Probabilistic Programming for Embedding Theory and Quantifying Uncertainty in Econometric Analysis

Thomas Heckelei[a]
together with Hugo Storm[a], Kathy Baylis[b]

Keynote, XVII EAAE Congress 2023 Rennes

[a]University of Bonn, [b]University of California Santa Barbara

Time of debate and methodological advances

- Debate on statistical malpractice and publication bias
- Literature suggests change of reporting rules, better training, “mindful” statistical analysis, solid theoretical basis, …
  (Heckelei et al. 2023, Gigerenzer 2004, 2018)
- Data science and algorithmic advances with potential to expand the econometricians’ toolbox
  (Storm et al. 2020; Athey, Imbens 2019)

© adobe stock

What I’d like to get to today

- **Bayesian inference** is intuitive and naturally embeds uncertainty - time to become Bayesian econometricians
- New algorithms and programming languages give rise to

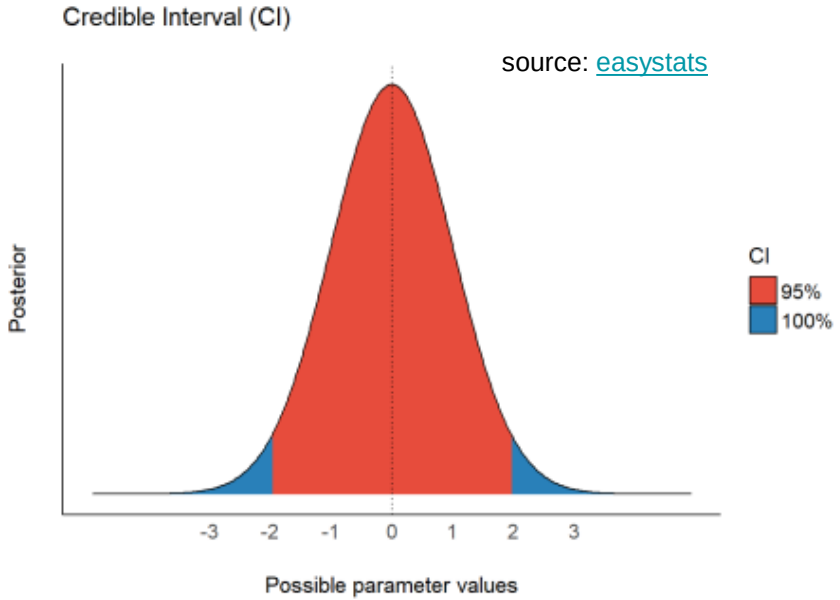
Probabilistic Programming

➔ semi-automatic Bayesian inference
➔ explicit data generation process rigorously connects theory & inference
➔ flexible inclusion of machine learning tools in causal identification setting

© adobe stock

Bayesian statistical inference - an intuitive treat

- Bayesian analysis offers an intuitive alternative to frequentist inference
- The posterior distribution summarizes the available information on a parameter of interest including uncertainty
- It allows a direct probabilistic interpretation



Given model assumptions, the probability that the true parameter value is between -2 and +2 is equal to 95%

What is the posterior (distribution)?

$$post(\theta) \propto prior(\theta) \cdot likelihood(\theta \mid data)$$

Remember: the likelihood is mathematically identical to the sampling distribution

direct link to the Data Generation Process (DGP)

Ignore the choice of the prior for now

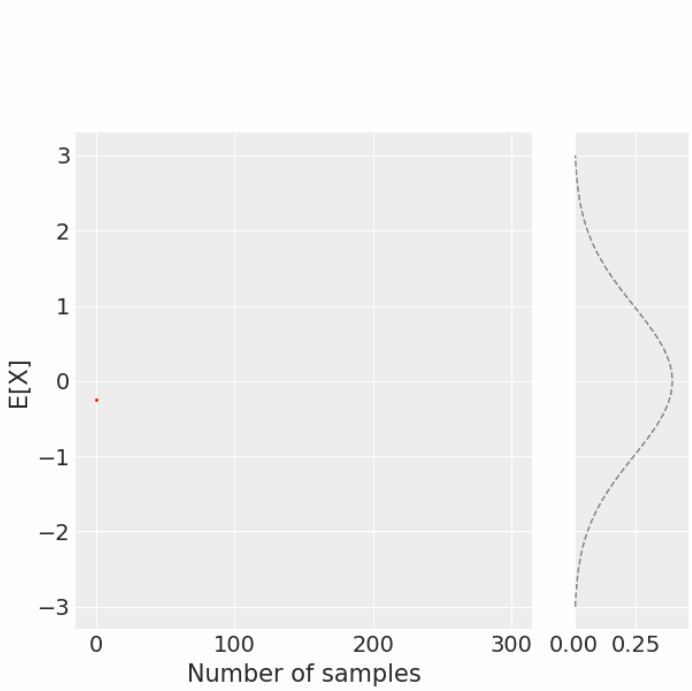
Sampling from the posterior allows statistical inference

$$E\left[f(\theta)\right] = \int f(\theta) post(\theta) d\theta$$

can be approximated arbitrarily closely with a sample of size N as

$$E_{app}\left[f(\theta)\right] = \sum_{N} f(\theta_i)$$

- posterior mean or variance
- probability to lie within interval
- …

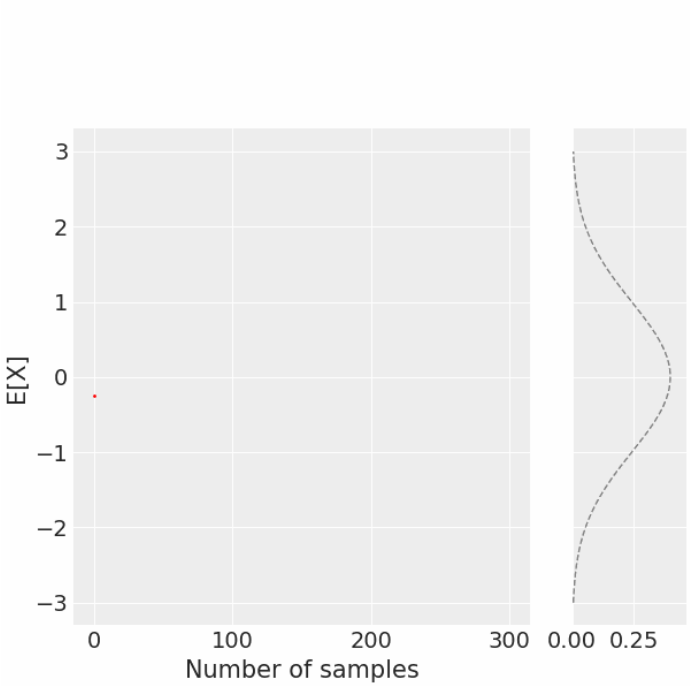


Sampling from the posterior allows statistical inference

- If you can sample from the posterior, any desirable inference is straightforward
- Sample enough outcomes and calculate function of interest

→ posterior distributions and expectations of functions of model parameters

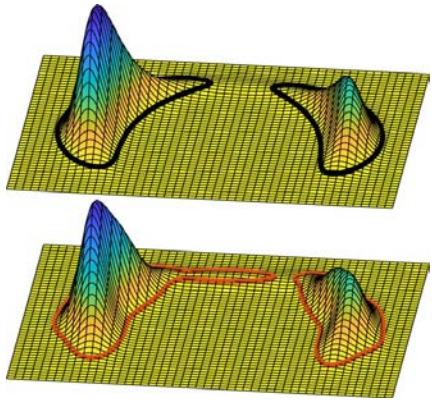
- We would perhaps already be all Bayesians if it always worked…



Why is the Bayesian approach not generally used?

- Sampling from posterior often not straightforward or even possible for flexible prior and likelihood
- Case-specific approaches had to be developed in the past
- Convergence can be a nightmare and technical skill requirements are high
- And more: Lack of training … derivation of likelihood can be hard…

Markov Chain Monte Carlo (MCMC)
Metropolis Hastings
Gibbs sampling
Stochastic Variational Inference (SVI)



Frisch and Hanebeck, 2020

New algorithms and programming languages can help

- Advances of MCMC and SVI make case-specific posterior inference procedures unnecessary
- New programming languages allow posterior inference based on the explicit formulation of the DGP

Probabilistic Programming

© adobe stock

Literature on Probabilistic Programming (PP)

"**There are several reasons why probabilistic programming could prove to be revolutionary for machine intelligence and scientific modelling** (...).

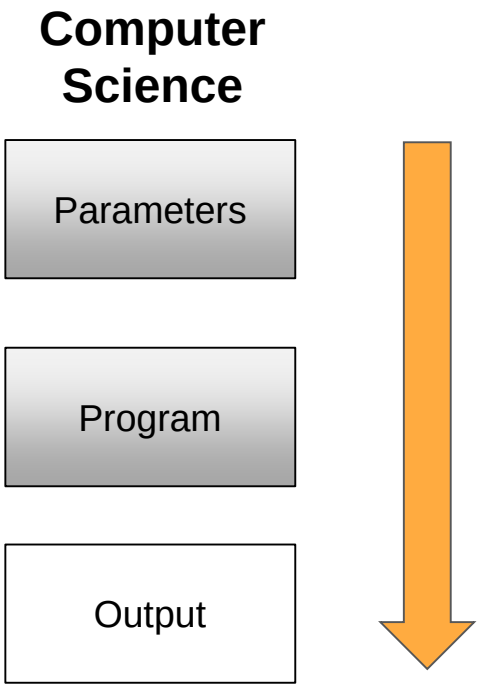
First, (...) obviates the need to manually derive inference methods for models (…). Second, (…) since it allows for rapid prototyping and testing of different models of data."

Ghahramani 2015, p.455

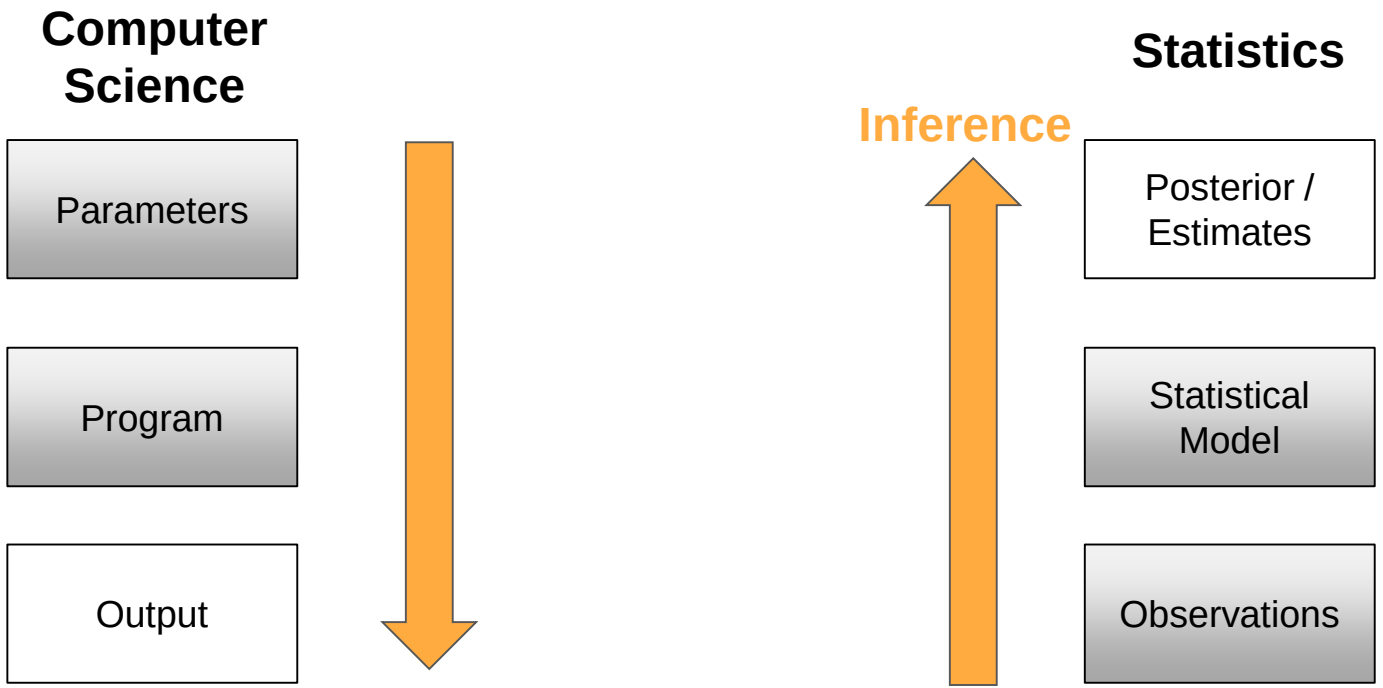
"**Probabilistic programming is about doing statistics using the tools of computer science**."

Meent et al. 2018, p.21

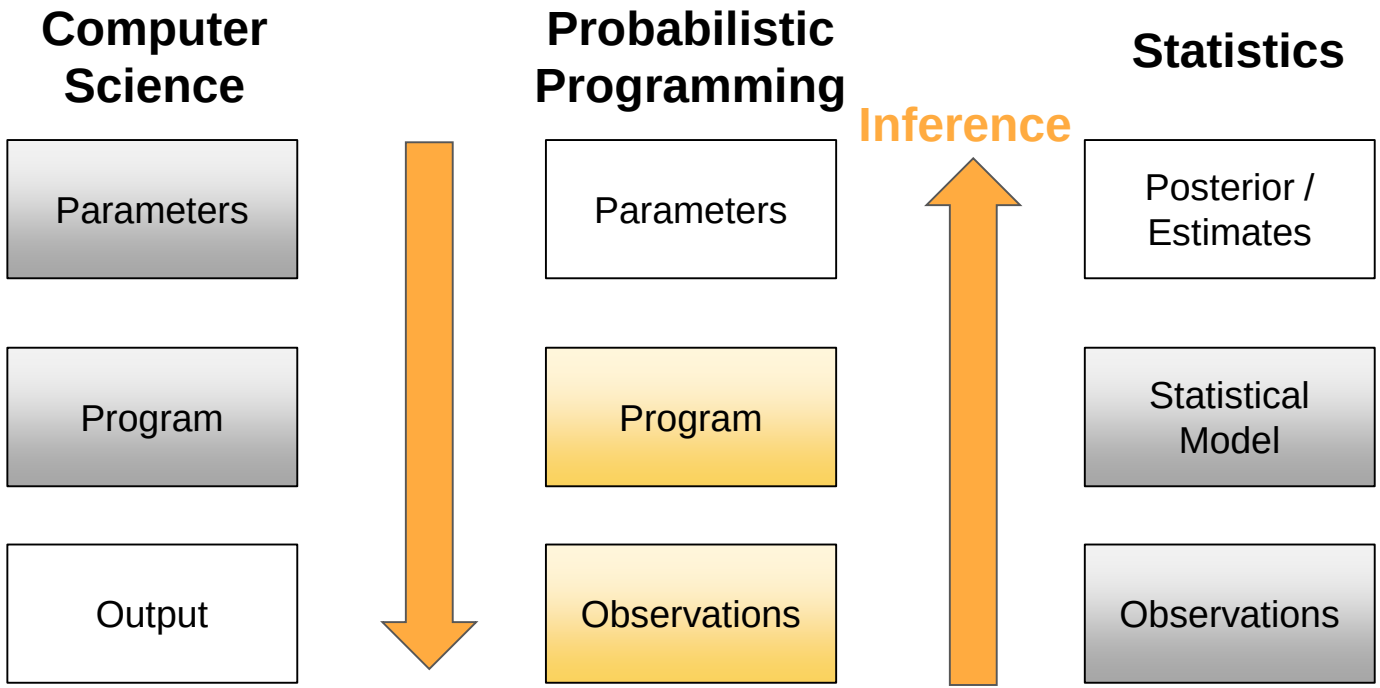
Intuition on PP (Adapted from Meent et al. 2018, p.22)



Intuition on PP (Adapted from Meent et al. 2018, p.22)



Intuition on PP (Adapted from Meent et al. 2018, p.22)



PP eco-system

Frameworks

pyro-ppl/**numpyro**

Probabilistic programming with NumPy powered by JAX for autograd and JIT compilation to GPU/TPU/CPU.

PyMC

RStan

pyro-ppl

Pyro - Deep Universal Probabilistic Programming

82 followers https://pyro.ai @PyroAi

tensorflow/ **probability**

Probabilistic reasoning and statistical analysis in TensorFlow

Diagnosis and Visualization

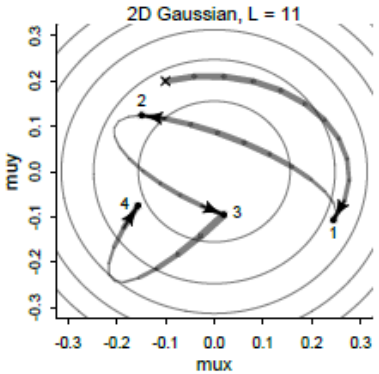
ArviZ

GPU support, auto-diff, and more

New automatic Bayesian inference algorithms

Hamiltonian Monte Carlo inference, using No U-Turn Sampler (NUTS)

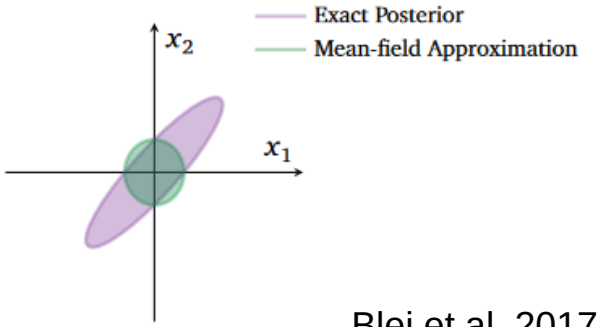
- MCMC samples with no/minimum hyperparameter tuning
- Large speed gain by running on GPU (using JAX gradients)
- Allows larger models and samples sizes (compared to old MCMC)



McElreath 2020

Stochastic Variational Inference (SVI)

- Approximate posterior inference
- much(!) faster than MCMC
- can deal with very large number of parameters and sample sizes



Blei et al. 2017

Linear regression in PP *(using NumPyro)*

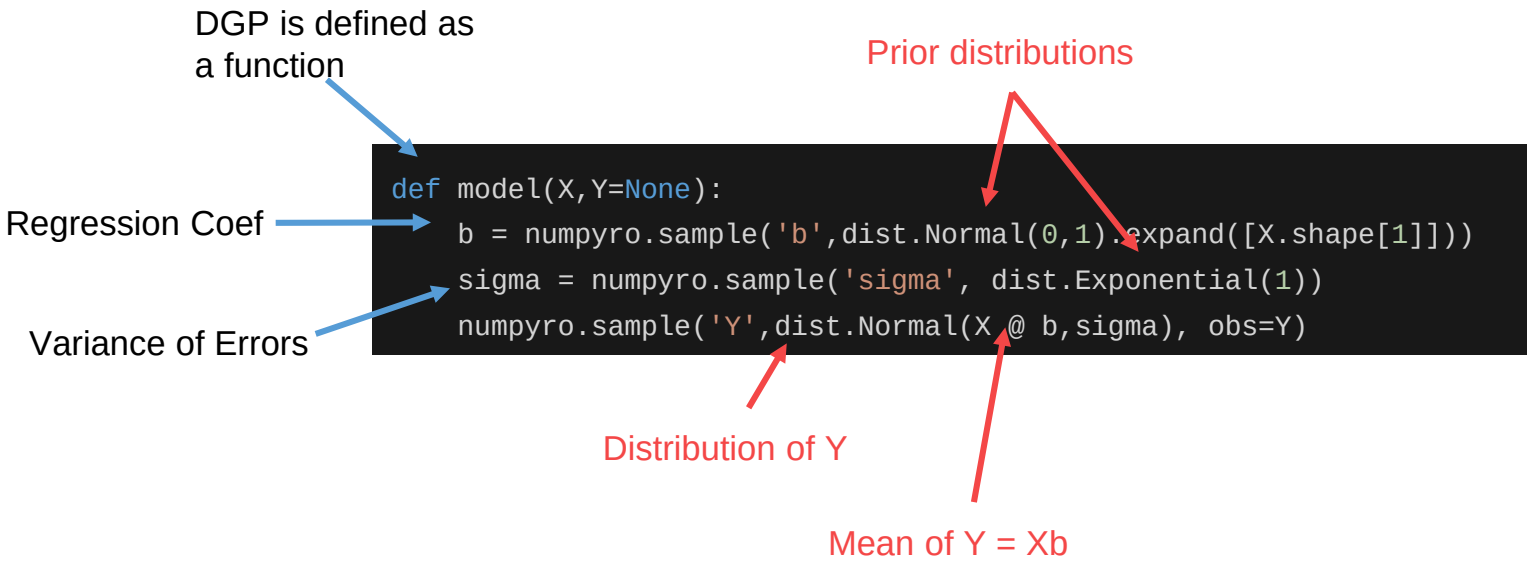
DGP is defined as a function

Regression Coef

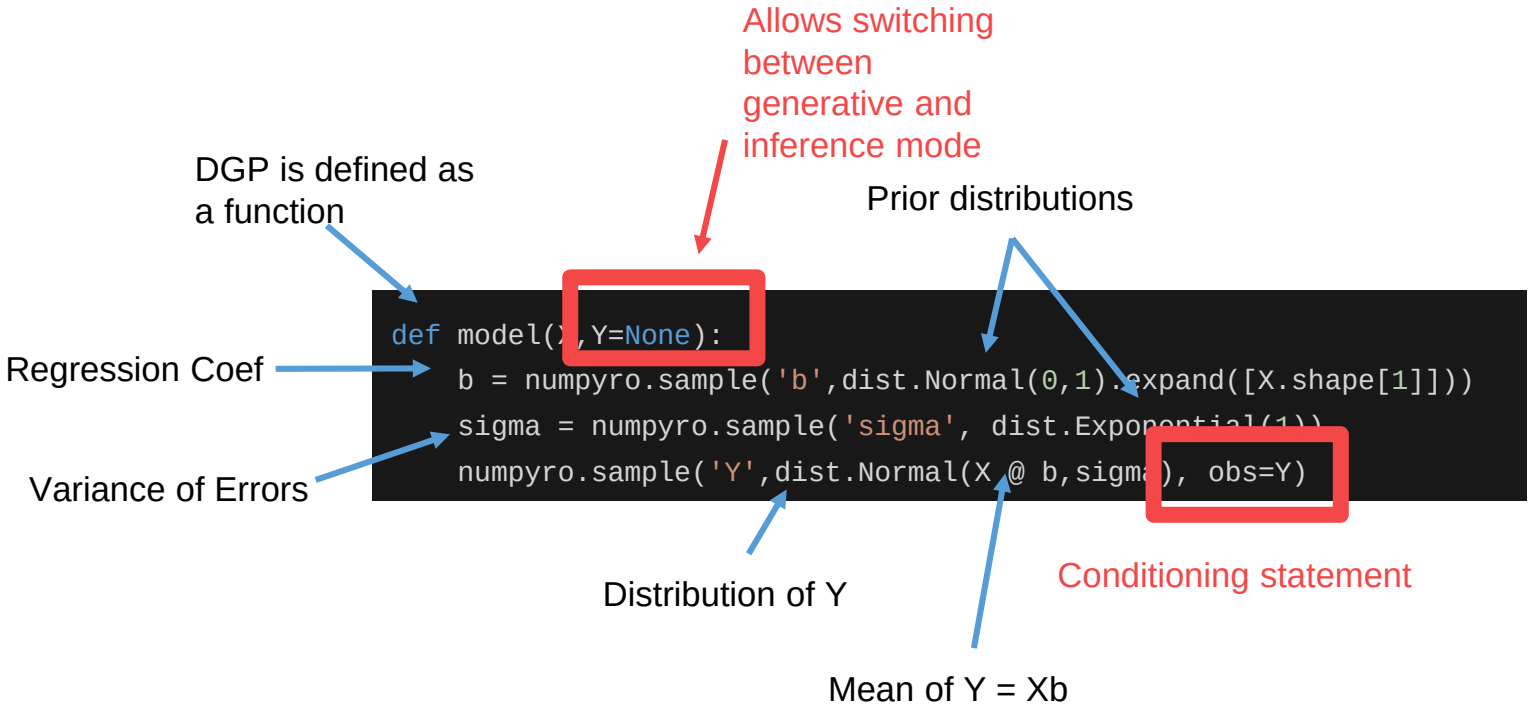
Variance of Errors

```
def model(X,Y=None):
    b = numpyro.sample('b',dist.Normal(0,1).expand([X.shape[1]]))
    sigma = numpyro.sample('sigma', dist.Exponential(1))
    numpyro.sample('Y',dist.Normal(X @ b,sigma), obs=Y)
```

Linear regression in PP *(using NumPyro)*



Linear regression in PP *(using NumPyro)*



Logit model in PP *(using NumPyro)*

Linear regression

```
def model(X,Y=None):
    b = numpyro.sample('b',dist.Normal(0,1).expand([X.shape[1]]))
    sigma = numpyro.sample('sigma', dist.Exponential(1))
    numpyro.sample('Y',dist.Normal(X @ b,sigma), obs=Y)
```

Logit model

```
def modelLogit(X,Y=None):
    b = numpyro.sample('b', dist.Normal(0,1).expand([X.shape[1]]))
    numpyro.sample('Y',dist.Bernoulli(logits=X @ b), obs=Y)
```

Bernoulli distribution
with logit link function

Example 1 - predict winter wheat yield

$$\text{yield} = b_0 + b_1\,\text{soilRating} + b_2\,\text{soilMoist}[\text{Oct}] + \cdots + b_{12}\,\text{soilMoist}[\text{Aug}]$$

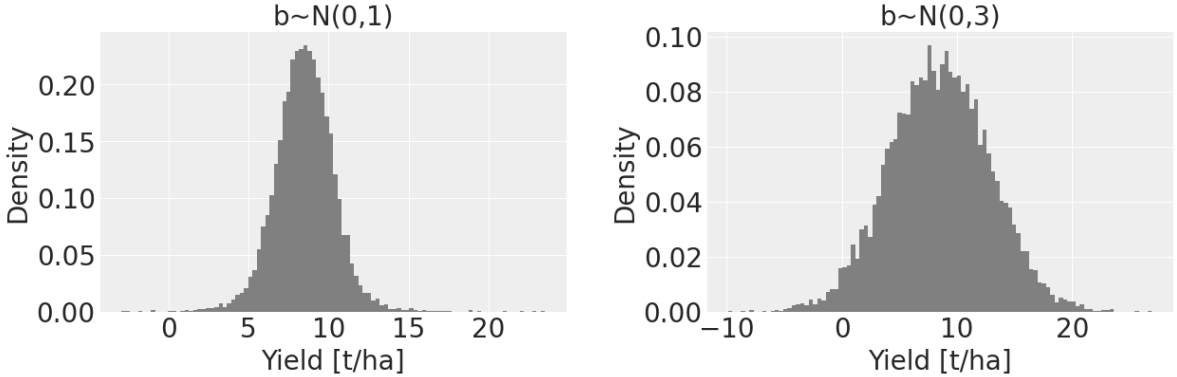
regional average yields, N = 148 (taken from Pahmeyer 2021)

Prior sampling: use of generative model to test model and priors:

1. Add a normal (b ~ N(0,1)) prior on coefficients (for normalized variables)
2. Sample from the prior distribution of parameters
3. Generate and plot predicted yields for prior sample

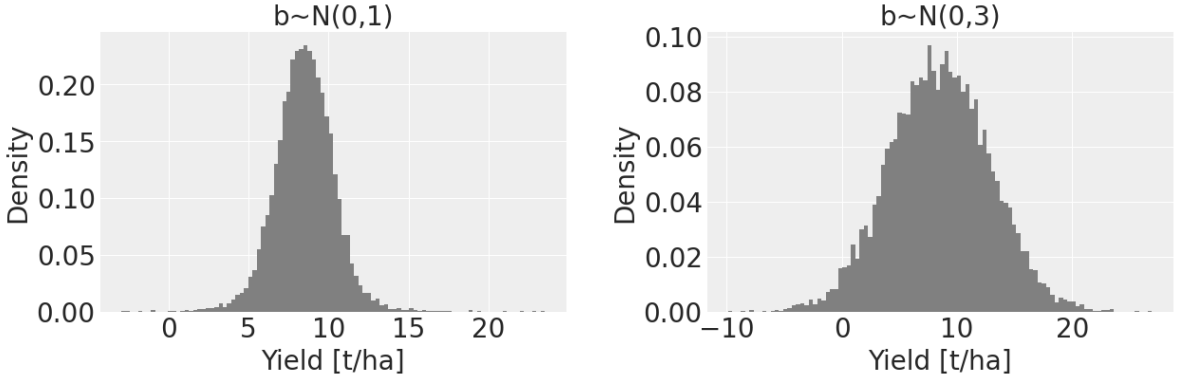
Prior sample outcomes of winter wheat yield

Prior yields



Prior sample outcomes of winter wheat yield

Prior yields



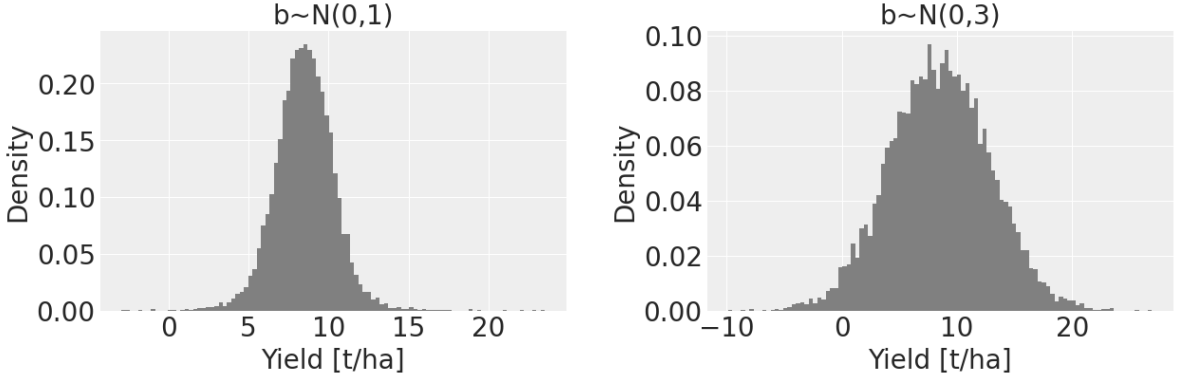
Lincolnshire farmer sets new world record for wheat yield with 17.96t/ha crop

2nd February 2023

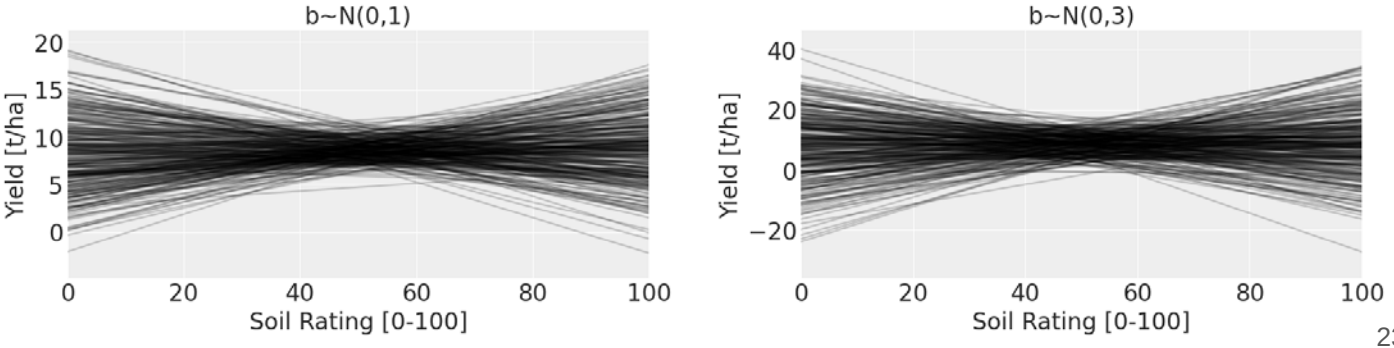
farmersguide.co.uk

Prior sample outcomes of winter wheat yield

Prior yields

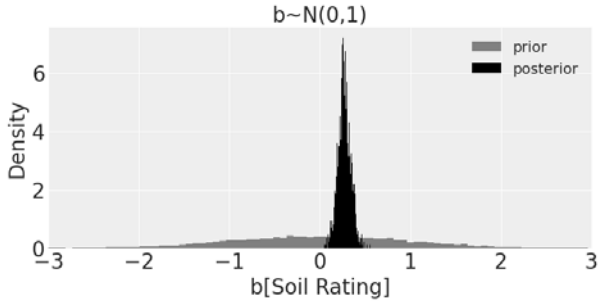


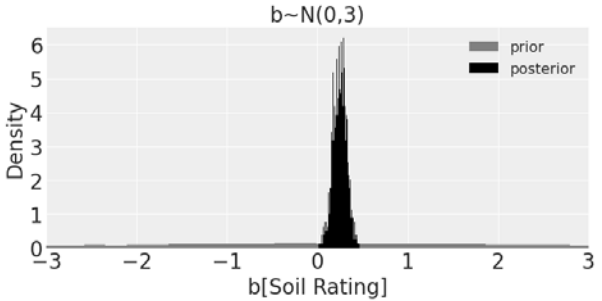
Prior regression lines



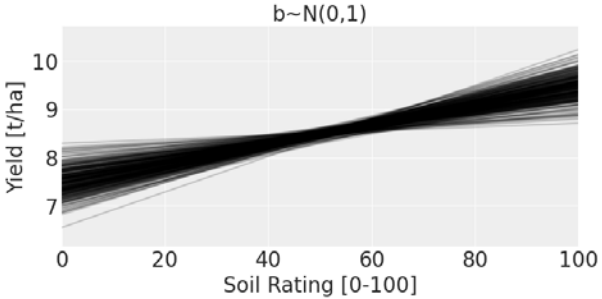
Inference mode: deriving posterior of coefficient(s)

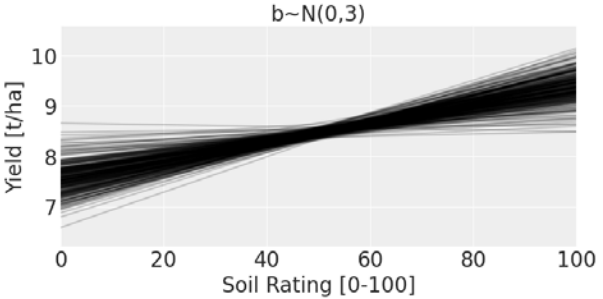
Posterior (and prior)





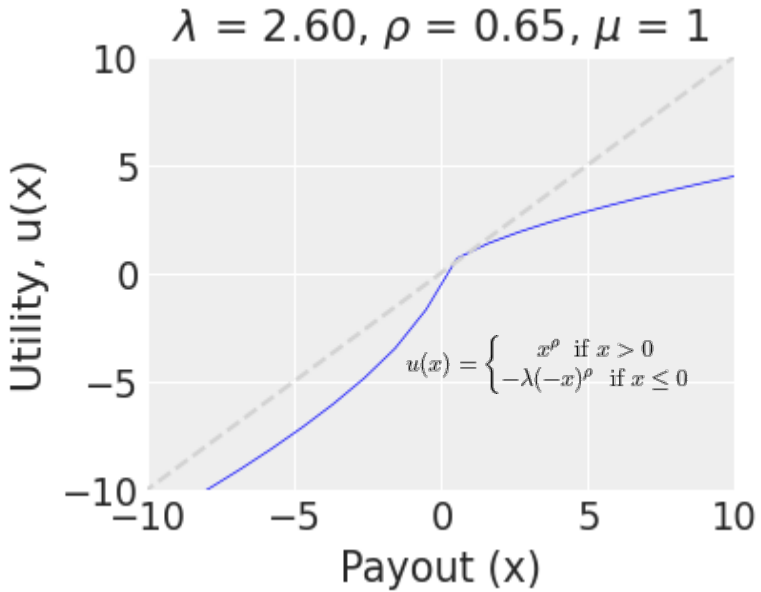
Posterior regression lines





Example 2 - Cumulative prospect theory model

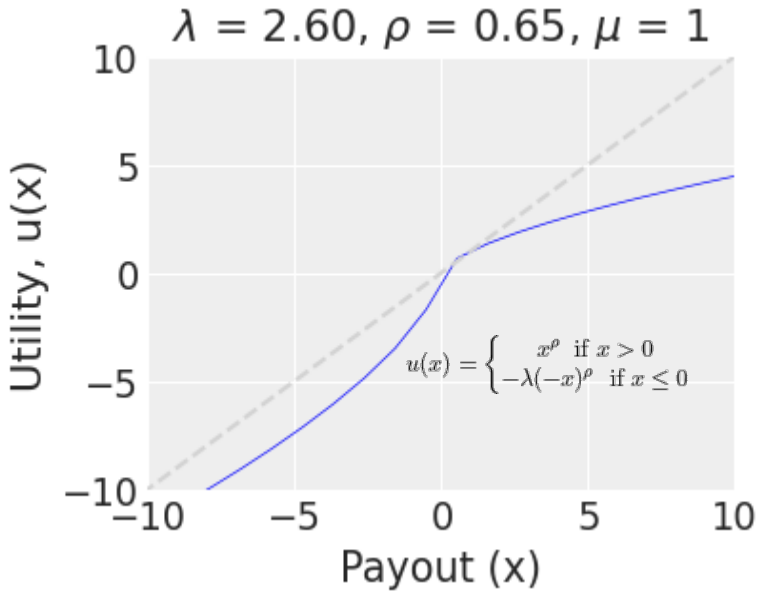
Game: Choose between 50/50 gamble or payout with certainty



Building on: www.thegreatstatsby.com

Example 2 - Cumulative prospect theory model

Game: Choose between 50/50 gamble or payout with certainty



$$u(reject) = u(certain)$$

$$u(accept) = (.5u(gain) + .5u(loss))$$

$$\Delta u = u(accept) - u(reject)$$

$$p(accept) = \frac{1}{1 + e^{-\mu\,\Delta u}}$$

Building on: www.thegreatstatsby.com

Cumulative prospect theory model: PP implementation

$$u(x) = \begin{cases} x^{\rho} & \text{if } x > 0 \\ -\lambda(-x)^{\rho} & \text{if } x \leq 0 \end{cases}$$

$$u(reject) = u(certain)$$

$$u(accept) = (.5u(gain) + .5u(loss))$$

$$\Delta u = u(accept) - u(reject)$$

$$p(accept) = \frac{1}{1 + e^{-\mu\ \Delta u}}$$

```python
def utility(x, lam, rho):
    ...
def model_PT(gain, loss, cert, took_gamble=None):
    # Define priors
    lam = numpyro.sample('lam', dist.TruncatedNormal(loc=2, scale=1.0,low=1., high=4.))
    rho = numpyro.sample('rho', dist.TruncatedNormal(loc=1, scale=1.0,low=0.5, high=1.))
    mu = numpyro.sample('mu', dist.Uniform(0.5, 1.5))
    # Calculate utility of gamble and certain option
    util_reject =  utility(cert, lam, rho)
    util_accept = 0.5 * utility(gain, lam, rho) + 0.5 * utility(loss, lam, rho)
    util_diff = util_accept - util_reject
    # Calculate probability of accepting gamble
    p_accept = 1/(1+jnp.exp(-mu*util_diff))
    # Choice to take gamble
    numpyro.sample('took_gamble', dist.BernoulliProbs(p_accept), obs=took_gamble)
```

Building on: www.thegreatstatsby.com

Cumulative prospect theory model: PP implementation

Definition of prior distributions including restrictions to theoretically plausible ranges

$$u(x) = \begin{cases} x^{\rho} & \text{if } x > 0 \\ -\lambda(-x)^{\rho} & \text{if } x \leq 0 \end{cases}$$

$$u(reject) = u(certain)$$

$$u(accept) = (.5u(gain) + .5u(loss))$$

$$\Delta u = u(accept) - u(reject)$$

$$p(accept) = \frac{1}{1 + e^{-\mu\ \Delta u}}$$

```
def utility(x, lam, rho):
    ...
def model_PT(gain, loss, cert, took_gamble=None):
    # Define priors
    lam = numpyro.sample('lam', dist.TruncatedNormal(loc=2, scale=1.0,low=1., high=4.))
    rho = numpyro.sample('rho', dist.TruncatedNormal(loc=1, scale=1.0,low=0.5, high=1.))
    mu = numpyro.sample('mu', dist.Uniform(0.5, 1.5))
    # Calculate utility of gamble and certain option
    util_reject =  utility(cert, lam, rho)
    util_accept = 0.5 * utility(gain, lam, rho) + 0.5 * utility(loss, lam, rho)
    util_diff = util_accept - util_reject
    # Calculate probability of accepting gamble
    p_accept = 1/(1+jnp.exp(-mu*util_diff))
    # Choice to take gamble
    numpyro.sample('took_gamble', dist.BernoulliProbs(p_accept), obs=took_gamble)
```

Building on: www.thegreatstatsby.com

Cumulative prospect theory model: PP implementation

Definition of prior distributions including restrictions to theoretically plausible ranges

$$u(x) = \begin{cases} x^{\rho} & \text{if } x > 0 \\ -\lambda(-x)^{\rho} & \text{if } x \leq 0 \end{cases}$$

$$u(reject) = u(certain)$$

$$u(accept) = (.5u(gain) + .5u(loss))$$

$$\Delta u = u(accept) - u(reject)$$

$$p(accept) = \frac{1}{1 + e^{-\mu\,\Delta u}}$$

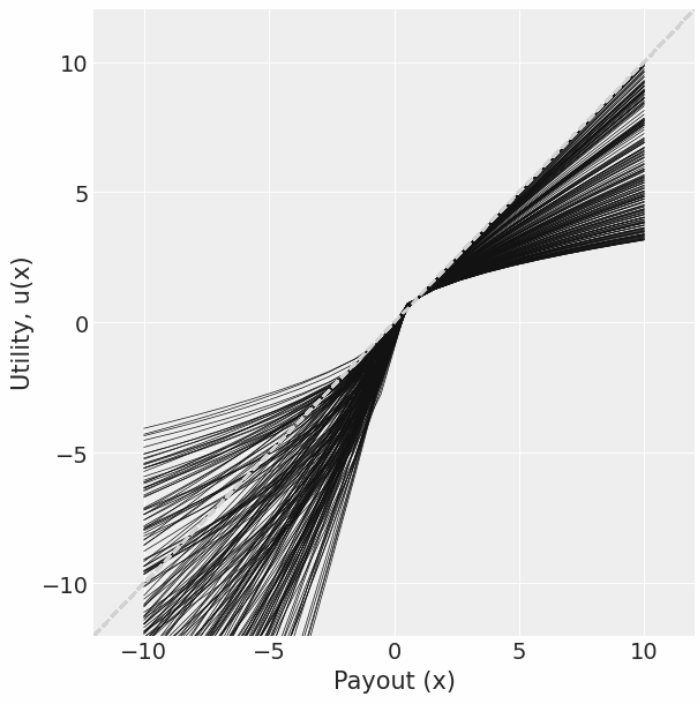
```
def utility(x, lam, rho, mu):
    ...
def model_PT(gain, loss, cert, took_gamble=None):
    # Define priors
    lam = numpyro.sample('lam', dist.TruncatedNormal(loc=2, scale=1.0,low=1., high=4.))
    rho = numpyro.sample('rho', dist.TruncatedNormal(loc=1, scale=1.0,low=0.5, high=1.))
    mu = numpyro.sample('mu', dist.Uniform(0.5, 1.5))
    # Calculate utility of gamble and certain option
    util_reject =  utility(cert, lam, rho)
    util_accept = 0.5 * utility(gain, lam, rho) + 0.5 * utility(loss, lam, rho)
    util_diff = util_accept - util_reject
    # Calculate probability of accepting gamble
    p_accept = 1/(1+jnp.exp(-mu*util_diff))
    # Choice to take gamble
    numpyro.sample('took_gamble', dist.BernoulliProbs(p_accept), obs=took_gamble)
```

Sampling statement for accept / reject decision

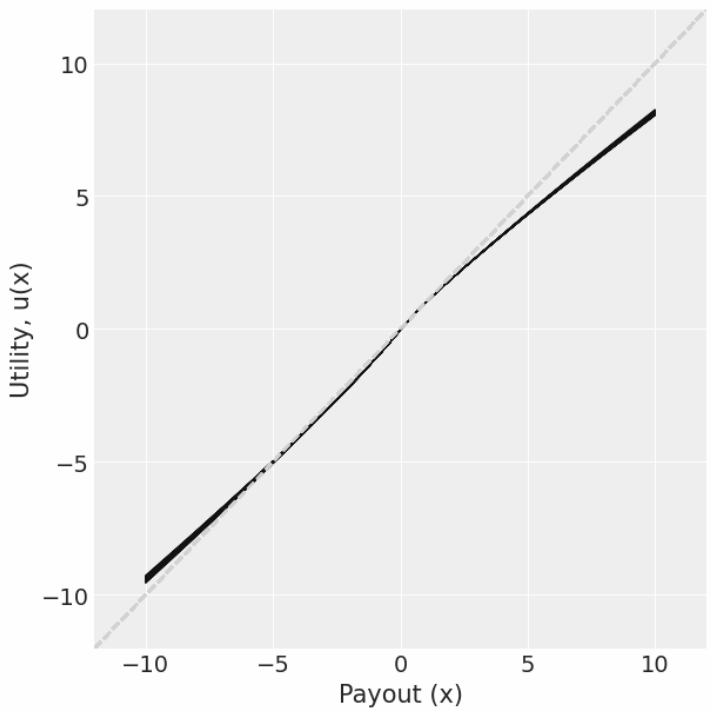
Building on: www.thegreatstatsby.com

Cumulative prospect theory model - prior vs. posterior

Prior utility functions



Posterior utility functions



Building on: www.thegreatstatsby.com

Example 3 - Potential outcome framework (Imbens 2020)

- Heterogeneous treatment effects depending on X
- Selection into treatment based on expected treatment effect

$$E[Y_i(0)] = \mathbf{x_i'}\alpha$$ Expected outcome without treatment

$$E[\tau_i] = \mathbf{x_i'}\beta$$ Expected (linear) treatment effect

$$E[Y_i(1)] = E[Y_i(0)] + E[\tau_i]$$ Expected outcome with treatment

Example 3 - Potential outcome framework

- Heterogeneous treatment effects depending on X
- Selection into treatment based on expected treatment effect

$E[Y_i(0)] = \mathbf{x_i'}\alpha$ Expected outcome without treatment

$E[\tau_i] = \mathbf{x_i'}\beta$ Expected (linear) treatment effect

$E[Y_i(1)] = E[Y_i(0)] + E[\tau_i]$ Expected outcome with treatment

$P(T_i = 1) = logit^{-1}(E[Y_i(1)] - E[Y_i(0)])$ Probability of treatment

$Y_{i,obs} \sim \mathcal{N}(T * E[Y_i(1)] + (1 - T) * E[Y_i(0)], \sigma_Y)$ Distribution of Y

Potential outcome framework - PP implementation

Prior distributions on
α, **β**, **σ**

```
def modelPotOutcome(X, T=None, Y=None):
    alpha = numpyro.sample("alpha",
dist.Normal(0.,1).expand([X.shape[1]]))
    beta = numpyro.sample("beta", dist.Normal(0.,1).expand([X.shape[1]]))
    sigma_Y = numpyro.sample("sigma_Y", dist.Exponential(1))

    EY0 = X @ alpha
    Etau = X @ beta
    EY1 = Y0 + tau
    T = numpyro.sample("T", dist.Bernoulli(logits=Y1 - Y0), obs=T)
    numpyro.sample("Y", dist.Normal(Y1*T + Y0*(1-T), sigma_Y), obs=Y)
```

$$E[Y_i(0)] = \mathbf{x_i'}\alpha$$
$$E[\tau_i] = \mathbf{x_i'}\beta$$
$$E[Y_i(1)] = E[Y_i(0)] + E[\tau_i]$$
$$P(T_i = 1) = logit^{-1}(E[Y_i(1)] - E[Y_i(0)])$$
$$Y_{i,obs} \sim \mathcal{N}(T * E[Y_i(1)] + (1 - T) * E[Y_i(0)], \sigma_Y)$$

Non-linear treatment effects

- Straightforward in PP to replace linear effect with non-linear functions

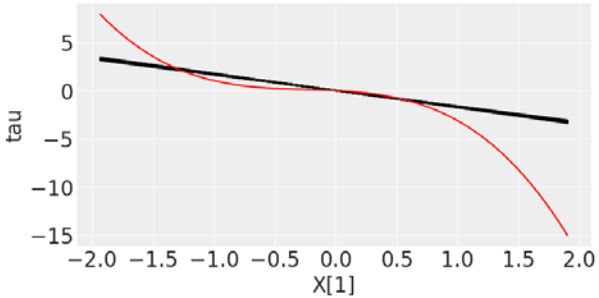
$$E[\tau_i] = \mathbf{x_i'}\beta \quad \Rightarrow \quad E[\tau_i] = f\left(\mathbf{x_i}\right)$$

- Those might even be flexible deep Neural Networks (includes training with dropout, batch learning, flexible number of layer/neurons)

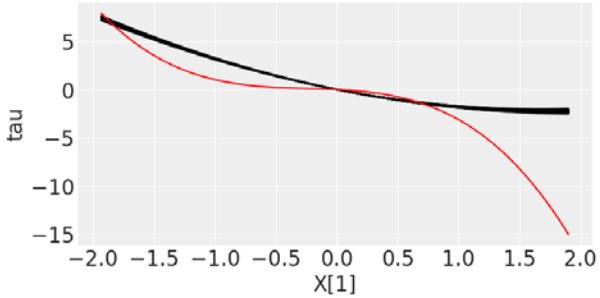
Knowing the functional form always helps, but…

True model is a **cubic treatment effect** (red line)

Linear treatment model

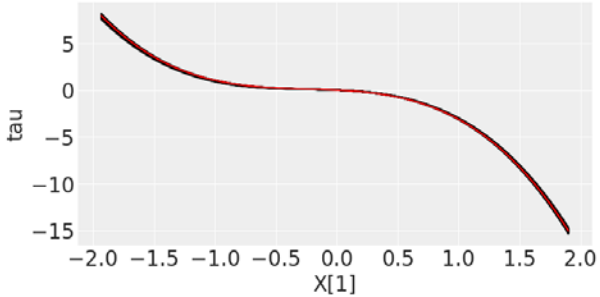


Squared treatment model

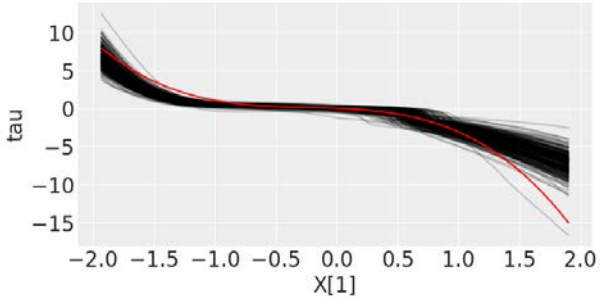


Limited functional form flexibility with potentially large errors

Cubic treatment model



"Deep" MLP treatment model

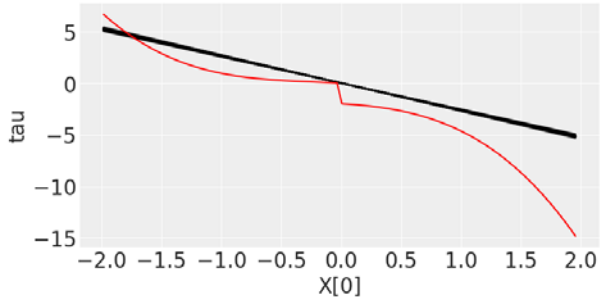


Knowing the functional form is best, but if we don't, ML techniques can be very useful

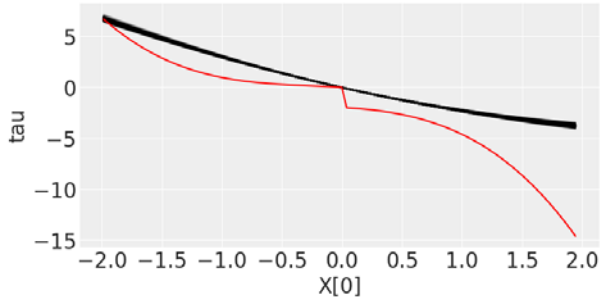
And even more non-linear treatment

True model is a cubic treatment effect **with a step function** (red line)

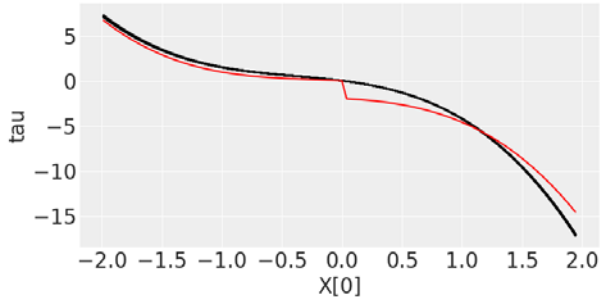
Linear treatment model



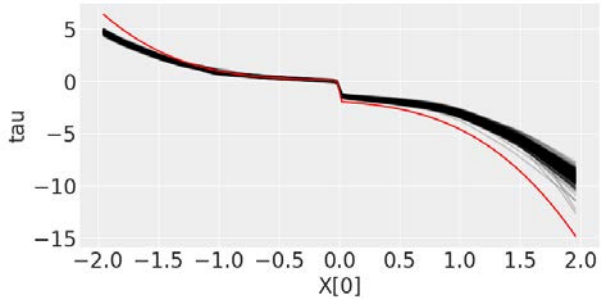
Squared treatment model



Cubic treatment model



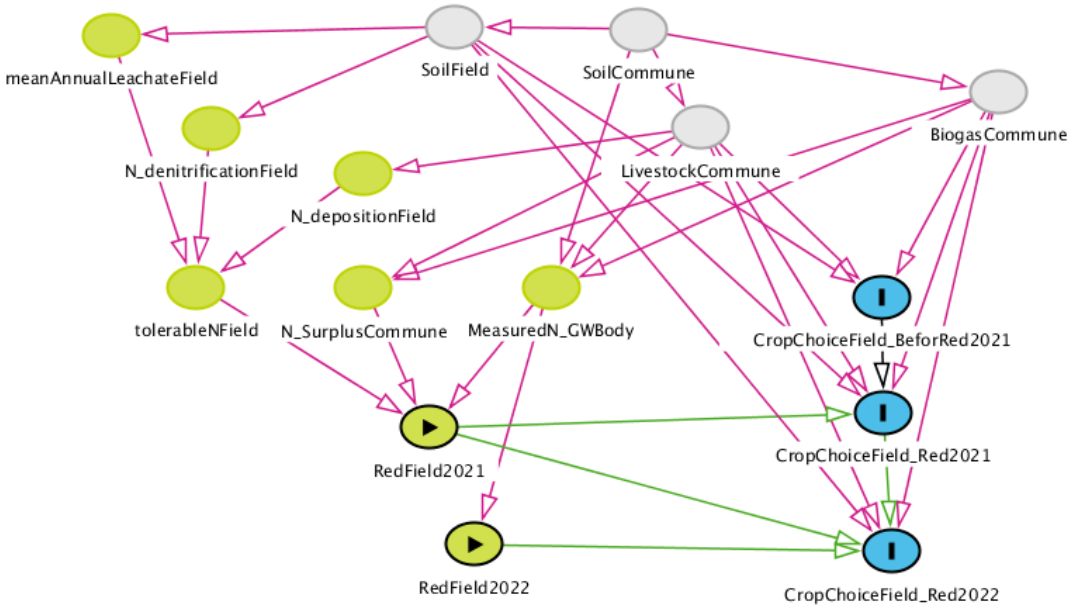
“Deep” NN (MLP) treatment



ML techniques can be quite amazing - here as part of a statistical, causal identification framework

Some additions on explicit formulation of DGPs

- More complex, individual data generating processes can be addressed with PP
- Raises awareness of identification issues
- Allows model testing and comparison
- In preparation, it made us unexpectedly think about the source of uncertainty in ‘standard’ models as well



Kuhn et al. 2023

Take home messages, code, poster

- Bayesian statistical analysis offers intuitive inference directly reflecting uncertainty
- Advances facilitate semi-automatic Bayesian analysis via Probabilistic Programming
- Explicitly formulating a generative model helps us to focus on the economics
- …and fosters statistical inference embedded in a holistic research design process
- Machine learning tools can contribute most to Econometrics in causal frameworks
- We need updating of econometrics curricula

https://github.com/hstorm/pp_eaae_rennes

Go Bayesian!

References

Athey, S., Imbens, G.W. 2019. “Machine Learning Methods That Economists Should Know About”.Annual Review of Economics 11(1), 685-725.

Blei, D.M.,Kucukelbir, A.,McAuliffe, J.D.. 2017. “Variational Inference: A Review for Statisticians.” Journal of the American Statistical Association 112 (518): 859–77.

Frisch, D., Hanebeck, U. 2020. “Progressive Bayesian Filtering with Coupled Gaussian and Dirac Mixtures”. 1-8. http://dx.doi.org/10.23919/FUSION45008.2020.9190540.

Ghahramani, Z. 2015. “Probabilistic Machine Learning and Artificial Intelligence.” Nature 521 (7553): 452–59.

Gigerenzer G., 2004. “Mindless Statistics”:.The Journal of Socio-Economics 33(5), 587-606.

Gigerenzer G., 2018. “Statistical Rituals: The Replication Delusion and How We Got There”. Advances in Methods and Practices in Psychological Science. 2018;1(2):198-218. doi:10.1177/2515245918771329

Heckelei, T., Hüttel, S., Odening, M., Rommel, J. 2023: “The p-Value Debate and Statistical (Mal)practice - Implications for the Agricultural and Food Economics Community”. German Journal of Agricultural Economics 72(1), 47-67.

Imbens, G. W. 2020. "Potential Outcome and Directed Acyclic Graph Approaches to Causality: Relevance for Empirical Practice in Economics." Journal of Economic Literature, 58 (4): 1129-79.

Kuhn, T., Pahmeyer, C., Storm, H. 2023. “Using Probabilistic Programming to Assess Policy Induced Adaptation of Crop Choices: A Case Study of the German Implementation of the EU Nitrates Directive. Poster presented at the XVII Congress of the European Association of Agricultural Economists, Rennes

McElreath, R. 2020. “Statistical Rethinking: A Bayesian Course with Examples in R and Stan”. Chapman and Hall/CRC.

Meent, van de, J.-W, Paige, B., Yang,H., Wood, F. 2018. “An Introduction to Probabilistic Programming.” arXiv [stat.ML]. arXiv. http://arxiv.org/abs/1809.10756.

Pahmeyer, C. 2021. “Estimating plot specific yields”. https://observablehq.com/d/3a2dd11e01f67dfb

Stillman, P.R. 2021.”Estimating Prospect Theory Parameters with Maximum Likelihood in R. The Great Statsby, https://www.thegreatstatsby.com/posts/2021-03-08-ml-prospect/. .