# Effective Portfolio Optimization

APPLICATION NOTE


Insightful

# Table of Contents

# Effective Portfolio Optimization

***Frank Block***

## Abstract

This Application Note shows you how to effectively construct an efficient portfolio using Insightful software. This example uses a small sample of stocks and determines the weight of each stock required to obtain specific risk-return characteristics.

## Introduction

In the past 25 years, there has been a revolution in the use of quantitative models within the financial world. This has been due to a number of important breakthroughs in academic theory, such as the modern portfolio theory of Markowitz and the option pricing theory of Merton, Black & Scholes, as well as the enormous increase in computing power and the availability of advanced analytical tools to implement these theories.

*The portfolio optimization problem requires maximizing the return for a given risk target or minimizing the risk given a target return.*

Today's markets are extremely demanding for all persons involved in asset management. The sheer quantity and variety of available investment tools (ranging from bonds and futures to derivatives, hedge funds, funds of funds, etc.) makes the task of optimizing the performance of invested assets while controlling risk a significant challenge. Advanced techniques are required to analyze market data, select instruments forming a portfolio with appropriate risk-return characteristics, and communicate the results to analysts and investors.

The portfolio optimization problem can be summarized as follows: Given the universe of assets under consideration, how much of each asset should we hold in the optimal portfolio so as to achieve the highest rate of return with a tolerance for certain risk, or to achieve a minimal risk given a desired rate of return?

The input to a portfolio optimization problem consists of the estimates of the expected returns of all assets in the portfolio, and the estimate of the covariance matrix of the returns. The output is simply the weight vector of the optimal portfolio.

In practice, we normally do not observe the returns of financial assets directly. Instead, we observe the prices of traded financial assets, and returns must be calculated from the price observations. In addition, the price observations usually have missing values due to market closures, and/or outliers due to abnormal market movement or data recording errors. Therefore, the pre-processing step of treating missing values and detecting outliers is usually necessary.

*This note describes the construction of optimal stock-only portfolios based on S&P500 index constituents.*

For simplicity, let's consider the universe as composed of Standard & Poor's 500 index constituents. This restricts us to a stock-only portfolio, but in practice, a portfolio can also include bonds, currencies, derivatives, etc. The input for the optimization consists of the vector of mean returns and the covariance matrix of the returns on the 500 stocks.

In simple cases, closed form solutions can be found to solve this problem. In practice, many other constraints are imposed, and we have to resort to numerical algorithms for finding the optimal portfolio, which solves either of the above optimization problems.

The example used in this example covers the following steps:

- Reading and transforming historical stock price data
- Stock screening
- Computing and exploring return data
- Computing efficient frontier and optimal portfolios

We take a completely new approach to solve this problem by using Insightful finance products: Insightful Miner, S-PLUS, S+FinMetrics and S+NuOPT. This enables us to integrate financial data with cutting-edge techniques for exploration and analysis. The vast range of available graphical and modeling capabilities speeds exploring universes of tens of thousands investment instruments, understanding causes and effects, and generating hypotheses about future behavior. Based on the comprehensive set of cutting-edge statistical and financial analysis functions, predictive models and large portfolio optimizations can be easily run and assessed with back-testing techniques. Insightful products also provide an easy-to-use visual programming environment, making the process of transforming and integrating multiple data sources a simple task for very large data sets.

*A new approach is shown for solving the portfolio optimization problem based on the integrated visual and scalable programming environment consisting of Insightful Miner, S-PLUS, S+FinMetrics, and S+NuOPT.*

Through Insightful Miner, the combined data analysis capabilities of all Insightful finance products can be accessed. This note shows the intuitive ease and flexibility of Insightful Miner's visual programming paradigm, which makes complex data analysis applications easier to understand and to share with others. Using Insightful Miner's highly scalable architecture, many large-data problems can be handled with ease. When the example in this note is complete, we obtain a workflow that looks like the following graphic.

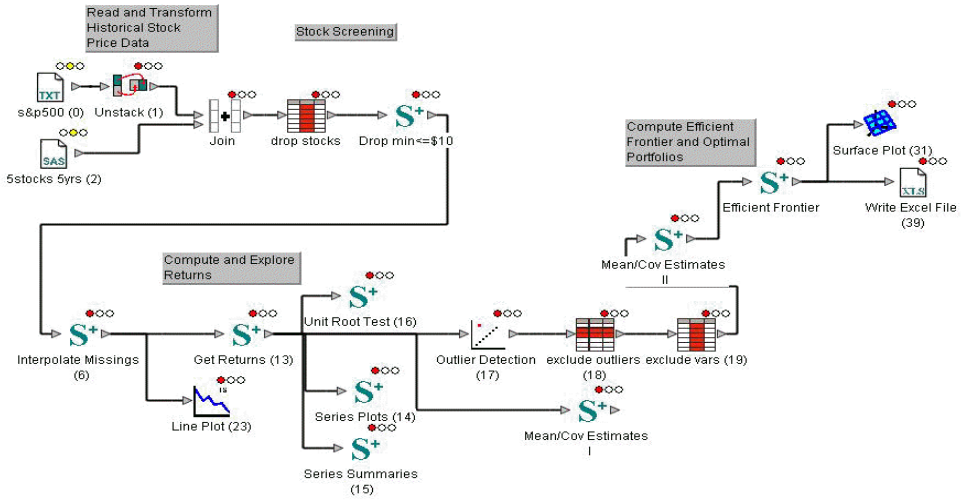



Figure 1

## Step 1: Reading and Transforming Historical Stock Price Data

In this example, we read historical stock price data from two data sources and join them into one single data set which will be used for subsequent analysis. After reading the data in, some data transformation is required to get us there, annotated in the following steps.Read stock price data from two different sources:

- **sp500hst.txt**: a text file with price data on the S&P 500 companies from May 10, 2002 until May 9, 2003. The data is comma separated.
- **5techstocks.sas7bdat**: a SAS file with price data on six more technology stocks (AMAT, CSCO, IBM, DELL, and INTC) from May 11, 1998 until May 9, 2003. The data contains the date and the closing price at each day for these stocks.

1. Transform the data into a format having one date column and one price column for each stock considered.
2. Join the data from two sources into a single data set.

*With Insightful Miner's powerful data reading and transformation capabilities, it is easy to read and join data from various sources and to transform raw data into formats suited for analysis.*

Open up a new Insightful Miner worksheet, and double-click a **Read Text File** node to move it into the empty worksheet. Click the **Browse** button and navigate to **sp500hst.txt**. Set the rest of the dialog parameters to match the screen shot in Figure 2, and then press the **Update Preview** button.

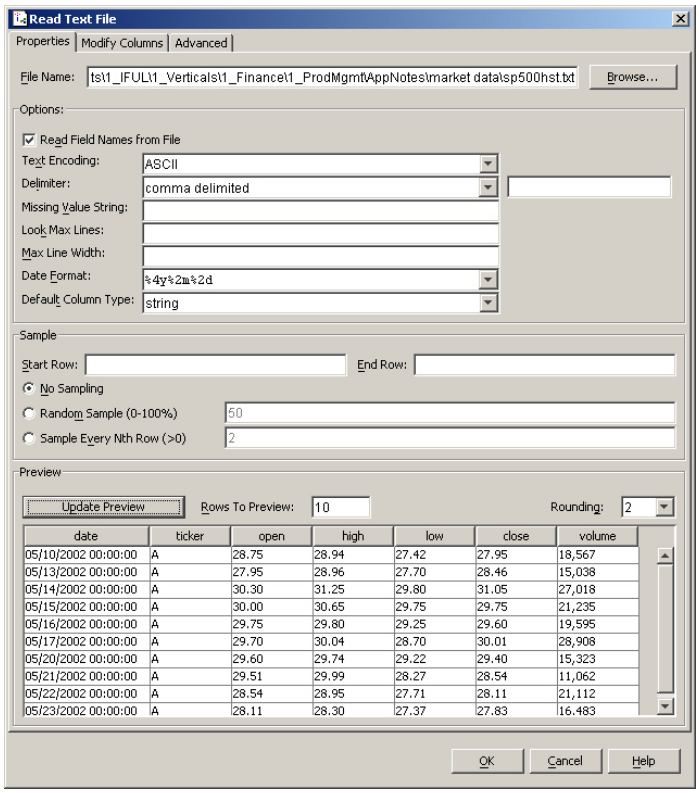



Figure 2

Click the **Modify Columns** tab and select the **date**, **ticker**, and **close** columns to select them, since those are the only variables we're interested in. Set **ticker** as a categorical by highlighting the **ticker** column and clicking the **Categorical** button under **New Types**, as shown in Figure 3.[1]


Figure 3

Run the node by right-clicking it and selecting **Run to Here**. Check the data was correctly read by right-clicking the node and selecting **Viewer**. The summary statistics are shown in Figure 4.

[1] Since there are more than 500 values of the ticker symbol in the data that are going to be read, we set the number of maximum levels of categorical variables to 600. You can set this by going to **File** | **Properties** and selecting the **Advanced** tab.

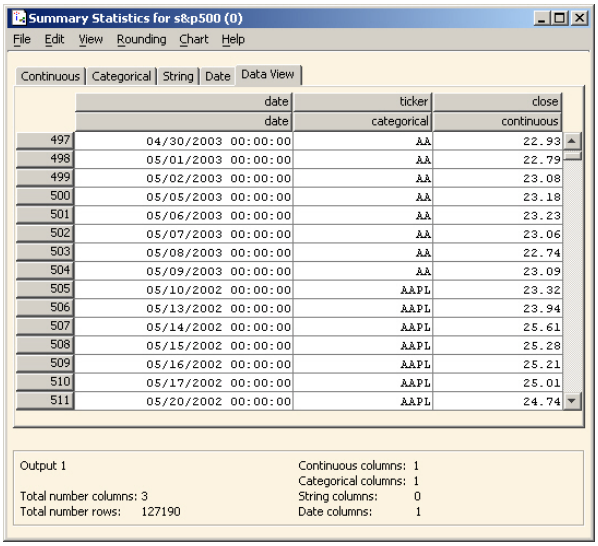



Figure 4

Let's transform this data into a different format, with a date column and one price column for each ticker symbol contained in this file. Double-click an **Unstack** node to move it to the worksheet, and connect it to the **Read Text File** node.

*With Insightful Miner's **Unstack** node, data can easily be transformed from a format with row-wise stacked stock price time series to a format with one stock price time series per column.*

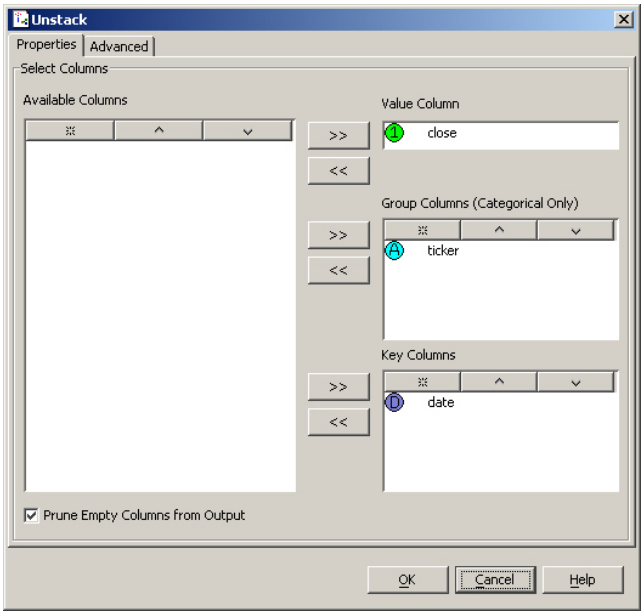
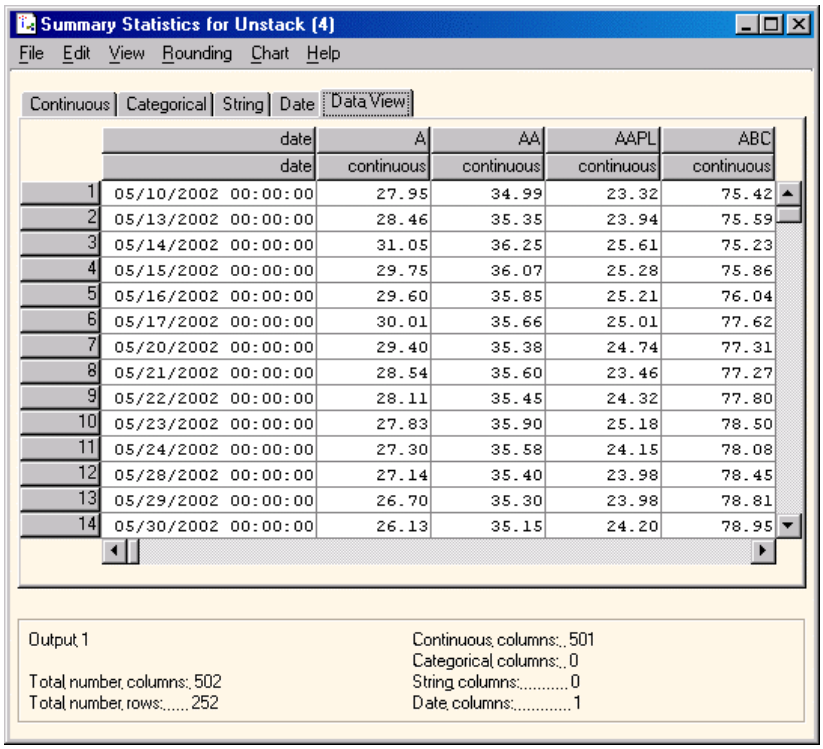



Figure 5

Select the columns from **Available Columns** and move them to the appropriate groupbox on the right, as shown in
Figure 5. These settings determine how the unstack transformation works.

Run the **Unstack** node by right-clicking it and selecting **Run to Here**. When it has completed, right-click the node and select **Viewer** to examine the data (Figure 6).




| | date | A | AA | AAPL | ABC |
|---|---|---|---|---|---|
| | date | continuous | continuous | continuous | continuous |
| 1 | 05/10/2002 00:00:00 | 27.95 | 34.99 | 23.32 | 75.42 |
| 2 | 05/13/2002 00:00:00 | 28.46 | 35.35 | 23.94 | 75.59 |
| 3 | 05/14/2002 00:00:00 | 31.05 | 36.25 | 25.61 | 75.23 |
| 4 | 05/15/2002 00:00:00 | 29.75 | 36.07 | 25.28 | 75.86 |
| 5 | 05/16/2002 00:00:00 | 29.60 | 35.85 | 25.21 | 76.04 |
| 6 | 05/17/2002 00:00:00 | 30.01 | 35.66 | 25.01 | 77.62 |
| 7 | 05/20/2002 00:00:00 | 29.40 | 35.38 | 24.74 | 77.31 |
| 8 | 05/21/2002 00:00:00 | 28.54 | 35.60 | 23.46 | 77.27 |
| 9 | 05/22/2002 00:00:00 | 28.11 | 35.45 | 24.32 | 77.80 |
| 10 | 05/23/2002 00:00:00 | 27.83 | 35.90 | 25.18 | 78.50 |
| 11 | 05/24/2002 00:00:00 | 27.30 | 35.58 | 24.15 | 78.08 |
| 12 | 05/28/2002 00:00:00 | 27.14 | 35.40 | 23.98 | 78.45 |
| 13 | 05/29/2002 00:00:00 | 26.70 | 35.30 | 23.98 | 78.81 |
| 14 | 05/30/2002 00:00:00 | 26.13 | 35.15 | 24.20 | 78.95 |

Figure 6

We now have separate columns for each ticker symbol; instead of 127,190 rows and three columns as before, we now have 252 columns (corresponding to one year of daily data) and 502 columns, one for each stock. The fact that there are more than 500 stocks is a consequence of some stocks not being included in the S&P500 at the last day of the time series (May 9, 2003).

We now turn our attention to the second data set. Double-click a **Read SAS File** component to move it to the worksheet, and then double-click it to open the **Properties** page. Click the **Browse** button and navigate to **5techstocks.sas7bdat**. Then click the **Update Preview** button, and the dialog should look like Figure 7.

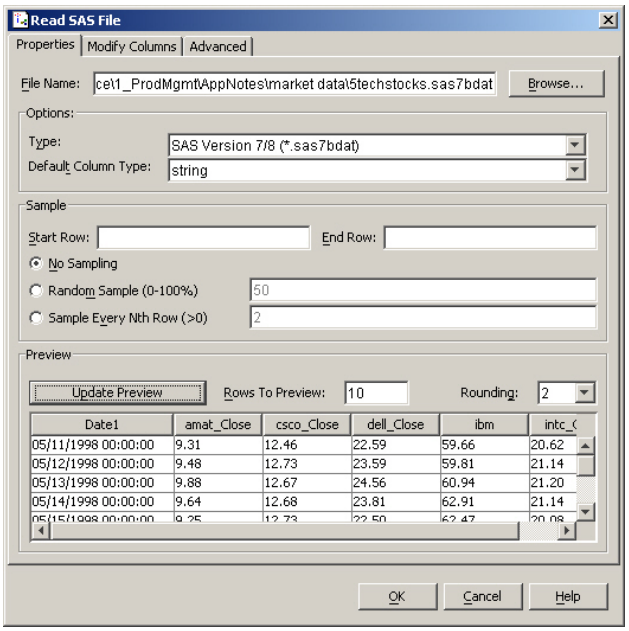



Figure 7

Figure 7 shows the expected data output format for this file: a date column and one price column for each of the five stocks. Right-click the node and select **Run to Here** to validate the node, and then right-click and select **Viewer** to display the data, shown in Figure 8.

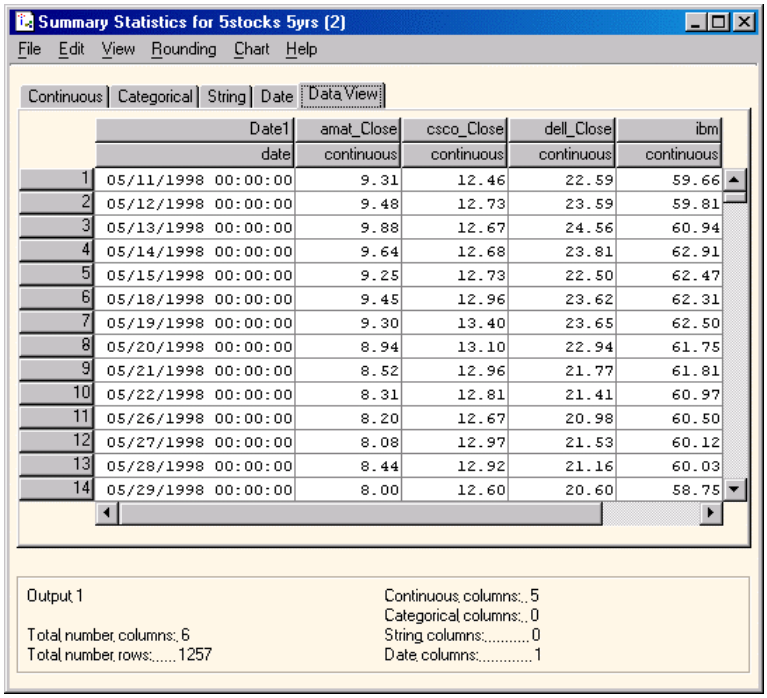
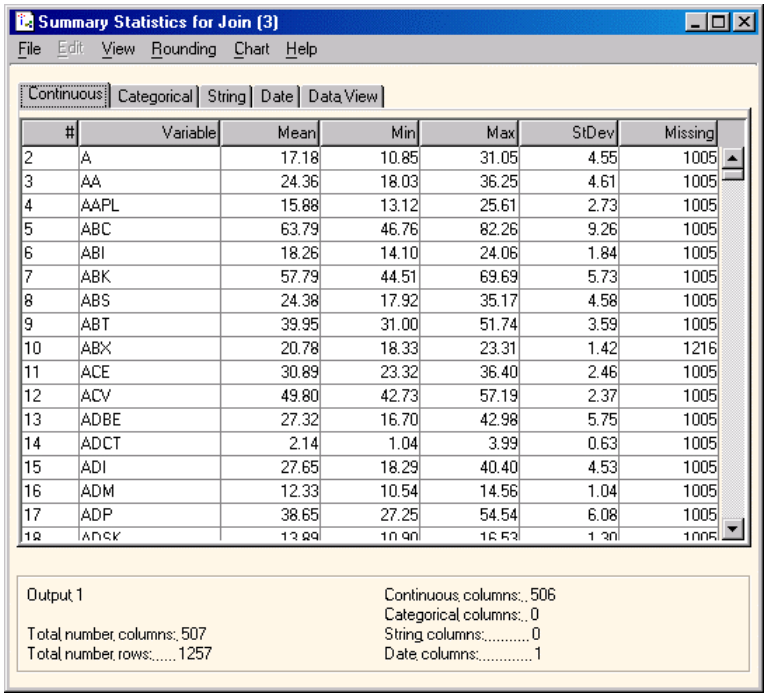



Figure 8

This data has the same format as the first source with S&P 500 data, so we can join these two data sources into a single data set using the **Join** node.

Double-click a **Join** component to move it to the worksheet, and connect the **Read Text File** and **Read SAS File** nodes to it. In the **Key Column** groupbox, select **date** for the **Read Text File** node and **Date1** for the **Read SAS File** node. Run the **Join** node, right-click and select **Viewer**, and the output should look like Figure 9.

*Th **Join** node allows the integration of data sources and gives the user full control over the details of the join operation.*




| # | Variable | Mean | Min | Max | StDev | Missing |
|---|---|---|---|---|---|---|
| 2 | A | 17.18 | 10.85 | 31.05 | 4.55 | 1005 |
| 3 | AA | 24.36 | 18.03 | 36.25 | 4.61 | 1005 |
| 4 | AAPL | 15.88 | 13.12 | 25.61 | 2.73 | 1005 |
| 5 | ABC | 63.79 | 46.76 | 82.26 | 9.26 | 1005 |
| 6 | ABI | 18.26 | 14.10 | 24.06 | 1.84 | 1005 |
| 7 | ABK | 57.79 | 44.51 | 69.69 | 5.73 | 1005 |
| 8 | ABS | 24.38 | 17.92 | 35.17 | 4.58 | 1005 |
| 9 | ABT | 39.95 | 31.00 | 51.74 | 3.59 | 1005 |
| 10 | ABX | 20.78 | 18.33 | 23.31 | 1.42 | 1216 |
| 11 | ACE | 30.89 | 23.32 | 36.40 | 2.46 | 1005 |
| 12 | ACV | 49.80 | 42.73 | 57.19 | 2.37 | 1005 |
| 13 | ADBE | 27.32 | 16.70 | 42.98 | 5.75 | 1005 |
| 14 | ADCT | 2.14 | 1.04 | 3.99 | 0.63 | 1005 |
| 15 | ADI | 27.65 | 18.29 | 40.40 | 4.53 | 1005 |
| 16 | ADM | 12.33 | 10.54 | 14.56 | 1.04 | 1005 |
| 17 | ADP | 38.65 | 27.25 | 54.54 | 6.08 | 1005 |
| 18 | ADSK | 13.89 | 10.90 | 16.53 | 1.30 | 1005 |

Figure 9

Figure 9 shows that we added five new columns, corresponding to the five tech stocks, so now we have an integrated data set. Click the **Data View** tab, and note that the time range is now the same for all stocks from May 10 of 2002 until May 9 of 2003, as shown in Figure 10.

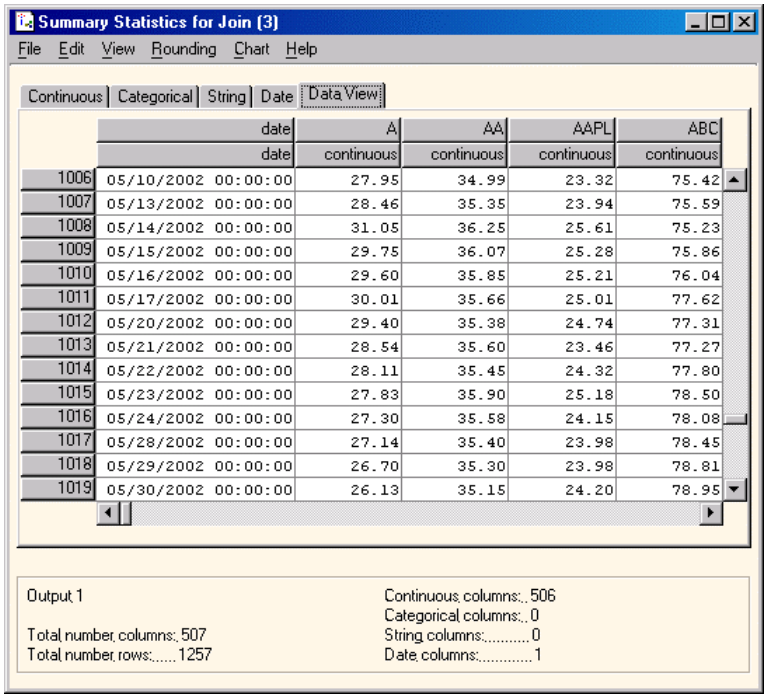



Figure 10

The worksheet so far looks like Figure 11:

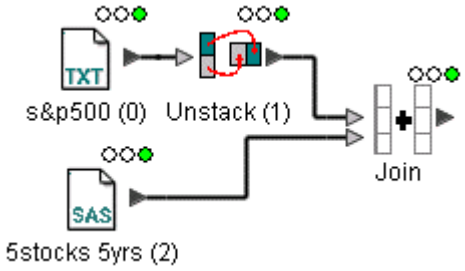



Figure 11

## Step 2: Stock Screening

*In this example, we assume that only a pre-defined subset of stocks is used for subsequent constructing of efficient portfolios.*

In the first step, we read in data (from two different sources) that consisted of price observations for all the assets in the considered portfolio from. Often, as in this case, more asset data is retrieved at this step than is actually required for the portfolio universe. Occasionally, analysts have a fixed list of the assets in a given portfolio, such as the previously mentioned S&P 500 portfolio example; more often, the portfolio under consideration is only defined using some selection criteria, and the constituents of the portfolio might even change over time. In this case, we need to construct an asset selection or stock screening step, which defines the constituents of the portfolio.

In this example, we want to restrict the size of our stock universe to a much smaller size than the currently available 500 stocks. In this case, we restrict it to a pre-defined list of 13 stocks and use a **Filter Columns** node to select them: ACV, AFL, AM, APC, DTE, KLAC, MSFT, TE, `amat_Close`, `csco_Close`, `dell_Close`, `ibm`, and `intc_Close`.

In this example, we use a subset of the stock universe, and consider only those stocks with a minimum price above $10 over the full time period for the optimization procedure. We introduce the **S-PLUS Script** component in Insightful Miner, which we title **Drop min<=$10**. The associated S-PLUS code is shown below:

*Extending the capabilities of Insightful Miner by using the S-PLUS programming language gives analysts the power they need to embed proprietary methods quickly.*

```
if (IM$test)
  {
    return(list(dynamic.outputs=T,
                in1.requirements=c("one.block","meta.data")))
  }
good.cols <- ifelse(is.na(IM$in1.column.min), T, (IM$in1.column.min>=10))
if (!any(good.cols)) stop("No columns pass the screen of Min >= 10")

IM$in1[,good.cols,drop=F]
```

This code is used to check the metadata for each column to find those columns with a minimum price value below $10, and drops those columns from the new data set. In this case, only two stocks will be excluded due to this condition: TE and CSCO.
The worksheet thus far looks like Figure 12.

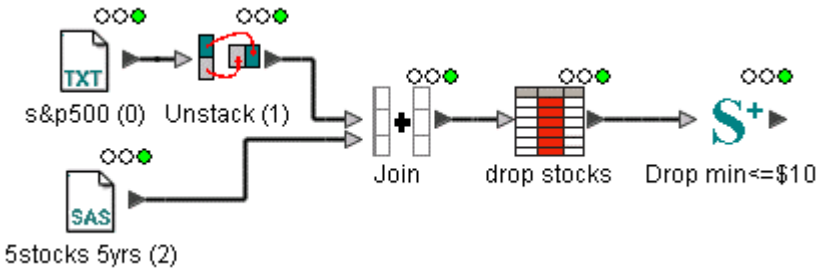


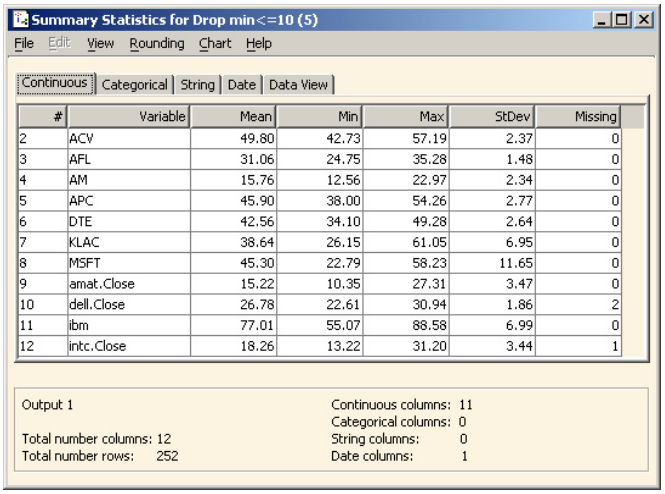

Figure 12

# Step 3: Computing and Exploring Returns

## Interpolating Missing Values

*Interpolation of missing values needs to be done prior to computing returns and is readily available with S+FinMetrics.*

If we run the **S-PLUS Script** node, right-click it, and select **Viewer**, we note that some time series elements still have missing values, as shown in Figure 13. It's better to address these missing data now, because one missing value in the price series creates two missing values in the corresponding returns series. Simple interpolation schemes, as provided by the `interpNA` function (available by attaching the S+FinMetrics module), are usually enough.




| # | Variable | Mean | Min | Max | StDev | Missing |
|---|---|---|---|---|---|---|
| 2 | ACV | 49.80 | 42.73 | 57.19 | 2.37 | 0 |
| 3 | AFL | 31.06 | 24.75 | 35.28 | 1.48 | 0 |
| 4 | AM | 15.76 | 12.56 | 22.97 | 2.34 | 0 |
| 5 | APC | 45.90 | 38.00 | 54.26 | 2.77 | 0 |
| 6 | DTE | 42.56 | 34.10 | 49.28 | 2.64 | 0 |
| 7 | KLAC | 38.64 | 26.15 | 61.05 | 6.95 | 0 |
| 8 | MSFT | 45.30 | 22.79 | 58.23 | 11.65 | 0 |
| 9 | amat.Close | 15.22 | 10.35 | 27.31 | 3.47 | 0 |
| 10 | dell.Close | 26.78 | 22.61 | 30.94 | 1.86 | 2 |
| 11 | ibm | 77.01 | 55.07 | 88.58 | 6.99 | 0 |
| 12 | intc.Close | 18.26 | 13.22 | 31.20 | 3.44 | 1 |

Figure 13

DELL and INTC have two and one missing values, respectively. We want to fix this for the price data, since each missing price results in two missing returns data points. To do this, we create another **S-PLUS Script** node and call `interpNA` (available by attaching the S+FinMetrics module), designed to interpolate missing values in time series. The content of the second **S-PLUS Script** node is shown on the following page.

```
# Interpolate missing values
if (IM$test) {
   if (!is.element("finmetrics", search())) module(finmetrics)
  }
# interpNA() doesn't handle timeDate cols
# only do interp for the other cols
isDateCol <- sapply(IM$in1, function(x) { data.class(x) == "timeDate" })
for (i in seq(along=isDateCol)) {
   if (!isDateCol[i])
     IM$in1[,i] <- interpNA(IM$in1[,i], method="spline", maxStartNA=5)
 }

# return the full data.frame
IM$in1
```

For each time series column with missing values, replacement values are interpolated using a spline method. After running this node, the data has no more missing values (Figure 14).

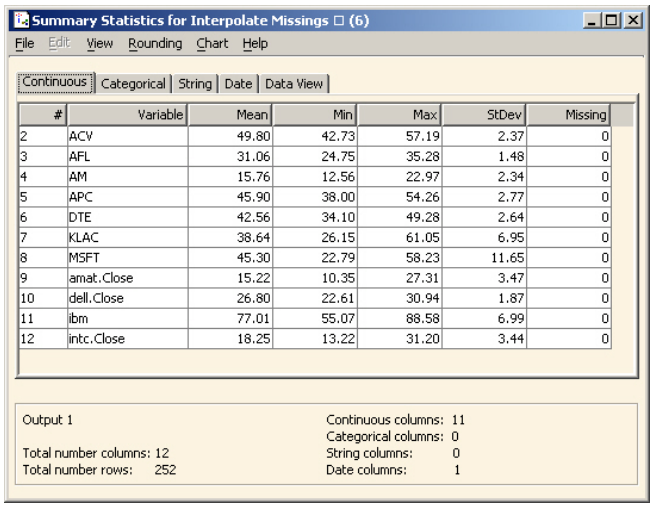



Figure 14

Although only returns data are usually used in portfolio optimization problems, it helps to visualize the original price data to check if there are any abnormal behavior or patterns in the data. You can use the technical indicators, which are usually based on the price data. In Figure 15, we see the output from a **Line Plot** node, used to create a time series graph with all currently selected stocks.

*Insightful Miner has powerful graphics for displaying time series.*

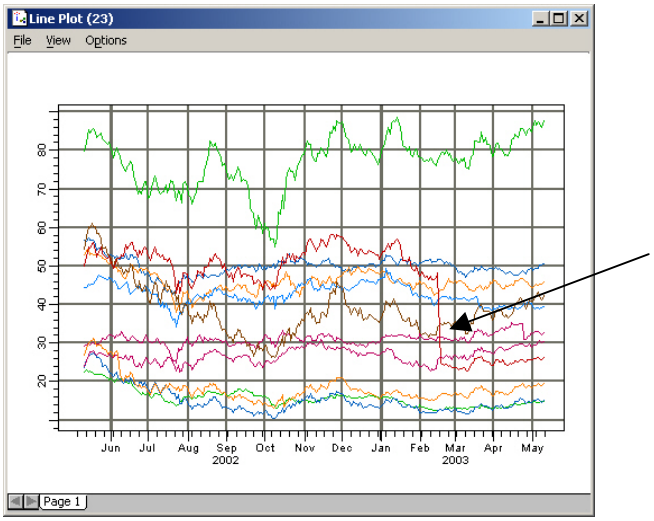



Figure 15

Note the one major price drop from this high-level view, indicated by the arrow in Figure 15.

The current worksheet for this project should look like Figure 16:

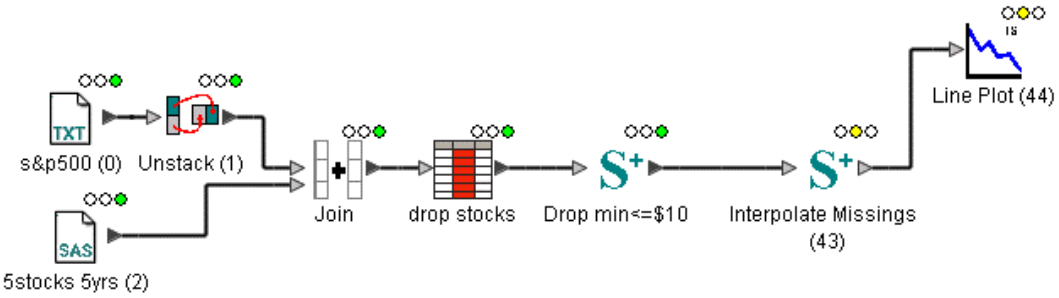



Figure 16

We are now ready to proceed with the calculation of returns.

## Computing Returns

*S+FinMetrics supports various methods for computing returns.*

Transformation from price data to returns data is usually easy to do, and either continuous compounding (the log difference) or discrete compounding (the percentage change) is used to compute returns.

First, double-click an **S-PLUS Script** component to move it to the worksheet, and use the S+FinMetrics function `getReturns` to compute the returns based on the available price data.

```
# Compute returns for numeric cols, leaving other
# columns unchanged
 if (IM$test) {
   if (!is.element("finmetrics", search())) {
     module(finmetrics)
   }
   return(list(out1=IM$in1))
 }
# Replace each numeric column with the returns.
# The returns are one element shorter than the input, put an
# NA at the front to match up the lengths and then drop the
# row at the end.
 for (i in 1:ncol(IM$in1)) {
   if (is.numeric(IM$in1[,i])) {
     IM$in1[,i] <- c(NA, getReturns(IM$in1[,i],
       type="discrete", percentage=F))
   }
 }
 return(IM$in1[-1,])
```

## Exploring Returns Data

*Specialized data exploration nodes can be built using integration with S+FinMetrics.*

After obtaining the returns data, the quantiles and the first four sample moments are usually computed for data summary purpose. We could also compute common tests and measures for each test, such as normality test, autocorrelation test, ARCH test, classical/robust alpha, classical/robust beta, etc. Visualization could also include scatter plots, pairwise plots, normal QQ-plots, etc. The scatter plots and pairwise plots help determine if there are any visual outliers in the data. For all these, we use S+FinMetrics function calls embedded in S-PLUS nodes.

Here, we restrict ourselves to only plotting the various returns time series, computing time series summaries and unit root tests.

First, we double-click an S-PLUS Script node to move it to the worksheet, and we plot all time series returns with the following code:

```
# This script creates a series plot
 if (IM$test) {
   if (!is.element("finmetrics", search())) {
     module(finmetrics)
   }
   return(list(simple=T))
 }
# Open a new graphics device
 java.graph()
# Create a timeSeries() using the first timeDate()
# col as the positions and dropping other date cols
 isDateCol <- sapply(IM$in1, function(x) {
   data.class(x) == "timeDate" })
 whichDateCols <- seq(1, ncol(IM$in1))[isDateCol]
 if (length(whichDateCols)==0) {
   warning(
     "No date column is available. Using default origin.")
   pos <- NULL
 }
 else {
   pos <- IM$in1[,whichDateCols[1]]
 }
# Create the timeSeries() object
 ts <- timeSeries(IM$in1[,!isDateCol, drop=F],
   position=pos)
# Series plot
 for (x in colIds(ts)) {
   plot(ts[,x])
```

```
   title(x)
   java.set.page.title(x)
 }
# Don't return anything
 return(NULL)
```

Running this node creates a series of plots, one for each returns time series, as shown in Figure 17.

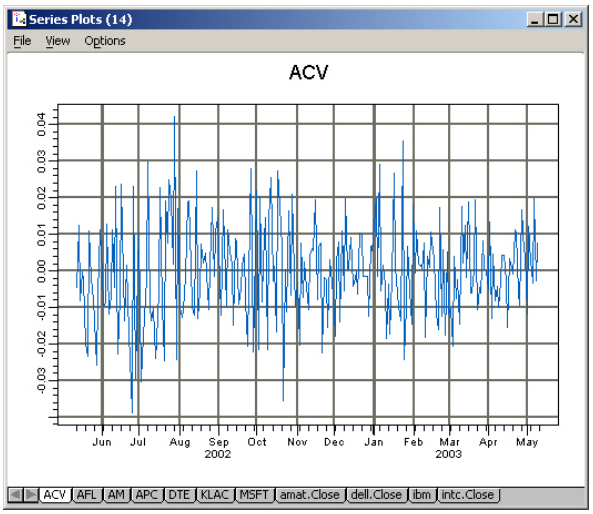



Figure 17

Now, we have a look at the summary statistics for the considered time series. The corresponding **S-PLUS Script** node uses this code:

```
# This script computes time series summary statistics
 if (IM$test) {
   if (!is.element("finmetrics", search())) {
     module(finmetrics)
   }
   return(list(simple=T))
 }
# Create a timeSeries() using the first timeDate()
# col as the positions and dropping other date cols
 isDateCol <- sapply(IM$in1, function(x) {
   data.class(x) == "timeDate" })
 whichDateCols <- seq(1, ncol(IM$in1))[isDateCol]
 if (length(whichDateCols)==0) {
   warning(
     "No date column is available. Using default origin.")
   pos <- NULL
 }
 else {
   pos <- IM$in1[,whichDateCols[1]]
 }
# Create the timeSeries() object
 ts <- timeSeries(IM$in1[,!isDateCol, drop=F],
   position=pos)
# Print summaries
 print(summaryStats(ts))
# Return nothing
 return(NULL)
```

As a result, for each time series it computes minimum, maximum, quartiles, median, mean, standard deviation, skewness, and kurtosis, as shown in Figure 18.

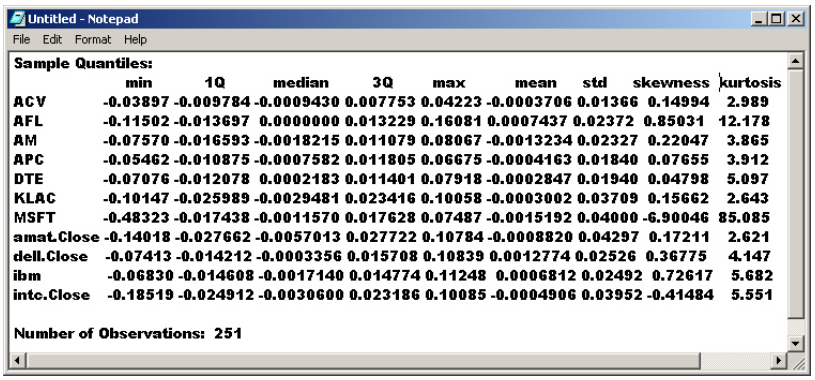



Figure 18

Note that MSFT has a large, negative skewness and a large kurtosis when compared to the other stocks.

Finally, we conclude this section with a unit root test, for which we use the S+FinMetrics function `unitroot` and some S-PLUS code to create another **S-PLUS Script** node:

```
# Compute unit root tests
 if (IM$test) {
   if (!is.element("finmetrics", search())) {
     module(finmetrics)
   }
   return(list(simple=T))
 }
# Perform the unit root test for all numeric columns
 for (i in 1:ncol(IM$in1)) {
   if (is.numeric(IM$in1[,i])) {
     cat("\n\t*** Unit Root Test for",
       names(IM$in1)[i], "***\n")
     print(unitroot(IM$in1[,i], trend="ct", method="adf",
       lags=10, na.rm=F))
   }
 }
# Don't return anything
 return(NULL)
```

Running this node generates the report shown in Figure 19.

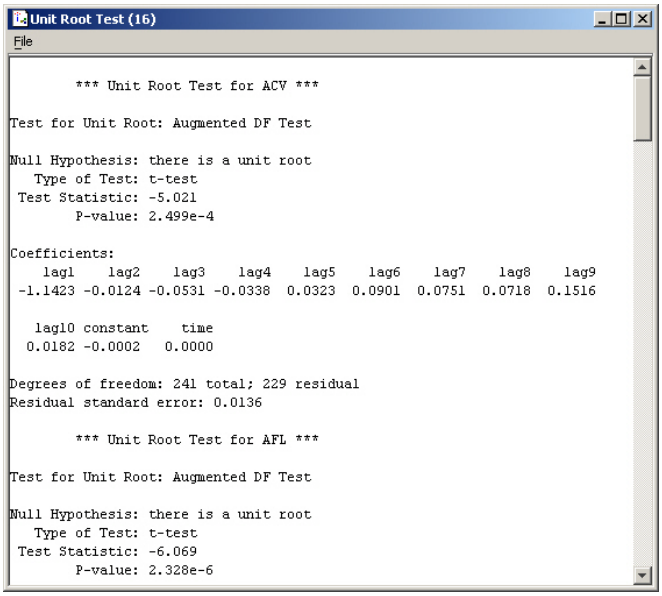



Figure 19

We have now concluded the exploratory part, and we summarize our progress by showing the current worksheet up to this point (Figure 20):

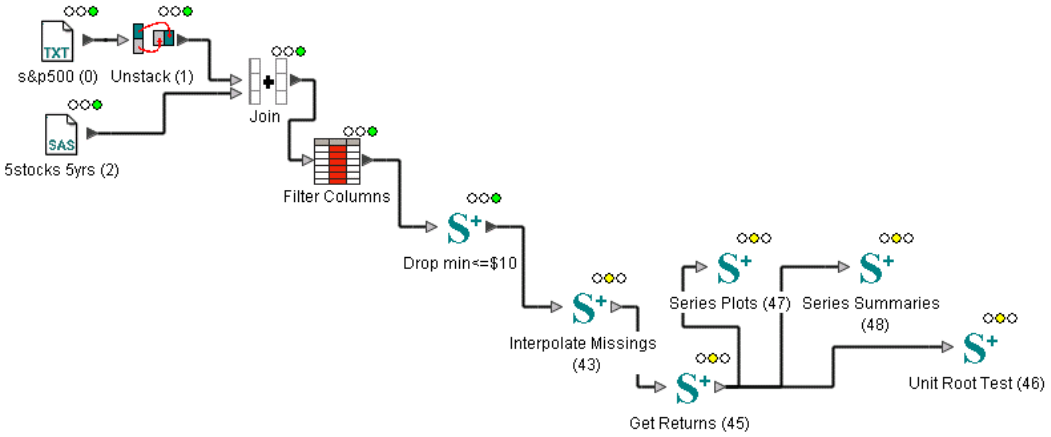



Figure 20

## Step 4: Computing the Efficient Frontier and Optimal Portfolios

As mentioned previously, the input to a portfolio optimization problem consists of estimates of expected returns and the covariance matrix. The most basic approach assumes the returns follow a multivariate normal distribution and simply computes the sample mean and sample covariance matrix. Alternatively, simple moving averages or EWMA estimates of the expected returns and covariance matrix can be used.

In practice, it has been found that factor models usually generate better estimates of expected returns and the covariance matrix for portfolio allocation purposes. The single market factor model simply states that for each asset, the returns are linearly correlated with the market returns, and thus a simple linear regression can be used to obtain expected returns and the covariance estimate.

The single market factor model provides a first cut of the factor model, but this can usually be improved. The BARRA type factor model (using firm specific fundamentals and sector indicators as factors) and statistical factor models (principal components as factors) are usually preferred. The `mfactor` function in S+FinMetrics implements the statistical factor model.

### Computing the Robust Mean-Covariance Matrix

We first compute the mean covariance matrix using the `fit.models` function from the S-PLUS robust library.

```
# This node computes mean and cov values
# for the optimization step
 isDateCol <- sapply(IM$in1, function(x) {
   data.class(x) == "timeDate" })
 if (IM$test) {
   if (!is.element("robust", search())) {
     library(robust, first=T)
   }
  return(list(in1.requirements="one.block",
    out1=cbind(Mean = rep(0, nrow(IM$in1)),
    IM$in1[!isDateCol])))
 }
java.graph()
 df <- IM$in1[, !isDateCol, drop=F]
 meanVec <- sapply(df, mean)
 varMat <- var(df)
 plot(fit.models(list(Robust = "covRob",
   Classical = "cov"),
   data = df), which = 3)
# Return matrix with mean as first col and then cov mat
 out <- data.frame(cbind(Mean=meanVec, varMat))
 return(list(out1=out))
```

*Using the robust library within Insightful Miner to detect outliers.*

Running this code creates a graphical matrix representation with the comparison of classical (in blue) and robust estimations (in black) in Figure 21.

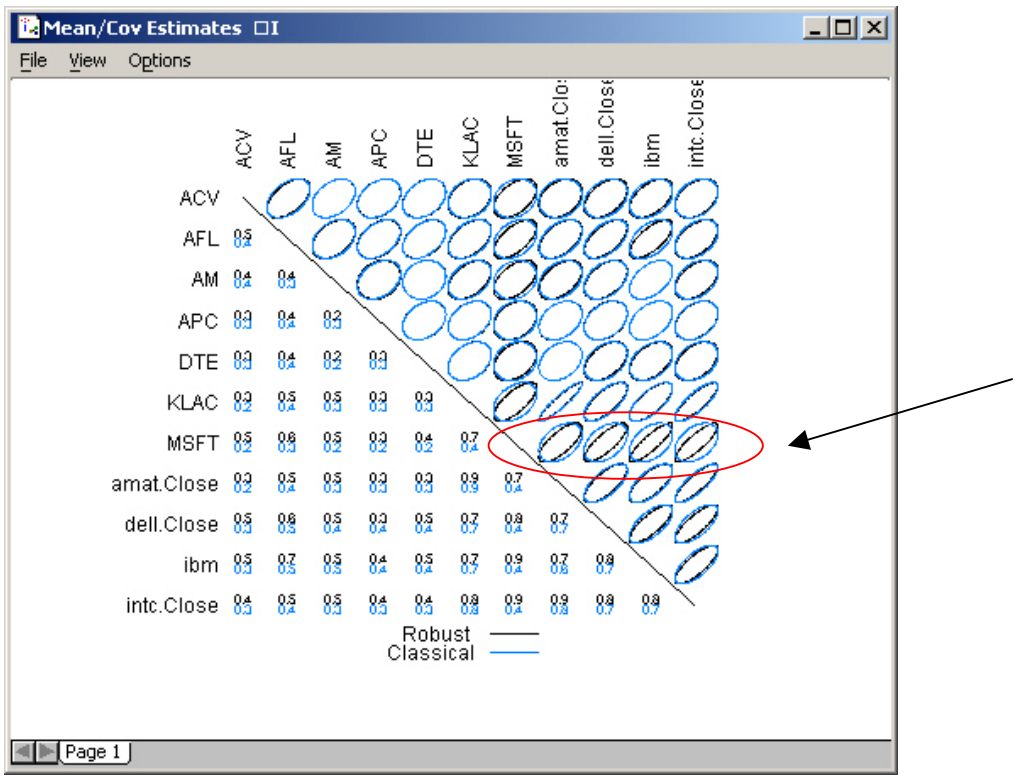



Figure 21

It is apparent that MSFT-related estimations show a large difference between both methods, which indicates that outliers may be an important consideration. To analyze this, we go back to the time series plot node and look at the time series returns plot for MSFT (right), and produce a time series plot for its price data (left), as shown in Figure 22:

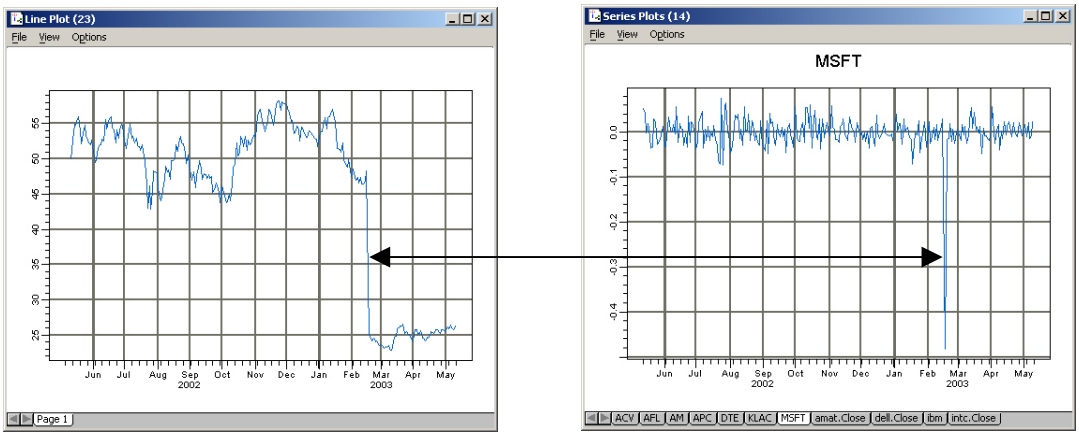



Figure 22

The returns outlier is clearly a consequence of the large price jump that occurred mid February 2003. Further investigation reveals that this is due to a MSFT stock split in that period.

We use a multi-dimensional outlier detection implemented in Insightful Miner's outlier node to isolate this and perhaps other similar outliers from our returns data. It computes the Mahalanobis distance based on which we exclude the outliers using a **Filter Rows** node.

Now we're ready to compute the mean covariance matrix for a second time based on the data, which is now clear of outliers. We use a copy of the **S-PLUS Script** node used above and obtain the result shown in Figure 23:

*Multidimensional (true) outliers can be easily removed with Insightful Miner's **Outlier Detection** and **Filter Rows** nodes.*

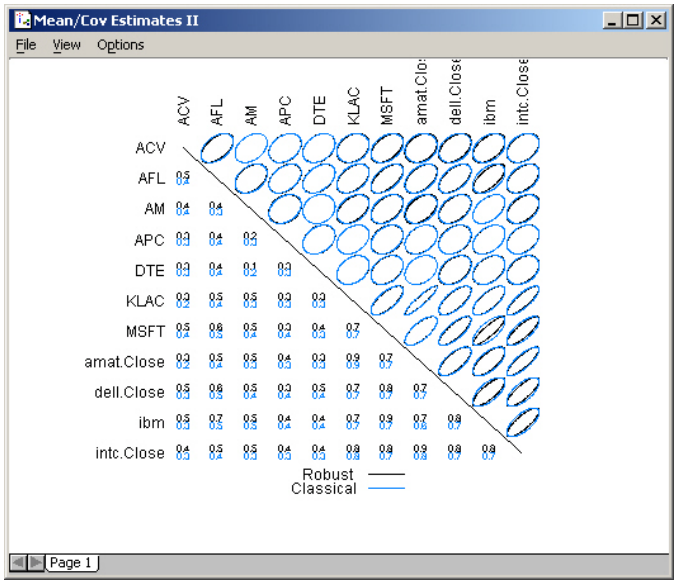



Figure 23

We observe a much smaller difference between the classical and robust estimations, which leads us to believe we've resolved the problem. We are now ready for the final step of constructing the efficient frontier with the given stock universe.

## Computing the Efficient Frontier

*Thanks to the integration with S+NuOPT, large-scale optimization problems with thousands of stocks and mixed integer constraints can be robustly solved.*

We double-click one last **S-PLUS Script** node to move it the worksheet. This time we use function calls to the S+NuOPT module, Insightful's optimizer for solving large-scale problems. The code used is shown below:

```
if (IM$test) {
   if (!is.element("nuopt", search())) { module(nuopt) }
   return(list(in1.requirements="one.block",
     out1=data.frame(SIGMA1=0, MU1=0, SIGMA2=0, MU2=0)))
}
module(nuopt)
java.graph()
meanVec <- IM$in1[,1]
names(meanVec) <- names(IM$in1)[-1]
varMat <- as.matrix(IM$in1[,-1, drop=F])
sigma <- sqrt(diag(varMat))
optfron <- portfolioFrontier(varMat, meanVec, wmin=-Inf,
  max.ret=max(meanVec)*1.5,n.ret=50)
optfron1 <- portfolioFrontier(varMat, meanVec, wmin=0,
  max.ret=max(meanVec),n.ret=50)
xlim <- range(optfron$sd,optfron1$sd,.045)
ylim <- range(optfron$ret,optfron$ret,0,.0025)
plot(optfron$sd,optfron$ret,xlim=xlim,ylim=ylim,
```

```
  type="l",xlab="SIGMA",ylab="MU")
lines(optfron1$sd,optfron1$ret, lty=3,lwd=4)
points(sigma,meanVec,pch=2)
text(sigma + 0.01 * xlim[2], meanVec, names(meanVec),
  adj= 0)
title(main="EFF. FRONTIERS WITH AND WITHOUT SHORT SELLING")
return(list(out1=data.frame(SIGMA1=optfron$sd,
  MU1=optfron$ret,SIGMA2=optfron1$sd, MU2=optfron1$ret)))
```

A total of 50 points is computed along the efficient frontier, which is computed for both cases with short selling (continuous line) and without short selling (dotted line). The graph created by this code shows the efficient frontiers and the location of the various stocks, shown in Figure 24.

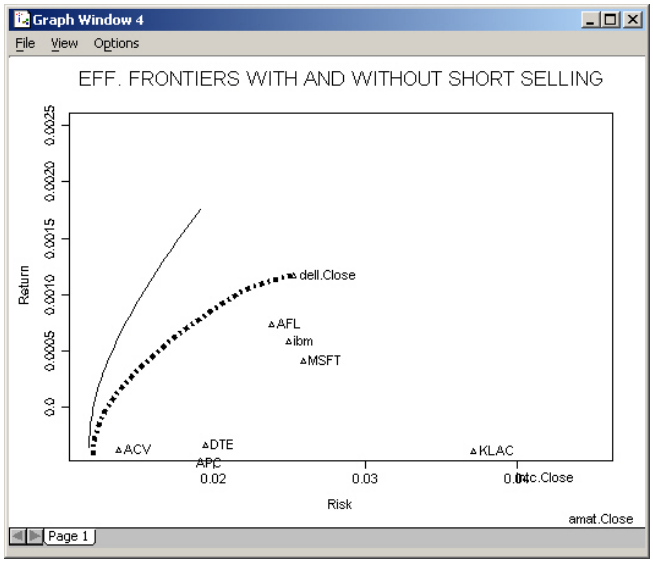
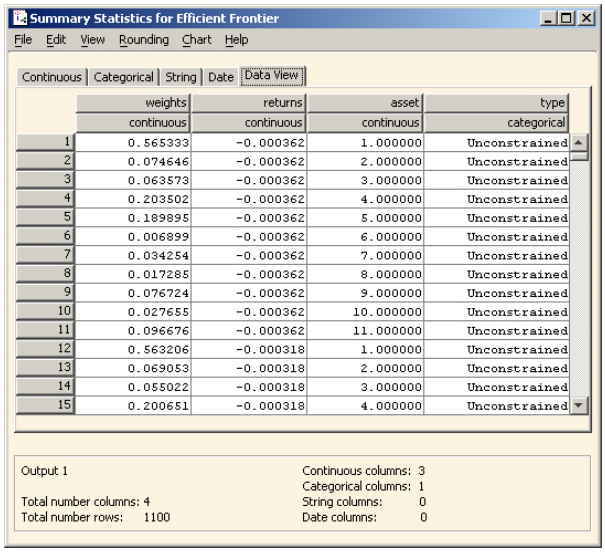



Figure 24

If we right-click the node and select **Viewer**, we can analyze the data. Select the **Data View** tab to show the entire output, seen in Figure 25.




| | weights | returns | asset | type |
|---|---|---|---|---|
| | continuous | continuous | continuous | categorical |
| 1 | 0.565333 | -0.000362 | 1.000000 | Unconstrained |
| 2 | 0.074646 | -0.000362 | 2.000000 | Unconstrained |
| 3 | 0.063573 | -0.000362 | 3.000000 | Unconstrained |
| 4 | 0.203502 | -0.000362 | 4.000000 | Unconstrained |
| 5 | 0.189895 | -0.000362 | 5.000000 | Unconstrained |
| 6 | 0.006899 | -0.000362 | 6.000000 | Unconstrained |
| 7 | 0.034254 | -0.000362 | 7.000000 | Unconstrained |
| 8 | 0.017285 | -0.000362 | 8.000000 | Unconstrained |
| 9 | 0.076724 | -0.000362 | 9.000000 | Unconstrained |
| 10 | 0.027655 | -0.000362 | 10.000000 | Unconstrained |
| 11 | 0.096676 | -0.000362 | 11.000000 | Unconstrained |
| 12 | 0.563206 | -0.000318 | 1.000000 | Unconstrained |
| 13 | 0.069053 | -0.000318 | 2.000000 | Unconstrained |
| 14 | 0.055022 | -0.000318 | 3.000000 | Unconstrained |
| 15 | 0.200651 | -0.000318 | 4.000000 | Unconstrained |

Figure 25

Select all the data in Figure 25 and then select **Chart** | **Three Columns** | **Surface Plot**, and create a graph to show the various weights for each asset for the two cases of short sales and no short sales. This is shown in Figure 26:

*Unique Trellis graphs provide powerful explorative capabilities.*

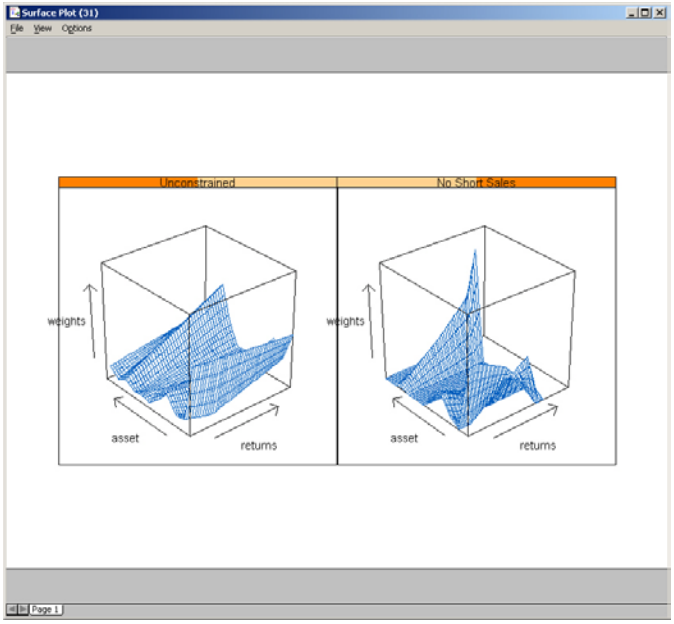
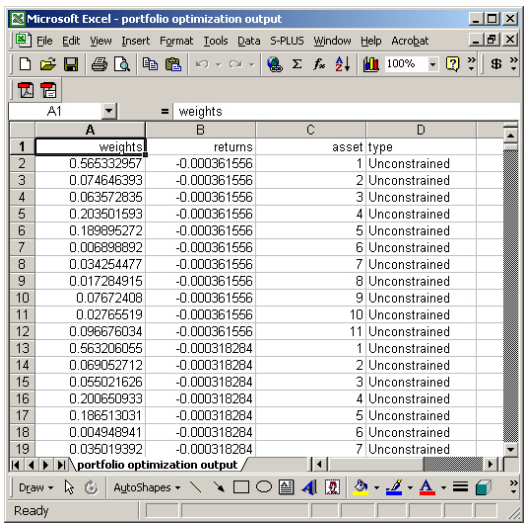



Figure 26

Finally, we export this data to construct a new portfolio or correct an existing portfolio according to the investor's risk/return objectives. To do this, we use a **Write Excel File** node, and the exported data looks like Figure 27 in Excel:

*Data export to numerous databases including Oracle, Sybase, DB-2, SQL Server and data formats such as S-PLUS, Excel, SPSS, SAS, is supported by Insightful Miner.*




| | A | B | C | D |
|---|---|---|---|---|
| 1 | weights | returns | asset | type |
| 2 | 0.565332957 | -0.000361556 | 1 | Unconstrained |
| 3 | 0.074646393 | -0.000361556 | 2 | Unconstrained |
| 4 | 0.063572835 | -0.000361556 | 3 | Unconstrained |
| 5 | 0.203501593 | -0.000361556 | 4 | Unconstrained |
| 6 | 0.189895272 | -0.000361556 | 5 | Unconstrained |
| 7 | 0.006898892 | -0.000361556 | 6 | Unconstrained |
| 8 | 0.034254477 | -0.000361556 | 7 | Unconstrained |
| 9 | 0.017284915 | -0.000361556 | 8 | Unconstrained |
| 10 | 0.07672408 | -0.000361556 | 9 | Unconstrained |
| 11 | 0.02765519 | -0.000361556 | 10 | Unconstrained |
| 12 | 0.096676034 | -0.000361556 | 11 | Unconstrained |
| 13 | 0.563206055 | -0.000318284 | 1 | Unconstrained |
| 14 | 0.069052712 | -0.000318284 | 2 | Unconstrained |
| 15 | 0.055021626 | -0.000318284 | 3 | Unconstrained |
| 16 | 0.200650933 | -0.000318284 | 4 | Unconstrained |
| 17 | 0.186513031 | -0.000318284 | 5 | Unconstrained |
| 18 | 0.004948941 | -0.000318284 | 6 | Unconstrained |
| 19 | 0.035019392 | -0.000318284 | 7 | Unconstrained |

Figure 27

The first column (`weights`) describes the relative weight of the asset identified in column `asset` for a specific return value given in the column `returns` and type of constraint as shown in column `type`. This output can then be used to generate trading orders of the corresponding assets.

## Conclusion

Using a small sample of stocks extracted from the S&P 500 constituents, we have shown how to effectively construct an efficient portfolio by optimizing the weight for each stock required to obtain specific risk-return characteristics. The integration of Insightful Miner, S-PLUS, S+NuOPT and S+FinMetrics results in a very powerful data analysis and modeling platform for this type of application.

The intuitive ease and flexibility of Insightful Miner's visual programming paradigm makes such data analysis applications easier to understand and share with others. Further, the highly scalable architecture allows handling large-data problems with no difficulty.

We have also shown how to use both pre-built analytic components and S-PLUS-based custom created components to solve typical asset management problems in a completely new way. Together, Insightful Miner along with S-PLUS, S+FinMetrics, and S+NuOPT form a powerful and seamlessly integrated analytical environment for advanced financial analysis, modeling, and deployment.

## About Insightful Corporation

Insightful Corporation (NASDAQ: IFUL) provides enterprises with scalable data analysis solutions that drive better decisions faster by revealing patterns, trends and relationships. The company is a leading supplier of software and services for statistical data analysis, data mining and knowledge access enabling clients to gain intelligence from numeric and text data. The company's products include S-PLUS®, Insightful Miner, StatServer®, S-PLUS Analytic Server®, S+FinMetrics™, S+Wavelets®, and S+NuOPT™. Headquartered in Seattle, Insightful has offices in New York City, North Carolina, France, Switzerland, and the United Kingdom with distributors around the world. For more information, visit http://www.insightful.com/, email info@insightful.com or call 1-800-569-0123.

## Contact Us

For more information contact Insightful at the information below or email info@insightful.com.

**Global Headquarters**
1700 Westlake Ave N
Suite 500
Seattle, WA 98103
Tel: 206.283.8802
Fax: 206.283.6310
info@insightful.com

**Insightful France**
7, rue Auber
31000 Toulouse
France
Tel: +33 0 5 62 27 70 60
Fax: +33 0 5 62 27 70 61
info.fr@insightful.com

**Insightful Switzerland**
Christoph Merian-Ring 11
4153 Reinach
Switzerland
Tel: +41 61 717 9340
Fax: +41 61 717 9341
info.ch@insightful.com

**Insightful UK**
5th Floor
Network House
Basing View
Basingstoke, Hampshire
RG21 4HG
Tel: +44 (0) 1256 339800
Fax: +44 (0) 1256 339839
info.uk@insightful.com