Bayesian Statistics for Genetics
Lec 6: Model Selection and Averaging

Ken Rice

Summer Institute in Statistical Genetics

July, 2018

Model selection and averaging

Diabetes example:

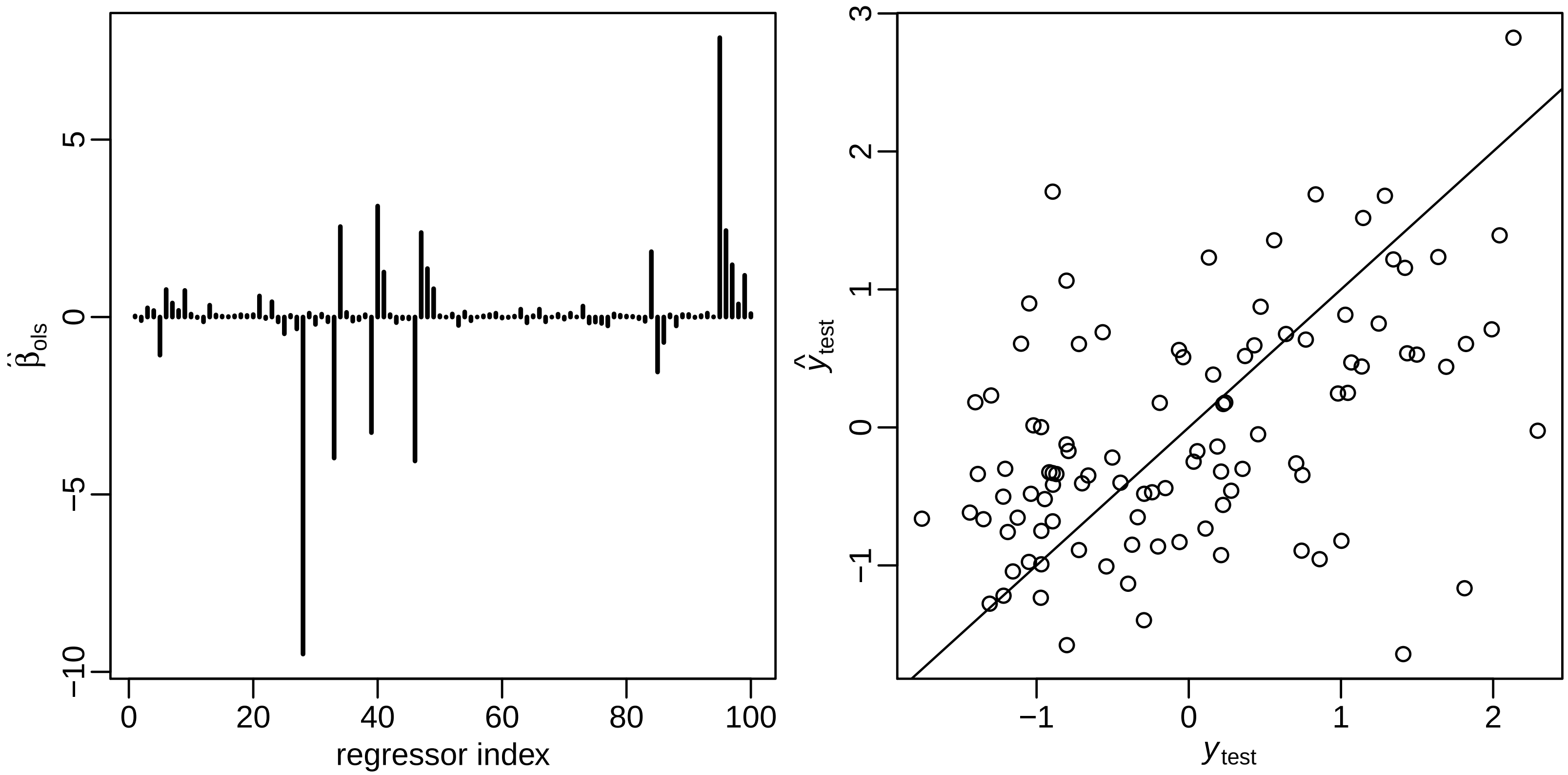
- 342 subjects
- $y_i =$ diabetes progression
- $\boldsymbol{x}_i =$ explanatory variables.

Each $\boldsymbol{x}_i$ includes

- 13 subject specific measurements $(x_{\text{age}}, x_{\text{sex}}, \ldots)$;
- $78 = \binom{13}{2}$ interaction terms $(x_{\text{age}} \cdot x_{\text{sex}}, \ldots)$;
- 9 quadratic terms (x_{sex} and three genetic variables are binary)

100 explanatory variables total!

OLS regression



$$\frac{1}{100}\sum(y_{\text{test},i}-\widehat{y}_{\text{test},i})^2 \;=\; 0.93$$

$$\frac{1}{100}\sum(y_{\text{test},i}-0)^2=\sum y^2_{\text{test},i} \;=\; 1.01$$

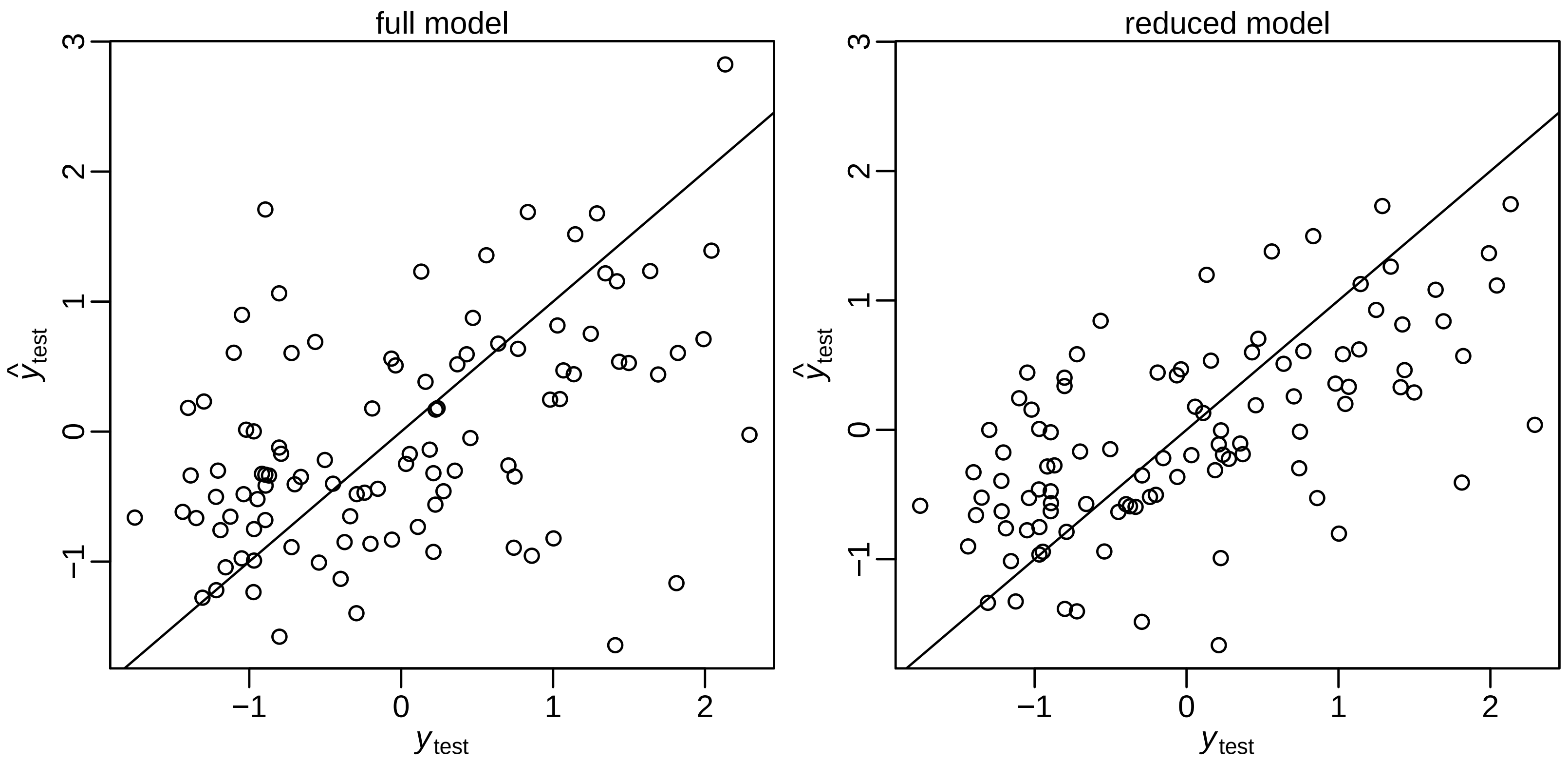
Backwards elimination

1. Obtain the estimator $\widehat{\boldsymbol{\beta}}_{\mathsf{ols}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y}$ and its t-statistics.

2. If there are any regressors j such that $|t_j| < t_{\mathsf{cutoff}}$,
   (a) find the regressor j_{min} having the smallest value of $|t_j|$ and remove column j_{min} from $\mathbf{X}$.
   (b) return to step 1.

3. If $|t_j| > t_{\mathsf{cutoff}}$ for all variables j remaining in the model, then stop.

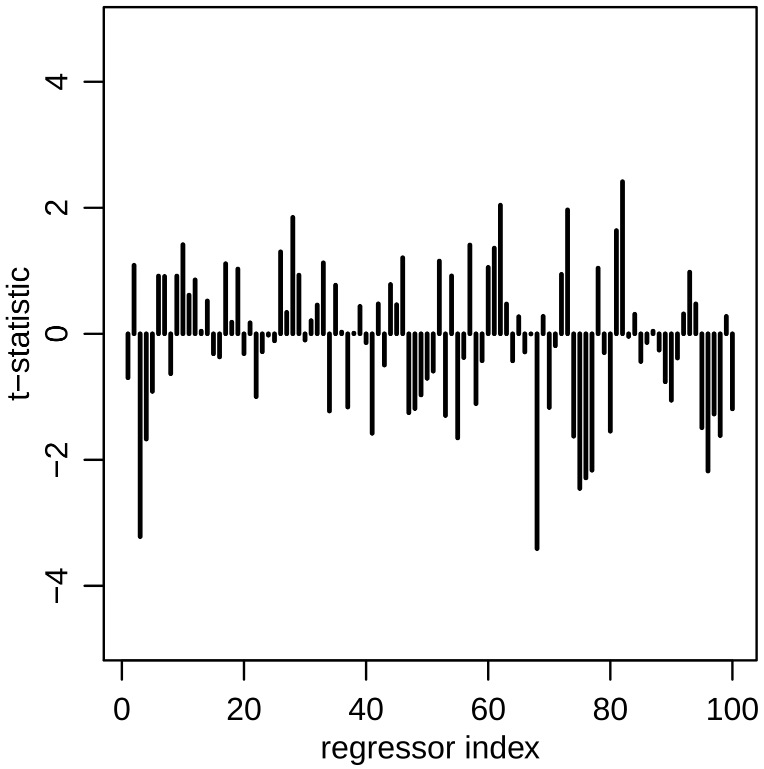
Backwards elimination



$$\frac{1}{100}\sum(y_{\text{test},i} - \hat{y}_{\text{test}^{bel},i})^2 = 0.64$$

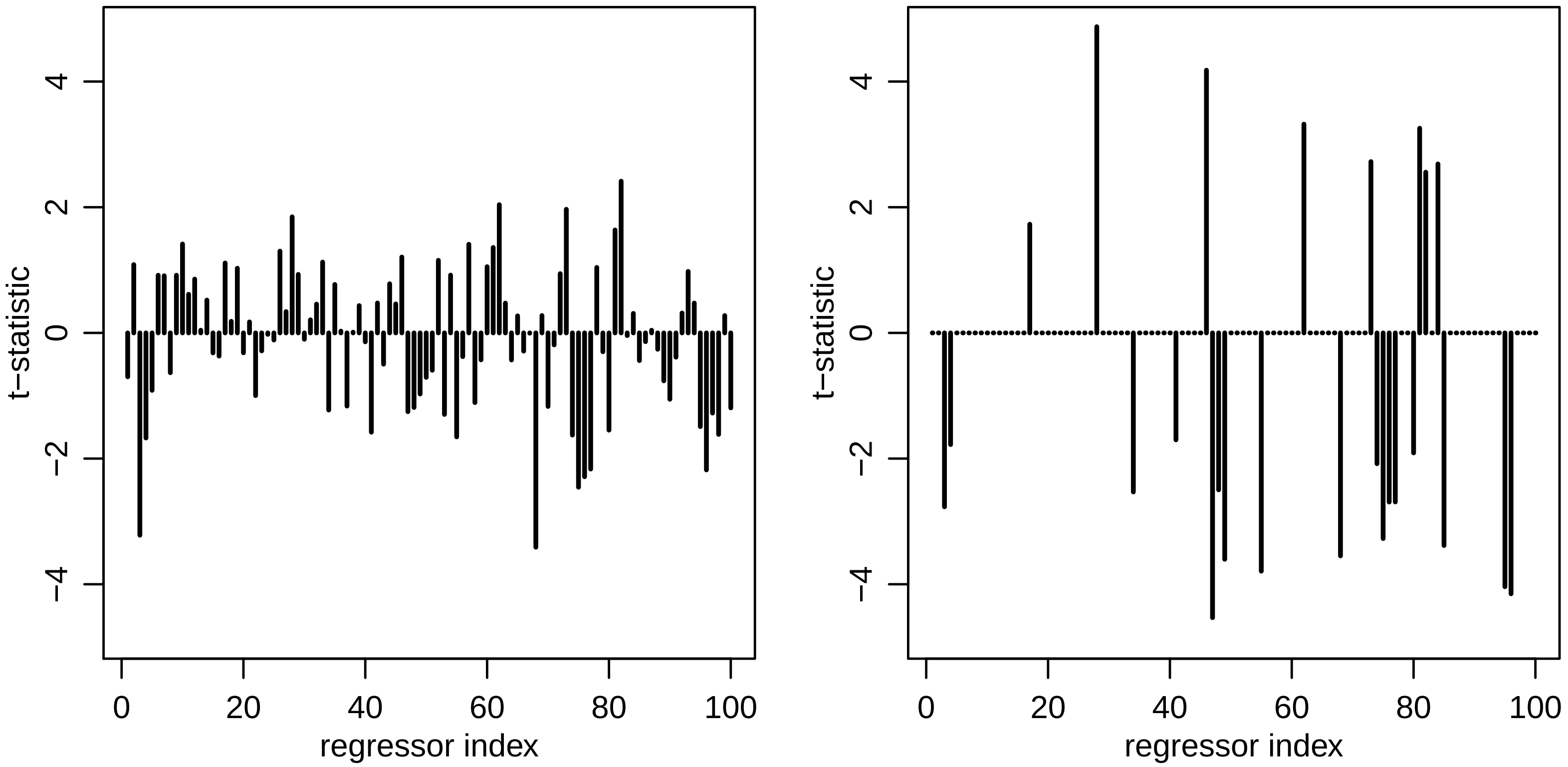
Spurious associations

Now try modeling permuted $y_{\pi(i)} = \boldsymbol{\beta}^T \boldsymbol{x}_i + \epsilon_i$ (and backwards-select)



Spurious associations

Now try modeling permuted $y_{\pi(i)} = \boldsymbol{\beta}^T \boldsymbol{x}_i + \epsilon_i$ (and backwards-select)



Spurious associations

```
sum(abs(t.bslperm)>2 )
## [1] 21
sum(abs(t.bslperm)>3 )
## [1] 12
sum(abs(t.bslperm)>4 )
## [1] 5
```

- 21 regressors have t-stats > 2 ($p \approx 0.05$)
- 12 regressors have t-stats > 3 ($p \approx 0.003$)
- 5 regressors have t-stats > 4 ($p \approx 0.00006$)

Often want some way to pick a sparse model – but this approach is not smart.

Bayesian model selection

Prior belief: $\beta_j \approx 0$ for many j's.

Formulation: Write $\beta_j = z_j \times b_j$, where $z_j \in \{0, 1\}$ and $b_j \in \mathbb{R}$.

$$y_i = z_1 b_1 x_{i,1} + \cdots + z_p b_p x_{i,p} + \epsilon_i.$$

For example, in the FTO experiment,

$$\begin{aligned}
\mathbb{E}[\,Y|\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{z} = (1, 0, 1, 0)\,] &= b_1 x_1 + b_3 x_3 \\
&= b_1 + b_3 \times \text{age} \\
\mathbb{E}[\,Y|\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{z} = (1, 1, 0, 0)\,] &= b_1 x_1 + b_2 x_2 \\
&= b_1 + b_2 \times \text{group} \\
\mathbb{E}[\,Y|\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{z} = (1, 1, 1, 0)\,] &= b_1 x_1 + b_2 x_2 + b_3 x_3 \\
&= b_1 + b_2 \times \text{group} + b_3 \times \text{age}.
\end{aligned}$$

Can think of each value of $\boldsymbol{z} = (z_1, \ldots, z_p)$ representing a different model.

Bayesian model selection

Or, think of z_j as unknown components in one (big) model – written informally as;

$$\begin{aligned} z_j &\overset{i.i.d.}{\sim} Bern(0.5) \\ b_j &\sim p(b_j) \\ \epsilon_i &\overset{i.i.d.}{\sim} N(0, \sigma^2) \\ \sigma^2 &\sim p(\sigma^2) \\ y_i &= z_1 b_1 x_{i,1} + \cdots + z_p b_p x_{i,p} + \epsilon_i \end{aligned}$$

Each of the 2^p possible values of of $\boldsymbol{z}$ has a posterior probability. (In the prior we treat them as a 'coin toss', equally likely to be 'in' or 'out'.)

Bayesian model comparison

Posterior probability

$$p(\boldsymbol{z}|\boldsymbol{y}, \mathsf{X}) = \frac{p(\boldsymbol{z})p(\boldsymbol{y}|\mathsf{X}, \boldsymbol{z})}{p(\boldsymbol{y}|\mathsf{X})}$$

Model comparison

$$\begin{array}{cccccc} \dfrac{p(\boldsymbol{z}_a|\boldsymbol{y}, \mathsf{X})}{p(\boldsymbol{z}_b|\boldsymbol{y}, \mathsf{X})} & = & \dfrac{p(\boldsymbol{z}_a)}{p(\boldsymbol{z}_b)} & \times & \dfrac{p(\boldsymbol{y}|\mathsf{X}, \boldsymbol{z}_a)}{p(\boldsymbol{y}|\mathsf{X}, \boldsymbol{z}_b)} \\ \text{posterior odds} & = & \text{prior odds} & \times & \text{“Bayes factor”} \end{array}$$

Note that the Bayes Factor (BF) does not depend on the prior for $\boldsymbol{z}$ – so the ‘coin toss’ prior is not crucial for this approach.

Parsimony

The formula for $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z})$ is complex, but

$$\frac{p(\boldsymbol{y}|\mathsf{X}, \boldsymbol{z}_a)}{p(\boldsymbol{y}|\mathsf{X}, \boldsymbol{z}_b)} = (1+n)^{(p_{z_b}-p_{z_a})/2} \left(\frac{s^2_{z_a}}{s^2_{z_b}}\right)^{1/2} \times \left(\frac{s^2_{z_b}+\mathsf{SSR}^{z_b}_g}{s^2_{z_a}+\mathsf{SSR}^{z_a}_g}\right)^{(n+1)/2} .$$

So a model $\boldsymbol{z}_a$ is penalized if;

- it is too complex (number of covariates p_A is large)
- it doesn't fit well (SSR^a_g is large)

FTO example

$$\begin{array}{rcccccccc}
\mathbb{E}[\,Y_i|\boldsymbol{\beta},\boldsymbol{x}_i\,] & = & \beta_1 x_{i,1} & + & \beta_2 x_{i,2} & + & \beta_3 x_{i,3} & + & \beta_4 x_{i,4} \\
& = & \beta_1 & + & \beta_2 \times \mathsf{grp}_i & + & \beta_3 \times \mathsf{age}_i & + & \beta_4 \times \mathsf{grp}_i \times \mathsf{age}_i\,.
\end{array}$$

effect of group $\Leftrightarrow$ one of more of β_2, β_4 not zero

$\boldsymbol{z}$	model	$\log p(\boldsymbol{y}\vert\mathbf{X},\boldsymbol{z})$	$p(\boldsymbol{z}\vert\boldsymbol{y},\mathbf{X})$
(1,0,0,0)	β_1	-71.82	0
(1,1,0,0)	$\beta_1 + \beta_2 \times \mathsf{grp}_i$	-70.04	0
(1,0,1,0)	$\beta_1 + \beta_3 \times \mathsf{age}_i$	-67.04	0
(1,1,1,0)	$\beta_1 + \beta_2 \times \mathsf{grp}_i + \beta_3 \times \mathsf{age}_i$	-61.19	0.63
(1,1,1,1)	$\beta_1 + \beta_2 \times \mathsf{grp}_i + \beta_3 \times \mathsf{age}_i + \beta_4 \times \mathsf{grp}_i \times \mathsf{age}_i$	-61.72	0.37

$$\begin{array}{rcl}
\mathbb{P}[\,\beta_2 \ \text{ or } \ \beta_4 \neq 0\,] & = & 0.60 \\
\mathbb{P}[\,\beta_2 \ \text{ or } \ \beta_4 \neq 0|\boldsymbol{y},\mathbf{X}\,] & \approx & 1
\end{array}$$

FTO example: using JAGS

Using the conjugate g-prior is a little artificial here;

- Each sub-model has a prior that corresponds to one observation's information, but those observations are not the same.
- It's strange to support the model with all $\beta_j = 0$, i.e. where $\mathbb{E}[\,Y_i | x_i\,]$ is exactly zero for everyone

So we'll instead use a general-purpose Gibbs sampler for the same model, but with $z_1 = 1$ (forcing an intercept) and

$$\begin{aligned} z_j &\overset{i.i.d.}{\sim} Bern(0.5) \\ b_j &\sim N(0, 10), \text{ for } j = 2, 3, 4 \\ \epsilon_i &\overset{i.i.d.}{\sim} N(0, \sigma^2) \\ 1/\sigma^2 &\sim \Gamma(0.5, 1.839) \text{ ... as in Lec 4} \\ y_i &= z_1 b_1 x_{i,1} + \cdots + z_p b_p x_{i,p} + \epsilon_i \end{aligned}$$

Reminder: Gibbs sampler

Goal: A *Monte Carlo* approximation to $p(x, y, z)$

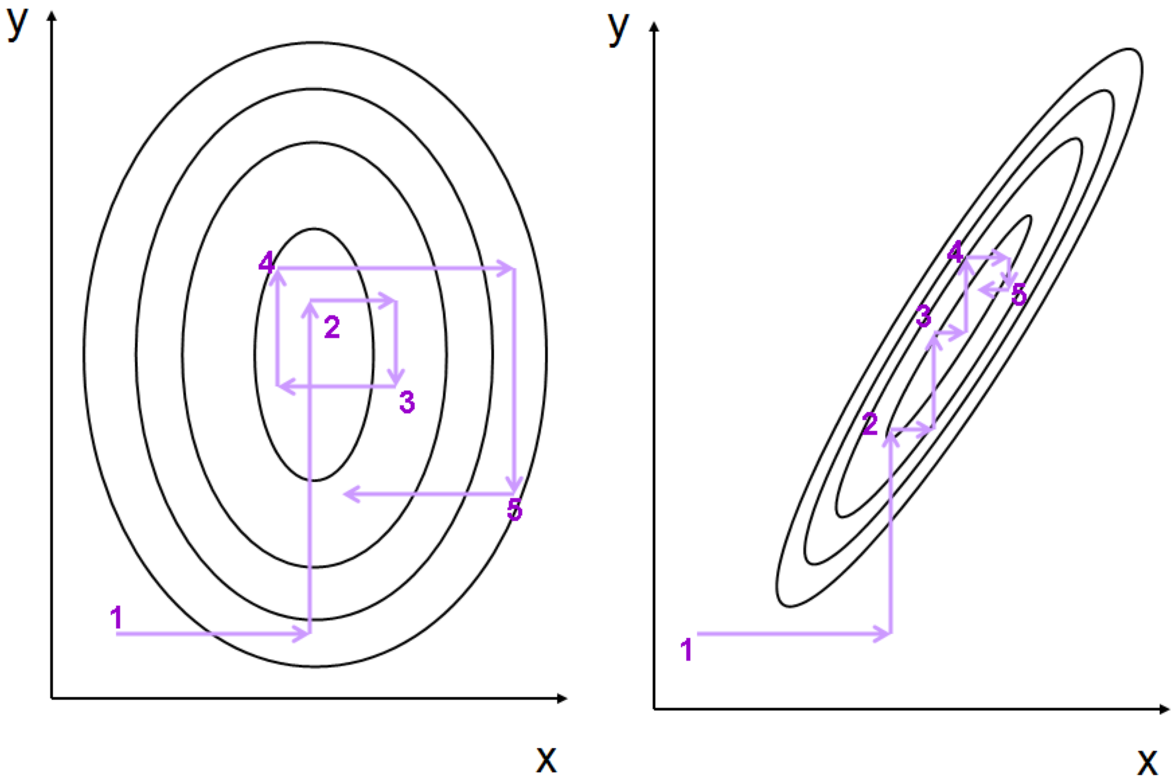
Given $\{x^{(s)}, y^{(s)}, z^{(s)}\}$,

1. simulate $x^{(s+1)} \sim p(x|y^{(s)}, z^{(s)})$,
2. simulate $y^{(s+1)} \sim p(y|x^{(s+1)}, z^{(s)})$,
3. simulate $z^{(s+1)} \sim p(z|x^{s+1)}, y^{(s+1)})$.

This generates $\{x^{(s+1)}, y^{(s+1)}, z^{(s+1)}\}$ – and then 'go round' again, many times. Repeated many times, this generates $\{x^{(1)}, y^{(1)}, z^{(1)}\}, \ldots, \{x^{(S)}, y^{(S)}, z^{(S)}\}$

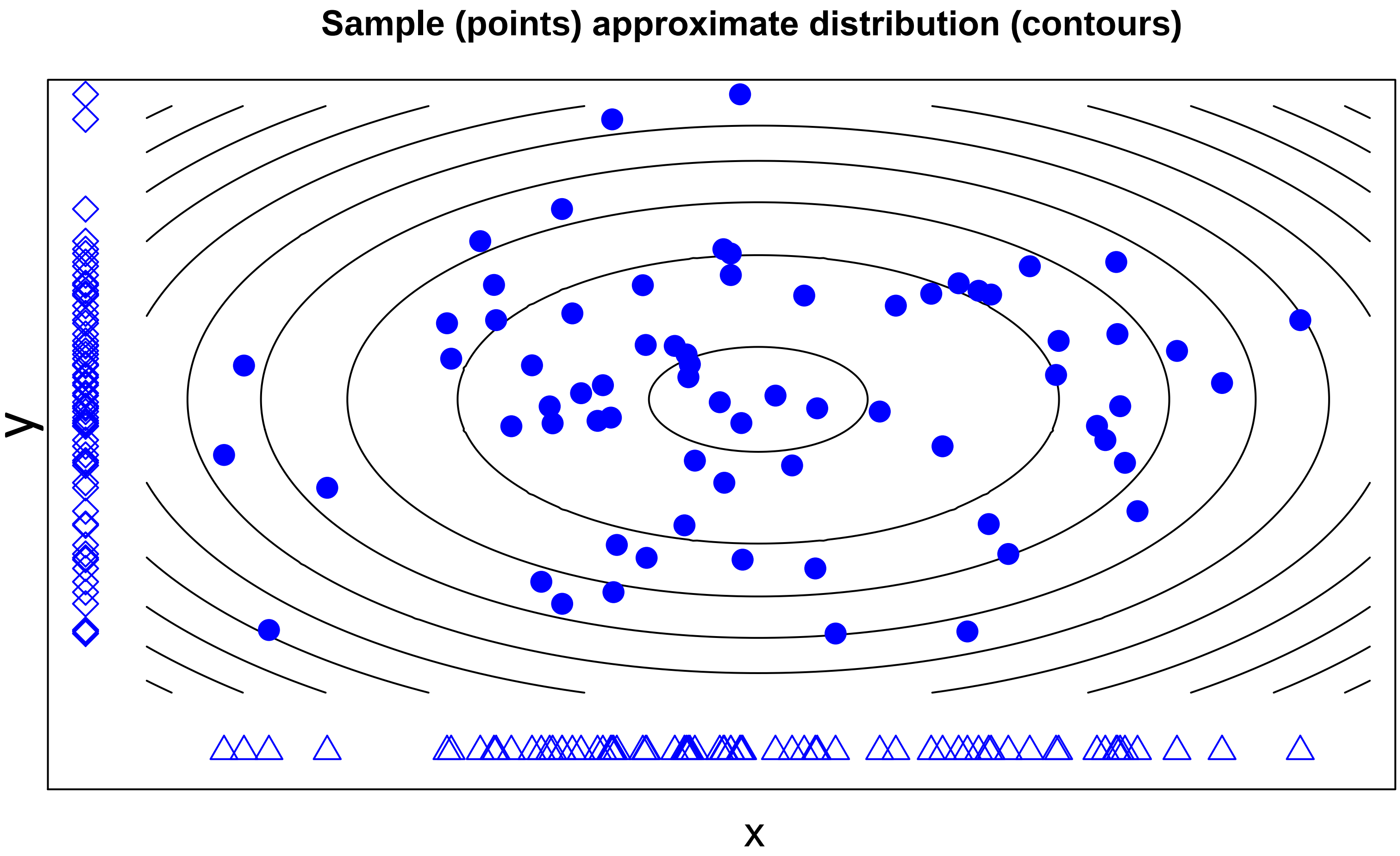
Reminder: Gibbs sampler

For a couple of two-dimensional examples;



Reminder: Gibbs sampler

Output from a short sampler;



Reminder: Gibbs sampler

Repeated many times, this gives $\{x^{(1)}, y^{(1)}, z^{(1)}\}, \ldots, \{x^{(S)}, y^{(S)}, z^{(S)}\}$

The distribution of this sequence approximates $p(x, y, z)$:

$$\begin{aligned}
\frac{1}{S}\sum x^{(s)} &\approx \mathbb{E}x = \int x\ p(x,y,z)\ dx\ dy\ dz \\
\frac{\#(x^{(s)} \in A)}{S} &\approx \text{Pr}(x \in A) = \int\int\int_A p(x,y,z)\ dx\ dy\ dz \\
\frac{\#(\{x^{(s)}, y^{(s)}, z^{(s)}\} \in B)}{S} &\approx \underset{B}{\int\int\int} p(x,y,z)\ dx\ dy\ dz
\end{aligned}$$

By necessity, the sequence will frequently visit regions where $p(x, y, z)$ is large.

Gibbs sampling: for model selection

Goal Approximate $p(z_1, \ldots, z_p | \boldsymbol{y}, \mathsf{X})$.

Gibbs sampler: Given $\boldsymbol{z}^{(s)} = (z_1^{(s)}, \ldots, z_p^{(s)})$,

$$\begin{aligned}
z_1^{(s+1)} &\sim p(z_1 | z_2^{(s)}, \ldots, z_p^{(s)}, \boldsymbol{y}, \mathsf{X}) \\
z_2^{(s+1)} &\sim p(z_2 | z_1^{(s+1)}, z_3^{(s)}, \ldots, z_p^{(s)}, \boldsymbol{y}, \mathsf{X}) \\
&\vdots \\
z_p^{(s+1)} &\sim p(z_p | z_1^{(s+1)}, \ldots, z_{p-1}^{(s+1)}, \boldsymbol{y}, \mathsf{X})
\end{aligned}$$

This generates $\boldsymbol{z}^{(s+1)}$ from $\boldsymbol{z}^{(s)}$.

Repeating this generates $z^{(1)}, \ldots, z^{(S)}$ with which to approximate $p(\boldsymbol{z} | \boldsymbol{y}, \mathsf{X})$.

FTO example: using JAGS

Stan doesn't handle discrete parameters like the z_i well (yet) so we use JAGS – *Just Another Gibbs Sampler*. Like Stan, it writes and executes MCMC code, given a model and data.

```
library("rjags")
# first, write the model as to a text file
cat(file="linearprog2.txt", "model{
   for(j in 1:p){
      b[j]~dnorm(0, 0.1)
   }
   z[1] <- 1
   for(j in 2:p){
      z[j] ~ dbern(0.5)
   }
inv.sigma2 ~ dgamma( 0.5, 1.839 )
   sigma <- sqrt(1/inv.sigma2)
   for(i in 1:n){
mu[i] <- x[i,1]*b[1]*z[1] + x[i,2]*b[2]*z[2] + x[i,3]*b[3]*z[3] + x[i,4]*b[4]*z[4
y[i] ~ dnorm(mu[i], inv.sigma2) }
}")
# compile code based on model and data, then run chain
jags1 <- jags.model("linearprog2.txt", data=list(y=y,x=X, n=nrow(X), p=ncol(X)) )
update(jags1, 50000) # initial iteraions
```

FTO example: using JAGS

And some of the output;

```
> jags1.out <- coda.samples(jags1, c("b","inv.sigma2", "z"), n.iter=100000)[[1]]
> summary(jags1.out)
Iterations = 50001:150000
Number of chains = 1
Sample size per chain = 1e+05
1. Empirical mean and standard deviation for each variable & std err of the mean:
             Mean      SD  Naive SE Time-series SE
b[1]       0.7593 1.26609 0.0040037      0.0184052
b[2]       1.2431 2.71152 0.0085746      0.0300475
b[3]       2.6202 0.39962 0.0012637      0.0057575
b[4]       2.1791 0.62138 0.0019650      0.0091990
inv.sigma2 0.2676 0.09069 0.0002868      0.0004338
z[1]       1.0000 0.00000 0.0000000      0.0000000
z[2]       0.5604 0.49634 0.0015696      0.0058886
z[3]       1.0000 0.00000 0.0000000      0.0000000
z[4]       0.9928 0.08431 0.0002666      0.0015052
```

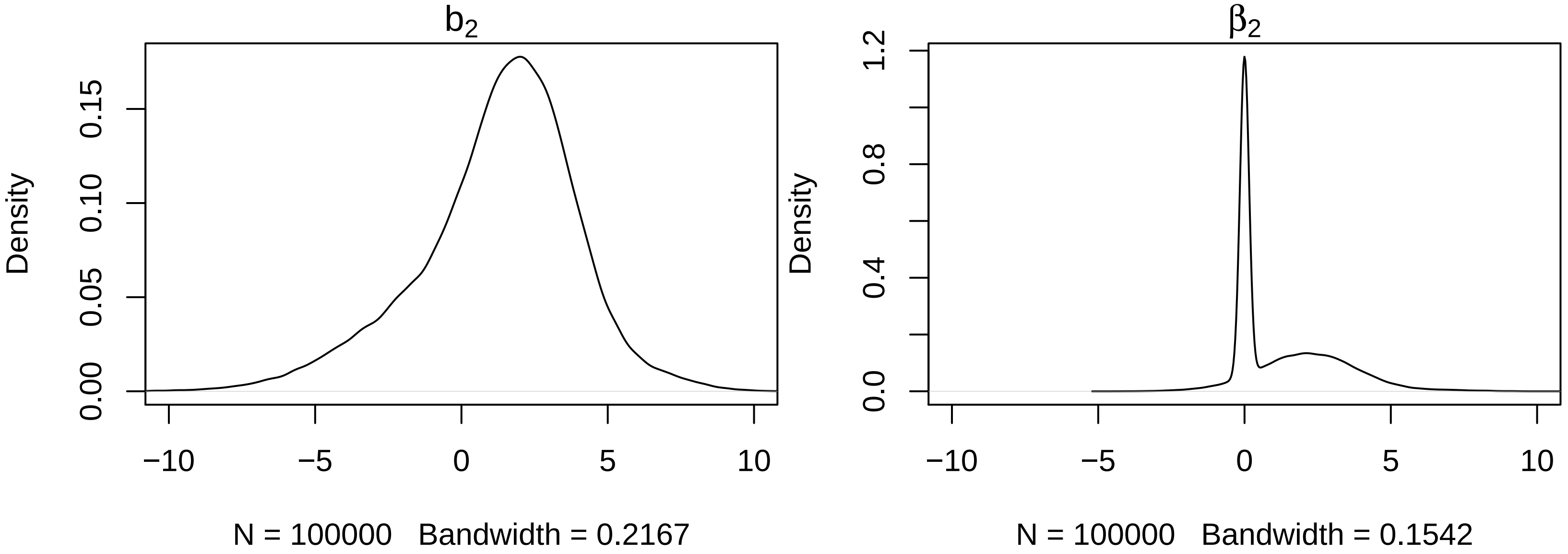
The coefficient of genotype is $\neq 0$ with 56% posterior support; the interaction term being $\neq 0$ has 99% support. The chain never moved from supporting age term $\neq 0$, so it has (approximately) 100% support.

FTO example: using JAGS

All 100,000 steps in the chain are stored, so we can assess posterior for other terms – for example the support for each set of included/excluded variables;

```
> table(apply(
    jags1.out[,c("z[1]","z[2]","z[3]","z[4]")], 1, paste, collapse="")
    )/100000
   1011    1110    1111
0.43851 0.00693 0.55456
```

And comparing the posteriors for b_2 to the posterior to the actual genotype coefficient, $\beta_2 = b_2 \times z_2$;



FTO example: using JAGS

Using MCMC, we have to start the ‘chain’ somewhere – but this arbitrary choice shouldn’t affect analysis

- After running long enough, the chains from any two starting points should *converge* to cover the posterior in the same way
- Less formally, after running long enough, chains forget where they started
- It’s pragmatic to run chains from a few different starting points, and check they give similar answers

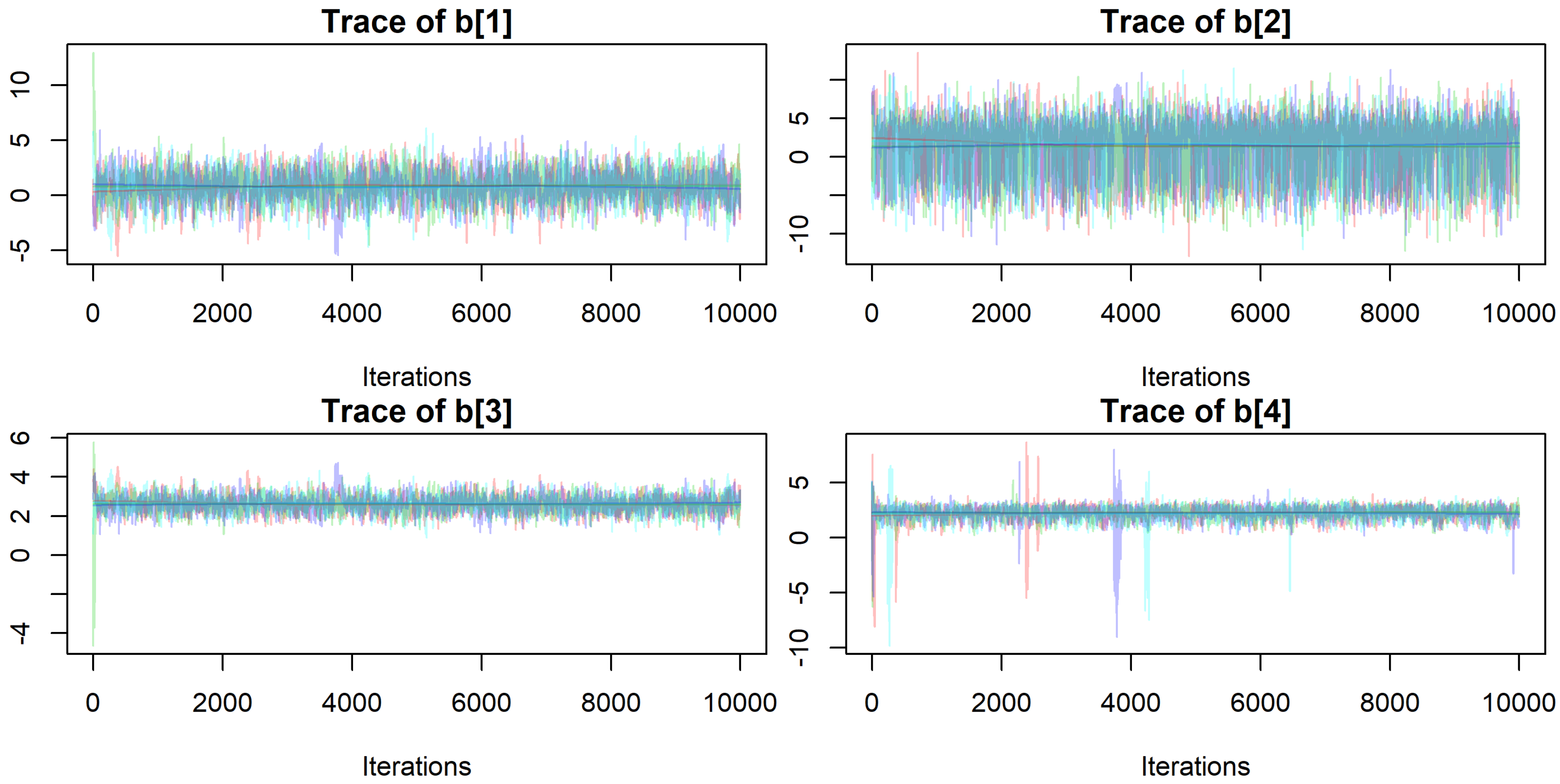
JAGS makes this fairly painless – here for 4 short chains;

```
set.seed(4)
inits1 <- list( b=rnorm(4,0,1),inv.sigma2=0.5,z=c(NA,0,1,0))
inits2 <- list( b=rnorm(4,0,1),inv.sigma2=0.5,z=c(NA,0,0,0))
inits3 <- list( b=rnorm(4,0,1),inv.sigma2=0.5,z=c(NA,1,1,0))
inits4 <- list( b=rnorm(4,0,1),inv.sigma2=0.5,z=c(NA,1,1,1))
jags2 <- jags.model("linearprog2.txt", data=list(y=y,x=X, n=nrow(X), p=ncol(X)),
inits=list( inits1, inits2, inits3, inits4), n.chains=4 )
jags2.out <- coda.samples(jags2, c("b"), n.iter=10000)
```

FTO example: using JAGS

An informal way to check for convergence is to look for differences in each chain's traceplot; (no issues seen here)

```
plot(jags2.out, trace=TRUE, density=FALSE, auto.layout=FALSE,
col=adjustcolor(2:5, alpha.f=0.25), lty=1)
```



FTO example: using JAGS

To more formally check convergence of the chains for individual parameters, the *Gelman-Rubin diagnostic* compares within-chain variance (W) to between-chain variance (B), using tools from mixed models.

For a converged chain we should get $R = W/B \approx 1\ldots$

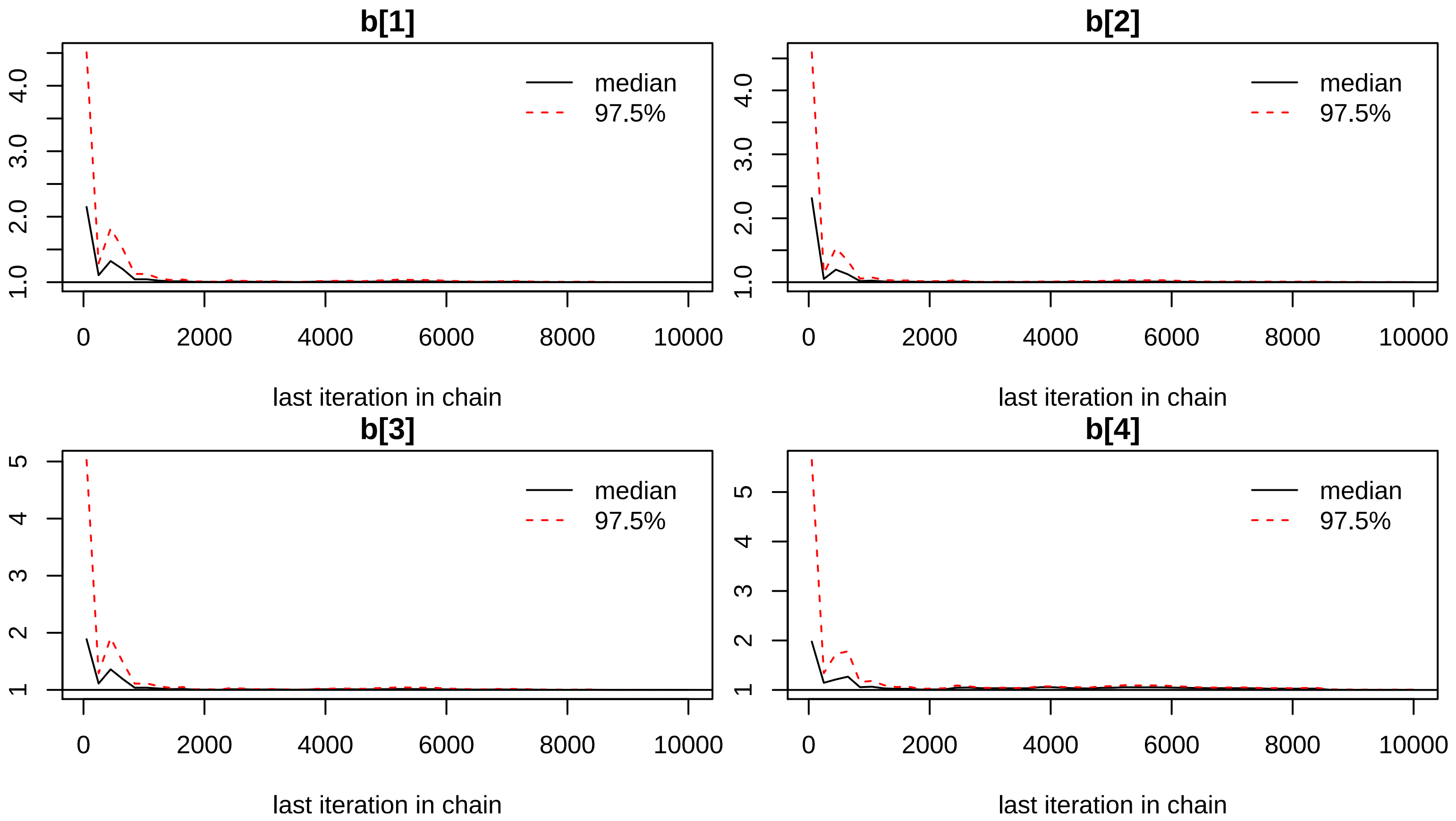
```
> gelman.diag(jags2.out)
Potential scale reduction factors:
     Point est. Upper C.I.
b[1]          1       1.00
b[2]          1       1.00
b[3]          1       1.00
b[4]          1       1.01
```

Similar ideas provide the effective sample size, i.e. roughly how big a simple random sample from the posterior is represented by the (auto-correlated) chain

```
> effectiveSize(jags2.out)
    b[1]     b[2]     b[3]     b[4]
1860.972 3057.044 1898.274 1586.170 # each from 40,000 iterations
```

FTO example: using JAGS

`gelman.plot(jags2.out)` shows how W/B evolves over iterations;



Ideally, don't start using the chain output until it looks like its converged – and even then, use as long a chain as you can manage – thinned, if memory is an issue.

Stochastic search: High dim'l regression

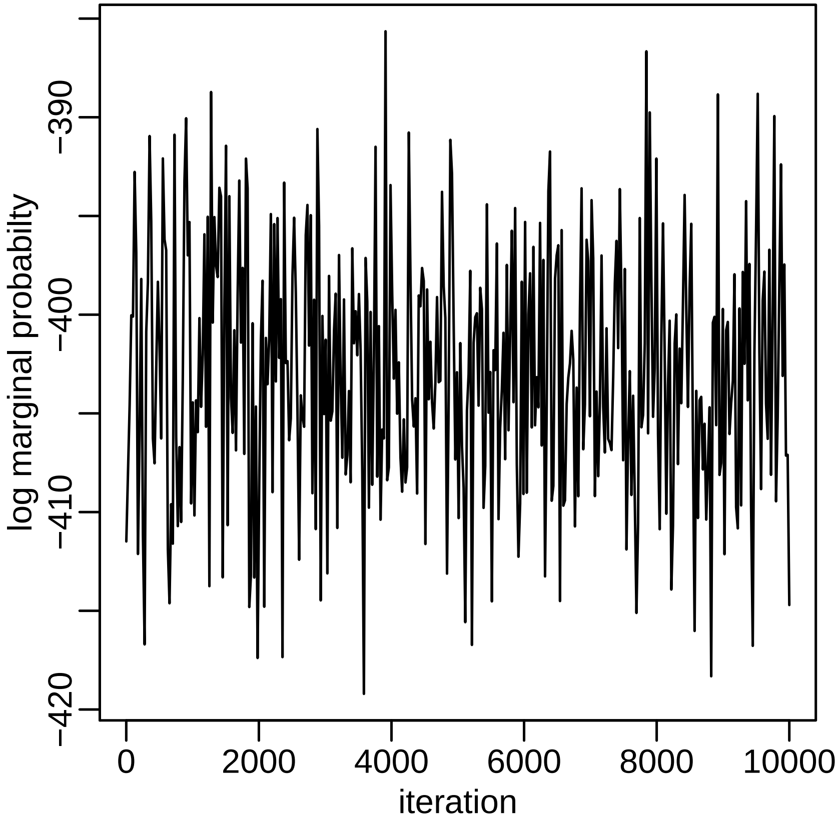
Diabetes example: $p = 100 \Rightarrow 2^{100} \approx 10^{30}$ models to consider.

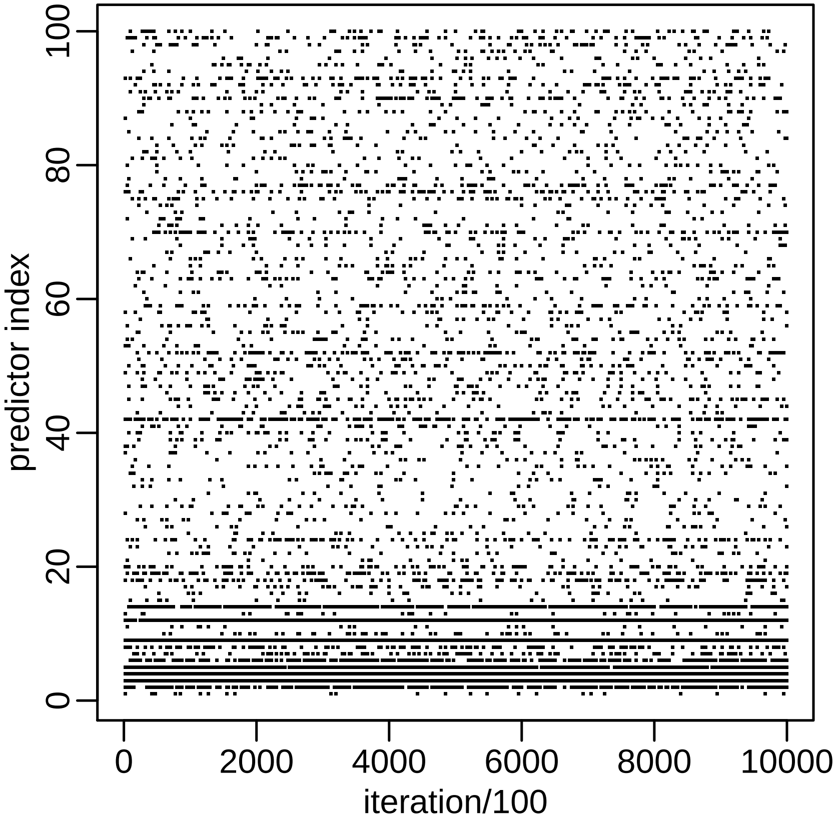
We can't compute $\mathbb{P}[\boldsymbol{z}|\boldsymbol{y},\mathbf{X}]$ for each $\boldsymbol{z}$. Instead, we hope to

- search for models $\boldsymbol{z}$ with high posterior probability;
- approximate $\beta_j = z_j \times b_j$ for each j;
- build a predictive model for $\boldsymbol{y}$.

We can view Gibbs Sampling here as a way to explore possible models – not to fully cover the whole parameter space. It will tend to sample models that better support the data.

Diabetes example





Marginal inference

What is the estimate of $\boldsymbol{\beta}$? Recall

$$\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p) = (b_1 z_1, \ldots, b_p, z_p)$$

Our Monte Carlo samples are

$$\begin{array}{lcllllll}
\boldsymbol{\beta}^{(1)} & = & (0 & -.299 & 0 & .427 & \cdots & .845) \\
\boldsymbol{\beta}^{(2)} & = & (0 & -.235 & .834 & .374 & \cdots & 0) \\
\vdots & & & & \vdots & & & \\
\boldsymbol{\beta}^{(S)} & = & (0 & -.315 & 0 & .536 & \cdots & 0)
\end{array}$$

A posterior mean for $\boldsymbol{\beta}$ is obtained in the usual way:

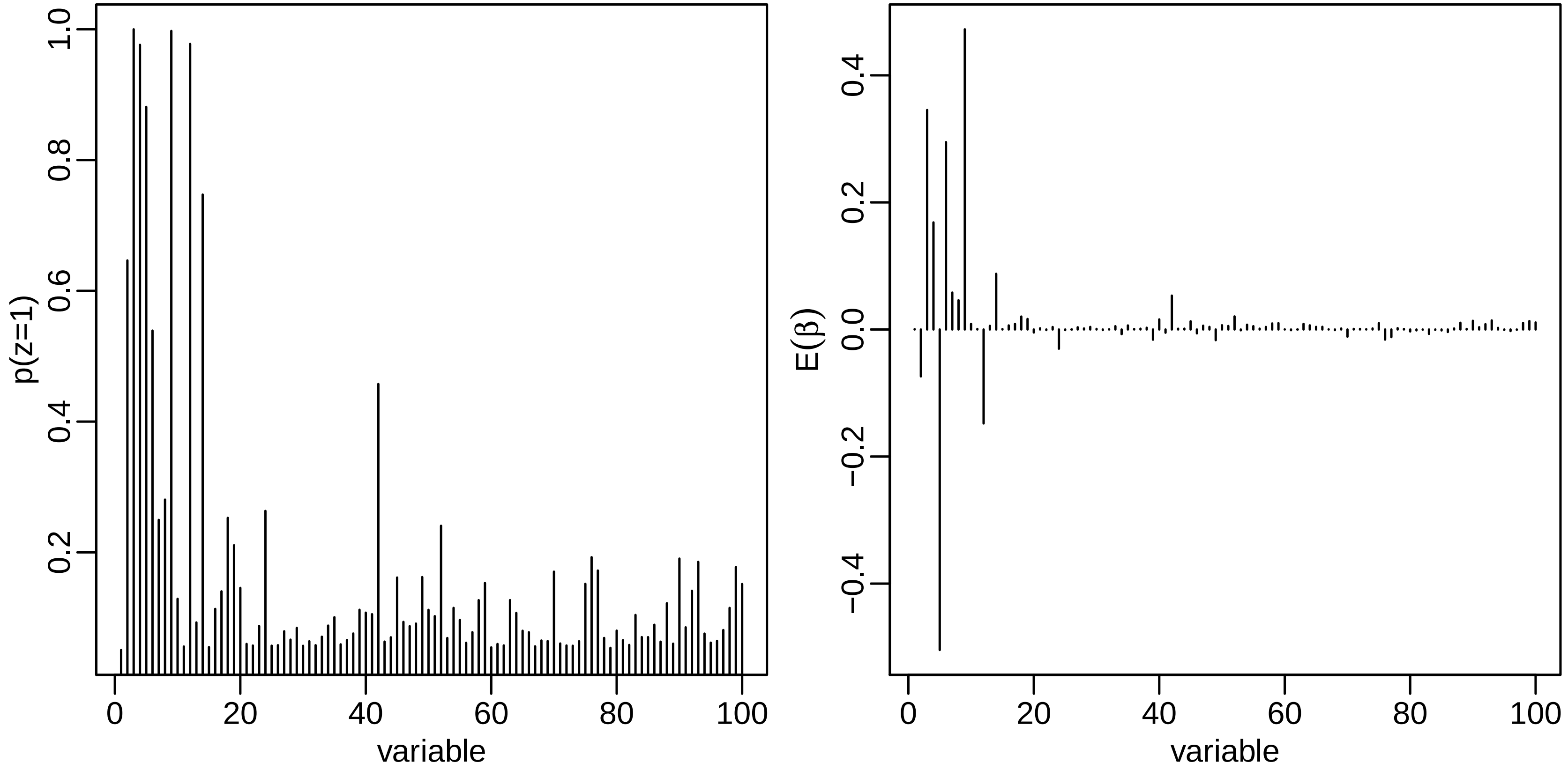
$$\hat{\boldsymbol{\beta}}^{\text{bayes}} = \frac{1}{S} \sum \boldsymbol{\beta}^{(s)} \approx \mathbb{E}\boldsymbol{\beta} | \boldsymbol{y}, \mathbf{X}$$

Out of sample predictions can be made with $\hat{\boldsymbol{\beta}}_{\text{bayes}}$:

$$\hat{y}^{bayes}_{\text{test,i}} = \hat{\boldsymbol{\beta}}^T_{\text{bayes}} \boldsymbol{x}_{\text{test,i}}$$

Out of sample prediction error: $\frac{1}{S} \sum (y_{\text{test,i}} - \hat{y}^{bayes}_{\text{test,i}})^2 = 0.485$

Marginal inference



Important variables

```
colnames(X)[ order(z.pmean,decreasing=TRUE)[1:10] ]
## [1] "bmi" "ltg" "g2" "map" "tc" "sex.age" "sex"
## [8] "ldl" "ltg.age" "tch"
```

```
colnames(X)[ order(b.pmean,decreasing=TRUE)[1:10] ]
## [1] "ltg" "bmi" "ldl" "map" "sex.age" "hdl" "ltg.age"
## [8] "tch" "glu.bmi" "map.sex"
```

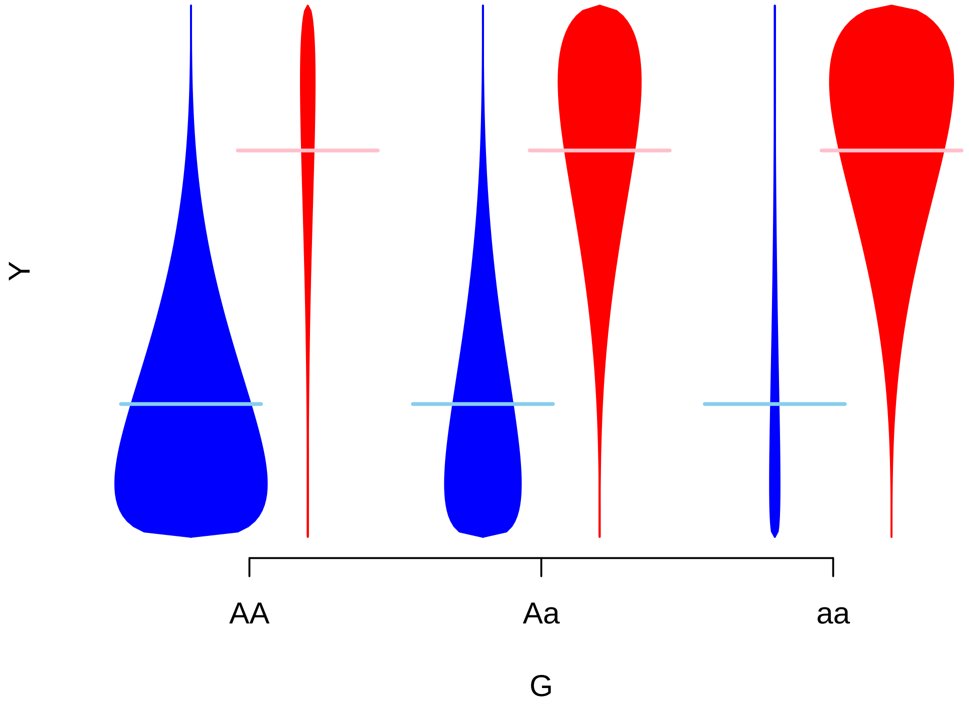
Other approaches, briefly

Model-averaging in this way gives an honest statement of uncertainty. But;

- Not all variables are in the model for the same reason – may want to ‘force’ some covariates into the model
- When selecting a single, parsimonious model, may want to maximize its ability to predict – not its probability of being true

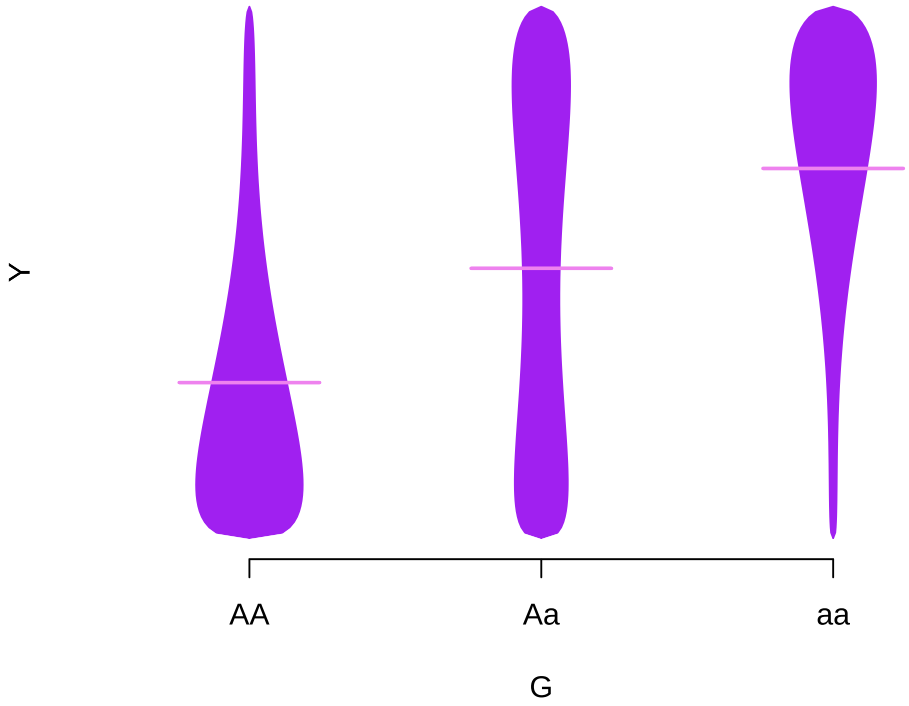
Confounding

‘Confounding’ means not being able to distinguish between a signal of interest, and some other cause. Here’s a genetic ‘signal’;



Confounding

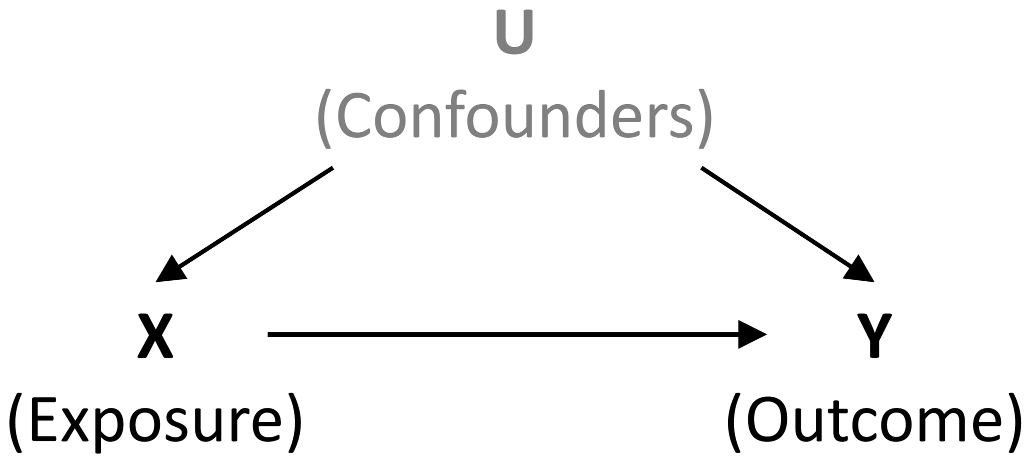
…which can be explained by ancestry, i.e. is confounded by ancestry



However, analysis that adjusts for ancestry would be of interest – even if models without it are better-supported.

Confounding

Directed Acyclic Graphs (DAGs) are a general language for confounding;



Arrows indicate causal relationships; confounding means 'back-door paths' exist; these can be removed by adjustment for confounders. In genetic association work, typically ancestry is the only plausible confounder - expression and methylation work is more complex.

Confounding

Bayesian Adjustment for Confounding (BAC, Wang et al 2012) specifies a model with

1. Dependence of outcome on the exposure and the set of confounders
2. Dependence of exposure on the set of confounders
3. Dependence between these models, making variable inclusion in (1) more likely if it is included in (2)

So BAC fits two set of z indicators, and links them. Modeling exposures is unusual – doing it well takes careful work.

The method is implemented in BEAU, a stand-alone R package, using approximate calculations for the posterior.

Prediction

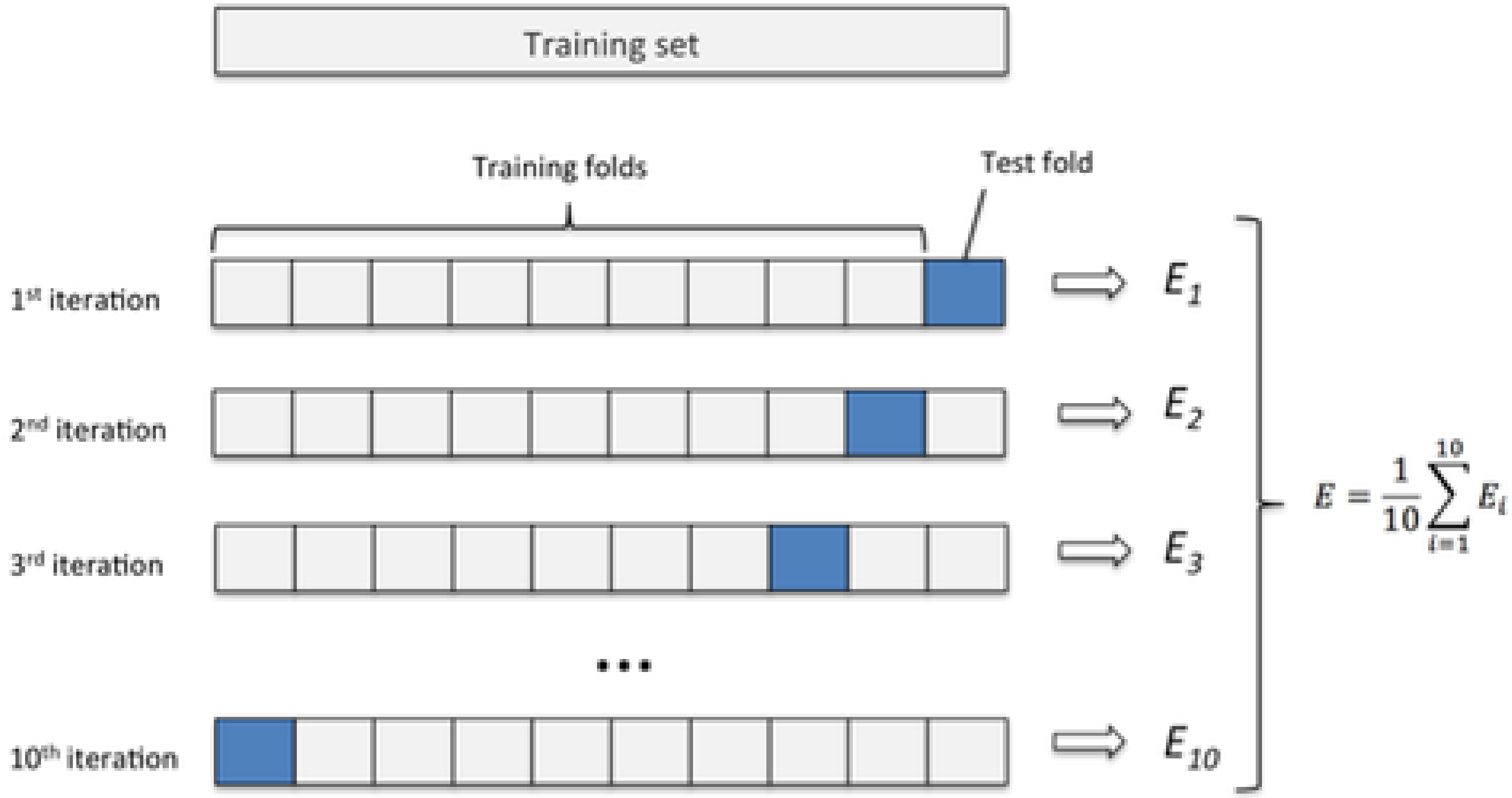
Understanding causes (and confounding) is often very important – but ability to predict can matter too;

- Remaining lifetime
- Drug response
- Telling 'good' genotyping from 'bad'

To pick a model here, it's reasonable to ask how well it would predict *in similarly-collected data*. This choice may not be the same as asking what the causes are, e.g. TV ownership rates predict child mortality but are not a cause.

Cross-validation

A natural way to assess how well a fitted model predicts is to fit it, and predict!



SSR is a common measure of predictive accuracy

Cross-validation

- SSR (squared error loss) is not the only option – need to consider the *loss* (*utility*) of particular predictions
- For categorical outcomes, could also weight misclassification rates (e.g. $P(1|0)$ and $P(0|1)$) – some mistakes may be worse than others
- Trickier still for dependent outcomes
- 10-fold cross-validation is typical
- Fitting multiple models with Gibbs sampling, and cross-validating each can be too slow

Approximate prediction measures

The standard ‘score’ is log *posterior predictive density*

$$\log p_{\text{ppost}}(y) = \log \int p(y|\theta)p(\theta|y)_{\text{obs}}d\theta).$$

Expected out-of-sample accuracy (over new datasets $\tilde{y}$) is defined as

$$elpd = E(\log p_{\text{ppost}}(\tilde{y})) = \int \log p_{\text{ppost}}(\tilde{y})q(\tilde{y})d\tilde{y}$$

for true density $q(\tilde{y})$. A natural way to estimate this is through the ‘in sample accuracy’,

$$lpd = \log \int p(y_{\text{obs}}|\theta)p(\theta|y)_{\text{obs}}d\theta,$$

but its double-use of the posterior leads to bias – worse with more parameters.

Approximate prediction measures

- Akaike's Information Criterion (AIC) approximates lpd by $\log p(y_{\text{obs}}|\widehat{\theta}_{MLE})$ – so is not Bayesian, and adds bias-correction k, the number of parameters
- Deviance Information Criterion (DIC) approximates lpd by $\log p(y_{\text{obs}}|E(\theta|y_{\text{obs}}))$ and adds the *effective number of parameters*,

$$p_D = 2\left(\log p(y_{\text{obs}}|\mathbb{E}[\,\theta|y_{\text{obs}}\,]) - \mathbb{E}_\theta[\,\log p(y_{\text{obs}}|\theta)\,]\right)$$

For either, in large samples – and under some conditions – choosing the model with the lowest value is equivalent to doing cross-validation.

Note: several other versions are available; AIC, DIC2, WAIC...

DIC examples

- Shriner and Yi 2009 use DIC in the context of multiple QTL Mapping – to select how many QTLs there are, and their locations
- Yu et al, 2012 use DIC studying gene×environment interactions, with a model that 'clusters' nearby* variants, so they have similar interaction effects. DIC is used to choose how many clusters

* ...using the Potts model