STAT5003

Week 10 : Monte Carlo

Dr. Justin Wishart

THE UNIVERSITY OF SYDNEY

Goal

Readings 🏛

- Easily accessible text in Shonkwiler and Mendivil (2009)

Monte Carlo Methods

THE UNIVERSITY OF SYDNEY

What are Monte Carlo methods?

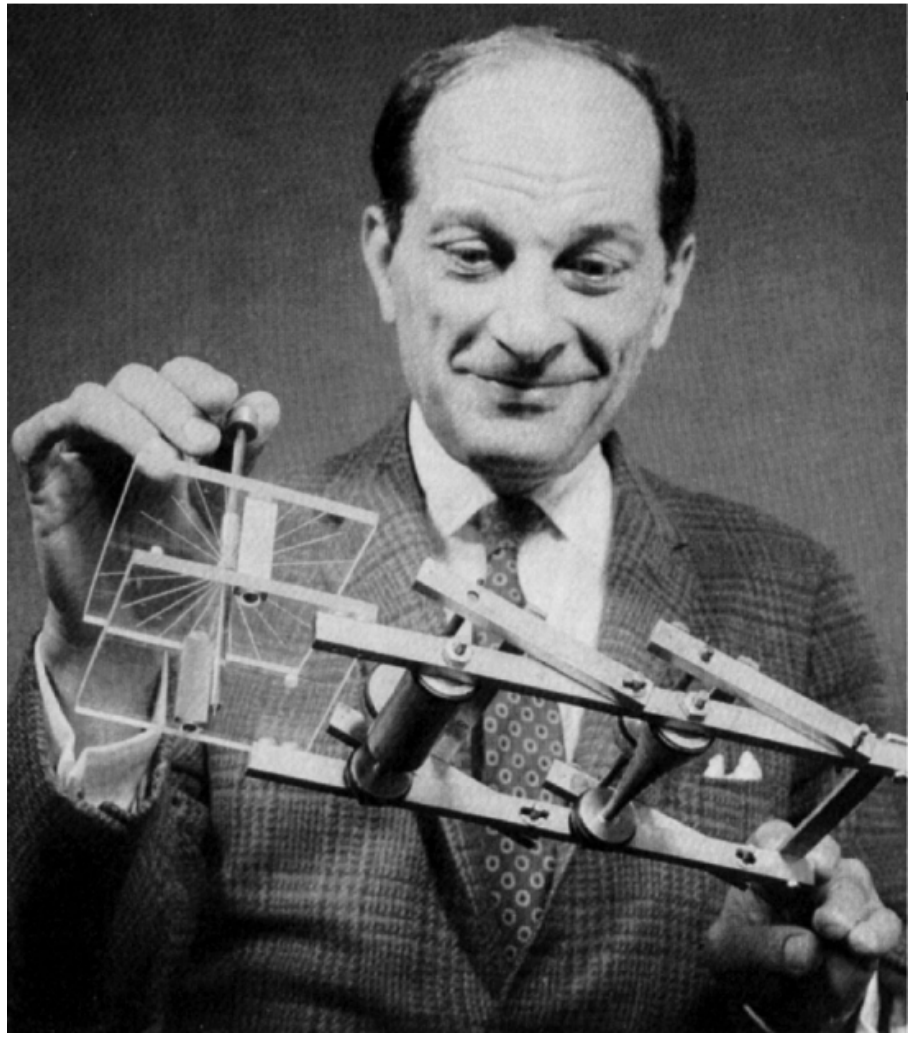
- Monte Carlo are a class of computational methods that can be applied to a wide range of problems
- Key aspect of Monte Carlo methods are that they rely on **random sampling**
- Generally provide approximate solutions
  - not exact solutions
- Used in cases where analytic solutions don't exist or are too difficult to implement
- Monte Carlo methods are sometimes referred to as stochastic simulation

History of the Monte Carlo method

- Invented by a Polish mathematician called Stanislaw Ulam in the late 1940s
- Inspiration during recovery from illness with a thought experiment:

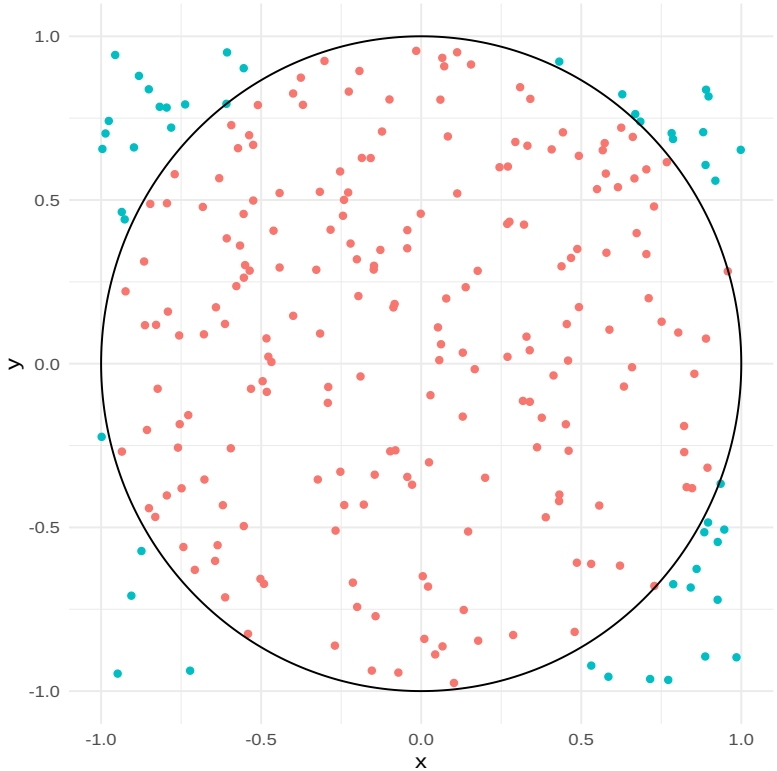
> The question was what are the chances that a Canfield Solitaire laid out with 52 cards will come out successfully? After spending a lot of time trying to estimate them by pure combinatorial calculations, I wondered whether a more practical method than "abstract thinking" might not be to lay it out say one hundred times and simply observe and count the number of successful plays.

- Monte Carlo was the code name given for the project.
  - Inspired by the Monte Carlo casino in Monaco where Ulam's uncle would borrow family money to gamble.



Monte Carlo estimation of π

```
## Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` in the `default_aes` field and elsewhere instead.
```



- Area of circle $= \pi r^2$ and area of square $= 4r^2$
  - (r is the radius)
- A Monte Carlo method to estimate π:
  1. Randomly draw N points within unit square at random
  2. Count the R points which are inside the unit circle
  3. Compute the ratio R/N and estimate π as $4R/N$

Using Monte Carlo to estimate π

In this example, the mathematical statistics of the procedure is:

- Write the parameter we want to estimate (π here) as an expectation
- Represent the parameter as a sample approximation

$$\mathbb{E}X = \int t f(t)\, dt \approx \frac{1}{N} \sum_{i=1}^{N} X_i$$

- The law of large numbers guarantees that as the number of samples we draw increase, the parameter converges to the true parameter value.

Expectation of a random variable

From last slide, we can write the expected value of a random variable X as a sum:

$$\mathbb{E}X = \int_A t \cdot f(t)\, dt \approx \frac{1}{N} \sum_{i=1}^{N} X_i$$

We can replace X with a function $g(X)$. Then the previous equation becomes:

$$\mathbb{E}g(X) = \int_A g(t) \cdot f(t)\, dt \approx \frac{1}{N} \sum_{i=1}^{N} g(X_i)$$

Where we assume to draw $x_i \sim f$

Monte Carlo Integration example

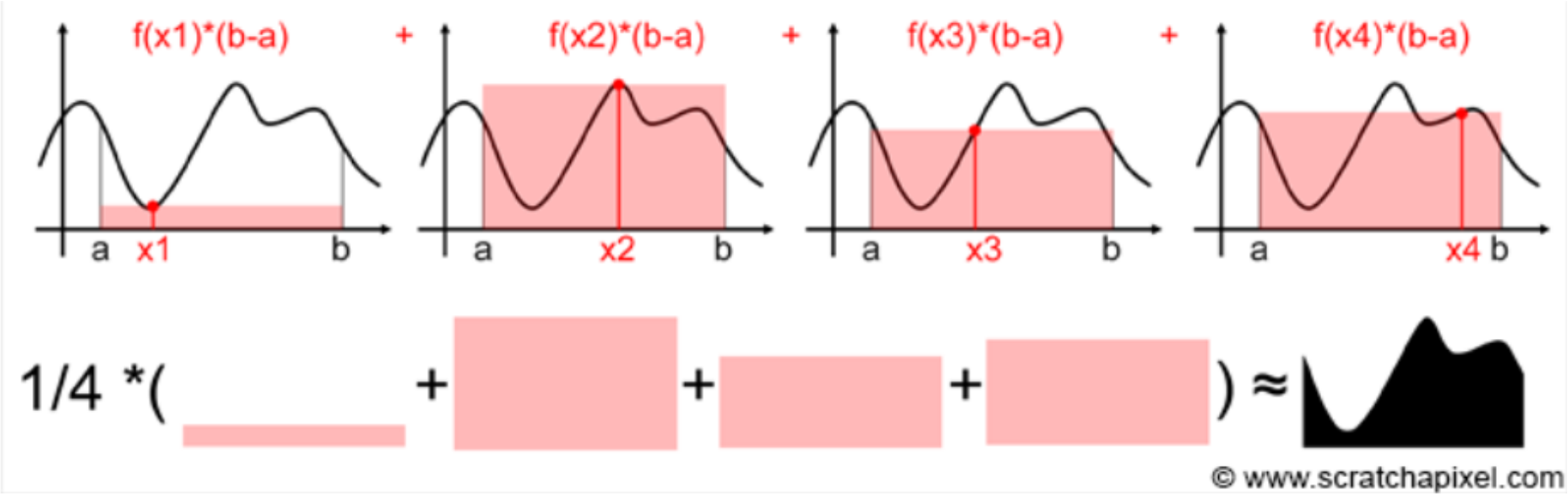
Goal is to evaluate the following definite integral

$$\int_0^1 e^{-x^2/2}\,dx (\approx \frac{1}{N}\sum_{i=1}^{N} g(X_i)).$$

Find a random variable such that the above holds. Need to be careful such that,

- Domain of the random variable and g agree with the above definition.
- One example is,
  - $g(x) = e^{-x^2/2}$
  - X_i is a uniform random variable on (0,1)
  - Then $\mathbb{E}g(X) = \int_0^1 e^{-x^2/2}\,dx \approx \frac{1}{N}\sum_{i=1}^{n} g(X_i)$

Monte Carlo visually



Simulating random variable

- Let's say we know how to simulate random variables that has a uniform distribution on some interval $U \sim [0, 1]$
  - Can do that in `R` with `runif`

> How do we simulate random variables that can have any probability distribution?

In `R`, we can draw random Gaussian variables using the function `rnorm` but how do these functions really work?

Two methods (others exist):

1. Inverse-transform method
2. Acceptance-rejection method

Inverse transform method

- Let's say X is a random variable that has a cumulative distribution function (cdf) F.
- Remember that cdf is the integral of the pdf i.e.

$$F(x) = \int_{-\infty}^{x} f(t)\, dt$$

- If the inverse of $F(x)$ exists, then we can generate X as:

$$X = F^{-1}(U)$$

Example - Exponential distribution

The exponential distribution has a pdf of the form:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

and a cdf of the form:

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

The inverse cdf is:

$$F^{-1}(x) = -\frac{\log(1-x)}{\lambda}$$

To sample from the exponential distribution, we can do the following:

1. Sample $U \sim \text{Uniform}(0, 1)$
2. Set $X = -\frac{\log(1-U)}{\lambda}$

Acceptance-Rejection method

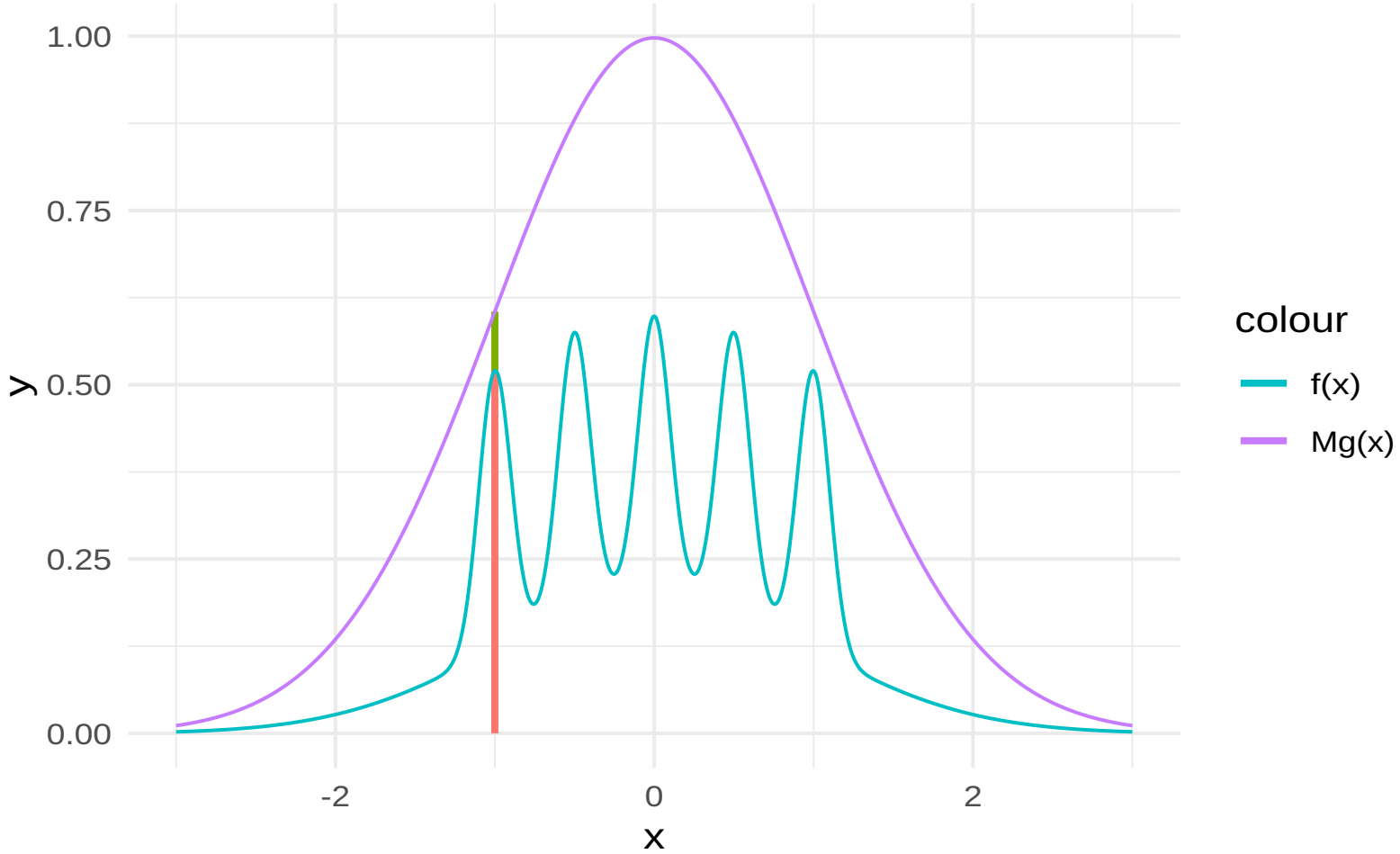
- The problem with the inverse transform method is that you need to know the functional from of the inverse of the cdf
- Acceptance-rejection method is a more general method can simulate 'difficult' distributions

Acceptance rejection method

- Given two probability densities $f(x)$ and $g(x)$, with $f(x) < Mg(x)$ for all x
- If $g(x)$ is a distribution that can be easily sampled, then we can sample $f(x)$ using the following procedure:
  1. Draw a random variable $X \sim g(x)$
  2. Accept X with probability $\frac{f(x)}{Mg(x)}$
  3. Repeat steps 1 and 2 until you have the desired number of random samples

Acceptance-rejection plot

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
```



Problems with the Acceptance-rejection method

- The envelop distribution $g(x)$ needs to closely resemble $f(x)$ for this method to work well
- If $g(x)$ does not match $f(x)$ well,
  - the method will draw a lot of unwanted samples hence is not very efficient
  - becomes very problematic when the data is high-dimensional.
  - Even at dimensions of ~ 10, this method become very inefficient – i.e. you need to draw lots of samples before one sample is accepted

Monte Carlo simulation examples

THE UNIVERSITY OF SYDNEY

Collecting Coles or Woolworths kids toys

- Collect one little shop when you spend $30
- How many $30 shops do I need to collect all 30 items?

Theoretical aspect (before we see the Monte Carlo)

Expected number of shops you need to do:

$$\mathbb{E}T = n\left(\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n}\right) \approx n\log n + 0.5772n + 0.5$$

n =30

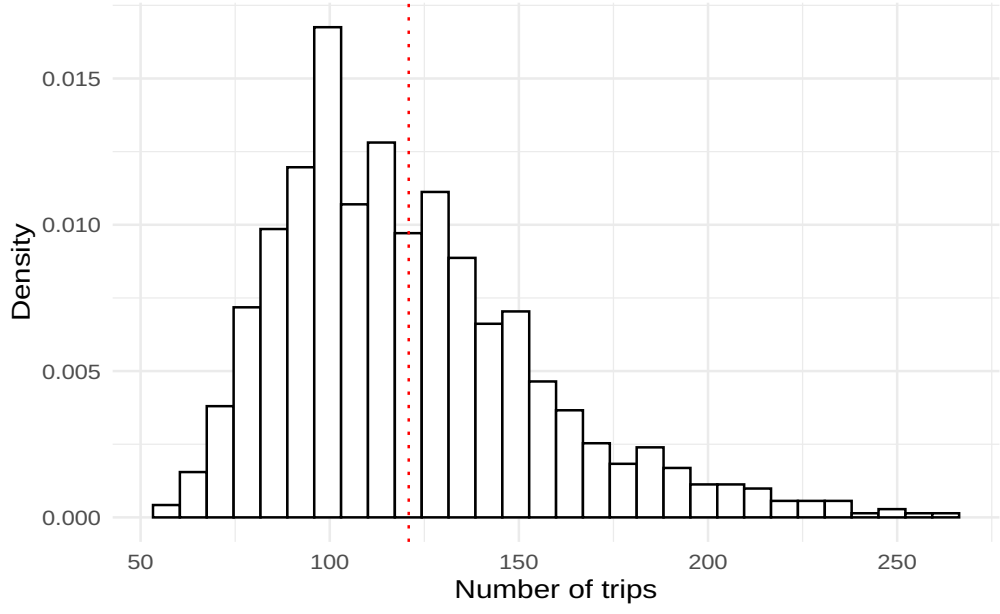
- where, T is the time to collect all items, n is the number of items to collect

So, for a Coles little shop would expect, $\mathbb{E}(T) \approx 120$

Monte Carlo simulation of Cole's little shop

```
n.shops.sim <- c()
for(i in 1:1000) {
  collected <- c()
  n.shops <- 0
  while(length(collected) < 30) {
    newitem <- sample(30,1)
    collected <- union(collected, newitem)
    n.shops <- n.shops + 1
  }
  n.shops.sim <- c(n.shops.sim, n.shops)
}
mean.sim <- mean(n.shops.sim)
ggplot(data.frame(x = n.shops.sim)) +
  theme_minimal() +
  geom_histogram(aes(x = x, y = ..density..),
                 colour = "black",
                 fill = "white") +
  labs(x = "Number of trips", y = "Density") +
  geom_vline(xintercept = mean.sim,
             linetype = "dotted",
             colour = "red")
```

```
## i Please use `after_stat(density)` instead.
```




```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
```

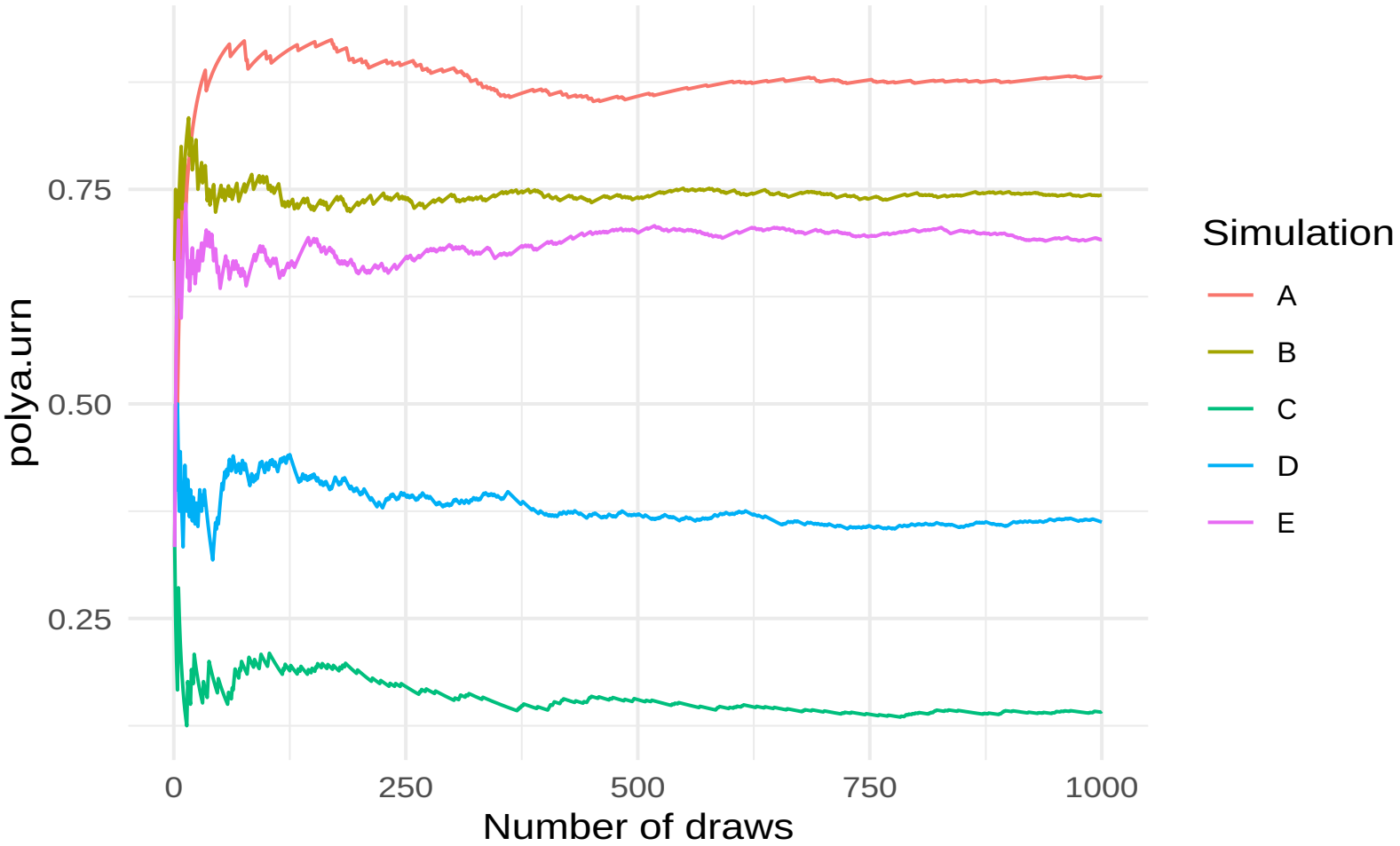
Polya urn problem

- Let an urn start with 1 black ball and 1 white ball.
- Repeatedly draw a single ball from the urn
- Each time a ball is drawn
  - you put back that ball plus one extra ball of the same colour back into the urn.

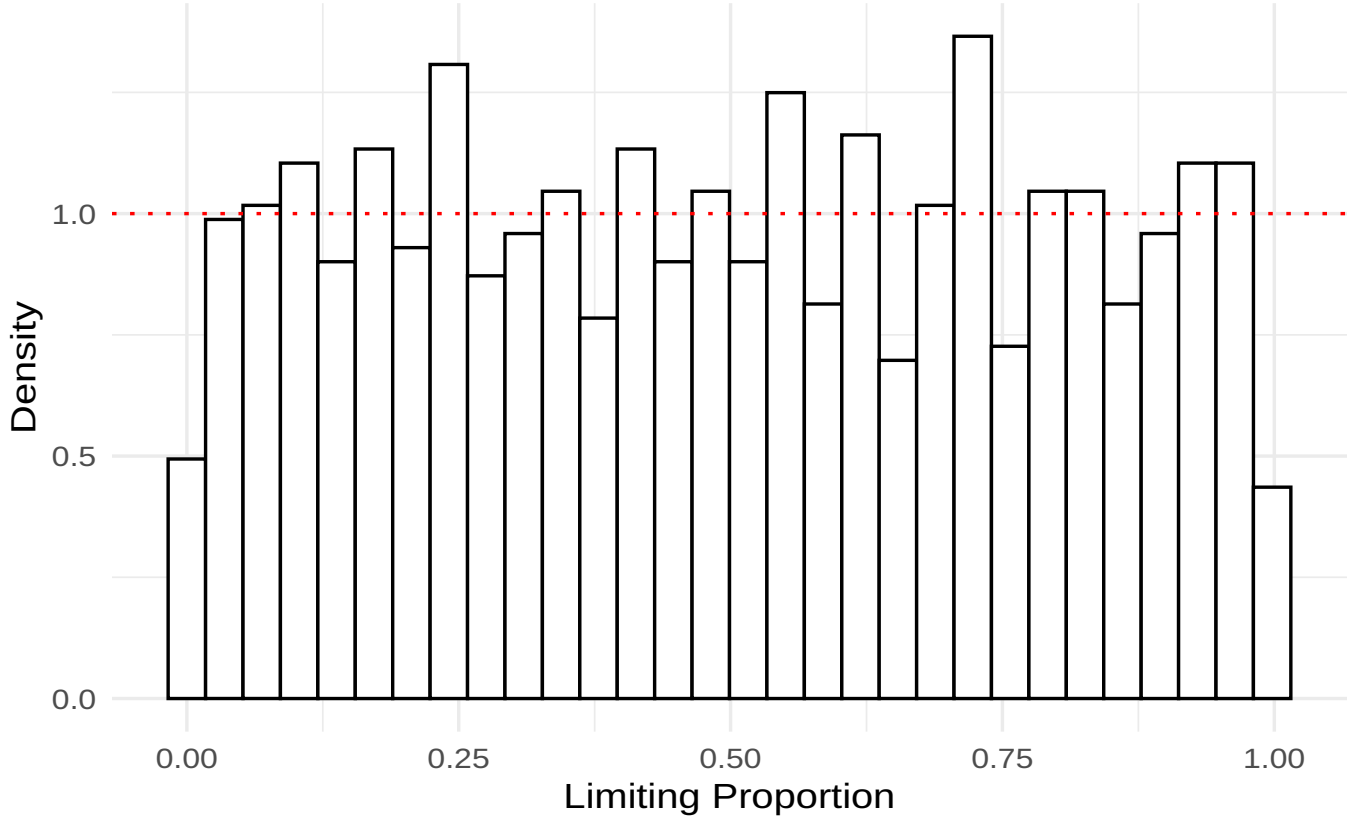
> After 1000 draws, how many balls of each colour do you expect to be in the urn?

- This is an example of using Monte Carlo to simulate a process

Monte Carlo simulation of the Polya urn problem



Asymptotic Uniform distribution of the proportion of black balls



- In the limit the distribution is $U(0,1)$

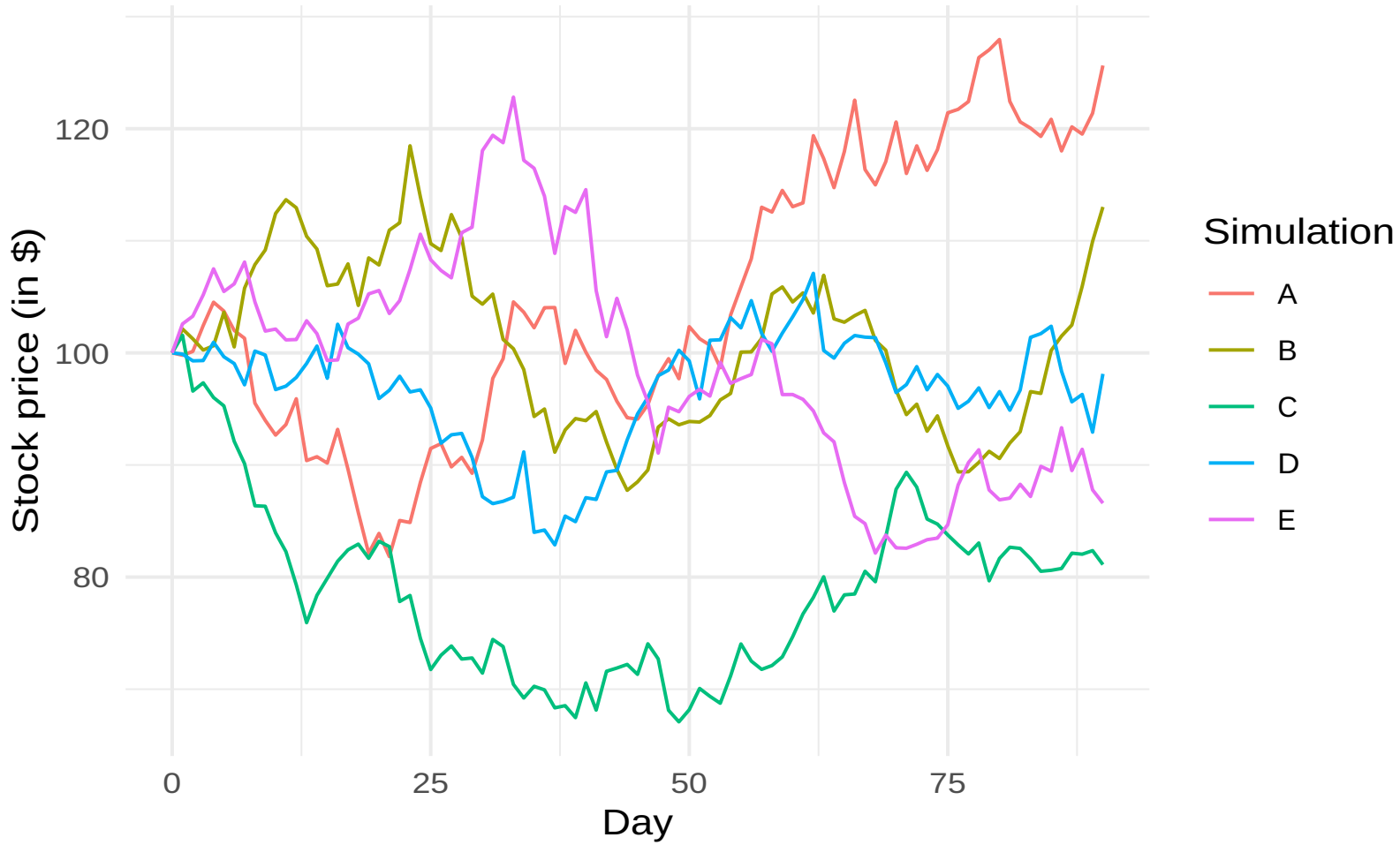
Simulate stock price as a random (stochastic) model

Equation for the future price S of a stock:

$$dS = S(\mu dT + \sigma\sqrt{dT}\mathcal{N})$$

- μ is the growth rate of the stock price
- σ is known as the volatility.
  - Think of this as the variance of the stock price
- $\mathcal{N}$ is a normal random variable
- dT is one unit of time

Examples of a stock price



Options

A call option gives you the right to buy a stock at a particular price at expiry

> E.g. 90 day call option with $105 strike. Let's say your stock is currently priced at $100, if it hits $108 at day 90, then you can exercise the option and make a $3 profit. If its price is below $105 at day 90, then you make nothing.

A put option gives you the right to sell a stock at a particular price at expiry

Option pricing with Monte Carlo

- How much would you pay for a call option? Think of this as the amount of insurance premium.
- Using Monte Carlo simulation, simulate many realisations of the stock price. For each realisation, calculate the option payoff
- The price you should pay for the option is the expected value of the profit you should make

References

Shonkwiler, R. W. and F. Mendivil (2009). *Explorations in Monte Carlo Methods*. Springer Science & Business Media.