5. Random walks

Last time

- Counting, frequencies
- Probability distribution
- Variability: mean and variance

Goals for today

- Suggestions from Problem Set 1
- Model of random motion: **random walk**
- Behaviour in time
- Different types of walks

Messages from Problem Set 1

- Make sure *code runs* —- version that you hand in
- Don't copy and paste: build on what you already have
- If there is a question, write down your best guess
- Include all your work, e.g. Python code
- If something doesn't work after some time, *ask for help*
- Save thinking time for the hard stuff: "how do I translate these words into code", not plotting stuff

- File name that includes PS number and your name (1 first name + 1 last name)
- `@belapsed` from BenchmarkTools: returns time
- Use Jupyter notebook from now on

Calculations with units!

- Can calculate with units:

```
using Unitful, Unitful.DefaultSymbols
ratio = 1s / 1μs
Unitful.dimension(ratio). # NoDims
upreferred(ratio)    # convert to SI units
```

Plotting

- `!` versions like `plot!()` add to pre-existing figure. Versions without `!` create new figure
- `plot()` creates new, empty figure
- `for` loops don't return anything. Evaluate plot object to see plot
- Use points to draw discrete data. Lines are only a guide to the eye

Brownian motion

- Watch a particle in water under microscope: follows
- Follows random path: **Brownian motion**.
- Has turned out to be fundamental dynamical process in many / all domains

- Biology – protein inside cell / monkey in forest
- Chemistry – reactant in gas phase
- Economics – stock price
- Engineering – jet noise
- Environmental sciences – pollutant spreading out
- Music: John Cage composition
- Physics – particle moving in fluid
- Mathematics – fundamental random process

Model Brownian motion: Random walk

- Expensive to simulate collisions of many particles
- Instead, **directly simulate** random kicks using random numbers.
- Start with *simplest model*
- What is simplest model?

Simple random walk

- Simplify: 1 spatial dimension
- Simplify: Particle jumps in discrete time steps
- Simplify (?): particle lives on integers $\mathbb{Z}$

Simple symmetric random walk

- 1 particle moving on integers in 1D
- Jumps:
  - left (-1) with probability $1/$
  - or right (displacement $+1$) with probability $1/2$
- How generate jumps ± 1 with uniform probability?

- One solution: `julia jump() = rand( (-1, +1) )`
- Another: random Boolean value (`true` or `false`) and convert to step:

```julia
r = rand(Bool)
Int(r)   # convert to integer
```

- How convert this to ± 1? Which is faster?
- Another: `rand() < 0.5`

Random walk process

- Now we know how to do a single jump, we put many of them together to create a random walk
- Now know: don't use global scope; *make a function*:

```
function walk(N)
    x = 0      # initial position
    positions = [x]  # store the positions

    for i in 1:N
        x += jump()
        push!(positions, x)
    end

    return positions
end
```

Interactive animation of walker position

- Now first instinct: Plot data and make interactive
- *Pre-generate* data so don't have different randomness each time:

```
using Interact

N = 100
positions = walk(N)

@manipulate for n in 1:N
    plot(positions[1:n], xlim=(0, N), ylim=(-20, 20), m=:o, ms=1)
end
```

Shape of random walk

- Plot several walks in single figure using `for`
- Since `for` returns `nothing`, evaluate graph to plot:

```
p = plot(leg=false)  # empty plot
N = 100

for i in 1:10    # number of walks
    plot!(walk(N))
end

p  # or plot!()
```

- **Exercise**: Animate position of several walkers simultaneously

Distribution of walker position

- Fix a time n, e.g. $n = 10$ and think about X_n
- Ask same questions as always:
  - What is mean position $\langle X_n \rangle$?
  - What is variance of X_n?
  - What does probability distribution of X_n look like?

Random processes

- Notation:
  - Steps $S_i = \pm 1$
  - Position X_n at step n
- $X_n = S_1 + S_2 + \cdots + S_n = \sum_{i=1}^{n} S_i$
- S_i are random variables; X_n is also random variable.
- Collection $(X_n)_{n=1}^{N}$ is **random process**: random variable at each time

Dynamics of random process

- Whole process is similar to (stochastic) dynamical system
- Questions:
  - What is dynamics *as a function of time*?
  - How does mean position change *as function of time*?
  - How does variance change *as function of time*?
  - Number of sites visited up to time n
  - First time to reach certain position
- Last two questions cannot be answered by looking at single time n

Probability distribution of X_n

- X_n is **discrete random variable**
- Run "cloud" ("ensemble") of **independent** walkers, i.e. *don't interact with one another*
- To generate data, could use `walk(N)`, but only need final position:

```
jump() = rand( (-1, +1) )
```

```
walk_position(N) = sum(jump() for i in 1:N)
```

- Faster to generate all random numbers at once: `julia walk_position2(N) = sum(rand((-1, +1), N))`

Many walkers

- Now can collect data on many walkers:

```
using StatsBase

T = 10
N = 100

data = [walk_position(T) for i in 1:N]

counts = countmap(data)
ks = sort(collect(keys(counts)))

bar(ks, [counts[k] for k in ks])
```

Time evolution of statistics

- How calculate time evolution of mean & variance as function of time n?
- Need access to all walker positions at all times
- Store whole history of each walker – $$$ in memory
- Or evolve walkers for m steps, calculate, then evolve futher.

Simulate many walkers

- How make several walkers?

```julia
num_walkers = 100
walkers = zeros(Int, num_walkers)
```

- Need function that *modifies* its argument
- Julia convention: `!` at end of function name:

```julia
function move!(walkers, i)
    walkers[i] += jump()
end
```

- Now move *all* walkers
- Use another method with *same name* since common functionality

```
function move!(walkers)
    for i in 1:length(walkers)
        move!(walkers, i)
    end
end
```

- Make *interactive visualization*: pre-generate data

Different types of walkers

- So far restricted to walker on integers
- Generalize
- E.g. steps uniformly distributed on interval $[-0.5, 0.5]$
- How generate?
- `rand()`: uniform random number in interval $[0, 1)$

- Make function:

```
continuous_jump() = rand() - 0.5
```

- Different “type of jump”
- Make new **abstraction**: random walker defined by given `jump` function
- Makes previous code more **generic**

Make code more generic – abstraction

- *Pass in jump function as argument* to previous function – *code is the same as before*!

```
function walk(jump, N)
    x = 0
    positions = [x]

    for i in 1:N
        x += jump()   # now calls custom jump function
        push!(positions, x)
    end

    return positions
end
```

Difficulties

- Walkers have position with different *types* and different *jump functions*
- `x = 0` defines `x` as *integer*
- In problem set 3 will have an internal state too
- Need a better solution

User-defined types

- Collect information for each walker in **new type**
- `DiscreteWalker` and `ContinuousWalker` are kinds of a supertype `Walker`:

```
abstract type Walker end

mutable struct DiscreteWalker <: Walker   # subtype
    x::Int
end

mutable struct ContinuousWalker <: Walker
    x::Float64
end

position(w::Walker) = w.x
```

Jump functions

- Rewrite `jump` functions:

```
jump(w::DiscreteWalker) = rand( (-1, +1) )
jump(w::ContinuousWalker) = rand() - 0.5
```

- Define `initialize!` function:

Walk function

- Rewrite `walk` function:

```
function walk!(w::Walker, N)    # modifies its argument
    positions = [position(w)]

    for i in 1:N
        x = position(w)
        new_x = x + jump(w)

        set_position!(w, new_x)    # now calls custom jump function
        push!(positions, new_x)
    end

    return positions
end
```

- Make walkers:

```
d = DiscreteWalker(0)
c = ContinuousWalker(0.0)

pos1 = walk(d, 10)
pos2 = walk(c, 10)
```

- Julia generates **specialized code** for each version

Moving to 2D

- How can we move to a model in 2D with coordinates x and y?
- Above code will not work
- Could use vector for coordinates, or make `jump!` function instead

Review

- Random walks model random motion in space: **diffusion**
- Spread out slowly
- Define Julia types to represent different kinds of objects