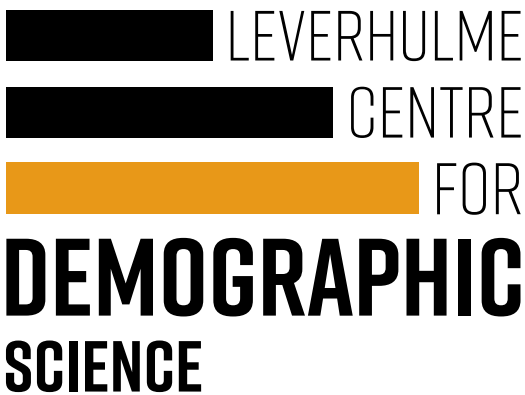
On the Responsible use of Pseudo-Random Number Generators in Scientific Research

Charlie Rahal

LCDS and Nuffield College, University of Oxford

Ox | Ber 2023



What is a PRNG?

ChatGPT

A pseudo-random number generator (PRNG) is a deterministic algorithm creating seemingly random number sequences using a specified seed. Unlike true random generators relying on unpredictable physical processes, PRNGs offer repeatability. This trait is beneficial in research, ensuring consistent results with the same seed, essential for reproducibility in various applications.

Pseudo-Random Number Generation

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

Pseudo-Random Number Generation

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

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

Pseudo-Random Number Generation

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?

Pseudo-Random Number Generation

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 to eliminate variation in algorithms with PRNGs?
  - This seems to be the current 'best practice' advice.

Pseudo-Random Number Generation

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 to eliminate variation in algorithms with PRNGs?
  - This seems to be the current 'best practice' advice.
- We argue this is the opposite of what we want!
  - We propose you pre-specify **multiple** (complex) seeds.

Pseudo-Random Number Generation

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 to eliminate variation in algorithms with PRNGs?
  - This seems to be the current 'best practice' advice.
- We argue this is the opposite of what we want!
  - We propose you pre-specify **multiple** (complex) seeds.
  - Or, alternatively, use a subset of the seeds **we** provide.

Pseudo-Random Number Generation (Cont.)

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

Pseudo-Random Number Generation (Cont.)

- We ~~want~~ need to assess possible variation independent of arbitrary seed choice.
- An extremely important and scarcely researched problem.

Pseudo-Random Number Generation (Cont.)

- We ~~want~~ need to assess possible variation independent of arbitrary seed choice.
- An extremely important and scarcely researched problem.
- Pseudo-Random Number Generators occur everywhere!

Pseudo-Random Number Generation (Cont.)

- We ~~want~~ need to assess possible variation independent of arbitrary seed choice.
- An extremely important and scarcely researched problem.
- Pseudo-Random Number Generators occur everywhere!
- The variation in estimand can be **huge** (as we'll show).

Pseudo-Random Number Generation (Cont.)

- We ~~want~~ need to assess possible variation independent of arbitrary seed choice.
- An extremely important and scarcely researched problem.
- Pseudo-Random Number Generators occur everywhere!
- 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.
- Seed variation at times: i.) can be eliminated, ii.) is not a concern, iii.) cannot be eliminated (requires examination).

The General Premise

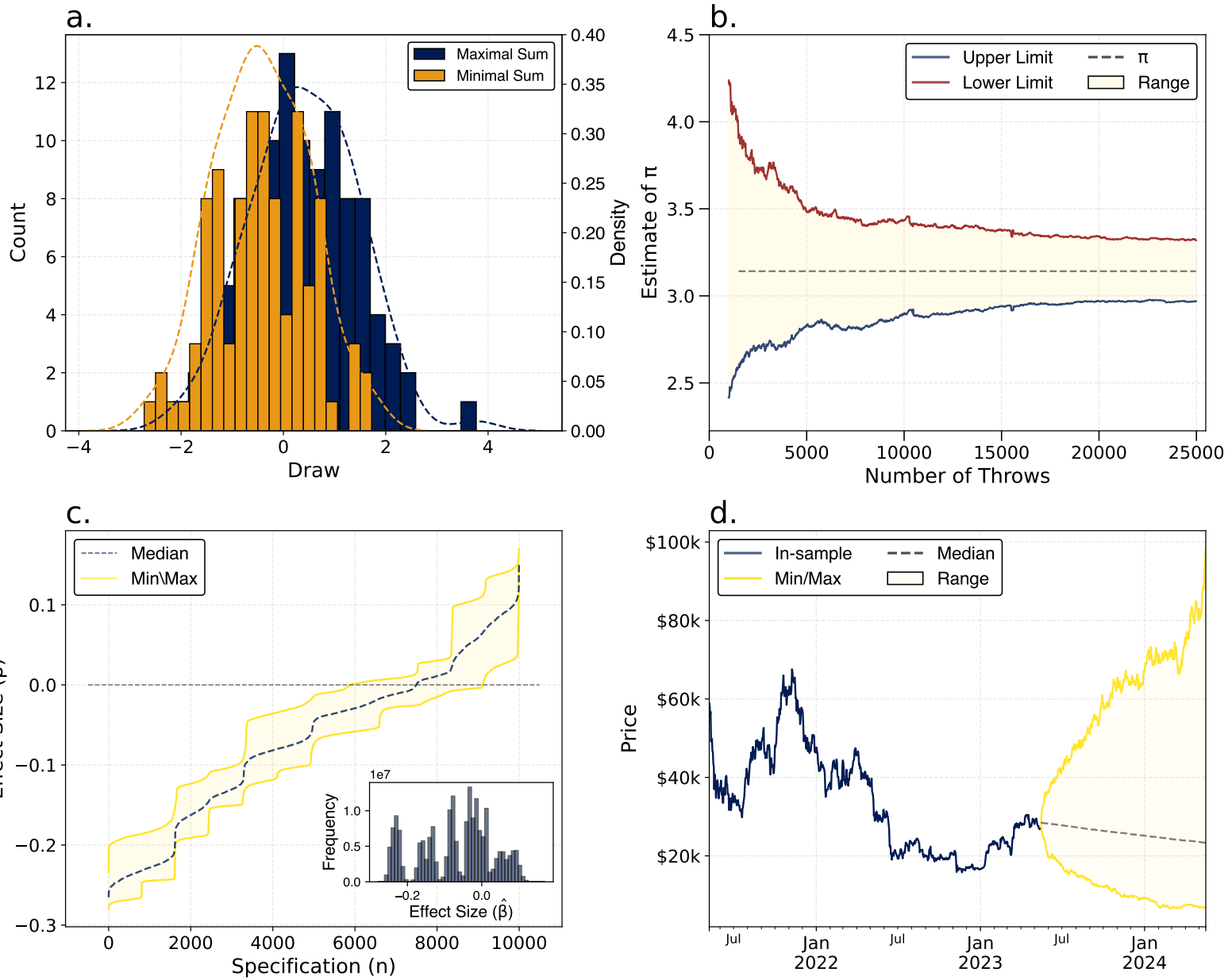
- **Problem Statement**: By setting **one** seed, we ignore the variation of our estimand as a function of how PRNGs were generated: computationally un-intensive, but scientifically dour.

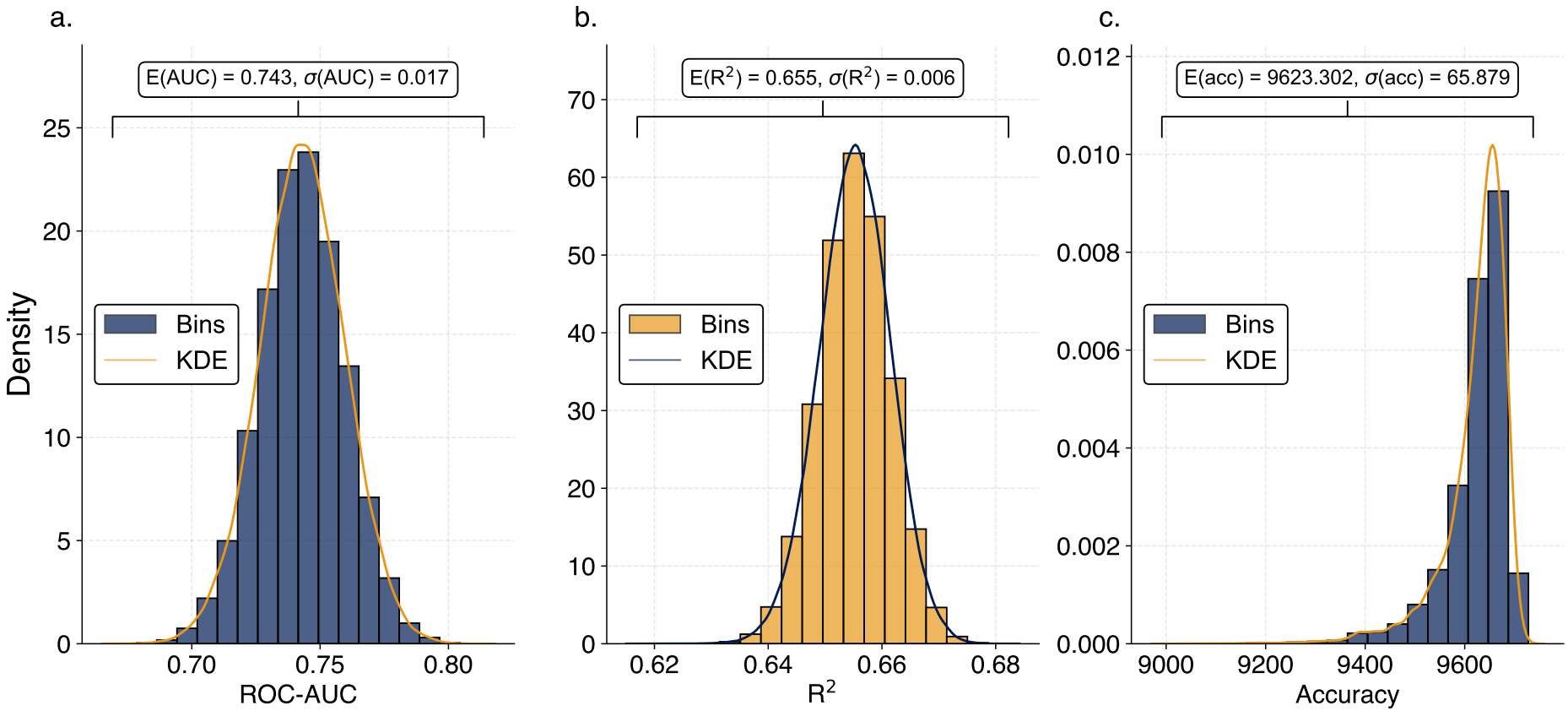
The General Premise

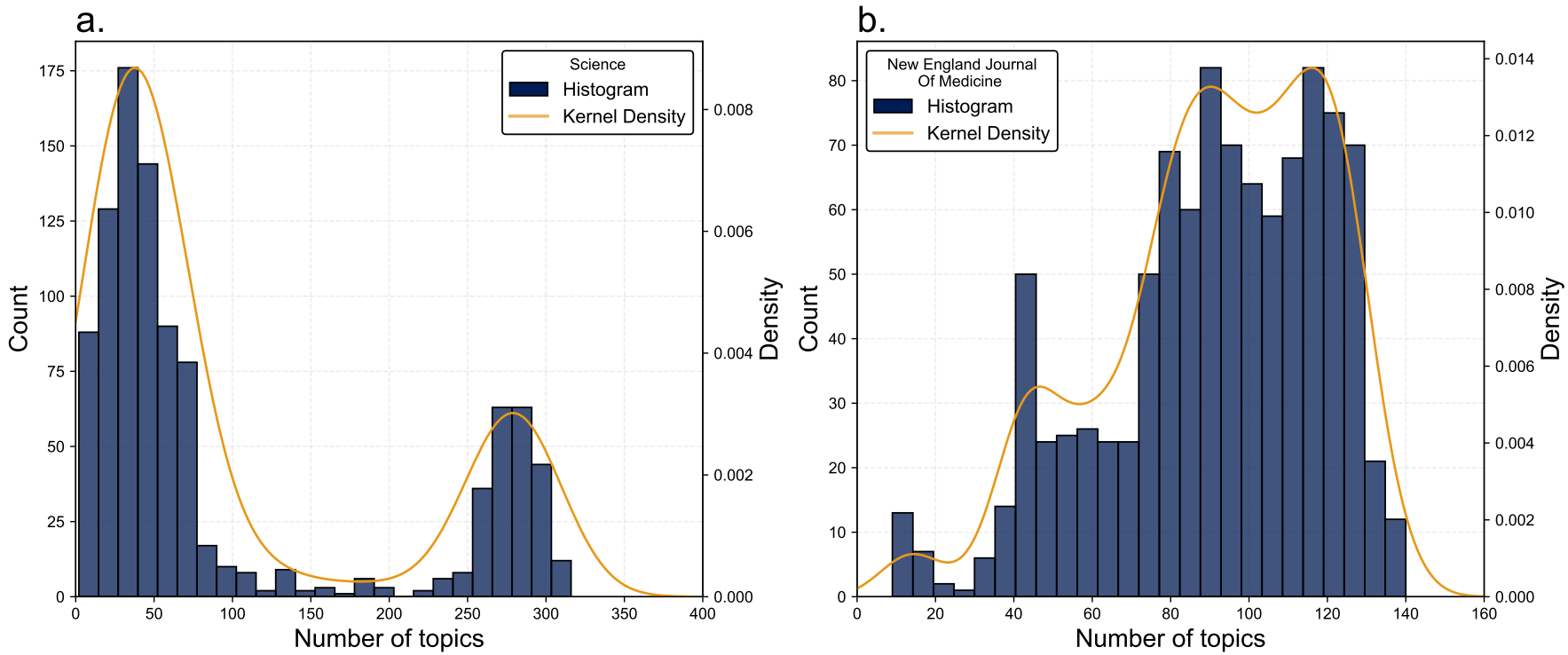
- **Problem Statement**: By setting **one** seed, we ignore the variation of our estimand as a function of how PRNGs were generated: computationally un-intensive, but scientifically dour.
- **Solution**: Visualize the outcome space of a **large number** (10k? 100k?) of seeds simultaneously. This is computationally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

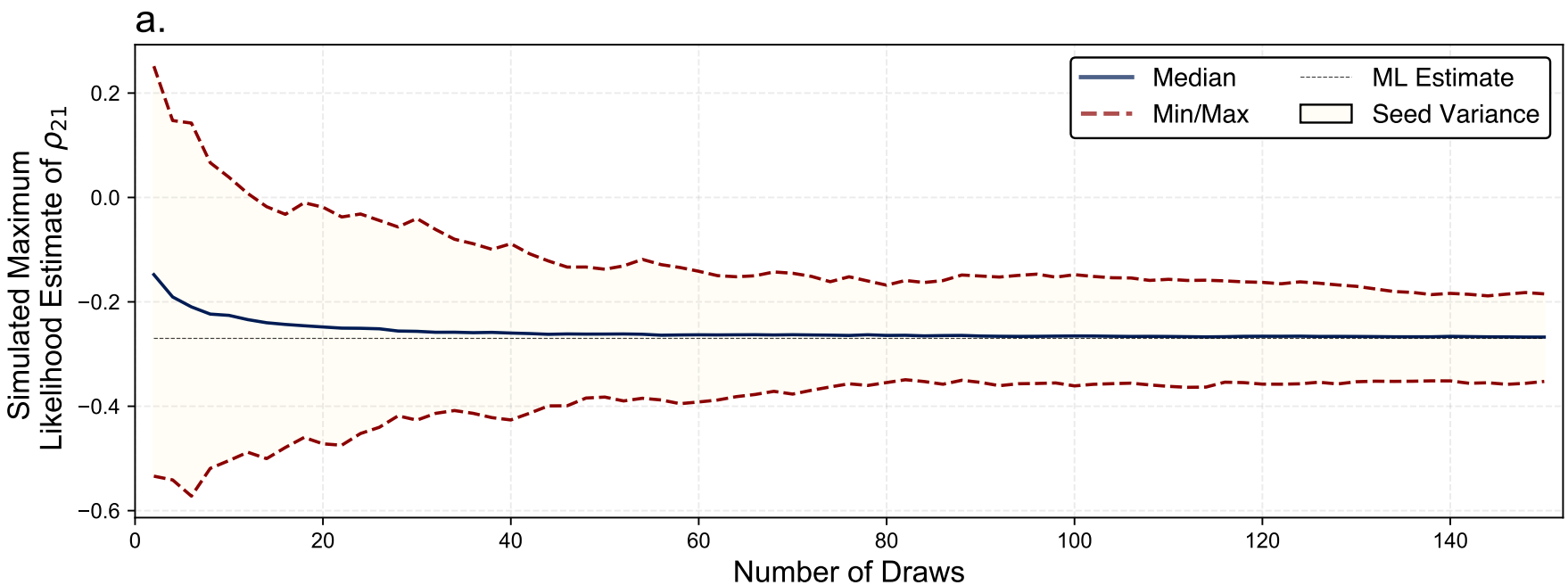
The General Premise

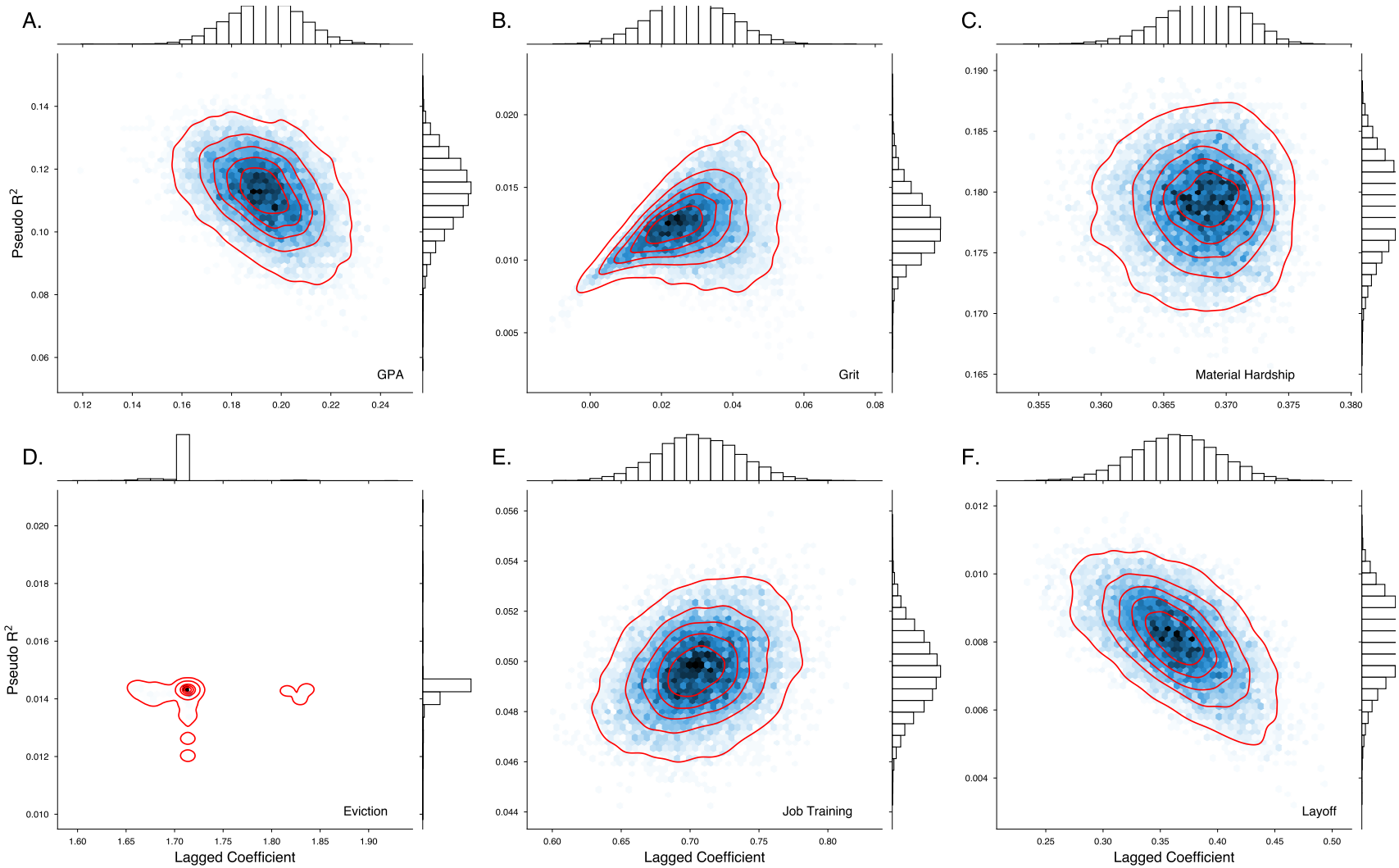
- **Problem Statement**: By setting **one** seed, we ignore the variation of our estimand as a function of how PRNGs were generated: computationally un-intensive, but scientifically dour.
- **Solution**: Visualize the outcome space of a **large number** (10k? 100k?) of seeds simultaneously. This is computationally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


a.
Maximal Sum
Minimal Sum
Count
Density
Draw
b.
Upper Limit
Lower Limit
π
Range
Estimate of π
Number of Throws
c.
Median
Min\Max
Effect Size (β̂)
Specification (n)
Frequency
1e7
Effect Size (β̂)
d.
In-sample
Min/Max
Median
Range
Price
Jul
Jan
2022
Jul
Jan
2023
Jul
Jan
2024


a.
E(AUC) = 0.743, σ(AUC) = 0.017
Bins
KDE
Density
ROC-AUC
b.
E(R²) = 0.655, σ(R²) = 0.006
Bins
KDE
R²
c.
E(acc) = 9623.302, σ(acc) = 65.879
Bins
KDE
Accuracy


a.
Science
Histogram
Kernel Density
Count
Density
Number of topics
b.
New England Journal
Of Medicine
Histogram
Kernel Density
Count
Density
Number of topics


a.
Median
ML Estimate
Min/Max
Seed Variance
Simulated Maximum Likelihood Estimate of ρ_{21}
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100
120
140
Number of Draws

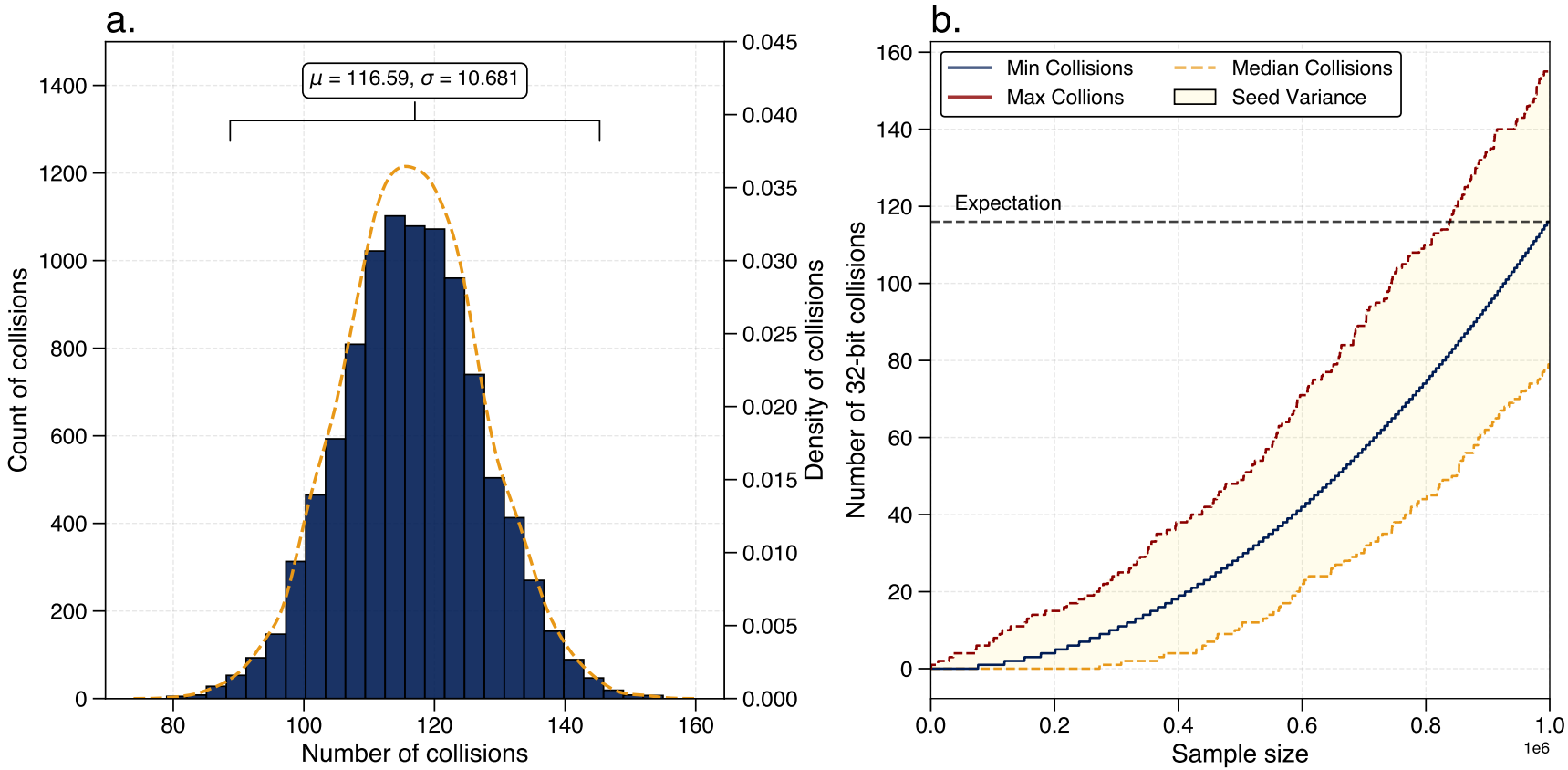

A.
B.
C.
D.
E.
F.
Pseudo R²
GPA
Grit
Material Hardship
Eviction
Job Training
Layoff
Lagged Coefficient

To make things more complicated...

- Hofert (2020): in *some* scenarios, **collisions** exist.
- Collisions: generation of duplicates b/c of insufficient complexity.
- An author of a leading LCDS publication recently stated to me:

  "We wanted to run it [the model] with a 100k seeds, but in reality, it was only 99,905 or something, because we were seeing that our results weren't unique when they should have been."

- This *only* occurs in 32-bit Mersenne Twister implementations.
  - With 32-bit MT: E(collisions) is 116.
  - With 64-bit MT: E(collisions is 2.71e-08.
- The default implementation of MT in `R` is 32-bit. . ..


a.
μ = 116.59, σ = 10.681
Count of collisions
Density of collisions
Number of collisions
b.
Min Collisions
Max Collions
Median Collisions
Seed Variance
Expectation
Number of 32-bit collisions
Sample size
1e6

Concluding Thoughts

- Matt Salganik commented: 'That sounds great, but expensive!'
  - Expense no-longer prohibitive: everything here ran locally.
- 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.