6 Robust Statistical Methods for Portfolio Construction

6.1 Outliers and Non-normal Returns

The value of robust statistical methods in portfolio construction arises because asset returns and other financial quantities often contain **outliers**. Outliers are data values that are well-separated from the bulk of the data values and are not predicted by univariate or multivariate normal distributions. Under normal distribution models, such an outlier sometimes occurs with exceedingly small probability. For example, if we fit a normal distribution to S & P 500 daily returns for various periods of time prior to the stock market crash of 1987, we find that the probability of occurrence of an event of that magnitude is so small that one would have to wait much longer than the history of civilization for another such occurrence.[1] Large outliers of this type are not limited to situations with extreme market movements—one can find many such examples in individual asset returns. For example, the five-year monthly returns for the microcap stock with ticker EVST shown in Figure 6.1 has an extremely large outlier in December 1988 with value 6.88. You can make this plot with the S-PLUS commands given in Code 6.1, in which we first extract EVST monthly stock returns for a five-year span from the `microcap.ts` time series object, and then plot the EVST time series:

```
EVST.returns.ts <- microcap.ts[,"EVST"]
plot(EVST.returns.ts, plot.args = list(type = "b",
  pch = "."), reference.grid = F, ylab = "RETURNS",
  main = "EVST RETURNS")
# Add text "OUTLIER" by left-clicking mouse at
# desired location
text(locator(1),"OUTLIER")
# Add line by left-clicking at each line end-point,
# then right click
lines(locator())
# This command and the next are equivalent
```

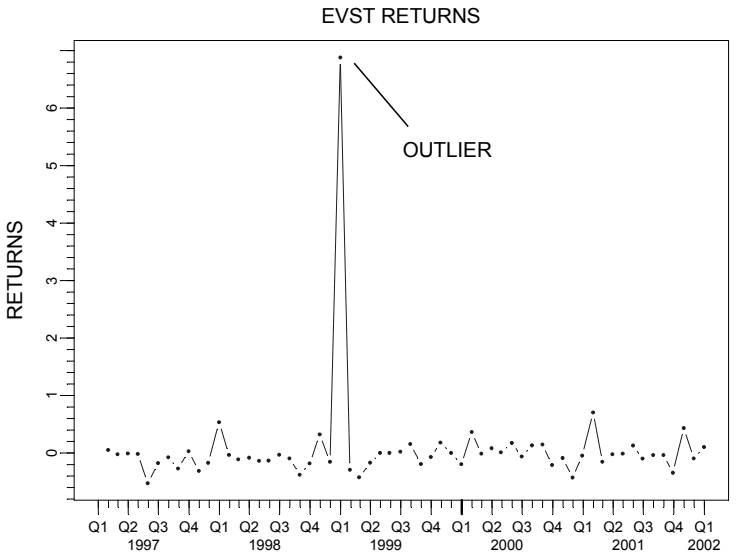

Figure 6.1 Time Series of EVST Returns

```
EVST.returns <- EVST.returns.ts@data[,1]
EVST.returns <- seriesData(EVST.returns.ts)[,1]
```

Code 6.1 Time Series Plot of EVST Returns

Note that `EVST.returns.ts` is an S-PLUS V4 time series object, the first part of which looks like:

```
> EVST.returns.ts
  Positions          EVST
  1/31/1997  0.050847456
  2/28/1997 -0.024193548
  3/31/1997 -0.008264462
         ...........
```

At the end of Code 6.1, the data from `EVST.returns.ts` is extracted and converted to a simple S-PLUS vector object.[2] You can now compute the mean and standard deviation of the EVST returns as follows:

```
> mean(EVST.returns)
[1] 0.07568058
> stdev(EVST.returns)
[1] 0.9197129
```

To get the value of the outlier and its time of occurrence, use the command

```
> EVST.returns.ts[EVST.returns.ts@data > 3,]
  Positions     EVST
 12/31/1998 6.878788
```

Now let's compute the probability of getting a return as large or larger than 6.88 for a normal distribution with mean .076 and standard deviation .92:

```
> 1-pnorm(6.88,.076,.92)
[1] 7.038814e-014
```

Under the normal distribution model, you would have to wait an unbelievable amount of time to see the recurrence of such an outlier in the monthly returns of EVST.

We can easily assess the non-normality of these returns using a normal Q-Q plot with a robustly fitted straight line, as shown in Figure 6.2.[3] (See Section 6.5 for a discussion of robust straight-line fitting in the context of estimating the CAPM beta.)

```
> qqnorm(EVST.returns, ylab="EVST.returns")
> qqline(EVST.returns)
```

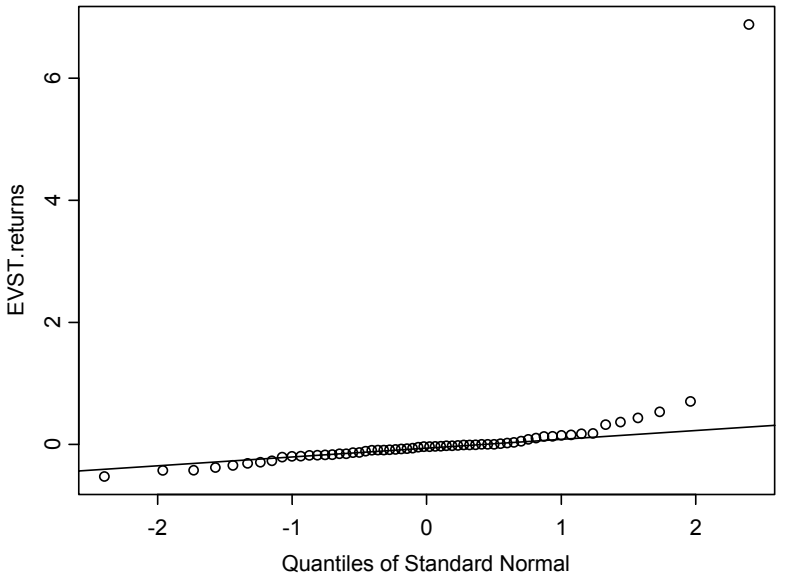

Figure 6.2 Normal Q-Q Plot of EVST Returns with Robust Line Fit

The normal Q-Q plot indicates that the returns are non-normal because of the single outlier and possibly because of the other deviations of the points from a

straight line. You can check this easily by making a normal Q-Q plot with the outlier deleted:

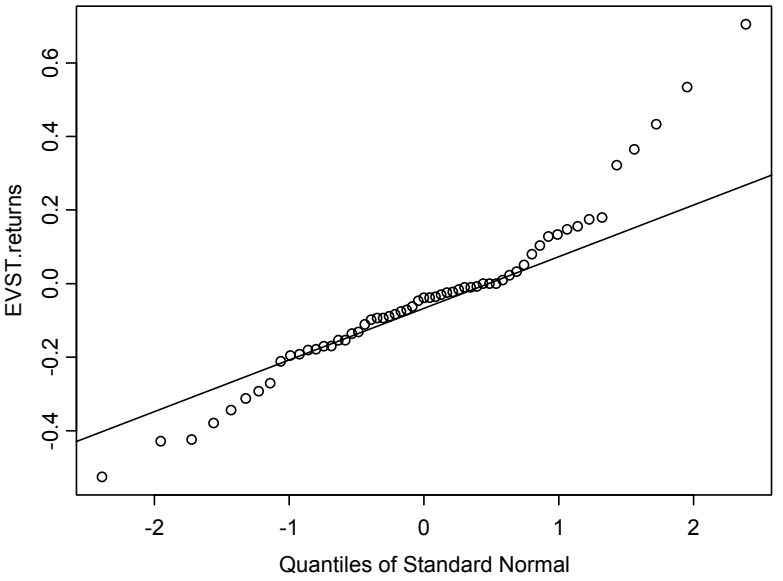

Figure 6.3 Normal QQ-Plot of EVST Returns with Outlier Removed

```
> qqnorm(EVST.returns[EVST.returns<2], ylab = "EVST
    Returns")
> qqline(EVST.returns[EVST.returns<2])
```

The result in Figure 6.3 shows that some outliers and non-normality are still present. How are we to judge non-normality from a Q-Q plot? The answer is to add 95% simulation confidence bands to plots like that of Figure 6.3, which you can do as follows:

```
> EVST.robfit <-
    lmRob(EVST.returns[EVST.returns<2]~1,eff=.95)
> plot(EVST.robfit,which.plots=2)
```

The function `lmRob` is a robust regression-fitting function that can estimate a mean robustly when used with a formula of the form `x ~ 1` as the first argument. The use of `lmRob` for robustly estimating mean returns will be discussed further in Section 6.3. It is used here only because the generic function plot invokes a special plot method for an `lmRob` object that computes and plots 95% simulation envelopes, which are useful for assessing whether residuals (the error term in the model) contain outliers.

Figure 6.4, showing the *standardized* residuals and the 95% simulation envelopes, indicates that there is still some non-normality in the form of incipient outliers (not quite outliers). This is due to a slightly heavier right-hand tail of the returns density.[4]

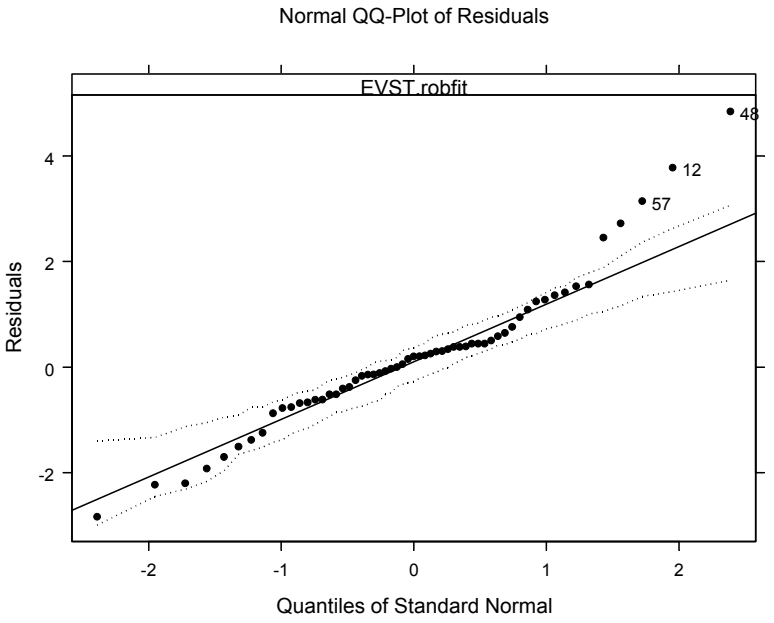

Figure 6.4 Normal Q-Q Plot with 95% Confidence Envelope

The single outlier in the EVST returns is highly **influential** in that it greatly increases the values of the mean and standard deviation relative to the values you would get if December 1988 were not an outlier. You can see the effect by doing the computation with the outlier deleted:

```
> mean(EVST.returns[EVST.returns < 2])
[1] -0.03962633
> stdev(EVST.returns[EVST.returns < 2])
[1] 0.2212708
```

The mean drops to –.040 and the standard deviation drops to .22. The latter is a little over four times smaller than the standard deviation that was calculated using all of the returns (.92). Consider the impact on the sample mean when you add the December 1998 return, assuming it was not originally there. A natural yardstick to measure the change is the standard deviation of the sample mean without the outlier, $.22/\sqrt{60} = .028$, so the influence of adding the outlier is to

shift the sample mean by $(.076 - (-.04))/.028 = 4.14$ standard deviations, which is a very large influence indeed.

It is well-known that mean returns are estimated poorly in that the standard deviation of the sample mean is typically a substantial fraction of the (absolute) value of the mean. In this example without the outlier, the standard deviation of the sample mean is .028, which is 70% of the size of the sample mean absolute value of .04. The single outliers increase the value of the sample mean by more than four standard deviations of the sample mean. *This example emphasizes (most painfully) that the problem of accurately estimating asset mean returns is greatly compounded by the presence of outliers*!

6.2 Robust Statistics versus Classical Statistics

One reason that we need to pay attention to outliers is that, as suggested by the example above, virtually all classical statistical parameter estimation and associated model-fitting methods lack **robustness** toward outliers. Even a single outlier can have an arbitrarily large adverse influence on classical model-fitting methods and classical statistical inference. Outliers can adversely influence not only mean and volatility estimates of returns but also covariance and correlation estimates, factor model parameter estimates, and optimal portfolio weights and related quantities such as Sharpe ratios. In data-oriented terms, a **robust statistical model-fitting method** is one that is not much influenced by outliers and provides a good fit to the bulk of the data.

A vivid example of the difference between a classical method and a robust method is shown in Figure 6.5, which displays a time series of annual earnings per share (EPS) from 1984 to 2001 for a company with ticker INVENSYS, along with two straight-line fits, one the classical least squares (LS) fit and one a highly robust fit (ROBUST). We describe the latter in Section 6.5 in the context of estimating the CAPM beta. It is quite clear that there are two outlier values of EPS. The LS line fit is highly influenced by these outliers and consequently does not provide a good fit to the bulk of the data, while the opposite is true of the robust fit, which is influenced very little by the outliers.

This particular example came to one of the authors from an analyst in the corporate finance office of a large, well-known firm. The analyst's task was to compute one-year-ahead forecasts of EPS for hundreds of firms as part of a portfolio stock-selection process. This example indicates clearly that use of LS through years 1998 and 1999 would have produced very poor predictions for years 1999 and 2000, respectively, and can be expected to produce a poor prediction for 2001. It is impossible to predict the time and direction of future outliers with any degree of accuracy, for if this were possible one could make a lot of money with an appropriate investment strategy. Thus, we cannot expect to

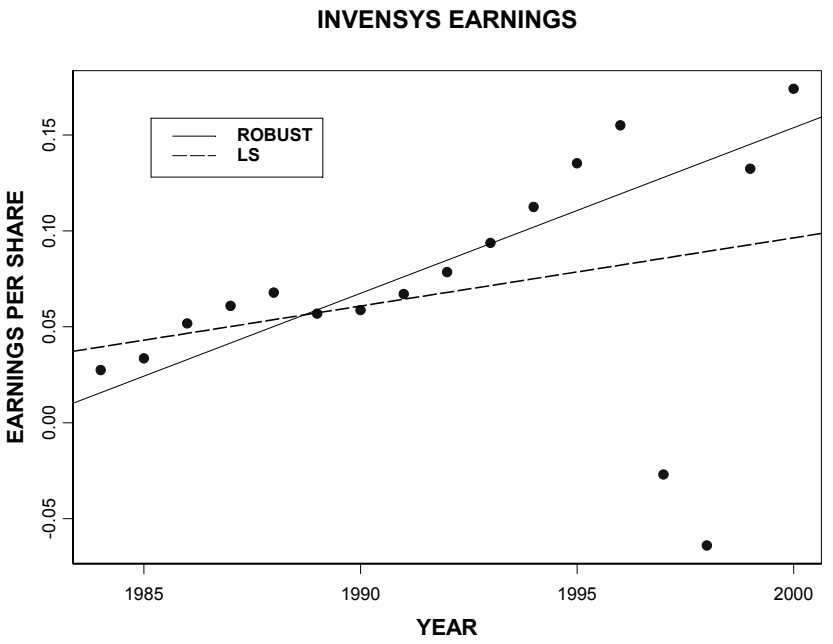

Figure 6.5 EPS versus Time with LS and Robust Line Fits

accurately predict the EPS outliers in 1997 and 1998. However, with a robust fit, we can compute good predictions for future data that are similar to the bulk of the outlier-free historical data.

While the data-oriented description of robust statistical model-fitting methods above has immediate appeal, it is important to know that there are rigorous probability-based statistical modeling foundations for robust statistics. These are probabilistic forms of stability of variance and minimization of bias under outlier-generating heavy-tailed deviations from a nominal (often normal) distribution model. An important approach for model parameter estimation is that of minimizing the maximum bias due to outlier contamination while at the same time achieving high statistical efficiency at the nominal model. For further details, see Martin and Zamar (1993) and Yohai and Zamar (1997).

So far, we have been talking about robustness of model parameter "point" estimates. It is important to note that robustness is also quite important with regard to methods of statistical inference such as hypothesis tests, confidence intervals, and model selection criteria. It turns out that outliers can seriously distort the level and power of a t test and the coverage probability and error rate of a confidence interval. A **robust hypothesis test** is one for which neither the nominal error rate nor the power of the test are much affected by the presence of outliers. A **robust confidence interval** is one for which neither the confidence level nor the expected confidence interval length are much affected by outliers.

The importance of robust methods in finance is immediately clear—one does not want an investment decision to be highly influenced by a small number of data points. When the historical asset price or returns data are of limited extent and exhibit at most a small number of outliers, it is typically impossible to predict with any degree of certainty whether there will be outliers in the future investment horizon.[5] In such situations, the bulk of the data is the only predictable part of the data, and an investment decision based on a robust statistical method may be preferred to the use of a procedure that is optimal under normality. On the other hand, robust statistics down-weight the influence of outlying returns, which reduces volatility estimates while reducing mean return estimates that have been influenced only by positive outliers and increasing mean return estimates that have been influenced only by negative outliers. Fund managers may rightly feel uncomfortable about making an investment decision based on a robust estimate. Such fund managers can nonetheless derive value from the use of robust methods as a diagnostic tool by comparing the results of the classical and robust methods. When both results agree, there is little worry about the possibility of outliers influencing the robust method, but when the two methods disagree substantially, the fund manager should be wary and look more closely at the data before making an investment decision.

6.3 Robust Estimates of Mean Returns

Suppose you have a set of identically distributed returns $(r_1, r_2, \cdots, r_n)$ with common mean $\mu = E(r_1)$. You can estimate μ with a variety of robust estimates that are influenced very little by outliers. The simplest and most transparent of these are the sample median and trimmed mean, both of which are based on the set of ordered returns $r_{1:n} \le r_{2:n} \le \cdots \le r_{n:n}$ (the **order statistics**). The **sample median** $\hat{\mu}_{MED}$ is the "middle" order statistic (i.e., the unique middle order statistic when n is odd, and the average of the two middle order statistics when n is even). For example, when $n = 11$, $\hat{\mu}_{MED} = r_{6:11}$, and when $n = 10$, $\hat{\mu}_{MED} = \frac{1}{2}(r_{5:11} + r_{6:11})$. An **α-trimmed mean**, $\hat{\mu}_{trim,\alpha}$, is computed by discarding a fraction α of the largest and smallest order statistic values and computing the sample mean of the remaining data points. For example, when $n = 10$, the 10% trimmed mean is

$$\hat{\mu}_{trim,\,.1} = \frac{1}{8}\sum_{i=2}^{9} r_{i:n}.$$

You can easily compute the median and trimmed mean in S-PLUS as illustrated below for the EVST returns.

```
> median(EVST.returns)
[1] -0.03863926
> mean(EVST.returns,trim=.1)
[1] -0.04756718
```

You can also compute the sample mean with the outlier deleted just as you did in Section 6.1:

```
> mean(EVST.returns[EVST.returns < 2])
[1] -0.0396
```

The result is almost identical to the sample median and not grossly different from the 10% trimmed mean.

While the simplicity of the median and trimmed means make them attractive robust estimators of location, they have some deficiencies that limit their general use. For example, it is not easy to construct confidence intervals for the sample median by any means other than bootstrapping, and the trimmed mean does not generalize nicely to other estimation problems such as fitting factor models. For this reason we introduce so-called **M-estimators** of location, a class of estimators that does generalize to many other model-fitting problems, including (as we shall see in Section 6.4) linear regression models.[6]

A location M-estimator $\hat{\mu}$ is a solution of the minimization problem

$$\min_{\mu} \sum_{t=1}^{n} \rho\left(\frac{r_t - \mu}{\hat{s}}\right), \tag{6.1}$$

where $\hat{s}$ is a robust scale estimate (see Section 6.4) and ρ is a "robustifying" loss function.

We obtain an **estimating equation** for $\hat{\mu}$ by differentiating the objective function (6.1) with respect to μ; this gives the M-estimator estimating equation

$$\sum_{i=1}^{n} \psi\left(\frac{r_t - \hat{\mu}}{\hat{s}}\right) = 0, \tag{6.2}$$

where $\psi = \rho'$. There is an intuitively appealing weighted least squares (WLS) interpretation of the M-estimator: if we set

$$w_t = w\left(\frac{r_t - \hat{\mu}}{\hat{s}}\right) \qquad w(u) = \frac{\psi(u)}{u}, \tag{6.3}$$

then we can rewrite the estimating equation (6.2) as a weighted least squares equation[7]

$$\sum_{t=1}^{n} w_t \cdot (r_t - \hat{\mu}) = 0. \tag{6.4}$$

When $\rho(t) = t^2$, the psi function is the identity function $\psi(t) \equiv t$, and the weights w_t are identically one. The solution is the least squares (LS) estimate of μ (i.e., the sample mean). The sample mean lacks robustness because the quadratic character of the least squares loss function causes outliers to have undue influence on the estimate. A robust estimate is obtained by using a ρ that grows more slowly than a quadratic function. The two main choices are: (a) a $\rho(t)$ that grows like $|t|$ for large *t*, and (b) a bounded $\rho(t)$. It is known that the former choice does not provide bias robustness (Martin, Yohai, and Zamar, 1989). Thus, we use a bounded $\rho(t)$ that was shown by Yohai and Zamar (1997) to be optimally bias-robust, subject to a constraint of specified efficiency, when the data are normally distributed. This $\rho(t)$ is implemented in the S-PLUS function `lmRob`. This type of $\rho(t)$ (RHO) is graphed in Figure 6.6 along with the corresponding $\psi(t)$ (PSI).

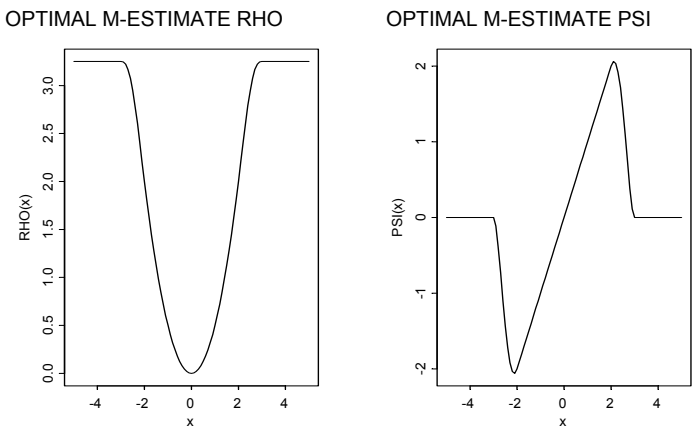

Figure 6.6 Optimal Bias Robust Rho and Psi Functions for 90% Gaussian Efficiency

The formula for the rho function $\rho(t)$ is

$$\rho(u) = \begin{cases} 0.5u^2, & |u| \le 2c \\ c^2\left(1.792 - .972\left(\frac{u}{c}\right)^2 + .432\left(\frac{u}{c}\right)^4 - .052\left(\frac{u}{c}\right)^6 + .002\left(\frac{u}{c}\right)^8\right), & 2c \le |u| \le 3c \\ 3.25c^2, & |u| \ge 3c. \end{cases}$$

The formula for the psi function $\psi(t)$ is easily obtained by differentiation. The role of the tuning constant c is to adjust the efficiency of the estimate to the desired level when the returns are normally distributed. **Efficiency when the returns are Gaussian** is defined to be the variance of the least squares (LS) estimator divided by the variance of the robust estimator. The graphs in Figure 6.6 show the ρ and ψ functions for an efficiency of 90%. This corresponds to an 11% increase in variance (only 3.3% in terms of increase in standard deviation) of the robust estimate over that of the least squares estimate (the sample mean in the present discussion). This increase in variance at nominal Gaussian returns is, in effect, a small "insurance premium" paid in exchange for protection against bias and inflated variance in the presence of outliers. The higher the premium paid, the more protection we get. If we require a Gaussian efficiency larger than 90%, the bound on the ρ grows, and the cutoff points where the ψ function goes to zero retreat further toward infinity. In the limit, when we require 100% Gaussian efficiency, the loss function becomes quadratic and we get the LS estimator.

The weight function associated with the optimal ψ function for 90% Gaussian efficiency has the shape shown in Figure 6.7. Note that the weight function is zero outside a finite interval; this means that the M-estimator will put zero weight w_t on any return r_t that has sufficiently large scaled residuals $\hat{\varepsilon} = (r_t - \hat{\mu})/\hat{s}$ (i.e., the outliers will be totally rejected). Returns whose scaled residuals are sufficiently small (typically the bulk of them) will get weights w_t equal to one.

The S-PLUS function `lmRob` uses a sophisticated form of a nonlinear optimization method proposed by Yohai, Stahel, and Zamar (1991) to solve the M-estimate minimization problem (6.1). `lmRob` was designed for robustly fitting a linear regression (factor) model and computing associated robust statistical inference quantities such as robust standard deviations, t-statistics, and p-values. As a special case, `lmRob` can compute a robust estimate of mean returns (in which we have only an intercept term). We compute a robust M-estimate of the mean and its robust standard deviation, and a plot of the weights, using Code 6.2.

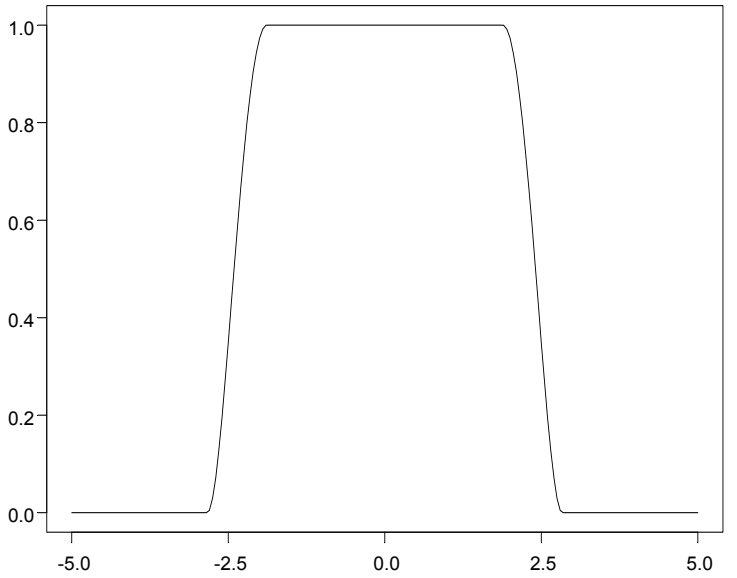

Figure 6.7 Optimal Weight Function for 90% Gaussian Efficiency

```
EVST.mean <- lmRob(EVST.returns~1, eff=.95)
coef(EVST.mean)
sqrt(EVST.mean$cov)
EVST.M.weights <- timeSeries(EVST.mean$M.weights,
  from="1/1/1997",by="months")
plot(EVST.M.weights,plot.args=list(type = "p"),
  reference.grid=F, xlab="TIME",ylab=WEIGHTS")
```

Code 6.2 Robust Location Estimate Weights

Use of the `eff = .95` argument gives us a robust estimate that would have 95% efficiency if the returns were normally distributed. In the weights plot of Figure 6.8, we see that the huge outlier in Q4 1988 gets zero weight, as do a few other large returns; this is not surprising in view of Figure 6.4.

We can repeat the commands above with a higher efficiency, say `eff =.98`, to perform a less severe down-weighting of data. This changes the robust mean returns estimate to –.066 and the standard deviation to .028 (which is not much of a change). The new weights, shown in Figure 6.9, are not very different from those for 95% Gaussian efficiency (shown in Figure 6.8). Note that as we change the efficiency, we change the cut-off values of the weight function in Figure 6.7. Higher efficiencies result in larger cut-off values and less severe rejection of outliers, while lower efficiencies result in smaller cutoff values and

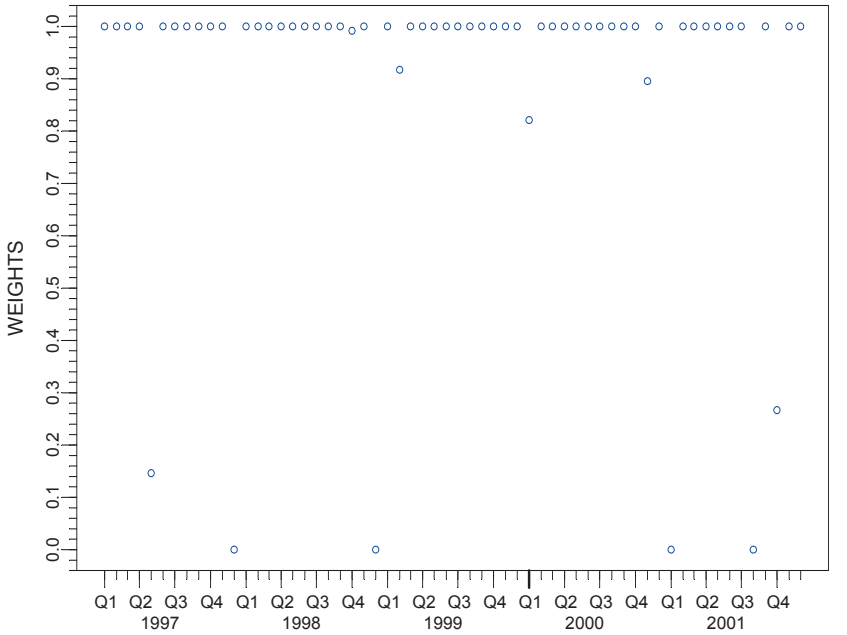

Figure 6.8 Robust M-Estimate Weights (95% Gaussian Efficiency)

more severe rejection of outliers. You are encouraged to try some lower values of efficiency, say .90 and .85. If you do so, you will see that while you get a few more zero weights and some other weights smaller than one, the robust estimate of mean returns and its standard deviation do not change much. This relative insensitivity of the robust estimate of mean returns with respect to the Gaussian efficiency/cutoff values is an attractive feature of the method.

There is also an S-PLUS function `location.m` that computes only a robust M-estimate of location (the mean); i.e., it does not compute a standard error like `lmRob`. It uses a somewhat different weight function (the Tukey biweight) that still gives weight zero to all sufficiently large residuals. With the previous data it gives a location estimate of –.06 instead of –.04, a difference less than one standard error (.28) of the sample mean without the large outlier:

```
> location.m(EVST.returns)
[1] -0.0597
```

Here are our recommendations on which estimator to use:

1. If experience with your data tells you that you have only a certain fraction of outliers, and you do not need a standard error estimate,

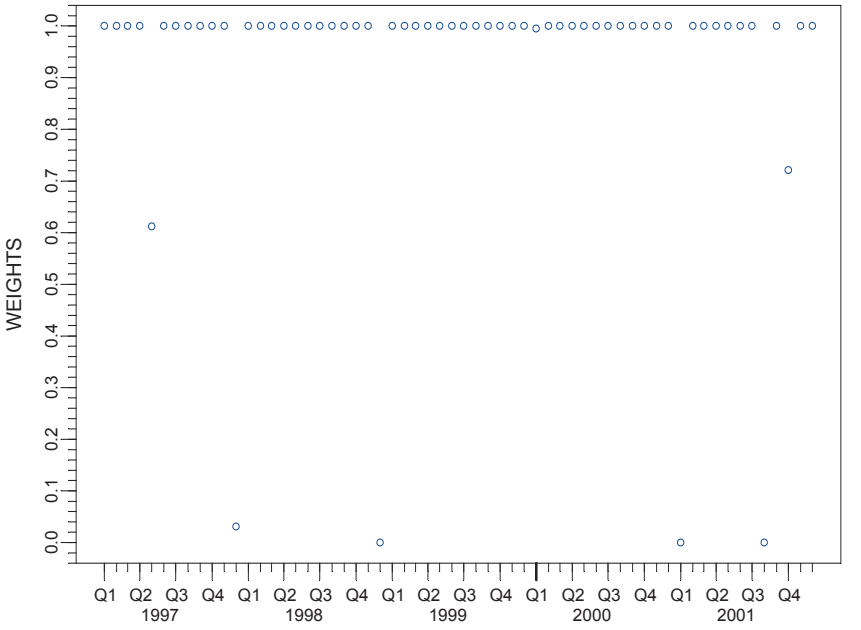

Figure 6.9 Robust M-Estimate Weights (98% Gaussian Efficiency)

use the trimmed mean `mean(returns, trim = alpha)` with trimming fraction `alpha` slightly greater than your worst case estimate of the fraction of outliers.

2. If you don't know much about the fraction of outliers and don't need a standard error, use `location.m`.
3. If you need a standard error estimate, use `lmRob` with the default Gaussian efficiency of 90%.

In S-PLUS, we can easily compute robust estimates of the mean returns for a collection of stocks with any of the robust mean (location) estimates above by using the `apply` function on the data frame of returns. To illustrate, we compute robust location estimates for five large cap stocks:

```
> rob.means <- apply(largecap.ts[, 3:8]@data,2,
     location.m)
> round(rob.means, 3)
   CAT    DD     G  GENZ    GM   HON
 0.007 0.003 0.01 0.038 0.009 0.02
```

6.4 Robust Estimates of Volatility

It is well-known that standard deviations (volatilities) of stock returns are usually estimated more accurately than means. Hence, there has been a tendency to assume that accuracy of estimation of standard deviations is not a problem. As the example in Section 6.1 clearly shows, however, the presence of even a single outlier can cause a dramatic change in the value of the standard deviation (in that particular case, a little over a fourfold increase from .22 if the outlier were not present to .92 with the outlier present).

There are several robust scale estimator functions in S-PLUS that provide robust estimates of returns volatilities and are approximately unbiased estimates of the standard deviation when the returns are normally distributed. These are the functions `mad` (median absolute deviation about the median), `scale.a`, and `scale.tau`. For the EVST returns, these functions give the following results:

```
> mad(EVST.returns)
[1] 0.1708058
> scale.a(EVST.returns)
[1] 0.1717936
> scale.tau(EVST.returns)
[1] 0.1864585
```

These values are all similar, and any one of the estimates can be used safely. Since the `mad` is the simplest and most transparent, it can be used as a default (though in some applications the smoothness properties and conceptual transparency of `scale.tau` might be preferred). See the S-PLUS help files for more details on these robust scale estimates.

6.4.1 Robustness Is Not Enough for Risk Management

Figure 6.10 shows the time series of ZIF returns along with a histogram and two normal density estimates of the returns. Clearly, the ZIF returns exhibit several outliers.

Code 6.3 gives the S-PLUS script to make the plot.

```
par(mfrow = c(2, 1))
ZIF.returns.ts = microcap.ts[,"ZIF"]
plot(ZIF.returns.ts, plot.args = list(type = "b",
  pch = "."), reference.grid = F, ylab = "RETURNS",
  main = "ZIF RETURNS")
```

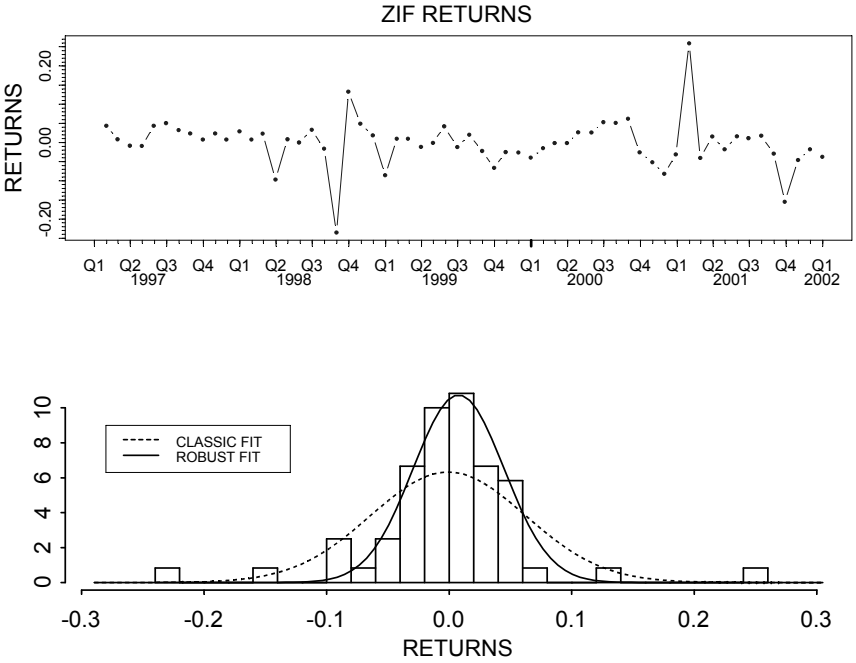

Figure 6.10 Time Series of ZIF Returns and Histogram and Two Density Estimates

```
returns = seriesData(ZIF.returns.ts)[,1]

mu = mean(returns)
sigma = stdev(returns)
mu.rob = median(returns)
sigma.rob = mad(returns)
k = 8

xlim = c(mu.rob - k * sigma.rob,
  mu.rob + k * sigma.rob)
hist(returns, nclass = "fd", col = 0, prob = T,
  xlim = xlim, xlab = "RETURNS")
x = seq(xlim[1], xlim[2], length = 100)
lines(x, dnorm(x, mu, sigma), lty = 8, lwd = 2)
lines(x, dnorm(x, mu.rob, sigma.rob), lwd = 2)
leg.names = c("CLASSIC FIT", "ROBUST FIT")
legend(-.28,9,legend=leg.names,lty=c(8,1),lwd=2)
par(mfrow = c(1, 1))
```

```
mu
mu.rob
sigma
sigma.rob
```

Code 6.3 Classical and Robust Normal PDF Fits

The classical and robust estimate values computed by the script above are:

```
> mu
[1] -0.0008619089
> mu.rob
[1] 0.007695105
> sigma
[1] 0.06311233
> sigma.rob
[1] 0.03714824
```

Suppose you want to calculate a 1% value-at-risk (VaR) (i.e., the lower 1% point of the fitted distribution). It is obvious from Figure 6.10 that neither the classical nor the robust fit of the normal distribution will suffice for this purpose. Let's examine the plots a bit more carefully.

Notice that although the mean and median (MED) differ, they are both rather close to zero because the positive and negative outliers tend to balance. On the other hand, the classical standard deviation is considerably larger than the median absolute deviation about the median (MAD). The dashed line is a normal density estimate based on the mean and standard deviation. The solid line is a normal density estimate based on MED and MAD. Because the mean and MED are quite close, both densities are well-centered on zero. But because of the difference between the standard deviation and MAD, the normal density based on the latter fits the bulk of the data well but fails to adequately describe the tails. This is to be expected since the data require a heavy-tailed density description. On the other hand, the normal density (fit with the classical estimates) is a very poor fit to the center of the data, but because of the inflated standard deviation estimate does a better job of estimating the tails (though it is still not good enough). This behavior of the normal fit explains why some value-at-risk (VaR) calculations are not so bad under normality at the 95% level but are quite inaccurate for 99% VaR. The robust MED and MAD indeed give you good estimates of the location and scale of the returns. But they are not a solution for getting better tail estimates! If you want good tail probability estimates, you need to fit a heavy-tailed distribution. Often for smallish sample sizes (such as five years of monthly data) a normal mixture with two or three components will do. For larger data sets (such as a year or more of daily data), one can do well by fitting stable distributions whose fat tails provide a good model for outliers (see Rachev and Mittnik, 2000).

6.4.2 Robust EWMA Estimates of Volatility

It is an overlooked fact that exponentially weighted moving average (EWMA) estimates often badly overestimate volatility following the occurrence of an isolated outlier return. Fortunately, it is quite easy to obtain a robust version of an EWMA volatility estimate by a simple modification of the classical EWMA algorithm, as we describe below. Figure 6.11 illustrates the dramatic difference between classical and robust EWMA estimates using two years of daily returns for a mid-cap stock with ticker ROH. The time series of returns, shown in the top panel of Figure 6.11, clearly reveals several outliers indicative of unusual movements in the price series. The middle panel shows a classical EWMA volatility time series estimate using a default smoothing parameter of .93. The classical EWMA estimate clearly grossly overestimates the volatility following the occurrences of the outliers, particularly after the two negative outliers in early Q2 and Q3 of 2000 and the outliers near the end of Q3 of 2001. The robust EWMA volatility estimate, shown in the bottom panel of Figure 6.11, does not suffer from this defect and produces much more reasonable-looking estimates of the volatility following the outliers.

The classical EWMA volatility estimate is the standard deviation series obtained as the square root of the variance estimates computed by the recursion

$$\hat{\sigma}_{t+1}^2 = \lambda \cdot \hat{\sigma}_t^2 + (1-\lambda) \cdot r_{t+1}^2, \qquad t \geq t_0. \tag{6.5}$$

Here t_0 is a starting time and $\hat{\sigma}_{t_0}^2$ is an initial variance estimate. We can easily construct a robust EWMA volatility estimate that in turn provides a robust unusual movement test (UMT) statistic with robustness of power as well as level of test. The robust EWMA algorithm is defined as follows:

$$\begin{aligned} \hat{\sigma}_{t+1}^2 &= \lambda \cdot \hat{\sigma}_t^2 + (1-\lambda) \cdot r_{t+1}^2, \quad \textit{if} \quad |r_{t+1}| \leq a \cdot \hat{\sigma}_t \\ &= \hat{\sigma}_t^2, \qquad\qquad\qquad\quad \textit{if} \quad |r_{t+1}| > a \cdot \hat{\sigma}_t . \end{aligned} \tag{6.6}$$

The parameter a is a rejection threshold. We recommend a default value of a = 2.5 for the rejection threshold; this results in using a pure prediction $\hat{\sigma}_{t+1}^2 = \hat{\sigma}_t^2$ about 1.2% of the time when the returns are normal and $\hat{\sigma}_t^2$ is equal to the true volatility of the return at time t.

In addition to having a good robust volatility estimate, one would sometimes like to have a good test statistic for providing an alert that a return is an outlier (and correspondingly that the asset price has made an unusually large movement). It is natural to use the following statistic, similar in form to a classical two-sided t-test:

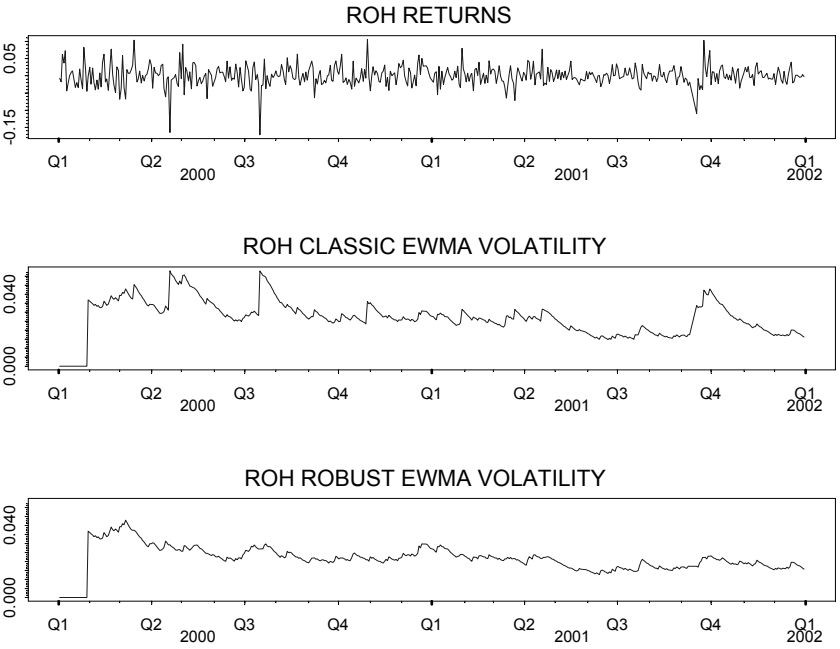

Figure 6.11 Time Series of Returns and Estimates of Volatility for ROH

$$UMT_t = \frac{|r_t|}{\hat{\sigma}_t}, \tag{6.7}$$

where r_t is the asset return at time t and $\hat{\sigma}_t$ is the classical EWMA estimate. This statistic is called the **classical UMT statistic**. However, since we know that the classical EWMA estimate overestimates volatility at times following an outlier, the denominator of the statistic will be larger than it would be without an outlier present. Thus we might anticipate it will result in a UMT with low power for detecting an unusual movement.[8] This suggests that one might obtain a robust UMT statistic by substituting the robust EWMA estimate $\hat{\sigma}_t$ in the denominator. Results for the classical and robust UMT's for the ROH returns are shown in Figure 6.12. The horizontal dashed line at $c = 3.5$ in Figure 6.12 is a test rejection threshold chosen to yield a false alarm rate of approximately .001. The classical UMT fails to detect any outlier returns or unusual movement in prices, while the robust UMT clearly detects the five largest outliers in Figure 6.11 and gives a weak indication of two others.

We remark that if the UMT test statistic had a standard normal distribution, a rejection threshold of c = 3.29 would yield a false alarm rate of .001. However, since this statistic is rather like a t-test, one has to naturally question the accuracy of a standard normal approximation. To answer this question, we make the Q-Q plot in Figure 6.13. This plot shows that the robust UMT statistic

follows the normal distribution closely, with the exception of a few outliers and straggling values.

This leads us to use a normal distribution as a crude approximation for purposes of computing a threshold value. We fit this normal distribution to the data with robust location and scale estimates. The median estimate of location turns out to be exactly zero (and the mean is essentially equal to the median, with value .0002), while the robust scale estimate is 1.068 (as compared with the standard deviation value of 1.21, which is inflated by the outliers). The upper .0005 quantile of an $N(0,(1.068)^2)$ distribution is 3.51, giving a false alarm rate of .001 for the two-sided test. These values are not quite fair because they were computed post hoc. But in practice one could apply the same approach using data prior to the times at which the test statistics were computed.

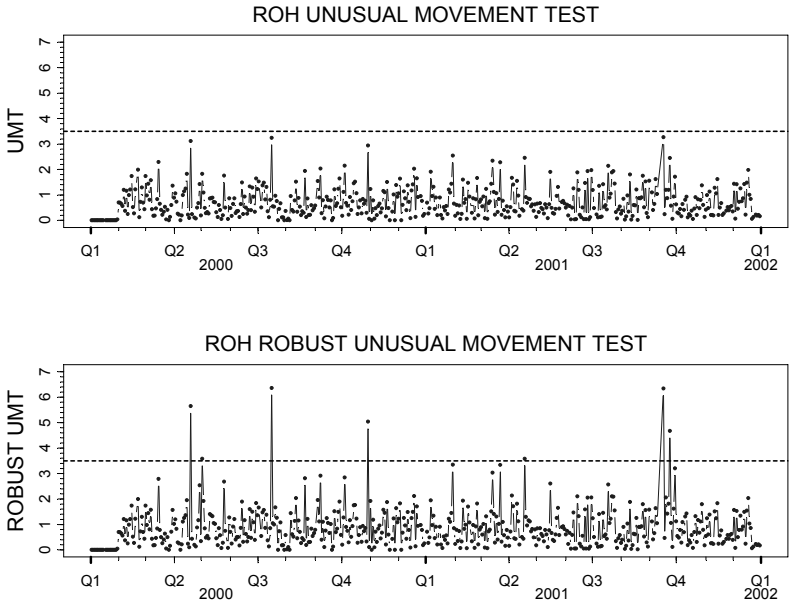

Figure 6.12 Classical and Robust UMT Test Statistics Time Series

Code 6.4 and Code 6.5 give an EWMA function and the script, respectively, for making the above computations and plots above.

```
ewma <- function(x, robust = T, lambda = 0.93,
  nstart = 20, a = 2.5)
{
  n <- length(x)
```

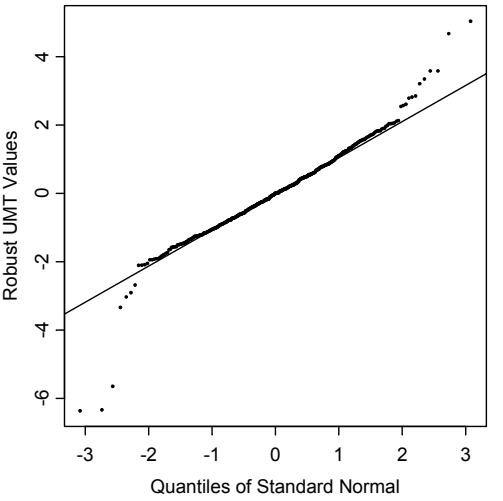

Figure 6.13 Robust EWMA and UMT for ROH

```
  # Compute initial variance estimate var.start
  var.start <- scale.tau(x[1:nstart])^2

  # Create output vector with padded zero's and
  # var.start
  varvec <- c(rep(0, nstart - 1), var.start,
     rep(0, n - nstart))

  # EWMA recursion
  var.old <- var.start
  ns1 <- nstart + 1
  for(i in ns1:n) {
     r2 <- x[i]^2
     if(robust && r2 > a^2 * var.old)
        r2 <- var.old
     var.new <- lambda * var.old +
        (1 - lambda) * r2
     var.old <- var.new
     varvec[i] <- var.new
  }
  varvec^0.5
}
```

Code 6.4 Function to Compute Classical and Robust EWMA

```
ticker <- "ROH"
tsdata <- midcapD.ts[,ticker]
returns <- tsdata@data[,1]
n <- length(returns)
lambda <-.93
nstart <- 20
a <- 2.5
thresh <- 3.5
vol.classic <- ewma(returns, robust=F,
     lambda=lambda, nstart=nstart,a=a)
vol.classic
vol.rob <- ewma(returns, lambda=lambda,
     nstart=nstart, a=a)
vol.rob
ylim <- range(vol.classic, vol.rob)
vol.classic.ts <- timeSeries(vol.classic,
  positions = tsdata@positions)
vol.rob.ts <- timeSeries(vol.rob,
  positions = tsdata@positions)
par(mfrow = c(3,1))
plot(tsdata[,ticker],reference.grid = F,
  main = paste(ticker,"RETURNS"))
plot(vol.classic.ts,ylim = ylim, reference.grid =
     F, main = paste(ticker,"CLASSIC EWMA
     VOLATILITY"))
plot(vol.rob.ts,reference.grid = F, ylim = ylim,
  main = paste(ticker,"ROBUST EWMA VOLATILITY"))
# Compute UMT
par(mfrow = c(2, 1))
umt.classic <-
  abs(returns[nstart:n])/vol.classic[nstart:n]
umt.classic <- c(rep(0,nstart-1),umt.classic)
umt.classic.ts = timeSeries(pos = tsdata@positions,
  data = umt.classic)
umt.robust <-
     abs(returns[nstart:n])/vol.rob[nstart:n]
umt.robust <- c(rep(0,nstart-1),umt.robust)
umt.robust.ts <- timeSeries(pos=tsdata@positions,
  data = umt.robust)
ylim <- 1.1*c(0,max(thresh,
  max(umt.classic,umt.robust)))
plot(umt.classic.ts, plot.args=list(type="b",
  pch="."), reference.grid=F, ylim=ylim,
  ylab="UMT",
  main=paste(ticker,"UNUSUAL MOVEMENT TEST"))
```

```
abline(thresh, 0,lty=8)
plot(umt.robust.ts, plot.args=list(type="b",
  pch="."), reference.grid=F, ylim=ylim,
  ylab="ROBUST UMT", main=paste(ticker,
  "ROBUST UNUSUAL MOVEMENT TEST"))
abline(thresh, 0,lty=8)
par(mfrow=c(1, 1))
# Q-Q plot and threshold estimate
par(pty="s")
umt.rob.signed <-
     returns[nstart:n]/vol.rob[nstart:n]
qqnorm(umt.rob.signed,
  ylab="Robust UMT Values",pch=".")
qqline(umt.rob.signed)
mean(umt.rob.signed)
sigma <- stdev(umt.rob.signed)
sigma
abs(qnorm(.0005,0,sigma))
sigma.rob <- scale.tau(umt.rob.signed)
mean(umt.rob.signed)
sigma.rob
abs(qnorm(.0005,0,sigma.rob))
par(pty = "")
```

Code 6.5 Compute and Plot Classical and Robust EWMA and UMT

We remark that the detection power of the classic UMT might be improved by using $\hat{\sigma}_{t-1}$ in the denominator instead of $\hat{\sigma}_t$. This should clearly improve the ability to detect isolated outliers, but it may not suffice to detect any additional outliers that follow in close time proximity to a first outlier. The reader could check this out by modifying Code 6.5 to use $\hat{\sigma}_{t-1}$ in place of $\hat{\sigma}_t$ in the test statistic (Exercise 5).

It is apparent that the robust EWMA volatility estimate has many potential uses in portfolio construction and risk management calculations, as would potential extensions to robust EWMA covariance matrix and mean return estimates. Clearly, in a time period subsequent to an isolated outlying return (positive or negative), one does not want to rebalance a portfolio based on a volatility estimate that grossly overestimates the true volatility and a covariance matrix estimate that is quite distorted by the outlying return. Generalizations of the robust UMT could be used to detect regime shifts from good times to bad times and vice versa.

6.5 Robust Betas

Beta estimates for assets are often used by portfolio managers to decide whether an asset should be added to a portfolio to increase or decrease its beta. It is therefore important to have reliable beta estimates that accurately reflect the risk and return characteristics of the assets under scrutiny. This section examines the impact of outliers on beta estimates and shows that even a single outlier in an asset's returns can adversely influence the conventional estimates of beta, which gives a completely misleading picture of the asset's risk and return characteristics.

CAPM[9] betas are typically estimated by fitting the single-factor market model

$$r_t = \alpha + r_{M,t}\beta + \varepsilon_t, \qquad t = 1, 2, \cdots, n, \tag{6.8}$$

where r_t is the return on an asset or portfolio at time t, $r_{M,t}$ is the market return at time t, and ε_t is the error term in the model. In the United States, the market return is often taken to be the return on a value-weighted index of stocks from the NASDAQ, New York, and American stock exchanges. The parameter estimates $\hat{\alpha}$ and $\hat{\beta}$ are obtained by fitting a straight line to the scatterplot of r_t versus $r_{M,t}$ by some "good" method. The sanctified "good" method is that of (ordinary) least squares (LS), i.e., $\hat{\alpha}$ and $\hat{\beta}$ are obtained by minimizing the sum of squared residuals

$$\sum_{t=1}^{n} \left(r_t - \alpha - r_{M,t}\beta \right)^2. \tag{6.9}$$

This is often (but not always) good enough for large cap stocks, as the Microsoft example in Figure 6.14 and Figure 6.15 shows. The first of these figures shows the monthly time series of Microsoft returns and market returns, while the second displays both the LS and the robust straight-line fits and corresponding beta estimates. The LS and robust betas are quite close to one another, both on a relative basis and with respect to the ordinary LS standard error value of .28.

The robust beta is computed using the optimal bias robust regression M-estimate method described in Section 6.3, with $r_t - \mu$ replaced by $r_t - \alpha - r_{M,t}\beta$. This is the method implemented by the function `lmRob`. The time series plots of Figure 6.14 and the classic and robust beta computations for Figure 6.15 can be replicated using Code 6.6.

```
mkt.ret.ts <- largecap.ts[,"market"]
ticker <- "MSFT"
stock.ret.ts <- largecap.ts[,ticker]
par(mfrow = c(2,1))
plot(stock.ret.ts,plot.args = list(type = "b",
  pch = "."), reference.grid = F, ylab = "RETURNS",
  main = ticker)
plot(mkt.ret.ts,plot.args = list(type = "b",
  pch = "."), reference.grid = F, ylab = "RETURNS",
  main = "MARKET")
par(mfrow = c(1,1))
par(pty = "s")
mkt.ret <- mkt.ret.ts@data[,1]
stock.ret <- stock.ret.ts@data[,1]
plot(mkt.ret,stock.ret, xlab = "MARKET RETURNS",
  ylab = paste(ticker,"RETURNS"))
beta.ls <- lm(stock.ret ~ mkt.ret)
abline(beta.ls,lty = 8)
beta.rob <- lmRob(stock.ret ~ mkt.ret)
abline(beta.rob)
text.ls <- as.character(round(coef(beta.ls),2)[2])
text.ls <- paste("LS BETA =",text.ls)
text.rob <-
     as.character(round(coef(beta.rob),2)[2])
text.rob <- paste("ROBUST BETA =",text.rob)
legend(-.15,.37,c(text.ls,text.rob),lty = c(8,1))
par(pty = "")
```

Code 6.6 Classic and Robust Betas

It can happen, of course, that the returns of a stock contain one or more highly influential outliers that adversely influence the LS beta. This behavior is particularly prevalent in small cap and microcap stocks and is vividly illustrated in Figure 6.16 for the microcap stock EVST, whose time series of returns contains one very large outlier value representing a return of close to 700% (recall Figure 6.1). You can produce Figure 6.16 by replacing `largecap.ts` by `microcap.ts` and `ticker = "MSFT"` with `ticker = "EVST"` in Code 6.6.

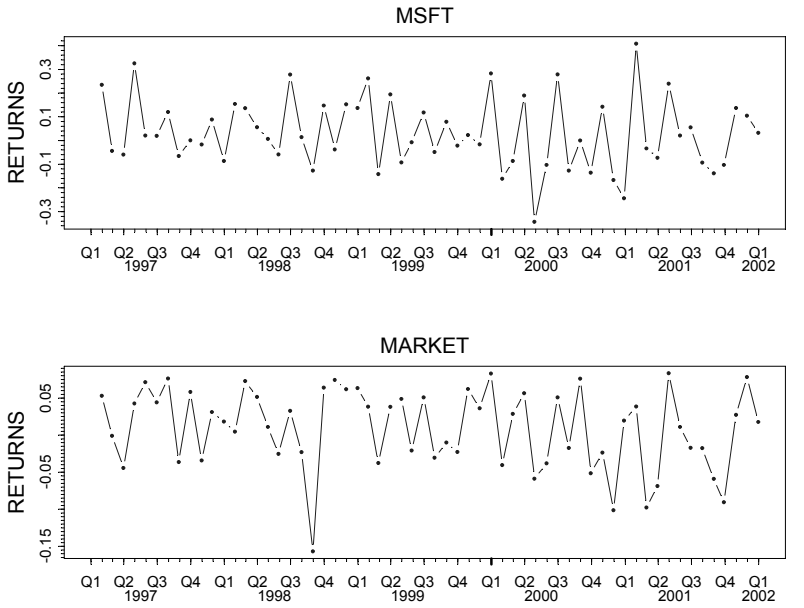

Figure 6.14 Microsoft and Market Monthly Returns for Five Years

Here the single outlier causes the LS line to fit the data quite poorly when the market returns are positive, whereas the robust line fit is not greatly affected by the outlier and fits the bulk of the data quite well. Note that in this example the outlier resulted in an LS beta of 3.17; a consumer of the LS beta would think that EVST has a high level of risk and high expected excess return relative to the market, even though this conclusion rests on a single data point (and one that was an amazingly high return at that). On the other hand, the robust beta value of 1.13 indicates that EVST behaves for the most part like the market, which is essentially the conclusion one would draw if the outlier were deleted.

Suppose that instead of the raw $\hat{\beta}$ estimates one computed an adjusted beta according to the fossilized shrinkage formula sometimes used by commercial financial data service providers,

$$\tilde{\beta} = .33 + .67 \cdot \hat{\beta}, \tag{6.10}$$

where $\hat{\beta}$ is either the LS or robust beta estimate.[10] This gives $\tilde{\beta}_{LS} = 2.45$ and $\tilde{\beta}_{ROBUST} = 1.09,$ respectively. The influence of the outlier would be reduced, but this adjustment would not solve the problem with the LS estimate.

It is a rather surprising fact that most commercial providers of beta estimates appear to be totally unaware of the impact of outliers on the betas that they deliver. This is documented in Martin and Simin (2003), who found that out of

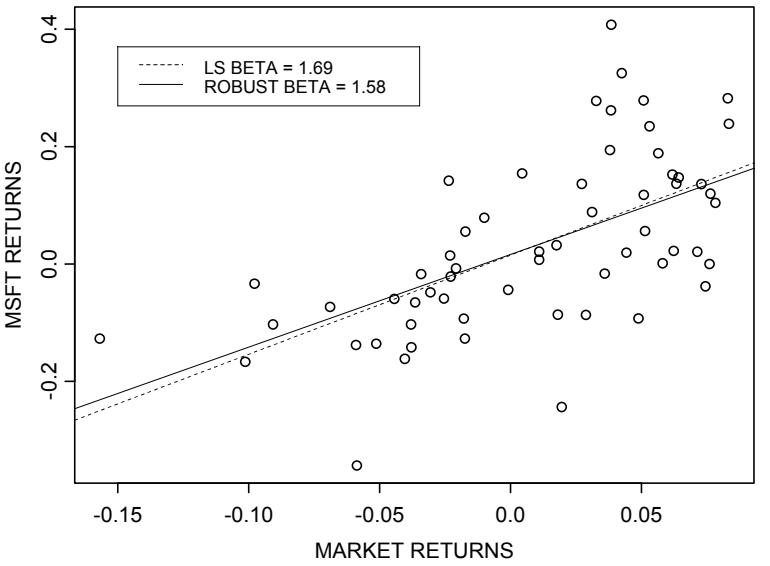

Figure 6.15 Least Squares and Robust Betas for Microsoft

the nine commercial providers of beta estimates that they surveyed, only two appeared to be aware of the issue.[11]

In order to more fully evaluate the value of a robust beta over an LS beta, one needs to know whether robust betas predict robust betas better than LS betas predict LS betas. Martin and Simin (2003) answer this question in the affirmative, giving further support to the use of robust betas in practice. Our recommendation is to compute both LS and robust betas and signal an alert that one or more outliers are probably influencing the LS estimate whenever the difference between the two estimates is larger than a user-supplied threshold. In this case, the provider should supply additional information such as a time series plot of returns, the time(s) of occurrence of the outlier(s), and potentially important related information such as corporate announcements, etc.

6.6 Robust Correlations and Covariances

In this section, we show that multivariate outliers in asset returns can have a substantial influence on correlation and covariance matrix estimates, and that one can use robust covariance matrix estimates to accurately measure the covariance and correlation structure of the bulk of the data. Figure 6.17 shows time series of 81 monthly returns for the following assets: U.S. alternative investments in DM (AI), high-quality German mortgage bonds (Pfand), U.S.

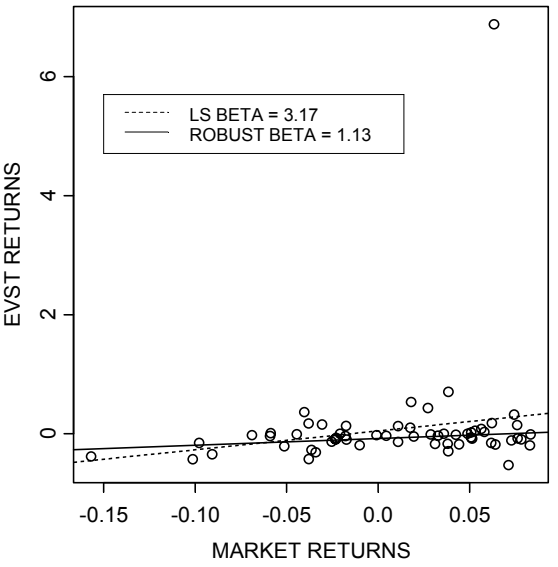

Figure 6.16 Least Squares and Robust Betas for EVST

private equity hedged in DM (PEHinDM), and U.S. high-yield bonds hedged in DM (USHYHinDM).[12]

The series of returns appear to have some distinct volatility regimes over time, and possibly a few outliers. For example, the PEHinDM and USHYHinDM series have relatively low volatility during the early time periods, while the Pfand series seems to have a lower volatility near the end of the series. The PEHinDM returns have an outlier during Q3 1998, and USHYHinDM appears to have an outlier in each of Q3 and Q4 1998 as well as an outlier in early 2001. The pairwise scatterplots in Figure 6.18 reveal some clear outliers and deviations from the elliptical shape of a multivariate normal distribution.

Figure 6.17 is a Trellis time series plot made with a modified version of the Trellis time series plotting function `seriesPlot` that comes with the S-PLUS add-on module S+FinMetrics.[13] Use the commands in Code 6.7 to make the Trellis time series plot and the pairwise scatterplots.

```
data.ts <- normal.vs.hectic.ts[-(1:60),2:5]
data <- seriesData(data.ts)
y.name <- colIds(data.ts)
seriesPlot(data.ts,one.plot=F,
  strip.text=y.name,col=1)
pairs(data)
```

Code 6.7 Trellis Time Series Plots and Pairwise Scatterplots

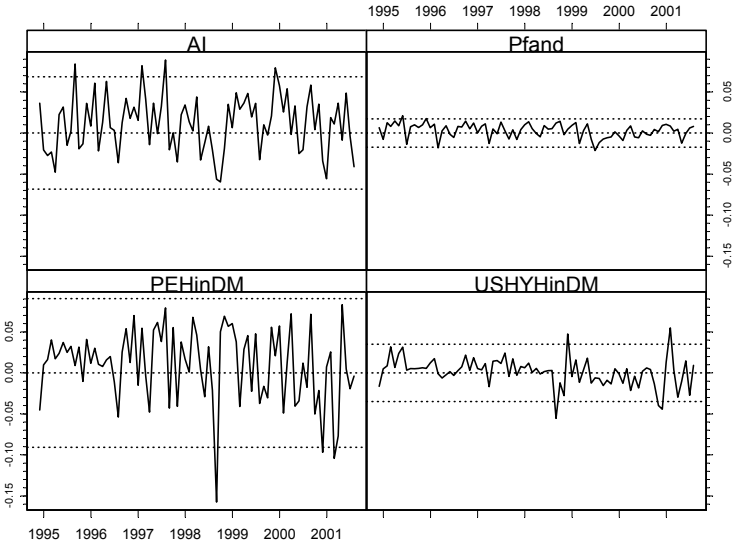

Figure 6.17 Monthly Time Series of Asset Returns

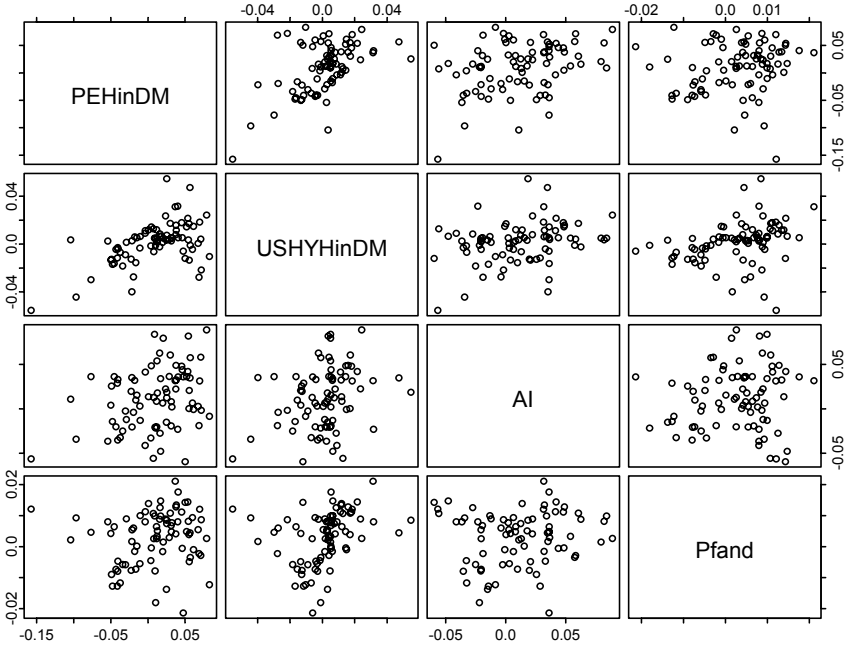

Figure 6.18 Pairwise Scatterplots of the Asset Returns

Figure 6.19 shows visual and tabular pairwise displays of classical and robust correlations for these returns. The robust correlations are obtained from a robust covariance matrix in a manner analogous to the way a classical correlation matrix is obtained from a classical covariance matrix: the elements of the robust covariance matrix are divided by the appropriate products of robust standard deviation (robust scale) estimates. For two of the pairs of returns, there are substantial differences between the classical and robust correlations: for Pfand and PEHinDM the classical correlation is .14 and the robust correlation is .49, and for Pfand and USHYHinDM, the classical correlation is .30 and the robust correlation is .66. These differences are consistent with the fact that the bulk of the data in the corresponding scatterplots clearly have a substantial positive correlation, while the outliers in these plots tend to make the data look more circular and hence less correlated.

Assuming you have already computed `data.ts` and `data` as in Code 6.7, Code 6.8 will produce Figure 6.19.

```
cov.fm <- fit.models(list(ROBUST = covRob(data),
  CLASSICAL = cov(data)))
plot(cov.fm,which.plots = 3)
```

Code 6.8 Robust Covariance Matrix and Correlation Display

The function `covRob`, appearing in Code 6.8, allows you to use any of several types of robust covariance matrices, with the default being the "Fast" Minimum Covariance Determinant (MCD) estimate of Rousseeuw and Van Driessen (1999). The MCD estimate computes the covariance matrix of the fraction `quan` of the data that yield the minimum covariance determinant, with the default `quan = .75`. The MCD estimate also returns a robust estimate of the multivariate mean (the mean returns in this application) consisting of the sample mean of the fraction `quan` of observations that yield the minimum covariance determinant. The reader is encouraged to experiment with different values of `quan` for the MCD estimate, and with the other robust covariance matrix estimates provided through `covRob` (see the online Robust Library User Guide (Insightful Corp., 2002) and help files for further details).

6.6.1 Uses of Robust Covariances and Correlations

There are at least three ways robust covariances and correlations can be used in portfolio construction:

1. As an exploratory data analysis (EDA) tool in order to discover whether the classical correlation and covariance estimates are influenced by outliers. In the case where the classical and robust methods agree, there is little need for concern, but when there are

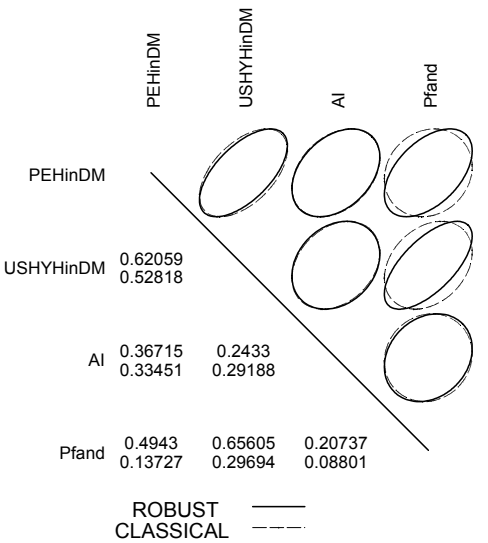

Figure 6.19 Classical and Robust Correlations for Asset Returns

substantial differences, one is well-advised to look more carefully at the data for possible explanations that will lead to better investment decisions. In some cases, influential outliers may be due to data errors, and in such cases the robust estimate will be more reliable than the classical estimate. In other cases, the influential outliers are valid data points, and the portfolio manager needs to decide whether they are representative of the future behavior of the returns, or are unique events that are unlikely to occur during the investment horizon under consideration and as such should be disregarded.

2. To construct robust multivariate distances for detecting unusual movements in multivariate returns (e.g., detecting normal times versus hectic times).
3. To obtain robust versions of Markowitz mean-variance optimal portfolios. By comparing the robust result with a classical mean-variance optimal portfolio, we will be alerted to the possibility that one or more outliers influence a particular optimal portfolio or Sharpe ratio.

The last two applications are described in the next two sections.

6.7 Robust Distances for Determining Normal Times versus Hectic Times

One way of defining "hectic" or "unusual" times, proposed by Scherer (2004), is based on the following statistical measure of the (squared) distance of a return vector $\mathbf{r}_t = (r_{t1}, r_{t2}, \cdots, r_{tk})$ from the vector of sample means $\hat{\boldsymbol{\mu}}$ over the history of interest:

$$d_t^2 = (\mathbf{r}_t - \hat{\boldsymbol{\mu}})' \hat{\boldsymbol{\Omega}}^{-1} (\mathbf{r}_t - \hat{\boldsymbol{\mu}}). \tag{6.11}$$

See also Chow, Jacquier, Kritzman, and Lowry (1999). Here $\hat{\boldsymbol{\Omega}}$ is the classical sample covariance matrix and $\hat{\boldsymbol{\Omega}}^{-1}$ is its inverse, and we assume a history of length n. In the statistical literature, this distance is called the **Mahalanobis** distance. When the returns have a multivariate normal distribution and n is not too small, the distances above have a distribution that is well-approximated by a chi-squared distribution with k degrees of freedom (dof). By definition, "unusual" times are those that do not happen very often and so represent a smallish fraction of the returns history during which the data have considerably different behavior than during the remaining majority of "normal" times. Thus it is reasonable to define **unusual times** as those for which the values of d_t are larger than the square root of an upper-tail percentage point of this chi-squared distribution (e.g., the square root of the upper 1%, 2.5%, or 5% point).

Scherer (2004) provides a convincing example of this approach to detecting unusual times when the classical sample mean and sample covariance estimates are used in the squared distance above. In particular, the example shows that it is possible to separate unusual times from normal times. In general, however, the use of the classical sample mean and covariance matrix may not yield a highly reliable method of detecting unusual times: since outliers can distort the sample mean and covariance estimates, the resulting squared distance may not be very reliable. Robust mean and covariance matrix estimates do not suffer from this drawback and therefore are ideal alternatives to the classical sample means and covariances for detecting unusual times. Thus we compute robust Mahalanobis distances by replacing the classical mean and covariance matrix estimates in the Mahalanobis distance with robust estimates. We illustrate this approach using the data shown in Figure 6.20.

Figure 6.20 shows the classical and robust (Mahalanobis) distances for the time series of multivariate returns (i.e., the values of d_t). The horizontal dashed line in the figure is the upper 2.5% point of a chi-squared distribution with four degrees of freedom. This figure reveals that the classical distance only detects three unusual times (two of these are just barely detected), while the robust distance clearly detects thirteen unusual times in two distinct temporal clusters, a

cluster of seven at the end of the series and a cluster of four near the middle of the series. There are also two other unusual times, month 56 and month 65.

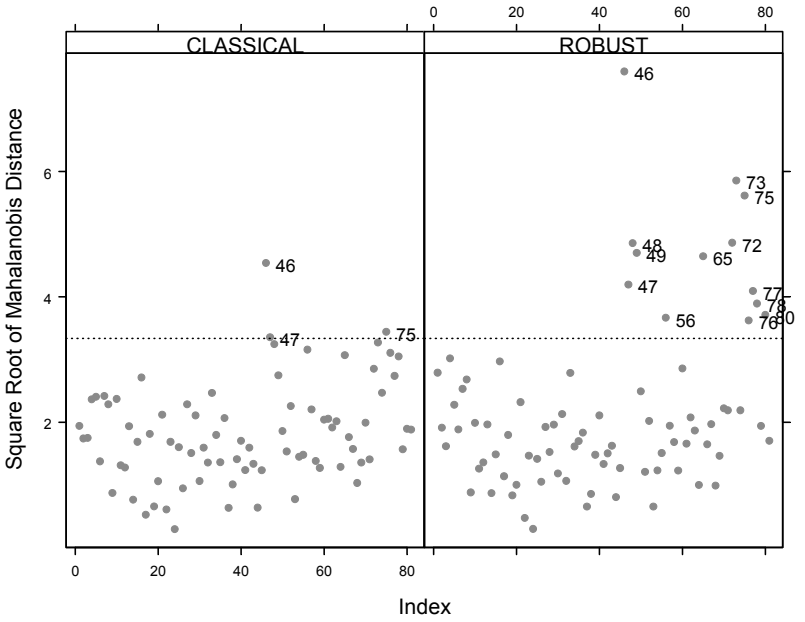

Figure 6.20 Robust and Classical Distances

Given the previous computation of `cov.fm` in Section 6.6, you can make the plot in Figure 6.20 with the command

```
plot(cov.fm,which.plots = 2,id.n = 14)
```

Now that we know the unusual times, we can repeat the application of classical and robust correlations and distances to the unusual portion of the data. The results are shown in Figure 6.21 and Figure 6.22 for the two clusters of unusual times.

The most striking bit of information revealed in Figure 6.21 is that the correlation between the German mortgage bond returns (Pfand) and returns on the other assets has switched from being positive during usual times to being substantially negative. This is natural during market downturns when there is a flight from equity investments. This behavior is also reflected in the pairwise scatterplots of Figure 6.23, which clearly reveal one or two outliers in the scatterplot of Al versus Pfand. The classical distances in Figure 6.22 completely miss the existence of these outliers, while the robust distances reveal two clear outliers and one marginal outlier.

Code 6.9 gives the S-PLUS code for producing the analysis of Figure 6.21, Figure 6.22, and Figure 6.23.

```
data <- data[c(46:49,72:80),]
cov.fm <- fit.models(list(ROBUST = covRob(data),
  CLASSICAL = cov(data)))
plot(cov.fm,which.plots = c(2,3),id.n = 14)
pairs(data)
```

Code 6.9 Analysis of Unusual Times Data

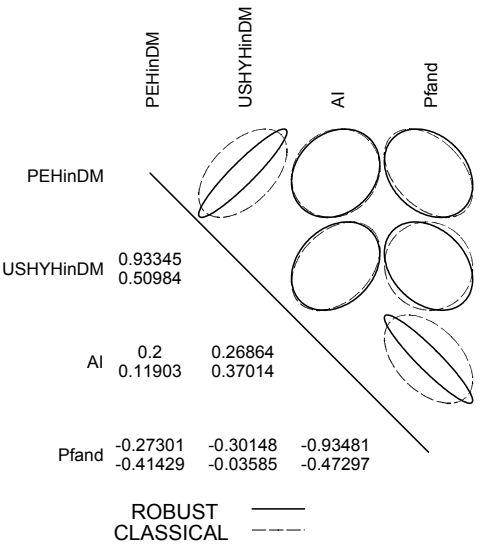

Figure 6.21 Classical and Robust Correlations for Two Clusters of Unusual Times

It is worth explaining why the distance measure (classical and robust) introduced above is the appropriate "statistical" distance. If you imagine an elliptical multivariate distribution for your returns (e.g., a multivariate normal or multivariate *t* distribution), then the right statistical distance is one that is the same for any data point lying along the same elliptical contour. This is what the Mahalanobis distance provides. A useful geometrical way to see what is going on with this distance is to consider the following re-expression of the (squared) distance. We assume that the true covariance matrix and mean return vector are $\mathbf{\Omega}$ and $\boldsymbol{\mu}$, and without loss of generality (by a shift of origin) assume that $\boldsymbol{\mu} = \mathbf{0}$. Then

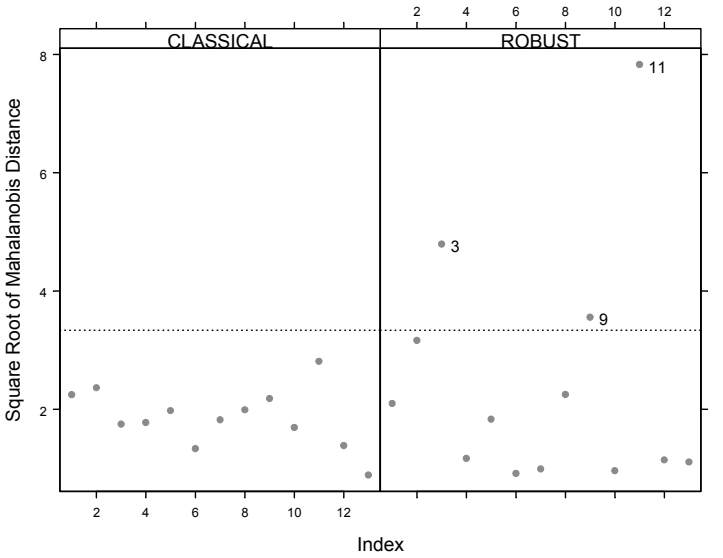

Figure 6.22 Robust and Classical Distances for Two Clusters of Unusual Times

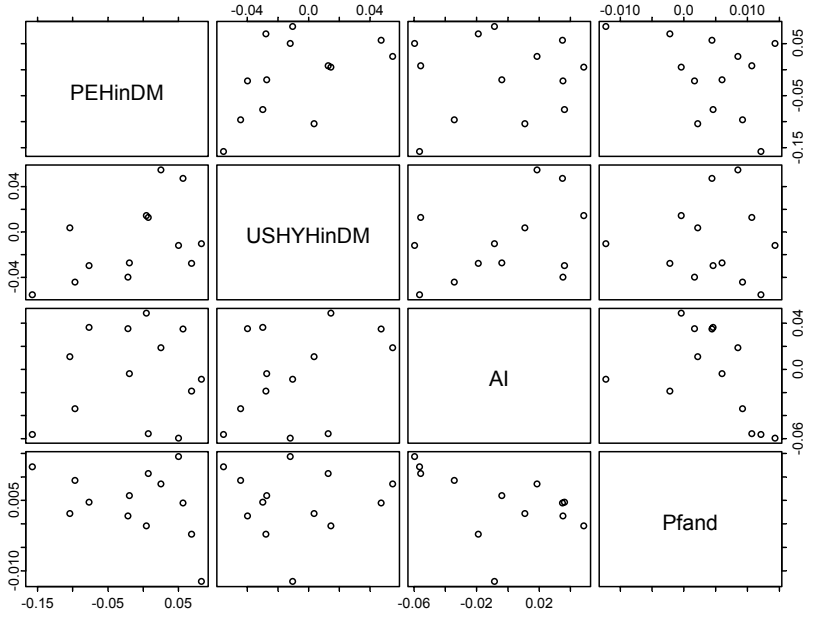

Figure 6.23 Pairwise Scatterplots for the Two Clusters of Unusual Times

$$\begin{aligned} d_t^2 &= \mathbf{r}_t' \boldsymbol{\Omega}^{-1} \mathbf{r}_t \\ &= \mathbf{r}_t' \boldsymbol{\Omega}^{-1/2} \cdot \boldsymbol{\Omega}^{-1/2} \mathbf{r}_t \\ &= \mathbf{z}_t' \cdot \mathbf{z}_t. \end{aligned} \tag{6.12}$$

You easily check that $\mathbf{z}_t'$ has the identity as its covariance matrix. Thus, using d_t is equivalent to making a transformation of the data so that the distribution is spherical rather than elliptical and then using the ordinary Euclidean distance in this new coordinate system.

The reason that a classical covariance matrix often fails to provide robust distances is that outliers often distort the estimated covariance matrix to such an extent that the transformation above does not result in a spherical scatter for the bulk of the data. Consequently, outliers are not reliably detected using the classical covariance matrix.

We emphasize the point that unusual times, consisting of locally extreme movements of one or more of the returns in a collection of returns, are frequently occurring behaviors by providing a second example, using monthly returns of four hedge fund indices: EMERGING MARKETS, EUROPE, EVENT DRIVEN, and EQUITY. The classical and robust correlations shown in Figure 6.24 clearly indicate that some outlying returns are influencing the classical covariance and correlation estimates.

Figure 6.25 shows that the classical distances give only a weak indication that there is something unusual going on at two or three time points, while the robust distances give a very strong indication of unusual movement at three to five time points in two clusters (time points 4 and 6 and time points 11, 12, 14, and 16).

A quick look at all pairwise scatterplots in Figure 6.26 reveals several multivariate outliers. The time series plots in Figure 6.27 reveal that the first cluster with unusual movement is in the emerging market returns in Q2 and Q3 of 1999, and the second cluster is joint unusual movements in the emerging market returns at the beginning of Q1 2000 and in the returns for Europe at the end of Q4 1999 and in the first and third months of Q1 2000.

Code 6.10 gives S-PLUS code for producing Figure 6.24 through Figure 6.27.

```
returns <- seriesData(hfunds.ts)
cov.fm <- fit.models(list(ROBUST = covRob(returns),
  CLASSICAL = cov(returns)))
plot(cov.fm,which.plots = c(2,3),id.n = 14)
pairs(returns)
par(mfrow = c(4,1))
y.name = colIds(hfunds.ts)
seriesPlot(hfunds.ts,one.plot=F,
  strip.text=y.name,col = 1)
```

Code 6.10 Robust Analysis for Hedge Fund Indices

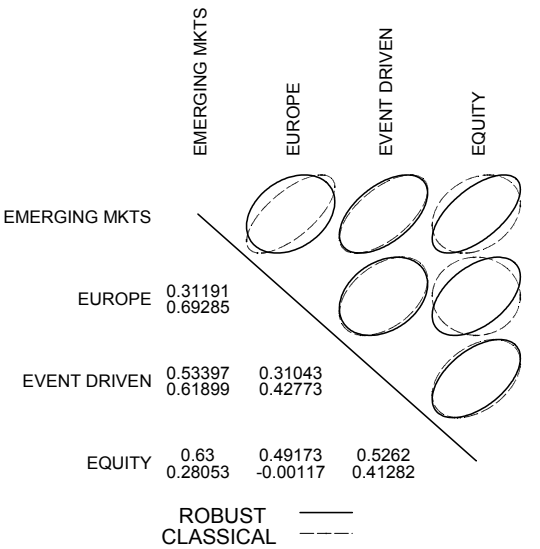

Figure 6.24 Classical and Robust Correlations for Four Hedge Fund Index Returns

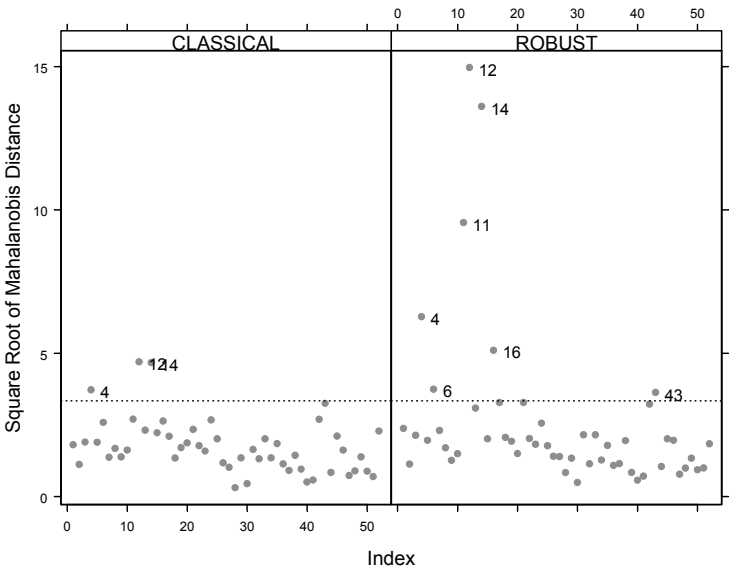

Figure 6.25 Robust and Classical Distances for Hedge Fund Index Returns

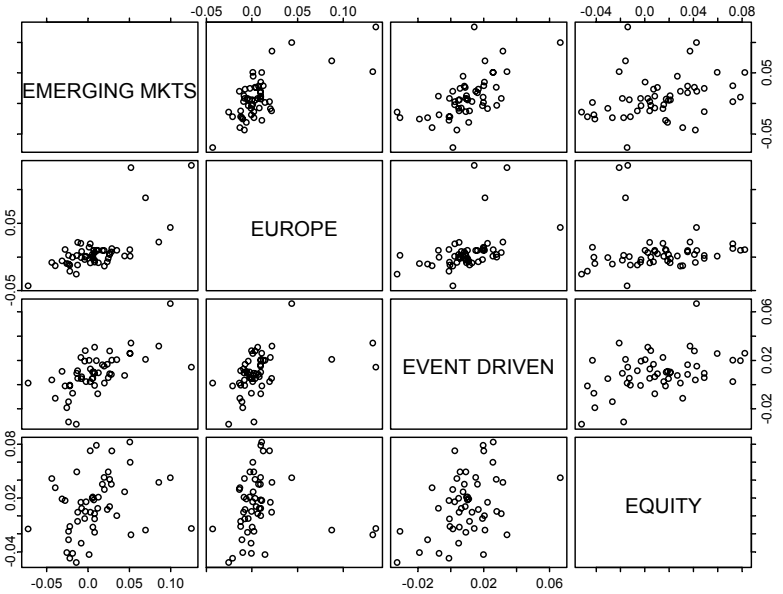

Figure 6.26 Pairwise Scatterplots for the Hedge Fund Index Returns

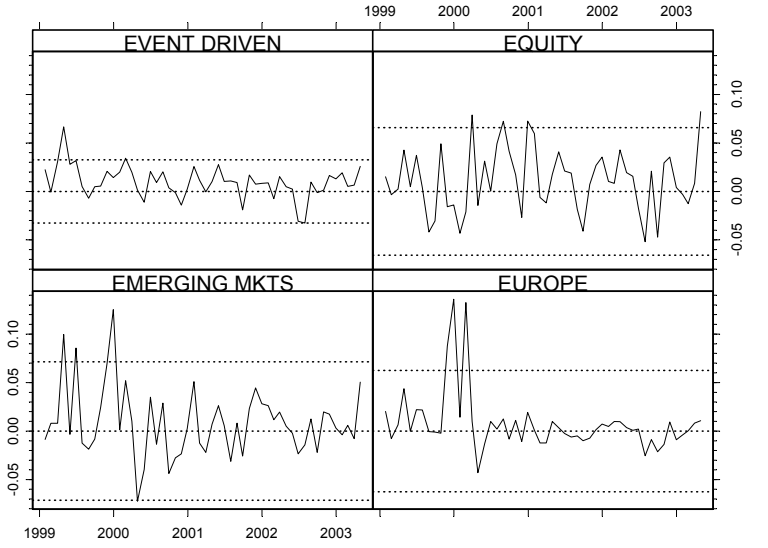

Figure 6.27 Time Series Plots of Hedge Fund Index Returns

6.8 Robust Covariances and Distances with Different Return Histories

It often happens that histories of returns for a collection of portfolio assets cover different periods of time. This situation is quite prevalent in "funds of funds" contexts where a manager is trying to select and optimize a portfolio of funds from a pool in which some funds have existed for only a few years while others have existed for ten years or more. In such a case one does not, at first blush, have an obvious way to compute classical or robust covariance matrices using all the data available. When confronted with this situation, most managers will opt to use the longest common history of the data by truncating all the data to the length of the shortest history available, a practice that often wastes useful information in the asset returns with longer histories. For example, Figure 6.28 shows five sets of hedge fund index returns, with the Emerging Markets (EM) index having the longest history (January 29, 1993 to March 31, 2003), and the High Yield (HY) and Health indices having common shortest histories (January 31, 1997 to March 31, 2003). All of the returns above exhibit a clear negative outlier in 1998, when markets took a dive following the Russian credit default. In addition, Health (a Health and Biotech index) exhibited a wild positive swing in 1999 prior to the dot-com collapse, as well as a wild negative swing following the dot-com collapse in spring of 2000, and Events exhibits a large positive outlier in 1995. A good detection method should reveal these unusual movements, along with others that may not be so apparent, at any time in the entire history of the series (from the earliest date of the longest series to the end of the series). It would be wasteful to throw away the Equity, EM, and Events returns prior to January 31, 1997, in order to compute robust covariance matrices and robust distances. To detect unusual times (or simply unusual data) at every instance over the entire time span of the data, we need a method to compute a robust covariance matrix of appropriate dimension and the associated robust distances.

Effective use of all the data is a classical missing data problem for which there exists a solution in the context of maximum likelihood estimation under multivariate normal returns (Stambaugh, 1997). Here we briefly explain the method in detail for the special case of two groups of assets, where each asset within a group has the same history, and indicate how the method is generalized to more than two groups.

Let the first group have k_1 assets and let the second group have k_2 assets, where the first group has the longer history, $t = 1, 2, \cdots, T$, and the second group has the shorter history, $t = s, s+1, \cdots, T$, with $s > 1$. Let $\hat{\boldsymbol{\mu}}_{long}^{ML}$ and $\hat{\boldsymbol{\Omega}}_{long,long}^{ML}$ be the Gaussian maximum likelihood estimators of the mean vector and covariance matrix of the first group with the long history (i.e., the usual sample mean vector

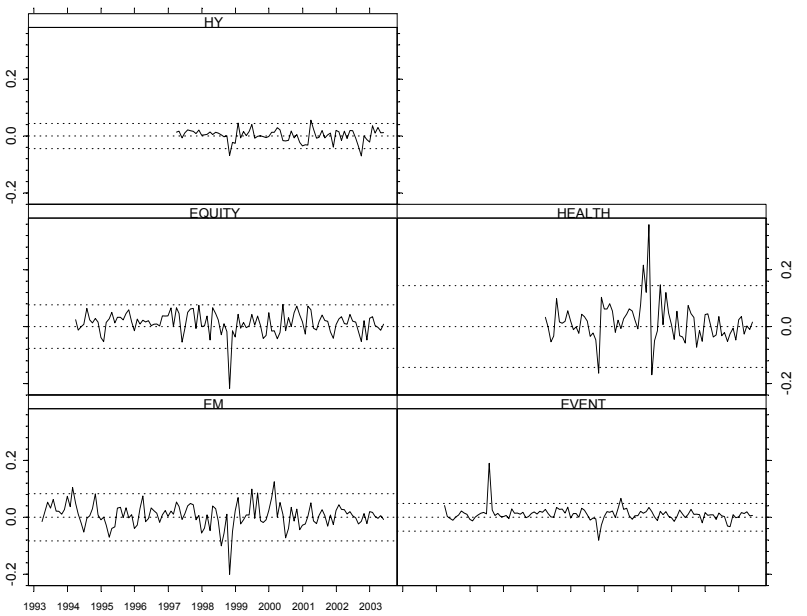

Figure 6.28 Hedge Fund Index Returns with Different Starting Dates

and sample covariance matrix with divisor T). Let $\hat{\boldsymbol{\mu}}_{long}^{truncated}$ be the sample mean vector of the longer group after truncating the returns to make their history have the same period as the shorter group, and let $\hat{\boldsymbol{\mu}}_{short}$ be the sample mean vector of the shorter group. It is important to note that this is not the maximum likelihood estimator of the shorter group's mean vector: the longer series is generally correlated with the shorter series and therefore contains information about the mean vector of the shorter series. Let $\mathbf{r}_{long,t}, \quad t = 1,2,\cdots,T,$ be the k_1-dimensional column vectors of returns of the first group, and let $\mathbf{r}_{short,t}, \quad t = s, s+1,\cdots,T$, be the k_2-dimensional column vectors of the second group. Consider the multivariate linear regression model

$$\mathbf{r}_{short,t} = \boldsymbol{\alpha} + \mathbf{B} \cdot \mathbf{r}_{long,t} + \boldsymbol{\varepsilon}_t, \quad t = s, s+1,\cdots,T \tag{6.13}$$

of the shorter set of returns on the longer set of returns over the shorter history. Let $\hat{\boldsymbol{\alpha}}$ and $\hat{\mathbf{B}}$ be the Gaussian maximum likelihood (least squares) estimates of the regression coefficients, and let $\hat{\boldsymbol{\Omega}}_{\varepsilon}$ be the sample covariance matrix of the residuals $\hat{\boldsymbol{\varepsilon}}_t, \ t = s, s+1,\cdots,T,$ from the maximum likelihood fit.

We can now summarize the overall maximum likelihood estimation results. The maximum likelihood estimate (MLE) of the mean vector of the shorter group series is

$$\hat{\boldsymbol{\mu}}_{short}^{ML} = \hat{\boldsymbol{\mu}}_{short} + \hat{\mathbf{B}} \cdot \left(\hat{\boldsymbol{\mu}}_{long}^{ML} - \hat{\boldsymbol{\mu}}_{long}^{truncated} \right). \tag{6.14}$$

The overall mean vector MLE is

$$\hat{\boldsymbol{\mu}}^{ML} = \left(\hat{\boldsymbol{\mu}}_{long}^{ML},\ \hat{\boldsymbol{\mu}}_{short}^{ML} \right). \tag{6.15}$$

The maximum likelihood estimate of the overall covariance matrix is

$$\hat{\boldsymbol{\Omega}}^{ML} = \begin{pmatrix} \hat{\boldsymbol{\Omega}}_{long,long}^{ML} & \hat{\boldsymbol{\Omega}}_{long,short}^{ML} \\ \hat{\boldsymbol{\Omega}}_{short,long}^{ML} & \hat{\boldsymbol{\Omega}}_{short,short}^{ML} \end{pmatrix}, \tag{6.16}$$

where

$$\hat{\boldsymbol{\Omega}}_{short,short}^{ML} = \hat{\boldsymbol{\Omega}}_{\varepsilon} + \hat{\mathbf{B}} \cdot \hat{\boldsymbol{\Omega}}_{short,short}^{ML} \cdot \hat{\mathbf{B}}' \tag{6.17}$$

and

$$\hat{\boldsymbol{\Omega}}_{short,long}^{ML} = \hat{\mathbf{B}} \cdot \hat{\boldsymbol{\Omega}}_{long,long}^{ML}. \tag{6.18}$$

In applications where there are more than two groups and more than two sets of common histories with different starting dates, the method above can be applied recursively to compute an overall Gaussian maximum likelihood estimate of the mean vector and covariance matrix. Details may be found in Section 4 of Stambaugh (1997).

6.8.1 Robustifying the Stambaugh Method

The Stambaugh method relies heavily on a multivariate Gaussian assumption for the returns. As we have seen, this is not a very safe assumption when dealing with asset returns. Furthermore, we need a robust version of the method that is not much influenced by a few outliers. Fortunately, it is rather straightforward to create a robust version by making the following three modifications: (a) replace sample mean estimates by robust location estimates, (b) replace the least squares multivariate regression estimates $\hat{\boldsymbol{\alpha}}$ and $\hat{\mathbf{B}}$ with robust regression estimates, and (c) replace each of the Gaussian MLE sample covariance matrix estimates above (including $\hat{\boldsymbol{\Omega}}_{\varepsilon}$) with a robust covariance matrix estimate. In the example below, we use (a) `location.m` for the robust location estimates, (b) `lmRob` to obtain the robust multivariate regression by computing a set of robust univariate

regressions, and (c) `covRob`, with the default fast MCD method and setting `quan = .9`. We note that the method is such that when all the component covariance matrix estimates below are positive definite, the overall robust covariance matrix

$$\hat{\mathbf{\Omega}}^{ROB} = \begin{pmatrix} \hat{\mathbf{\Omega}}^{ROB}_{long,long} & \hat{\mathbf{\Omega}}^{ROB}_{long,short} \\ \hat{\mathbf{\Omega}}^{ROB}_{short,long} & \hat{\mathbf{\Omega}}^{ROB}_{short,short} \end{pmatrix} \tag{6.19}$$

will be positive definite.

6.8.2 Robust Distances and Degrees of Freedom at Different Time Points

In order to compute a robust distance at each point in time, one needs to use the appropriate robust covariance matrix. Suppose that as you move through the history of a set of returns with different starting dates you encounter returns of dimensions $k_1, k_2, \cdots, k_M$. Then at a time point where there exist k_i returns, you use the corresponding $k_i \times k_i$ robust covariance matrix. Then the corresponding degrees of freedom for the chi-squared upper 2.5% cutoff point is k_i.

6.8.3 The Hedge Fund Indices Example

We used the robustified Stambaugh method[14] to obtain robust covariance matrices $\hat{\mathbf{\Omega}}^{ROB}_i$, $i = 1, 2, 3,$ and associated robust distances for the hedge fund indices whose time series were displayed at the beginning of this section (Figure 6.28). The results are shown in Figure 6.29. Note the increasing staircase behavior of the chi-squared upper 2.5% thresholds for each of the three groups with common starting dates owing to the increase in chi-squared degrees of freedom as more assets came online in 1994 and 1997. The robust distances detect outliers with greater power than the classical distances, thereby clearly revealing multivariate outliers that the classical method detects only weakly or not at all.

The robust Stambaugh method code is long, and is not included with this book.

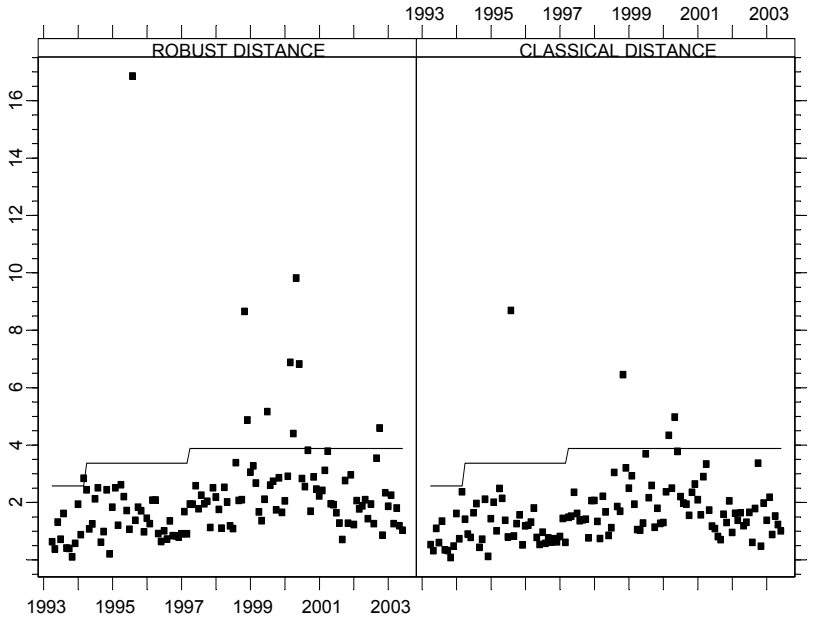

Figure 6.29 Classical and Robust Distances for Hedge Fund Indices

6.8.4 Comment on Using Chi-Square Percentage Points

One should be aware of several points concerning use of the (square root of the) 2.5% upper percentage point of the chi-squared distribution as a detection threshold. First, the threshold is somewhat arbitrary, and one could equally well use an upper 5% point or upper 1% point, the former yielding a larger false alarm rate and the latter yielding a smaller false alarm rate than the 2.5% point. We advise against using anything smaller than the upper 5% point since false alarm rates that are too high can lead to detection of outliers and unusual times even when the data are perfectly stationary and normally distributed (e.g., the upper 25% point recommended by Chow, Jacquier, Kritzman, and Lowry (1999) would exhibit such behavior). As the chi-squared approximation is not very reliable under non-normality (see, for example, Rocke and Woodruff, 1996), one may prefer to make a kernel density estimate of the classical and robust distances and look for clustering of unusual times in terms of multimodality of the density estimates.

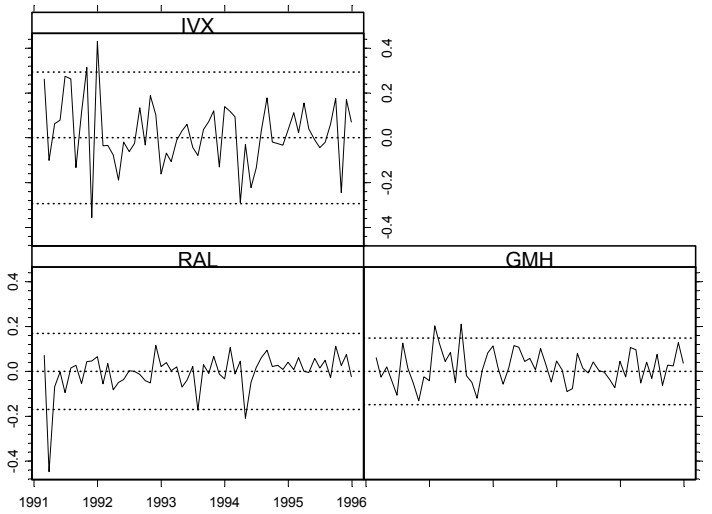

Figure 6.30 Time Series of RAL, GMH, and IVX Returns

6.9 Robust Portfolio Optimization

Since classical estimates of mean returns and covariances can be adversely influenced by the presence of one or more outliers, it should not be surprising to find that Markowitz mean-variance optimal portfolios based on these classical estimates can also be adversely influenced by such outliers. As an example of the extent to which outliers can influence the estimated Markowitz efficient frontier, consider the time series of highly volatile monthly stock returns for RAL, GMH, and IVX from February 28, 1991 to December 29, 1995, shown in Figure 6.30.

RAL is distinguished by having a single negative outlier at the beginning of the series, while GHM and IVX have a relatively high volatility in the early time periods when compared with the rest of the series. The values of robust and classical sample means and standard deviations for these returns are shown in Table 6.1.

It is evident that the outliers have the largest impact on RAL, where the sample mean and robust mean values are .003 and .009, respectively, and the classical and robust standard deviations are .085 and .055. The differences in correlations between the two estimates are shown in Figure 6.31.

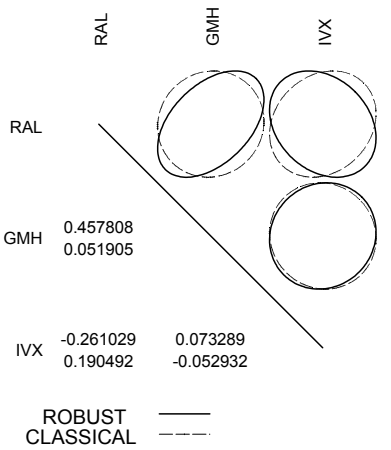

Figure 6.31 Classical and Robust Correlations for RAL, GHM, and IVX

Table 6.1 Means and Standard Deviations for RAL, GHM, and IVX

	RAL	GMH	IVX
Classic Mean	–.003	.019	.021
Robust Mean	.009	.018	.020
Classic Std. Dev.	.085	.074	.147
Robust Std. Dev.	.055	.082	.131

We see a substantial shift of +.41 for the RAL/GHM correlation and –.45 for the RAL/IVX correlation when substituting a robust correlation for a classical correlation. The S-PLUS code for the plots above is similar to Code 6.7 and Code 6.8 provided in Section 6.6.

We made the plots in Figure 6.30 and Figure 6.31 with Code 6.7 and Code 6.8 using `returns.three.ts` in place of `normal.vs.hect.ts` and deleting the `pairs` command in Code 6.7.

Now we use NUOPT to compute a robust efficient frontier with a constraint of no short-selling by simply replacing the classical sample mean returns and sample covariance estimates with robust estimates. The resulting efficient frontier is displayed in Figure 6.32 along with the classical efficient frontier and the maximum Sharpe ratios based on a monthly risk-free rate of .003. The display also shows the classical and robust means and standard deviations of each of the three stocks along with their ticker symbols.

The (maximum) Sharpe ratios are approximately the same for both frontiers. However, the classical efficient frontier indicates that the investor can achieve about 10 to 20 basis points (monthly) more than with the robust frontiers for sufficiently high levels of risk. On the other hand, the robust efficient frontier

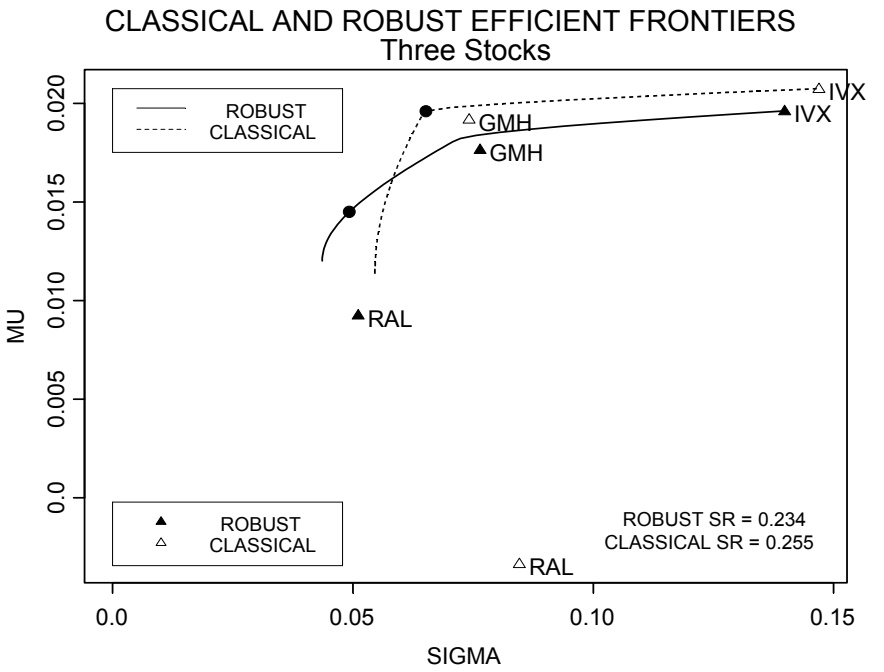

Figure 6.32 Classical and Robust Efficient Frontiers for Three-Stock Portfolio

offers higher levels of return than the classical frontier for lower levels of risk and a considerably lower risk for the (global) minimum variance portfolio. The means and standard deviations of the individual stocks in the plot above are the classical sample mean and sample standard deviation. Note that the main difference between the values of the classical and robust means and standard deviations are for RAL. The optimal weights for the classical and robust efficient frontiers are displayed in Figure 6.33.

It should not be surprising to see that, for small levels of risk, the classical portfolio gives considerably less weight to RAL than does the robust portfolio (recall that RAL has one large negative outlier at the beginning of the series) and that for the largest levels of risk both portfolios give about the same relative weights to GMH and IVX. We also see that the robust portfolio gives no weight to GMH in the minimum variance portfolio and also gives considerably less weight to GMH than does the classic portfolio for lower levels of risk.

Code 6.11 creates the function `rob.mv.efronts` for computing and displaying mean-variance and robust efficient frontiers and optionally plotting the portfolio weights for each. The last line of Code 6.11 executes the function on the three-asset time series object `returns.three.ts`:

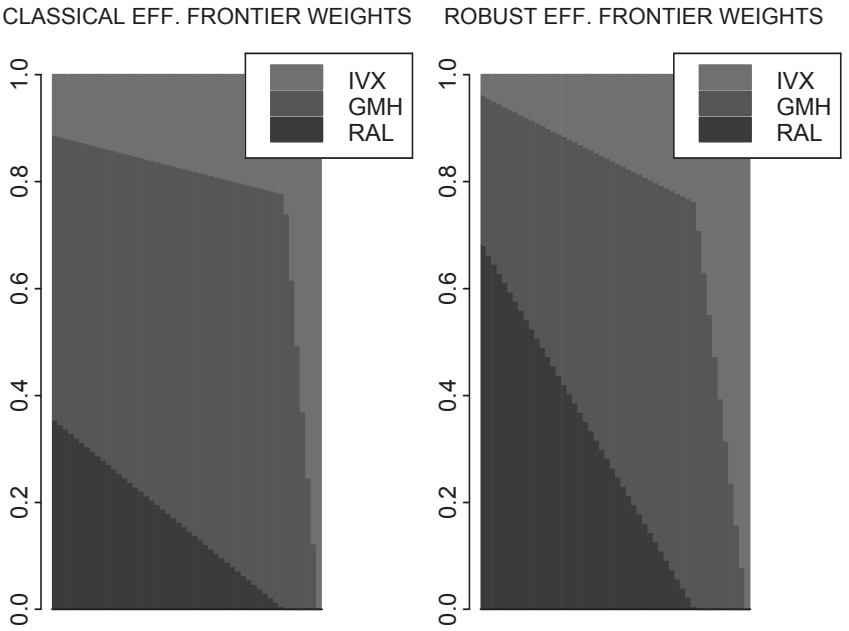

Figure 6.33 Weights for Classical and Robust Efficient Frontier Portfolios

```
rob.mv.efronts <- function(returns.ts, rf=.003,
  n.ret=50, plot.weights=F, a=1, sharpe=T,
  display.points=T, display.names=T,
  display.letters=F)
{
  returns <- seriesData(returns.ts)
  p <- ncol(returns)
  # Parameter comments
  # Use a= 1 for no short selling, and
  # adjust a > 1 for short selling
  # If using display.letters=T, set
  # display.points=F and display.names=F)
  # Compute Classical Efficient Frontier
  meanVec1 <- apply(returns,2,mean)
  covMat1 <- var(returns)
  sigma1 <- diag(covMat1)^.5
  max.ret1 <- max(meanVec1)*a
  ef.classic <- portfolioFrontier(covMat1,
     meanVec1,
     wmin=0, max.ret=max.ret1, n.ret=n.ret)
  # Compute Robust Efficient Frontier
```

```
meanVec2 <- apply(returns,2,location.m)
covMat2 <- covRob(returns,estim = "mcd",
   quan = .9)$cov
sigma2 <- diag(covMat2)^.5
max.ret2 <- max(meanVec2)*a
ef.robust <- portfolioFrontier(covMat2, meanVec2,
   wmin=0, max.ret=max.ret2, n.ret=n.ret)
# Plot Efficient Frontiers
xlim <- range(ef.classic$sd,ef.robust$sd,
   sigma1,sigma2,0)
ylim <- range(ef.classic$ret,ef.robust$ret,
   meanVec1,meanVec2,0)
plot(ef.classic$sd,ef.classic$ret,xlim=xlim,
   ylim=ylim, type = "n" ,xlab="SIGMA",ylab="MU")
lines(ef.classic$sd,ef.classic$ret,lty = 8)
lines(ef.robust$sd,ef.robust$ret,lwd=2)
# Plot Stock Mu's and Sigma's and Add Legend
title(main="CLASSICAL AND ROBUST EFFICIENT
   FRONTIERS\n Three Stocks")
if(display.letters) {
   for(i in 1:p) {
      points(sigma1[i],meanVec1[i],
         pch = letters[i])
      points(sigma2[i],meanVec2[i],
         pch = LETTERS[i])
   }
   if(display.names){
      text(sigma1 + 0.002, meanVec1,
         names(meanVec1), adj=0)
      text(sigma2 + 0.002,meanVec2,
         names(meanVec2), adj=0)
   }
   x = xlim[1]+.15*(xlim[2]-xlim[1])
   y = ylim[1]+.10*(ylim[2]-ylim[1])
   leg.names = c("A,B,.. Robust Mu's,Sigma's",
      "a,b,. Classical Mu's,Sigma's")
   text(x,y,leg.names[1])
   text(x,y-.002,leg.names[2])
} # endif display.letters

if(display.points){
   points(sigma1,meanVec1,pch = 2)
   points(sigma2,meanVec2,pch = 17)
   if(display.names){
      text(sigma1 + 0.002, meanVec1,
```

```
            names(meanVec1), adj= 0)
         text(sigma2 + 0.002, meanVec2,
            names(meanVec2), adj= 0)}
         x = xlim[1]+.00*(xlim[2]-xlim[1])
         y = ylim[1]+.13*(ylim[2]-ylim[1])
         leg.names = c("   ROBUST","CLASSICAL")
         legend(x,y,leg.names, marks = c(17,2))
} #endif display.points

# Add legend
x = xlim[1]+.0*(xlim[2]-xlim[1])
y = ylim[2]-.0*(ylim[2]-ylim[1])
leg.names = c("   ROBUST","CLASSICAL")
legend(x,y,leg.names,lty=c(1,8))

# Compute and Display Maximum Sharpe Ratio's,
# and Add Bullets
if(sharpe) {
   i.maxsr.classic = order((ef.classic$ret-
      rf)/ef.classic$sd)[n.ret]
   i.maxsr.robust = order((ef.robust$ret-
      rf)/ef.robust$sd)[n.ret]
   sr.classic = ((ef.classic$ret-
      rf)/ef.classic$sd)[i.maxsr.classic]
   sr.robust = ((ef.robust$ret-
      rf)/ef.robust$sd)[i.maxsr.robust]
   points(ef.classic$sd[i.maxsr.classic],
      ef.classic$ret[i.maxsr.classic],pch = 16)
   points(ef.robust$sd[i.maxsr.robust],
      ef.robust$ret[i.maxsr.robust],pch = 16)
   x = xlim[1]+.85*(xlim[2]-xlim[1])
   y = ylim[1]+.1*(ylim[2]-ylim[1])
   text(x,y,paste("  ROBUST SR =",
      round(sr.robust,3)))
   y = ylim[1]+.05*(ylim[2]-ylim[1])
   text(x,y,paste("CLASSICAL SR =",
      round(sr.classic,3)))
} #endif sharpe

# Plot Portfolio Weights for Both Efficient
# Frontiers
if(plot.weights) {
   par(mfrow = c(1,2))
   barplot(ef.classic$weights,
      legend = names(returns))
```

```
      title(main = "CLASSICAL EFF. FRONTIER
         WEIGHTS")
      barplot(ef.robust$weights,
         legend = names(returns))
      title(main = "ROBUST EFF. FRONTIER WEIGHTS")
      par(mfrow = c(1,1))
   }#endif plot.weights
 } # end function definition

 rob.mv.efronts(returns.three.ts,plot.weights = T)
```

Code 6.11 Robust Efficient Frontiers

6.9.1 Effect of Outliers on the Sample Mean versus the Sample Covariance Matrix

By making small modifications to Code 6.11 above, you can easily do a sensitivity analysis to see whether the influence of the outliers on the classical efficient frontier is primarily through distortion of the mean estimate or primarily through distortion of the covariance matrix estimate. To use only a robust covariance estimate, replace `location.m` with `mean` in the expression

```
meanVec2 <- apply(returns,2,location.m)
```

in the Code 6.11 function. To use only a robust mean estimate (and the classical covariance estimate), change the code line

```
covMat2 <- covRob(returns)$cov
```

to

```
covMat2 <- var(returns).
```

The results of making these two changes separately are shown in Figure 6.34 and Figure 6.35, respectively.

These displays indicate that it is not enough to use only robust means or only robust covariances. Combining the information in Figure 6.33, Figure 6.34, and Figure 6.35, it appears that the difference between the classical and robust efficient frontiers is a result of outliers influencing both the sample mean and sample covariance estimates.

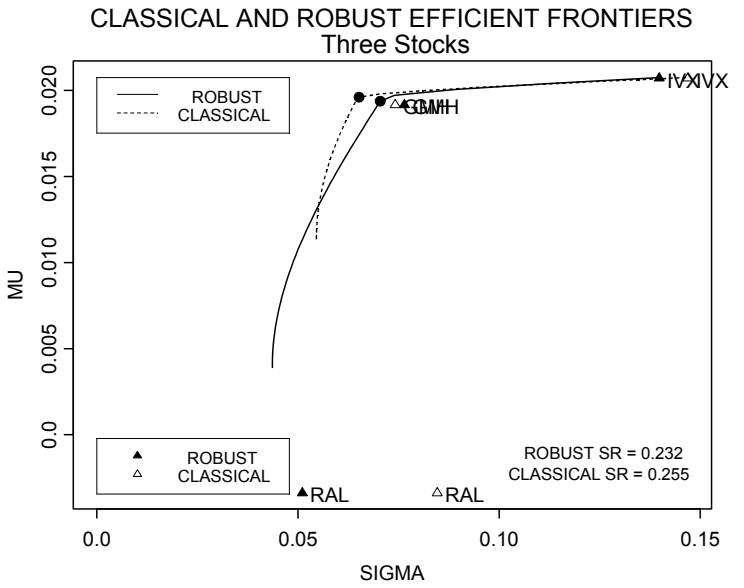

Figure 6.34 Robust Efficient Frontier with Robust Covariance Estimate Only

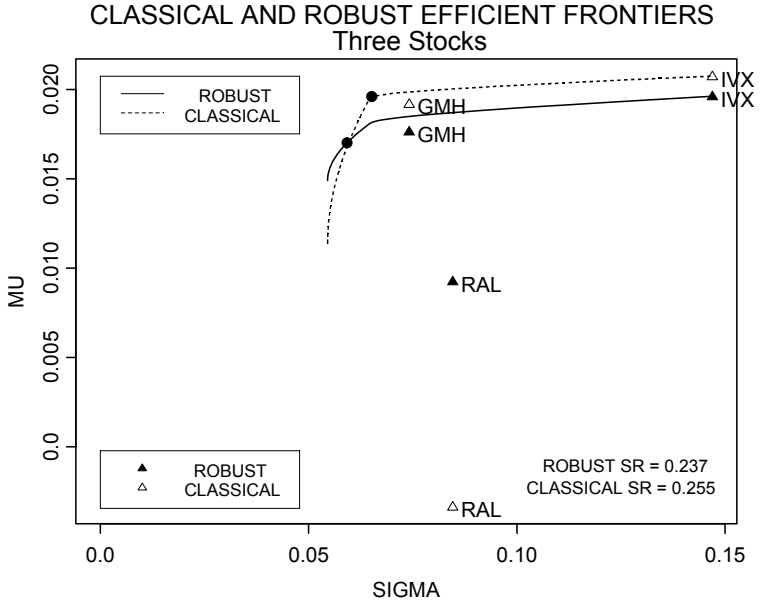

Figure 6.35 Robust Efficient Frontier with Robust Mean Only

6.9.2 Other Examples and Alternative Asset Plot Labels

You can easily find many examples where the classical and robust efficient frontiers differ by a considerable amount. This is particularly true for microcap and small cap stocks, but you can also find examples of this type for mid-cap and large cap stocks. We give three examples in Figure 6.36 through Figure 6.38 in support of this claim using options in Code 6.11 that provide for alternative displays of both classical and robust means and standard deviations.

In Figure 6.36, we display the two efficient frontiers for a group of five small cap stocks with a solid (open) triangle symbol for the robust (classical) means and standard deviations of the returns of the individual stocks. Although we can see that there are substantial differences in the robust and classical means and standard deviations, we cannot see their individual changes.

We make the plot of Figure 6.36 with the commands

```
tickers <- c("TOPP","KWD","HAR","RARE","IBC")
returns.ts <- smallcap.ts[,tickers]
rob.mv.efronts(returns.ts)
y.name <- colIds(returns.ts)
seriesPlot(returns.ts,one.plot=F,strip.text=y.name,
  col = 1)
```

Figure 6.37 and Figure 6.38 are made by modifying the code above in obvious ways.

We note that, in general, mid-cap and large cap stocks are less prone to having large outliers than small caps and microcaps, and when there are no influential outliers in returns, the values for classical and robust means, standard deviations, and covariances will be close to one another. In such situations, the classical and robust efficient frontiers will be very similar, as in Figure 6.38, and one need not worry about influential outliers.

In the case of many stocks, the ticker symbols may overlap a lot, and you may prefer to use uppercase and lowercase letters in order to visualize the changes in individual means and standard deviations, as in Figure 6.39. You can get these kinds of labels in your efficient frontier plot by using the optional arguments `display.points = F`, `display.names = F`, and `display.letters = T` in the function `rob.mv.efronts`.

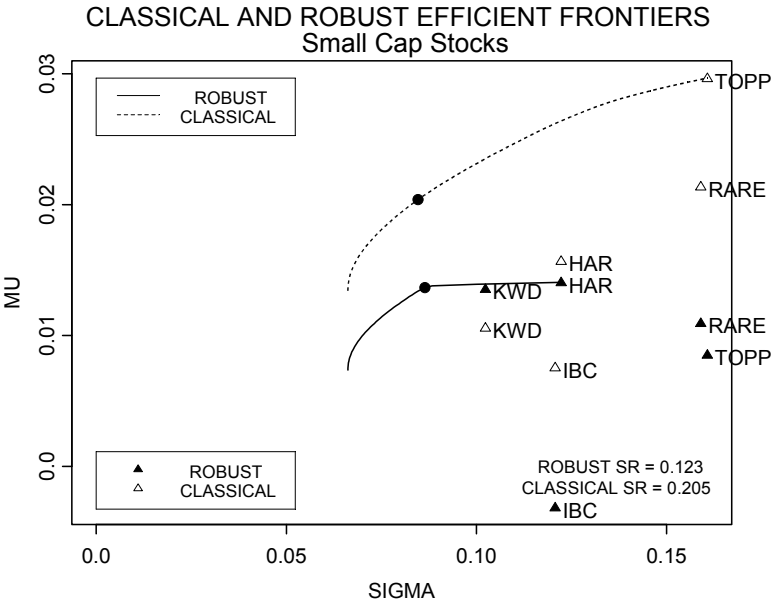

Figure 6.36 Efficient Frontiers for Small Cap Stocks with Classical and Robust Mu and Sigma

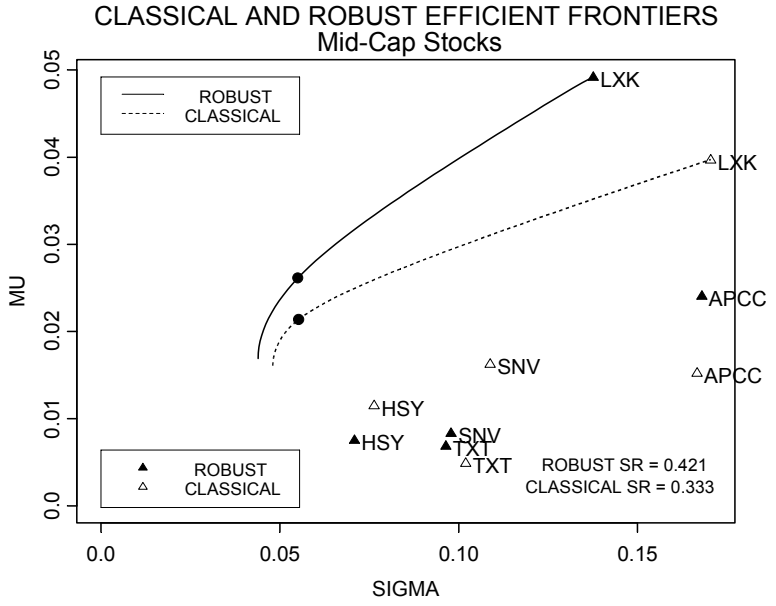

Figure 6.37 Efficient Frontiers for Mid-Cap Stocks with Ticker Symbols

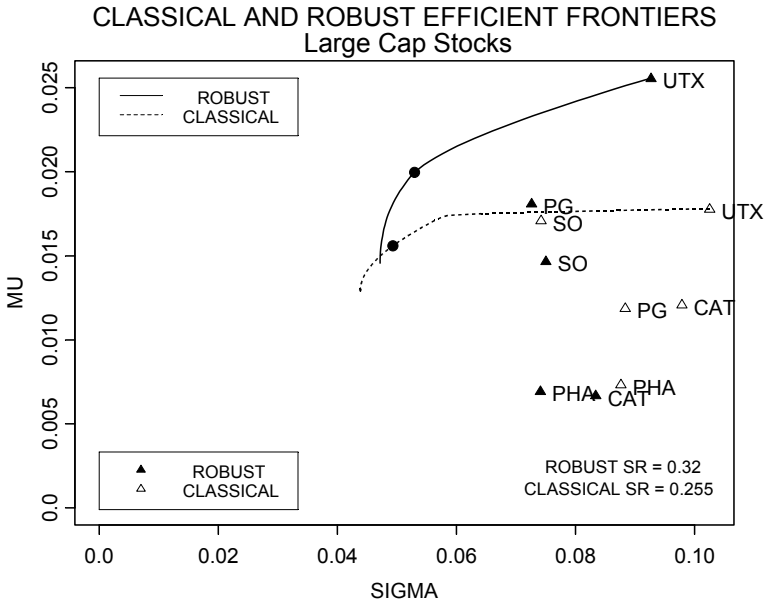

Figure 6.38 Efficient Frontiers for Large Cap Stocks with Letters for Mu and Sigma

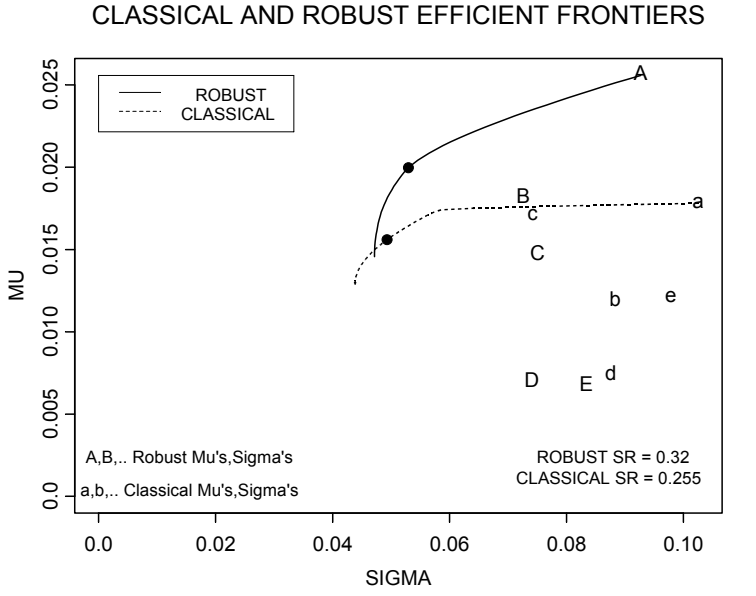

Figure 6.39 Typical Efficient Frontiers for Large Cap Stocks

6.9.3 Classical or Robust Efficient Frontier: Which to Use?

It is important to keep in mind that a robust efficient frontier is based on robust estimates of the means and covariance matrix, which themselves represent the mean and covariance of the bulk of the returns. As such, a robust efficient frontier represents the bulk of the data. Whether this is an adequate representation of the future behavior of your returns is open to serious question. Therefore, at this point it is not clear whether you should prefer making an investment decision based on a robust efficient frontier rather than a classical efficient frontier.

Of one thing we are sure: a robust efficient frontier is a valuable diagnostic tool. When the robust and classical frontiers are quite close to one another, as in Figure 6.39, the returns are quite likely to be free of influential outliers and well-approximated by a multivariate normal distribution. In this case, one can feel reasonably confident in using the mean-variance efficient frontier. When the two efficient frontiers differ by a significant amount, as in Figure 6.36 though Figure 6.38 above, there are likely to be influential outlying returns, and it is unlikely that the returns are well-approximated by a multivariate normal distribution. In such cases one is alerted to the need to carry out some exploratory data analysis (EDA) of the returns data and think carefully about what to do. One way to start such an EDA is by making time series plots of your returns to see if there are any obvious outliers, whether those outliers are positive or negative, and where they occur in the series of returns (e.g., early, middle or late in the period of interest). We illustrate what this initial step can reveal in the examples above.

The Trellis time series plots of the stocks in the small cap portfolio above, provided in Figure 6.40, reveal that three of the series, TOPP, RARE, and IBC, have one or more dominant positive returns outliers and that there are no dominant negative outliers in any of the series of returns. In the efficient frontiers display of Figure 6.36, you see that the robust means and standard deviations of the returns for TOPP, RARE, and IBC are substantially smaller than those of classical means and standard deviations, as might be anticipated. Correspondingly, the robust efficient frontier is lower and slightly to the left of the classical frontier. An investor who uses the robust efficient frontier is taking a conservative view with regard to the potential occurrence of future positive outliers in one or more of the series TOPP, RARE, and IBC (i.e., he is not betting on such occurrences in the future). Such an investor is, in a sense, implementing a Bayesian approach based on his own subjective prior distributions about the probability of future positive outliers.

The situation for the time series plots of the mid-cap stocks in Figure 6.41 is different. APCC has a number of substantially negative values but no clearly dominant outliers, while LXK has three or four dominant negative outliers. Furthermore, in each case there are no equivalent offsetting positive values. For

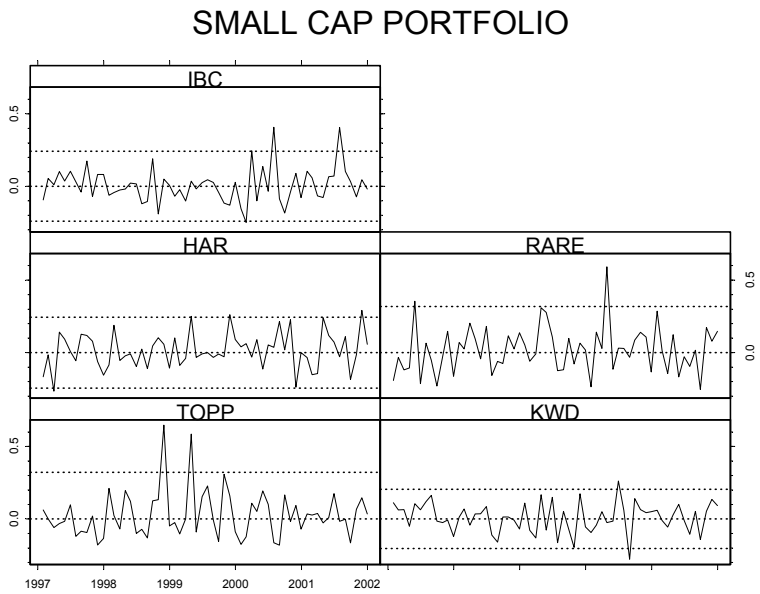

Figure 6.40 Small Cap Portfolio Time Series

TXT, the positive and negative extreme values appear to be roughly offsetting, and SNV has a single positive outlier near the beginning of the series. This information is depicted in the efficient frontiers of Figure 6.36: the robust means of the APCC and LXK returns are larger than their sample means; the robust standard deviation of LXK is smaller than its classical sample standard deviation, while the robust and classical standard deviations of APCC have almost the same values; the robust mean and standard deviation of SNV are smaller than the classical sample mean and standard deviation; and there is a small difference between the robust and classical means and standard deviations for TXT. Given the different behaviors of these means and standard deviations, one might not expect the robust efficient frontier to dominate the classical frontier at all levels of risk. That this is the case reflects the fact that the larger values of the robust means along with the large values of the robust standard deviations result in considerable "leverage" effects in determining the location of the robust efficient frontier relative to the classical efficient frontier.

A word of caution is in order: in this situation, would a wise investor trust the higher returns achievable with the robust efficient frontier? Since the negative returns for APCC and LXK are near the end of the series, an investor may well be wary of assuming that such returns will not occur again in the near future and therefore reject the optimism of the robust efficient frontier.

Figure 6.42 shows the Trellis time series plots of returns for the stocks of the large cap portfolio whose efficient frontiers are shown in Figure 6.38. The tickers UTX, PG, PHA, SO, and CAT correspond to the letters "A," "B," "C,"

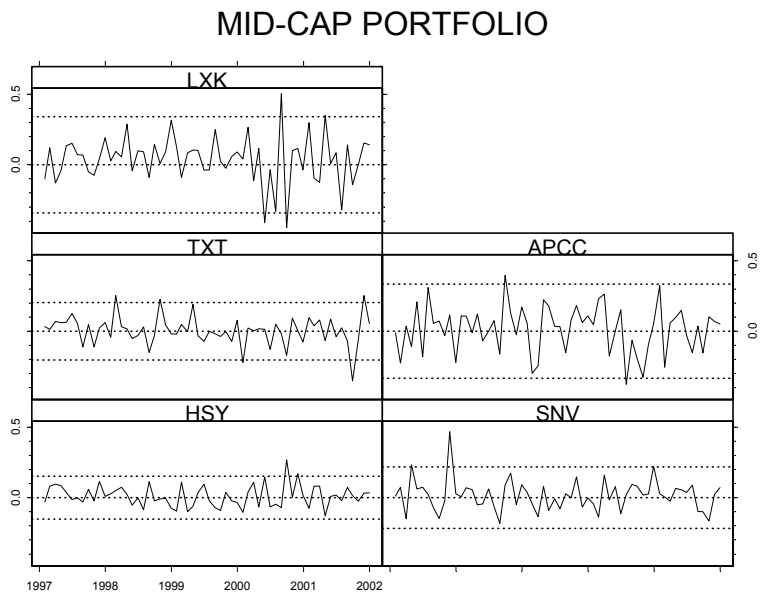

Figure 6.41 Mid-Cap Portfolio Time Series

"D," "E" (and "a," "b," "c," "d," "e"). The time series plots reveal that UTX has one or two possible negative outliers, PG has one large negative outlier, PHA has one positive outlier and one negative outlier, and CAT has one large positive outlier. The corresponding differences in locations of the robust means and standard deviations (the points labeled "A," "B," "C," and "E," respectively) and the classical means and standard deviations (the points labeled "a," "b," "c," and "e," respectively) are what one would expect. In this case the overall configuration of the outliers in the returns and the resulting robust versus classical means and standard deviations is rather complex, and one cannot easily guess the relative positioning of the robust and classical efficient frontiers.

While the analysis above may provide some guidance in choosing between a robust and classical efficient frontier when making an investment decision, better tools are needed to determine the relative performance of investments made with these two approaches. One such tool is a **bootstrapped efficient frontier** that can help determine whether the difference between a robust and classical efficient frontier is "real" or whether it is just a result of statistical variability.

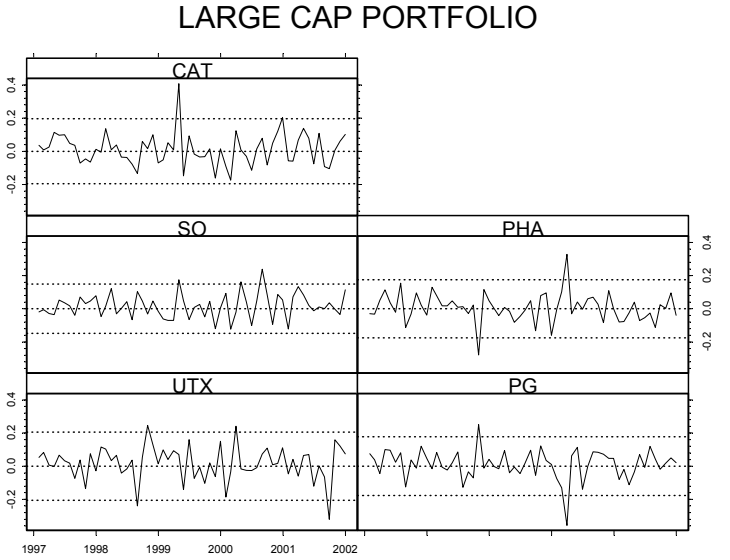

Figure 6.42 Large Cap Portfolio Time Series

6.9.4 Bootstrapped Efficient Frontiers and Sharpe Ratios

Classical and robust efficient frontiers are complicated functionals of the underlying distribution of the returns, and the exact distributions of efficient frontiers (even the mean-variance efficient frontier) are generally intractable. In Chapter 4, we discussed parametric portfolio resampling in which multivariate normal samples are generated based on the sample mean and covariance of the returns. In Section 4.5.4, we noted the lack of statistical foundation of the variant proposed by Jorion (1992) and Michaud (1998), and in Section 4.6 we applied nonparametric bootstrap methods to estimate confidence intervals for the Sharpe ratio.[15] In this section we continue to use the nonparametric bootstrap to calculate and visualize the variability of both robust and classical mean-variance efficient frontiers and their maximum Sharpe ratios.

A primary advantage of using the nonparametric bootstrap to assess the average behavior and variability of these two types of efficient frontiers is that the results tell us all that the data have to say about the unknown distribution of the multivariate returns. In this kind of bootstrap sampling, some samples will have fewer outliers (sometimes zero outliers) than in the original sample and some will have more outliers than in the original sample. In this way, the efficient frontier variability is reflecting the various possible future efficient

frontier curves based on samples coming from a nonparametric estimate of the unknown returns distribution.

The dashed lines in Figure 6.43 show 30 bootstrapped classical mean-variance efficient frontiers for the small cap portfolios of Figure 6.36 and Figure 6.40, and the dashed lines in Figure 6.44 show 30 bootstrapped robust efficient frontiers for the same portfolio. The solid line in each figure is the efficient frontier based on the original set of returns. The same bootstrap replicates are used for each figure (i.e., for each mean-variance efficient frontier there is a corresponding robust efficient frontier computed with the same bootstrap sample). Each solid dot is the "bullet point" representing the (global) minimum variance portfolio for the corresponding bootstrap sample and associated efficient frontier. It may be noted immediately that both the original efficient frontiers are biased in that they are not centrally located in the scatter of bootstrap efficient frontiers. Since the horizontal and vertical ranges of both axes are the same in the two figures, you can deduce that the mean return of the robust minimum variance portfolio appears to be less than that of the mean-variance minimum variance portfolio, while the risk of the former is at least as small as the risk of the latter.

Figure 6.45 shows boxplots of the differences for each bootstrap sample between the mean returns and risks of the robust and mean-variance minimum variance portfolios along with the differences between the Sharpe ratios. The

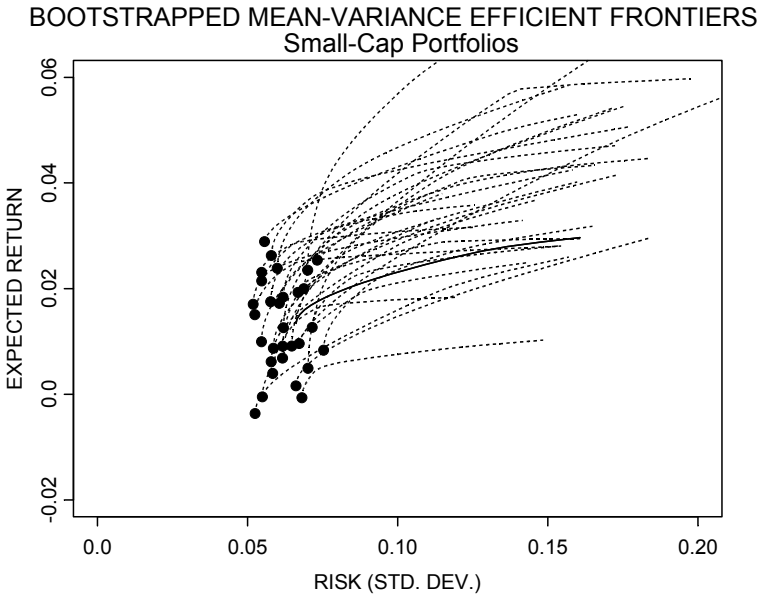

Figure 6.43 Bootstrapped Mean-Variance Efficient Frontiers for Small Cap Portfolio

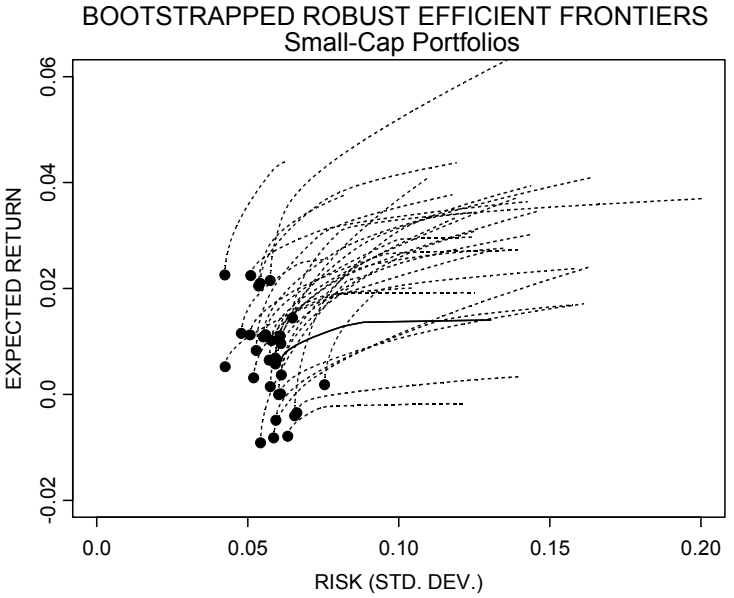

Figure 6.44 Bootstrapped Robust Efficient Frontiers for Small Cap Portfolio

notches in the boxplots correspond to approximate 95% confidence intervals for the median differences. None of these confidence intervals contains zero, indicating that the performance of the robust optimal portfolio is significantly lower than that of the mean-variance portfolio at a level of 5%. The median difference between the mean-variance and robust Sharpe ratios is only about –.06, which is not likely to be of much financial consequence.

The choice $B = 30$ for the number of bootstrap replicates may well be too small to draw firm conclusions. To get an idea of how things will change with an increasing number of replicates, we ran the bootstrap program with $B = 100$ replicates (without plotting efficient frontiers); the resulting boxplots are shown in Figure 6.46. For this particular example, the results are not substantially different from those of the bootstrap with $B = 30$.

The S-PLUS and NUOPT code for the above computations is provided below in the form of the two functions `boot.efronts` (Code 6.12) and `efront.nuopt.forboot` (Code 6.13) and a short script (Code 6.14) for calling `boot.efronts` using the small cap returns. When running Code 6.12 and Code 6.13, one must allow sufficient time for the bootstrap computations to finish.

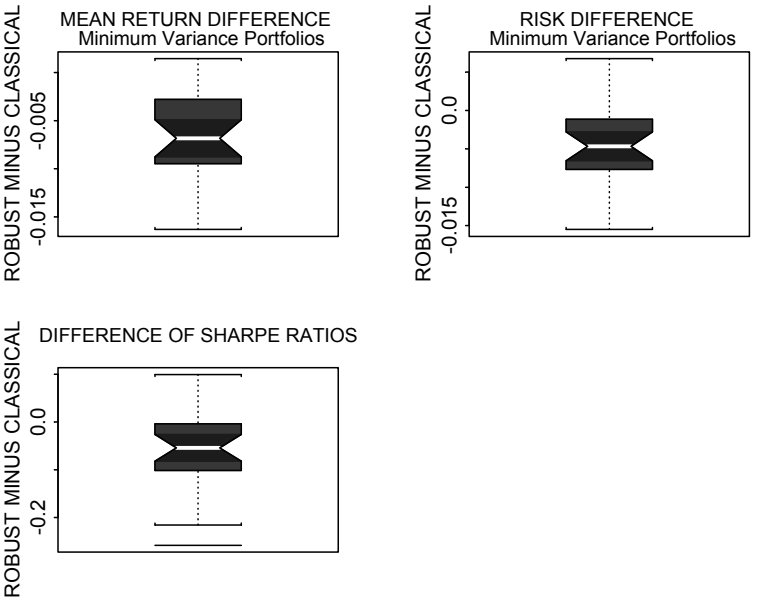

Figure 6.45 Bootstrap Portfolio Performance Differences for $B = 30$

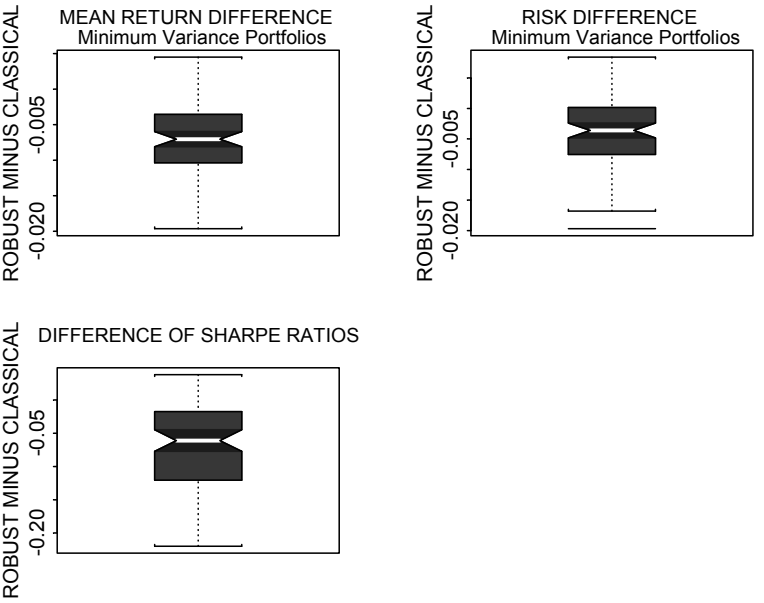

Figure 6.46 Bootstrap Portfolio Performance Differences for $B = 100$

```
boot.efronts <- function(returns.ts, B=30, rf=.03,
  npoints=20, k.mu=4, k.sigma=1.5, mv=T, tan=F,
  plotit=F, estim = "mcd", quan=.9)
{
  # B is the number of bootstrap samples
  # k.mu controls vertical axis plotting range
  # k.sigma controls horizontal axis plotting range
  # Adjust k.mu, k.sigma to minimize plot
  # "Line out of bounds" Warnings
  # mv = T to display bullet at minimum var.
     portfolio
  # tan = T to display bullet at tangency portfolio
  # plotit = T to plot efficient frontiers
  # estim = "mcd" to use MCD est. (avoid auto.
  # default choice)
  # quan is the fraction of data used by the MCD
  # Compute Bootstrap Samples Indices
  returns <- seriesData(returns.ts)
  n <- nrow(returns)
  m <- ncol(returns)
  B <- 30
  boot.index <- samp.boot.mc(n,B)
  # Compute Classic Efficient Frontier
  covmat <- var(returns)
  mu <- apply(returns,2,mean)
  max.ret <- max(mu)
  ef <- portfolioFrontier(covmat, mu, wmin=0,
     max.ret=max.ret,n.ret=npoints)
  sd = ef$sd
  ret = ef$returns
  # Compute Robust Efficient Frontier
  cov.rob <- covRob(returns,estim = estim,
     quan = quan)
  covmat.rob <- cov.rob$cov
  #mu.rob <- cov.rob$center
  mu.rob <- apply(returns,2,location.m)
  max.ret <- max(mu.rob)
  ef.rob <- portfolioFrontier(covmat.rob, mu.rob,
     wmin=0, max.ret=max.ret, n.ret=npoints)
  sd.rob <- ef.rob$sd
  ret.rob <- ef.rob$returns
  # Set Axis Limits
  xlim <- k.sigma*c(0,max(sd,sd.rob))
  ylim <- k.mu*c(0,max(ret,ret.rob))
  xlim <- c(0,.2)
```

```
ylim <- c(-.02, .06)
# Plot Original Classic Efficient Frontier
if(plotit) {
   plot(sd, ret, xlim = xlim, ylim = ylim,
      type = "l", xlab ="RISK (STD. DEV.)",
      ylab ="EXPECTED RETURN")
   lines(sd,ret,lwd = 2)
   title(main="BOOTSTRAP MEAN-VARIANCE EFFICIENT
      FRONTIERS \n Small cap Portfolios")
} #endif plotit

# Compute and Plot Classic Bootstrapped Frontiers
names <- c("SD.MV","MU.MV","SD.TAN","MU.TAN",
   "SHARPE")
out <- matrix(rep(0,5*B),ncol = 5)
dimnames(out) <- list(NULL,names)

for(i in 1:B) {
   ef.classic = efront.nuopt.forboot(
      returns[boot.index[,i],],plotit,
      robust = F, estim = estim, quan = quan,
      rf = rf, mv = mv, tan = tan)
   out[i,] = ef.classic
} # endfor i in 1:B
round(out,3)
# Plot Original Robust Efficient Frontier
if(plotit) {
   plot(sd.rob, ret.rob, xlim = xlim, ylim =
   ylim,
      xlab = "RISK (STD. DEV.)",
      ylab = "EXPECTED RETURN",type = "l")
   lines(sd.rob,ret.rob,lwd = 2)
   title(main="BOOTSTRAP ROBUST EFFICIENT
   FRONTIERS
      \n Small cap Portfolios")
} # endif plotit

# Compute and Plot Robust Bootstrapped Frontiers
names <- c("SD.MV","MU.MV","SD.TAN","MU.TAN",
   "SHARPE")
out.rob <- matrix(rep(0,5*B),ncol = 5)
dimnames(out.rob) <- list(NULL,names)
for(i in 1:B){
   ef.rob = efront.nuopt.forboot(
      returns[boot.index[,i],],plotit,
```

```
         robust = T, estim = estim, quan = quan,
         rf = rf, mv = mv, tan = tan)
      out.rob[i,] = ef.rob
}
   round(out.rob,3)
   par(mfrow = c(2,2))
   boxplot(out.rob[,2]-out[,2],notch = T,
      ylab = c("ROBUST MINUS CLASSICAL"))
   title(main = "MEAN RETURN DIFFERENCE \n Minimum
      Variance Portfolios")
   boxplot(out.rob[,1]-out[,1],notch = T,
      ylab = c("ROBUST MINUS CLASSICAL"))
   title(main = "RISK DIFFERENCE \n Minimum
      Variance Portfolios")
   boxplot(out.rob[,5]-out[,5],notch = T,
      ylab = c("ROBUST MINUS CLASSICAL"),
      ylim = range(out.rob[,5]-out[,5]))
   title(main = "DIFFERENCE OF SHARPE RATIOS")
   par(mfrow = c(1,1))
 }
```

Code 6.12 Bootstrapped Efficient Frontiers and Sharpe Ratios

```
 efront.nuopt.forboot <- function(returns, plotit =
   T, robust, estim, quan, rf = 0.005, mv = T, tan =
   F, npoints = 50)
 {
   if(robust) {
      covmat <- covRob(returns,estim = estim,
         quan = quan)$cov
      mu <- apply(returns,2,location.m)
   }
   else {
      covmat <- var(returns)
      mu <- apply(returns,2,mean)
   }
   #sd <- apply(returns, 2, stdev)
   ef <- portfolioFrontier(covmat, mu,wmin = 0,
      max.ret = max(mu), n.ret = npoints)
   sdopt <- ef$sd

   muopt <- ef$returns
   # Compute minimum variance portfolio
   port.mv <- c(sdopt[1], muopt[1])
   names(port.mv) <- c("SD.MV", "MU.MV")
   # Compute tangency portfolio
```

```
  sharpe <- (muopt - rf)/sdopt
  iopt <- order(sharpe)[npoints]
  sharpe.max <- sharpe[iopt]

  names(sharpe.max) <- "SHARPE"
  port.tan <- c(sdopt[iopt], muopt[iopt])
  names(port.tan) <- c("SD.TAN", "MU.TAN")
  # Plot results
  if(plotit) {
     lines(sdopt, muopt, lty = 8)
     #points(max(sdopt), max(muopt), pch = ".")
     if(mv == T)
        points(port.mv[1], port.mv[2], pch = 16)
     if(tan == T)
        points(port.tan[1], port.tan[2], pch = 18)
  }
  c(port.mv, port.tan,sharpe.max)
}
```

Code 6.13 NUOPT Efficient Frontiers for Bootstrap Function

```
tickers <- c("TOPP","KWD","HAR","RARE","IBC")
returns.ts <- smallcap.ts[,tickers]
boot.efronts(returns.ts,plotit = T)
```

Code 6.14 Bootstrap Efficient Frontiers Example

The reader is encouraged to experiment with Codes 6.12–6.14 on a variety of portfolios using the returns data set included with this book.

6.9.5 Efficient Frontiers Based on the Classical and Robust Stambaugh Methods

In Section 6.7, we discussed the Stambaugh normal distribution maximum likelihood method of estimating a mean vector and covariance matrix for asset returns having unequal histories and showed how to make the method robust. With these two types of mean vector and covariance matrix estimates in hand, we can proceed as usual to compute both a classical and a robust efficient frontier. Figure 6.47 and Figure 6.48 show the results of doing this using the returns pictured in Figure 6.28. Note the dominant role of Health along with Events in determining the limits of the classical efficient frontier as compared with the dominant role of Equity along with Events in determining the robust efficient frontier. Note also that the reduction in risk when moving from classical sample standard deviations to robust standard deviations is roughly the same for all indices except Health, which exhibits a much more substantial

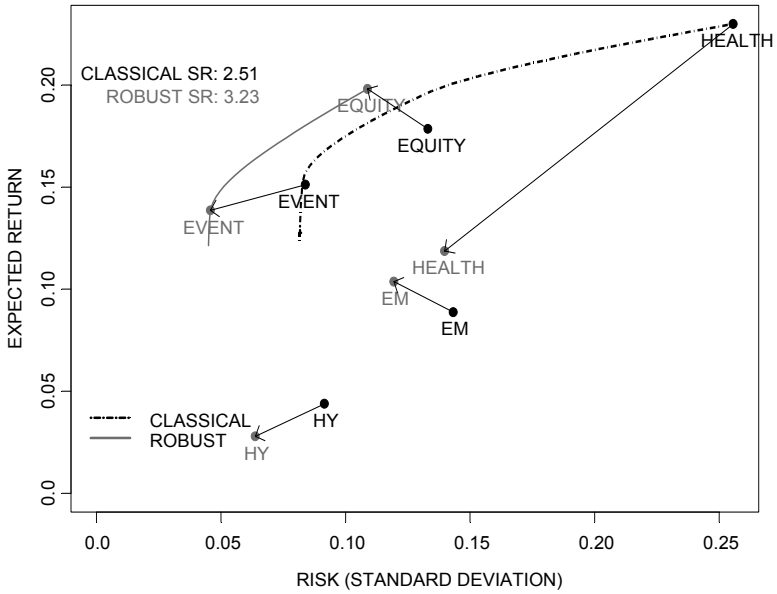

Figure 6.47 Efficient Frontiers for Index Returns with Unequal Histories

reduction in risk. Health also exhibits a substantial reduction in mean return when moving from the sample mean estimate to a robust location estimate. The distribution of portfolio weights along the two efficient frontiers is quite different, with the classical portfolio weights relying more heavily on Health at higher levels of risk and return. By way of contrast, the robust portfolio relies more heavily on Equity at higher levels of return (and risk) and gives Health a zero weight for all possible portfolios.

A glance at the time series of returns for the indices in Figure 6.28 reveals that Health was giving exceptional gains during 1999 (probably by riding the dot-com bubble) and exhibited exceptional losses during the dot-com crash in 2000, followed by relatively lower volatility and unexceptional returns during 2001, 2002, and early 2003. Therefore, it would not be surprising to find many investors preferring the robust efficient frontier for making their investment decision. One can say the robust approach is an automatic method for down-weighting the anomalous returns in the data, thereby calculating an efficient frontier that represents the "normal" behavior of the data. Lacking special information, the "normally" behaving data are the only part of the data that is predictable.

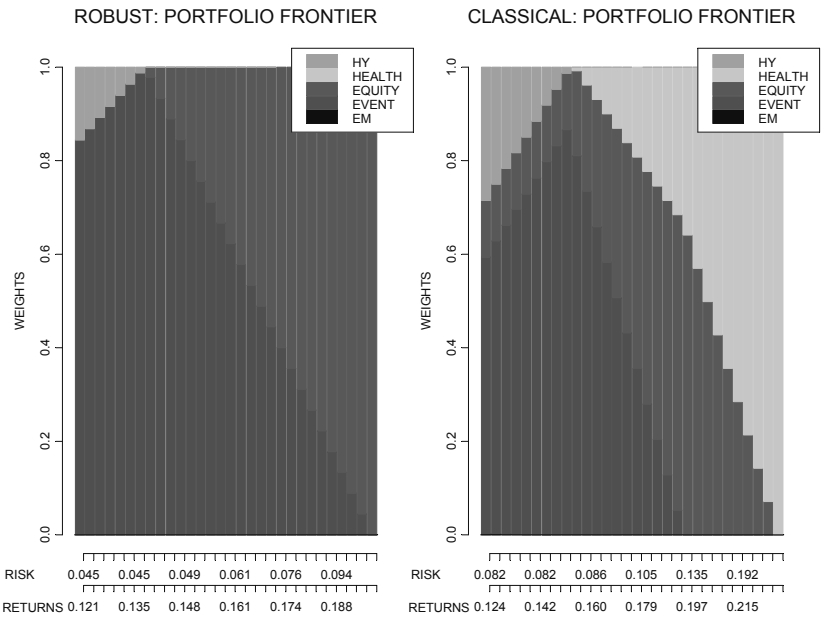

Figure 6.48 Classic and Robust Portfolio Weights for Hedge Fund Indices

6.10 Conditional Value-at-Risk Frontiers: Classical and Robust

A definition of coherent risk measure introduced by Artzner et al. (1997, 1999) was discussed in Section 5.6.2, where it was pointed out that value-at-risk (VaR) is not a coherent risk measure.[16] It is also the case that standard deviation is not a coherent risk measure, which makes the classical Markowitz mean-variance method suspect. On the other hand, it was pointed out in Sections 5.6.2 and 5.6.3 that conditional value-at-risk (CVaR) is a coherent risk measure that leads to a computationally attractive portfolio optimization approach. The question, therefore, is how one might make the CVaR method of portfolio optimization robust. We note that a CVaR optimal portfolio (CVaR portfolio for short) does not involve an estimate of the covariance matrix; it only involves estimation of the mean returns. Thus, at first glance one can make a CVaR robust by simply replacing the sample mean estimates of the μ_i by one of the robust location estimates in Section 6.2. However, outliers can also influence the

individual scenario returns $\sum_{i=1}^{n} w_i r_{i,s}$ in $e_s = \max\left[0, VaR - \sum_{i=1}^{n} w_i r_{i,s}\right]$ and hence can influence the value of the objective function $CVaR = VaR - \frac{1}{\alpha \cdot m}\sum_{i=1}^{m} e_s$ (see Section 5.6.3). In order to control the influence of outliers on this objective function, one needs to down-weight the values of the $r_{i,s}$ appearing above.

With regard to down-weighting the $r_{i,s}$, there are two questions. First, should one down-weight these values at all? After all, the whole point of using CVaR is to use tail risk to find optimal portfolio weights, and in general one expects that one or more large negative returns in a given asset will tend to reduce the weight in that asset in the optimal CVaR portfolio. Down-weighting such large negative returns might be counterproductive. Second, how should one down-weight the values?

We defer the second question for a moment and address the first. As we pointed out earlier, the use of robust portfolio computations is not a be-all and end-all. The most important value of a robust portfolio is its diagnostic value: when the robust and classical efficient frontiers agree, there is no need to worry, and when they differ the portfolio manager needs to make a decision on which to use based on all the other information available. In the end, the manager may be inclined to take one of the following positions:

> The *robust view,* in which the manager does not trust that *any* past outlier returns will repeat themselves with any degree of predictability and therefore uses a robust portfolio solution since it reflects the behavior of the bulk of the data, excluding outliers;
>
> The *pessimistic view,* in which the manager does not trust that past *positive* outliers are to be expected over the investment horizon, but that past negative returns outliers indicate possible future negative returns outliers, and therefore *down-weights only positive returns outliers*;
>
> The *optimistic view,* in which the manager does not believe that past *negative* returns outliers will repeat themselves over the investment horizon, but that past positive returns outliers indicate possible future positive returns outliers, and therefore *down-weights only negative returns outliers*.

The optimistic or pessimistic views might be taken, for example, when the corresponding outlier or outliers occur only during the early part of the history used to optimize the portfolio or when the manager has other information about some or all of the portfolio assets under consideration.

With these three managerial views in mind, one might first think to use robust distances based on a robust covariance matrix estimate as in Section 6.6 to create weights for each scenario vector $r_s = (r_{1,s}, r_{2,s}, \cdots, r_{n,s}),\ s = 1, 2, \cdots, m.$ There are at least two reasons why this is not a highly appealing approach. First, it is not clear how to modify the robust distance approach in a simple manner to accommodate the three distinct manager views. Second, since CVaR portfolio optimization does not require computation of a covariance matrix (which can be computationally burdensome when dealing with a large portfolio), it is attractive to avoid this approach. Consequently, we propose a much simpler approach based on down-weighting outliers in each set of asset returns, one at a time, according to which of the views above the manager takes. While this simpler approach has the deficiency that it is not able to detect and down-weight potentially influential multivariate returns outliers that are not univariate outliers, it has the virtue of simplicity and appears to help in many situations occurring in practice.

We use a special form of the outlier-down-weighting approach, often called **trimming**, which is done as follows: for each set of returns $r_{i,s},\ s = 1, 2, \cdots, m,$ compute a robust location estimate $\hat{\mu}_i$, a robust scale estimate $\hat{\sigma}_i$, and the resulting residuals $res_{i,s} = r_{i,s} - \hat{\mu}_i$. For a manager with a robust view, compute the symmetrically trimmed returns

$$\tilde{r}_{i,s} = \begin{cases} r_{i,s}, & |res_{i,s}| \le a \cdot \hat{\sigma}_i \\ \hat{\mu}_i, & |res_{i,s}| > a \cdot \hat{\sigma}_i. \end{cases} \tag{6.20}$$

For a manager with a pessimistic view, compute the positive-trimmed returns

$$\tilde{r}_{i,s} = \begin{cases} r_{i,s}, & res_{i,s} \le a \cdot \hat{\sigma}_i \\ \hat{\mu}_i, & res_{i,s} > a \cdot \hat{\sigma}_i, \end{cases} \tag{6.21}$$

and for a manager with an optimistic view, compute the negative-trimmed returns

$$\tilde{r}_{i,s} = \begin{cases} \hat{\mu}_i, & res_{i,s} \le -a \cdot \hat{\sigma}_i \\ r_{i,s}, & res_{i,s} > -a \cdot \hat{\sigma}_i. \end{cases} \tag{6.22}$$

We use a default value $a = 3$, which results in about 2.6 symmetric trimmed residuals out of 1000 for normally distributed returns and about 1.3 out of 1000 for the two other cases.

Code 6.15 gives the S-PLUS code for the function `trimmed.returns` that computes the above trimmed returns:

```
trimmed.returns <- function(x, view = "robust",
  a = 3)
```

```
{
  p <- ncol(x)
  n <- nrow(x)
  ind <- matrix(0,nrow = n, ncol = p)
  for(j in 1:p) {
     mu <- location.m(x[,j])
     scale <- scale.tau(x[,j])
     resid <- x[,j]- mu
     if(view == "pessimistic") {
        x[resid > a*scale,j] <- mu
        ind[resid > a*scale,j] <- 1
     }
     else if(view == "optimistic") {
        x[resid < -a*scale,j] <- mu
        ind[resid < -a*scale,j] <- 1
     }
     else if(view == "robust") {
        x[abs(resid) > a*scale,j] <- mu
        ind[abs(resid) > a*scale,j] <- 1
     }
     else
        stop("view must be \"pessimistic\",
           \"optimistic\", or \"robust\"")
     ind <- data.frame(ind)
     names(ind) <- names(x)
  }

  list(returns.trimmed = x,ind = ind)
}
```

Code 6.15 Trimmed Returns

Once the matrix of returns has been trimmed using the function above, you compute a CVaR efficient frontier in a manner similar to that used in Section 5.6.3. We note that in Section 5.6.3 the function `CVaR.frontier` computes a frontier for long-only portfolios and for target returns ranging from the minimum return sample mean return to the maximum return sample mean. Consequently, the resulting frontier typically contains inefficient positions. In order to compute a CVaR efficient frontier, we first need to find the global minimum CVaR portfolio. This is easily done as follows. Remove the line of code that specifies a return constraint from the function `CVaR.model` in Section 5.6.3:

```
Sum(mu.bar[i]*w[i],i) == mu.target.
```

Name the resulting function `CVaR.globalmin.model`. Now you need to modify the function `CVaR.portfolio` by replacing the code line

```
call(CVaR.model)
```

in that function with

```
call(CVaR.globalmin.model)
```

Name the new function `CVaR.globalmin.portfolio`. You should also add the lines

```
S <- as.matrix(S)
dimnames(S) <- NULL
```

at the beginning of these two functions, the first to allow a data frame as an argument to the function and the second to remove the column names, which the current version of NUOPT does not accept. Since the function `CVaR.globalmin.portfolio` differs in a few other places from `CVaR.portfolio`, Code 6.16 gives the code for the revised version of `CVaR.globalmin.portfolio`:

```
CVaR.globalmin.portfolio <- function(S, alpha)
{
  S <- as.matrix(S)
  dimnames(S) <- NULL
  call(CVaR.globalmin.model)
  CVaR.system <-
     System(CVaR.globalmin.model,S,alpha)
  solution <- solve(CVaR.system, trace=T)
  weight <- solution$variable$w$current
  w <- as.matrix(weight)
  mu <- as.matrix(apply(S,2,mean))
  mu.min <- as.numeric(t(w)%*%mu)
  risk <- solution$objective

  return(mu.min,risk)
}
```

Code 6.16 Global Minimum CVaR Portfolio

The argument `alpha` above, and in what follows, specifies the tail probability for CVaR.

Code 6.17 provides `CVaR.eff.frontier`, a slightly modified version of the function `CVaR.frontier` in Section 5.6.3, that computes and optionally plots a CVaR efficient frontier:

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

  for(i in 2:n.pf) {
     x <- CVaR.portfolio(S,alpha,
        mu.target=mu.range[i])
     Risk[i] <- x$risk
     Return[i] <- mu.range[i]
     weight <- cbind(weight,x$weight)
  }
  # Convert CVaR to a positive quantity
  Risk = - Risk
  if(plot) {
     par(mfrow=c(1,2))
     plot(Risk, Return, type="b",xlab = "RISK",
        ylab = "RETURN")
     title("MEAN vs. CVaR EFFICIENT FRONTIER")
     barplot(weight,legend = names(S))
     title("FRONTIER PORTFOLIOS")
  }
  list(Risk = Risk, Return = Return, Weights =
     weight)
}
```

Code 6.17 CVaR Efficient Frontier

Note that in order to use the function `CVaR.eff.frontier`, as in the examples below, you need to have already created the functions `CVaR.model` (Code 5.9) and `CVaR.portfolio` (Code 5.10).

Figure 6.49 shows the time series of five years of monthly returns for four stocks, with tickers BKE, GG, GYMB, and KRON, for which we will compute a CVaR efficient frontier.

The plots shown in Figure 6.49 were made by extracting the stocks from the `smallcap.ts` time series object and using the `seriesPlot` (see Code 6.7) function as follows:

```
tickers <- c("BKE","GG","GYMB","KRON")
returns.ts <- smallcap.ts[,tickers]
seriesPlot(returns.ts,strip.text =
  colIds(returns.ts),
  trellis.args = list(as.table = T,type = "l"),
  one.plot = F)
```

Note the positive and negative returns outliers and that depending upon the investor's knowledge he may wish to take any one of the three views we have proposed. For example, the investor may know that the large outlier in the GG returns was associated with a singular event that is not expected to recur in the next year or two and may feel that the two positive outlier returns in GYMB present an overly optimistic view of future performance. Consequently, he will want a CVaR optimal portfolio constructed with a pessimistic view. On the other hand, the investor may feel that most of the negative outliers are sufficiently far in the past, or, as in the case of KRON, are left in the dust by a strong positive trend, leading him to construct a CVaR portfolio based on a positive view. Finally, the investor may feel that the positive and negative outliers tend to have cancelling effects and have no good reason to believe they will occur in the next year. Consequently, he will compute a CVaR portfolio based on a robust view that reflects the behavior of the bulk of the returns.

Assuming we have created the `returns.ts` object as above, we can compute the standard CVaR efficient frontier and barplot of weights in Figure 6.50 with the commands (the computation takes noticeably longer than a mean-variance optimal frontier):

```
returns = returns.ts@data
CVaR.eff.frontier(returns, alpha = .05, n.pf = 10)
```

You can now use `trimmed.returns` to compute the CVaR efficient frontier based on one of the three possible manager views. For the robust view, use:

```
returns.tr <-
     trimmed.returns(returns)$returns.trimmed
```

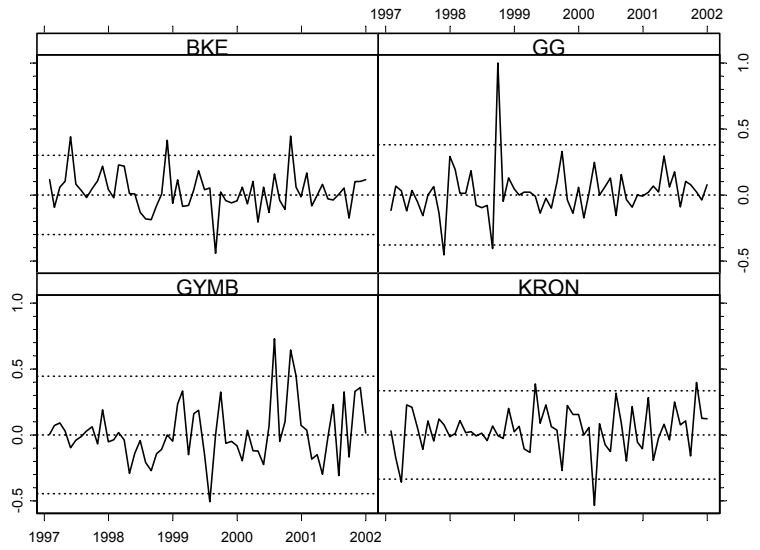

Figure 6.49 Time Series of Returns for Four Stocks

```
CVaR.eff.frontier(returns.tr,alpha = .05,n.pf = 10)
```

The results are the CVaR efficient frontier and portfolio weights in Figure 6.51, where upon careful inspection you notice the change in efficient frontier location and the change in weights relative to those in Figure 6.50.

Of course, what we really want to have is overlaid efficient frontiers and displayed values of the mean return and CVaR for each stock as in Section 6.8, where standard deviations were used as the risk measure. We can easily do this by modifying Code 6.11. The only additional function specific to the CVaR context that we need is a little function to compute the CVaR of each set of returns, rather than the standard deviation, so that we can display each stock in the mean return versus CVaR coordinates. This simple function is given in Code 6.18.

```
CVaR.simple <- function(x, alpha = .05) {
  k = floor(length(x)*alpha)
  #convert CVaR to a positive quantity
  -mean(sort(x)[1:k])
}
```

Code 6.18 CVaR Computation Function

Now we show a few examples before providing the code needed to produce them. Figure 6.52 provides an overlaid version of the CVaR efficient frontiers of

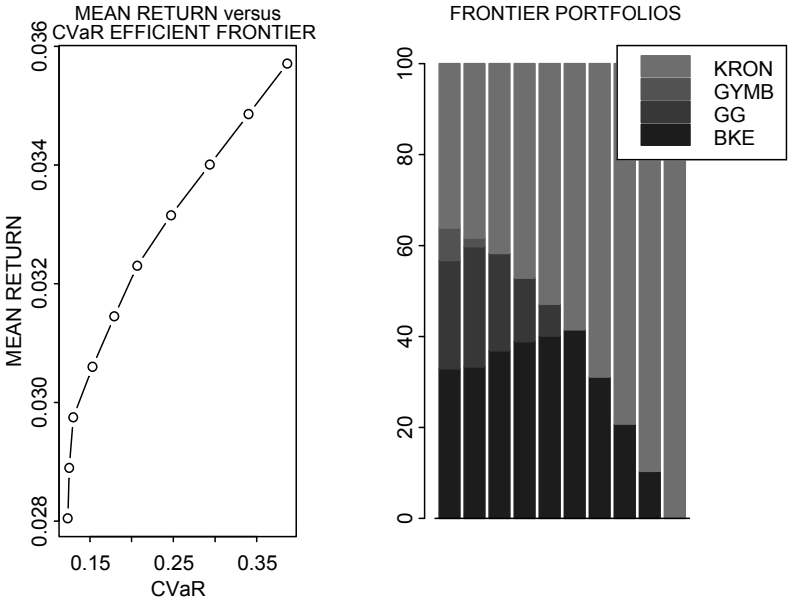

Figure 6.50 CVaR Efficient Frontier and Weights for Stock Returns of Figure 6.48

Figure 6.50 and Figure 6.51 along with the individual stocks' sample means and sample CVaRs.

In Figure 6.52 one sees that the robust CVaR efficient frontier yields larger returns than the standard CVaR efficient frontier, with the difference increasing with increasing CVaR risk. Figure 6.53 and Figure 6.54 show the results for pessimistic and optimistic views, respectively.

Figure 6.53 shows that the manager with a pessimistic view gets lower returns than the standard CVaR manager at all levels of CVaR below that of KRON, with the largest difference at the global minimum CVaR values. Finally, Figure 6.54 shows that the manager with an optimistic view gets mean returns that are uniformly higher than the classic CVaR returns at all levels of CVaR. Note also that the gain of the optimistic CVaR portfolio in Figure 6.53 relative to the robust view CVaR portfolio in Figure 6.54 is most substantial for the smaller values of CVaR. This is quite understandable based on the differences in trimming for these two views and the fact that both negative and positive outlier returns are evident in Figure 6.49.

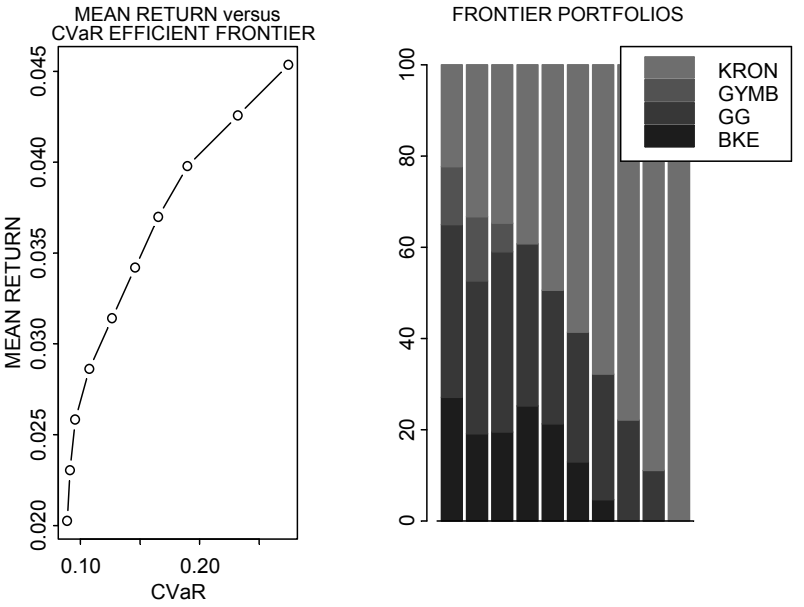

Figure 6.51 CVaR Efficient Frontier and Weights with Robust View Trimming of Returns

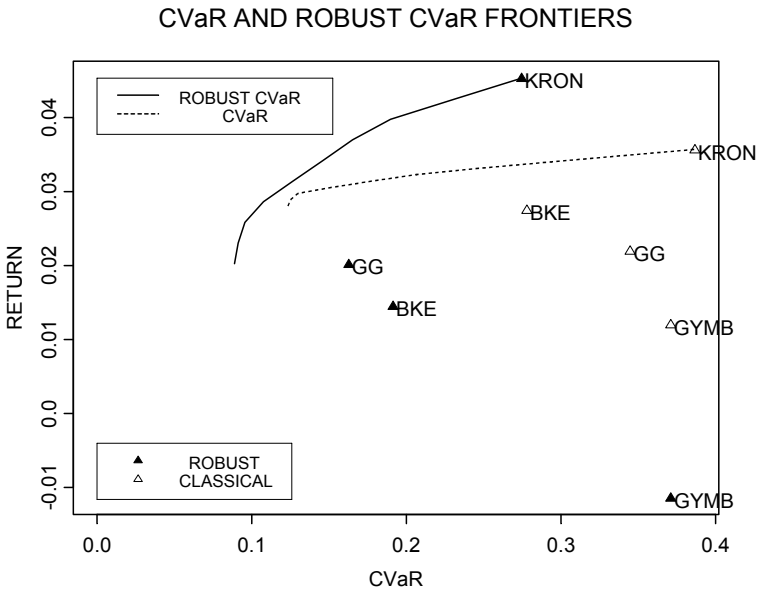

Figure 6.52 CVaR and Robust View CVaR Efficient Frontiers

Code 6.19 gives the code for making the plots above (just change `view = "robust"` to `view = "pessimistic"` and `view = "optimistic"` to get Figure 6.53 and Figure 6.54, respectively):

```
tickers <- c("BKE","GG","GYMB","KRON")
returns.ts <- smallcap.ts[,tickers]
returns <- returns.ts@data
p <- ncol(returns)
# Parameters
alpha <- .05
view <- "robust"
display.letters <- F
display.points <- T
display.names <- T
n.pf <- 10
plot.weights <- F
series.plots <- F
# Time Series Plots
if(series.plots)
  seriesPlot(returns.ts,
     strip.text = colIds(returns.ts),
     trellis.args = list(as.table = T,type = "l"),
     one.plot = F)
# Compute Standard CVaR Efficient Frontier
ef.cvar <- CVaR.eff.frontier(returns, alpha, n.pf,
  plot = F)
ef.cvar$Weights
# Compute Robust CVaR Efficient Frontier
ret.trimmed <-
  trimmed.returns(returns,view)$returns.trimmed
ef.cvar.robust <- CVaR.eff.frontier(ret.trimmed,
  alpha, n.pf, plot = F)
ef.cvar.robust$Weights

# Plot Efficient Frontiers
if(display.letters || display.points) {
  mu1 <- apply(returns,2,mean)
  mu2 <- apply(ret.trimmed,2,mean)
  cvar1 <- apply(returns,2,CVaR.simple,alpha=alpha)
  cvar2 <- apply(ret.trimmed,2,CVaR.simple,
     alpha=alpha)
  xlim <- range(ef.cvar$Risk,ef.cvar.robust$Risk,
     cvar1,cvar2,0)
```

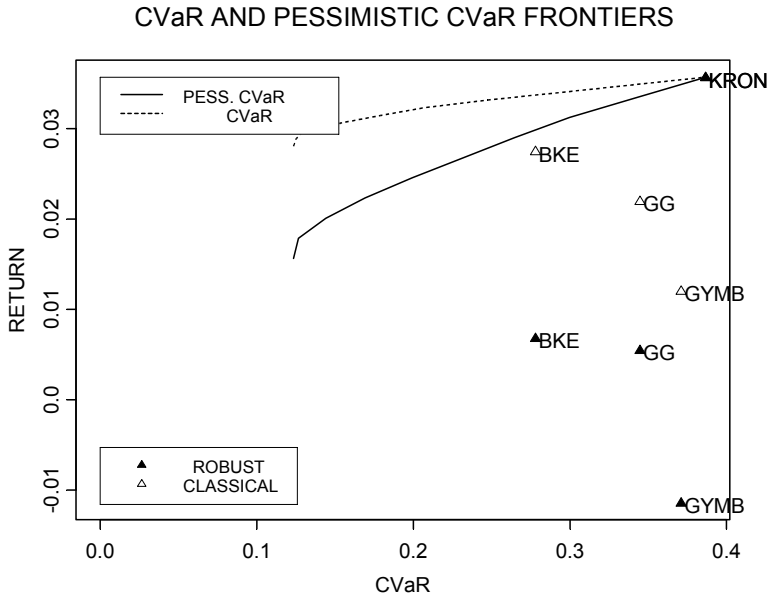

Figure 6.53 CVaR and Pessimistic View CVaR Efficient Frontiers

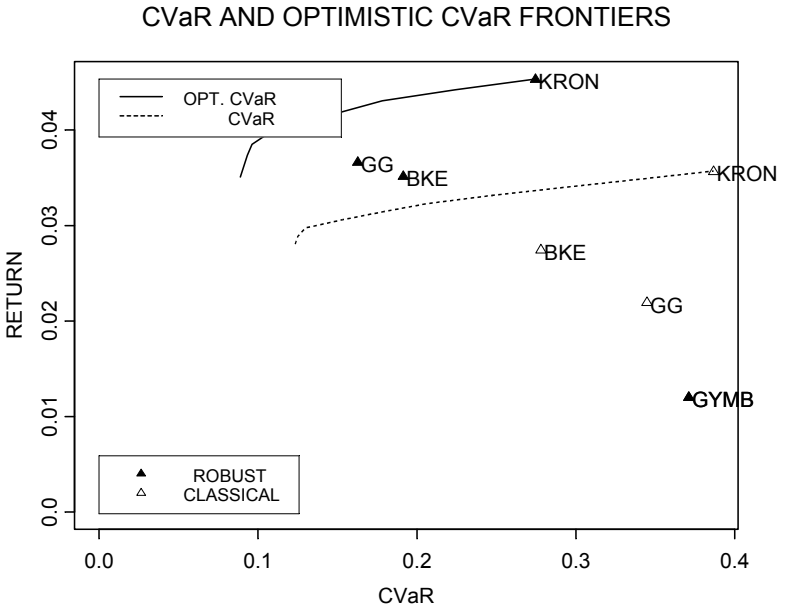

Figure 6.54 CVaR and Optimistic View CVaR Efficient Frontiers

```
  ylim <-
     range(ef.cvar$Return,ef.cvar.robust$Return,
     mu1,mu2,0)
}
else {
  xlim <- range(ef.cvar$Risk,ef.cvar.robust$Risk,0)
  ylim <- range(ef.cvar$Return,
     ef.cvar.robust$Return,0)
}
plot(ef.cvar$Risk,ef.cvar$Return,xlim=xlim,
  ylim=ylim,  type = "n",xlab="CVaR",ylab="RETURN")
lines(ef.cvar$Risk,ef.cvar$Return,lty = 8,lwd = 2)
lines(ef.cvar.robust$Risk,ef.cvar.robust$Return,
  lwd=2)
if(view == "robust") {
  title(main="CVaR AND ROBUST CVaR FRONTIERS")
}
else if(view == "pessimistic") {
  title(main="CVaR AND PESSIMISTIC CVaR FRONTIERS")
}
else if(view == "optimistic") {
  title(main="CVaR AND OPTIMISTIC CVaR FRONTIERS")
}

# Add Frontiers Legend
x <- xlim[1]+.0*(xlim[2]-xlim[1])
y <- ylim[2]-.0*(ylim[2]-ylim[1])
if(view == "robust") {
  leg.names <- c("ROBUST CVaR","          CVaR")
}
else if(view == "pessimistic") {
  leg.names <- c("PESS. CVaR","          CVaR")
}
else if(view == "optimistic") {
  leg.names <- c("OPT. CVaR","          CVaR")
}
legend(x,y,leg.names,lty=c(1,8), lwd = 2)

# Plot Stock Mu's and CVaR's and Add Legend
if(display.letters){
  for(i in 1:p) {
     points(cvar1[i],mu1[i],pch = letters[i])
     points(cvar2[i],mu2[i],pch = LETTERS[i])
  }
```

```
  if(display.names) {
     text(cvar1 + 0.002, mu1, names(mu1), adj= 0)
     text(cvar2 + 0.002, mu2, names(mu2), adj= 0)
  }
  x <- xlim[1]+.15*(xlim[2]-xlim[1])
  y <- ylim[1]+.10*(ylim[2]-ylim[1])
  leg.names <- c("A,B,. Robust Mu's,CVaR's",
     "a,b,. Classic Mu's,CVaR's")
  text(x,y,leg.names[1])
  text(x,y-.002,leg.names[2])
}
if(display.points) {
  points(cvar1,mu1,pch = 2)
  points(cvar2,mu2,pch = 17)
  if(display.names) {
     text(cvar1 + 0.002, mu1, names(mu1), adj= 0)
     text(cvar2 + 0.002, mu2, names(mu2), adj= 0)
  }
  x <- xlim[1]+.00*(xlim[2]-xlim[1])
  y <- ylim[1]+.13*(ylim[2]-ylim[1])
  leg.names <- c("  ROBUST","CLASSICAL")
  legend(x,y,leg.names, marks = c(17,2))
}
# Plot Portfolio Weights for Both Efficient
# Frontiers
if(plot.weights) {
  par(mfrow = c(1,2))
  barplot(ef.cvar$Weights,legend = names(returns))
  title(main = "CVaR FRONTIER WEIGHTS")
  barplot(ef.cvar.robust$Weights,
     legend = names(returns))
  if(view == "robust") {
     title(main="ROBUST CVaR FRONTIER WEIGHTS")
  }
  else if(view == "pessimistic") {
     title(main="PESSIMISTIC CVaR FRONTIER
        WEIGHTS")
  }
  else if(view == "optimistic") {
     title(main="OPTIMISTIC CVaR FRONTIER WEIGHTS")
  }
  par(mfrow = c(1,1))
}
```

Code 6.19 CVaR Efficient Frontier Plots

6.10.1 Manager Views and What-If Predictive Diagnostics

We want to stress the predictive and diagnostic nature of the optimal CVaR efficient frontiers above. First of all, any kind of efficient frontier calculation is an "in-sample" predictive model in the sense that when one selects a particular portfolio to use based on such an efficient frontier, one is predicting that the efficient frontier is a reasonable predictor of future mean return-risk trade-off. Second, it may often be the case that an asset manager is at first unwilling to take any one of the views proposed above (robust, pessimistic, optimistic) or even a "standard" view that there should be no outlier treatment. In such cases, the manager may derive considerable diagnostic benefit from a "what-if" analysis based on computing CVaR efficient frontiers for each of the views. If the results are all in reasonable agreement, there is little cause to worry about influential outliers. But if there are substantial differences between two or more of the views, such analysis can act as a catalyst for the asset manager to investigate any unusual positive or negative returns that may be influencing the results. This may lead the manager to adopt a particular view based on a deeper knowledge of what has caused the events and his belief about whether they are likely to occur during the investment horizon.

We also remark that the simple return trimming method used here for CVaR portfolio optimization could also be used as a preprocessor for Markowitz mean-variance optimization. The additional computational burden of the robust covariance matrix calculation (in the case of a large number of assets) could then be avoided by instead using the classical mean and covariance matrix estimates based on univariate trimmed returns.

6.10.2 Choice of Alpha

Section 5.6.1 shows that estimates of CVaR are much more variable than estimates of VaR, which are in turn much more variable than estimates of standard error. This is natural in that CVaR is based on the mean value of the smallest $\alpha\%$ of the returns. Consequently, one can expect that CVaR efficient frontiers will be more variable than Markowitz mean-variance frontiers. (This could be checked with bootstrap experiments.) One way to mitigate this problem is to increase the value of α, say, to .1 or .2. Note that when $\alpha = .5$, CVaR is the mean of the returns below the median, which differs from lower semi-variance only by using the median in place of the overall mean. This choice may be interesting to asset managers in view of its very simple interpretation as the average losses below the median loss. Of course, the resulting portfolio weights do not pay as much relative attention to the downside returns when using larger values of alpha as when using smaller values of alpha. Examples of CVaR

frontiers for $\alpha = .05, .2,$ and $.5$ are provided in Figure 6.55, which is produced with Code 6.20.

```
tickers <- c("BKE","GG","GYMB","KRON")
returns <- smallcap.ts[,tickers]@data
cvar05 <- CVaR.eff.frontier(returns, alpha = .05,
  n.pf = 10, plot = F)
cvar2 <- CVaR.eff.frontier(returns, alpha = .2,
  n.pf = 10, plot = F)
cvar5 <- CVaR.eff.frontier(returns, alpha = .5,
  n.pf = 10, plot = F)
xlim <- range(cvar05$Risk,cvar2$Risk,cvar5$Risk)
ylim <-
     range(cvar05$Return,cvar2$Return,cvar5$Return)
plot(cvar05$Risk,cvar05$Return, type = "l",
  xlim = xlim, ylim = ylim,
  xlab = "CVaR", ylab = "MEAN RETURNS")
title(main = "CVaR EFFICIENT FRONTIERS FOR VARIOUS
  ALPHAS")
lines(cvar05$Risk,cvar05$Return, lty = 1, lwd =2)
lines(cvar2$Risk,cvar2$Return, lty = 4, lwd = 2)
lines(cvar5$Risk,cvar5$Return, lty = 8, lwd = 2)
leg.names = c("ALPHA = .05","ALPHA = .1",
  "ALPHA = .5")
legend(.26,.029,legend = leg.names, lty = c(1,4,8),
  lwd = 2)
```

Code 6.20 CVaR Efficient Frontiers for Different Values of Alpha

From Figure 6.55 it is clear that the efficient frontiers are not obtained from one another simply by a uniform scaling with respect to CVaR values. It will be useful to compare the portfolio weight profile for each of the frontiers (Exercise 11).

6.11 Influence Functions for Portfolios

Influence functions are powerful statistical tools for characterizing three key aspects of an estimator, namely: (a) the influence of individual data values on the estimator, particularly the influence of outliers; (b) the maximum bias of the estimator caused by small fractions of outliers; and (c) the asymptotic variance of the estimators.[17] While influence functions have been widely applied in statistics (see, for example, Hampel et al., 1986), there has been almost no use of them in finance and in portfolio construction in particular. This section

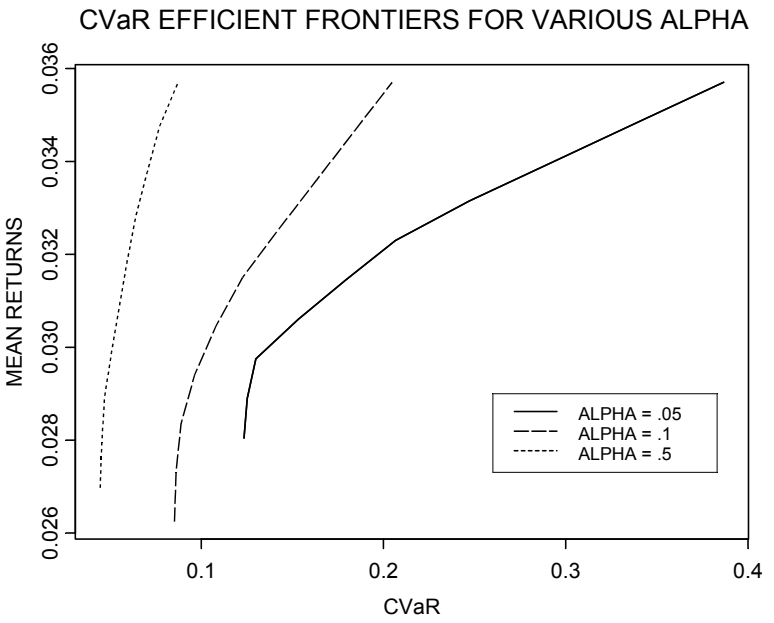

Figure 6.55 CVaR Efficient Frontiers for Three Values of Alpha

introduces the basic definitions and uses of influence functions and provides some initial applications to portfolio construction for the purpose of sensitivity analysis.

6.11.1 Introduction to Influence Functions

Influence functions have both finite sample and asymptotic versions. The intuitive motivation for the asymptotic influence function (**influence function** for short) comes from a finite sample form, one version of which is as follows. Let $\hat{\theta}_n = \hat{\theta}_n(x)$ be an estimator of a parameter θ based on a sample of data $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ of size n, and let x be an additional data point. To fix ideas you can think of $\hat{\theta}_n$ as a sample mean of returns or a sample standard deviation estimate of volatility for a particular stock. Then an empirical influence function (EIF) of $\hat{\theta}_n$ at $\boldsymbol{x}$ is the function of x given by

$$\text{EIF}(x; \hat{\theta}_n, \boldsymbol{x}) = (n+1) \cdot \left(\hat{\theta}_n(x, \boldsymbol{x}) - \hat{\theta}_n(\boldsymbol{x}) \right), \tag{6.23}$$

where the factor $n+1$ is used to normalize the result across sample sizes.[18] For a few simple estimators (such as the sample mean and sample median), it is

possible to compute an analytical expression for the EIF, but for most robust estimators this is not possible. However, one can numerically compute the EIF for a "typical" sample of returns $\boldsymbol{x}$ and plot the results as a function of the value of the additional data point x. Thinking of a normally distributed sample of returns, one can choose $\boldsymbol{x}$ to be normal random numbers whose mean and standard deviation are those of typical returns. A better approach that eliminates the variability of the random sample and gives a good rendition for small as well as large sample sizes is to let the sample $\boldsymbol{x}$ be the quantiles of a normal distribution.

Figure 6.56 displays the EIFs of the sample mean, the sample median, a 10% trimmed mean, and the optimal location M-estimate obtained from `lmRob`, as described in Section 6.3, using twenty quantiles of a standard normal distribution for the prototype data sample $\boldsymbol{x}$. The main messages from these EIFs are:

(a) The unbounded character of the EIF for the sample mean reflects the fact that a single outlier can cause an arbitrarily large influence on the sample mean.
(b) All the other estimates have bounded EIFs, reflecting the fact that a single outlier can only influence the estimate by a limited amount.
(c) The median has a nearly discontinuous EIF, which reflects the fact that the median has a certain "roughness" character.
(d) Very large outliers have zero EIF values for the optimal M-estimate, reflecting the fact that this estimate "rejects" sufficiently large outliers.

Note that the trimmed mean, which at first glance appears to discard large outliers, does not in fact accomplish this goal in the same effective way as the optimal M-estimate. The computations and plots of Figure 6.56 are produced by Code 6.21.

```
n <- 20
probs <- (1:n - .5)/n
xn <- qnorm(probs)
x <- seq(-5,5,.1)
k <- length(x)
eif <- rep(0,k)
par(mfrow = c(2,2))
par(pty = "s")
for(i in 1:k) {
  eif[i] <- (n+1)*(mean(c(x[i],xn))-mean(xn))
}
```

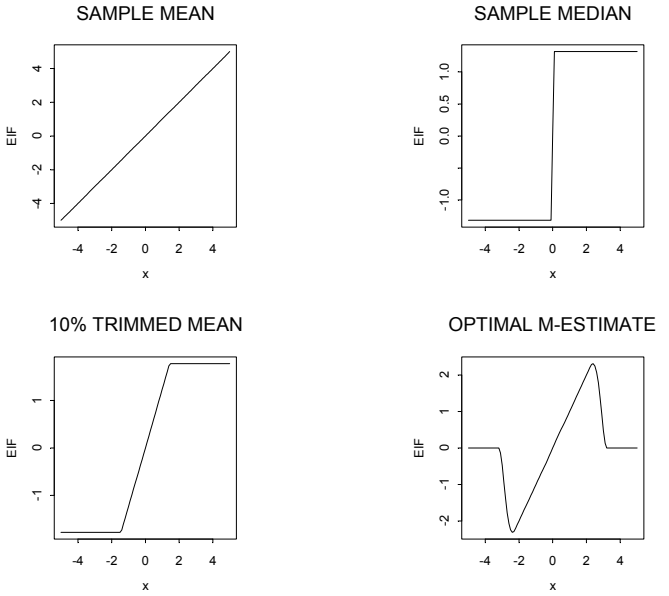

Figure 6.56 EIFs for Sample Mean, Median, Trimmed Mean, and Huber M-Estimate

```
plot(x,eif,type = "l",ylab = "EIF",
  main = "SAMPLE MEAN")
for(i in 1:k) {
  eif[i] <- (n+1)*(median(c(x[i],xn))-median(xn))
}
plot(x,eif,type = "l",ylab = "EIF",
  main = "SAMPLE MEDIAN")
for(i in 1:k) {
  eif[i] <- (n+1)*(mean(c(x[i],xn),trim=.1)-
     mean(xn,trim=.1))
}
plot(x,eif,type = "l",ylab = "EIF",
  main = "10% TRIMMED MEAN")
for(i in 1:k) {
  eif[i]=(n+1)*(coef(lmRob(c(x[i],xn)~1))-
     coef(lmRob(y~1)))
}
plot(x,eif,type = "l",ylab = "EIF",
  main = "OPTIMAL M-ESTIMATE")
```

Code 6.21 EIFs for Mean Returns Estimates

You can easily compute EIFs for the sample standard deviation volatility estimate by replacing the estimator function (e.g., `mean` in one of the "for" loops in Code 6.21 with the function `stdev`. You can do likewise for a robust scale estimate of volatility, for example, by replacing the function `mean` with the function `scale.tau`. The results, shown in Figure 6.57, show that (a) a single outlier has rapidly unbounded influence on the conventional standard deviation volatility estimate, and (b) outliers have only a bounded influence on the tau-scale estimate. With respect to the sample standard deviation EIF, note that when the additional data point is close to zero, which is the value of the sample mean for the prototype $\boldsymbol{x}$, the additional data point is an **inlier** that results in a negative value of the EIF because such an inlier decreases the value of the standard deviation. Note that the tau-scale volatility estimate EIF has a shape similar to that of the standard deviation in the central region from –2 to 2, except that small values of the added data point do not have so much negative influence except right at zero.

As the sample size tends towards infinity, the empirical influence function will (under regularity conditions) converge to the influence function defined as follows. It is assumed that the data are generated by a parametric distribution F_θ, where θ is the true parameter value. Let $\theta(F)$ be the asymptotic value of the parameter estimate $\hat{\theta}_n = \hat{\theta}_n(\boldsymbol{x})$ when the data have an arbitrary distribution function F, and note that typically $\theta(F) \neq \theta$ for an arbitrary F. It is also assumed that the parameter estimate is **consistent** (i.e., $\hat{\theta}_n$ converges to θ_o in probability), and that $\theta = \theta(F_\theta)$.[19] We represent the asymptotic version of the prototype sample $\boldsymbol{x}$ for an arbitrary distribution F and additional data point x by the mixture distribution

$$F_\gamma = (1-\gamma)\cdot F + \gamma \cdot \delta_x, \tag{6.24}$$

where γ is the mixture probability and δ_x is a point mass probability distribution located at x. The influence function $\mathrm{IF}(x) = \mathrm{IF}(x;\theta(F),F)$ is defined as

$$\mathrm{IF}(x) = \lim_{\gamma \downarrow 0} \frac{\theta(F_\gamma) - \theta(F_o)}{\gamma}. \tag{6.25}$$

Equivalently, $\mathrm{IF}(x)$ is the derivative of $\theta(F_\gamma)$ evaluated at $\gamma = 0$[20]:

$$\begin{aligned} \mathrm{IF}(x) &= \lim_{\gamma \downarrow 0} \frac{\theta(F_\gamma) - \theta(F_o)}{\gamma} \\ &= \frac{d}{d\gamma}\theta(F_\gamma)\Big|_{\gamma=0}. \end{aligned} \tag{6.26}$$

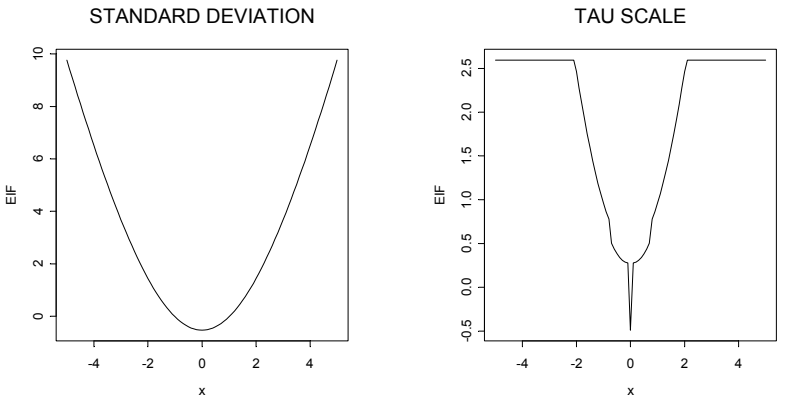

Figure 6.57 EIFs of Standard Deviation and Robust Tau-Scale Volatility Estimates

A very important property of the influence function is that it provides an approximate expression for the large sample bias

$$BIAS(x;\gamma) \triangleq \theta(F_\gamma) - \theta(F_\theta) = \theta(F_\gamma) - \theta$$

due to a small fraction γ of data located at x,

$$BIAS(x;\gamma) \approx \gamma \cdot \mathrm{IF}(x), \tag{6.27}$$

where the influence function $\mathrm{IF}(x)$ is evaluated at F_θ. There is evidence that for good robust estimators this local linear approximation of the bias is reasonably good for fractions γ as large as 5% to 10%, which covers many situations of importance in finance.[21]

An easy computation shows that the influence function of the sample mean estimate $\overline{x}$ is

$$IF(x;\overline{x}) = x - \mu, \tag{6.28}$$

where $\theta = \mu$ is the true mean value (Exercise 13). A slightly more involved computation shows that the influence function of an M-estimate $\hat{\mu}$ of location (see Equation 6.2) is

$$IF(x;\hat{\mu}) = \frac{\psi_{\mu,s}(x)}{E_F\left[\psi_{\mu,s}(x)\right]}, \tag{6.29}$$

where

$$\psi_{\mu,s}(x) = \psi\left(\frac{x-\mu}{s}\right) \tag{6.30}$$

and μ and s are the true location and scale parameters of the returns (Exercise 14). This influence function is the same as $\psi_{\mu,s}(x)$ except for the scale factor in the denominator. So, for the optimal bias-robust location M-estimate ψ function shown in the right-hand plot of Figure 6.6, the corresponding approximating EIF (shown in the lower-right-hand plot of Figure 6.56) differs from Figure 6.6 only by a scale factor and a small finite-sample approximation error.

6.11.2 Influence Functions for Sample Mean and Covariance Estimates of Returns

The definition of influence function extends in the obvious way to the case of multivariate data sets of returns and multidimensional parameter estimates. The functional representation of the sample mean estimate $\bar{\boldsymbol{x}} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_k)'$ is $\boldsymbol{\mu}(F) = \int \boldsymbol{x} dF(\boldsymbol{x})$. A straightforward calculation leads to (Exercise 15)

$$\mathrm{IF}(\boldsymbol{x}; \bar{\boldsymbol{x}}) = \boldsymbol{x} - \boldsymbol{\mu}. \tag{6.31}$$

For the sample covariance matrix estimate

$$\hat{\boldsymbol{\Omega}} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})', \tag{6.32}$$

one finds that the influence function is (Exercise 16)

$$\mathrm{IF}(\boldsymbol{x}; \hat{\boldsymbol{\Omega}}) = (\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})' - \boldsymbol{\Omega}. \tag{6.33}$$

For the case of a single asset where $k = 1$, the covariance matrix estimate becomes a sample variance, $\boldsymbol{\Omega} = \sigma^2$, and we get the influence function of the sample variance:

$$\mathrm{IF}(x; \hat{\sigma}^2) = (x - \mu)^2 - \sigma^2. \tag{6.34}$$

This shows that the influence of an outlier on the sample variance is quadratically unbounded, while that of an "inlier" at $x = \mu$ is $-\sigma^2$. It is easy to

see from its definition that the influence function of the sample standard deviation $\hat{\sigma} = \sqrt{\hat{\sigma}^2}$ is

$$\mathrm{IF}(x; \hat{\sigma}) = \frac{1}{2\sigma}\left((x-\mu)^2 - \sigma^2 \right),$$

which for a standard normal distribution is $.5 \cdot (x^2 - 1)$. In the left-hand plot of Figure 6.57, you see that the EIF for the sample standard deviation is a good approximation to $\mathrm{IF}(x; \hat{\sigma})$ for the standard normal distribution.

6.11.3 Influence Functions for Mean-Variance Optimal Tangency Portfolios

The Markowitz mean-variance efficient frontier for unconstrained portfolios is completely determined by the mean vector and covariance matrix of the returns. In order to estimate quantities on the efficient frontier, such as the mean return and risk of the global minimum variance and tangency portfolios and the maximum Sharpe ratio, one substitutes estimates of the mean vector and covariance matrix for their true values in the corresponding formulas. For the case of the tangency portfolio with return vector $\boldsymbol{\mu}$ and an excess return vector $\boldsymbol{\mu}_e = \boldsymbol{\mu} - \mathbf{1} \cdot r_f$, the weights vector, mean return, and variance estimates, respectively, are

$$\hat{\mathbf{w}}_T = \frac{\hat{\boldsymbol{\Omega}}^{-1}\hat{\boldsymbol{\mu}}_e}{\mathbf{1}'\hat{\boldsymbol{\Omega}}^{-1}\hat{\boldsymbol{\mu}}_e}, \tag{6.35}$$

$$\hat{\mu}_T = \hat{\boldsymbol{\mu}}_e'\hat{\mathbf{w}}_T = \frac{\hat{\boldsymbol{\mu}}_e'\hat{\boldsymbol{\Omega}}^{-1}\hat{\boldsymbol{\mu}}_e}{\mathbf{1}'\hat{\boldsymbol{\Omega}}^{-1}\hat{\boldsymbol{\mu}}_e}, \tag{6.36}$$

$$\hat{\sigma}_T^2 = \hat{\mathbf{w}}_T'\hat{\boldsymbol{\Omega}}\hat{\mathbf{w}}_T = \frac{\hat{\boldsymbol{\mu}}_e'\hat{\boldsymbol{\Omega}}^{-1}\hat{\boldsymbol{\mu}}_e}{\left(\mathbf{1}'\hat{\boldsymbol{\Omega}}^{-1}\hat{\boldsymbol{\mu}}_e\right)^2}. \tag{6.37}$$

The functional representation of the quantities above, needed for computing their influence functions, is obtained by replacing the mean and covariance matrix estimates by their functional representations $\boldsymbol{\mu}_e(\gamma) = \int \boldsymbol{x}\, dF_\gamma(\boldsymbol{x}) - \mathbf{1} \cdot r_f$ and $\boldsymbol{\Omega}(\gamma) = \int (\boldsymbol{x} - \boldsymbol{\mu}(\gamma))(\boldsymbol{x} - \boldsymbol{\mu}(\gamma))'\, dF_\gamma(\boldsymbol{x})$. This results in corresponding tangency portfolio functional representations for $\mathbf{w}_T(\gamma)$, $\mu_T(\gamma)$, and $\sigma_T^2(\gamma)$, from which one can compute influence functions. The influence function for the tangency portfolio weights is

$$\begin{aligned} \mathrm{IF}_{w_T}(\boldsymbol{x}) &= \frac{d}{d\gamma}\frac{\boldsymbol{\Omega}^{-1}(\gamma)\boldsymbol{\mu}_e(\gamma)}{\mathbf{1}'\boldsymbol{\Omega}^{-1}(\gamma)\boldsymbol{\mu}_e(\gamma)}\bigg|_{\gamma=0} \\ &= \frac{\mathbf{1}'\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e - \boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e\mathbf{1}'}{(\mathbf{1}'\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e)^2}\cdot\left[-\boldsymbol{\Omega}^{-1}\mathrm{IF}_{\Omega}(\boldsymbol{x})\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e + \boldsymbol{\Omega}^{-1}\mathrm{IF}_{\mu}(\boldsymbol{x})\right], \end{aligned} \tag{6.38}$$

as can be verified by careful calculation (Exercise 17). Since the influence function $\mathrm{IF}_{\Omega}(\boldsymbol{x})$ for the covariance matrix is quadratically unbounded in $\boldsymbol{x}$ (i.e., the value increases quadratically with size $\boldsymbol{x}$), the same is true of the influence function for the weights given above. Since $\mu_T(\gamma) = \boldsymbol{\mu}_e'(\gamma)\cdot\mathbf{w}_T(\gamma)$, one gets (Exercise 18)

$$\mathrm{IF}_{\mu_T}(\boldsymbol{x}) = \boldsymbol{\mu}_e'\cdot\mathrm{IF}_{w_T}(\boldsymbol{x}) + (\boldsymbol{x}-\boldsymbol{\mu})'\cdot\mathbf{w}_T. \tag{6.39}$$

One can also show that (Martin and Zhang, 2004)

$$\begin{aligned} \mathrm{IF}_{\sigma_T}(\boldsymbol{x}) &= \tfrac{1}{2\sigma_T}\mathrm{IF}_{\sigma_T^2}(\boldsymbol{x}). \\ &= \tfrac{1}{2\sigma_T}\left[2\cdot\mathrm{IF}_{w_T}'(\boldsymbol{x})\cdot\boldsymbol{\Omega}\cdot\mathbf{w}_T + \mathbf{w}_T'\cdot\mathrm{IF}_{\Omega}(\boldsymbol{x})\cdot\mathbf{w}_T\right] \end{aligned} \tag{6.40}$$

Notice that when the additional data value $\boldsymbol{x}$ is located at the mean return vector $\boldsymbol{\mu}$, (i.e., $\boldsymbol{x} = \boldsymbol{\mu}$), the influence function for the weight vector is

$$\begin{aligned} \mathrm{IF}_{w_T}(\boldsymbol{\mu}) &= \frac{\mathbf{1}'\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e - \boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e\mathbf{1}'}{(\mathbf{1}'\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e)^2}\cdot\left[-\boldsymbol{\Omega}^{-1}(-\boldsymbol{\Omega})\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e + \mathbf{0}\right] \\ &= \frac{\mathbf{1}'\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e - \boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e\mathbf{1}'\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e}{(\mathbf{1}'\boldsymbol{\Omega}^{-1}\boldsymbol{\mu}_e)^2} \\ &= \mathbf{0}. \end{aligned} \tag{6.41}$$

One can also show that $\mathrm{IF}_{\mu_T}(\boldsymbol{x}) = 0$ and $\mathrm{IF}_{\sigma_T}(\boldsymbol{x}) = -\frac{1}{2}\sigma_T$. These results are intuitively appealing in that when a data value $\boldsymbol{x}$ is located at the mean returns vector $\boldsymbol{\mu}$, it is reasonable that it have no perturbing influence on the portfolio weights or mean return, and the negative influence on the portfolio risk is consistent with that of the influence of an inlier on a simple standard deviation. Some other interesting results on portfolio influence functions may be found in Martin and Zhang (2004).

We test the use of the influence formulas above on a very simple case where we know the tangency portfolio solution immediately and can interpret the influence function results most easily. Suppose we have 60 monthly observations of returns on two stocks with equal mean returns, equal volatilities, and a diagonal covariance matrix constructed as follows:

```
> stock.names <- c("STOCK.1", "STOCK.2")
> mutest <- c(0.01, 0.01)
> names(mu.test) <- stock.names
> Vtest <- diag(c(0.003, 0.003))
> dimnames(Vtest) <- list(stock.names, stock.names)
> mutest
[1] 0.01 0.01
> Vtest
        STOCK.1 STOCK.2
STOCK.1   0.003   0.000
STOCK.2   0.000   0.003
```

Our influence function code (Code 6.22) incorporates a function `port.tan` for computing the tangency portfolio:

```
port.tan <- function(V, mu, rf)
{
  p <- length(mu)
  one <- rep(1, p)
  mue <- mu - rf
  a <- solve(V, mue)
  Vinv <- solve(V)
  d <- as.numeric(inprod(one, a))
  wts <- a/d
  n <- as.numeric(qform(mue, Vinv))
  muep <- n/d
  sigma <- n^0.5/abs(d)
  sr <- sign(d) * n^0.5
  list(Weights = wts, Mu.e = muep, Sigma = sigma,
     "Sharpe Ratio" = sr)
}
```

Code 6.22 Unconstrained Tangency Portfolio

Code 6.23 gives three simple functions for computing an inner product and quadratic form in `port.tan`, as well as an outer product function needed for our influence function calculations:

```
inprod <- function(x, y) {
  as.numeric(t(matrix(x)) %*% matrix(y))
}

qform <- function(x, A) {
  x = matrix(x)
```

```
  as.numeric(t(x) %*% A %*% x)
}

outprod = function(x, y) {
  matrix(x) %*% t(matrix(y))
}
```

Code 6.23 Inner Product, Outer Product, and Quadratic Form Functions

Now we compute the influence function value for a data point $\boldsymbol{x} = (.065, -.065)$ on the tangency portfolio weights, mean return, risk (standard deviation), and Sharpe ratio, assuming a risk-free rate of .004 and using $\gamma = 1/61$ in the bias approximation $BIAS(x;\gamma) \approx \gamma \cdot \mathrm{IF}(x)$:

```
rf <- .004
Gamma <- 1/61
x <- c(.0648,-.0448)

if.tan(x, Vtest, mutest, rf, print.results = T,
  Gamma, IF.relative = T)

* TANGENCY PORTFOLIO WEIGHTS AND PERFORMANCE *

     WT1     WT2  MUE   SIGMA SHARPE
     0.5     0.5 0.006 0.0387 0.1549

* INFLUENCE FUNCTION OF TANGENCY PORTFOLIO *

[1] "GAMMA = 0.016"

        X1     X2  WT1   WT2 MUE  SIGMA SHARPE
[1,] 1.001 -1.001 0.15 -0.15   0 -0.008  0.008
```

The tangency portfolio weights `(WT1, WT2)`, excess mean return `(MUE)`, risk `(SIGMA)`, and Sharpe ratio `(SHARPE)` provided as the first output above are exactly as one expects. In the last line of output the `X1` and `X2` are standardized versions of $\boldsymbol{x} = (.065, -.065)$ (i.e., they represent one standard deviations of the added data from the mean returns). Because we used the optional argument `IF.relative = T`, the influence function values `WT1`, `WT2`, `MUE`, `SIGMA`, `SHARPE` are all relative to the true tangency portfolio values. For example, the `WT1` value of `.15` represents an increase in the weight of 15%, etc.

Code 6.24 gives the S-PLUS code for the calculation above.

```
if.tan = function(x, V, mu, rf, print.results = T,
  Gamma = 1, IF.relative = F)
{
  # Compute tangency portfolio for IF.relative
  # calculations
  tanport <- port.tan(V, mu, rf)
  wts.tan <- tanport$Weights
  mu.tan <- tanport$Mu
  sigma.tan <- tanport$Sigma
  sr.tan <- tanport$"Sharpe Ratio"
  opt.tan <- c(wts.tan, mu.tan, sigma.tan, sr.tan)

  p <- length(mu)
  # Optionally print tangency portfolio
  if(print.results) {
     names(opt.tan) = c(paste("WT", 1:p, sep = ""),
        "MUE","SIGMA", "SHARPE")
        cat("\n* TANGENCY PORTFOLIO WEIGHTS AND
        PERFORMANCE *\n\n")
     print(round(opt.tan, 4)); cat("\n")
  }
  # Compute excess returns, vector of 1's,
  # Vinverse*mu.e, Vinverse
  mu.e <- mu - rf
  one <- rep(1, p)
  xcent <- x - mu
  a <- solve(V, mu.e)
  Vinv <- solve(V)
  # Compute influence functions
  IF.cov <- outprod(xcent, xcent) - V
  A <- Vinv %*% matrix(xcent) -
     Vinv %*% IF.cov %*% a
  B <- 1/inprod(one, a)
  IF.wts <- B * A - (B^2) * a * inprod(one, A)
  IF.mu <- inprod(mu.e, IF.wts) +
     inprod(xcent,wts.tan) #IF mu.e
  IF.sigma <- (2*t(wts.tan) %*% V %*% IF.wts +
     qform(wts.tan,IF.cov))/(2*sigma.tan)
  IF.sr <- (sr.tan^2 - 1 +
     (inprod(a,xcent)-1)^2)/(2*sr.tan)
  IF <- Gamma * c(t(IF.wts), IF.mu, IF.sigma,
     IF.sr)
  if(IF.relative) {
     IF <- IF/opt.tan
```

```
      digits <- 3
      x <- xcent/as.vector(diag(V)^0.5)
   }
   else {
      digits <- 4
   }
   IF <- t(as.matrix(c(x, IF)))
   dimnames(IF)[[2]] <- c(paste("X", 1:p, sep = ""),
      paste("WT", 1:p, sep = ""), "MUE", "SIGMA",
      "SHARPE")

   if(print.results) {
      cat("\n* INFLUENCE FUNCTION OF TANGENCY
         PORTFOLIO *\n\n")
      cat(paste("GAMMA =", round(Gamma,3)))
      cat("\n\n")
   }
   round(IF, digits)
}
```

Code 6.24 Influence Function of Tangency Portfolio

Now we can use the function `if.tan` in a function `if.comp2`, given in Code 6.25 to compute contour plots of the tangency portfolio influence function for weights, mean return, and risk for a range of values of $\boldsymbol{x} = (x_1, x_2)$. The resulting influence function contours shown in Figure 6.58 for the tangency portfolio weight w_1 show that when X1 = X2 there is no influence bias on the weights, while X1 > X2 results in positive bias and X2 > X1 results in negative bias.

Figure 6.59 and Figure 6.60 show corresponding results for the tangency portfolio mean return and standard deviation.

Figure 6.59 shows that when X1+X2 > 0, the influence bias in mean return is positive, and when X1 + X2 < 0, this bias is negative. Figure 6.60 shows that the standard deviation of the tangency portfolio increases with increasing values of |X1+X2|.

The results in Figure 6.58, Figure 6.59, and Figure 6.60 may seem intuitively reasonable based on the very simple structure of the assumed mean vector and covariance matrix. In any event, by careful examination of the expressions for the influence functions for the weight vectors, mean return, and standard deviation, one can easily check that the results are qualitatively correct (Exercise 19).

Here is a short bit of code that uses the function `if.comp2` in Code 6.25 to carry out the computation above and produce the plot of Figure 6.58:

```
rf <- .004
Gamma <- 1/61
if.comp2(Vtest, mutest, rf, plotchoice = "if.wts",
  k=3, nsteps=9, Gamma,IF.relative = T)
```

To get the plots of Figure 6.59 and Figure 6.60, use the optional arguments `plotchoice = "if.mu"` and `plotchoice = "if.sigma"`, respectively.

```
if.comp2 <- function(V, mu, rf, plotchoice =
     "if.wts", k=5, nsteps=3, Gamma = 1,
     IF.relative = T)
{
  sigma <- diag(V)^0.5
  rng1 <- c(mu[1]-k*sigma[1], mu[1]+k*sigma[1])
  rng2 <- c(mu[2]-k*sigma[2], mu[2]+k*sigma[2])
  x1 <- rep(seq(rng1[1], rng1[2], length = nsteps),
     times = nsteps)
  x2 <- rep(seq(rng2[1], rng2[2], length = nsteps),
     times = rep(nsteps, times = nsteps))
```

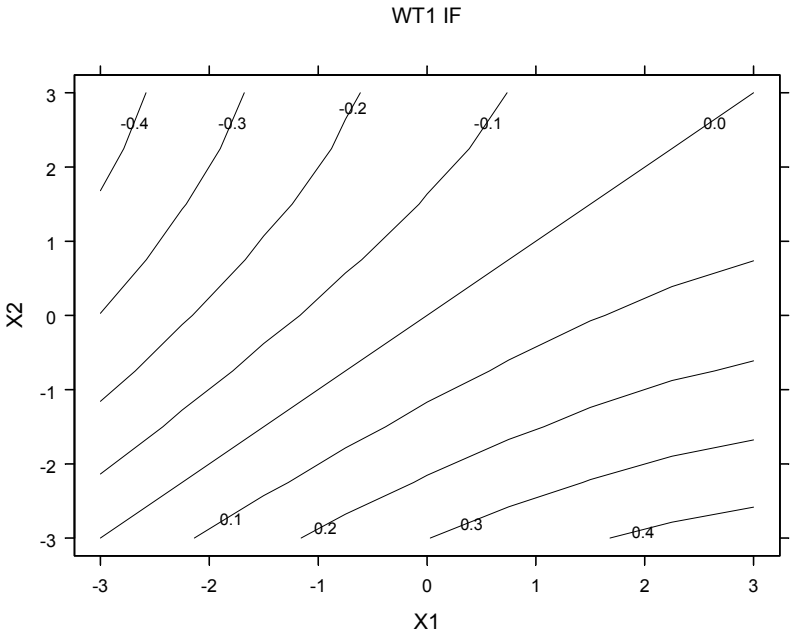

Figure 6.58 Influence Function Contours for Tangency Portfolio Weights

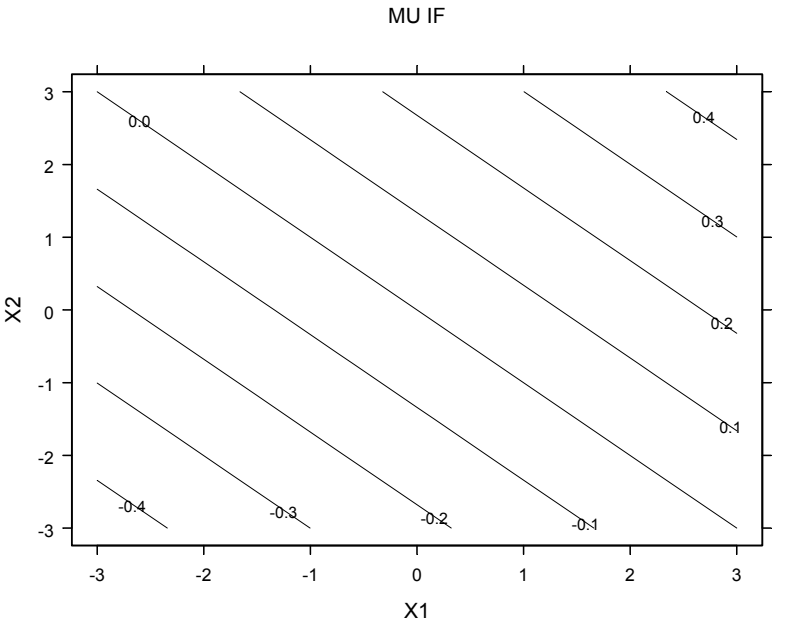

Figure 6.59 Influence Function Contours for Tangency Portfolio Mean Return

```
x <- cbind(x1, x2)
IF <- data.frame(matrix(rep(0, 7 * nsteps^2),
   ncol = 7))

for(i in 1:(nsteps^2)) {
   IF[i,  ] <- if.tan(x[i,  ], V, mu, rf,
      print.results = F, Gamma, IF.relative)
}

names(IF) <- dimnames(if.tan(x[1, ],V,mu,rf,
   print.results = F))[[2]]

if(plotchoice == "if.wts") {
   contourplot(WT1 ~ X1*X2, data = IF,
      main = "WT1 IF")
}
else if(plotchoice == "if.mu") {
   contourplot(MUE ~ X1*X2, data = IF,
      main = "MU IF")
}
else if(plotchoice == "if.sigma") [
```

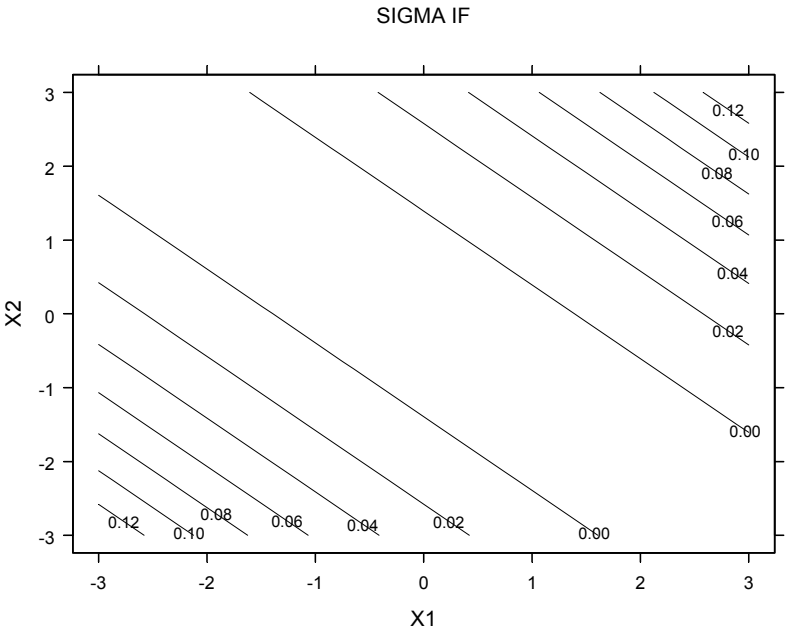

Figure 6.60 Influence Function Contours for Tangency Portfolio Risk

```
    contourplot(SIGMA ~ X1*X2, data = IF,
       main = "SIGMA IF")
  }
}
```

Code 6.25 IF Plots for Tangency Portfolio

We recommend that the reader experiment with the tangency portfolio influence functions above using a more realistic set of mean returns and covariance matrix that may arise in practice. For example, you might use the following monthly mean returns, covariance matrix, and volatilities (Exercise 20)[22]:

```
> mu3
  SP500 GOV.BOND SMALL.CAP
 0.0101   0.0043    0.0137

> V3
             SP500 GOV.BOND SMALL.CAP
    SP500 0.00325  0.00023   0.00420
 GOV.BOND 0.00023  0.00050   0.00019
SMALL.CAP 0.00420  0.00019   0.00764
```

```
> round(sqrt(diag(V3)),4)
[1] 0.0570 0.0224 0.0874
```

You can get the correlation matrix from the covariance matrix with the function `cov.to.corr` in Code 6.26.

```
cov.to.corr <- function(v) {
  dimnames <- dimnames(v)
  sigma <- diag(v)^0.5
  s.inv <- diag(1/sigma)
  rho <- s.inv %*% v %*% s.inv
  dimnames(rho) <- dimnames
  rho
}
```

Code 6.26 Convert Covariance Matrix to Correlation Matrix

```
> round(cov.to.corr(V3),4)
           SP500 GOV.BOND SMALL.CAP
    SP500 1.0000   0.1804    0.8429
 GOV.BOND 0.1804   1.0000    0.0972
SMALL.CAP 0.8429   0.0972    1.0000
```

You can compute the tangency influence function for all three of these assets with `if.tan`, but when using `if.comp` you need to work with two at a time. Note that if you use SP500 and SMALL.CAP, you have a high correlation, but if you use GOV.BOND and SMALL.CAP, you have a small correlation. Do not be surprised if you find that the influence of additional outlier data is greater in the case of high correlation between assets.

6.11.4 Influence of Outliers on the Sharpe Ratio

It turns out that the influence function for the Sharpe ratio of the tangency portfolio has the simple form (see Martin and Zhang, 2004)

$$\mathrm{IF}_{SR}(\boldsymbol{x}) = \frac{1}{2SR}\left(SR^2 + 1 - (y-1)^2 \right), \tag{6.42}$$

where

$$y = \boldsymbol{\mu}_e \boldsymbol{\Omega}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) \tag{6.43}$$

Thus, somewhat surprisingly, the value of $IF_{SR}(\boldsymbol{x})$ is bounded above by $\left(SR^2+1\right)/2SR$ (i.e., an outlier can cause at most a bounded positive bias in the maximum Sharpe ratio). But since y can be made arbitrarily large by making the size of $\boldsymbol{x}$ arbitrarily large, an outlier can cause an arbitrarily large negative bias of the Sharpe ratio. This represents a fundamental kind of asymmetry in the potential influence of outliers on the maximum Sharpe ratio. The reader is encouraged to explore the influence of outliers on the maximum Sharpe ratio by appropriately modifying the function `if.comp2`, first in such a way as to explore the influence of outliers in one coordinate direction at a time (i.e., in one set of returns at a time).

6.11.5 Empirical Influence Functions for Unconstrained and Constrained Portfolios

It is a straightforward matter to compute influence functions for weights, mean return, and risk of other unconstrained portfolios (such as the global minimum variance portfolio). Since the method is an infinitesimal one, providing a valid approximation for small fractions of influential data, it could in principle be applied to a portfolio optimized under constraints, provided none of the constraints are binding and the influence of an outlier does not cause some of the constraints to become binding. The obvious first approach to computing influence functions for constrained portfolios (e.g., long-only portfolios, sector constraints, etc.) is to compute empirical influence functions (EIFs) along the lines of the calculations leading to Figure 6.56. While this will be computationally burdensome since one has to solve a QP or LP problem for each outlier data position, it can no doubt be done. A first step would be to compute the empirical influence function for the tangency portfolio quantities in the unconstrained case as a check on its accuracy. Our earlier results with EIFs for simple location estimates were quite encouraging, but since the tangency portfolio estimates are much more complicated, this initial check will be useful. A deeper study of the QP optimization structure might lead to some efficient methods of computing influence functions or EIFs for optimal portfolios under constraints. This is a topic for further research.

Exercises

1. Use the `plot` function in a `for` loop to plot time series of returns for each microcap stock in `microcap.ts` so that you can get an overall visual grasp of the behavior of each stock in this microcap group with respect to outliers and time-varying volatility. Use the `plot` function with arguments as in Code 6.1, except use a generic first argument `returns`, setting `returns` equal to one of the stock's returns series for each cycle of the `for` loop. Use the graphics command `par(mfrow(n1,n2))` to automate plotting a number of time series of returns per page for each of the market cap groups. Use the S-PLUS Windows toolbar menu choice "Options > Graph Options" and select "Every Graph" from the Auto Pages drop-down list in the "Traditional Graphics" region of the dialog, as this will result in each new page of plots appearing as a separate page of the Graph Sheet. Make similar plots for each of the groups `smallcap.ts`, `midcap.ts`, and `largecap.ts`.

2. Make Q-Q plots of returns for a few time series from each of the four market cap groups, both with and without 95% simulation envelopes. Automate making Q-Q plots for all stock returns in the four market cap groups in a manner similar to the way you plotted all the time series in Problem 1.

3. Compute classical and robust means and standard deviations of returns for all the stocks in the microcap group, and plot the means versus standard deviations for these estimates. Use one plotting symbol for the classical estimates and another plotting symbol for the robust estimates. Add the ticker symbols as text labels. (Warning: with a lot of stocks there may be confusing overlap of these text labels.)

4. Use the classical and robust EWMA volatility estimate and UMT functions (Code 6.4 and Code 6.5) on a few stocks from each of the four market cap categories. What do you conclude about the prevalence of outliers and overestimation of volatility following isolated outliers? What do you conclude about the potential usefulness of a robust UMT?

5. Modify the UMT Code 6.4 to use $\hat{\sigma}_{t-1}$ in place of $\hat{\sigma}_t$ in the test statistic and evaluate the improvement in detecting initial outlier returns (unusual price movements). Do you think the modified method is adequate for detecting a returns outlier that occurs shortly after another returns outlier?

6. Convert Code 6.6 into an S-PLUS function that makes an array of plots (like that of Figure 6.17) of the input returns series, overlays the LS and robust regression lines, and places the legend and annotations automatically. Then run the function on a few of the stock returns in each of the time series data sets `microcap.ts`, `smallcap.ts`, `midcap.ts`, `largecap.ts`. You might want to do this with a `for` loop as in Problem 1, in which case you can do it for all twenty stock returns in each of the market-cap groups. (Alternatively, the Trellis graphics functions in S-Plus, if you are familiar with them, provide a clean way to do this.) For which stock returns do the least squares and robust betas differ significantly because of the presence of outliers?

7. Alphas are of considerable interest to investors because they represent excess returns obtainable from investing in a given stock over and above what is predicted by the Capital Asset Pricing Model (CAPM). Modify the code in Exercise 6 so that you obtain the least squares and robust alphas and their standard errors. For what firms do the least squares and robust alphas differ significantly? What do you conclude about the usefulness of robust alphas?

8. Explore small subsets of four to six multivariate stock returns in one of the time series data sets `microcap.ts`, `smallcap.ts`, `midcap.ts`, `largecap.ts`, and `midcapD.ts` by making pairwise scatterplots and classical versus robust correlations to find a subset that exhibits one or more substantial differences between the classic and robust correlations. Explain why the substantial difference or differences between classical and robust correlations are reasonable given the nature of the data. For such a subset, compute and display classical and robust Mahalanobis distances, and comment on any interesting aspects of the multidimensional outliers found.

9. Use the multivariate returns data set of Exercise 8 for this exercise. Compute and display classical and robust mean-variance efficient frontiers, and discuss how you would use the results to guide a portfolio selection investment decision.

10. Use the multivariate returns data of Exercises 8 and 9 or other similarly interesting multivariate returns data for this problem. Compute and display bootstrapped classical and robust efficient frontiers and boxplots of paired differences in classical and robust Sharpe ratios. What is the effect of changing the number of bootstrap samples? How many bootstrap samples appear to be adequate to you?

11. Compute and display the portfolio weights for the three efficient frontiers in Figure 6.55. What do you find? Do the results make sense?

12. Use the multivariate returns data of Exercises 8 and 9 or other similarly interesting multivariate returns data for this problem. For alpha equal to .05, compute robust, optimistic, and pessimistic CVaR optimal portfolios and display the results of each along with the classical CVaR optimal portfolio (also using an alpha value of .05). Discuss how you would use the results to guide a portfolio selection investment decision. Do likewise for alpha values of .1, .2, and .5.

13. Derive the expression for the influence function of the sample mean of a single-asset return.

14. Derive the expression for the influence function of a location M-estimate.

15. As a slight extension to Exercise 13, derive the expression for the influence function of a sample mean of multivariate returns.

16. Derive the expression for the influence function of the sample covariance matrix.

17. Verify the expression for $\mathrm{IF}_{w_T}(\boldsymbol{x})$.

18. Verify the expression for $\mathrm{IF}_{\mu_T}(\boldsymbol{x})$.

19. Verify that the results in Figure 6.58, Figure 6.59, and Figure 6.60 are qualitatively correct.

20. Compute and display tangency portfolio influence functions assuming that the mean returns vector is `mu3` and the covariance matrix is `V3`. Explain why the results are reasonable.

Endnotes

[1] This is true even when adjusting for time-varying volatility with a GARCH model. See, for example, calculations by Menn (2003).

[2] If you wanted to get a data frame object with a single variable rather than a vector object (the default simple data object in S-PLUS), you would drop the subscript part "`[,1]`" in the following two commands. Some functions in S-PLUS will work on S-PLUS V4 time series objects directly (e.g., `mean` and `var` will do so, though with slightly different output formats, but `stdev` will not). Many functions that do not work on S-PLUS V4 time series objects will work on the data frame component of a time series object that you would obtain as described above. Unfortunately, some functions, such as `stdev`, will not even work on a data frame but will work on a vector object, which is why we elected to extract the data from `returns.ts` as a vector in this example.

[3] The functions `qqnorm` and `qqline` are other examples of functions that do not work on data frames.

[4] See the online manual for the Robust Library in S-PLUS 6 for further details on the Q-Q plot simulation envelopes. See also Atkinson (1985).

[5] We remark that when there is a lot of data (e.g., a few hundred observations), one may be able to fit a heavy-tailed distribution to asset returns with a reasonably high degree of accuracy (see, for example, Rachev and Mittnik, 2000). Then although one cannot predict just when a future outlier will occur, one can be certain that a certain number will occur on average over a certain time interval, and this is can be very useful in the context of risk management.

[6] M-estimators are generalizations of maximum likelihood estimators introduced by Huber (1964) for estimates of location and by Huber (1973) for regression. See also Huber (2004) and Hampel et al. (1986)

[7] It should be noted that the weights in this case are data-dependent, which means that this weighted least squares equation is nonlinear.

[8] This is similar to the fact that the classical *t*-test lacks robustness of power toward heavy-tailed deviations from normality.

[9] Capital Asset Pricing Model.

[10] This general form of shrinkage estimator was justified using a Bayesian argument by Vasicek (1973) and by Blume (1971) using an argument of regression toward the mean.

[11] See the "Current Commercial Practice" section of Martin and Simin (2003) for further details.

[12] The horizontal dashed lines are located at ± 2.5 times the scaled median absolute deviation about the median (MADM) robust scale estimate, which is an approximately unbiased estimate of the standard deviation when the returns are normally distributed.

[13] The modified function `seriesPlot`, and a modified function `panel.superpose.ts` that is called by `seriesPlot`, are provided in the code archive for this book (see the Preface).

[14] The script `multi.start.function.ssc`, written by Heiko Bailer and included in the code archive for this book (see the Preface), implements the classical and robust versions of the Stambaugh method.

[15] We note that there is nothing wrong with using a parametric bootstrap so long as the parametric model is adequate. The problem with the Jorion and Michaud approach is the evaluation of the performance of their resampled portfolios using the original sample mean and covariance rather than the resampled means and covariances.

[16] See also Bradley and Taqqu (2003).

[17] For a thorough introduction to influence functions in the context of robust statistics, see Hampel et al. (1986), which discusses all the key properties of influence functions. Here we focus primarily on the influence of outliers and the approximate bias caused by outliers.

[18] This is one of several possible definitions of a finite sample influence function. See, for example, Mallows (1975).

[19] The latter condition is called Fisher consistency in the statistical literature. See, for example, Huber (2004) or Hampel et al. (1986).

[20] This is a directional or Gateaux derivative of the functional $\theta(F)$ at F_o in the "direction" F_γ.

[21] See, for example, Hampel et al. (1986) and the maximum bias curves in Martin, Yohai, and Zamar (1989).

[22] These are the values used in Rockafellar and Uryasev (2000) but have been slightly rounded.