SIM507 · WEEK 1 · PLAIN-LANGUAGE EDITION

The Research Simulator

Run · Artefact · Manifest — the three words that turn a computer experiment into evidence other people can trust. No prior simulation experience needed.

Jakub Dubec · jakub.dubec@stuba.sk · FIIT STU

Lecture 0 · First, the plain picture

What this field is, the words you need, and why anyone bothers

What is a research simulator? (one sentence)

If you remember one thing from this whole week, remember this.

IN ONE SENTENCE

A research simulator is a **computer program that runs an experiment you could not safely or cheaply run in the real world** — and records every decision it made, so that someone else can run the exact same experiment and get the **exact same answer**.

> 🧪 Like a chemistry lab that writes down every step, every amount, and every brand of every chemical — so a stranger across the world can repeat your experiment and reach your result.

The words you need

Simulator
A program that imitates a piece of reality so you can test ideas safely.

Run
One execution of a simulator with one specific set of choices.

Artefact
A file a run leaves behind: data, a summary, or a figure.

Manifest
A packing list that names every artefact a run should produce.

Seed
A single number that pins down all the "random" choices a run makes.

Campaign
A research question plus the runs gathered to answer it.

Every one of these comes back in context later — you don't have to memorise them now.

Why bother? The real-world stakes

SAFETY

Test a fire-drill or a packed-platform crowd **without** ever putting real people at risk.

COST

Run a thousand scenarios overnight for the price of electricity — impossible with real volunteers.

HONESTY

When a reviewer asks **"how do you know?"**, you can hand them the experiment to re-run themselves.

Science has a **reproducibility problem**: many published results cannot be repeated, and some get retracted. A simulator that records every choice is one honest answer to that problem.

Define

What a research simulator is, precisely

Before the formula: a baking recipe

> 🍰 Bake a cake with the **same recipe**, the **same ingredients**, and the **same model of oven** — and you get the same cake, every single time. Change any one of the three, and the cake changes.

A simulator is exactly that: ingredients + method + version → a result.

A simulator is a function

We turn the recipe into a tidy statement, one piece at a time.

1. **The ingredients (parameters).** What you set up: how big the room, how many people, which radio frequency. We call this bundle **theta**.
2. **The one random knob (seed).** A single number that fixes every "random" choice. We call it **s**.
3. **The exact version (code).** Which exact build of the program ran. We call it **c**.
4. **The result.** A fixed set of files left behind — the **artefacts**.

$$\text{simulator}(\,\theta,\; s,\; c\,) \;\longmapsto\; \{\,a_1,\, a_2,\, \ldots,\, a_n\,\}$$

Read it as a sentence: "feed in ingredients **θ**, random knob **s**, and exact version **c** — get out a fixed set of files $\mathbf{a_1} \ldots \mathbf{a_n}$." Nothing here mentions crowds or radio: the recipe is the same whatever the domain.

Two properties make it an instrument

A simulator becomes a **scientific instrument** only when it is

deterministic — same θ, s, c always give the same files

and **self-describing** — the files arrive with a manifest

Classify

Four kinds of simulator

Four kinds — the everyday version first

🚶 **The crowd mover** — like a tiny, realistic video game of people walking. You set the floor and the headcount; it tells you where everyone is, frame by frame.

📏 **The measurer** — takes someone else's walking paths and works out the numbers: how crowded, how fast. It moves nobody; it only measures.

📡 **The radio tracer** — traces how a wireless signal bounces around a room full of people, the way a ray-traced game traces light.

📓 **The notebook** — a worksheet of code run start-to-finish; the finished worksheet, figures and all, *is* the result.

Class 1 · The agent-based engine

> **Agent-based engine** — a simulator that moves many interacting people forward in time: you give it a floor and a headcount, it gives back where everyone is at every moment.

> 🐜 Like watching an ant farm in fast-forward: lots of little agents, each following simple rules, and you record every position.

In: floor geometry + agent count · **Out:** per-frame positions (a "trajectory").

Class 2 · The measurement library

> **Measurement library** — a simulator that takes walking paths it did *not* create and computes physical numbers from them: density, speed, and the like.

> 📐 Like a referee with a stopwatch and a tape measure: it produces no motion of its own, it only **measures** the motion that already happened.

In: a trajectory (from Class 1) · **Out:** metrics — numbers about the crowd.

Class 3 · The ray-traced channel

> **Ray-traced channel** — a simulator that sends a radio signal through a room full of walls and bodies and reports what the wireless channel looks like.

> 🔦 Like a video game tracing light bouncing off walls — only here it traces **radio** bouncing off walls and people, because bodies (mostly water) reflect radio.

In: a scene + the crowd's motion · **Out:** the radio channel (the signal's fingerprint).

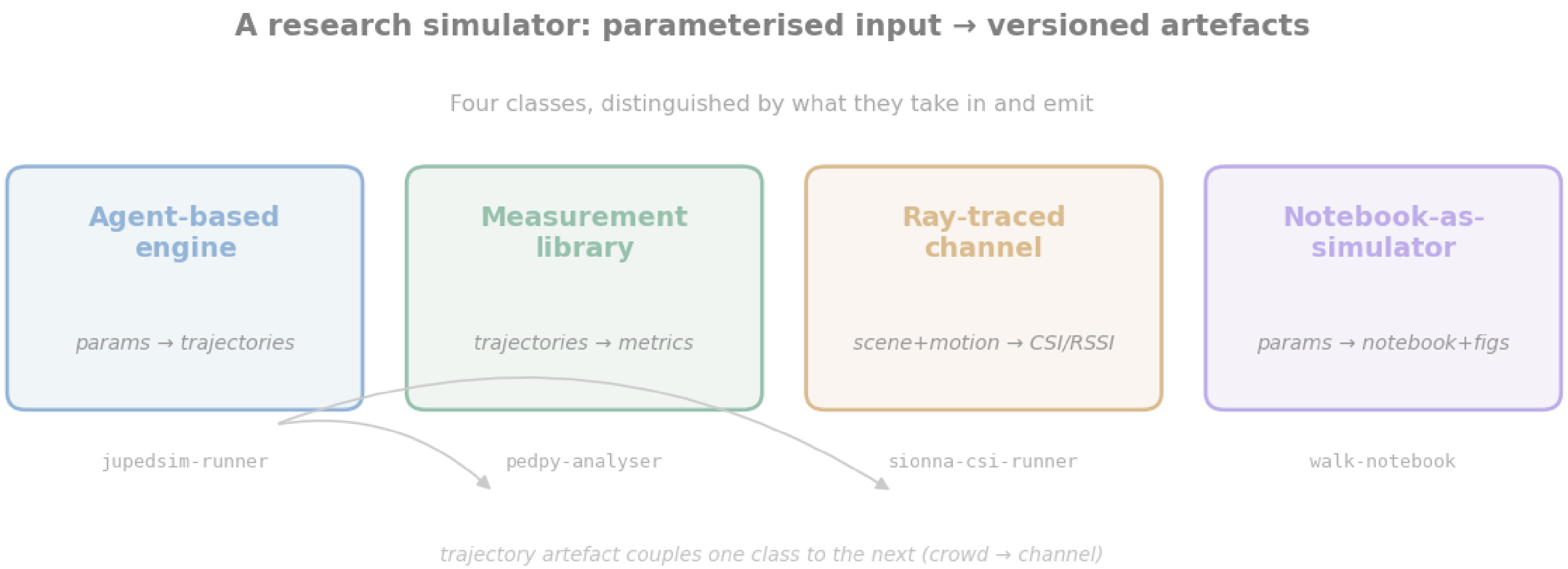
Class 4 · The notebook-as-simulator

> **Notebook-as-simulator** — a worksheet of code, run from top to bottom with chosen settings; the finished worksheet (text, numbers, and figures) is itself the result.

> 📒 Like a lab notebook that fills *itself* in: you set the inputs at the top, press run, and every page below — calculations and charts — writes out automatically.

In: parameters · **Out:** an executed notebook + its figures.

The four are not islands



Inspect

A real run and its packing list

Before the manifest: a shipping packing list

> 📦 Every box that leaves a warehouse carries a **packing list**: exactly what is inside, how many, and which items are essential versus optional. Open the box, check it against the list — nothing missing, nothing wrong.

> **Manifest** — a run's packing list: it names every artefact the run should produce, what kind of file each is, what role each plays, and whether each is required.

A real run from the lab

Abstractions earn their keep against a real example. Here is one real, gate-passed run.

WHAT THE LAB RAN

A pair of simulators on a small sandbox experiment: the crowd mover put **12 people** through a **180-second** scenario, and the measurer reported how crowded and how fast they moved.

CROWD MOVER

3601 frames · 12 agents · 180.0 s

MEASURER

density 0.125 ped/m^2 · speed 1.147 m/s

GATE

the platform checked the files against the packing list — **passed**

The six numbers of this run

Quantity	Value	Measured by
Frames recorded	**3601**	crowd mover
People (agents)	**12**	crowd mover
Duration	**180.0 s**	crowd mover
Mean crowd density	**0.125 ped/m²**	measurer
Mean walking speed	**1.147 m/s**	measurer
Mean "flow balance" error	**0.0769**	measurer

> **ped/m²** — pedestrians per square metre, the natural unit of how crowded a patch of floor is.

> **Flow balance error** — a bookkeeping check: of all the people who entered a patch of floor, how badly does "people in minus people out" fail to match the change in how many are standing there? A small number (here 0.077) means almost nobody is being miscounted.

What each file is for — its role

A run is not its console output. A run is its **files** plus the packing list that describes them. Each file has a **role**.

> **Role** — what a file is *for*: a headline summary, the raw motion data, a reduced table, a figure, and so on.

scalar-summary

A small file of headline numbers (checked against a contract).

trajectory

The big table of where everyone was, frame by frame.

per-bin-summary

A reduced table — one row per slice of time.

Three more roles

domain-specific

A result meaningful only to this field (e.g. the flow-balance check).

upstream-input

A file produced by one simulator and re-used by the next — how the platform records that the crowd mover fed the measurer.

figure

A rendered picture (a chart saved as an image).

> 🔗 The **upstream-input** role is the paper trail: it records, in the packing list itself, that the measurer's input box came straight from the crowd mover's output box.

The shape of a run’s evidence

- **8 files**, sorted into six roles
- Most are **derived**: 2 summaries, 2 figures
- One heavy **trajectory** anchors them all
- One **upstream-input** re-use = the crowd → measurement link

> **Gate** — the platform's check that the delivered files match the packing list. If a required file is missing or fails its contract, the run is **refused**. This run **passed**.

Reproduce

What a seed actually pins

Before the maths: a shuffled deck

🃏 Shuffle a deck of cards, then **cut it at exactly the same spot** every time, and the order comes out identical, run after run. The shuffle *looks* random — but pinned to the same starting cut, it repeats perfectly.

Seed — the "starting cut": a single number that pins every random choice a run makes. Same seed → same choices → same result.

A seed pins a random stream

Write the deck rule as a tidy statement.

1. Call the random generator **g**, and the seed **s**.
2. Feed it the **same** seed twice — you get the same numbers.
3. Feed it a **different** seed — you almost surely get different numbers.

$$g(s) = g(s) \quad \text{always,} \qquad g(s) \neq g(s') \ \ \text{for} \ \ s \neq s'$$

In words: same seed s → byte-identical numbers, every time; a different seed s′ → almost surely different numbers. Re-running with a fixed seed is **not** probabilistic — only *choosing* the seed is.

Two seeds, one run

A subtlety, because the real run shows it.

THE CATCH

The run records **two different seeds** — one for the campaign's run-selection, one for the crowd mover's scenario. They are not a contradiction; they pin **two different random streams**.

🎲 Two dice on the table, each with its own starting cut. Knowing one tells you nothing about the other — so you must write down **both**.

A reproducible run records every seed it depends on.

A version is a frozen edition

📕 "The recipe" is not enough — you need **which printing** of the cookbook. A book's ISBN, or a wax seal on an envelope, names *one exact frozen edition* and proves it has not changed.

Code version — the exact frozen edition of the program that ran: a unique fingerprint of the source code, plus a unique fingerprint of the packaged program. Same fingerprints → same program, guaranteed.

A seed alone is not enough

To reproduce a run, pin the **full triple**

θ (the ingredients) · **s** (every seed) · **c** (the frozen edition)

Pin all three and the **0.125 ped/m²** re-appears.

Change one and you should *expect* a different number — and that difference is information.

Operate

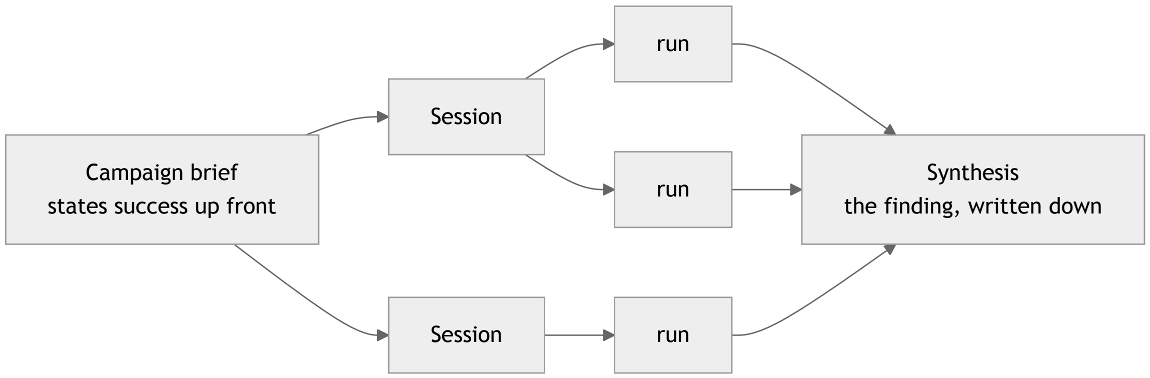
From a heap of runs to citable evidence

Why structure? One run is not a study

A single reproducible run is necessary but not sufficient. Research makes **many** runs.

> 🗄️ A thousand loose photos in a shoebox is not an album. The platform adds three layers of order so a pile of runs becomes something a colleague can navigate and cite.

Three layers of structure



- **Run** — one reproducible execution (everything from the Inspect section).
- **Campaign** — a research question with a written brief and explicit success criteria.
- **Synthesis** — reading across the runs and writing the finding back down.

The point: citability under reproduction

When a reviewer asks **"how do you know?"**,

the answer is a run they can **re-execute from the packing list alone**,

at the pinned recipe, seeds, and frozen edition,

and watch the same **0.125 ped/m²** come back.

Recap — the five verbs

- **Define** — a simulator = ingredients + seed + version → a fixed set of files; an *instrument* only when deterministic and self-describing.
- **Classify** — four kinds: crowd mover, measurer, radio tracer, notebook; the crowd mover's trajectory feeds the other two.
- **Inspect** — a run is its files plus a manifest (packing list); each file has a role; the gate refuses files that miss the contract.
- **Reproduce** — a seed pins one random stream; reproduce the full triple θ, s, c.
- **Operate** — run → campaign → synthesis turns a heap of runs into evidence **citable under reproduction**.

> **Next week:** the crowd-mover + measurer pair end-to-end — rebuilding the speed-vs-density curve from this very run.

The throughline

A simulation is a scientific instrument — and an instrument you cannot reproduce, version, and validate is not evidence. It is an opinion with a plot attached.

End of Week 1 · Questions welcome.

Further reading

SIM507 Curriculum

The course arc and where this week sits in it.

IP-078 — Simulation Platform

The run / artefact / manifest contract and the artefact store.

IP-081 — Research Campaigns

The brief → sessions → runs → synthesis workflow.