A short course in practical financial optimisation

Enrico Schumann

Rimini, May 2016

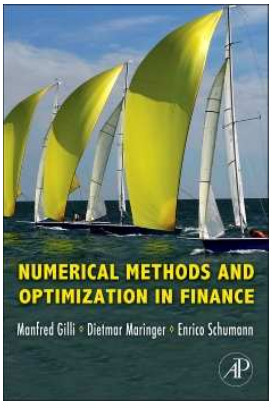
Outline

- practical optimisation: models, data, tools/software
- optimisation
  - financial models and how to solve them with heuristics
  - specific algorithms: Local Search, Threshold Accepting, Differential Evolution, ...
- portfolio optimisation with heuristics
- practical sessions

E. Schumann 2

Reading + Software

Gilli, Maringer, and Schumann (2011)



E. Schumann 3

Outline

finance/economics not a science → all about opinion

in academia, different people do different things, but they publish

at firms, different people do different things, but they do not publish

E. Schumann 4

Computers: more is possible

more computing power/storage

- larger datasets
- look at more models
- use more complicated models
- visualise models/data
- rely more on brute force
- but also: better software (user-friendly)

Software

typical tasks

- collect data, prepare data, store data, update data (databases)
- clean data (remove errors, missing values, make consistent)
- 'look at your data'
- explore data: visualise, summarise
- create reports
- do 'research'

→ data analysis requires programming

→ do not expect to do everything with one language/tool

Software

tools

- high-level languages: R, Perl, Python, ...
- low-level languages: C, ...
- 'little' languages, regular expressions
- Unix/GNU/Linux tools (`diff`, `wc`, ...)
- database software

R

- `http://www.r-project.org/`
- elegant and powerful high-level language
- free, as in free speech and as in free beer
- platform-independent
- graphics
- reporting: integration with LaTeX, HTML and other markup languages
- text computations (regular expressions, ...)
- connections: internet, databases
- lower-level language interfaces
- GUIs
- large community

E. Schumann 8

Principles

- be systematic!
  - analyse good and bad results
  - keep diaries/logs
  - write code as if you had to make it public (and do that)
- there is a 'good enough' for everything!
  - optimal vs not-optimal
  - nothing, something better, something pretty good, ...
- go experiment!

  how do you *know* how to ...? → how do you *decide* how to ...?
  → reinventing the wheel is a good idea (usually)

E. Schumann 9

Problems and models

given: a question

- what assets to buy for a retirement savings account?
- how to price a security?
- ...

modelling: objective function and constraints

$$\min_x \; f(x, \text{data})$$

- financial considerations (how to measure risk/reward?)
- statistical considerations (estimate/forecast/approximate/simulate...)
- computational considerations

Problems

what to model?

- goals – be careful what you wish for [if you don't ask for it, you probably won't get it]
- constraints – empirical, technological, regulatory, manpower, ...

example: asset allocation

- assets, data availability, legal constraints
- investment process (frequency of rebalancing, risk characteristics, stop loss, ...)
- forecasts
- scenarios for risk management
- how to evaluate portfolios?

Examples – asset allocation

portfolio optimisation: allocate wealth among a given set of assets

Markowitz: select portfolio that is mean–variance efficient

...many alternative models
based on distribution of portfolio returns

- moments (variance, skewness, ...)
- conditional moments (expected shortfall, ...), partial moments (semivariance, ...)
- quantiles (VaR, ...), corresponding probabilities (shortfall probability, ...)

based on trajectory of portfolio wealth

- drawdown, time under water, ...

more constraints

- several objectives
- cardinalities, min-thresholds
- sectors, factor exposures
- difference between current portfolio and 'suggested' portfolio
- ...

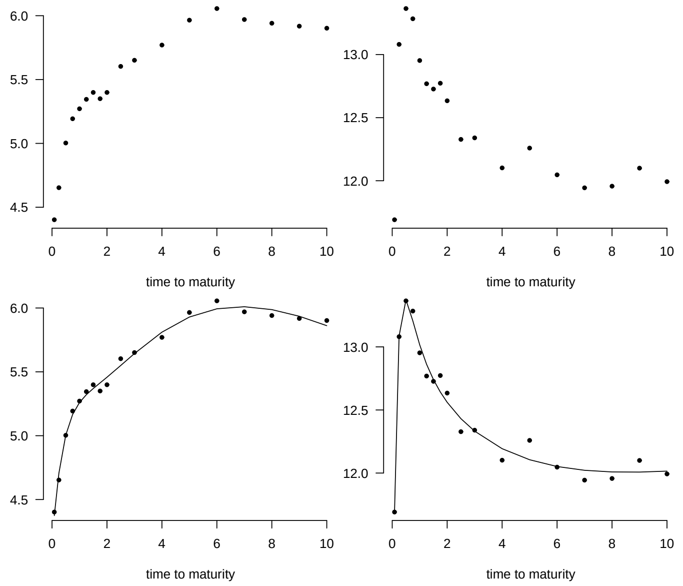
Examples – interest rate modelling

given: prices/quotes of bonds, deposits, ...

aim: interpolate given prices → yield curve

model should be flexible, simple, interpretable

examples (data from Diebold and Li, 2006)



Examples – option pricing

- position/portfolio optimisation: select strikes/maturities
- calibration: given prices/quotes → obtain vol curve
- dynamic strategies: hedging, vol trading, ...

Examples – accounting

k managed accounts

buy $n = n_1 + n_2 + \cdots + n_k$ shares

m executions

aim: assign the executions in a fair way (roughly same entry prices for all clients)

	account 1	account 2	account 3
n_i	20 000	10 000	2000
1000 @ 100.15	800	100	100
300 @ 100.01	100	50	150
1200 @ 100.31	80	1100	20
...			

Examples – mechanical trading

automatic execution vs PL-driven

set of rules when to buy or sell assets, depending only on past prices of assets (and possibly the rules' own performance)

model → buy/sell recommendation → PL

Examples – selection problems

subset selection

- select variables for model
- select assets from subset
- select traders
- ...

clustering

- how many clusters?
- include all observations?
- ...

How to solve a model?

$$\min_x \quad f(x, \text{data}) \quad \text{subject to constraints}$$

no solution – the solution x^*

no solution $\rightarrow$ some solution $\rightarrow$ a better solution $\rightarrow \cdots \rightarrow$ best solution

no solution $\rightarrow$ some solution $\rightarrow$ a better solution $\rightarrow \cdots \rightarrow$ $\boxed{\text{good enough}}$

How to solve a model?

$$\min_x \quad f(x, \text{data}) \quad \text{subject to constraints}$$

strategy 1: random x

good

- simple
- can be scaled/distributed
- independent of dimension of x

bad

- slow, particularly in higher dimensions

How to solve a model?

$$\min_x \quad f(x, \text{data}) \quad \text{subject to constraints}$$

strategy 2: discretise x (enumeration, grid search)

```
for x in x-values do
    evaluate f(x, ...)
end for
```

How to solve a model?

$$\min_x \quad f(x, \text{data}) \quad \text{subject to constraints}$$

strategy 2: discretise x (enumeration, grid search)

for x_1 in x_1-values **do**
  for x_2 in x_2-values **do**
    for x_3 in x_3-values **do**
      ...
      evaluate $f(x_1, x_2, ...)$
      ...
    end for
  end for
end for

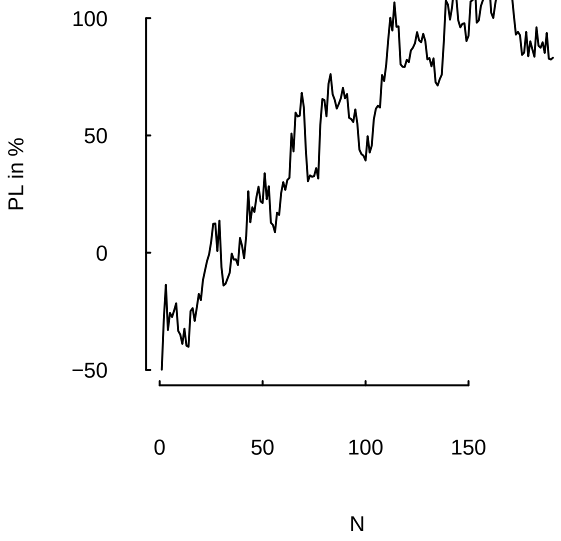
Grid search example

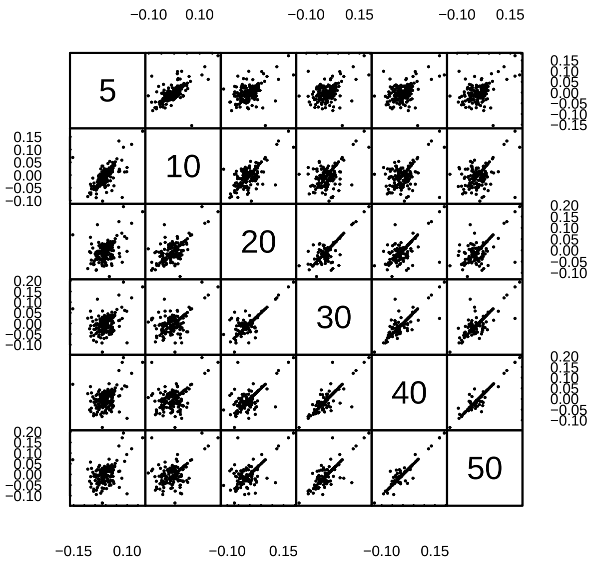
a very simple trading rule

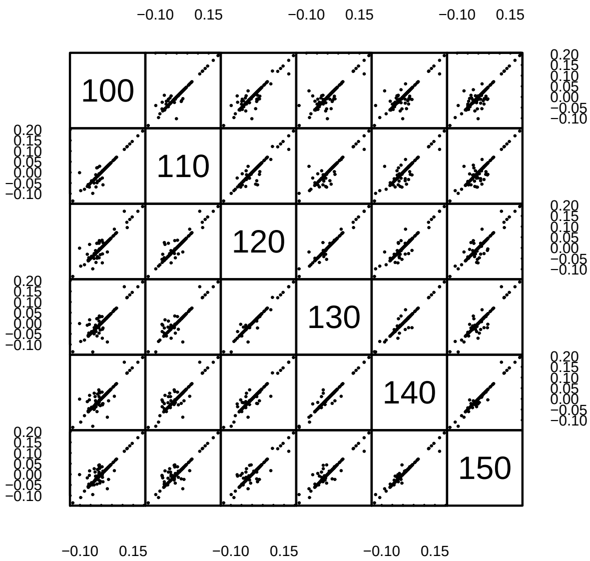
- compute N-day moving average M of price P
- if $M \geq P \rightarrow$ sell
- if $M < P \rightarrow$ buy

test data: DAX, daily close 2005-01-01 – 2011-08-12

Grid search example







How to solve a model?

$$\min_x \quad f(x, \text{data}) \quad \text{subject to constraints}$$

strategy 3: constructive methods

example: travelling salesman problem

example: low-volatility portfolios example: low-volatility portfolios

sort cross-section of stocks by volatility and invest equal-weight in n stocks with lowest volatility

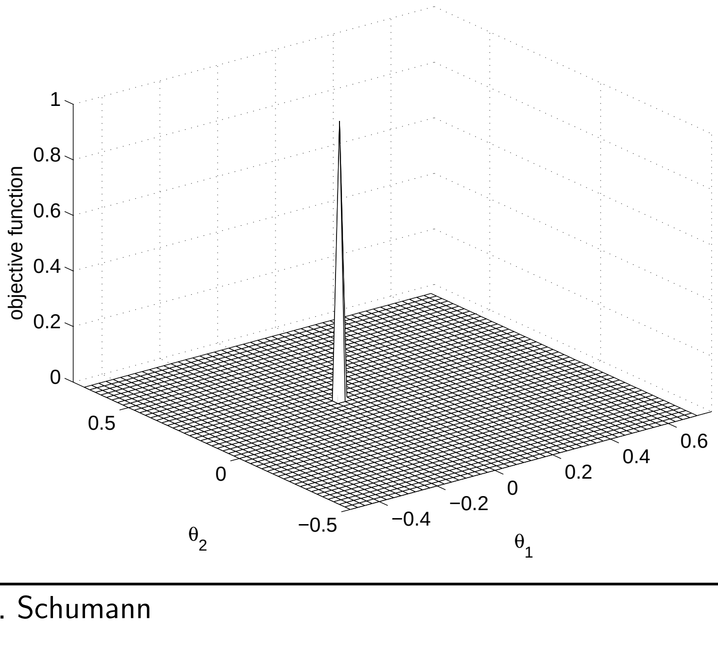
How to solve a model?

$$\min_x \quad f(x, \text{data}) \quad \text{subject to constraints}$$

strategy 4: iterative improvement

start with some solution $x_0 \rightarrow$ improve it gradually

how/when does it work?



How to solve a model?

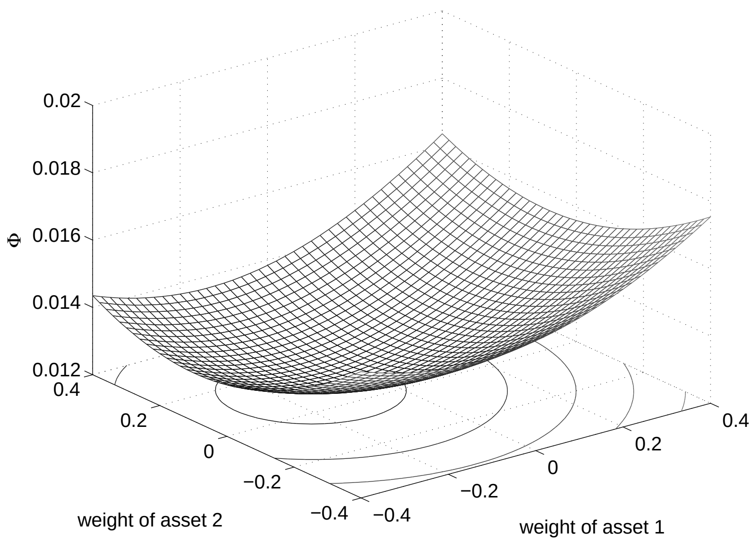
$$\min_x \quad f(x, \text{data}) \quad \text{subject to constraints}$$

strategy 4: iterative improvement

example: minimum variance for a portfolio of three assets

$f = x'\Sigma x$

- x portfolio weights
- Σ forecast of variance–covariance matrix



E. Schumann 26

How to solve a model?

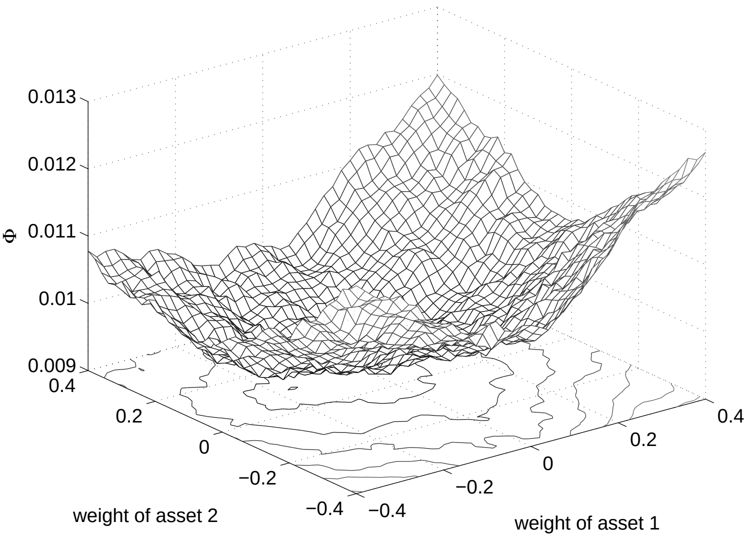
$$\min_x \quad f(x, \text{data}) \quad \text{subject to constraints}$$

strategy 4: iterative improvement

example: Value-at-Risk for a portfolio of three assets

$f = Q(r)$

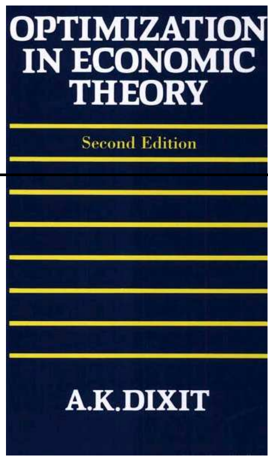
- R return scenarios
- $r = Rx$ portfolio returns in scenarios



Methods

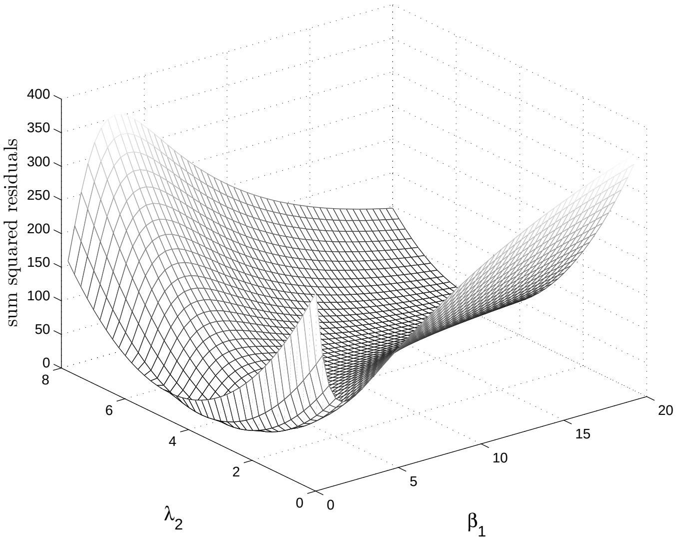
standard methods require 'nice' model

- smooth functions
- derivatives
- no multiple local optima



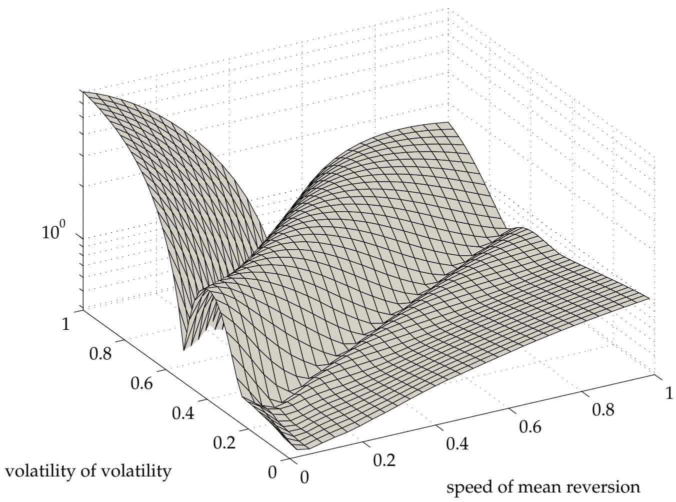
Examples

yield curve models



Examples

option pricing



Examples

robust/resistant regression

$$y = \underbrace{\begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix}}_{\theta} + \epsilon$$

$$r = y - X\hat{\theta}$$

good fit means to make r 'small'

Examples

Least (Mean of) Squares (LS)

$$\hat{\theta}_{\mathrm{LS}} = \underset{\theta}{\operatorname{argmin}} \; \frac{1}{n_{\mathrm{O}}} \sum_{i=1}^{n_{\mathrm{O}}} r_i^2$$

Least Quantile of Squares (LQS)

$$\hat{\theta}_{\mathrm{LQS}} = \underset{\theta}{\operatorname{argmin}} \; \mathcal{Q}_q(r_i^2)$$

with

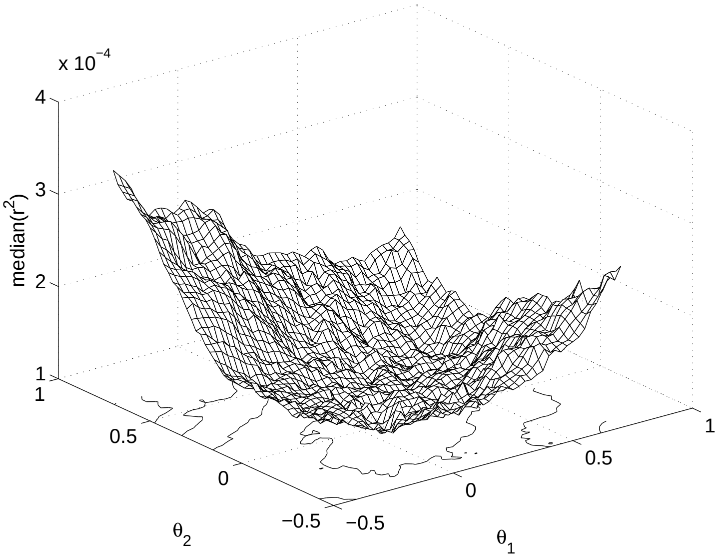
$$\mathcal{Q}_q = \mathrm{CDF}^{-1}(q) = \min\{r^2 \mid \mathrm{CDF}(r) \geq q\} \qquad q \in \{0\% \ldots 100\%\}$$

Least Trimmed Squares (LTS)

$$\hat{\theta}_{\mathrm{LTS}} = \underset{\theta}{\operatorname{argmin}} \; \frac{1}{k} \sum_{i=1}^{k} r_{[i]}^2$$

Examples

Least Median of Squares



E. Schumann 33

What makes these models hard to solve

continuous models

- multiple minima
- noise
- jumps: discontinuities, no derivatives

combinatorial models

- number of possibilities!

  example: choose 20 stocks out of 500 that best fulfil some criterion. Assuming 1 000 000 portfolios can be listed and evaluated per second, the whole process takes 8×10^{21} years.

E. Schumann 34

Finding a peak

a mountainous region, fog → aim: to find the peak!

help: a GPS device that measures position and altitude

E. Schumann 35

Finding a peak

- move
- uphill is good
- sometimes you need to go downhill
- tradeoff time/altitude
- it would help if you could walk very fast

Two approaches

approach 1:
(re)formulate the model until solvable with standard methods

examples: portfolio optimisation → mean–variance, Expected Shortfall

downside: very problem-specific, not flexible (constraints!), not always possible

approach 2:
use a heuristic

examples: 'solve' any model

downside: no exact solution (at least you cannot prove it)

Heuristics

- used in many fields: mathematics, psychology/judgement and decision making, computer science/artificial intelligence, software engineering. . .
  - 'heuristics and biases'
  - fast and frugal
- associated with optimisation, rules of thumb, search
- optimality cannot be proved

Off-topic: recognition heuristic

when choosing between two objects, choose the one you recognise

(the recognized object is 'better')

example:
Which city has a larger population: San Diego or San Antonio?

Heuristics

. . . in the sense of numerical optimisation techniques

- inspired by nature and physical processes
- practical because of today's computing power
- 'good' stochastic approximation of optimum ('good': solution quality/computing time)
- robust to changes to the given model and to changes in the parameter settings of the heuristic ([changes in] solution quality/computing time)
- easy and simple
- not subjective

Iterative improvement

generate initial solution x^c
evaluate $f(x^c)$
while stopping condition not met **do**
    create new solution $x^n \in N(x^c)$
    evaluate $f(x^n)$
    if $A(x^n, ...)$ **then** $x^c = x^n$
end while
return x^c

x a solution
f objective function (goal function, fitness function, . . .)
N neighbourhood
A acceptance (or selection)
stopping rule

Optimisation heuristics

given: optimisation model min $f(x)$ and some solution x

'rule':

- simple
- change $x \rightarrow$ on average improve $f(x)$

$\rightarrow$ apply rule many times over (thus computationally intensive) so that *on average/in the long run* the solution is improved

guidelines for rule (Schumann and Ardia, 2011)

- don't be greedy
- trust your luck

Heuristics

good:
all heuristics are based on just a few principles

bad:
all heuristics are based on just a few principles

good:

- precisely-described algorithms exist ('canonical versions')
- algorithms robust for different settings
- heal bad settings through more computing time
- judge for yourself: run experiments – and stop when satisfied

E. Schumann 43

Classification

- single solution – several solutions (population-based)
  - single solution: Local Search
  - multiple solutions: Genetic Algorithms
- generation of new solutions: step size, use of past information, . . .
- acceptance of new solutions

E. Schumann 44

Heuristics: Examples

- Local Search/Threshold Accepting/Simulated Annealing
- Tabu Search
- Genetic Algorithms
- Differential Evolution
- Particle Swarm

E. Schumann 45

Local Search

not a simple rule: a gradient search

pick initial solution x^c
evaluate quality of solution $f(x^c)$
while stopping condition not met **do**
    create new solution $x^n = x^c - \alpha \nabla f(x^c)$
    evaluate quality of solution $f(x^n)$
    accept new solution if $f(x^n) < f(x^c)$
end while
return x^c

assumptions: gradient exists/meaningful

Local Search

a simple rule: a Local Search

generate initial solution x^c
evaluate quality of solution $f(x^c)$
while stopping condition not met **do**
    create new solution $x^n = x^c + \epsilon$
    evaluate quality of solution $f(x^n)$
    accept new solution if $f(x^n) \leq f(x^c)$
end while
return x^c

no gradient needed

Local Search

a simple rule: a Local Search

generate initial solution x^c
evaluate quality of solution $f(x^c)$
while stopping condition not met **do**
    create new solution $x^n = x^c + \epsilon$
    evaluate quality of solution $f(x^n)$
    accept new solution if $f(x^n) \leq f(x^c)$
end while
return x^c

no gradient needed

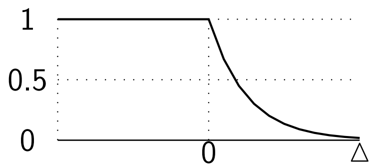
Characteristics of Local Search

- still ‘greedy’: will never move uphill
- stochastic: repeated restarts from same starting point may result in different solutions; there is a chance to avoid local minima

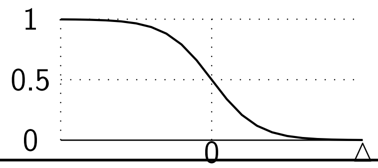
Simulated Annealing

- inspired by process of annealing (Kirkpatrick et al., 1983)
- as in Local Search: better solutions are always accepted
- different from Local Search: worse solutions are also accepted, but only with probability

Metropolis function



Barker criterion

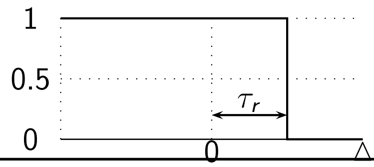


Simulated Annealing

set $R_{\max}$ and T
randomly generate current solution x^c
for $r = 1$ to $R_{\max}$ **do**
  while stopping criteria not met **do**
    generate $x^n \in \mathcal{N}(x^c)$ (neighbor to current solution)
    compute $\triangle = f(x^n) - f(x^c)$ and generate u (uniform random variable)
    if $(\triangle < 0)$ or $(e^{-\triangle/T} > u)$ **then** $x^c = x^n$
  end while
  reduce T
end for
return best solution

Threshold Accepting

- deterministic variant of Simulated Annealing (Dueck and Scheuer, 1990)
- as in Local Search: better solutions are always accepted
- different from Local Search: worse solutions are also accepted, but only *if not too much worse*
- first heuristic applied to portfolio optimisation (Dueck and Winker, 1992)



Threshold Accepting

set n_{rounds} and n_{steps}
compute threshold sequence τ_r
randomly generate current solution x^c
for $r = 1 : n_{\text{rounds}}$ **do**
  for $i = 1 : n_{\text{steps}}$ **do**
    generate $x^n \in N(x^c)$ and compute $\Delta = f(x^n) - f(x^c)$
    if $\Delta < \tau_r$ **then** $x^c = x^n$
  end for
end for
return best solution

τ_r thresholds

Example

`LS_TA.R`

Tabu Search

- developed for combinatorial models with finite neighbours (Glover, 1986)
- allows uphill moves, but no random elements
- in most implementations greedy in neighbourhood search

generate current solution x^c
while stopping criteria not met **do**
  compute all neighbours $V = \{x | x \in \mathcal{N}(x^c)\}$
  select $x^n = \min(V)$
  if $f(x^n) < f(x^c)$ **then** $x^c = x^n$
end while
return x^c

Tabu Search

```
initialize tabu list T = ∅
generate current solution x^c
while stopping criteria not met do
    compute V = {x | x ∈ N(x^c)} \ T
    select x^n = min(V)
    x^c = x^n   and   T = T ∪ x^n
    update memory
end while
return best solution
```


Iterative improvement

```
generate initial solution  x^c
evaluate f(x^c)
while stopping condition not met do
    create new solution  x^n ∈ N(x^c)
    evaluate f(x^n)
    if  A(x^n, ...)  then  x^c = x^n
end while
return x^c
```

x a solution
f objective function (goal function, fitness function, . . .)
N neighbourhood
A acceptance (or selection)
stopping rule

Decisions

- representation: encoding solutions x and data
- evaluation: objective function $f(\cdot)$
- how to change/evolve solutions: $N(\cdot)$
  - variation, but correlation
- how to accept solutions: $A(\cdot)$
- when to stop?
  - convergence: trade-off resources/quality

Representing solutions

model representation: a vector, a matrix, a graph, ...

internal representation: a vector, ...

LS/TA example: regression model

$$y = X\beta + \epsilon$$

- representation: numeric vector β
- objective function: $f(y - X\beta)$
- change solutions: randomly pick element in β, add noise
- how to accept solutions: when better; when not too much worse
- stopping criterion: number of iterations

LS/TA example: accounting

representation: a matrix (restriction on row sums, column sums)

	account 1	account 2	account 3
n_i	20 000	10 000	2000
1000 @ 100.15	800	100	100
300 @ 100.01	100	50	150
1200 @ 100.31	80	1100	20
...			

change solutions: randomly pick two columns, two rows

$$\begin{array}{ccc} +1 & \dots & -1 \\ \dots & \dots & \dots \\ -1 & \dots & +1 \end{array}$$

Representing solutions

example: large number of funds, pick a small number

vector
`0 0 0 0 0.21 0 0 0 0 ... 0 0.14 0 0 0 ...`

list

- asset indices (`5, 43, 98, 105, ...`)
- asset weights (`0.21, 0.14, 0.15, 0.20, ...`)

store computations with solution

E. Schumann 62

Objective functions

called many times → try to make it fast

→ profile code, improve code

example: compute moving average (MA)

given: time series y_t for $t = 1, \dots, N$

MA of order O:

$$M_t = \frac{\sum_{o=1}^{O} y_{t-o+1}}{O}$$

E. Schumann 63

Objective functions

called many times → try to make it fast

```
> N <- 1000L      ## length of series
> O <- 50L        ## order
> trials <- 500L  ## run 500 times to measure time
> y <- rnorm(N)   ## random series
```

E. Schumann 64

heuristics: generic iterative methods

called many times → try to make it fast

```
> MA1 <- numeric(N)
> system.time(
      for (i in seq_len(trials))
          for (t in O:N)
              MA1[t] <- mean(y[(t-O+1L):t])
  )
```

```
   user  system elapsed
  3.560   0.000   3.556
```


Objective functions

called many times → try to make it fast

$$M_t = \frac{y_t}{O} + \frac{y_{t-1}}{O} + \frac{y_{t-2}}{O} + \cdots + \frac{y_{t-O+1}}{O}$$

$$M_{t+1} = \frac{y_{t+1}}{O} + \underbrace{\frac{y_t}{O} + \frac{y_{t-1}}{O} + \cdots + \frac{y_{t-O+1}}{O} + \frac{y_{t-O+1}}{O}}_{M_t} - \frac{y_{t-O+1}}{O}$$

$$M_{t+1} = M_t + \frac{y_{t+1}}{O} - \frac{y_{t-O+1}}{O}$$

Objective functions

called many times → try to make it fast

```
> MA3 <- numeric(N)
> system.time(
      for(i in seq_len(trials)) {
          MA3[O] <- sum(y[seq_len(O)])/O
          for(t in (O+1L):N)
              MA3[t] <- MA3[t-1L]+y[t]/O - y[t-O]/O
      }
  )
```

```
   user  system elapsed
  0.996   0.000   0.996
```

E. Schumann 67

Objective functions

called many times → try to make it fast

```
> MA4 <- numeric(N)
> system.time(
      for (i in seq_len(trials))
          MA4 <- filter(y, rep(1/O, O), sides = 1L)
  )
```

```
   user  system elapsed
  0.264   0.000   0.262
```

E. Schumann 68

Objective functions

called many times → try to make it fast

```
> MA5 <- numeric(N)
> system.time(
      for(i in seq_len(trials)) {
          MA5 <- cumsum(y)/O
          MA5[O:N] <- MA5[O:N] - c(0, MA5[1:(N-O)])
      }
  )
```

```
   user  system elapsed
  0.040   0.000   0.039
```

→ speedup of about 150

E. Schumann 69

Objective functions

called many times → try to make it fast

- check what is available
- experiment

E. Schumann 70

Objective functions

called many times → try to make it fast

→ 'make it run'
→ 'make it run *correctly*'
→ 'make it run *correctly & efficiently*'

time to . . .

- . . . write code
- . . . test
- . . . maintain/change
- . . . debug/handle errors
- . . . run

many other dimensions of 'good' code

Changing solutions

- functions of current solution(s)
- random elements
- meaningful variation, but quality should be correlated

Iterative improvement

```
generate initial solution  x^c
evaluate f(x^c)
while stopping condition not met do
    create new solution  x^n ∈ N(x^c)
    evaluate f(x^n)
    if  A(x^n, ...)  then  x^c = x^n
end while
return x^c
```

x a solution
f objective function (goal function, fitness function, . . .)
N neighbourhood
A acceptance (or selection)
stopping rule

Multiple solutions

generate initial solutions X^c
evaluate $f(X^c)$
while stopping condition not met **do**
  create new solutions $X^n \in N(X^c)$
  evaluate $f(X^n)$
  if $A(X^n, \ldots)$ **then** $X^c = X^n$
end while
return X^c

x a solution
f objective function (goal function, fitness function, . . .)
N neighbourhood
A acceptance (or selection)
stopping rule

Genetic Algorithm

Evolutionary algorithms: developed by various authors

- David Fogel (Evolutionary programming)
- John Holland (Genetic Algorithms)
- Ingo Rechenberg/Hans-Paul Schwefel (Evolutionary strategies)

(and probably others)

- build on evolutionary principles: random variation and natural selection

Genetic Algorithm

- representation: binary (solutions coded as 0s and 1s)
- evaluation: objective function $f(\cdot)$ ('fitness function')
- change/evolve solutions: crossover and mutation
- how to accept solutions: sort, tournament, pairwise comparison, . . .
- stopping criterion: number of iterations ('generations')

Genetic Algorithm

generate initial population P of solutions
while stopping criteria not met **do**
  select $P' \subset P$ (mating pool), initialise $P'' = \emptyset$ (set of children)
  for $i = 1$ to n **do**
    select individuals x^{a} and x^{b} at random from P'
    apply crossover to x^{a} and x^{b} to produce x^{child}
    randomly mutate produced child x^{child}
    $P'' = P'' \cup x^{\mathrm{child}}$
  end for
  $P = \mathrm{survive}(P', P'')$
end while

E. Schumann 77

Genetic Algorithm

0 1 1 1 0 0 0 1

two parents

0 1 1 1 0 0 0 1
1 1 0 0 1 1 1 0

. . . and children

0 1 1 1 1 1 1 0
1 1 0 0 0 0 0 1

crossover

original solution

0 1 1 1 0 0 0 1

. . . and mutant

0 1 1 1 0 1 0 1

mutation

E. Schumann 78

GA example: variable selection

- representation: logical vector – `TRUE` (variable included), `FALSE` (variable excluded)
- evaluation: evaluate specified model (eg, information criteria)
- change/evolve solutions: mix existing solutions, randomly switch elements
- how to accept solutions: sort, tournament, pairwise comparison, . . .
- stopping criterion: number of iterations (‘generations’)

E. Schumann 79

Differential Evolution

- introduced in Storn and Price (1997); for continuous objective functions
- representation: numeric vectors → matrices
- evaluation: objective function $f(\cdot)$
- change/evolve solutions: specified rules
- how to accept solutions: specified rules
- stopping criterion: number of iterations

Differential Evolution

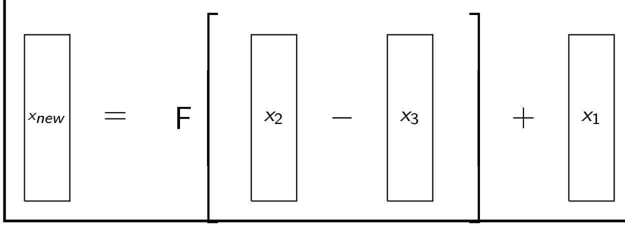
set n_P, n_G, F and CR
randomly generate initial population $P^{(1)}_{j,i}$, $j = 1, \ldots, d$, $i = 1, \ldots, n_P$
for $k = 1$ to n_G **do**
    $P^{(0)} = P^{(1)}$
    for $i = 1$ to n_P **do**
        randomly generate r_1, r_2, $r_3 \in \{1, \ldots, n_P\}$, $r_1 \neq r_2 \neq r_3 \neq i$
        compute $P^{(v)}_{\cdot,i} = P^{(0)}_{\cdot,r_1} + \texttt{F} \times (P^{(0)}_{\cdot,r_2} - P^{(0)}_{\cdot,r_3})$
        for $j = 1$ to d **do**
            if rand < CR then $P^{(u)}_{j,i} = P^{(v)}_{j,i}$ else $P^{(u)}_{j,i} = P^{(0)}_{j,i}$
        end for
        if $f(P^{(u)}_{\cdot,i}) < f(P^{(0)}_{\cdot,i})$ then $P^{(1)}_{\cdot,i} = P^{(u)}_{\cdot,i}$ else $P^{(1)}_{\cdot,i} = P^{(0)}_{\cdot,i}$
    end for
end for
return best solution

Differential Evolution

in every iteration, for every member of the population x_i:

- randomly pick three other solutions x_1, x_2, x_3
- compute $x_{new} = \text{F}(x_2 - x_3) + x_1$ for some F
- pointwise crossover between x_{new} and x_i with probability CR

when new solution is better, replace x_i



Particle Swarm

- introduced in Eberhart and Kennedy (1995)
- for continuous functions
- representation: numeric vectors → matrices
- evaluation: objective function $f(\cdot)$
- change/evolve solutions: specified rules
- how to accept solutions: specified rules
- stopping criterion: number of iterations

implementation

set n_P, n_G and c_1, c_2
randomly generate initial population $P_i^{(0)}$ and velocity $v_i^{(0)}$, $i = 1, \dots, n_\mathrm{P}$
evaluate objective function $F_i = f(P_i^{(0)})$, $i = 1, \dots, n_\mathrm{P}$
$Pbest = P^{(0)}$, $Fbest = F$, $Gbest = \min_i(F_i)$, $gbest = \mathrm{argmin}_i(F_i)$
for $k = 1$ to n_G **do**
    for $i = 1$ to n_P **do**
        $\triangle v_i = c_1\, u_1\, (Pbest_i - P_i^{(k-1)}) + c_2\, u_2\, (Pbest_{gbest} - P_i^{(k-1)})$
        $v_i^{(k)} = v_i^{(k-1)} + \triangle v_i$
        $P_i^{(k)} = P_i^{(k-1)} + v_i^{(k)}$
    end for
    evaluate objective function $F_i = f(P_i^{(k)})$, $i = 1, \dots, n_\mathrm{P}$
    for $i = 1$ to n_P **do**
        if $F_i < Fbest_i$ **then** $Pbest_i = P_i^{(k)}$ and $Fbest_i = F_i$
        if $F_i < Gbest$ **then** $Gbest = F_i$ and $gbest = i$
    end for
end for
return best solution

Particle Swarm

- current solutions
- the best solution for every member of population
- velocities

→ solution is a position in vector space; velocity is the direction of movement (ie, change)

in every iteration, for every member of the population x_i: velocity is updated toward best personal solution and overall best solution

$$\begin{aligned} v_{\text{new}} = & v_{\text{old}} + \\ & \text{random} \cdot c_1(\text{best solution - current solution}) + \\ & \text{random} \cdot c_2(\text{overall best solution - current solution}) \end{aligned}$$

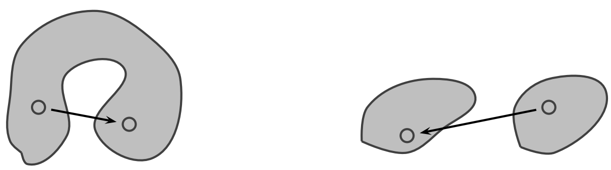
if new solution is better, update the best solution

Example

`DE_PS.R`

Constraints

- discard infeasible solutions
- always construct feasible solutions
  - example: budget constraint
- repair solutions
  - example: budget constraint, cardinality constraints
- penalise infeasible solutions



How to choose a method?

- natural representation of problem: TA, GA for combinatorial models; DE for continuous models
- simple is better: do not mix methods

Stochastics

result of optimisation: random variable ϕ with unknown distribution $\mathcal{D}$

assumption: change seed for each run
easy to sample from $\mathcal{D}$: run restarts $r = 1, \dots, R \rightarrow$ collect ϕ_r

E. Schumann 89

Stochastics

Value-at-Risk for a portfolio of financial assets

$f = Q(r)$

R return scenarios $r = Rx$ portfolio returns in scenarios

dataset: 20 assets, 100 weekly return scenarios
restrictions: long-only, maximum weight 10%

E. Schumann 90

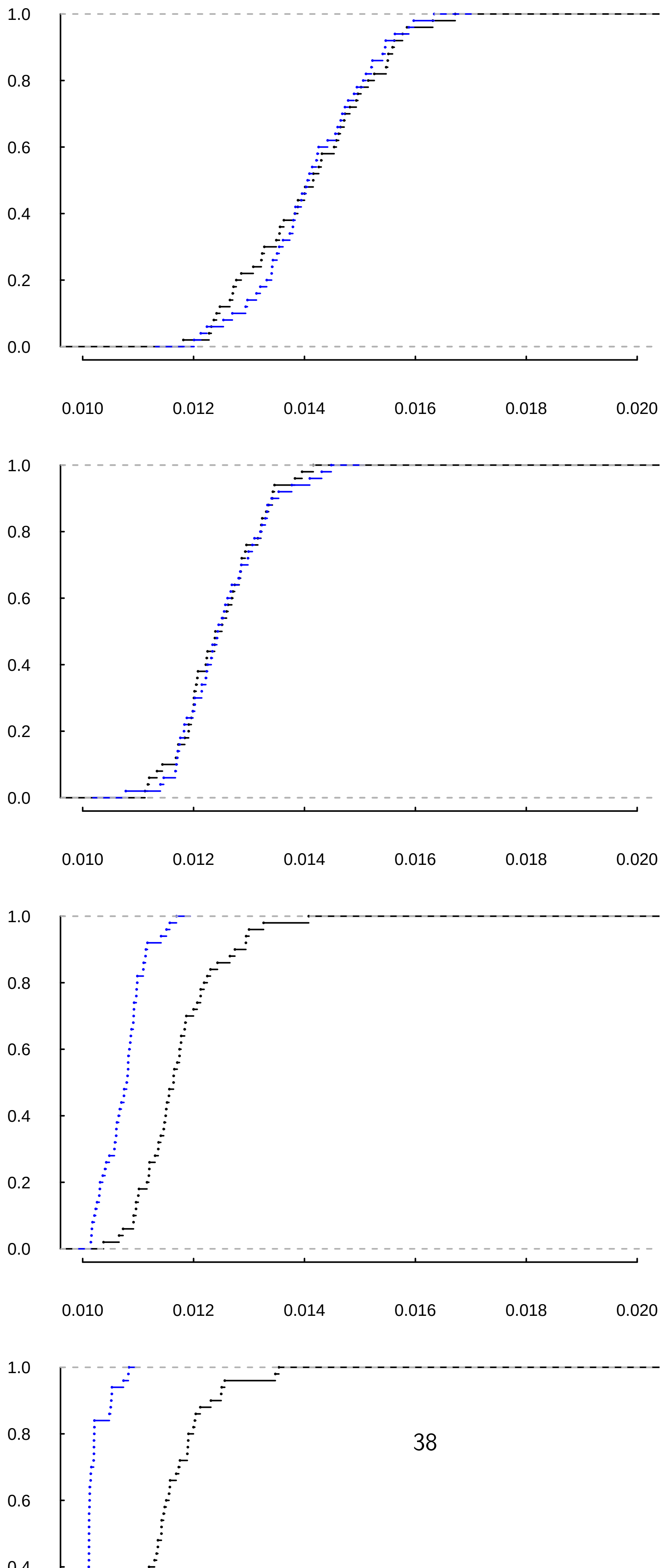
Stochastics

```
TAopt(OF, algo = list(), ...)
```

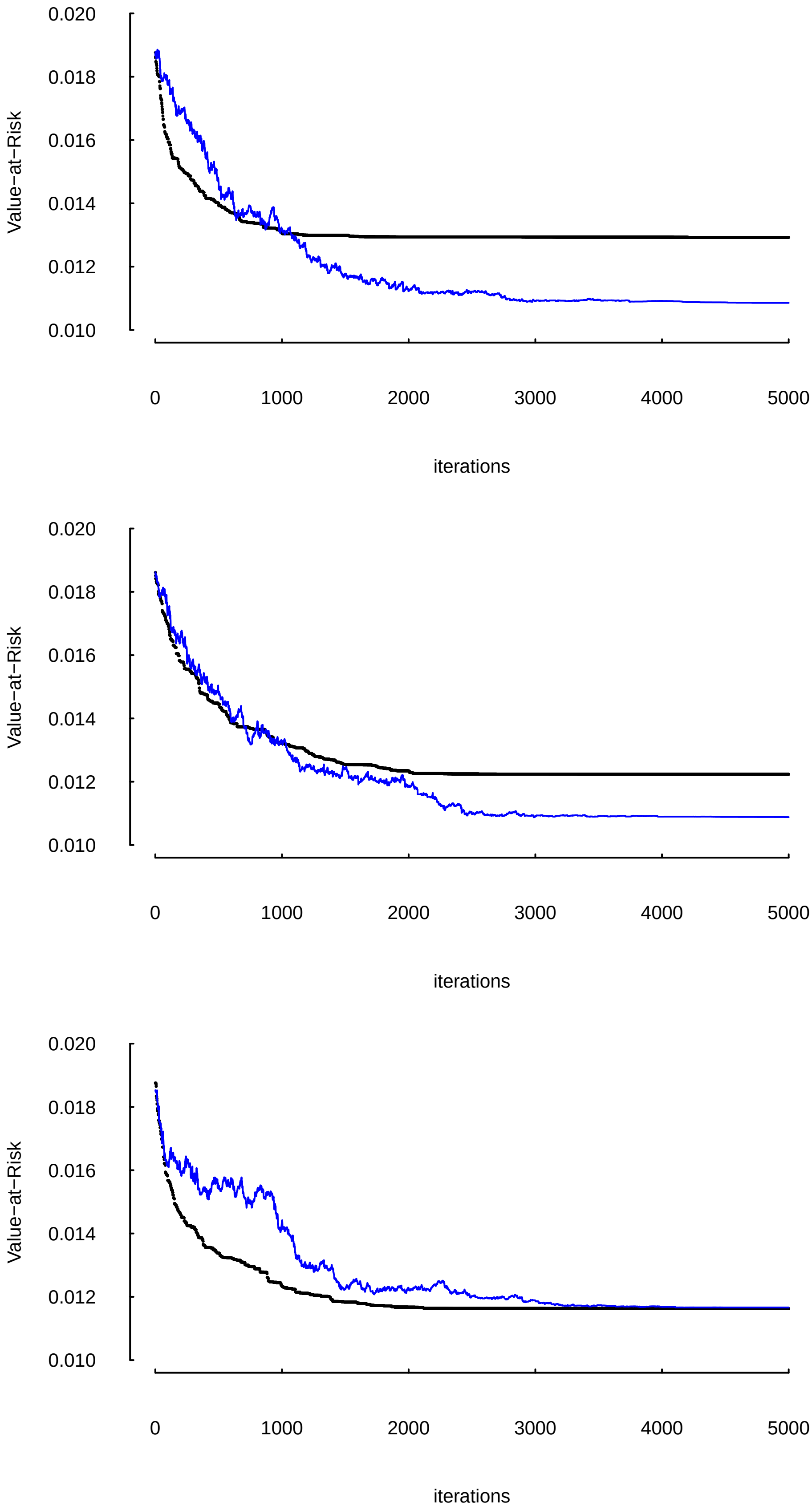
```
> TAopt(OF, algo, data)$OFvalue
[1] 0.010375

> TAopt(OF, algo, data)$OFvalue
[1] 0.010963

> TAopt(OF, algo, data)$OFvalue
[1] 0.010163
```

E. Schumann 91

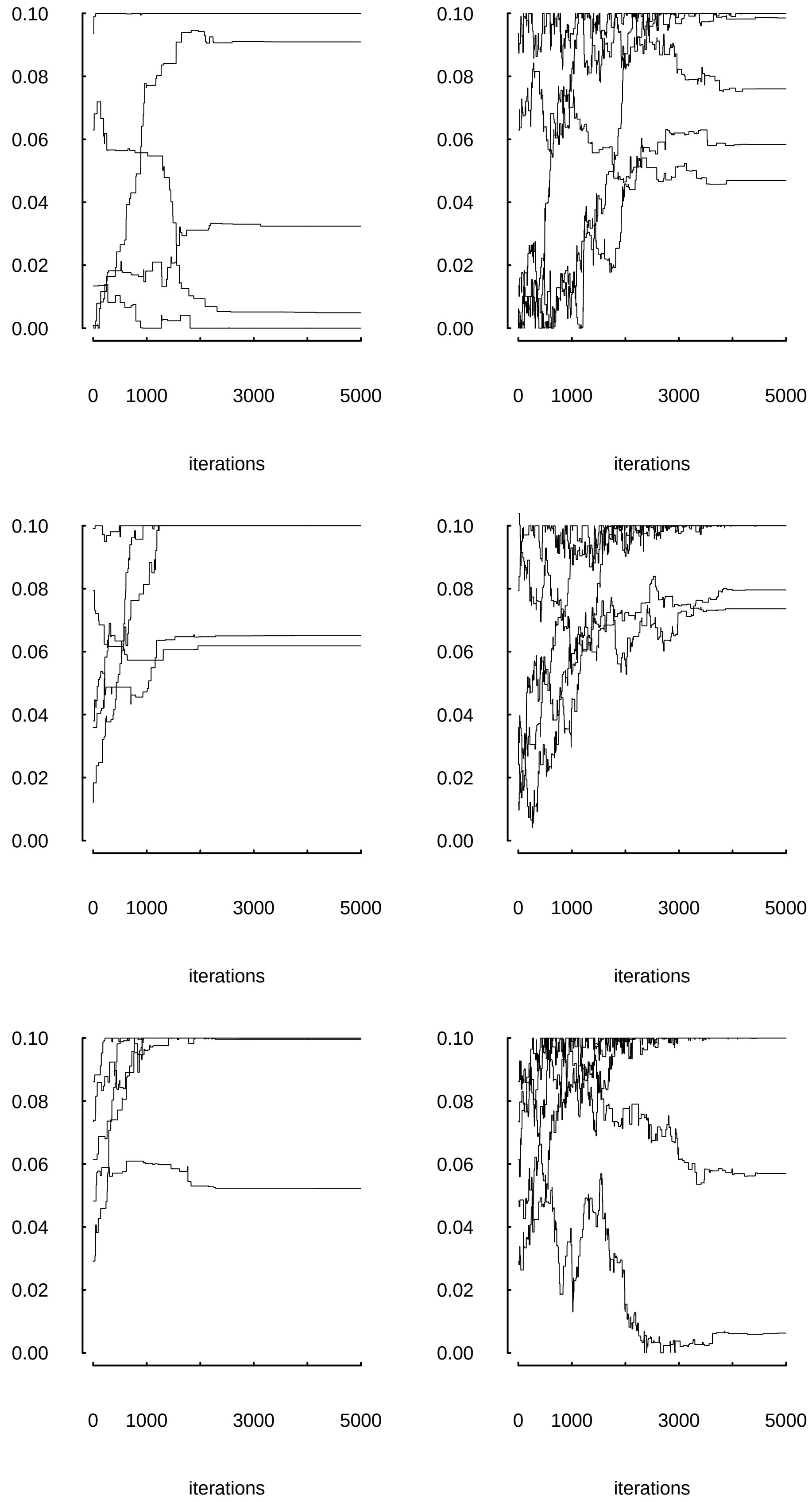
Stochastics (TA in blue, LS in black)



Stochastics (TA in blue, LS in black)



Weights with LS (left) and TA (right)



Portfolio selection

select assets according to investor's needs and goals

- assets, data availability
- investment process (frequency of rebalancing, risk characteristics, stop loss, ...)
- forecasts
- scenarios for risk management
- how to evaluate portfolios?

Portfolio selection: textbook case

Markowitz (1952)

one period: what to buy and sell (not when) one period: what to buy and sell (not when)

investor's needs and goals: high return is good; return variability is bad

GOTO statement

→ mean–variance

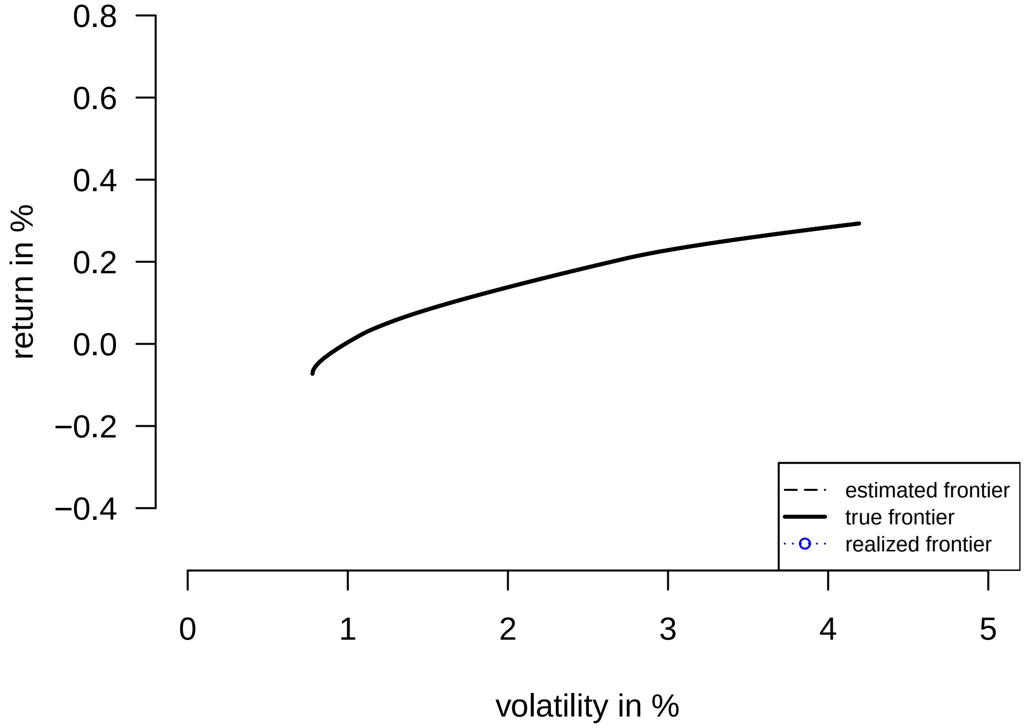
The efficient frontier

tracing out the frontier
$\mu' w - \gamma w' \Sigma w$

dataset `fundData`

- 25 assets
- 100 scenarios

The efficient frontier

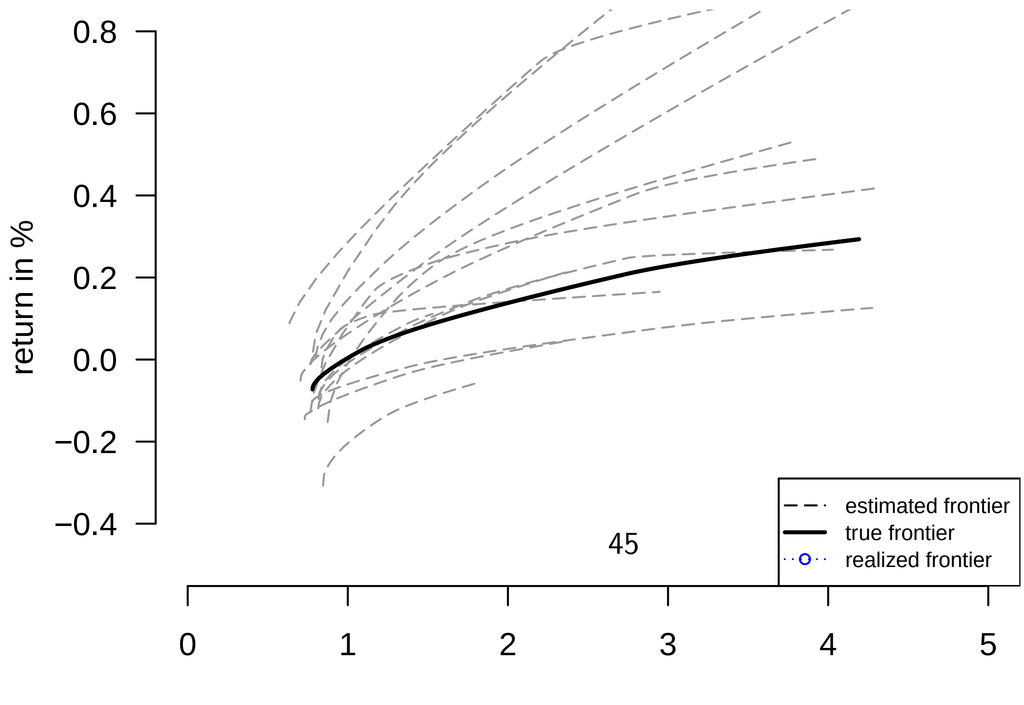


tracing out the frontier
$\mu' w - \gamma w' \Sigma w$

with long-only constraint

1: set true expected returns m_* and a true variance–covariance matrix Σ_*
2: simulate data with these parameters
3: estimate parameters m and Σ
4: compute optimal portfolio weights w

true frontier	$-w_*' \Sigma_* w_* + \gamma m_*' w_*$
'expected' frontier	$-w' \Sigma w + \gamma w' m$
realized frontier	$-w' \Sigma_* w + \gamma w' m_*$



Managing mean–variance

Markowitz (1952):

> "The process of selecting a portfolio may be divided into two stages. The first stage starts with observation and experience and ends with beliefs about the future performances of available securities. The second stage starts with the relevant beliefs about future performances and ends with the choice of portfolio. This paper is concerned with the second stage."

Managing mean–variance

Trouble stems from expected returns and supposed trade-off between risk and reward.

What to do:

- don't use it
- massage inputs
- massage outputs

Portfolio selection

goal: allocate wealth

- modelling
- data
- optimisation

Portfolio selection: modelling and data

goal: allocate wealth

- assets, data availability
- investment process (frequency of rebalancing, risk characteristics, stop loss, . . .)
- forecasts
- scenarios for risk management
- how to evaluate portfolios?

Shortcomings of mean–variance

Markowitz: allocate wealth among n_A assets; select portfolio that is mean–variance efficient $\rightarrow$ find weights w that maximise

$$w'\mu - \gamma w'\Sigma w$$

1: collect historical data $R_{\ (T \times n_\text{A})}$
2: estimate $\hat{\mu} = \frac{1}{T}\iota' R$
3: estimate $\hat{\Sigma} = \frac{1}{T}R'\big(I - \frac{1}{T}\iota\iota'\big)R$
4: find weights w that maximise $w'\hat{\mu} - \gamma w'\hat{\Sigma} w$

estimation problems: eg, Jobson and Korkie (1980) and many others
theoretical concerns: Artzner et al. (1999) and many others

- test alternative risk measures & objective functions empirically
- test alternative forecasting and scenario generation methods

there is more: dynamic models, alternative assets (derivatives, bonds), ...

Alternative objective functions: risk & reward

$$\underbrace{w'\mu}_{\text{reward}} - \underbrace{\gamma w'\Sigma w}_{\text{risk}}$$

replace reward and risk by alternative functions

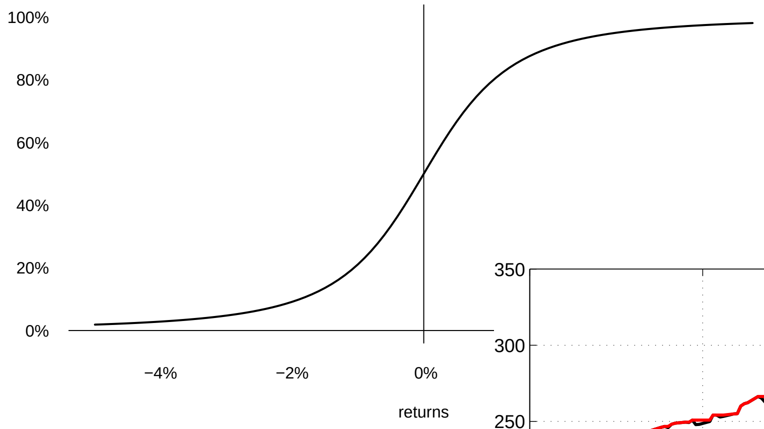
$$\min_w \frac{\text{risk}}{\text{reward}}$$

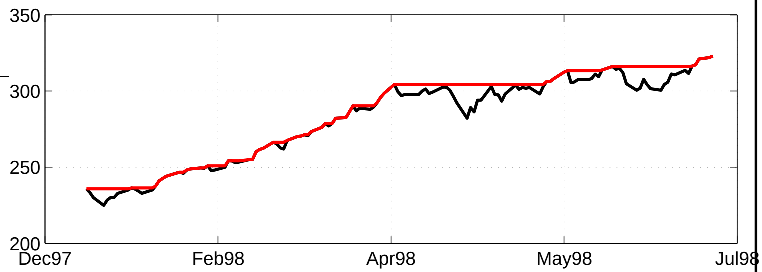
$$\min_w \frac{\text{risk}}{\text{reward}} = \Phi$$

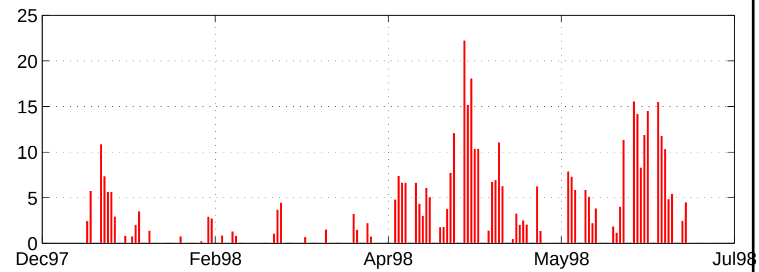
$$\min_w f(\text{risk}_1, \text{risk}_2, \dots, -\text{reward}_1, -\text{reward}_2, \dots) = \Phi$$

Alternative objective functions: building blocks

based on distribution of portfolio returns







- moments (variance, skewness, . . .)
- conditional moments (expected shor
- quantiles (VaR, . . .), corresponding

based on trajectory of portfolio wealth

- drawdown, time under water, . . .

objective function: do as you please

Alternative objective functions: partial moments

capture non-symmetric returns Fishburn, 1977

$$r = \underbrace{r_d}_{\text{desired return}} + \underbrace{(r - r_d)^+}_{\text{upside}} - \underbrace{(r_d - r)^+}_{\text{downside}}$$

$$\mathcal{P}_\gamma^+(r_d) = \frac{1}{T} \sum_{r > r_d} (r - r_d)^\gamma$$

$$\mathcal{P}_\gamma^-(r_d) = \frac{1}{T} \sum_{r < r_d} (r_d - r)^\gamma$$

example: semi-variance

E. Schumann 107

Alternative objective functions: conditional moments

capture non-symmetric returns

$$r = \underbrace{r_d}_{\text{desired return}} + \underbrace{(r - r_d)^+}_{\text{upside}} - \underbrace{(r_d - r)^+}_{\text{downside}}$$

$$\mathcal{C}_\gamma^+(r_d) = \frac{1}{\#\{r > r_d\}} \sum_{r > r_d} (r - r_d)^\gamma$$

$$\mathcal{C}_\gamma^-(r_d) = \frac{1}{\#\{r < r_d\}} \sum_{r < r_d} (r_d - r)^\gamma$$

example: Expected Shortfall conditional vs partial moments

$$\mathcal{P}_\gamma^+(r_d) = \mathcal{C}_\gamma^+(r_d) \underbrace{\mathcal{P}_0^+(r_d)}_{\pi \text{ of } r > r_d}$$

$$\mathcal{P}_\gamma^-(r_d) = \mathcal{C}_\gamma^-(r_d) \underbrace{\mathcal{P}_0^-(r_d)}_{\pi \text{ of } r < r_d}$$

E. Schumann 108

Alternative objective functions: quantiles

$$\mathcal{Q}_q = \text{CDF}^{-1}(q) = \min\{r \mid \text{CDF}(r) \geq q\}\,,$$

example: VaR

E. Schumann 109

Alternative objective functions: drawdown

drawdown of portfolio value v

$$\text{drawdown}_t = v_t^{\max} - v_t$$

$v_t^{\max} = \max\{v_{t'} \mid t' \in [0, t]\}$

```
> dd <- function(v)      ## absolute drawdown (currency units)
      cummax(v) - v
> dd <- function(v) {   ## percentage drawdown
      maxv <- cummax(v)
      (maxv - v) / maxv
  }
```

E. Schumann 110

Alternative objective functions: examples

reward	risk	
constant	$\mathcal{C}_1^-(\mathcal{Q}_q)$	minimise Expected Shortfall for qth quantile
constant	$\mathcal{Q}_0$	minimise maximum loss
$\frac{1}{n_S}\sum r$	$\sqrt{\mathcal{P}_2^-(r_d)}$	Sortino ratio
$\mathcal{P}_1^+(r_d)$	$\sqrt{\mathcal{P}_2^-(r_d)}$	Upside Potential ratio
$\mathcal{P}_1^+(r_d)$	$\mathcal{P}_1^-(r_d)$	Omega for threshold r_d
$\frac{1}{n_S}\sum r$	$\mathcal{D}_{\max}$	Calmar ratio
$\mathcal{C}_\gamma^+(\mathcal{Q}_p)$	$\mathcal{C}_\delta^-(\mathcal{Q}_q)$	Rachev Generalised ratio for exponents γ and δ

E. Schumann 111

Constraints

required: budget constraint

$$\sum w = 1$$

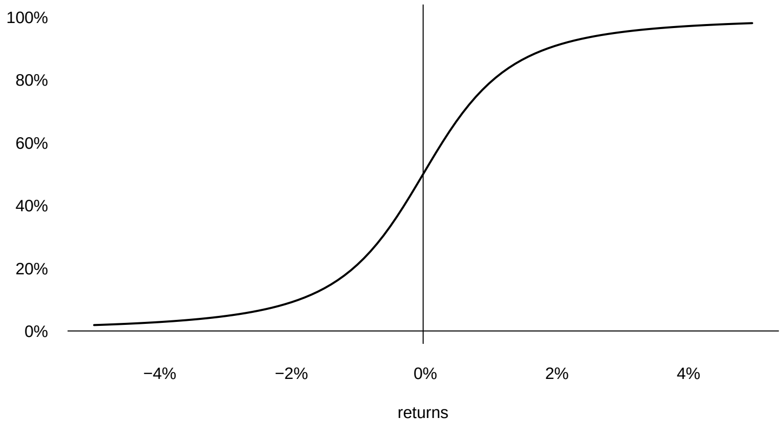
'classical':

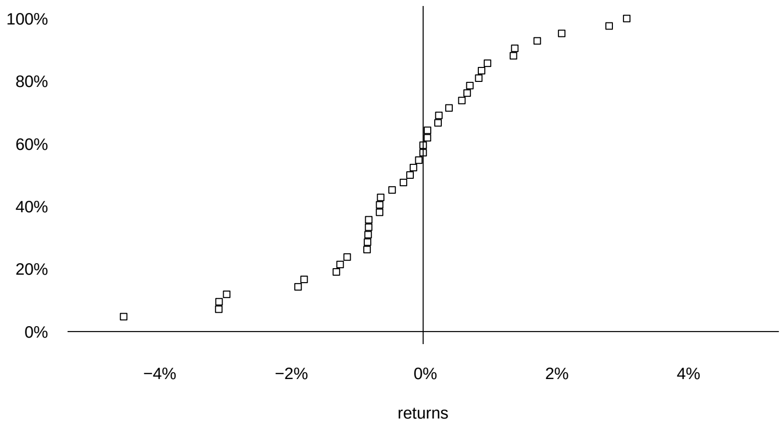
- all w nonnegative
- more general: minimum and maximum holding sizes
- several objectives
- cardinalities, minimum-thresholds
- sectors, factor exposures
- difference to current portfolio
- ...

Optimisation: constraints

- discard infeasible solutions
- always construct feasible solutions
  - example: budget constraint
- repair solutions
  - example: budget constraint, cardinality constraints
- penalise infeasible solutions

Forecasts





empirical distribution of historical portfolio returns
(order statistics $r_{[1]} \leq r_{[2]} \leq \cdots \leq r_{[T]}$)

Scenario generation

- historical returns
- bootstrapping: single observations, blocks ...
- bootstrapping from model residuals

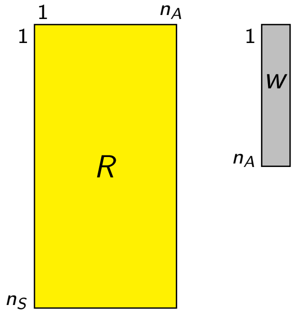
Forecasting (not estimating)

Scenario generation

arbitrage opportunities in data sample

$$\min_w w'\iota \qquad \max_w (Rw)'\iota$$
$$Rw \geq 0 \qquad Rw \geq 0$$
$$w'\iota = 0$$

(see for example Scherer, 2004)



Portfolio optimisation with R

examples

- asset selection
- single-period optimisation

Asset selection: model

given a universe of n_A assets, select K of them according to some criterion

$$\min_w \; w'\Sigma w$$
$$w_j = {}^1\!/_K \quad \text{for } j \in \mathcal{J}$$
$$\#\{\mathcal{J}\} = K$$
$$K_{inf} \leq K \leq K_{sup}$$

Local Search

```
set $n_{steps}$
randomly generate current solution $x^c$
for $i = 1 : n_{steps}$ do
    generate $x^n \in N(x^c)$ and compute $\triangle = \Phi(x^n) - \Phi(x^c)$
    if $\triangle < 0$ then $x^c = x^n$
end for
$x^{sol} = x^c$
```

- representation: encoding solutions x
- evaluation: objective function Φ
- how to evolve solutions? N
- how to accept solutions?
- when to stop?

Encoding solutions

weights `[w 0 0 0 w w 0 0 w 0 0 ... 0 w 0]`

weights `w [1 0 0 0 1 1 0 0 1 0 0 ... 0 1 0]`

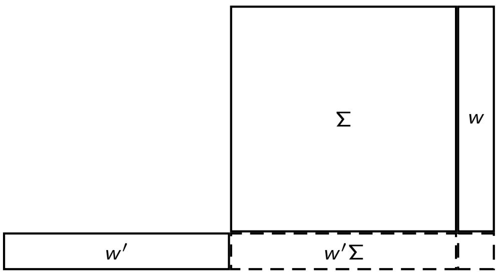
weights `w * x`

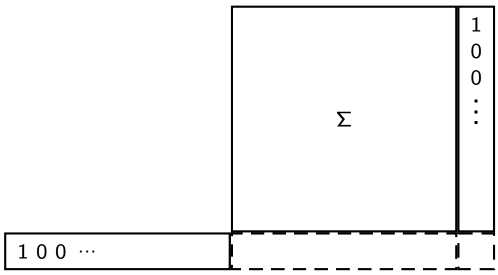
`x` is `logical`; `data` is a list that stores data

```
> OF <- function(x, data) {
      w <- x/sum(x)
      w %*% data$Sigma %*% w
  }
```

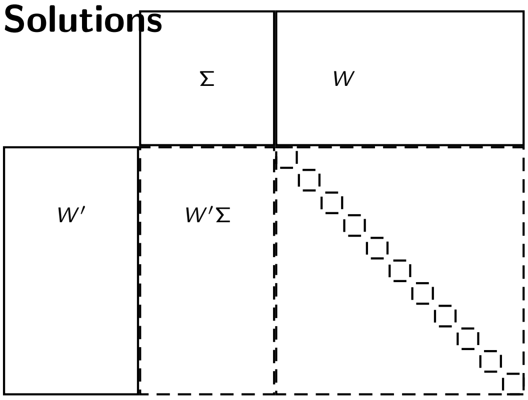
subset: `w[x]` and `Sigma[x,x]`

Solutions





Solutions



$$\mathrm{diag}(W'\Sigma W) \qquad \iota' \underbrace{\overbrace{\Sigma W}^{\text{matrix multiplication}}\ W}_{\text{elementwise multiplication}}$$

Neighbourhood

neighbour `x0` → `x1`

pick any one asset; if not in portfolio, add; else remove
`x` is logical:

```
x[i] <- FALSE      ## asset i is not in portfolio
x[i] <- !x[i]      ## asset i is in portfolio
```

```
> neighbour <- function(x, data) {
      p <- sample.int(data$na, 1L)
      x[p] <- !x[p]
      x
  }
```

neighbourhood

```
> data <- list(na = 5L)
> (x <- logical(5L))
[1] FALSE FALSE FALSE FALSE FALSE
> (x <- neighbour(x, data))
[1] FALSE FALSE  TRUE FALSE FALSE
> (x <- neighbour(x, data))
[1] FALSE FALSE  TRUE FALSE  TRUE
> (x <- neighbour(x, data))
[1] FALSE FALSE FALSE FALSE  TRUE
> (x <- neighbour(x, data))
[1]  TRUE FALSE FALSE FALSE  TRUE
> (x <- neighbour(x, data))
[1]  TRUE FALSE FALSE  TRUE  TRUE
```


constraints

unconstrained solution: `all(x == FALSE)`

```
> neighbour <- function(xc, data) {
      xn <- xc
      p <- sample.int(data$na, 1L)
      xn[p] <- !xn[p]

      sumx <- sum(xn)
      if ( (sumx > data$Ksup) || (sumx < data$Kinf) )
          xc else xn
  }
```

solutions

```
> LSopt <- function(OF, algo = list(x0 = x0, nS = nS,
                        neighbour = neighbour), ...) {
      xc <- algo$x0
      xcF <- OF(xc, ...)
      for (s in seq_len(algo$nS)) {
          xn  <- algo$neighbour(xc, ...)
          xnF <- OF(xn, ...)
          if (xnF <= xcF) {
              xc  <- xn
              xcF <- xnF
          }
      }
      list(xbest = xc, OFvalue = xcF)
  }
```


Experiment

random data: 500 assets, pairwise correlation 0.5, vols between 20% and 40%

→ select between 30 and 60 assets

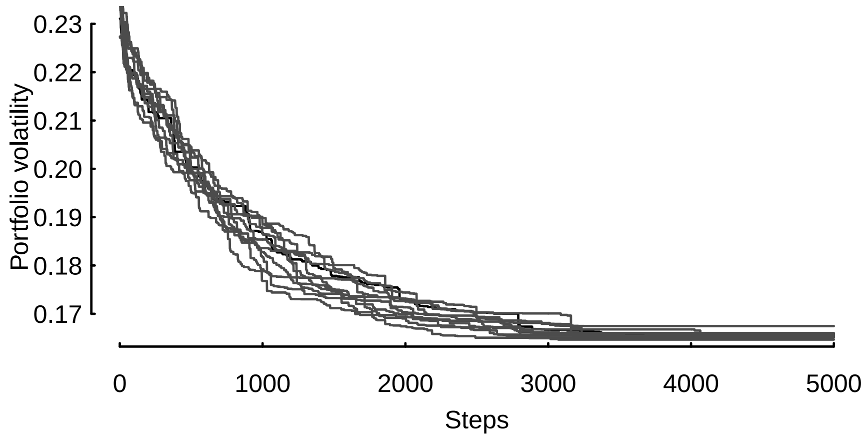
```
> na <- 500L
> C <- array(0.5, dim = c(na,na))
> diag(C) <- 1
> minVol <- 0.20
> maxVol <- 0.40
> Vols <- (maxVol - minVol) * runif(na) + minVol
> Sigma <- outer(Vols, Vols) * C
> data <- list(Sigma = Sigma, Kinf = 30L, Ksup = 60L,
               na = na)
```

Solutions

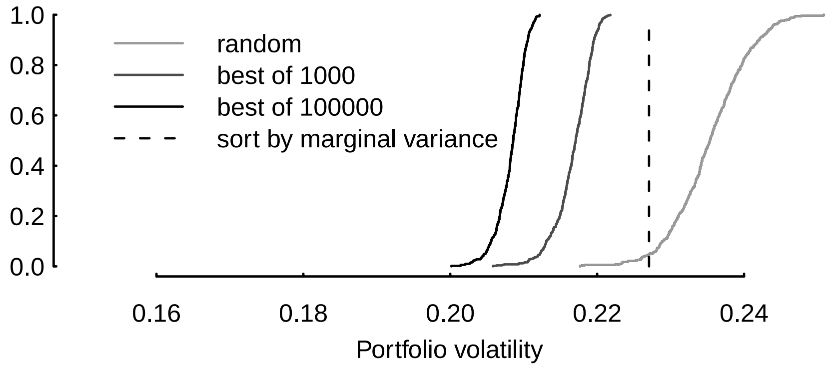
```
> card0 <- sample(data$Kinf:data$Ksup, 1L, replace = FALSE)
> assets <- sample.int(na, card0, replace = FALSE)
> x0 <- logical(na)
> x0[assets] <- TRUE
> algo <- list(x0 = x0, neighbour = neighbour, nS = 5000L,
               printDetail = FALSE, printBar = FALSE)
> system.time(solLS <- LSopt(OF, algo = algo, data = data))
```

```
   user  system elapsed
  0.240   0.004   0.241
```

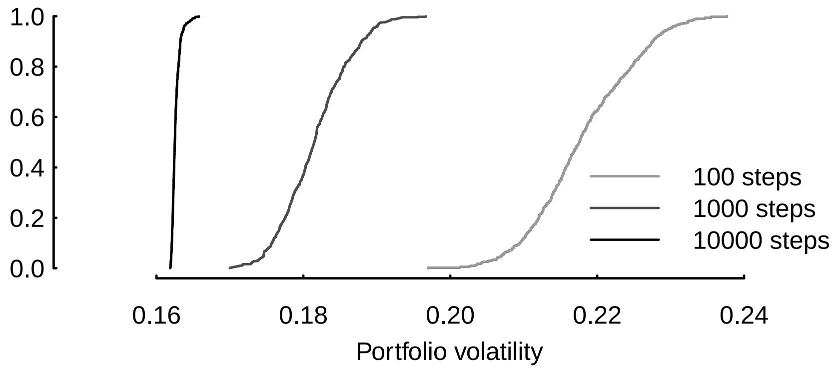

Solutions



Solutions

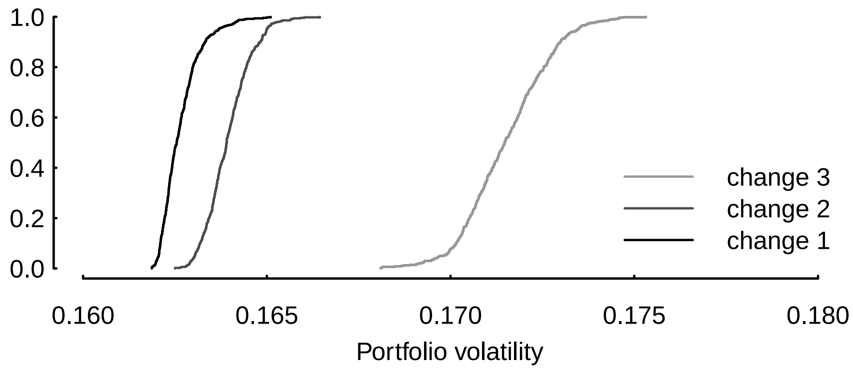


Solutions



E. Schumann 131

Solutions



E. Schumann 132

Portfolio optimisation

$$\min_{w} \ \Phi$$

$$w'\iota = 1\,,$$

$$0 \leq w_j \leq w_j^{\max} \qquad \text{for } j = 1, 2, \ldots, n_A$$

w weight vector

$w_j^{\max}$ maximum weight 5%

Φ squared portfolio return

E. Schumann 133

Portfolio optimisation

mean–variance

weights	+	returns	$\rightarrow$	portfolio return
w		m		$m'w$
weights	+	covariance matrix	$\rightarrow$	portfolio variance
w		Σ		$w'\Sigma w$

scenario optimisation

scenario matrix R (rows: scenarios, columns: assets)

weights	+	scenarios	$\rightarrow$	portfolio returns	$\rightarrow$	any portfolio statistic
w		R		Rw		$f(Rw)$

Setting up the model

portfolio weights: numeric vector w

objective function: $f(Rw)$

neighbourhood: pick two assets; increase one weight, decrease one weight

1: set ϵ
2: randomly select asset i
3: set $w_i = w_i - \epsilon$
4: randomly select asset i
5: set $w_i = w_i + \epsilon$

$\rightarrow$ enforces budget constraint (and possibly `wmin`/`wmax`)

Setting up the model

dataset `fundData`: 500 weekly return scenarios for 200 funds

```
> Data <- list(R = t(fundData),
               na = dim(fundData)[2L],  ## number of assets
               ns = dim(fundData)[1L],  ## number of scenarios
               eps = 0.5/100,           ## stepsize
               wmin = 0.00,
               wmax = 0.05,
               resample = function(x, ...)
                              x[sample.int(length(x), ...)])
```

Portfolio optimisation

objective function

- compute Rw
- evaluate f(Rw)

```
> OF <- function(w, Data) {
      Rw <- crossprod(Data$R, w)
      crossprod(Rw)
  }
```

E. Schumann 137

Portfolio optimisation

```
> neighbour <- function(w, Data) {
      toSell <- w > Data$wmin
      toBuy  <- w < Data$wmax
      i <- Data$resample(which(toSell), size = 1L)
      j <- Data$resample(which(toBuy),  size = 1L)
      eps <- runif(1L) * Data$eps
      eps <- min(w[i] - Data$wmin, Data$wmax - w[j], eps)
      w[i] <- w[i] - eps
      w[j] <- w[j] + eps
      w
  }
```

E. Schumann 138

Portfolio optimisation

set up and run TAopt

```
> w0 <- runif(Data$na); w0 <- w0/sum(w0)  ## a random solution
> algo <- list(x0 = w0,
               neighbour = neighbour,
               nS = 2000L,
               nT = 10L,
               q = 0.10,
               printBar = FALSE)
```

E. Schumann 139

Portfolio optimisation

```
> res <- TAopt(OF,algo,Data)
```

```
Threshold Accepting.

Computing thresholds ... OK.
Estimated remaining running time: 2.98 secs.


Running Threshold Accepting...
Initial solution:  0.24271
Finished.
Best solution overall: 0.0056618
```

scale solution: divide by `ns`; take square root; multiply by 100

```
[1] 0.33651
```

E. Schumann 140

Portfolio optimisation

check constraints

```
> min(res$xbest) ## should not be smaller than Data$wmin
```

```
[1] 0
```

```
> max(res$xbest) ## should not be greater than Data$wmax
```

```
[1] 0.05
```

```
> sum(res$xbest) ## should be one
```

```
[1] 1
```

E. Schumann 141

Portfolio optimisation

compare with `quadprog`

```
 OF (scaled) QP: 0.33612
 OF (scaled) TA: 0.33651
```

(`scaled`: divide by `ns`; take square root; multiply by 100)

E. Schumann 142

Computing time

mean–variance: estimate means/variance–covariance-matrix, aggregate for portfolio

$\rightarrow$ computing time varies with n_{A}

heuristics: compute scenarios/time series of portfolio returns, then compute Φ

$\rightarrow$ computing time varies with n_{A} and number of scenarios n_{S}

Computing time

matrix R of observations of size $n_{\mathrm{S}} \times n_{\mathrm{A}}$
$\rightarrow$ Rw for each function evaluation

for LS/TA: updating possible

$$\begin{aligned} w^{\mathrm{n}} &= \quad w^{\mathrm{c}} + w^{\Delta} \\ Rw^{\mathrm{n}} &= R(w^{\mathrm{c}} + w^{\Delta}) = \underbrace{Rw^{\mathrm{c}}}_{\text{known}} + Rw^{\Delta} \end{aligned}$$

R_*: changed columns (size $n_{\mathrm{S}} \times 2$)
w_*^{Δ}: vector of changes (size 2×1)

$Rw^{\Delta} \rightarrow R_* w_*^{\Delta}$.

Computing time

```
set n_steps
randomly generate current solution x^c
for i = 1 : n_steps do
    generate x^n ∈ N(x^c)  and compute  △= Φ(x^n) − Φ(x^c)
    if △< 0 then x^c = x^n
end for
x^sol = x^c
```

manipulate solution x through Φ and N $\rightarrow$ code x as list containing

- w
- Rw

$\rightarrow$ change in N

Updating

with updating

```
> OFU <- function(sol, Data)
      crossprod(sol$Rw)
> neighbourU <- function(sol, Data){
      wn <- sol$w
      toSell <- wn > Data$wmin; toBuy  <- wn < Data$wmax
      i <- Data$resample(which(toSell), size = 1L)
      j <- Data$resample(which(toBuy), size = 1L)
      eps <- runif(1) * Data$eps
      eps <- min(wn[i] - Data$wmin, Data$wmax - wn[j], eps)
      wn[i] <- wn[i] - eps; wn[j] <- wn[j] + eps
      Rw <- sol$Rw + Data$R[ ,c(i,j)] %*% c(-eps,eps)
      list(w = wn, Rw = Rw)
  }
```


Updating

```
> w0 <- runif(Data$na); w0 <- w0/sum(w0)  ## a random solution
> Data$R <- fundData
> sol <- list(w = w0, Rw = Data$R %*% w0)
> algo <- list(x0 = sol,
               neighbour = neighbourU,
               nS = 2000L,
               nT = 10L,
               q = 0.10,
               printBar = FALSE,
               printDetail = FALSE)
> res <- TAopt(OFU,algo,Data)
```

Robustness

the weight of asset 200

```
> wqp[200]
```

```
[1] 0.000000000000000027105
```

```
> fundData <- cbind(fundData, fundData[, 200L])
> dim(fundData)
```

```
[1] 500 201
```

```
> qr(fundData)$rank
```

```
[1] 200
```

```
> qr(cov(fundData))$rank
```

```
[1] 200
```

E. Schumann 148

Robustness

```
> cat(try(result.QP <- solve.QP(Dmat = covMatrix,
                               dvec = rep(0, Data$na),
                               Amat = t(rbind(A,B)),
                               bvec = rbind(a,b),
                               meq = 1L)))
```

```
Error in solve.QP(Dmat = covMatrix, dvec = rep(0, Data$na), Amat = t(rbind(A,  :
  matrix D in quadratic function is not positive definite!
```

E. Schumann 149

Robustness

```
> res2 <- TAopt(OFU, algo, Data)
```

```
[1] 0.33636
```

weights 200 and 201

```
> res2$xbest$w[200:201]
```

```
[1] 0 0
```

E. Schumann 150

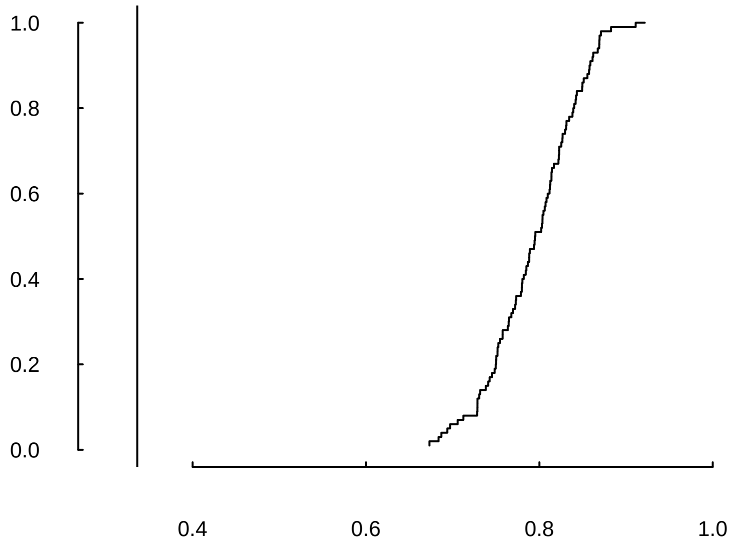
Other objective functions

$$\frac{1}{n_S}\sum_{r_i<\theta}(\theta - r_i)^2$$

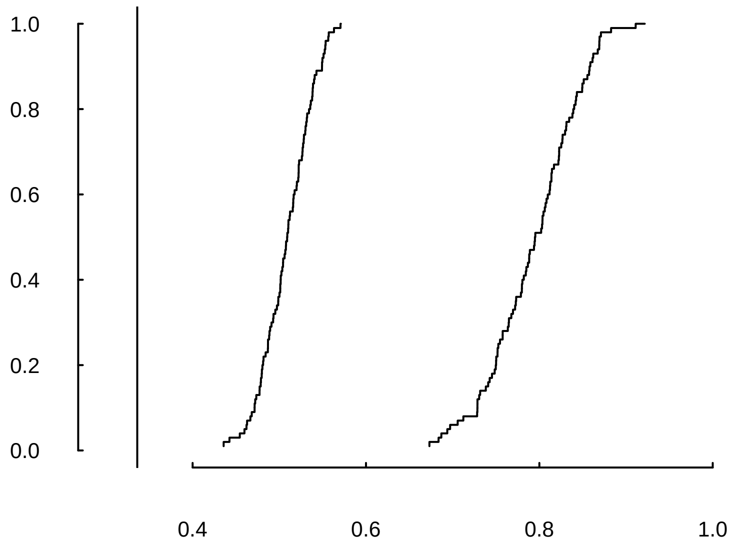
```
> OF <- function(w, Data) { ## semi-variance
      Rw <- crossprod(Data$R, w) - Data$theta
      Rw <- Rw - abs(Rw)
      sum(Rw*Rw) / (4 * Data$ns)
  }

> OF <- function(w, Data) { ## Omega
      Rw <- crossprod(Data$R, w) - Data$theta
      -sum(Rw - abs(Rw)) / sum(Rw + abs(Rw))
  }
```

More iterations

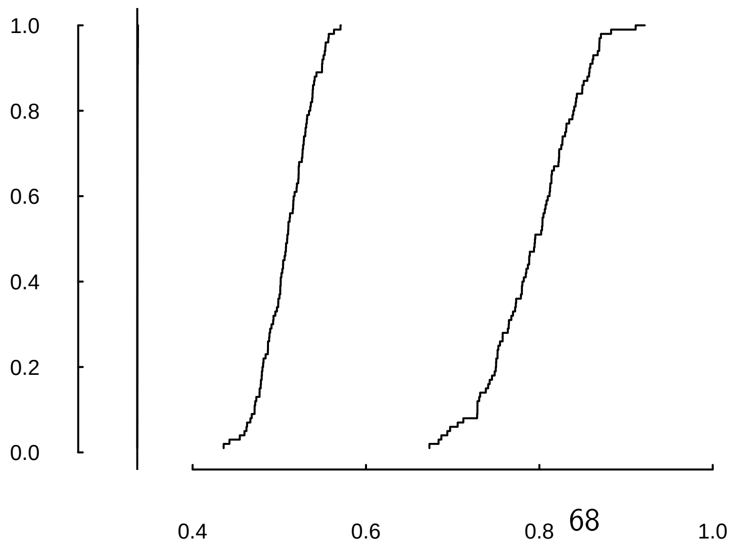
1500 iterations



1500 iterations – 2500 iterations



1500 iterations – 2500 iterations – 15 000 iterations

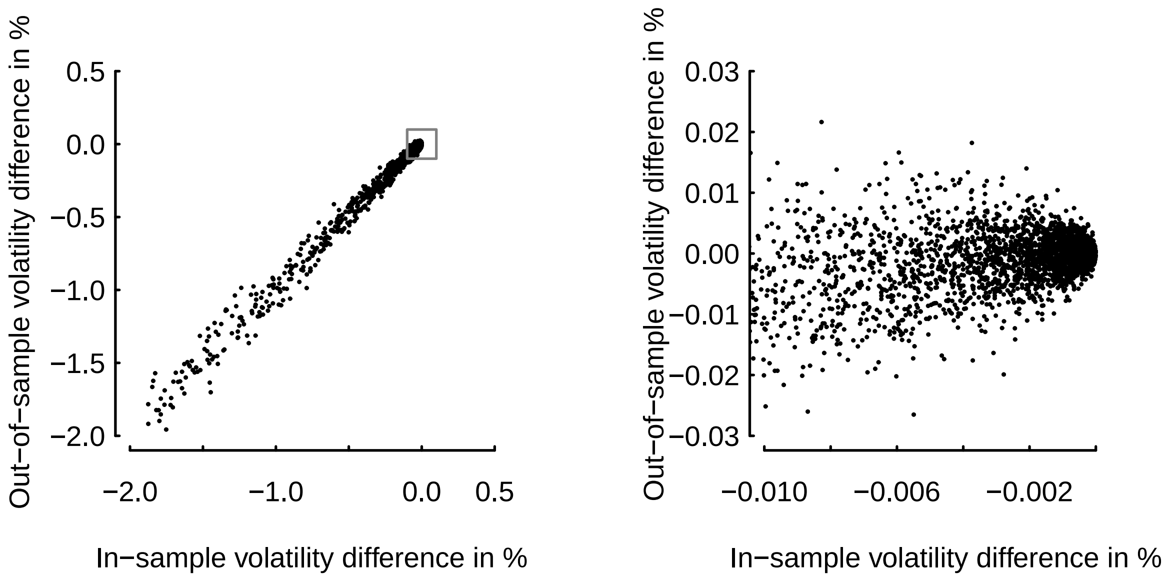


Good enough?

for $i = 1 : 5000$ **do**
  sample 400 scenarios without replacement
  compute optimal portfolio with QP
  set $n_{\text{steps}} = i$
  compute portfolio with TA, compute in-sample difference between QP and TA
  compute out-of-sample difference for QP and TA on remaining 100 scenarios
end for

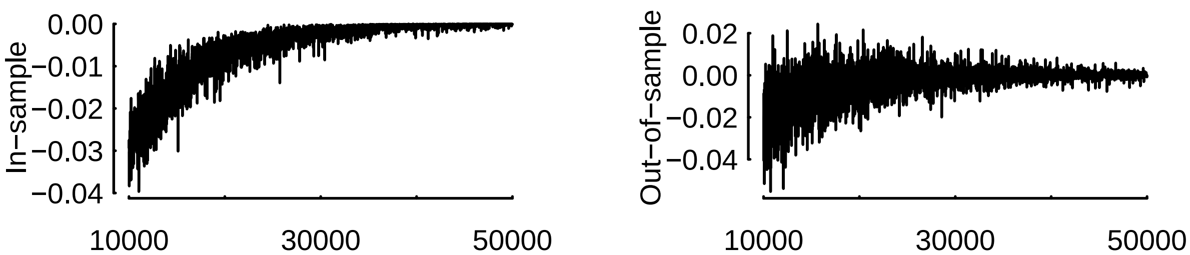
objective function value of QP − objective function value of TA

Good enough?



in-sample versus out-of-sample difference

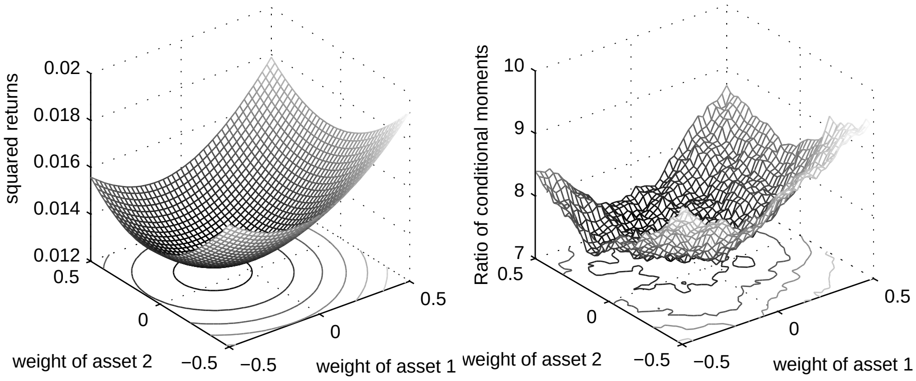
scenario optimisation



in-sample versus out-of-sample difference depending on the number of iterations

applications

portfolio optimisation



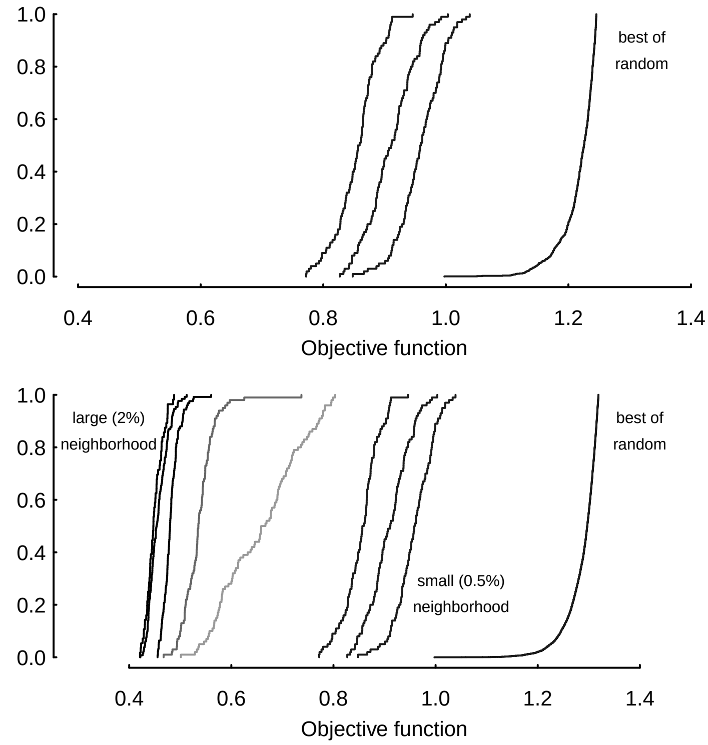
applications

$$\min_w \Phi$$

$$w'\iota = 1$$
$$0 \leq w_j \leq w_j^{\text{sup}} \quad \text{for } j = 1, 2, \ldots, n_A$$
$$K_{\text{inf}} \leq K \leq K_{\text{sup}}$$

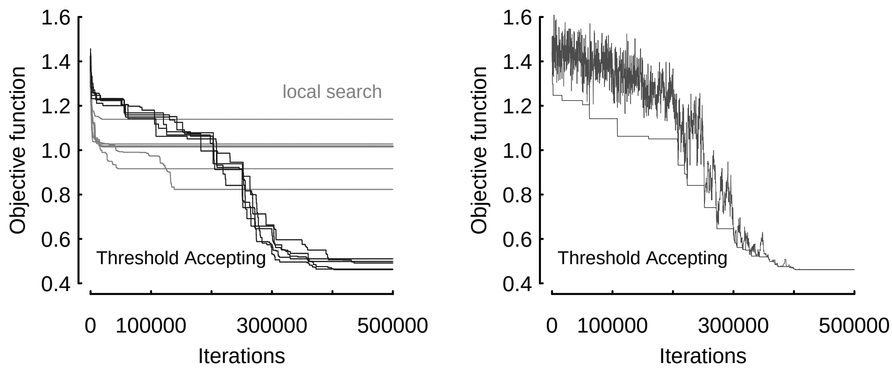
$$\Phi = \frac{\mathcal{C}^-_{\gamma-}}{\mathcal{C}^+_{\gamma+}}$$

solutions



E. Schumann 158

solutions



E. Schumann 159

a case study

$$\min_x \ \Phi(x)$$

$$\sum_{j\in\mathcal{J}} x_j\, p_{0j} = v_0$$

$$x_j^{\text{inf}} \leq x_j \leq x_j^{\text{sup}} \qquad j \in \mathcal{J}$$

$$K_{\text{inf}} \leq \#\{\mathcal{J}\} \leq K_{\text{sup}}$$

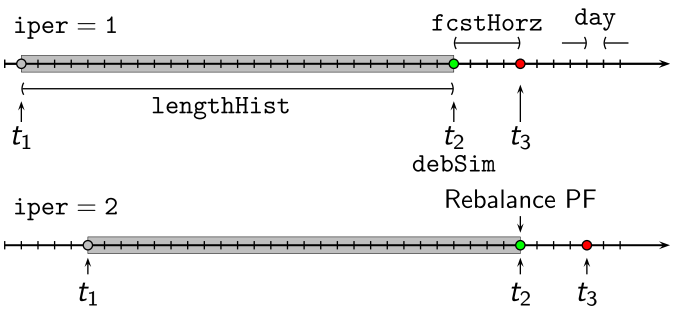
$$\vdots$$

(x = numbers of shares, $\mathcal{A}$ = all assets, $\mathcal{J}$ = assets included in portfolio)

→ Threshold Accepting

data and methodology

- 600 assets (EUR) from DJ STOXX (7-Jan-1999 — 19-Mar-2008)
- market capitalisation $> 4 \times 10^7$ Euros (248 assets)
- historical window (52 weeks), horizon (12 weeks) → 40 rebalancings



- 10bp variable cost for long positions
  minimum trading size (5 000)
- holding size constraints
- sector allocation constraints

estimation

bootstrapping returns (r^B) from a simple regression model:

$$r_{it} = \alpha_i + \beta_i \; r_{Mt} + \cdots + \epsilon_{it} \qquad \begin{array}{l} i = 1, \dots, n_{\mathrm{A}} \\ t = 1, \dots, T \end{array}$$

regressors: indices, PCA ...

estimate $\hat{\alpha}_i$, $\hat{\beta}_i$, ... $i = 1, \dots, n_{\mathrm{A}}$ from model
for $k = 1 : n_S$ **do**
  draw with replacement $\tau_M \in \{1, \dots, T\}$
  for $i = 1 : n_{\mathcal{A}}$ **do**
    draw with replacement $\tau_i \in \{1, \dots, T\}$
    $r^B_{ik} = \hat{\alpha}_i + \hat{\beta}_i \; r_{M\tau_M} + \epsilon_{i\tau_i}$
  end for
end for

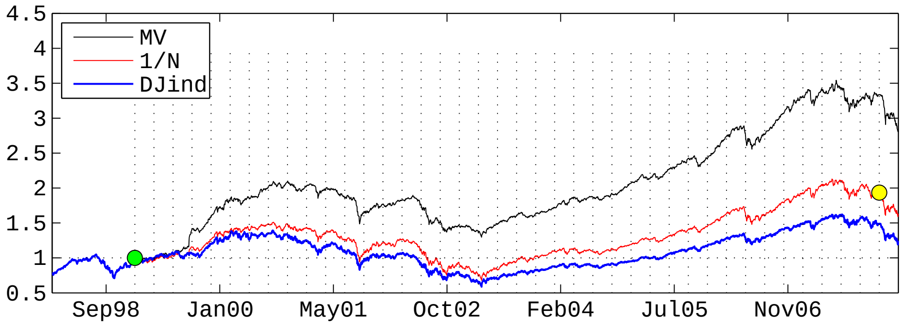
benchmark: minimum-variance portfolio long-only

- Σ estimated as covariance matrix $\hat{\Sigma}$ from historical observations

$$\min_{w} \; w' \hat{\Sigma} w$$
$$\sum w = 1$$
$$0 \leq w_j \leq w_j^{\sup} \qquad j = 1, \dots, n_{\mathrm{A}}$$

- optimisation with maximum holding size and sector allocation constraints done with Matlab's `quadprog`

benchmark: minimum-variance portfolio long-only

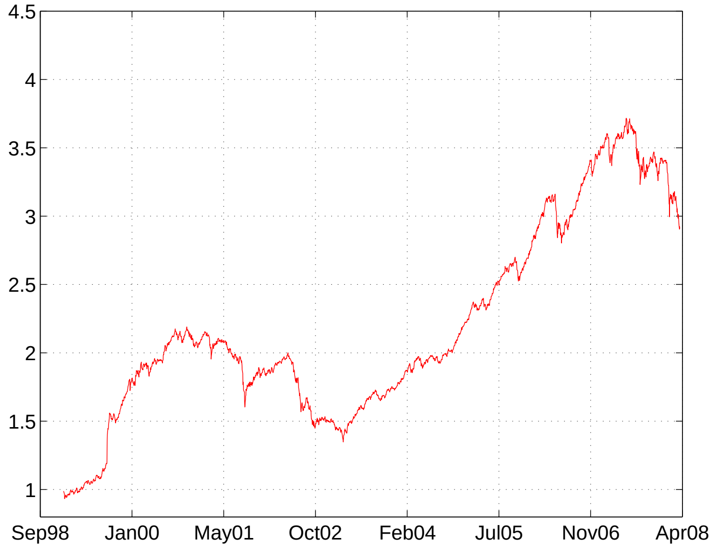


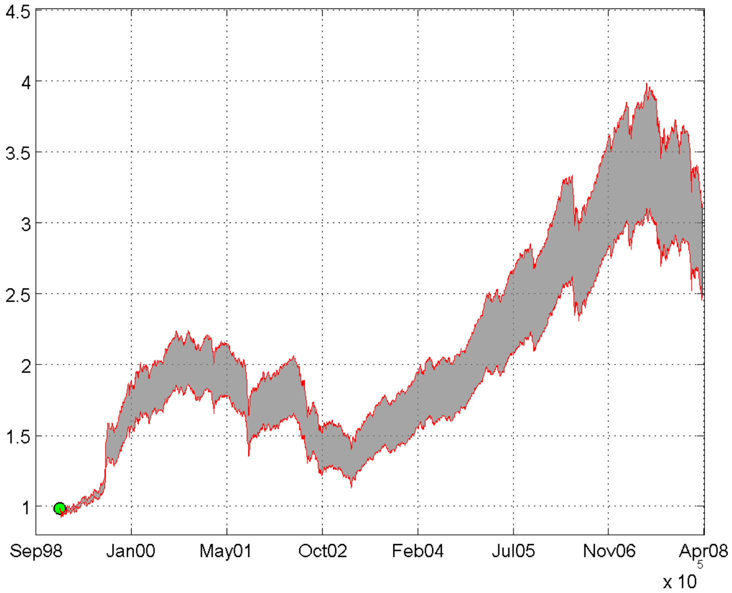
introducing uncertainty: draw T daily data with replacement, compute minimum-variance portfolio from bootstrapped time series

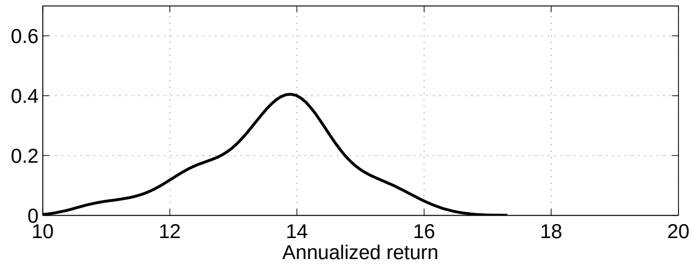
for $k = 1 : 100$ **do**
  for $j = 1 : T$ **do**
    draw with replacement $\tau \in \{1, \dots, T\}$
    $R^B_{j\bullet} = R_{\tau\bullet}$
  end for
  compute MV portfolio
end for

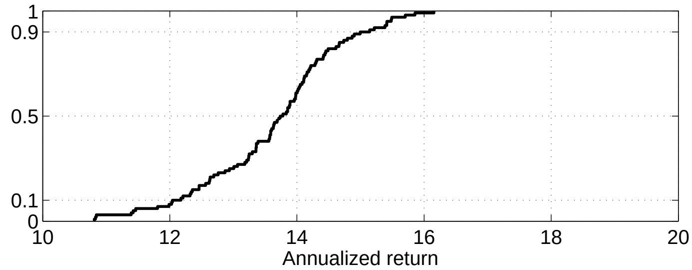
alternatively: use jackknife

benchmark: MV long-only

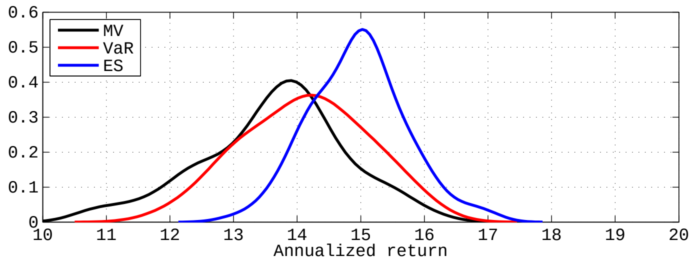


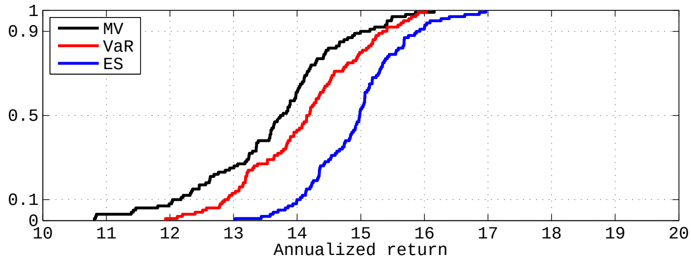




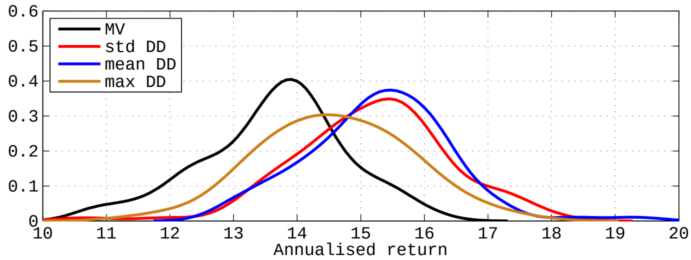


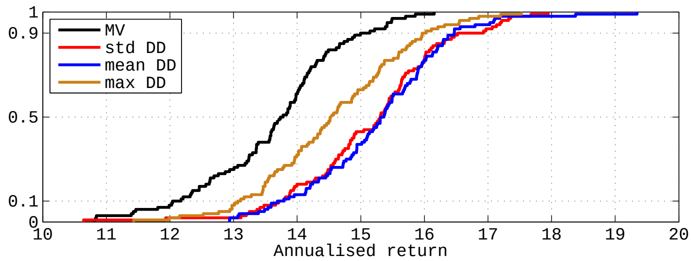
results VaR/Expected Shortfall



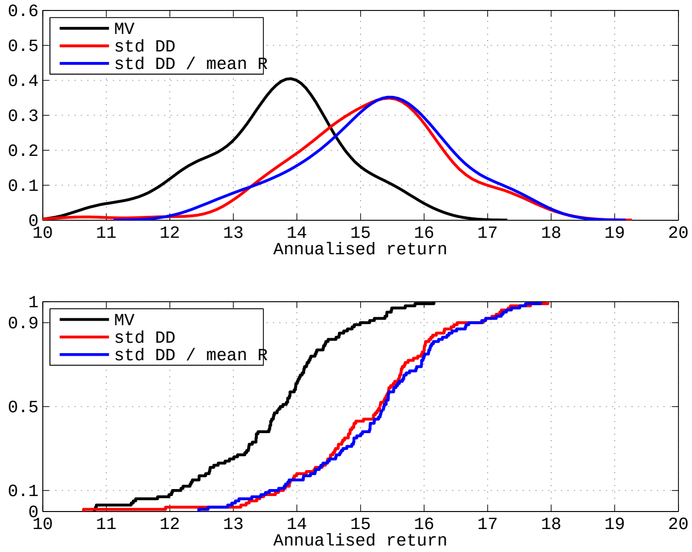


results drawdowns

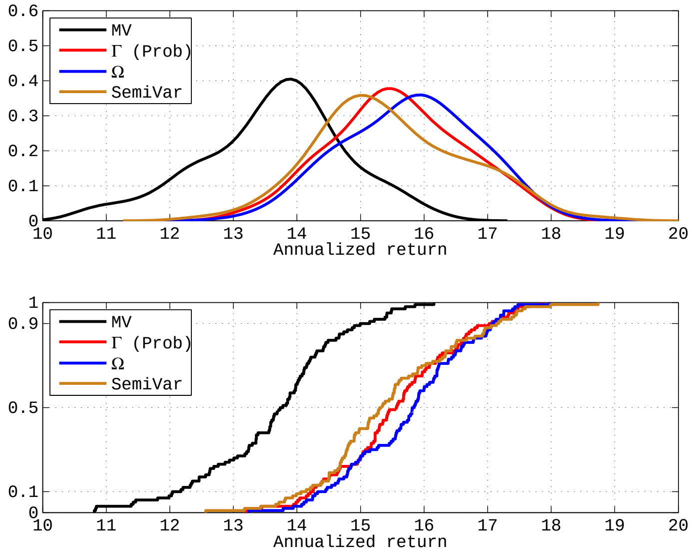




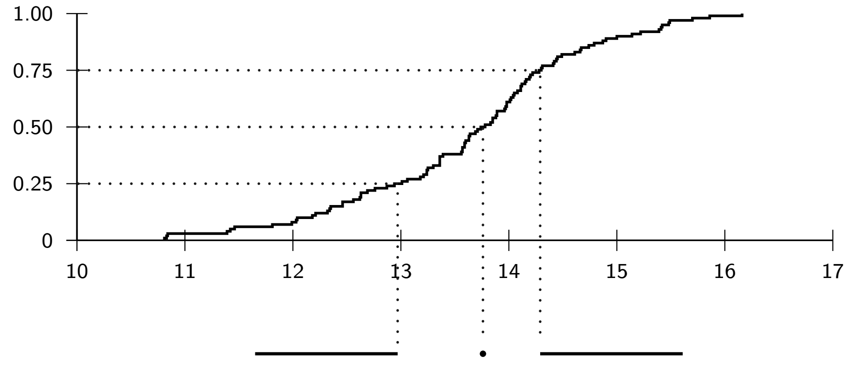
results drawdowns



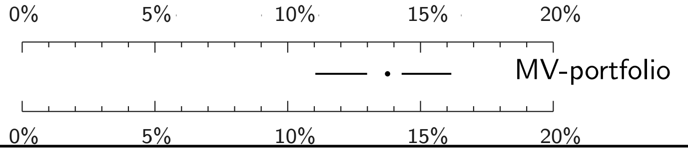
results partial moments



more results – MV portfolio

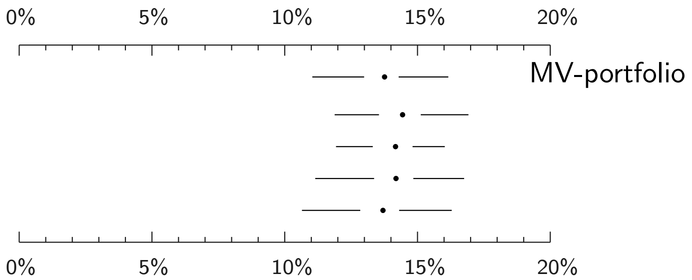


reward	risk	median
c	$\mathcal{M}_2$	13.76

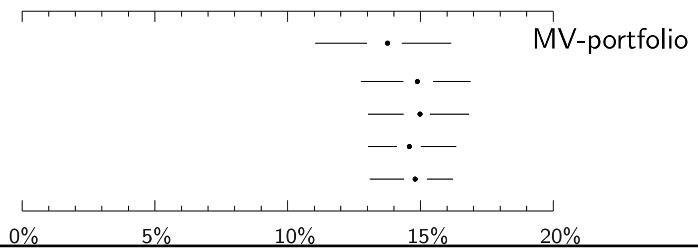


results `VaR`, `ES`

reward	risk	median
c	$\mathcal{M}_2$	13.76
c	$\mathcal{Q}_1$	14.44
c	$\mathcal{Q}_5$	14.17
c	$\mathcal{Q}_{10}$	14.19
c	$\mathcal{Q}_{20}$	13.70

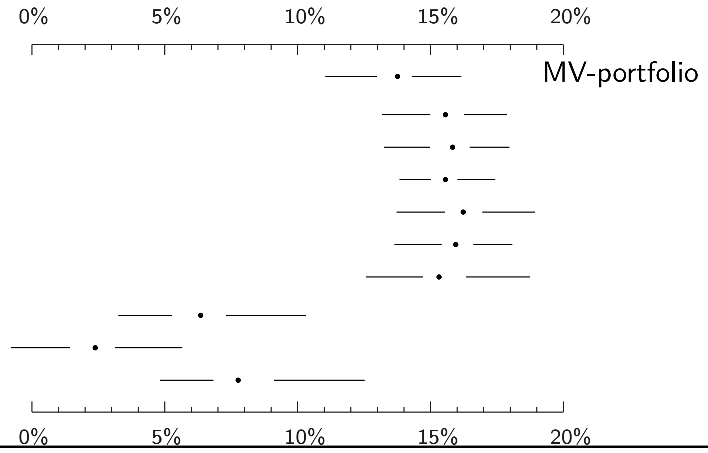


reward	risk	median
c	$\mathcal{M}_2$	13.76
c	$\mathcal{C}_1^-(\mathcal{Q}_1)$	14.88
c	$\mathcal{C}_1^-(\mathcal{Q}_5)$	14.98
c	$\mathcal{C}_1^-(\mathcal{Q}_{10})$	14.58
c	$\mathcal{C}_1^-(\mathcal{Q}_{20})$	14.80

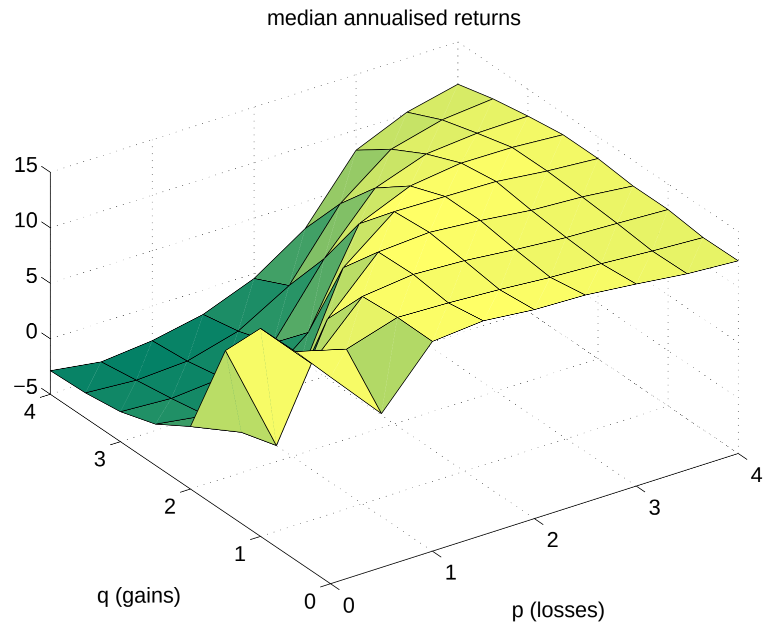


results partial moments

reward	risk	median
c	$\mathcal{M}_2$	13.76
$\mathcal{P}_0^+$	$\mathcal{P}_0^-$	15.56
$\mathcal{P}_1^+$	$\mathcal{P}_1^-$	15.83
$\mathcal{P}_1^+$	$\mathcal{P}_2^-$	15.56
$\mathcal{P}_{1.5}^+$	$\mathcal{P}_{1.5}^-$	16.23
$\mathcal{P}_{1.5}^+$	$\mathcal{P}_2^-$	15.95
$\mathcal{P}_2^+$	$\mathcal{P}_2^-$	15.32
$\mathcal{P}_3^+$	$\mathcal{P}_2^-$	6.35
$\mathcal{P}_4^+$	$\mathcal{P}_2^-$	2.38
$\mathcal{P}_4^+$	$\mathcal{P}_3^-$	7.76



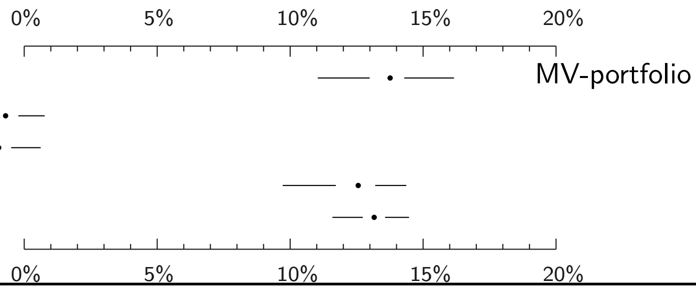
results partial moments



E. Schumann 173

results partial moments

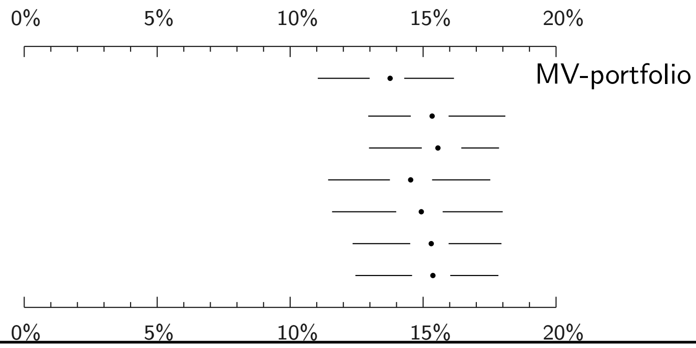
reward	risk	median
c	$\mathcal{M}_2$	13.76
$\mathcal{P}_2^+$	c	−0.70
$\mathcal{P}_3^+$	c	−0.96
c	$\mathcal{P}_2^+$	12.56
c	$\mathcal{P}_3^+$	13.16



E. Schumann 174

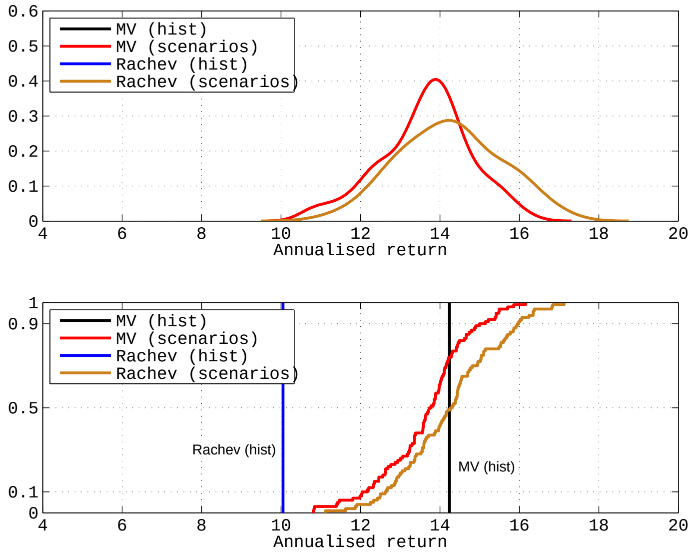
results drawdowns

reward	risk	median
c	$\mathcal{M}_2$	13.76
c	$\mathcal{D}_{\text{mean}}$	15.34
$\mathcal{M}_1$	$\mathcal{D}_{\text{mean}}$	15.56
c	$\mathcal{D}_{\text{max}}$	14.53
$\mathcal{M}_1$	$\mathcal{D}_{\text{max}}$	14.93
c	$\mathcal{D}_{\text{std}}$	15.31
$\mathcal{M}_1$	$\mathcal{D}_{\text{std}}$	15.37



E. Schumann 175

scenarios vs historical data



When is a solution 'optimal enough'?

set of 600 European equities

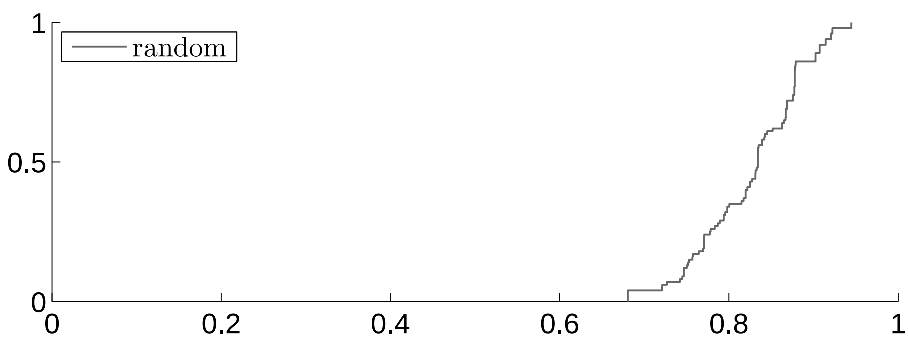
aim: to find portfolio that minimises semi-variance, subject to

(i) only 20–50 assets in portfolio

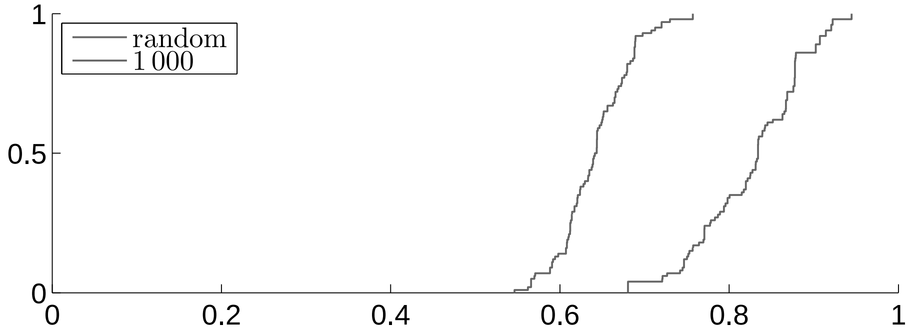
(ii) all weights between 1% and 5%

→ characterise solution by objective function value
→ run large number of optimisations → compare distribution of solutions

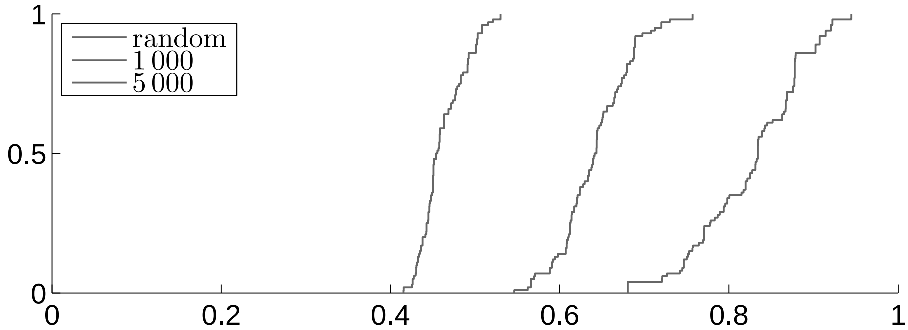
in-sample convergence



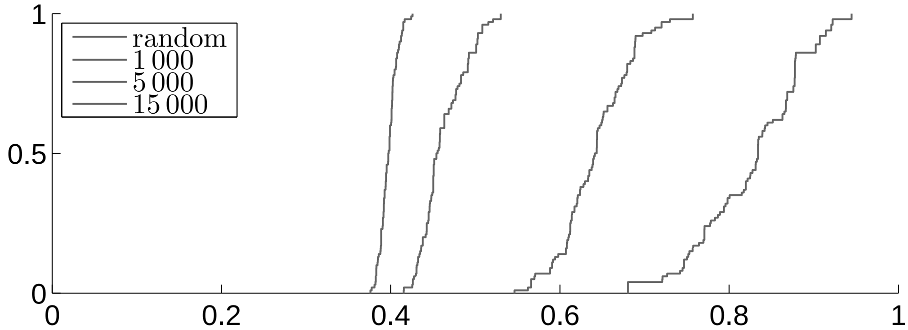
in-sample convergence



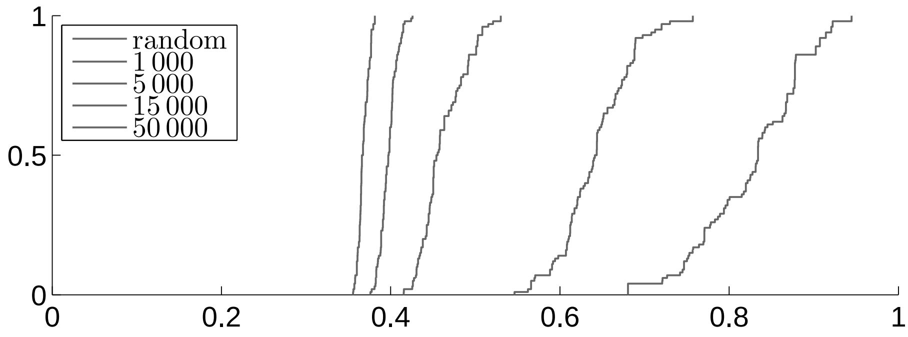
in-sample convergence



in-sample convergence

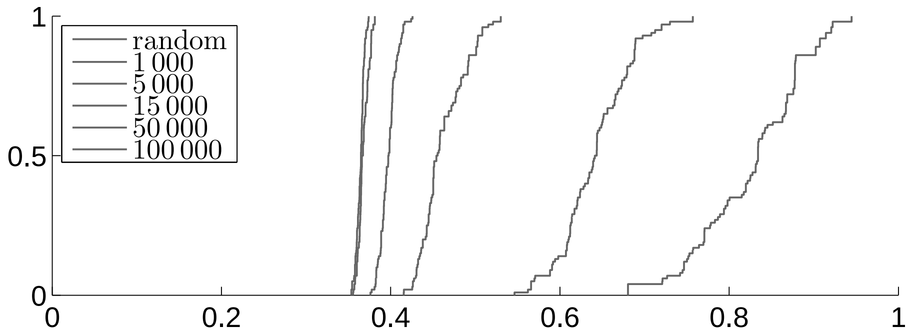


in-sample convergence



E. Schumann 182

in-sample convergence



E. Schumann 183

In-sample and out-of-sample

solution: vector of portfolio weights x

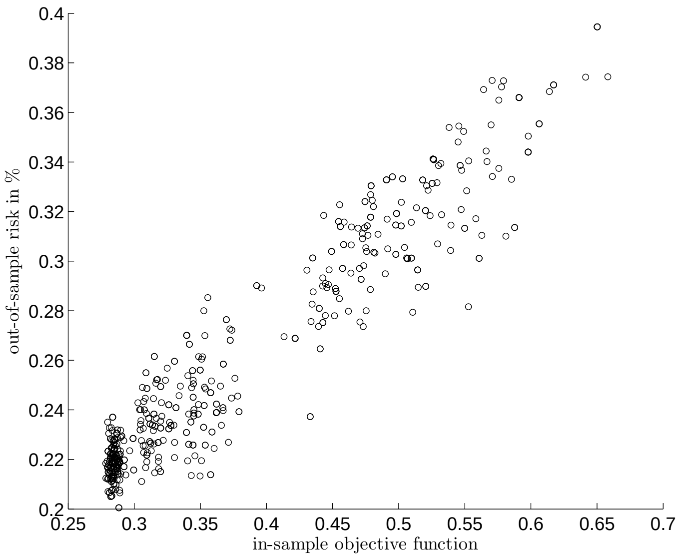
characterise x by objective function value $f(x)$

$\rightarrow$ run R restarts with differing numbers of steps

$\rightarrow$ compare $\phi_1, \dots, \phi_R$ in-sample and out-of-sample

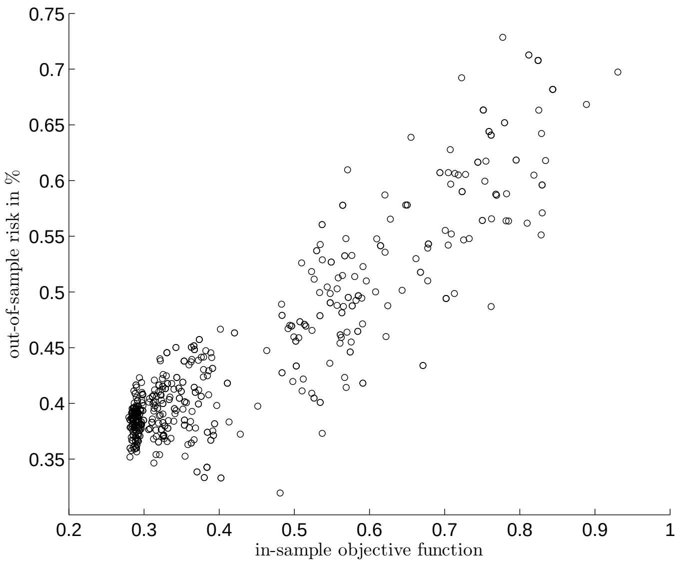
E. Schumann 184

out-of-sample: single period risk



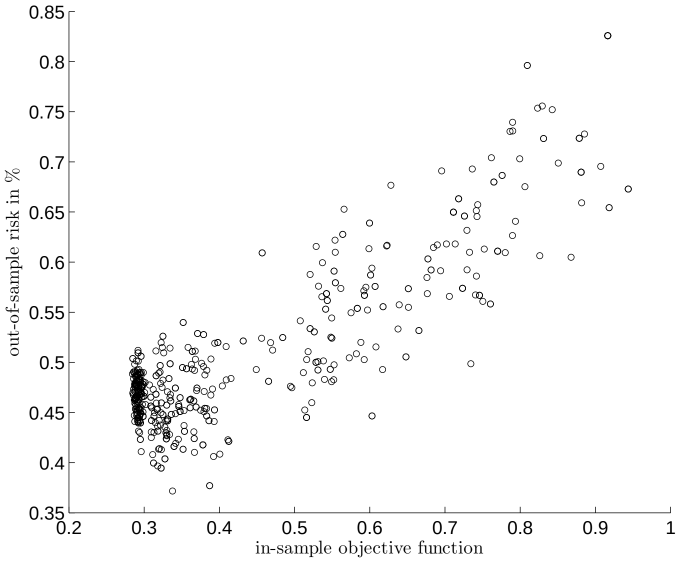
E. Schumann 185

out-of-sample: single period risk

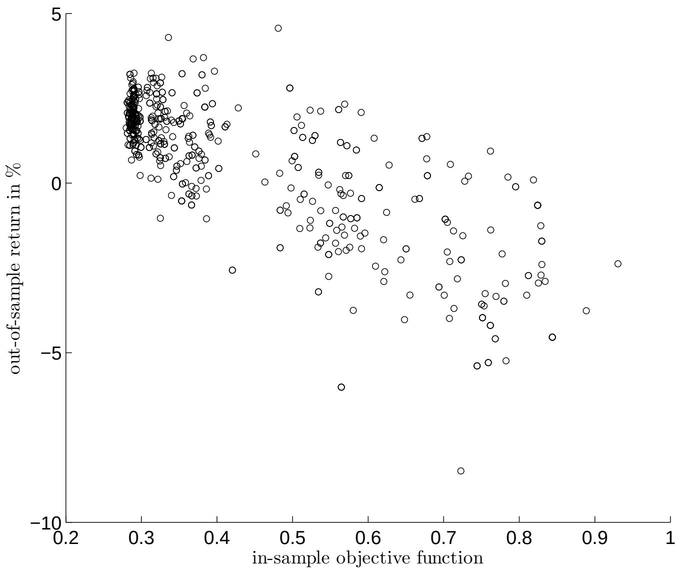


E. Schumann 186

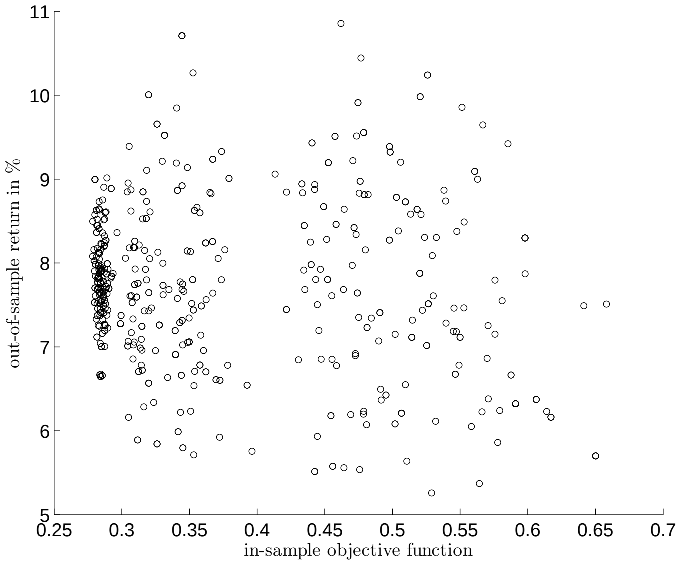
out-of-sample: single period risk



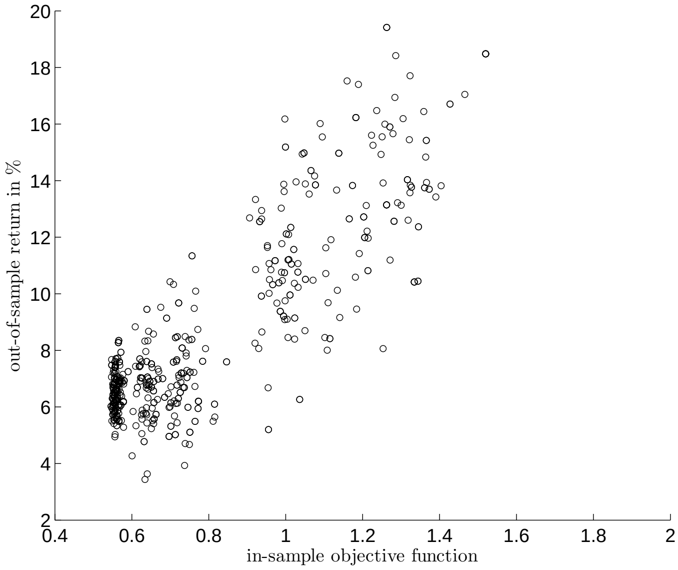
out-of-sample: single period return



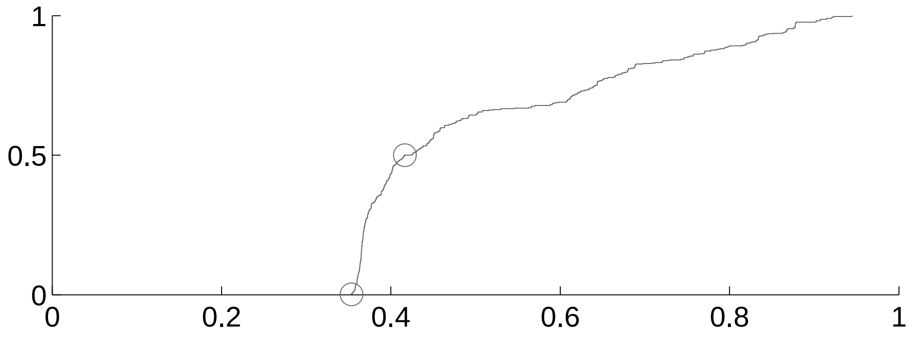
out-of-sample convergence: single period return



out-of-sample convergence: single period return

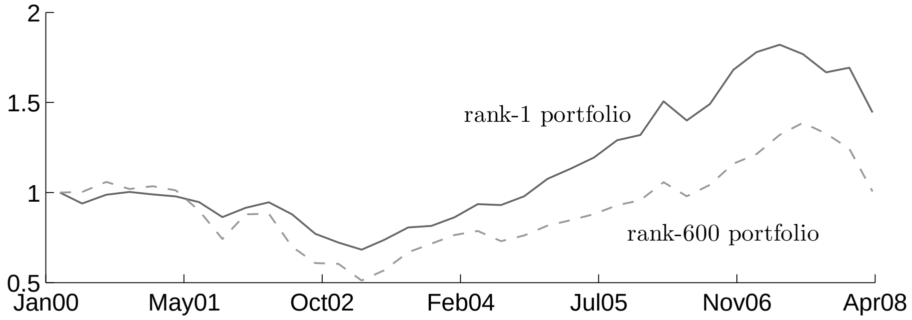


Rank portfolios



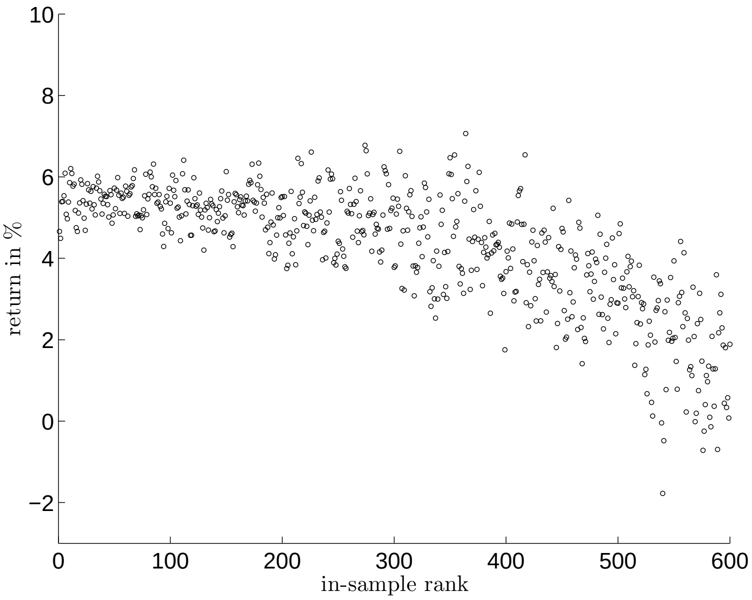
E. Schumann 191

Rank portfolios, paths



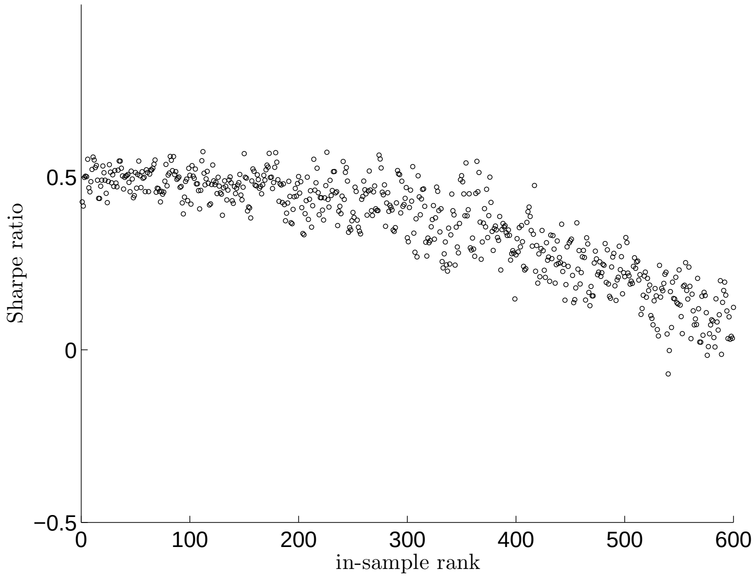
E. Schumann 192

Rank portfolios, out-of-sample final wealth

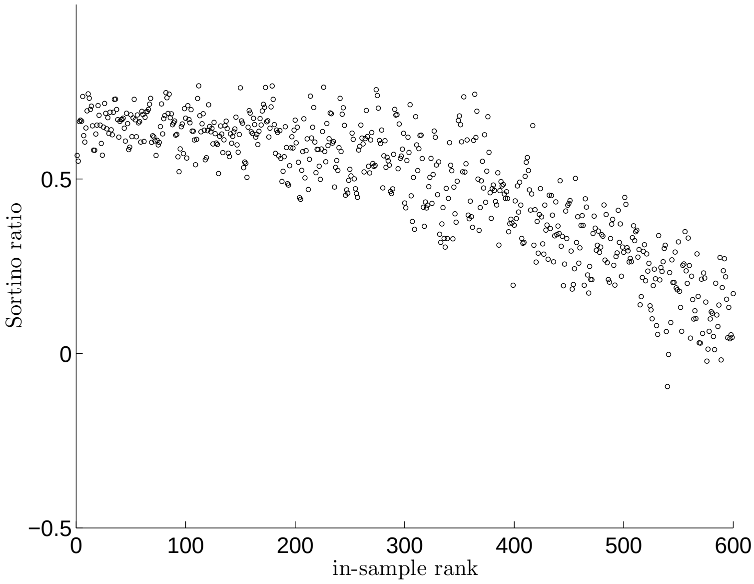


E. Schumann 193

Rank portfolios, out-of-sample Sharpe ratios



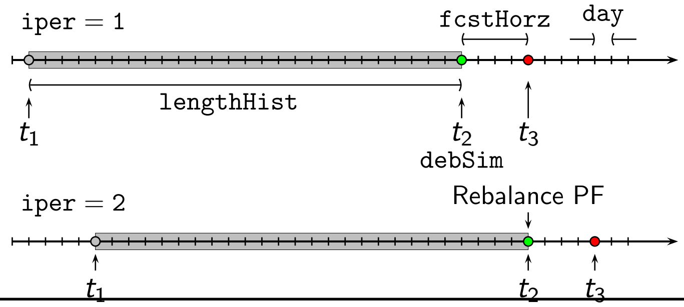
Rank portfolios, out-of-sample Sortino ratios



Sensitity of model and stochastics of optimisation

Gilli and Schumann (2016)

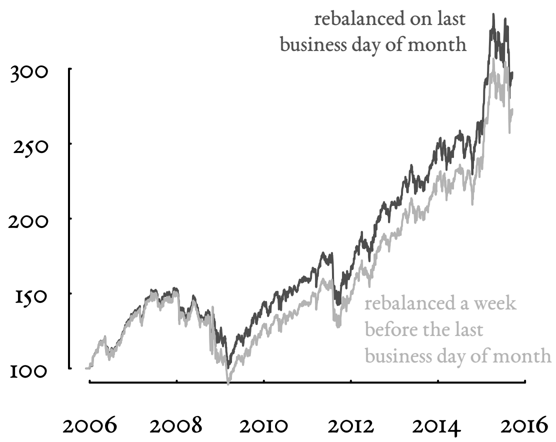
- dataset: DAX stocks Jan 2004 – Sep 2015
- compute long-only minimum-variance portfolio
- weights between 0 and 10%
- historical window (1 year), horizon (1 month)



E. Schumann 196

Sensitity of model and stochastics of optimisation

out-of-sample path of walkforward, with QP: 11.8% p.a. *v* 10.8% p.a.

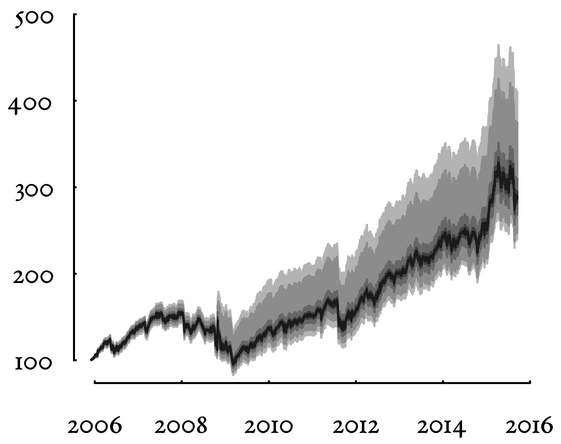


E. Schumann 197

Sensitity of model and stochastics of optimisation

10 000 out-of-sample paths with QP

- ◦ random rebalancing dates, 20–80 days business days apart
- ◦ random historical window, 120–500 business days

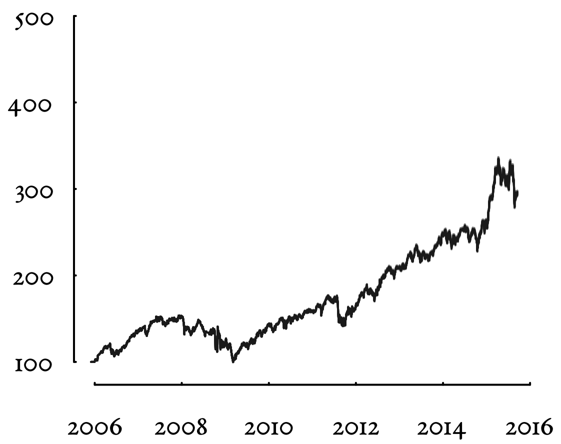


E. Schumann 198

Sensitity of model and stochastics of optimisation

10 000 out-of-sample paths with Local Search

- ◦ fixed rebalancing dates (end of month)
- ◦ fixed historical window, 260 business days



E. Schumann 199

Conclusions

- alternative portfolio selection models
  - optimisation more difficult, but manageable
  - add value over mean–variance
  - ‘good’ solutions suffice
- using historical data (alone) leads to overfitting
- concentrating on risk (rather than reward) more important
- problem very sensitive to data

Subset sum problem

[R] Solving an optimization problem: selecting an ‘optimal’ subset

More information

the NMOF package is on CRAN/R-Forge

```
> install.packages("NMOF") ## CRAN
> install.packages("NMOF",
                   repos = "http://R-Forge.R-project.org")


> require("NMOF")
> showExamples("tria.R")  ## load code examples from book
```

mailing list: NMOF-News
https://lists.r-forge.r-project.org/cgi-bin/mailman/listinfo/nmof-news

and also at gmane.comp.finance.nmof.announce

references

Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

Gunter Dueck and Tobias Scheuer. Threshold Accepting. A general purpose optimization algorithm superior to Simulated Annealing. *Journal of Computational Physics*, 90(1):161–175, 1990.

Gunter Dueck and Peter Winker. New concepts and algorithms for portfolio choice. *Applied Stochastic Models and Data Analysis*, 8(3):159–178, 1992.

Russell C. Eberhart and James Kennedy. A new optimizer using Particle Swarm theory. In *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, pages 39–43, Nagoya, Japan, 1995.

Peter C. Fishburn. Mean–risk analysis with risk associated with below-target returns. *American Economic Review*, 67(2):116–126, 1977.

Manfred Gilli and Enrico Schumann. Optimal enough? *Journal of Heuristics*, 17(4):373–387, 2011.

Manfred Gilli and Enrico Schumann. Accuracy and precision in finance. In Otto Hieronymi and Martino Lo Cascio, editors, *A New Social Market Economy for the 21st Century*. Aracne, 2016. ISBN 978-88-548-9041-1.

Manfred Gilli, Dietmar Maringer, and Enrico Schumann. *Numerical Methods and Optimization in Finance*. Elsevier/Academic Press, 2011. URL `http://nmof.net`.

Fred Glover. Future paths for integer programming and links to Artificial Intelligence. *Computers and Operations Research*, 13(5):533–549, 1986.

J. Dave Jobson and Bob Korkie. Estimation for Markowitz efficient portfolios. *Journal of the American Statistical Association*, 75(371):544–554, 1980. ISSN 01621459.

Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by Simulated Annealing. *Science*, 220 (4598):671–680, 1983.

Harry M. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

Bernd Scherer. *Portfolio Construction and Risk Budgeting*. Risk Books, 2nd edition, 2004.

Enrico Schumann and David Ardia. Heuristic methods in finance. *Statistical Computing & Statistical Graphics Newsletter*, 22(1):13–19, 2011.

Rainer M. Storn and Kenneth V. Price. Differential Evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.