November 2024

Introduction to C++: Workshop Five

Dr. Alexander Hill

a.d.hill@liverpool.ac.uk

LIV.INNO

Last Week

- Classes

Resources

- alex-hill94.github.io/#WS5
- https://www.w3schools.com/cpp/cpp_classes.asp
- History of Monte Carlo - https://library.lanl.gov/cgi-bin/getfile?00326866.pdf
- Basics of Monte Carlo Simulations - https://www.youtube.com/watch?v=OgO1gpXSUzU

Challenge Eight (Homework)

- Create a class called Country with attributes:
  - Population, size, national language
- Include a method called Greet, which outputs “Hello!”
- In main(), create objects from the country class called: UK, France, Spain
- Use constructors to initialise the attributes mentioned above
- Print all attributes and run Greet
- Bonus: inside Greet include an ‘if’ statement that changes language based on the national language

Archie

```cpp
class Country{
public:
int population;
double size;
string language;
Country(int x, double y, string z){
population = x;
size = y;
language = z;
}
void greet();
};

void Country::greet(){
if(language == "French"){
cout << "Bonjour"<<endl;
}else{
if(language == "Spanish"){
cout << "Hola" << endl;
}else{
cout << "Hello" << endl;
}
}
}
```

```cpp
int main(){
Country England(60,10,"English");
Country France (50,20,"French");
Country Spain (40,30,"Spanish");
cout<< "England population: "<< England.population << endl;
cout<<"England size: " << France.size << endl;
cout << "England greeting: ";
England.greet();
cout << "France population: " << France.population << endl;
cout << "France size: " << France.size << endl;
cout << "France greeting: ";
France.greet();
cout << "Spain population: " << Spain.population << endl;
cout << "Spain size: " << Spain.size << endl;
cout << "Spain greeting: ";
Spain.greet();
return 0;
}
```

Good use of constructors

Clean class structure

If statements work, but are a little confusing

```
$ ./a.out
England population: 60
England size: 20
England greeting: Hello
France population: 50
France size: 20
France greeting: Bonjour
Spain population: 40
Spain size: 30
Spain greeting: Hola
```

```
class country{
public:
int population, size;
string national_language;
country(int x, int y, string z){
population = x;
size = y;
national_language= z;

}
void Greet(string l);
};


void country::Greet(string l)
{ if(l=="English")cout<< "Hello"<<endl;
if(l=="French")cout<< "Bonjour"<<endl;
if(l=="Spanish")cout<< "Hola"<<endl;
}
```

```
int main()
{
country UK(67600000, 244376,"English"), France(66548530,547557,
"French"), Spain(47473373,504782,"Spanish");

cout<<"Total population of UK is: "<<UK.population<<endl;
cout<<"Total population of France is: "<<France.population<<endl;
cout<<"Total population of Spain is: "<<Spain.population<<endl;
cout<<"\n"<<endl;


...
```

Shirsendu

Good use of constructors

Clean class structure

What if national_language is not a match? Consider a default or a warning

Total population of UK is: 67600000
Total population of France is: 66548530
Total population of Spain is: 47473373

Total size of UK is: 244376 sq km
Total size of France is: 547557 sq km
Total size of Spain is: 504782 sq km

National Language of UK is: English
National Language of France is: French
National Language of Spain is: Spanish

Hello
Bonjour
Hola

Rosie

```
class country {
public:
int population;
int size;
string national_language;
country(int p, int s, string nl) {
population = p;
size = s;
national_language = nl;
}
void greet() {
if(national_language == "English") {
cout <<"Hello!" <<"\n";
}
if(national_language == "French") {
cout <<"Bonjour!" <<"\n";
}
if(national_language == "Spanish") {
cout <<"Hola!" <<"\n";
}
}
};
```

UK: 68350000, 243610, English
Hello!
France: 68170000, 551695, French
Bonjour!
Spain: 48370000, 506030, Spanish
Hola!

What if national_language is not a match? Consider a default or a warning

Ben

```
class Country {
public:
int population;
double size;
string national_language;

Country (int pop, double si, string nat_lang) {
population = pop;
size = si;
national_language = nat_lang;
}

void greet() {
if (national_language == "English") {
cout << "Hello!" << endl;
}
if (national_language == "French") {
cout << "Bonjour!" << endl;
}
if (national_language == "Spanish") {
cout << "Hola!" << endl;
}
}
};
```

```
$ ./a.out
1234234
2934.2
Spanish
Hello!
Bonjour!
Hola!
```

What if national_language is not a match? Consider a default or a warning

Structs

- Similar to classes
- Used mainly to group similar/related data
- Structs can only inherit from other structs, not classes

```
struct Point {
int x;
int y;
};

int main() {
Point p1;
p1.x = 10;
p1.y = 20;

std::cout << "Point p1: (" << p1.x << ", " << p1.y << ")" <<
std::endl;
return 0;
}
```

Challenge Nine

- Download https://alex-hill94.github.io/Workshop/Intro_to_C++/soln.cpp
- Adapt this script so that the write_out function is replaced with a class
- Use the constructor to create the data.py file, and use a method within the class to save the vectors

Headers

- To avoid your script becoming too verbose, you can save useful functions and classes in a header file

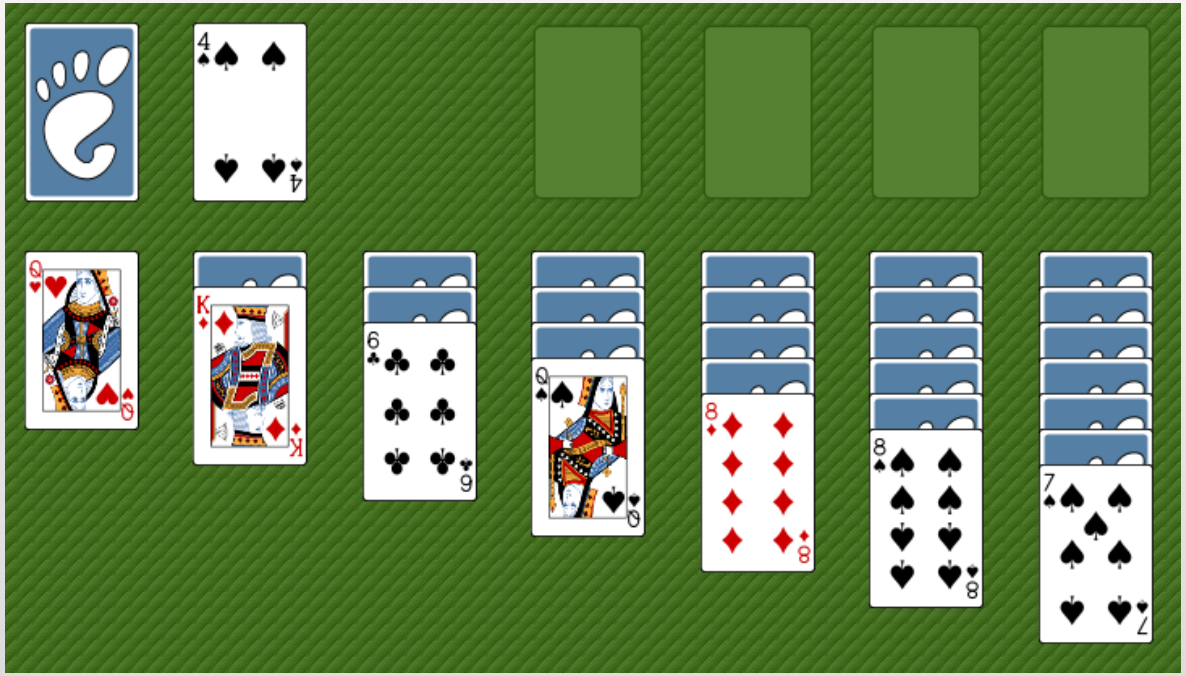
- Save this file as my_funcs.h, and import using `#include "my_funcs.h"`

Monte Carlo Methods

- History, general concepts
- Examples
- Markov Chains

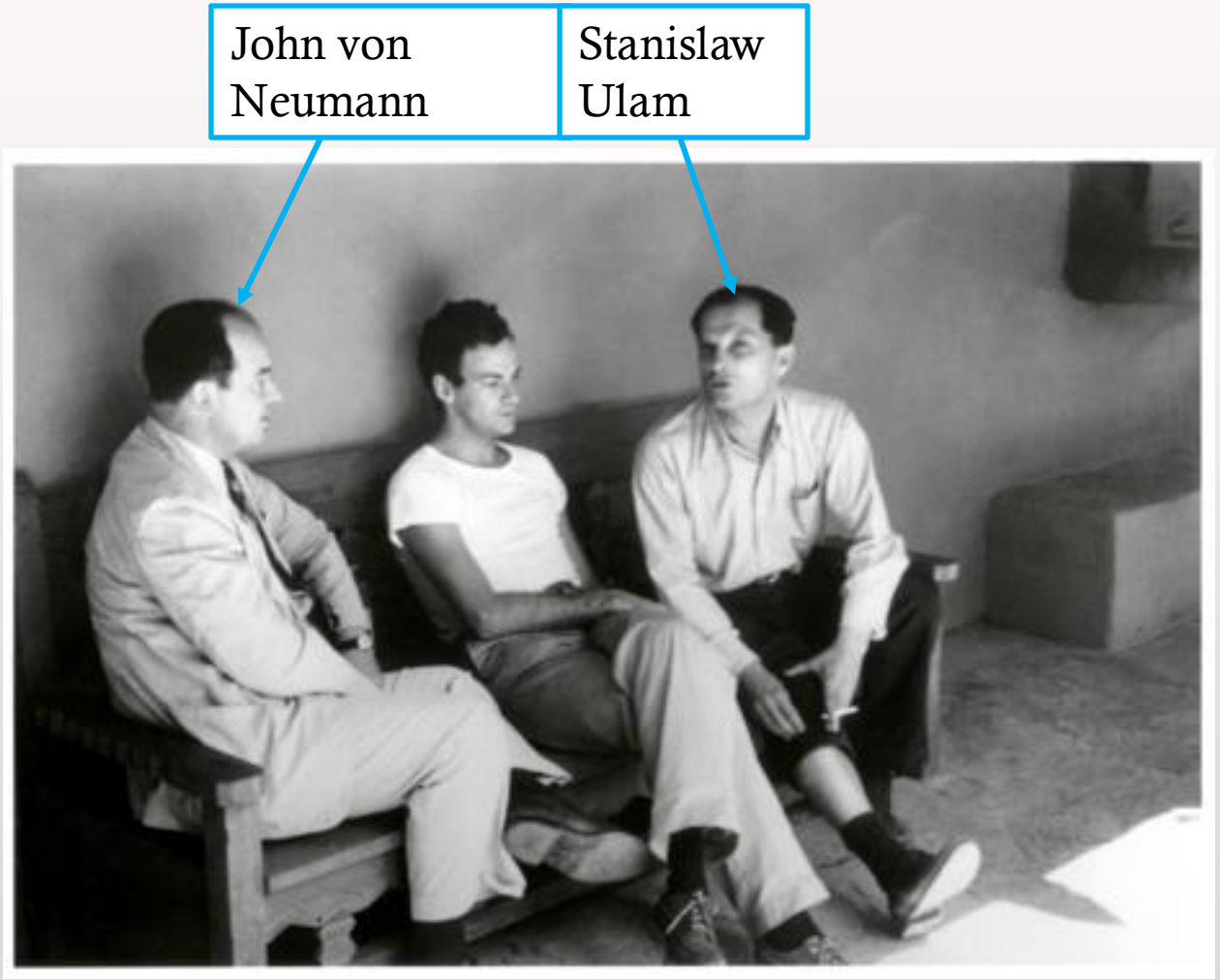
Monte Carlo History

- Initially created by Stanislaw Ulam, Polish-American scientist
- While ill, he wanted to know the probability of winning a game of solitaire
- Finding the pure calculations too hard, he adopted a statistical approach
- Playing the game would take too long, so he asked von Neumann to simulate it on an early (huge) computer



Monte Carlo History

- The Monte Carlo Method is a mathematical technique used to estimate the outcomes of an uncertain event
- Developed during WWII in relation to the Manhattan project
- Named after the Monte Carlo Casino, frequented by Ulam's uncle, as chance is important to the modelling approach
- Idea: use random sampling of inputs to explore outputs complex systems



Monte Carlo History

- The challenge of constructing the atom bomb involved **neutron diffusion**
- Too challenging to be addressed analytically, needed a numerical approach
- Some of the first computers tried an exhaustive numerical approach (plug in many numbers into the equation and calculate the result), but this was too slow due to high dimensionality of the problem
- Monte Carlo methods were found to be remarkably successful

MANIAC Computer

Monte Carlo Basics

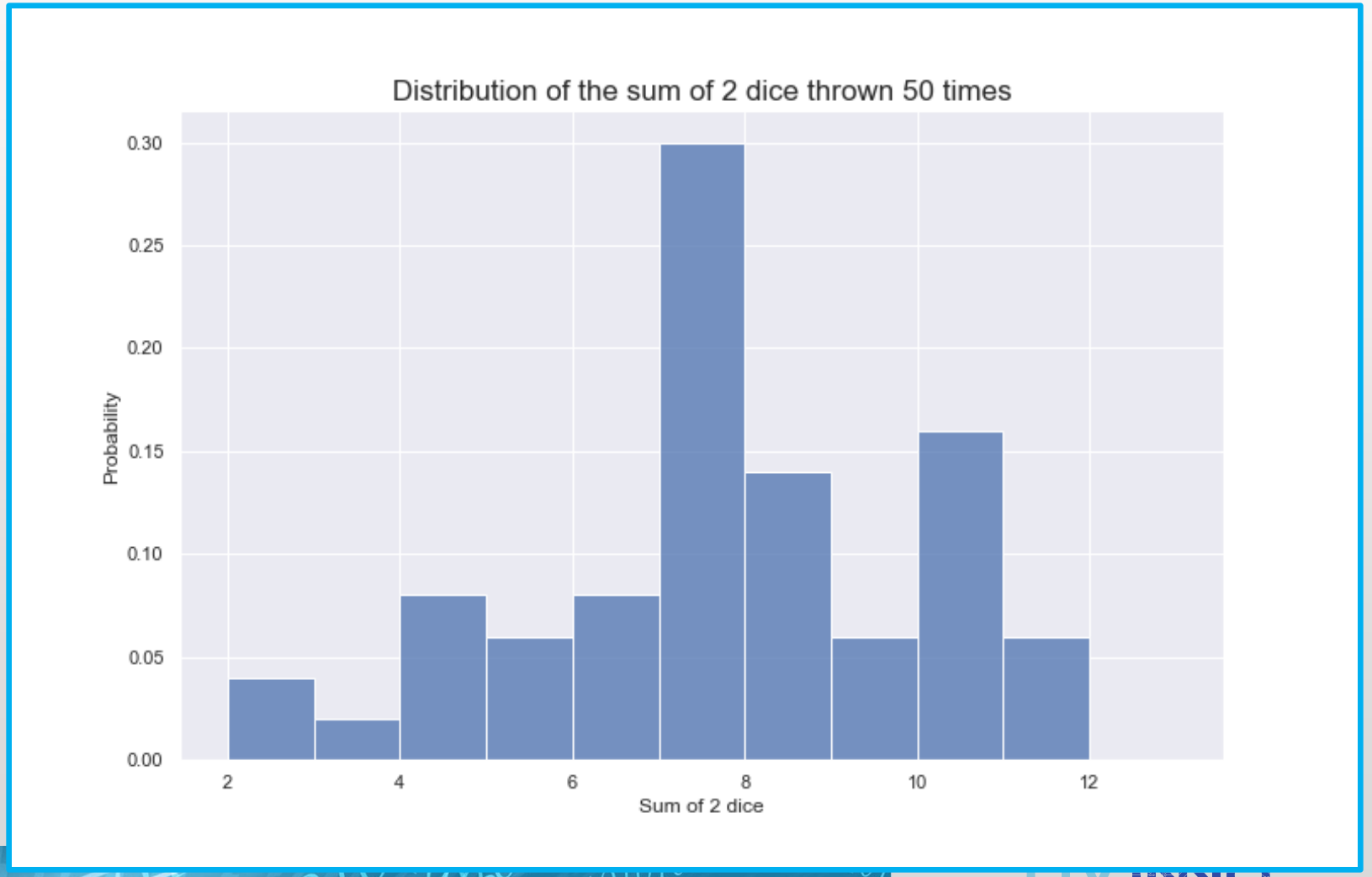
- A method of estimating the value of an unknown quantity using the principles of inferential statistics
- Key concepts:
  - Population: the universe of possible examples
  - Sample: a proper subset of a population
  - A random sample tends to exhibit the same properties as the population from which it is drawn
  - Use the sample to infer the statistics of the population

Law of Large Numbers

- Law of large numbers:
  - In repeated independent tests with the same probability, *p*, of a particular outcome in each test, the chance that the fraction of times that the outcome occurs differs from *p* converges to zero as the number of trials goes to infinity

Law of large numbers

- Example: roll two dice to predict probability of getting a given number in total



Results

Simulate betting a pocket for 20 trials of 1000 spins each
Exp. return for Fair Roulette = 3.68%, +/- 27.189% with 95% confidence
Exp. return for European Roulette = -5.5%, +/- 35.042% with 95% confidence
Exp. return for American Roulette = -4.24%, +/- 26.494% with 95% confidence

Simulate betting a pocket for 20 trials of 100000 spins each
Exp. return for Fair Roulette = 0.125%, +/- 3.999% with 95% confidence
Exp. return for European Roulette = -3.313%, +/- 3.515% with 95% confidence
Exp. return for American Roulette = -5.594%, +/- 4.287% with 95% confidence

Simulate betting a pocket for 20 trials of 1000000 spins each
Exp. return for Fair Roulette = 0.012%, +/- 0.846% with 95% confidence
Exp. return for European Roulette = -2.679%, +/- 0.948% with 95% confidence
Exp. return for American Roulette = -5.176%, +/- 1.214% with 95% confidence

Regression to the Mean

- Following an extreme, independent random event, the next random event is likely to be less extreme
- Example: 10 reds in a row in roulette is $\frac{1}{2^{10}}$
- In the next 10 spins, it is likely that you will get fewer than 10 reds (E[red] = 5)
- For the 20 spins in total, you will get closer to the expected mean of 50% reds than you had in the first 10 spins
- Subtly different to the gambler's fallacy…

Gambler's Fallacy

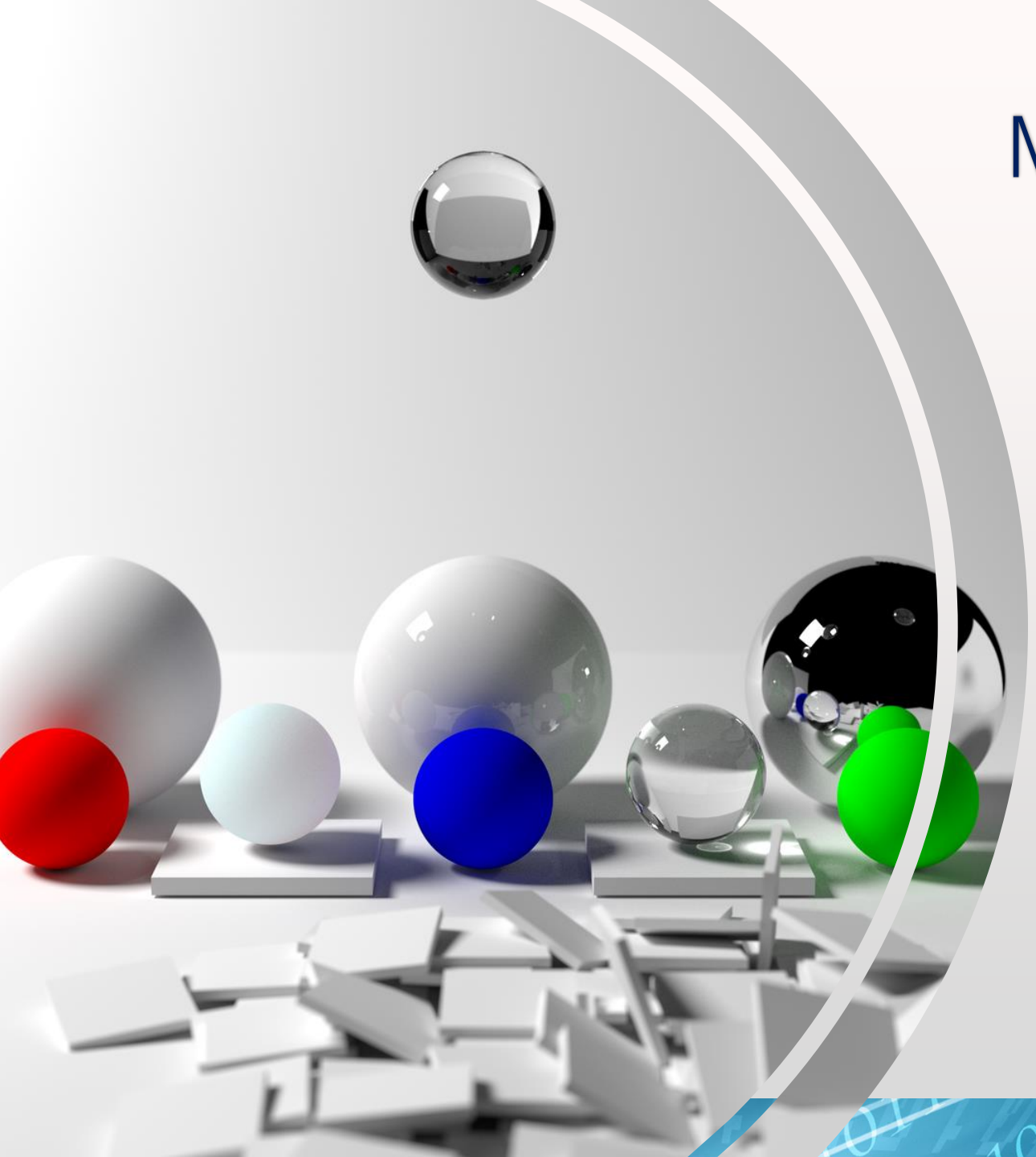
- This is the idea that following a statistically unlikely set of results (e.g. many instances of red in a row in roulette), people believe that the next sample will work to bring you back to the average
- Example in 1913 at Monte Carlo casino: after 15 blacks in a row, there was a rush to bet on red
- After 26 reds in a row, the casino had made a fortune

How many samples do we need?

- Depends on the sample variance:
  - $\text{Var} = \frac{\sum_{x \in X}(x - \mu)^2}{N}$
  - $\sigma = \sqrt{\frac{\sum_{x \in X}(x - \mu)^2}{N}}$
- The larger the variance, the more samples you need to accurately estimate the distribution

Monte Carlo Experiments

- Generate random inputs according to some distribution
- Use these to conduct an experiment
- Analyse the output using statistics and make predictions of the simulation's or process' properties



Monte Carlo Examples

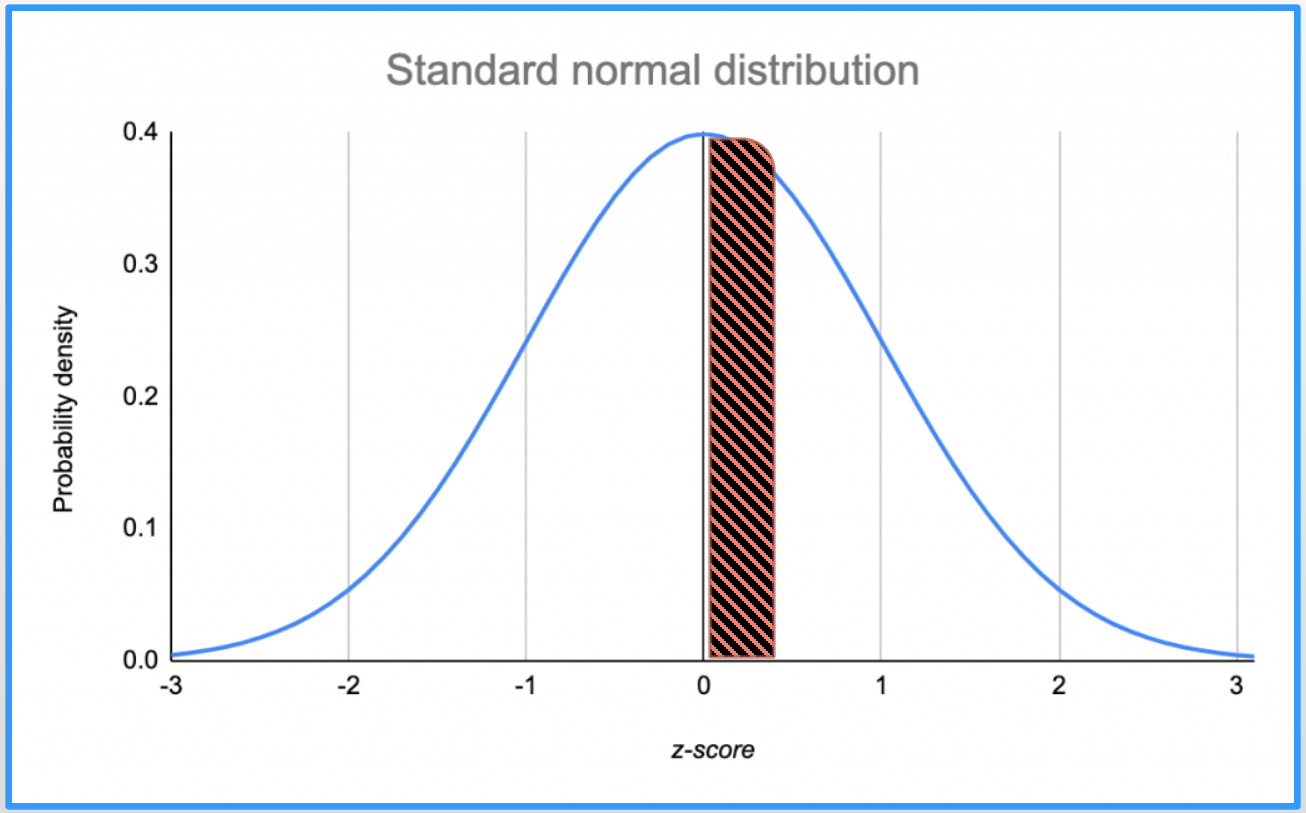
- Business applications:
  - Estimate probability of cost overrun in large projects
  - Estimate likelihood of an asset gaining/losing value
- Path tracing in video games (trace random samples of possible light paths)
- AI for video games (tree search for next best move)
- Computational physics, physical chemistry

Random Numbers

- For random events, the probability that any given random event occurs is given via probability distribution function
- This captures the notion of relative frequency with which random events occur
- For discrete random events (heads, tails), the sum of the probability of each value occurring must add up to one
- For continuous events (temperature) it's trickier, as we can't give infinite real numbers separate probability values

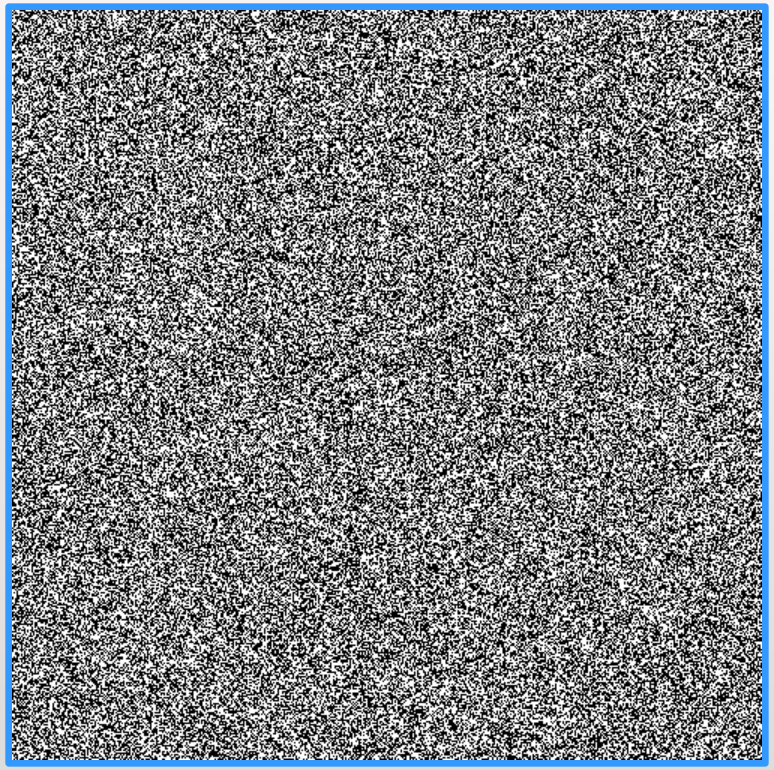
Continuous random variables

- Continuous random events are defined using a probability density function
- This gives the probability of a random variable lying between two values
- It defines a curve in $x - y$, where the range spanned by x are the possible values, and the area under the curve between two points gives the probability that a sample will fall in that range



Generating Random Numbers in C++

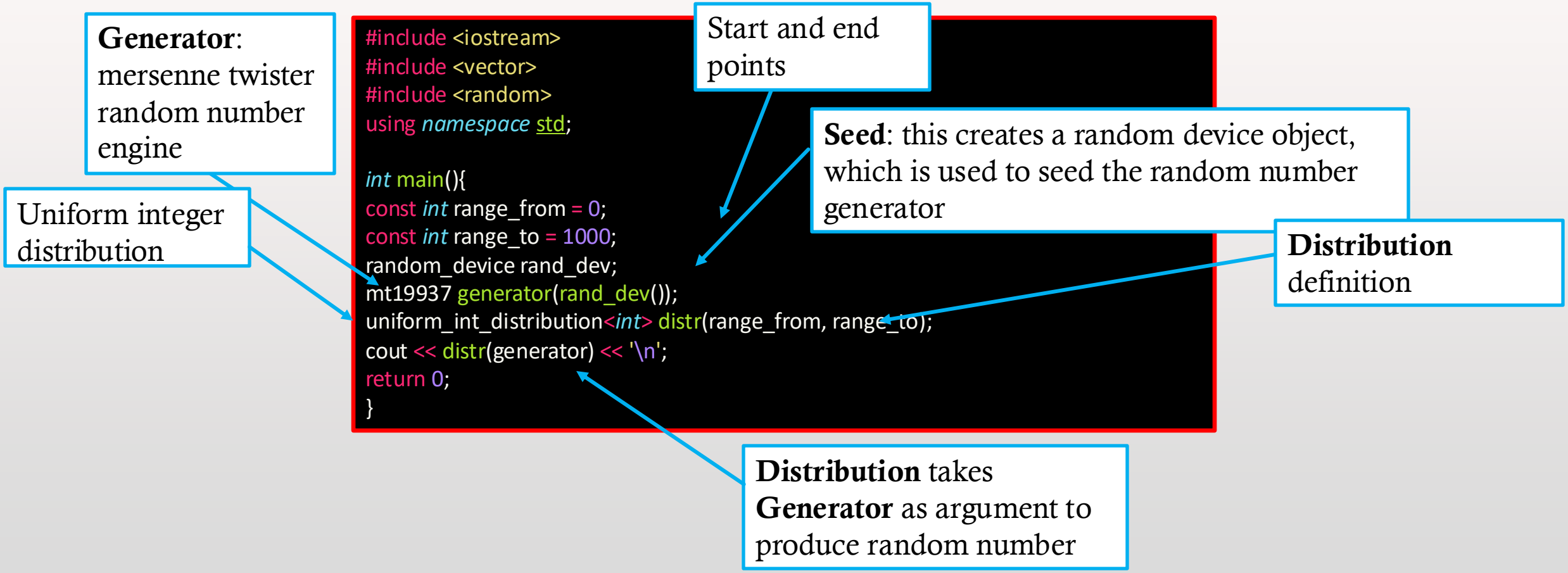
- We'll use the <random> header to generate random numbers (https://cplusplus.com/reference/random/)
- This uses **Generators** and **Distributions** to generate random numbers
- **Generator:** an object that generates uniformly distributed numbers
- **Distribution:** an object that transforms sequences of generated numbers into numbers numbers that follow a specific random variable distribution



Generating Random Numbers in C++

1. Create a random number seed
2. Create a **generator** using the seed
3. Create a **distribution** object
4. Pass the generator into the distribution

Generating random numbers in C++




```
#include <iostream>
#include <vector>
#include <random>
using namespace std;

int main(){
const int range_from = 0;
const int range_to = 1000;
random_device rand_dev;
mt19937 generator(rand_dev());
uniform_int_distribution<int> distr(range_from, range_to);
cout << distr(generator) << '\n';
return 0;
}
```

Example

- A ‘simulation’ takes one input (uniform dist.) and produces one output
- What is the output distribution?

```
#include <iostream>
#include <vector>
#include <random>
#include <cmath>
#include <fstream>
#include "funcs.h"
using namespace std;

int main(){
int n_sim_runs = 1e5;
vector<double> inputs(n_sim_runs);
vector<double> outputs;
string fname = "data.py";
string xname = "x_range";
string yname = ”y_range";

for (auto i = 0; i < inputs.size(); ++i){
rnum(inputs.at(i));
}

simulation(inputs, outputs);

write_out_vec(fname, xname, inputs);
write_out_vec(fname, yname, outputs, false);
system("ipython test.py &");
return 0;
}
```

Some auxiliary functions I’ve made

Create inputs with length ‘n’

Blank outputs

Replaces inputs elements with random numbers

Runs ‘simulation’, outputs is now the simulation’s output

Writes out data

Runs plotting python script

Example

Random number function takes an element the the input vector, and makes it a random float between -1000 and 1000

```
void rnum(double& input){
const double range_from = -1000;
const double range_to = 1000;
random_device rand_dev;
mt19937 gen(rand_dev());
uniform_real_distribution<double> distr(range_from, range_to);
input = distr(gen);
}
```

Example

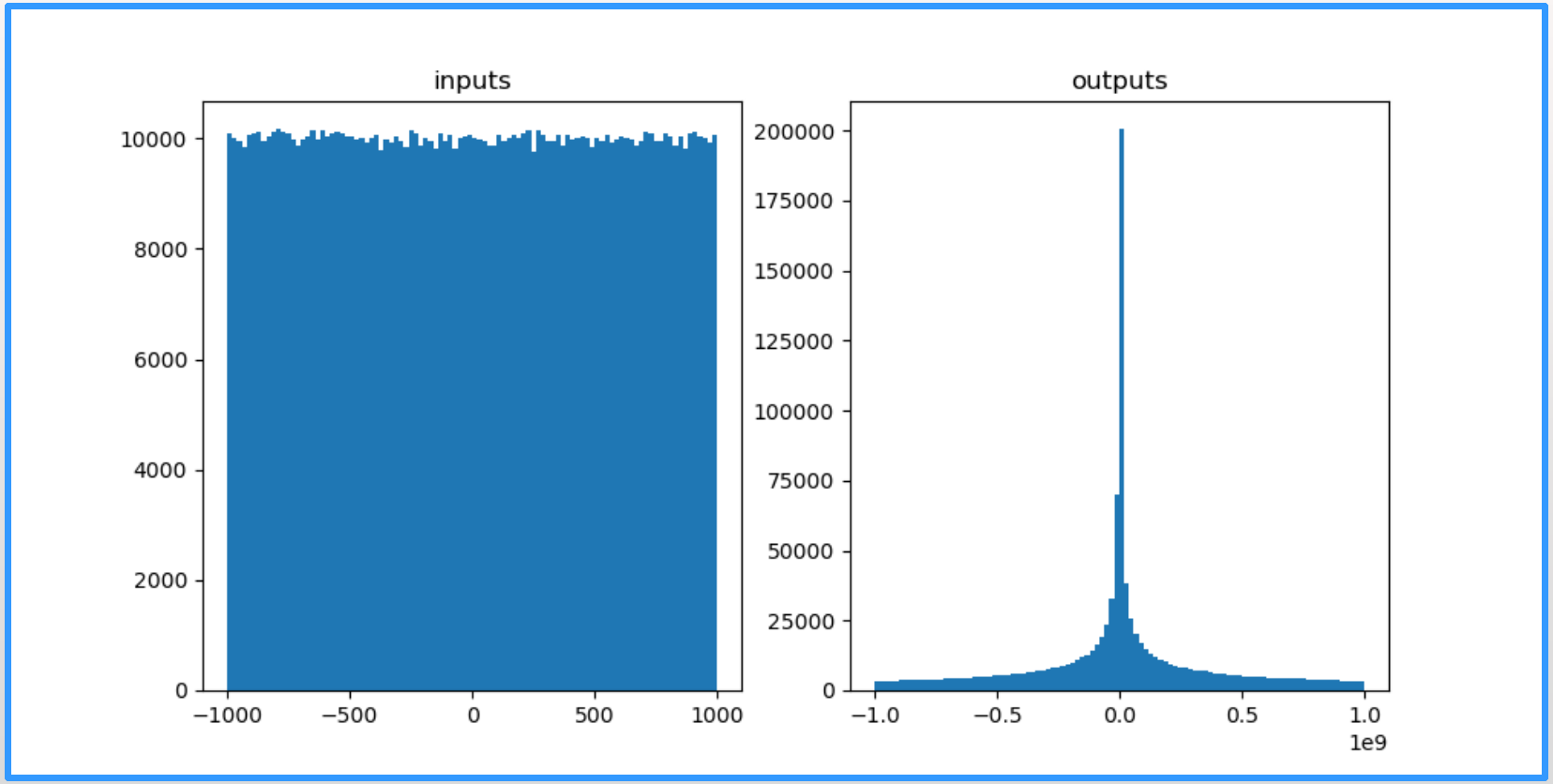
Input is fixed, output can change

```
void buffer(const double& input, double& output){
// Computes a polynomial from a given input, modifies the output accordingly
output = pow(input, 3) - 3*pow(input, 2) + 9.2 * pow(input,1) + 4.3;
}

void simulation(const vector<double>& input, vector<double>& output)
{
// Simulation loops over inputs and modifies the output vector
double temp_val;
for (auto i: input){
buffer(i, temp_val);
output.push_back(temp_val);
}
}
```

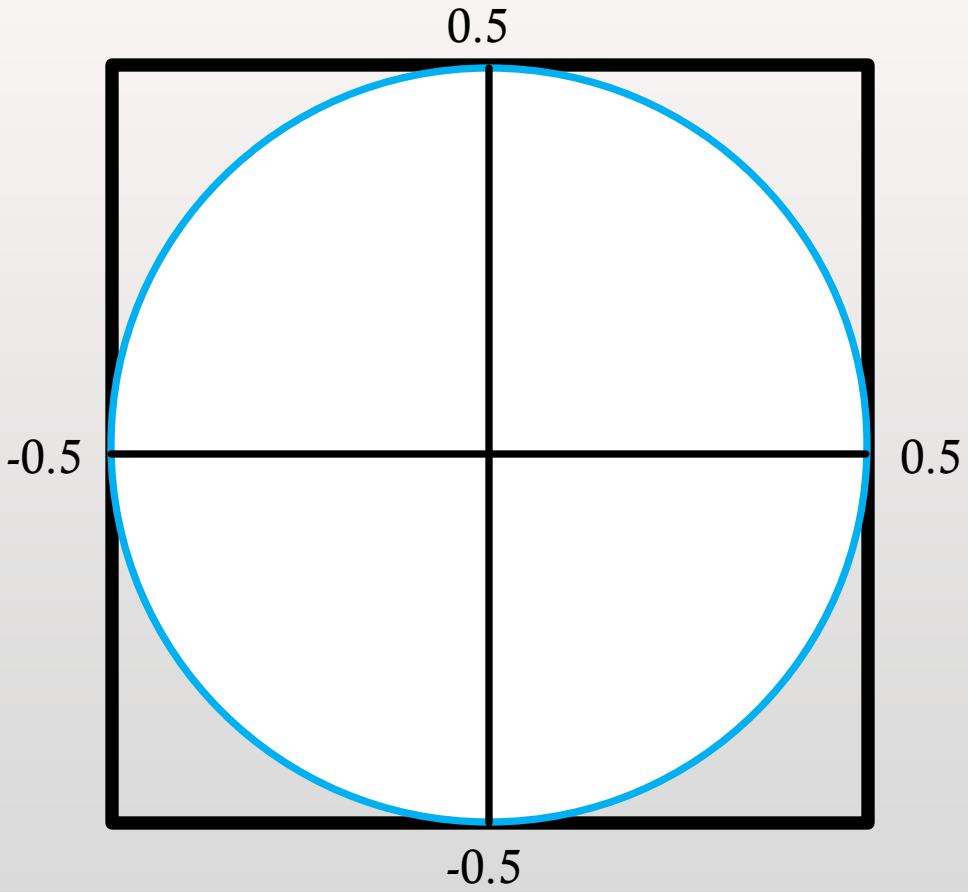
Example

1e1 -> 1e6
draws



Challenge Ten: Area of a Circle

- For the circle described by $x^2 + y^2 = 0.5^2$, compute its area using a Monte Carlo method
- i.e. draw random x and y between -0.5 and 0.5, and compute the fraction of draws that satisfy $x^2 + y^2 < 0.5^2$
- How many draws do you need to get ~1% error on πr^2 ?



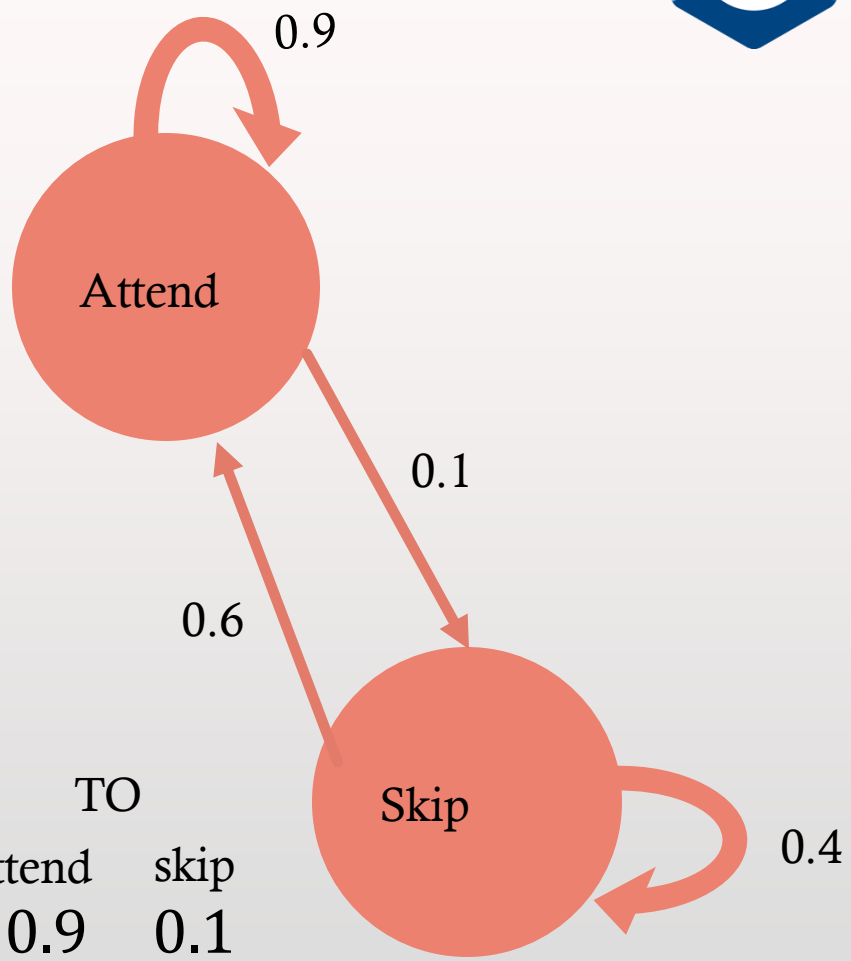
Markov Chain Monte Carlo (MCMC)

- Monte Carlo: estimate the expected value or probability density of some unknown space by drawing independent random values
- For high-dimension probabilistic models, Monte Carlo sampling may not be effective, as volume of sample space grows exponentially with additional parameters
- MCMCs try to sample more intelligently, the next random draw depends on the current one

Andrey Markov

Markov Chains

- A simple Markov Chain uses stochastic processes to determine the evolving state of a system
- Consider this system, it describes whether someone attends class given their previous attendance
- E.g. if you attend class one week, there's a 90% chance you will the next

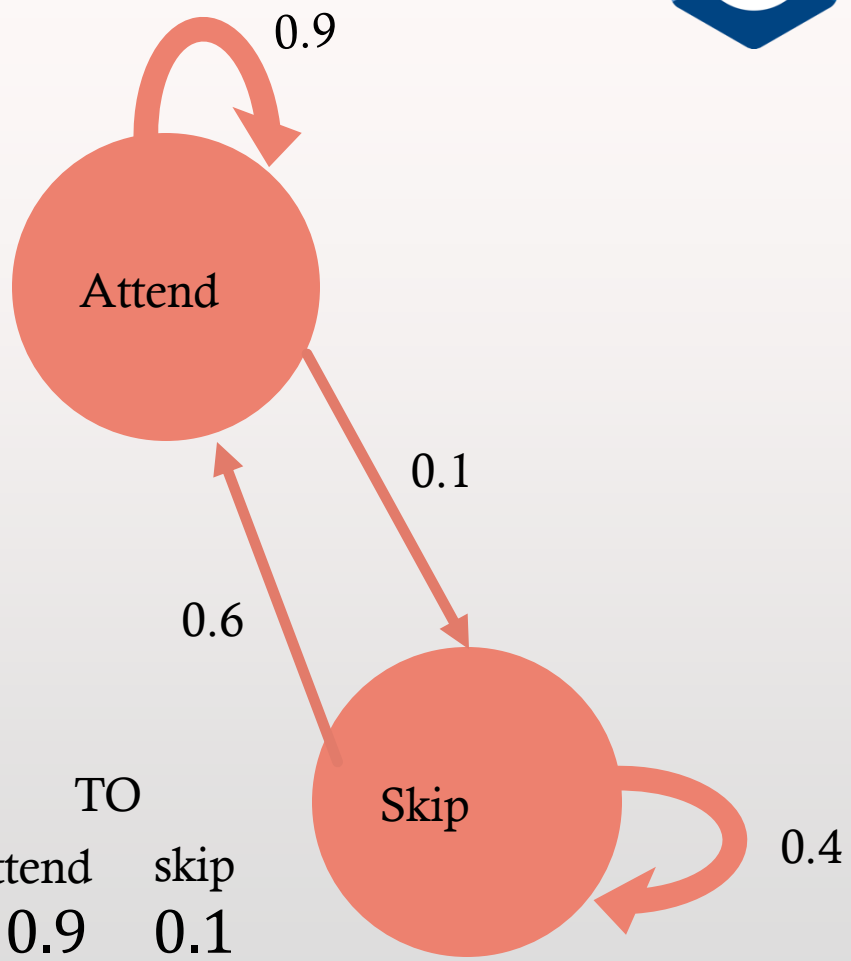


		TO attend	skip
FROM	attend	0.9	0.1
	skip	0.6	0.4

Transition matrix

Markov Chains

- Start with initial state of attendance: $X_0 = [1,0]$
- $X_1 = X_0 T = [0.9, 0.1]$
- $X_2 = X_1 T = [0.87, 0.13]$
- In the long-run, you approach a **steady state**, i.e. $X_{n+1} = X_n$

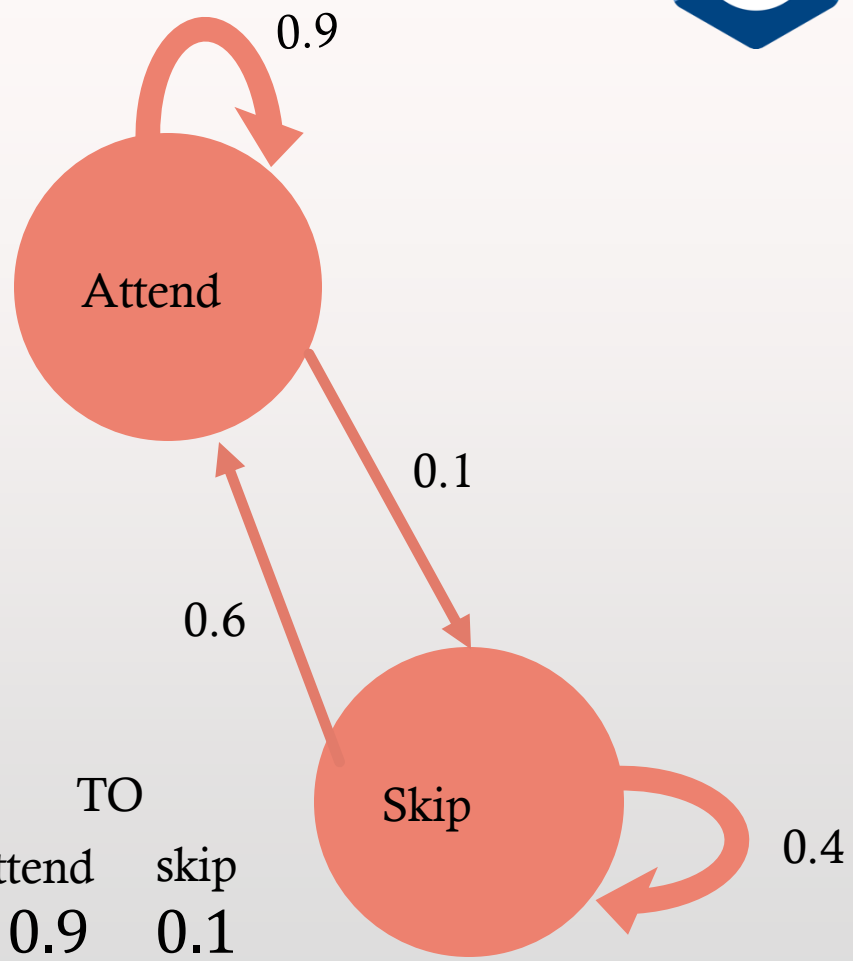


		TO attend	skip
FROM	attend	0.9	0.1
	skip	0.6	0.4

Transition matrix

Markov Chains

- $X_s - X_s T = 0$
- $X_s(I - T) = 0$
- $[x, y]\begin{pmatrix} 0.1 & -0.1 \\ -0.6 & 0.6 \end{pmatrix} = 0$
- $0.1x - 0.6y = 0$ and $x + y = 1$
- $X_s = \left[\frac{6}{7}, \frac{1}{7}\right]$



		TO	
		attend	skip
FROM	attend	0.9	0.1
	skip	0.6	0.4

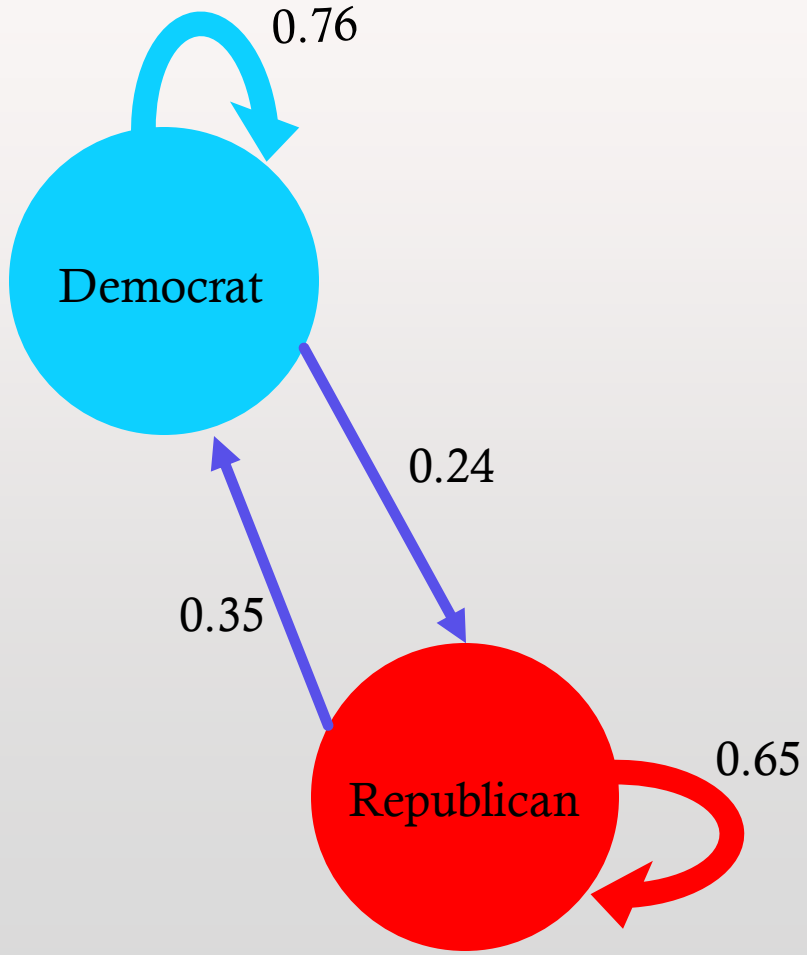
Transition matrix

Markov Chains

- Markov chains can also be used to generate a sequence of random variables where the current value is dependent on the value of the prior value
- MCMCs are Monte Carlo methods where a Markov chain is used to draw samples
- The idea is that the chain will settle (find equilibrium) on the desired quantity we are inferring

Challenge Eleven

- Create a class that generates random numbers from a uniform distribution
- Create a Markov Chain class that predicts which US party will win the next election (lookup matrices, matrix mutliplication, if statements etc.)
- Assume initially a Dem is in power X0 = [1,0], create a method in the MC class that calculates numerically the steady state vector, i.e. the probability that in a given year the holder will be Democrat or Republican
- Create another method that uses random draws from the random number class to stochastically predict who will be in power for each of the next 20 cycles
- Create a figure showing how the holder of office changed over the 20 cycles



Next week

- Introduction to your group project

Thanks!