Stat 107: Introduction to Business and Financial Statistics

Class 7: The Bootstrap and Simulation

Resampling = the Bootstrap

- The Bootstrap is another name for resampling "pick yourself up by your bootstraps…"
- Let the data tell you about itself.

"Pulling oneself up by one's bootstraps"

"I found myself stunned, and in a hole nine fathoms under the grass, when I recovered, hardly knowing how to get out again. Looking down, I observed that I had on a pair of boots with exceptionally sturdy straps. Grasping them firmly, I pulled with all my might. Soon I had hoist myself to the top and stepped out on terra firma without further ado." -- Campaigns and Adventures of Baron Munchausen, 1786.

More Interesting Examples

- We’ve seen that bootstrapping (resampling) replicates a result we know to be true from theory.
- Often in the real world we either don’t know the ‘true’ distributional properties of a random variable…
- …or are too busy to find out.
- This is when bootstrapping really comes in handy.

The Bootstrap in R

- Although resampling/bootstrap code is easy to write, there is a bootstrap routine built into R.
- library(boot)
- The function call:

```
bootstrap(x,f,R=1000)
```

The data

The function to bootstrap

The number of bootstraps

The function to bootstrap

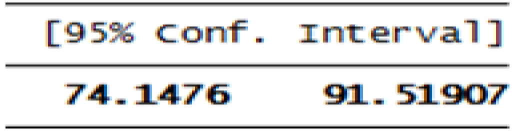
- Need to write a function that takes as arguments d (the data) and i (which rows to grab).
- For bootstrapping the mean, the function would be

```
f=function(d, i)
     {return(mean(d[i]))}
```

The Mean Example Again

- Running this in R

```
> bfit=boot(exam,f,R=1000)
> boot.ci(bfit)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = bfit)

Intervals :
Level      Normal              Basic
95%   (75.45, 89.98 )   (76.33, 90.66 )

Level     Percentile            BCa
95%   (75.00, 89.33 )   (73.27, 88.58 )
```



[95% Conf.	Interval]
74.1476	91.51907

Bootstrap the Correlation

- Start with monthly data

```
getSymbols("MRK",from="2012-01-01")
getSymbols("LULU",from="2012-01-01")
mrk.ret=as.numeric(monthlyReturn(Ad(MRK)))
lulu.ret=as.numeric(monthlyReturn(Ad(LULU)))
```

```
> library(nortest)
> ad.test(mrk.ret)
        Anderson-Darling normality test
data:  mrk.ret
A = 0.18428, p-value = 0.9034

> ad.test(lulu.ret)
        Anderson-Darling normality test
data:  lulu.ret
A = 0.23419, p-value = 0.7819
```

The Usual Interval

- The data appears normally distributed so we will calculate the usual interval

```
> cor.test(mrk.ret,lulu.ret)

        Pearson's product-moment correlation

data:  mrk.ret and lulu.ret
t = -0.82268, df = 43, p-value = 0.4152
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4032802  0.1754632
sample estimates:
       cor
-0.1244821
```

Bootstrap

- Have to write function

```
f <- function(d, i){
        d2 <- d[i,]
        return(cor(d2[,1], d2[,2]))
}
mydata=cbind(mrk.ret,lulu.ret)
bfit=boot(mydata,f,R=1000)
boot.ci(bfit)
```

Similar!

```
Intervals :
Level      Normal                    Basic
95%   (-0.4243,  0.1680 )   (-0.4428,  0.1643 )

Level     Percentile                  BCa
95%   (-0.4132,  0.1938 )   (-0.4283,  0.1579 )
```

Now Daily Data

- Daily data is not normal

```
> getSymbols("MRK",from="2012-01-01")
[1] "MRK"
> getSymbols("LULU",from="2012-01-01")
[1] "LULU"
> mrk.ret=as.numeric(dailyReturn(Ad(MRK)))
> lulu.ret=as.numeric(dailyReturn(Ad(LULU)))
> ad.test(mrk.ret)
        Anderson-Darling normality test
data:  mrk.ret
A = 6.5887, p-value = 3.664e-16

> ad.test(lulu.ret)
        Anderson-Darling normality test
data:  lulu.ret
A = 15.78, p-value < 2.2e-16
```

Correlation Confidence Interval

- Takes longer since much more data

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = bfit)

Intervals :
Level      Normal              Basic
95%   ( 0.0067,  0.2160 )   ( 0.0176,  0.2275 )

Level     Percentile            BCa
95%   (-0.0027,  0.2071 )   (-0.0410,  0.1903 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
```

The point is you can’t use the regular interval since not normal data

Remember the Bootstrap

- The bootstrap (resampling) is an incredibly powerful tool to keep around.
- It is easy to explain to people
- Enables one to create confidence intervals for anything you can estimate from the data, (correlation, VaR, Omega, whatever) without worrying about complex math.

Not all statistics behave well when bootstrapped

- Many Ω's "work well" (give good CIs and inferences) in bootstraps: means, variances, typical inferential stats, eigenvalues, etc.
- Some Ω's require larger n's to work well: medians, percentiles
- Some Ω's don't work well at all: min, max, unique values
- How to know which ones will work? The higher the variance and skewness of Ω, the less well bootstraps work. You might try simulating to get an idea about these.

Monte Carlo- Historical Note

- The name Monte Carlo simulation comes from the computer simulations performed during the 1930s and 1940s to estimate the probability that the chain reaction needed for an atom bomb to detonate would work successfully.
- The physicists involved in this work were big fans of gambling, so they gave the simulations the code name Monte Carlo.

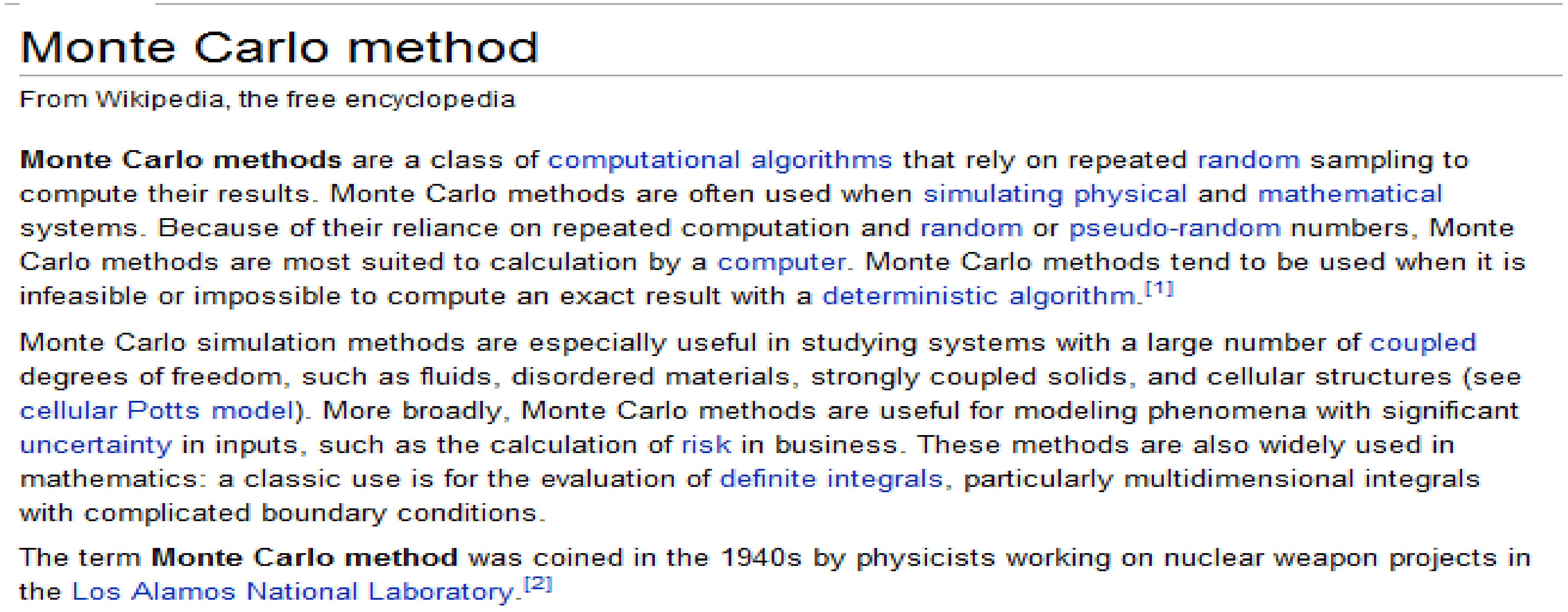
Uses of Monte Carlo Simulation

- Ford, Proctor and Gamble, Pfizer, Bristol-Myers Squibb, and Eli Lilly use simulation to estimate both the average return and the risk factor of new products.
- Proctor and Gamble uses simulation to model and optimally hedge foreign exchange risk.
- Sears uses simulation to determine how many units of each product line should be ordered from suppliers—for example, the number of pairs of Dockers trousers that should be ordered this year.
- Financial planners use Monte Carlo simulation to determine optimal investment strategies for their clients' retirement.

The Monte Carlo Method

- The method is usually used to understand how a process or estimator will perform “in practice”.
- The Monte Carlo method is often performed when the computational resources of a researcher are more abundant than the researchers mental resources.

Wikipedia



Monte Carlo method

From Wikipedia, the free encyclopedia

Monte Carlo methods are a class of computational algorithms that rely on repeated random sampling to compute their results. Monte Carlo methods are often used when simulating physical and mathematical systems. Because of their reliance on repeated computation and random or pseudo-random numbers, Monte Carlo methods are most suited to calculation by a computer. Monte Carlo methods tend to be used when it is infeasible or impossible to compute an exact result with a deterministic algorithm.[1]

Monte Carlo simulation methods are especially useful in studying systems with a large number of coupled degrees of freedom, such as fluids, disordered materials, strongly coupled solids, and cellular structures (see cellular Potts model). More broadly, Monte Carlo methods are useful for modeling phenomena with significant uncertainty in inputs, such as the calculation of risk in business. These methods are also widely used in mathematics: a classic use is for the evaluation of definite integrals, particularly multidimensional integrals with complicated boundary conditions.

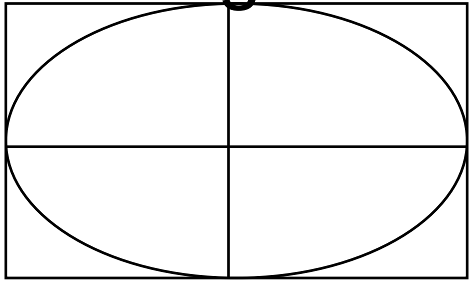
The term **Monte Carlo method** was coined in the 1940s by physicists working on nuclear weapon projects in the Los Alamos National Laboratory.[2]

Example: Estimating π

- Did you know that you can calculate π (3.14159..) by throwing darts at a dartboard ?
- Unfortunately, you have to be pretty bad at darts for it to work, though.

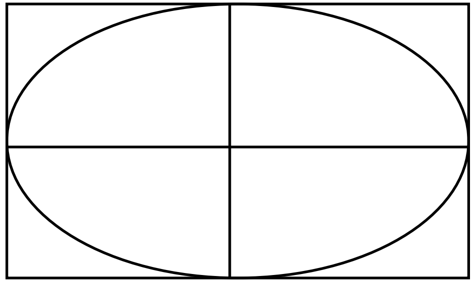
Geometry Review

- Suppose we have a circle of radius 1, with a center at (0,0).
- Then the unit circle is given by the equation $x^2+y^2=1$.

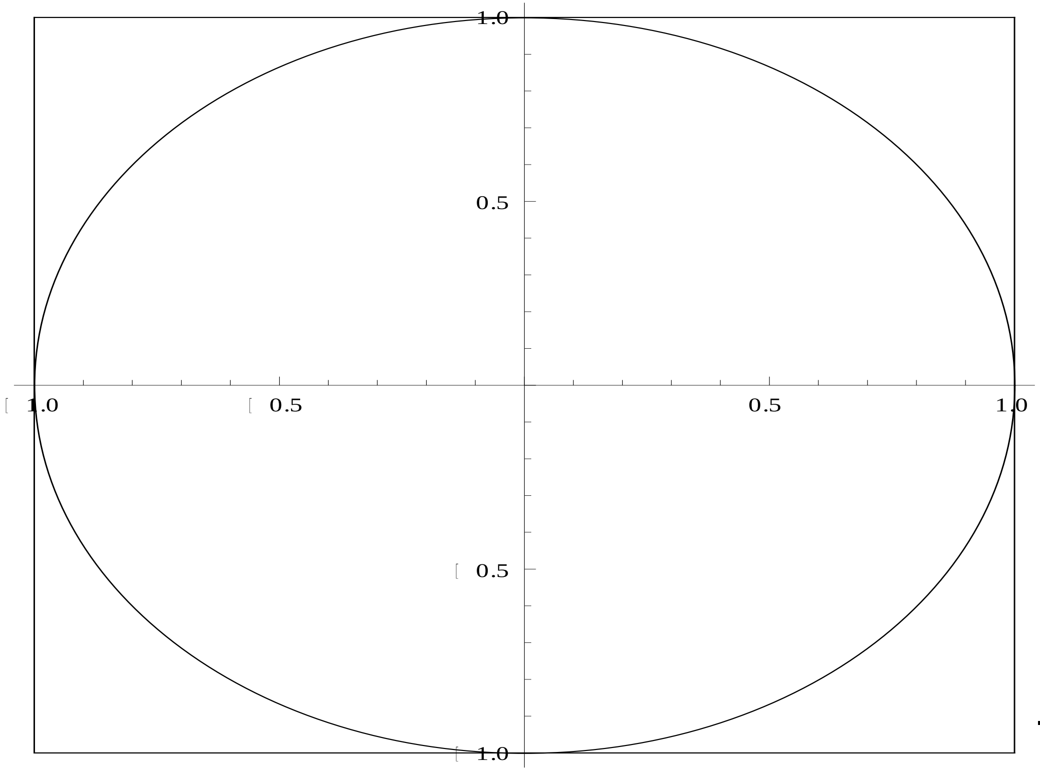


The Set Up

- Suppose our dartboard is a 2x2 square and we inscribe a circle of radius 1 inside the square, with a center at (0,0).



Computing π: Unit circle inscribed in unit square



- The square has area 4
- Unit circle has area π

The Algorithm

- We randomly throw darts at this square and keep track of how many land inside the circle. That would give us a relative estimation of what percentage of the box is in the circle.
- The area of the box is 4 and the area of the circle is π.
- So the ratio of circle area to box area is $\pi/4$.
- Hence, the percentage of darts that land inside the circle should be approximately $\pi/4$.

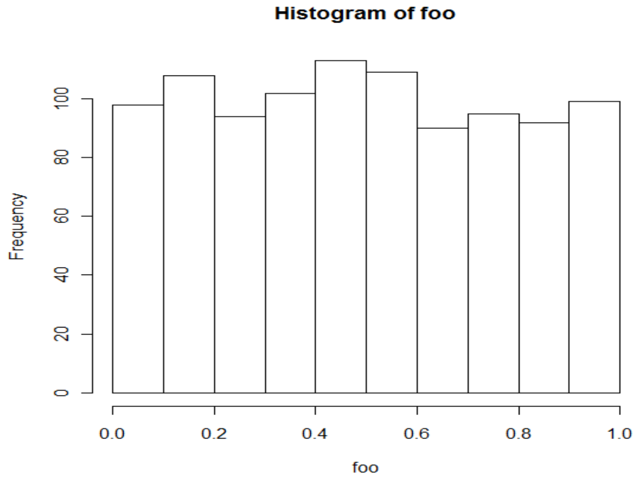
The runif() command

- To simulate throwing darts at the dartboard, we need to learn about the R command runif():

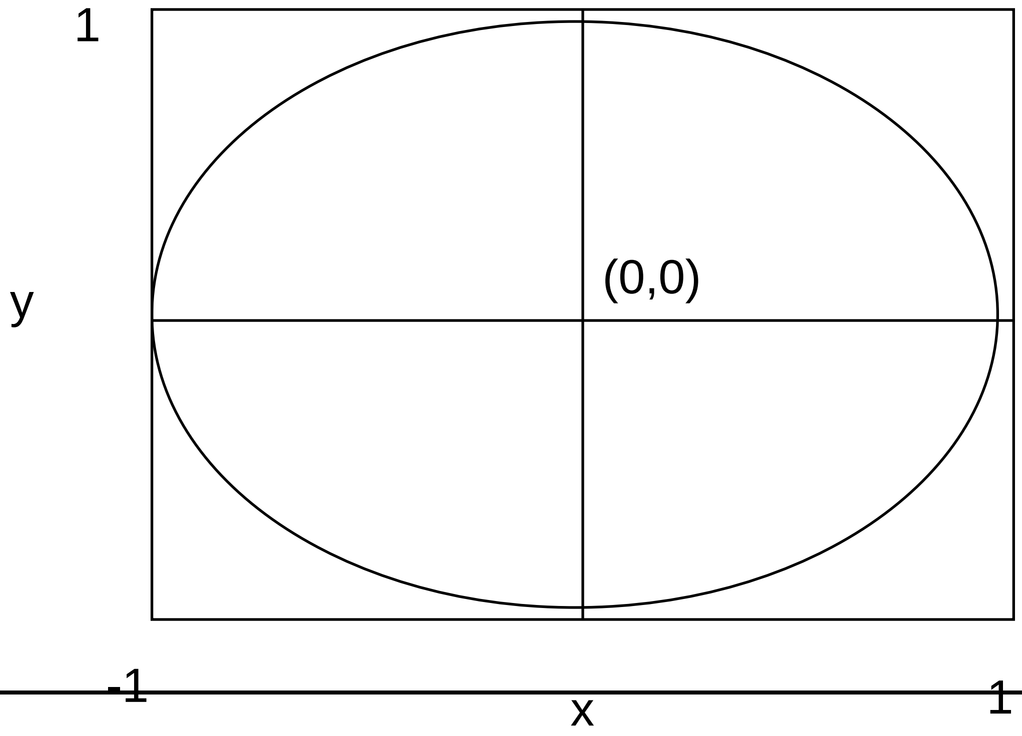
> **R Command**: runif(n)
> uniform random number generator that returns n random numbers between 0 and 1.

Examples

```
> runif(5)
[1] 0.7000905 0.6717379 0.9857697 0.6081811 0.8611683
>
> runif(5)
[1] 0.1719788 0.5086680 0.6567634 0.1922734 0.8315596
>
> foo=runif(1000)
> hist(foo)
>
```



Our circle is as follows:



How do we simulate darts thrown at this board ?

The location of the dart is given by its (x,y) point.

Ponder

- What do the following R commands
- produce?

```
x = 2*runif(1)-1
y = 2*runif(1)-1
```

- Lets plot this and see

Some R Code

- Consider the following code

```
n=500
x=matrix(nrow=n,ncol=1)
y=matrix(nrow=n,ncol=1)

for(i in 1:n) {
  x[i] = 2*runif(1)-1
  y[i] = 2*runif(1)-1
}
```

Let’s run it

- I cut and paste it into R and get:

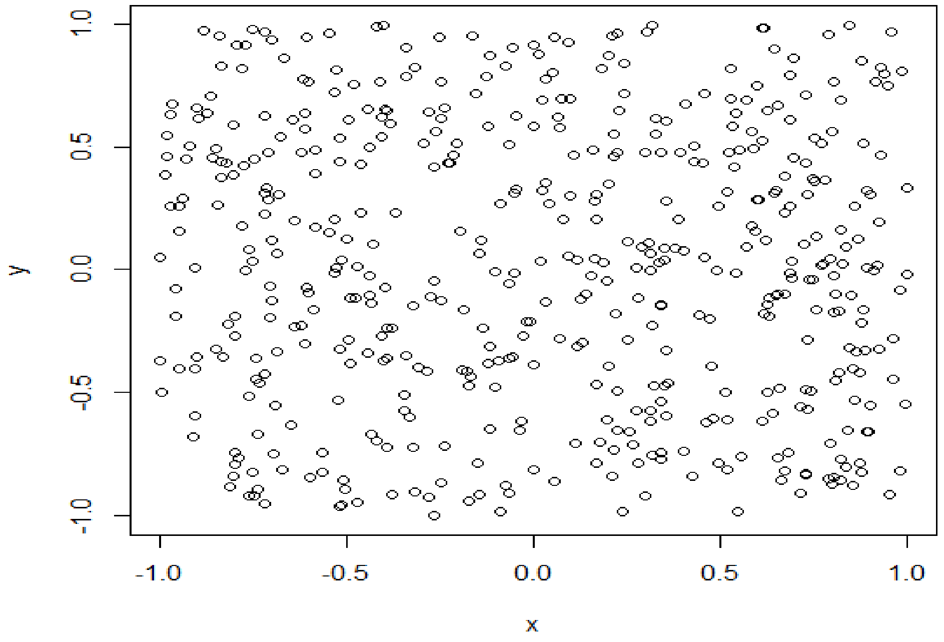
```
> n=500
> x=matrix(nrow=n,ncol=1)
> y=matrix(nrow=n,ncol=1)
>
> for(i in 1:n) {
+   x[i] = 2*runif(1)-1
+   y[i] = 2*runif(1)-1
+ }
>
```

.

That is, I get nothing….or do I????
New variables x and y are created

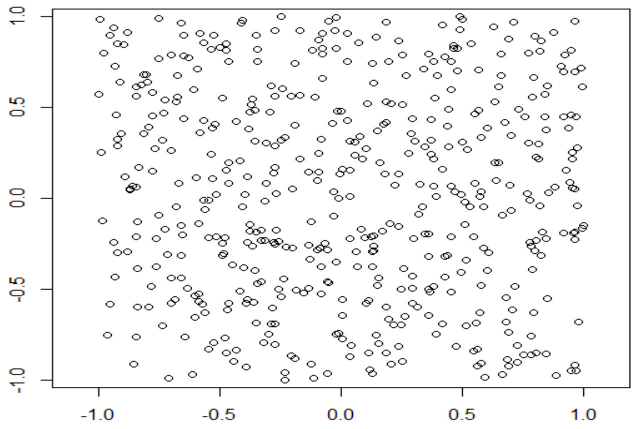
Plot the new variables

- plot(x,y)



Different plot command

- eqscplot(cbind(x,y))
- Makes the x and y axis the exact same length (1 inch in x direction=1 inch in y direction)

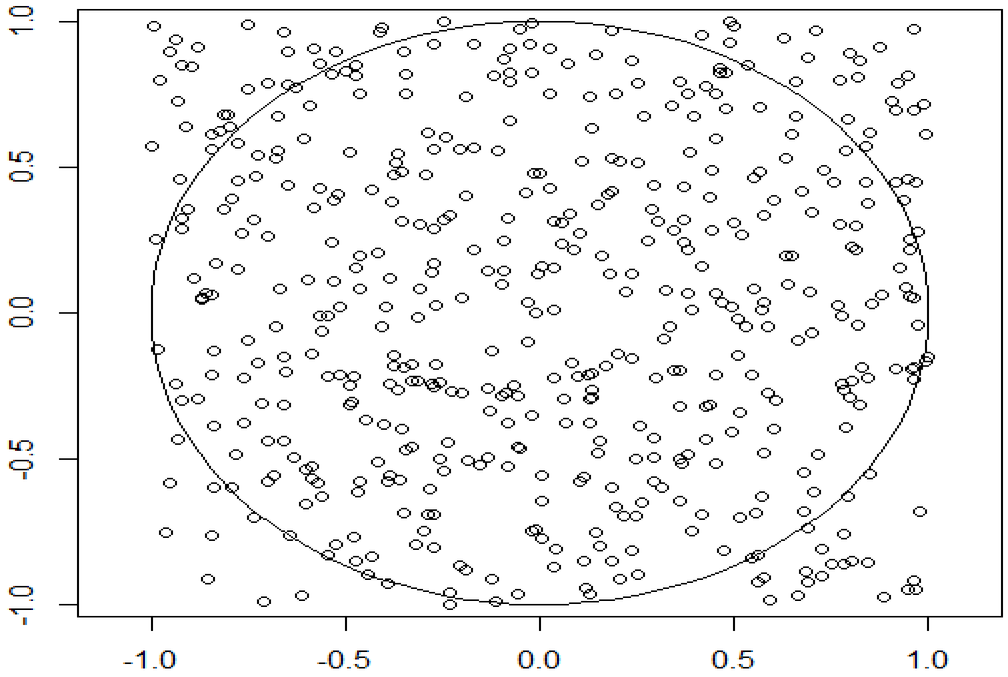


Add a circle

- This is a trick. We need a new package called “plotrix” and can then use its draw.circle routine.

- `library(plotrix)`
- `draw.circle(0,0,1)`

The Graph



So far the code looks to be working

Code Fragment

- Examine the following code fragment

```
x = 2*runif(1)-1
y = 2*runif(1)-1
if (x*x+y*y<=1) numincircle = numincircle+1
```

- What does this code do ?

The Complete Function

```
simulpi = function(n) {

numincircle = 0

for(i in 1:n) {
   x = 2*runif(1)-1
   y = 2*runif(1)-1
   if (x*x+y*y<1) numincircle = numincircle+1
}
cat("Number of simulations = ",n,"\n")
cat("Estimate of PI = ",4*numincircle/n,"\n")
}
```

Keep count of darts inside circle

do n simulations

Output

```
> simulpi(10)
Number of simulations =  10
Estimate of PI =  3.2
> simulpi(100)
Number of simulations =  100
Estimate of PI =  3.04
> simulpi(1000)
Number of simulations =  1000
Estimate of PI =  3.136
> simulpi(1000)
Number of simulations =  1000
Estimate of PI =  3.184
> simulpi(100000)
Number of simulations =  1e+05
Estimate of PI =  3.1472
```

1000 iterations-why aren’t the values the same each time ?

Example: Uncertain Profits

- A firm is introducing a product into a new market. As a marketing manager, you are trying to estimate the profit that will result from the product introductions. The items on which profit depends are
  - Sales in units
  - Price per unit
  - Unit cost: The marginal cost per unit of production, marketing and sales
  - Fixed costs: Fixed overhead, advertising and so on. These are known to be $30,000

Profit = Sales *(Price-Unit Cost) - Fixed Cost

Market Scenarios

- Because this is a new market, there is significant uncertainty.
- It is generally believed that either a low or high volume market can occur with equal likelihood.

Market Scenarios

- If there is a low volume market, sales of roughly 60,000 units are expected at a price of $10 per unit.
- If there is a high volume market, the good news is that your company's sales are expected to be roughly 100,000 units. The bad news is that under this scenario, the market is so hot that it brings in competition. This, in turn, drives the expected price down to $8 per unit.

Summary

- These market scenarios are summarized in the following table:

	Low Volume	High Volume	Average
Probability	50%	50%	
Units	60,000	100,000	80,000
Price	$10.00	$8.00	$9.00

Unit Cost

- The VP of production believes that the cost per unit will be $7.50.
- But you have been advised that depending on the cost of raw materials and actual production experience, this cost might be as low as $6.00 or as high as $9.00. This uncertainty is summarized in the following table:

Low	Most likely	High
$6.00	$7.50	$9.00

Enter the Boss

- Typically, employees entrusted to work on problems like these have bosses who ask questions such as the following:
- The Boss: What is the profit going to be for the new product ?
- You: I’m not sure what profit’s going to be because I don’t know what sales or prices are going to be.
- The Boss: What are we paying you for you lousy Harvard grad ! GIVE ME A NUMBER. I want it now, 15 minutes at the latest.

The Simple Answer

- We know that
- Profit = Sales *(Price-Unit Cost) - Fixed Cost
- Plugging in average values we can give The Boss an answer very quickly.

- Profit = 80000(9-7.5) -30000 = $90,000
- Is this a good result???

Monte Carlo Simulation

- Given the scenarios for demand and cost, we can run a Monte Carlo simulation.
- The basic steps of running a Monte Carlo simulation are
- Build a model of the uncertain situation
- Specify the simulation setting
- Run the simulation and examine the results

Modeling Market Scenarios

- There are two market scenarios, each with a 50% chance of happening.
- To model which one is occurring, we draw a random number between 0 and 1. If the number is less than .5 it is scenario 1; if it is between .5 and 1 it is scenario 2.
- We use the runif() command in R

Code Fragment 1

```
for (i in 1:10) {

which=runif(1)
if (which < 0.5) {
   units=60000
   price=10
} else {
   units=100000
   price=8
}
cat("Sim ",i,"Units = ",units,"Price = ",price,
      "\n")
```

```
}
```

Output

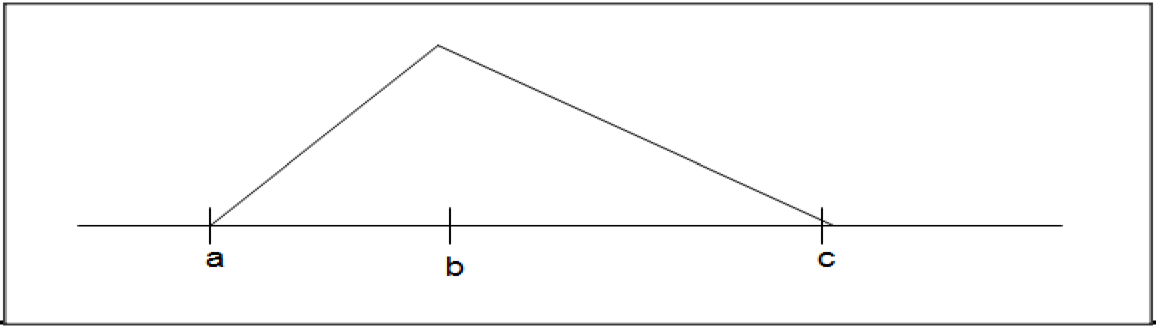
```
Sim  1 Units =  100000 Price =  8
Sim  2 Units =  60000 Price =  10
Sim  3 Units =  60000 Price =  10
Sim  4 Units =  100000 Price =  8
Sim  5 Units =  100000 Price =  8
Sim  6 Units =  100000 Price =  8
Sim  7 Units =  60000 Price =  10
Sim  8 Units =  100000 Price =  8
Sim  9 Units =  100000 Price =  8
Sim  10 Units =  60000 Price =  10
```

Modeling Unit Cost

- This is an unusual probability fact that is useful to know in the business world.
- When you have estimates of a low, most likely and high value for an uncertainty, it is often reasonable to use a random number generator with what is known as a Triangular Distribution.
- (Consultant trick)

The Triangular Distribtuion

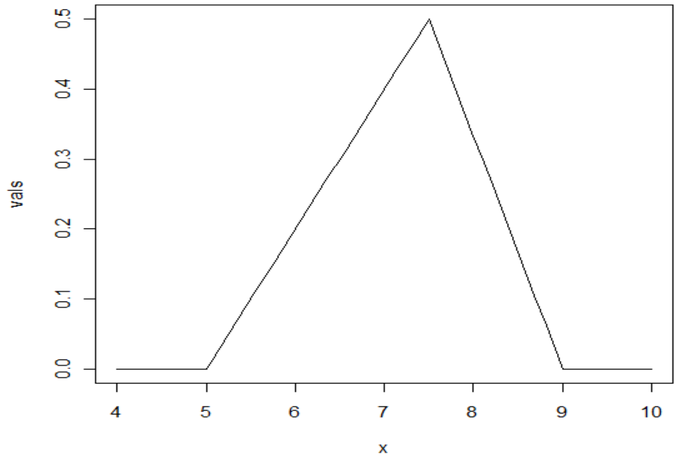
- The triangular distribution is described by 3 values (a,b,c)-the lowest value possible, the most likely value, and the maximum value possible.



Generating Random Numbers

- The R package is called triangle
- install.packages(“triangle”)

```
> x=seq(4,10,.1)
> vals=dtriangle(x,5,9,7.5)
> plot(x,vals,type="l")
```



Code Fragment #2

```
for (i in 1:10) {

which=runif(1)
if (which < 0.5) {
   units=60000
   price=10
} else {
   units=100000
   price=8
}

cost=rtriangle(1,6,9,7.5)

cat("Sim ",i,"Units = ",units,"Price = ",price,
       "Cost =",round(cost,2),"\n")
```

Program Output

```
Sim  1 Units =  100000 Price =  8 Cost = 6.32
Sim  2 Units =  100000 Price =  8 Cost = 6.89
Sim  3 Units =  100000 Price =  8 Cost = 8.49
Sim  4 Units =  100000 Price =  8 Cost = 6.97
Sim  5 Units =  100000 Price =  8 Cost = 7.32
Sim  6 Units =  100000 Price =  8 Cost = 8.34
Sim  7 Units =  100000 Price =  8 Cost = 8.45
Sim  8 Units =  60000 Price =  10 Cost = 7.77
Sim  9 Units =  60000 Price =  10 Cost = 7.33
Sim  10 Units =  100000 Price =  8 Cost = 7.31
```

Putting it all together (fragment #3)

```
for (i in 1:10) {

which=runif(1)
if (which < 0.5) {
   units=60000
   price=10
} else {
   units=100000
   price=8
}

cost=rtriangle(1,6,9,7.5)

profit=units*(price-cost)-30000

cat("Sim ",i,"Units = ",units,"Price = ",price,
          "Cost =",round(cost,2),"\n")
cat("Profit =",round(profit,2),"\n\n")

}
```

Some Output

```
Sim  1 Units =  60000 Price =  10 Cost = 7.52
Profit = 118842.08

Sim  2 Units =  100000 Price =  8 Cost = 6.57
Profit = 112894.61

Sim  3 Units =  60000 Price =  10 Cost = 7.15
Profit = 140743.9

Sim  4 Units =  60000 Price =  10 Cost = 6.83
Profit = 160330.54

Sim  5 Units =  100000 Price =  8 Cost = 8.01
Profit = -31125.26

Sim  6 Units =  100000 Price =  8 Cost = 6.85
Profit = 84909.09

Sim  7 Units =  100000 Price =  8 Cost = 8.66
Profit = -96098.35

Sim  8 Units =  60000 Price =  10 Cost = 7
Profit = 149707.84

Sim  9 Units =  100000 Price =  8 Cost = 7.5
Profit = 20270.83

Sim  10 Units =  60000 Price =  10 Cost = 8.05
Profit = 86798.2
```

Analyze

- We just ran the model 10 times.
- In two instances we lost money !
- There seems to be huge fluctuations as to what the profit for this project will be.
- We will run the simulation 500 times and analyze the resulting output

Code Frag #4

```
profit=1:500

for (i in 1:500) {


which=runif(1)
if (which < 0.5) {
   units=60000
   price=10
} else {
   units=100000
   price=8
}

cost=rtriangle(1,6,9,7.5)

profit[i]=units*(price-cost)-30000

#cat("Sim ",i,"Units = ",units,"Price = ",price,
#         "Cost =",round(cost,2),"\n")
#cat("Profit =",round(profit,2),"\n\n")

}
```

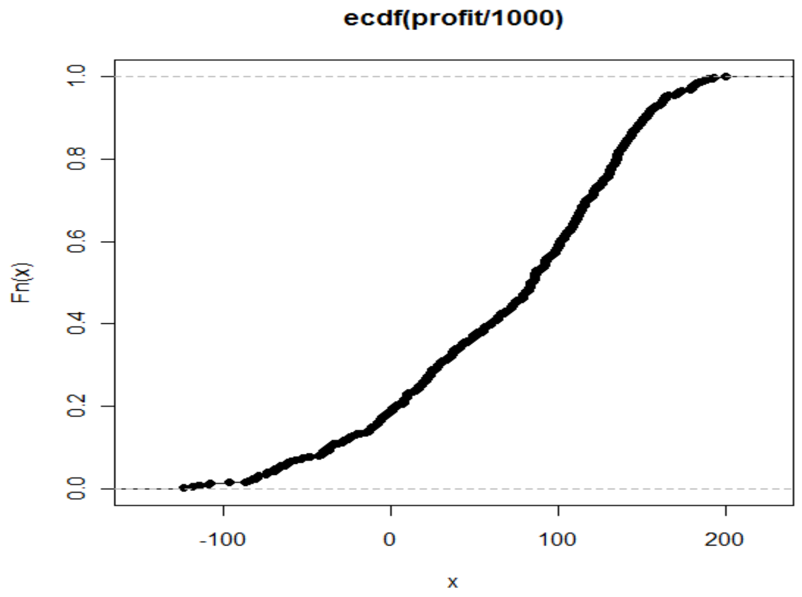
Summarize

- **Uh oh!** Although results will vary from run to run, the average value of profit will almost surely be less than the $90,000 profit you were about to tell the boss a few slides back. In fact, the average profit is around $70,000.
- There is also a lot of variability and a chance you are going to lose money on the project.

```
> summary(profit)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-123500   18350   84560   69820  127400  199900
> sd(profit)
[1] 71486.39
>
```

Empirical Distribution Function

- **An easy to read graphical display for the output is called an empirical CDF (cumulative distribution function):**
- **R command** `plot(ecdf(profit/1000))`



Back to Boss

- You: Well, Boss, if you average out all the things that might happened with this product, you should expect a profit of about $90,000. But that doesn’t tell the whole story….
- Boss: What do you mean not the whole story ? I told you I want a number!
- You: Well, here is a graph with all the possible scenarios. There is a 20% chance the product will lose money.
- Boss: That sucks.
- You: But also a 20% chance the product will make more than $140,000 !
- Boss: Glad we hired someone from Harvard…………...

Stock Returns

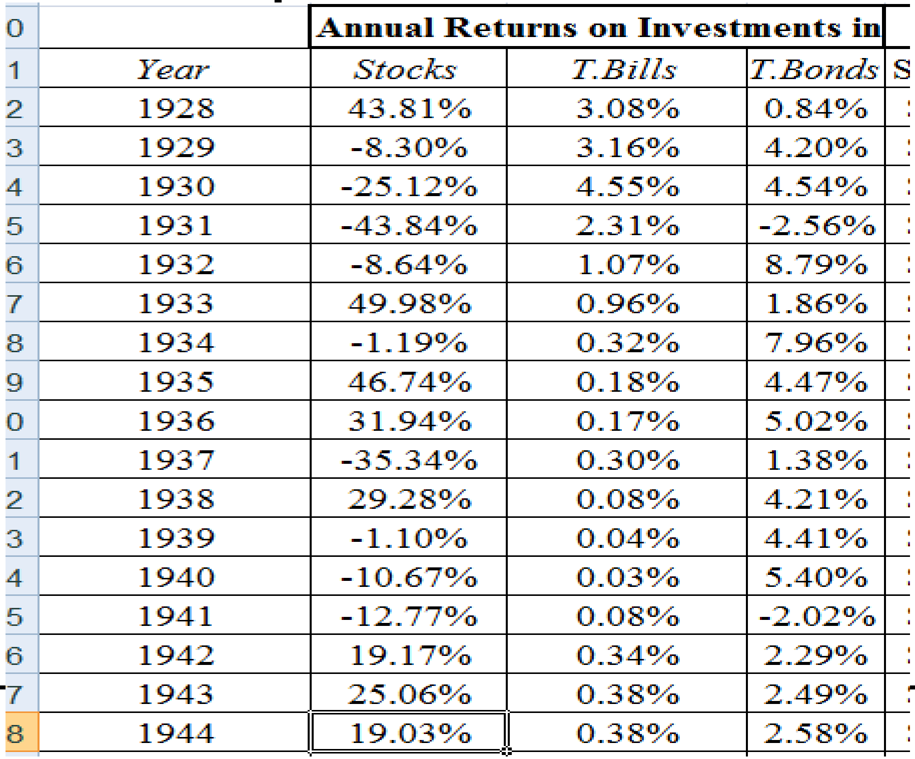
- Suppose you put $1000 into an S&P 500 index fund on January 1. How much will you have at the end of the year?
- We can easily simulate this using Monte Carlo.
- How do we simulate yearly stock returns? There are several methods.

Simulating Stock Returns

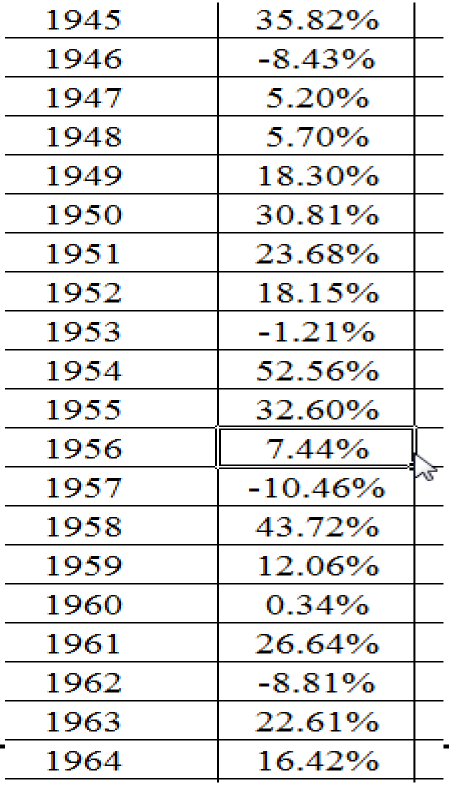
- Assuming normality of returns is not that good an idea since stock returns demonstrate tail behavior, but it is less common than what the normal distribution would dictate.
- We will produce the returns using a procedure called resampling. We have a dataset with the yearly S&P 500 returns from 1926 to 2010.

The Historical Returns

- A snapshot

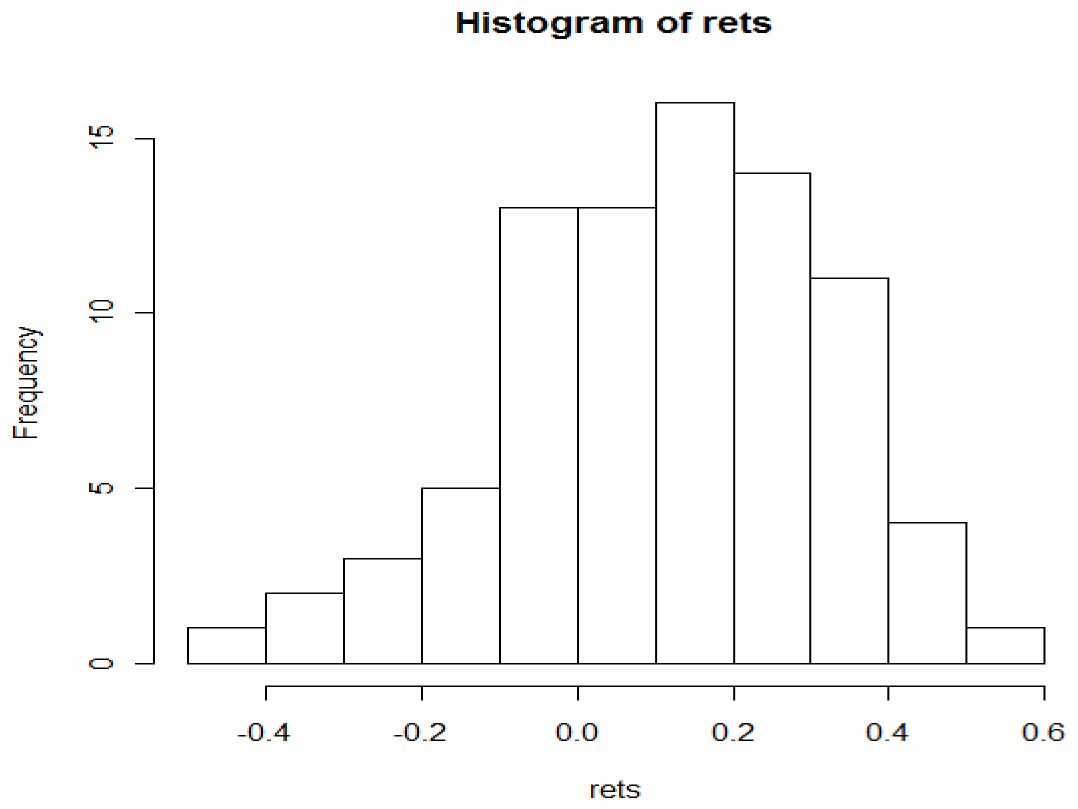


0		Annual Returns on Investments in			
1	*Year*	*Stocks*	*T.Bills*	*T.Bonds*	S
2	1928	43.81%	3.08%	0.84%	
3	1929	-8.30%	3.16%	4.20%	
4	1930	-25.12%	4.55%	4.54%	
5	1931	-43.84%	2.31%	-2.56%	
6	1932	-8.64%	1.07%	8.79%	
7	1933	49.98%	0.96%	1.86%	
8	1934	-1.19%	0.32%	7.96%	
9	1935	46.74%	0.18%	4.47%	
0	1936	31.94%	0.17%	5.02%	
1	1937	-35.34%	0.30%	1.38%	
2	1938	29.28%	0.08%	4.21%	
3	1939	-1.10%	0.04%	4.41%	
4	1940	-10.67%	0.03%	5.40%	
5	1941	-12.77%	0.08%	-2.02%	
6	1942	19.17%	0.34%	2.29%	
7	1943	25.06%	0.38%	2.49%	
8	1944	19.03%	0.38%	2.58%	



1945	35.82%
1946	-8.43%
1947	5.20%
1948	5.70%
1949	18.30%
1950	30.81%
1951	23.68%
1952	18.15%
1953	-1.21%
1954	52.56%
1955	32.60%
1956	7.44%
1957	-10.46%
1958	43.72%
1959	12.06%
1960	0.34%
1961	26.64%
1962	-8.81%
1963	22.61%
1964	16.42%

The Histogram of Historical Returns



Resampling

- To produce a return in the future for the index, we resample from our 83 historical observations-that is, we just randomly pick one out of the 83 past yearly returns.
- This scheme doesn’t build in any ideas like good years are more likely to be followed by good years, etc… but it is a technique used by several software packages that do financial planning.
- To do this in R, we use the sample command

The SAMPLE command

- The command we use is

```
sample(x, size=<<see below>>, replace=F)
```

Collection of items to sample from

How many times to sample

Sample with or without replacement

Example Output

```
> sample(rets,1)
[1] 0.2380297
> sample(rets,1)
[1] 0.09967052
> sample(rets,1)
[1] 0.06146142
> sample(rets,1)
[1] 0.3023484
> sample(rets,1)
[1] -0.01188566
> sample(rets,1)
[1] -0.01208205
```

The Monte Carlo

- We are now ready to code

How many iterations

```
year.ret = function(rets,howmany) {

        store = 1:howmany

        for(i in 1:howmany)
         store[i] = 1000*(1+sample(rets,1))

        return(store)
}
```

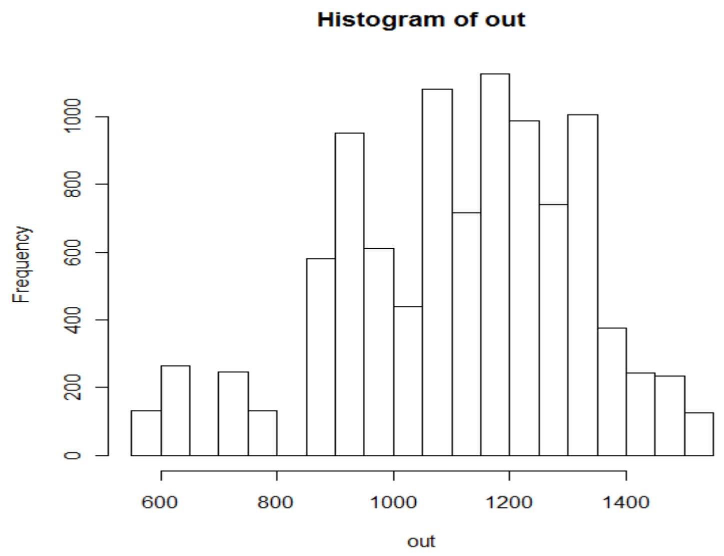
Creates a place to store our results

Running the Monte Carlo

- Cut and paste the code into R

```
> out=year.ret(rets,10000)
> hist(out)
```

The output is not normal.
Should it be?



Summarize the Results

```
> summary(out)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
  561.6    969.4   1142.0   1113.0   1259.0   1526.0 
>
```

The probability of making a positive return is

```
> sum(out>1000)/10000
[1] 0.708
```

This makes sense since

```
> sum(rets>0)/83
[1] 0.7108434
```

71% of the historical returns are positive so it makes sense that we made more than $1000 71% of the time.

Convert to CAGR…What is CAGR?

- CAGR=compound annual growth rate
- The compound annual growth rate is calculated by taking the nth root of the total percentage growth rate, where n is the number of years in the

$$\text{CAGR} = \left(\frac{\text{Ending Value}}{\text{Beginning Value}}\right)^{\left(\frac{1}{\text{\# of years}}\right)} - 1$$

Can Convert to CAGR if desired

- Recall CAGR

$$\text{CAGR} = \left(\frac{\text{Ending Value}}{\text{Beginning Value}}\right)^{\left(\frac{1}{\#\text{ of years}}\right)} - 1$$

```
> summary(out)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  561.6   969.4  1142.0  1113.0  1259.0  1526.0
>
```

CAGR= - 43.8%

CAGR=52.6%

- But this isn't an interesting example since it is just what happens in one year.

Growth over 30 years

- We can do the same idea for 30 years in a row

```
money = 1000
for(i in 1:30) {
  ret = sample(rets,size=1)
  money = money*(1+ret)
}
cat("In 30 years you have ",round(money),"\n")
```

Lets make it into a function

```
mysim = function(numyears) {

money = 1000
for(i in 1:numyears) {
  ret = sample(rets,size=1)
  money = money*(1+ret)
}
cat("In ",numyears," years you have ",round(money),"\n")
}
```

Running this code in R

```
> mysim(30)
In 30 years you have  66089
> mysim(30)
In 30 years you have  8522
> mysim(30)
In 30 years you have  20847
> mysim(30)
In 30 years you have  38044
> mysim(30)
In 30 years you have  22745
> mysim(30)
In 30 years you have  148696
>
```

Keeping Track of the Results

```
simul=function(rets,howmany) {
values = 1:howmany
for(j in 1:howmany) {

   money = 1000
   for(i in 1:30) {
      ret = sample(rets,size=1)
      money = money*(1+ret)
}
values[j] = money
}

return(values)
}
```

This creates a vector to store our results

This double looping makes the problem difficult to do in Excel

Run the code in R

- Cut and paste it into R

```
> summary(out)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   249.1   7229.0  15370.0  24810.0  30870.0 489900.0
>
```

CAGR= 9.5%

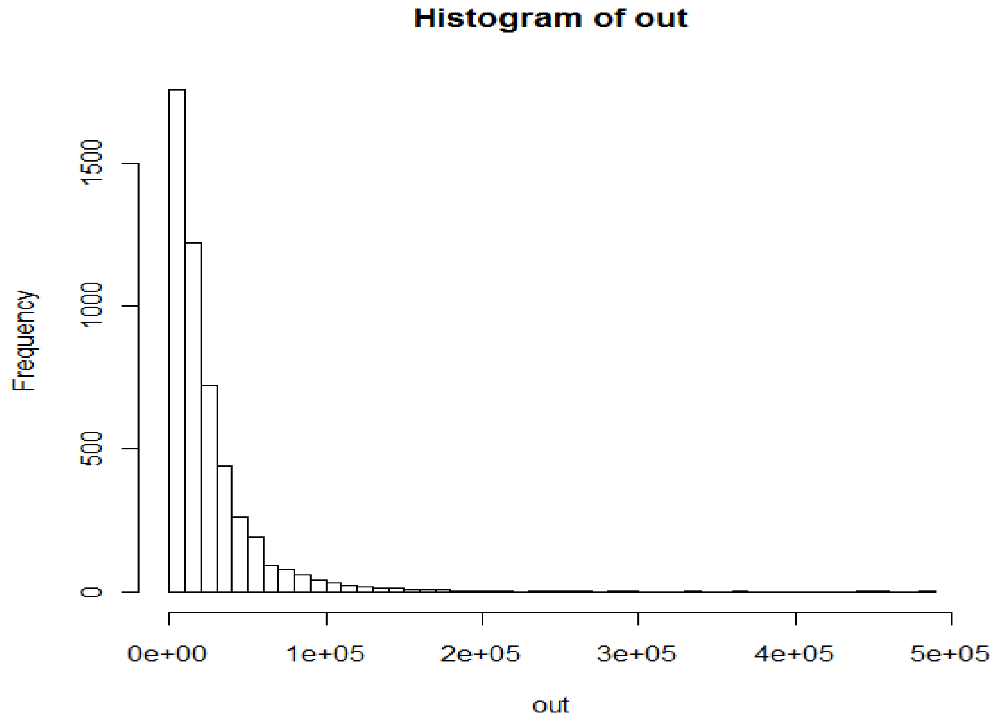
```
> (15370/1000)^(1/30) - 1
[1] 0.09535727
>
```

CAGR=22.9%

```
> (489900/1000)^(1/30) - 1
[1] 0.229335
```

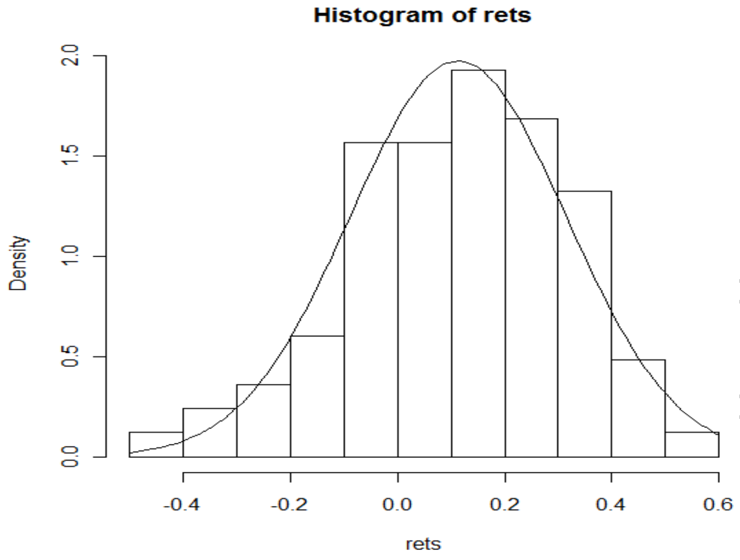
Run the code in R

```
hist(out,breaks=50)
```



Go Normal

- What if we wanted to assume the returns were normal?



Sample statistics

```
> mean(rets)
[1] 0.1131480
> sd(rets)
[1] 0.2021068
```

The rnorm command

```
simul=function(howmany) {
values = 1:howmany
for(j in 1:howmany) {

   money = 1000
   for(i in 1:30) {
      ret=rnorm(1,mean=.1131,s=.2021)
      money = money*(1+ret)
}
values[j] = money
}

return(values)
}
```

Different Results (of course)

Normal

```
> summary(out)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
   119.2   7360.0  14490.0  24240.0  28670.0 454000.0 
>
```

Resampled

```
> summary(out)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
   249.1   7229.0  15370.0  24810.0  30870.0 489900.0 
>
```

- Which model for returns allows for slightly more upside?
- Which is more realistic, which would we show a client?
- What would logspline routine show?

Go Semiparametric

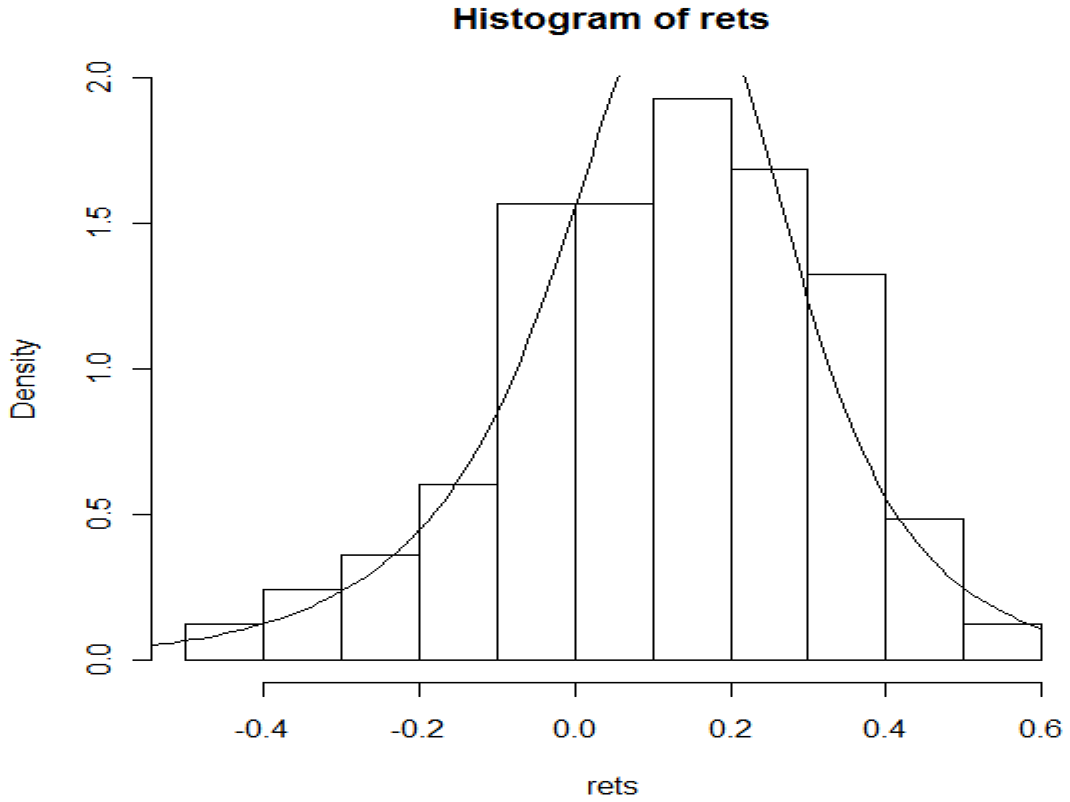
- Density Estimation refers to determining a probability density for a given set of data.
- This can be as simple as assuming the data is normal, and using the sample mean and standard deviation as parameters.
- Or it can involve some fancy math.
- We use the logspline package.

The historical stock returns

- Just a few lines of code

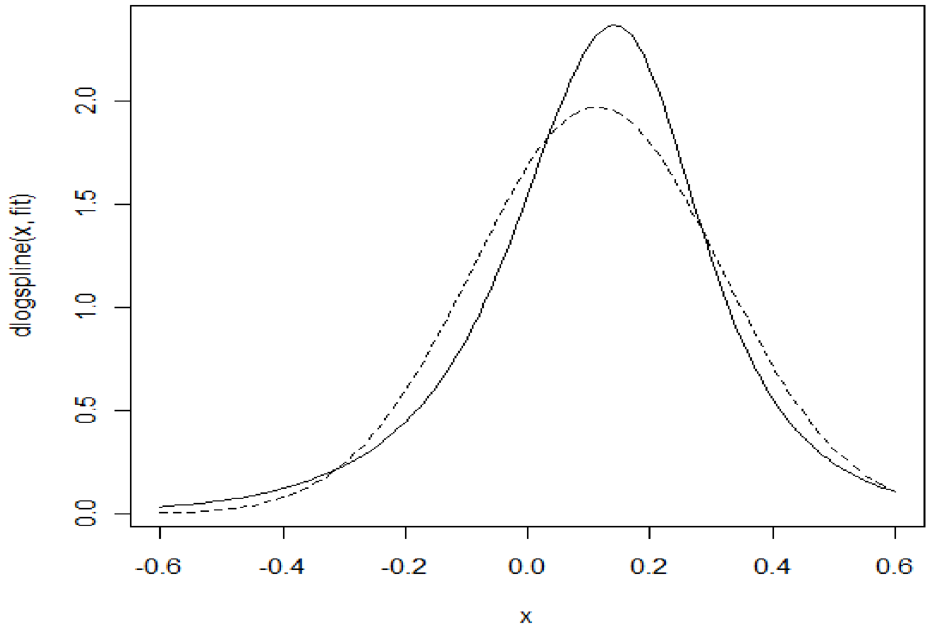
```
➢fit=logspline(rets)


> hist(rets,probability=TRUE)
> x=seq(-.6,.6,.01)
> lines(x,dlogspline(x,fit))
```

The resulting density



Compare with the Normal

- Longer tail



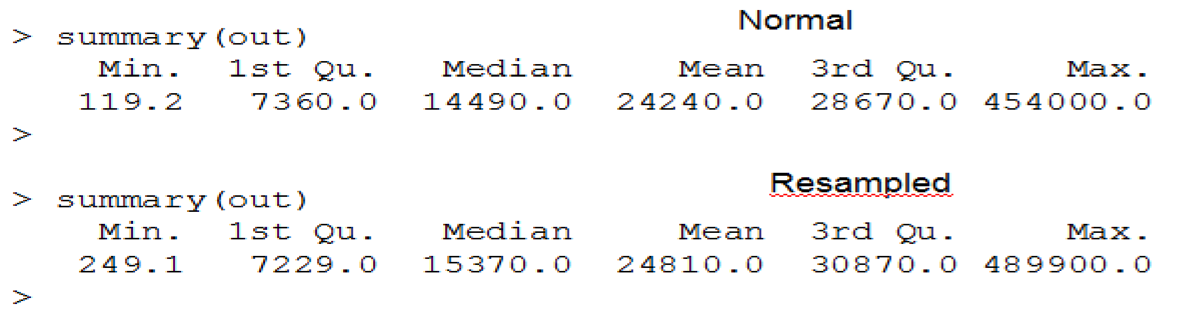
Can draw from this density

- A powerful ability of the logspline routine is that one can draw random values from the fitted density function.
- This is easily done using the rlogspline routine.

Another Simulation

```
fit = logspline(rets)

simul=function(fit,howmany) {
values = 1:howmany
for(j in 1:howmany) {

   money = 1000
   for(i in 1:30) {
      ret=rlogspline(1,fit)
      money = money*(1+ret)
}
values[j] = money
}

return(values)
}
```

Results



```
> summary(out)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
  119.2   7360.0  14490.0  24240.0  28670.0 454000.0
>
```

```
> summary(out)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
  249.1   7229.0  15370.0  24810.0  30870.0 489900.0
>
```

- More long tailed behavior

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -41290    6102   14000   24950   29360  485700
```

```
> plogspline(-.3,fit)
[1] 0.03610169

> pnorm(-.3,mean(rets),sd(rets))
[1] 0.02046658
```

Almost double the probability of a return being less than -0.3!

Example: Retirement Planning (hw)

- Joe Renk is a 35 year old freelance writer with $30,000 in an IRA. He figures he can put in $2000 (the maximum allowed every year) for the next 30 years.
- John wants to have a simple portfolio so he decides to put all his money into an S&P500 index fund, which has an annual management fee of 0.5%.
- How much money will John have in 30 years ?
- What is the probability he is a millionaire ?

A code fragment (modify for hw)

```
money = 30000
for(i in 1:30) {
ret = sample(rets,size=1)
money = (money+2000)*(1+ret)
fee = .005*money
money = money-fee
}
cat("In 30 years Joe has ",round(money),"\n")
```

Cut and pasting this into R, you get results such as

```
In 30 years Joe has  197919
In 30 years Joe has  364987
 In 30 years Joe has  1040814
```

Run this code many times

```
values = 1:1000
for(j in 1:1000) {

money = 30000
for(i in 1:30) {
  ret = sample(rets,size=1)
  money = (money+2000)*(1+ret)
  fee = .005*money
  money = money-fee
}
values[j] = money
}
```

Results:

```
> summary(values)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  36930  207900  353700  501400  608200 9083000
> length(values)
[1] 1000
> sum(values >= 1000000)
[1] 108
> 108/1000
[1] 0.108
```

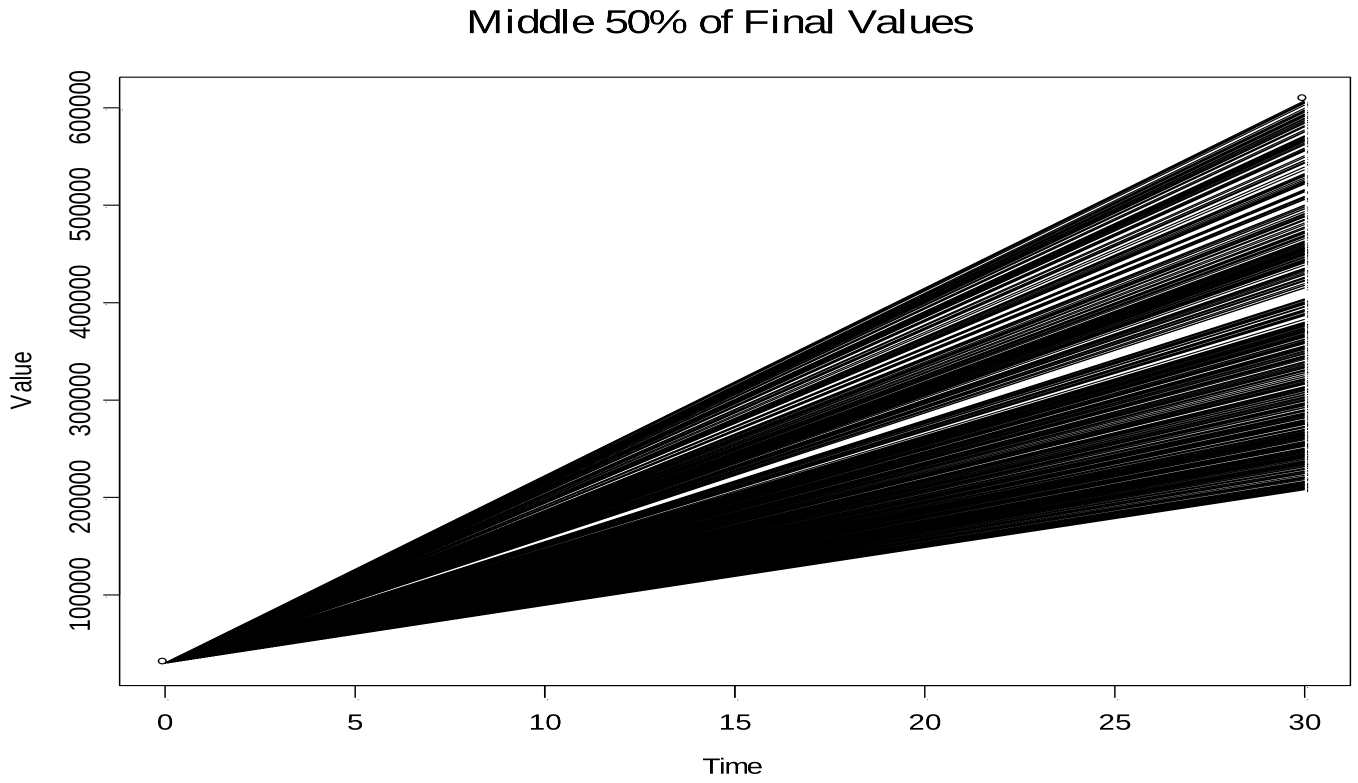
11% chance john will have a million or more.

```
> sum(values<100000)
[1] 65
```

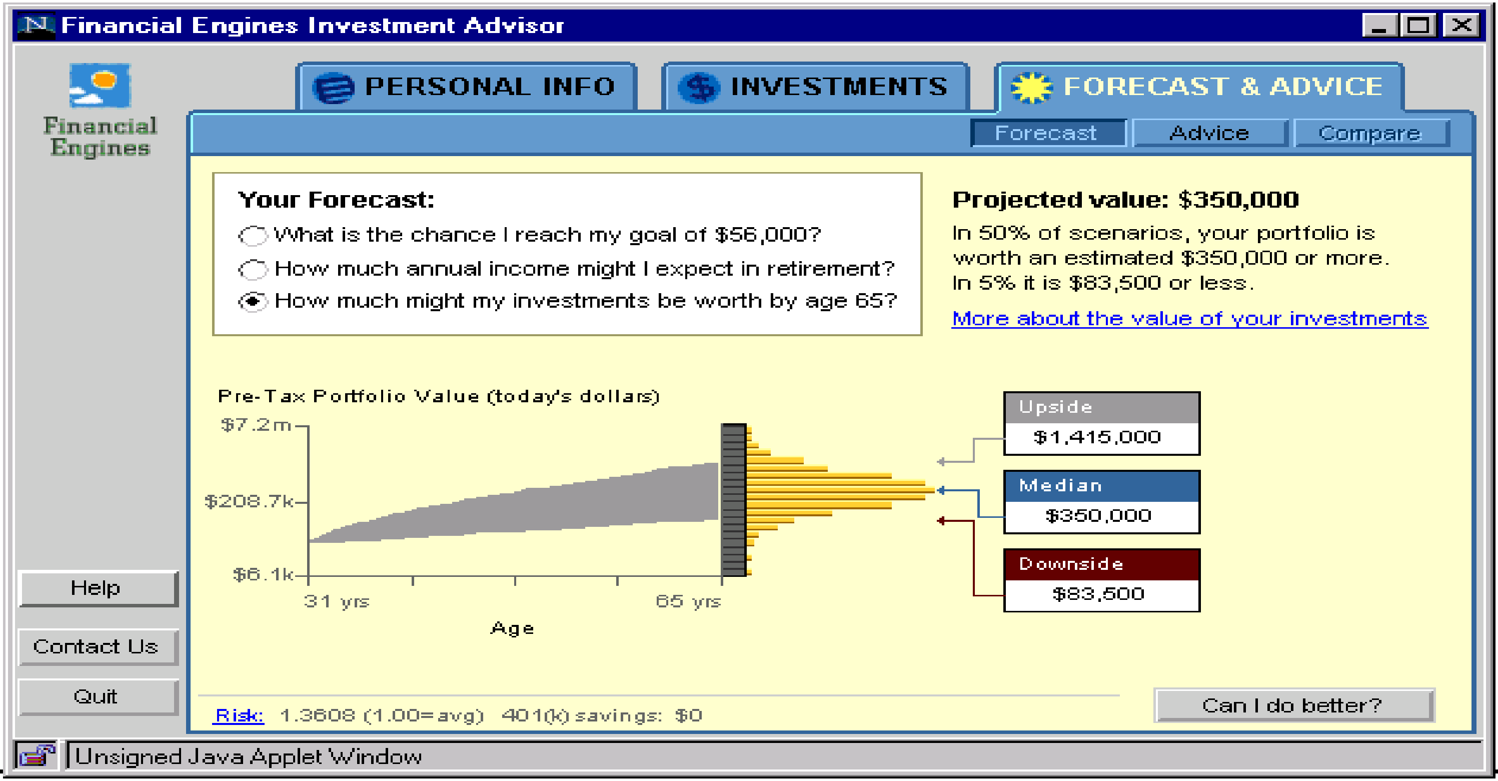
6.5% chance john will have less than 100000.

```
> sum(values>1400000)
[1] 46
```

4.6% chance john will have more than 1.4 million


Middle 50% of Final Values
Value
Time
100000
200000
300000
400000
500000
600000
0
5
10
15
20
25
30

Our numbers are close to financialengines.com, a website founded by Nobel Prize winner Bill Sharpe:



Your Forecast:

( ) What is the chance I reach my goal of $56,000?
( ) How much annual income might I expect in retirement?
(●) How much might my investments be worth by age 65?

Projected value: $350,000

In 50% of scenarios, your portfolio is worth an estimated $350,000 or more. In 5% it is $83,500 or less.

More about the value of your investments

Pre-Tax Portfolio Value (today's dollars)

Upside	$1,415,000
Median	$350,000
Downside	$83,500

Risk: 1.3608 (1.00=avg) 401(k) savings: $0