Monte Carlo Simulation

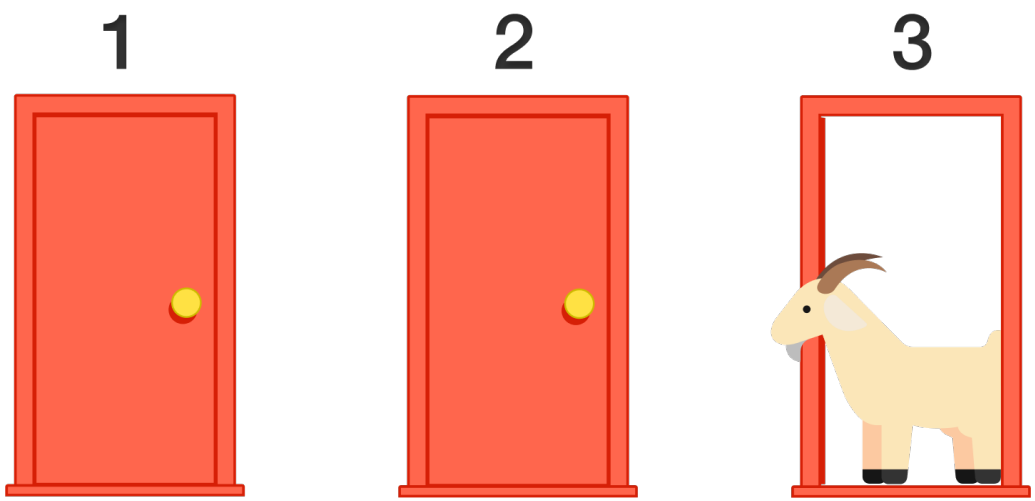
Dootika Vats

Department of Mathematics and Statistics
Indian Institute of Technology, Kanpur

ISI Bangalore, October 10, 2019

Example: Monty Hall Problem (Khullja Sim Sim)

You are on a game show, being asked to choose between three doors. One door has a car, and the other two have goats. The host, Monty Hall, opens one of the other doors, which he knows has a goat behind it. Monty then asks whether you would like to switch your choice of door to the other remaining door. *Do you choose to switch or not to switch?*



https://brilliant.org/wiki/monty-hall-problem/

Example: Monty Hall Problem (Khullja Sim Sim)

You should always switch!

$$\Pr(\text{Winning if you don't switch}) = \frac{1}{3}$$

$$\Pr(\text{Winning if you switch}) = \frac{2}{3}$$

Example: Monty Hall Problem (Khullja Sim Sim)

You should always switch!

$$\Pr(\text{Winning if you don't switch}) = \frac{1}{3}$$

$$\Pr(\text{Winning if you switch}) = \frac{2}{3}$$

If this phenomenon is hard to believe, you're not alone. Vos Savant (1997) writes

> *"even Nobel physicists systematically give the wrong answer, and that they insist on it, and they are ready to berate in print those who propose the right answer"*

Example: Monty Hall simulation

```
doors <- 1:3
prize <- sample(1:3, 1) #choose a door at random
repeats <- 1e4
win.no.switch <- numeric(length = repeats)
win.switch <- numeric(length = repeats)

for(r in 1:repeats)    # Repeat process many times
{
  chosen.door <- sample(1:3, 1)          # choose a door
  reveal <- (1:3)[-c(prize,chosen.door)][1] #reveal a door

  win.no.switch[r] <- chosen.door == prize  #don't change door
  chosen.door <- (1:3)[-c(reveal, chosen.door)] #change door
  win.switch[r] <- chosen.door == prize
}
mean(win.no.switch) #Prob of winning if you don't switch
[1] 0.3316
mean(win.switch) # Prob of winning if you switch
[1] 0.6684
```

Monte Carlo Simulation

Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated *random sampling* to obtain numerical results.

In the Monte Hall problem, we repeated the experiment 10,000 times and observed whether we won or not. This is *Monte Carlo simulation*.

Then we use the simulated draws to *estimate* the probability of winning in each case, this is *Monte Carlo integration*.

Example: Birthday Candles

It's my 30th birthday, and my friends bought me a cake with 30 candles on it. I make a wish and try to blow them out. Every time, I blow out a random number of candles between one and the number that remain, including one and that other number. How many times, on average, do I blow before all the candles are extinguished?

Example: Birthday Candles

It's my 30th birthday, and my friends bought me a cake with 30 candles on it. I make a wish and try to blow them out. Every time, I blow out a random number of candles between one and the number that remain, including one and that other number. How many times, on average, do I blow before all the candles are extinguished?

Monte Carlo simulation: Repeat experiment multiple times and note down the number of attempts. That is, let F be the distribution of the number of blows required. Generate

$$X_1, X_2, \ldots, X_N \overset{iid}{\sim} F$$

Monte Carlo integration: Estimate $\mathrm{E}_F[X]$ by the average of simulated values

$$\frac{1}{N} \sum_{t=1}^{N} X_t$$

Example: Monte Carlo

```
n <- 30      # no. of candles
repeats <- 1e4
iter <- numeric(length = repeats)

for(r in 1:repeats)
{
  candles <- n
  remain <- n
  while(remain != 0) # as long as candles are left
  {
      iter[r] <- iter[r] + 1
      blow.out <- sample(1:remain, 1) #randomly draw number of candles
          that will blow out
      remain <- remain - blow.out
  }
}
head(iter) # First 6 draws of Monte Carlo simulation
[1] 6 3 3 6 9 5
mean(iter) # Monte Carlo integration
[1] 3.9912
```

Monte Carlo in Applications

Engineering, biology, climate science, neuroscience, ecology, finance, etc.

Essentially, there are two main places where Monte Carlo is used

- If a complicated model determines the quantity, like climate models, bridge breaking points, stock markets etc.
- If it is not possible to analytically determine the quantity of interest.

Monte Carlo in Applications

Engineering, biology, climate science, neuroscience, ecology, finance, etc.

Essentially, there are two main places where Monte Carlo is used

- If a complicated model determines the quantity, like climate models, bridge breaking points, stock markets etc.
- If it is not possible to analytically determine the quantity of interest.

  In statistics, this is where it is often used.

Monte Carlo in Statistics

Let's say we are faced with a difficult integral

$$\mu = \int_0^\pi e^{\sin(x)} dx \, .$$

A closed-form solution is tough. Instead of *evaluating* the integral, we will *estimate* the integral. See that

Monte Carlo in Statistics

Let's say we are faced with a difficult integral

$$\mu = \int_0^\pi e^{\sin(x)} dx \,.$$

A closed-form solution is tough. Instead of *evaluating* the integral, we will *estimate* the integral. See that

$$\mu = \int_0^\pi e^{\sin(x)} dx = \pi \int_0^\pi e^{\sin(x)} \frac{1}{\pi} dx = \pi \, \mathsf{E}_F \left[e^{\sin(x)} \right] \qquad \text{where } F \text{ is Unif}[0, \pi]$$

Monte Carlo in Statistics

Let's say we are faced with a difficult integral

$$\mu = \int_0^\pi e^{\sin(x)} dx \,.$$

A closed-form solution is tough. Instead of *evaluating* the integral, we will *estimate* the integral. See that

$$\mu = \int_0^\pi e^{\sin(x)} dx = \pi \int_0^\pi e^{\sin(x)} \frac{1}{\pi} dx = \pi \, \mathsf{E}_F \left[e^{\sin(x)} \right] \qquad \text{where } F \text{ is Unif}[0, \pi]$$

```
N <- 1e4
draws <- runif(N, min = 0, max = pi) # Draw from Unif[0,pi]

MC.samples <- pi*exp(sin(draws)) # Monte Carlo simulation
(MC.est <- mean(MC.samples))    # Monte Carlo integration
[1] 6.216101
```

Monte Carlo Integration: why it works?

Monte Carlo simulation: $X_1, X_2, \ldots, X_N \overset{iid}{\sim} F$ with mean μ and variance σ^2.

Monte Carlo estimator: By the strong law of large numbers,

$$\hat{\mu}_N := \frac{1}{N} \sum_{t=1}^{N} X_t \overset{a.s.}{\to} \mu \quad \text{as } N \to \infty .$$

So the classical laws of statistics applies.

Monte Carlo Integration: why it works?

Monte Carlo simulation: $X_1, X_2, \ldots, X_N \overset{iid}{\sim} F$ with mean μ and variance σ^2.

Monte Carlo estimator: By the strong law of large numbers,

$$\hat{\mu}_N := \frac{1}{N} \sum_{t=1}^{N} X_t \overset{a.s.}{\to} \mu \quad \text{as } N \to \infty \,.$$

So the classical laws of statistics applies. Similarly, the CLT holds

$$\sqrt{N}\,(\hat{\mu}_N - \mu) \overset{d}{\to} N(0, \sigma^2)\,.$$

Monte Carlo error

We can estimate σ^2 with the usual sample variance

$$s^2 := \frac{1}{N-1} \sum_{t=1}^{N} (X_t - \hat{\mu}_N)^2 \ ,$$

and

$$\frac{\hat{\mu}_N - \mu}{\sqrt{s^2/N}} \approx \ t_{N-1}$$

So, a 95% (large-sample) confidence interval is

$$\hat{\mu}_N \pm t_{.975,N-1} \sqrt{\frac{s^2}{N}} \ .$$

In repeated simulations, on average, 95% of such confidence intervals will contain the truth.

Confidence Interval MC simulation

```
# Confidence interval
repeats <- 1e4
N <- 1e4
truth <- 6.20876 # True value of the integral
conf.track <- numeric(length = repeats)
for(r in 1:repeats)
{
  draws <- runif(N, min = 0, max = pi)
  MC.samples <- pi*exp(sin(draws))
  MC.est <- mean(MC.samples)    # Repeating this process

  var.est <- var(MC.samples) # Calculate variance and half-width
  half.width <- qt(.975, df = N-1)*sqrt(var.est/N)

# If truth is between upper and lower bounds
  conf.track[r] <- abs(truth - MC.est) < half.width
}
(mean(conf.track)) #about 95% contain the truth
[1] 0.9486
```

Estimating more than one quantity

Suppose we are interested in estimating

$$\mu_1 = \int_0^\pi e^{\sin(x)}\,dx\,dy \quad \text{and } \mu_2 = \int_0^\pi e^{\cos(x)}$$

We can use the same samples to estimate both these quantities.

Estimating more than one quantity

Suppose we are interested in estimating

$$\mu_1 = \int_0^\pi e^{\sin(x)}\,dx\,dy \quad \text{and } \mu_2 = \int_0^\pi e^{\cos(x)}$$

We can use the same samples to estimate both these quantities.

```
N <- 1e4
draws <- runif(N, min = 0, max = pi)
(MC.est1 <- mean(pi*exp(sin(draws))) ) #mu1
(MC.est2 <- mean(pi*exp(cos(draws))) ) #mu2
```

But, since we are estimating two things *together*, confidence intervals suffer.

Joint confidence intervals

We have two CLTs, one for each estimator

$$\sqrt{N}\,(\hat{\mu}_1 - \mu_1) \xrightarrow{d} N(0, \sigma_1^2) \quad \text{and} \quad \sqrt{N}\,(\hat{\mu}_2 - \mu_2) \xrightarrow{d} N(0, \sigma_2^2)$$

These give two corresponding confidence intervals

$$\hat{\mu}_1 \pm t_{.975,N-1}\sqrt{\frac{s_1^2}{N}} \quad \text{and} \quad \hat{\mu}_2 \pm t_{.975,N-1}\sqrt{\frac{s_2^2}{N}}$$

In repeated simulations, 95% of the 1st confidence interval will contain μ_1 and 95% of the 2nd confidence interval will contain μ_2.

Joint confidence intervals

We have two CLTs, one for each estimator

$$\sqrt{N}\,(\hat{\mu}_1 - \mu_1) \xrightarrow{d} N(0, \sigma_1^2) \quad \text{and} \quad \sqrt{N}\,(\hat{\mu}_2 - \mu_2) \xrightarrow{d} N(0, \sigma_2^2)$$

These give two corresponding confidence intervals

$$\hat{\mu}_1 \pm t_{.975,N-1}\sqrt{\frac{s_1^2}{N}} \quad \text{and} \quad \hat{\mu}_2 \pm t_{.975,N-1}\sqrt{\frac{s_2^2}{N}}$$

In repeated simulations, 95% of the 1st confidence interval will contain μ_1 and 95% of the 2nd confidence interval will contain μ_2.

But, the chances that both intervals contain their respective truths in repeated simulations is around $95\% * 95\% \approx 90\%$.

Estimating more than one quantity

```
conf.track <- matrix(0, nrow = repeats, ncol = 3)
for(r in 1:repeats)
{
  draws <- runif(N, min = 0, max = pi)
  MC.est1 <- mean(pi*exp(sin(draws)) )
  MC.est2 <- mean(pi*exp(cos(draws)))

  var.est1 <- var(pi*exp(sin(draws)))
  var.est2 <- var(pi*exp(cos(draws)))
  half.width1 <- qt(.975, df = N-1)*sqrt(var.est1/N)
  half.width2 <- qt(.975, df = N-1)*sqrt(var.est2/N)

  conf.track[r,1] <- abs(truth1 - MC.est1) < half.width1
  conf.track[r,2] <- abs(truth2 - MC.est2) < half.width2
  conf.track[r,3] <- conf.track[r,1] && conf.track[r,2] # joint CI
}
colMeans(conf.track)
  mu1       mu2 mu1_&_mu2
0.9500    0.9514    0.9052 #Joint estimation has lesser than 95% coverage
```

Correcting for multiple quantities

There are many ways of correcting for multiple quantities, but one of the most common is to increase the individual coverage probabilities, .95 to .975, since $.975 * .975 \approx .95$. In general for estimating p quantities, a formula for this is

$$\alpha^* = \alpha/p$$

This is called the Bonferroni correction.

This is not specific to Monte Carlo, but for almost any joint confidence interval construction.

Correcting for multiple quantities

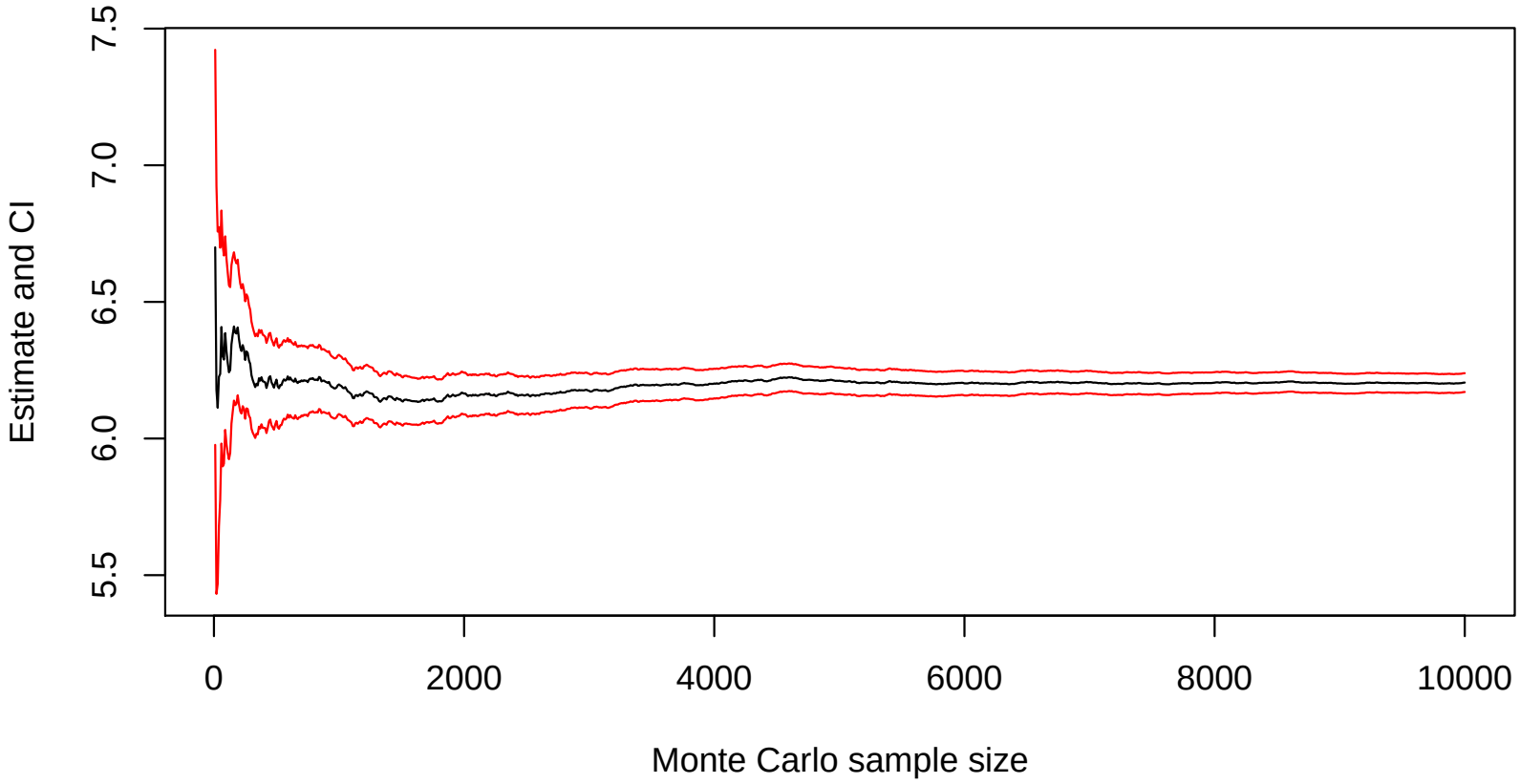
```
for(r in 1:repeats)
{
  draws <- runif(N, min = 0, max = pi)
  MC.est1 <- mean(pi*exp(sin(draws)))
  MC.est2 <- mean(pi*exp(cos(draws)))

  var.est1 <- var(pi*exp(sin(draws)))
  var.est2 <- var(pi*exp(cos(draws)))

  # new half-widths from adjusted t-quantile
  new.half.width1 <- qt( 1- .05/4, df = N-1)*sqrt(var.est1/N)
  new.half.width2 <- qt( 1- .05/4, df = N-1)*sqrt(var.est2/N)

  conf.track[r,1] <- abs(truth1 - MC.est1) < new.half.width1
  conf.track[r,2] <- abs(truth2 - MC.est2) < new.half.width2
  conf.track[r,3] <- conf.track[r,1] && conf.track[r,2]
}
# Individual CIs have .975 coverage, but joint CI has good coverage
colMeans(conf.track)
  mu1       mu2 mu1_&_mu2
0.9751    0.9743    0.9509
```

Why confidence intervals?

The reason confidence intervals are useful is because unlike real life, we can continue simulating draws on a computer. So when should we stop?

Why confidence intervals?

The reason confidence intervals are useful is because unlike real life, we can continue simulating draws on a computer. So when should we stop?



We can stop when the width of the CI is small, so accuracy is high.

Problem: Monte Carlo Sampling

One major problem is that Monte Carlo sampling is not always so easy. Consider estimating

$$\theta = \int_{x^2+y^2 \leq 1} \sin(x^2 + y^2) dx\, dy$$

Problem: Monte Carlo Sampling

One major problem is that Monte Carlo sampling is not always so easy. Consider estimating

$$\theta = \int_{x^2+y^2 \leq 1} \sin(x^2 + y^2) dx\, dy = \pi \int_{x^2+y^2 \leq 1} \frac{1}{\pi} \sin(x^2 + y^2) dx\, dy\,.$$

Define density f being the uniform density over a unit circle

$$f(x, y) = \frac{1}{\pi} I(x^2 + y^2 \leq 1)\,.$$

Problem: Monte Carlo Sampling

One major problem is that Monte Carlo sampling is not always so easy. Consider estimating

$$\theta = \int_{x^2+y^2 \leq 1} \sin(x^2 + y^2) dx\, dy = \pi \int_{x^2+y^2 \leq 1} \frac{1}{\pi} \sin(x^2 + y^2) dx\, dy\,.$$

Define density f being the uniform density over a unit circle

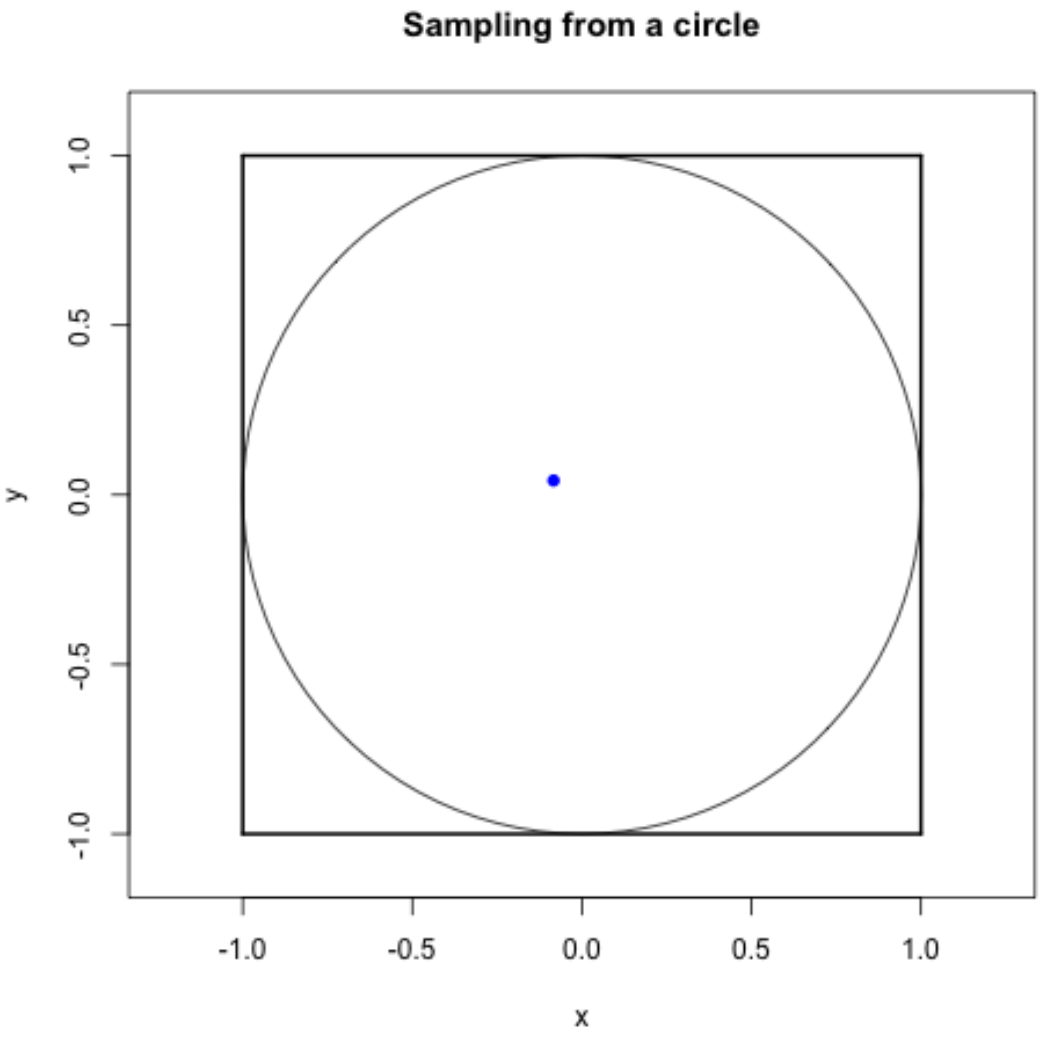
$$f(x, y) = \frac{1}{\pi} I(x^2 + y^2 \leq 1)\,.$$

Then,

$$\mu = \pi \int \sin(x^2 + y^2) f(x, y) dx\, dy = \pi\, \mathsf{E}_F \left[\sin(X^2 + Y^2)\right]\,.$$

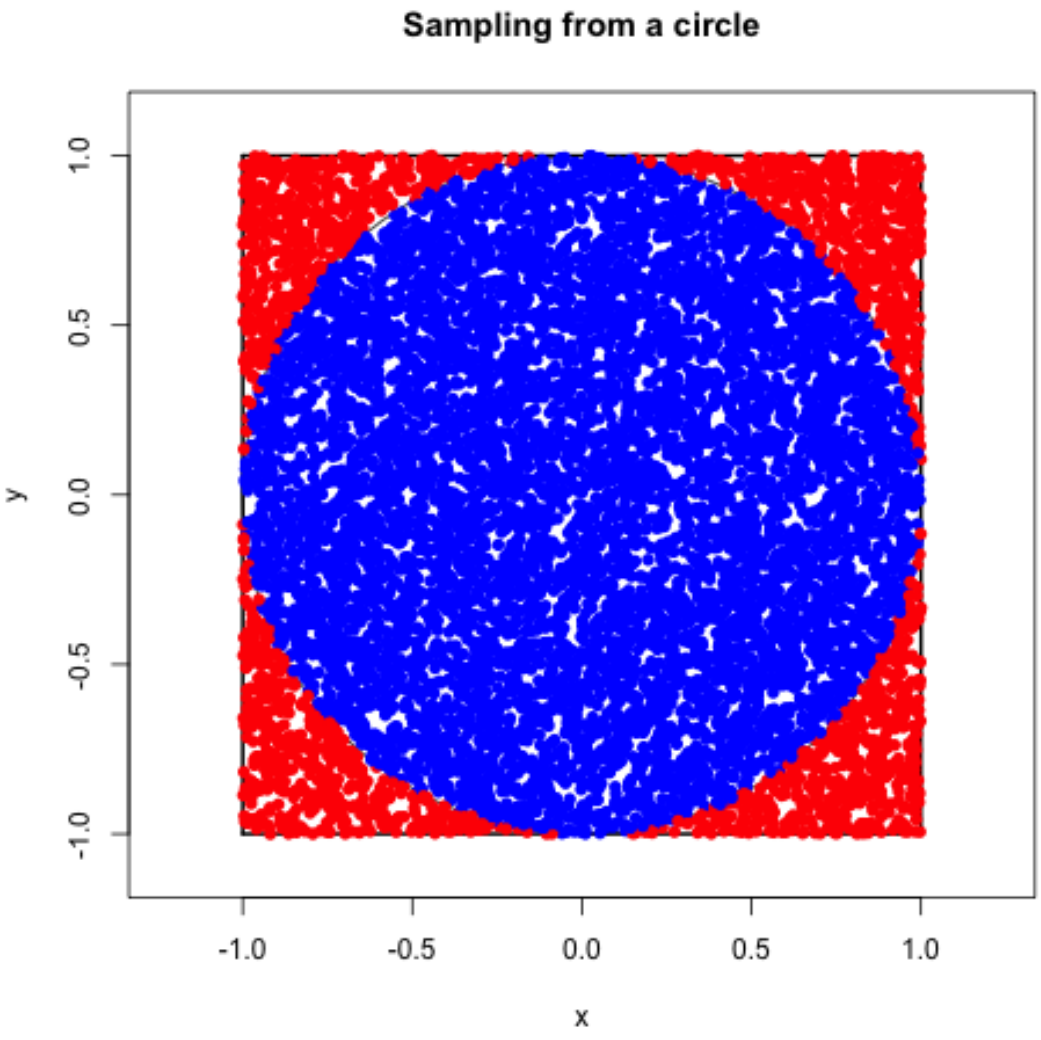
So we want to draw $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$ from the Uniform circle.

Sampling from a circle




```r
N <- 1e4
count <- 0
x <- numeric(length = N)
y <- numeric(length = N)
while(count < N)
{
  # sample from box
  from_box <- runif(2, min = -1,
        max = 1)

  # Accept if inside circle, accept
  if(from_box[1]^2 + from_box[2]^2
        <= 1)
  {
    x[count] <- from_box[1]
    y[count] <- from_box[2]
    count <- count + 1
  }
}
```

Sampling from a circle




```
N <- 1e4
count <- 0
x <- numeric(length = N)
y <- numeric(length = N)
while(count < N)
{
  # sample from box
  from_box <- runif(2, min = -1,
        max = 1)

  # Accept if inside circle, accept
  if(from_box[1]^2 + from_box[2]^2
        <= 1)
  {
    x[count] <- from_box[1]
    y[count] <- from_box[2]
    count <- count + 1
  }
}
```

```
# Monte Carlo Integration
mean(pi*sin(x^2 + y^2))
[1] 1.450068
```

Sampling from a circle

How efficient is this method?

$$\Pr\left(\text{accepting draws}\right) = \frac{\text{Area of circle}}{\text{Area of square}} = \frac{\pi}{4} \approx .785\,.$$

Pretty efficient.

Sampling from a circle

How efficient is this method?

$$\Pr\left(\text{accepting draws}\right) = \frac{\text{Area of circle}}{\text{Area of square}} = \frac{\pi}{4} \approx .785\,.$$

Pretty efficient. But, suppose you want to sample from a p-dimensional sphere. Same principle, sample from a p-cuboid, and accept if inside the sphere

$$\sum_{i=1}^{p} x_i^2 \leq 1\,.$$

$$\Pr\left(\text{accepting draws}\right) = \frac{\text{Area of sphere}}{\text{Area of cuboid}} = \frac{\pi^{p/2}}{\Gamma(p/2+1)2^p}\,.$$

Sampling from a circle

How efficient is this method?

$$\Pr(\text{accepting draws}) = \frac{\text{Area of circle}}{\text{Area of square}} = \frac{\pi}{4} \approx .785\,.$$

Pretty efficient. But, suppose you want to sample from a p-dimensional sphere. Same principle, sample from a p-cuboid, and accept if inside the sphere

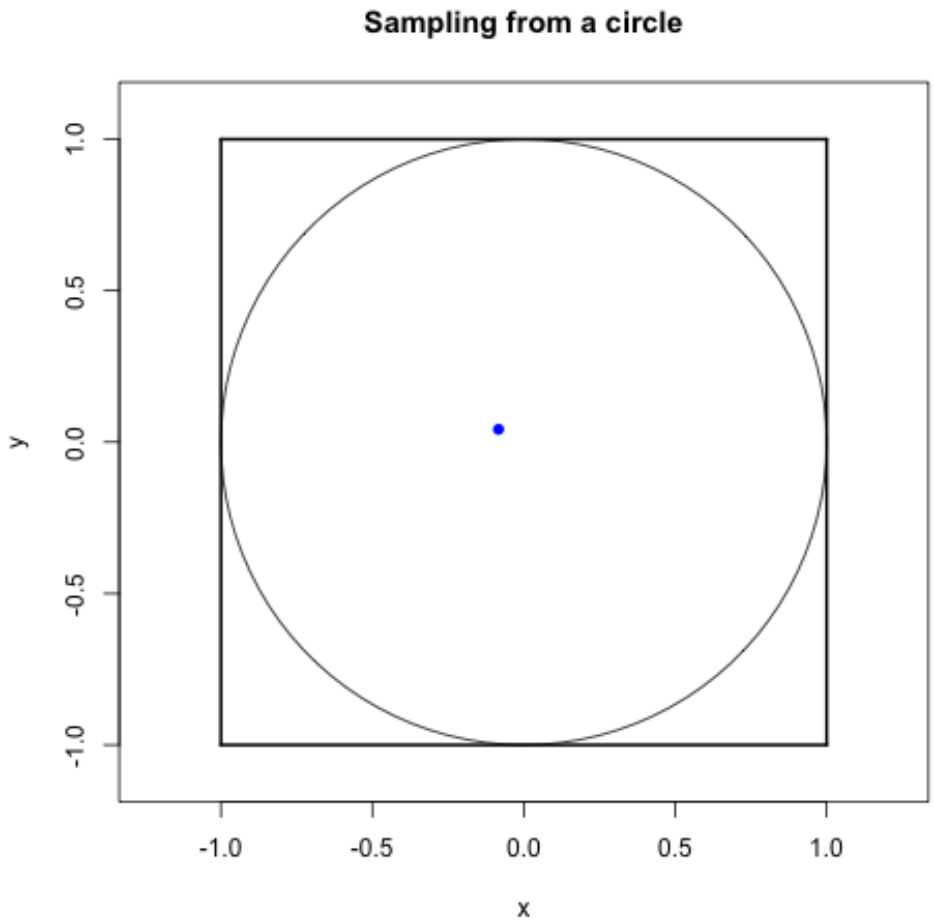
$$\sum_{i=1}^{p} x_i^2 \leq 1\,.$$

$$\Pr(\text{accepting draws}) = \frac{\text{Area of sphere}}{\text{Area of cuboid}} = \frac{\pi^{p/2}}{\Gamma(p/2+1)2^p}\,.$$

For $p = 5$ it is .164 and say $p = 15$, probability of accepting is .0000116 !

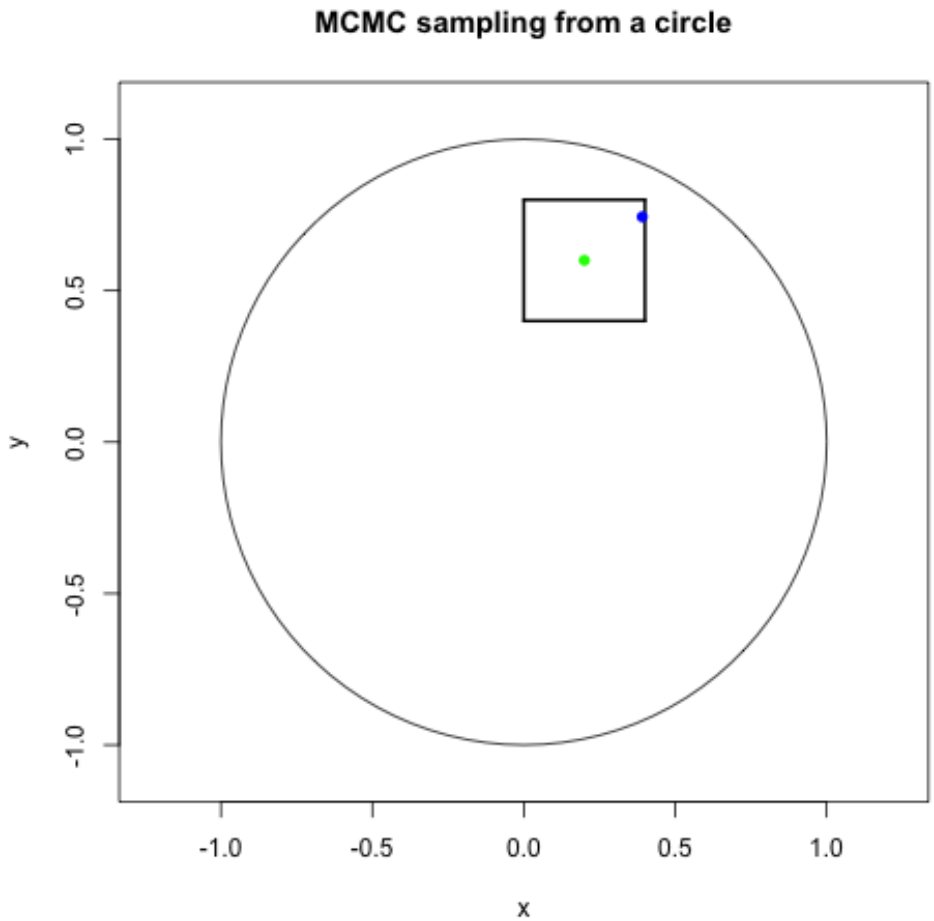
Markov chain Monte Carlo (MCMC)

The low efficiency in high dimensions is because information about past acceptances is not retained.



MCMC: sampling from a circle

What if we used the information about our "current" sample to propose the next sample?



How is MCMC different from traditional Monte Carlo?

The samples obtained are not iid. So classical statistics cannot be used.

- Does a law of large numbers hold?
- Does a CLT hold?
- If yes, what is the variance in the CLT?
- Can we still make confidence intervals?

These are all valid research questions. There are answers available. References:

- Handbook of Markov chain Monte Carlo (Brooks et al., 2011)
- Monte Carlo statistical methods (Robert and Casella, 2013)
- Simulation and the Monte Carlo method (Rubinstein and Kroese, 2016)

Thank you.

Questions?

References

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). Handbook of Markov chain Monte Carlo. CRC press.

Robert, C. and Casella, G. (2013). Monte Carlo statistical methods. Springer Science & Business Media.

Rubinstein, R. Y. and Kroese, D. P. (2016). Simulation and the Monte Carlo method, volume 10. John Wiley & Sons.

Vos Savant, M. (1997). The power of logical thinking: easy lessons in the art of reasoning... and hard facts about its absence in our lives. Macmillan.