Week 6 — Markov Chains + Networks

Friday, June 5, 2026

Prof. Joseph Austerweil

Agenda

Welcome + a Markov chain you already know	0:00
Markov chains: states, transitions, the Markov property	0:10
Stationary distribution	0:25
Networks: graphs as structure	0:40
Random walk on a network	0:52
Break	1:07
Memory search as a random walk (Abbott 2012)	1:12
Network properties + PageRank + Week 7 bridge	1:48

A Markov chain you already know

Admin — Assignment 2

Assignment 2 (Generalization) is due Fri Jun 19, 8:00 PM — worth 7.5%.

- Build your own Bayesian generalization model: design a hypothesis space, then compute posteriors and predictions under weak vs. strong sampling.
- **Walkthrough + stencils:** hml.chibatech.dev/assignments.html — read the PDF first, then pick a stencil (GenJAX / Python / R), one-click “Open in Colab.”
- Reminder: **3 free late days pooled** across the four programming assignments.
- Questions? **DM me, or drop them in the class channel.**

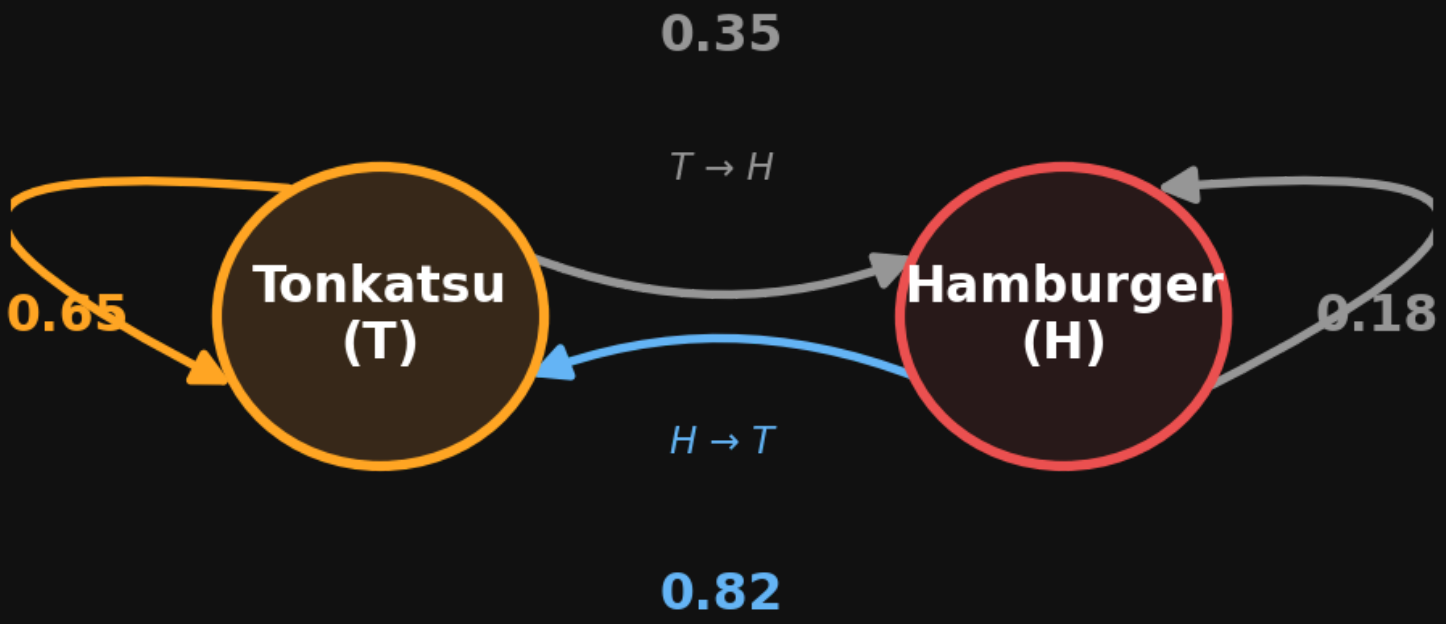
Chibany's bento habit

New twist on Chibany this week: students now bring Chibany **both** a **tonkatsu** and a **hamburger** bento every day — and Chibany **chooses** which one to eat.

There's a habit in the choosing: Chibany **loves tonkatsu**, so after a tonkatsu day they usually want it *again* — but every so often they fancy a change. And after a **hamburger** day, they almost always swing back to tonkatsu.

If Chibany chose tonkatsu **yesterday**, what will they *probably* eat **today**? And does it matter what they ate **last Tuesday**?

Draw the habit



Two **states** (X_t = today's bento), arrows = probabilities of tomorrow's choice. Today depends **only** on yesterday. Chibany **loves tonkatsu**: sticks with it (0.65) and swings back to it after a burger (0.82).

Markov

Really past

Past

Less past

Given the present, the future is independent of the past.

Name it: the Markov property

The future is independent of the past, given the present.

$$P(X_{t+1} \mid X_t, X_{t-1}, \ldots, X_0) \;=\; P(X_{t+1} \mid X_t)$$

- Once you know **today** (X_t), the whole earlier history tells you nothing more about **tomorrow**.
- A sequence with this property is a **Markov chain**.

Where we are

Welcome + a Markov chain you already know	0:00
Markov chains: states, transitions, the Markov property	0:10
Stationary distribution	0:25
Networks: graphs as structure	0:40
Random walk on a network	0:52
Break	1:07
Memory search as a random walk (Abbott 2012)	1:12
Network properties + PageRank + Week 7 bridge	1:48

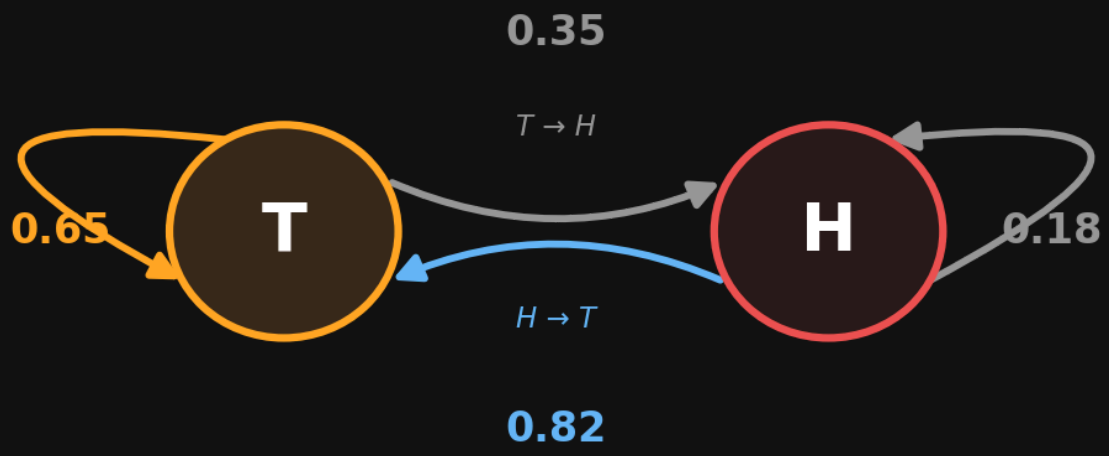
Markov chains: the machinery

Notation lock-in

- X_t — the **state** at time t (one of a finite set $\mathcal{S}$). *Chibany:* $X_t \in \{\mathrm{T}, \mathrm{H}\}$.
- P — the **transition matrix**: $P_{ij} = P(X_{t+1} = j \mid X_t = i)$.
- π — the **stationary distribution** (the long-run fractions — defined fully next segment).

Each **row** of P is a probability distribution over next states → it sums to 1. Such a matrix is called **row-stochastic**.

Two views of the same chain



Matrix view:

$$P = \begin{pmatrix} 0.65 & 0.35 \\ 0.82 & 0.18 \end{pmatrix}$$

Row 1 = “if **T** today, then tomorrow.”

Row 2 = “if **H** today.” *(columns are T, H)*

Same chain, two pictures.

Picture view: Chibany’s chain — states + labelled arrows (the same one from “Draw the habit”).

The matrix is a sampler

How does the chain actually *take a step*? You roll a (continuous) die.

Say today is **tonkatsu (T)**, so the row is $[\,0.65,\ 0.35\,]$ — 0.65 stay T, 0.35 switch to H.

1. Draw a random number u evenly between 0 and 1. *(Say $u = 0.42$.)*
2. $0.42 < 0.65 \rightarrow$ **stay at T**. *(If it were $> 0.65 \rightarrow$ switch to H.)*

Repeat tomorrow, and the next day… **the matrix plus a stream of random numbers generates the whole sequence.** Hold this idea — it returns at the end of class.

Run it: what a chain produces

Five runs of **Chibany's chain**, each starting at **T**:

```
T H T H T T H T H T T T H T H T T T T H
T T H T H H T T T T T H T H T T T T T T
T T H T H T T H T T T H T T T H T H T T
T T T T H T T H T T T T T T T H T T T
T T H T H T T H T T T H T T T T H T T T
```

Mostly **T**, with brief **H** interruptions (hamburger almost never repeats).

Over time, roughly **70% T, 30% H**.

Where we are

Welcome + a Markov chain you already know	0:00
Markov chains: states, transitions, the Markov property	0:10
Stationary distribution	0:25
Networks: graphs as structure	0:40
Random walk on a network	0:52
Break	1:07
Memory search as a random walk (Abbott 2012)	1:12
Network properties + PageRank + Week 7 bridge	1:48

The stationary distribution

Shuffling a deck

- **State** = the deck's current ordering
- **Transition** = "move a random card to the top," over and over

You shuffle a deck before a game. What are you actually trying to achieve — what's the *computational goal* of shuffling?

The goal: a distribution that stops changing

The goal of shuffling: make **every ordering equally likely** — a *uniform* distribution over all $52!$ orderings.

- Shuffle once more from there, and the distribution is **still uniform** — shuffling can't make it "more shuffled."
- That self-perpetuating, **no-longer-changing** distribution is what this whole lecture segment is about.

We were never after one ordering — we were after a **distribution the process settles into**. Let's make that precise.

The long-run question

Run a chain for a very long time. **What fraction of the time do you spend in each state?**

Over a whole semester, what fraction of Chibany's lunches are tonkatsu?

Call that distribution π.

Definition: π is *stationary* if running one more step doesn't change it:

$$\pi P = \pi$$

If you're already distributed as π, you stay distributed as π forever.

How to find π — just run it

Start from **any** distribution and step repeatedly — i.e. multiply by P over and over:

$$\mathbf{v},\ \mathbf{v}P,\ \mathbf{v}P^2,\ \mathbf{v}P^3,\ \ldots\ \longrightarrow\ \pi$$

This is called **power iteration**. Let’s watch it on **Chibany’s chain**.

Poll — does the start matter?

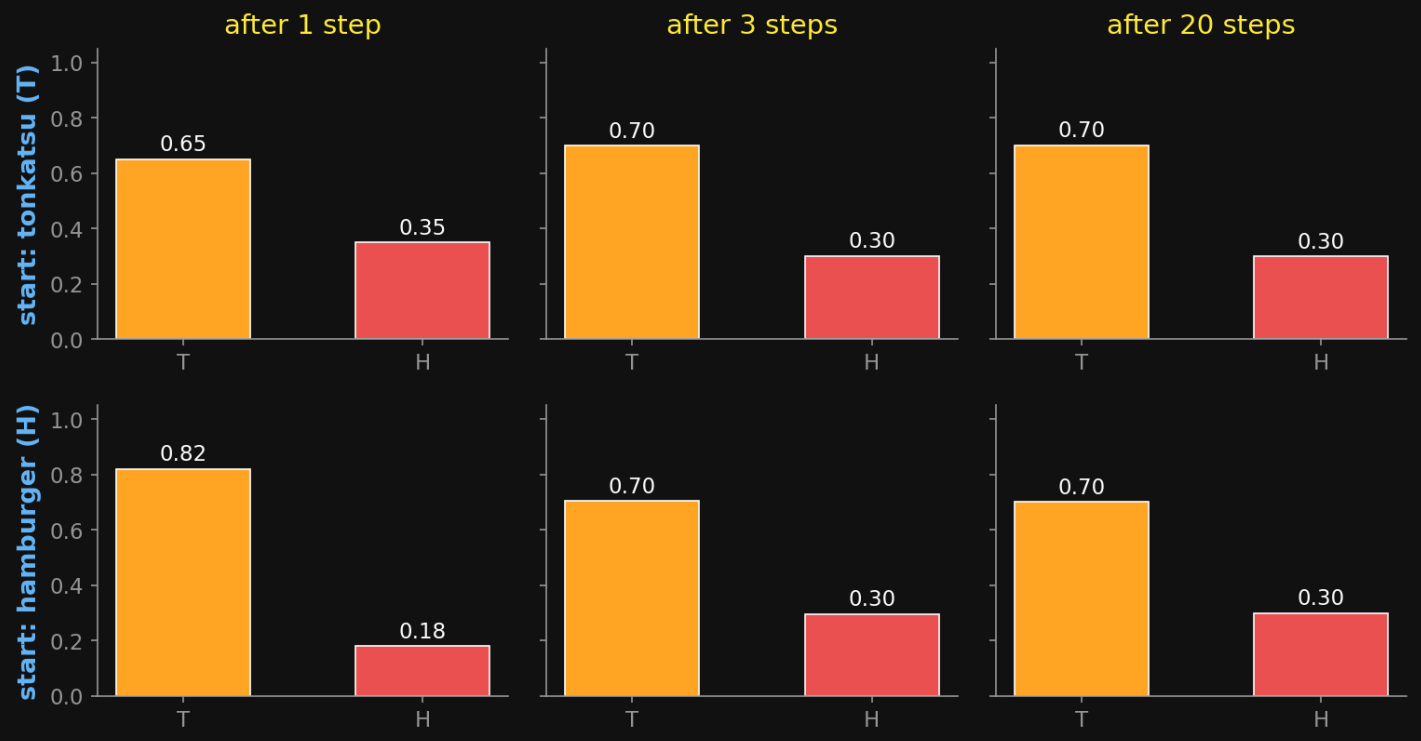
We'll run Chibany's chain two ways — he **always starts on tonkatsu**, or **always on hamburger** — for **20 days**.

Before we look: is the chance he's eating **tonkatsu** on day 20 the same either way?

- **A.** essentially the same
- **B.** clearly different — the start sticks
- **C.** it depends on the exact matrix

Two different starts — the reveal

Same start always-tonkatsu (top) vs always-hamburger (bottom). After k steps:



By 20 steps both starts give the SAME 70/30 split — the chain forgets where it began.

A. Essentially the same. After 1 step they differ; by 20 steps both reach **70% T / 30% H**. The chain **forgot where it started**.

Why? The chain mixes

After enough days the chain **forgets where it started** — it *mixes*. The long-run distribution is the stationary π (Chibany's **70/30**), and (for a well-behaved chain) it's the **same from every starting point**.

The technical condition is **ergodicity** — you can get from any state to any other, and the chain doesn't get trapped in a cycle.

Where does π come from?

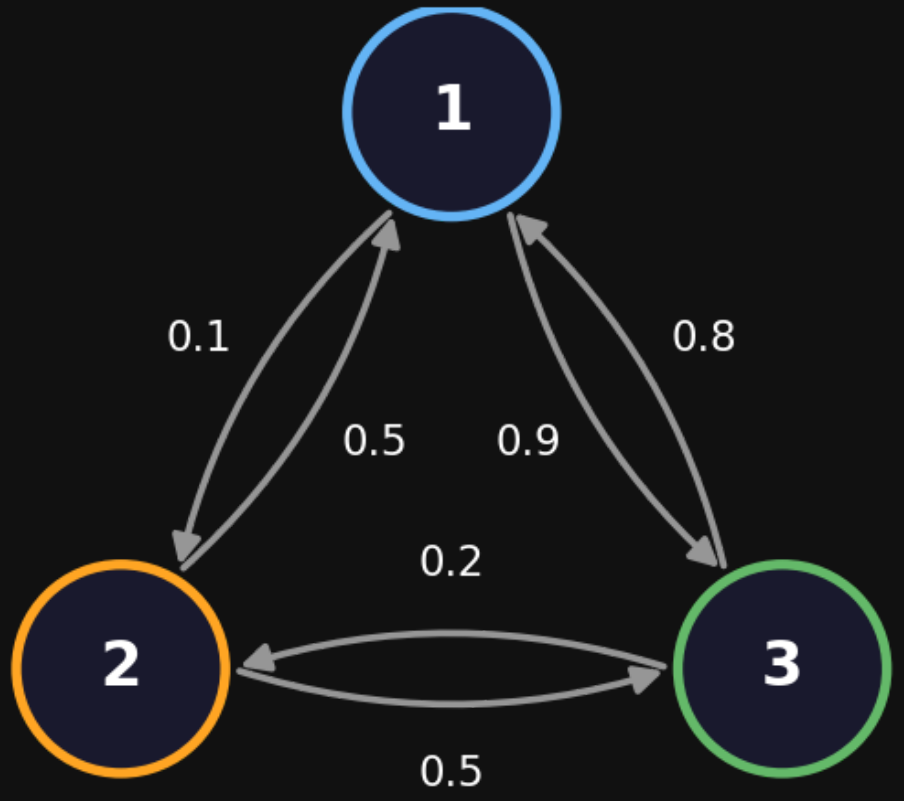
Power iteration **converges** to π. But what *is* π, exactly?

π is the **left eigenvector of P with eigenvalue 1**.

- An **eigenvector** of a matrix is a vector whose *direction* is unchanged when you multiply by the matrix — only its length scales, by a factor called the **eigenvalue**.
- Since $\pi P = \pi$ (unchanged, scale $= 1$), π is exactly the eigenvector with eigenvalue 1. Every row-stochastic matrix has one.

The bar charts showed this: once the shape stopped changing under $\times P$, you'd found π.

Now try a harder one — together

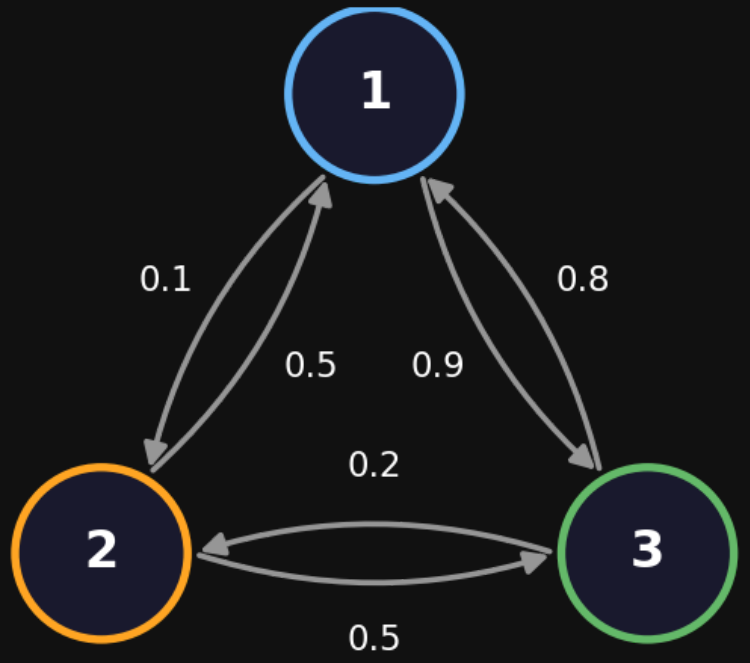


A **3-state** random walk (states 1, 2, 3). Same rules as Chibany — just more states. Its transition matrix:

$$A = \begin{pmatrix} 0 & 0.1 & 0.9 \\ 0.5 & 0 & 0.5 \\ 0.8 & 0.2 & 0 \end{pmatrix}$$

Row i = where you go from state i. (No self-loops here — the diagonal is 0.)

Poll — guess before we compute



Run this 3-state walk a long time. **Which state does it visit MOST often?**

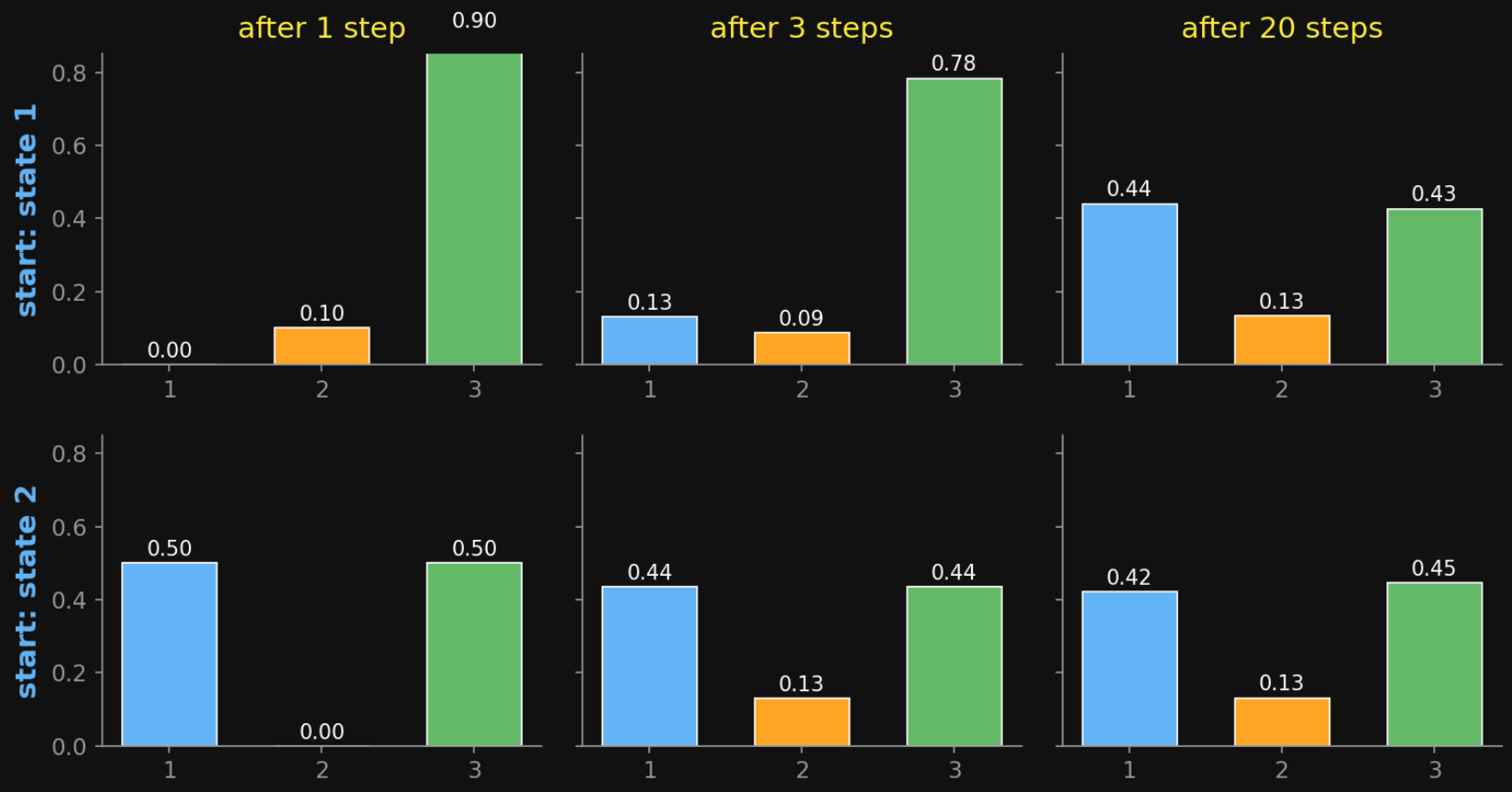
Which state do the *others* tend to send you toward?

- **A.** state 1
- **B.** state 2
- **C.** state 3
- **D.** all about equal

$$A = \begin{pmatrix} 0 & 0.1 & 0.9 \\ 0.5 & 0 & 0.5 \\ 0.8 & 0.2 & 0 \end{pmatrix}$$

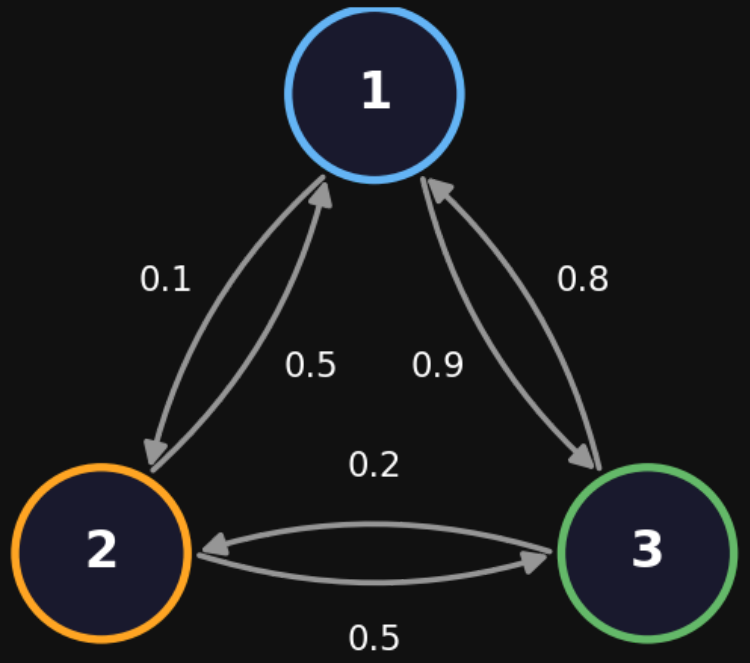
Work it out — power iteration

Same method as Chibany: pick two starts (**state 1** / **state 2**), multiply by A over and over.



Same story, bigger chain: by 20 steps both starts agree — π ≈ (0.42, 0.13, 0.45).

Poll — does the start matter? (again)



Run this walk for **20 steps**. Is the chance of being in **state 2** the same from **state 1** as from **state 2**?

Same question we asked for Chibany — does it still hold with 3 states?

- **A.** essentially the same
- **B.** clearly different
- **C.** depends on the matrix

$$A = \begin{pmatrix} 0 & 0.1 & 0.9 \\ 0.5 & 0 & 0.5 \\ 0.8 & 0.2 & 0 \end{pmatrix}$$

The 3-state stationary distribution

A. Essentially the same — and the walk's long-run home is:

$$\pi \approx (\,0.42,\ 0.13,\ 0.45\,)$$

- **States 1 and 3** soak up most of the visits (everyone points toward them).
- **State 2** is rarely visited — exactly the poll intuition.
- Two states, three states, or 52! card orderings — **same method, same kind of answer.**

Where we are

Welcome + a Markov chain you already know	0:00
Markov chains: states, transitions, the Markov property	0:10
Stationary distribution	0:25
Networks: graphs as structure	0:40
Random walk on a network	0:52
Break	1:07
Memory search as a random walk (Abbott 2012)	1:12
Network properties + PageRank + Week 7 bridge	1:48

Networks: graphs as structure

What’s a graph?

A **graph** $G = (V, E)$ is just **nodes** (V) joined by **edges** (E).

- Edges can be **undirected** (mutual) or **directed** (one-way)
- Edges can be **weighted** (stronger / weaker)

You already met graphs **last week** — a **Bayes net** is a directed graph. There an edge meant *“depends on”*; the graphs today use edges to mean *“is related / connected to.”*

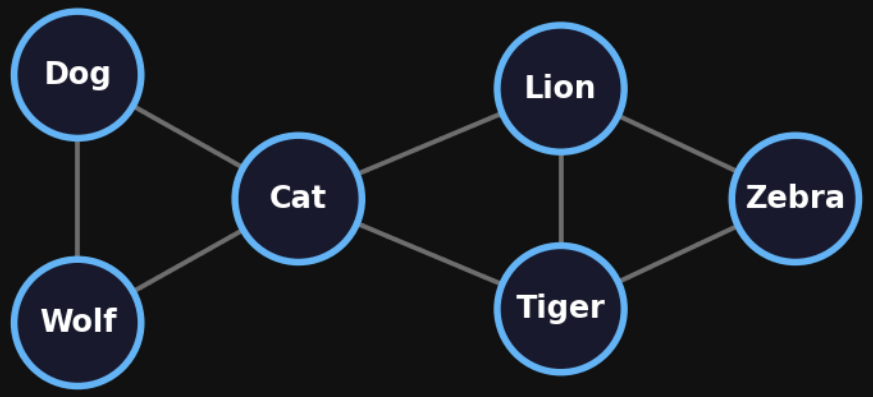
Graphs are everywhere

The power of the abstraction: **one object, many domains.** What are the **nodes** and **edges**?

Graph	Nodes	Edges
Semantic network (cog sci)	concepts / words	“is related / associated”
The Web	web pages	hyperlinks (directed)
Co-authorship network	researchers	“wrote a paper together”
Social network	people	friendships / follows
Road map	intersections	roads (weighted = distance)
Neural network (brain)	neurons	synapses

Today’s star is the **semantic network** — concepts as nodes, associations as edges. Cognitive science has used it for decades (Collins & Quillian 1969; Collins & Loftus 1975).

The graph as a matrix

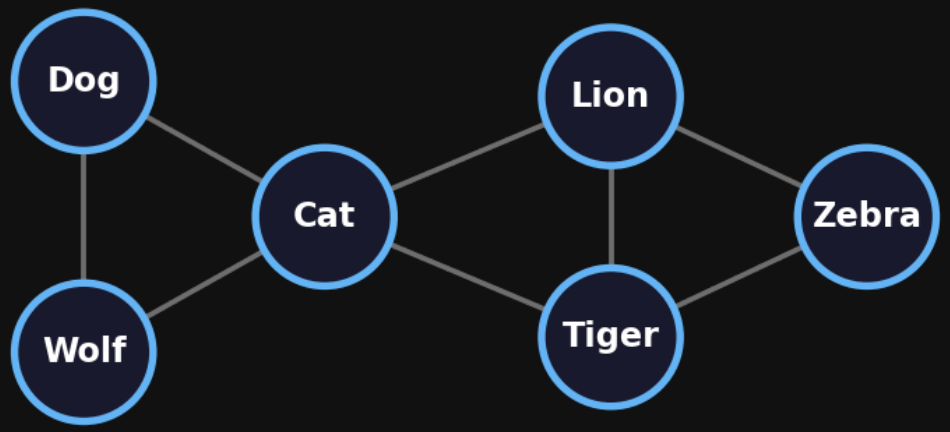


Adjacency matrix L: $L_{ij} = 1$ when nodes i and j share an edge, else 0.

$$L = \begin{array}{r|cccccc} & \mathrm{D} & \mathrm{W} & \mathrm{C} & \mathrm{L} & \mathrm{T} & \mathrm{Z} \\ \hline \mathrm{Dog} & 0 & 1 & 1 & 0 & 0 & 0 \\ \mathrm{Wolf} & 1 & 0 & 1 & 0 & 0 & 0 \\ \mathrm{Cat} & 1 & 1 & 0 & 1 & 1 & 0 \\ \mathrm{Lion} & 0 & 0 & 1 & 0 & 1 & 1 \\ \mathrm{Tiger} & 0 & 0 & 1 & 1 & 0 & 1 \\ \mathrm{Zebra} & 0 & 0 & 0 & 1 & 1 & 0 \end{array}$$

Symmetric (undirected). Each **row sum** = that node's **degree** — Cat's row sums to 4.

From a graph to a transition matrix



The hinge of the whole lecture:

1. List the edges in an **adjacency matrix** L: $L_{ij} = 1$ if i–j are connected (or a positive weight).
2. **Normalize each row** to sum to 1 → a transition matrix P.

Now P describes a walker who, at each node, **steps to a random neighbour**.

Structure (the graph) + process (the walk) — a Markov chain whose states are nodes.

Where we are

Welcome + a Markov chain you already know	0:00
Markov chains: states, transitions, the Markov property	0:10
Stationary distribution	0:25
Networks: graphs as structure	0:40
Random walk on a network	0:52
Break	1:07
Memory search as a random walk (Abbott 2012)	1:12
Network properties + PageRank + Week 7 bridge	1:48

Random walk on a network

Take a walk — step 1

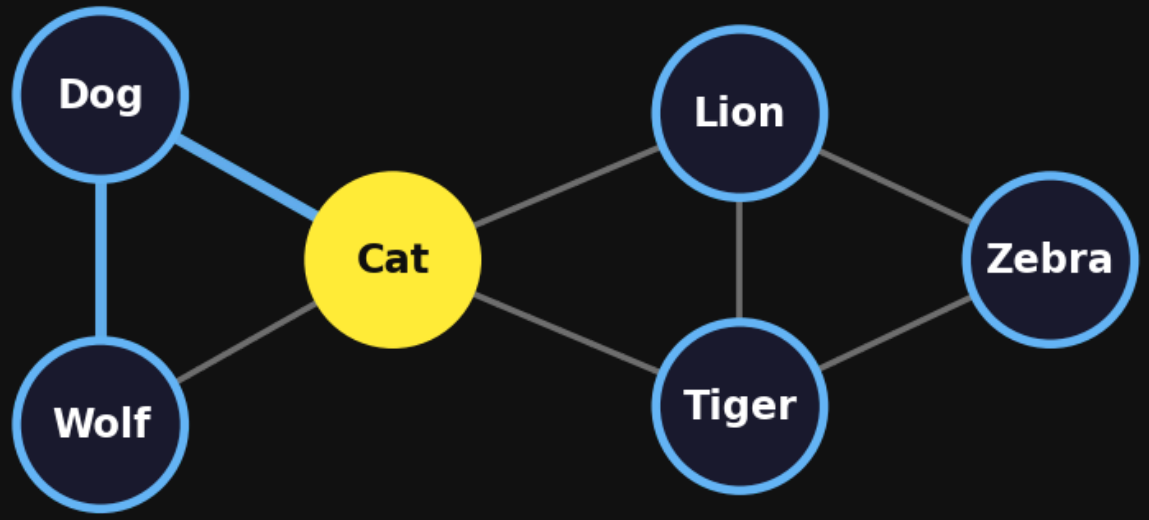
visited: Wolf → Dog

Start at **Wolf**. Pick a random neighbour → **Dog**.

The sequence of nodes we visit *is* a Markov chain.

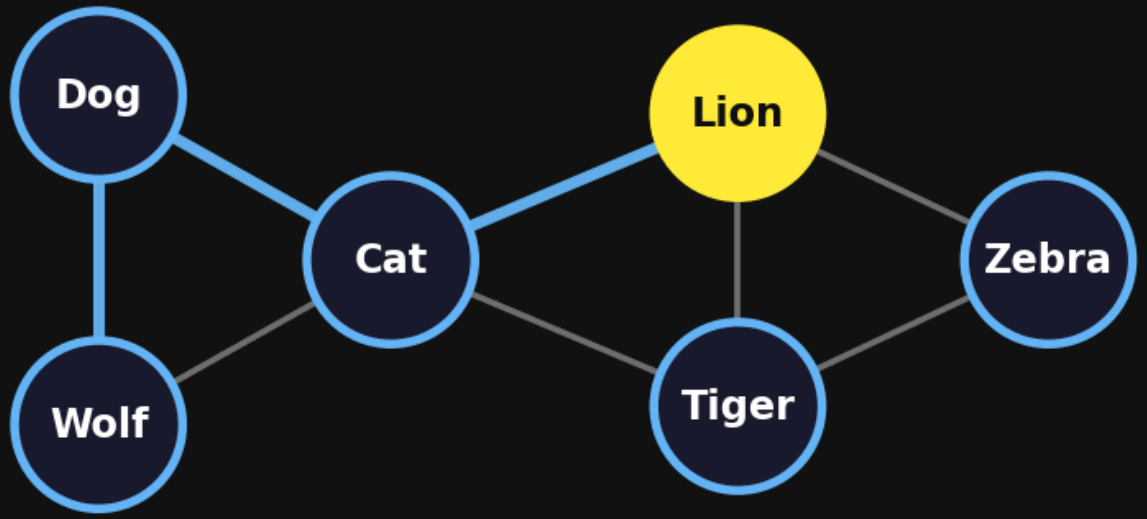
Take a walk — step 2

Dog → random neighbour → **Cat**.



visited: Wolf → Dog → Cat

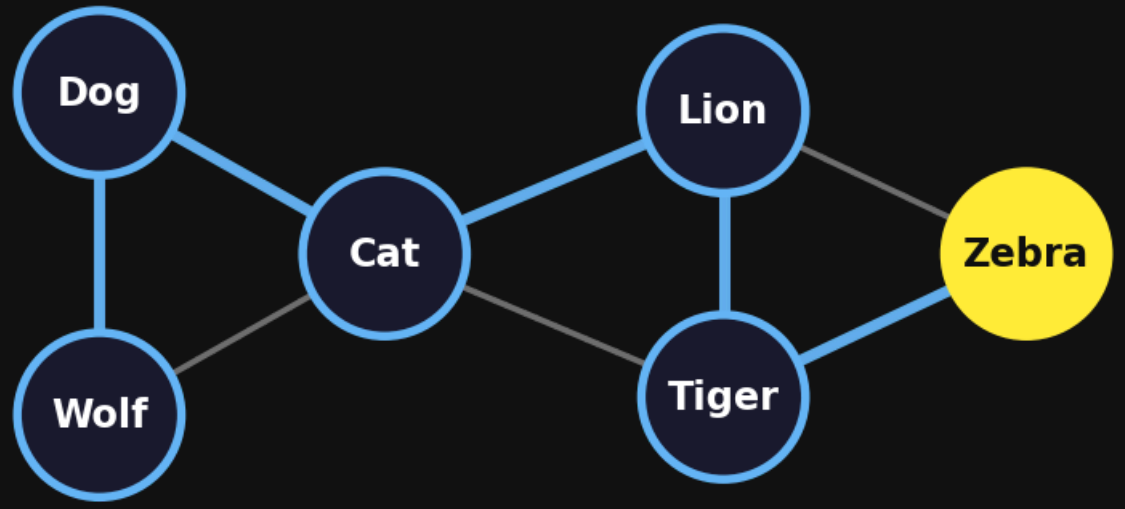
Take a walk — step 3



visited: Wolf → Dog → Cat → Lion

Cat → Lion. The walk crossed the bridge from the “pets” side to the “big cats.”

Take a walk — and keep going



visited: Wolf → Dog → Cat → Lion → Tiger → Zebra

…**Lion** → **Tiger** → **Zebra.** Visited so far: **Wolf** → **Dog** → **Cat** → **Lion** → **Tiger** → **Zebra**.

Run it long enough — which nodes get visited most?

The stationary distribution of a walk

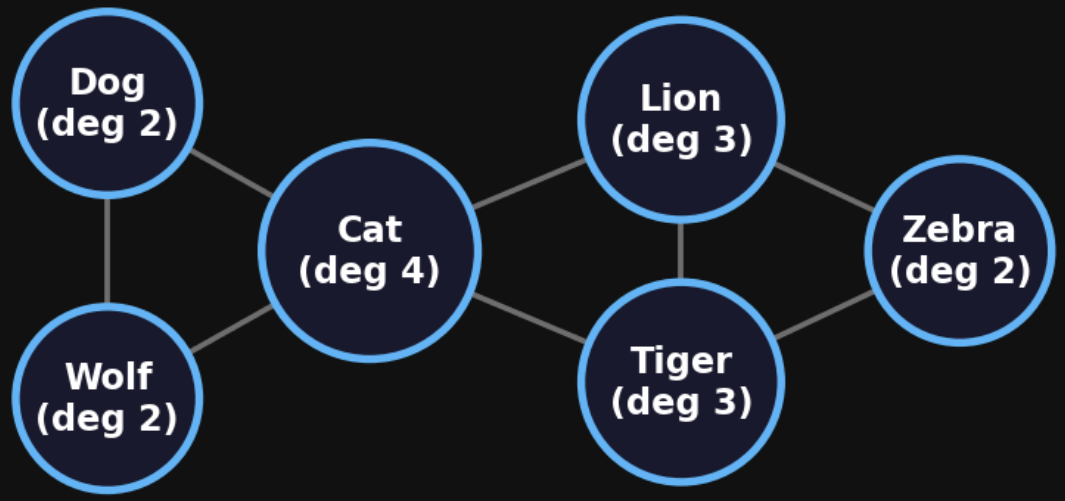
For an **undirected, unweighted** network, the long-run visit frequency has a beautifully simple form:

$$\pi_i \;\propto\; \deg(i)$$

where $\deg(i)$ = the **degree** of node i = how many edges touch it, and $\propto$ means *"grows with"* — double the edges, roughly double the visits.

More connected → visited more often. No eigen-solve needed for this case — the degree *is* the answer.

Poll — which node wins?



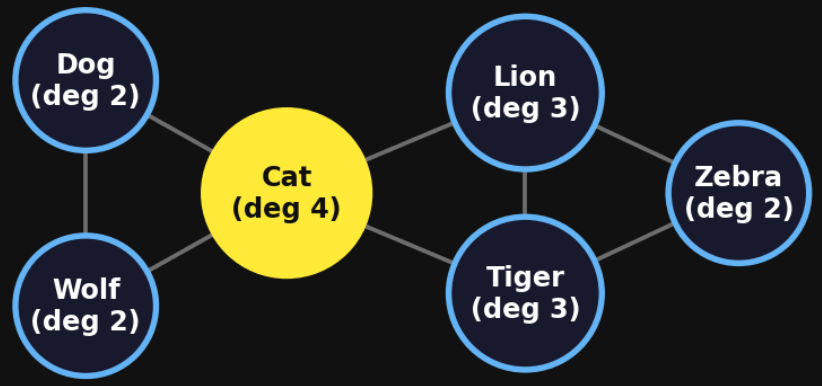
Run a random walk on this network for a long time.

Which node do you visit **most often**?

- **A.** Cat
- **B.** Lion
- **C.** Wolf
- **D.** Zebra

Poll — answer

A. Cat — it has the highest degree.



$\pi_i \propto \deg(i)$, and **Cat** touches the most edges (degree 4) — it's the **bridge** between the two clusters. Long-run behaviour is dictated by **structure**: no extra assumption needed, and **no eigen-solve** for an undirected graph — the degree *is* the answer.

Hold onto this: the most-visited concept is the most-connected one. That's about to explain how people *recall*.

Where we are

Welcome + a Markov chain you already know	0:00
Markov chains: states, transitions, the Markov property	0:10
Stationary distribution	0:25
Networks: graphs as structure	0:40
Random walk on a network	0:52
Break	1:07
Memory search as a random walk (Abbott 2012)	1:12
Network properties + PageRank + Week 7 bridge	1:48


Break
5 minutes
Chiba Tech
School of
Design & Science
EN Press L

Memory search as a random walk

Try it yourself

List as many animals as you can — 60 seconds.

Go.

What your list looks like

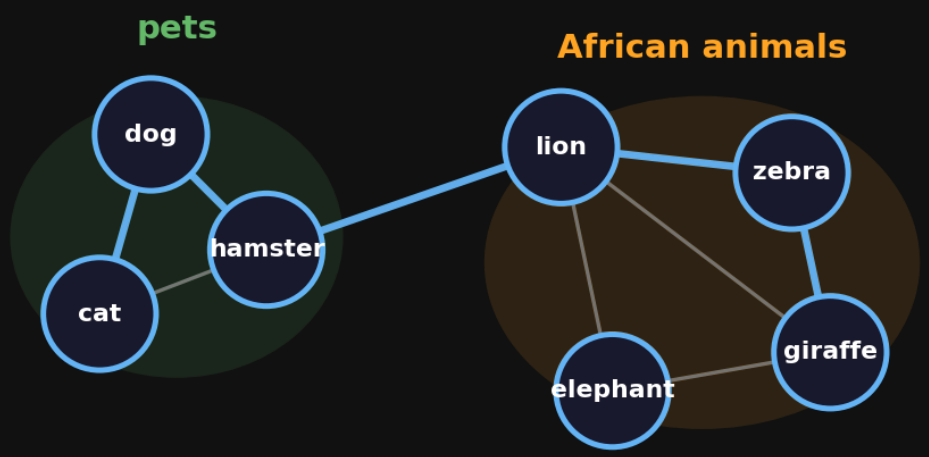
People don’t list animals at random. They come out in **bursts by category**:

> *wolf, lion, giraffe, zebra … dog, cat, hamster …*

A run of African animals, a pause, a run of pets. **Then a switch.**

Why this structure? And why *those* animals, in *that* order?

The model: recall is a random walk



recall: "cat, dog, hamster, lion, zebra, giraffe …"

Abbott, Austerweil & Griffiths (2012):

Your semantic memory is a **network**; recall is a **random walk** on it. The list you produce *is* the sequence of nodes the walk visits.

- **Clusters** = the walk lingering in a densely-connected community
- **Switches** = the walk crossing a bridge edge to another community

No extra “search strategy” needed — it falls out of the network.

The catch: the walk ≠ the list

A random walk **revisits** nodes and wanders through **non-animals** — but a fluency list has neither. So the walk is *not* the list directly.

The censoring function (Abbott et al. 2012) is the missing link: you **report a word only the first time the walk hits it**, and only if it's an animal. Everything else is **censored** (hidden).

- $\tau(k)$ = the **first hitting time** of the k-th unique animal
- The gap between first-hits → the **inter-item response time (IRT)** — *what the data actually measures.*

Censoring — worked example

Reported fluency list: dog, cat

$$\mathrm{IRT}(\mathrm{cat}) = \tau(\mathrm{cat}) - \tau(\mathrm{dog}) + \mathrm{len}(\mathrm{cat}) = 5 - 2 + 3 = 6$$

The walk visits **animal** → **dog** → **house** → **dog** → **cat**. Reported: **dog, cat**. *house* (not an animal) and the *second dog* (a revisit) are censored. The IRT to “cat” is the step-gap between first-hits, plus the word’s length.

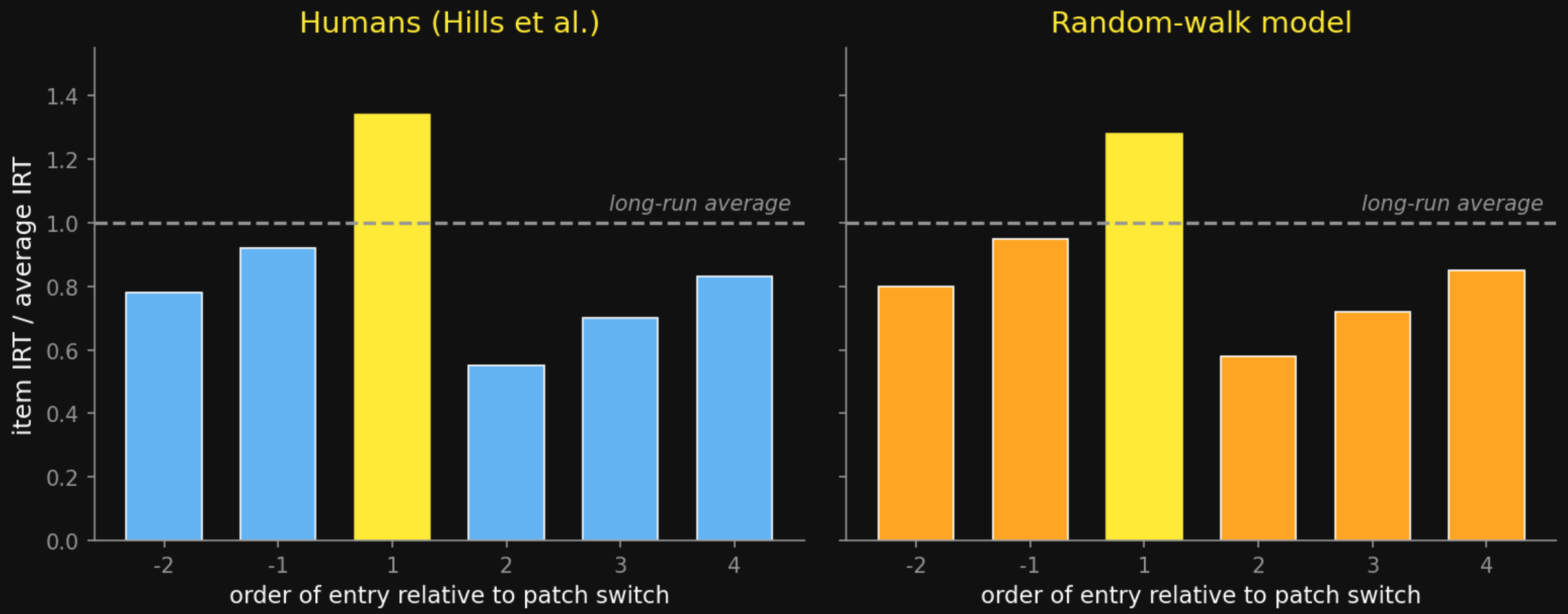
The signature: slow down, then switch

The famous **optimal-foraging** finding (Hills et al. 2012): line up each animal by **when it appears relative to a category switch**.

- Within a patch, items come **faster and faster** — until they slow.
- The **first word of a new patch** is the **slowest** of all — it’s above your average IRT. That’s the “switch cost.”

Marginal Value Theorem: leave a patch when its return rate drops below your overall average. People look like they’re foraging.

…and the random walk reproduces it



Position 1 (first word of a new patch) is SLOWEST — the switch cost. The walk reproduces it with no explicit switch rule.

The censored random walk — **one process, no switch rule** — produces the *same* curve as humans: position 1 slowest, position 2 fastest. The “decision to switch” was never needed; it falls out of the network’s structure.

Why this is a big claim

The same **structure + process** split we've used since Marr (Week 2):

- **Structure** — the semantic network (estimated from *other* behaviour, e.g. word-association norms)
- **Process** — the random walk (a Markov chain — *memoryless*)
- **Behaviour** — the fluency list (which animals, in which order, with which pauses)

Structure + process **jointly predict** behaviour — and they fit human data well, with no recall-specific machinery bolted on.

Discuss

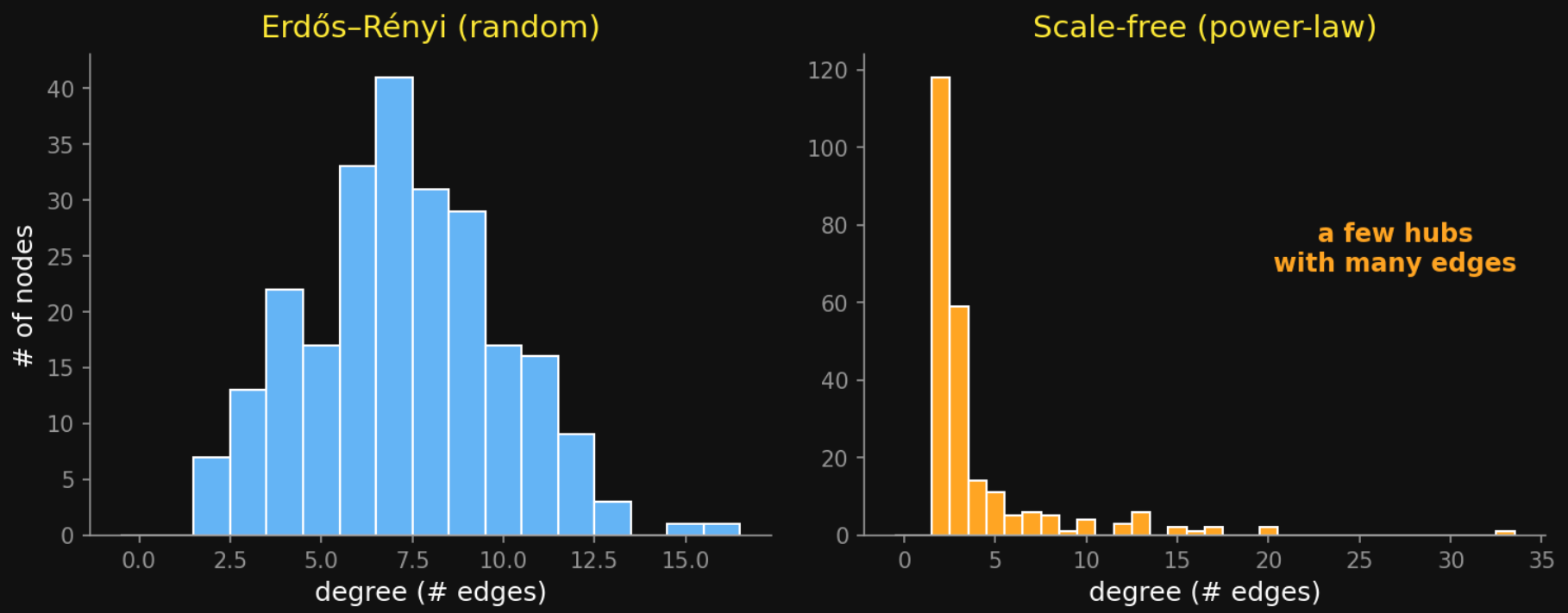
- The walk is **memoryless** — but people rarely repeat themselves. Fatal flaw, or fixable?
- The network is **estimated from other data**. Is that a clean test, or is some explanation smuggled in?
- Does the **stationary distribution** of your memory mean anything real — salience? accessibility? frequency?

Network properties + the road to Week 7

A few network terms

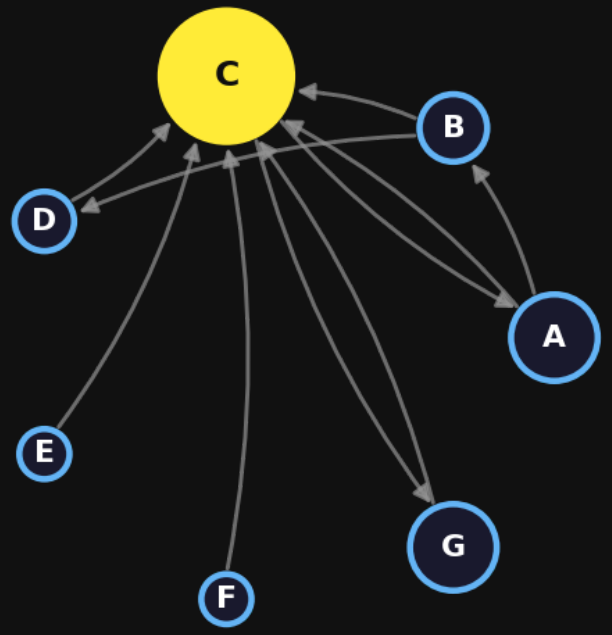
- **Degree** of a node = number of edges touching it. (We used this: $\pi_i \propto \deg(i)$.)
- **Shortest path length** (geodesic distance) = fewest edges between two nodes.
- **Diameter** = the longest shortest-path in the network.
- **Degree distribution** = the histogram of all nodes' degrees — it tells you what *kind* of network you have.

Two kinds of network



- **Random (Erdős–Rényi):** Binomial degrees, everyone similar
- **Scale-free / power-law:** a few high-degree **hubs** — real semantic, social, and web networks look like this

Hubs win: PageRank

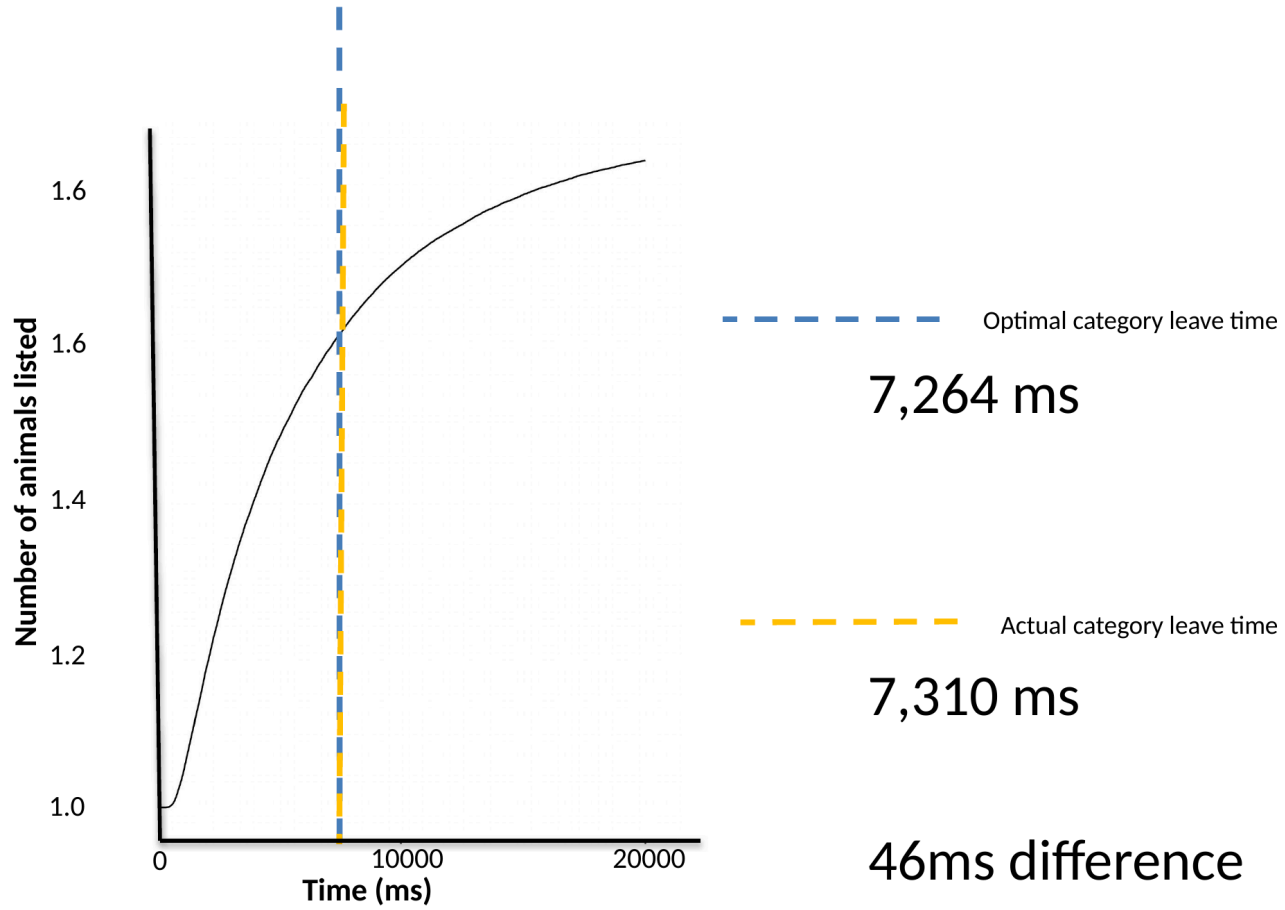


node size ∝ PageRank = long-run visit frequency of a random surfer

PageRank (the original Google algorithm) ranks pages by the **stationary distribution of a random walk** over the link graph — exactly the π from this morning, at web scale.

Griffiths, Steyvers & Firl (2007), *Google and the mind*: PageRank over a **semantic** network predicts human word-fluency. Same algorithm, same π.

Optional: do people *actually* leave at the optimum?



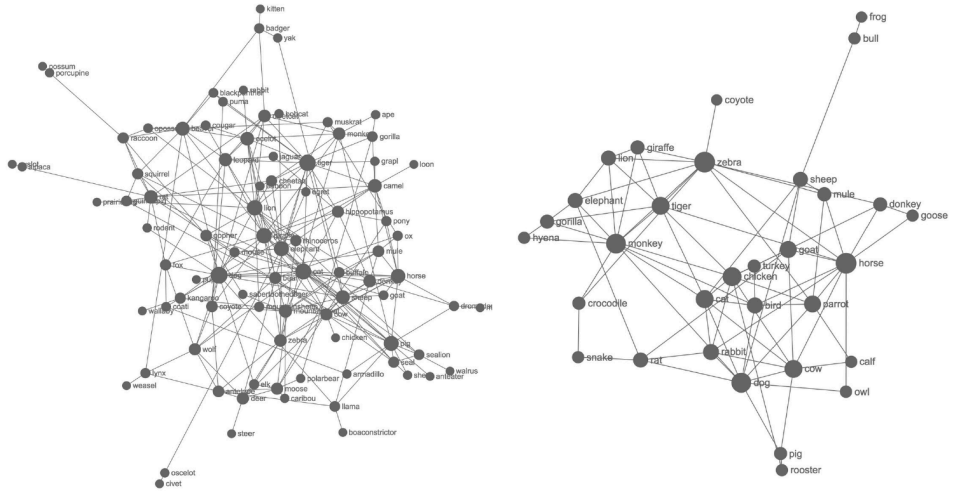
(Zemla, Gooding, & Austerweil, 2023)

Zemla, Gooding & Austerweil (2023): actual category-leave times track the MVT optimum almost exactly (**46 ms** apart).

And **optimal switching is preserved with age** — older adults switch *less*, but *strategically*, not from a deficit.

Optional: invert the walk → estimate the network

Estimated individual semantic networks for control and AD



Representative control Representative AD

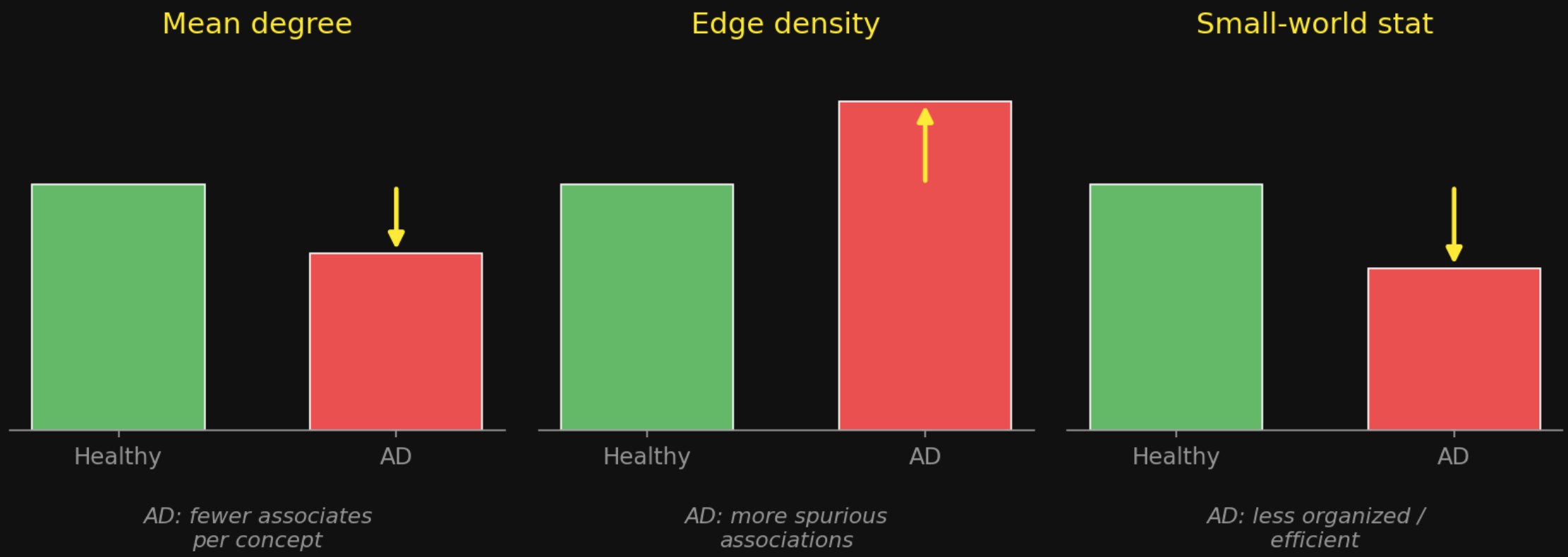
(Zemla & Austerweil, 2019)

So far: network → walk → fluency list. You can also run it **backwards** — given fluency lists, **infer the most likely network** (U-INVITE; Zemla & Austerweil 2018).

That makes the network a **measurement instrument**: estimate someone's semantic network from a few fluency lists, then compare groups.

Tool: **SNAFU** — `github.com/AusterweilLab/snafu-py`

Optional: Alzheimer’s changes the network



Zemla & Austerweil (2019): AD changes the STRUCTURE of the estimated network — a representation deficit, measurable from fluency lists.

Zemla & Austerweil (2019) estimated networks for 41 AD patients vs. healthy controls. Three structural differences point to an **impaired representation** — fewer associates per concept, more spurious links, less organized — *measured from fluency lists alone.* (A separate **retrieval** deficit — faulty monitoring of what’s already been said — shows up too.)

Bridge — we've been doing Monte Carlo

Look back at the day:

- **Running the chain** (Chibany's bento, the card shuffle) **drew samples** by comparing a random number to the row of P.
- The **stationary distribution** we found by **running the chain** and watching where it settles.

We were **estimating a distribution by sampling** — that's **Monte Carlo**.

Next week — Week 7

Monte Carlo — and we flip the problem.

Today the chain came first and we *found* its stationary distribution. **Next week we reverse it:** start with a target distribution we *want* to sample (e.g. a Bayesian posterior), and **design a Markov chain whose stationary distribution is that target.**

That's **Markov chain Monte Carlo (MCMC)** — and it leads to a wild idea: people may sample from their own posteriors (*MCMC with People*).

Required reading + presenter on the readings page.