

On the Responsible use of Pseudo-Random Number Generators in Scientific Research

Charlie Rahal

LCDS and Nuffield College, University of Oxford

ECSR

[REDACTED] LEVERHULME
[REDACTED] CENTRE
[REDACTED] FOR
DEMOGRAPHIC
SCIENCE



What is a PRNG?

"A pseudo-random number generator (PRNG) is an algorithm that generates a sequence of numbers approximating true randomness. However, unlike true random numbers, PRGNs are deterministic, meaning they rely on an initial value called a 'seed' and follow a predictable pattern. Though not truly random, they are widely used in computer simulations, cryptography, and games due to their efficiency and ability to produce long sequences of seemingly random numbers with minimal computational resources."

- GPT-4o, (2024).

PRGNs are just one type of RNG!

	'RN'	'RN*' and 'Hardware'	'RN*' and 'Quantum'	'RN*' and 'Pseudo'	'RN*' and 'Quasi'
Health Sciences	0.406	0.000646	0.000407	0.00109	0.004201
Life Sciences	0.292	0.000516	0.001336	0.00179	0.001202
Physical Sciences	0.225	0.005430	0.007717	0.00726	0.003884
Social Sciences	0.092	0.000216	0.000242	0.00055	0.002420

A cursory scientometric analysis of Random Numbers (RN*). Data comes from OpenAlex API. All numbers are % of the entire scientific record.

- More 'PRNG papers' in CS (4222) than others combined.
- %PRGN papers since 1970: 0.0008% → 0.0048%.

PRNGs occur everywhere.

Computational Sciences:

'Applied' (Health/Social) Sciences:

- Numerical Integration
 - Cryptography
 - Genetic Algorithms
 - Signal Processing
 - Deep Learning
 - Weather Forecasting
 - Procedural Content
 - ...
- RCTs/Survey Sampling
 - Bootstrapping
 - Epidemiological Modeling
 - Econometric Estimation
 - Data Imputation
 - Topic Modeling
 - ABMs
 - ...

MT64: ubiquitous PRNG implementation (R, Stata, Python, ...)

So, why do we care about PRNGs?

An invisible source of uncertainty in the scientific record: PRNGs!

So, why do we care about PRNGs?

An invisible source of uncertainty in the scientific record: PRNGs!

- **Q:** Has anyone ran a program twice, with different results?

So, why do we care about PRNGs?

An invisible source of uncertainty in the scientific record: PRNGs!

- **Q:** Has anyone ran a program twice, with different results?
- **Q:** Has anyone here ever set a '[seed](#)'? Which seed?

So, why do we care about PRNGs?

An invisible source of uncertainty in the scientific record: PRNGs!

- **Q:** Has anyone ran a program twice, with different results?
- **Q:** Has anyone here ever set a '[seed](#)'? Which seed?
 - Maybe you prefer `set.seed(42)`?

So, why do we care about PRNGs?

An invisible source of uncertainty in the scientific record: PRNGs!

- **Q:** Has anyone ran a program twice, with different results?
- **Q:** Has anyone here ever set a ‘seed’? Which seed?
 - Maybe you prefer `set.seed(42)`?
 - Or `numpy.random.seed(123)`?

So, why do we care about PRNGs?

An invisible source of uncertainty in the scientific record: PRNGs!

- **Q:** Has anyone ran a program twice, with different results?
- **Q:** Has anyone here ever set a ‘seed’? Which seed?
 - Maybe you prefer `set.seed(42)`?
 - Or `numpy.random.seed(123)`?
- Why did you do this?

So, why do we care about PRNGs?

An invisible source of uncertainty in the scientific record: PRNGs!

- **Q:** Has anyone ran a program twice, with different results?
- **Q:** Has anyone here ever set a ‘seed’? Which seed?
 - Maybe you prefer `set.seed(42)`?
 - Or `numpy.random.seed(123)`?
- Why did you do this?
 - Maybe to eliminate variation in algorithms with PRNGs?

So, why do we care about PRNGs?

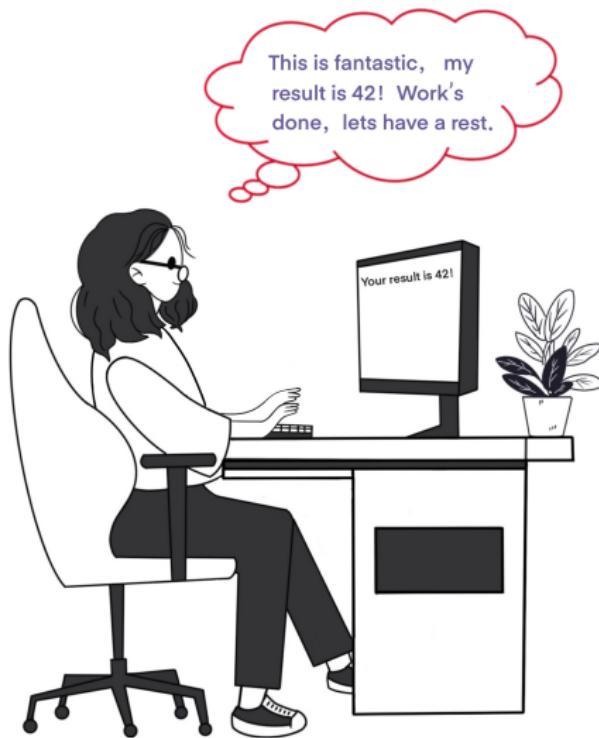
An invisible source of uncertainty in the scientific record: PRNGs!

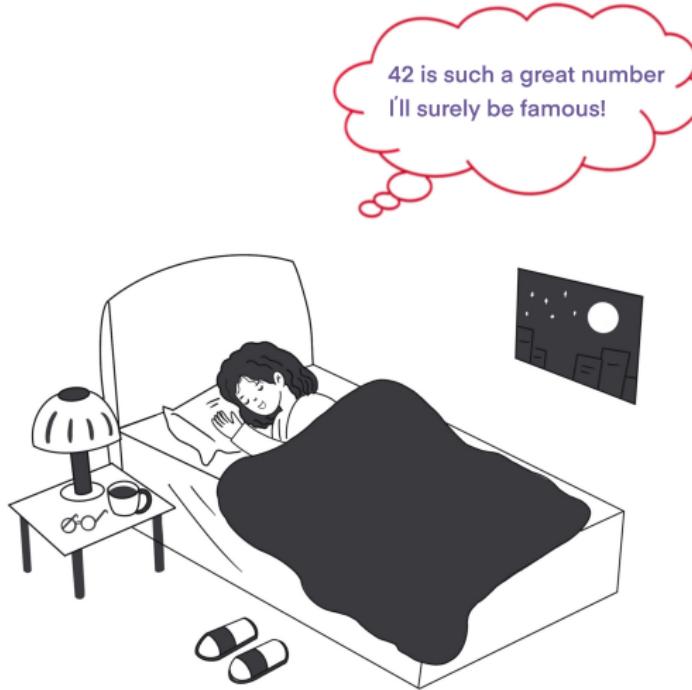
- **Q:** Has anyone ran a program twice, with different results?
- **Q:** Has anyone here ever set a ‘seed’? Which seed?
 - Maybe you prefer `set.seed(42)`?
 - Or `numpy.random.seed(123)`?
- Why did you do this?
 - Maybe to eliminate variation in algorithms with PRNGs?
 - This seems to be the current ‘best practice’ advice.

So, why do we care about PRNGs?

An invisible source of uncertainty in the scientific record: PRNGs!

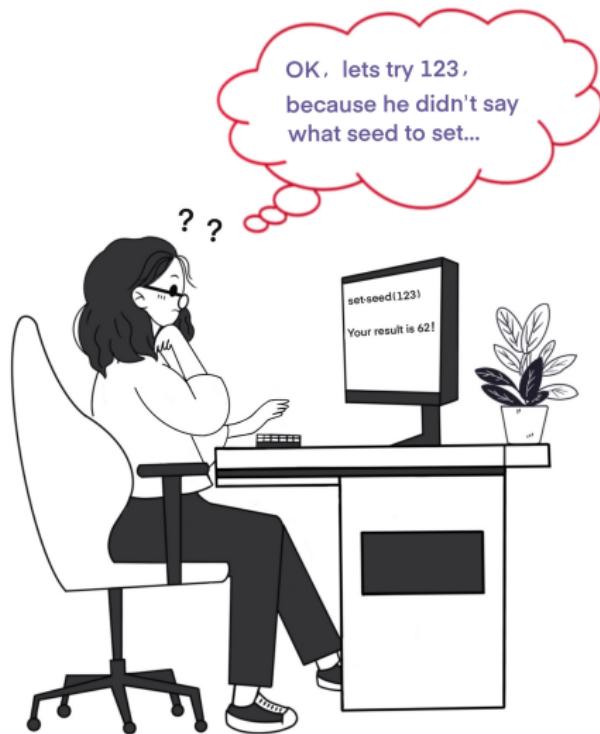
- **Q:** Has anyone ran a program twice, with different results?
- **Q:** Has anyone here ever set a ‘seed’? Which seed?
 - Maybe you prefer `set.seed(42)`?
 - Or `numpy.random.seed(123)`?
- Why did you do this?
 - Maybe to eliminate variation in algorithms with PRNGs?
 - This seems to be the current ‘best practice’ advice.
- This is very well intentioned: it allows reproducibility. Yay!

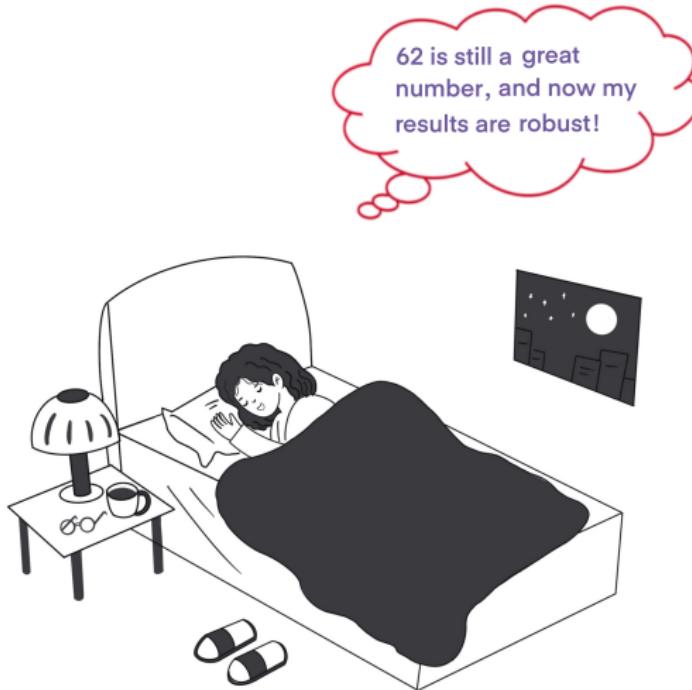


















Pseudo-Random Number Generation (Cont.)

- We want need to assess variation independent of seed choice.

Pseudo-Random Number Generation (Cont.)

- We ~~want~~ need to assess variation independent of seed choice.
- An extremely important and scarcely researched problem.
 - In well designed research, it shouldn't matter. But, it does.

Pseudo-Random Number Generation (Cont.)

- We ~~want~~ need to assess variation independent of seed choice.
- An extremely important and scarcely researched problem.
 - In well designed research, it shouldn't matter. But, it does.
- Pockets within CS/Physics community where this appreciated.
 - Some CS conferences do request examination of instantiation.

Pseudo-Random Number Generation (Cont.)

- We ~~want~~ need to assess variation independent of seed choice.
- An extremely important and scarcely researched problem.
 - In well designed research, it shouldn't matter. But, it does.
- Pockets within CS/Physics community where this appreciated.
 - Some CS conferences do request examination of instantiation.
- Outside of these narrow fields, it's **severly** under-appreciated.

Pseudo-Random Number Generation (Cont.)

- We ~~want~~ need to assess variation independent of seed choice.
- An extremely important and scarcely researched problem.
 - In well designed research, it shouldn't matter. But, it does.
- Pockets within CS/Physics community where this appreciated.
 - Some CS conferences do request examination of instantiation.
- Outside of these narrow fields, it's **severely** under-appreciated.
- The variation in estimand can be **huge** (as we'll show).

Pseudo-Random Number Generation (Cont.)

- We ~~want~~ need to assess variation independent of seed choice.
- An extremely important and scarcely researched problem.
 - In well designed research, it shouldn't matter. But, it does.
- Pockets within CS/Physics community where this appreciated.
 - Some CS conferences do request examination of instantiation.
- Outside of these narrow fields, it's **severly** under-appreciated.
- The variation in estimand can be **huge** (as we'll show).
- We bring attention to this through multiple types of replications.
 - Simulations, machine learning, NLP, and inferential research.

“If any research design is susceptible to this kind of variation, there is a problem”.

– Anonymous Colleague, Oxford (2019).

The General Premise

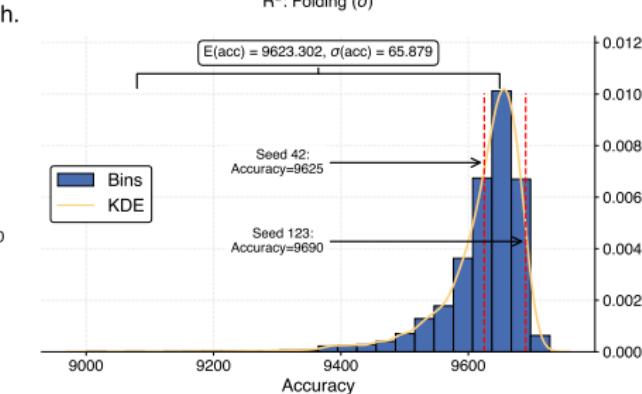
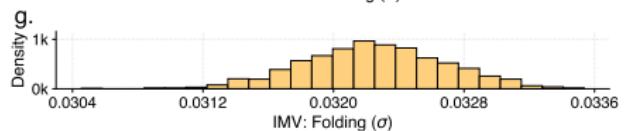
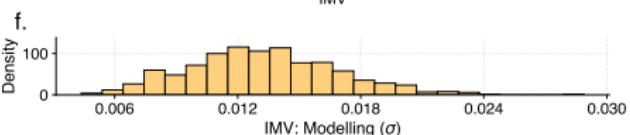
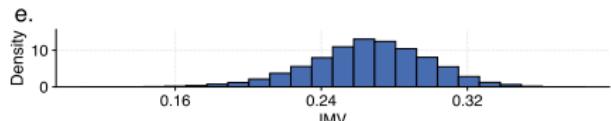
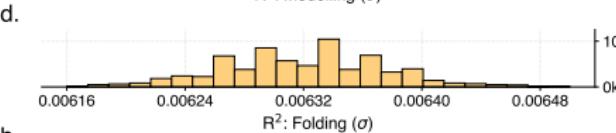
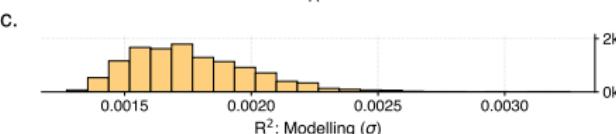
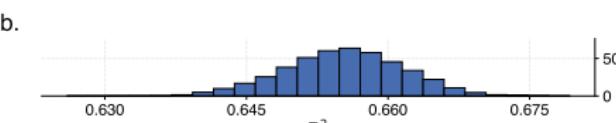
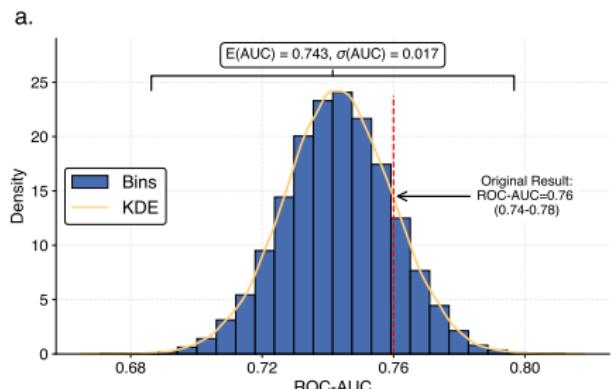
- **Problem Statement:** By setting **one** seed in applied algorithmic pipelines, we ignore the variation of our estimand as a function of how PRNGs were instantiated, This is computationally un-intensive, but scientifically dangerous.

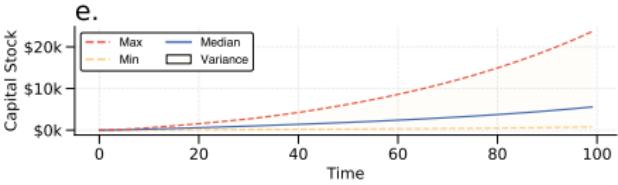
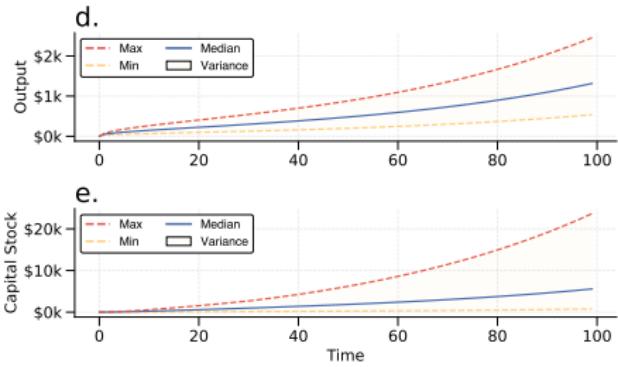
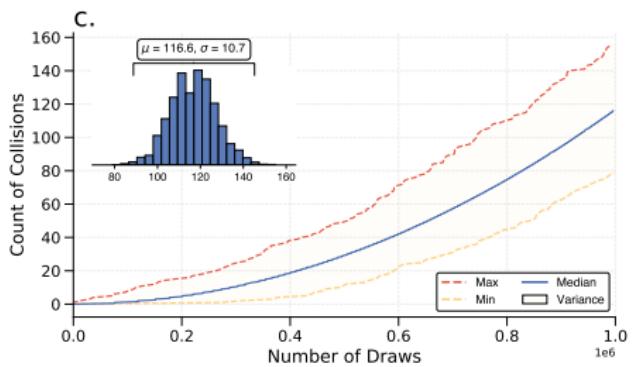
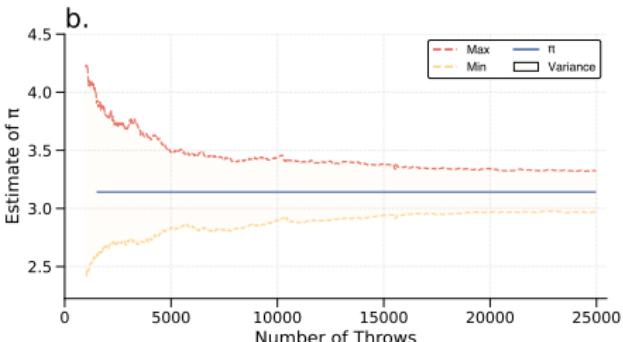
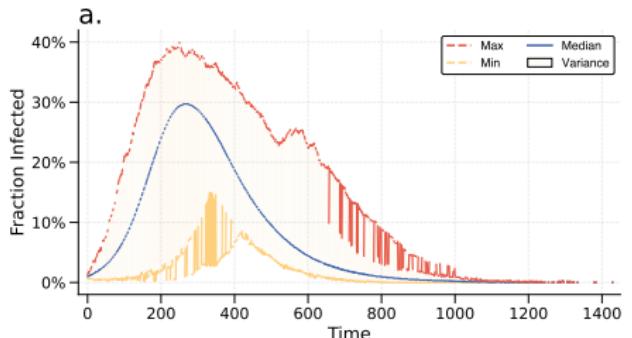
The General Premise

- **Problem Statement:** By setting **one** seed in applied algorithmic pipelines, we ignore the variation of our estimand as a function of how PRNGs were instantiated. This is computationally un-intensive, but scientifically dangerous.
- **Solution:** Visualize the outcome space of a **large number** (10k? 100k?) of seeds simultaneously. This is computationally intensive, but scientifically responsible.

The General Premise

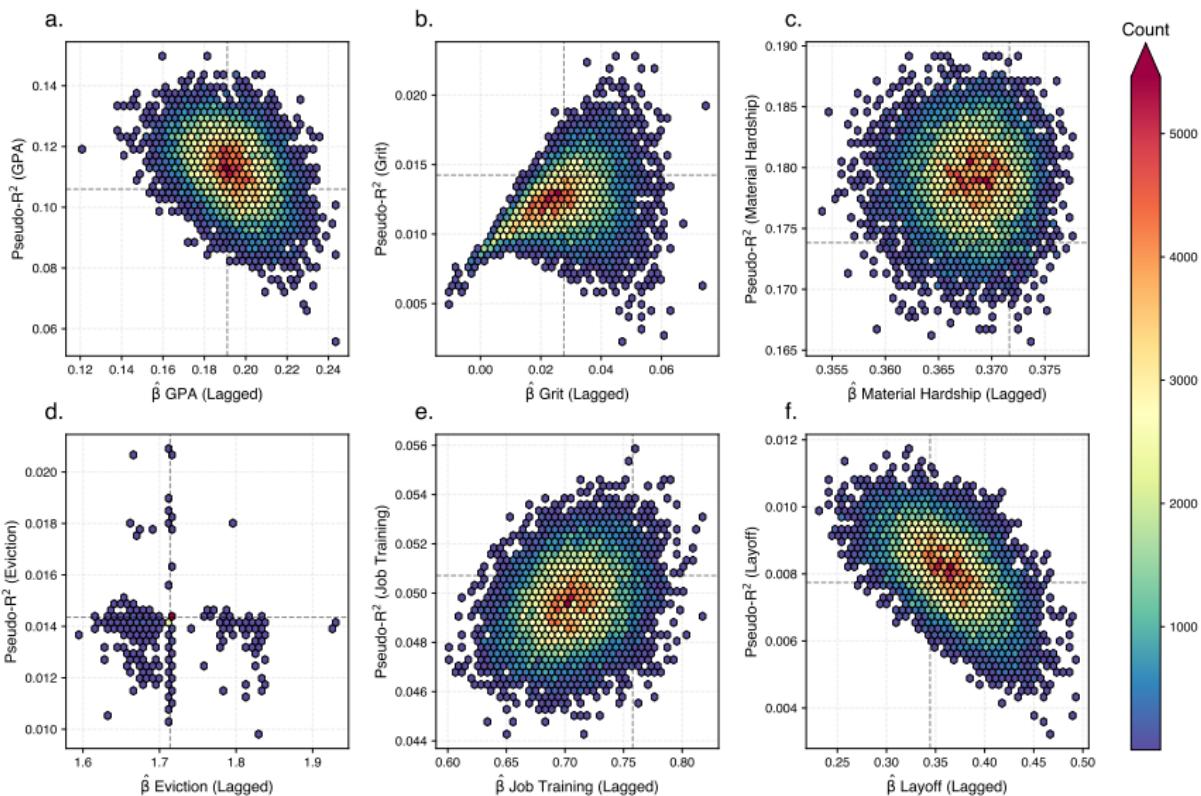
- **Problem Statement:** By setting **one** seed in applied algorithmic pipelines, we ignore the variation of our estimand as a function of how PRNGs were instantiated. This is computationally un-intensive, but scientifically dangerous.
- **Solution:** Visualize the outcome space of a **large number** (10k? 100k?) of seeds simultaneously. This is computationally intensive, but scientifically responsible.
- **Replication:** Similar to specification curves and p-hacking, we can retrospectively consider whether an original result is in the tail/IQR of the distribution of possible outcome space.





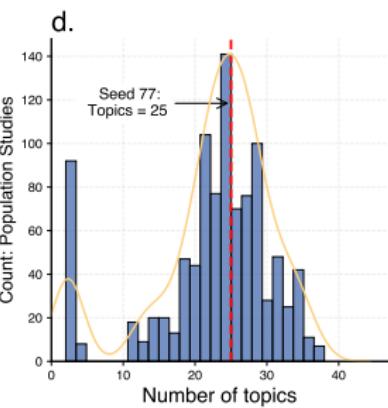
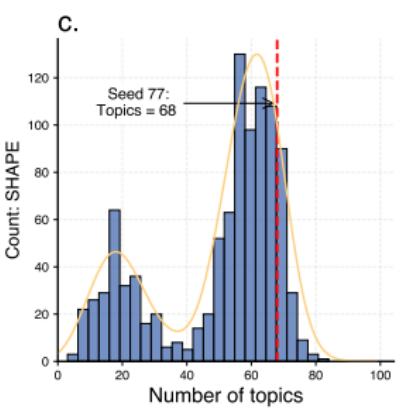
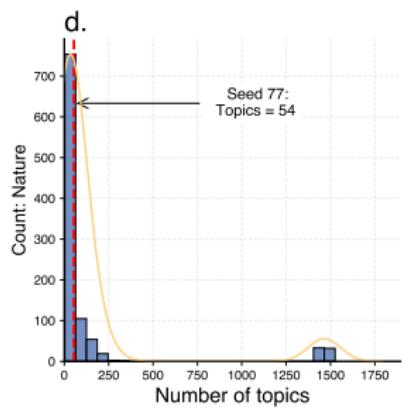
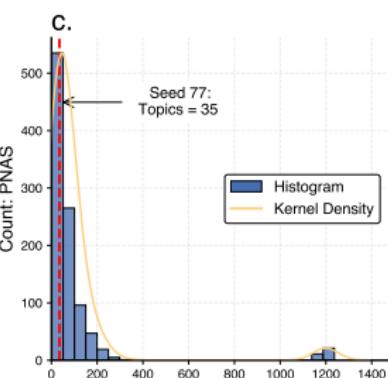
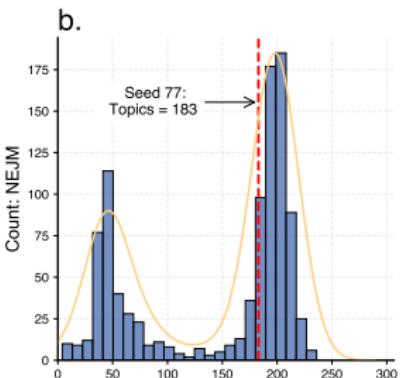
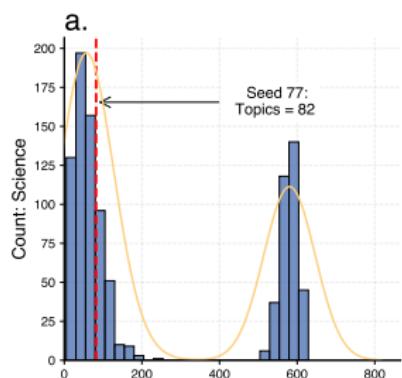
More Complex Replications

- Lets move into more complex and direct replications.
 - Major focus on the quantitative/computational social sciences.
 - Machine/Deep Learning
 - Sociology
 - Digital Humanities
 - Econometrics
 - Time Series Analysis
 - Urban Segregation
- Both recent and classical papers.
 - For each example, we'll give intuition as to what's going on.
- In total, this project replicates about 15 different papers.



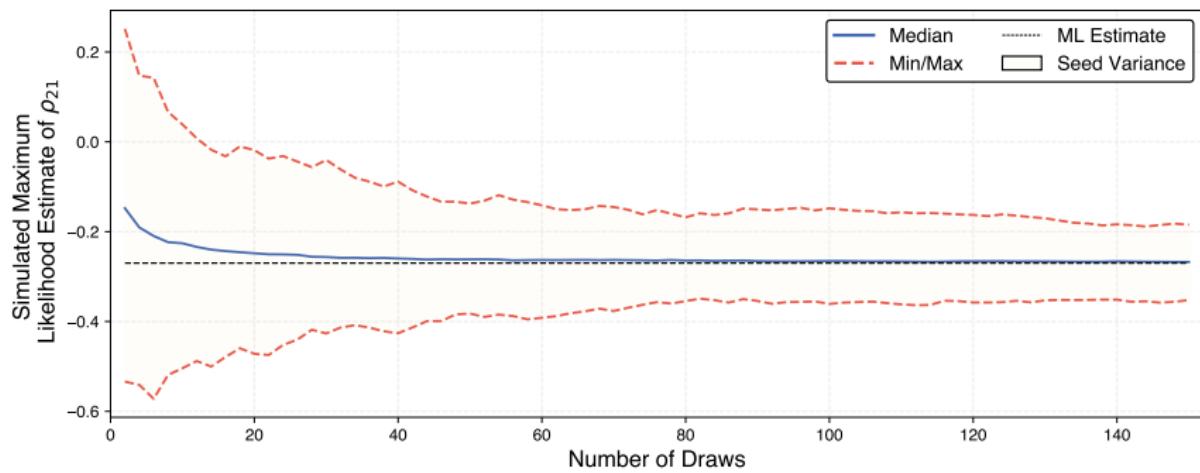
What's going on here?

- Amelia fills in missing data by drawing values from distributions.
- These distributions are based on observed data patterns.
- The imputed values reflect the uncertainty in the missing data.
- This is a form of Multiple Imputation.
- This results in entirely different models being built.
- This has implications for both inference and prediction.



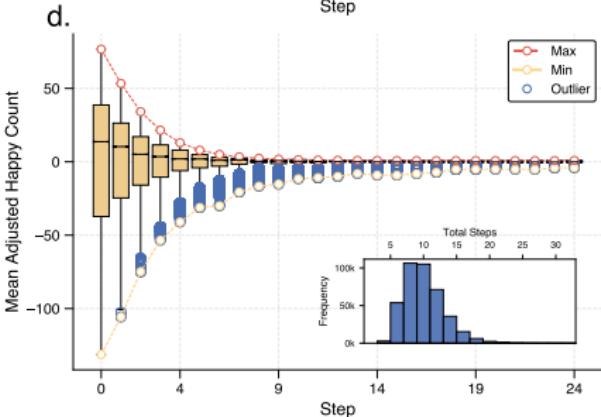
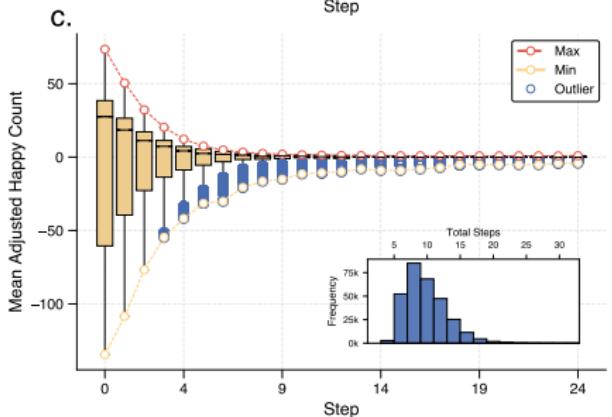
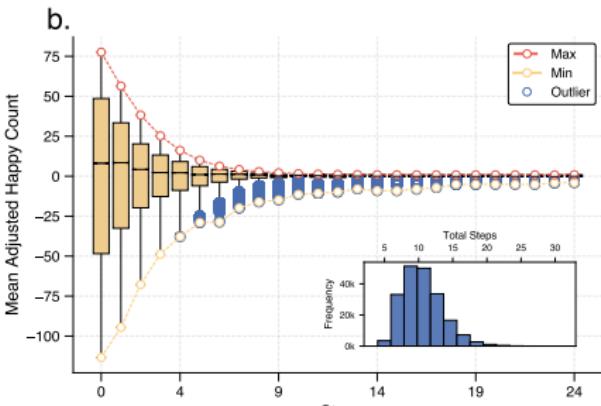
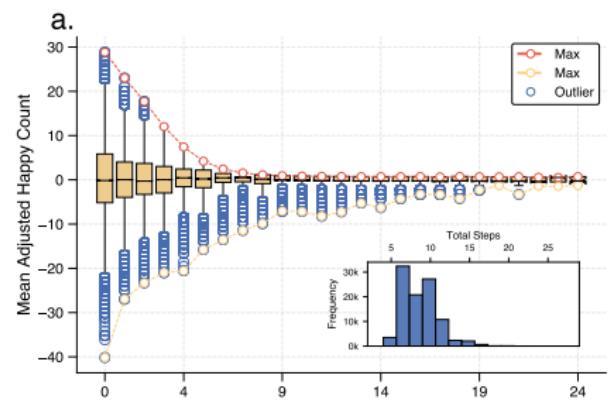
What's going on here?

- BERTopic has many process: one particularly volatile to seeds.
- It commonly involves dimension reduction via UMAP.
 - UMAP: causes different low-dimensional embeddings.
- Note: CUML implementation of UMAP gives variable outputs for spectral initialisation *even when setting the seed*.
- If using HDBScan for clustering, no seed variability.
- In general here, the seed variance looks **enormous**.



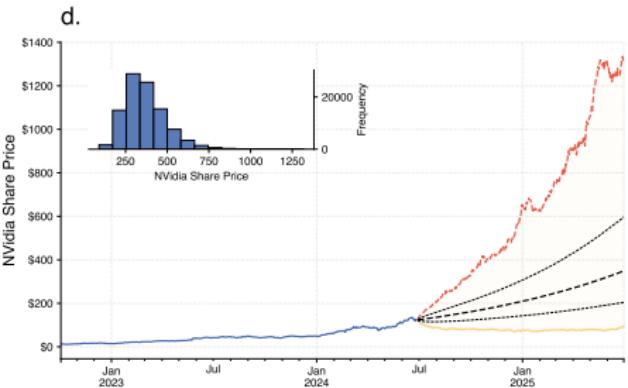
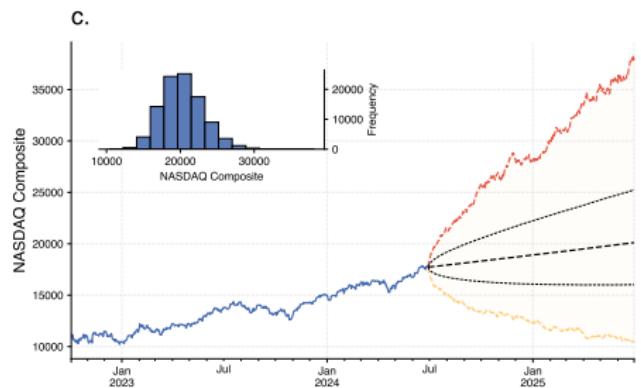
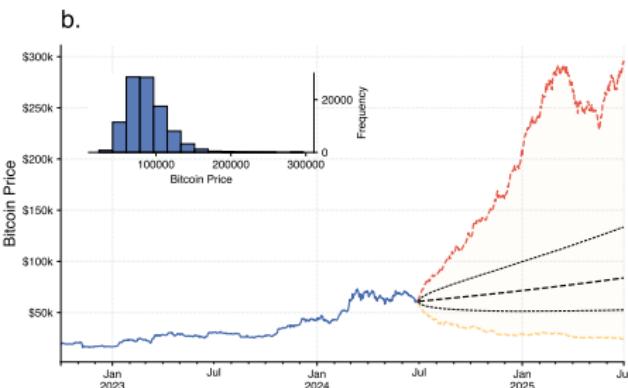
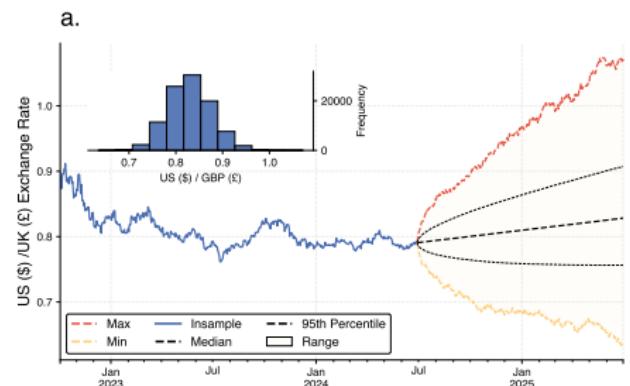
What's going on here?

- The `mvprobit` estimator requires evaluating M integrals jointly.
- This can be rewritten into a multivariate normal distributions.
 - Then, a sequence of normal distributions with certain properties.
- `mvprobit` samples from this to get at the underlying likelihood.
- Rewrite each individual normal to its CDF, which allows you to sample by just entering a uniform value between $[0,1]$.
 - This random value is seed-dependent.
- Where CDF is steep on a small interval, seeds matter a lot.



What's going on here?

- There are *multiple* (!) sources of PRNG in the Schelling Model:
 1. The initialisation of the agents on the grid.
 2. The shuffling of unhappy agents and empty cells.
- Agents might start more/less segregated from the beginning.
 - This affects the speed of total segregation over time.
- Even if the model reaches similar segregation, configuration of agents and trajectories will differ.
- As shown in the previous slide, parameterisation matters (a lot).



What's going on here?

- Seeds modify sequences of PRNG draws, affecting y_t .
- Moments of previous changes ($\mu, \sigma/2$) control magnitude.
- Draws of $\mathcal{N}(\mu, \sigma/2)$ impact asset price trajectory.
- The seed affects long-term trends by influencing both direction and size of early random steps.

This can be formally written as:

$$y_t = y_{t-1} + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(\mu, \sigma/2) \quad (1)$$

- Previously thought that nothing outperforms RW for ForEx.
- RWs commonly used as a benchmark for structural models.

Concluding Thoughts (Part One)

- Matt Salganik commented: 'That sounds great, but expensive!'
 - Expense no-longer prohibitive: everything here ran locally.
 - We live in an age of vast computational resources.

Concluding Thoughts (Part One)

- Matt Salganik commented: 'That sounds great, but expensive!'
 - Expense no-longer prohibitive: everything here ran locally.
 - We live in an age of vast computational resources.
- What are our suggestions?

Concluding Thoughts (Part One)

- Matt Salganik commented: 'That sounds great, but expensive!'
 - Expense no-longer prohibitive: everything here ran locally.
 - We live in an age of vast computational resources.
- What are our suggestions?
 - **Researchers:** Visualize your seed variability where affordable!
 - There are times when you can eliminate/reduce seed variability.
 - Pedagogically, let's teach students to parallelize code effectively.

Concluding Thoughts (Part One)

- Matt Salganik commented: 'That sounds great, but expensive!'
 - Expense no-longer prohibitive: everything here ran locally.
 - We live in an age of vast computational resources.
- What are our suggestions?
 - **Researchers:** Visualize your seed variability where affordable!
 - There are times when you can eliminate/reduce seed variability.
 - Pedagogically, lets teach students to parallelize code effectively.
 - **Reviewers:** Question papers you read – is there stochasticity?

Concluding Thoughts (Part One)

- Matt Salganik commented: 'That sounds great, but expensive!'
 - Expense no-longer prohibitive: everything here ran locally.
 - We live in an age of vast computational resources.
- What are our suggestions?
 - **Researchers:** Visualize your seed variability where affordable!
 - There are times when you can eliminate/reduce seed variability.
 - Pedagogically, let's teach students to parallelize code effectively.
 - **Reviewers:** Question papers you read – is there stochasticity?
 - Even well established results are susceptible.

Concluding Thoughts (Part One)

- Matt Salganik commented: 'That sounds great, but expensive!'
 - Expense no-longer prohibitive: everything here ran locally.
 - We live in an age of vast computational resources.
- What are our suggestions?
 - **Researchers:** Visualize your seed variability where affordable!
 - There are times when you can eliminate/reduce seed variability.
 - Pedagogically, let's teach students to parallelize code effectively.
 - **Reviewers:** Question papers you read – is there stochasticity?
 - Even well established results are susceptible.
 - **Editors:** Ubiquitous mandate for responsible use of PRNGs!

simple_seed_algorithm.py

```
import numpy as np # for randomisation
import matplotlib.pyplot as plt # for plotting

def analytical_function(input, seed):
    """ Simple analytical function: can be anything"""
    np.random.seed(seed)
    return input*np.random.normal()

results = [] # store results
inputs = 42 # dataset, figure path, etc.

for seed in range(0, 100000): # simple loop; can be distributed
    results.append(analytical_function(inputs, seed))

plt.hist(results) # plot results
```

better_seed_algorithm.py

```
import numpy as np # for randomisation
import matplotlib.pyplot as plt # for plotting

def analytical_function(input, seed):
    """ Simple analytical function: can be anything """
    np.random.seed(seed)
    return input*np.random.normal()

results = [] # store results
inputs = 42 # dataset, figure path, etc.

# Instead, use list of predefined, complex 'secret' seeds we provide
with open(os.path.join( '..', 'assets', 'seed_list.txt')) as f:
    seed_list = [int(line.rstrip('\n')) for line in f]

for seed in seed_list: # simple loop; can be distributed
    results.append(analytical_function(inputs, seed))

plt.hist(results) # plot results
```

Concluding Thoughts (Part Two)

- Ideally, we want to move towards distributional reproducibility.
- Moving forward, PRGNs shouldn't exist (QRNGs the norm).
- Does anyone have ideas for other types of seed variability?
- Can this be corrective? Index historical seed variability?
- Prospectively: we make available a list of seeds (replication materials), encouraging their use as a pre-specified set.
- **TLDR:** when variation can't be eliminated, should be visualised.

Thanks to my Coauthors!



Mark Verhagen



Jiani Yan

Thanks to my Coauthors!



Mark Verhagen



Jiani Yan

And thanks to you for your attention and attendance!