I-priors in Bayesian Variable Selection: From Reproducing Kernel Hilbert Spaces to Hamiltonian Monte Carlo

Haziq Jamil

Social Statistics (Year 3)
London School of Economics & Political Science

3 November 2016

Social Statistics Meeting

follow along at: **https://haziqjamil.github.io/**

Outline

The I-prior Bayesian Variable Selection model

- For centred responses y_i and standardised covariates $x_{i1}, \ldots, x_{ip}$,

$$y_i = \gamma_1 \beta_1 x_{i1} + \cdots + \gamma_p \beta_p x_{ip} + \epsilon_i$$
$$\epsilon_i \sim \mathsf{N}(0, \psi^{-1})$$
$$i = 1, \ldots, n$$

$$\tag{1}$$

$$\underline{\text{Priors}}$$
$$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, \psi \boldsymbol{\Lambda} \mathbf{X}^\top \mathbf{X} \boldsymbol{\Lambda}), \text{ where } \boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_p]$$
$$\gamma_j \sim \mathrm{Bern}(p_j), \ j = 1, \ldots, p$$
$$\psi, \lambda_1^{-2}, \ldots, \lambda_p^{-2} \sim \Gamma(c, d)$$

- Use MCMC methods to sample from posterior using software such as JAGS. Interested in two things:
  - ▶ Posterior model probabilities $\mathrm{P}[\boldsymbol{\gamma} = \boldsymbol{\gamma}' | \mathbf{y}]$ for model $\boldsymbol{\gamma}'$.
  - ▶ Posterior inclusion probabilities $\mathrm{P}[\gamma_j = 1 | \mathbf{y}]$ for variable X_j.

Why Bayesian Variable Selection?

Some criticisms

- The end-game of model selection is often prediction. If so, better methods exist e.g. Lasso

- Why not just put a reasonable prior?

- Unreliable Gibbs sampler - likely to get stuck in multiple modes.

Why Bayesian Variable Selection?

Some criticisms

- The end-game of model selection is often prediction. If so, better methods exist e.g. Lasso
- Why not just put a reasonable prior?
- Unreliable Gibbs sampler - likely to get stuck in multiple modes.

But actually,

- Sometimes there is a need to know what is the most plausible, interpretable, and parsimonious model.
- Valid applications in social sciences, but perhaps not the $p > n$ cases.
- Gibbs sampler not too terrible.
- For as many critics to this "combinatorial approach", there are equally as many proponents.
- Prediction through Bayesian model averaging.

Simulation results are good...
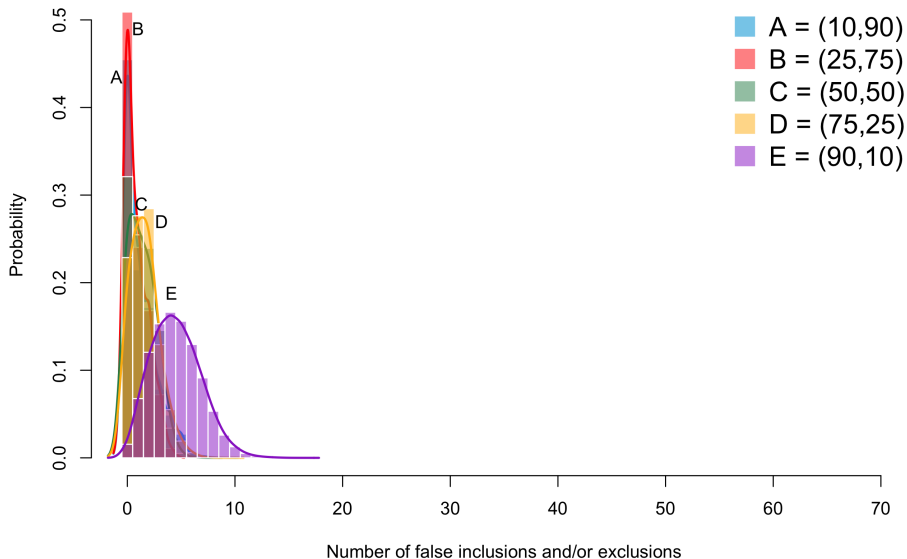
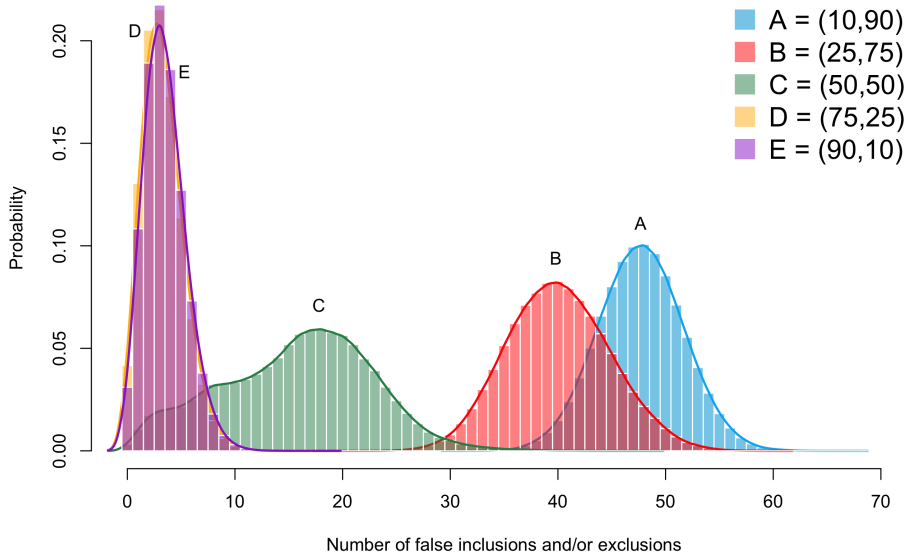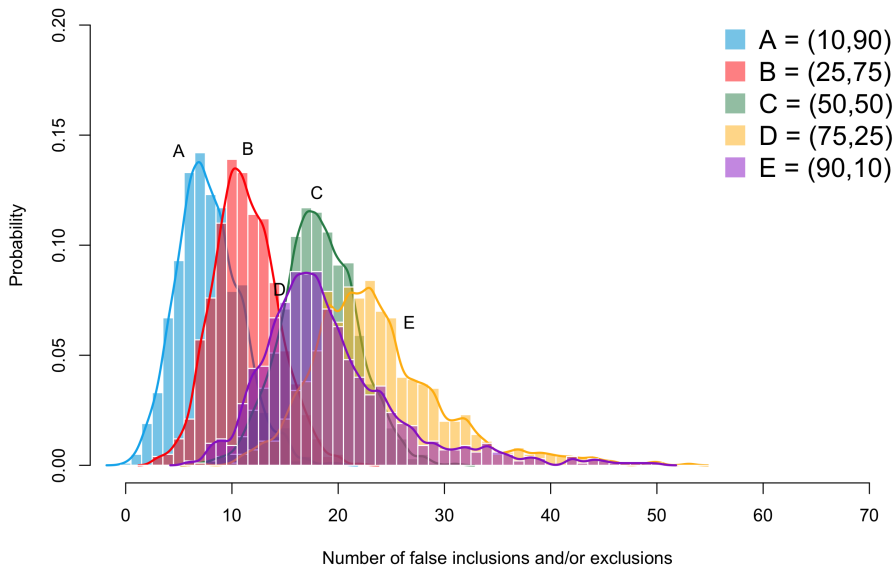# ...in comparison to: SSVS (George & McCulloch, 1993)

Number of false inclusions and/or exclusions
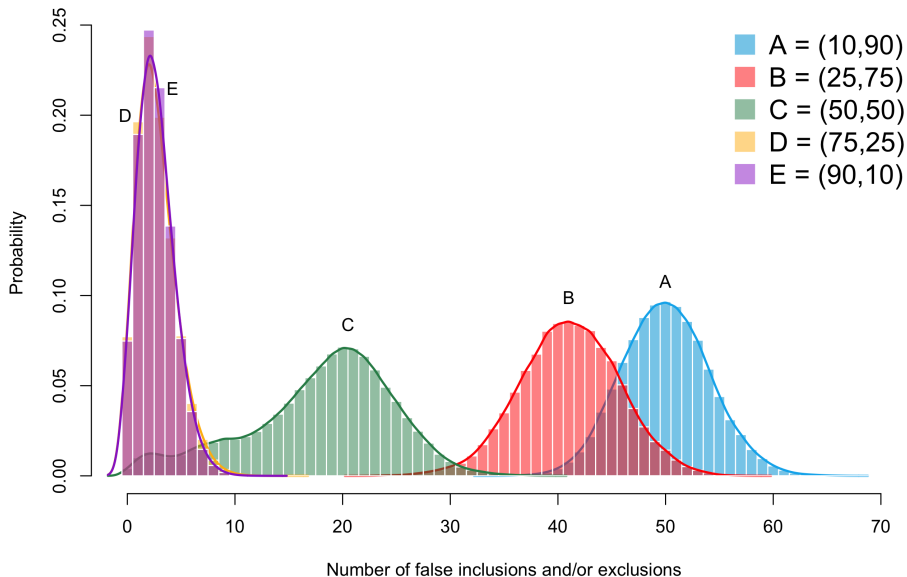
...in comparison to: KM (Kuo & Mallick, 1998)
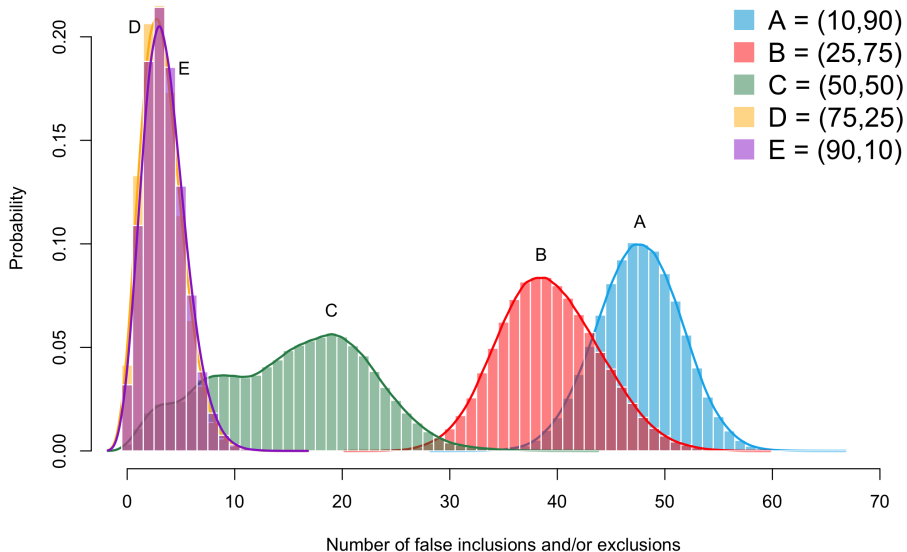
...in comparison to: GVS (Dellaportas et. al., 2011)

...in comparison to: Lasso (Tibshirani, 1994)

...and so are some real world applications

1 **Modelling aerobic fitness through some exercise data** ($n = 30$, $p = 6$) [Kuo and Mallick, 1998]
   ▶ Agreed with forward selection and backward elimination procedure except in the `Age` variable.
   ▶ `Age` negatively correlated with `MaxPulse`.

2 **Effects of air pollution on mortality rate** ($n = 60$, $p = 15$) [McDonald and Schwing, 1973]
   ▶ Which of `HC`, `NOx`, and/or `SO2` affects mortality rate in U.S. metropolitan areas?
   ▶ Agreed with "ridge trace analysis" in identifying `SO2`.

3 **Factors affecting ozone depletion** ($n = 178$, $p = 12, 90$) [Casella and Moreno, 2006]
   ▶ Model obtained had smaller out-of-sample RMSE.
   ▶ Selection of squared and two-way interaction terms to improve RMSE without overcomplicating the model.

1 Bayesian Variable Selection

2 I-priors

3 Bayesian I-prior linear models

4 Hamiltonian Monte Carlo

5 Summary

Introduction

- For $i = 1, \ldots, n$, consider the regression model

$$y_i = \alpha + f(\mathbf{x}_i) + \epsilon_i$$
$$(\epsilon_1, \ldots, \epsilon_n) \sim \mathsf{N}(\mathbf{0}, \mathbf{\Psi}^{-1})$$

where $f \in \mathcal{F}$, $y_i \in \mathbb{R}$, and $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathcal{X}$.

Introduction

- For $i = 1, \ldots, n$, consider the regression model

$$y_i = \alpha + f(\mathbf{x}_i) + \epsilon_i$$
$$(\epsilon_1, \ldots, \epsilon_n) \sim \mathsf{N}(\mathbf{0}, \mathbf{\Psi}^{-1})$$

where $f \in \mathcal{F}$, $y_i \in \mathbb{R}$, and $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathcal{X}$.

- Definition **(I-priors)**

  For the regression model above, let $\mathcal{F}$ be a reproducing kernel Hilbert space (RKHS) with kernel $h_{\boldsymbol{\lambda}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Then, assuming it exists, the Fisher information for $I[f]$ for the function f is given by

$$I[f(\mathbf{x}_i), f(\mathbf{x}_i')] = \sum_{k=1}^{n} \sum_{l=1}^{n} \psi_{kl} h_{\boldsymbol{\lambda}}(\mathbf{x}_i, \mathbf{x}_k) h_{\boldsymbol{\lambda}}(\mathbf{x}_i', \mathbf{x}_l).$$

  Let π be a Gaussian distribution on the random vector f with mean f_0 and covariance kernel $I[f]$. Then π is called an I-prior for f.

Function spaces and kernels

- There is a bijection between the set of all positive-definite functions (reproducing kernels) $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and the set of all RKHS.

$\mathcal{X} = \{x_i\}$	**Effect**	**Vector space** $\mathcal{F}$	**Kernel** $h(x_i, x_k)$
Real	"Straight line" functions	Canonical	$x_i x_k$
Real	"Curvy" functions (smoothing)	Fractional Brownian Motion (FBM)	$\|x_i\|^{2\gamma} + \|x_k\|^{2\gamma} - \|x_i - x_k\|^{2\gamma}$ with $\gamma \in (0, 1)$
Nominal	Grouping	Pearson	$\frac{\mathbb{1}[x_i = x_k]}{p_i} - 1$ where $p_i = \mathsf{P}[X = x_i]$

The w I-prior model

- The I-prior for f has the random-effect representation

$$f(\mathbf{x}_i) = \alpha + f_0(\mathbf{x}_i) + \sum_{k=1}^{n} h_\lambda(\mathbf{x}_i, \mathbf{x}_k) w_k$$

$$(w_1, \ldots, w_n) \sim \mathrm{N}(\mathbf{0}, \mathbf{\Psi})$$

The w I-prior model

- The I-prior for f has the random-effect representation

$$f(\mathbf{x}_i) = \alpha + f_0(\mathbf{x}_i) + \sum_{k=1}^{n} h_\lambda(\mathbf{x}_i, \mathbf{x}_k) w_k$$

$$(w_1, \ldots, w_n) \sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Psi})$$

- Putting this back into our regression model, we obtain the w *I-prior* model

$$y_i = \alpha + f_0(\mathbf{x}_i) + \sum_{k=1}^{n} h_\lambda(\mathbf{x}_i, \mathbf{x}_k) w_k + \epsilon_i$$

$$(w_1, \ldots, w_n) \sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Psi})$$

$$(\epsilon_1, \ldots, \epsilon_n) \sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Psi}^{-1})$$

The *w* I-prior model

- The I-prior for f has the random-effect representation

$$f(\mathbf{x}_i) = \alpha + f_0(\mathbf{x}_i) + \sum_{k=1}^{n} h_\lambda(\mathbf{x}_i, \mathbf{x}_k) w_k$$

$$(w_1, \ldots, w_n) \sim \mathrm{N}(\mathbf{0}, \mathbf{\Psi})$$

- Putting this back into our regression model, we obtain the *w I-prior* model

$$\mathbf{y} = \boldsymbol{\alpha} + \mathbf{f}_0 + \mathbf{H}_\lambda \mathbf{w} + \boldsymbol{\epsilon}$$

$$\mathbf{w} \sim \mathrm{N}(\mathbf{0}, \mathbf{\Psi})$$

$$\boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \mathbf{\Psi}^{-1})$$

The w I-prior model

- The I-prior for f has the random-effect representation

$$f(\mathbf{x}_i) = \alpha + f_0(\mathbf{x}_i) + \sum_{k=1}^{n} h_{\boldsymbol{\lambda}}(\mathbf{x}_i, \mathbf{x}_k) w_k$$

$$(w_1, \ldots, w_n) \sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Psi})$$

- Putting this back into our regression model, we obtain the w *I-prior* model

$$\mathbf{y} = \boldsymbol{\alpha} + \mathbf{f}_0 + \mathbf{H}_{\boldsymbol{\lambda}}\mathbf{w} + \boldsymbol{\epsilon}$$

$$\mathbf{w} \sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Psi})$$

$$\boldsymbol{\epsilon} \sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Psi}^{-1})$$

- Typically, $(\mathbf{H}_{\boldsymbol{\lambda}})_{ij} = \sum_{k=1}^{p} \lambda_k h_k(x_{ik}, x_{jk})$

The *w* I-prior model

- The I-prior for f has the random-effect representation

$$f(\mathbf{x}_i) = \alpha + f_0(\mathbf{x}_i) + \sum_{k=1}^{n} h_\lambda(\mathbf{x}_i, \mathbf{x}_k) w_k$$

$$(w_1, \ldots, w_n) \sim N(\mathbf{0}, \mathbf{\Psi})$$

- Putting this back into our regression model, we obtain the *w I-prior* model

$$\mathbf{y} = \boldsymbol{\alpha} + \mathbf{f}_0 + \mathbf{H}_\lambda \mathbf{w} + \boldsymbol{\epsilon}$$

$$\mathbf{w} \sim N(\mathbf{0}, \mathbf{\Psi})$$

$$\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{\Psi}^{-1})$$

- Typically, $\mathbf{H}_\lambda = \lambda_1 \mathbf{H}_1 + \cdots + \lambda_p \mathbf{H}_p$

The w I-prior model

- The I-prior for f has the random-effect representation

$$f(\mathbf{x}_i) = \alpha + f_0(\mathbf{x}_i) + \sum_{k=1}^{n} h_\lambda(\mathbf{x}_i, \mathbf{x}_k) w_k$$

$$(w_1, \ldots, w_n) \sim N(\mathbf{0}, \mathbf{\Psi})$$

- Putting this back into our regression model, we obtain the w I-prior model

$$\mathbf{y} = \alpha + \mathbf{f}_0 + \mathbf{H}_\lambda \mathbf{w} + \boldsymbol{\epsilon}$$

$$\mathbf{w} \sim N(\mathbf{0}, \mathbf{\Psi})$$

$$\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{\Psi}^{-1})$$

- Typically, $\mathbf{H}_\lambda = \lambda_1 \mathbf{H}_1 + \cdots + \lambda_p \mathbf{H}_p$, $\mathbf{\Psi} = \psi \mathbf{I}_n$, and $\mathbf{f}_0 = \mathbf{0}$.

The *w* I-prior model

- The I-prior for f has the random-effect representation

$$f(\mathbf{x}_i) = \alpha + f_0(\mathbf{x}_i) + \sum_{k=1}^{n} h_\lambda(\mathbf{x}_i, \mathbf{x}_k) w_k$$

$$(w_1, \ldots, w_n) \sim \mathsf{N}(\mathbf{0}, \mathbf{\Psi})$$

- Putting this back into our regression model, we obtain the *w I-prior* model

$$\mathbf{y} = \alpha + \mathbf{f}_0 + \mathbf{H}_\lambda \mathbf{w} + \boldsymbol{\epsilon}$$

$$\mathbf{w} \sim \mathsf{N}(\mathbf{0}, \mathbf{\Psi})$$

$$\boldsymbol{\epsilon} \sim \mathsf{N}(\mathbf{0}, \mathbf{\Psi}^{-1})$$

- Typically, $\mathbf{H}_\lambda = \lambda_1 \mathbf{H}_1 + \cdots + \lambda_p \mathbf{H}_p$, $\mathbf{\Psi} = \psi \mathbf{I}_n$, and $\mathbf{f}_0 = \mathbf{0}$.
- Parameters of interest are $\boldsymbol{\theta} = (\alpha, \lambda_1, \ldots, \lambda_p, \psi)$.

Maximum likelihood

- The marginal distribution of **y** is normal with mean and variance

$$E[\mathbf{y}] = \boldsymbol{\alpha}$$
$$\mathrm{Var}[\mathbf{y}] = \psi \mathbf{H}_\lambda^2 + \psi^{-1} \mathbf{I}_n =: \mathbf{V}_y$$

Maximum likelihood

- The marginal distribution of $\mathbf{y}$ is normal with mean and variance

$$\mathsf{E}[\mathbf{y}] = \boldsymbol{\alpha}$$
$$\mathsf{Var}[\mathbf{y}] = \psi \mathbf{H}_{\boldsymbol{\lambda}}^2 + \psi^{-1} \mathbf{I}_n =: \mathbf{V}_y$$

and thus, the marginal log-likelihood is given by

$$l(\boldsymbol{\theta}) = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{V}_y| - \frac{1}{2}(\mathbf{y} - \boldsymbol{\alpha})^\top \mathbf{V}_y^{-1}(\mathbf{y} - \boldsymbol{\alpha}).$$

Maximum likelihood

- The marginal distribution of $\mathbf{y}$ is normal with mean and variance

$$\mathrm{E}[\mathbf{y}] = \boldsymbol{\alpha}$$
$$\mathrm{Var}[\mathbf{y}] = \psi \mathbf{H}_{\boldsymbol{\lambda}}^2 + \psi^{-1} \mathbf{I}_n =: \mathbf{V}_y$$

and thus, the marginal log-likelihood is given by

$$l(\boldsymbol{\theta}) = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{V}_y| - \frac{1}{2}(\mathbf{y} - \boldsymbol{\alpha})^\top \mathbf{V}_y^{-1}(\mathbf{y} - \boldsymbol{\alpha}).$$

- MLE for intercept is $\hat{\alpha} := \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i.$

Maximum likelihood

- The marginal distribution of $\mathbf{y}$ is normal with mean and variance

$$\mathsf{E}[\mathbf{y}] = \boldsymbol{\alpha}$$
$$\mathsf{Var}[\mathbf{y}] = \psi\mathbf{H}_{\boldsymbol{\lambda}}^2 + \psi^{-1}\mathbf{I}_n =: \mathbf{V}_y$$

and thus, the marginal log-likelihood is given by

$$l(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{V}_y| - \frac{1}{2}(\mathbf{y} - \boldsymbol{\alpha})^{\top}\mathbf{V}_y^{-1}(\mathbf{y} - \boldsymbol{\alpha}).$$

- MLE for intercept is $\hat{\alpha} := \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.
- Usually no closed form estimates for $\boldsymbol{\lambda}$ and ψ, so use numerical optimisation to find MLE.

Maximum likelihood

- The marginal distribution of $\mathbf{y}$ is normal with mean and variance

$$E[\mathbf{y}] = \boldsymbol{\alpha}$$
$$\text{Var}[\mathbf{y}] = \psi \mathbf{H}_{\lambda}^2 + \psi^{-1} \mathbf{I}_n =: \mathbf{V}_y$$

  and thus, the marginal log-likelihood is given by

$$l(\boldsymbol{\theta}) = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{V}_y| - \frac{1}{2}(\mathbf{y} - \boldsymbol{\alpha})^\top \mathbf{V}_y^{-1}(\mathbf{y} - \boldsymbol{\alpha}).$$

- MLE for intercept is $\hat{\alpha} := \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$.
- Usually no closed form estimates for $\boldsymbol{\lambda}$ and ψ, so use numerical optimisation to find MLE.
- *Problem: Convergence is difficult when there are a lot of scale parameters.*

EM algorithm

- A more stable method is using the EM algorithm. Treat the random effects **w** as "missing".

EM algorithm

- A more stable method is using the EM algorithm. Treat the random effects $\mathbf{w}$ as "missing".

- The relevant distributions are easy enough to obtain:
    - $\mathbf{y} \sim \mathsf{N}(\boldsymbol{\alpha}, \mathbf{V}_y)$
    - $\mathbf{w} \sim \mathsf{N}(\mathbf{0}, \psi \mathbf{I}_n)$
    - $\begin{pmatrix} \mathbf{y} \\ \mathbf{w} \end{pmatrix} \sim \mathsf{N}\left(\begin{pmatrix} \boldsymbol{\alpha} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{V}_y & \psi \mathbf{H}_{\boldsymbol{\lambda}} \\ \psi \mathbf{H}_{\boldsymbol{\lambda}} & \psi \mathbf{I}_n \end{pmatrix} \right)$
    - $\mathbf{w}|\mathbf{y} \sim \mathsf{N}\left(\psi \mathbf{H}_{\boldsymbol{\lambda}} \mathbf{V}_y^{-1}(\mathbf{y} - \boldsymbol{\alpha}), \mathbf{V}_y^{-1} \right)$

EM algorithm

- A more stable method is using the EM algorithm. Treat the random effects $\mathbf{w}$ as "missing".
- The relevant distributions are easy enough to obtain:
  - $\mathbf{y} \sim \mathrm{N}(\boldsymbol{\alpha}, \mathbf{V}_y)$
  - $\mathbf{w} \sim \mathrm{N}(\mathbf{0}, \psi \mathbf{I}_n)$
  - $\begin{pmatrix} \mathbf{y} \\ \mathbf{w} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \boldsymbol{\alpha} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{V}_y & \psi \mathbf{H}_{\boldsymbol{\lambda}} \\ \psi \mathbf{H}_{\boldsymbol{\lambda}} & \psi \mathbf{I}_n \end{pmatrix} \right)$
  - $\mathbf{w}|\mathbf{y} \sim \mathrm{N}\left(\psi \mathbf{H}_{\boldsymbol{\lambda}} \mathbf{V}_y^{-1}(\mathbf{y} - \boldsymbol{\alpha}), \mathbf{V}_y^{-1} \right)$
- For $t = 0, 1, \ldots,$ do:
  - E-step: Calculate $Q(\boldsymbol{\lambda}, \psi) = \mathsf{E}_{\mathbf{w}}\left[\log f(\mathbf{y}, \mathbf{w}; \boldsymbol{\theta}) | \mathbf{y}; \boldsymbol{\lambda}^{(t)}, \psi^{(t)}, \hat{\alpha} \right]$.
  - M-step: $(\boldsymbol{\lambda}^{(t+1)}, \psi^{(t+1)}) \leftarrow \arg\max_{(\boldsymbol{\lambda}, \psi)} Q(\boldsymbol{\lambda}, \psi)$.

EM algorithm

- A more stable method is using the EM algorithm. Treat the random effects $\mathbf{w}$ as "missing".
- The relevant distributions are easy enough to obtain:
  - $\mathbf{y} \sim \mathsf{N}(\boldsymbol{\alpha}, \mathbf{V}_y)$
  - $\mathbf{w} \sim \mathsf{N}(\mathbf{0}, \psi\mathbf{I}_n)$
  - $\begin{pmatrix} \mathbf{y} \\ \mathbf{w} \end{pmatrix} \sim \mathsf{N}\left(\begin{pmatrix} \boldsymbol{\alpha} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{V}_y & \psi\mathbf{H}_{\boldsymbol{\lambda}} \\ \psi\mathbf{H}_{\boldsymbol{\lambda}} & \psi\mathbf{I}_n \end{pmatrix} \right)$
  - $\mathbf{w}|\mathbf{y} \sim \mathsf{N}\left(\psi\mathbf{H}_{\boldsymbol{\lambda}}\mathbf{V}_y^{-1}(\mathbf{y} - \boldsymbol{\alpha}), \mathbf{V}_y^{-1} \right)$
- For $t = 0, 1, \ldots,$ do:
  - E-step: Calculate $Q(\boldsymbol{\lambda}, \psi) = \mathsf{E}_{\mathbf{w}}\left[\log f(\mathbf{y}, \mathbf{w}; \boldsymbol{\theta})|\mathbf{y}; \boldsymbol{\lambda}^{(t)}, \psi^{(t)}, \hat{\alpha} \right]$.
  - M-step: $(\boldsymbol{\lambda}^{(t+1)}, \psi^{(t+1)}) \leftarrow \arg\max_{(\boldsymbol{\lambda}, \psi)} Q(\boldsymbol{\lambda}, \psi)$.
- *Problem: May be very slow to converge.*

The R/iprior package

- An R package for regression modelling using I-priors.
    - ▶ Similar syntax to R's `lm()`.
    - ▶ Parameters estimated using maximum likelihood.
    - ▶ Available on CRAN and GitHub.

The R/iprior package

- An R package for regression modelling using I-priors.
  - ▸ Similar syntax to R's `lm()`.
  - ▸ Parameters estimated using maximum likelihood.
  - ▸ Available on CRAN and GitHub.

- Example: Look at how students' mathematics achievement varies across different high schools (High School & Beyond dataset).

```
str(hsbsmall)

## 'data.frame': 661 obs. of  3 variables:
## $ mathach : num  16.663 -2.155 0.085 18.804 2.409 ...
## $ ses     : num  0.322 0.212 0.682 -0.148 -0.468 0.842 ..
## $ schoolid: Factor w/ 16 levels "1374","1433",..: 1 1 1..
```

The R/iprior package

- Fit a straight line regressing `mathach` against `ses`.

```
system.time(
  mod <- iprior(mathach ~ ses, data = hsbsmall)
)

## Iteration 0:    Log-likelihood = -19755.905 .......
## Iteration 100:  Log-likelihood = -2169.8515 ........
## Iteration 200:  Log-likelihood = -2169.8481 ....
## Iteration 258:  Log-likelihood = -2169.8481
## EM complete.
##     user   system  elapsed
##   90.677    1.946   92.752
```

The R/iprior package

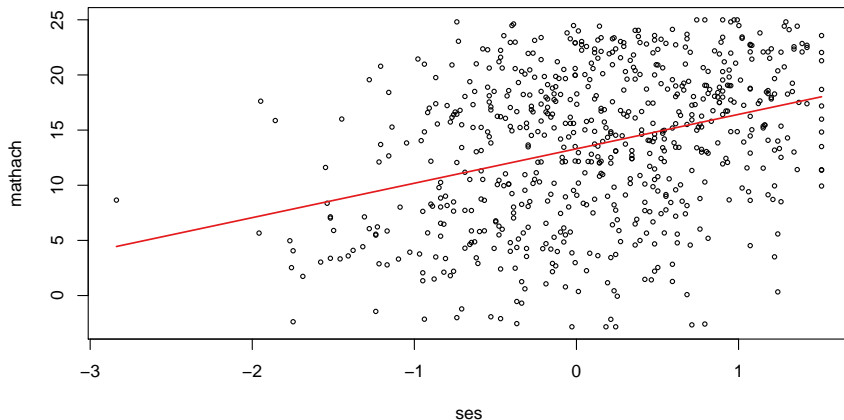- Obtain the parameter estimates. Can also do summary(mod).

```
print(mod)

##
## Call:
## iprior(formula = mathach ~ ses, data = hsbsmall)
##
## RKHS used: Canonical, with a single scale parameter.
##
##
## Parameter estimates:
## (Intercept)        lambda           psi
## 13.68325416    1.06084515    0.02421674
```

The R/iprior package
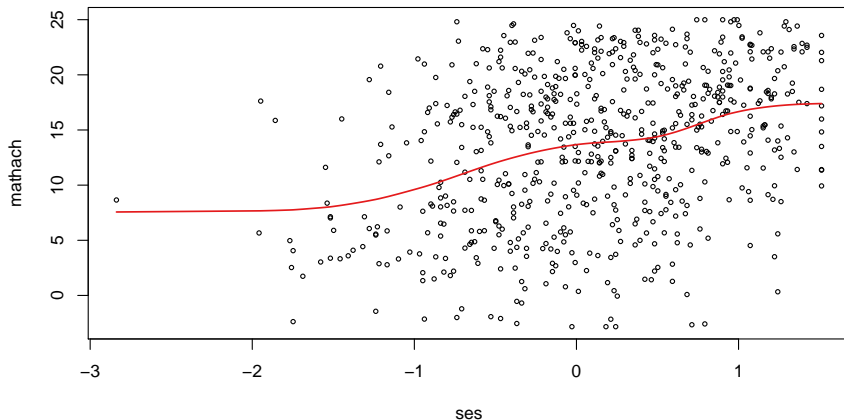
```
plot(mod, plots = "fitted")
```

Fitted regression curve

The R/iprior package

```
plot(
  iprior(mathach ~ ses, hsbsmall, model = list(kernel = "FBM")
)
```

Fitted regression curve

The R/iprior package

```
plot(
  iprior(mathach ~ ses + schoolid + ses:schoolid, hsbsmall)
)
```

Fitted regression curve

The R/iprior package

```
plot(
   iprior(mathach ~ . ^ 2, hsbsmall, model = list(kernel = "FBM"))
)
```
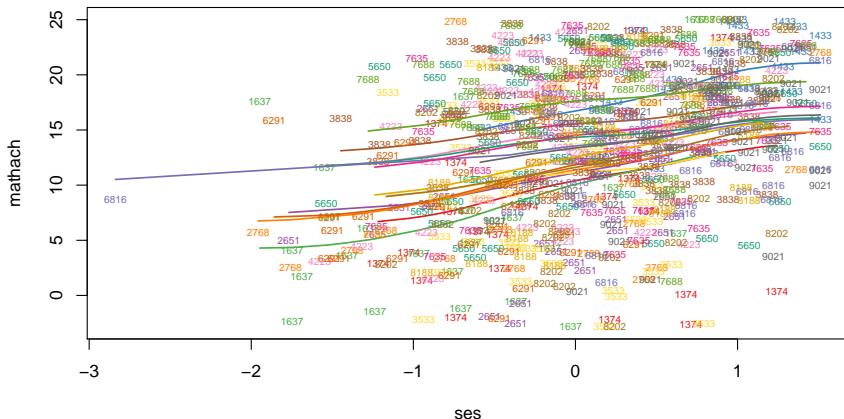
Fitted regression curve

The R/iprior package

- Compare mean squared errors and log-likelihood values of models.

```
##            Canonical       FBM Can. w/ intr FBM w/ intr
## MSE           41.232     40.86       34.809       34.31
## logLik     -2169.850  -2171.18    -2137.800    -2138.64
```

The R/iprior package

- Compare mean squared errors and log-likelihood values of models.

```
##           Canonical        FBM  Can. w/ intr  FBM w/ intr
## MSE         41.232       40.86        34.809        34.31
## logLik   -2169.850   -2171.18     -2137.800     -2138.64
```

- Other things available:
    - ▸ fitted() for fitted values.
    - ▸ predict() for fitted values of a new set of covariates.
    - ▸ resid() for model residuals.
    - ▸ logLik() and deviance() for model log-likelihood and deviance values respectively.
    - ▸ ipriorOptim() is a routine which combines EM algorithm and direct optimisation.

1 Bayesian Variable Selection

2 I-priors

3 Bayesian I-prior linear models

4 Hamiltonian Monte Carlo

5 Summary

The beta I-prior (linear) model

- For "straight line" functions in the Canonical RKHS, its kernel
  $h_\lambda : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined as

$$h_\lambda(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p} \lambda_k x_{ik} x_{jk}$$

The beta I-prior (linear) model

- For "straight line" functions in the Canonical RKHS, its kernel $h_\lambda : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined as

$$\mathbf{H}_\lambda = \lambda_1 \mathbf{X}_1 \mathbf{X}_1^\top + \cdots + \lambda_p \mathbf{X}_p \mathbf{X}_p^\top$$

The beta I-prior (linear) model

- For "straight line" functions in the Canonical RKHS, its kernel $h_\lambda : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined as

$$\begin{aligned} \mathbf{H}_\lambda &= \lambda_1 \mathbf{X}_1 \mathbf{X}_1^\top + \cdots + \lambda_p \mathbf{X}_p \mathbf{X}_p^\top \\ &= \mathbf{X} \mathbf{\Lambda} \mathbf{X}^\top, \end{aligned}$$

where $\mathbf{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_p]$.

The beta I-prior (linear) model

- For "straight line" functions in the Canonical RKHS, its kernel $h_\lambda : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined as

$$\mathbf{H}_\lambda = \lambda_1 \mathbf{X}_1 \mathbf{X}_1^\top + \cdots + \lambda_p \mathbf{X}_p \mathbf{X}_p^\top$$
$$= \mathbf{X} \mathbf{\Lambda} \mathbf{X}^\top,$$

  where $\mathbf{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_p]$.

- Putting this into the w I-prior model we have

$$\mathbf{y} = \boldsymbol{\alpha} + \mathbf{H}_\lambda \mathbf{w} + \boldsymbol{\epsilon}$$
$$= \boldsymbol{\alpha} + \mathbf{X} \underbrace{\mathbf{\Lambda} \mathbf{X}^\top \mathbf{w}}_{\beta} + \boldsymbol{\epsilon}$$

The beta I-prior (linear) model

- For "straight line" functions in the Canonical RKHS, its kernel $h_\lambda : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined as

$$\mathbf{H}_\lambda = \lambda_1 \mathbf{X}_1 \mathbf{X}_1^\top + \cdots + \lambda_p \mathbf{X}_p \mathbf{X}_p^\top$$
$$= \mathbf{X} \mathbf{\Lambda} \mathbf{X}^\top,$$

where $\mathbf{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_p]$.

- Putting this into the w I-prior model we have

$$\mathbf{y} = \boldsymbol{\alpha} + \mathbf{H}_\lambda \mathbf{w} + \boldsymbol{\epsilon}$$
$$= \boldsymbol{\alpha} + \mathbf{X} \underbrace{\mathbf{\Lambda} \mathbf{X}^\top \mathbf{w}}_{\beta} + \boldsymbol{\epsilon}$$

which implies $\mathsf{E}[\boldsymbol{\beta}] = \mathbf{0}$ and $\mathsf{Var}[\boldsymbol{\beta}] = \psi \mathbf{\Lambda} \mathbf{X}^\top \mathbf{X} \mathbf{\Lambda}$.

The beta I-prior (linear) model cont.

- The standard multiple regression model with an I-prior on $\boldsymbol{\beta}$

$$
\begin{aligned}
y_i &= \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i \\
\boldsymbol{\beta} &\sim \mathsf{N}(\mathbf{0}, \psi \boldsymbol{\Lambda} \mathbf{X}^\top \mathbf{X} \boldsymbol{\Lambda}), \text{ where } \boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_p] \\
\epsilon_i &\sim \mathsf{N}(0, \psi^{-1}) \\
i &= 1, \ldots, n
\end{aligned}
\tag{2}
$$

  is an equivalent representation of the w I-prior model under the Canonical kernel.

The beta I-prior (linear) model cont.

- The standard multiple regression model with an I-prior on $\boldsymbol{\beta}$

$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i$$
$$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, \psi \boldsymbol{\Lambda} \mathbf{X}^\top \mathbf{X} \boldsymbol{\Lambda}), \text{ where } \boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_p]$$
$$\epsilon_i \sim \mathsf{N}(0, \psi^{-1})$$
$$i = 1, \ldots, n$$

(2)

  is an equivalent representation of the w I-prior model under the Canonical kernel.

- Estimate this model via ML methods as before, or fully Bayes, as we will see soon.

Shrinkage properties of I-priors

- Comparison to ridge regression and Lasso

$$Ridge : \hat{\boldsymbol{\beta}}^R = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

$$Lasso : \hat{\boldsymbol{\beta}}^L = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Shrinkage properties of I-priors

- Comparison to ridge regression and Lasso

$$Ridge : \beta_1, \ldots, \beta_p \sim \mathsf{N}(0, 1/\lambda)$$
$$Lasso : \beta_1, \ldots, \beta_p \sim \mathsf{Laplace}(0, 1/\lambda)$$

Shrinkage properties of I-priors

- Comparison to ridge regression and Lasso

$$Ridge : \beta_1, \ldots, \beta_p \sim N(0, 1/\lambda)$$
$$Lasso : \beta_1, \ldots, \beta_p \sim \text{Laplace}(0, 1/\lambda)$$

These only (typically) use one common shrinkage parameter λ.

Shrinkage properties of I-priors

- Comparison to ridge regression and Lasso

$$Ridge : \beta_1, \ldots, \beta_p \sim \mathsf{N}(0, 1/\lambda)$$
$$Lasso : \beta_1, \ldots, \beta_p \sim \mathsf{Laplace}(0, 1/\lambda)$$

These only (typically) use one common shrinkage parameter λ.

- Other Bayesian Variable Selection priors
  - ▸ After standardising data $\mathbf{X}$, use weakly informative priors

$$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, 10\mathbf{I}_p).$$

Shrinkage properties of I-priors

- Comparison to ridge regression and Lasso

$$Ridge : \beta_1, \ldots, \beta_p \sim \mathsf{N}(0, 1/\lambda)$$
$$Lasso : \beta_1, \ldots, \beta_p \sim \mathsf{Laplace}(0, 1/\lambda)$$

  These only (typically) use one common shrinkage parameter λ.

- Other Bayesian Variable Selection priors
  - ▶ After standardising data **X**, use weakly informative priors

    $$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, 10\mathbf{I}_p).$$

  - ▶ The objective g-prior is also popular

    $$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, g(\mathbf{X}^\top \mathbf{X})^{-1}).$$

Shrinkage properties of I-priors

- Comparison to ridge regression and Lasso

$$Ridge : \beta_1, \ldots, \beta_p \sim \mathsf{N}(0, 1/\lambda)$$
$$Lasso : \beta_1, \ldots, \beta_p \sim \mathsf{Laplace}(0, 1/\lambda)$$

  These only (typically) use one common shrinkage parameter λ.

- Other Bayesian Variable Selection priors
  - After standardising data $\mathbf{X}$, use weakly informative priors

  $$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, 10\mathbf{I}_p).$$

  - The objective g-prior is also popular

  $$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, g(\mathbf{X}^\top \mathbf{X})^{-1}).$$

- I-priors have individual shrinkage coefficients on the $\boldsymbol{\beta}$, and also makes them correlated a priori.

Shrinkage properties of I-priors cont.

Demo

https://haziqjamil.shinyapps.io/iprior/

Full Bayes estimation

- The fully Bayes beta I-prior model is the following hierarchical model

$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i$$
$$\epsilon_i \sim \mathsf{N}(0, \psi^{-1})$$
$$i = 1, \ldots, n$$

(3)

$$\underline{\text{Priors}}$$
$$\alpha \sim \mathsf{N}(0, a^2)$$
$$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, \psi \boldsymbol{\Lambda} \mathbf{X}^\top \mathbf{X} \boldsymbol{\Lambda}), \text{ where } \boldsymbol{\Lambda} = \text{diag}[\lambda_1, \ldots, \lambda_p]$$
$$\psi, \lambda_1^{-2}, \ldots, \lambda_p^{-2} \sim \Gamma(c, d)$$

Full Bayes estimation

- The fully Bayes beta I-prior model is the following hierarchical model

$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i$$
$$\epsilon_i \sim \mathsf{N}(0, \psi^{-1})$$
$$i = 1, \ldots, n$$

$$(3)$$

Priors
$$\alpha \sim \mathsf{N}(0, a^2)$$
$$\boldsymbol{\beta} \sim \mathsf{N}(\mathbf{0}, \psi \boldsymbol{\Lambda} \mathbf{X}^\top \mathbf{X} \boldsymbol{\Lambda}), \text{ where } \boldsymbol{\Lambda} = \operatorname{diag}[\lambda_1, \ldots, \lambda_p]$$
$$\psi, \lambda_1^{-2}, \ldots, \lambda_p^{-2} \sim \Gamma(c, d)$$

- The posterior distribution is

$$f(\alpha, \boldsymbol{\beta}, \psi, \boldsymbol{\lambda} | \mathbf{y}) \propto f(\mathbf{y} | \alpha, \boldsymbol{\beta}, \psi, \boldsymbol{\lambda}) f(\alpha, \boldsymbol{\beta}, \psi, \boldsymbol{\lambda})$$
$$\propto f(\mathbf{y} | \alpha, \boldsymbol{\beta}, \psi) f(\alpha) f(\boldsymbol{\beta} | \psi, \boldsymbol{\lambda}) f(\psi) f(\lambda_1) \cdots f(\lambda_p)$$

Estimation using JAGS

- Fit Bayesian models using JAGS (or WinBUGS or OpenBUGS).
- In R, many packages to run JAGS models: `rjags`, `R2Jags`, `runjags`.
- We will use `runjags` as it allows easy parallelisation of chains.
- Simulate a dataset:

```
n <- 100
p <- 2
beta.true <- matrix(c(10, 0), ncol = 1)
X <- matrix(rnorm(n * p, ncol = p)
Y <- X %*% beta.true + rnorm(n, mean = 0, sd = 2)
```

Estimation using JAGS

- Fit Bayesian models using JAGS (or WinBUGS or OpenBUGS).
- In R, many packages to run JAGS models: `rjags`, `R2Jags`, `runjags`.
- We will use `runjags` as it allows easy parallelisation of chains.
- Simulate a dataset:

```
n <- 100
p <- 2
beta.true <- matrix(c(10, 0), ncol = 1)
X <- matrix(rnorm(n * p, ncol = p)
Y <- X %*% beta.true + rnorm(n, mean = 0, sd = 2)
```

Estimation using JAGS

```
mod <- "
  model {
    for (i in 1:n) {
      Y[i] ~ dnorm(mu[i], psi)
      mu[i] <- alpha + inprod(X[i,1:p], beta[1:p])
    }

    alpha ~ dnorm(0, 0.0001)
    psi ~ dgamma(0.1, 0.0001)
    for (j in 1:p) {
      lambdasq[j] ~ dgamma(0.0001, 0.0001)
      for (k in 1:p) { LambdaInv[j, k] <- equals(j,k) * pow(lambdasq[k], -0.5) }
    }
    BetaPrec <- LambdaInv[1:p, 1:p] %*% XTX.inv %*% LambdaInv[1:p, 1:p] / psi
    beta[1:p] ~ dmnorm(rep(0, p), BetaPrec)

    sigma <- pow(psi, -0.5)
    lambda[1:p] <- pow(lambdasq[1:p], 0.5)
  }
  #data# Y, X, XTX.inv, n, p
  #inits# alpha, beta, psi, lambdasq
  #monitor# alpha, beta, sigma, lambda
"
```

Estimation using JAGS

```
(mod.fit <- run.jags(mod, n.chains = 4, sample = 2500, thin = 10,
                     method = "parallel", n.sims = 4))


##
## JAGS model summary statistics from 10000 samples (thin = 10; chains = 4; adapt+b
##
##              Lower95     Median    Upper95       Mean        SD        Mode
## alpha       -0.39905   -0.053978   0.36319  -0.052545   0.19233   -0.039251
## beta[1]       9.3746     9.7564     10.134     9.7574   0.19342      9.7704
## beta[2]     -0.010512  2.8816e-30 0.020854  0.0033068  0.030077  5.9629e-23
## sigma         1.6469     1.8978     2.1805     1.9044   0.13755      1.8887
## lambda[1]    0.55642     2.7259     16.499     4.6509    5.0124      1.7294
## lambda[2] 1.0291e-77  1.4207e-21 0.0052381 0.0018802  0.012068  3.0584e-23
##
##               MCerr  MC%ofSD  SSeff     AC.100     psrf
## alpha       0.0019229       1  10003   0.014252   1.0008
## beta[1]     0.0019342       1  10000   0.020871   0.9999
## beta[2]     0.0008697     2.9   1196    0.10176   1.0591
## sigma       0.0013755       1  10000  -0.0037105  1.0004
## lambda[1]    0.26974     5.4    345    0.53237   1.0284
## lambda[2] 0.00067477     5.6    320    0.39277   1.2695
##
## Total time taken: 10.2 seconds
```
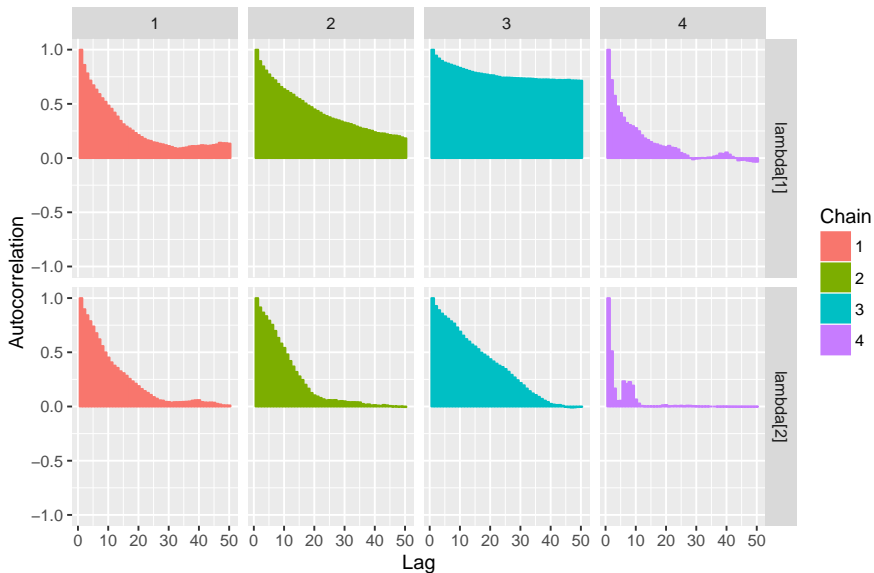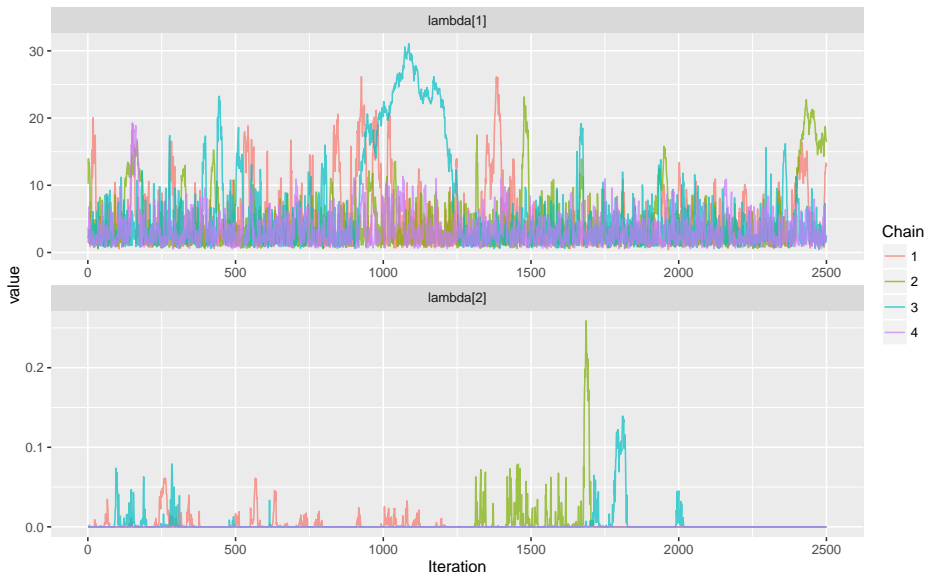
Estimation using JAGS

Estimation using JAGS

Problems

- Trace plot of λ_k can be very erratic, and samples found to be severely autocorrelated.

- Since the scale parameters are very important for Bayesian Variable Selection, it is imperative that these are estimated correctly.

- Suggestions:
    - Improve samples - Hamiltonian Monte Carlo?
    - Treat $\boldsymbol{\lambda}$ as fixed, replacing them with estimates obtained using ML methods.

1 Bayesian Variable Selection

2 I-priors

3 Bayesian I-prior linear models

4 Hamiltonian Monte Carlo

5 Summary

Introduction

- Introduced as Hybrid Monte Carlo [Duane et al., 1987] for use in lattice models of quantum theory. Statistical applications started appearing sparsely in the 1990s.

- Development of HMC software (Stan) began in 2011, motivated by the difficulties faced when doing full Bayesian inference on multilevel generalised linear models.

- The basic idea behind HMC is to use Hamiltonian dynamics to propose new states, instead of "random walks".

$$\mathbf{x} \xrightarrow[\substack{\text{Markov} \\ \text{Transition}}]{} \mathbf{x}^*$$

- High probability of acceptance
- Distant move

Hamiltonian dynamics

- A reformulation of classical mechanics which describes motion through Hamilton's equations:

$$\frac{d\mathbf{x}}{dt} = \frac{\partial H}{\partial \mathbf{p}} \ \text{ and } \ \frac{d\mathbf{p}}{dt} = -\frac{\partial H}{\partial \mathbf{x}},$$

where $H = H(\mathbf{x}, \mathbf{p})$ is the Hamiltonian of the system (total energy), and $(\mathbf{x}, \mathbf{p})$ are the position and momentum coordinates of the body in motion.

Hamiltonian dynamics

- A reformulation of classical mechanics which describes motion through Hamilton's equations:

$$\frac{d\mathbf{x}}{dt} = \frac{\partial H}{\partial \mathbf{p}} \ \text{ and } \ \frac{d\mathbf{p}}{dt} = -\frac{\partial H}{\partial \mathbf{x}},$$

where $H = H(\mathbf{x}, \mathbf{p})$ is the Hamiltonian of the system (total energy), and $(\mathbf{x}, \mathbf{p})$ are the position and momentum coordinates of the body in motion.

- In a closed system,

$$H(\mathbf{x}, \mathbf{p}) = \quad K(\mathbf{p}) \quad + \quad U(\mathbf{x})$$

Kinetic energy   Potential energy

Hamiltonian dynamics

- A reformulation of classical mechanics which describes motion through Hamilton's equations:

$$\frac{d\mathbf{x}}{dt} = \frac{\partial H}{\partial \mathbf{p}} = \frac{\partial}{\partial \mathbf{p}} K(\mathbf{p}) \ \text{ and } \ \frac{d\mathbf{p}}{dt} = -\frac{\partial H}{\partial \mathbf{x}} = -\frac{\partial}{\partial \mathbf{x}} U(\mathbf{x}),$$

  where $H = H(\mathbf{x}, \mathbf{p})$ is the Hamiltonian of the system (total energy), and $(\mathbf{x}, \mathbf{p})$ are the position and momentum coordinates of the body in motion.
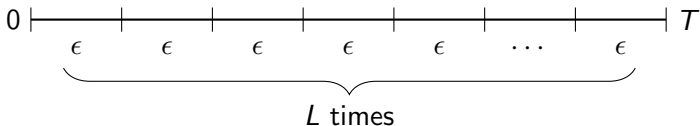
- In a closed system,

$$H(\mathbf{x}, \mathbf{p}) = \quad K(\mathbf{p}) \quad + \quad U(\mathbf{x})$$

  Kinetic energy   Potential energy

Hamiltonian dynamics cont.

- To describe the evolution of $\big(\mathbf{x}(t), \mathbf{p}(t)\big)$ from time t to $t + T$, it is necessary to discretise time and split $T = L \cdot \epsilon$.
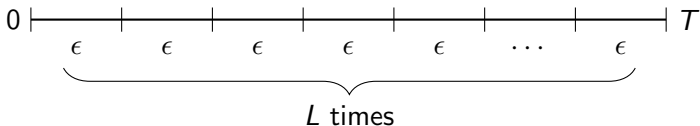
Hamiltonian dynamics cont.

- To describe the evolution of $\big(\mathbf{x}(t), \mathbf{p}(t)\big)$ from time t to $t + T$, it is necessary to discretise time and split $T = L \cdot \epsilon$.

- Solve the system of differential equations using Euler's method, or the more commonly used leapfrog integration:

$$\text{Step 1: } \mathbf{p}(t + \epsilon/2) = \mathbf{p}(t) - \frac{\epsilon}{2} \cdot \frac{\partial}{\partial \mathbf{x}} U\big(\mathbf{x}(t)\big)$$

Hamiltonian dynamics cont.

- To describe the evolution of $\big(\mathbf{x}(t), \mathbf{p}(t)\big)$ from time t to $t + T$, it is necessary to discretise time and split $T = L \cdot \epsilon$.
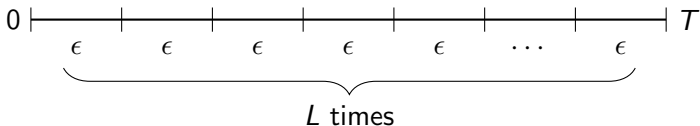

- Solve the system of differential equations using Euler's method, or the more commonly used leapfrog integration:

$$\text{Step 1: } \mathbf{p}(t + \epsilon/2) = \mathbf{p}(t) - \frac{\epsilon}{2} \cdot \frac{\partial}{\partial \mathbf{x}} U\big(\mathbf{x}(t)\big)$$

$$\text{Step 2: } \quad \mathbf{x}(t + \epsilon) = \mathbf{x}(t) + \epsilon \cdot \frac{\partial}{\partial \mathbf{p}} K\big(\mathbf{p}(t + \epsilon/2)\big)$$

Hamiltonian dynamics cont.

- To describe the evolution of $\big(\mathbf{x}(t), \mathbf{p}(t)\big)$ from time t to $t + T$, it is necessary to discretise time and split $T = L \cdot \epsilon$.

$$0 \vdash\!\!\!\underbrace{\!\!\!\underset{\epsilon}{\rule{0pt}{0pt}}\quad\underset{\epsilon}{\rule{0pt}{0pt}}\quad\underset{\epsilon}{\rule{0pt}{0pt}}\quad\underset{\epsilon}{\rule{0pt}{0pt}}\quad\underset{\epsilon}{\rule{0pt}{0pt}}\quad\underset{\cdots}{\rule{0pt}{0pt}}\quad\underset{\epsilon}{\rule{0pt}{0pt}}\!\!\!}_{L \text{ times}}\!\!\!\dashv T$$
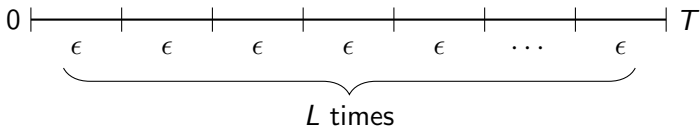
- Solve the system of differential equations using Euler's method, or the more commonly used leapfrog integration:

$$\text{Step 1: } \mathbf{p}(t + \epsilon/2) = \mathbf{p}(t) - \frac{\epsilon}{2} \cdot \frac{\partial}{\partial \mathbf{x}} U\big(\mathbf{x}(t)\big)$$

$$\text{Step 2: } \quad \mathbf{x}(t + \epsilon) = \mathbf{x}(t) + \epsilon \cdot \frac{\partial}{\partial \mathbf{p}} K\big(\mathbf{p}(t + \epsilon/2)\big)$$

$$\text{Step 3: } \quad \mathbf{p}(t + \epsilon) = \mathbf{p}(t + \epsilon/2) - \frac{\epsilon}{2} \cdot \frac{\partial}{\partial \mathbf{x}} U\big(\mathbf{x}(t + \epsilon)\big)$$

Hamiltonian dynamics cont.

- To describe the evolution of $\big(\mathbf{x}(t), \mathbf{p}(t)\big)$ from time t to $t + T$, it is necessary to discretise time and split $T = L \cdot \epsilon$.

- Solve the system of differential equations using Euler's method, or the more commonly used leapfrog integration:
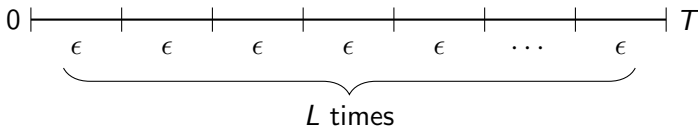
$$\text{Step 1: } \mathbf{p}(t + \epsilon/2) = \mathbf{p}(t) - \frac{\epsilon}{2} \cdot \frac{\partial}{\partial \mathbf{x}} U\big(\mathbf{x}(t)\big)$$

$$\text{Step 2: } \quad \mathbf{x}(t + \epsilon) = \mathbf{x}(t) + \epsilon \cdot \frac{\partial}{\partial \mathbf{p}} K\big(\mathbf{p}(t + \epsilon/2)\big)$$

$$\text{Step 3: } \quad \mathbf{p}(t + \epsilon) = \mathbf{p}(t + \epsilon/2) - \frac{\epsilon}{2} \cdot \frac{\partial}{\partial \mathbf{x}} U\big(\mathbf{x}(t + \epsilon)\big)$$

Steps 1-3 are repeated L times.

Hamiltonian dynamics cont.

Demo

https://haziqjamil.shinyapps.io/hmc1/

Probability and the Hamiltonian

- Given some energy function $E(\boldsymbol{\theta})$ over states $\boldsymbol{\theta}$, the *canonical distribution* of the states $\boldsymbol{\theta}$ is given by the pdf

$$f(\boldsymbol{\theta}) = \frac{1}{Z} \exp\left[-\frac{E(\boldsymbol{\theta})}{kT} \right].$$

where k is Boltzmann's constant, T is the absolute temperature of the system, and Z is a normalising constant.

Probability and the Hamiltonian

- Given some energy function $E(\boldsymbol{\theta})$ over states $\boldsymbol{\theta}$, the *canonical distribution* of the states $\boldsymbol{\theta}$ is given by the pdf

$$f(\boldsymbol{\theta}) = \frac{1}{Z} \exp\left[-\frac{E(\boldsymbol{\theta})}{kT}\right].$$

  where k is Boltzmann's constant, T is the absolute temperature of the system, and Z is a normalising constant.

- The Hamiltonian $H(\mathbf{x}, \mathbf{p}) = K(\mathbf{p}) + U(\mathbf{x})$ is one such energy function over states $(\mathbf{x}, \mathbf{p})$.

Probability and the Hamiltonian

- Given some energy function $E(\theta)$ over states θ, the *canonical distribution* of the states θ is given by the pdf

$$f(\mathbf{x}, \mathbf{p}) = \frac{1}{Z} \exp\left[-\frac{H(\mathbf{x}, \mathbf{p})}{kT} \right].$$

where k is Boltzmann's constant, T is the absolute temperature of the system, and Z is a normalising constant.

- The Hamiltonian $H(\mathbf{x}, \mathbf{p}) = K(\mathbf{p}) + U(\mathbf{x})$ is one such energy function over states $(\mathbf{x}, \mathbf{p})$.

Probability and the Hamiltonian

- Given some energy function $E(\boldsymbol{\theta})$ over states $\boldsymbol{\theta}$, the *canonical distribution* of the states $\boldsymbol{\theta}$ is given by the pdf

$$f(\mathbf{x}, \mathbf{p}) = \frac{1}{Z} \exp \left[-\frac{H(\mathbf{x}, \mathbf{p})}{kT} \right].$$

where k is Boltzmann's constant, T is the absolute temperature of the system, and Z is a normalising constant.

- The Hamiltonian $H(\mathbf{x}, \mathbf{p}) = K(\mathbf{p}) + U(\mathbf{x})$ is one such energy function over states $(\mathbf{x}, \mathbf{p})$.

- Notice that the distribution for $\mathbf{x}$ and $\mathbf{p}$ are independent:

$$f(\mathbf{x}, \mathbf{p}) \propto \exp \left[-\frac{K(\mathbf{p})}{kT} \right] \exp \left[-\frac{U(\mathbf{x})}{kT} \right] = f(\mathbf{x}) f(\mathbf{p}).$$

Probability and the Hamiltonian

- Given some energy function $E(\theta)$ over states θ, the *canonical distribution* of the states θ is given by the pdf

$$f(\mathbf{x}, \mathbf{p}) = \frac{1}{Z} \exp\left[-\frac{H(\mathbf{x}, \mathbf{p})}{kT}\right].$$

where k is Boltzmann's constant, T is the absolute temperature of the system, and Z is a normalising constant.

- The Hamiltonian $H(\mathbf{x}, \mathbf{p}) = K(\mathbf{p}) + U(\mathbf{x})$ is one such energy function over states $(\mathbf{x}, \mathbf{p})$.

- Notice that the distribution for $\mathbf{x}$ and $\mathbf{p}$ are independent:

$$f(\mathbf{x}, \mathbf{p}) \propto \exp\left[-\frac{K(\mathbf{p})}{kT}\right] \exp\left[-\frac{U(\mathbf{x})}{kT}\right] = f(\mathbf{x})f(\mathbf{p}).$$

- Typically, choose T such that $kT = 1$.

Choosing the energy functions

- Using a *quadratic kinetic energy function* $K(\mathbf{p}) = \mathbf{p}^\top \mathbf{M}^{-1} \mathbf{p}/2$ yields the probability density function

$$f(\mathbf{p}) \propto \exp\left[-\frac{1}{2} \mathbf{p}^\top \mathbf{M}^{-1} \mathbf{p} \right]$$

Choosing the energy functions

- Using a *quadratic kinetic energy function* $K(\mathbf{p}) = \mathbf{p}^\top \mathbf{M}^{-1} \mathbf{p}/2$ yields the probability density function

$$f(\mathbf{p}) \propto \exp\left[-\frac{1}{2}\mathbf{p}^\top \mathbf{M}^{-1}\mathbf{p}\right],$$

implying $\mathbf{p} \sim \mathrm{N}_d(\mathbf{0}, \mathbf{M})$, where $\mathbf{M} = \mathrm{diag}[m_1, \ldots, m_d]$ is the mass matrix.

Choosing the energy functions

- Using a *quadratic kinetic energy function* $K(\mathbf{p}) = \mathbf{p}^\top \mathbf{M}^{-1} \mathbf{p}/2$ yields the probability density function

$$f(\mathbf{p}) \propto \exp\left[-\frac{1}{2}\mathbf{p}^\top \mathbf{M}^{-1}\mathbf{p}\right],$$

implying $\mathbf{p} \sim \mathrm{N}_d(\mathbf{0}, \mathbf{M})$, where $\mathbf{M} = \mathrm{diag}[m_1, \ldots, m_d]$ is the mass matrix.

- As for the potential energy, choose a function such that

$$U(\mathbf{x}) = -\log f(\mathbf{x})$$

Choosing the energy functions

- Using a *quadratic kinetic energy function* $K(\mathbf{p}) = \mathbf{p}^\top \mathbf{M}^{-1} \mathbf{p}/2$ yields the probability density function

$$f(\mathbf{p}) \propto \exp\left[-\frac{1}{2}\mathbf{p}^\top \mathbf{M}^{-1} \mathbf{p}\right],$$

implying $\mathbf{p} \sim N_d(\mathbf{0}, \mathbf{M})$, where $\mathbf{M} = \text{diag}[m_1, \ldots, m_d]$ is the mass matrix.

- As for the potential energy, choose a function such that

$$U(\mathbf{x}) = -\log f(\mathbf{x}),$$

since $f(\mathbf{x}) \propto \exp[-U(\mathbf{x})]$, where $f(\mathbf{x})$ is the target density from which we wish to sample.

Hamiltonian Monte Carlo

- To sample variables $\mathbf{x}$, introduce momentum variables $\mathbf{p}$ and sample jointly from $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x})f(\mathbf{p})$.

Hamiltonian Monte Carlo

- To sample variables $\mathbf{x}$, introduce momentum variables $\mathbf{p}$ and sample jointly from $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x})f(\mathbf{p})$.
- The Hamiltonian Monte Carlo (HMC) algorithm
  Step 1 *Perturb momentum*. Draw $\mathbf{p}$ from $N_d(\mathbf{0}, \mathbf{M})$.

Hamiltonian Monte Carlo

- To sample variables $\mathbf{x}$, introduce momentum variables $\mathbf{p}$ and sample jointly from $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x})f(\mathbf{p})$.
- The Hamiltonian Monte Carlo (HMC) algorithm
  - Step 1 *Perturb momentum.* Draw $\mathbf{p}$ from $\mathrm{N}_d(\mathbf{0}, \mathbf{M})$.
  - Step 2 *Metropolis update.* Simulate Hamiltonian dynamics using L leapfrogs of step-size ϵ and obtain a new state $(\mathbf{x}^*, \mathbf{p}^*)$. Accept the proposal state with probability $\min(1, A)$, where

$$A = \frac{f(\mathbf{x}^*, \mathbf{p}^*)}{f(\mathbf{x}, \mathbf{p})} = \exp\left[H(\mathbf{x}, \mathbf{p}) - H(\mathbf{x}^*, \mathbf{p}^*)\right].$$
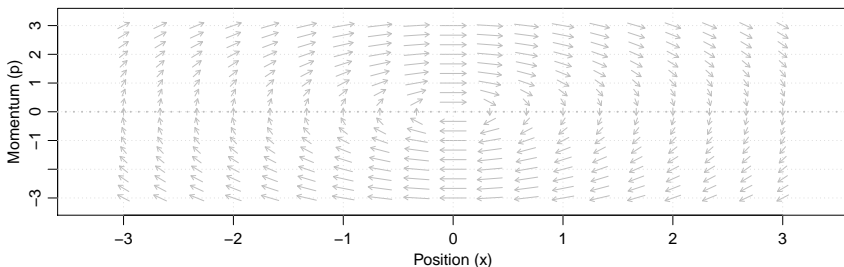
Hamiltonian Monte Carlo

- To sample variables $\mathbf{x}$, introduce momentum variables $\mathbf{p}$ and sample jointly from $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x})f(\mathbf{p})$.
- The Hamiltonian Monte Carlo (HMC) algorithm
  - Step 1 *Perturb momentum*. Draw $\mathbf{p}$ from $N_d(\mathbf{0}, \mathbf{M})$.
  - Step 2 *Metropolis update*. Simulate Hamiltonian dynamics using L leapfrogs of step-size ϵ and obtain a new state $(\mathbf{x}^*, \mathbf{p}^*)$. Accept the proposal state with probability $\min(1, A)$, where

$$A = \frac{f(\mathbf{x}^*, \mathbf{p}^*)}{f(\mathbf{x}, \mathbf{p})} = \exp\left[H(\mathbf{x}, \mathbf{p}) - H(\mathbf{x}^*, \mathbf{p}^*)\right].$$
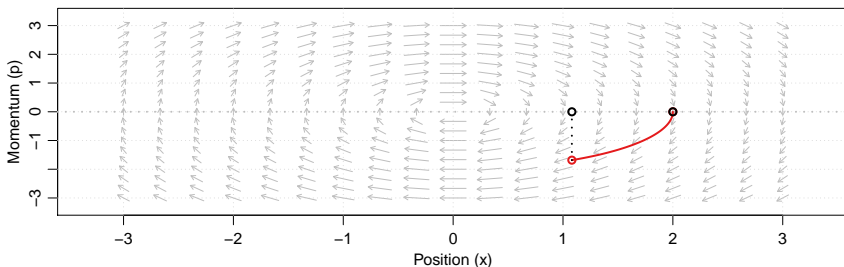
Hamiltonian Monte Carlo

- To sample variables $\mathbf{x}$, introduce momentum variables $\mathbf{p}$ and sample jointly from $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x})f(\mathbf{p})$.
- The Hamiltonian Monte Carlo (HMC) algorithm
  - Step 1 *Perturb momentum.* Draw $\mathbf{p}$ from $N_d(\mathbf{0}, \mathbf{M})$.
  - Step 2 *Metropolis update.* Simulate Hamiltonian dynamics using L leapfrogs of step-size ϵ and obtain a new state $(\mathbf{x}^*, \mathbf{p}^*)$. Accept the proposal state with probability $\min(1, A)$, where

$$A = \frac{f(\mathbf{x}^*, \mathbf{p}^*)}{f(\mathbf{x}, \mathbf{p})} = \exp\left[H(\mathbf{x}, \mathbf{p}) - H(\mathbf{x}^*, \mathbf{p}^*)\right].$$
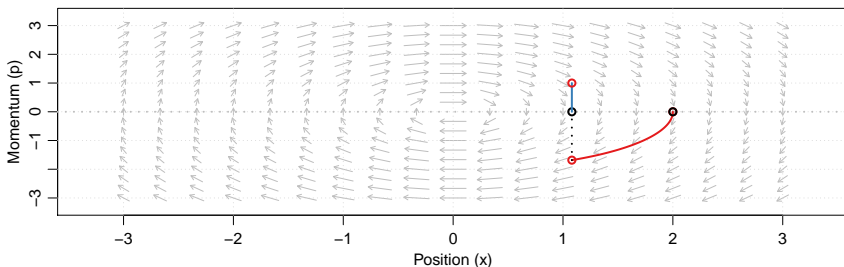
Hamiltonian Monte Carlo

- To sample variables $\mathbf{x}$, introduce momentum variables $\mathbf{p}$ and sample jointly from $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x})f(\mathbf{p})$.
- The Hamiltonian Monte Carlo (HMC) algorithm
    - Step 1 *Perturb momentum.* Draw $\mathbf{p}$ from $N_d(\mathbf{0}, \mathbf{M})$.
    - Step 2 *Metropolis update.* Simulate Hamiltonian dynamics using L leapfrogs of step-size ϵ and obtain a new state $(\mathbf{x}^*, \mathbf{p}^*)$. Accept the proposal state with probability $\min(1, A)$, where

$$A = \frac{f(\mathbf{x}^*, \mathbf{p}^*)}{f(\mathbf{x}, \mathbf{p})} = \exp\left[H(\mathbf{x}, \mathbf{p}) - H(\mathbf{x}^*, \mathbf{p}^*)\right].$$

Hamiltonian Monte Carlo

- To sample variables $\mathbf{x}$, introduce momentum variables $\mathbf{p}$ and sample jointly from $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x})f(\mathbf{p})$.
- The Hamiltonian Monte Carlo (HMC) algorithm

  Step 1 *Perturb momentum.* Draw $\mathbf{p}$ from $N_d(\mathbf{0}, \mathbf{M})$.

  Step 2 *Metropolis update.* Simulate Hamiltonian dynamics using L leapfrogs of step-size ϵ and obtain a new state $(\mathbf{x}^*, \mathbf{p}^*)$. Accept the proposal state with probability $\min(1, A)$, where

$$A = \frac{f(\mathbf{x}^*, \mathbf{p}^*)}{f(\mathbf{x}, \mathbf{p})} = \exp\left[H(\mathbf{x}, \mathbf{p}) - H(\mathbf{x}^*, \mathbf{p}^*)\right].$$
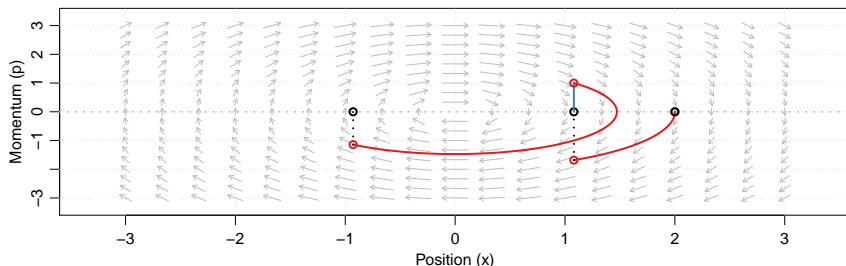
Hamiltonian Monte Carlo

- To sample variables $\mathbf{x}$, introduce momentum variables $\mathbf{p}$ and sample jointly from $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x})f(\mathbf{p})$.
- The Hamiltonian Monte Carlo (HMC) algorithm
  - Step 1 *Perturb momentum.* Draw $\mathbf{p}$ from $N_d(\mathbf{0}, \mathbf{M})$.
  - Step 2 *Metropolis update.* Simulate Hamiltonian dynamics using L leapfrogs of step-size ϵ and obtain a new state $(\mathbf{x}^*, \mathbf{p}^*)$. Accept the proposal state with probability $\min(1, A)$, where

$$A = \frac{f(\mathbf{x}^*, \mathbf{p}^*)}{f(\mathbf{x}, \mathbf{p})} = \exp\left[H(\mathbf{x}, \mathbf{p}) - H(\mathbf{x}^*, \mathbf{p}^*)\right].$$
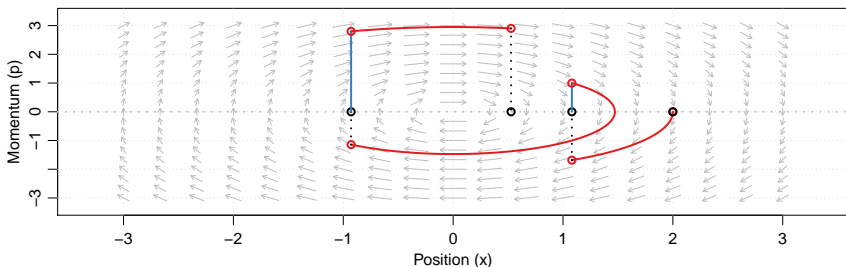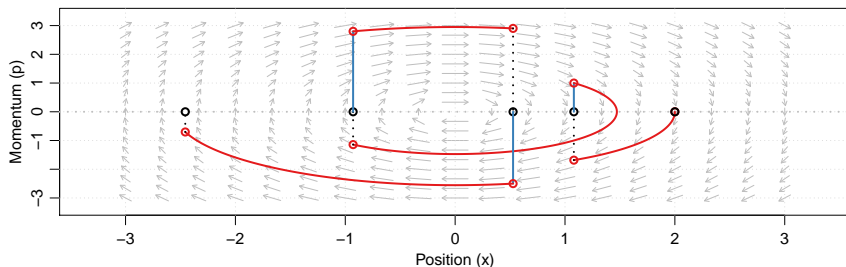
Hamiltonian Monte Carlo

- To sample variables $\mathbf{x}$, introduce momentum variables $\mathbf{p}$ and sample jointly from $f(\mathbf{x}, \mathbf{p}) = f(\mathbf{x})f(\mathbf{p})$.
- The Hamiltonian Monte Carlo (HMC) algorithm
  Step 1 *Perturb momentum.* Draw $\mathbf{p}$ from $N_d(\mathbf{0}, \mathbf{M})$.
  Step 2 *Metropolis update.* Simulate Hamiltonian dynamics using L leapfrogs of step-size ϵ and obtain a new state $(\mathbf{x}^*, \mathbf{p}^*)$. Accept the proposal state with probability $\min(1, A)$, where

$$A = \frac{f(\mathbf{x}^*, \mathbf{p}^*)}{f(\mathbf{x}, \mathbf{p})} = \exp\left[H(\mathbf{x}, \mathbf{p}) - H(\mathbf{x}^*, \mathbf{p}^*)\right].$$

Hamiltonian Monte Carlo cont.

Demo

https://haziqjamil.shinyapps.io/hmc2/

Stan

http://mc-stan.org

- Stan interfaces: R, Python, shell, MATLAB, Julia, Stata, and Mathematica. Runs on Linux, Mac and Windows.
- R package `rstan` uses Stan modelling language. For expression-based Bayesian regression modelling, package `rstanarm` is available.
- Nice things about Stan
  - ▶ Tuning is done automatically.
  - ▶ Vast library of differentiable probability functions, or code your own.
  - ▶ Conjugacy has no computational advantage.
  - ▶ Optimising for efficiency possible, e.g. vectorisation.

Stan example

```
stan.iprior.mod <- "
  function {
    ...
  }
  data {
    int n; // number of data
    int p; // number of parameters
    vector[n] Y; // responses
    matrix[n, p] X; // (centred) data
  }
  transformed data {
    matrix[p, p] XTX;
    XTX = X' * X;
  }
  parameters {
    real alpha;  // intercept
    real<lower=0> sigma;  // s.d. of errors
    vector[p] beta;  // regression coefficients
    vector<lower=0>[p] lambda;  // I-prior scale parameters
  }
```

Stan example

```
transformed parameters {
  vector[p] lambdasq;
  cov_matrix[p] Sigma;
  vector[n] mu;
  lambdasq = lambda .* lambda;
  Sigma = diag_matrix(lambda) * XTX * diag_matrix(lambda) ./ (sigma ^ 2);
  mu = alpha + X * beta;
}
model {
  target += inv_gamma_lpdf(lambdasq | 0.0001, 0.0001);
  target += multi_normal_lpdf(beta | rep_vector(0, p), Sigma);
  target += normal_lpdf(Y | mu, sigma);
}
generated quantities {
  ...
}
"
```

Stan example

- Compile the Stan model.

```
m <- stan_model(model_code = stan.mod)
m@model_name <- "iprior"
```

Stan example

- Compile the Stan model.

```
m <- stan_model(model_code = stan.mod)
m@model_name <- "iprior"
```

- Set the data for Stan to use.

```
stan.dat <- list(Y = as.vector(Y), X = Xs, n = n, p = p)
```

Stan example

- Compile the Stan model.

```
m <- stan_model(model_code = stan.mod)
m@model_name <- "iprior"
```

- Set the data for Stan to use.

```
stan.dat <- list(Y = as.vector(Y), X = Xs, n = n, p = p)
```

- Begin sampling

```
fit.stan <- stan(model_code = stan.mod, data = stan.dat,
                 pars = c("alpha", "beta", "lambda", "sigma"),
                 iter = 50000, chains = 4, thin = 10)
```

Stan example

```
print(fit.stan)

## Inference for Stan model: iprior.
## 4 chains, each with iter=50000; warmup=25000; thin=10;
## post-warmup draws per chain=2500, total post-warmup draws=10000.
##
##             mean se_mean   sd    2.5%     25%     50%     75%    97.5%
## alpha      -1.04    0.00 0.19   -1.40   -1.16   -1.04   -0.92   -0.68
## beta[1]     9.71    0.00 0.20    9.32    9.58    9.71    9.84   10.10
## beta[2]     0.07    0.00 0.10   -0.10    0.01    0.06    0.12    0.31
## lambda[1]   2.26    0.03 2.55    0.70    1.14    1.61    2.47    7.78
## lambda[2]   0.03    0.00 0.06    0.01    0.01    0.02    0.03    0.09
## sigma       1.91    0.00 0.13    1.68    1.82    1.91    2.00    2.20
## lp__     -224.20    0.02 1.85 -228.60 -225.24 -223.88 -222.82 -221.62
##           n_eff Rhat
## alpha     10000    1
## beta[1]   10000    1
## beta[2]    9410    1
## lambda[1] 10000    1
## lambda[2]  9972    1
```

Stan example
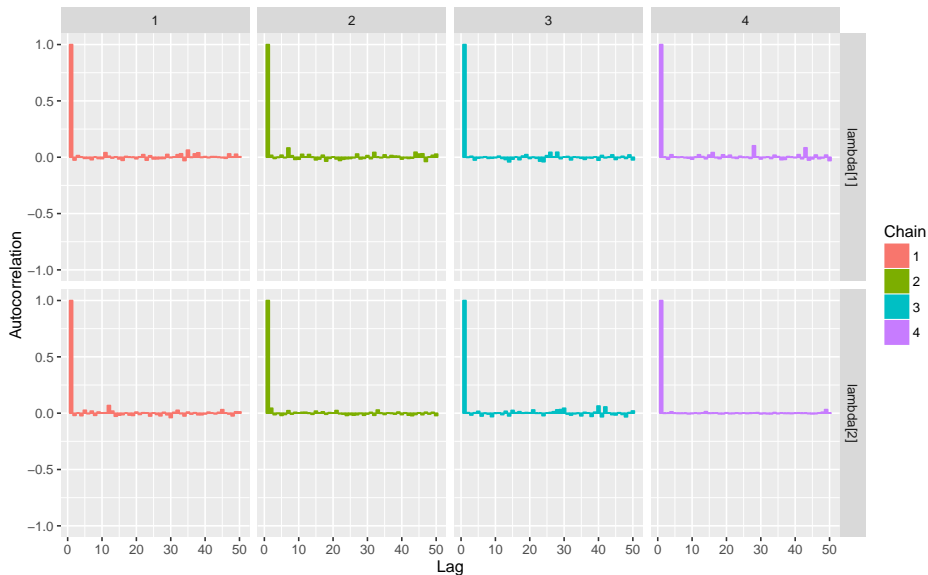
Stan example
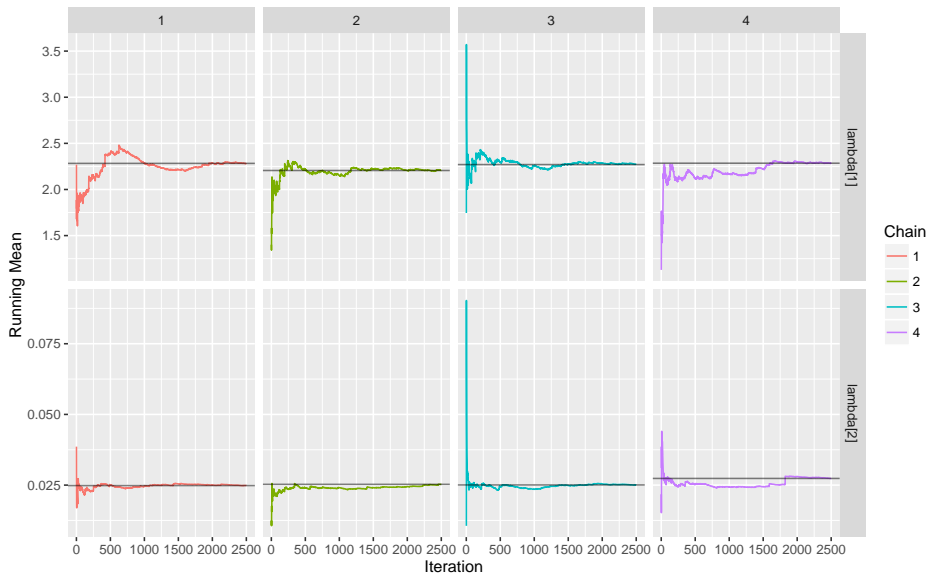
HMC unable to sample from discrete distributions

- HMC requires that the domain of $f(\mathbf{x})$ is continuous and $\partial \log f(\mathbf{x})/\partial \mathbf{x}$ is inexpensive to compute.
- This is a problem for our Bayesian Variable Selection model because we need posterior samples of $\gamma \in \{0, 1\}^p$.
- Three ideas:
  - ▶ Marginalise the discrete variables.
  - ▶ Use an underlying latent continuous variable.
  - ▶ Augment with Gibbs sampling.

Approach 1: Marginalise

- Let $\boldsymbol{\theta}$ be some continuous parameters and $\boldsymbol{\gamma}$ be some discrete parameters in the model with data $\mathbf{y}$.

- Since unable to sample from $f(\boldsymbol{\gamma}|\mathbf{y})$, integrate out $\boldsymbol{\gamma}$ from the model, and just sample from the posterior of $\boldsymbol{\theta}$

$$f(\boldsymbol{\theta}|\mathbf{y}) = \sum_{\boldsymbol{\gamma}} f(\boldsymbol{\theta}, \boldsymbol{\gamma}|\mathbf{y}) = f(\boldsymbol{\theta}) \sum_{\boldsymbol{\gamma}} f(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\gamma}) f(\boldsymbol{\gamma})$$

- The unnormalised posterior probability mass function for $\boldsymbol{\gamma}$ is

$$q(\boldsymbol{\gamma}) = \frac{1}{M} \sum_{m=1}^{M} f(\boldsymbol{\theta}^{(m)}, \boldsymbol{\gamma}|\mathbf{y})$$

where $m = 1, \ldots, M$ is the index for the posterior draws.

- *Problem: For Bayesian Variable Selection models, this is intractable because need to sum over all 2^p models.*

Approach 2: Latent continuous variables

- For the Bayesian Variable Selection model, assume there is underlying standard normal random variable Z_j for each $j = 1, \ldots, p$ such that

$$\gamma_j = \begin{cases} 1 & Z_j \geq 0 \\ 0 & Z_j < 0 \end{cases}$$

- Probabilities are preserved: $P[\gamma_j = 1] = P[Z_j \geq 0] = 0.5$.
- *Problems:*
    - *Does this make sense?*
    - *The discrete variables still "exist", so possibly derivatives will break.*

Approach 3: Use Gibbs sampler

- Sample the continuous parameters $\boldsymbol{\theta}$ using HMC.
- At each iteration m, use $\boldsymbol{\theta}^{(m)}$ in the Gibbs conditional densities to sample $\boldsymbol{\gamma}$.
- *Problem: Have to write code for the HMC sampler, which won't include all the automatic tuning that Stan has.*

1 Bayesian Variable Selection

2 I-priors

3 Bayesian I-prior linear models

4 Hamiltonian Monte Carlo

5 Summary

Summary

- For our I-prior Bayesian Variable Selection model
  - ▶ Promising results in both simulated and real-world data.
  - ▶ The individual scale parameters $\lambda_1, \ldots, \lambda_p$ are important.
  - ▶ We have used ML estimate for $\boldsymbol{\lambda}$ in our Bayesian model.

Summary

- For our I-prior Bayesian Variable Selection model
    - ▸ Promising results in both simulated and real-world data.
    - ▸ The individual scale parameters $\lambda_1, \ldots, \lambda_p$ are important.
    - ▸ We have used ML estimate for $\boldsymbol{\lambda}$ in our Bayesian model.
- To-do list:
    - ▸ Any model consistency results for Bayesian variable selection models?
    - ▸ Any mathematical justification as to why we should use individual scale parameters?
    - ▸ Does the off-diagonal elements in the I-prior covariance matrix affect variable selection results in multicollinearity situations?

Summary

- For our I-prior Bayesian Variable Selection model
    - Promising results in both simulated and real-world data.
    - The individual scale parameters $\lambda_1, \ldots, \lambda_p$ are important.
    - We have used ML estimate for $\boldsymbol{\lambda}$ in our Bayesian model.
- To-do list:
    - Any model consistency results for Bayesian variable selection models?
    - Any mathematical justification as to why we should use individual scale parameters?
    - Does the off-diagonal elements in the I-prior covariance matrix affect variable selection results in multicollinearity situations?
- Wishlist: Make HMC work for Bayesian variable selection models.

What we've seen today

1 I-prior models estimated using ML methods (EM algorithm) and use of the iprior package in R.

2 Shrinkage properties of I-priors for use in Bayesian variable selection.

3 Bayesian estimation in JAGS.

4 Shiny apps for reactive programming.

5 Hamiltonian dynamics and Hamiltonian Monte Carlo.

6 Bayesian inference using HMC via Stan.

7 knitr for combining (evaluated) R code and plots into documents.

8 Git and GitHub for version control.

knitr example

- You type:

```
<<chunk.name, echo = TRUE>>=
x <- rnorm(100)
max(x)
hist(x)
@
```

- The output:

```
x <- rnorm(100)
max(x)

## [1] 2.375356
```

```
hist(x)
```


Histogram of x
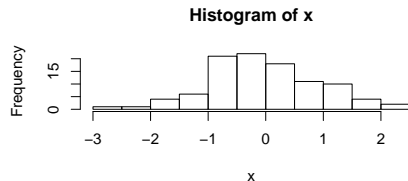
References I

Casella, G. and Moreno, E. (2006).
Objective Bayesian Variable Selection.
Journal of the American Statistical Association, 101(473):157–167.

Dellaportas, P., Forster, J. J., and Ntzoufras, I. (2002).
On Bayesian model and variable selection using MCMC.
Statistics and Computing, 12(1):27–36.

George, E. I. and McCulloch, R. E. (1993).
Variable Selection Via Gibbs Sampling.
Journal of the American Statistical Association, 88(423):881–889.

Kuo, L. and Mallick, B. (1998).
Variable selection for regression models.
Sankhya: The Indian Journal of Statistics, Series B, 60(1):65–81.

References II

McDonald, G. C. and Schwing, R. C. (1973).
Instabilities of regression estimates relating air pollution to mortality.
Technometrics, 15(3):463–481.

Ntzoufras, I. (2011).
Bayesian Modeling Using WinBUGS.
Wiley.

Zellner, A. (1986).
On assessing prior distributions and Bayesian regresison analysis with
g-prior distributions.
*Bayesian Inference and Decision Techniques: Essays in Honor of Bruno
de Finetti*, pages 233–243.

HMC References

Betancourt, M. (2014).
Efficient Bayesian inference with Hamiltonian Monte Carlo.
Machine Learning Summer School (Iceland 2014).
https://www.youtube.com/watch?v=pHsuIaPbNbY.

Betancourt, M. (2016).
Scalable Bayesian Inference with Hamiltonian Monte Carlo.
Tokyo Stan. https://www.youtube.com/watch?v=VnNdhsm0rJQ.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987).
Hybrid monte carlo.
Physics letters B, 195(2):216–222.

Neal, R. M. et al. (2011).
MCMC using Hamiltonian dynamics.
Handbook of Markov Chain Monte Carlo, 2:113–162.

Stuff I

Chang, W., Cheng, J., Allaire, J., Xie, Y., and McPherson, J. (2016).
shiny: Web Application Framework for R.
R package version 0.13.2.

GitHub.
GitHub Guides.
https://guides.github.com.

Jamil, H. (2016).
iprior: Linear Regression using I-Priors.
R package version 0.6.2. https://haziqjamil.github.io/iprior.

Plummer, M. (2016).
rjags: Bayesian Graphical Models using MCMC.
R package version 4-6.

Stuff II

RStudio Team (2015).
RStudio: Integrated Development Environment for R.
RStudio, Inc., Boston, MA. http://www.rstudio.com.

Stan Development Team (2015).
Stan: A C++ Library for Probability and Sampling.
Version 2.10.0. http://mc-stan.org.

Wickham, H. (2015).
R Packages: Organize, Test, Document, and Share Your Code.
O'Reilly Media, Inc. http://r-pkgs.had.co.nz.

Xie, Y. (2016).
knitr: A General-Purpose Package for Dynamic Report Generation in R.
R package version 1.14. http://yihui.name/knitr.

End

Thank you!

Minimising profiled deviance (à la `lme4`)

- Very fast algorithm to obtain MLEs of mixed-effects models by using sparse Cholesky decomposition.
- Consider the mixed-effects model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}$$
$$\boldsymbol{\epsilon} \sim \mathsf{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$$
$$\mathbf{b} \sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Sigma})$$

- Suppose that $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{\Lambda}_{\boldsymbol{\theta}} \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{\top}$. Then the following model is equivalent, where we have used the subsitution $\mathbf{b} = \boldsymbol{\Lambda}_{\boldsymbol{\theta}} \mathbf{u}$:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\Lambda}_{\boldsymbol{\theta}}\mathbf{u} + \boldsymbol{\epsilon}$$
$$\boldsymbol{\epsilon} \sim \mathsf{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$$
$$\mathbf{u} \sim \mathsf{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$$

Minimising profiled deviance (à la `lme4`) cont.

- The density of interest is $f(\mathbf{y}) = \int h(\mathbf{u}) \, d\mathbf{u}$, where

$$
\begin{aligned}
h(\mathbf{u}) &= f(\mathbf{y}|\mathbf{u})f(\mathbf{u}) \\
&= (2\pi\sigma^2)^{-(n+q)/2} \exp\left[-\frac{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\boldsymbol{\Lambda}_{\boldsymbol{\theta}}\mathbf{u}\|^2 + \|\mathbf{u}\|^2}{2\sigma^2}\right]
\end{aligned}
$$

Minimising profiled deviance (à la `lme4`) cont.

- The density of interest is $f(\mathbf{y}) = \int h(\mathbf{u}) \, d\mathbf{u}$, where

$$h(\mathbf{u}) = f(\mathbf{y}|\mathbf{u})f(\mathbf{u})$$
$$= (2\pi\sigma^2)^{-(n+q)/2} \exp\left[-\frac{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\boldsymbol{\Lambda}_{\boldsymbol{\theta}}\mathbf{u}\|^2 + \|\mathbf{u}\|^2}{2\sigma^2}\right]$$

- Each calculation of $f(\mathbf{y})$ involves obtaining the conditional modes

$$\tilde{\mathbf{u}}(\boldsymbol{\theta}, \boldsymbol{\beta}) = \arg\min_{\mathbf{u}} d(\mathbf{u}, \boldsymbol{\theta}, \boldsymbol{\beta})$$

by computing the sparse Cholesky factorisation

$$\mathbf{L}_{\boldsymbol{\theta}}\mathbf{L}_{\boldsymbol{\theta}}^{\top} = \boldsymbol{\Lambda}_{\boldsymbol{\theta}}^{\top}\mathbf{Z}^{\top}\mathbf{Z}\boldsymbol{\Lambda}_{\boldsymbol{\theta}} + \mathbf{I}_q,$$

and solving $\mathbf{L}_{\boldsymbol{\theta}}^{\top}\tilde{\mathbf{u}} = c(\boldsymbol{\theta}, \boldsymbol{\beta})$ by back substitution.

Minimising profiled deviance (à la `lme4`) cont.

- For linear mixed models, $f(\mathbf{y}) = \int h(\mathbf{u})\, d\mathbf{u}$ has a closed-form expression in terms of $\mathbf{L}_{\boldsymbol{\theta}}$ and $\tilde{\mathbf{u}}(\boldsymbol{\theta}, \boldsymbol{\beta})$:

$$f(\mathbf{y}) = (2\pi\sigma^2)^{-n/2} |\mathbf{L}_{\boldsymbol{\theta}}|^{-1} \exp\left[-\frac{d(\tilde{\mathbf{u}}, \boldsymbol{\theta}, \boldsymbol{\beta})}{2\sigma^2}\right]. \qquad (4)$$

- On the deviance scale, we have $D(\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma) = -2\log f(\mathbf{y})$. The value of σ which minimises the deviance is

$$\sigma^2(\boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{d(\tilde{\mathbf{u}}, \boldsymbol{\theta}, \boldsymbol{\beta})}{n}.$$

- Plugging this back into (4), we obtain the profiled deviance

$$D(\boldsymbol{\theta}, \boldsymbol{\beta}) = 2\log|\mathbf{L}_{\boldsymbol{\theta}}| + n\left(1 + \log\left(2\pi\frac{d(\tilde{\mathbf{u}}, \boldsymbol{\theta}, \boldsymbol{\beta})}{n}\right)\right)$$

  which is then minimised to obtain MLEs $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\beta}}$ and $\sigma^2(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}})$.

Minimising profiled deviance (à la `lme4`) cont.

- "Eliminate" fixed effects $\boldsymbol{\beta}$.
  - Find conditional modes $\tilde{\boldsymbol{\beta}}(\boldsymbol{\theta})$

  $$\begin{pmatrix} \tilde{\mathbf{u}}(\boldsymbol{\theta}) \\ \tilde{\boldsymbol{\beta}}(\boldsymbol{\theta}) \end{pmatrix} = \underset{(\mathbf{u}, \boldsymbol{\beta})}{\arg\min}\, d(\mathbf{u}, \boldsymbol{\beta}, \boldsymbol{\theta})$$

  via a sparse Cholesky decomposition. Following a similar method as before, obtain a profiled deviance which depends only on $\boldsymbol{\theta}$

  $$D(\boldsymbol{\theta}) = 2\log|\mathbf{L}_{\boldsymbol{\theta}}| + n\left(1 + \log\left(2\pi\frac{d(\tilde{\mathbf{u}}, \boldsymbol{\theta}, \tilde{\boldsymbol{\beta}})}{n}\right)\right).$$

Minimising profiled deviance (à la `lme4`) cont.

- "Eliminate" fixed effects $\boldsymbol{\beta}$.
  - ▸ Find conditional modes $\tilde{\boldsymbol{\beta}}(\boldsymbol{\theta})$

  $$\begin{pmatrix} \tilde{\mathbf{u}}(\boldsymbol{\theta}) \\ \tilde{\boldsymbol{\beta}}(\boldsymbol{\theta}) \end{pmatrix} = \underset{(\mathbf{u}, \boldsymbol{\beta})}{\arg\min} \, d(\mathbf{u}, \boldsymbol{\beta}, \boldsymbol{\theta})$$

  via a sparse Cholesky decomposition. Following a similar method as before, obtain a profiled deviance which depends only on $\boldsymbol{\theta}$

  $$D(\boldsymbol{\theta}) = 2 \log |\mathbf{L}_{\boldsymbol{\theta}}| + n \left(1 + \log \left(2\pi \frac{d(\tilde{\mathbf{u}}, \boldsymbol{\theta}, \tilde{\boldsymbol{\beta}})}{n} \right) \right).$$

  - ▸ In addition, use the restricted maximum likelihood (REML) criterion

  $$D_R(\boldsymbol{\theta}, \sigma) = -2 \log \int f(\mathbf{y}) \, d\boldsymbol{\beta}.$$

  Again, follow similar steps to obtain the profiled REML criterion

  $$D_R(\boldsymbol{\theta}) = 2 \log(|\mathbf{L}_{\boldsymbol{\theta}}||\mathbf{L}_{\mathbf{X}}|) + (n - p) \left(1 + \log \left(2\pi \frac{d(\tilde{\mathbf{u}}, \boldsymbol{\theta}, \tilde{\boldsymbol{\beta}})}{n - p} \right) \right).$$

Coerce the *w* I-prior model into a mixed-model

$$\mathbf{y} = \boldsymbol{\alpha} + (\lambda_1 \mathbf{H}_1 + \cdots + \lambda_p \mathbf{H}_p)\mathbf{w} + \boldsymbol{\epsilon}$$

$$= \underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} \alpha \end{bmatrix}}_{\beta} + \underbrace{\begin{bmatrix} \mathbf{H}_1 & \cdots & \mathbf{H}_p \end{bmatrix}}_{\mathbf{Z}} \underbrace{\begin{bmatrix} \lambda_1 \mathbf{I}_n \\ \vdots \\ \lambda_p \mathbf{I}_n \end{bmatrix}}_{\boldsymbol{\Lambda}_\lambda} \mathbf{w} + \boldsymbol{\epsilon}$$

$$= \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\Lambda}_\lambda \mathbf{w} + \boldsymbol{\epsilon}$$

$$= \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\underbrace{\left(\frac{1}{\sigma^2}\boldsymbol{\Lambda}_\lambda\right)}_{\boldsymbol{\Lambda}_\theta}\underbrace{(\sigma^2 \mathbf{w})}_{\mathbf{u}} + \boldsymbol{\epsilon}$$

$$= \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\Lambda}_\theta \mathbf{u} + \boldsymbol{\epsilon}$$

- Our scale parameters are contained in $\boldsymbol{\theta} = (\lambda_1/\sigma^2, \ldots, \lambda_p/\sigma^2)$.
- *Problem: Our **Z** matrix is dense, so not able to use sparse Cholesky methods.*

MLE vs Bayes for scale parameters

- For each $k = 1, \ldots, p$, the maximum a posteriori (MAP) estimate is

$$\hat{\lambda}_k^{MAP} = \arg\max_{\lambda_k} f(\alpha, \boldsymbol{\beta}, \psi, \boldsymbol{\lambda}|\mathbf{y})$$

$$= \arg\max_{\lambda_k} f(\mathbf{y}, \boldsymbol{\beta}|\alpha, \psi, \boldsymbol{\lambda}) f(\psi) f(\lambda_1) \cdots f(\lambda_p)$$

$$= \arg\max_{\lambda_k} f(\mathbf{y}, \boldsymbol{\beta}|\alpha, \psi, \boldsymbol{\lambda}) f(\lambda_k)$$

whereas the ML estimate is

$$\hat{\lambda}_k^{ML} = \arg\max_{\lambda_k} f(\mathbf{y}; \boldsymbol{\lambda}, \psi)$$

$$= \arg\max_{\lambda_k} \int f(\mathbf{y}, \boldsymbol{\beta}; \boldsymbol{\lambda}, \psi) \, d\boldsymbol{\beta}$$

- $\hat{\lambda}_k^{MAP} = \hat{\lambda}_k^{ML}$ if the beta I-prior model is marginalised over $\boldsymbol{\beta}$, and a uniform prior is used for each λ_k.

`lme4` References I

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2015).
Fitting linear mixed-effects models using lme4.
Journal of Statistical Software, 67(1):1–48.