Using Random Portfolios with R

Patrick Burns
http://www.burns-stat.com

June 2009

This was presented June 03 at the Thalesians.
http://www.thalesians.com

Outline

- Random portfolios
- The R language
- Demo of random portfolios in R

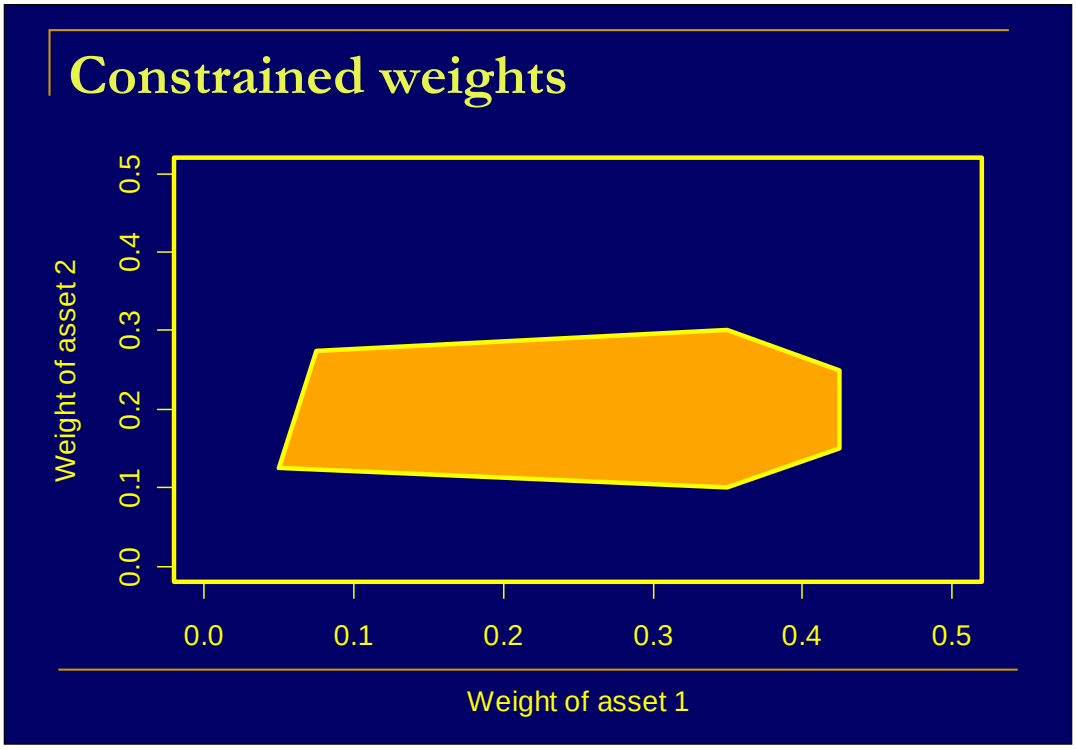
Outline

- Random portfolios
- The R language
- Demo of random portfolios in R

More information about random portfolios can be found on the random portfolios page of the Burns Statistics website:

http://www.burns-stat.com/pages/Finance/random_portfolios.html

Random portfolios are intimately tied to constraints – no constraints, no random portfolios.



The idea of random portfolios is to take a random sample from the set of portfolios that obey the given set of constraints.

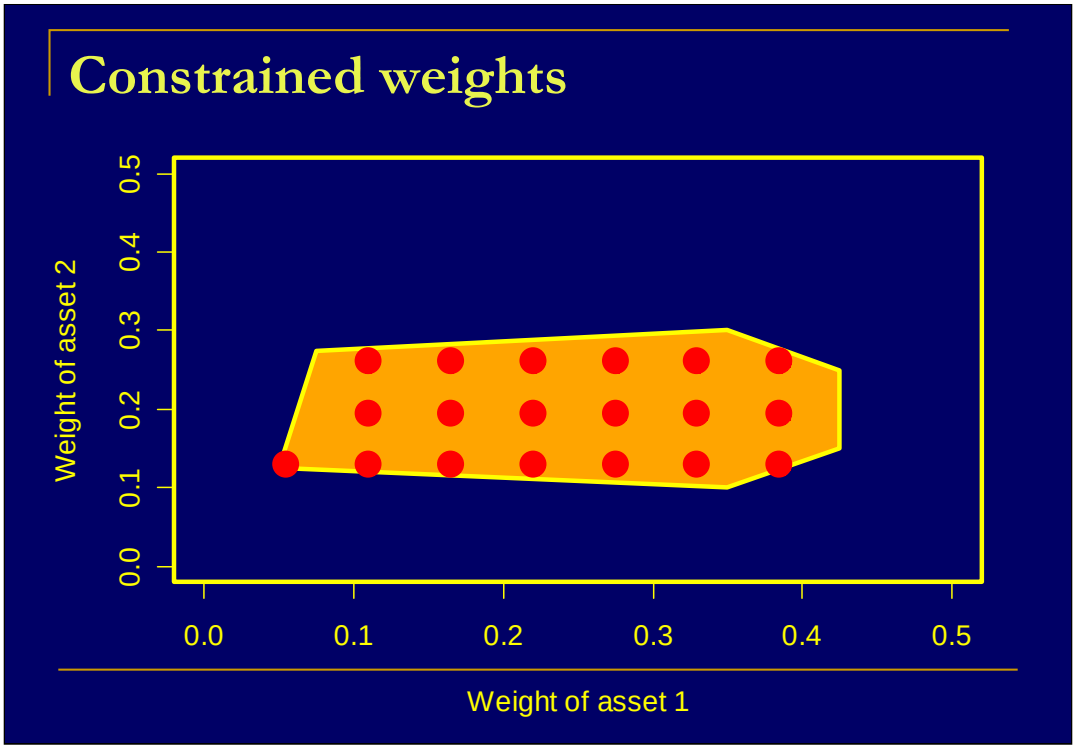
This picture shows an example with a three-asset problem.

We want to sample from the orange region.

There are two things wrong with this picture.

The first thing that is wrong is that all of the lines are straight, meaning that all the constraints are linear. Constraints are not always linear.

The second thing wrong is that the picture implies that we have a continuous space. Real portfolios are discrete – you can't buy 9.457 shares of Vodafone.



So we are really on a lattice like the red dots.

Applications

- **Performance Measurement**
- **Testing Trading Strategies**
- **Evaluating Constraints**
- **Validating Risk Models**
- **…**

The first three of these applications will be discussed shortly.

The fourth application will be the subject of the demonstration.

There is a large number of additional applications of random portfolios. Some of them known, most of them yet to be discovered.

Applications

- Performance Measurement
- Testing Trading Strategies
- Evaluating Constraints
- Validating Risk Models
- …

Perfect Performance Measurement

- **Look at all possible portfolios that the manager might have held**
- **Take the return of each of these portfolios over the time period**
- **Compare actual return to the distribution from all possibilities**

Fund managers have a cloud of portfolios that they MIGHT hold, and they pick one. We get a sense of how good they are by where the return of the actual fund fits into the distribution of returns of all the portfolios they might have held.

Some Caveats

- **Should account for implicit as well as explicit constraints**
- **Return need not be the measure**
- **Trading is allowed**

Funds can have implicit constraints (growth-oriented perhaps) as well as explicit constraints (maximum weight, sector, etc). Ideally we should take both into account.

For simplicity I speak of return as the performance measure, but it doesn't have to be. Risk-adjusted returns are a possible alternative.

The statement about a fund manager picking one portfolio out of their cloud implies that no trading is done during the time period of interest.

When we allow for trading, then a path is traced through the cloud rather than having a single point. A single point is easier to visualize but the path doesn't really complicate our analysis.

Perfect Performance Measurement

- **Number of possible portfolios is finite but astronomical**

While the number of portfolios that a fund might select is finite, that number is astronomical. We are never going to be able to deal with the entire set.

But we can take a random sample.

Random Portfolio Measurement

- **Take a random sample from the set of all portfolios that might have been**
- **Fraction of random portfolios with a larger return than the fund is a p-value**
- **Null Hypothesis is zero skill**

We can perform a statistical hypothesis test in the same spirit as a random permutation test.

Peer Groups

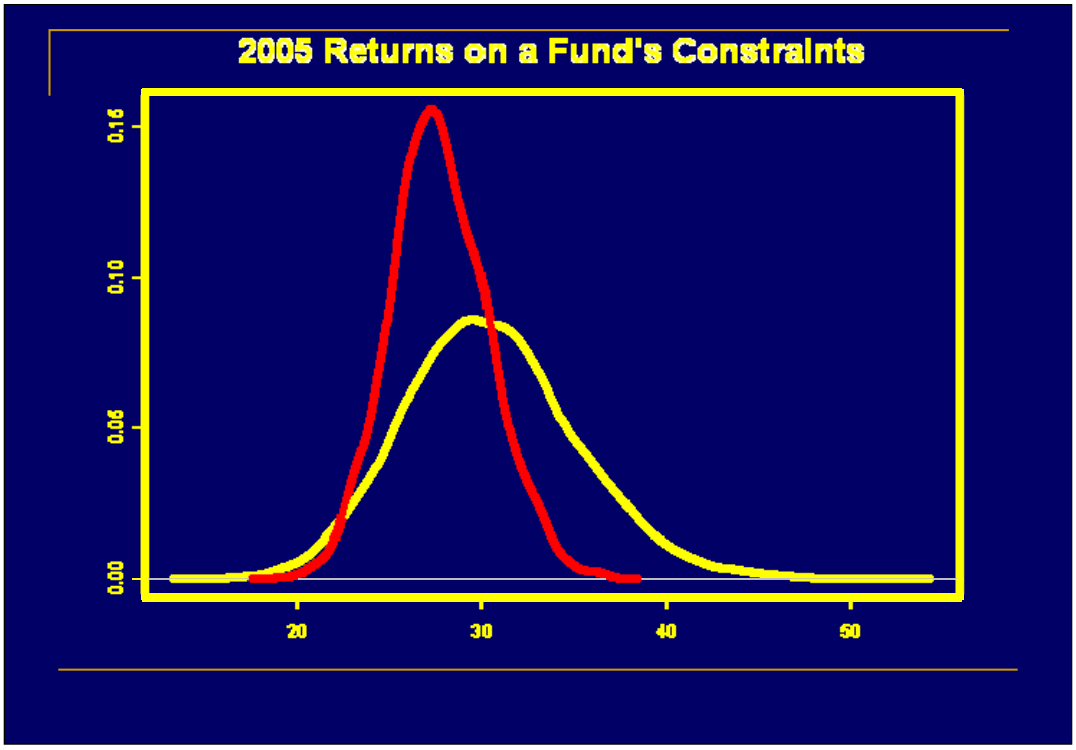
- **Superficially like random portfolios**
- **Comparing against "random" portfolios of unknown skill**

There are two methods of performance measurement in common use: benchmarks and peer groups.

Using a benchmark requires a long time (as in decades) to get reasonable statistical power.

Using peer groups looks superficially a lot like using random portfolios. Both use a single time period, both compare our fund of interest to a group of alternative portfolios.

The key difference is that we know the skill level of random portfolios (it is zero skill) but we have no idea of the skill of the peers. Hence we don't know the meaning of our results when we use peer groups.



The distribution in yellow is the return over the calendar year given a certain set of constraints. The distribution in red is the distribution under the same set of constraints, plus we are starting at a particular portfolio at the beginning of the year and we have slightly over 200% turnover (buys plus sells) throughout the year.

Knowledge of the initial positions of a fund is information that random portfolios can take advantage of but benchmarks and peer groups can not.

The next slide highlights the dramatic improvement in inference for this case.

Upper tail probabilities

Return	uncond	cond
30%	.52	.19
31%	.43	.10
32%	.35	.06
33%	.28	.03
35%	.16	.004
40%	.03	0

If we are looking for a 3% p-value, then conditional on this starting portfolio we only need to see a 33% return in the fund while we need to see a 40% return unconditionally.

Applications

- Performance Measurement
- Testing Trading Strategies
- Evaluating Constraints
- Validating Risk Models
- …

The Situation

- Alpha model
- Set of constraints for the portfolio
- Trading strategy
- Want to know if we'll make money

The Alpha Model

- MA_{26} = equally weighted mean of the previous 26 daily returns
- MA_{12} = equally weighted mean of the previous 12 daily returns
- Alpha = $MA_{26} - MA_{12}$

The Constraints

- 45 – 50 names
- Long-short
- Net close to zero
  - Try to keep it less than 5% of gross in absolute value
  - Try very hard to keep it less than 10% of gross
- Try to keep maximum weight less than 10% for any stock

Trading

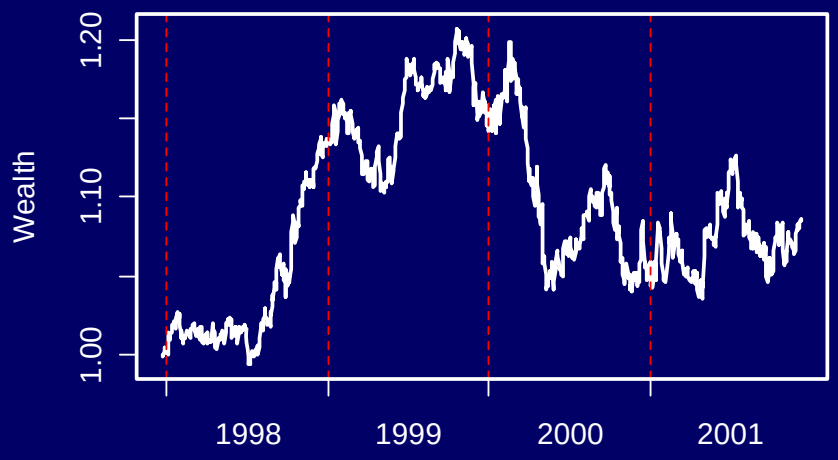
- Variance matrix is statistical factor model based on last 500 daily returns
- Objective is to maximise the information ratio
- Trade every day
- Restrict turnover (buys plus sells) to 400% per year
- Starts with a particular portfolio of 50 stocks

Data

- **Unsystematic collection of 186 US equities, both large cap and small cap**
- **Data start at beginning of 1996**
- **Daily data**
- **Trading period is essentially 1998 through 2001**

We trade for 1000 days.

Results



How Good Is It?

- **8.5% gain over 4 years**
- **Take away a lot for trading costs**
- **If quit at start of 2000: happy**
- **If start in early 2000: sad**

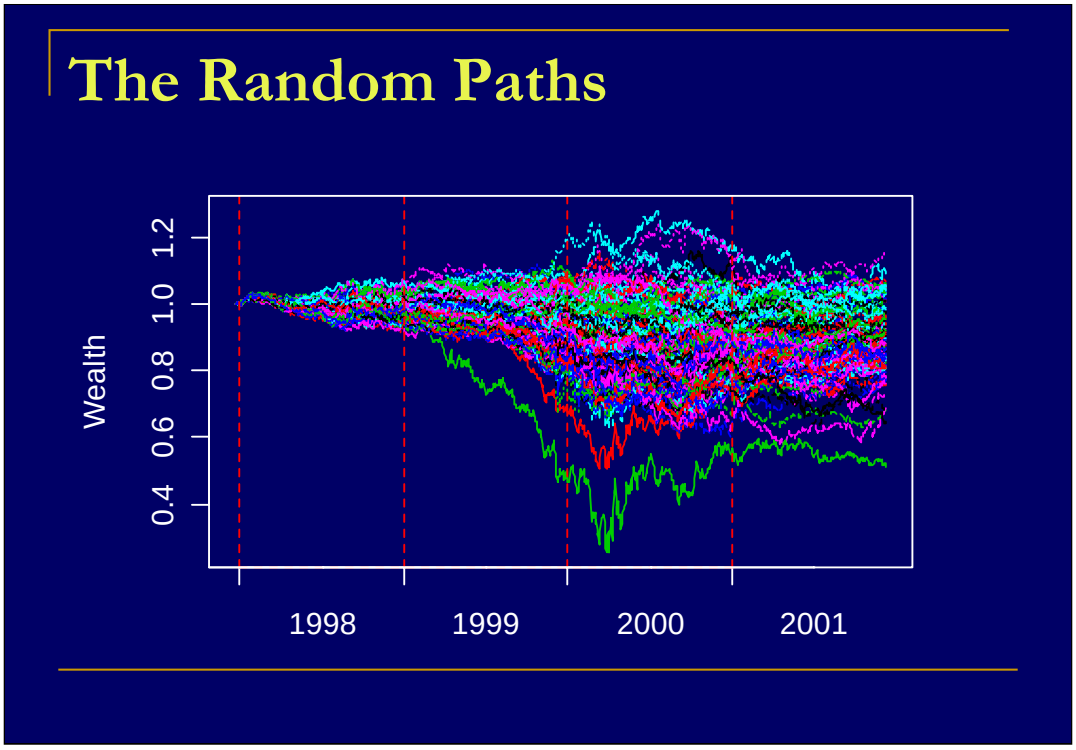
As far as I know this is about as specific as we can be about the quality of the results without using random portfolios. A qualitative analysis is possible, but random portfolios allow us to make a quantitative analysis.

Generate Random Trades

- **Want to mimic the optimisation process**
- **Generate 100 random paths**
- **All 100 paths start with the initial portfolio**
- **On each day, each random portfolio performs a trade that satisfies the constraints**
- **So a total of 100,000 random trades performed**

We will create 100 different paths that mimic the original backtest, but that are random and ignore our return predictions. We create paths with zero skill.

At a given point in time each of the 100 portfolios will be in a different state, so the possible trades that maintain the constraints will be different for each of them.

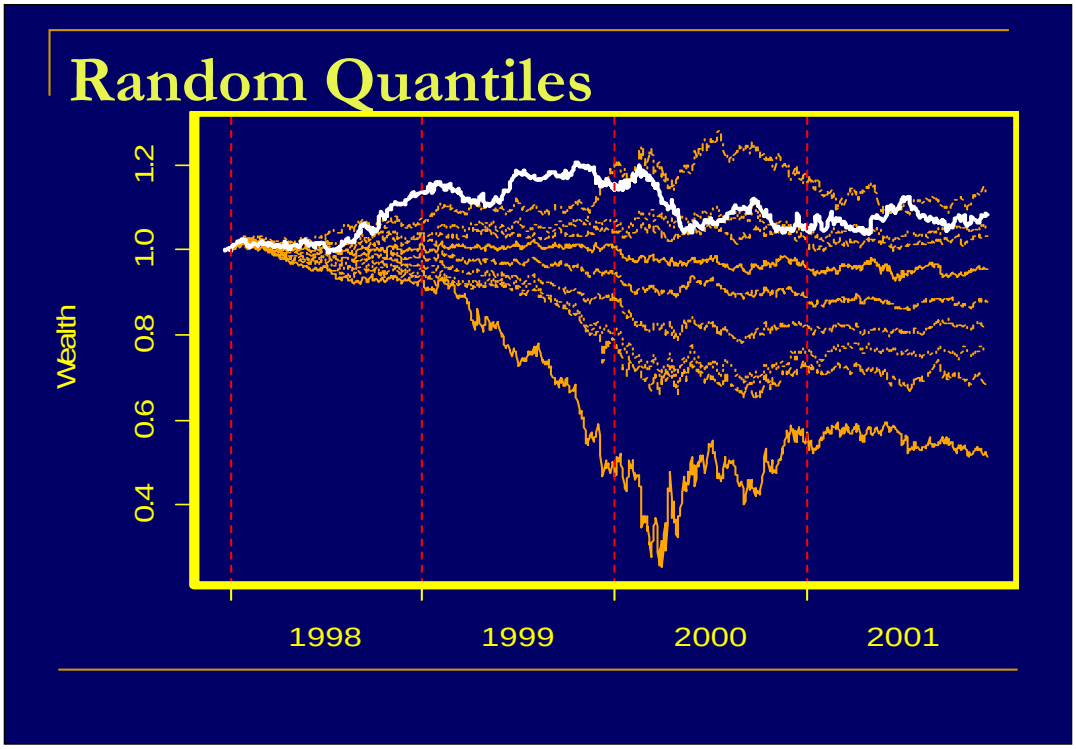


Here are the 100 paths. A pretty picture but not especially informative.

Random Quantiles

- **On each day, plot:**
  - **Maximum**
  - **95% quantile**
  - **90% quantile**
  - **75% quantile**
  - **50% quantile**
  - **25% quantile**
  - **10% quantile**
  - **5% quantile**
  - **minimum**

More informative is a plot that gives the quantiles of the distribution of paths on each day. These quantiles are not a specific path – typically the quantile corresponds to some particular path for some number of days and then switches to another path.



Here we can see how the original backtest (in white) compares to the random paths.

In mid 1998 the test was decidedly inside the scatter of the random paths. Then it gains so that by the start of 1999 it is significantly better than the best random path.

When the dot.com bubble burst, so did our portfolio. It then goes mostly sideways. It ends being better than all but 2 or 3 of the random paths. So this is a quite good result.

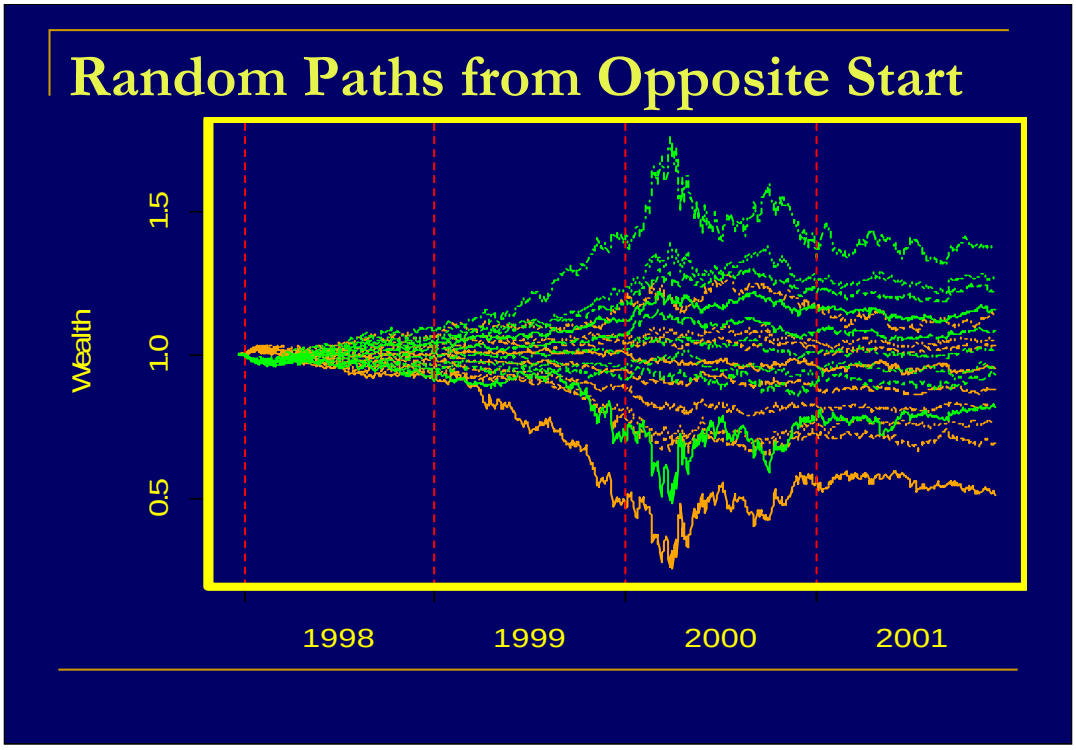
But we aren't really interested in how good the strategy is starting from this particular initial portfolio, we care about it being good starting from any portfolio. So we want to do this whole process a number of times starting from different portfolios.

A Bias for Losing

- **Random paths mostly slope down**
- **Initial portfolio loses over time period**
- **Persistent influence of initial portfolio**

There remains the question of why a path that gained only about 2% a year on average would test so well.

Close inspection of the previous plot shows that the quantiles are tending downwards in general. It turns out that the initial portfolio performs poorly over the time period, and there is surprising persistence of the influence of the initial portfolio.



In this plot the orange lines are the same as before. The green lines are quantiles for random paths that started with the opposite of the original initial portfolio. That is, long positions become short and short positions become long.

At the start of 1999 the range of the random paths for the two cases are quite similar. At that point there has been 400% turnover, so conceptually we should have got rid of the initial portfolio twice over.

The two distributions diverge significantly after that – implying that the effect of the initial portfolio is still there.

Applications

- Performance Measurement
- Testing Trading Strategies
- Evaluating Constraints
- Validating Risk Models
- …

Constraints

- Why do we impose constraints?
- Insurance
- What protection are we buying?
- What price is the premium?

FTSE Example

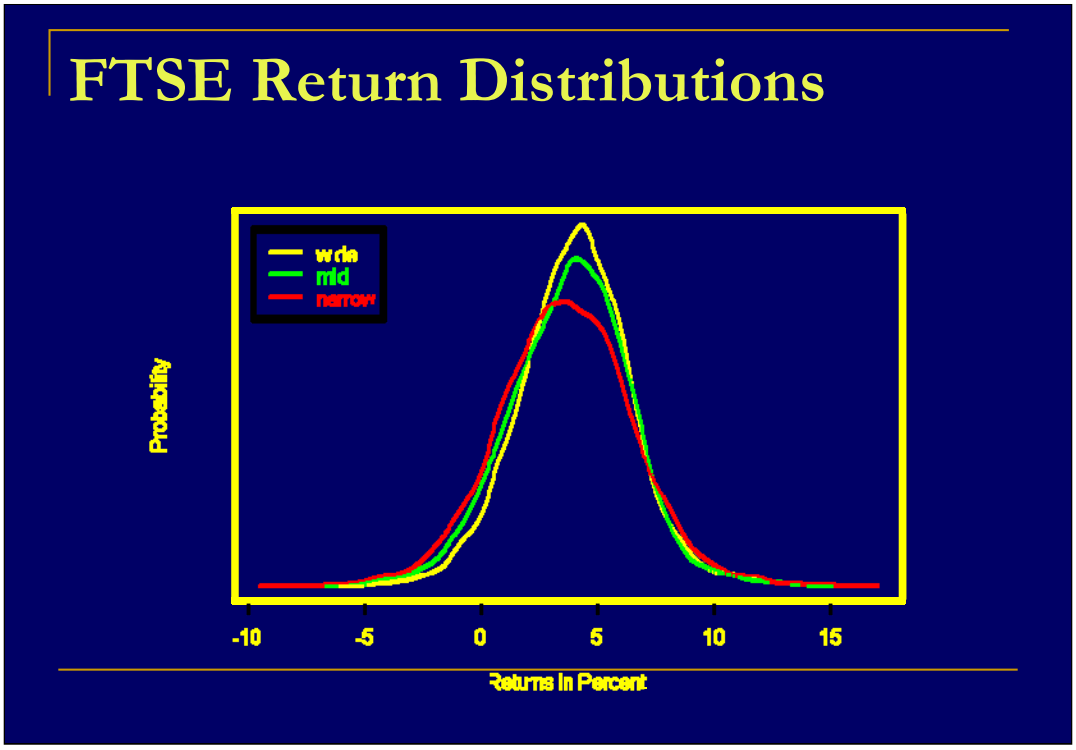
- FTSE 350 Data
- 10,000 portfolios generated for each set of constraints
- Returns: 2006 Jan 01 – 2006 June 01
- Long-only
- 90 – 100 assets in portfolio
- Nested set of linear constraints

FTSE Linear Constraints

- **Large cap versus Mid cap**
  - **10% - 30% 70% - 90%**
  - **13% - 27% 73% - 87%**
  - **17% - 23% 77% - 83%**
- **High yield versus Low yield**
  - **50% - 70% 30% - 50%**
  - **53% - 67% 33% - 47%**
  - **57% - 63% 37% - 43%**

FTSE Linear Constraints

- **5 Sectors**
  - **10% - 30%**
  - **13% - 27%**
  - **17% - 23%**

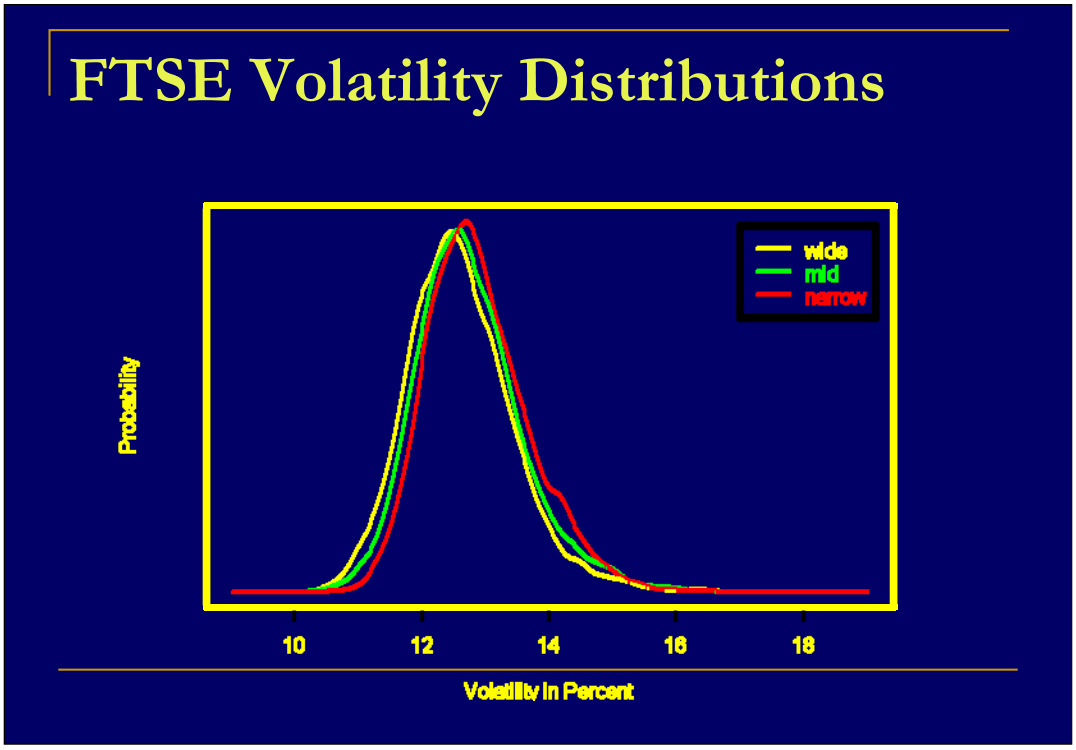


The results we see are just the opposite of what we expect.

The more constrained we are the wider the distribution, whereas our expectation is that the most constrained distribution should be narrower.

Why would this happen?

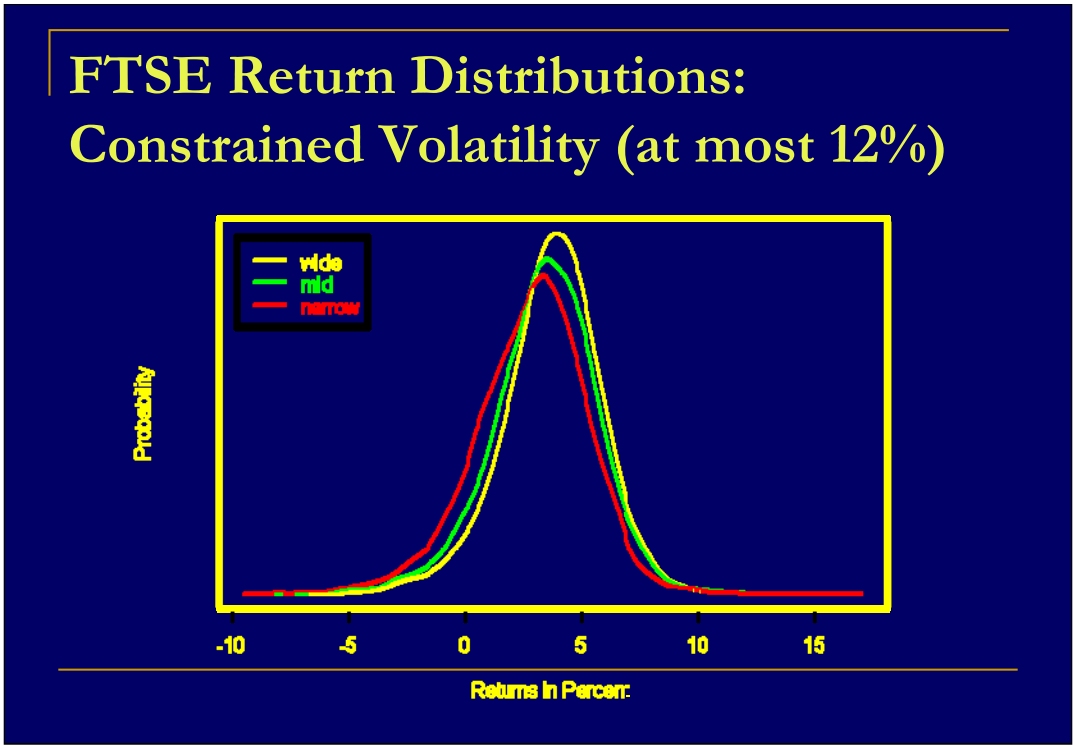
I don't know, but one possibility is that we are constraining into a high volatility region.



It does seem to be the case that we are constraining into volatility.

Note that this picture uses precisely the same random portfolios as last slide. The difference is that we are looking at volatility here rather than returns.

We can generate a new set of random portfolios that constrain volatility to be no more than 12%.



With the addition of the volatility constraint, the most constrained distribution now has smaller returns as well as a wider distribution.

Outline

- Random portfolios
- The R language
- Demo of random portfolios in R

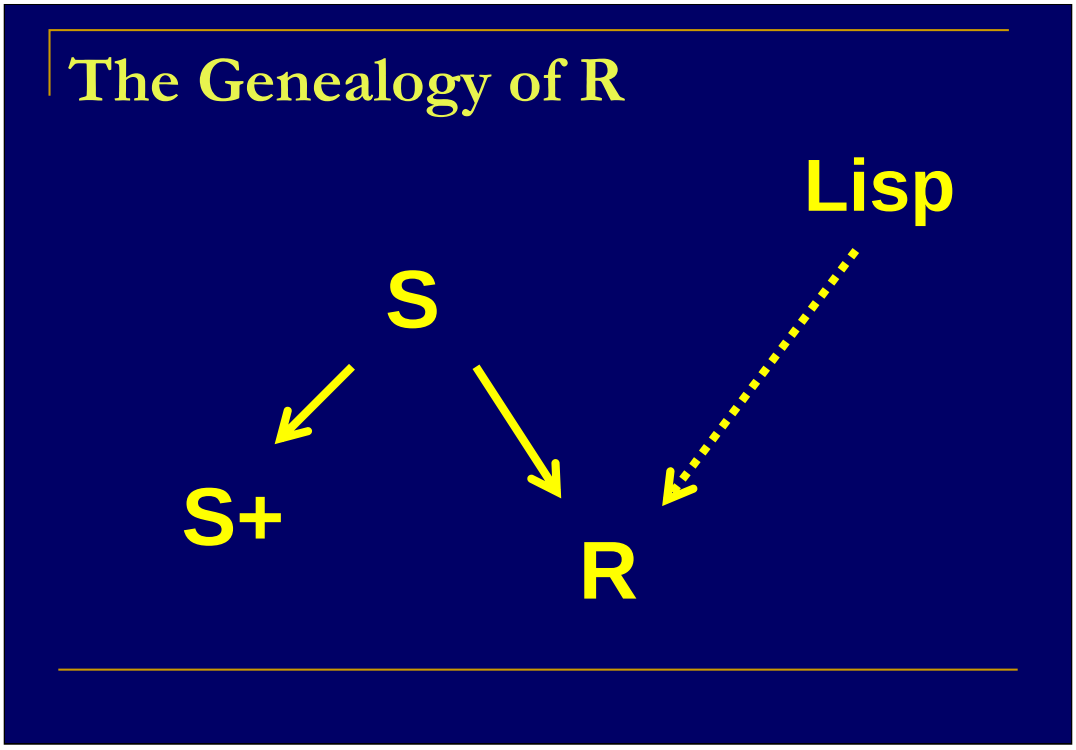
The R Language

- **Data analysis**
- **Graphics**
- **Good alternative to spreadsheets**
- **Free and open source**
- **http://www.r-project.org**

The R language was designed for the purpose of data analysis. A large part of good data analysis is graphics.

Much of what is currently done in spreadsheets would be better done in R. See "Spreadsheet Addiction" for why spreadsheets are inherently unsafe.

http://www.burns-stat.com/pages/Tutor/spreadsheet_addiction.html



In the mid 1970's the Data Analysis group at Bell Labs started a research project into a computing environment for data analysis.

In the mid 1980s that research project started to escape out into the wild in the form of the S language.

In the late 1980s S spawned a commercial product called S-PLUS.

In the early 1990s R was written. That first version of R might be characterized as Scheme with a large crust of syntactic sugar on top that made it look like the S language.

In the mid 1990s the R Project formed.

R has been growing ever since.

Language characteristics

- Rich in data structures
- Vector oriented
- Syntax similar to C
- Influenced by functional programming
- There exists object-orientation

A key difference with C is that indexing starts at 1, not 0.

The functional programming influence means that it is difficult to destroy your data accidentally.

Data structures

- **Atomic vectors**
  - **Logical**
  - **Numeric**
  - **Complex**
  - **Character**
- **All these have NA (missing value)**

Atomic vectors are each only of one type. You can have logical values in a vector or numbers in the vector, but not both.

Data structures

- **Lists**
  - **Components can be anything, including a list**
- **Functions**
  - **May have default values for arguments**
  - **Objects in the language**

Lists allow quite general data structures.

Default values of arguments in functions provides both convenience and flexibility.

Data structures

- **Attributes**
  - **A named list of meta-data about the object**
- **“class” attribute implies object orientation**
- **“dim” attribute implies matrix or higher dimensional array**

Attributes are a stroke of genius. They expand our universe from the earth with the sun and moon and a few planets spinning around it to a place with billions of galaxies.

Attributes provide the possibility of virtually any sort of data structure.

Some functions change their behavior based on the `class` of the object given as an argument.

If an object has a `dim` that is c(3, 2), then we know it is a matrix with three rows and two columns.

If an object has a `dim` that is c(3, 2, 4), then it is a three-dimensional array with three rows, two columns and four slices.

Subscripting

- **Can subscript with:**
  - **Positive integers**
  - **Negative integers**
  - **Characters**
  - **Logicals**

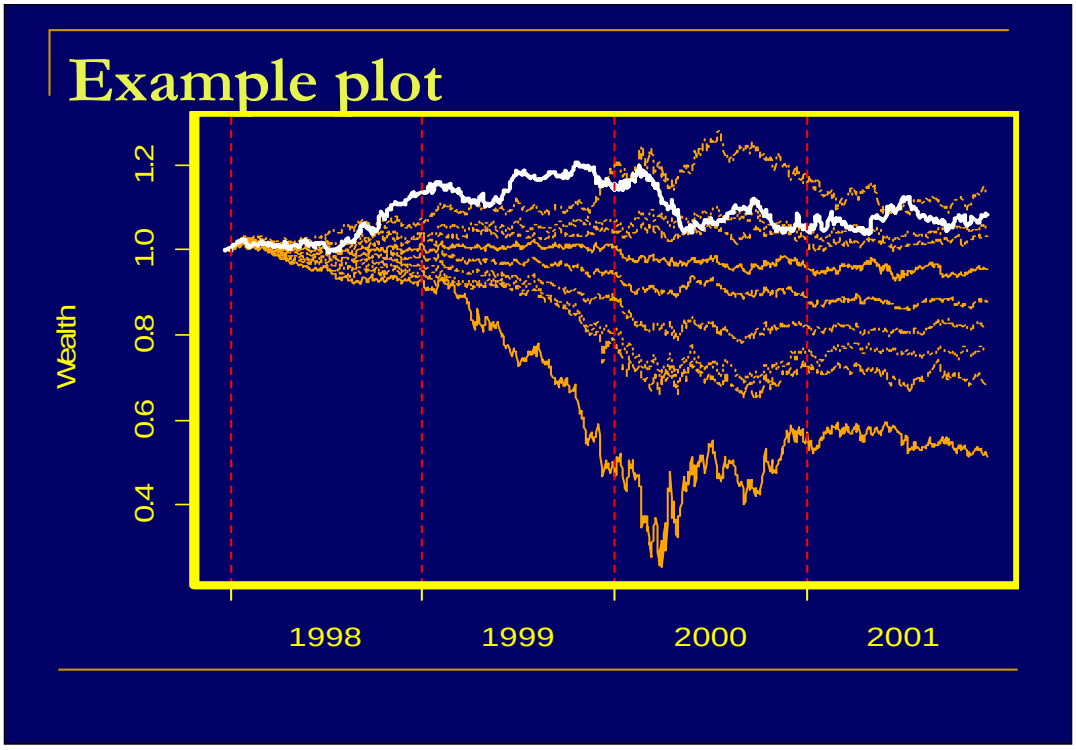
Subscripting is very important in R, partly because it is so flexible.

```
X[1:5] # selects the first five elements of X

X[-1:-5] # selects all but the first five elements of X

X[c('dog', 'cat')] # selects the elements with those names

X[c(TRUE, FALSE)] # selects the odd numbered elements
                  #  the logical vector is replicated to
                  #  the length of X
```



This plot has a custom x-axis that allows us to see quite well when events are happening.

The usual alternatives would be one of two cases:

1) The x-axis is treated as numbers from 1 to 1000.
2) There is a large number of dates written (possibly diagonally). You probably can't read the dates, and even if you can, you can not easily relate them to what is happening in the plot.

A key advantage of R is that it allows you to create graphics that do what they should, and not just what some programmer thinks you need.

Example plot

```
matplot(quantile.matrix, axes=FALSE,
  type='l', col='orange')
lines(orig, col='white')
axis(2)
custom.timeaxis(quantile.matrix)
box()
```

The previous plot was created with these commands[1]. Graphics don't have to be created with just one command, they can be built up with a number of commands.

The key bit here is a function that I wrote to create the x-axis as I wanted it to appear.

Note that when doing typical data analysis, a single graphics command will generally get you what you want – this technique of using multiple commands is mainly for presentation graphics. It would also be possible to write a function that did this sort of plot, so we'd be back to one command again.

[1] The first command should also include the argument:

`ylab='Wealth'`

But that didn't fit in the slide well.

Example: random portfolio

```
IBM  MSFT   C
237   84   -391
```

Named numeric vector:

```
rp <- c(IBM=237, MSFT=84, C=-391)

rp <- c(237, 84, -391)
names(rp) <- c("IBM", "MSFT", "C")
```

A natural way to represent a single random portfolio is as a named numeric vector.

Such an object can be created all in one go, or the data can be assigned to the object and then the `names` attribute added later. The resulting object is exactly the same in either case.

Example: random portfolios

- **We want a set of random portfolios**
- **List with length equal to number of random portfolios**
- **Attributes include “class” and “call” and a timestamp**

A natural representation of a set of random portfolios is as a list with the length of the list equal to the number of random portfolios, and each component of the list being a named numeric vector describing a single random portfolio.

The ‘`call`’ (attribute in this case, often a component of a list) is an image of the command that created the object. Objects that include their call are self-describing, and there are additional advantages.

R mailing lists

- R-help
- R-devel
- R-sig-finance
- R-sig-hpc
- …

There are several mailing lists for R.

‘sig’ stands for Special Interest Group.

And ‘hpc’ is High Performance Computing.

The dark side of R

- **Inconsistencies**
- **Redundancies**
- **RAM bound**
- **Slightly challenging to search on**
- **Drinking from a firehose**

There are many inconsistencies in R. A key reason for this is that the S language was a research project long after it was being used for substantial work. Backward compatibility has been a big issue for 20 years now.

'The R Inferno' tries to help you past the inconsistencies.

http://www.burns-stat.com/pages/Tutor/R_inferno.pdf

There are redundancies both because of the on-going research problem, and because there may be several contributed packages that do essentially the same thing.

The freedom of having any data structure we like has the price that the data needs to all be in RAM. That is a problem for large data -- generally there are workarounds, though.

If you think you want to know all of R, give up that idea.

The future of R

- **Three companies producing supported versions**
- **New statistical techniques appear first in R**
- **So bright I gotta wear shades**

For those of you who don't listen to Radio Paradise enough, the allusion in the last line is to a song by Timbuk3.

The momentum of R in the last few years is phenomenal.

I'm particularly surprised by the momentum in finance – the area that I know best.

Further reading:

"An Introduction to the S Language" provides a few more arguments of why R is a good thing.

http://www.burns-stat.com/pages/Tutor/slanguage.html

"A Guide for the Unwilling S User" is a brief introduction to using R or S+.

http://www.burns-stat.com/pages/Tutor/unwilling_S.pdf

Outline

- Random portfolios
- The R language
- Demo of random portfolios in R

The demonstration was of the POP commercial software from Burns Statistics as used in R. The exercise was to give some hints of how we would go about validating a risk model with random portfolios.

The data available were 600 days of returns for a cohort of stocks.

The first 500 days of data were used to fit the risk model (variance matrix) and we left the last 100 days as the out-of-sample test period.

The risk model being "tested" was a statistical factor model.

Some random portfolios were generated, and their ex-ante volatilities were compared to the realized volatilities in the out-of-sample period. The realized volatilities were substantially higher. If that consistently happened, then we would be concerned. However, the out-of-sample period starts in May 2007 so we shouldn't be surprised that we get higher volatilities.

There were two graphics functions that were heavily used:

`qqnorm` creates QQ-plots for the normal distribution

`boxplot` creates boxplots.