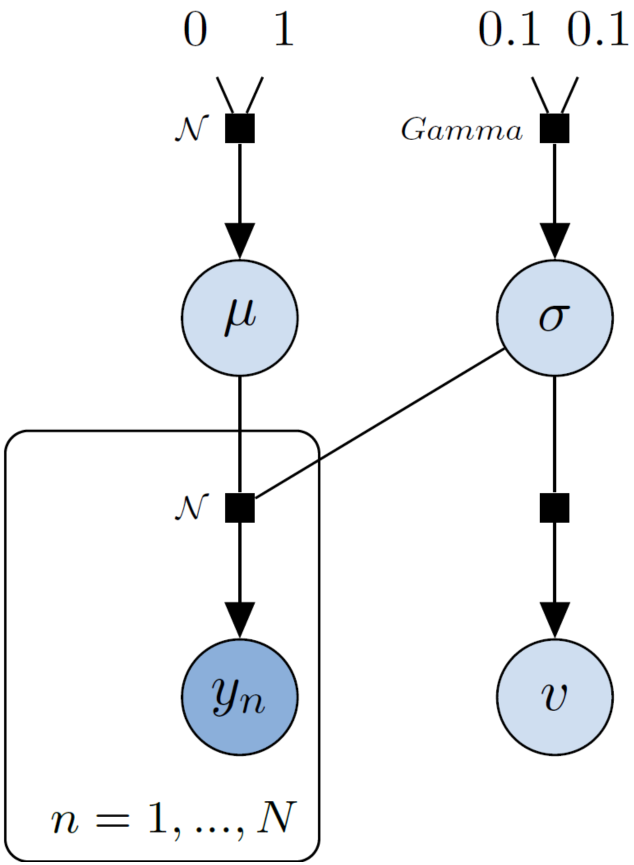
SlicStan

Optimising Probabilistic Programs using Information Flow Analysis

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

What is Stan?

```
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;
}
```



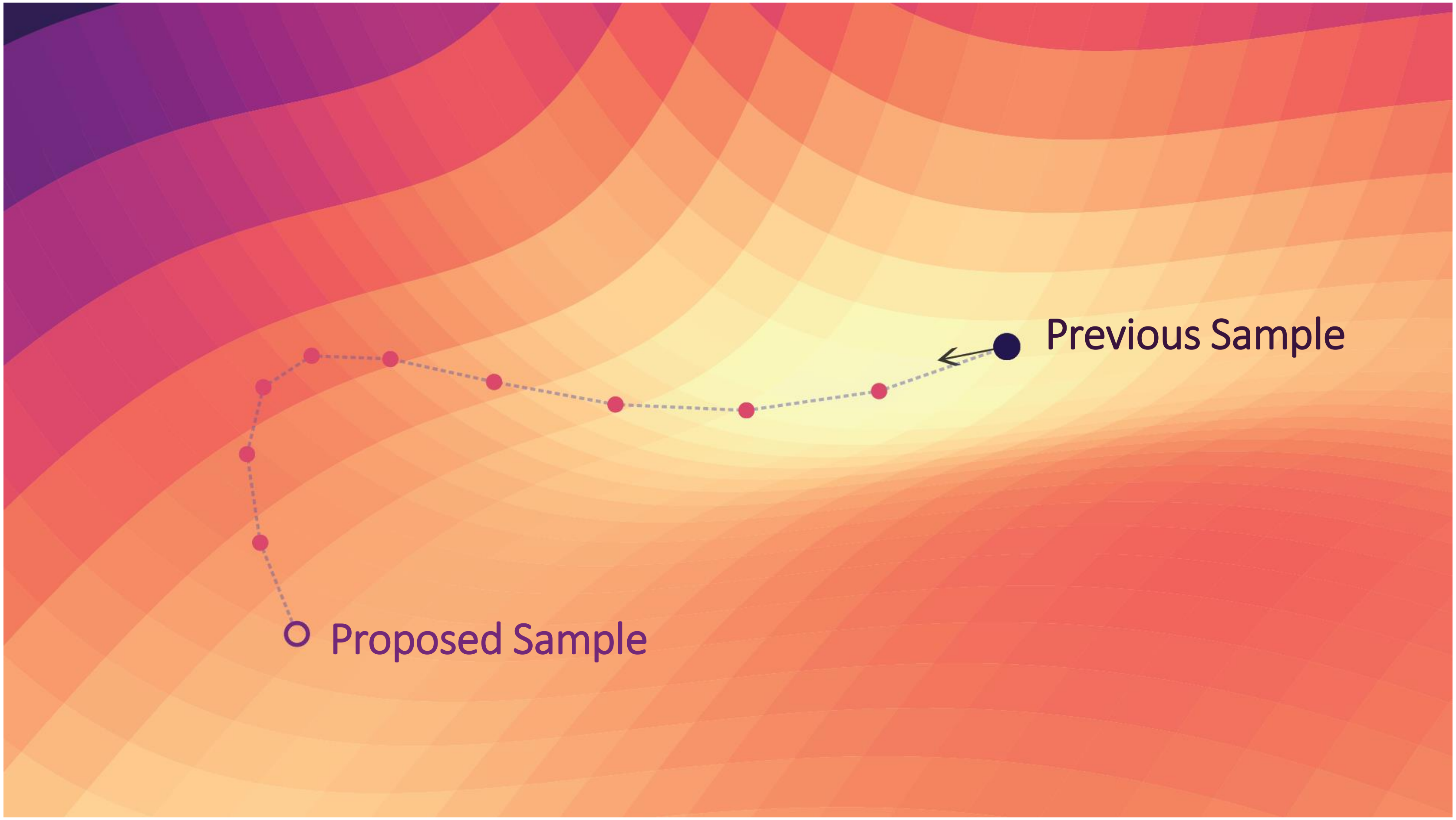
Why blocks matter?

?

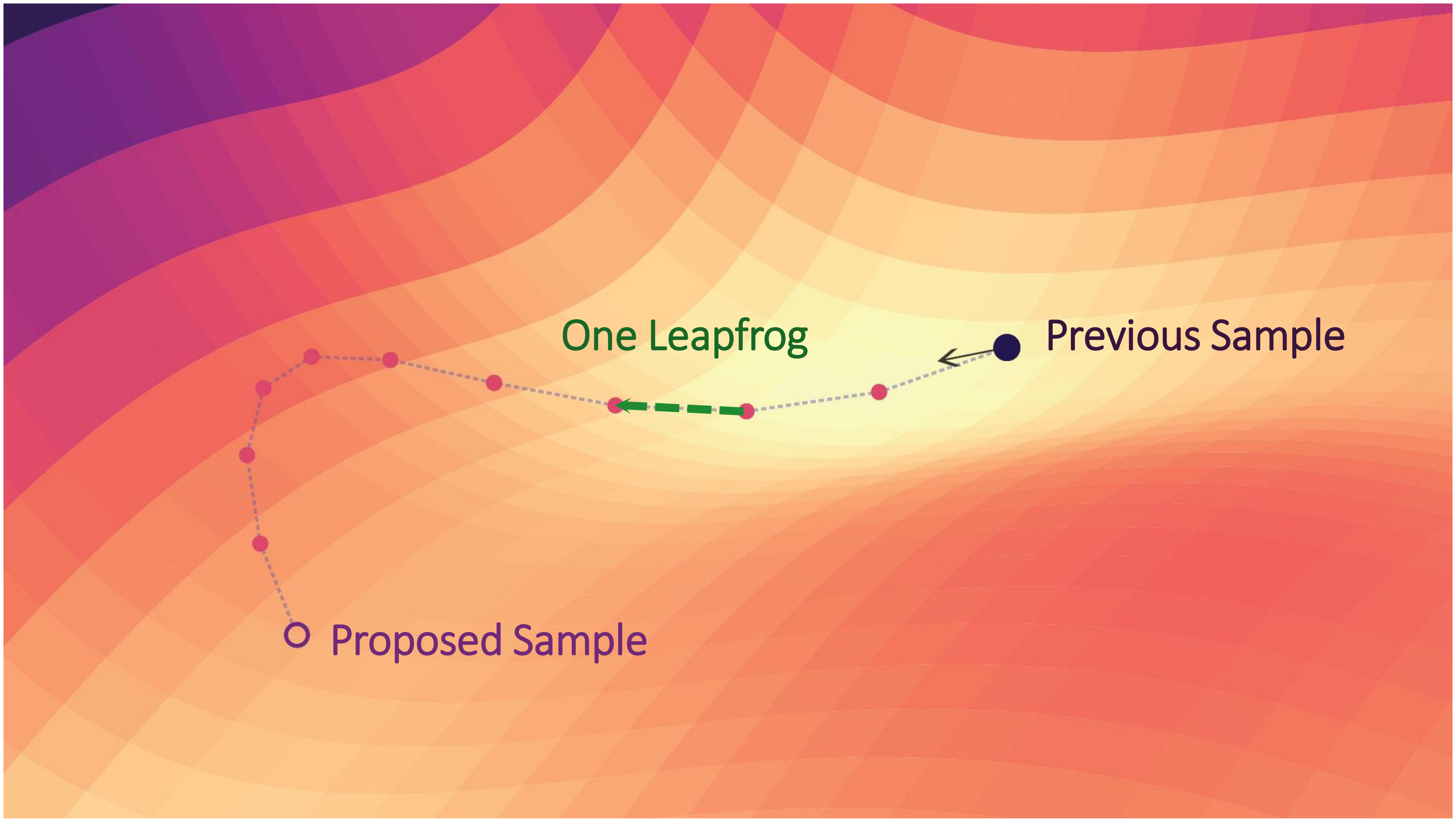
```
real x;
```

- **data**
- **transformed data**
- **parameters**
- **transformed parameters**
- **model**
- **generated quantities**

Inference in Stan: Hamiltonian Monte Carlo



Inference in Stan: Hamiltonian Monte Carlo



SlicStan vs. Stan

```
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

- Transfer of information between two variables

```
y = x + 5

if x > 5 then y = 1 else y = 0
```

Information Flow

$$\mathrm{PUBLIC} < \mathrm{SECRET}$$

p: PUBLIC, s: SECRET

✓ `s = p`

✗ `p = s`

✗ `if s > 5 then p = 1 else p = 0`

Information Flow in Stan

```
data {
   int N;
   real y[N];
   real mu_mu;
   real sigma_mu;
}
transformed data {
   real alpha = 0.1;
   real beta = 0.1;
}
parameters {
   real mu_y;
   real tau_y;
}
transformed parameters {
   real sigma_y;
   sigma_y = pow(tau_y, -0.5);
}
model {
   tau_y ~ gamma(alpha, beta);
   mu_y ~ normal(mu_mu, sigma_mu);
   y ~ normal(mu_y, sigma_y);
}
generated quantities {
   real variance_y;
   variance_y = sigma_y * sigma_y;
}
```

DATA

$\leq$

MODEL

$\leq$

GENQUANT

Key idea

- Find all **possible roles** a variable can have during inference, w.r.t. the **information flow**:

$$\text{DATA} \leq \text{MODEL} \leq \text{GENQUANT}$$

  - `data real x` → `level(x)` = DATA
  - `real x, ≠` → `level(x)` ≥ MODEL
  - `x = foo(y)` → `level(x)` ≥ `level(y)`
  - `x ~ foo(y)` → `level(x)` ≤ MODEL and `level(y)` ≤ MODEL

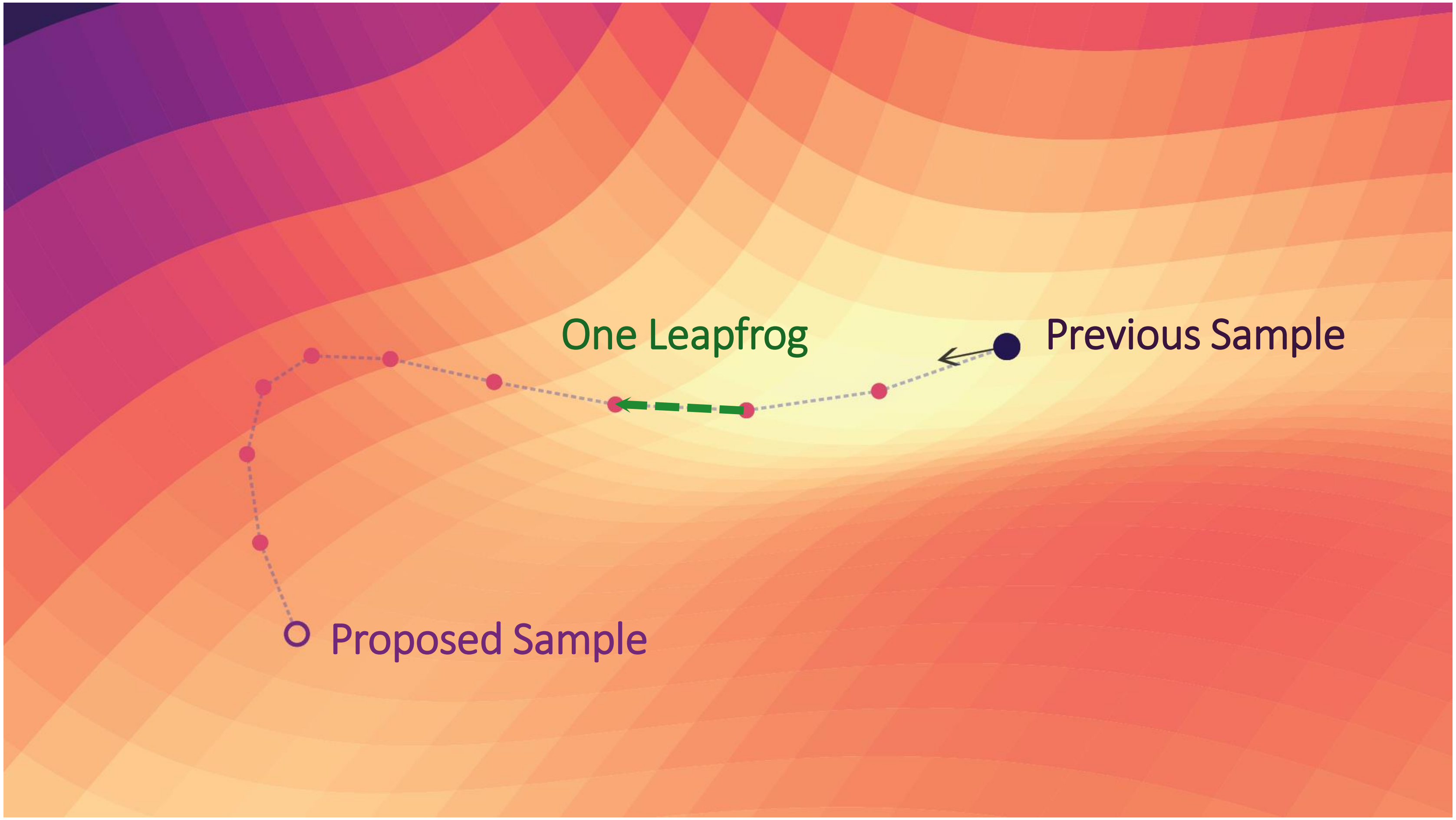
Key idea

- Find all **possible roles** a variable can have during inference, w.r.t. the **information flow**:

$$\text{DATA} \leq \text{MODEL} \leq \text{GENQUANT}$$

  - `data real x` → `level(x) =` DATA
  - `real x, ≠` → `level(x) ≥` MODEL
  - `x = foo(y)` → `level(x) ≥ level(y)`
  - `x ~ foo(y)` → `level(x) ≤` MODEL `and level(y) ≤` MODEL
- Not unique… Which one do we choose?

Hamiltonian Monte Carlo



Performance ordering

Block	Execution	Level
`data`	---	DATA
`transformed data`	once	DATA
`parameters`	once per leapfrog	MODEL
`transformed parameters`	once per leapfrog	MODEL
`model`	once per leapfrog	MODEL
`generated quantities`	once per sample	GENQUANT

$$\text{DATA} < \text{GENQUANT} < \text{MODEL}$$

Key insight

- Find all **possible roles** a variable can have during inference, w.r.t. the **information flow**:

$$\mathrm{DATA} \quad \leq \quad \mathrm{MODEL} \quad \leq \quad \mathrm{GENQUANT}$$

- Choose the most **optimal role**, w.r.t. the **performance ordering**:

$$\mathrm{DATA} \quad < \quad \mathrm{GENQUANT} \quad < \quad \mathrm{MODEL}$$

SlicStan

```
real alpha = 0.1;
real beta = 0.1;
real tau_y ~ gamma(alpha, beta);

data real mu_mu;
data real sigma_mu;
real mu_y ~ normal(mu_mu, sigma_mu);

real sigma_y = pow(tau_y, -0.5);
data int N;
data real[N] y;
y ~ normal(mu_y, sigma_y);

real variance_y = pow(sigma_y, 2);
```

SlicStan

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

data DATA real mu_mu;
data DATA real sigma_mu;
MODEL real mu_y ~ normal(mu_mu, sigma_mu);

MODEL real sigma_y = pow(tau_y, -0.5);
data DATA int N;
data DATA real[N] y;
y ~ normal(mu_y, sigma_y);

GENQUANT real variance_y = pow(sigma_y, 2);
```

SlicStan

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

data DATA real mu_mu;
data DATA real sigma_mu;
MODEL real mu_y ~ normal(mu_mu, sigma_mu);

MODEL real sigma_y = pow(tau_y, -0.5);
data DATA int N;
data DATA real[N] y;
y ~ normal(mu_y, sigma_y);

GENQUANT real variance_y = pow(sigma_y, 2);
```

Stan

```
data {
   int N;
   real y[N];
   real mu_mu;
   real sigma_mu;
}
transformed data {
   real alpha = 0.1;
   real beta = 0.1;
}
parameters {
   real mu_y;
   real tau_y;
}
transformed parameters {
   real sigma_y;
   sigma_y = pow(tau_y, -0.5);
}
model {
   tau_y ~ gamma(alpha, beta);
   mu_y ~ normal(mu_mu, sigma_mu);
   y ~ normal(mu_y, sigma_y);
}
generated quantities {
   real variance_y;
   variance_y = sigma_y * sigma_y;
}
```

Neal's Funnel

SlicStan

```
real nc_normal(real m, real s) {
   real raw ~ normal(0, 1);
   return s * raw + m;
}
real y = nc_normal(0, 3);
real x = nc_normal(0, exp(y/2));
```

Stan (efficient)

```
parameters {
   real y_raw;
   real x_raw;
}
transformed parameters {
   real y;
   real x;
   y = 3.0 * y_raw;
   x = exp(y/2) * x_raw;
}
model {
   y_raw ~ normal(0, 1);
   x_raw ~ normal(0, 1);
}
```

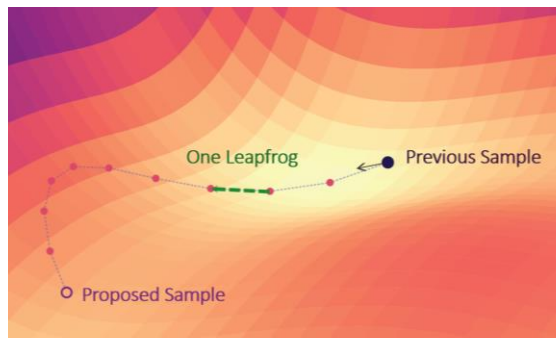
SlicStan: Optimising Probabilistic Programs using Information Flow Analysis

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

?

```
real x;
```

- `data`
- `transformed data`
- `parameters`
- `transformed parameters`
- `model`
- `generated quantities`



$$\mathrm{DATA} \leq \mathrm{MODEL} \leq \mathrm{GENQUANT}$$

Information Flow

$$\mathrm{DATA} < \mathrm{GENQUANT} < \mathrm{MODEL}$$

Performance Ordering

Email: m.gorinova@ed.ac.uk

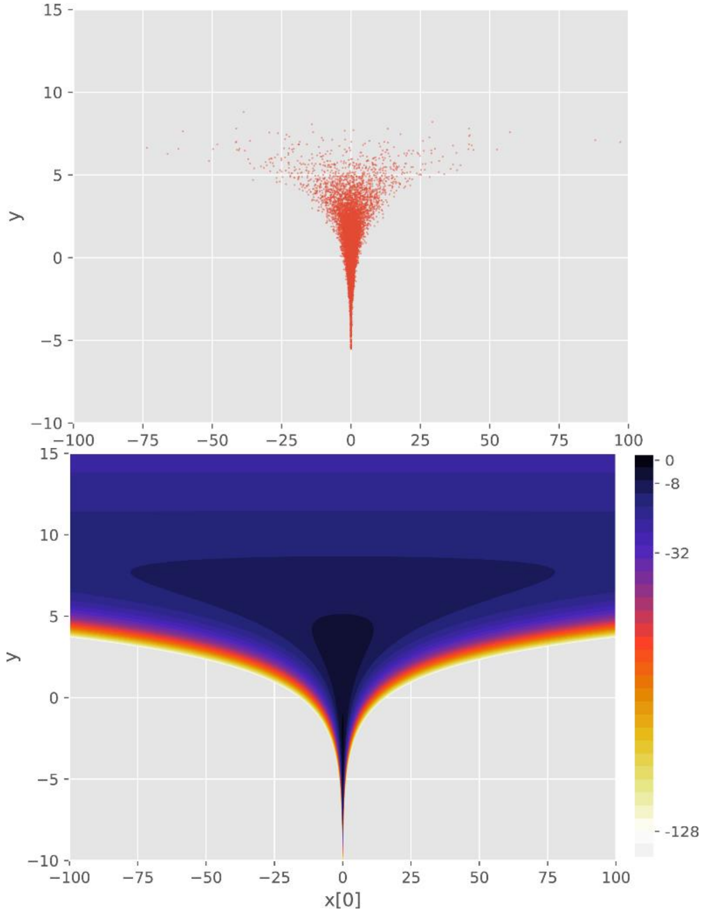
SlicStan PPS page: https://tiny.cc/slicstan

Technical paper available soon: ask me for details!

THE UNIVERSITY of EDINBURGH

EPSRC

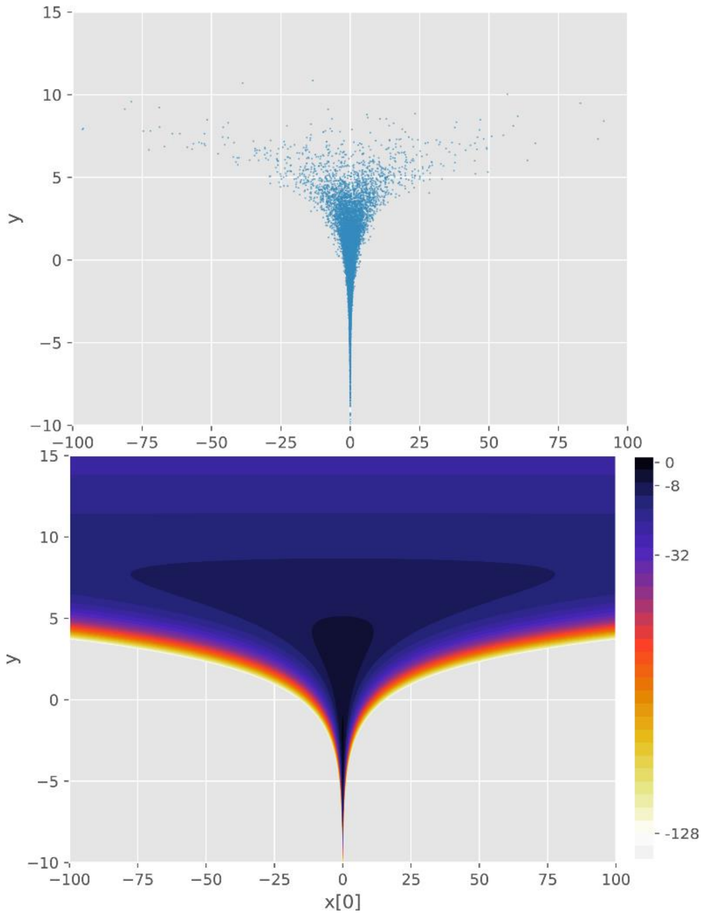
Neal’s Funnel



Stan (direct implementation)

```
parameters {
   real y;
   real x;
}
model {
   y ~ normal(0, 3.0);
   x ~ normal(0, exp(y/2));
}
```

Neal’s Funnel



Stan (efficient)

```
parameters {
    real y_raw;
    real x_raw;
}
transformed parameters {
    real y;
    real x;
    y = 3.0 * y_raw;
    x = exp(y/2) * x_raw;
}
model {
    y_raw ~ normal(0, 1);
    x_raw ~ normal(0, 1);
}
```

Neal's Funnel

SlicStan

```
real nc_normal(real m, real s) {
   real raw ~ normal(0, 1);
   return s * raw + m;
}
real y = nc_normal(0, 3);
real x = nc_normal(0, exp(y/2));
```

Stan (efficient)

```
parameters {
   real y_raw;
   real x_raw;
}
transformed parameters {
   real y;
   real x;
   y = 3.0 * y_raw;
   x = exp(y/2) * x_raw;
}
model {
   y_raw ~ normal(0, 1);
   x_raw ~ normal(0, 1);
}
```