Bayesian Causal Inference via Probabilistic Program Synthesis

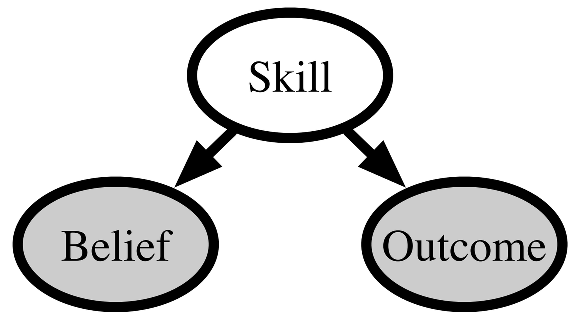
Sam Witty[1]*, Alexander Lew[2]*, David Jensen[1], Vikash Mansinghka[2]
[1]UMass Amherst Knowledge Discovery Lab, [2]MIT Probabilistic Computing Project
*Equal Contribution

1. Introduction

How can we learn causal models from diverse sources of observational and experimental data?

Causal Graph		Observational MiniStan Program	Encouragement Design MiniStan Program

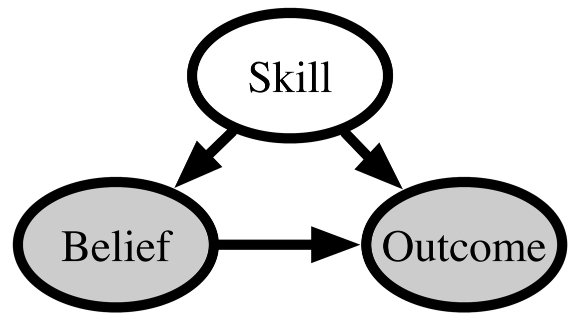
H1: "Only skill matters"




```
quote
  s ∼ normal(mu_s, sigma_s)
  b ∼ normal(s, sigma_b)
  logit_o = s * lambda_so
  o ∼ bernoulli(1/(1+exp(-logit_o)))
end
```

```
quote
  s ∼ normal(mu_s, sigma_s)
  b ∼ normal(s + 3, sigma_b)
  logit_o = s * lambda_so
  o ∼ bernoulli(1/(1+exp(-logit_o)))
end
```

H2: "Belief and skill matter"




```
quote
  s ∼ normal(mu_s, sigma_s)
  b ∼ normal(s, sigma_b)
  logit_o = s * lambda_so + b * lambda_bo
  o ∼ bernoulli(1/(1+exp(-logit_o)))
end
```

```
quote
  s ∼ normal(mu_s, sigma_s)
  b ∼ normal(s + 3, sigma_b)
  logit_o = s * lambda_so + b * lambda_bo
  o ∼ bernoulli(1/(1+exp(-logit_o)))
end
```

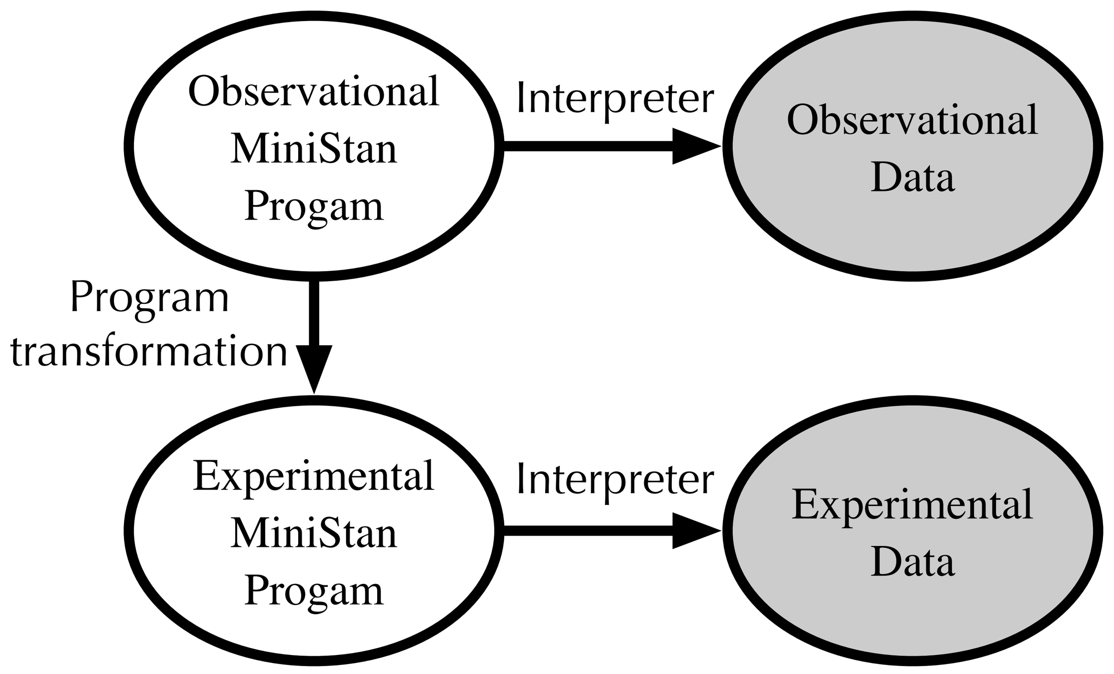
- Stochastically generate MiniStan programs [5] to encode a prior distribution over causal model structures and parameters.
- Programmatically edit the generated MiniStan programs to reflect experimental conditions.
- Interpret the MiniStan programs to induce a likelihood given observational and experimental data.

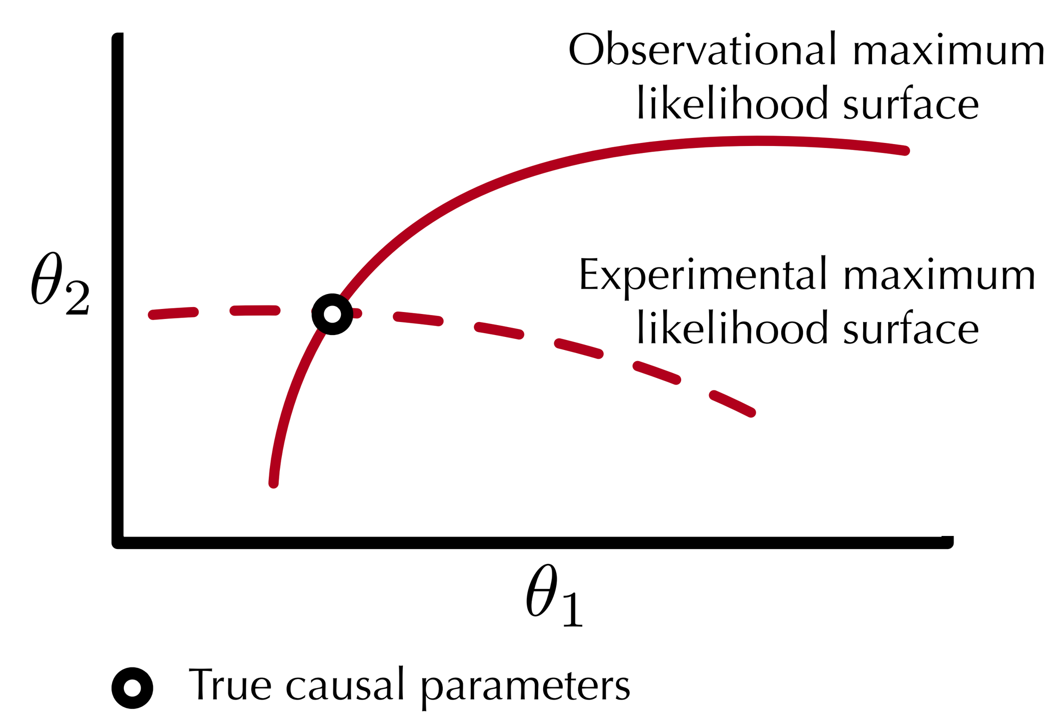
2. Interventions

Experiments can be represented as the composition of MiniStan program transformations.

```
function applyShiftIntervention(program, var, shift)
  walk(program) do expr
    @match expr begin
      :($x ∼ normal($mean, $std)) && if x == var end => :($x ∼ normal($mean + $shift, $std))
      :($x ∼ uniform($a, $b)) && if x == var end => :($x ∼ uniform($a + $shift, $b + $shift))
      :($x = $value) && if x == var end => :($x = $value + $shift)
      _ => expr
    end
  end
end
```

```
function applyDoIntervention(program, var, newValue)
  walk(program) do expr
    @match expr begin
      :($x = $val)  && if x == var end => :($var = $newValue)
      :($x ∼ $dist) && if x == var end => :($var = $newValue)
      _ => expr
    end
  end
end
```





3. Generative model over MiniStan Programs

MiniStan Program Prior

```
@gen function generate_causal_model()
  mu_s = @trace(normal(0, 1), :mu_s)
  sigma_s = @trace(uniform(0, 1), :sigma_s)
  sigma_b = @trace(uniform(0, 1), :sigma_b)
  lambda_so = @trace(uniform(0, 1), :so_weight)
  lambda_bo = @trace(uniform(0, 1), :bo_weight)
  edge = @trace(bernoulli(0.5), :edge)

  if edge
    logit_o_expr = quote s * $so_weight + b * $bo_weight end
  else
    logit_o_expr = quote s * $so_weight end
  end

  causal_model = quote
    s ∼ normal($mu_s, $sigma_s)
    b ∼ normal(s, $sigma_b)
    logit_o = $logit_o_expr
    o ∼ bernoulli(1/(1+exp(-logit_o)))
  end
  return causal_model
end
```

MiniStan Prior Samples

```
quote
  s ∼ normal(-0.592, 0.302)
  b ∼ normal(s, 0.724)
  logit_o = s * 0.503 + b * 0.491
  o ∼ bernoulli(1/(1 + exp(-logit_o)))
end
```

```
quote
  s ∼ normal(1.892, 0.108)
  b ∼ normal(s, 0.301)
  logit_o = s * 0.542
  o ∼ bernoulli(1/(1 + exp(-logit_o)))
end
```

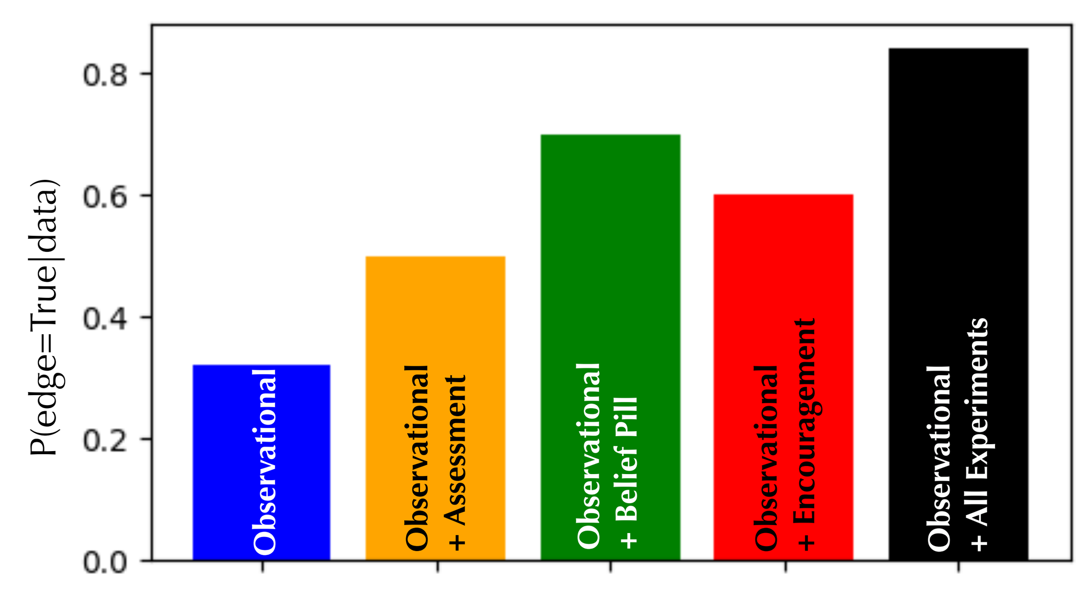
- **Belief pill.** Students are given a pill that sets their belief in their ability to a fixed value. The belief pill experiment is implemented using a single "do" intervention.
- **Encouragement.** Students are encouraged by their advisor, increasing their belief in their ability by a fixed value. The encouragement experiment is implemented using a single "shift" intervention.
- **Assessment.** Students are given an assessment which (i) reduces the student's uncertainty about their academic ability and (ii) increases their skill. The assessment experiment is implemented by composing a "variance scaling" intervention and a "shift" intervention.

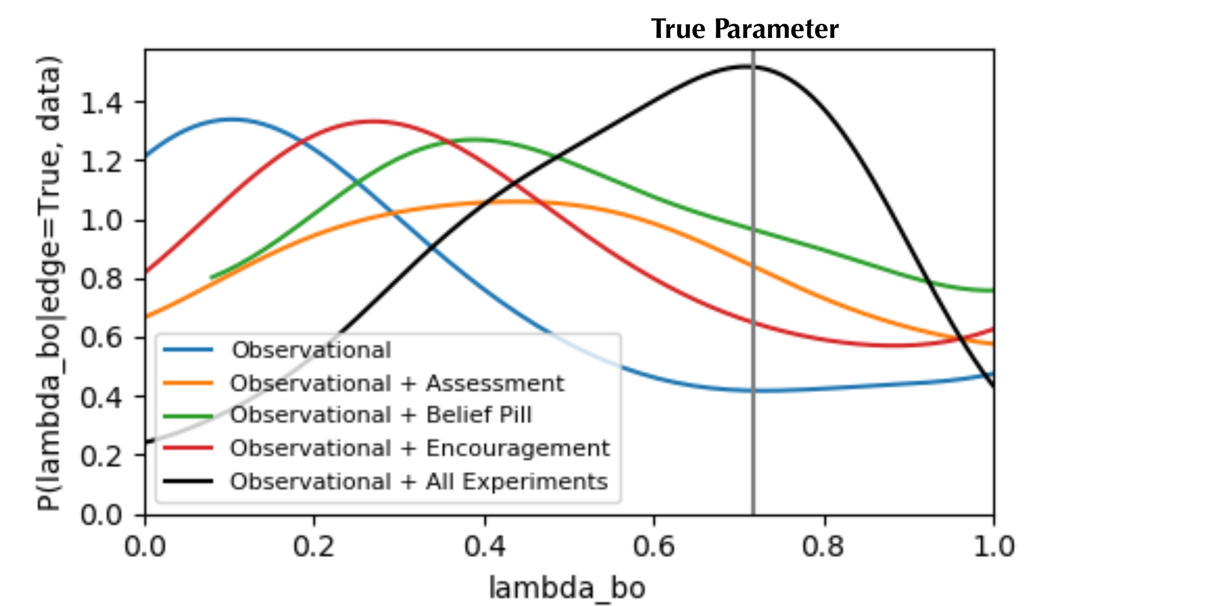
Observational and Experimental Data Likelihood

```
@gen function generate_data(NObs, NBeliefPill, NEncouragement, NAssessment)
  observational_model = @trace(generate_causal_model())
  belief_pill_model = applyDoIntervention(observational_model, :b, 5)
  encouragement_model = applyShiftIntervention(observational_model, :b, 3)
  assessment_model = applyVarianceScalingIntervention(applyShiftIntervention(observational_model, :s, 2),
                                                      :b, 1/100)

  observational_data = @trace(interpretMiniStan(observational_model, n_runs=NObs), :obs)
  belief_pill_data = @trace(interpretMiniStan(belief_pill_model, n_runs=NBeliefPill), :belief_pill)
  encouragement_data = @trace(interpretMiniStan(encouragement_model, n_runs=NEncouragement), :encouragement)
  assessment_data = @trace(interpretMiniStan(assessment_model, n_runs=NAssessment, :assessment)
end
```

4. Experiments and Discussion





- **Causal models are represented as code.** Causal programs can express many kinds of causal relationships, including context-dependent causal dependencies. ***Future work:*** Can more expressive grammars of causal programs [6] help us learn more realistic causal models of the world?
- **Prior beliefs are represented as code generators.** Naive priors over causal graphs fail to focus on a promising subset of the super-exponentially many hypotheses, and assume that all causal parameters are independent. ***Future work:*** How can the Bayesian synthesis approach be used to encode more richly structured priors [3, 8] over causal models using probabilistic programs?
- **Interventions/experiments are represented as code transformations.** The Bayesian synthesis approach to causal inference can be used to process data from any experiment that can be described in terms of programs that modify causal program source code. This differs from "do" interventions in causal graphs [4], or manually specified per-model experiment effects [2]. ***Future work:*** How can the Bayesian synthesis approach be used to model realistic [7] experimental conditions?

Acknowledgments — Thanks to Javier Burroni, Dan Garant, Zenna Tavares, and Reilly Grant for thoughtful discussion. This material is based upon work supported by the United States Air Force under Contract No. FA8750-17-C-0120 and a philanthropic gifts from the Siegil Family Foundation and the Aphorism Foundation. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

[1] Cusumano-Towner, Marco F., et al. "Gen: a general-purpose probabilistic programming system with programmable inference." *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2019.
[2] Foster, Adam, et al. "Variational Bayesian optimal experimental design." *Advances in Neural Information Processing Systems*. 2019.
[3] Mansinghka, V. K., et al. "Structured priors for structure learning." *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI 2006)*. 2006.
[4] Pearl, Judea. *Causality*. Cambridge university press, 2009.
[5] Saad, Feras A., et al. "Bayesian synthesis of probabilistic programs for automatic data modeling." *Proceedings of the ACM on Programming Languages* 3. POPL (2019): 1-32
[6] Tavares, Zenna, et al. "A language for counterfactual generative models." (2018).
[7] West, Stephen G., et al. "Alternatives to the randomized controlled trial." *American journal of public health* 98.8 (2008): 1359-1366.
[8] Witty, Sam A., et al. "Causal Inference using Gaussian Processes with Structured Latent Confounders." *International Conference on Machine Learning*. 2020.