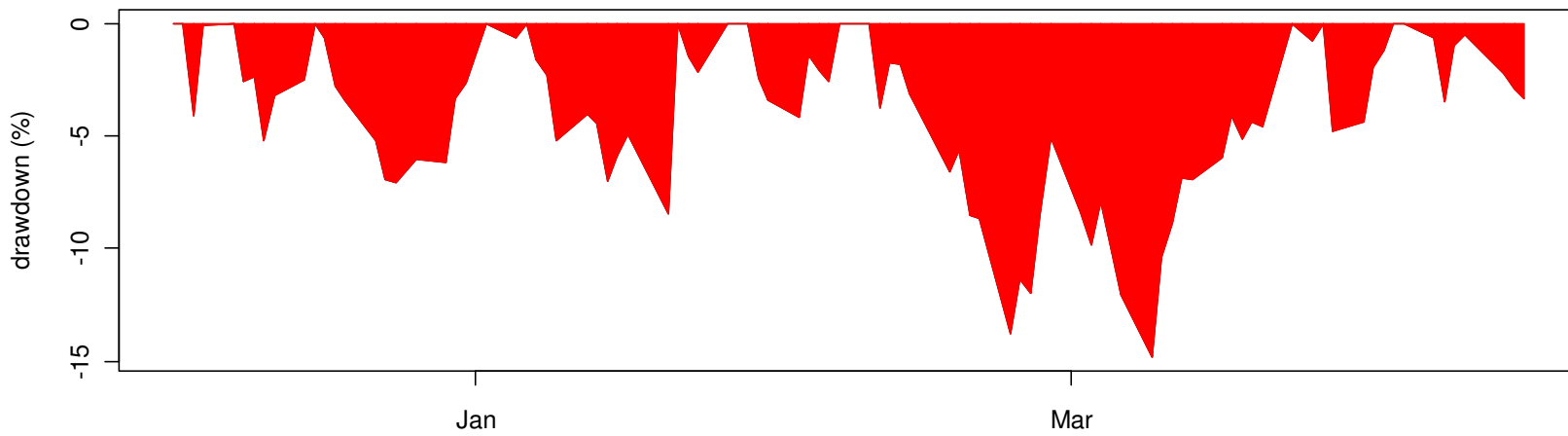


# Maximum Drawdown

IBM: 12/02/2008 - 04/15/2009



IBM Underwater Graph



# Maximum Drawdown Optimization

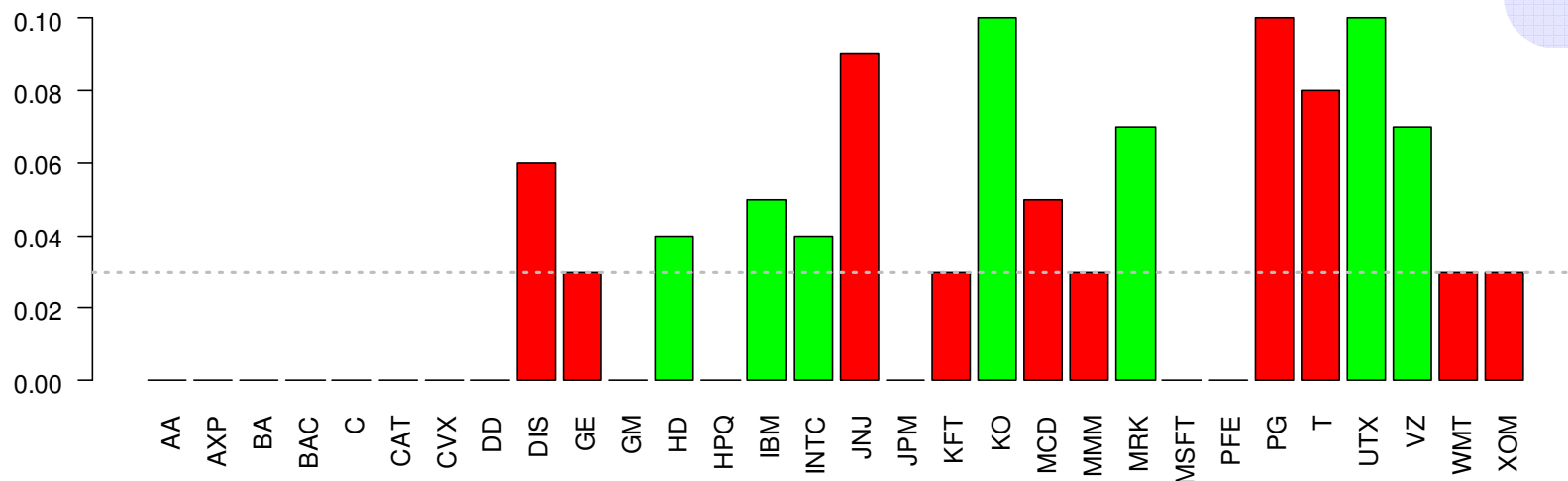
```
# max drawdown with non-zero weights between 3% & 10%
optMDD.gt3 = function(x, ret)
{
  retu = ret %*% x
  obj = mddx(retu, 1)
  weight.penalty = 100*(1-sum(x))^2
  small.weight.penalty = 100*sum(x[x<0.03])
  return( obj + weight.penalty + small.weight.penalty )
}

res = DEoptim(optMDD.gt3, lower, upper,
  control=list(NP=2000, itermax=1000, F=0.2, CR=0.8),
  ret=coredata(r))
```

function return  
the mean of the  
top n drawdowns  
(in this case n=1)

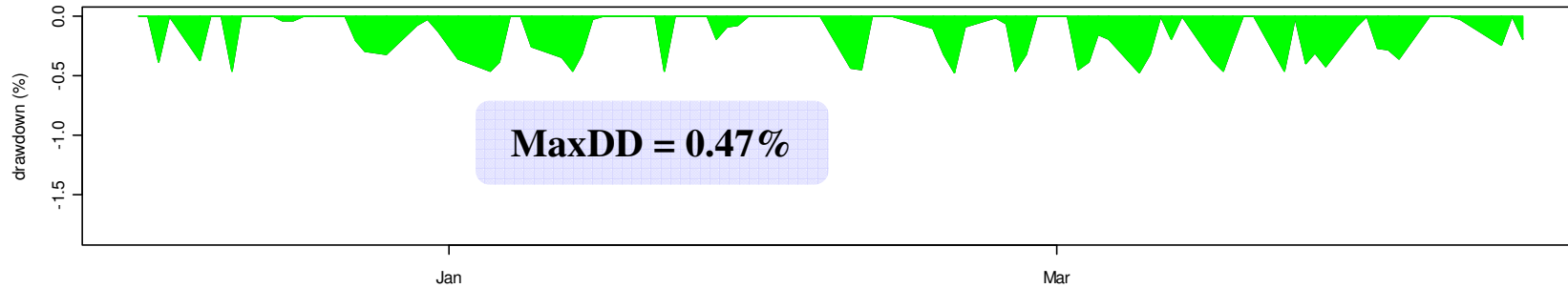
could readily  
implement  
optimization  
on Calmar or  
Sterling  
ratios

Weights of Portfolio Optimized on Maximum Drawdown

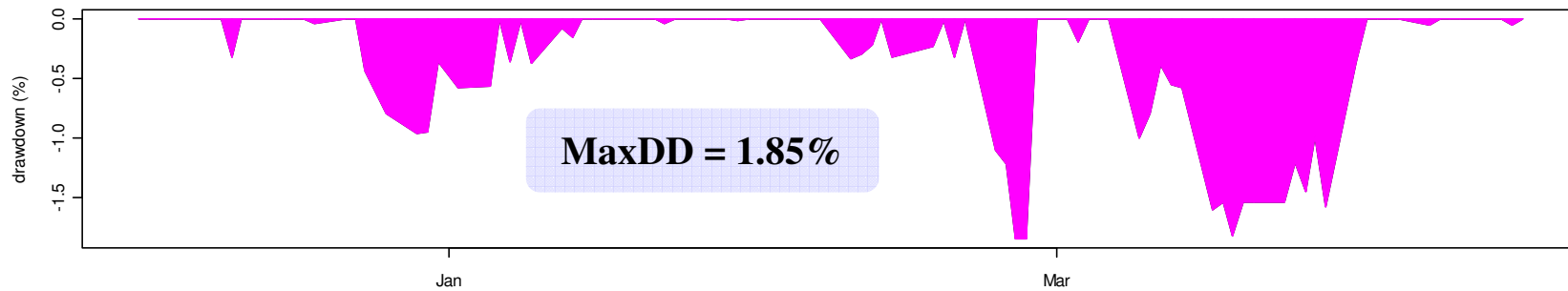


# Maximum Drawdown Optimization

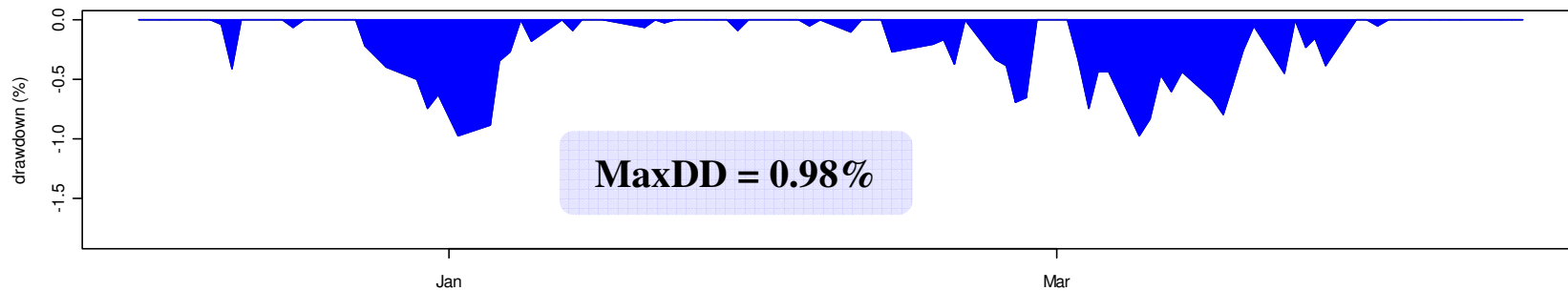
Optimized on Max Drawdown



Optimized on Omega Ratio

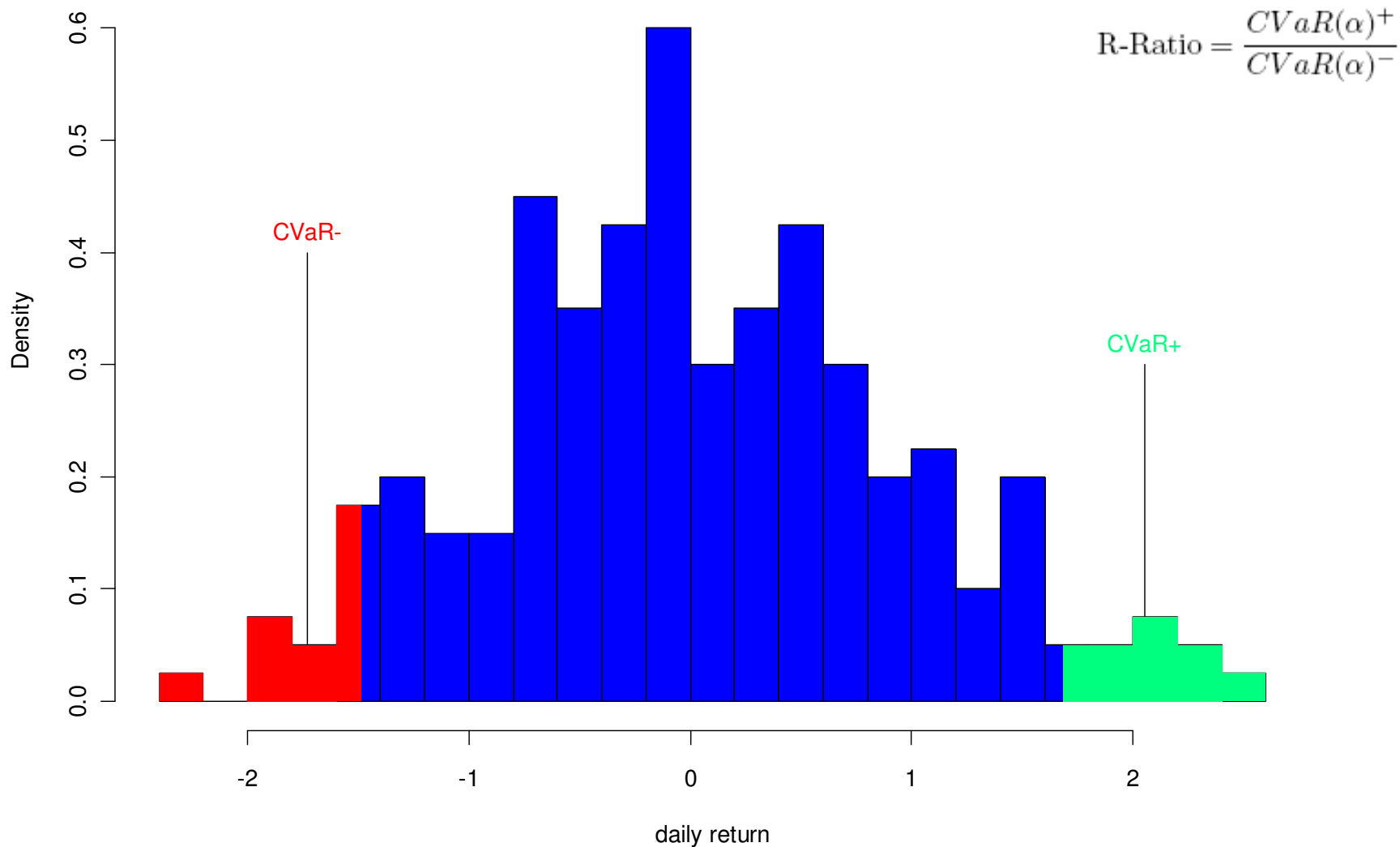


Optimized on Sharpe Ratio



# Rachev Ratio (R-Ratio)

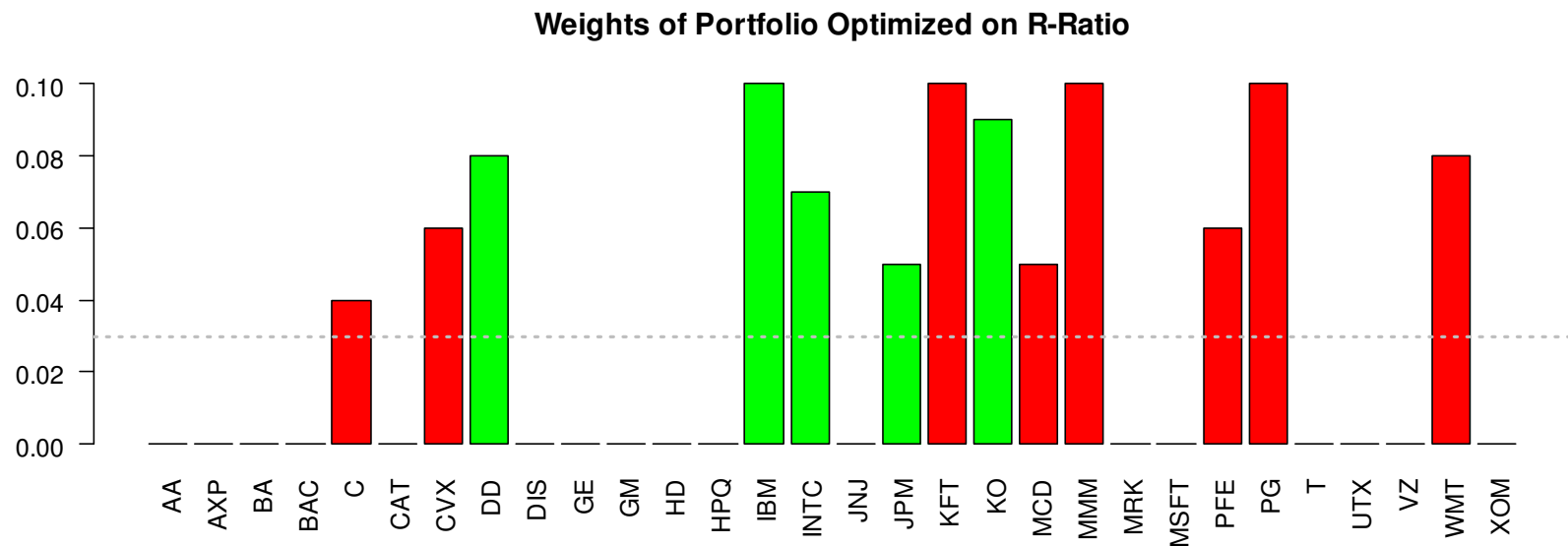
P/L Distribution



# R-Ratio Optimization

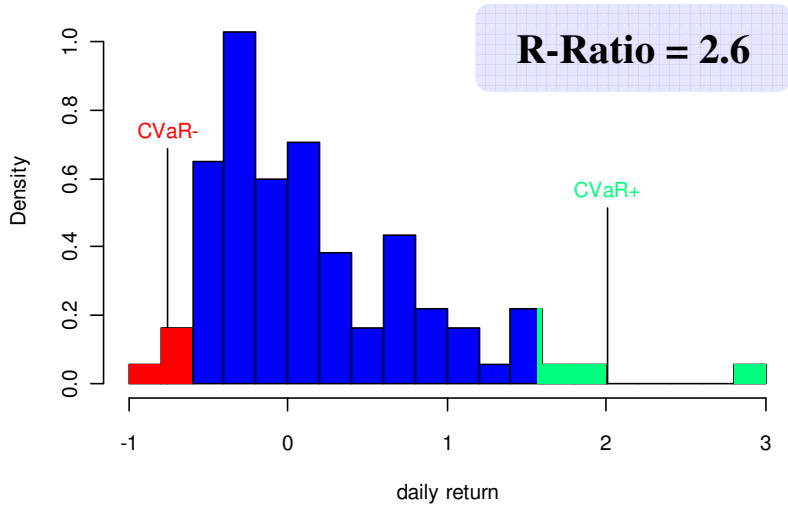
```
optRR.gt3 = function(x, ret)
{
  retu = ret %*% x
  obj = -CVaR(-retu)/CVaR(retu)
  weight.penalty = 100*(1-sum(x))^2
  small.weight.penalty = 100*sum(x[x<0.03])
  return( obj + weight.penalty + small.weight.penalty )
}

res = DEoptim(optRR.gt3, lower, upper,
  control=list(NP=2000, itermax=1000, F=0.2, CR=0.8),
  ret=coredata(r))
```

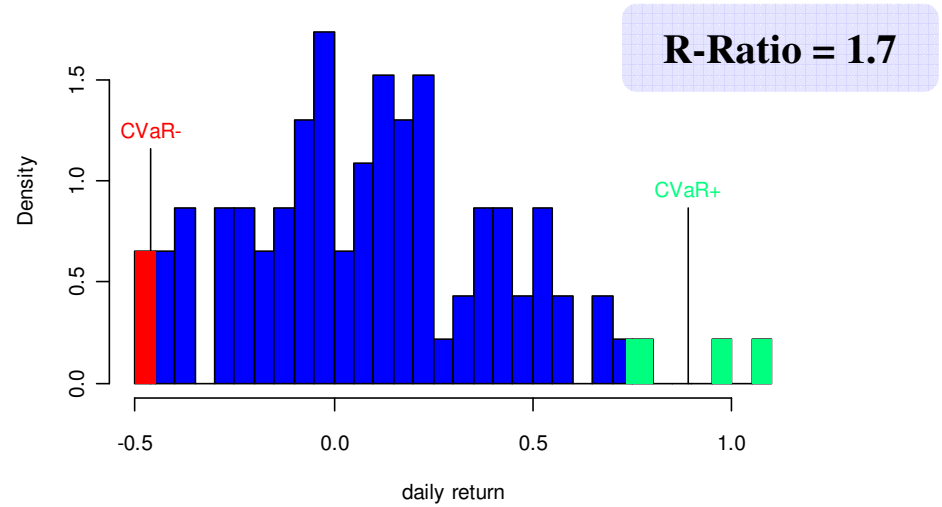


# R-Ratio Comparison

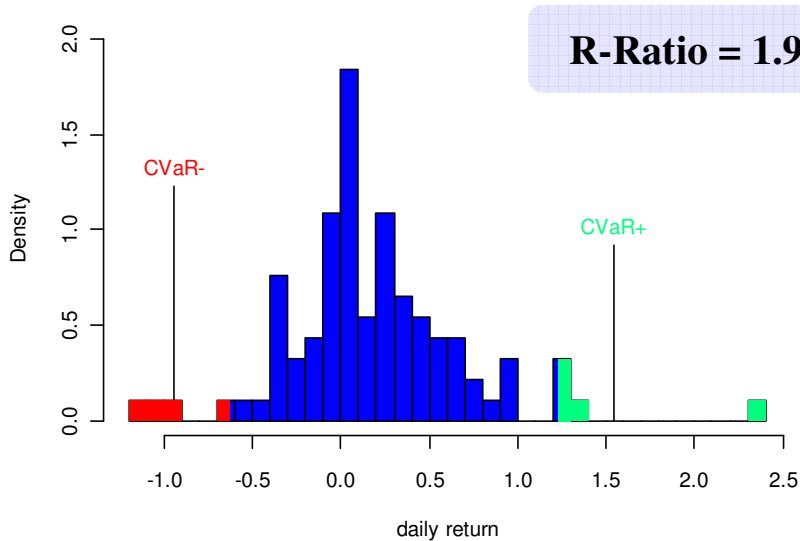
Optimal R-Ratio



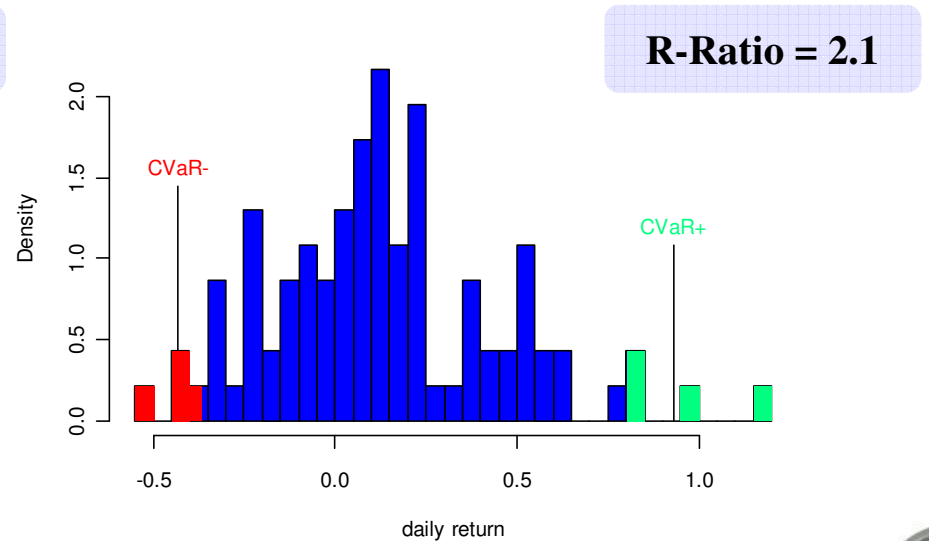
Optimal Max DD



Optimal Omega



Optimal Sharpe



# Summary

- Mean-Variance Portfolio Optimization
  - high-level function: `portfolio.optim`
  - low-level function: `solve.QP`
- Linear Programming Optimization
  - `Rglpk_solve_LP()`
    - Conditional Value at Risk (CVaR):
    - MAD, Semivariance & others
- General-purpose non-linear optimization
  - `DEoptim`
    - Omega
    - non-linear constraints
    - maximum drawdown
    - R-Ratio
    - 130/30 portfolio optimization

# Thank You

- Questions & Answers
- Contact the Author
  - Guy Yollin
  - [guy.yollin@rotellacapital.com](mailto:guy.yollin@rotellacapital.com)
- Learn more about Rotella Capital Management
  - [businessdevelopment@rotellacapital.com](mailto:businessdevelopment@rotellacapital.com)
  - (425)-213-5700