4. Probability II & random walks

Last time

- Fixed points of dynamical systems
- Probability as long-term frequencies
- Concepts: Random variable and probability distribution
- Bar graph for categorical data

Goals for today

- Characterising variability
- Probability distribution + summary statistics
- Modeling random motion: random walks

Recall: Rolling a die in Julia

- `rand(1:6)` gives random (uniform) integer between 1 and 6
- `rand(1:6, N)` gives N of them
- **Count** outcomes

```julia
function count_outcomes(data, sides)

    counts = zeros(Int, sides)

    for result in data
        counts[result] += 1
    end

    return counts
end
```

Plotting bar graphs interactively

- Suppose we roll one die at a time and want to update the statistics and redraw
- We can do this using `Interact`, but…
- We don't want to regenerate new data each time, but rather use the same data
- So pre-generate data *before* plotting
- Plot only relevant *portion* of data

Frequencies

- Instead of counts, plot **proportion** or **frequency**
- Compare to expected result:

```
bar(counts ./ N, leg=false)
hline!( [1/6], ls=:dash, lw=2)    # horizontal line
```

- Here the `.` means **broadcast**: apply the operation element by element

Probability distribution

- Heights of bars are **probability** that each value occured in the **sample**.
- Collection of all probabilities (proportions / frequencies) is called the **probability distribution**.
- Gives Prob$(X = i)$ for each possible outcome i in *discrete* set

Variability

- We have **finite sample** from ideal **population**
- If we repeat experiment, get different sample with different counts.
- How characterize this *variability*?

Characterising variability

- Focus on n_1 := total number of 1s out of N
- Also a random variable; ask same questions:
  - what are possible outcomes?
  - how often does each outcome occur?
- I.e. want **probability distribution** of n_1
- Should be "close to" $N/6$.
- **Variability**: how *far away* from $N/6$ can n_1 be?
- How count number of rolls that give 1?

Probability distribution of n_1

- Use `for` loop – **exercise**
- Julia has tools to simplify (but not quicker? – **benchmark!**):

```
n1(N) = count( rand(1:6) == 1 for i in 1:N )    # or count
```

- *Generator expression* $\equiv$ array comprehension *without creating array*
- `count` counts the number of `true`s in the expression

Support of a distribution

- What is **support** of n_1: set of possible outcomes?
- Minimum possible value is 0, maximum is N; all values in between are possible.
- Intuitively those extreme values are *very* unlikely (improbable = low probability).
- **Exercise**: How improbable?
- Can calculate mathematically and/or do *computer experiment*.

n_1 experiment

- Run experiment for n_1:

```
N = 1000  # number of die rolls
num_experiments = 10000

# repeat experiment:
n1_data = [n1(N) for i in 1:num_experiments]

counts = count_outcomes(n1_data, N)
# better to use a Dict?

bar(counts)
```

Shape of distribution

- See that n_1 “clusters around” a middle value: **expected value**.
- Values near extremes “never” occur.
- Characterise using **summary statistics**: numbers that summarise aspects of **distribution**.
- Simplest: **sample mean** = average value

Mean

- Given outcomes x_i for $i = 1, \ldots, N$, (arithmetic) mean is

$$\bar{x} := \frac{1}{N} \sum_{i=1}^{N} x_i$$

- Calculate in Julia:

```
mean(data) = sum(data) / length(data)
```

```
m = sum(n1_data) / length(n1_data)
```

- NB: `mean` already defined in `StatsBase` standard library package (no need to install)

- Add to plot:

Centre the data

- Distribution “spreads out” from mean
- How can we measure *how far* it spreads
- **Centre** data by subtracting mean:

```
n1_centered = n1_data .- m

bar(count_outcomes(n1_centered, N))
vline!([0], c=:red, lw=2, ls=:dash)
```

Spread

- Want to measure spread as some kind of "average spread away from mean" = "average *distance* from mean"
- If just take `mean` of new data, get tiny result near 0:

```
mean(n1_centered)
```

- (`1e-14` means 1×10^{-14}, i.e. a value that is effectively 0.)
- Why? Negative values *cancel out* positive values.
- Need to *avoid cancellation*. How?

Spread II

- Options to avoid cancellation of displacements from mean:
  - take **absolute value**
  - take **square**:

```
spread = mean(abs.(n1_centered))    # no standard name?

variance = mean(n1_centered .^ 2)
σ = √variance

@show spread, variance
```

- (Sample) **variance** defined by squaring, so must take $\sqrt{}$ for “correct units” (metres vs. metres^2)
- σ is called **standard deviation**

Spread III

- Let's plot these:

```
bar(countmap(n1_centered))
vline!([-σ, σ], c=:red, lw=2, ls=:dash)
vline!([-2σ, 2σ], c=:green, lw=2, ls=:dash)
```

- Most data is in interval $[\mu - 2\sigma, \mu + 2\sigma]$.
- How much? Calculate!:

```
count(-2σ .< n1_centered .< 2σ) / length(n1_centered)
```

- Approx. 95%: "universal" in many (but *not all* situations) – see later

Effect of increasing data size

- What happens if take more data, i.e. larger N?
- Still expect $n_1(N) \simeq N/6$
- What happens to spread?

- Expect to spread out more – *how much*?
- Can calculate analytically using probability theory
- Or *do computational experiment*
- How does spread depend on N? *Problem set 2*

Random walks

Brownian motion

- Watch a particle in water under microscope.
- Follows a random path: **Brownian motion**.
- Fundamental dynamical process in many domains:
  - Biology – protein inside cell
  - Chemistry – reactant in
  - Physics – particle in fluid
  - Engineering – jet noise
  - Economics – stock price
  - Environmental sciences – pollutant spreading out
  - Mathematics – fundamental random process

Modelling random motion: Random walk

- Expensive to simulate collisions of many particles
- Instead, **directly simulate** random kicks using random numbers.
- Simplest model: **simple random walk**:
  - 1 particle moving on integers in 1D
  - Jumps left (displacement -1) or right (displacement $+1$) with probability $1/2$
- How can we generate jumps ± 1 with uniform probability?

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

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

- How convert this to ± 1? Which is faster?

Simple random walk

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

Interactive animation of walker position

- First instinct by now: Plot data and make it 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,
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
- Can ask same questions as before:
  - What is mean position $\langle X_n \rangle$?
  - What is the 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 a **random process** – i.e. a 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))`
- Probability distribution: Problem set 2

Review

- Characterise variability using **mean** and **variance** or **standard deviation**
- Most data within 2 standard deviations of mean in common distributions
- Random walk is simple model of random motion