Probabilistic Programming with Densities in SlicStan

Efficient, Flexible and Deterministic

Maria Gorinova, Andy Gordon, and Charles Sutton

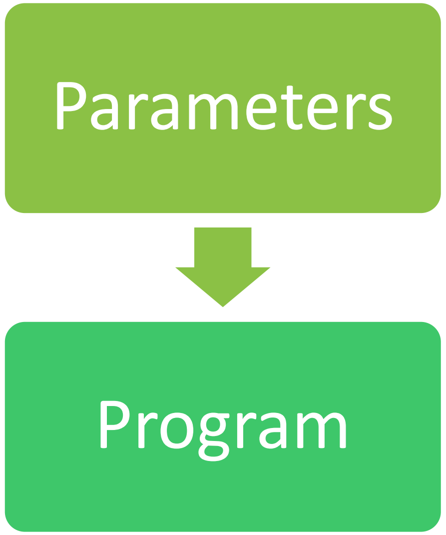
THE UNIVERSITY *of* EDINBURGH

EPSRC

A probabilistic program

```
sigma ~ gamma(0.1, 0.1);
mu ~ normal(0, 1);


y ~ normal(mu, sigma);
```

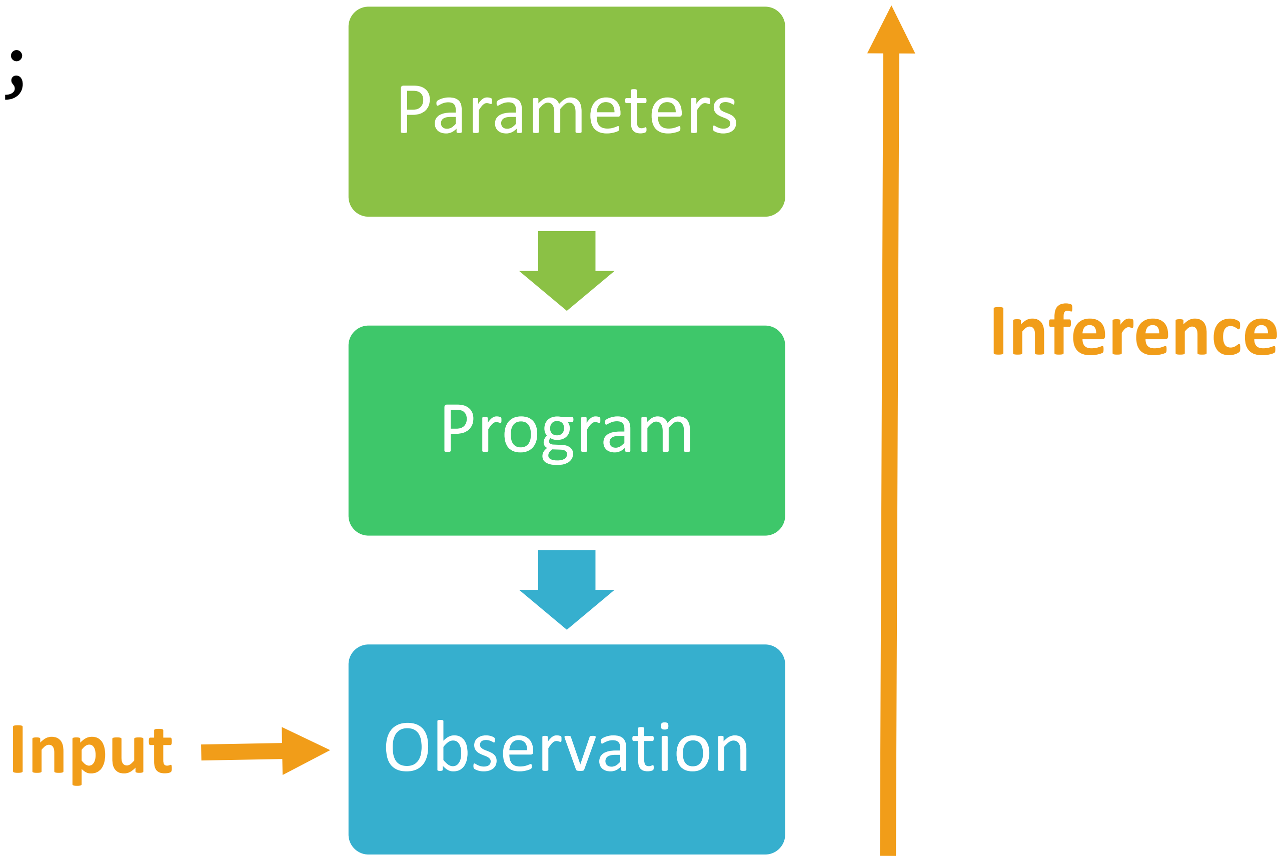


A probabilistic program

```
sigma ~ gamma(0.1, 0.1);
mu ~ normal(0, 1);

y ~ normal(mu, sigma);

observe y = 2.1;
```

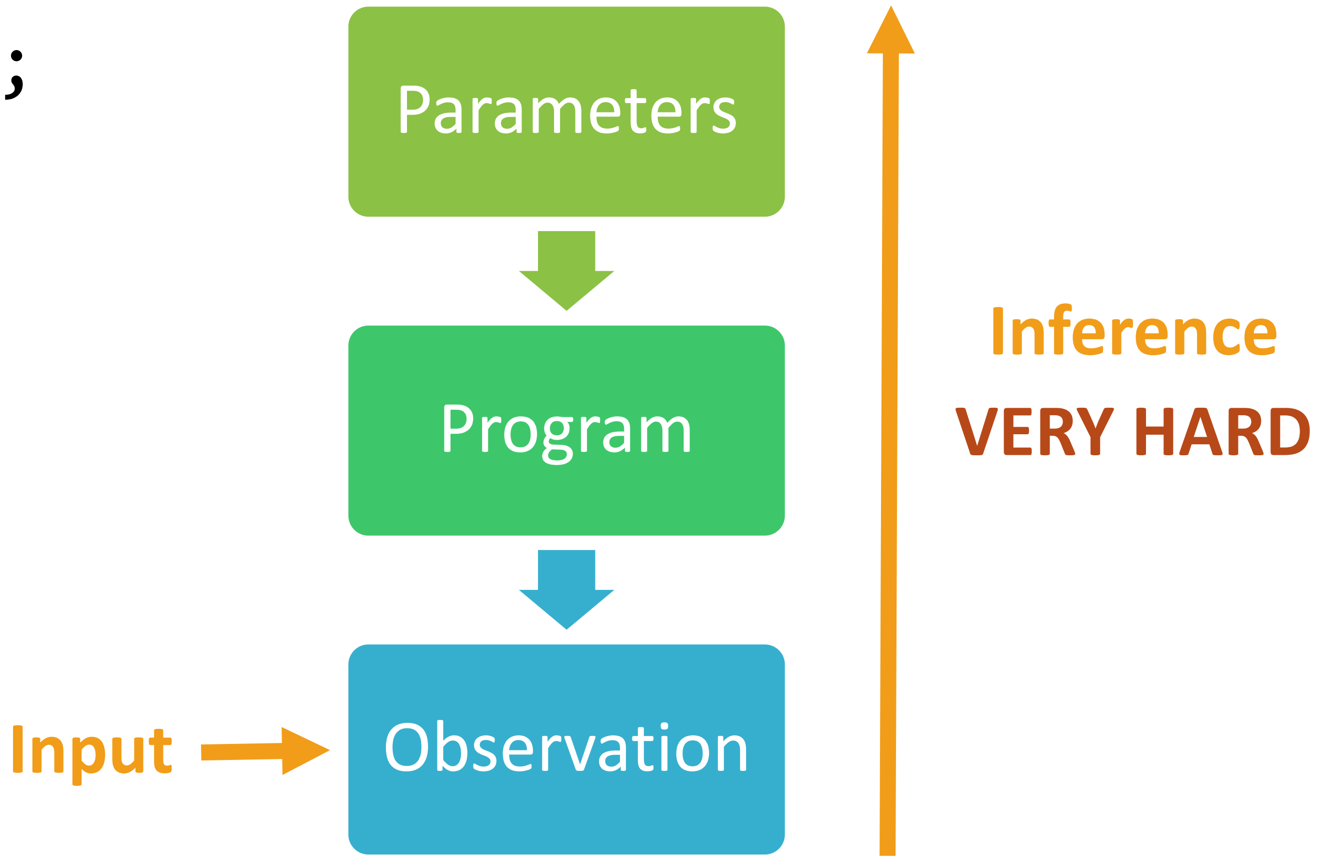


A probabilistic program

```
sigma ~ gamma(0.1, 0.1);
mu ~ normal(0, 1);

y ~ normal(mu, sigma);

observe y = 2.1;
```



https://mc-stan.org

Stan

Stan® is a state-of-the-art platform for statistical modeling and high-performance statistical computation. Thousands of users rely on Stan for statistical modeling, data analysis, and prediction in the social, biological, and physical sciences, engineering, and business.

Users specify log density functions in Stan's probabilistic programming language and get:

- full Bayesian statistical inference with MCMC sampling (NUTS, HMC)
- approximate Bayesian inference with variational inference (ADVI)
- penalized maximum likelihood estimation with optimization (L-BFGS)

https://mc-stan.org

Stan

Stan® is a state-of-the-art platform for statistical modeling and high-performance statistical computation. Thousands of users rely on Stan for statistical modeling, data analysis, and prediction in the social, biological, and physical sciences, engineering, and business.

Users specify log density functions in Stan's probabilistic programming language and get:

- full Bayesian statistical inference with MCMC sampling (NUTS, HMC)
- approximate Bayesian inference with variational inference (ADVI)
- penalized maximum likelihood estimation with optimization (L-BFGS)

2,000+
Forum Users

300,000+
RStan Downloads

150+
StanCon Attendees

```
data {
    int N;
    real y[N];
}
parameters {
    real mu;
    real sigma;
}
model {
    sigma ~ gamma(0.1, 0.1);
    mu ~ normal(0, 1);
    y ~ normal(mu, sigma);
}
generated quantities {
    real variance;
    variance = sigma * sigma;
}
```

Fast, but has unusual syntax

Goals: Understand the language principles behind Stan’s efficient black-box inference.

&

Make Stan more compositional.

Goal: Understand the principles behind Stan's inference and design a compositional alternative.

1. Stan programs are **deterministic.**

Density-Based Semantics

=

Standard big-step operational semantics

+

$\boldsymbol{\theta} \models \Gamma_p$ and $\mathcal{D} \models \Gamma_d$

$((\mathcal{D}, \boldsymbol{\theta}, \textbf{target} \mapsto 0), S) \Downarrow s'$

$\log p^* (\boldsymbol{\theta} \mid \mathcal{D}) \triangleq s'[\textbf{target}]$

2. Blocks correspond to different **information-flow levels.**

Information Flow	Performance Ordering
DATA < MODEL < GENQUANT	DATA ≤ GENQUANT ≤ MODEL

SlicStan

1. A Stan program is **deterministic.**

```
data {
    int N;
    real y[N];
}
parameters {
    real mu;
    real sigma;
}
model {
    sigma ~ gamma(0.1, 0.1);
    mu ~ normal(0, 1);
    y ~ normal(mu, sigma);
}
generated quantities {
    real variance;
    variance = sigma * sigma;
}
```

- Data $\mathfrak{D} = \{N: 2,\ \boldsymbol{y}: [0, 2]\}$
- Parameters $\boldsymbol{\theta}$
- Way of generating $\boldsymbol{\theta}$ and $\mathfrak{D}$
- Other variables

```
data {
   int N;
   real y[N];
}
parameters {
   real mu;
   real sigma;
}
model {
    sigma ~ gamma(0.1, 0.1);
    mu ~ normal(0, 1);
    y ~ normal(mu, sigma);
}
generated quantities {
   real variance;
   variance = sigma * sigma;
}
```

- Data $\mathfrak{D} = \{N: 2,\ y: [0, 2]\}$
- Parameters $\boldsymbol{\theta}$
- Way of generating $\boldsymbol{\theta}$ and $\mathfrak{D}$
- Other variables

```
data {
    int N;
    real y[N];
}
parameters {
    real mu;
    real sigma;
}
model {
    target += gamma_lpdf(sigma | 0.1, 0.1);
    target += normal_lpdf(mu | 0, 1);
    target += normal_lpdf(y | mu, sigma);
}
generated quantities {
    real variance;
    variance = sigma * sigma;
}
```

$$
\begin{aligned}
\log p(\mathfrak{D}, \theta) &= \log Gamma(\sigma, 0.1, 0.1) \\
&\quad + \log \mathcal{N}(\mu, 0, 1) \\
&\quad + \log \mathcal{N}(y_1, \mu, \sigma) \\
&\quad + \log \mathcal{N}(y_2, \mu, \sigma)
\end{aligned}
$$

```
data {
    int N;
    real y[N];
}
parameters {
    real mu;
    real sigma;
}
model {
    target += gamma_lpdf(sigma | 0.1, 0.1);
    target += normal_lpdf(mu | 0, 1);
    target += normal_lpdf(y | mu, sigma);
}
generated quantities {
    real variance;
    variance = sigma * sigma;
}
```

Fixed! → $\mathfrak{D} = \{N: 2,\ \boldsymbol{y}: [0, 2]\}$

θ

$\log p(\mathfrak{D}, \theta)$

```
data {
    int N;
    real y[N];
}
parameters {
    real mu;
    real sigma;
}
model {
    target += gamma_lpdf(sigma | 0.1, 0.1);
    target += normal_lpdf(mu | 0, 1);
    target += normal_lpdf(y | mu, sigma);
}
generated quantities {
    real variance;
    variance = sigma * sigma;
}
```

$\mathfrak{D} = \{N: 2,\ \boldsymbol{y}: [0, 2]\}$

$\theta = \{\mu: 0, \sigma: 1\}$

$$\log p(\mathfrak{D}, \theta) = 0$$
$$+ (\text{-}0.1)$$
$$+ (\text{-}0.4)$$
$$+ ((\text{-}0.4) + (\text{-}0.3))$$
$$= \text{-}1.2$$

$$v = \sigma * \sigma = 1$$

Inference

- Observed variables: $\mathfrak{D}$
- Parameters: θ
- $f(\theta) = \log p(\theta|\mathfrak{D}) + const_{\mathfrak{D}}$
- Inference: evaluate $f(\theta)$ repeatedly!

Inference

- Observed variables: $\mathfrak{D}$
- Parameters: θ
- $f(\theta) = \log p(\theta|\mathfrak{D}) + const_{\mathfrak{D}}$
- Inference: evaluate $f(\theta)$!

```
model {
    target +=
        gamma_lpdf(sigma | 0.1, 0.1);
    target +=
        normal_lpdf(mu | 0, 1);
    target +=
        normal_lpdf(y | mu, sigma);
}
```

$$\log p(\theta, \mathfrak{D}) = \log p(\theta|\mathfrak{D}) + const_{\mathfrak{D}}$$ ← **Bayes rule in log space**

$$\Rightarrow f(\theta) = \log p(\theta, \mathfrak{D}) = \log p^*(\theta|\mathfrak{D})$$ ← **Unnormalised posterior**

Stan: Density-Based Semantics

$P ::=$

$$
\begin{array}{l}
\textbf{data} \; \{ \; \Gamma_d \; \} \\
\textbf{transformed data} \; \{ \; \Gamma_{td}, \mathrm{S}_{td} \; \} \\
\textbf{parameters} \; \{ \; \Gamma_p \; \} \\
\textbf{transformed parameters} \; \{ \; \Gamma_{tp}, \mathrm{S}_{tp} \; \} \\
\textbf{model} \; \{ \; \mathrm{S}_m \; \} \\
\textbf{generated quantities} \; \{ \; \Gamma_{gq}, \mathrm{S}_{gq} \; \}
\end{array}
$$

Standard big-step operational semantics

$$(s, S) \Downarrow s'$$

Stan: Density-Based Semantics

P ::=

data { Γ_d }
transformed data { Γ_{td}, S_{td} }
parameters { Γ_p }
transformed parameters { Γ_{tp}, S_{tp} }
model { S_m }
generated quantities { Γ_{gq}, S_{gq} }

Standard big-step operational semantics

+

$$S = S_{td}; S_{tp}; S_m; S_{gq}$$

$$\boldsymbol{\theta} \models \Gamma_p \text{ and } \mathcal{D} \models \Gamma_d$$

$$((\mathcal{D}, \boldsymbol{\theta}, \textbf{target} \mapsto 0), S) \Downarrow s'$$

$$\log p^*(\boldsymbol{\theta} \mid \mathcal{D}) \triangleq s'[\textbf{target}]$$

An Extended Stan Program

```
data {
      int N;
      real y[N];
}
transformed data {
      real alpha = 0.1;
      real beta = 0.1;
}
parameters {
      real mu;
      real tau;
}
```

```
transformed parameters {
      real sigma;
      sigma = pow(tau, -0.5);
}
model {
      tau ~ gamma(alpha, beta);
      mu ~ normal(0, 1);
      y ~ normal(mu, sigma);
}
generated quantities {
      real variance;
      variance = sigma * sigma;
}
```

2. Blocks correspond to different **information-flow levels**

Information Flow in Stan

```
data {
     int N;
     real y[N];
}
transformed data {
     real alpha = 0.1;
     real beta = 0.1;
}
parameters {
     real mu;
     real tau;
}
transformed parameters {
     real sigma;
     sigma = pow(tau, -0.5);
}
model {
     tau ~ gamma(alpha, beta);
     mu ~ normal(0, 1);
     y ~ normal(mu, sigma);
}
generated quantities {
     real variance;
     variance = sigma * sigma;
}
```

DATA

<

MODEL

<

GENQUANT

SlicStan

```
real mu ~ normal(0, 1);

real alpha = 0.1;
real beta = 0.1;
real tau ~ gamma(alpha, beta);
real sigma = pow(tau, -0.5);

data int N;
data real[N] y ~ normal(mu, sigma);

real variance = sigma * sigma;
```

→

```
data {
	int N;
	real y[N];
}
transformed data {
	real alpha = 0.1;
	real beta = 0.1;
}
parameters {
	real mu;
	real tau;
}
transformed parameters {
	real sigma;
	sigma = pow(tau, -0.5);
}
model {
	tau ~ gamma(alpha, beta);
	mu ~ normal(0, 1);
	y ~ normal(mu, sigma);
}
generated quantities {
	real variance;
	variance = sigma * sigma;
}
```

Information Flow in SlicStan

```
DATA real alpha = 0.1;
DATA real beta = 0.1;
MODEL real tau ~ gamma(alpha, beta);

MODEL real mu ~ normal(0, 1);


MODEL real sigma = pow(tau, -0.5);
data DATA int N;
data DATA real[N] y;
y ~ normal(mu, sigma);


GENQUANT real variance = sigma*sigma
```

Standard* information flow type system

(Assign)

$$\frac{\Gamma(L) = (\tau, \ell) \quad \Gamma \vdash E : (\tau, \ell)}{\Gamma \vdash (L = E) : \ell}$$

(ESub)

$$\frac{\Gamma \vdash E : (\tau, \ell) \quad \ell \leq \ell'}{\Gamma \vdash E : (\tau, \ell')}$$

(Seq)

$$\frac{\Gamma \vdash S_1 : \ell \quad \Gamma \vdash S_2 : \ell \quad \mathcal{S}(S_1, S_2)}{\Gamma \vdash (S_1; S_2) : \ell}$$

Volpano, Irvine, and Smith. A sound type system for secure flow analysis. Journal of computer security 4, 2-3 (1996), 167–187.

Information Flow in SlicStan

```
DATA real alpha = 0.1;
DATA real beta = 0.1;
MODEL real tau ~ gamma(alpha, beta);

MODEL real mu ~ normal(0, 1);


MODEL real sigma = pow(tau, -0.5);
data DATA int N;
data DATA real[N] y;
y ~ normal(mu, sigma);


GENQUANT real variance = sigma*sigma;
```

Standard* information flow type system

+

target : (**real**, MODEL)

Derived rule:

$$\frac{\Gamma \vdash E : T \quad \Gamma \vdash E_i : T_i \quad \forall i \in 1..n}{\Gamma \vdash E \sim \text{D\_dist}(E_1, \ldots E_n) : \text{MODEL}}$$

Translation of SlicStan to Stan

In the paper:

- Formal semantics of SlicStan.
- Formal *elaboration, slicing* and *translation* procedures.
- Proof of *semantic preservation.*

Goal: Understand the principles behind Stan's inference and design a compositional alternative.

1. Stan programs are **deterministic.**

Density-Based Semantics
=
Standard big-step operational semantics
+

$\boldsymbol{\theta} \models \Gamma_p$ and $\mathcal{D} \models \Gamma_d$

$((\mathcal{D}, \boldsymbol{\theta}, \textbf{target} \mapsto 0), S) \Downarrow s'$

$\log p^* (\boldsymbol{\theta} \mid \mathcal{D}) \triangleq s'[\textbf{target}]$

2. Blocks correspond to different **information-flow levels.**

Information Flow	Performance Ordering
DATA	DATA
$<$	$\leq$
MODEL	GENQUANT
$<$	$\leq$
GENQUANT	MODEL

SlicStan

Probabilistic Programming with Densities in SlicStan:

Efficient, Flexible and Deterministic

Maria I. Gorinova, Andrew D. Gordon, Charles Sutton

mc-stan.org

Density-Based Semantics

=

Standard big-step operational semantics

+

$\boldsymbol{\theta} \models \Gamma_p$ and $\mathcal{D} \models \Gamma_d$

$((\mathcal{D}, \boldsymbol{\theta}, \textbf{target} \mapsto 0), S) \Downarrow s'$

$\log p^* (\boldsymbol{\theta} \mid \mathcal{D}) \triangleq s'[\textbf{target}]$

DATA	DATA
$<$	$\leq$
MODEL	GENQUANT
$<$	$\leq$
GENQUANT	MODEL
Information Flow	Performance Ordering

SlicStan

Email: m.gorinova@ed.ac.uk

THE UNIVERSITY of EDINBURGH

EPSRC

Comparison with sampling-based semantics

(Eval Model)

$$\frac{(s, E) \Downarrow V \quad (s, E_i) \Downarrow V_i \quad \forall i \in 1..n \quad V' = s(\textbf{target}) + \text{d\_lpdf}(V, V_1, \ldots, V_n)}{(s, E \sim \text{d}(E_1, \ldots, E_n)) \Downarrow s[\textbf{target} \mapsto V']}$$

(Sampling Model)

$$\frac{v \in \text{Val} \quad p = \text{Dist}(s(\overline{\theta}))(v)}{(s, x \sim \text{Dist}(\overline{\theta})) \Downarrow^{x \mapsto [v]} (s[x \mapsto v], p)}$$