Automatic Reparameterisation in Probabilistic Programming

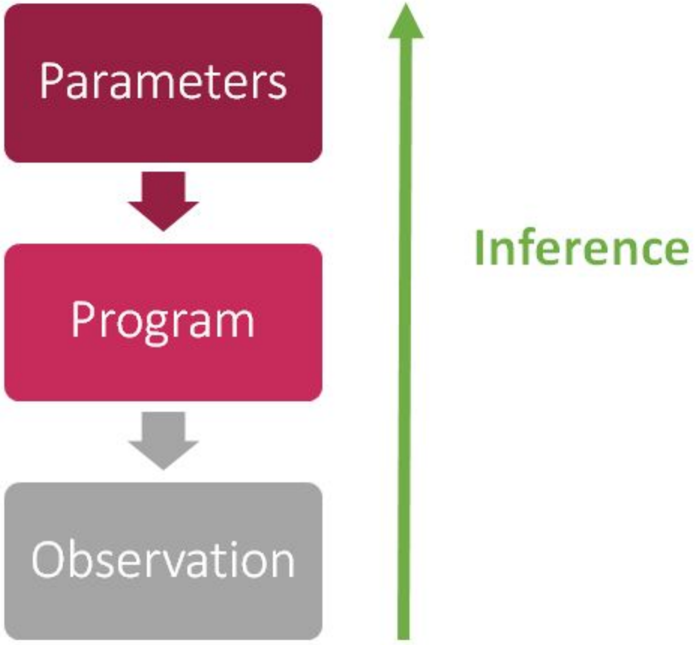
Maria I. Gorinova, Dave Moore, Matthew D. Hoffman

A probabilistic program: Schools

```
mu ~ normal(0, 5)
tau ~ halfCauchy(0, 5)

for(n in 1 .. 3)
  theta[n] ~ normal(mu, tau)
  y[n] ~ normal(theta, sigma[n])

observe y = [28, 8, -3]
observe sigma = [15, 10, 16]
```

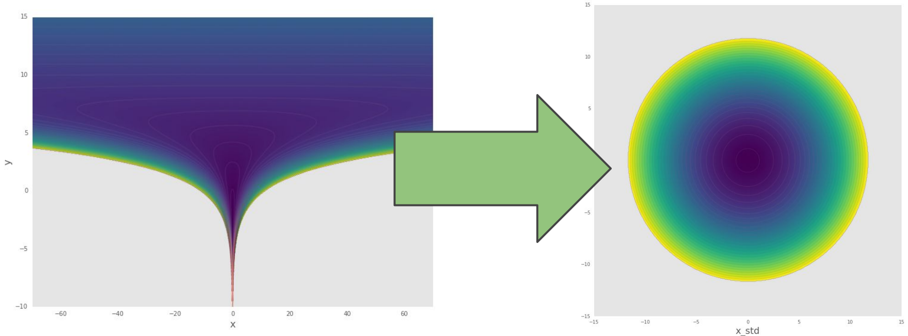


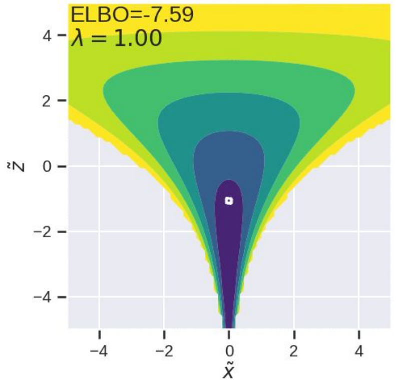
Automatic reparameterisation overview

1. What is reparameterisation and why is it difficult?

2. Reparameterisation in probabilistic programming

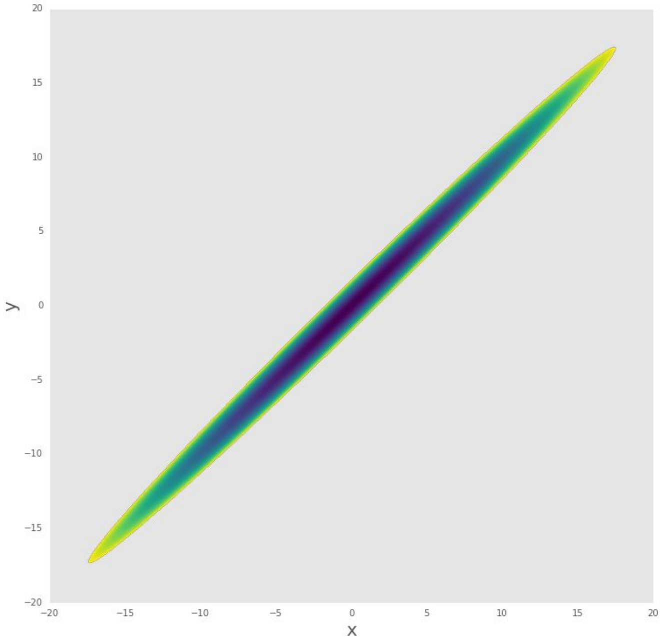
3. Variationally inferred parameterisation (VIP)



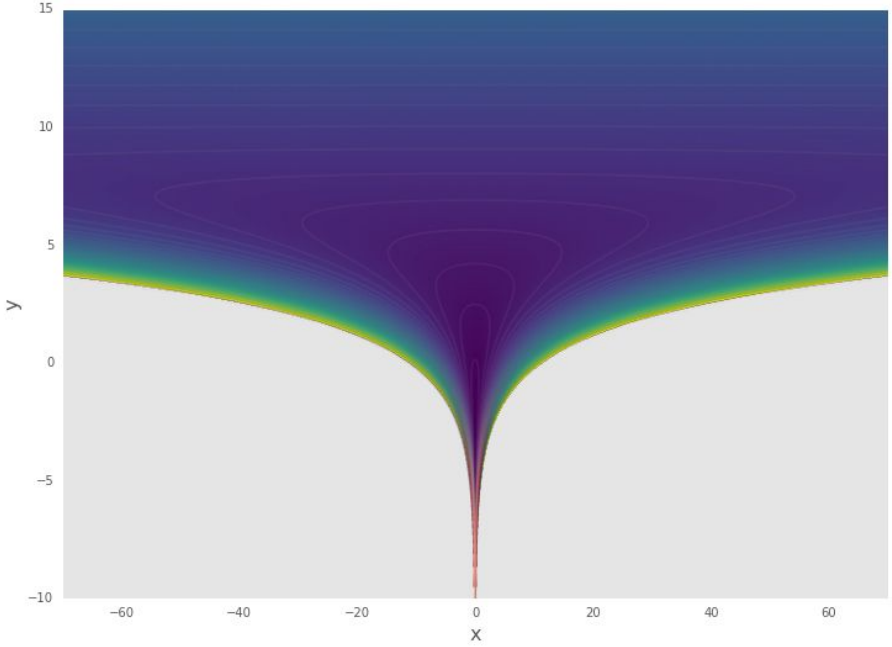


Problem: The posterior geometry affects quality of inference

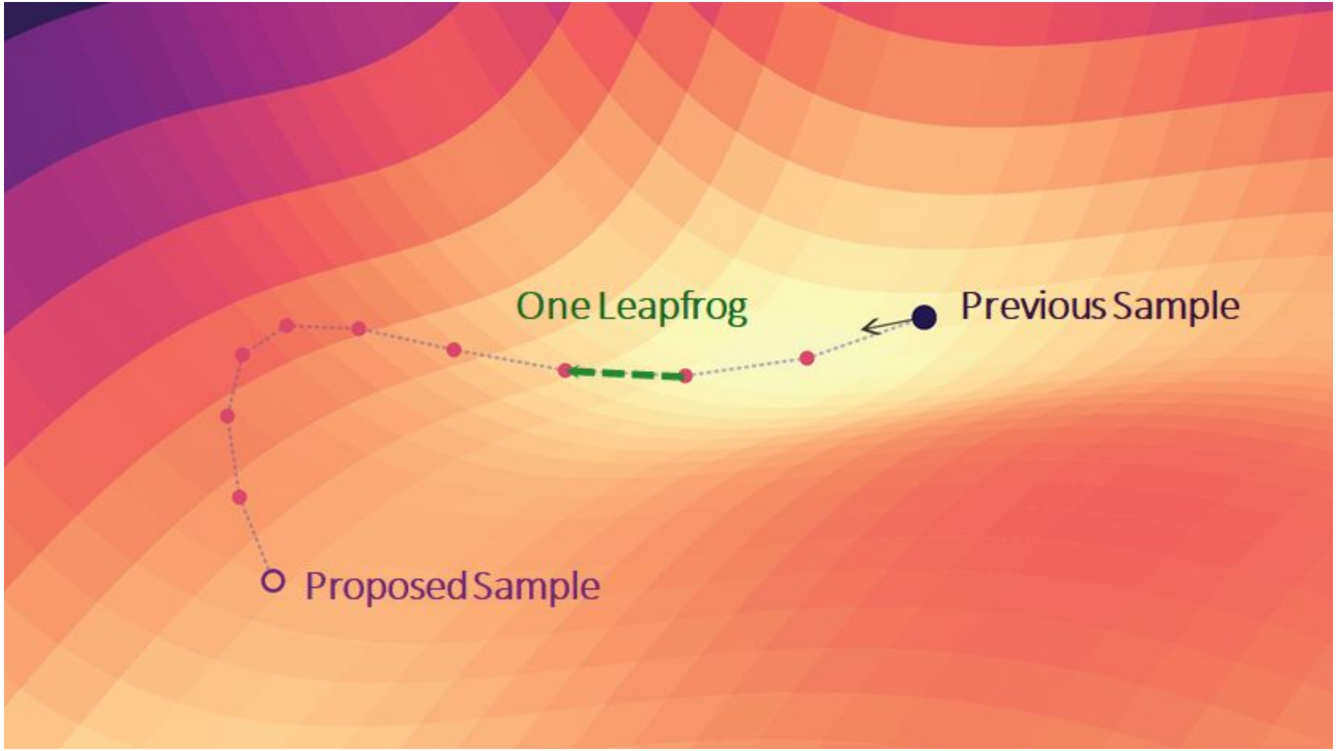
$$y \sim \mathcal{N}(0, 1)$$
$$x \sim \mathcal{N}(y, 0.1)$$



$$y \sim \mathcal{N}(0, 3)$$
$$x \sim \mathcal{N}(0, e^{y/2})$$

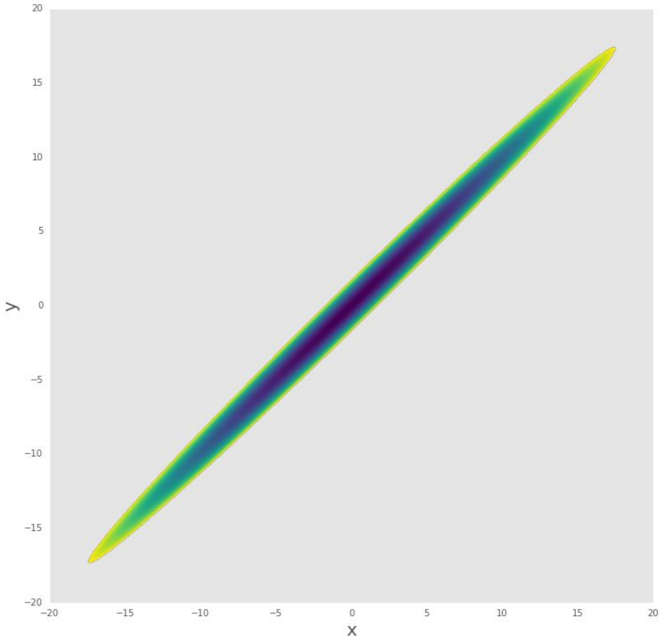


Hamiltonian Monte Carlo (HMC)

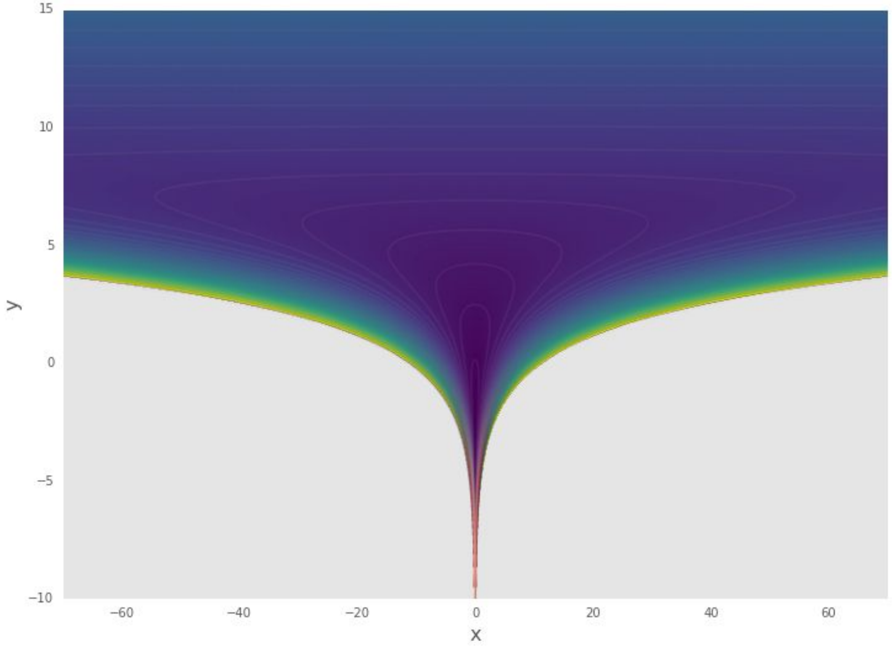


Problem: The posterior geometry affects quality of inference

$$y \sim \mathcal{N}(0, 1)$$
$$x \sim \mathcal{N}(y, 0.1)$$

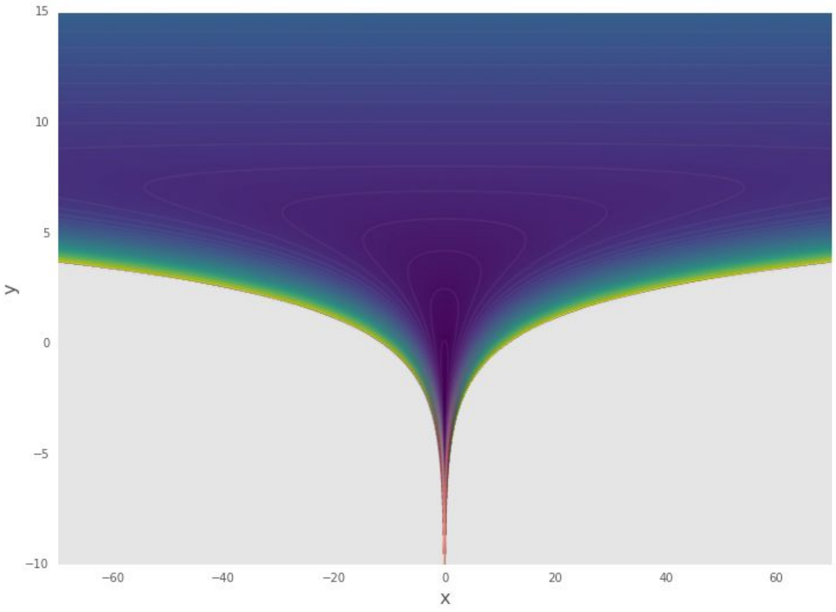


$$y \sim \mathcal{N}(0, 3)$$
$$x \sim \mathcal{N}(0, e^{y/2})$$



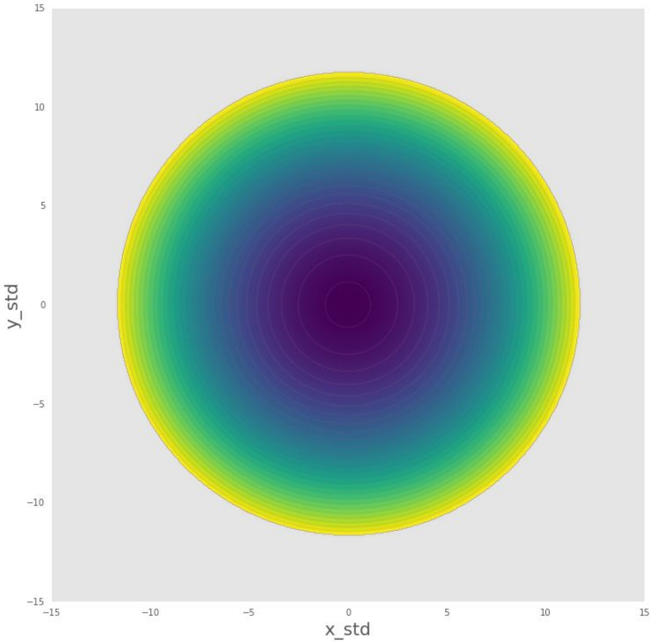
What is model reparameterisation?

$$y \sim \mathcal{N}(0, 3)$$
$$x \sim \mathcal{N}(0, e^{y/2})$$



$$\tilde{y} \sim \mathcal{N}(0, 1)$$
$$\tilde{x} \sim \mathcal{N}(0, 1)$$

$$y = \tilde{y} \times 3$$
$$x = \tilde{x} \times e^{y/2}$$

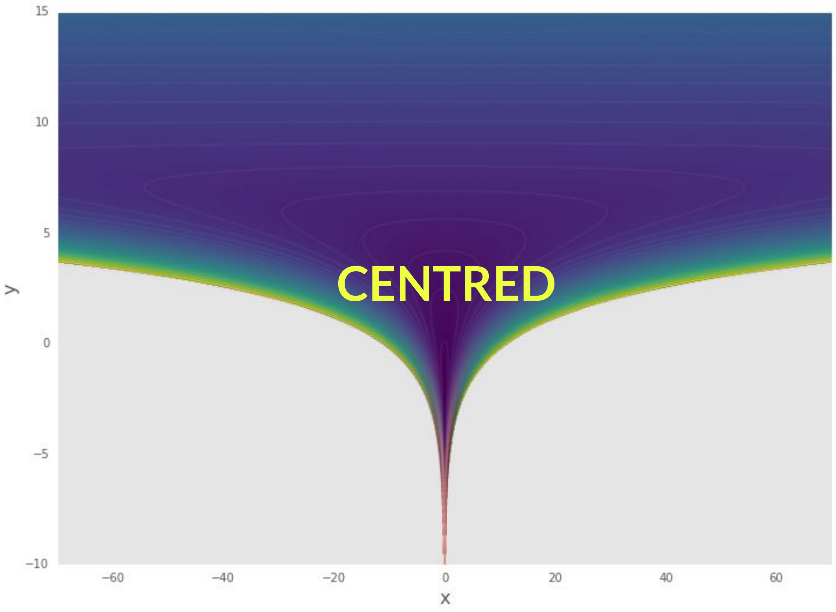


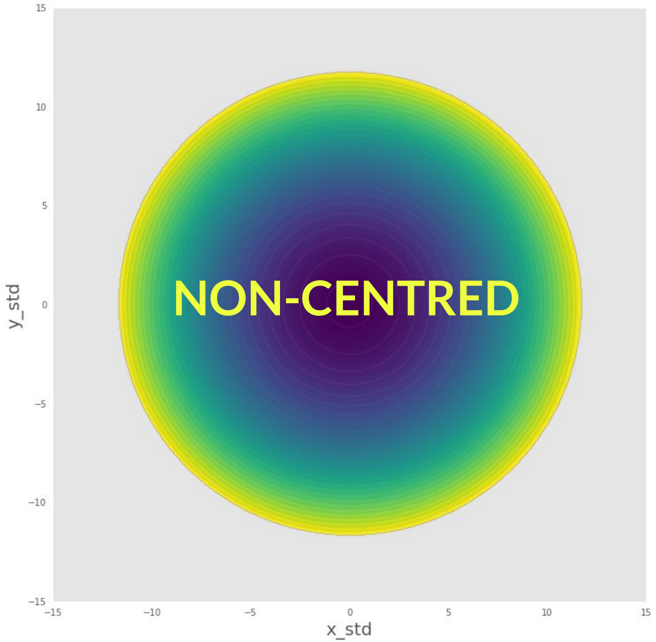
What is model reparameterisation?

$$y \sim \mathcal{N}(0, 3)$$
$$x \sim \mathcal{N}(0, e^{y/2})$$

$$\tilde{y} \sim \mathcal{N}(0, 1)$$
$$\tilde{x} \sim \mathcal{N}(0, 1)$$

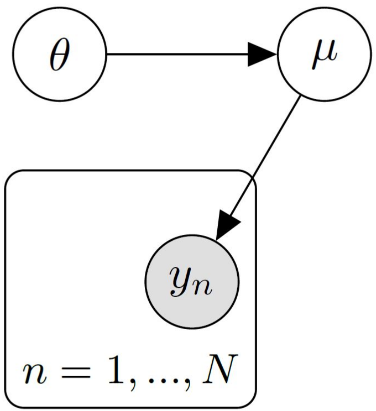
$$y = \tilde{y} \times 3$$
$$x = \tilde{x} \times e^{y/2}$$





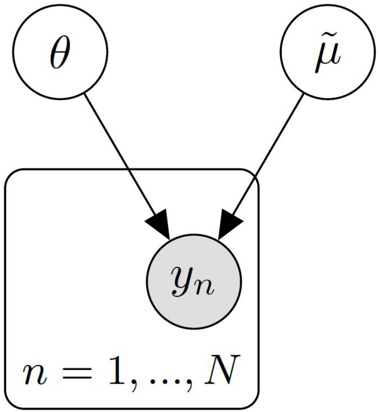
Understanding Reparameterisation Effects

Centred



$$\theta \sim \mathcal{N}(0, 1) \qquad \mu \sim \mathcal{N}(\theta, \sigma_\mu)$$
$$y_n \sim \mathcal{N}(\mu, \sigma) \text{ for all } n \in 1 \ldots N$$

Non-centred



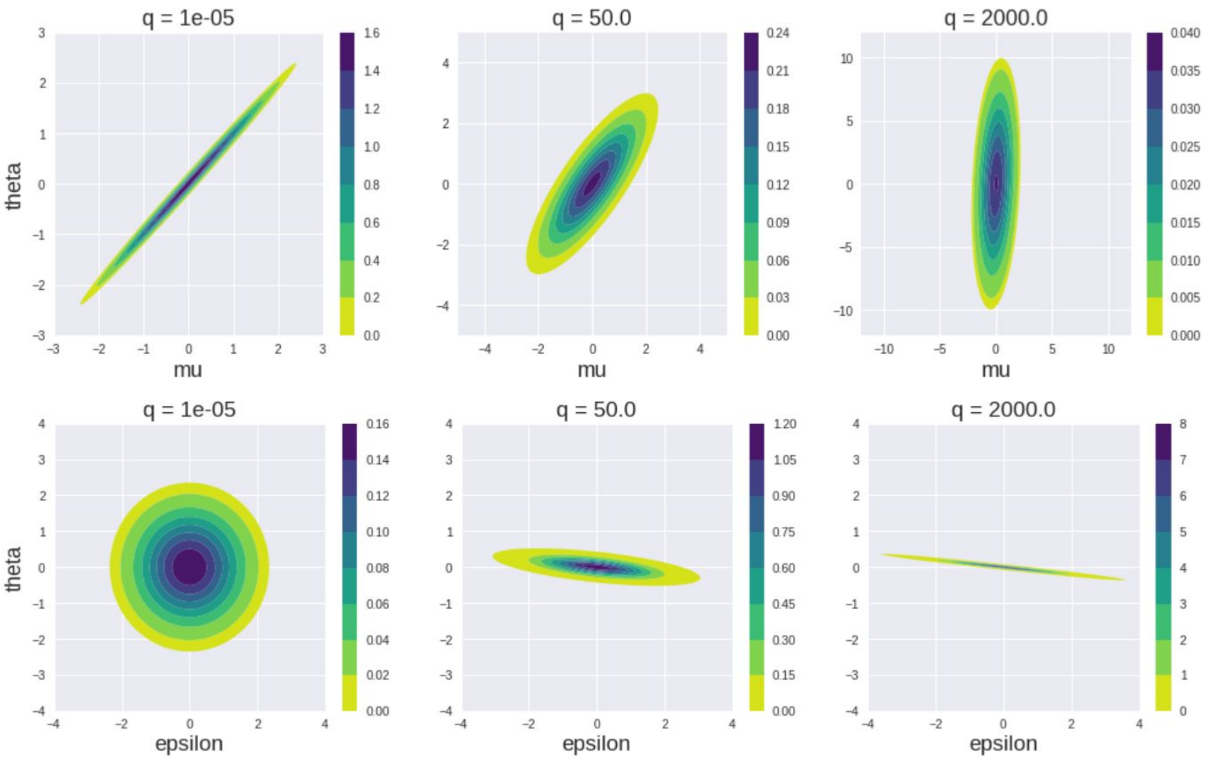
$$\theta \sim \mathcal{N}(0, 1) \qquad \epsilon \sim \mathcal{N}(0, 1) \qquad \mu = \theta + \sigma_\mu \epsilon$$
$$y_n \sim \mathcal{N}(\theta + \sigma_\mu \epsilon, \sigma) \text{ for all } n \in 1 \ldots N$$

Understanding Reparameterisation Effects

$$q = N/\sigma$$

Centred

Non-centred

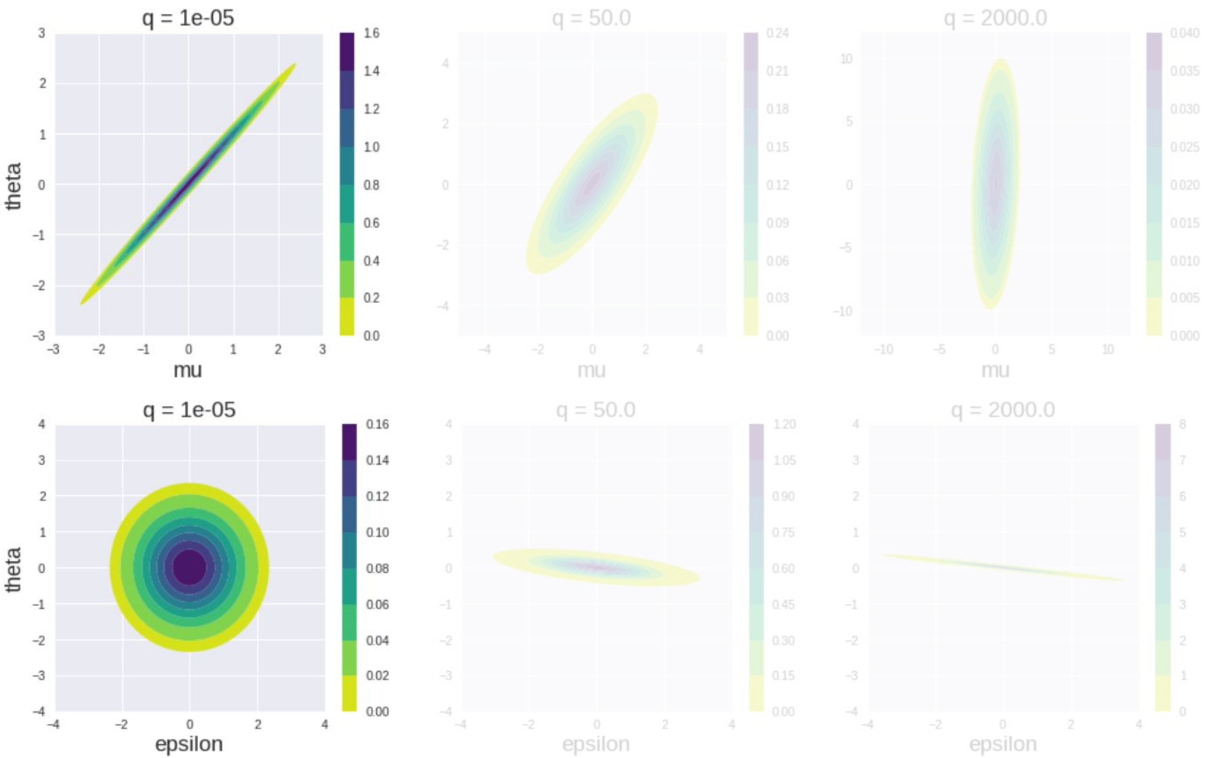


Understanding Reparameterisation Effects

$$q = N/\sigma$$

Centred

Non-centred

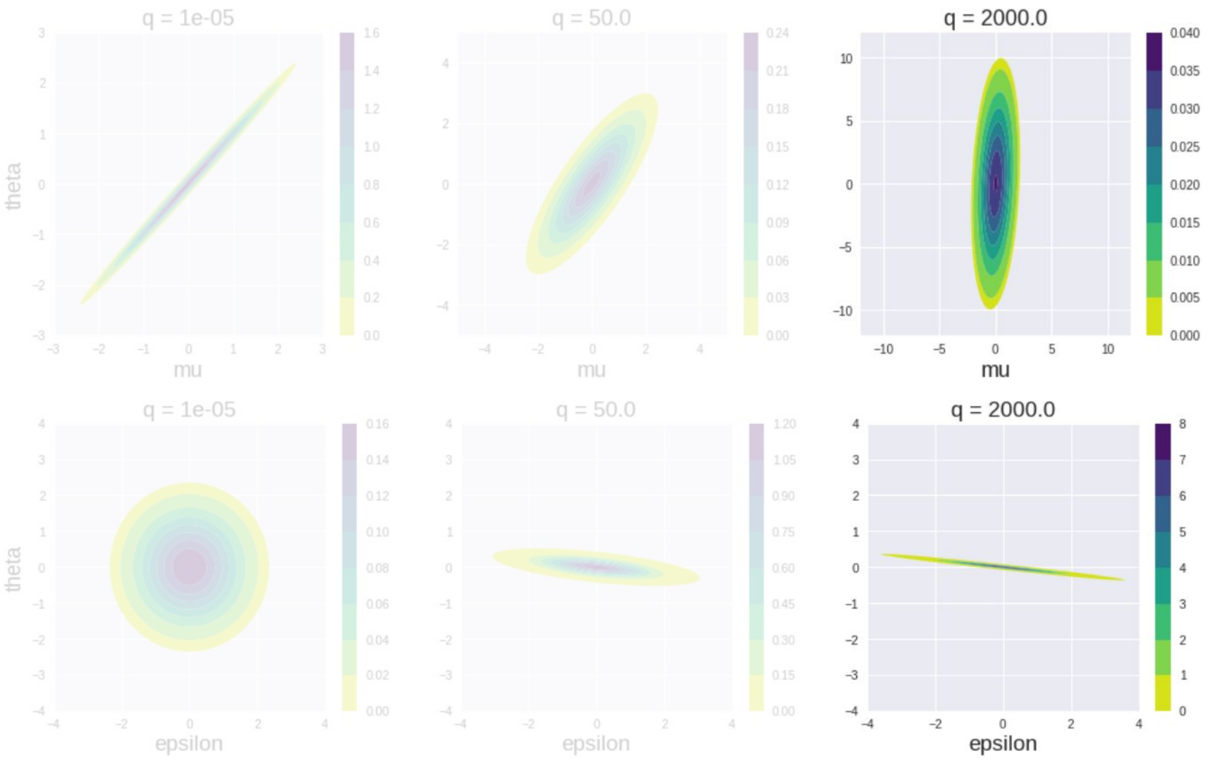


Understanding Reparameterisation Effects

$$q = N/\sigma$$

Centred

Non-centred



Stan diagnostics example

Goal: Free modellers of the need to choose model parameterisation

Reparameterisation in probabilistic programming

Centred

```
def model(N, sigma, sigma_mu):

  theta = Normal(0., 3.)


  mu = Normal(theta, sigma_mu)


  y = Normal(mu, sigma)
  return y
```

Non-centred

```
def model_ncp(N, sigma, sigma_mu):

  theta_std = Normal(0., 1.)
  theta = 3. * theta_std

  mu_std = Normal(0., 1.)
  mu = theta + mu_std * sigma_mu

  y = Normal(mu, sigma)
  return y
```

Algebraic effect handlers

```
handler h {
    read(_) -> "I <3 PPLs"
}
```

```
x = read(file1)
y = read(file2)
print(concatenate(x, y))
```

```
> "Contents of file1Contents of file2"
```

```
with h handle:
    x = read(file1)
    y = read(file2)
    print(concatenate(x, y))
```

```
> "I <3 PPLsI <3 PPLs"
```

Algebraic effect handlers

```
handler h {
   v ~ dist →
      print("I <3 PPLs")
      v = dist.sample()
}

x ~ normal(0, 1)
y ~ normal(0, 1)
print(x * y)

with h handle:
   x ~ normal(0, 1)
   y ~ normal(0, 1)
   print(x * y)
```

```
> -1.891424147896625
```

```
> "I <3 PPLs"
> "I <3 PPLs"
> -1.891424147896625
```

Reparameterisation in probabilistic programming

Effect Handler: A function that may change how and if a RV is constructed

```
def ncp(rv_constr, **args):
  if is_location_scale(rv_constr):
   std = rv_constr(0., 1.)
   return args["scale"] * std + args["loc"]

with handler(ncp):
  theta = Normal(0., 3.)
  mu = Normal(theta, 1.)
```

Reparameterisation in probabilistic programming

Effect Handler: A function that may change how and if a RV is constructed

```
def ncp(rv_constr, **args):
  if is_location_scale(rv_constr):
   std = rv_constr(0., 1.)
   return args[“scale”] * std + args[“loc”]
```

```
with handler(ncp):
  theta = Normal(0., 3.)
  mu = Normal(theta, 1.)
```

=

```
theta_std = Normal(0., 1.)
theta = 3. * theta_std
mu_std = Normal(0., 1.)
mu = mu_std + theta
```

Reparameterisation in probabilistic programming

Centred

```python
def model(N, sigma, sigma_mu):

  theta = Normal(0., 3.)


  mu = Normal(theta, sigma_mu)


  y = Normal(mu, sigma)
  return y
```

Non-centred

```python
def model_ncp(N, sigma, sigma_mu):

  theta_std = Normal(0., 1.)
  theta = 3. * theta_std

  mu_std = Normal(0., 1.)
  mu = theta + mu_std * sigma_mu

  y = Normal(mu, sigma)
  return y
```

Variationally Inferred Parameterisation (VIP)

$$\tilde{z} \sim \mathcal{N}(\lambda\mu, \sigma^{\lambda})$$

$$z = \mu + \sigma^{1-\lambda}(\tilde{z} - \lambda\mu)$$

Variationally Inferred Parameterisation (VIP)

$$\tilde{z} \sim \mathcal{N}(\lambda\mu, \sigma^{\lambda})$$

$$z = \mu + \sigma^{1-\lambda}(\tilde{z} - \lambda\mu)$$

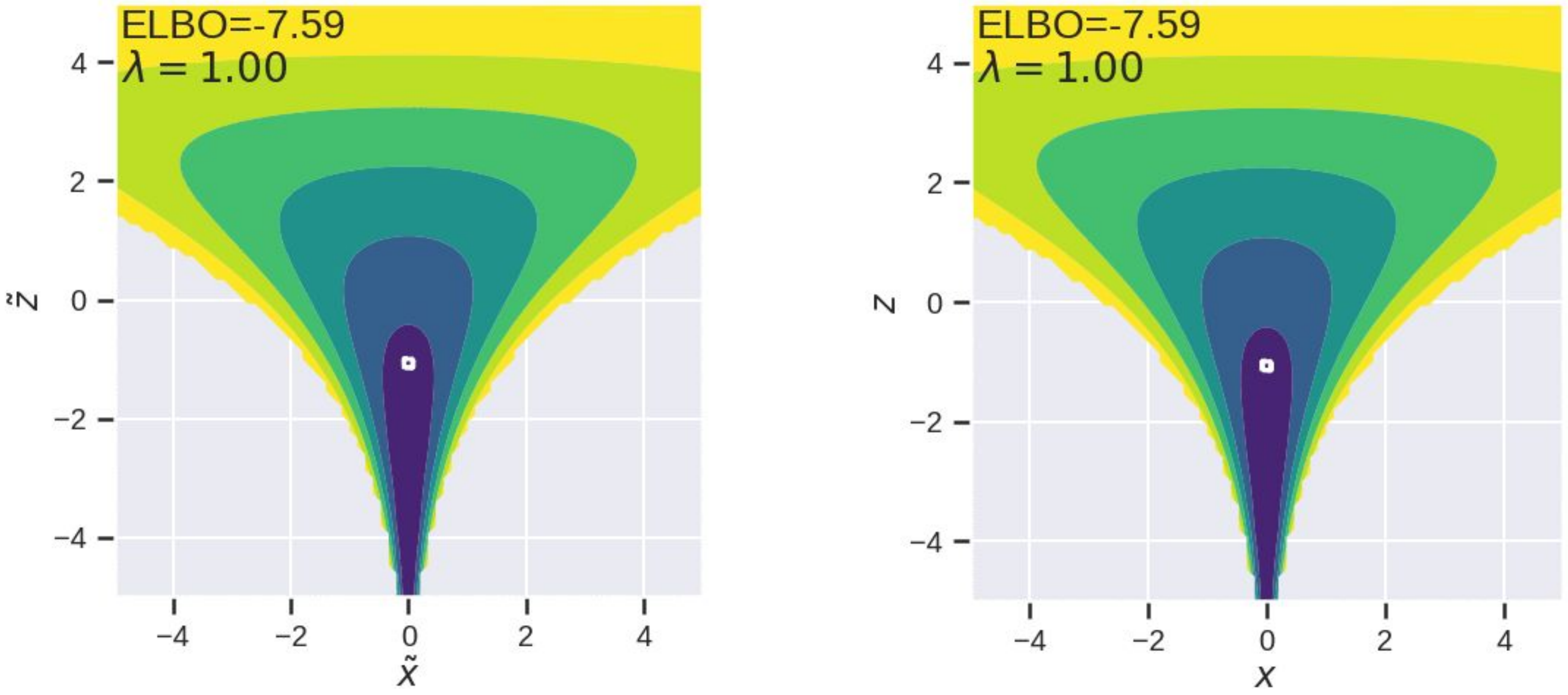
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbb{E}_{q(\tilde{\mathbf{z}};\boldsymbol{\theta})}\left(\log p(\mathbf{x}, \tilde{\mathbf{z}}; \boldsymbol{\lambda}) - \log q(\tilde{\mathbf{z}}; \boldsymbol{\theta})\right)$$

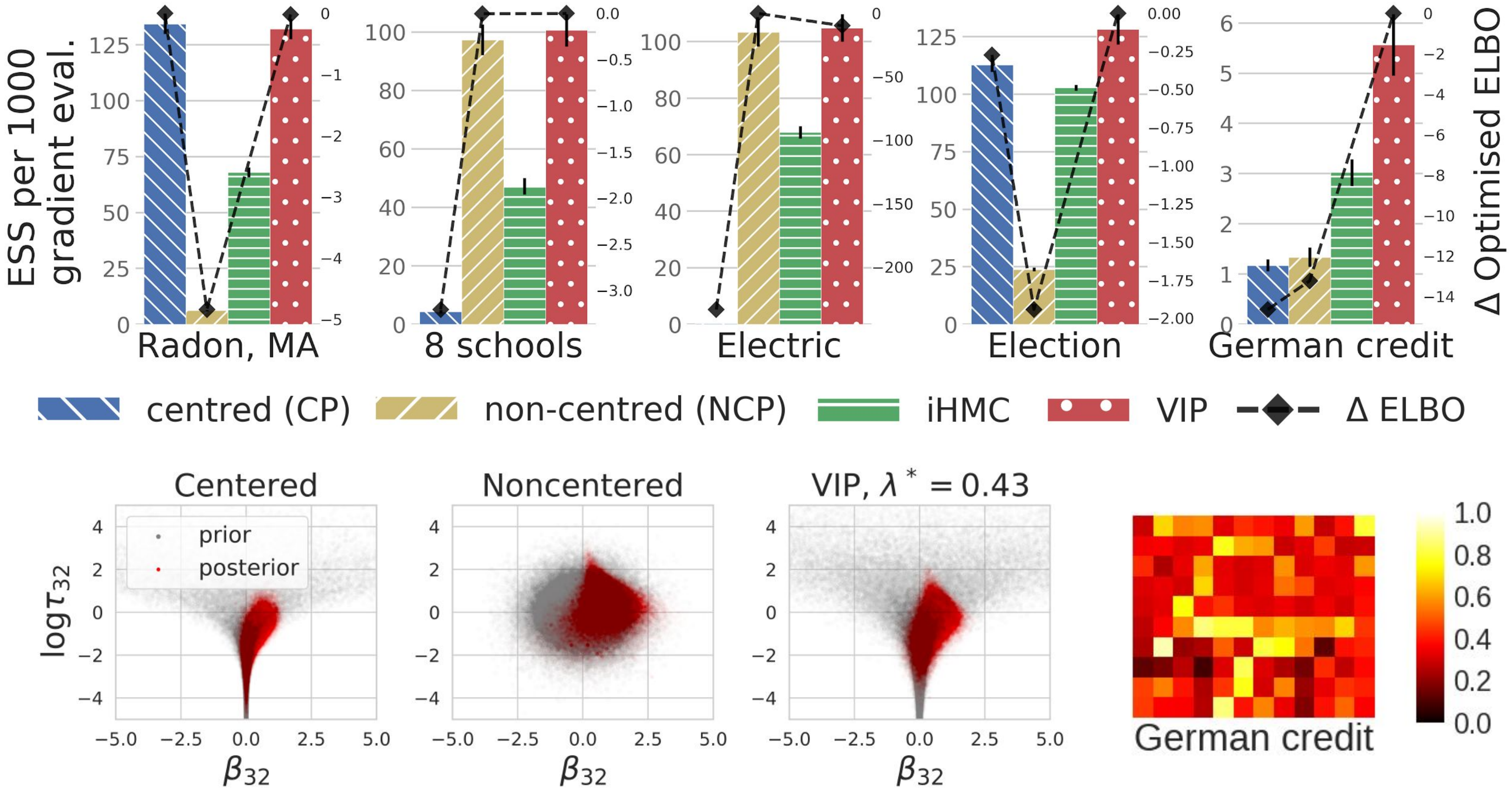
Example: Neal's funnel VIP

$z \sim \mathcal{N}(0, 3) \qquad x \sim \mathcal{N}(0, e^{z/2})$

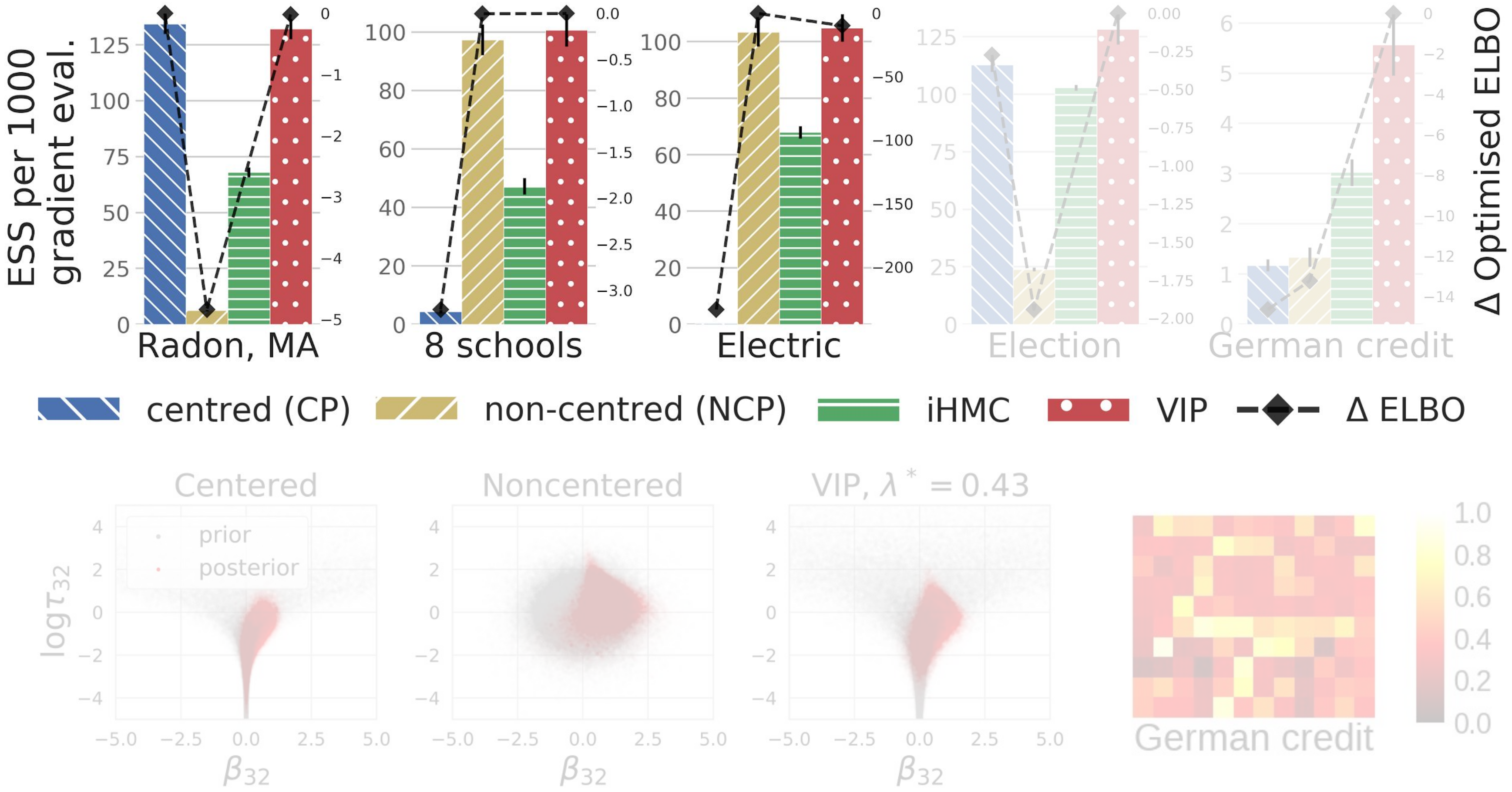
Reparameterized coordinates:

Original coordinates:

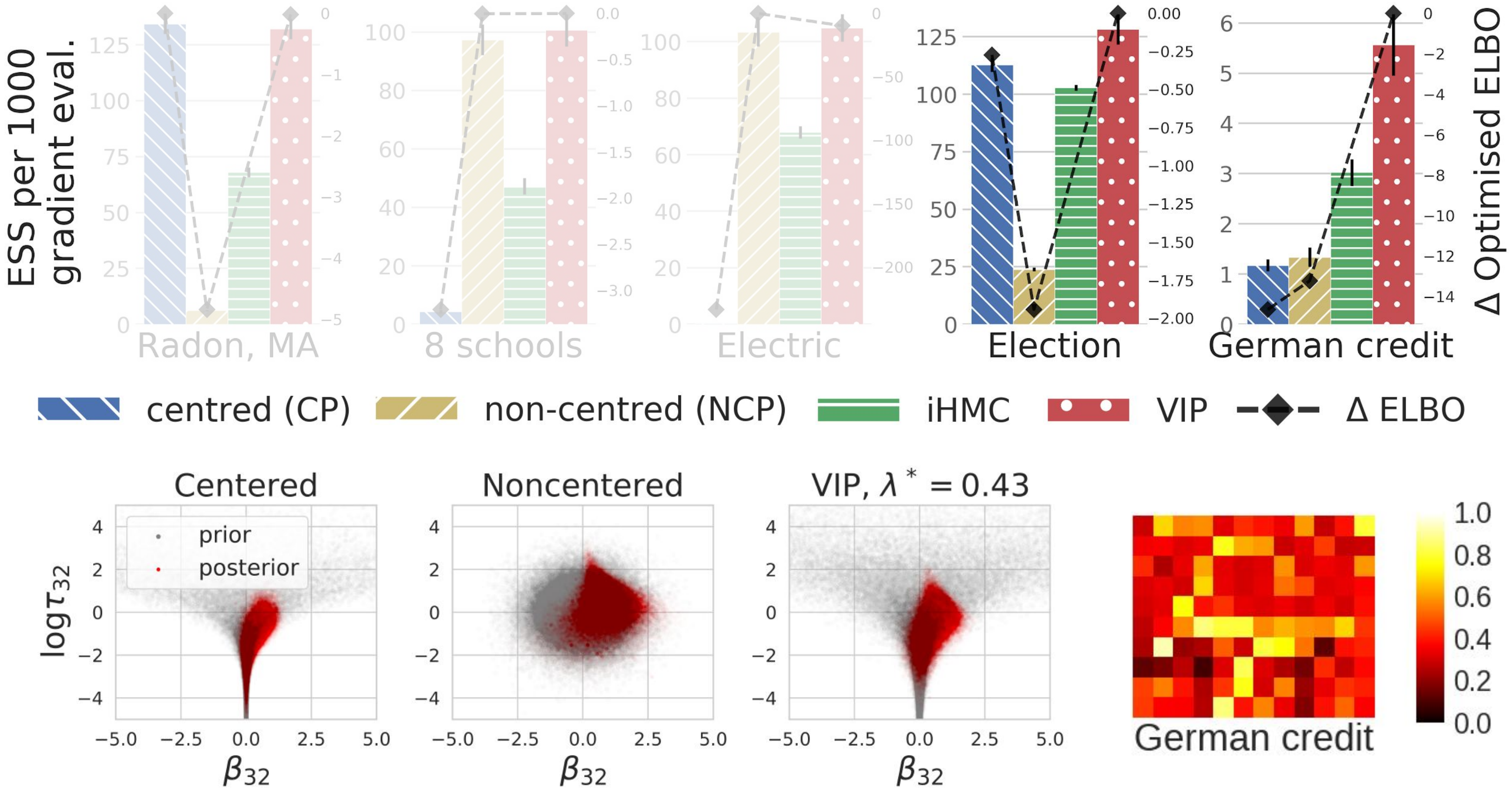




ESS per 1000 gradient eval.
Δ Optimised ELBO
Radon, MA
8 schools
Electric
Election
German credit
centred (CP)
non-centred (NCP)
iHMC
VIP
Δ ELBO
Centered
Noncentered
VIP, $\lambda^* = 0.43$
prior
posterior
$\log \tau_{32}$
β_{32}
German credit

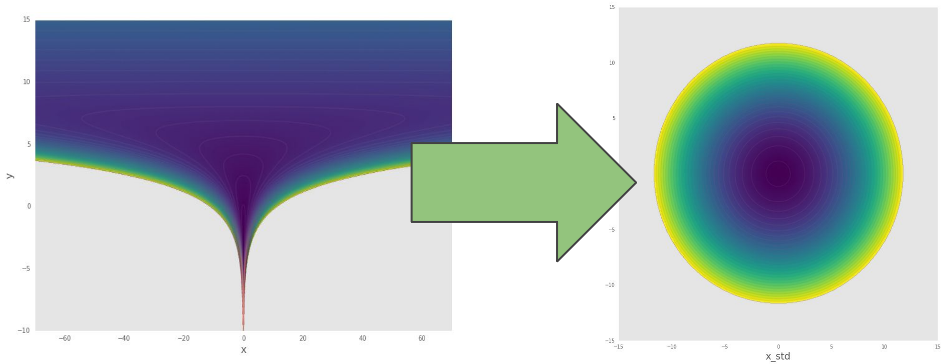


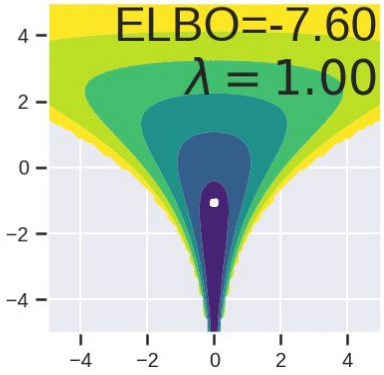
ESS per 1000 gradient eval.
Δ Optimised ELBO
Radon, MA
8 schools
Electric
Election
German credit
centred (CP)
non-centred (NCP)
iHMC
VIP
Δ ELBO
Centered
Noncentered
VIP, $\lambda^* = 0.43$
prior
posterior
$\log \tau_{32}$
β_{32}
German credit


ESS per 1000 gradient eval.
Δ Optimised ELBO
Radon, MA
8 schools
Electric
Election
German credit
centred (CP)
non-centred (NCP)
iHMC
VIP
Δ ELBO
Centered
Noncentered
VIP, λ* = 0.43
prior
posterior
log τ32
β32
German credit

Automatic Reparameterisation in Probabilistic Programming

Maria I. Gorinova, Dave Moore, Matthew D. Hoffman





https://git.io/fpKeO   M.Gorinova@gmail.com   @migorinova

Thank you!