Portfolio Selection with Probabilistic Utility: Revisited

Dr. Bernhard Pfaff
bernhard_pfaff@fra.invesco.com

Invesco Asset Management GmbH
Frankfurt am Main

R in Finance
17 – 18 May 2013
Chicago

Contents

Motivation
Overview

- Portfolio selection problems derived from utility functions.
- E.g. mean-variance optimisation:
  $U = \lambda \boldsymbol{\omega}' \boldsymbol{\mu} - (1 - \lambda) \boldsymbol{\omega}' \Sigma \boldsymbol{\omega}$.
- Allocation sensitive to parameters $\boldsymbol{\mu}, \Sigma, \lambda$.
- Problem-solving approaches: robust/bayesian estimators and/or robust optimization.
- Nota bene: $\boldsymbol{\mu}$ and Σ are random variables; as such the allocation vector $\boldsymbol{\omega}$ is a random variable itself.
- In this talk: probabilistic interpretation of utility functions.

Motivation
Probabilistic Utility I

- Approach introduced by Rossi et al. (2002) and Marschinski et al. (2007).
- Utility function is interpreted as the logarithm of the probability density for a portfolio.
- Optimal allocation is defined as the expected value of the portfolio's weights with respect to that probability, *i.e.*, the weights are viewed as parameters of this distribution.

Motivation
Probabilistic Utility II

- Given: $u = u(\boldsymbol{\omega}, U, \theta)$, whereby $\boldsymbol{\omega}$ is weight vector, U the assumed utility function and θ a catch-all parameter vector (*e.g.* expected returns, dispersion, risk sensitivity).
- Expected utility is proportional to the logarithm of a probability measure:
  $\boldsymbol{\omega} \sim P(\boldsymbol{\omega}|U, \theta) = Z^{-1}(\nu, U, \theta) \exp(\nu u(\boldsymbol{\omega}, U, \theta))$.
- Normalizing constant: $Z(\nu, U, \theta) = \int_{\mathfrak{D}(\boldsymbol{\omega})} [\mathrm{d}\boldsymbol{\omega}] \exp(\nu u(\boldsymbol{\omega}, U, \theta))$.
- Convergence to maximum utility ($\nu \to \infty$) or equal-weight solution ($\nu \to 0$) is controlled by: $\nu = pN^{\gamma}$.
- Portfolio solution is then defined as:
  $\bar{\boldsymbol{\omega}}(U, \theta) = Z^{-1}(\nu, U, \theta) \int_{\mathfrak{D}(\boldsymbol{\omega})} [\mathrm{d}\boldsymbol{\omega}] \boldsymbol{\omega} \exp(\nu u(\boldsymbol{\omega}, U, \theta))$

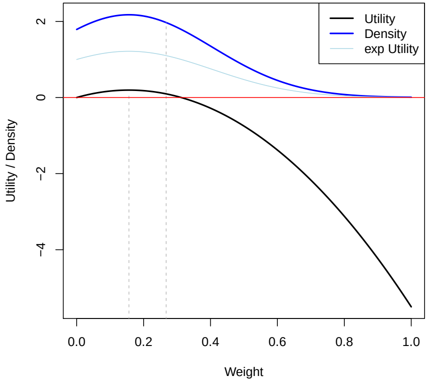
Motivation

Example: quadratic utility, one risky asset, I

```
## Utility function
U1 <- function(x, mu, risk, lambda = 0.5){
  lambda * x * mu - (1 - lambda) * risk * x^2
}
## Sequence of possible weights
x <- seq(0, 1, length.out = 1000)
## Utility
u1 <- U1(x, mu = 5, risk = 16, lambda = 0.5)
## Optimal Allocation (in percentage)
MUopt <- round(x[which.max(u1)] * 100, 2)
## Now introducing concept of probabilistic utility
U1DU <- function(x, mu, risk, lambda = 0.5, nu = 1){
  exp(nu * U1(x = x, mu = mu, risk = risk, lambda = lambda))
}
u1u <- U1DU(x, mu = 5, risk = 16, lambda = 0.5, nu = 1)
## Density
U1DS <- function(x, mu, risk, lambda = 0.5, nu = 1){
  Dconst <- integrate(U1DU, lower = 0, upper = 1, mu = mu,
                      risk = risk, lambda = lambda, nu = nu)$value
  1 / Dconst * U1DU(x = x, mu = mu, risk = risk, lambda = lambda, nu = nu)
}
## Compute expected value as optimal weight for risky asset
PUopt <- round(mean(x * U1DS(x = x, mu = 5, risk = 16, lambda = 0.5, nu = 1)) * 100, 2)
## Associated utility
U1MU <- U1(MUopt / 100, mu = 2, risk = 9, lambda = 0.5)
U1PU <- U1(PUopt / 100, mu = 2, risk = 9, lambda = 0.5)
```

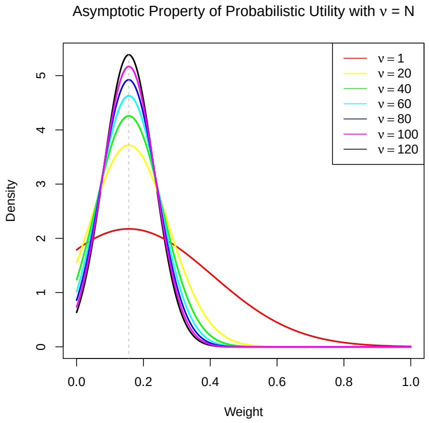
Motivation

Example: quadratic utility, one risky asset, II



Motivation

Example: quadratic utility, one risky asset, III



Markov Chain Monte Carlo

Overview

- Class of algorithms for sampling from a probability distribution; shape of density suffices.
- Purpose of MCMC is the numeric evaluation of multi-dimensional integrals, by (i) searching and (ii) evaluating the state space.
- The state space is searched by means of a Markov chain-type progression of the parameters.
- Evaluating proposed move (accepting/rejecting) ordinarily by Metropolis-Hastings algorithm.
- R resources: numerous R packages are available; see CRAN and task view 'Bayesian' for an annotated listing.
- Book resources: Gilks et al. (1995) and Brooks et al. (2011).

Markov Chain Monte Carlo

Hybrid Monte Carlo I

- Introduced by Duane et al. (1987) (see Neal (2011) for a more textbook-like exposition).
- Inclusion of an auxilliary momentum vector and taking the gradient of the target distribution into account.
- Purpose/aim:
  1. Moving through state space in larger steps.
  2. Autocorrelation in Markov Chains less pronounced compared to other approaches (thinning in principal not necessary).
  3. High acceptance rate, ideally all moves are accepted.
  4. Faster convergence to equilibrium distribution.

Markov Chain Monte Carlo

Hybrid Monte Carlo II

- Amending density by conjugate variables $\mathbf{p}$:

$$G(\mathbf{q},\mathbf{p}) \sim \exp\left(U(\mathbf{q}) - \frac{\mathbf{p}'\mathbf{p}}{2}\right) \tag{1}$$

- Algorithm: Starting from a pair $(\mathbf{q}_n, \mathbf{p}_n)$
  1. Sample $\boldsymbol{\eta}$ from standard normal.
  2. For a time interval T, integrate Hamiltonion equations:

$$\frac{\mathrm{d}p_i}{\mathrm{d}t} = -\frac{\delta U}{\delta p_i} \tag{2a}$$

$$\frac{\mathrm{d}q_i}{\mathrm{d}t} = p_i \tag{2b}$$

     together with the boundary constraints $\mathbf{p}(0) = \boldsymbol{\eta}$ and $\mathbf{q}(0) = \mathbf{q}_n$.
  3. Accept $\mathbf{q}_{n+1} = \mathbf{q}(T)$ with probability:

$$\beta = min(1, \exp\left(G(\mathbf{q}(T), \mathbf{p}(T)) - G(\mathbf{q}_n, \boldsymbol{\eta})\right)), \tag{3}$$

     else set $\mathbf{q}_{n+1} = \mathbf{q}_n$.

Markov Chain Monte Carlo I

Hybrid Monte Carlo III

See http://www.cs.utoronto.ca/~radford/GRIMS.html (adopted version)

```
hybridMC <- function(logDens, cState, eps, L, ...){
  q <- cState
  p <- rnorm(length(q), 0, 1)  ## independent standard normal variates
  cMom <- p
  ## Make a half step for momentum at the beginning
  p <- p + eps * grad(func = logDens, x = q, ...) / 2
  ## Alternate full steps for position and momentum
  for (i in 1:L){
    ## Make a full step for the position
    q <- q + eps * p
    ## Check lower bound
    lbidx <- which(q < 0)
    if(length(lbidx) > 0){
      q[lbidx] <- -q[lbidx]
      p[lbidx] <- -p[lbidx]
    }
    ## Check budget constraint
    qsum <- sum(q)
    q <- q / qsum
    ## Make a full step for the momentum, except at end of trajectory
    if (i!=L) p <- p + eps * grad(func = logDens, x = q, ...)
  }
  ## Make a half step for momentum at the end.
  p <- p + eps * grad(func = logDens, x = q, ...) / 2
  ## Negate momentum at end of trajectory to make the proposal symmetric
  p <- -p
```

Markov Chain Monte Carlo II

Hybrid Monte Carlo III

```
  ## Evaluate potential and kinetic energies at start and end of trajectory
  clogDens <- logDens(cState, ...)
  cK <- sum(cMom^2) / 2
  Hinit <- pexp(clogDens - cK)
  plogDens <- logDens(q, ...)
  pK <- sum(p^2) / 2
  Hprop <- pexp(plogDens - pK)
  delta <- Hprop - Hinit
  ## Accept or reject the state at end of trajectory, returning either
  ## the position at the end of the trajectory or the initial position
  apr <- min(1, exp(delta))
  ifelse(runif(1) < apr, return(q), return(cState))
}
## Quadratic Utility Funtion
U <- function(x, mu, Sigma, lambda = 0.5){
  c(lambda * t(x) %*% mu) - c((1 - lambda) * t(x) %*% Sigma %*% x)
}
## Log-density of quadratic utility
LUdens <- function(x, mu, Sigma, lambda = 0.5, nu){
  nu * U(x = x, mu = mu, Sigma = Sigma, lambda = lambda)
}
## Expected utility of Quadratic Utility Function
PUopt <- function(logDens, MCSteps, BurnIn, eps, L, mu, Sigma, lambda = 0.5, nu){
  J <- length(mu)
  MCMC <- matrix(NA, ncol = J, nrow = MCSteps)
  MCMC[1, ] <- rep(1/J, J)
  for(i in 2:MCSteps){
```

Markov Chain Monte Carlo III

Hybrid Monte Carlo III

```
    MCMC[i, ] <- hybridMC(logDens = logDens, cState =  MCMC[i - 1, ],
                          eps = epsf(eps), L = L, mu = mu, Sigma = Sigma,
                          lambda = lambda, nu = nu)
  }
  MCMC <- MCMC[-c(1:BurnIn), ]
  MCMC
}
## Maximization of Quadratic Utility Function
MUopt <- function(mu, Sigma, lambda){
    V <- (1 - lambda) * 2 * Sigma
    N <- ncol(Sigma)
    a1 <- rep(1, N)
    b1 <- 1
    a2 <- diag(N)
    b2 <- rep(0, N)
    Amat <- cbind(a1, a2)
    Bvec <- c(b1, b2)
    meq <- c(1, rep(0, N))
    opt <- solve.QP(Dmat = V, dvec = lambda * mu, Amat = Amat, bvec = Bvec, meq = meq)
    opt$solution
}
```

Comparative Simulation
Design

- Michaud-type simulation (see Michaud, 1989, 1998) as in Marschinski et al. (2007):
  1. Treat estimates of location and dispersion as true population parameters for a given sample.
  2. Obtain optimal 'true' MU allocations and hence utility.
  3. Draw K random samples of length L from these 'population' parameters and obtain MU and PU solutions.
  4. Compare distances of these K solutions with 'true' utility.
- Settings: Sample sizes (L) of 24, 30, 36, 48, 54, 60, 72, 84, 96, 108 and 120 observations; length of MC 250 (150 burn-in-periods) and K equals 100.
- Applied to end-of-month multi-asset data set contained in R package `FRAPO` (see Pfaff, 2012), sample period 2004:11 – 2011:11.

Comparative Simulation I
R Code

```
## Load packages
library(FRAPO)
library(MASS)
library(numDeriv)
library(parallel)
library(compiler)
enableJIT(3)
## Loading data and computing returns
data(MultiAsset)
Assets <- timeSeries(MultiAsset, charvec = rownames(MultiAsset))
R <- returns(Assets, method = "discrete", percentage = TRUE)
J <- ncol(R)
N <- nrow(R)
## Population moments, max util weights and utility
MuPop <- apply(R, 2, mean)
SigmaPop <- cov(R)
WeightsPop <- MUopt(m = MuPop, S = SigmaPop, lambda = 0.9)
UtilPop <- U(WeightsPop, mu = MuPop, Sigma = SigmaPop, lambda = 0.9)
## Parameters and initialising of simulation
Draws <- 100
Idx <- 1:Draws
Samples <- c(24, 30, 36, 48, 54, 60, 72, 84, 96, 108, 120)
LS <- length(Samples)
PU <- matrix(NA, ncol = LS, nrow = Draws)
MU <- matrix(NA, ncol = LS, nrow = Draws)
colnames(PU) <- colnames(MU) <- paste("S", Samples, sep = "")
```

Comparative Simulation II

R Code

```
PUW <- array(NA, dim = c(Draws, J, LS))
MUW <- array(NA, dim = c(Draws, J, LS))

## Parallel processing
cl <- makeCluster(3)
clusterExport(cl = cl, c("MUopt", "PUopt", "solve.QP", "U", "hybridMC", "grad", "LUdens"))

## Utility simulation: function for computing and evaluating MU and PU
Util <- function(x, MCSteps, BurnIn, eps, L, lambda, nu, MuPop, SigmaPop){
  J <- ncol(x)
  mu <- apply(x, 2, mean)
  sigma <- cov(x)
  ## Max Utility for sample weights, with population moments
  MUW <- MUopt(mu, sigma, lambda)
  MU <- U(MUW, MuPop, SigmaPop, lambda)
  ## Prob Utility for sample weights, with population moments
  MCMC <- PUopt(LUdens, MCSteps, BurnIn, eps, L, mu, sigma, lambda, nu)
  PUW <- colMeans(MCMC)
  PU <- U(PUW, MuPop, SigmaPop, lambda)
  list(U = c(MU, PU), PUW = PUW, MUW = MUW)
}
```

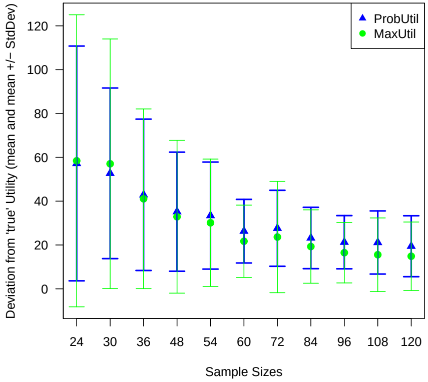
Comparative Simulation III

R Code

```
for(i in 1:LS){
  cat(paste("Computing for Sample Size", Samples[i], "\n"))
  SampleL <- Samples[i]
  ListData <- lapply(Idx, function(x) mvrnorm(n = SampleL, mu = MuPop, Sigma = SigmaPop))
  MuPu <- parLapplyLB(cl = cl, ListData, Util, MCSteps = 250, BurnIn = 150,
                      eps = 1 / SampleL, L = SampleL,
                      lambda = 0.9, nu = SampleL, MuPop = MuPop, SigmaPop = SigmaPop)
  MU[, i] <- unlist(lapply(MuPu, function(x) x$U[1]))
  PU[, i] <- unlist(lapply(MuPu, function(x) x$U[2]))
  PUW[, , i] <- matrix(unlist(lapply(MuPu, function(x) x$PUW)), ncol = ncol(R), nrow = Draws, byrow = TRUE)
  MUW[, , i] <- matrix(unlist(lapply(MuPu, function(x) x$MUW)), ncol = ncol(R), nrow = Draws, byrow = TRUE)
}
stopCluster(cl = cl)
## Computing distances
MUD <- (UtilPop - MU) / UtilPop * 100
PUD <- (UtilPop - PU) / UtilPop * 100
```

Comparative Simulation

Distances from true utility



Sensitivity with respect to ν and λ

Design

1. Sensitivity with respect to ν
   - Recall: $\nu = \rho N^{\gamma}$; vary ν: $\nu_1 = 1$, $\nu_2 = \sqrt{N}$, and $\nu_3 = N$.
   - Apply to multi-asset portfolio as above.
   - Comparison of weights with MU solution.
2. Sensitivity with respect to ν
   - Vary $\lambda \in [0, 1]$; *i.e.* moving along the efficient frontier from MVP to MRP.
   - Conduct analysis for complete and sub-sample of multi-asset data set.
   - Usage of $C = \sum_{j=1}^{J} \omega_j^2, \quad \in [1/J, 1]$ as concentration measure.

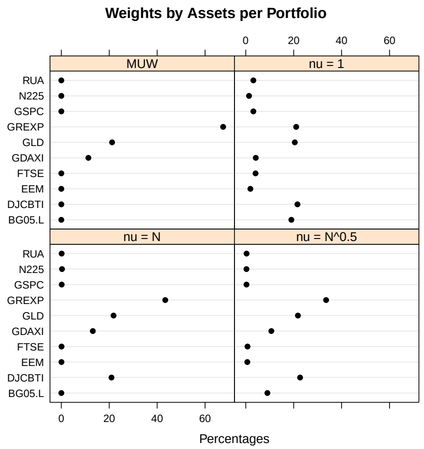
Sensitivity with respect to ν I

R Code

```r
## MU weights
MUW <- MUopt(MuPop, SigmaPop, lambda = 0.9) * 100
## PU weights for various nu values
cl <- makeCluster(3)
clusterExport(cl = cl, c("PUopt", "U", "hybridMC", "grad", "LUdens", "MuPop", "SigmaPop", "N"))
nu <- c(1, sqrt(N), N)
PUWs <- parLapplyLB(cl = cl, nu, function(i) PUopt(LUdens, MCSteps = 250,
                      BurnIn = 150, eps = 1 / N, L = N, mu = MuPop,
                      Sigma = SigmaPop, lambda = 0.9, nu = i)
                    )
stopCluster(cl = cl)
Wlist <- lapply(PUWs, function(i) colMeans(i))
PUW <- matrix(unlist(Wlist), ncol = 3, nrow = J) * 100
library(lattice)
latdat <- cbind(MUW, PUW)
colnames(latdat) <- c("MUW", "nu = 1", "nu = N^0.5", "nu = N")
rownames(latdat) <- colnames(R)
Assets <- factor(rep(rownames(latdat), ncol(latdat)), levels = sort(rownames(latdat)))
Port <- factor(rep(colnames(latdat), each = length(rownames(latdat))), levels = colnames(latdat))
Wdf <- data.frame(W = c(latdat), Port, Assets)
dotplot(Assets ~ W | Port, groups = Port, data = Wdf,
        xlab = "Percentages",
        main = "Weights by Assets per Portfolio",
        col = "black", pch = 19, as.table = TRUE)
```

Sensitivity with respect to ν

Results



Sensitivity with respect to λ I

R Code

```
## Sample estimates
MuAll <- apply(R, 2, mean)
SigmaAll <- cov(R)
MuSub <- apply(head(R, 48), 2, mean)
SigmaSub <- cov(head(R, 48))
## Initialising output matrices
ra <- seq(0.02, 0.98, by = 0.04)
## Function for parallel execution
RaSens <- function(x){
  MuwAll <- sum(MUopt(MuAll, SigmaAll, lambda = x)^2)
  MuwSub <- sum(MUopt(MuSub, SigmaSub, lambda = x)^2)
  MCMC <- PUopt(LUdens, MCSteps = 250, BurnIn = 150,
                eps = 1 / N, L = N,
                mu = MuAll, Sigma = SigmaAll,
                lambda = x, nu = N)
  PuwAll <- sum(colMeans(MCMC)^2)
  MCMC <- PUopt(LUdens, MCSteps = 250, BurnIn = 150,
                eps = 1 / 48, L = 48,
                mu = MuSub, Sigma = SigmaSub,
                lambda = x, nu = 48)
  PuwSub <- sum(colMeans(MCMC)^2)
  c(MuwAll, MuwSub, PuwAll, PuwSub)
}
## Parallel computation
cl <- makeCluster(3)
clusterExport(cl = cl, c("MUopt", "solve.QP", "PUopt", "U", "hybridMC", "grad", "LUdens", "MuAll",
                         "SigmaAll", "MuSub", "SigmaSub", "N"))
```

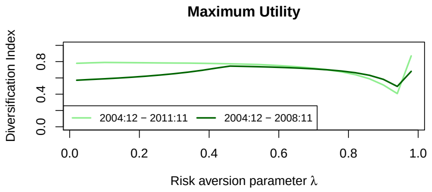
Sensitivity with respect to λ II

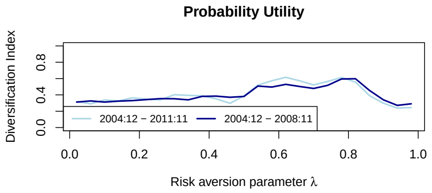
R Code

```
RASens <- parLapplyLB(cl = cl, ra, RaSens)
stopCluster(cl = cl)
## Convert to matrix
RAS <- matrix(unlist(RASens), ncol = 4, nrow = length(ra), byrow = TRUE)
par(mfrow = c(2, 1))
plot(RAS[, 1], RAS[, 2], col = "lightgreen", type = "l",
     ylim = c(0, 1), lwd = 2,
     xlab = expression(paste("Risk aversion parameter ", lambda)),
     ylab = "Diversification Index", main = "Maximum Utility")
lines(RAS[, 1], RAS[, 3], col = "darkgreen", lwd = 2)
legend("bottomleft", legend = c("2004:12 - 2011:11", "2004:12 - 2008:11"),
       col = c("lightgreen", "darkgreen"), lty = 1, lwd = 2, ncol = 2, cex = 0.8)
plot(RAS[, 1], RAS[, 4], col = "lightblue", type = "l",
     ylim = c(0, 1), lwd = 2,
     xlab = expression(paste("Risk aversion parameter ", lambda)),
     ylab = "Diversification Index", main = "Probability Utility")
lines(RAS[, 1], RAS[, 5], col = "darkblue", lwd = 2)
legend("bottomleft", legend = c("2004:12 - 2011:11", "2004:12 - 2008:11"),
       col = c("lightblue", "darkblue"), lty = 1, lwd = 2, ncol = 2, cex = 0.8)
```

Sensitivity with respect to λ

Results





Summary

- Reinterpretation of utility function as log-density.
- Optimal allocation defined as expected utility.
- High-dimensional density evaluated by means of HMC.
- Promising simulation results.
- However, some arbitrariness with respect to ν, but in general corner-solutions as in MU-settings can be circumvented.
- In a nutshell: Probabilistic utility approach is worth a second look.

Bibliography

Brooks, S., A. Gelman, G. Jones, and X.-L. Meng (Eds.) (2011). *Handbook of Markow Chain Monte Carlo*. Boca Raton, FL: Chapman & Hall / CRC.

Duane, S., A. Kennedy, B. Pendleton, and D. Roweth (1987). Hybrid monte carlo. *Physical Letters B195*, 216–222.

Gilks, W., S. Richardson, and D. Spiegelhalter (1995). *Markov Chain Monte Carlo in Practice*. Interdisciplinary Statistics. Boca Raton, FL.: Chapman & Hall / CRC.

Marschinski, R., P. Rossi, M. Tavoni, and F. Cocco (2007). Portfolio selection with probabilistic utility. *Annals of Operations Research 151*, 223–239.

Michaud, R. (1989). The markowitz optimization enigma: Is optimized optimal. *Financial Analyst Journal 45*, 31–42.

Michaud, R. (1998). *Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization and Asset Allocation*. New York: Oxford University Press.

Neal, R. (2011). *Handbook of Markov Chain Monte Carlo*, Chapter MCMC using Hamiltonian dynamics, pp. 113–162. Handbooks of Modern Statistical Methods. Boca Raton, FL: Chapman & Hall / CRC.

Pfaff, B. (2012). *Financial Risk Modelling and Portfolio Optimisation with R*. Chichester, UK: John Wiley & Sons Ltd.

Rossi, P., M. Tavoni, F. Cocco, and R. Marschinski (2002, November). Portfolio selection with probabilistic utility, bayesian statistics and markov chain monte carlo. *eprint arXiv arXiv:cond-mat/0211480*, 1–27. http://arxiv.org.