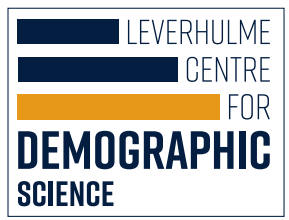
RobustiPy: The next generation of model uncertainty software

Daniel Valdenegro, Jiani Yan, Duiyi Dai, and **Charlie Rahal**

A Talk at the BDI Software Engineers Group

May 26, 2026



RESEARCH ARTICLE | SOCIAL SCIENCES |

Observing many researchers using the same data and hypothesis reveals a hidden universe of uncertainty

Nate Breznau, Eike Mark Rinke, Alexander Wuttke, +162, and Tomasz Żółtak Authors Info & Affiliations

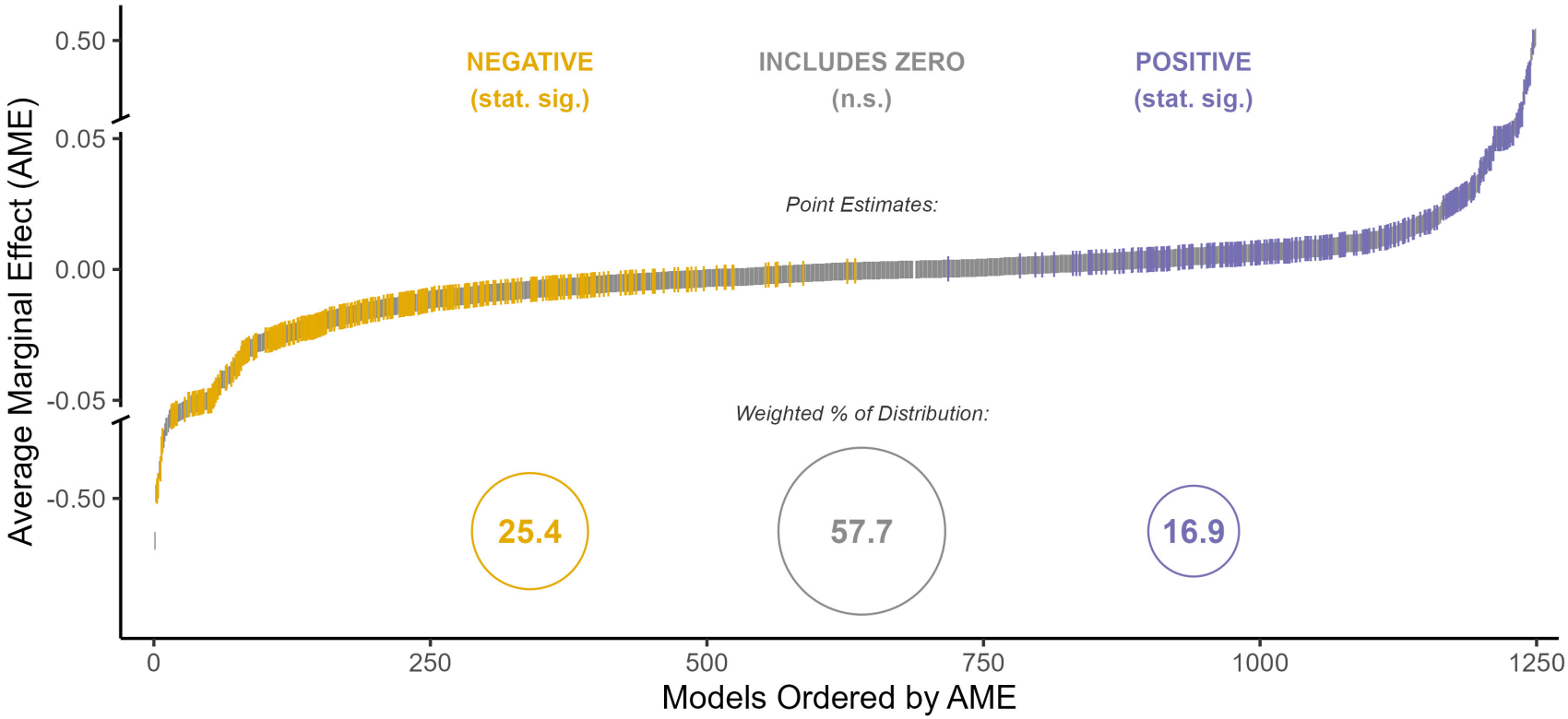
Edited by Douglas Massey, Princeton University, Princeton, NJ; received March 6, 2022; accepted August 22, 2022

October 28, 2022 | 119 (44) e2203150119 | https://doi.org/10.1073/pnas.2203150119

THIS ARTICLE HAS BEEN UPDATED

THIS ARTICLE HAS BEEN CORRECTED + VIEW RELATED CONTENT +

65,932 | 58 PDF/EPUB


NEGATIVE
(stat. sig.)
INCLUDES ZERO
(n.s.)
POSITIVE
(stat. sig.)
Point Estimates:
Weighted % of Distribution:
25.4
57.7
16.9
Average Marginal Effect (AME)
0.50
0.05
0.00
-0.05
-0.50
0
250
500
750
1000
1250
Models Ordered by AME

Types of Model Uncertainty

- **Parameter Uncertainty**:
  - Variability due to estimation from limited data.

Types of Model Uncertainty

- **Parameter Uncertainty**:
  - Variability due to estimation from limited data.
- **Structural Uncertainty**:
  - Inadequate model to capture system complexities fully.
  - Simplifications that may oversimplify or miss critical aspects.

Types of Model Uncertainty

- **Parameter Uncertainty**:
  - Variability due to estimation from limited data.
- **Structural Uncertainty**:
  - Inadequate model to capture system complexities fully.
  - Simplifications that may oversimplify or miss critical aspects.
- **Data Uncertainty**:
  - Errors in data collection or measurement.
  - Outliers affecting model stability or accuracy.

Types of Model Uncertainty

- **Parameter Uncertainty**:
  - Variability due to estimation from limited data.
- **Structural Uncertainty**:
  - Inadequate model to capture system complexities fully.
  - Simplifications that may oversimplify or miss critical aspects.
- **Data Uncertainty**:
  - Errors in data collection or measurement.
  - Outliers affecting model stability or accuracy.
- **Model Formulation Uncertainty**:
  - Choice of mathematical model structure.
  - Assumptions about relationships between variables.

Types of Model Uncertainty

- **Parameter Uncertainty**:
  - Variability due to estimation from limited data.
- **Structural Uncertainty**:
  - Inadequate model to capture system complexities fully.
  - Simplifications that may oversimplify or miss critical aspects.
- **Data Uncertainty**:
  - Errors in data collection or measurement.
  - Outliers affecting model stability or accuracy.
- **Model Formulation Uncertainty**:
  - Choice of mathematical model structure.
  - Assumptions about relationships between variables.

Some types of uncertainty can be handled by researchers, some can't.

Let's consider a simple example

- Imagine we want to know the effect of one variable x_1, on an output variable y.
- Example: x_1 is the effect of inequality on y, state-level crime rate.
- That is to say: our 'estimand' is β_1 in the following regression:

$$y_i = \alpha + \beta_1 x_{i,1} + \varepsilon_i \tag{1}$$

OLS Regression Results

Dep. Variable:	R	R-squared:	0.032
Model:	OLS	Adj. R-squared:	0.011
Method:	Least Squares	F-statistic:	2.010
Date:	Sun, 12 Oct 2025	Prob (F-statistic):	0.163
Time:	05:04:45	Log-Likelihood:	-237.21
No. Observations:	47	AIC:	478.4
Df Residuals:	45	BIC:	482.1
Df Model:	1		
Covariance Type:	HC1		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	124.1773	26.789	4.635	0.000	71.672	176.682
Inequality	-0.1736	0.122	-1.418	0.156	-0.413	0.066

Omnibus:	8.646	Durbin-Watson:	2.461
Prob(Omnibus):	0.013	Jarque-Bera (JB):	7.831
Skew:	0.948	Prob(JB):	0.0199
Kurtosis:	3.636	Cond. No.	993.

But don’t we also need to correct for things like age?

OLS Regression Results

Dep. Variable:	R	R-squared:	0.033
Model:	OLS	Adj. R-squared:	-0.011
Method:	Least Squares	F-statistic:	0.9856
Date:	Sun, 12 Oct 2025	Prob (F-statistic):	0.381
Time:	05:05:32	Log-Likelihood:	-237.19
No. Observations:	47	AIC:	480.4
Df Residuals:	44	BIC:	485.9
Df Model:	2		
Covariance Type:	HC1		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	111.2518	46.834	2.375	0.018	19.459	203.044
Inequality	-0.1997	0.172	-1.160	0.246	-0.537	0.138
Age	0.1299	0.449	0.289	0.773	-0.751	1.011

Omnibus:	8.799	Durbin-Watson:	2.457
Prob(Omnibus):	0.012	Jarque-Bera (JB):	8.002
Skew:	0.956	Prob(JB):	0.0183
Kurtosis:	3.655	Cond. No.	2.79e+03

What about if we build out a fuller model, in order to help this estimation meet assumptions for the Gauss-Markov theorem?

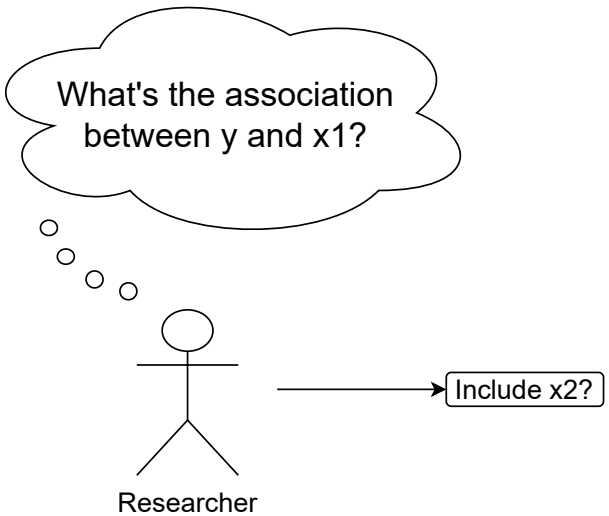
(Are there any social theorists here? Do we expect inequality to be positively or negatively associated with crime?)

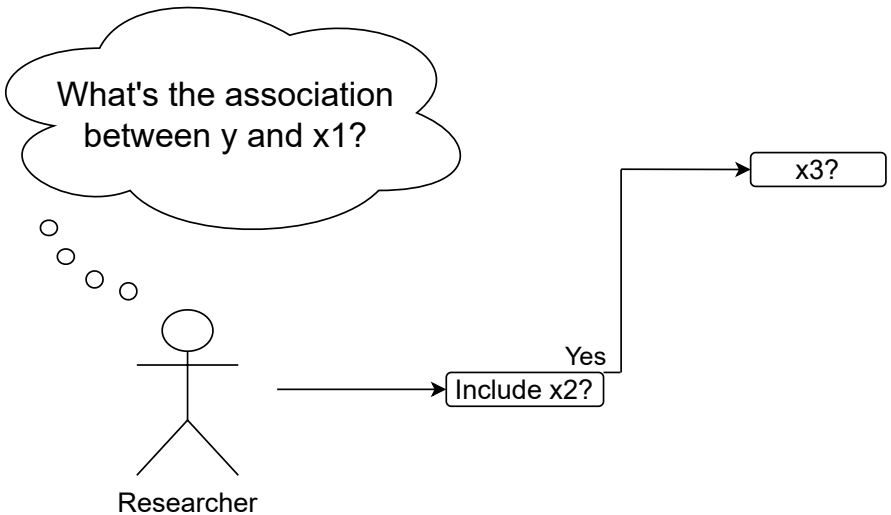
OLS Regression Results

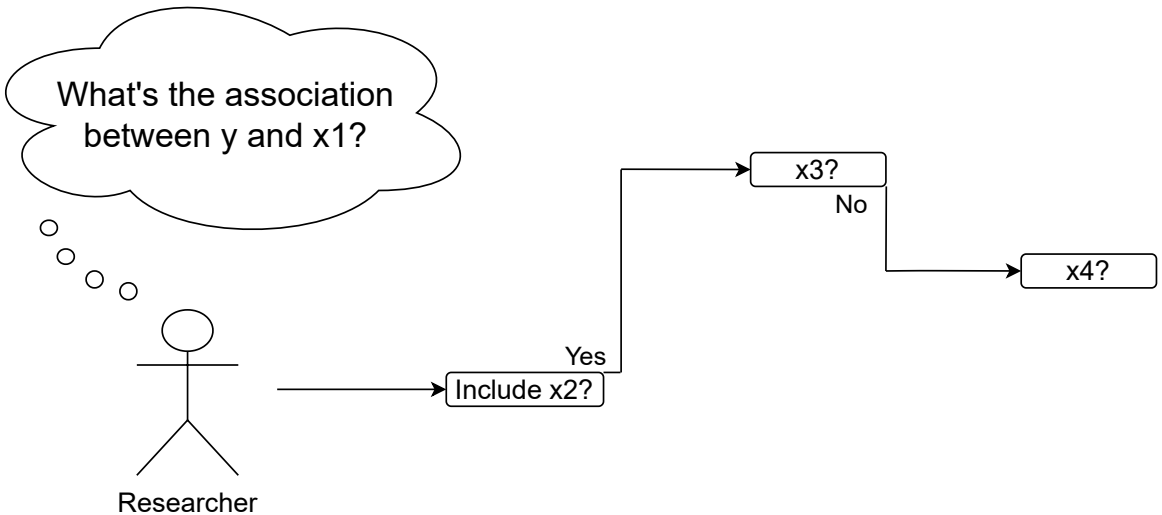
Dep. Variable:	R	R-squared:	0.522
Model:	OLS	Adj. R-squared:	0.436
Method:	Least Squares	F-statistic:	7.276
Date:	Sun, 12 Oct 2025	Prob (F-statistic):	1.39e-05
Time:	05:06:00	Log-Likelihood:	-220.64
No. Observations:	47	AIC:	457.3
Df Residuals:	39	BIC:	472.1
Df Model:	7		
Covariance Type:	HC1		

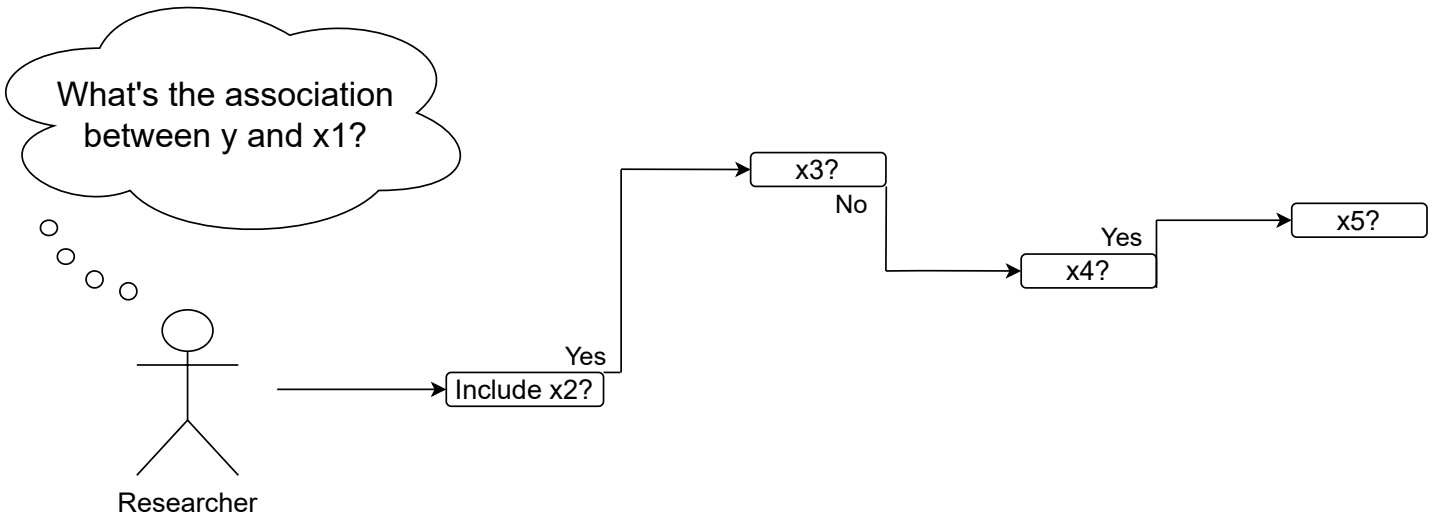
	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-764.2379	223.962	-3.412	0.001	-1203.196	-325.280
Inequality	0.8445	0.264	3.196	0.001	0.327	1.363
Age	0.9750	0.578	1.688	0.091	-0.157	2.107
Ed	0.9510	0.793	1.199	0.231	-0.604	2.506
Unemployment	-0.0593	0.302	-0.196	0.844	-0.652	0.533
Males	0.2211	0.241	0.918	0.359	-0.251	0.693
N	0.2714	0.117	2.326	0.020	0.043	0.500
Wealth	0.4449	0.125	3.559	0.000	0.200	0.690

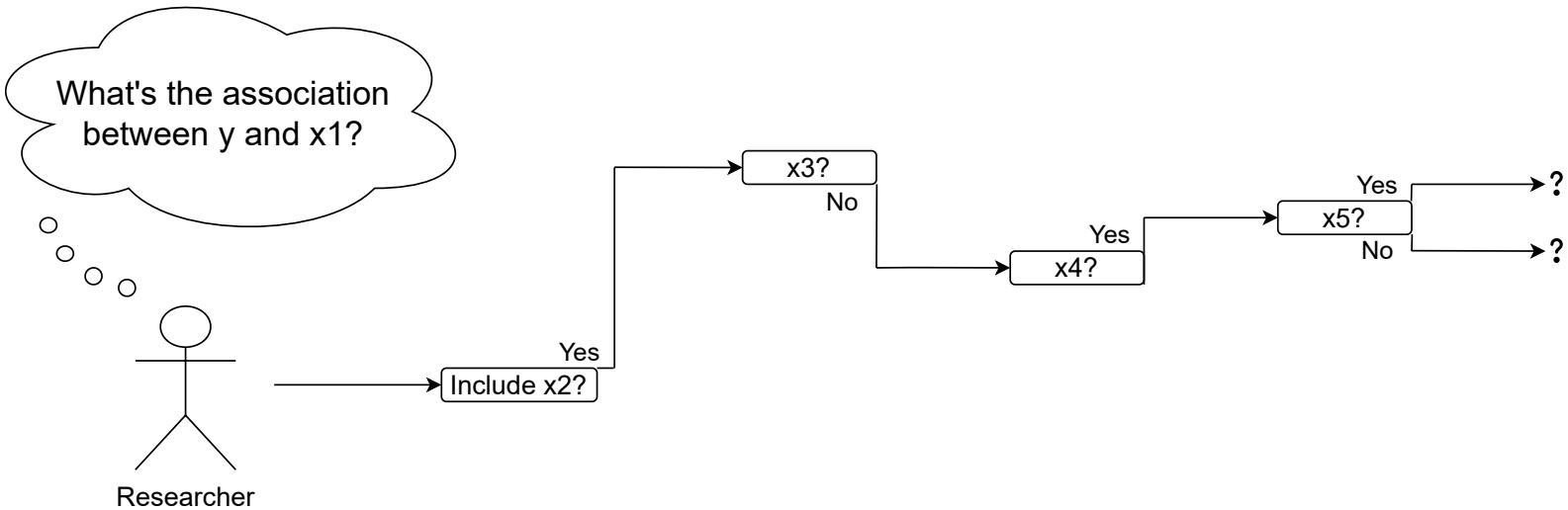
Omnibus:	11.181	Durbin-Watson:	1.794
Prob(Omnibus):	0.004	Jarque-Bera (JB):	10.897
Skew:	1.061	Prob(JB):	0.00430
Kurtosis:	4.031	Cond. No.	5.08e+04

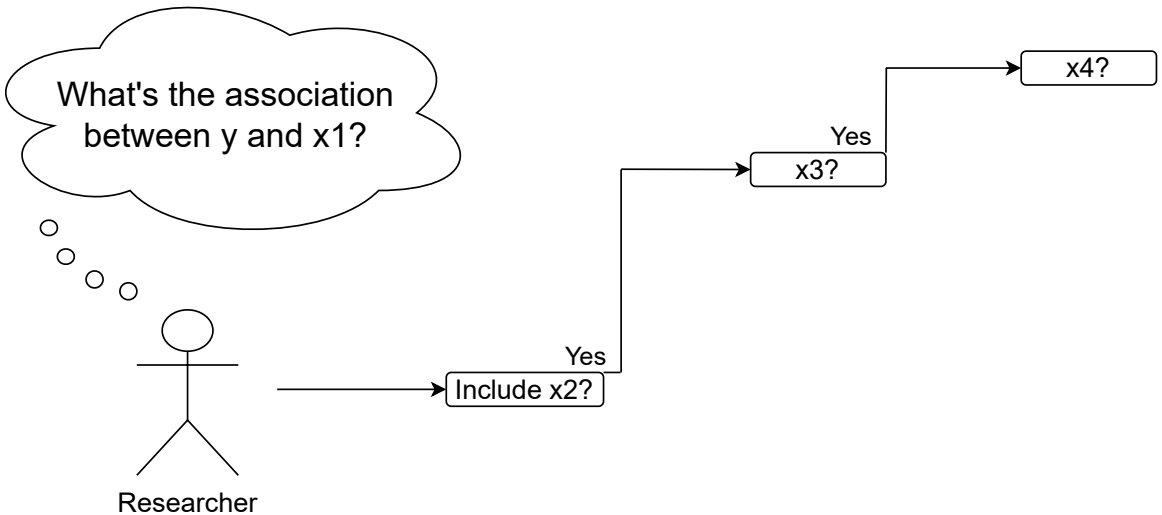

What's the association between y and x1?
Include x2?
Researcher

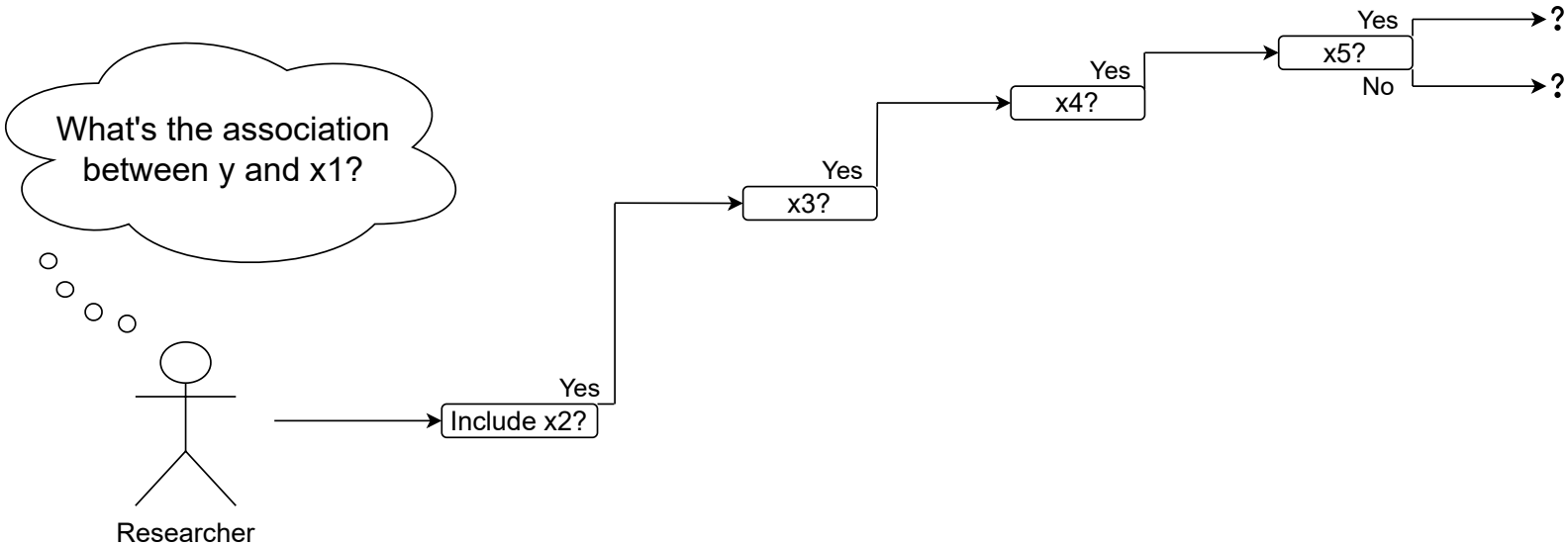

What's the association between y and x1?
x3?
Yes
Include x2?
Researcher

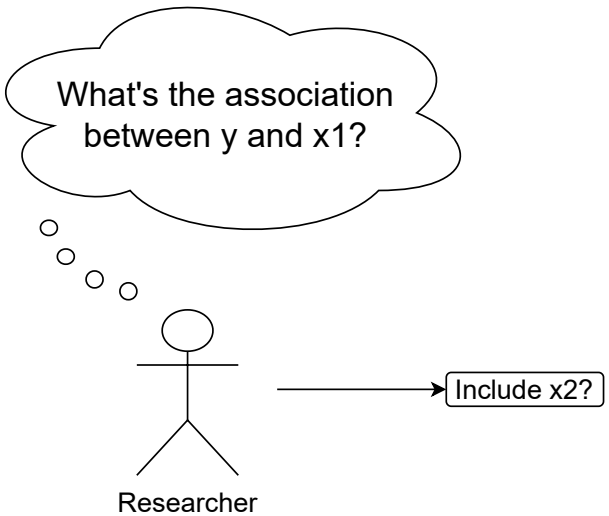

What's the association between y and x1?
Researcher
Include x2?
Yes
x3?
No
x4?

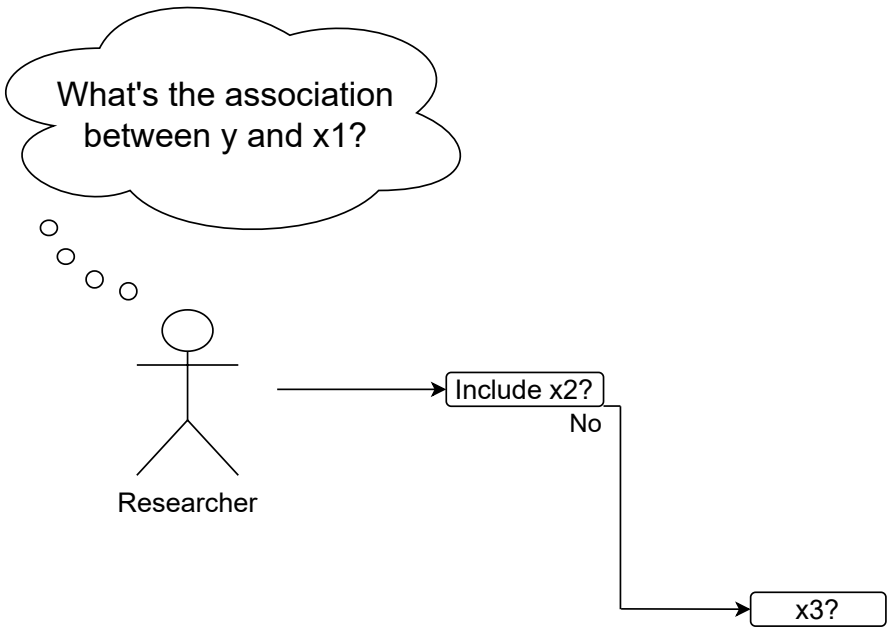

What's the association between y and x1?
Researcher
Include x2?
Yes
x3?
No
x4?
Yes
x5?

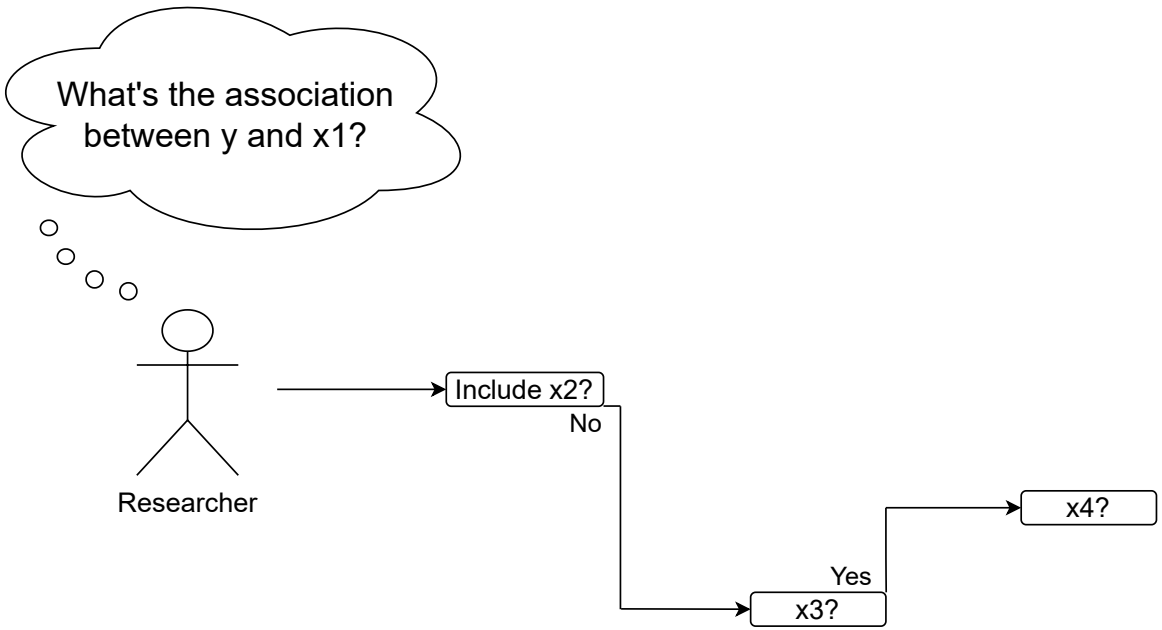

What's the association between y and x1?
Researcher
Include x2?
Yes
x3?
No
x4?
Yes
x5?
Yes
No
?
?

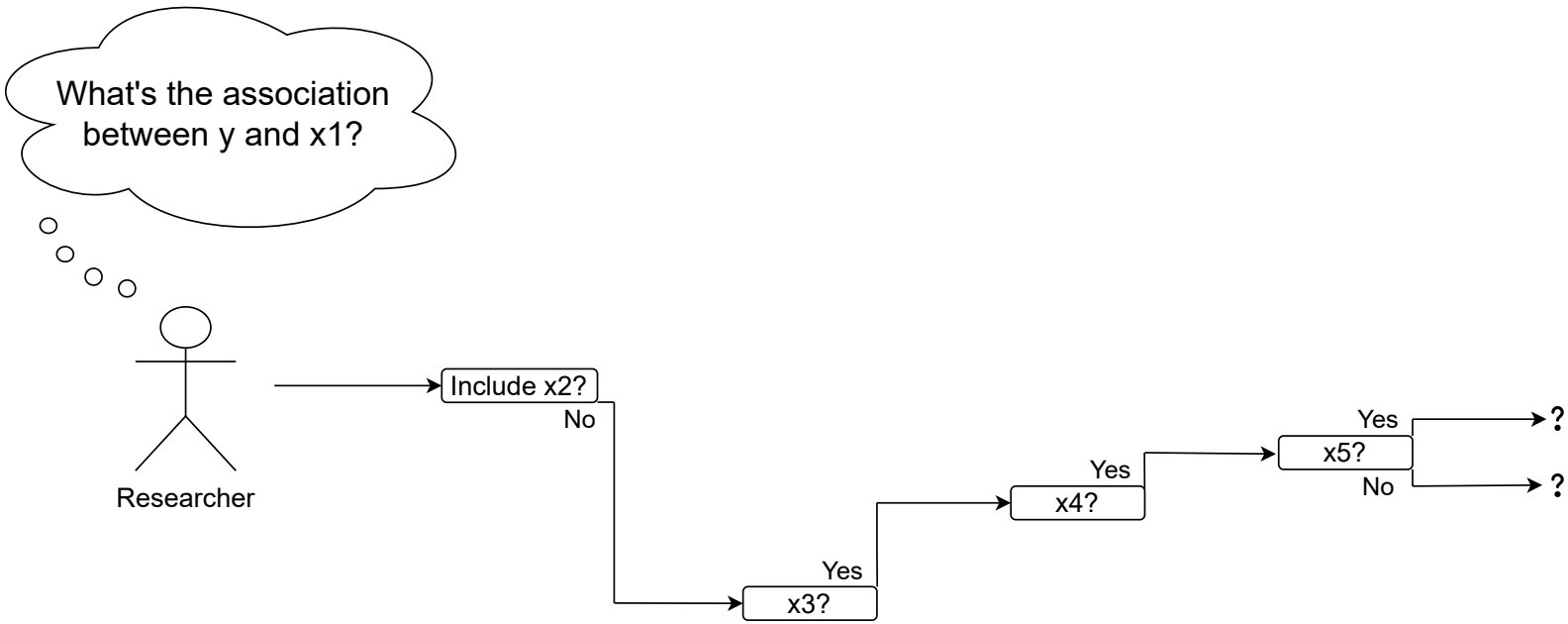

What's the association between y and x1?
Researcher
Include x2?
Yes
x3?
Yes
x4?

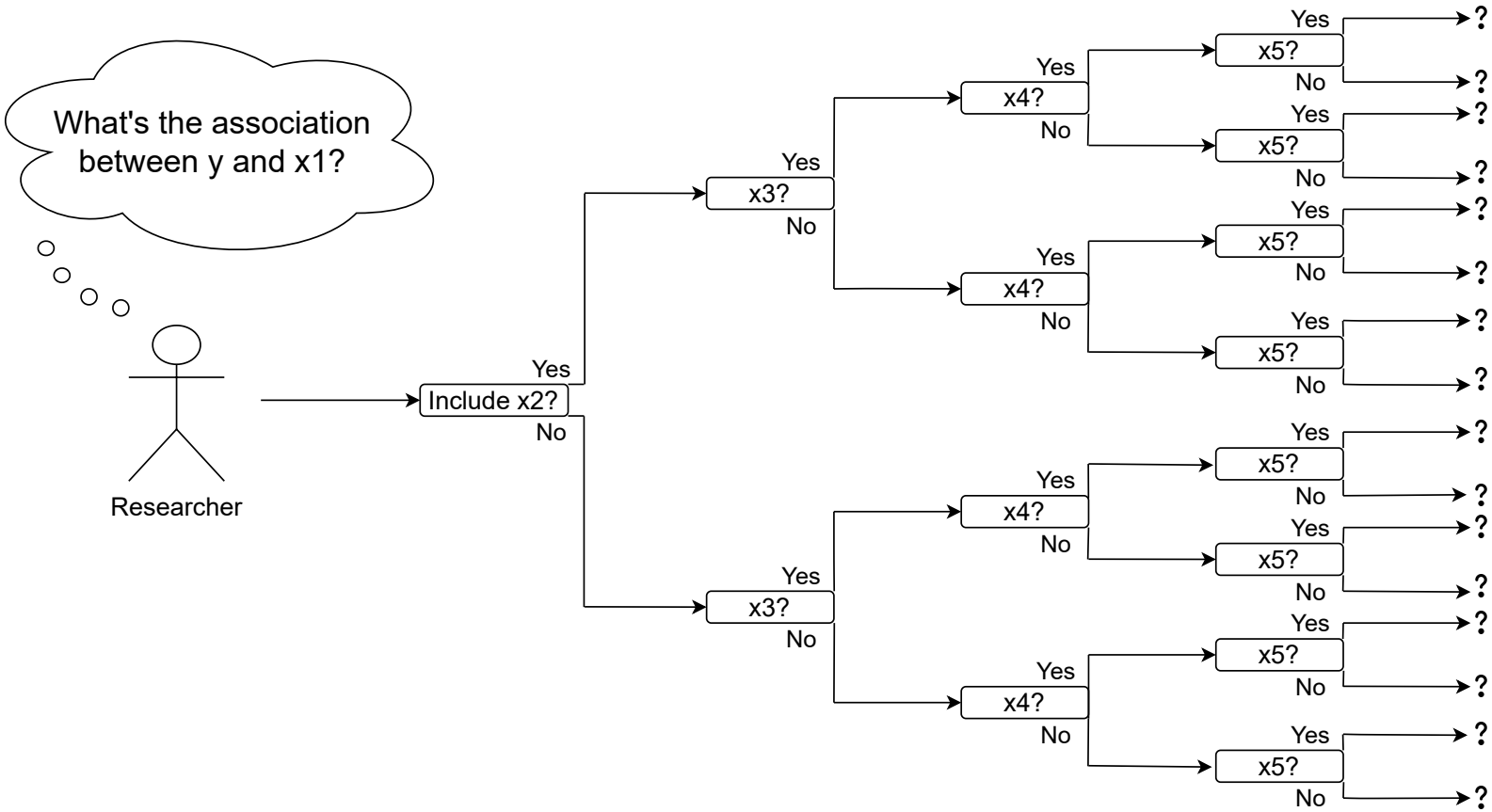

What's the association between y and x1?
Researcher
Include x2?
Yes
x3?
Yes
x4?
Yes
x5?
Yes
No
?
?


What's the association between y and x1?
Include x2?
Researcher


What's the association between y and x1?
Researcher
Include x2?
No
x3?


What's the association between y and x1?
Researcher
Include x2?
No
x4?
Yes
x3?


What's the association between y and x1?
Researcher
Include x2?
No
Yes
x3?
Yes
x4?
Yes
x5?
No
?
?


What's the association between y and x1?
Researcher
Include x2?
Yes
No
x3?
x4?
x5?
?

This type of problem has enormous implications for p-hacking, HARKing, and the validity of the scientific record.

Introducing the ‘Feasible Model Space’

- From this simple choice of four auxiliary control variables, we have 16 possible models.

Introducing the 'Feasible Model Space'

- From this simple choice of four auxiliary control variables, we have 16 possible models.
  - In general, this is 2^{n+1}.

Introducing the 'Feasible Model Space'

- From this simple choice of four auxiliary control variables, we have 16 possible models.
  - In general, this is 2^{n+1}.
  - The '+1' comes from models with or without a constant.

Introducing the 'Feasible Model Space'

- From this simple choice of four auxiliary control variables, we have 16 possible models.
  - In general, this is 2^{n+1}.
  - The '+1' comes from models with or without a constant.
  - With a choice of eight possible controls: $2^9 = 512$ models.

Introducing the 'Feasible Model Space'

- From this simple choice of four auxiliary control variables, we have 16 possible models.
  - In general, this is 2^{n+1}.
  - The '+1' comes from models with or without a constant.
  - With a choice of eight possible controls: $2^9 = 512$ models.
  - This can quickly get large.

Introducing the 'Feasible Model Space'

- From this simple choice of four auxiliary control variables, we have 16 possible models.
  - In general, this is 2^{n+1}.
  - The '+1' comes from models with or without a constant.
  - With a choice of eight possible controls: $2^9 = 512$ models.
  - This can quickly get large.
  - With 20 possible controls: $2^{21} = 2,097,152$ models.

Introducing the 'Feasible Model Space'

- From this simple choice of four auxiliary control variables, we have 16 possible models.
  - In general, this is 2^{n+1}.
  - The '+1' comes from models with or without a constant.
  - With a choice of eight possible controls: $2^9 = 512$ models.
  - This can quickly get large.
  - With 20 possible controls: $2^{21} = 2,097,152$ models.
- In an age of high performance computing: no problem!

Introducing the 'Feasible Model Space'

- From this simple choice of four auxiliary control variables, we have 16 possible models.
  - In general, this is 2^{n+1}.
  - The '+1' comes from models with or without a constant.
  - With a choice of eight possible controls: $2^9 = 512$ models.
  - This can quickly get large.
  - With 20 possible controls: $2^{21} = 2,097,152$ models.
- In an age of high performance computing: no problem!
  - Do, however, spare a thought for the carbon cost... 🌳

Introducing the 'Feasible Model Space'

- From this simple choice of four auxiliary control variables, we have 16 possible models.
  - In general, this is 2^{n+1}.
  - The '+1' comes from models with or without a constant.
  - With a choice of eight possible controls: $2^9 = 512$ models.
  - This can quickly get large.
  - With 20 possible controls: $2^{21} = 2,097,152$ models.
- In an age of high performance computing: no problem!
  - Do, however, spare a thought for the carbon cost... 🌳
- The main challenge: systematically reporting all these results.

Introducing the ‘Feasible Model Space’

- What’s critical here is that we consider only a ‘feasible model space’.

Introducing the 'Feasible Model Space'

- What's critical here is that we consider only a 'feasible model space'.
- We should **not** be including in this space noisy variables.

Introducing the 'Feasible Model Space'

- What's critical here is that we consider only a 'feasible model space'.
- We should **not** be including in this space noisy variables.
- This is where the role of 'Theory' becomes critical.

Introducing the 'Feasible Model Space'

- What's critical here is that we consider only a 'feasible model space'.
- We should **not** be including in this space noisy variables.
- This is where the role of 'Theory' becomes critical.
- Systematic inclusion/omission allows holistic tests of theory.

Introducing the 'Feasible Model Space'

- What's critical here is that we consider only a 'feasible model space'.
- We should **not** be including in this space noisy variables.
- This is where the role of 'Theory' becomes critical.
- Systematic inclusion/omission allows holistic tests of theory.
- This does not prohibit us from having *one* preferred model.

Introducing the 'Feasible Model Space'

- What's critical here is that we consider only a 'feasible model space'.
- We should **not** be including in this space noisy variables.
- This is where the role of 'Theory' becomes critical.
- Systematic inclusion/omission allows holistic tests of theory.
- This does not prohibit us from having *one* preferred model.
  - This could be the one that is classically built in isolation.

Introducing the 'Feasible Model Space'

- What's critical here is that we consider only a 'feasible model space'.
- We should **not** be including in this space noisy variables.
- This is where the role of 'Theory' becomes critical.
- Systematic inclusion/omission allows holistic tests of theory.
- This does not prohibit us from having *one* preferred model.
  - This could be the one that is classically built in isolation.
  - But, now we can compare it with **all** others.

nature human behaviour

View all journals Search Log in

Explore content About the journal Publish with us

nature > nature human behaviour > resources > article

Resource | Published: 27 July 2020

Specification curve analysis

Uri Simonsohn, Joseph P. Simmons & Leif D. Nelson

Nature Human Behaviour **4**, 1208–1214 (2020) | Cite this article

8225 Accesses | **237** Citations | **68** Altmetric | Metrics

A Publisher Correction to this article was published on 09 October 2020

This article has been updated

Sociological Methods & Research

Impact Factor: **6.5** / 5-Year Impact Factor: **7.1**

Available access | Research article | First published online October 23, 2015 | Request permissions

Model Uncertainty and Robustness: A Computational Framework for Multimodel Analysis

Cristobal Young and Katherine Holsteen View all authors and affiliations

Volume 46, Issue 1 | https://doi.org/10.1177/0049124115610347

Abstract

Model uncertainty is pervasive in social science. A key question is how robust empirical results are to sensible changes in model specification. We present a new approach and applied statistical software for computational multimodel analysis. Our approach proceeds in two steps: First, we estimate the modeling distribution of estimates across all combinations of possible controls as well as specified functional form issues, variable definitions, standard error calculations, and estimation commands. This allows analysts to present their core, preferred estimate in the context of a distribution of plausible estimates. Second, we develop a model influence analysis showing how each model ingredient affects the coefficient of interest. This shows which model assumptions, if any, are critical to obtaining an empirical result. We demonstrate the architecture and interpretation of multimodel analysis using data on the union wage premium, gender dynamics in mortgage lending, and tax flight migration among U.S. states. These illustrate how initial results can be strongly robust to alternative model specifications or remarkably dependent on a knife-edge specification.


WHAT PEOPLE THINK
YOU NEED TO DO
MULTIVERSE ANALYSIS:
FOR LOOP
UNTIL
CONVERGENCE
432
UNIVERSES
NO BUGS
STILL
RUNNING
HELP.
EAT
SLEEP
CODE
REPEAT
Advanced
C++
3rd Edition
REQUIRED:
• 10 YEARS CODING EXPERIENCE
• 3 MONITORS
• 7 ENERGY DRINKS
• WILLINGNESS TO SACRIFICE SLEEP, SOCIAL LIFE, AND SANITY
WHAT YOU
ACTUALLY NEED:
I HAVE
QUESTIONS.
THEY HAVE
TOOLS.
IT'S A
MULTIVERSE.
LET'S GO!
POWERFUL TOOLS.
NO CODING
PH.D. REQUIRED.
JUST CURIOSITY
(AND MAYBE COFFEE)
RobustiPy
spec-f
mrobust
MULTIVERSE ANALYSIS?
EASY. ♡
DATA
>
DRAMA
TODAY'S PLAN:
ASK QUESTION
RUN ROBUSTNESS
CHANGE THE WORLD
YOU DON'T NEED TO CODE THE UNIVERSE.
JUST UNDERSTAND IT.

However, these existing tools are limited

- They don't include any real indication of model fit.

However, these existing tools are limited

- They don't include any real indication of model fit.
- They simply visualise distributions of the estimand.

However, these existing tools are limited

- They don't include any real indication of model fit.
- They simply visualise distributions of the estimand.
- There is no out-of-sample validity.

However, these existing tools are limited

- They don't include any real indication of model fit.
- They simply visualise distributions of the estimand.
- There is no out-of-sample validity.
- There's no way to identify individual models easily.

However, these existing tools are limited

- They don't include any real indication of model fit.
- They simply visualise distributions of the estimand.
- There is no out-of-sample validity.
- There's no way to identify individual models easily.
- They place a uniform likelihood over all models.

However, these existing tools are limited

- They don't include any real indication of model fit.
- They simply visualise distributions of the estimand.
- There is no out-of-sample validity.
- There's no way to identify individual models easily.
- They place a uniform likelihood over all models.
- Some of them are actually quite difficult to use. . .

We set ourselves the task:

We set ourselves the task:

Can we create a reliable tool to conduct this/adjacent analysis?

Toolkit Comparison

Feature	`multiverse` (R)	`specr` (R)	`MULTIVRS` (Stata)	`spec_curve` (Py)	`Boba` (Py DSL)	`RobustiPy` (Py)
Multiverse analysis	✓	–	✓	–	✓	✓[L]
Specification curve analysis	–	✓	–	✓	–	✓
High-abstraction syntax	✓	✓[P]	—	–	✓	✓
Visualisation of results	✓	✓[I]	✓[K]	✓[B]	✓	✓[C]
Joint inference	–	✓	–	✓	✓	✓
Influence analysis	–	–	✓	–	✓	✓
Bootstrapping	–	✓	–	✓	✓[P]	✓
Out-of-sample	–	–	–	–	–	✓
Multiple estimands	–	✓[L]	–	✓[P]	–	✓

P = partial; L = limited; I = built-in; K = kdensity; B = basic; C = custom.

RobustiPy

ROBUST INFERENCE ACROSS THE ANALYTICAL MULTIVERSE

1 Repository organization

- src/robustipy
  - models.py
  - figures.py
  - utils.py
  - prototypes.py
  - __init__.py
- empirical_examples
- simulated_examples
- tests
- docs

pip install robustipy

2 Specify the multiverse

Y	X	Z (controls)
	X_1 X_2 ⋯ X_{k_k}	Z_1 Z_2 ⋯ Z_{k_z}
y_1	● ● ⋯ ●	● ● ⋯ ●
y_2	● ● ⋯ ●	● ● ⋯ ●
⋮	⋮ ⋮ ⋯ ⋮	⋮ ⋮ ⋯ ⋮
y_m	● ● ⋯ ●	● ● ⋯ ●

$$\hat{y} = f(X, Z) + \varepsilon$$

Defensible operationalisations

Y = $\{y_1, y_2, \ldots, y_m\}$ | X = $\{x_1, x_2, \ldots, x_{k_x}\}$ | Z = $\{z_1, z_2, \ldots, z_{k_z}\}$

All subsets / powerset

$$\prod = \text{mean}\,(P(Y)\,\{0\}) \times P(X) \times P(Z)$$

Enumerates the full specification space

3 Fit robust models

```
from robustipy.models
import OLSRobust

model = OLSRobust(
  y=['y1', 'y2'],
  x=['x1', 'x2'],
  data=data
)
model.fit(
  controls=c,
  group='firm',
  draws=1000,
  kfold=10,
  seed=192735
)
results = model.get_results()
```

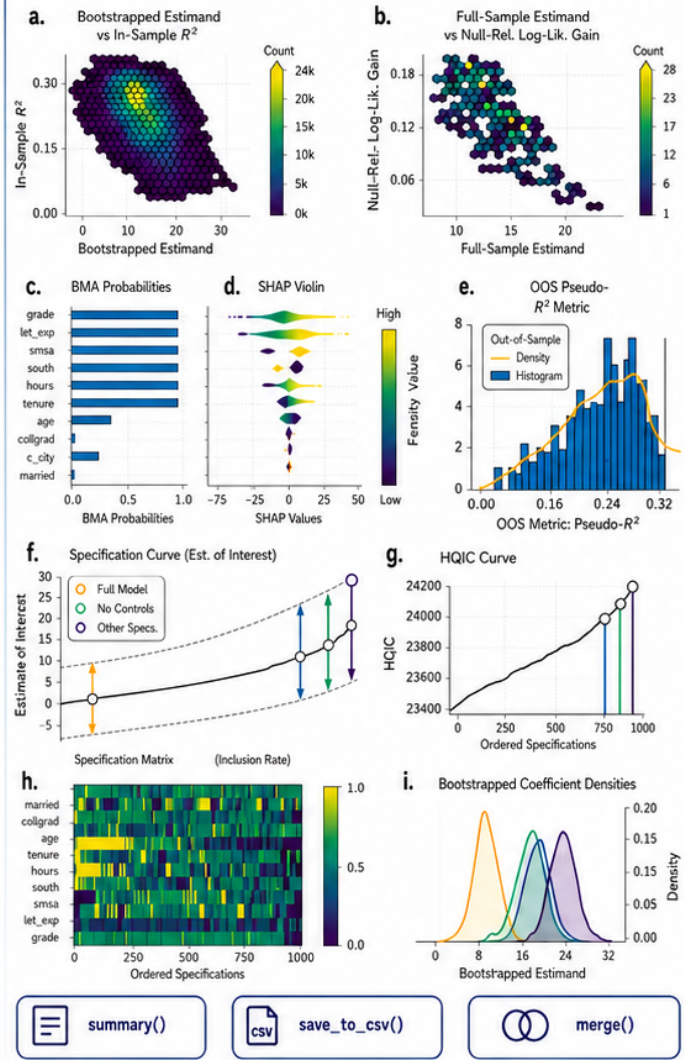
OLS | Logistic | Fixed effects

multi-y | multi-x | len(x) > 1

- Bootstrap resampling (draws)
- K-fold out-of-sample evaluation
- Information criteria (AIC, BIC, HQIC)
- Bayesian model averaging (BMA)
- SHAP / explainable AI
- Joint inference (Stouffer / Fisher)
- Parallel computation (joblib)
- Multiple dependent variables
- Multiple focal predictors

4 Inspect outputs

`results.plot()`



summary() | save_to_csv() | merge()

README Code of conduct Contributing GPL-3.0 license

RobustiPy

Purpose Research | Python 3.11 | License GNU3.0 | DOI 10.5281/zenodo.15700698

Welcome to the home of `RobustiPy`, a Python library for the creation of a more robust and stable model space. RobustiPy does a large number of things, included but not limited to: high dimensional visualisation, Bayesian Model Averaging, bootstrapped resampling, (in- and)out-of-sample model evaluation, model selection via Information Criterion, explainable AI (via SHAP), and joint inference tests (as per Simonsohn et al. 2019).

Full documentation is available on Read the Docs. The first release is indexed onto Zenodo here.

`RobustiPy` performs Multiversal/Specification Curve Analysis which attempts to compute most or all reasonable specifications of a statistical model, understanding a specification as a single attempt to create an estimand of interest, whether through a particular choice of covariates, hyperparameters, data cleaning decisions, and so forth.

RobustiPy: An efficient next-generation multiversal library with model selection, averaging, resampling, and explainable AI

Daniel Valdenegro Ibarra[1,3,5], Jiani Yan[1,2,3,4], Duiyi Dai[1,3], and Charles Rahal[1,3,5]

[1]Leverhulme Centre for Demographic Science, University of Oxford
[2]Department of Sociology, University of Oxford
[3]ESRC Centre for Care, University of Oxford
[4]Wolfson College, University of Oxford
[5]Nuffield College, University of Oxford

May 20, 2026

Abstract

Scientific inference is often undermined by the vast but rarely explored 'multiverse' of defensible modeling choices which can generate results as variable as the phenomena under study. We introduce `RobustiPy`, an open-source Python library that systematizes multiverse analysis and model-uncertainty quantification at scale. `RobustiPy` unifies bootstrap-based inference, combinatorial specification search, model selection and averaging, joint-inference routines, and explainable AI methods within a modular, reproducible framework. Beyond exhaustive specification curves, it supports rigorous out-of-sample validation and quantifies the marginal contribution of each covariate. We demonstrate its utility across five simulation designs and ten empirical and high-profile replications spanning economics, sociology, psychology, and medicine, including a re-analysis of widely cited findings with documented discrepancies. Benchmarking on ∼672 million simulated regressions shows that `RobustiPy` delivers state-of-the-art computational efficiency while expanding transparency in empirical research. By standardizing and accelerating methods for robustness, `RobustiPy` transforms how researchers interrogate sensitivity across the analytical multiverse, offering a practical foundation for more reproducible and interpretable computational science.

RobustiPy: The formal problem

Consider the key association between variables Y and X, where a set of covariates $\mathbf{Z}$ can influence the relationship between the former as follows:

$$Y = F(X, \mathbf{Z}) + \varepsilon. \tag{2}$$

Let's now observe that:

- Usually Y and X are imprecisely defined latent variables.
- The covariate set $\mathbf{Z}$ can also be composed of imprecisely defined variables; the number of them can be unknown and/or non-finite.
- Finally, $F()$ is an unknown data generating function.

RobustiPy: The formal problem

Let's define the set of *reasonable* operationalisations of Y, X, $\mathbf{Z}$ and $F()$ as $\overleftrightarrow{Y}$, $\overleftrightarrow{X}$, $\overleftrightarrow{\mathbf{Z}}$ and $\overleftrightarrow{F()}$. Then, we have:

$$\overleftrightarrow{Y}_{k_Y} = \overleftrightarrow{F}_{k_f}(\overleftrightarrow{X}_{k_X}, \overleftrightarrow{\mathbf{Z}}_{k_\mathbf{Z}}) + \varepsilon. \tag{3}$$

- Eq. 3 corresponds to a single possible choice π from the set of all possible choices Π, such that $\pi \in \Pi$.
- The total number of estimations is the number of distinct combinations obtained by selecting:
  1. One (composite) outcome operationalisation,
  2. One functional form (or estimator),
  3. One focal predictor specification,
  4. One subset of candidate controls.

RobustiPy: The formal problem

Let

$$m_F := |\Pi_{\underset{F()}{\leftrightarrow}}|, \qquad m_X := |\Pi_{\underset{\mathbf{X}}{\leftrightarrow}}|.$$

If there are d_Z candidate control variables and d_Y candidate components of a composite dependent variable, then

$$|\Pi_{\underset{\mathbf{Z}}{\leftrightarrow}}| = 2^{d_Z}, \qquad |\Pi_{\underset{\mathbf{Y}}{\leftrightarrow}}| = 2^{d_Y} - 1.$$

Hence:

$$|\Pi| = (2^{d_Y} - 1)\, m_F\, m_X\, 2^{d_Z}. \tag{4}$$

For example, with two functional forms ($m_F = 2$), two target variables ($d_Y = 2$), two alternative focal predictors ($m_X = 2$), and four candidate controls ($d_Z = 4$), we have

$$|\Pi| = (2^2 - 1) \times 2 \times 2 \times 2^4 = 192.$$

- The main problem with current research is that the sample of Π reported by researchers is not random.

How The Code Works I: Package Layout

- `prototypes.py`: abstract base classes plus shared setup. This file defines `Protomodel`, `Protoresult`, and especially `BaseRobust`.
- `models.py`: the execution layer. This is where `OLSRobust.fit()` and `LRobust.fit()` actually build and estimate specifications.
- `utils.py`: low-level helpers such as `all_subsets`, `sample_y_masks`, `sample_z_masks`, stripped OLS/logit routines, and result concatenation.
- `figures.py`: plotting/reporting methods that read from stored result objects rather than from live fitted models.

How The Code Works II: `BaseRobust`

- Every user-facing model inherits from `BaseRobust(y, x, data, ...)`. The constructor does the boring but important work first: type checks, column existence checks, and missingness warnings.
- Composite outcomes are not hard-coded by hand. `multiple_y()` standardizes candidate outcome columns, enumerates or samples subsets, and stores the resulting composite series in `self.y_composites`.
- The object also keeps a `parameters` dictionary, so the run configuration, sampled masks, and generated specifications can travel with the eventual results object.

How The Code Works III: Fitting Engine

- `OLSRobust.fit()` and `LRobust.fit()` turn the abstract design into many concrete design matrices: one choice of Y, one focal predictor, one estimator, one control mask.
- For controls, the code can enumerate the full power set or sample it using `sample_z_specs()` and `sample_z_masks()`, which matters when the full space is too large to run exhaustively.
- Each specification is then sent to a lean numerical routine such as `stripped_ols()` or the statsmodels-based logistic path, with extra handling for fixed effects, constants, and collinearity.

How The Code Works IV: Parallelism and Resampling

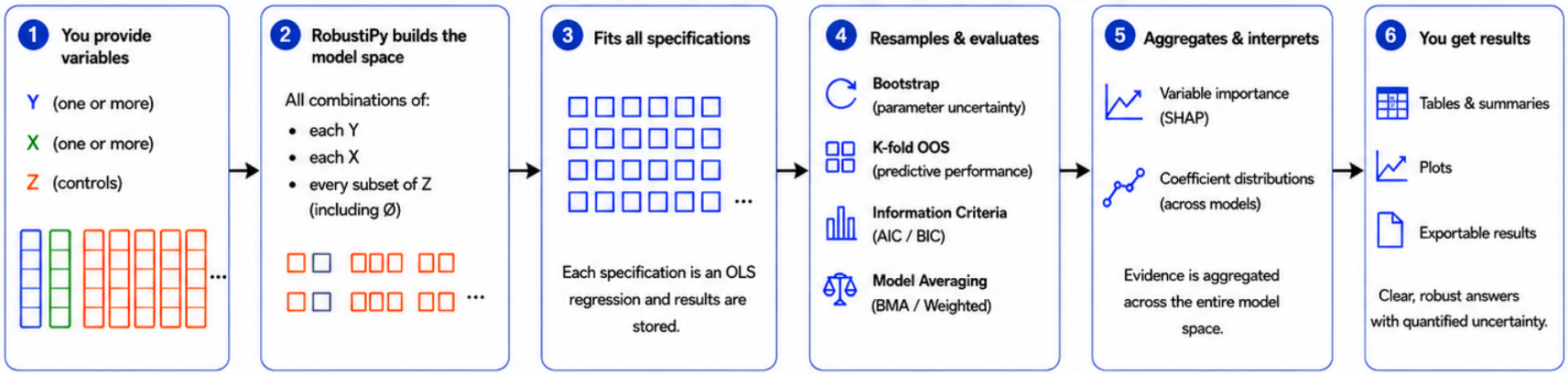
- The heavy work is not a giant `for` loop in notebook code; it sits inside `models.py` and is dispatched with `joblib.Parallel` using seeded batches.
- Bootstrap draws go through helpers such as `_run_parallel_seed_batches()` and grouped-bootstrap utilities, so clustered resampling and ordinary resampling share the same engine.
- Out-of-sample evaluation also plugs into that machinery via `KFold`, `GroupKFold`, and `StratifiedKFold`, instead of being a separate subsystem.

How The Code Works V: Result Objects

- The stable abstraction is the result container, not the transient fitted model. `OLSResult` stores coefficients, p-values, fit metrics, bootstrap outputs, and the metadata telling us which specification produced each row.
- That is why the user API is mostly methods on results objects: `summary()`, `plot()`, serialization, and merging via `MergedResult` or `concat_results()`.
- In other words, the Python architecture is: validate in `BaseRobust`, execute in `models.py`, support the numerics in `utils.py`, and expose analysis through result objects.

How RobustiPy Computes So Much at Once

RobustiPy takes your Y, X, and Z, then automatically builds, estimates, and evaluates thousands of reasonable linear models in one call.



Programmatic view (as in the empirical examples)

1 Prepare your data and variable lists

```
import pandas as pd
from robustipy import OLSRobust  # or LRobust for (generalized) linear models

data = pd.read_csv('...')
y = ['log_homicide']
x = ['police_per_capita']
z = ['poverty', 'unemployment', 'pc_income', 'education',
     'population', 'gini', 'urban', 'alcohol']
```

You pass your DataFrame and the lists of Y, X, and Z. That's all the model-space information you provide.

2 Create the RobustiPy object

```
rp = OLSRobust(data=data, y=y, x=x, z=z)
```

RobustiPy stores your data and choices.

3 Fit once — RobustiPy does the rest

```
rp.fit(controls='all',        # consider all controls in Z
       draws=1000,            # bootstrap draws
       kfold=10,              # k-fold for OOS
       seed=123,              # reproducibility
       oos_metric='pseudo-r2',# predictive metric (default: pseudo-r2)
       threshold= 0.01)       # omit very unlikely models
```

With a single .fit ( ) call, RobustiPy automatically:

- creates every combination of Y × X × P(Z) (all subsets of controls, including Ø),
- estimates each model (OLS),
- bootstraps for parameter uncertainty,
- runs k-fold OOS for predictive performance,
- computes AIC/BIC and model weights (BMA),
- ranks and aggregates evidence.

4 Get results

```
results = rp.get_results()
```

All results across thousands of specifications are collected in a Results object.

5 Explore and export

```
results.summary()              # tables, metrics, top models
results.plot('shap')           # SHAP importance plot
results.plot('coef_dists')     # coefficient distributions
results.export('results.csv')  # everything to a file
```

Use built-in methods to summarize, visualize, and export—no manual enumeration required.

Key idea

You specify the variables. RobustiPy automatically **searches the entire reasonable model space,** evaluates every model rigorously, and aggregates the evidence.

Typical scale

- If |Z| = 8 controls → $2^8 = 256$ subsets
- With 1 Y and 1 X → 256 specifications
- With multiple Y's or X's → scales accordingly
- Each evaluated with bootstrap, k-fold, AIC/BIC, BMA, SHAP, and more

Functionality and Examples

- RobustiPy undertakes five novel types of common research design.

Functionality and Examples

- RobustiPy undertakes five novel types of common research design.
- It comes with:

Functionality and Examples

- RobustiPy undertakes five novel types of common research design.
- It comes with:
  - Five simulated examples

Functionality and Examples

- RobustiPy undertakes five novel types of common research design.
- It comes with:
  - Five simulated examples
  - Ten empirical examples

Functionality and Examples

- RobustiPy undertakes five novel types of common research design.
- It comes with:
  - Five simulated examples
  - Ten empirical examples
  - Full time profiling

Functionality and Examples

- RobustiPy undertakes five novel types of common research design.
- It comes with:
  - Five simulated examples
  - Ten empirical examples
  - Full time profiling
- It gives you back:

Functionality and Examples

- RobustiPy undertakes five novel types of common research design.
- It comes with:
  - Five simulated examples
  - Ten empirical examples
  - Full time profiling
- It gives you back:
  - Static but comprehrensive visualisations

Functionality and Examples

- RobustiPy undertakes five novel types of common research design.
- It comes with:
  - Five simulated examples
  - Ten empirical examples
  - Full time profiling
- It gives you back:
  - Static but comprehrensive visualisations
  - Vast summary printouts

Functionality and Examples

- RobustiPy undertakes five novel types of common research design.
- It comes with:
  - Five simulated examples
  - Ten empirical examples
  - Full time profiling
- It gives you back:
  - Static but comprehrensive visualisations
  - Vast summary printouts
  - Output results for custom analysis

Functionality 1: Vanilla Computation

- Baseline case: single outcome Y, single predictor X, and varying controls Z.

Functionality 1: Vanilla Computation

- Baseline case: single outcome Y, single predictor X, and varying controls Z.
- Data generating process:

$$Y = \alpha + \beta X + \gamma Z + \varepsilon$$

Functionality 1: Vanilla Computation

- Baseline case: single outcome Y, single predictor X, and varying controls Z.
- Data generating process:

$$Y = \alpha + \beta X + \gamma Z + \varepsilon$$

- Specification space reduces to $\Pi_{\overleftrightarrow{Z}}$ only.

Functionality 1: Vanilla Computation

- Baseline case: single outcome Y, single predictor X, and varying controls Z.
- Data generating process:

$$Y = \alpha + \beta X + \gamma Z + \varepsilon$$

- Specification space reduces to $\Pi_{\overleftrightarrow{Z}}$ only.
- Estimation: OLS with k-fold CV and bootstrap resampling.

Functionality 1: Vanilla Computation

- Baseline case: single outcome Y, single predictor X, and varying controls Z.
- Data generating process:

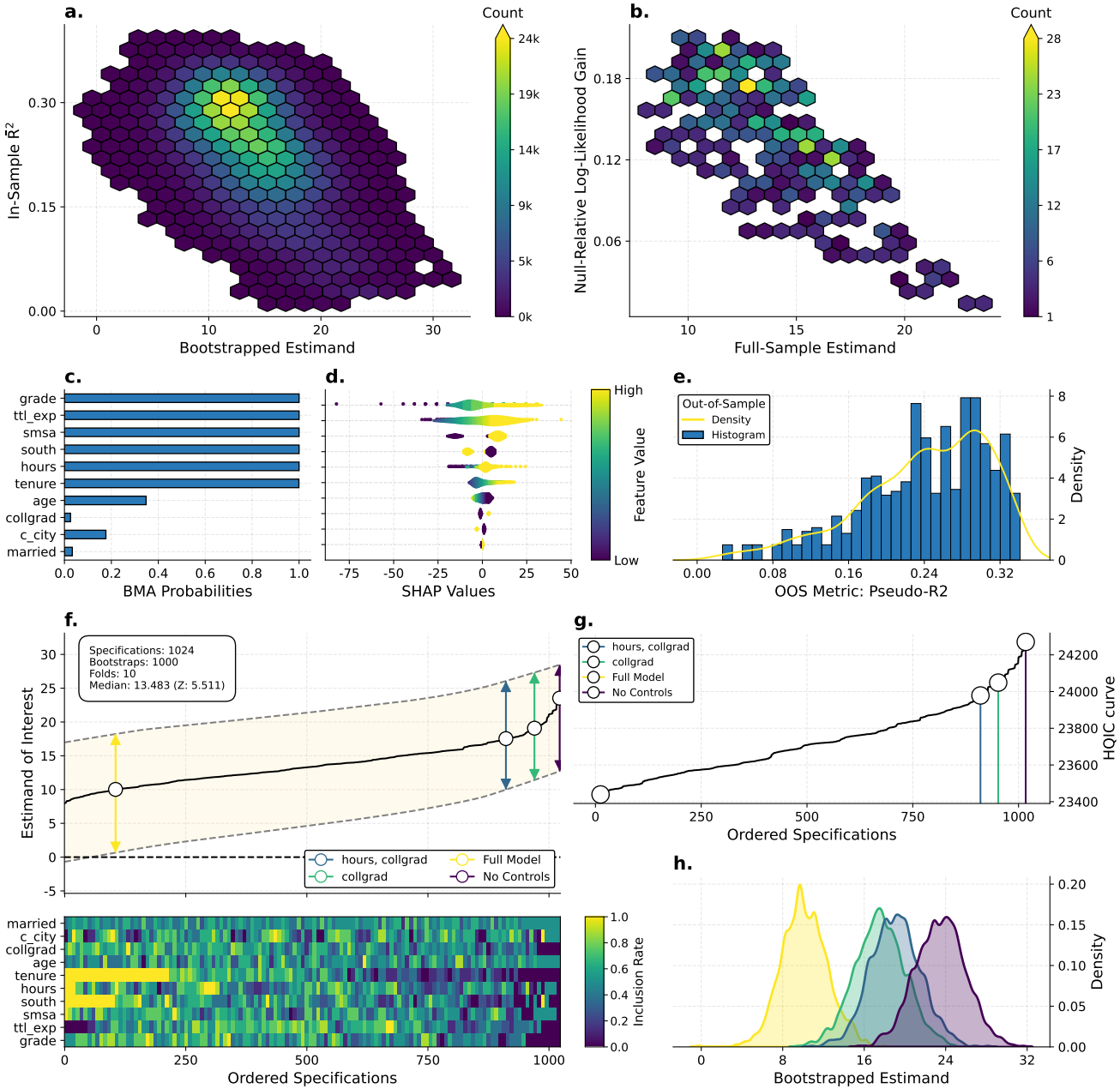
$$Y = \alpha + \beta X + \gamma Z + \varepsilon$$

- Specification space reduces to $\Pi_{\overleftrightarrow{Z}}$ only.
- Estimation: OLS with k-fold CV and bootstrap resampling.
- Outputs: distribution of $\hat{\beta}$ across $2^{|Z|}$ models, with null vs full model.

Code Block 1: Vanilla computation

```python
import numpy as np
import pandas as pd
from robustipy.models import OLSRobust


def sim1(project_name):
    # 1. Setup
    n = 100
    rng = np.random.default_rng(192735)
    z = [f"z{i}" for i in range(1, 5)]

    # 2. Simulate covariates
    X = rng.standard_normal((n, 1))
    Z = rng.standard_normal((n, len(z)))
    df = pd.DataFrame(np.hstack([X, Z]), columns=["x1"] + z)

    # 3. Generate outcome y1
    df["y1"] = (1 + 2 * df["x1"] + 0.5 * df[z].sum(axis=1)
                + rng.standard_normal(n) * 0.5)

    # 4. Fit robust OLS with cross-validation
    vanilla_obj = OLSRobust(y=["y1"], x=["x1"], data=df)
    vanilla_obj.fit(controls=z, draws=1000, kfold=10, seed=192735)

    # 5. Retrieve and plot results
    vanilla_res = vanilla_obj.get_results()
    vanilla_res.plot(specs=[z[1:3]], figsize=(16, 16), ci=0.95,
                     ic="bic", ext="svg", figpath='../figures',
                     project_name=project_name)
    vanilla_res.summary()


if __name__ == "__main__":
    sim1('sim1_example')
```

==========
1. Model Summary
==========
Model: OLS Robust
Inference Tests: Yes
Dependent variable: R
Independent variable: Inequality
Number of possible controls: 7
...
==========
2. Model Robustness Metrics
==========
2.1 Inference Metrics
==========
Median β (specs, no resampling): 0.6976 (null-calibrated p: 0)
Min Adj-R^2: -0.0310, Specs: ['Age', 'Unemployment']
...
Share β <0 & significant (specs, no resampling): 0.0000
Share β <0 & significant (bootstraps × specs): 0.0163 | null-calibrated p: 0.001998
Stouffer's Z = 29.1123 (two-sided p = 2.51e-186)
==========
2.2 In-Sample Metrics (Full Sample)
==========
Min AIC: 507.5432, Specs: ['Unemployment', 'Expenditure', 'N', 'Wealth', 'Males', 'Age', 'Ed']
Max Adj-R^2: 0.6922, Specs: ['Expenditure', 'Wealth', 'Males', 'Age', 'Ed']
Min Adj-R^2: -0.0310, Specs: ['Age', 'Unemployment']
==========
2.3 Out-Of-Sample Metrics (pseudo-r2 averaged across folds)
==========
Max Average: 0.4925, Specs: ['Age', 'Unemployment', 'Expenditure', 'Ed']
Min Average: -0.5057, Specs: ['Age', 'Unemployment', 'Ed', 'Males']
Mean Average: 0.12
Median Average: 0.2026


a.
Count
24k
19k
14k
9k
5k
0k
In-Sample $\bar{R}^2$
0.00
0.15
0.30
0
10
20
30
Bootstrapped Estimand
b.
Count
28
23
17
12
6
1
Null-Relative Log-Likelihood Gain
0.06
0.12
0.18
10
15
20
Full-Sample Estimand
c.
grade
ttl_exp
smsa
south
hours
tenure
age
collgrad
c_city
married
0.0
0.2
0.4
0.6
0.8
1.0
BMA Probabilities
d.
-75
-50
-25
0
25
50
SHAP Values
High
Feature Value
Low
e.
Out-of-Sample
Density
Histogram
0.00
0.08
0.16
0.24
0.32
OOS Metric: Pseudo-R2
Density
0
2
4
6
8
f.
Specifications: 1024
Bootstraps: 1000
Folds: 10
Median: 13.483 (Z: 5.511)
Estimand of Interest
hours, collgrad
collgrad
Full Model
No Controls
married
c_city
collgrad
age
tenure
hours
south
smsa
ttl_exp
grade
0
250
500
750
1000
Ordered Specifications
Inclusion Rate
0.0
0.2
0.4
0.6
0.8
1.0
g.
hours, collgrad
collgrad
Full Model
No Controls
23400
23600
23800
24000
24200
HQIC curve
0
250
500
750
1000
Ordered Specifications
h.
0.00
0.05
0.10
0.15
0.20
Density
0
8
16
24
32
Bootstrapped Estimand

Importantly, note the opportunity to highlight individual specifications!

Importantly, note the opportunity to highlight individual specifications!

This allows us to consider where classical models live in the permissible model space.

Functionality 2: Array of Never-Changing Variables

- Some predictors are fixed (always included in $\overleftrightarrow{X}$).

Functionality 2: Array of Never-Changing Variables

- Some predictors are fixed (always included in $\overleftrightarrow{X}$).
- Reduces computational cost by shrinking the specification space.

Functionality 2: Array of Never-Changing Variables

- Some predictors are fixed (always included in $\overleftrightarrow{X}$).
- Reduces computational cost by shrinking the specification space.
- Empirical motivation: theoretically indispensable covariates.

Functionality 2: Array of Never-Changing Variables

- Some predictors are fixed (always included in $\overleftrightarrow{X}$).
- Reduces computational cost by shrinking the specification space.
- Empirical motivation: theoretically indispensable covariates.
- Specification space:

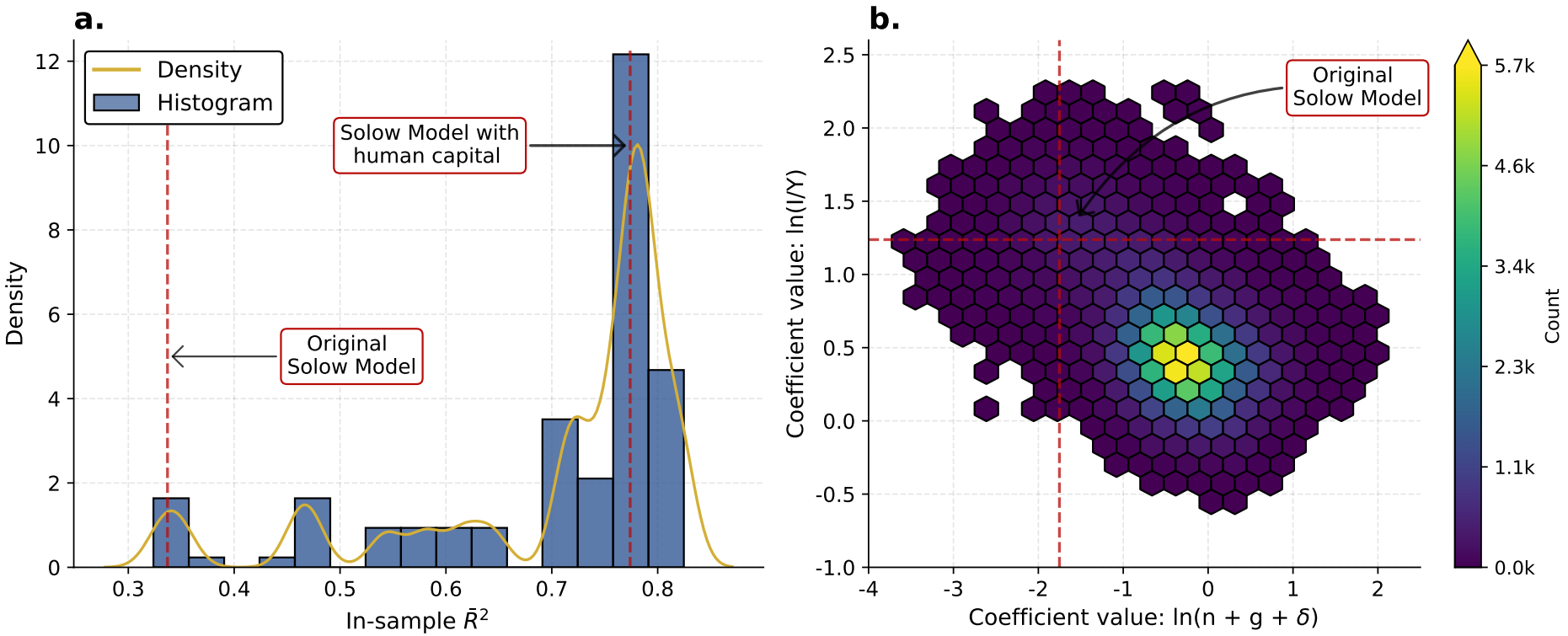
$$\Pi = \Pi_{\overleftrightarrow{Z}} \quad \text{with fixed } X_f$$

Functionality 2: Array of Never-Changing Variables

- Some predictors are fixed (always included in $\overleftrightarrow{X}$).
- Reduces computational cost by shrinking the specification space.
- Empirical motivation: theoretically indispensable covariates.
- Specification space:

$$\Pi = \Pi_{\overleftrightarrow{Z}} \quad \text{with fixed } X_f$$

- Results: interpretable estimates with reduced variance and runtime.


a.
Density
Histogram
Solow Model with human capital
Original Solow Model
Density
In-sample $\bar{R}^2$
b.
Original Solow Model
Coefficient value: ln(I/Y)
Coefficient value: ln(n + g + δ)
Count
5.7k
4.6k
3.4k
2.3k
1.1k
0.0k

Functionality 3: Fixed Effects OLS

- Designed for panel/longitudinal data with grouping variable g.

Functionality 3: Fixed Effects OLS

- Designed for panel/longitudinal data with grouping variable g.
- Within-transformation removes group-specific heterogeneity:

$$Y_{it} - \bar{Y}_i = \beta(X_{it} - \bar{X}_i) + \varepsilon_{it}$$

Functionality 3: Fixed Effects OLS

- Designed for panel/longitudinal data with grouping variable g.
- Within-transformation removes group-specific heterogeneity:

$$Y_{it} - \bar{Y}_i = \beta(X_{it} - \bar{X}_i) + \varepsilon_{it}$$

- Implemented by demeaning variables by g.

Functionality 3: Fixed Effects OLS

- Designed for panel/longitudinal data with grouping variable g.
- Within-transformation removes group-specific heterogeneity:

$$Y_{it} - \bar{Y}_i = \beta(X_{it} - \bar{X}_i) + \varepsilon_{it}$$

- Implemented by demeaning variables by g.
- $\Pi_{\underset{Z}{\leftrightarrow}}$ still varied, but estimation is within-groups.

Functionality 3: Fixed Effects OLS

- Designed for panel/longitudinal data with grouping variable g.
- Within-transformation removes group-specific heterogeneity:

$$Y_{it} - \bar{Y}_i = \beta(X_{it} - \bar{X}_i) + \varepsilon_{it}$$

- Implemented by demeaning variables by g.
- $\Pi_{\underset{Z}{\leftrightarrow}}$ still varied, but estimation is within-groups.
- Useful for causal inference in repeated-measures data.

Functionality 4: Binary Dependents

- Logistic regression estimator for binary $\mathcal{Y} \in \{0, 1\}$.

Functionality 4: Binary Dependents

- Logistic regression estimator for binary $Y \in \{0, 1\}$.
- Model:

$$\Pr(Y = 1 \mid X, Z) = \frac{1}{1 + \exp(-(\beta X + \gamma Z))}$$

Functionality 4: Binary Dependents

- Logistic regression estimator for binary $Y \in \{0, 1\}$.
- Model:

$$\Pr(Y = 1 \mid X, Z) = \frac{1}{1 + \exp(-(\beta X + \gamma Z))}$$

- $\Pi_{\underset{Z}{\leftrightarrow}}$ explored as before, with log-likelihood fit.

Functionality 4: Binary Dependents

- Logistic regression estimator for binary $Y \in \{0, 1\}$.
- Model:

$$\Pr(Y = 1 \mid X, Z) = \frac{1}{1 + \exp(-(\beta X + \gamma Z))}$$

- $\Pi_{\overleftrightarrow{Z}}$ explored as before, with log-likelihood fit.
- Supports odds ratio interpretation of $\hat{\beta}$.

Functionality 4: Binary Dependents

- Logistic regression estimator for binary $Y \in \{0, 1\}$.
- Model:

$$\Pr(Y = 1 \mid X, Z) = \frac{1}{1 + \exp(-(\beta X + \gamma Z))}$$

- $\Pi_{\overleftrightarrow{Z}}$ explored as before, with log-likelihood fit.
- Supports odds ratio interpretation of $\hat{\beta}$.
- Alternative: linear probability model with OLSRobust.

Functionality 5: Multiple Dependents

- Allows multiple operationalisations of Y.

Functionality 5: Multiple Dependents

- Allows multiple operationalisations of Y.
- Construct composites by averaging z-scored outcomes:

$$Y_{i,S}^{\text{comp}} = \frac{1}{|S|} \sum_{j \in S} \tilde{Y}_i^{(j)}$$

Functionality 5: Multiple Dependents

- Allows multiple operationalisations of Y.
- Construct composites by averaging z-scored outcomes:

$$Y_{i,S}^{\text{comp}} = \frac{1}{|S|} \sum_{j \in S} \tilde{Y}_i^{(j)}$$

- Specification space enlarges: $\Pi_{\overleftrightarrow{Y}} \times \Pi_{\overleftrightarrow{Z}}$.

Functionality 5: Multiple Dependents

- Allows multiple operationalisations of Y.
- Construct composites by averaging z-scored outcomes:

$$Y_{i,S}^{\text{comp}} = \frac{1}{|S|} \sum_{j \in S} \tilde{Y}_i^{(j)}$$

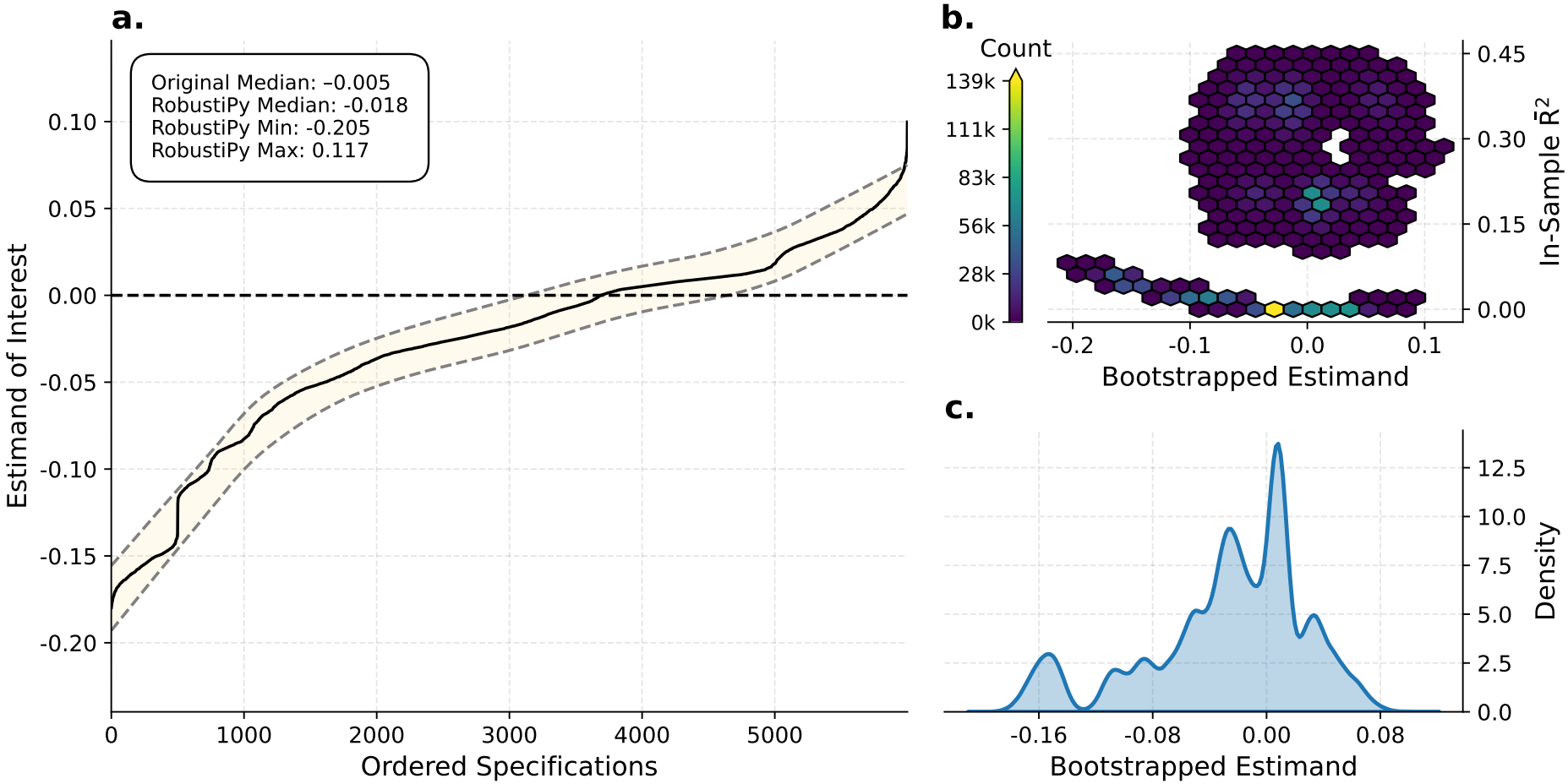
- Specification space enlarges: $\Pi_{\overleftrightarrow{Y}} \times \Pi_{\overleftrightarrow{Z}}$.
- Provides robustness to measurement uncertainty in the outcome.

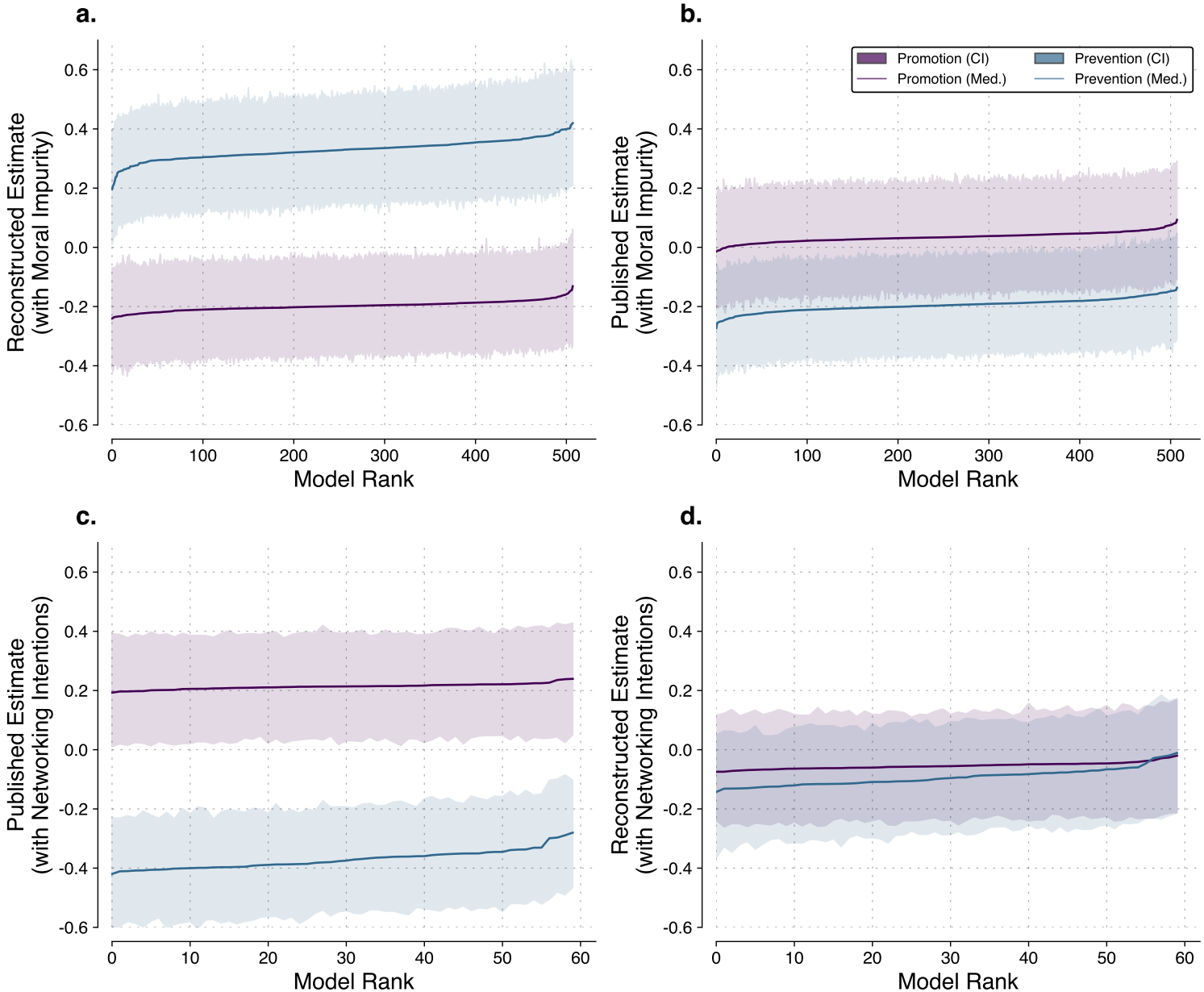
Functionality 5: Multiple Dependents

- Allows multiple operationalisations of Y.
- Construct composites by averaging z-scored outcomes:

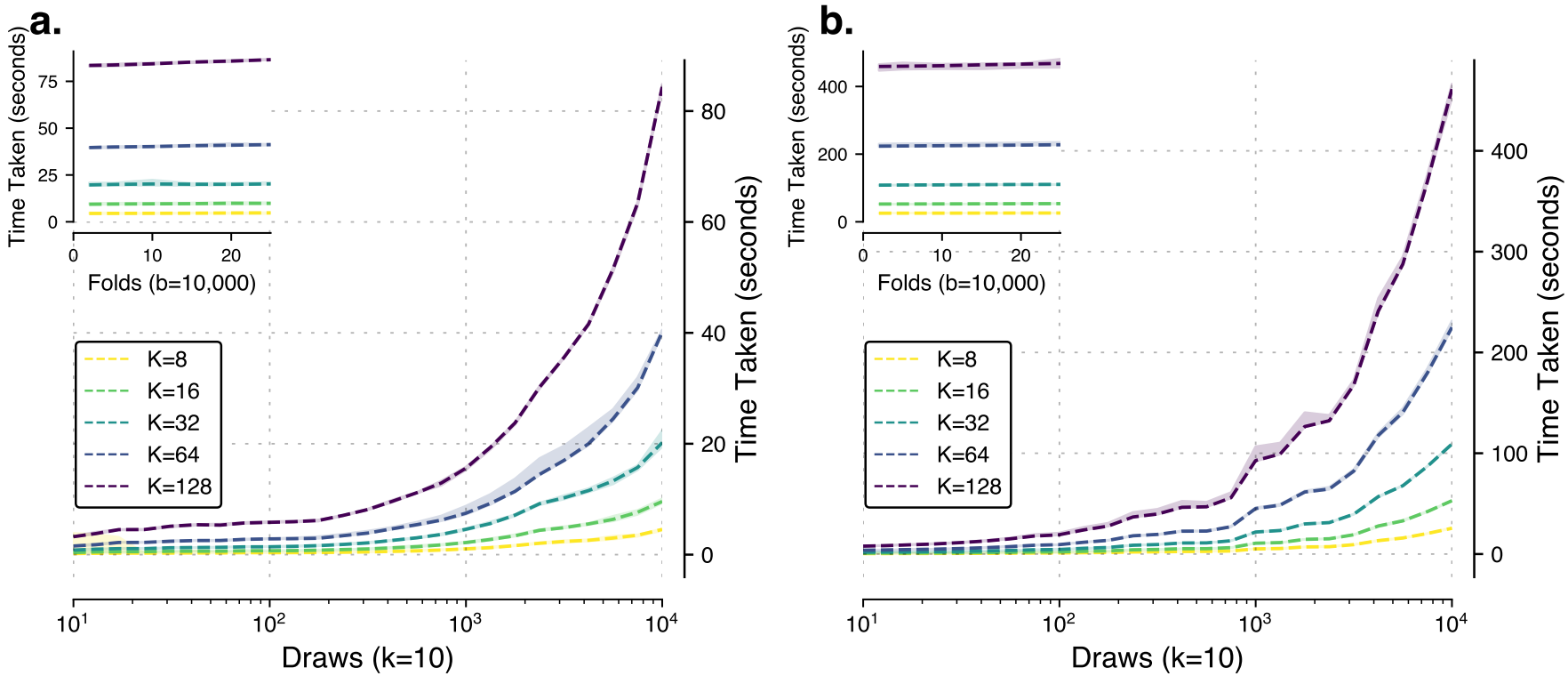
$$Y_{i,S}^{\text{comp}} = \frac{1}{|S|} \sum_{j \in S} \tilde{Y}_i^{(j)}$$

- Specification space enlarges: $\Pi_{\leftrightarrow Y} \times \Pi_{\leftrightarrow Z}$.
- Provides robustness to measurement uncertainty in the outcome.
- Outputs: specification curves across all dependent definitions/composites.


a.
Original Median: –0.005
RobustiPy Median: -0.018
RobustiPy Min: -0.205
RobustiPy Max: 0.117
Estimand of Interest
Ordered Specifications
b.
Count
139k
111k
83k
56k
28k
0k
In-Sample $\bar{R}^2$
Bootstrapped Estimand
c.
Density
Bootstrapped Estimand


a.
b.
c.
d.
Promotion (CI)
Promotion (Med.)
Prevention (CI)
Prevention (Med.)
Reconstructed Estimate (with Moral Impurity)
Published Estimate (with Moral Impurity)
Published Estimate (with Networking Intentions)
Reconstructed Estimate (with Networking Intentions)
Model Rank

Time Profiling



Conclusion

- Model uncertainty: measurement/functional form/covariate choices.
  - A feasible specification space makes assumptions explicit.

Conclusion

- Model uncertainty: measurement/functional form/covariate choices.
  - A feasible specification space makes assumptions explicit.
- RobustiPy automates exploration of Π across $\overleftrightarrow{Y}$, $\overleftrightarrow{X}$, $\overleftrightarrow{Z}$, $\overleftrightarrow{F()}$.
  - CV, bootstraps, spec curves, averaging, selection, etc.

Conclusion

- Model uncertainty: measurement/functional form/covariate choices.
  - A feasible specification space makes assumptions explicit.
- RobustiPy automates exploration of Π across $\overleftrightarrow{Y}$, $\overleftrightarrow{X}$, $\overleftrightarrow{Z}$, $\overleftrightarrow{F()}$.
  - CV, bootstraps, spec curves, averaging, selection, etc.
- Spotlight preferred models or undertake joint inference.

Conclusion

- Model uncertainty: measurement/functional form/covariate choices.
  - A feasible specification space makes assumptions explicit.
- RobustiPy automates exploration of Π across $\overleftrightarrow{Y}$, $\overleftrightarrow{X}$, $\overleftrightarrow{Z}$, $\overleftrightarrow{F()}$.
  - CV, bootstraps, spec curves, averaging, selection, etc.
- Spotlight preferred models or undertake joint inference.
- Workflow encourages transparent, theory-led reporting.
  - It reduces room for p-hacking or HARKing.

Conclusion

- Model uncertainty: measurement/functional form/covariate choices.
  - A feasible specification space makes assumptions explicit.
- RobustiPy automates exploration of Π across $\overleftrightarrow{Y}$, $\overleftrightarrow{X}$, $\overleftrightarrow{Z}$, $\overleftrightarrow{F()}$.
  - CV, bootstraps, spec curves, averaging, selection, etc.
- Spotlight preferred models or undertake joint inference.
- Workflow encourages transparent, theory-led reporting.
  - It reduces room for p-hacking or HARKing.
  - Open-source and ready to use: github.com/RobustiPy.
    - Try the notebooks and tell us what to add next!
    - We're specifically looking for help with more estimators!

Sincere thanks to my **wonderful** co-authors!