Graph Tracking in Dynamic Probabilistic Programs via Source Transformations

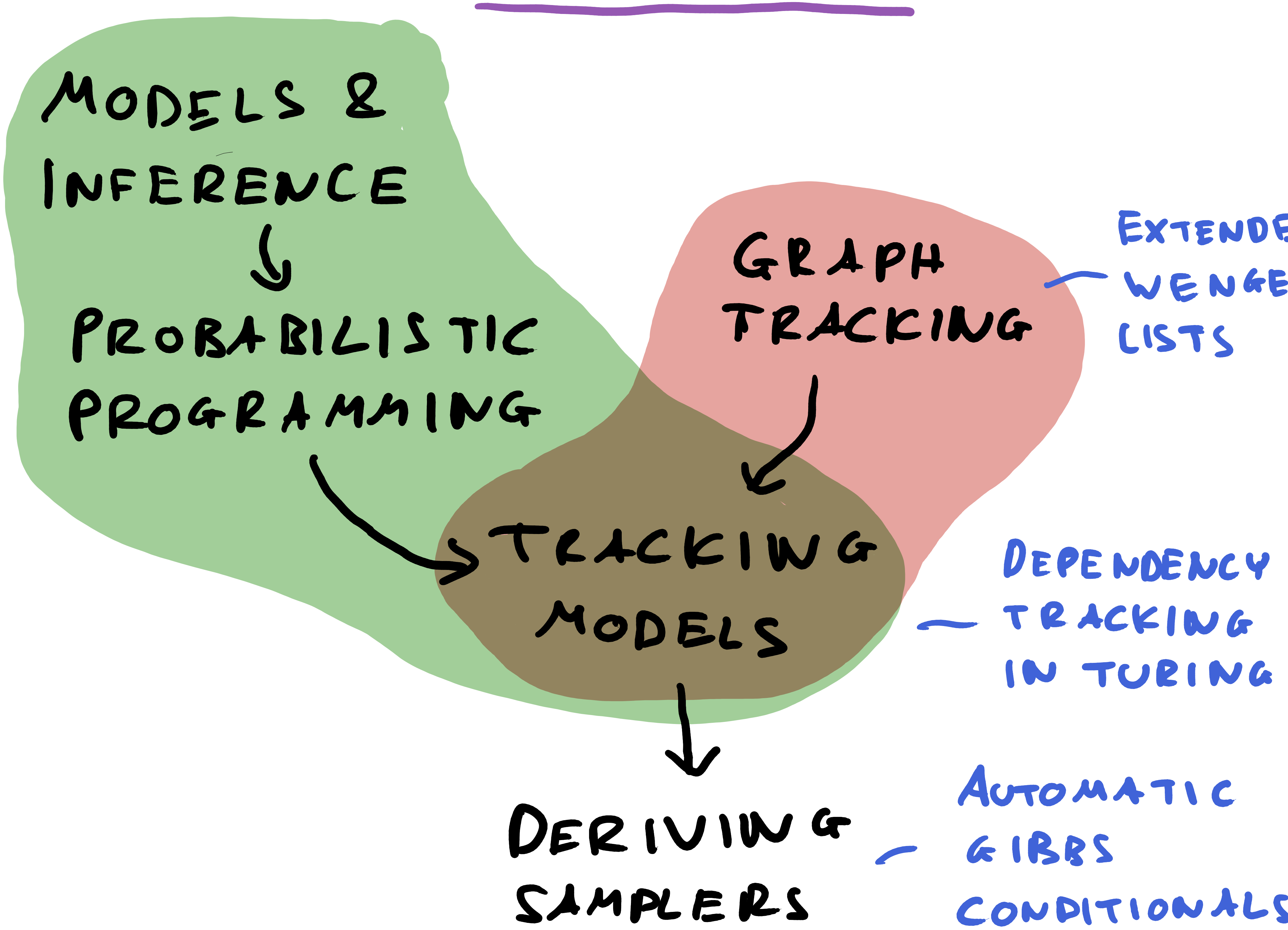
Philipp Gabler

2021-03-11

ABSTRACT

- Q: HOW TO ADD STRUCTURE TO TURING MODELS?
- OUTCOMES:
  - "EXTENDED WENGERT LIST" TRACKING TECHNIQUE
  - DEPENDENCY GRAPH EXTRACTION (SLICING) FROM TURING MODELS
  - AUTOMATIC GIBBS SAMPLER

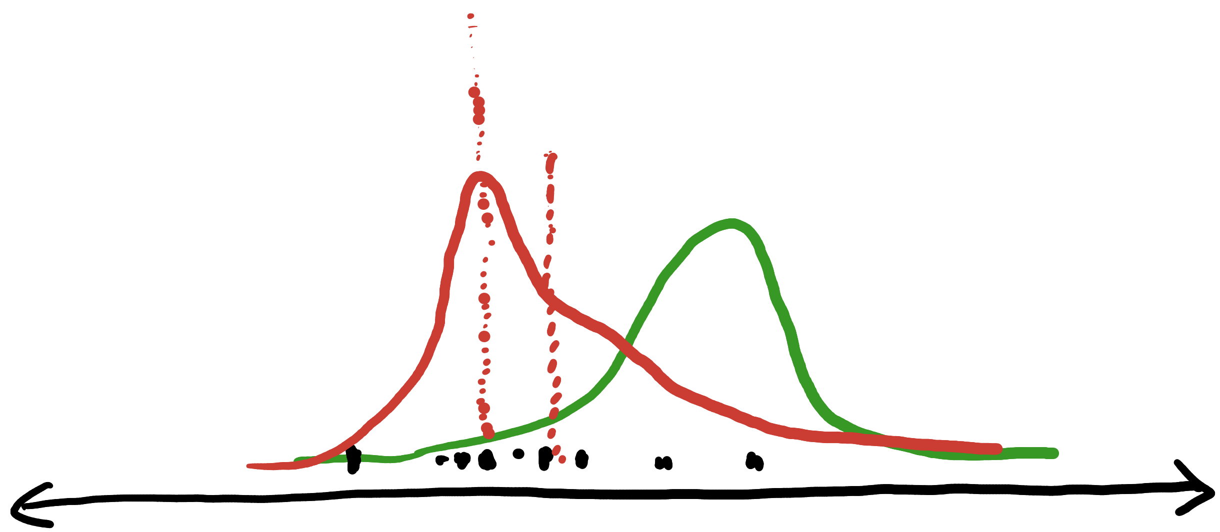
ROADMAP



MODELS & INFERENCE
PROBABILISTIC PROGRAMMING
GRAPH TRACKING
EXTENDED WENGERT LISTS
TRACKING MODELS
DEPENDENCY TRACKING IN TURING
DERIVING SAMPLERS
AUTOMATIC GIBBS CONDITIONALS

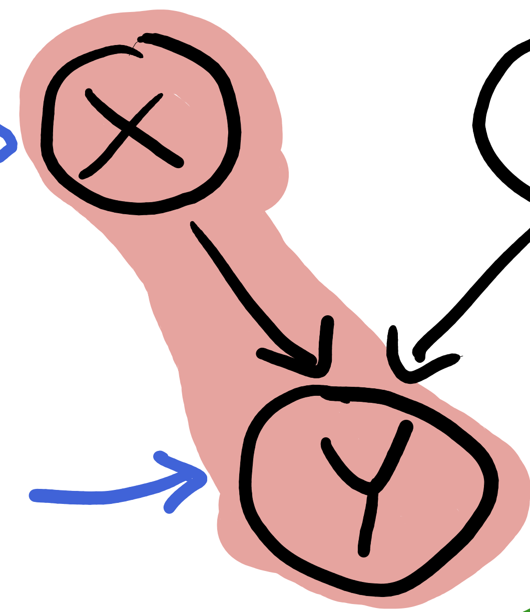
7

BAYES-ICS



GENERATIVE MODELS

INPUT



PARAMETERS, LATENT VARS, ...

OUTPUT

PRIOR

LIKELIHOOD

$$P(Y, \Theta \mid X) = P(\Theta)\, P(Y \mid X, \Theta)$$

GIVEN: GENERATIVE PROCESS

$$P(\Theta \mid D) = \frac{P(Y, \Theta \mid X)}{P(Y \mid X)}$$

WANTED: POSTERIOR

OBSERVED DATA

$D = \{(X_1, Y_1), \dots\}$

EVIDENCE

POSTERIOR EXPECTATION

DATA

- $\mathbb{E}[f(\theta) \mid D]$

POSTERIOR

$$= \int f(\vartheta)\, p(\vartheta \mid D)\, d\nu(\vartheta)$$

- INTERESTING QUANTITIES (OFTEN) ARE INTRACTABLE INTEGRALS!

E.G. PPD, CREDIBLE INTERVALS, MAP, MODEL COMPARISON

Approximate Inference

- VI: Minimize

$$D(q_\lambda(D) \,\|\, p(\vartheta \mid D))$$

- Sampling:

USE LLN!

$$I^{(N)}(f) = \frac{1}{N} \sum_n f(Y^{(n)})$$

$$\xrightarrow{a.s.} \mathbb{E}[f(\Theta) \mid D]$$

THE MARKOV CHAIN

- CONSTRUCT TRANSITION KERNEL:

$$\mathbb{P}[Y^{(k+1)} \in dy \mid Y^{(k)} = y^{(k)}, \dots, Y^{(1)} = y^{(1)}]$$
$$= K(y^{(k)}, dy)$$
$$\text{s.t. } \underbrace{p(\vartheta \mid D)\, K(\vartheta, S) = \mathbb{P}[\theta \in S \mid D] \ \forall S}_{\hat{=}\ \pi K = \pi}$$

- SCHEMA:
  - PROPOSE NEW $Y^{(i)}$ FROM $Y^{(i-1)}$
  - ACCEPT/REJECT

LIVE DEMO

KERNELS ≈ SAMPLERS

- METROPOLIS-HASTINGS, HMC, ...
- OFTEN EASY: GIBBS CONDITIONALS!

$$p(\vartheta_1 \mid \vartheta_2, D) \qquad p(\vartheta_2 \mid \vartheta_1, D)$$

NO ACCEPTANCE STEP!

- "WITHIN-GIBBS": MIX SAMPLERS PER VARIABLE

2

PROBABILISTIC PROGRAMMING

$$X \sim \mathcal{N}(\mu, \sigma_1)$$

$$Y \sim \mathcal{N}(X, \sigma_2)$$

PROBABILISTIC PROGRAMS I

HOW TO IMPLEMENT MODELS?

PARAMETERS

OBSERVED DATA

```
@model function hierarchical_gaussian(x)
    λ ~ Gamma(2.0, inv(3.0))
    m ~ Normal(0, sqrt(1 / λ))
    x ~ Normal(m, sqrt(1 / λ))
end
```

JULIA FUNCTIONS

PROBABILISTIC PROGRAMS II

```
@model function normal_mixture(x, K, m, s, σ)
    N = length(x)

    μ = Vector{Float64}(undef, K)
    for k = 1:K
        μ[k] ~ Normal(m, s)
    end

    z = Vector{Int}(undef, N)
    for n = 1:N
        z[n] ~ Categorical(K)
    end

    for n = 1:N
        x[n] ~ Normal(μ[z[n]], σ)
    end

    return x
end
```

PARAMETERS

DATA STRUCTURES

MUTATION

LOOPS / CONTROL FLOW

PPL CHARACTERISTICS

- DENSITY EVALUATION
- MODEL STRUCTURE
- AUTOMATIC DIFFERENTIATION
- SAMPLING, DATA GENERATION
- DIAGNOSTICS, EVALUATION
- PROGRAMMING & EXTERNAL LIBS
- COMPOSITION

PPL IMPLEMENTATIONS

METAPROGRAMMING

SOURCE → DENSITY PROGRAM → SAMPLER

RESTRICTED DYNAMICS

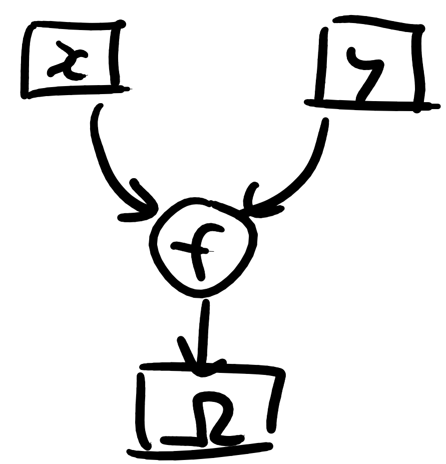
NO STRUCTURE

TRACED PROGRAM ⇄ SAMPLER

CALLBACKS

3

GRAPH TRACKING



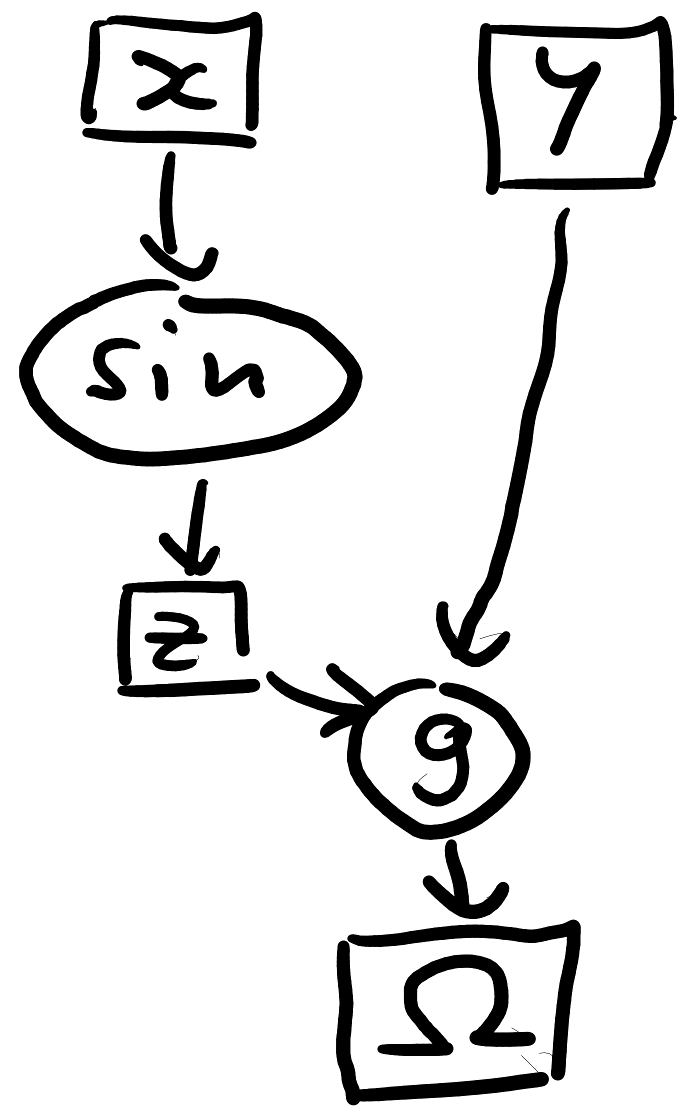
COMPUTATION GRAPHS

$$g(\sin(x), y)$$

EXPRESSION / PROGRAM

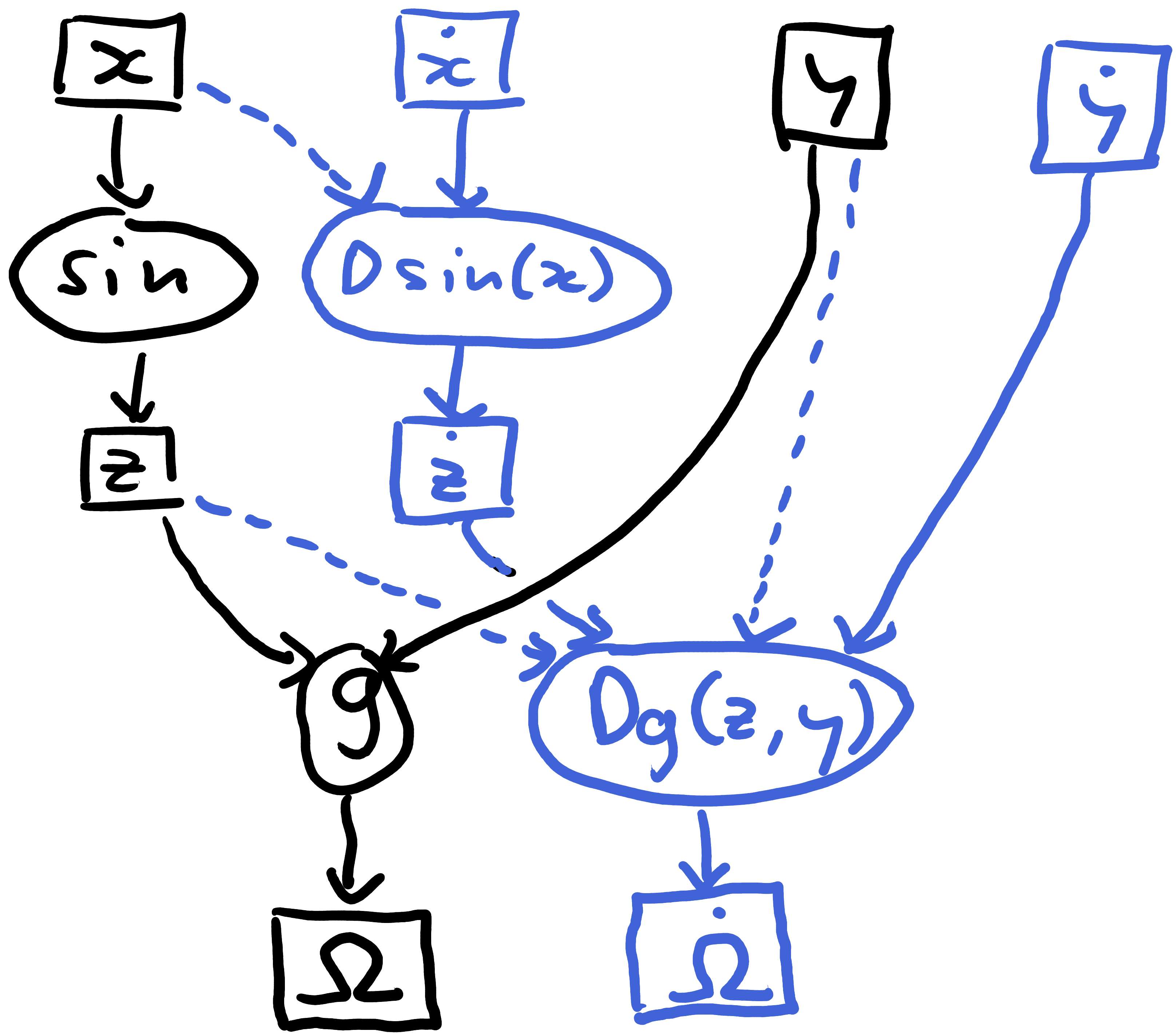
$$x = ?$$
$$y = ?$$
$$z = \sin(x)$$
$$\Omega = g(z, y)$$

SSA FORM



GRAPH

GRAPHS IN AD



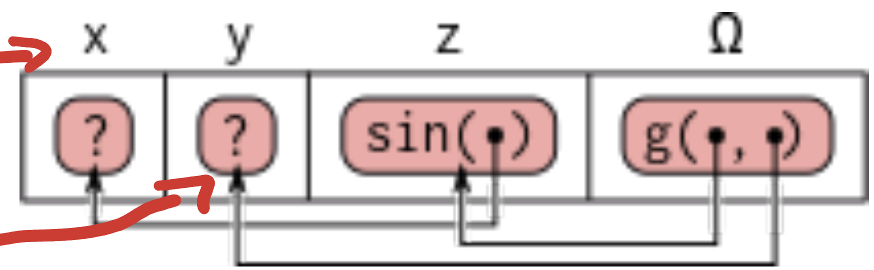
IMPLEMENTATON TECHNIQUES

- OPERATOR OVERLOADING:

$$x = \text{TRACKED}(?)$$

$$y = \text{TRACKED}(?)$$

$$\vdots$$



WENGERT LIST,
A.K.A. TAPE

- SOURCE TRANSFORMATION:

$$x = ? \qquad z = \sin(x)$$

$$\dot{x} = \delta_1 \qquad \dot{z} = D\sin(x)(\dot{x})$$

$$y = ? \qquad \Omega = g(z, y)$$

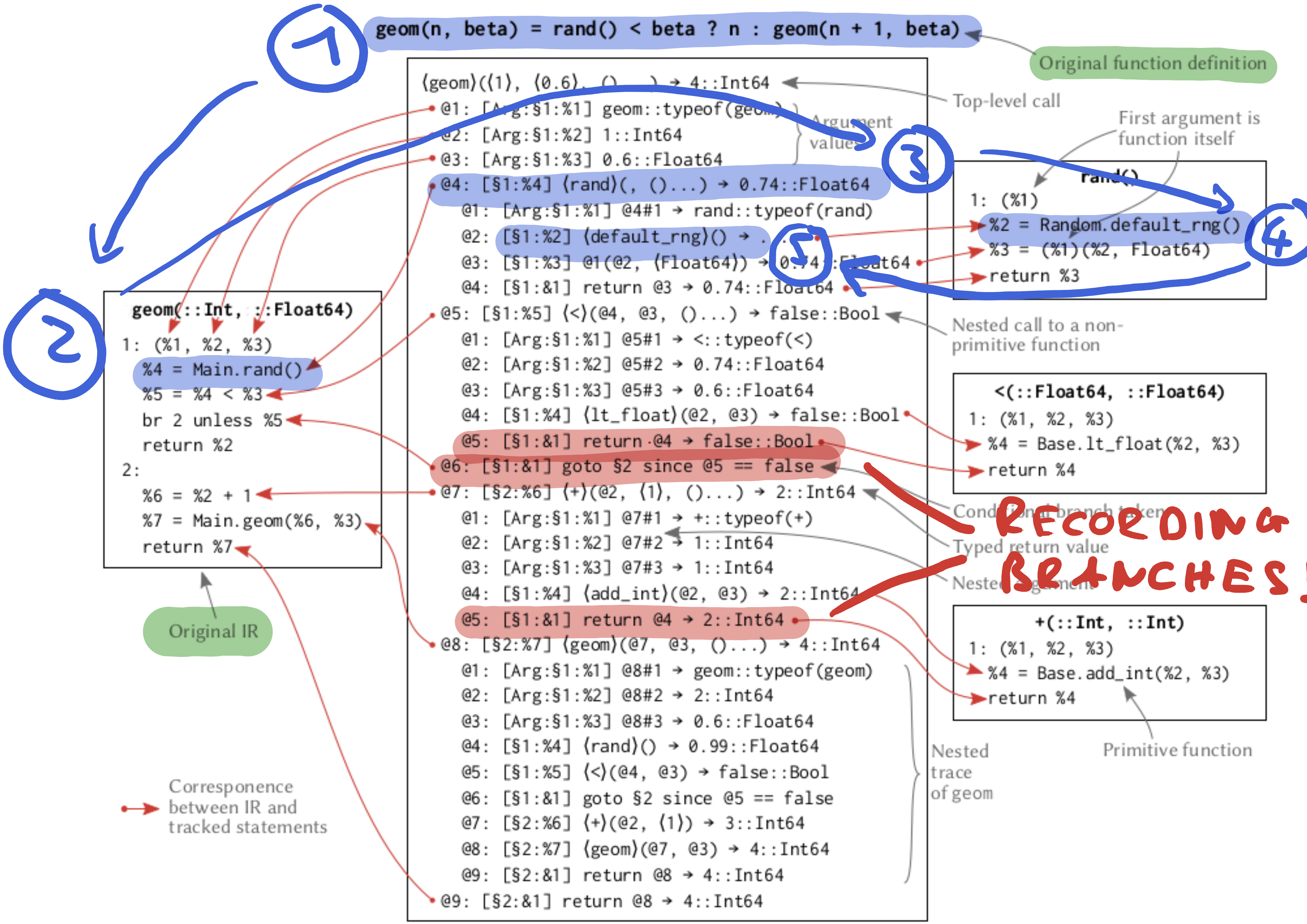
$$\dot{y} = \delta_2 \qquad \dot{\Omega} = Dg(z, y)(\dot{z}, \dot{y})$$

Julia IR

```
f(x,y) = y > 0 ? g(sin(x),y) : y
```

```
       f     x     y
1: (%1, %2, %3)                          3:
    %4 = %3 > 0                              return %3
    br 3 unless(%4)
2:
    %5 = sin(%2)         ← z = sin(x)
    %6 = g(%5, %3)       ← Ω = g(z,y)
    return %6
```

Extended Wengert List



IR TRANSFORMATION

TRACKED IR

```
%15 = record!(%5, %14)
%16 = TapeConstant(Main.rand)
%17 = Base.tuple()
%18 = trackedcall(%5, %16, %17, $(QuoteNode(§1:%4)))
%19 = record!(%5, %18)
%20 = TapeConstant(Main.:<)
%21 = trackedvariable(%5, $(QuoteNode(%4)), %19)
%22 = trackedvariable(%5, $(QuoteNode(%3)), %3)
%23 = Base.tuple(%21, %22)
%24 = trackedcall(%5, %20, %23, $(QuoteNode(§1:%5)))
%25 = record!(%5, %24)
%26 = Base.tuple()
%27 = trackedvariable(%5, $(QuoteNode(%5)), %25)
%28 = trackedjump(%5, 2, %26, %27, $(QuoteNode(§1:&1)))
%29 = trackedvariable(%5, $(QuoteNode(%2)), %2)
%30 = trackedreturn(%5, %29, $(QuoteNode(§1:&2)))
br 2 (%28) unless %25
br 3 (%2, %30)
```

Actual jump is recorded

ORIGINAL IR

```
1: (%1, %2, %3)
  %4 = Main.rand()
  %5 = %4 < %3
  br 2 unless %5
  return %2
```

Jumps and returns are passed down to the next block

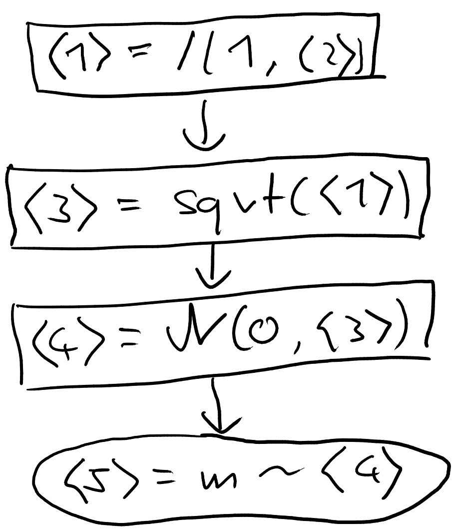
⋮

```
3: (%46, %47)
  %48 = record!(%5, %47)
  return %46
```

Special extra block for return values

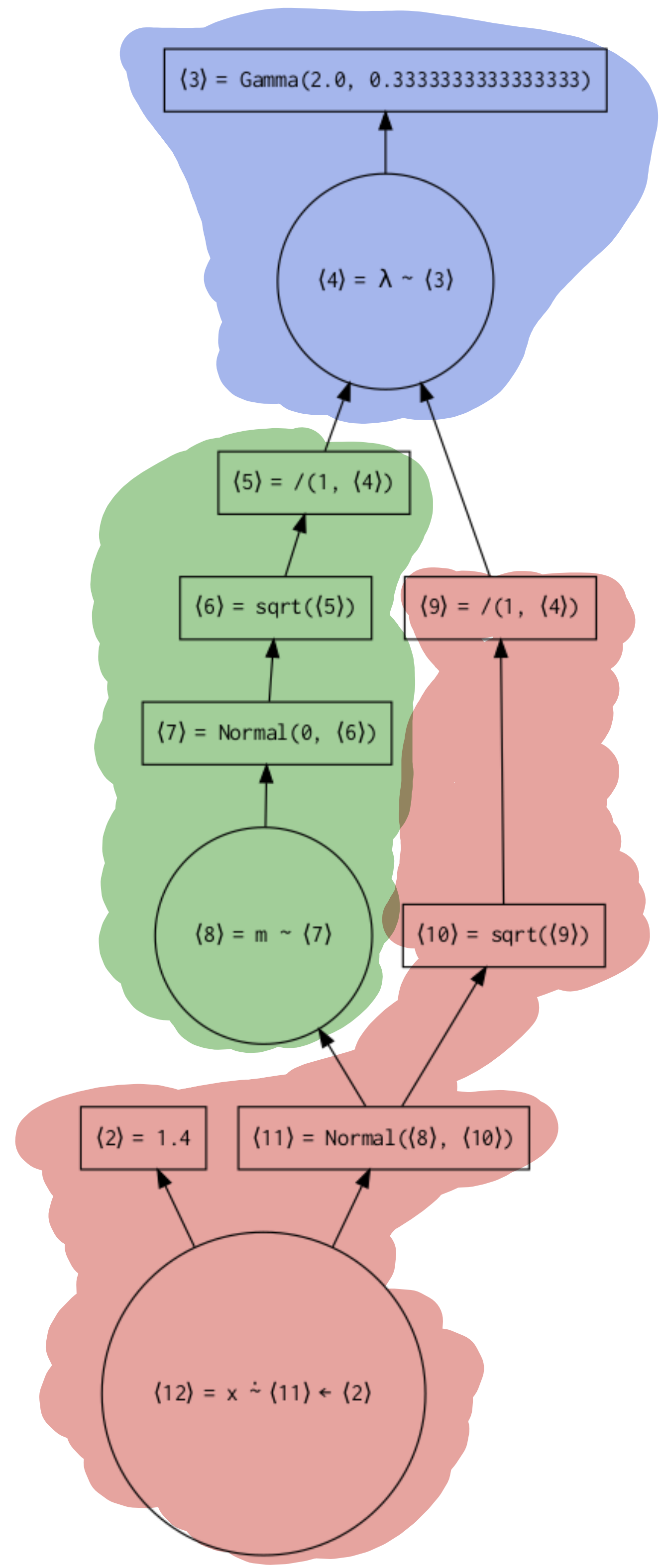
4

TRACKING TURING MODELS

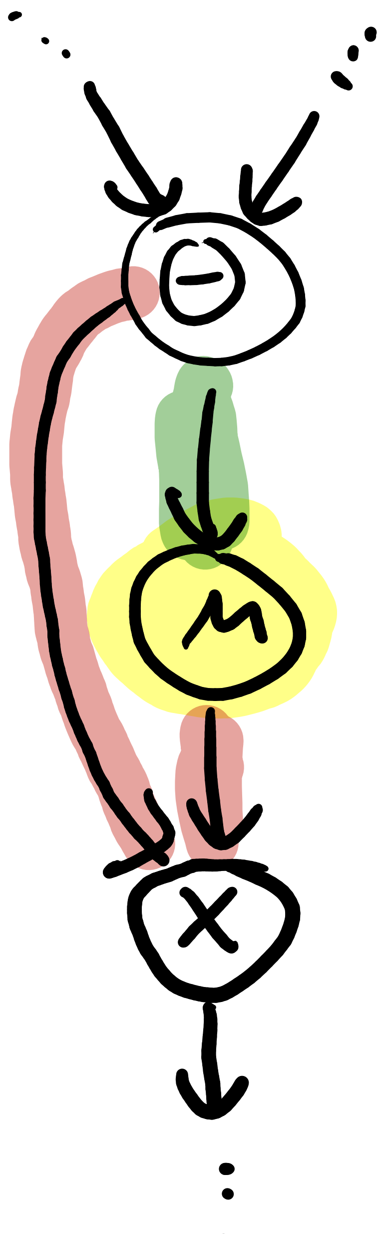


DEPENDENCY EXTRACTION

```
@model function hierarchical_gaussian(x)
    λ ~ Gamma(2.0, inv(3.0))
    m ~ Normal(0, sqrt(1 / λ))
    x ~ Normal(m, sqrt(1 / λ))
end
```



(DISCRETE) GIBBS CONDITIONALS



$$p(m, \vartheta, x)$$
$$= p(\vartheta)\, p(m \mid \vartheta)\, p(x \mid m, \vartheta)$$

MARKOV BLANKET

$$p(m \mid \vartheta, x) = \frac{p(m \mid \vartheta)\, p(x \mid m, \vartheta)}{Z}$$

$$Z = \sum_m p(m \mid \vartheta)\, p(x \mid m, \vartheta)$$

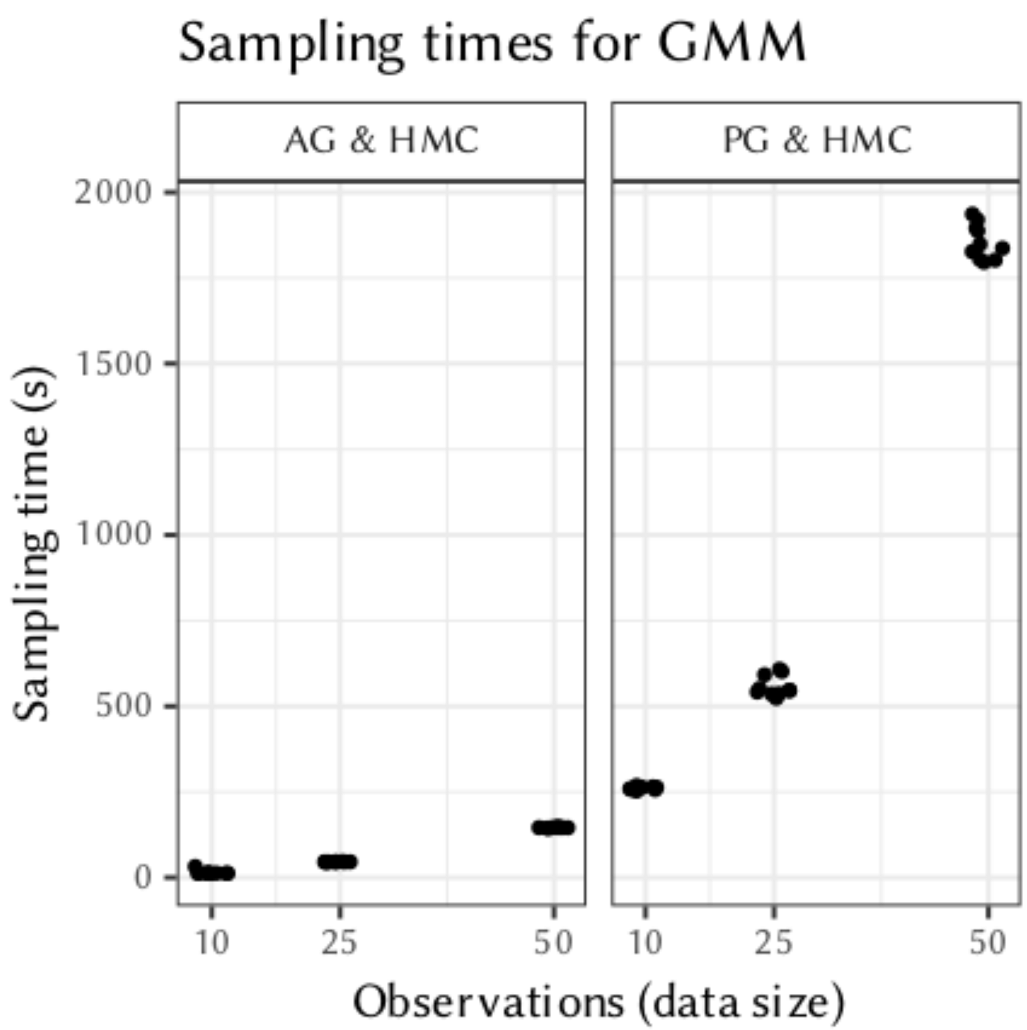
A Test Model

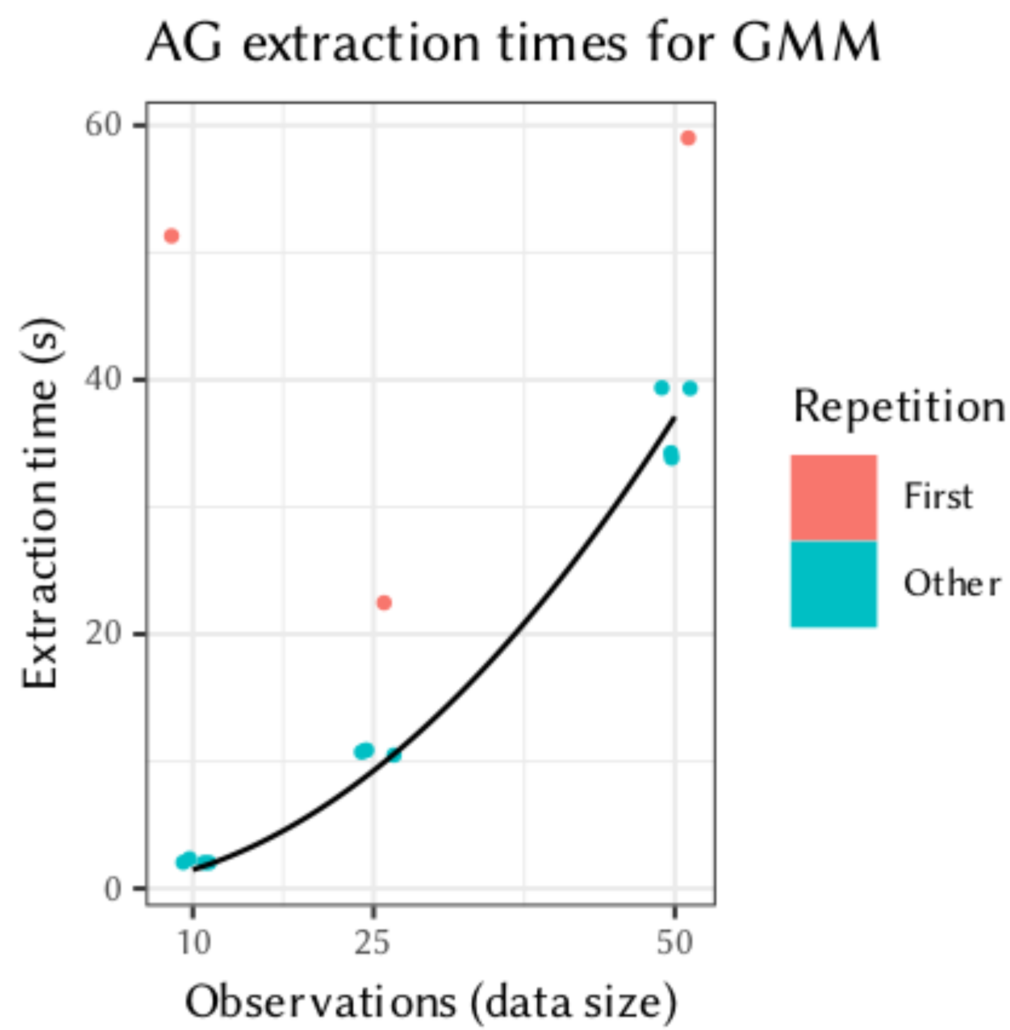
```
@model function gmm(x, K)
    N = length(x)
    w ~ Dirichlet(K, 1/K)  # Cluster association prior
    z ~ filldist(Categorical(w), N)  # Cluster assignments
    μ ~ filldist(Normal(0.0, s1_gmm), K)  # Cluster centers

    for n = 1:N
        x[n] ~ Normal(μ[z[n]], s2_gmm)  # Observations
    end
end
```

z: AutoGibbs vs. Particle Gibbs

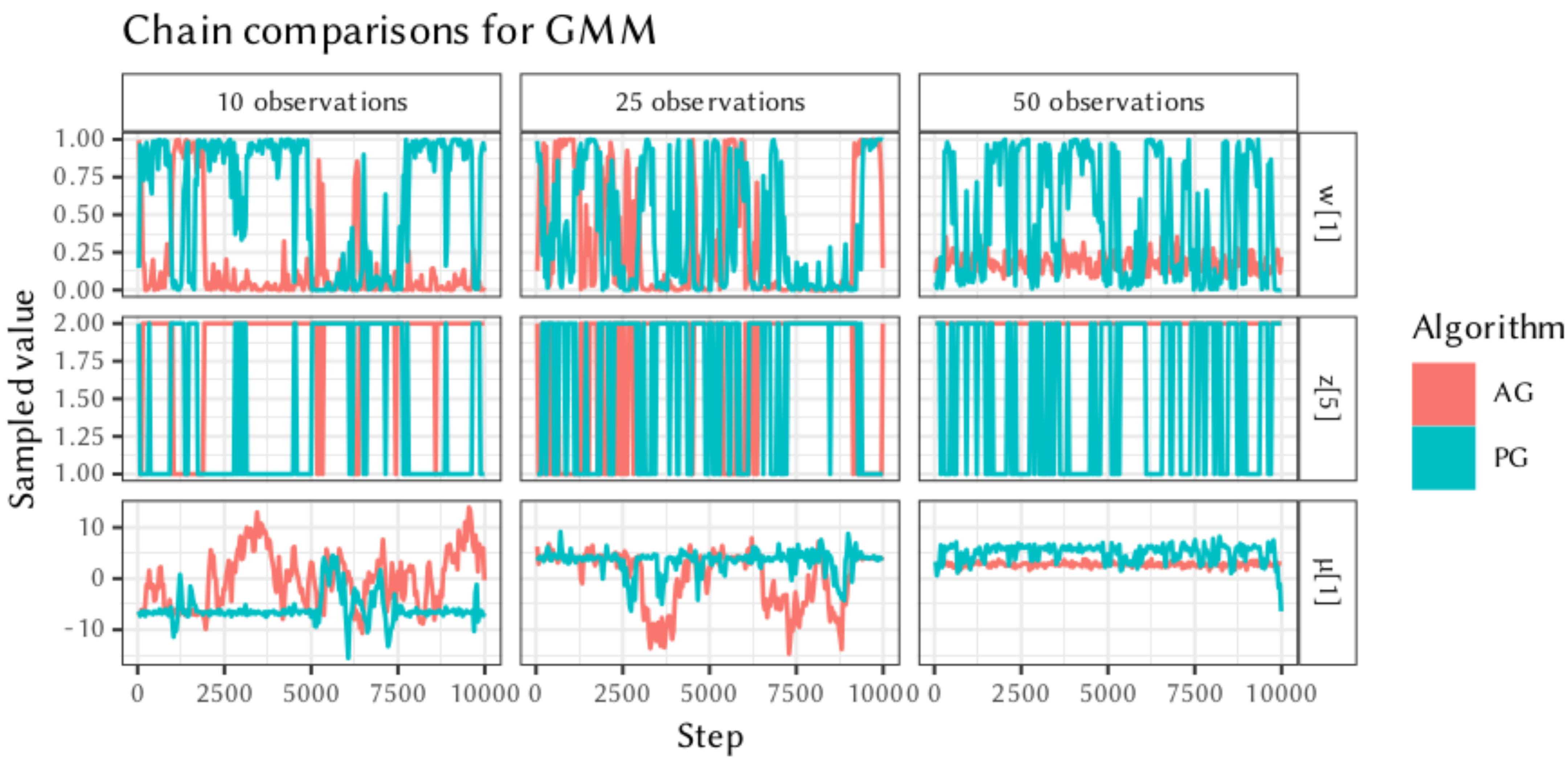
w, μ: HMC

EVALUATION: TIME

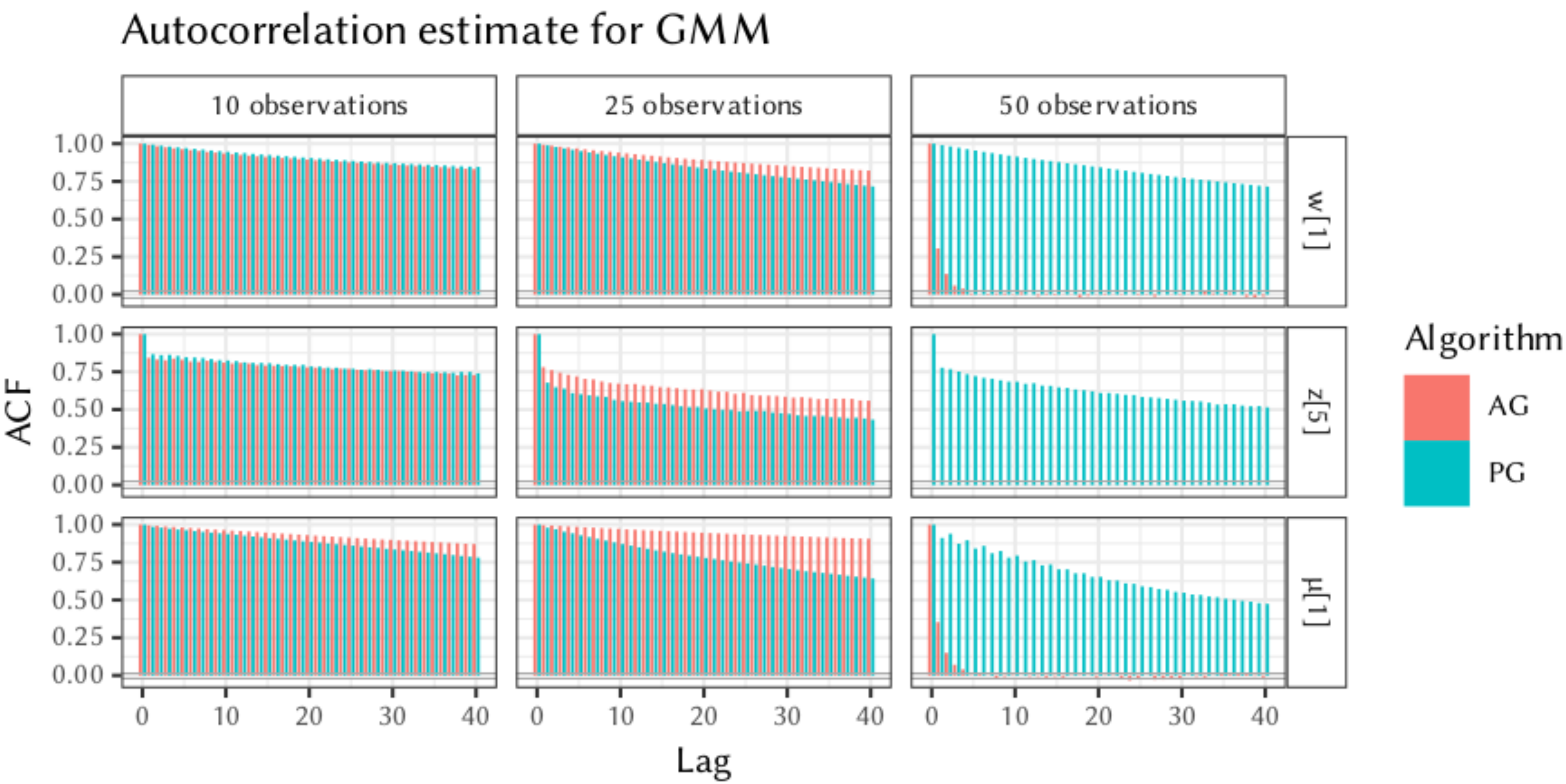




EVALUATION: CHAINS



Evaluation: Convergence

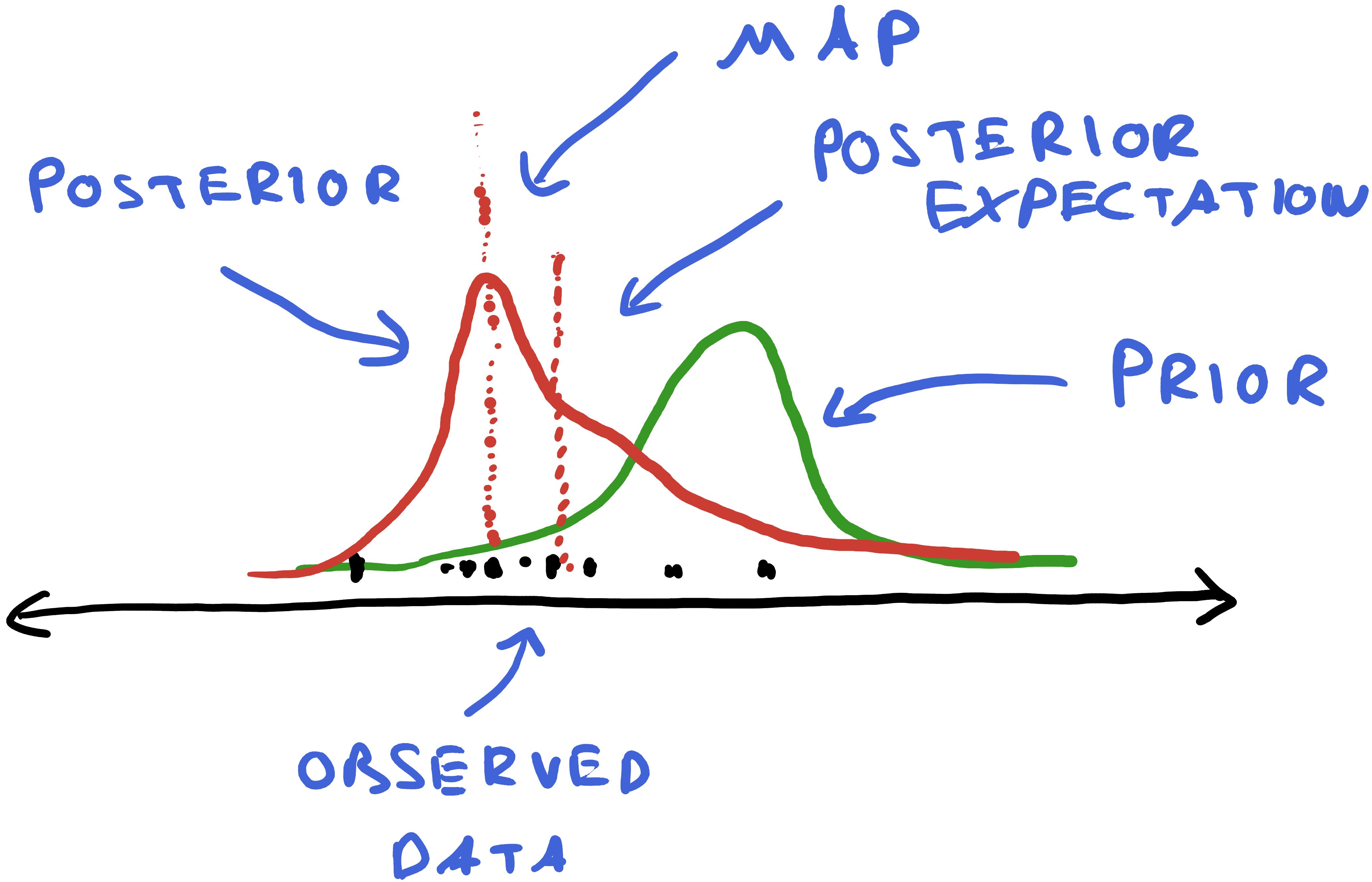


CONCLUSIONS

- STATIC DEPENDENCIES, FINITE CONDITIONALS: 
- SLICING DYNAMIC MODELS: 
- RECOVERING DYNAMIC STRUCTURE: 
- FUTURE PROOF: 

scratch

"BAYES-ICS"



Prediction

- Plugin Estimator:

arg max!

$$\hat{p}(y|x,D) = p(y|x,\vartheta^*(D))$$

- Posterior Predictive:

$$p(y|x,D) = \int p(y|x,\vartheta)\,p(\vartheta|D)\,d\nu(\vartheta)$$

Posterior expectation!