5 Scenario Optimization: Addressing Non-normality

5.1 Scenario Optimization

5.1.1 Foundations of Scenario Optimization

In the case of portfolio optimization, the uncertainty in the optimization process stems from the uncertainty of returns. One way to solve this problem (in the sense of addressing uncertainty, not estimation error) is to solve a very large-scale deterministic program instead, where a large number of scenarios try to capture randomness. For example, we can simulate 100,000 scenarios for four assets from the predictive distribution of portfolio returns. After the draws have been made, the uncertainty is removed and we are left with solving a deterministic problem. This procedure is called **scenario optimization**. We will see later in this chapter that for many objectives scenario optimization can be solved as a linear program. Key to successful scenario optimization is the quality of the sampled scenarios. In particular, scenarios must be

- **Representative** – Scenarios must offer a realistic description of the relevant problem and not induce estimation error.
- **Parsimonious** – Scenarios should use a relatively small number of samples to save computing time.
- **Arbitrage-free** – Scenarios should not allow the optimization algorithm to find highly attractive solutions that make no economic sense.

Scenario optimization that is based on only a few unrepresentative data might over-adjust and lead to an overly optimistic assessment of what could be achieved, while scenario optimization with a very large set of scenarios and assets might become computationally infeasible.

It is well-known that under normally distributed returns there is no need for scenario optimization, as the efficient set of solutions under arbitrary objective

functions would still coincide with the efficient set in a traditional mean-variance optimization. Otherwise complicated calculations become quite simple. As a first step we just find the mean-variance solutions. We then calculate the risk measure on the efficient set of portfolio solutions in a second step. For a given return expectation, the portfolio with the smallest value-at-risk (VaR) or lower partial moment will still be the portfolio with the minimum variance.

In order to deviate from the normality assumption, we have to ask ourselves a series of questions.

- Are returns non-normal?
- Are deviations from normality statistically significant?
- Are deviations stable (i.e., can we forecast them over time)?
- Will the non-normality vanish over time?

There is no general dogmatic answer to the questions above. At the asset class level, this is an empirical problem. However, modelers should also be aware of what will be lost if we discard the normality assumption. We lose portfolio aggregation as well as time aggregation (risk measures often do not have closed forms under non-elliptical distributions). Additionally, we need a new equilibrium model where skewness and kurtosis are also priced. On the instrument level, it is clear that nonlinear derivatives (options, collateralized debt obligations (CDOs), etc.) require the explicit modeling of non-normalities that have been deliberately engineered. We discuss a relevant problem within the set of exercises.

Let us now start with a visual inspection of two return series to illustrate the problem of non-normality. Figure 5.1 shows histogram (empirical frequency distribution), empirical cumulative frequency distribution (versus assumed normal distribution), and Q-Q plots (plots of empirical quantile versus hypothetical quantile of assumed distribution) for monthly returns on emerging market bonds (JPM.EMBI) and U. S. dollar returns versus those for the Japanese yen (USD.YEN).[1]

```
graphsheet()
par(mfrow=c(1,3))
hist(Dollar.Yen)
Normal <- rnorm(10000, mean(Dollar.Yen),
  sqrt(var(Dollar.Yen)))
cdf.compare(Dollar.Yen, Normal, cex=0.7)
ks.gof(Dollar.Yen, Normal)
qqnorm(Dollar.Yen)
qqline(Dollar.Yen)
```

It is straightforward to see that returns on emerging market bonds show negative skewness (too many large negative returns), while currency returns are

approximately normal. We can also use the Kolmogorov-Smirnov test (calculating how distant both cumulative distributions are) for a more formal assessment.

```
ks.gof(Dollar.Yen, Normal)

  Two-Sample Kolmogorov-Smirnov Test

data:  Dollar.Yen and Normal

ks = 0.0748, p-value = 0.2116
alternative hypothesis:
  cdf of Dollar.Yen does not equal the cdf of
  Normal for at least one sample point.
```

The high p-value (0.21) confirms that the empirical distribution is not significantly different from the normal distribution. We can also use the Kolmogorov-Smirnov test to check for multivariate normality. Note that individual marginal distributions could all be normally distributed, while the corresponding multivariate distribution still might not be normal. Under multivariate normality, we know that $\mathbf{d}_m^T \bar{\mathbf{\Omega}}^{-1} \mathbf{d}_m$ is distributed as $\chi^2(n)$, where $\mathbf{d}_m$ reflects the distance vector at time m (period returns minus mean return). All we need is to compare the cumulative distribution of $\mathbf{d}_m^T \bar{\mathbf{\Omega}}^{-1} \mathbf{d}_m$ with $\chi^2(n)$. This is a straightforward test for multivariate normality.

Even if period-by-period returns are non-normally distributed, it is most likely that multiperiod returns are (log) normally distributed: the Central Limit Theorem states that the product of independent and identical distributed variables (with finite variance) will approach log-normality after approximately 30 random drawings. We can check this using the built-in `bootstrap()` function to generate 36 month returns from the series of one month returns.

```
JPM.EMBI.36month <- bootstrap(JPM.EMBI,
  prod(1+sample(JPM.EMBI, 36)), 10000)$replicates
hist(JPM.EMBI.36month)
```

Figure 5.2 agrees with our intuition. It looks very much like a log-normal distribution, confirming our previous considerations that non-normality will tend to vanish as we move away from the very short time horizon.

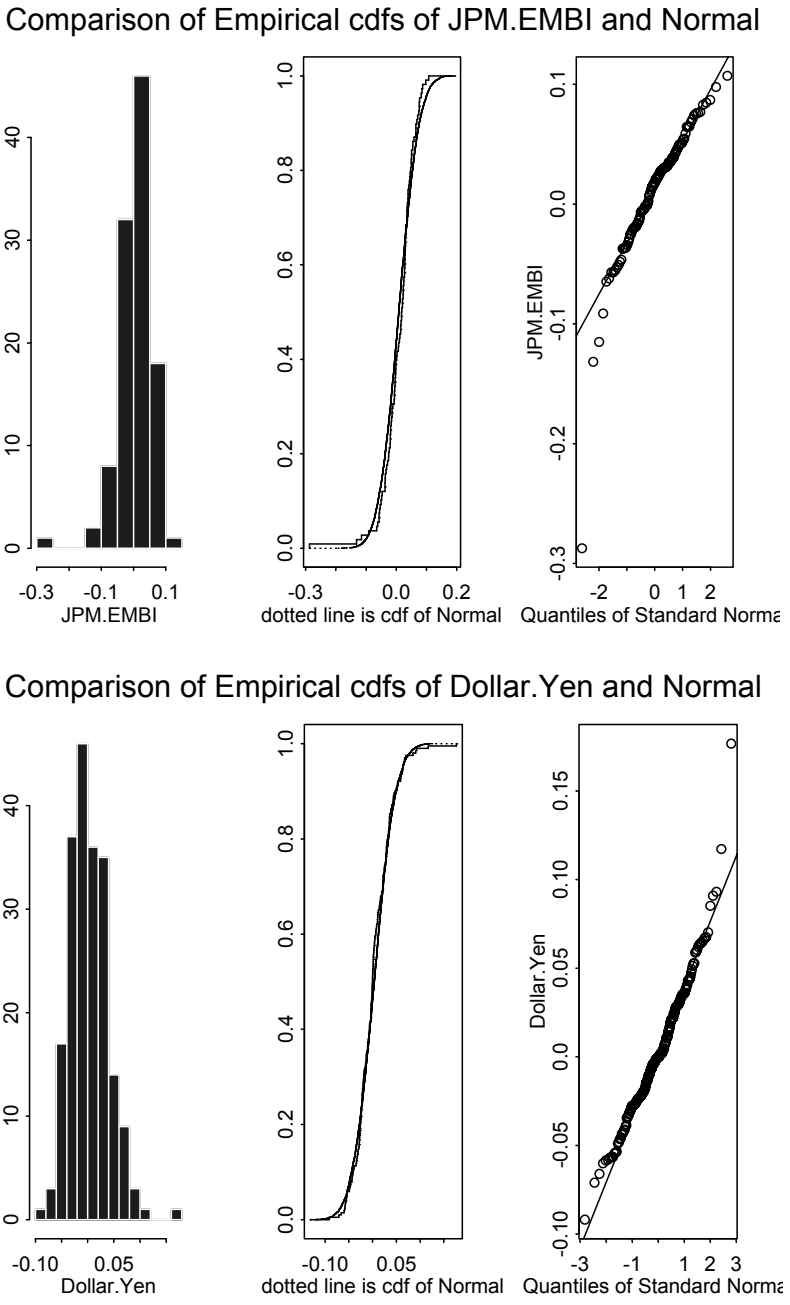

Figure 5.1 Visual Inspection of Asset Returns in S-PLUS

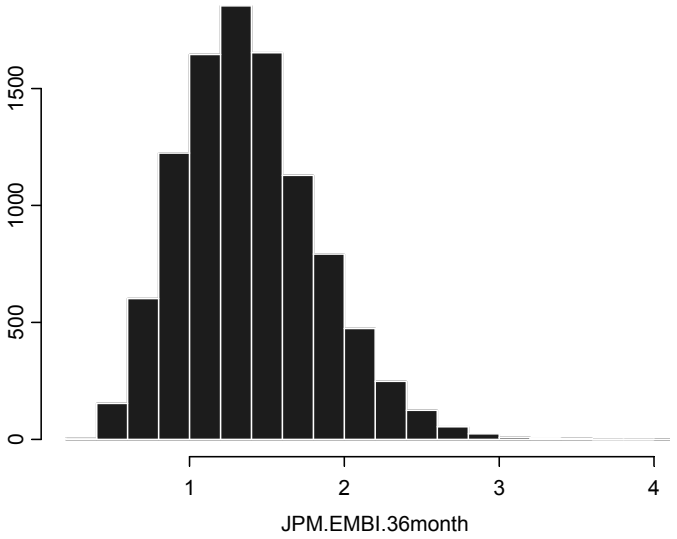

Figure 5.2 Multiperiod Returns for JPM.EMBI

5.1.2 Implied Returns and Arbitrary Preferences (Utilities) and Distributions

One problem with non-normal returns is that we lose the ability to back out implied returns using reversed optimization as seen in Chapter 1. What can we do to back out the implied returns for investors with different preferences under arbitrary return distributions?[2] Suppose our investor maximizes expected utility

$$E(U) = \sum_{s=1}^{m} \pi_s U\left(1 + \sum_{i=1}^{n} w_i r_{is}\right), \tag{5.1}$$

where we use the same notation as throughout the previous chapters. Expected utility is calculated as the average utility over m simulated scenarios. Each scenario is drawn with probability $\pi_s = \frac{1}{m}$. In our example, utility itself is defined as

$$U(1+r) = \begin{cases} \dfrac{(1+r)^{1-\gamma}}{1-\gamma}, & \gamma \geq 0 \\ \ln(1+r), & \gamma = 1, \end{cases} \tag{5.2}$$

where γ denotes the risk aversion coefficient. Note that any series of historic returns can be written as

$$r_{is} = c + \mu_i + \sigma_i z_{is}, \tag{5.3}$$

where we can isolate the degree of non-normality captured in the empirical distribution of z_{is} from our forward-looking assumptions on risk premiums (μ) as well as volatilities (σ). Applying (5.3) to the definition of benchmark returns $r_{bs} = \sum_{i=1}^{n} w_i r_{is}$, we get

$$r_{bs} = c + \mu_b + \sum_{i=1}^{n} w_i \sigma_i z_{is}. \tag{5.4}$$

We know from standard valuation theory that[3]

$$\pi_s^* = \pi_s \frac{U'(W_s)}{\sum_{s=1}^{m} \pi_s U'(W_s)}, \tag{5.5}$$

where π_s^* denotes the risk-neutral probabilities. The risk-neutral probability will be high in states where marginal utility is high (wealth is low). Hence, large weight is given to those states where wealth levels are depressed. Under the assumed utility function (5.3), we get

$$\pi_s^* = \pi_s \frac{(1+R_{bs})^{-\gamma}}{\sum_{s=1}^{m} \pi_s (1+R_{bs})^{-\gamma}}. \tag{5.6}$$

Risk-neutral probabilities equalize all expected returns, as they correct for risk via (5.5). Assuming our investor finds the current (benchmark) portfolio optimal, we hence know that he prices all assets according to

$$\sum_{s=1}^{m} \pi_s^* r_{is} = \sum_{s=1}^{m} \pi_s^* r_{bs}. \tag{5.7}$$

Inserting (5.3) and (5.4) into (5.7), we arrive at the implied return for the *i*-th asset,

$$\mu_i = \mu_b + \sum_{s=1}^{m} \pi_s^* \left(\sum_{i=1}^{n} w_i \sigma_i z_{is} - \sigma_i z_{is} \right). \tag{5.8}$$

Changing the risk aversion parameter will change the implied returns. The very risk-averse investor will require large compensations for assets that show considerable "tail risk."

5.1.3 Generation of Non-normally Distributed Scenarios

Suppose we want to generate a set of returns for future scenarios. Also assume we want to dismiss normality and generalize our simulation methodology in two main aspects. First, we aim to allow marginal distributions that could take any arbitrary form (i.e., they could follow a blend of a normal distribution plus an extreme value distribution for large losses, a mixture of normals, etc.). Second, we want to relax the modeling of dependence beyond the concept of correlation, as it is well-known that empirical distributions show tail dependence that is not explained by correlation alone. How can we glue arbitrary return distributions together and still maintain their correlation structure? How can we model tail dependence and still keep the same marginal distributions?

The answer to these questions is the concept of **copula functions**. Recall that a typical Monte Carlo simulation of random returns requires us to draw a uniform random number $u_s \sim uniform(0,1)$ and then invert the cumulative distribution function to arrive at a simulated return observation $r_{i,s} = F_{r_i}^{-1}(u_s)$. What do we do in a multivariate context? An n-dimensional copula is a multivariate cumulative distribution function with uniformly distributed marginals. Alternatively, we can think of it as a random vector of uniformly distributed variables that share a specified dependence structure,

$$C(u_1,\ldots,u_n) = prob\left(\tilde{u}_1 \le u_1,\ldots,\tilde{u}_n \le u_n\right). \tag{5.9}$$

As soon as we know the realization of the uniform random numbers $(u_1,\ldots,u_n)$, we can calculate the marginals from $F_{r_1}^{-1}(u_1),\cdots,F_{r_n}^{-1}(u_n)$. In general, we can say that if $F(r_1,\cdots,r_n)$ denotes a multivariate distribution function with continuous marginals, it will have a unique copula representation,

$$F(r_1,\cdots,r_n) = C\left(F_{r_1},\cdots,F_{r_n}\right). \tag{5.10}$$

We can hence separate the univariate margins and the multivariate dependence structure.[4] This proves to be very convenient in scenario generation. In what follows, we will not elaborate on how best to estimate the copula function (and the marginals) in (5.10). We rather work on the assumption that the marginals and copula are given.

In order to appreciate how scenario-based solutions (which will be presented later in this chapter) differ from a simple mean-variance approach, we use the copula approach to glue four mixtures of normals together, maintaining a specified correlation structure and modeling (symmetric) tail dependence according to a *t* copula. To simulate a *t* copula with υ degrees of freedom, we have to proceed according to the following steps.

1. Find the Cholesky decomposition $\mathbf{C}_{Cholesky}$ of the correlation matrix $\mathbf{C}$ (dimension: $n \times n$).
2. Draw a vector of n standard normals $\mathbf{u}$ and calculate $\mathbf{C}_{Cholesky}\mathbf{u}$. Alternatively, you might want to combine both steps and draw directly from a multivariate normal.
3. Draw from $s \sim \chi^2_\upsilon$ and multiply the result of the second step by $\sqrt{\upsilon}/\sqrt{s}$, i.e., calculate $\mathbf{x} = \mathbf{C}_{Cholesky}\mathbf{u}\frac{\sqrt{\upsilon}}{\sqrt{s}}$.
4. Each element of $\mathbf{x}$ $(x_1, \cdots, x_n)$ is inserted into the cumulative distribution function to arrive at uniformly distributed variables $u_i \sim t_\upsilon(x_i)$.
5. Repeat steps 2 to 4 many (m) times.

What looks like a complicated procedure can be performed in S-PLUS using a single line of code:

```
Corr <- matrix(c(1.0,0.8,0.2,0.2,
                 0.8,1.0,0.6,0.2,
                 0.2,0.6,1.0,0.2,
                 0.2,0.2,0.2,1.0), ncol=4, nrow=4)
m <- 100000
v <- 2
copula <- pt(rmvnorm(m, mean=rep(0,ncol(Corr)),
  cov=Corr)*sqrt(v)/sqrt(rchisq(m,v)),v)
```

Figure 5.3 and Figure 5.4 can be replicated with the following code:

```
x <- matrix(qnorm(copula), ncol=4)
graphsheet()
pairs(x, label=c("asset 1", "asset 2", "asset 3",
  "asset 4"))
xx <- rmvnorm(m, mean=rep(0,ncol(Corr)), cov=Corr)
graphsheet()
pairs(xx,label=c("asset 1", "asset 2", "asset 3",
  "asset 4"))
```

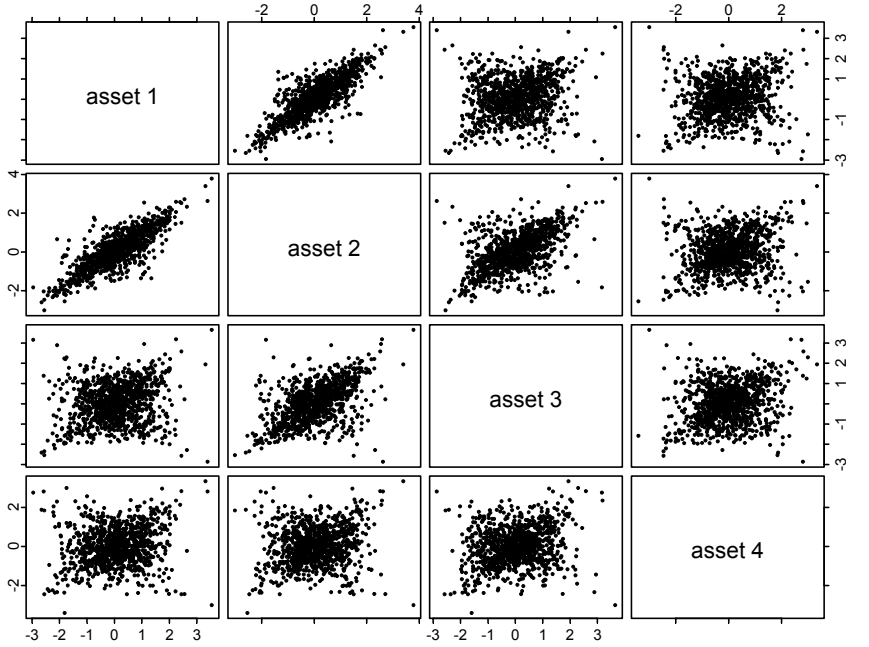

Figure 5.3 *t* copula with 2 Degrees of Freedom and Standard Normal Margins

Figure 5.5 is the result of the commands

```
graphsheet()
plot(x[,1],x[,2], xlab="asset 1", ylab="asset 2",
  pch=1)
points(xx[,1],xx[,2], pch=3)
```

Marginal distributions for each of the four assets are assumed to be drawn from a mixture of normals and are plotted in Figure 5.6.

```
asset.1 <- exp(c(rnorm(500,  0.05, 0.05),
  rnorm(9500, 0.05, 0.05)))-1
asset.2 <- exp(c(rnorm(500, -0.3, 0.01),
  rnorm(9500, 0.08, 0.05)))-1
asset.3 <- exp(c(rnorm(200, +0.4, 0.01),
  rnorm(9800, 0.1, 0.17)))-1
asset.4 <- exp(c(rnorm(500, -0.6, 0.1),
  rnorm(9500, 0.12, 0.25)))-1
```

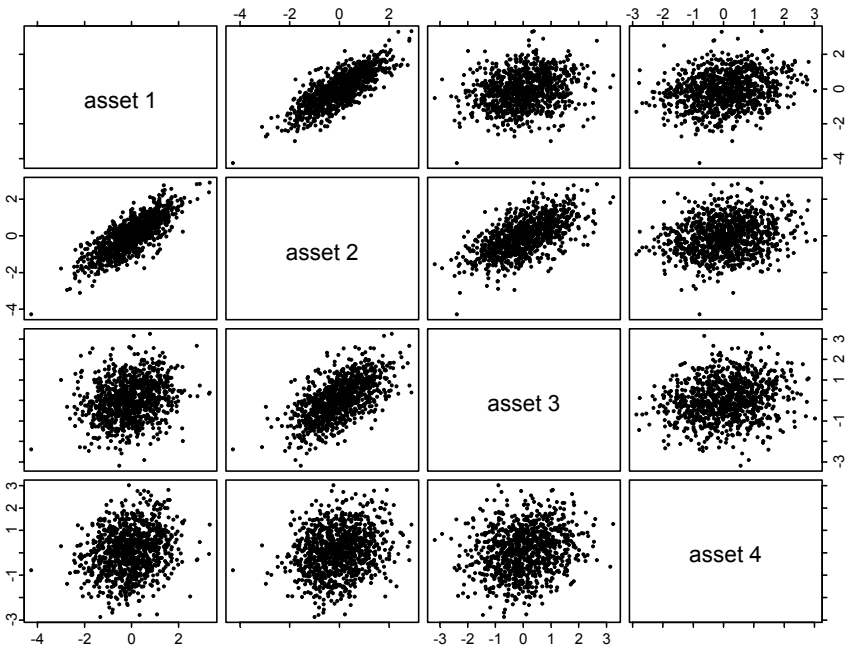

Figure 5.4 Multivariate Standard Normal

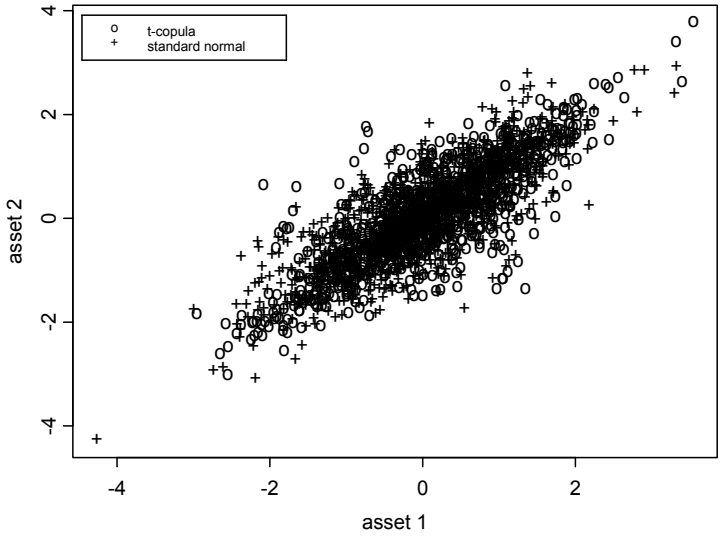

Figure 5.5 Scatterplot for Normal Distribution versus *t* copula

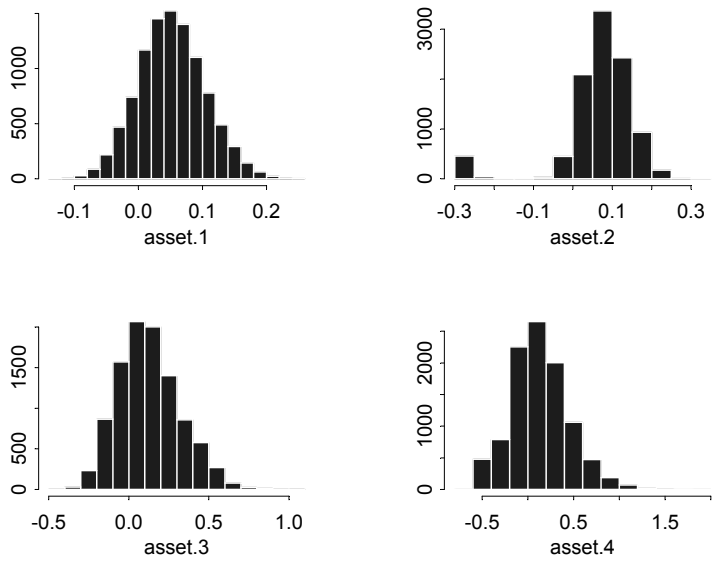

Figure 5.6 Marginal Distributions in Four-Asset Test Case

```
graphsheet()
par(mfrow=c(2,2))
hist(asset.1)
hist(asset.2)
hist(asset.3)
hist(asset.4)
```

Marginal distributions are glued together with the use of a *t* copula and stored in the scenario matrix $\mathbf{S}$. The necessary operations are

```
asset.1 <- matrix(quantile(asset.1, copula[,1]))
asset.2 <- matrix(quantile(asset.2, copula[,2]))
asset.3 <- matrix(quantile(asset.3, copula[,3]))
asset.4 <- matrix(quantile(asset.4, copula[,4]))
S <- cbind(asset.1, asset.2, asset.3, asset.4)
```

We can now code a scenario-based Markowitz optimization (Code 5.1), where portfolio variance is calculated using all scenarios for each weight allocation rather than by supplying a single covariance matrix.

```
MV.model <- function(S, mu.target)
{
  m <- nrow(S)
```

```
  n <- ncol(S)
  mu.bar <- apply(S, 2, mean)
  asset <- Set()
  period <- Set()
  i <- Element(set=asset)
  s <- Element(set=period)
  S <- Parameter(S, index=dprod(s,i))
  mu.bar <- Parameter(as.array(mu.bar), index=i)
  mu.target <- Parameter(mu.target, changeable=T)
  w <- Variable(index=i)
  r <- Variable(index=s)
  r[s] == Sum((S[s,i]-mu.bar[i])*w[i],i)
  risk <- Objective(type="minimize")
  risk ~ Sum(r[s]^2,s)/(m-1)
  Sum(mu.bar[i]*w[i],i) >= mu.target
  Sum(w[i],i) == 1
  w[i] >= 0
}

MV.portfolio <- function(S, mu.target)
{
  call(MV.model)
  MV.system <- System(MV.model, S, mu.target)
  solution <- solve(MV.system, trace=T)
  weight <-
     matrix(round(solution$variable$w$current,
     digit=5)*100, ncol=1)
  risk <- solution$objective
  return(weight,risk)
}

MV.frontier <- function(S, n.pf)
{
  call(MV.portfolio)
  Risk <- matrix(0, ncol=1, nrow=n.pf)
  Return <- matrix(0, ncol=1, nrow=n.pf)
  m <- nrow(S)
  mu.min <- min(apply(S,2,mean))
  mu.max <- max(apply(S, 2, mean))
  mu.range <- seq(mu.min, mu.max,
     (mu.max-mu.min)/(n.pf-1))
  x <- MV.portfolio(S, mu.target=mu.min)
  weight <- x$weight
  Risk[1,1] <- x$risk
  Return[1,1] <- mu.min
```

```
  for(i in 2:n.pf){
     x <- MV.portfolio(S, mu.target=mu.range[i])
     Risk[i,1] <- x$risk
     Return[i,1] <- mu.range[i]
     weight <- cbind(weight,x$weight)
  }
  graphsheet()
  par(mfrow=c(1,2))
  plot(Risk, Return, type="b")
  title("Mean - Variance Frontier")
  barplot(weight)
  title("Frontier Portfolios")
  list("optimal.weights" = weight)
}
```

Code 5.1 Mean-Variance Scenario Optimization

Typing `x <- MV.frontier(S, n.pf=10)` will trace out an efficient frontier with ten portfolios.

```
> x$optimal.weights
numeric matrix: 4 rows, 10 columns.
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
[1,] 97.058 88.816 71.458 52.225 32.993 13.760
[2,]  0.000  0.000  8.032 18.572 29.112 39.652
[3,]  2.942 10.403 17.268 23.561 29.854 36.148
[4,]  0.000  0.781  3.243  5.642  8.041 10.440

       [,7]   [,8]   [,9] [,10]
[1,]  0.000  0.000  0.000     0
[2,] 42.869 27.675 12.481     0
[3,] 44.131 56.363 68.596   100
[4,] 13.000 15.962 18.923     0
```

Figure 5.7 shows the solutions for ten return points along the efficient frontier. All portfolios look reasonably diversified. We can use these solutions as a reference point for the following scenario optimizations.

5.2 Mean Absolute Deviation

The first scenario-based alternative to Markowitz optimization is the **Mean Absolute Deviation** model (MAD).[5] It involves the minimization of the probability-weighted (where p_s denotes the probability of scenario s) sum of

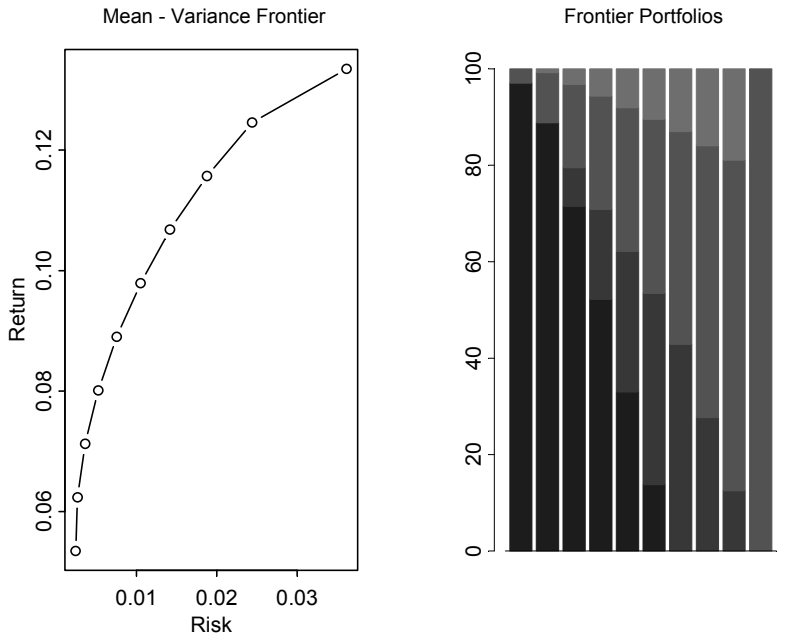

Figure 5.7 Mean-Variance Solutions

absolute deviations subject to the usual constraints. Risk is measured in the context of MAD as an absolute deviation from the mean rather than the squared deviation as in the case of variance.

$$\begin{aligned} &MAD = \sum_{s=1}^{m} p_s \left| \sum_{i=1}^{n} w_i \left(r_{i,s} - \bar{\mu}_i \right) \right| \\ &\sum_{i=1}^{n} w_i \bar{\mu}_i = \bar{\mu} \\ &\sum_{i=1}^{n} w_i = 1 \\ &w_i \geq 0. \end{aligned} \tag{5.11}$$

While the square function $(\cdot)^2$ penalizes larger deviations at an increasing rate, this is not the case with MAD. In fact, MAD implies that an additional unit of underperformance relative to the mean creates the same disutility no matter how big the loss already is. However, one advantage of MAD is that we can specify the costs of deviations above and below the mean differently, putting greater weight (costs) on underperformance rather than outperformance. If we work on simulated data ($p_s = \frac{1}{m}$) and denote the absolute deviation of the scenario

portfolio return from the average portfolio return by ad_s, we can transform (5.11) into a linear program.

$$\begin{aligned}
&\min_{ad_s, w_i} \ \tfrac{1}{m}\sum_{s=1}^{m} ad_s \\
&\sum_{i=1}^{n} w_i \left(r_{i,s} - \bar{\mu}_i \right) \le ad_s \\
&\sum_{i=1}^{n} w_i \left(r_{i,s} - \bar{\mu}_i \right) \ge -ad_s \\
&\sum_{i=1}^{n} w_i \mu_i = \bar{\mu} \\
&\sum_{i=1}^{n} w_i = 1 \\
&ad_s \ge 0 \\
&w_i \ge 0.
\end{aligned} \tag{5.12}$$

The MAD-based portfolio selection shown here offers a range of appealing properties versus variance-based models:

1. There is no need to calculate a covariance matrix because we use the scenario matrix, which can be constructed from time series of asset returns. However, this is only true if we rely on historical data for scenario generation; simulated scenarios from a parametric distribution have to be drawn using a covariance matrix.
2. Solving a linear program is much easier than mean–variance optimization. The number of constraints $(2m + 2$ in the case of MAD) depends on the number of scenarios, not the number of assets.
3. The upper bound on the number of assets in the optimal solution is related to the number of scenarios $(2m + 2$ in the case of MAD).

We leave the solution of (5.12) in NUOPT for S-PLUS as an exercise and use SIMPLE instead. Let us add one more layer of complexity by attaching different costs to upside and downside deviations, $\sum_{i=1}^{n} w_i \left(r_{i,s} - \bar{\mu}_i \right) \ge 0$ and $\sum_{i=1}^{n} w_i \left(r_{i,s} - \bar{\mu}_i \right) \ge 0$, respectively. The objective now becomes

$$\begin{aligned}
&\tfrac{1}{m}\sum_{s=1}^{m} \left(c^{+} ad_s^{+} + c^{-} ad_s^{-} \right) \\
&ad_s^{+} = \begin{cases} ad_s & \text{if } ad_s > 0 \\ 0 & \text{else} \end{cases} \\
&ad_s^{-} = \begin{cases} ad_s & \text{if } ad_s < 0 \\ 0 & \text{else.} \end{cases}
\end{aligned} \tag{5.13}$$

The corresponding code is given in Code 5.2.

```
MAD.model <- function(S, cost.up, cost.dn,
     mu.target)
{
  m <- nrow(S)
  n <- ncol(S)
  mu.bar <- apply(S, 2, mean)
  asset <- Set()
  period <- Set()
  i <- Element(set=asset)
  s <- Element(set=period)
  S <- Parameter(S, index=dprod(s,i))
  mu.bar <- Parameter(as.array(mu.bar), index=i)
  mu.target <- Parameter(mu.target, changeable=T)
  cost.up <- Parameter(cost.up, changeable=T)
  cost.dn <- Parameter(cost.dn, changeable=T)
  w <- Variable(index=i)
  up <- Variable(index=s)
  dn <- Variable(index=s)
  up[s] >= 0
  dn[s] >= 0
  up[s]-dn[s] == Sum((S[s,i]-mu.bar[i])*w[i],i)
  risk <- Objective(type="minimize")
  risk ~ Sum((cost.up*up[s]+cost.dn*dn[s]),s)/(m-1)
  Sum(mu.bar[i]*w[i],i) >= mu.target
  Sum(w[i],i) == 1
  w[i] >= 0
}

MAD.portfolio <- function(S, cost.up, cost.dn,
  mu.target)
{
  call(MAD.model)
  MAD.system <- System(MAD.model, S, cost.up,
     cost.dn, mu.target)
  solution <- solve(MAD.system, trace=T)
  weight <-
     matrix(round(solution$variable$w$current,
     digit=5)*100, ncol=1)
  risk <- solution$objective
  return(weight,risk)
}
```

```
MAD.frontier <- function(S, cost.up, cost.dn, n.pf)
{
  call(MAD.portfolio)
  Risk <- matrix(0, ncol=1, nrow=n.pf)
  Return <- matrix(0, ncol=1, nrow=n.pf)
  m <- nrow(S)
  mu.min <- min(apply(S,2,mean))
  mu.max <- max(apply(S, 2, mean))
  mu.range <- seq(mu.min, mu.max,
     (mu.max-mu.min)/(n.pf-1))
  x <- MAD.portfolio(S, cost.up, cost.dn,
     mu.target=mu.min)
  weight <- x$weight
  Risk[1,1] <- x$risk
  Return[1,1] <- mu.min
  for(i in 2:n.pf){
     x <- MAD.portfolio(S, cost.up, cost.dn,
        mu.target=mu.range[i])
     Risk[i,1] <- x$risk
     Return[i,1] <- mu.range[i]
     weight <- cbind(weight,x$weight)
  }
  graphsheet()
  par(mfrow=c(1,2))
  plot(Risk, Return, type="b")
  title("Mean - Mean Absolute Deviation Frontier")
  barplot(weight)

  title("Frontier Portfolios")
  list("optimal.weights" = weight)
}
```

Code 5.2 Scenario Optimization Using Mean Absolute Deviation

Hence, typing `MAD.frontier(S,cost.up=1,cost.dn=1,n.pf=10)` will trace out an efficient frontier with ten portfolios, as shown in Figure 5.8.

```
> x$optimal.weights
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
[1,] 94.733 88.889 62.241 32.043  0.012  0.000
[2,]  0.005  0.000 20.447 45.543 73.145 57.860
[3,]  5.261 11.109 15.002 17.091 18.986 30.461
[4,]  0.000  0.002  2.311  5.323  7.857 11.679
```

```
        [,7]   [,8]   [,9]  [,10]
[1,]   0.009  0.000  0.000     0
[2,]  42.540 27.174 11.857     0
[3,]  41.901 52.826 64.188   100
[4,]  15.549 20.000 23.955     0
```

Portfolios constructed with a symmetric understanding of Mean Absolute Deviation (up and down costs of deviations equal one) are close to mean-variance solutions. This is not surprising, as variance is also a symmetric measure of investment risk.

5.3 Semi-variance and Generalized Semi-variance Optimization

5.3.1 Properties of Semi-variance

Mean-variance-based portfolio construction has always suffered from the implicit assumption of normality that dictated that risk be measured as the variance of returns. One of the earliest alternative risk measures is **semi-variance**. While variance uses all return realizations, semi-variance utilizes only those returns that either fall below the average return (lower semi-variance, sv^-) or above the average return (upper semi-variance, sv^+):

$$\begin{aligned} sv^- &= \frac{1}{m}\sum_{s=1}^{m}(r_s-\mu)^2\,\delta_s \\ sv^+ &= \frac{1}{m}\sum_{s=1}^{m}(r_s-\mu)^2\,(1-\delta_s) \\ \delta_s &= \begin{cases} 0 & r_s > \bar{\mu} \\ 1 & r_s \le \bar{\mu}. \end{cases} \end{aligned} \tag{5.14}$$

We can combine lower and upper semi-variances to arrive at variance again. To see this, note that if $\delta_s = 1$, it must follow that $1-\delta_s = 0$ and vice versa.

$$\begin{aligned} sv^- + sv^+ &= \frac{1}{m}\sum_{s=1}^{m}(r_s-\mu)^2\,\delta_s + \frac{1}{m}\sum_{s=1}^{m}(r_s-\mu)^2\,(1-\delta_s) \\ &= \frac{1}{m}\sum_{s=1}^{m}(r_s-\mu)^2 \\ &= \sigma^2. \end{aligned} \tag{5.15}$$

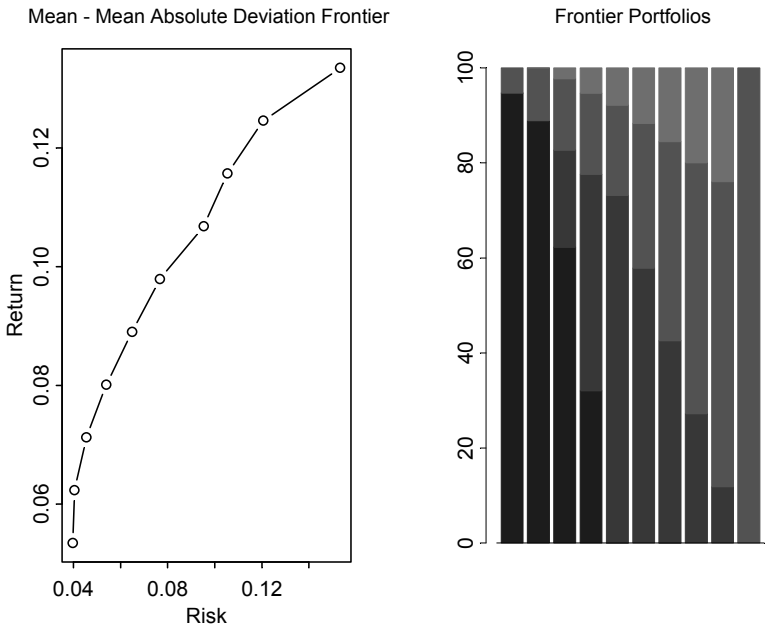

Figure 5.8 Mean Absolute Deviation Frontier

In the case of symmetry (for every return deviation below the mean, there is an equal return deviation above the mean), we get $sv^{-} = sv^{+}$ and

$$2sv = \sigma^2. \tag{5.16}$$

Equation (5.16) often serves as a simple check for symmetry. If the variance is roughly twice the semi-variance, the distribution is close to symmetric. (Symmetry does not imply normality, as a distribution might be symmetric but still exhibit fat tails.) For many assets, we can regress the difference between the variance and twice the semi-variance against a constant to see whether deviations are statistically different from zero:

$$\left(2sv_n - \sigma^2\right) = \alpha + \varepsilon_n. \tag{5.17}$$

Alternatively, we can test whether skewness is persistent across time. Suppose we have observations of returns for a number of time series. We can split the observations into two subperiods and test for persistence in deviations from symmetry,

$$\left(2sv - \sigma^2\right)_{t+1} = a + b\left(2sv - \sigma^2\right)_t + e_{t+1}. \tag{5.18}$$

Persistence is indicated by a significant b and a high R^2.

5.3.2 A General Semi-variance Model

Traditionally, semi-variance optimization has centered around lower semi-variance. However, there is no reason not to merge upper and lower semi-variances into a combined risk measure. This allows us every flexibility in expressing a mixture of risk-averse and risk-seeking behaviors.

The proposed measure uses a weighted linear combination of upper and lower semi-variances and is hence called the **weighted semi-variance model**[6] (see Figure 5.9):

$$sv_{weighted} = \omega sv^{-} + (1-\omega)sv^{+}. \tag{5.19}$$

Note that the range of ω is restricted to lie between zero and one. We can plot the new penalty function (5.19) for various weights.

```
deviation.from.mean.return <- seq(-60, 60, 1)
l.sv <- ifelse(deviation.from.mean.return <= 0,
  deviation.from.mean.return^2, 0)
u.sv <- ifelse(deviation.from.mean.return > 0,
  deviation.from.mean.return^2, 0)
V <- deviation.from.mean.return^2
DB <- 2*ifelse(deviation.from.mean.return <= 0,
  0.75*deviation.from.mean.return^2,
  0.25*deviation.from.mean.return^2)
UB <- 2*ifelse(deviation.from.mean.return <= 0,
  0.25*deviation.from.mean.return^2,
  0.75*deviation.from.mean.return^2)
graphsheet()
par(mfrow=c(2,2))
plot(deviation.from.mean.return, l.sv, type="l",
  ylab="penalty")
title("Lower semi-variance")

plot(deviation.from.mean.return, DB, type="l",
  ylab="penalty")
title("75% weight on lower semi-variance")
plot(deviation.from.mean.return, V, type="l",
  ylab="penalty")
title("Variance")
plot(deviation.from.mean.return, UB, type="l",
```

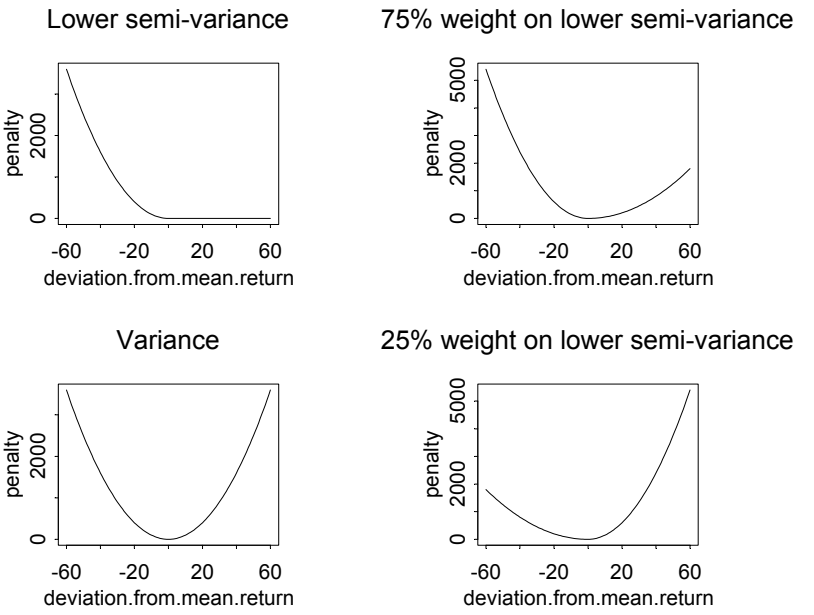

Figure 5.9 Weighted Semi-variance Measure

```
  ylab="penalty")
title("25% weight on lower semi-variance")
```

The weighted semi-variance portfolio optimization model becomes

$$\text{minimize}$$
$$\omega\left[\frac{1}{m}\sum_{s=1}^{m}\left(r_{pf,s}-\bar{\mu}\right)^2\delta_s\right]+(1-\omega)\left[\frac{1}{m}\sum_{s=1}^{m}\left(r_{pf,s}-\bar{\mu}\right)^2(1-\delta_s)\right]$$
$$\begin{aligned}
\sum_{i=1}^{n} w_i r_{i,s} &= r_{pf,s} \\
\sum_{i=1}^{n} w_i \mu_i &= \bar{\mu} \\
\sum_{i=1}^{n} w_i &= 1 \\
\delta_s &= \begin{cases} 0 & r_{pf,s} > \bar{\mu} \\ 1 & r_{pf,s} \le \bar{\mu} \end{cases} \\
w_i &\ge 0,
\end{aligned} \tag{5.20}$$

where $r_{pf,s}$ denotes the portfolio return in scenario s. Code 5.3 illustrates how the model given in (5.20) can be translated into NUOPT for S-PLUS.

```
WSV.model <- function(S, mu.target,
     downside.weight)
{
  if(downside.weight < 0 | downside.weight > 1)
     stop("downside weight must range between
        0 and 1")
  m <- nrow(S)
  n <- ncol(S)
  mu.bar <- apply(S, 2, mean)
  asset <- Set()
  period <- Set()
  i <- Element(set=asset)
  s <- Element(set=period)
  S <- Parameter(S, index=dprod(s,i))
  mu.bar <- Parameter(as.array(mu.bar), index=i)
  mu.target <- Parameter(as.numeric(mu.target),
     changeable=T)
  dw <- Parameter(downside.weight, changeable=T)
  w <- Variable(index=i)
  up <- Variable(index=s)
  dn <- Variable(index=s)
  up[s] >= 0
  dn[s] >= 0
  up[s]-dn[s] == Sum((S[s,i]-mu.bar[i])*w[i],i)
  risk <- Objective(type="minimize")
  risk ~ Sum(dw*dn[s]^2+(1-dw)*up[s]^2,s)/(m-1)
  Sum(mu.bar[i]*w[i],i) == mu.target
  Sum(w[i],i) == 1
  w[i] >= 0
}

WSV.portfolio <- function(S, mu.target,
  downside.weight)
{
  call(WSV.model)
  WSV.system <- System(WSV.model, S, mu.target,
     downside.weight)
  solution <- solve(WSV.system, trace=T)
  weight <-
     matrix(round(solution$variable$w$current,
     digit=5)*100, ncol=1)
```

```
  risk <- solution$objective
  return(weight,risk)
}

WSV.frontier <- function(S, n.pf, downside.weight)
{
  call(WSV.portfolio)
  Risk <- matrix(0, ncol=1, nrow=n.pf)
  Return <- matrix(0, ncol=1, nrow=n.pf)
  m <- nrow(S)
  mu.min <- min(apply(S, 2, mean))
  x <- WSV.portfolio(S, mu.target=mu.min,
     downside.weight)
  mu.max <- max(apply(S, 2, mean))
  mu.range <- seq(mu.min, mu.max,
     (mu.max-mu.min)/(n.pf-1))
  weight <- x$weight
  Risk[1,1] <- x$risk
  Return[1,1] <- mu.min

  for(i in 2:n.pf)
  {
     x <- WSV.portfolio(S, mu.target=mu.range[i],
        downside.weight)
     Risk[i,1] <- x$risk
     Return[i,1] <- mu.range[i]
     weight <- cbind(weight,x$weight)
  }

  graphsheet()
  par(mfrow=c(1,2))
  plot(Risk, Return, type="b")
  title("Mean - Weighted Semi-variance Frontier")
  barplot(weight)
  title("Frontier Portfolios")
  litle("optimal.weights"=weight)
}
```

Code 5.3 Weighted Semi-variance Model

As usual, we run an example optimization with a simulated data set:

```
> x <- WSV.frontier(S, n.pf=10,
     downside.weight=0.8)
```

```
> x$optimal.weights
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
[1,]    100 88.825 77.449 66.080 50.272 34.251
[2,]      0  0.000  0.000  0.019  5.985 12.241
[3,]      0 10.483 19.010 27.736 35.148 42.527
[4,]      0  0.693  3.542  6.166  8.595 10.981

       [,7]   [,8]   [,9] [,10]
[1,] 18.080  1.977  0.000     0
[2,] 18.696 25.061 12.531     0
[3,] 49.850 57.192 68.953   100
[4,] 13.374 15.771 18.516     0
```

Weighted semi-variance solutions invest more cautiously in asset 2 due to the obvious non-normality in its returns. Although the flexibility of the weighted semi-variance model is appealing, little guidance can be given on how to weight upper and lower semi-variances. For most practical applications, investors will hence stick with the lower semi-variance model. Figure 5.10 illustrates the weighted semi-variance frontier.

5.4 Probability-Based Risk/Return Measures

5.4.1 Shortfall Probability, Lower Partial Moment, and Value-at-Risk

Regulatory pressures (bankruptcy/default occurs if wealth falls below a liability threshold) as well as investment intuition (probability statements seem to be easier to understand than volatility numbers) often guide investors towards shortfall probability as their preferred measure of risk.[7] We start with the general observation that uncertain investment returns can be decomposed into a threshold return (γ) plus an upside measure, expressed as $\max[r-\gamma,0]$ which is either positive or zero, minus a downside risk measure, denoted by $\max[\gamma-r,0]$ which is also either positive or zero. In combination, we get

$$r=\gamma+\max[r-\gamma,0]-\max[\gamma-r,0]. \tag{5.21}$$

Measures that focus on the downside of a return distribution are called **lower partial moments**, while measures that focus on the upside are called **upper partial moments**. If return distributions become non-normal, risk measures that capture non-normality become attractive. We have already discussed the special case of $\gamma=\bar{\mu}$, in which we distinguished between upper and lower

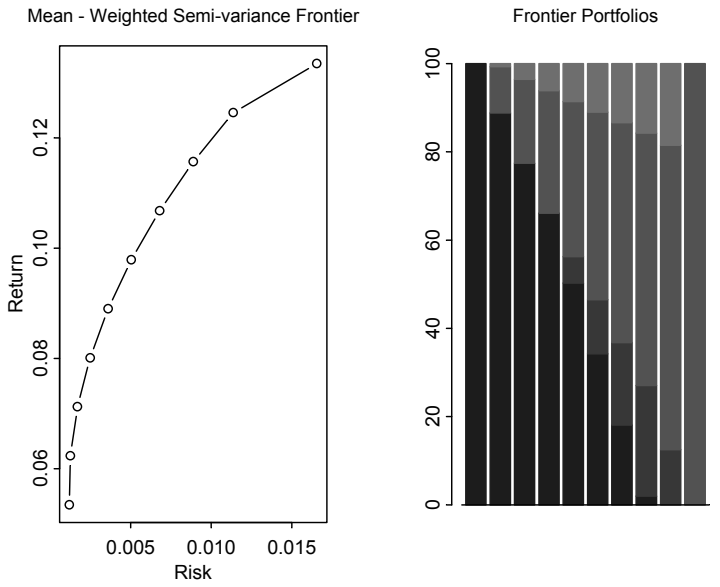

Figure 5.10 Weighted Semi-variance Frontier

semi-variances. Lower partial moments characterize the moments of a return distribution below the specified threshold return. In general, we define the lower partial moment of degree k in its discrete form (working on realized return scenarios rather than on a continuous distribution) as

$$\begin{aligned} lpm_{\gamma}^{k} &= \tfrac{1}{m}\sum\nolimits_{s=1}^{m}(r_s-\gamma)^k\,\delta_s, \\ \delta_s &= \begin{cases} 0, & r_s > \gamma \\ 1, & r_s \le \gamma. \end{cases} \end{aligned} \tag{5.22}$$

Again we use the same notation as in the previous chapter, where δ_s denotes an integer variable that assumes either one or zero. Effectively, δ_s will decide which observations enter the calculation on a go/no-go basis and hence can be modeled using integer variables. Apart from the threshold level, we also control the choice of the moment parameter k. For $k=0,$ we get the shortfall probability, for $k=1$ we get the average shortfall, and for $k=2,3,4,\cdots,$ we find shortfall variance, skewness, kurtosis, etc. In this section, we will focus on $k=0,$

$$lpm_{\gamma}^{0} = \tfrac{1}{m}\sum\nolimits_{s=1}^{m}(r_s-\gamma)^0\,\delta_s = \tfrac{1}{m}\sum\nolimits_{s=1}^{m}\delta_s,$$

because shortfall probability is closely related to value-at-risk. Note that as value-at-risk denotes the maximum loss (to be calculated) likely to occur in normal circumstances (i.e., 95% of all times, with a significance level of $\alpha = 5\%$),

$$VaR = F^{-1}(\alpha),$$

while the shortfall probability denotes the probability (to be calculated) that the loss will fall below a prespecified loss of amount γ:

$$lpm_{\gamma}^{0} = prob(r \leq \gamma).$$

While we fix probability in value-at-risk calculations, we fix our loss threshold in the calculation of shortfall probability. Hence both measures coincide if we set $\gamma = VaR$:

$$VaR = F^{-1}\left(lpm_{\gamma}^{0}\right).$$

In short, the value-at-risk at a significance level of α denotes a loss with shortfall probability α.[8] After this short digression on value-at-risk, shortfall probability, and lower partial moments, we proceed with the implementation of shortfall probability in NUOPT for S-PLUS. We focus on the calculation of shortfall probability with the use of scenarios, in which case the minimization of shortfall probabilities requires the use of integer variables.

5.4.2 Portfolio Construction and Shortfall Probability

We focus on an investor who aims to minimize shortfall risk (relative to a return threshold) subject to a specified return target. Equations (5.23) and (5.24) provide the appropriate switches,

$$\sum_{i=1}^{n} w_i r_{i,s} - \gamma \geq e - \delta_s E, \tag{5.23}$$

$$\sum_{i=1}^{n} w_i r_{i,s} - \gamma \leq (1 - \delta_s) E, \tag{5.24}$$

where e denotes a very small number and E represents a very large number. Each time the portfolio return is higher than the threshold return, $\delta_s = 0$ will simultaneously satisfy (5.23) and (5.24). Note that $\delta_s = 1$ will only satisfy (5.23). The portfolio construction problem becomes

$$
\begin{aligned}
&\text{minimize } \tfrac{1}{m}\sum_{s=1}^{m}\delta_s \\
&\sum_{i=1}^{n} w_i \mu_i = \bar{\mu} \\
&\sum_{i=1}^{n} w_i = 1 \\
&\sum_{i=1}^{n} w_i r_{i,s} - \gamma \geq e - \delta_s E \\
&\sum_{i=1}^{n} w_i r_{i,s} - \gamma \leq (1-\delta_s) E \\
&w_i \geq 0 \\
&\delta_s \in \{0,1\}.
\end{aligned}
\tag{5.25}
$$

The system (5.25) can also be brought into S-PLUS code (Code 5.4):

```
SF.model <- function(S, mu.target, mu.threshold)
{
  m <- nrow(S)
  n <- ncol(S)
  mu.bar <- apply(S, 2, mean)
  names(mu.bar) <- NULL
  asset <- Set()
  period <- Set()
  i <- Element(set=asset)
  s <- Element(set=period)
  S <- Parameter(S, index=dprod(s,i))
  mu.bar <- Parameter(as.array(mu.bar), index=i)
  mu.target <- Parameter(mu.target, changeable=T)
  mu.threshold <- Parameter(mu.threshold,
     changeable=T)
  w <- Variable(index=i)
  dummy <- IntegerVariable(index=s, type=binary)
  Sum(S[s,i]*w[i],i)-mu.threshold <=
     (1-dummy[s])*10
  Sum(S[s,i]*w[i],i)-mu.threshold >=
      -1*dummy[s]*10+0.000001
  risk <- Objective(type="minimize")
  risk ~ 1/m*Sum(dummy[s],s)
  Sum(mu.bar[i]*w[i],i) >= mu.target
  Sum(w[i],i) == 1
  w[i] >= 0
}
```

```
SF.portfolio <- function(S, mu.target,
     mu.threshold)
{
  SF.system <- System(SF.model, S, mu.target,
     mu.threshold)
  nuopt.options(maxitn=1000)
  solution <- solve(SF.system, risk, trace=T)
  weight <-
     matrix(round(solution$variable$w$current,
     digit=5)*100, ncol=1)
  risk <- solution$objective
  return(weight,risk)
}

SF.frontier <- function(S, mu.threshold, n.pf)
{
  call(SF.portfolio)
  Risk <- matrix(0, ncol=1, nrow=n.pf)
  Return <- matrix(0, ncol=1, nrow=n.pf)
  m <- nrow(S)
  n <- ncol(S)
  mu.min <- min(apply(S, 2, mean))
  x <- SF.portfolio(S, mu.target=mu.min,
     mu.threshold)
  mu.min <- t(x$weight) %*% apply(S, 2, mean)/100
  weight <- x$weight
  Risk[1,1] <- c(x$risk[2])


  Return[1,1] <- mu.min
  mu.max <- max(apply(S, 2, mean))
  mu.range <- seq(mu.min, mu.max,
     (mu.max-mu.min)/(n.pf-1))
  for(i in 2:n.pf){
     x <- SF.portfolio(S, mu.range[i],
     mu.threshold)
     Risk[i,1] <- x$risk[2]
     Return[i,1] <- mu.range[i]
     weight <- cbind(weight,x$weight)
  }
  graphsheet()
  par(mfrow=c(1,2))
  plot(Risk, Return, type="b")
  title("Mean - Shortfall Frontier")
  barplot(weight)
```

```
  title("Frontier Portfolios")
  list("optimal.weights" = weight)
}
```

Code 5.4 Shortfall Efficient Model

```
> x <- SF.frontier(S, mu.threshold, n.pf=50)
> x$optimal.weights
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
[1,]  0.000  0.000  0.000  0.000  0.063  0.000
[2,] 77.436 68.604 58.863 49.981 40.926 27.247
[3,] 16.393 24.086 25.763 32.667 39.386 50.680
[4,]  6.172  7.310 15.375 17.352 19.625 22.073

       [,7]   [,8]   [,9] [,10]
[1,]  1.467  1.710  0.000     0
[2,] 21.324 11.716  4.907     0
[3,] 54.751 61.965 64.596   100
[4,] 22.458 24.609 30.497     0
```

The inspection of the left part of Figure 5.11 is disappointing. However, as VaR is a nonconvex function with respect to portfolio weights (and hence possesses many local minima), it is not surprising that standard optimization techniques will not always find the optimal solution. After all, heuristics are needed if objective functions are nonconvex. This difficulty in finding optimal portfolios when using VaR as a risk measure in scenario optimization is one of the major obstacles to its use. It is not only inconvenient but also directly related to its theoretical deficiencies (i.e., its lack of subadditivity; see Section 5.6).

5.4.3 Probability of Outperformance

Many portfolio managers and plan sponsors are given performance objectives. Hence their interest is to outperform their given investment targets. The problem of maximizing the probability of outperformance can be written as

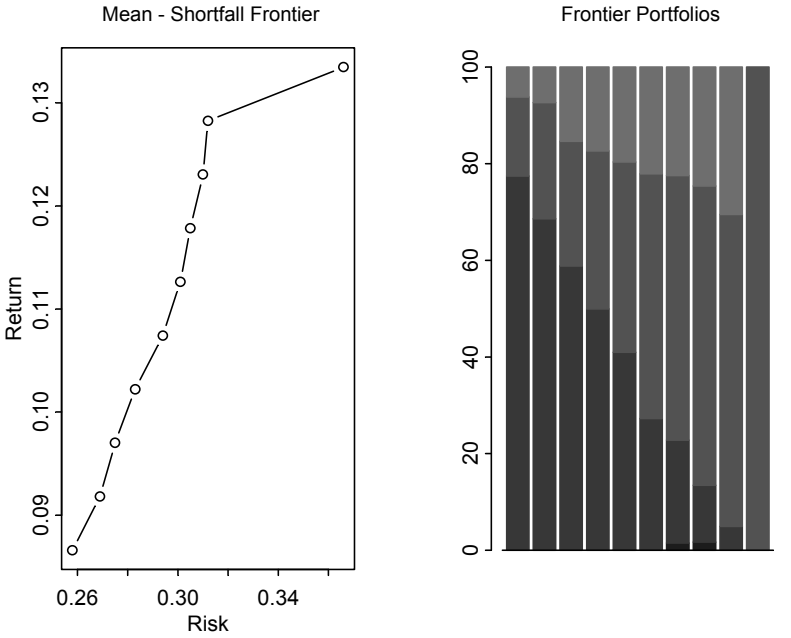

Figure 5.11 Shortfall Efficient Portfolios

$$
\begin{aligned}
&\text{maximize } 1-\tfrac{1}{m}\sum_{s=1}^{m}\delta_s \\
&\sum_{i=1}^{n} w_i = 1 \\
&\sum_{i=1}^{n} w_i r_{i,s} - \gamma \geq e - \delta_s E \\
&\sum_{i=1}^{n} w_i r_{i,s} - \gamma \leq (1-\delta_s) E \\
&w_i \geq 0,\ \delta_s \in \{0,1\}.
\end{aligned}
\tag{5.26}
$$

We leave the implementation of (5.26) to the reader. Do you think this investment objective makes sense?

5.5 Minimum Regret

Suppose we are again given the scenario matrix $\mathbf{S}$, either from historical returns or from a scenario simulation exercise. As in basic decision theory, we could choose minimax criteria, as illustrated in Figure 5.12 (i.e., we might want to minimize the maximum portfolio loss—minimizing regret).[9] This could be the

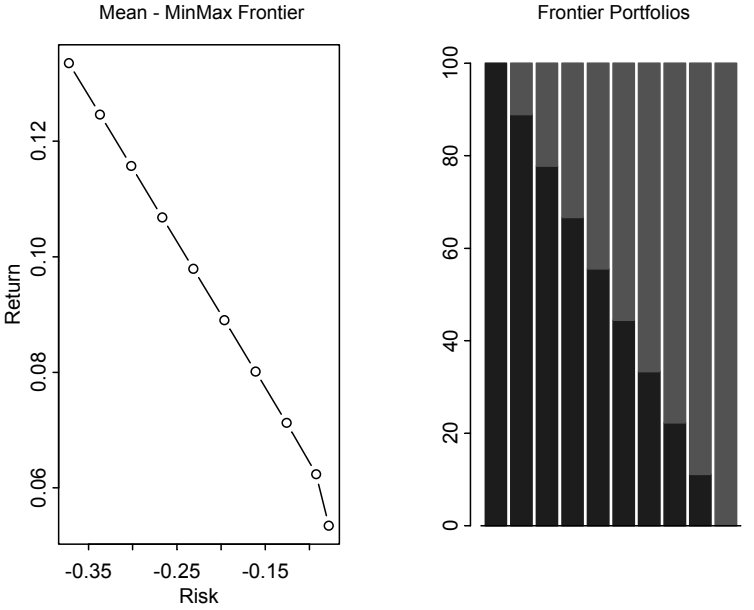

Figure 5.12 MinMax Efficient Frontier

optimal strategy for investors who have to make sure under all means (scenarios) that they never experience a particular size of loss. Focusing on extreme events will have its merits if returns either substantially deviate from normality or if investors are extremely risk-averse. Minimizing the maximum loss can be written as a linear program:

$$\begin{aligned} &\max\ r_{\min} \\ &\sum_{i=1}^{n} w_i r_{i,s} - r_{\min} \geq 0 \\ &\sum_{i=1}^{n} w_i \mu_i = \bar{\mu} \\ &\sum_{i=1}^{n} w_i = 1 \\ &w_i \geq 0. \end{aligned} \tag{5.27}$$

The first constraint $\sum_{i=1}^{n} w_i r_{i,s} - r_{\min} \geq 0$ ensures that there is no scenario for which the portfolio return is worse than the minimum return. As $r_{\min}$ is a variable as well as the objective in the system (5.27), it will take on the value of the minimum maximum loss. An alternative (equivalent) formulation to (5.27) is

to maximize return subject to the restriction that there is no scenario for which the portfolio return falls below a threshold return $r_{\min}$.

$$\begin{aligned} &\max \sum_{i=1}^{n} w_i \mu_i \\ &\sum_{i=1}^{n} w_i r_{i,s} \geq r_{\min} \\ &\sum_{i=1}^{n} w_i = 1 \\ &w_i \geq 0. \end{aligned} \tag{5.28}$$

We choose the program of (5.27) for implementation in NUOPT for S-PLUS (shown in Code 5.5).

```
MinMax.model <- function(S, mu.target)
{
  m <- nrow(S)
  n <- ncol(S)
  mu.bar <- apply(S, 2, mean)
  asset <- Set()
  period <- Set()
  i <- Element(set=asset)
  s <- Element(set=period)
  S <- Parameter(S, index=dprod(s,i))
  mu.bar <- Parameter(as.array(mu.bar), index=i)
  mu.target <- Parameter(mu.target, changeable=T)
  w <- Variable(index=i)
  mu.Min <- Variable()
  Sum(S[s,i]*w[i],i)-mu.Min >= 0
  MinMax <- Objective(type="maximize")
  MinMax ~ mu.Min
  Sum(mu.bar[i]*w[i],i) == mu.target
  Sum(w[i],i) == 1
  w[i] >= 0
}

MinMax.portfolio <- function(S, mu.target)
{
  call(MinMax.model)
  MinMax.system <- System(MinMax.model, S,
     mu.target)
  solution <- solve(MinMax.system, trace=T)
  weight <-
     matrix(round(solution$variable$w$current,
```

```
    digit=5)*100, ncol=1)
  risk <- solution$objective
  return(weight,risk)
}

MinMax.frontier <- function(S, n.pf)
{
  call(MinMax.portfolio)
  Risk <- matrix(0, ncol=1, nrow=n.pf)
  Return <- matrix(0, ncol=1, nrow=n.pf)
  m <- nrow(S)
  mu.min <- min(apply(S,2,mean))
  x <- MinMax.portfolio(S, mu.target=mu.min)
  weight <- x$weight
  Risk[1,1] <- x$risk
  Return[1,1] <- mu.min
  mu.max <- max(apply(S, 2, mean))
  mu.range <- seq(mu.min, mu.max,
     (mu.max-mu.min)/(n.pf-1))
  for(i in 2:n.pf){
     x <- MinMax.portfolio(S,
     mu.target=mu.range[i])
     Risk[i,1] <- x$risk
     Return[i,1] <- mu.range[i]
     weight <- cbind(weight,x$weight)
  }
  graphsheet
  par(mfrow=c(1,2))
  plot(Risk, Return, type="b")
  title("Mean - MinMax Frontier")
  barplot(weight)
  title("Frontier Portfolios")
  list("optimal.weights" = weight)
}
```

Code 5.5 Regret Minimization

```
> x <- MinMax.frontier(S, n.pf=50)
> x$optimal.weights
     [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
[1,]  100 88.889 77.778 66.667 55.556 44.444
[2,]    0  0.000  0.000  0.000  0.000  0.000
[3,]    0 11.111 22.222 33.333 44.444 55.556
[4,]    0  0.000  0.000  0.000  0.000  0.000
```

```
        [,7]   [,8]   [,9]  [,10]
[1,] 33.333 22.222 11.111     0
[2,]  0.000  0.000  0.000     0
[3,] 66.667 77.778 88.889   100
[4,]  0.000  0.000  0.000     0
```

Minimizing maximum regret leads to concentrated portfolios. The highly non-normal assets 2 and 4 never enter the optimal solution.

5.6 Conditional Value-at-Risk

5.6.1 CVaR, Tail Conditional Loss, and VaR

Suppose we sampled discrete realizations of portfolio returns from a continuous distribution to arrive at m realizations $\{r_s\}_{s=1,\ldots,m}$ of random returns. To make matters transparent, just think of this as a sequence of returns $\{6,-10,-3,4,5,\cdots,1\}_{m=100}$.[10] We now define the order statistics (simply by ordering the returns starting with the smallest return from the left) $r_{1:m} \le r_{2:m} \le \ldots \le r_{m:m}$ that result in the sorted returns $\{-10,-5,-3,-3,-3,-3,-3,\cdots,16\}_{m=100}$. If we need to estimate the $\alpha\%$-quantile (value-at-risk), we simply look for

$$VaR_\alpha = r_{\alpha m:m}. \tag{5.29}$$

If we set $\alpha = 5\%$, $m = 100$, we arrive at $VaR = r_{5:100} = -3$ (5th out of 100 returns) in the example above.

The estimator for the expected loss in $\alpha\%$ of all cases, also called **conditional value-at-risk** ($CVaR$) or **expected shortfall** (ES), is calculated from

$$ES_\alpha = \frac{1}{m\alpha}\sum_{s=1}^{m\alpha} r_{s:m}. \tag{5.30}$$

For the example above we get

$$ES_{5\%} = \tfrac{1}{5}\sum_{s=1}^{5} r_{s:100} = \tfrac{1}{5}(-10-5-3-3-3) = -4.8.$$

A measure similar to (5.30) is the "tail conditional loss" (TCL) defined as $E(r|r \le VaR_\alpha)$, which looks the same at first sight.

$$TCL_\alpha = \frac{\sum_{s=1}^{m} \delta_s r_s}{\sum_{s=1}^{m} \delta_s} \text{ where } \delta_s = \begin{cases} 1, & r_s \le r_{\alpha m:m} \\ 0, & \text{otherwise.} \end{cases} \tag{5.31}$$

This, however, is only true for continuous distributions, as the discrete example above shows.

$$TCL_\alpha = (-10-5-3-3-3-3-3)/7 = -4.3.$$

Tail conditional loss and expected shortfall differ. In general, we will find that expected shortfall is at least as large as tail conditional loss.

$$ES_\alpha = TCL_\alpha + \phi(TCL_\alpha - VaR_\alpha), \tag{5.32}$$

where

$$\phi = \frac{prob(r \le VaR)}{\alpha} - 1 \ge 0.$$

The reason for this is that for discrete distributions $prob(r \le VaR) \ge \alpha$. In our example, we find that $prob(r \le VaR) = \frac{7}{100} = 7\%$, which is larger than 5%. Substituting the appropriate values into (5.32), we get

$$-4.8 = -4.3 + \left(\frac{7\%}{5\%} - 1\right)(-4.3 - 3).$$

We now show various ways to calculate the numbers above in S-PLUS. Suppose we simulate a mixture (of two normals) to generate a data set for the sample calculations below.

```
returns <- c(rnorm(50, -0.4, 0.3),
  rnorm(950, 0.07, 0.2))
graphsheet()
par(mfrow=c(1,2))
hist(returns)
qqnorm(returns)
qqline(returns)
```

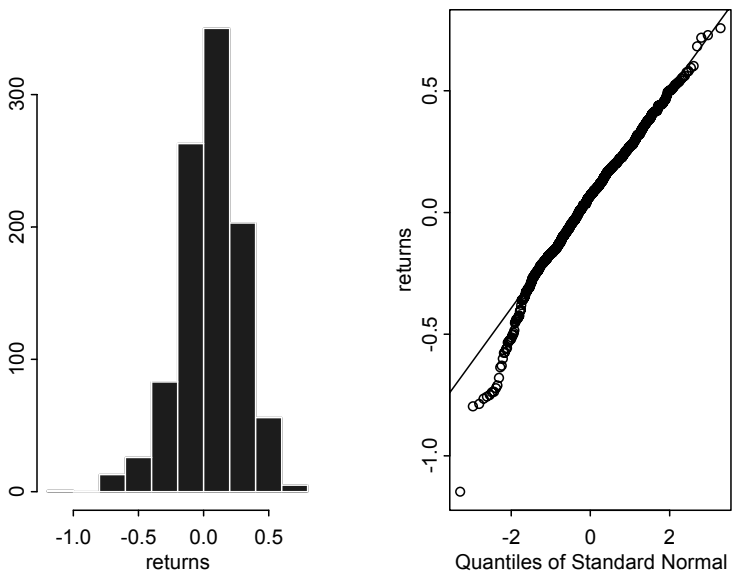

Figure 5.13 Sample Data

The data are plotted in Figure 5.13. It is apparent that the distribution differs significantly from the normal distribution in its left tail (the Q-Q plot deviates substantially from a straight line). S-PLUS functions that generate the required risk measures are given below (in Code 5.6) and can also be used to generate the distribution of our estimated risk measure via repeated resampling (bootstrapping). Bootstrapped results are shown in Figure 5.14.

```
VaR <- function(returns, alpha){
  sort(returns)[trunc(length(returns)*alpha)]
}

CVaR <- function(returns, alpha){
  mean(sort(returns)[1:trunc(length(returns)*alpha)
     ])
}

TCL <- function(VaR, returns){
  mean(returns[returns<=VaR(returns, alpha)])
}

bs.VaR <- bootstrap(returns, VaR(returns, alpha))
bs.CVaR <- bootstrap(returns, CVaR(returns, alpha))
```

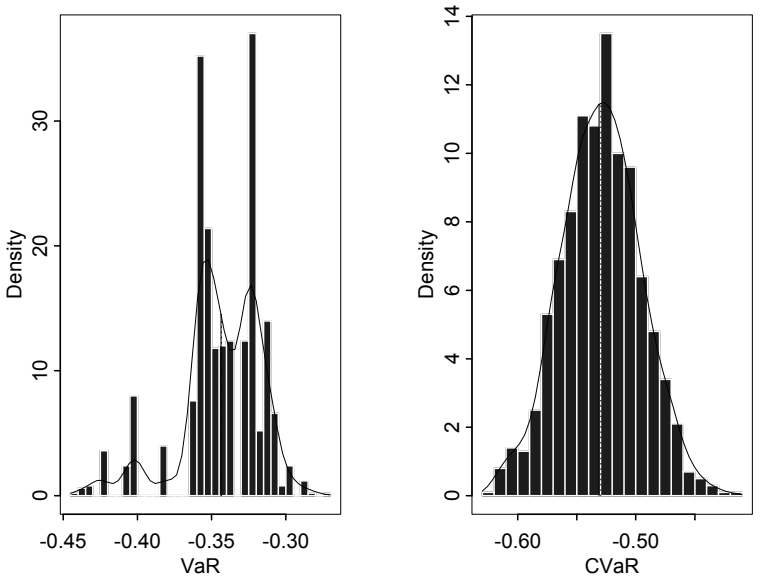

Figure 5.14 Resampled VaR and CVaR Calculations

```
graphsheet()
par(mfrow=c(1,2))
plot(bs.VaR, xlab="VaR", main="")
plot(bs.CVaR, xlab="CVaR", main="")
```

Code 5.6 Bootstrap Distributions of Various Risk Measures

This way, the bootstrap function can be used to investigate which risk concept (VaR, CVaR, or volatility) needs more data to be estimated with the same precision. Note that we can address the resampled risk measures (see Figure 5.15) directly using

```
bs.VaR <- bootstrap(returns,
  VaR(returns, alpha))$replicates
```

for example. We can then use the `boxplot()` command to visualize the estimation error in all three risk measures. For this purpose, we remove the means of the estimates of our risk measures to plot them on the same level in Code 5.7.

```
bs.VaR <- bootstrap(returns,
  VaR(returns, alpha))$replicates

bs.CVaR <- bootstrap(returns,
```

```
  CVaR(returns, alpha))$replicates

bs.Vol <- bootstrap(returns,
  Vol(returns))$replicates

demean <- function(x){x-mean(x)}

boxplot(demean(bs.VaR), demean(bs.CVaR),
  demean(bs.Vol),
  names=c("VaR", "CVaR", "Volatility"))
```

Code 5.7 Estimation Error in Various Risk Measures

We see that the estimation error for CVaR is larger than for volatility (where precision is highest) or VaR. This makes intuitive sense, as CVaR looks deeper into the tail and is hence outlier-dependent, while large outliers do not affect VaR (which is economically a central weakness). Volatility, on the other hand, uses all the data in a sample and arrives at a very precise estimate (that might be completely useless in the case of serious non-normality).

Rather than using sampled data, we could also employ the numerical integration techniques offered by S-PLUS to calculate the required risk measures. Suppose we are given the distribution behind Figure 5.13 in continuous form, $pf(x,\mu_1,\sigma_1)+(1-p)f(x,\mu_2,\sigma_2)$, where $f(x,\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$. The value-at-risk for the 2.5% level is –45.89%. To check this, just calculate

$$p\int_{-\infty}^{-45.89\%} f(x,\mu_1,\sigma_1)+(1-p)\int_{-\infty}^{-45.89\%}(x,\mu_2,\sigma_2)=2.5\%. \qquad (5.33)$$

```
integrand.1 <- function(x){
  (0.05*dnorm(x,-0.4,0.3) + 0.95*dnorm(x,0.07,0.2))
}
integrate(integrand.1,  - Inf, -0.4589)$integral
[1] 0.02499479
```

We can also calculate the expected shortfall, or conditional value-at-risk (see Code 5.8),

$$\frac{p\int_{-\infty}^{-45.89\%} xf(x,\mu_1,\sigma_1)dx+(1-p)\int_{-\infty}^{-45.89\%} xf(x,\mu_2,\sigma_2)dx}{p\int_{-\infty}^{-45.89\%} f(x,\mu_1,\sigma_1)dx+(1-p)\int_{-\infty}^{-45.89\%} f(x,\mu_2,\sigma_2)dx}=-65.366\%\ . \qquad (5.34)$$

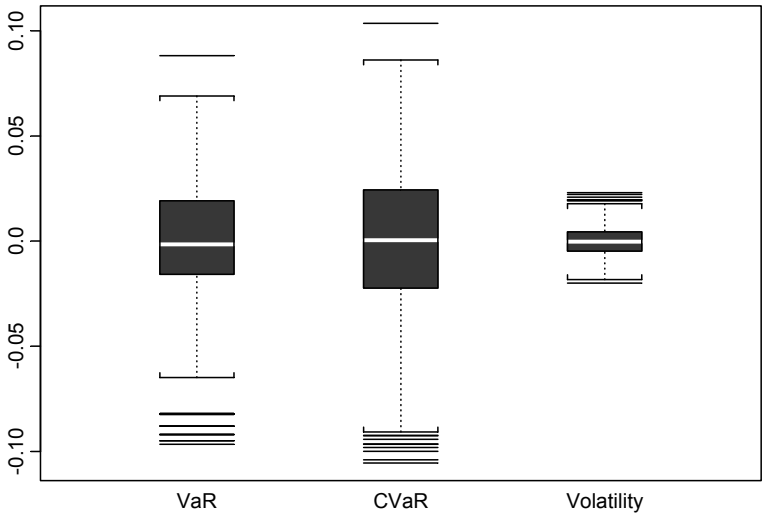

Figure 5.15 Boxplot for Resampled Risk Measures

```
integrand.2 <- function(x){
  (0.05*dnorm(x,-0.4,0.3) +
     0.95*dnorm(x,0.07,0.2))*x
}
integrate(integrand.2,-Inf,
  -0.458915)$integral/integrate(integrand.1,-Inf,
  -0.458915)$integral
[1] -0.6536664
```

Code 5.8 Risk Estimates via Numerical Integration

Numerical integration allows us to calculate arbitrary risk figures as soon as a continuous distribution has been fit to the data.

5.6.2 What Do We Require from a Risk Measure?

For portfolio managers, risk managers, and plan sponsors, there is the vital question: what properties are needed for a statistic of portfolio returns to qualify as a risk measure? The answer to this question has been given through a complete set of axioms. They define what has been called a **coherent risk measure**.[11] A coherent risk measure is a function (that translates returns into a risk figure) that is

1. **Monotonous**. Larger losses translate into higher risks.

2. **Positive homogeneous**. If we multiply holdings (positions, exposures) by a linear factor, risk also rises by this factor.
3. **Invariant to translations**. Adding a constant to our losses does not change risk.
4. **Subadditive**. The risk of a portfolio is at most the combined risks of the single positions.

The last axiom catches diversification. Adding two portfolios together must not create higher risks than both on a stand-alone basis. These axioms define the nature of the concept with a minimum set of precise formulations (requirements). A risk measure that violates one of the axioms above will lead to paradoxical results. Note that return statistics that do not fit into the axiomatic framework cannot be called risk measures (by the very definition).

Let's see how VaR, CVaR, volatility, and shortfall probability do in a simple setting. A plan sponsor budgets risks given to individual managers. His scenarios of two very diversifying managers (i.e., negative returns can only occur in different states) are given in Table 5.1.[12]

Table 5.1 Data for Manager Combination: Active Returns

Scenario	Manager 1	Manager 2	Manager 1+2	Probability
1	–20%	2%	–9%	3%
2	–3%	2%	–0.5%	2%
3	2%	–20%	–9%	3%
4	2%	–3%	–0.5%	2%
5	2%	2%	2%	90%

We can now calculate the risk measures mentioned above. The outcome is summarized in Table 5.2. While volatility and CVaR are decreasing as we move from a stand-alone approach to a combination of managers, this is not the case for shortfall probability and VaR. Hence, both statistics are not suitable risk measures.

Table 5.2 Risk Measures in Multiple-Manager Example

Risk Measure	Manager 1	Manager 2	Manager 1+2
Volatility	3.80%	3.80%	2.63%
VaR	–3%	–3%	–9%
Shortfall Probability	5%	5%	10%
CVaR	–13.20%	–13.20%	–5.60%

The reason for these paradoxical results lies in the concept of value-at-risk. It ignores the large –20% losses that are waiting undetected in the tail of the distribution. However, when we average across portfolios, these returns will be diversified into the portfolio risk measure and will increase risk, as they have been ignored before. No investor would find a risk measure that attaches equal weight to small losses and complete bankruptcy satisfying. Value-at-risk fails in detecting tail risks.

Value-at-risk and shortfall probability are not coherent risk measures (i.e., they should not be called risk measures at all). Value-at-risk will only be subadditive in special circumstances (i.e., for so-called elliptical distributions, such as the normal distribution, Student's *t* distribution, and the Cauchy distribution). In fact, value-at-risk and volatility share the same properties when the underlying distribution is elliptical.[13] However, the distributions of many assets involved in portfolio construction do not belong to this class. They either naturally deviate from asset class characteristics (hedge fund, credit risk, etc.) or are deliberately created to do so using heavily skewed distributions. Interestingly, value-at-risk, which once was regarded as the Holy Grail in risk management, fails when return distributions are not elliptical. Casually stated, value-at-risk cannot deal with non-normality. Table 5.3 gives a concise summary of our discussion.

Table 5.3 VaR versus CVaR

Criterion	VaR	CVaR
Subadditivity?	No	Yes
Tail risk measure?	No	Yes
Handle nonnormality?	Constrained to cases for non-normal elliptical distributions	Yes
Data requirements?	Needs more data than the volatility measure to be measured with the same precision	Needs more data than VaR to be measured with the same precision

It is difficult to understand why value-at-risk is still so popular. We do warn against its use in portfolio optimization, as the problems above are likely to increase further when a portfolio optimizer leverages axiomatic shortcomings of VaR.

5.6.3 The Use of CVaR in Portfolio Construction

Not only does CVaR offer a much sounder theoretical basis for risk management decisions but it is also computationally more efficient. While portfolio

optimization using VaR becomes a complicated integer-programming problem, CVaR optimization only requires well-established and widely available linear programming tools. First, we define an auxiliary variable,

$$e_s = \max\left[0, VaR - \sum_{i=1}^{n} w_i r_{i,s}\right]. \tag{5.35}$$

Suppose $VaR = -20$ (percent), while the portfolio return for scenario s turns out to be $\sum_{i=1}^{n} w_i r_{i,s} = -25$. Equation (5.35) would then find an excess of $e_s = \max[0,(-20)-(-25)] = \max[0,5] = 5$. Conditional value-at-risk equals value-at-risk plus the average of all losses in excess of value-at-risk. Hence we can write

$$CVaR = VaR - \left(\frac{1}{m}\sum_{s=1}^{m} e_s\right)/\alpha\,. \tag{5.36}$$

Note that $\frac{1}{m}\sum_{s=1}^{m} e_s$ reflects the average excess loss across all m scenarios. In order to scale this loss up to the average excess loss, if an excess loss occurs, we have to divide it by the probability of an excess loss (α). As VaR is a negative number, $VaR - \left(\frac{1}{m}\sum_{s=1}^{m} e_s\right)/\alpha$ will be even more negative. A risk-averse investor will hence want to maximize $CVaR$ (–5 is larger than –20). The complete portfolio optimization problem can now be written down as

$$\begin{aligned}
&\max_{w_i, e_s, VaR} \; VaR - \left(\frac{1}{m}\sum_{s=1}^{m} e_s\right)/\alpha \\
&\sum_{i=1}^{n} w_i \mu_i \geq \bar{\mu} \\
&e_s \geq VaR - \sum_{i=1} w_i r_{i,s} \\
&e_s \geq 0 \\
&w_i \geq 0.
\end{aligned} \tag{5.37}$$

A closer look at (5.37) will reveal some of the mechanics. Note that excesses are forced to be positive. Any excess ($e_s \geq VaR - \sum_{i=1} w_i r_{i,s}, e_s \geq 0$) will hence have a negative impact on the objective function. Excesses can be kept small by choosing the appropriate set of weights in order to prevent $VaR - \sum_{i=1} w_i r_{i,s}$ from becoming a large positive number.

If portfolio returns in all scenarios were positive, how could we prevent e_s from becoming negative? The optimizer can now increase VaR and therefore positively impact the objective function. However, moving VaR up too much will result in increasing excesses that will counterbalance this effect. The CVaR model is given in Code 5.9.

```
CVaR.model <- function(S, alpha, mu.target)
{
  m <- nrow(S)
  n <- ncol(S)
  mu.bar <- apply(S, 2, mean)
  asset <- Set()
  period <- Set()
  mu.VaR <- Set(1)
  i <- Element(set=asset)
  s <- Element(set=period)
  mu.VaR <- Element(set=mu.VaR)
  S <- Parameter(S, index=dprod(s,i))
  mu.bar <- Parameter(as.array(mu.bar), index=i)
  mu.target <- Parameter(mu.target, changeable=T)
  w <- Variable(index=i)
  e <- Variable(index=s)
  mu.VaR <- Variable(index=mu.VaR)
  e[s] >= mu.VaR[1]-Sum(S[s,i]*w[i],i)
  e[s] >= 0
  CVAR <- Objective(type="maximize")
  CVAR ~ mu.VaR[1]-(1/m)*(1/alpha)*Sum(e[s],s)
  Sum(mu.bar[i]*w[i],i) == mu.target
  Sum(w[i],i) == 1
  w[i] >= 0
}
```

Code 5.9 CVaR Optimization

Suppose we run the model on $m = 100$ scenarios with $\alpha = 0.05$ and $\bar{\mu} = 0.04$. What are the values for VaR and CVaR?

```
mu.target <- 0.04
alpha <- 0.05
S.mvnorm <- matrix(rmvnorm(100, mean=c(0.02,
   0.04, 0.05, 0.08), cov=diag(rep(0.2,4))),
   ncol=4)
CVaR.system <- System(CVaR.model, S.mvnorm,
   alpha, mu.target)
solution <- solve(CVaR.system, trace=T)
```

The value for CVaR can be obtained from `solution$objective`, which returns a value of `−0.7469582`. We compare this to VaR, which can be retrieved from `solution$variables$mu.VaR$current` and amounts to `-0.478263`. Finally, we plot the cumulative distribution for portfolio returns as well as for a normal distribution in Figure 5.16 and the CVaR frontier and optimal portfolios in Figure 5.17.

```
normal <- rnorm(100000, mean(returns),
  sqrt(var(returns)))
cdf.compare(returns, normal)

CVaR.portfolio <- function(S, alpha, mu.target)
{
  call(CVaR.model)
  CVaR.system <- System(CVaR.model, S, alpha,
     mu.target)
  solution <- solve(CVaR.system, trace=T)
  weight <-
     matrix(round(solution$variable$w$current,
     digit=5)*100, ncol=1)
  risk <- -solution$objective
  return(weight,risk)
}

CVaR.frontier <- function(S, alpha, n.pf)
{
  call(CVaR.portfolio)
  Risk <- matrix(0, ncol=1, nrow=n.pf)
  Return <- matrix(0, ncol=1, nrow=n.pf)
  mu.min <- min(apply(S,2,mean))
  mu.max <- max(apply(S,2,mean))
  mu.range <- seq(mu.min, mu.max,
     (mu.max-mu.min)/(n.pf-1))
  x <- CVaR.portfolio(S, alpha, mu.target=mu.min)
  weight <- x$weight
  Risk[1,1] <- x$risk
  Return[1,1] <- mu.min
  for(i in 2:n.pf){
     x <- CVaR.portfolio(S, alpha,
        mu.target=mu.range[i])
     Risk[i,1] <- x$risk
     Return[i,1] <- mu.range[i]
     weight <- cbind(weight,x$weight)
```

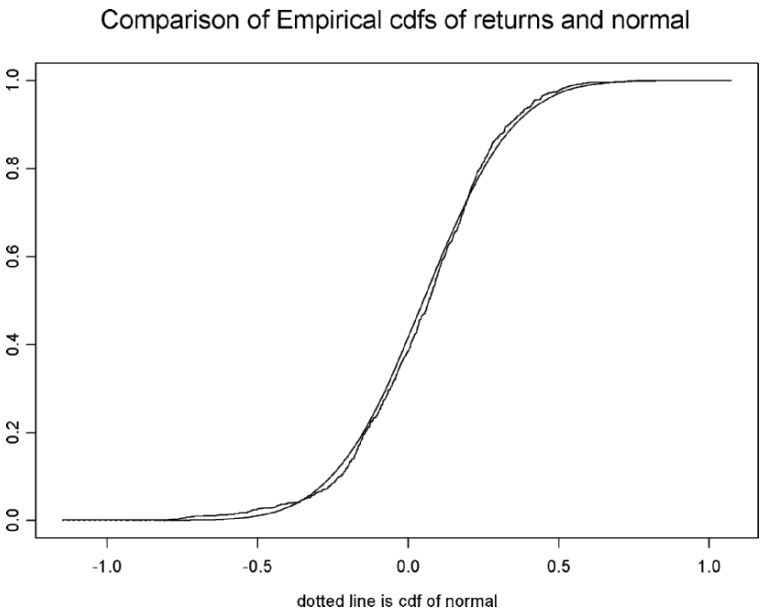

Figure 5.16 Cumulative Distribution of Portfolio Returns

```
  }
  graphsheet()
  par(mfrow=c(1,2))
  plot(Risk, Return, type="b")
  title("Mean - CVaR Frontier")
  barplot(weight)
  title("Frontier Portfolios")
  list("optimal.weights" = weight)
}
```

Code 5.10 CVaR Frontier

```
> x <- CVaR.frontier(S, alpha, n.pf=20)
> x$optimal.weights

       [,7]   [,8]   [,9]  [,10]  [,11]  [,12]
[1,] 67.944 62.631 57.295 52.035 46.684 41.375
[2,]  0.000  0.000  0.000  0.000  0.000  0.000
[3,] 26.918 31.703 36.253 41.548 45.949 50.761
[4,]  5.138  5.666  6.452  6.416  7.368  7.864

      [,13]  [,14]  [,15]  [,16]  [,17]  [,18]
[1,] 36.067 30.777 25.496 20.204 14.912  9.646
```

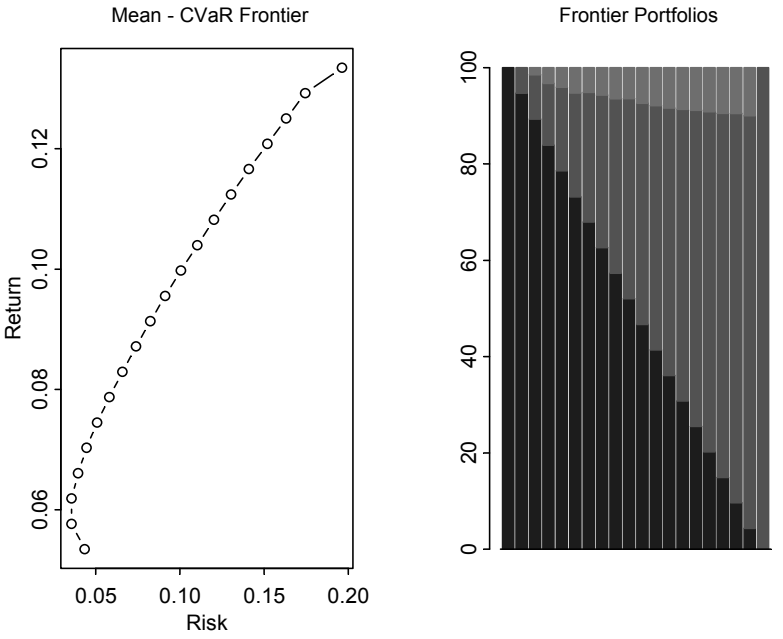

Figure 5.17 CVaR Frontier and Optimal Portfolios

```
[2,]  0.000  0.000  0.000  0.000  0.000  0.000
[3,] 55.592 60.589 65.685 70.666 75.647 80.882
[4,]  8.341  8.634  8.819  9.130  9.441  9.471

      [,19] [,20]
[1,]  4.338     0
[2,]  0.000     0
[3,] 85.705   100
[4,]  9.957     0
```

Conditional value-at-risk looks deeply into the tail of a distribution. In contrast with mean-variance-based solutions, we note that the highly non-normal assets 2 and 4 enter and are given much less weight.

5.6.4 VaR Approximation Using CVaR

It is well-known that VaR for discrete distributions is a nonsmooth, nonconvex, and multiextremum function with respect to w_i. VaR and the related shortfall risk are therefore difficult to optimize, as we have already seen in Section 5.4.2. This section will present a heuristic that attempts to minimize VaR by solving a

sequence of CVaR problems.[14] As CVaR presents an upper bound on VaR (CVaR will always be greater than VaR, as it adds the average of the excess losses to VaR), one approach to minimizing VaR is to minimize the upper bound in a sequence of CVaR problems, gradually discarding scenarios that exhibit losses larger than VaR.

Step 0. Let us assume we have generated $m = 1000$ scenarios for $n = 4$ assets. We start with a standard CVaR minimization for a prespecified α and return target $\bar{\mu}$ as described in (5.37) using all m scenarios. The resulting CVaR (call it α-CVaR) represents the first upper bound on VaR for the prespecified α (call this α-VaR). Now we split the set of total scenarios into active scenarios (those used for further CVaR minimizations) and inactive scenarios (those discarded). From all scenarios that show losses larger than α-VaR, we discard a fraction ξ. For example, if 50 portfolio returns fall below VaR, we discard the largest 25 losses for further use.

Step 1. Start a new CVaR optimization on the remaining set of active scenarios (if we discard 25 out of 1000 scenarios we are left with 975 scenarios). However, we have to modify our CVaR optimization in two important respects. First we need to take into account that we discarded a number of scenarios. As we are interested in the α-VaR, we need to ensure that the α_1-CVaR optimization in Step 2 focuses on the same quantile. The new α_1 for the α_1-CVaR optimization needs to satisfy

$$\alpha_i = 1 - (1 - \alpha)\left(1 - \frac{\sum_{step=0,i} discarded\ scenarios\ in\ step_i}{m}\right)^{-1}.$$

An example has been calculated in Table 5.4. As the number of discarded scenarios rises, we need to go further into the tail to maintain the quantile with respect to the original set of scenarios.

Table 5.4 Evolution of α_i for α=0.05 and ξ=0.5

Step i	# Discarded Scenarios	α_i
0	0	5.00%
1	25	2.56%
2	13	1.30%
3	6	0.65%
4	3	0.33%
5	2	0.16%

Second, we need to ensure that the allocation resulting from Step 1 does not create losses for the active scenarios that are larger than those for the inactive scenarios. This is important if we want to gradually reduce the number of active scenarios in a meaningful way. We therefore have to add m constraints and one new free variable (γ) to problem (5.37):

$$\sum_{i=1} w_i r_{i,s} \geq \gamma,\ s \in \textit{active scenario}$$
$$\sum_{i=1} w_i r_{i,s} \leq \gamma,\ s \in \textit{inactive scenario}$$

The constraints above will always be satisfied, as the optimizer could always default to the optimal allocation from the previous optimization (which we used to split scenarios into active and inactive scenarios). As before, we calculate α-VaR for the optimal solution in Step 1 and use it to split up scenarios further (reducing the number of scenarios stepwise) for use in the next step. It is obvious that α_1-CVaR needs to be smaller than α-CVaR, as we discarded the largest losses but adjusted the quantile for the inactive scenarios. Note that the VaR calculation is not affected, as all losses below VaR are equally counted, no matter how large they are. A refined CVaR code that takes account of this is shown in Code 5.11.

```
CVAR.model <- function(S.in, S.out, alpha, mu.bar,
  mu.target, VaR.cutoff)
{
  m.in <- nrow(S.in)
  m.out <- nrow(S.out)
  m <- m.in+m.out
  n <- ncol(S.in)
  asset <- Set()
  period.in <- Set()
  period.out <- Set()
  mu.VaR <- Set(1)
  i <- Element(set=asset)
  s <- Element(set=period.in)
  ss <- Element(set=period.out)
  mu.VaR <- Element(set=mu.VaR)
  S.in <- Parameter(S.in, index=dprod(s,i))
  S.out <- Parameter(S.out, index=dprod(ss,i))
  mu.bar <- Parameter(as.array(mu.bar), index=i)
  mu.target <- Parameter(mu.target, changeable=T)
  w <- Variable(index=i)
  e <- Variable(index=s)
  mu.VaR <- Variable(index=mu.VaR)
```

```
  g <- Variable(VaR.cutoff)
  e[s] >= mu.VaR[1]-Sum(S.in[s,i]*w[i],i)
  e[s] >= 0
  CVAR <- Objective(type="maximize")
  CVAR ~ mu.VaR[1]-(1/m)*(1/alpha)*Sum(e[s],s)
  Sum(mu.bar[i]*w[i],i) == mu.target
  Sum(w[i],i) == 1
  Sum(w[i]*S.in[s,i],i) >= g
  Sum(w[i]*S.out[ss,i],i) <= g
  w[i] >= 0
}
```

Code 5.11 VaR Approximation Using CVaR

Step 2. Repeat Step 1 as long as there are scenarios left to be discarded. The more scenarios we discard, the closer α_i-CVaR and α_i-VaR will become as more and more scenarios with large losses are removed (reducing CVaR), while this removal does not affect VaR as long as α_i is properly set. Implementing the algorithm above leads to the result shown in Figure 5.18 for our sample data set.

5.7 CDO Valuation using Scenario Optimization

Suppose we know the loss distribution of an underlying pool of assets valued today at 100 (i.e., we know $f(l)$). Note that $\tilde{l}$ can assume positive values (losses) as well as negative values (profits). This asset pool is financed via three different tranches. The first tranche is called *equity* (or sometimes a *junior note*). Liabilities for holders of this tranche are limited to l_{α_1}, which denotes the α_1 percentile of the loss distribution. Payoff to equity can be expressed as

$$CF_{equity} = l_{\alpha_1} - \tilde{l} + \max\left(\tilde{l} - l_{\alpha_1}, 0\right). \tag{5.38}$$

If the losses exceed l_{α_1}, the equity is wiped out and losses will start to eat into the second tranche (also called the *mezzanine*). The second tranche promises to pay an amount $l_{\alpha_1} - l_{\alpha_2}$. Losses larger than l_{α_2} lead to a complete loss of the second tranche and eat into the last tranche,

$$CF_{mezzanine} = l_{\alpha_2} - l_{\alpha_1} - \max\left(\tilde{l} - l_{\alpha_1}, 0\right) + \max\left(\tilde{l} - l_{\alpha_2}, 0\right). \tag{5.39}$$

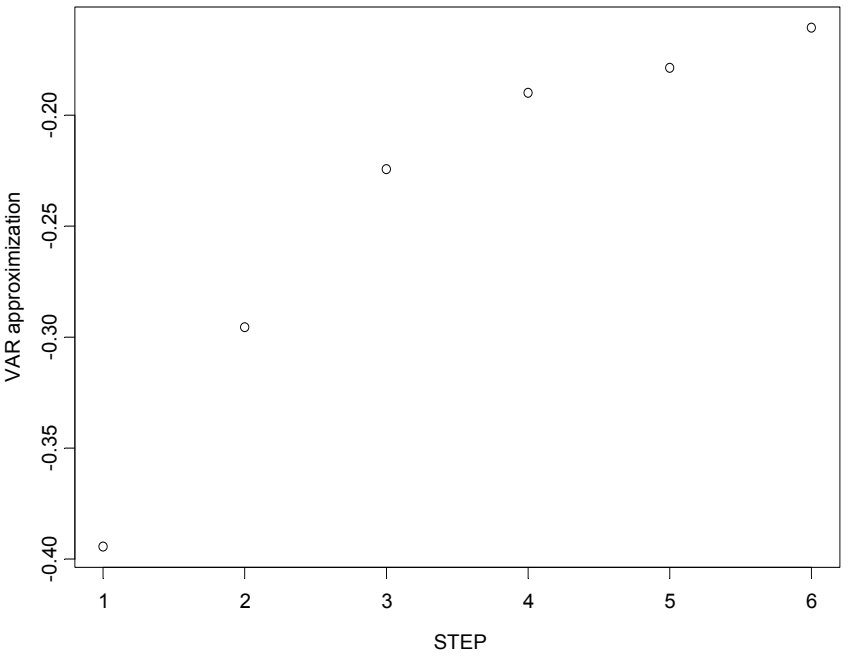

Figure 5.18 VaR Approximation Using CvaR for a Sample Data Set

The last tranche is called a *senior note*. Losses will only eat into the senior note after the first two tranches have been wiped out,

$$CF_{senior} = 100 - l_{\alpha_1} - \left(l_{\alpha_2} - l_{\alpha_1}\right) - \max\left(\tilde{l} - l_{\alpha_2}, 0\right). \tag{5.40}$$

We can see that investors in senior notes write a limited-liability option to holders of mezzanine debt $\max\left(\tilde{l} - l_{\alpha_2}, 0\right)$, while mezzanine debt investors write a limited-liability option $\max\left(\tilde{l} - l_{\alpha_1}, 0\right)$ to equity investors. If we add all positions, we arrive at

$$\begin{aligned} CF_{equity} + CF_{mezzanine} + CF_{senior} &= l_{\alpha_1} - \tilde{l} + \max\left(\tilde{l} - l_{\alpha_1}, 0\right) \\ &\quad + l_{\alpha_2} - l_{\alpha_1} - \max\left(\tilde{l} - l_{\alpha_1}, 0\right) \\ &\quad + \max\left(\tilde{l} - l_{\alpha_2}, 0\right) + 100 \\ &\quad - l_{\alpha_1} - \left(l_{\alpha_2} - l_{\alpha_1}\right) - \max\left(\tilde{l} - l_{\alpha_2}, 0\right) \\ &= 100 - \tilde{l}, \end{aligned} \tag{5.41}$$

which is exactly the payoff to the asset pool. Assuming that the loss distribution is log-normal, we can use standard Monte Carlo simulation techniques to evaluate the attached options. However, how can we value options under arbitrary distributions? One simple way is to use

$$\pi_s^* = \pi_s \frac{U'\left(W_s^{optimal}\right)}{\sum_{s=1}^{m} \pi_s U'\left(W_s^{optimal}\right)}, \tag{5.42}$$

where $W_s^{optimal}$ denotes the wealth in scenario s attached to the optimal solution of $\max_w \sum_{s=1}^{m} \pi_s U(W_s)$ and $W_s = 1 + wr_f + (1-w)r_s$. Cash flows from a particular CDO tranche are valued with

$$value_{tranche_i} = \sum_{s=1}^{m} \pi_s^* \left(\frac{cash - flow_s^{tranche_i}}{1 + r_f} \right) \tag{5.43}$$

assuming a particular form for $U(W_s)$.

We finish this section with an example. Suppose $r_f = 0.03$, $l_{\alpha_1} = 2$, and $l_{\alpha_2} = 10$. The returns of the underlying assets in a hypothetical CDO are assumed to be drawn from a normal distribution with mean 6% and volatility 9%. Applying standard risk-neutral valuation theory, we arrive at the following values for the three tranches:

$$\begin{aligned} value_{junior} &= 6.58 \\ value_{mezzanine} &= 6.42 \\ value_{senior} &= 87.00. \end{aligned}$$

These prices are required to subsequently calculate the returns for the respective tranches, as shown in Figure 5.19 and Figure 5.20.

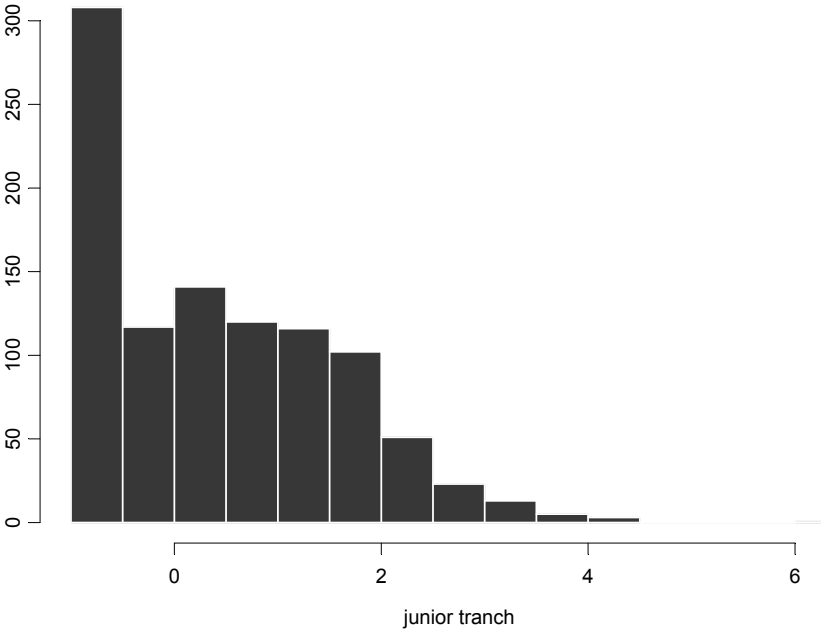

Figure 5.19 Return Distribution for Junior Tranche

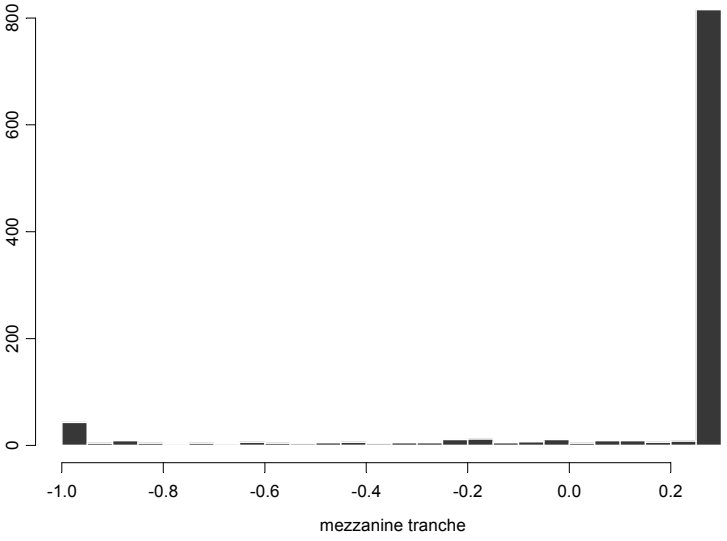

Figure 5.20 Return Distribution for Mezzanine Tranche

Exercises

1. Write a program that uses the methodology in Section 5.1.2.

2. Solve (5.12), first in Excel and then with `solveQP()`. How do we need to rewrite (5.12) in the case of different upper and lower costs of mean deviations?

3. Download data on a high yield corporate bond index.
   (a) Calculate the value of tranches within a standard CDO that is financed via equity (taking the first 5% losses), mezzanine debt (taking the next 10% of losses), and senior debt. Assume a risk-free rate of 3% and use Monte Carlo simulation.
   (b) Use the option prices of (a) to generate scenarios that are consistent with CDO pricing.
   (c) Add at least two more asset classes to the junior note and construct a CVaR efficient frontier.
   (d) How does the CVaR frontier differ from a mean-variance frontier?

4. Can you approximate semi-variance optimization using the MAD model and a piecewise linearization? Hint: See Hamza and Janssen (1995).

5. Extend the failed model (5.25) for shortfall probability to lower partial moments of degrees $k = 1, 2, 3, 4$. What do you observe?

6. Include fixed and proportional transaction costs in the weighted semi-variance model.

7. Assume a mixture of (two) normal distributions and write a program that calculates lower and upper partial moments for arbitrary threshold returns and moments using numerical integration. Check your results using Monte Carlo simulation.

8. Write a program that does the calculations in Section 5.6.4. Experiment with the number of scenarios discarded in each step. What do you observe?

Endnotes

[1] The time series stretches from January 1994 to December 2002 for Emerging Markets (JPM.EMBI) and from February 1986 to December 2002 for the dollar/yen exchange rate (DOLLAR.YEN).
[2] See Grinold (1999).
[3] See Zimmermann (1998, p.67, equation 4.2).
[4] See Sklar (1996).
[5] This model was introduced into the literature by Konno and Yamazaki (1991) and investigated further in Feinstein and Thapa (1993).
[6] See Hamza and Janssen (1995).
[7] Satchell and Sortino (2001) provide an excellent review on downside-based risk measures.
[8] This relationship is only true for continuous distributions, as we show in Section 5.6.1.
[9] See Young (1998).
[10] This section draws heavily on Acerbi and Tasche (2001).
[11] See Artzner et al. (1997).
[12] A similar example can be found in Acerbi et al. (2001).
[13] See Embrechts, McNeil, and Straumann (2002).
[14] The heuristic is described in Larsen, Mausser and Uryasev (2002)