COUNT BAYESIE

PROBABLY A PROBABILITY BLOG

BLOG ALL POSTS ABOUT BOOKS SUBSCRIBE

6 Neat Tricks with Monte Carlo Simulations

MARCH 24, 2015

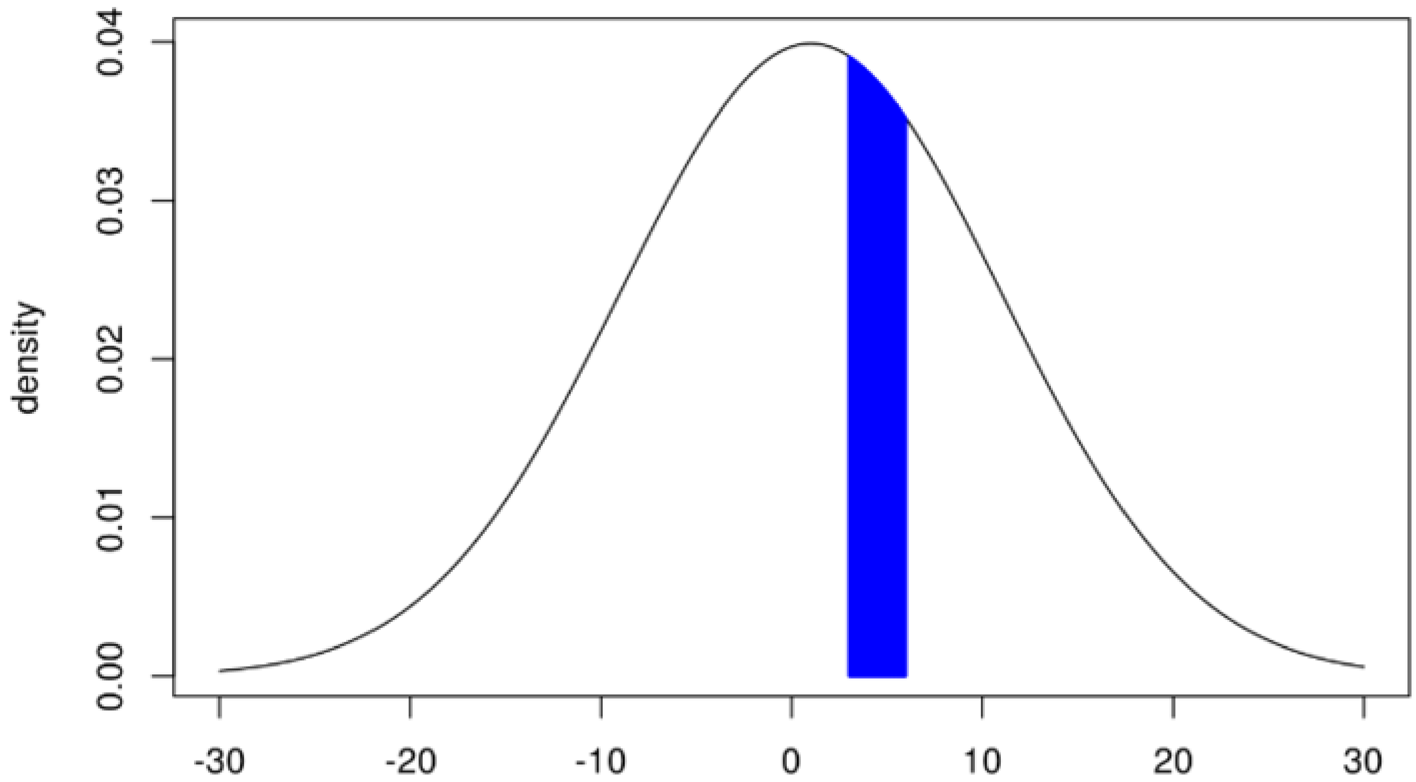
If there is one trick you should know about probability, its how to write a Monte Carlo simulation. If you can program, even just a little, you can write a Monte Carlo simulation. Most of my work is in either R or Python, these examples will all be in R since out-of-the-box R has more tools to run simulations. The basics of a Monte Carlo simulation are simply to model your problem, and than randomly simulate it until you get an answer. The best way to explain is to just run through a bunch of examples, so let's go!

Integration

We'll start with basic integration. Suppose we have an instance of a Normal distribution with a mean of 1 and a standard deviation of 10. Then we want to find the integral from 3 to 6

$$\int_3^6 \frac{1}{10\sqrt{2\pi}} e^{-\frac{(x-1)^2}{2 \cdot 10^2}} dx$$

as visualized below



x

In blue is the area we wish to integrate over

We can simply write a simulation that samples from this distribution 100,000 times and see how many values are between 3 and 6.

```
runs <- 100000
sims <- rnorm(runs,mean=1,sd=10)
mc.integral <- sum(sims >= 3 & sims <= 6)/runs
```

The result we get is:

mc.integral = 0.1122

Which isn't too far off from the 0.112203 that Wolfram Alpha gives us.

Approximating the Binomial Distribution

We flip a coin 10 times and we want to know the probability of getting more than 3 heads. Now this is a trivial problem for the Binomial distribution, but suppose we have forgotten about this or never learned it in the first place. We can easily solve this problem with a Monte Carlo Simulation. We'll use the common trick of representing tails with 0 and heads with 1, then simulate 10 coin tosses 100,000 times and see how often that happens.

```
runs <- 100000
#one.trail simulates a single round of toss 10 coins
#and returns true if the number of heads is > 3
one.trial <- function(){
  sum(sample(c(0,1),10,replace=T)) > 3
}
#now we repeat that trial 'runs' times.
mc.binom <- sum(replicate(runs,one.trial()))/runs
```

For our ad hoc Binomial distribution we get
mc.binom = 0.8279

Which we can compare to R's builtin Binomial distribution function
pbinom(3,10,0.5,lower.tail=FALSE) = 0.8281

Approximating Pi

Next we'll move on to something a bit trickier, approximating Pi!

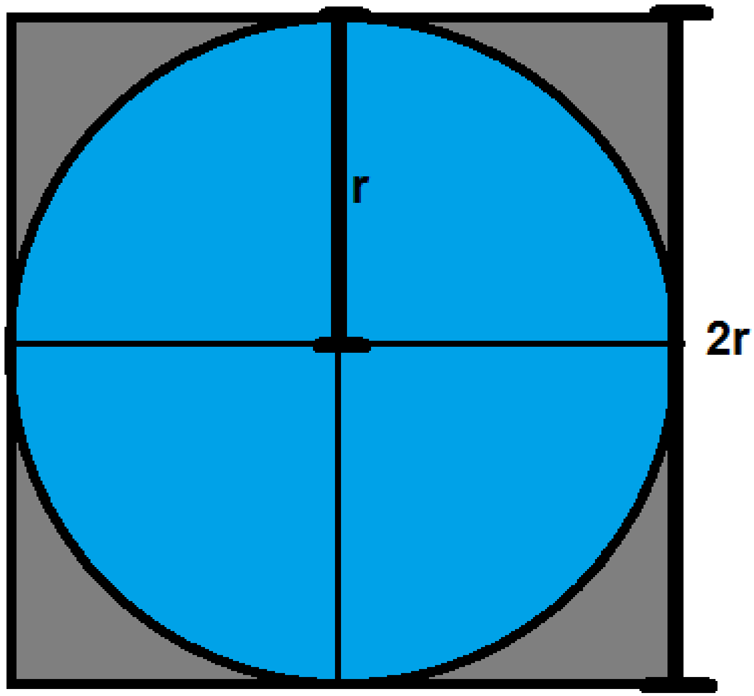
We'll start by refreshing on some basic facts. The area of a circle is

$$A_{\text{circle}} = \pi r^2$$

and if we draw a square containing that circle its area will be

$$A_{\text{square}} = 4r^2$$

this is because each side is simply $2r$ as can be seen in this image:



Basic properties of a circle

Now how do we get π? The ratio of the area of the circle to the area of the square is

$$\frac{\pi r^2}{4r^2}$$

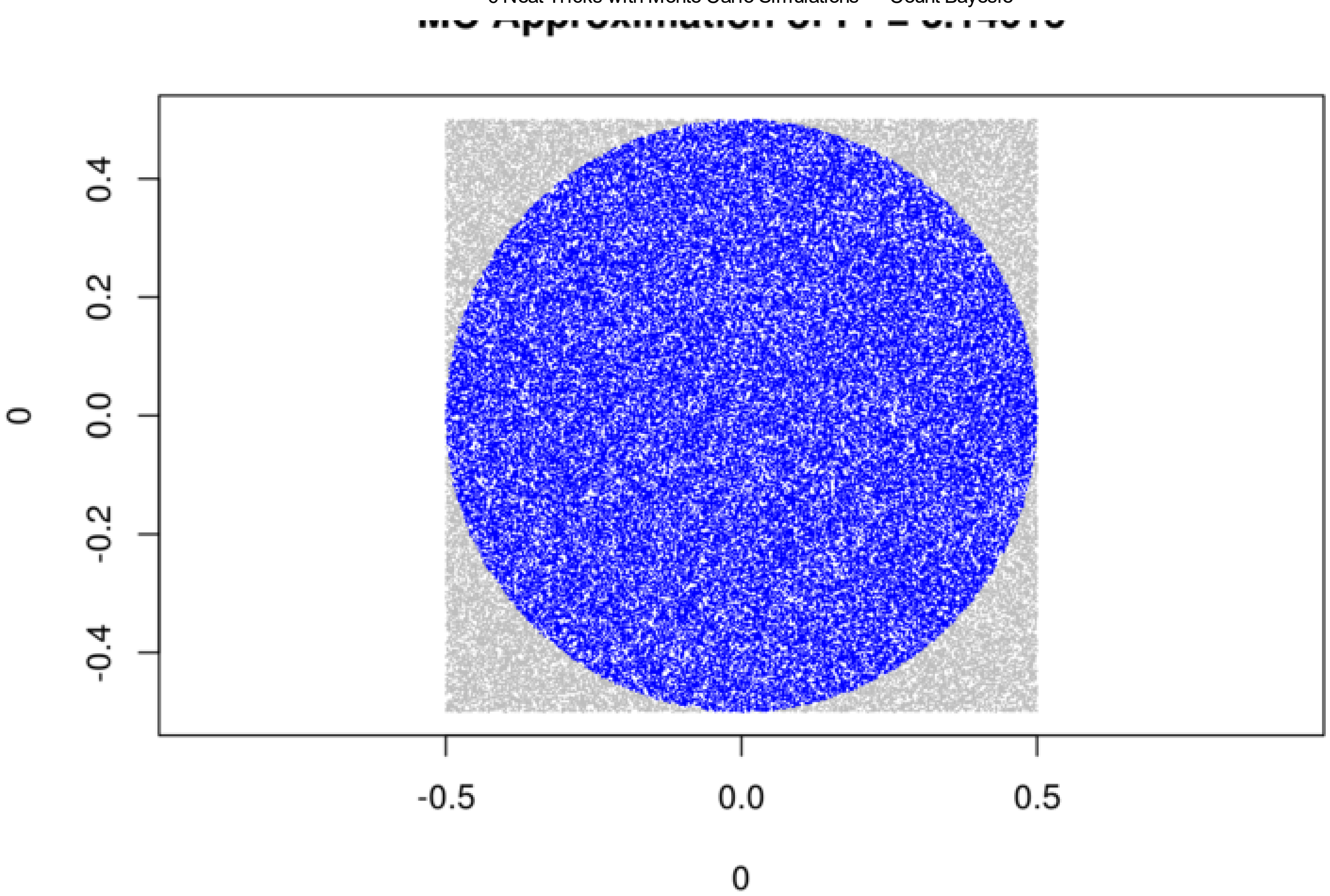
which we can reduce to simply

$$\frac{\pi}{4}$$

Given this fact, if we can empiricaly determine the ratio of the area of the circle to the area of the square we can simply multiply this number by 4 and we'll get our approximation of π.

To do this we can randomly sample x and y values from a unit square centered around 0. If $x^2 + y^2 \leq r^2$ then the point is in the circle (in this case $r = 0.5$). The ratio of points in the circle to total points sample multiplied by 4 should then approximate pi.

```
runs <- 100000
#runif samples from a uniform distribution
xs <- runif(runs,min=-0.5,max=0.5)
ys <- runif(runs,min=-0.5,max=0.5)
in.circle <- xs^2 + ys^2 <= 0.5^2
mc.pi <- (sum(in.circle)/runs)*4
plot(xs,ys,pch='.',col=ifelse(in.circle,"blue","grey")
     ,xlab='',ylab='',asp=1,
     main=paste("MC Approximation of Pi =",mc.pi))
```

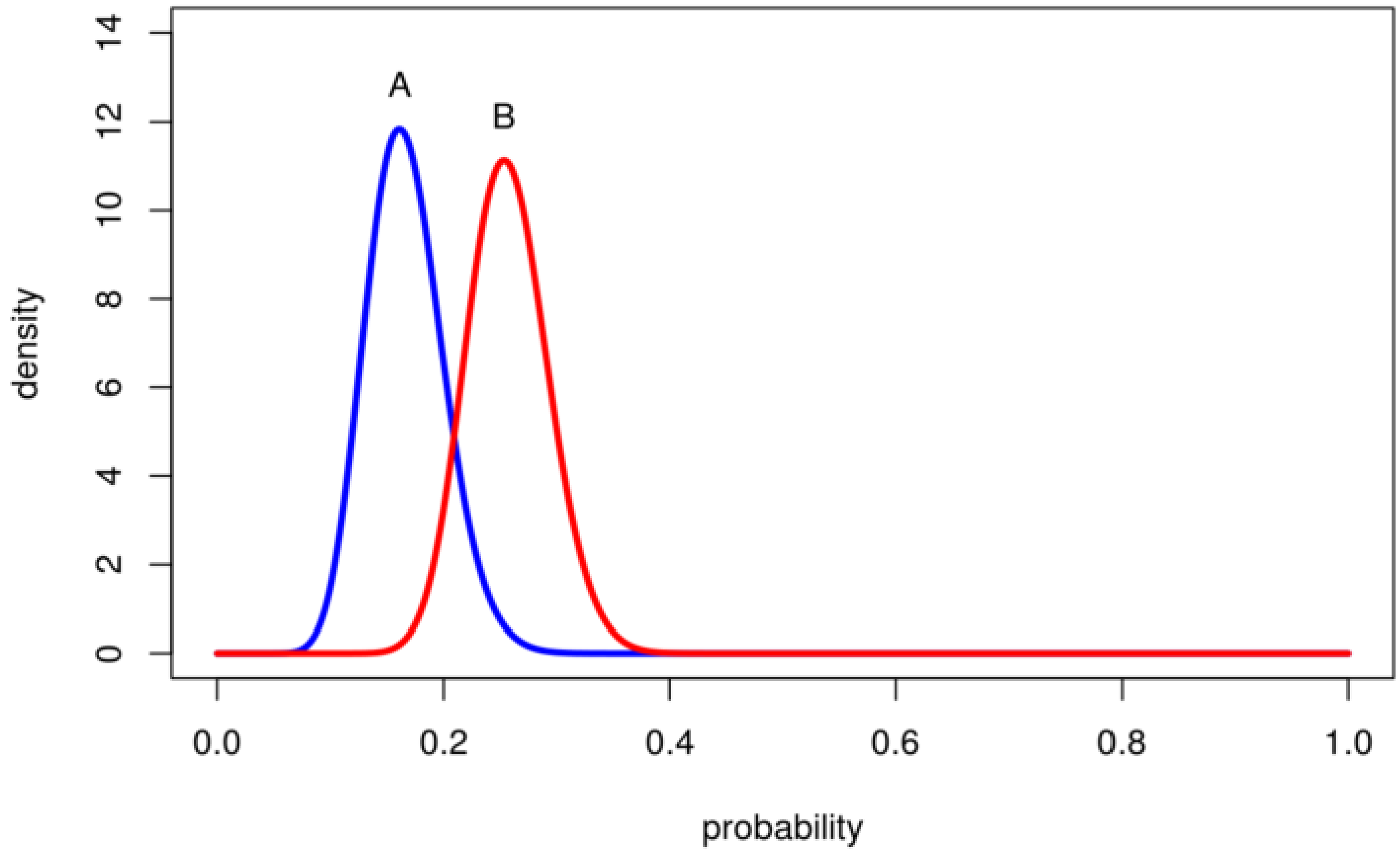


Sampling 100,000 points inside and outside of a circle

The more runs we have the more accurately we can approximate π.

Finding our own p-values

Let's suppose we're comparing two webpages to see which one converts our customers to "sign up" at a higher rate (This is commonly referred to as an A/B Test). For page A we have seen 20 convert and 100 not convert, for page B we have 38 converting and 110 not converting. We'll model this as two Beta distributions as we can see below:



Visualizing the possible overlap between two Beta distributions

It certainly looks like B is the winner, but we'd really like to know how likely this is. We could of course run a single tailed t-test, that would require that we assume that these are Normal distributions (which isn't a terrible approximation in this case). However we can also solve this via a Monte Carlo simulation! We're going to take 100,000 samples from A and 100,000 samples from B and see how often A ends up being larger than B.

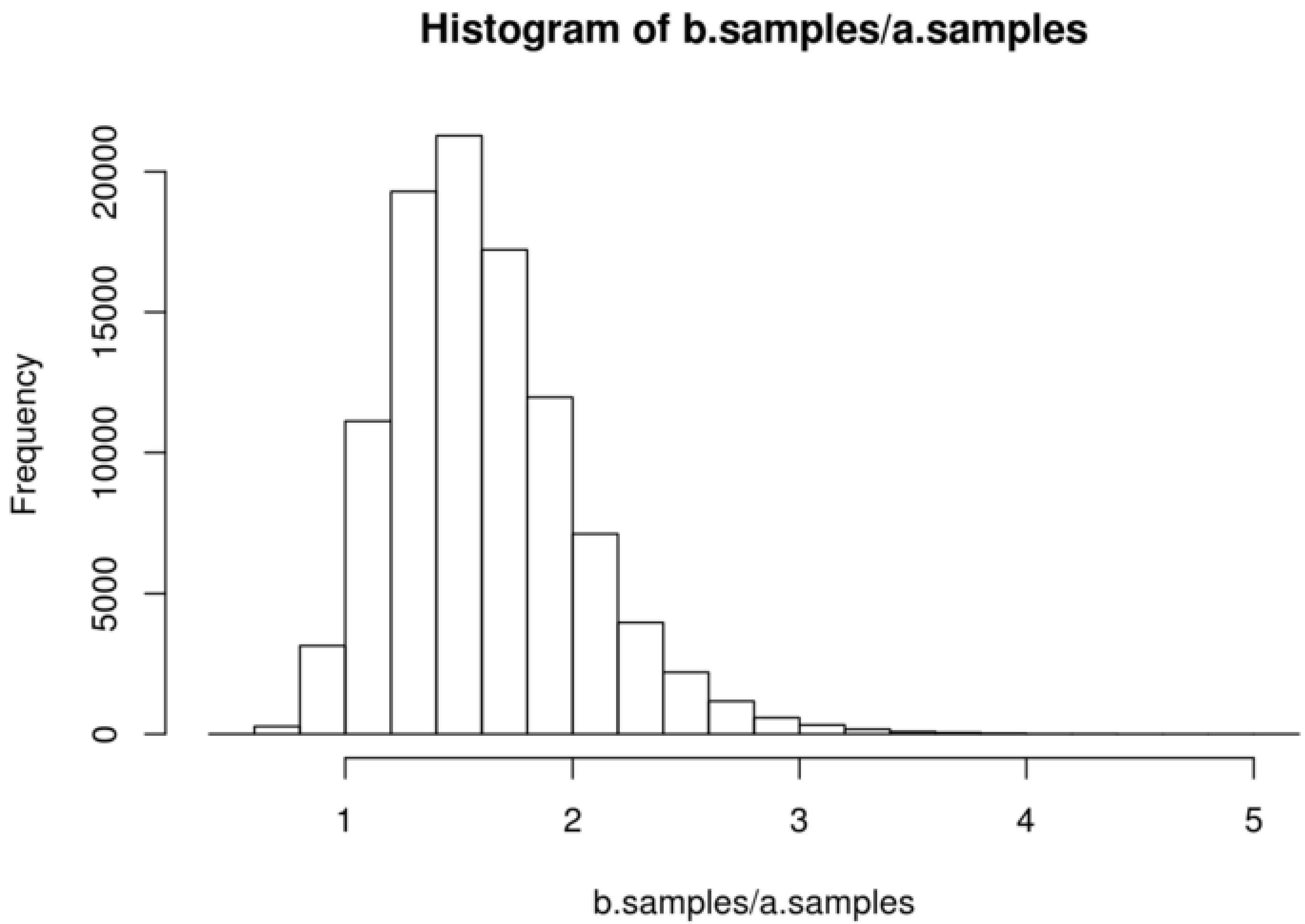
```
runs <- 100000
a.samples <- rbeta(runs,20,100)
b.samples <- rbeta(runs,38,110)
mc.p.value <- sum(a.samples > b.samples)/runs
```

And we have mc.p.value = 0.0348

Awesome our results are "statistically significant"!

..but wait, there's more! We can also plot out a histogram for of the differences to see how big a difference there might be between our two tests!

```
hist(b.samples/a.samples)
```

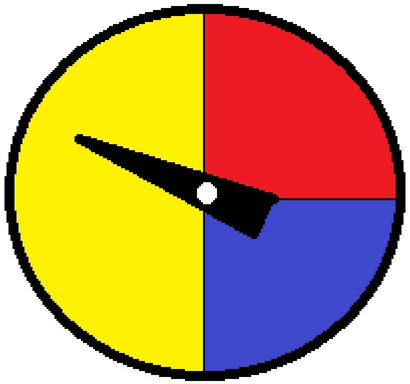


By simulating samples between two distributions we can see the average improvement.

Now we can actually reason about how much of a risk we are taking if we go with B over A!

Games of chance

If we bring back the spinner from the post on Expectation we can play a new game!



The great spinner of probability!

In this game landing on 'yellow' you gain 1 point, 'red' you lose 1 point and 'blue' you gain 2 points. We can easily calculate the expectation:

$$E(\text{game}) = 1/2 \cdot 1 + 1/4 \cdot -1 + 1/4 \cdot 2 = 0.75$$

This could have been calculated with a Monte Carlo simulation, but the hand calculation is really easy. Let's ask a trickier question "After 10 spins what is the probabilitly that you'll have less then 0 points?" There are methods to analytically solve this type of problem, but by the time they are even explained we could have already written our simulation!

To solve this with a Monte Carlo simulation we're going to sample from our Spinner 10 times, and return 1 if we're below 0 other wise we'll return 0. We'll repeat this 100,000 times to see how often it happens!

```
runs <- 100000
#simulates on game of 10 spins, returns whether the sum of all the spins is < 1
play.game <- function(){
  results <- sample(c(1,1,-1,2),10,replace=T)
  return(sum(results) < 0)
}
mc.prob <- sum(replicate(runs,play.game()))/runs
```

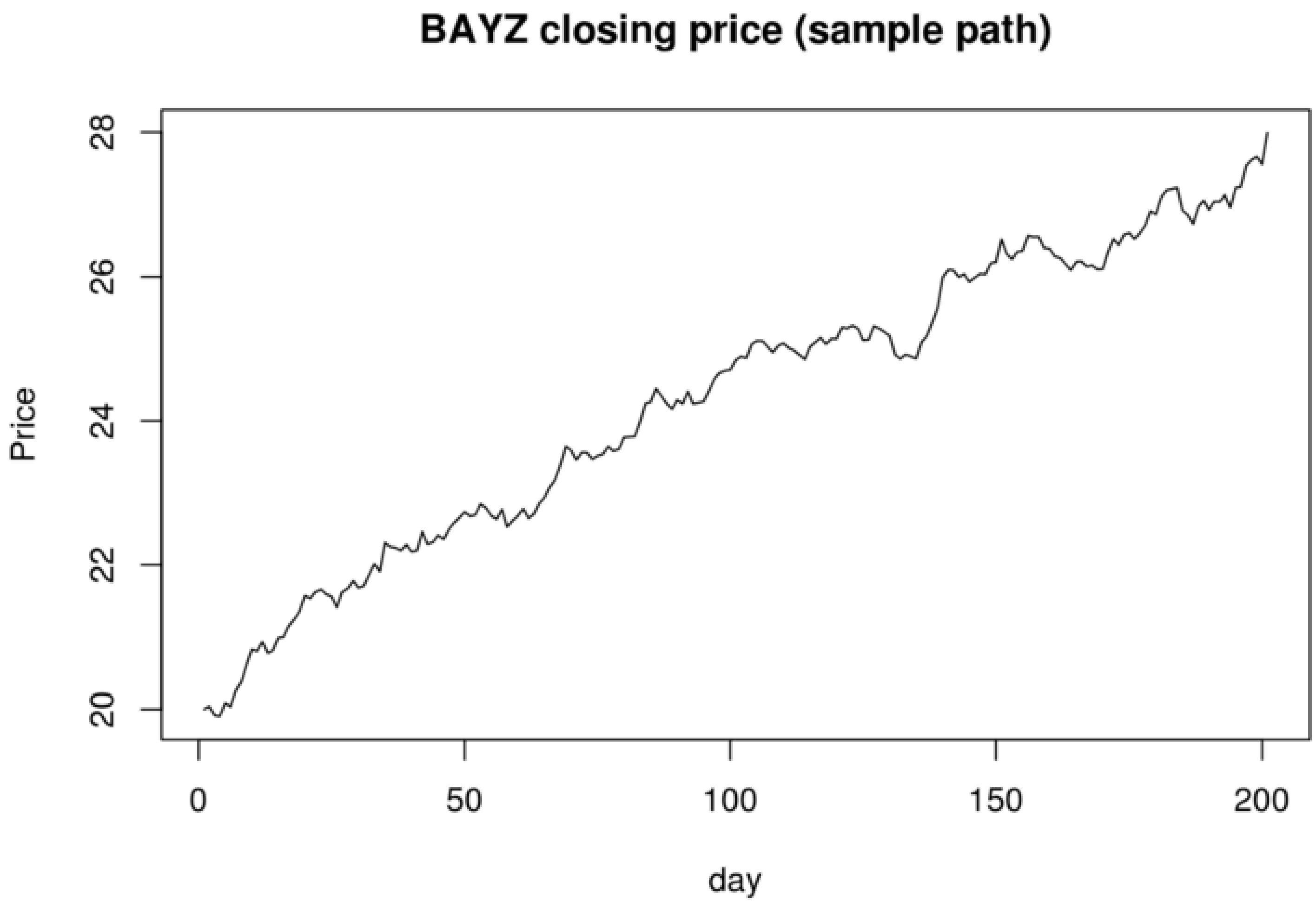
And now we have our estimate on the likelihood of having negative points after just 10 spins.

mc.prob = 0.01314

Predicting the Stock Market

Finally CountBayes.com has IPO'd! It trades under the ticker symbol BAYZ. On average it gains 1.001 times its opening price during the trading day, but that can vary by a standard deviation of 0.005 on any given day (this is its volatility). We can simulate a single sample path for BAYZ by taking the cumulative product from a Normal distribution with a mean of 1.001 and a sd of 0.005. Assuming BAYZ opens at $20/per share here is a sample path for 200 days of BAYZ trading.

```
days <- 200
changes <- rnorm(200,mean=1.001,sd=0.005)
plot(cumprod(c(20,changes)),type='l',ylab="Price",xlab="day",main="BAYZ closing price
```



BAYZ Stock Ticker

But this is just one possible future! If you are thinking of investing in BAYZ you want to know what are the possible closing prices of the stock at the end of 200. To assess risk in this stock we need to know what are reasonable upper and lower bounds on the future price.

To solve this we'll just simulate 100,000 different possible paths the stock could take and then look at the distribution of closing prices.

```
runs <- 100000
#simulates future movements and returns the closing price on day 200
generate.path <- function(){
  days <- 200
  changes <- rnorm(200,mean=1.001,sd=0.005)
  sample.path <- cumprod(c(20,changes))
  closing.price <- sample.path[days+1] #+1 because we add the opening price
  return(closing.price)
}

mc.closing <- replicate(runs,generate.path())
```

The median price of BAYZ at the end of 200 days is simply median(mc.closing) = 24.36
but we can also see the upper and lower 95th percentiles

quantile(mc.closing,0.95) = 27.35

quantile(mc.closing,0.05) = 21.69

NB - This is a toy model of stock market movements, even models that are generally considered poor models of stock prices at the very least would use a log-normal distribution. But those details deserve a post of their own! Real world quantitative finance makes heavy use of Monte Carlo simulations.

Just the beginning!

By now it should be clear that a few lines of R can create extremely good estimates to a whole host of problems in probability and statistics. There comes a point in problems involving probability where we are often left no other choice than to use a Monte Carlo simulation. This is just the beginning of the incredible things that can be done with some extraordinarily simple tools. It also turns out that Monte Carlo simulations are at the heart of many forms of Bayesian inference.

For more examples of using Monte Carlo Simulations check out these posts:

- Build your own Rejection Sampler in R.
- Using a MC simulation to solve a variant of the Coupon Collectors Puzzle.
- As an alternative, Solve Markov Chains with Linear Algebra instead of Monte Carlo Methods

If you enjoyed this post please subscribe to keep up to date and follow @willkurt!

MARCH 2015, POPULAR

37 LIKES SHARE

Newer Older

Comments (19)

Newest First Subscribe via e-mail

Preview POST COMMENT…

JerzyT 2 weeks ago · 1 like

Many years ago, in a Physical Chemistry course in college, we got an assignment to calculate the probability of an electron being within a certain confined range of a hydrogen atom. In essence, it was a nasty triple integral, not solvable analytically, only numerically.

I used Lotus 123, an early ancestor of Excel, to run a Monte Carlo simulation to solve the problem. I was the only one in the class with a solution. The funny thing was that is was so easy it felt like cheating.

The professor put that on my recommendation letter for grad school.

Siva 11 months ago

Hi Will,

Are there any resources that you find useful to perform or implement monte carlo simulations for real world examples?

P.S. - I have used Monte Carlo methods with R and the books listed on your site.

Thanks.

muraii A year ago

Late to the party, but I'm working my way through all your posts, somewhat sequentially. I'm not grokking everything but it's definitely building layers of familiarity.

In the stock market example, you touch on something I'm curious about. You use the average and standard deviation gains, and thereafter model performance by sampling from a Normal distribution. I wonder if you can confirm my intuition that to use the average and standard deviation, you're implicitly assuming normality. I think this is obvious but I ask because I've read cases in which folks are using the arithmetic mean and standard deviation without stating assumptions, or perhaps without knowing or assuming normality holds.

I feel that average and standard deviation are useless if you don't assume (or better, demonstrate) normality, just as using the expectation value for a Beta distribution to estimate for an apparently Poisson-distributed sample. But I also don't want to stay too hard-and-fast. Maybe it's useful to calculate an average, for instance, even if it's not the expectation value for the sample at-hand.

Have you written on this already, or would this line up with something you'd like to write about in the future?

Will Kurt A year ago · 1 like

In my posts on Expectation and Variance one of the things I always try to point out is that these are just mathematical operations with no inherent meaning for exactly the issue you bring up. It's fine to use mean and standard deviation (or Expectation on Variance) on any data, but the key thing is that many of the useful assumptions those values provide deeply depend on the assumption of normality.

I'm actually thinking, because of this comment, that I might touch on the topic of random walks sooner than planned as it might provide some more useful insight.

I hope that helps answer your question and as always thanks so much for reading!

muraii A year ago

Looking forward to it. +)

Brandon Harrison A year ago

I really like how you've modeled changes in stock price with a Monte Carlo simulation. I generally run regressions on stock and index datasets to build polynomial models (y = a + bx + cx^2 + dx^3...), but the further my forecast goes out in time, my model typically jumps off a cliff. Seems like using a Monte Carlo simulation would be just as good if not more reasonable?

Akshay Rao A year ago

As always awesome post. It is truly enlightening if one work through each example! :)
I wonder if you could do a "Count Bayesie" style post on common distribution functions in the wild. Would definitely love to read your take on them

Will Kurt A year ago · 1 like

And as always thank you for reading! I have a couple of posts on some probability distributions coming up, but for a long time I've wanted to build a catalog of interesting distributions which is something I'll definitely add to the queue! Thanks for the idea!

Tim A year ago

I really really like your articles on probability. Eventhough I have a basic statistical education, your articles light up a new perspective on the matter. Really love your work. Keep it up! :)

Will Kurt A year ago

Thanks so much! I'm working hard to keep new posts coming!

lurker A year ago

simulates on game of 10 spins, returns whether the sum of all the spins is < 1
...
return(cumsum(results) < 0)

did you meant "... all the spins is < 0" ?

Will Kurt A year ago

There was actually an error in the original code which as been fixed, cumsum should have been just 'sum' which is the case now.

lurker A year ago

I think you have a typo in the approximation of pi you state "If x2+y2≥r2 then the point is in the circle"
it should be the opposite, if x^2 + y^2 <= r^2

Vincent DM A year ago

Great post! Small typo in this paragraph: "To do this we can randomly sample x and y values from a unit square centered around 0. If x2+y2≥r2 then the point is in the circle".

I think the formula should be x2+y2≤r2 (as in the example code)

Rasmus Arnling Bååth A year ago

Hmm, I'm not sure your p-value is a p-value, looks more like "just" a standard posterior probability (which I think is a good thing!)

Will Kurt A year ago

Thanks for reading! You may be technically correct from a theoretical standpoint, I'm not entirely sure as you are correct that this is a more Bayesian approach. I do know that in the past when I've used this technique in Bayesian A/B tests the value I get is nearly identical to the p-value from a singled-tail t-test

Rasmus Arnling Bååth A year ago

Ah, but technically correct is the best kind of correct! :) In certain cases p-values and posterior probabilities are numerically identically. What makes a p-value a p-value is how you motivated it's calculation. For example, a p-value is always calculated with respect to a "null-hypothesis".

Asmund A year ago

I liked it. BTW, there's a typo in your criterion for whether a point is inside the circle.

Will Kurt A year ago

Thanks so much! The typo should be fixed now.