PS Angewandte Systemwissenschaften 1

Einheit 8: Szenarien, Analyse & Zwischenstand

Nachtrag aus Einheit 7, dann Projektarbeit und kurzer Status pro Gruppe

David Maier
Institut für Umweltsystemwissenschaften
Universität Graz

Heute

Szenarien nachholen

Vom animierten Modell zu reproduzierbaren Simulationsexperimenten.

Auf Projekte übertragen

Welche Parameter, Messgrößen und Vergleichsfälle passen zu euren Modellen?

Zwischenstand

Kurzer Status, nächste Arbeitsschritte, betreute Projektzeit.

Ziel der Einheit: Jede Gruppe hat mindestens eine konkrete Szenarioidee, eine Messgröße und den nächsten Commit vor Augen.

Stand nach Einheit 7

Geschafft

- Repositories und GitHub-Zugänge geprüft
- Git-Workflow besprochen
- Projektarbeit in den Gruppen gestartet

Heute nachholen

- Was ist ein Szenario?
- Warum brauchen wir Replikate?
- Wie lesen wir Modelloutputs systematisch?

> Ein ABM wird erst dann wissenschaftlich brauchbar, wenn wir nicht nur eine Animation sehen, sondern kontrollierte Läufe vergleichen.

Nachtrag

Szenarien & Sensitivität

Warum wir Modelle nicht nur anschauen, sondern systematisch experimentieren lassen.

Animation ist noch keine Analyse

Animation

Wir sehen, ob das Modell grundsätzlich funktioniert und welche Muster entstehen.

Experiment

Wir speichern Daten, wiederholen Läufe und vergleichen Bedingungen kontrolliert.

Railsback & Grimm: Ein lauffähiges ABM ist ein virtuelles Labor. Wir verstehen es durch kontrollierte Simulationsexperimente.

Szenario und Replikat

Szenario

Eine konkrete Kombination aus Parameterwerten, Anfangsbedingungen und Modellannahmen.

```
slowdown_prob = 0.3
n_cars = 22
safe_distance = 2
```

Replikat

Derselbe Parameter-Setup noch einmal, aber mit anderen Zufallsereignissen, also einem anderen Seed.

```
seed = 1
seed = 2
seed = 3
```

> Bei stochastischen ABMs ist ein einzelner Lauf nur eine Beobachtung, nicht schon ein Ergebnis.

Was ist ein Random Seed?

Pseudozufall

Computer erzeugen Zufallszahlen mit einem Algorithmus. Der **Seed** ist der Startwert für diese Folge.

```
random.seed(42)
```

Reproduzierbarkeit

Gleicher Seed bedeutet gleiche Zufallsereignisse. Unterschiedliche Seeds erzeugen unterschiedliche Replikate.

```
scenario A, seed 1
scenario A, seed 2
scenario A, seed 3
```

Seeds helfen uns, Zufall kontrolliert zu verwenden: wir können Ergebnisse wiederholen und trotzdem Streuung messen.

Was ist Sensitivitätsanalyse?

Sensitivitätsanalyse fragt, wie stark Modelloutputs reagieren, wenn wir Parameterwerte verändern.

Parameter

Annahmen im Modell, z.B. Zufallsbremsen, Sicherheitsabstand oder Verkehrsdichte.

Output

Gemessene Ergebnisse, z.B. Durchschnittsgeschwindigkeit oder Anzahl langsamer Fahrzeuge.

Reaktion

Ändert sich fast nichts, etwas, oder kippt das System in ein anderes Verhalten?

Railsback & Grimm (2019), Kap. 23

Warum machen wir Parameter-Sweeps?

Verstehen

- Welche Annahmen kontrollieren das Ergebnis?
- Wo entstehen Kipppunkte?
- Welche Muster sind robust?

Vertrauen prüfen

- Hängt unser Ergebnis an einem einzigen Wert?
- Ist der Effekt größer als Zufallsschwankung?
- Welche Parameter müssen wir genauer begründen?

Hohe Sensitivität ist nicht automatisch schlecht. Sie zeigt, welcher Prozess im Modell besonders wichtig ist.

Eine einfache Strategie

1. Ein Modell vorläufig einfrieren: Code läuft, Grundverhalten ist plausibel.
2. Eine Messgröße wählen: z.B. durchschnittliche Geschwindigkeit.
3. Einen Parameter über sinnvolle Werte variieren.
4. Pro Wert mehrere Replikate mit unterschiedlichen Seeds ausführen.
5. Ergebnisse plotten und als Modellverhalten interpretieren.

> Für den Einstieg reicht oft eine Single-Parameter-Sensitivitätsanalyse: ein Parameter nach dem anderen.

Unsere Messgrößen im Traffic-Modell

Im Traffic-Modell speichern wir bewusst einfache Größen, die man ohne Statistik-Vorwissen lesen kann.

Zeitreihe

- `time`: Simulationszeit
- `average_speed`: mittlere Geschwindigkeit

Stau-Indikatoren

- `stopped_cars`: stehende Fahrzeuge
- `slow_cars`: langsame Fahrzeuge

Eine Messgröße ist immer eine Entscheidung: Sie bestimmt, was wir am Modell als wichtig ansehen.

Was ist ein CLI?

Command Line Interface

Ein CLI ist eine Bedienoberfläche im Terminal. Wir starten ein Programm mit einem Befehl und geben Werte als Argumente mit.

```
uv run main.py sweep slowdown_prob 0.1 0.2 0.3
```

Warum für Szenarien?

- Parameter können von außen gesetzt werden
- Gleiche Läufe lassen sich exakt wiederholen
- Befehle können im README dokumentiert werden
- Viele Szenarien können automatisch gestartet werden

> Eine Animation hilft beim Verstehen. Ein CLI hilft beim Analysieren, weil Experimente reproduzierbar und automatisierbar werden.

CLI in Python mit argparse

Argumente definieren

```python
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("parameter")
parser.add_argument("values", nargs="+", type=float)
parser.add_argument("--replicates", type=int)

args = parser.parse_args()
```

Argumente verwenden

```python
run_sweep(
    args.parameter,
    args.values,
    args.replicates,
)
```

`argparse` liest Text aus dem Terminal und macht daraus normale Python-Werte, z.B. Strings, Zahlen und Listen.

Beispiel: Ein einfacher Sweep

Terminal

```
uv run main.py sweep slowdown_prob 0.0 0.1 0.2 0.3 0.4 \
  --replicates 5 \
  --sim-duration 60 \
  --output runs/slowdown-sweep
```

Leitfragen

- Ab wann sehen wir regelmäßig Staus?
- Ist der Effekt größer als die Streuung zwischen Replikaten?
- Bleibt das Ergebnis bei anderen plausiblen Parametern ähnlich?

Theorie und Code treffen sich hier: Wir formulieren eine Erwartung, lassen das Modell laufen und lesen die Daten.

Beispiele für Szenariofragen

Gittermodelle

- Waldbrand: Wie verändert Baumdichte die Brandgröße?
- Schelling: Ab welchem Toleranzwert entsteht Segregation?

Netzwerk und Umwelt

- SIR: Wie stark verändert Netzwerkdichte die Epidemiegröße?
- Fishery: Werden bei koordinierter Befischung mehr Fische gefangen?

Achtung: Nicht zu viele Dinge gleichzeitig verändern. Sonst ist am Ende unklar, welcher Mechanismus das Ergebnis erzeugt.