

Bayesian Statistics for Genetics Lecture 4: Linear models

July, 2021

In this session

- An important conjugacy result Normal prior, Normal likelihood
- Linear regression the standard approach for continuous outcomes (this material mostly review?)
- Bayesian linear regression
- A first look at Markov Chain Monte Carlo much more in later sessions

Normal prior, Normal likelihood

The best-known and most widely-used statistical is the Normal or Gaussian distribution, denoted $N(\mu, \sigma^2)$ for mean μ and variance σ^2 .

In sampling distributions, approximately Normal observations are expected when the variable is the sum of many similarly-sized small influences (e.g. height as a sum of SNP effects). This is called the *Central Limit Theorem*.



In priors, Normals are a convenient way to state beliefs, mainly because they're so familiar. They can also make some calculations easier (which is why Gauss liked them).

Normal prior, Normal likelihood: conjugacy



- Posterior mean is a weighted average of prior mean and sample mean
- Weights are \propto precision, i.e. 1/variance.
- Posterior precision = prior precision + precision from data
- With large *n* Bayes gives essentially standard result: point estimate \bar{Y} , 95% interval is $\bar{Y} \pm 1.96\sigma/\sqrt{n}$. With tiny *n*, posterior \approx prior.

FTO example: design

We'll soon use a version of Normal/Normal conjugacy in an analysis of the FTO gene, involved in growth and obesity.

Experimental design:

- 10 *fto* + /- mice
- 10 *fto* /- mice
- Mice are sacrificed at the end of 1-5 weeks of age (not 16 weeks, as here)
- Two mice in each group are sacrificed at each age
- n = 20 overall, so a small but well-controlled study



16-week-old

FTO example: data



FTO example: critical modeling assumption

In this example:

- Y = weight, which will be Normal **conditional** on covariate-specific mean
- $x_g = \text{indicator of fto heterozygote} \in \{0, 1\} = \text{number of "+" alleles}$
- $x_a = \text{age in weeks} \in \{1, 2, 3, 4, 5\}$

The critical modeling choice is how mean Y depends on the covariates. Without further assumptions, we could choose to model $\mathbb{E}[Y|X = x]$ as

This can't be 'wrong' – there is a mean weight for each genotype at each age – but we'd need to estimate 10 parameters from n = 20 data points.

But pragmatically, we instead assume smoothness in mean weight as a function of age. For each group, we assume

$$\mathbb{E}[Y|X_a = x_a] = \alpha_0 + \alpha_1 x_a$$

for some unknown α_0 and α_1 , that we'll estimate.

- Now we can *borrow strength* across the ages. For example, the difference in 4-5 week mice weights tells us about the difference in 1-2 week weights
- Can also write the assumption as

$$y = \alpha_0 + \alpha_1 x_a + \epsilon,$$

where ϵ is some mean-zero 'noise'

• A more complex linear regression might assume e.g. $y = \alpha_0 + \alpha_1 x_a + \alpha_2 x_a^2 + \alpha_3 x_a^3 + \epsilon$, — linearity means "linear in the parameters", i.e. adding several covariates, each multiplied by its own α coefficient.

We can also create variables, to describe models for both groups simultaneously:

 $\begin{array}{rcl} x_{i,1} &=& 1 \mbox{ for each subject } i \\ x_{i,2} &=& 0 \mbox{ if subject } i \mbox{ is homozygous, 1 if heterozygous} \\ x_{i,3} &=& age \mbox{ of subject } i, \mbox{ in weeks} \\ x_{i,4} &=& x_{i,2} \times x_{i,3} \\ \mathbb{E}[Y_i | \mathbf{X} = \mathbf{x}] &=& \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,3} + \beta_4 x_{i,4}, \\ \dots \mbox{ or write this as } Y_i &=& \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,3} + \beta_4 x_{i,4} + \epsilon_i \end{array}$

for some mean-zero ϵ_i . Note that under this model,

$$\mathbb{E}[Y|\mathbf{x},] = \beta_1 + \beta_3 \times \text{age if } x_2 = 0, \text{ and}$$
$$\mathbb{E}[Y|\mathbf{x}] = (\beta_1 + \beta_2) + (\beta_3 + \beta_4) \times \text{age if } x_2 = 1$$

Multiple linear regression

For graphical thinkers...



Normal linear regression

How do the Y_i vary around $\mathbb{E}[Y_i|\boldsymbol{\beta}, \boldsymbol{x}_i]$, ?

$$Y_i = \boldsymbol{\beta}^T \boldsymbol{x}_i + \epsilon_i$$

 $\epsilon_1, \dots, \epsilon_n \sim \text{ i.i.d. Normal}(0, \sigma^2).$

Assuming independent identically-distributed Normal errors gives the likelihood:



$$p(y_1, \dots, y_n | \boldsymbol{x}_1, \dots \boldsymbol{x}_n, \boldsymbol{\beta}, \sigma^2) = \prod_{i=1}^n p(y_i | \boldsymbol{x}_i, \boldsymbol{\beta}, \sigma^2)$$
$$= (2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \boldsymbol{\beta}^T \boldsymbol{x}_i)^2\right).$$

Note: in large(r) sample sizes, what we learn about the β_j is "robust" to the Normality assumption—but with n = 20 we do **rely** on the mean being linear in the x's, and on the ϵ_i 's variance σ^2 being constant with respect to x.

Linear regression: non-Bayes methods (*)

What values of β are consistent with our data y, X? Recall that

$$p(y_1, \dots, y_n | x_1, \dots, x_n, \beta, \sigma^2) = (2\pi\sigma^2)^{-n/2} \exp\{-\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \beta^T x_i)^2\}.$$

This is big when $SSR(\beta) = \sum (y_i - \beta^T x_i)^2$ is small. It can also be written as

$$SSR(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \boldsymbol{x}_i)^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) = \boldsymbol{y}^T \boldsymbol{y} - 2\boldsymbol{\beta}^T \boldsymbol{X}^T \boldsymbol{y} + \boldsymbol{\beta}^T \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{\beta},$$

where we use matrix notation, i.e.

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \boldsymbol{X}\boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{x}_1 \rightarrow \\ \boldsymbol{x}_2 \rightarrow \\ \vdots \\ \boldsymbol{x}_n \rightarrow \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^T \boldsymbol{\beta} \\ \boldsymbol{x}_2^T \boldsymbol{\beta} \\ \vdots \\ \boldsymbol{x}_n^T \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \beta_1 x_{1,1} + \dots + \beta_p x_{1,p} \\ \beta_2 x_{2,1} + \dots + \beta_p x_{2,p} \\ \vdots \\ \beta_1 x_{n,1} + \dots + \beta_p x_{n,p} \end{pmatrix}.$$

Linear regression: non-Bayes methods (*, almost)

Classical linear regression uses the $\hat{\beta}$ that makes SSR smallest, i.e. finds the 'best-fitting' model. To minimize SSR, 'recall' that...

- 1. a minimum of a function g(z) occurs at a value z such that $\frac{d}{dz}g(z) = 0$;
- 2. the derivative of g(z) = az is a and the derivative of $g(z) = bz^2$ is 2bz.

Doing this for SSR...

$$\frac{d}{d\beta} SSR(\beta) = \frac{d}{d\beta} \left(y^T y - 2\beta^T X^T y + \beta^T X^T X \beta \right)$$

$$= -2X^T y + 2X^T X \beta ,$$
and so $\frac{d}{d\beta} SSR(\beta) = 0 \iff -2X^T y + 2X^T X \beta = 0$

$$\Leftrightarrow X^T X \beta = X^T y \Leftrightarrow \beta = (X^T X)^{-1} X^T y .$$

 $\hat{\boldsymbol{\beta}}_{ols} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y$ is the Ordinary Least Squares (OLS) estimator of $\boldsymbol{\beta}$.

Linear regression: non-Bayes methods in R

```
### OLS estimate
> beta.ols <- solve( t(X)%*%X )%*%t(X)%*%y
> beta.ols
                   [,1]
(Intercept) -0.06821632
           2.94485495
xg
           2.84420729
xa
            1.72947648
xg:xa
### or less painfully, using lm() and R's formula syntax:
> fit.ols<-lm( y~ xg*xa )</pre>
> coef( summary(fit.ols) )
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.06821632 1.4222970 -0.04796208 9.623401e-01
            2.94485495 2.0114316 1.46405917 1.625482e-01
xg
            2.84420729 0.4288387 6.63234803 5.760923e-06
xa
            1.72947648 0.6064695 2.85171239 1.154001e-02
xg:xa
```

Linear regression: non-Bayes methods in R



EstimateStd. Errort valuePr(>|t|)(Intercept)-0.068216321.4222970-0.047962089.623401e-01xg2.944854952.01143161.464059171.625482e-01xa2.844207290.42883876.632348035.760923e-06xg:xa1.729476480.60646952.851712391.154001e-02

(The classic version estimates the standard error of each $\hat{\beta}_j$, and gives *p*-values indicating how significantly they differ from zero. t_j denotes $\hat{\beta}_j$ /Est.StdErr[$\hat{\beta}_j$], which has a *t* distribution if $\beta_j=0$.)

4.14

Multivariate Normal distribution

Before Bayesian linear regression, a final piece of math notation.

The *multivariate Normal* distribution describes random vectors. For mean m and variance matrix^{*} V, it is written mvn(m, V) – see right for an example:

So we can write a full linear regression model as just

$$\boldsymbol{y}|\mathbf{X},\boldsymbol{\beta},\sigma^2\sim \mathsf{mvn}(\mathbf{X}\boldsymbol{\beta},\sigma^2\boldsymbol{I}),$$

elements $Cov[Y_i, Y_i]$



where σ^2 is the (assumed-constant) variance of each observation around its mean, and I denotes an *identity* matrix, with 1s on the diagonal and 0 elsewhere. * For random vector Y_i , variance matrix Var[Y] has diagonal elements Var[Y_i] and off diagonal

Bayesian inference for linear regression models

With σ^2 known, Normal priors for β are conjugate. With

- Posterior precision (inverse-variance) is the sum of precision from prior and sampling model
- Posterior mean is a kind of weighted average of prior mean on $\hat{\beta}_{OLS}$, with precision/inverse-variance weights
- If $\Sigma_0^{-1} \ll \mathbf{X}^T \mathbf{X} / \sigma^2$, then $\boldsymbol{\beta}_n \approx \hat{\boldsymbol{\beta}}_{ols}$, if $\Sigma_0^{-1} \gg \mathbf{X}^T \mathbf{X} / \sigma^2$, then $\boldsymbol{\beta}_n \approx \boldsymbol{\beta}_0$ sensibly

The *g*-prior

But how to pick β_0, Σ_0 ? A classical suggestion (Zellner, 1986) uses the *g*-prior:

$$\boldsymbol{\beta} \sim \text{mvn}(\mathbf{0}, g\sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}).$$

Why this variance? The variance of the OLS estimate $\hat{\beta}_{ols}$ is

$$\operatorname{Var}\widehat{\boldsymbol{\beta}}_{\mathsf{ols}} = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1} = \frac{\sigma^2}{n} (\mathbf{X}^T \mathbf{X}/n)^{-1},$$

which can be thought of as (roughly) the uncertainty in β from *n* observations. In the *g*-prior the variance is

$$g\sigma^2 (\mathbf{X}^T \mathbf{X})^{-1} = \frac{\sigma^2}{n/g} (\mathbf{X}^T \mathbf{X}/n)^{-1},$$

which can be viewed as the uncertainty from n/g observations.

For example, g = n means the prior has the same amount of info as 1 observation – so (roughly!) not much, in a large study.

Posterior distributions under the *g*-prior

Following the conjugacy results, with σ^2 known it turns out that

$$\boldsymbol{\beta}|\boldsymbol{y}, \boldsymbol{X}, \sigma^2 \sim \operatorname{mvn}(\boldsymbol{\beta}_n, \boldsymbol{\Sigma}_n),$$

where
$$\Sigma_n = \operatorname{Var}[\boldsymbol{\beta}|\boldsymbol{y}, \mathbf{X}, \sigma^2] = \frac{g}{g+1} \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}$$

 $\boldsymbol{\beta}_n = \mathbb{E}[\boldsymbol{\beta}|\boldsymbol{y}, \mathbf{X}, \sigma^2] = \frac{g}{g+1} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{y}$

- The posterior mean estimate β_n is simply $\frac{g}{q+1}\hat{\beta}_{ols}$.
- The posterior variance of β is simply $\frac{g}{g+1}$ Var $\hat{\beta}_{ols}$.
- $\beta_n^T \Sigma_n^{-1} \beta_n$ is $\frac{g}{g+1}$ times smaller than $\hat{\beta}_{OLS}^T \text{Var}[\hat{\beta}_{OLS}]^{-1} \hat{\beta}_{OLS}$, the usual signal-to-noise measure we turn into *p*-values
- g shrinks the coefficients towards 0 and so can prevent overfitting to the data
- If $g \propto n$, then as n increases, inference approximates classical methods.

But σ^2 is rarely known exactly, so more realistic analysis captures that uncertainty via a prior. A convenient & popular choice is the *inverse-Gamma* distribution:

prior $1/\sigma^2 \sim \text{gamma}(\nu_0/2, \nu_0 \sigma_0^2/2)$ sampling model $y \sim \text{mvn}(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I})$ posterior $1/\sigma^2 | y, \mathbf{X} \sim \text{gamma}([\nu_0 + n]/2, [\nu_0 \sigma_0^2 + \text{SSR}_g]/2)$

...where SSR_q is complicated but basically a sum of squared residuals.

A natural next step is be to use the conjugate posteriors for σ and β together – usually to get the marginal posterior for β . But this requires integration:

$$p(\boldsymbol{\beta}|\boldsymbol{y},\boldsymbol{X}) = \int_{\sigma^2 > 0} p(\boldsymbol{\beta},\sigma^2|\boldsymbol{y},\boldsymbol{X}) d\sigma^2 = \int_{\sigma^2 > 0} p(\boldsymbol{\beta}|\sigma^2,\boldsymbol{y},\boldsymbol{X}) p(\sigma^2|\boldsymbol{y},\boldsymbol{X}) d\sigma^2$$

This takes work, and is error-prone. Instead we'll use Monte Carlo methods.

Monte Carlo simulation



Large sample (points) to estimate posterior (contours)

- we can approximate **any** property of that distribution.

Monte Carlo simulation

It may *seem* like sampling would require full knowledge of the posterior, and be at least as much work as the integration.

But this isn't true: one way to do it is the *Gibbs Sampler* – here shown for two examples



Gibbs updates parameters 'one at a time', using only $p(\theta_1|\theta_2)$, then $p(\theta_2|\theta_1)$ – so no hard integration to do. The sequence of samples $\theta^{(s)}$ (a *Markov Chain*) are autocorrelated, i.e. dependent, but the posterior **is** covered appropriately.

Monte Carlo simulation

Recall the posterior can be written as $p(\theta_1, \theta_2 | \mathbf{Y}) = p(\theta_2 | \mathbf{Y}) \times p(\theta_1 | \theta_2, \mathbf{Y})$, so if we can generate from conditional $p(\theta_1 | \theta_2, \mathbf{Y})$ – in one variable – with frequency $p(\theta_2 | \mathbf{Y})$ for each θ_2 , we get a sample from the full posterior.

Gibbs does this, using *just* the conditionals, iterating between these steps:

$$\begin{array}{ll} \theta_1^{(s)} &\sim p(\theta_1 | \theta_2^{(s-1)}, \mathbf{Y}) \\ \theta_2^{(s)} &\sim p(\theta_2 | \theta_1^{(s)}, \mathbf{Y}) \end{array} \end{array}$$

to produce a *chain* $(\theta_1^{(0)}, \theta_2^{(0)}), (\theta_1^{(1)}, \theta_2^{(1)}), ..., (\theta_1^{(s)}, \theta_2^{(s)}), ...$

- If the chain is long enough $(s \to \infty)$, this sequence is a sample from $p(\theta_1, \theta_2 | \mathbf{Y})$, no matter where you started
- For more parameters, update each single θ_k in turn, then start again

Stan works out all the conditionals for us, and how to update based on them ;



- Specify just a model, including priors, and tell Stan what the data are
- Stan writes code to sample from the posterior, by 'walking around' actually it runs the *No U-Turn Sampler*, a more advanced algorithm for exploring the parameter space
- Stan runs this code, and reports back all the samples
- The rstan package lets you run chains from R
- Some modeling limitations no discrete parameters but becoming very popular; works well with some models where other software would struggle
- Requires declarations (like C++) unlike R so models require a bit more typing...

```
For our linear regression, with unknown \sigma^2;
```

```
cat(file="FTOexample.stan", "
data {
  int n: //the number of observations
  int p; //the number of columns in the model matrix
  real y[n]; //the response
  matrix[n,p] X; //the model matrix
  real g; // Zellner scale factor
  vector[p] mu; // Zellner prior mean (all zeros)
  matrix[p,p] XtXinv; // information matrix
}
parameters {
  vector[p] beta; //the regression parameters
  real invsigma2; //the precision, a.k.a. inverse-variance
}
transformed parameters {
  vector[n] linpred;
  cov_matrix[p] Sigma;
```

```
real sigma;
  linpred = X*beta;
  sigma = 1/sqrt(invsigma2);
  for (j in 1:p){
    for (k in 1:p){
      Sigma[j,k] = g*sigma^2*XtXinv[j,k];
      model {
  beta ~ multi_normal(mu, Sigma);
  y ~ normal(linpred, sigma);
  invsigma2 ~ gamma(0.5, 1.839); // we took nu0=1, sigma0=1.91
}
")
# do the MCMC, store the results
library("rstan")
stan2 <- stan(file = "FTOexample.stan",</pre>
data = list(n=n,p=p, y=y, X=X, g=n, mu=rep(0,p), XtXinv=solve(crossprod(X)) ),
iter = 100000, chains = 1, pars=c("beta","sigma"))
```

Summarize the posterior by summarizing the MCMC samples:

```
> print(stan2)
Inference for Stan model: FTOexample.
1 chains, each with iter=1e+05; warmup=50000; thin=1;
post-warmup draws per chain=50000, total post-warmup draws=50000.
                      sd 2.5%
                                        50% 75% 97.5% n_eff Rhat
                                  25%
         mean se_mean
        0.50
                0.02 2.47 -4.39 -1.11 0.50 2.10 5.39 13849
beta[1]
                                                                 1
beta[2] 1.48 0.03 3.50 -5.42 -0.80 1.47 3.78 8.43 13720
                                                                 1
                                             2.61 3.61 14000
beta[3] 2.13 0.01 0.75 0.65 1.64 2.13
                                                                 1
beta[4] 2.56 0.01 1.06 0.46 1.87 2.56 3.24 4.65 14015
                                                                 1
      3.39 0.00 0.55 2.52 3.00 3.32 3.70 4.66 20814
sigma
                                                                 1
                0.01 1.64 -45.04 -41.79 -40.61 -39.75 -38.79 16583
       -40.95
lp__
                                                                 1
> # compare with the non-Bayes version
> round(cbind(summary(lm(y<sup>x</sup>g*xa))$coef[,1:2], confint(lm(y<sup>x</sup>g*xa))), 2)
          Estimate Std. Error 2.5 % 97.5 %
(Intercept)
                        1.06 - 1.75 2.73
              0.49
              1.59
                        1.49 - 1.57 4.76
xg
                        0.32 1.57 2.92
             2.24
xa
              2.68
                        0.45 1.72 3.63
xg:xa
```



The 'best guess' estimate is $\hat{\sigma} = \sigma_0 = 1.91 - but$ the prior also supports much larger values – with which the data don't strongly disagree.

To see where the 'chain' went...

> traceplot(stan2)

Tip: save these plots as PNGs not PDFs!



With the posterior samples in R, we can evaluate the posterior of any function of them. For example, the genetic effect at each of the five ages;

 $\mathbb{E}[Y|\mathsf{age}, +/-] - \mathbb{E}[Y|\mathsf{age}, -/-] = (\beta_1 + \beta_2) + (\beta_3 + \beta_4) \times \mathsf{age} - (\beta_1 + \beta_3 \times \mathsf{age}) = \beta_2 + \beta_4 \times \mathsf{age}$



MCMC: summary so far

- Stan and other similar software may look like overkill for this 'conjugate' problem, but Stan can provide posteriors for almost any model
- The 'modeling' language is based on R
- Users do have to decide how long a chain to run, and how long to 'burn in' for at the start of the chain. These are not easy to answer! We'll see some diagnostics in later sessions

Summary

- Linear regressions are of great applied interest
- Conjugacy helps us quickly explore how informative the prior is, versus the data
- ...but with MCMC we can fit a huge variety of models
- Inference using MCMC output is also flexible; no restriction on parameters we can learn about

Bonus tracks: what if the model's wrong? (***)

Different types of violation—in decreasing order of how much they typically matter in practice

- Just have the wrong data (!) i.e. not the data you claim to have
- Observations are not independent, e.g. repeated measures on same mouse over time
- Mean model is incorrect
- Error terms do not have constant variance
- Error terms are not Normally distributed

Having the wrong data, where possible, should be addressed before analyses. Here we'll focus on how much post-analysis violations matter.

Wrong model: dependent outcomes

- Observations from the same mouse are more likely to be similar than those from different mice (even if they have same age and genotype)
- SBP from subjects (even with same age, genotype etc) in the same family are more likely to be similar than those in different familes perhaps unmeasured common diet?
- Spatial and temporal relationships also tend to induce correlation

If the pattern of relationship is known, can allow for it – typically in "random effects modes" – see later session.

If not, treat results with caution! Precision is likely over-stated.

Even when the scientific background is highly informative about the variables of interest (e.g. we want to know about the association of Y with x_1 , adjusting for $x_2, x_3...$) there is rarely strong information about the form of the model

- Does mean weight increase with age? age²? age³?
- Could the effect of genotype also change non-linearly with age?

Including quadratic terms is a common approach – but quadratics are sensitive to the tails. Instead, including "spline" representations of covariates allows the model to capture many patterns.

Including interaction terms (as we did with $x_{i,2} \times x_{i,3}$) lets one covariate's effect vary with another.

(Deciding which covariates to use is addressed in the Model Choice session.)

Wrong model: non-constant variance

This is plausible in many situations; perhaps e.g. young mice are harder to measure, i.e. more variables. Or perhaps the FTO variant affects weight regulation — again, more variance.

- Having different variances at different covariate values is known as *het-eroskedasticity*
- Unaddressed, it can result in over- or under-statement of precision

The most obvious approach is to model the variance, i.e.

$$Y_i = \boldsymbol{\beta}^T \boldsymbol{x}_i + \boldsymbol{\epsilon}_i, \\ \boldsymbol{\epsilon}_i \sim \text{Normal}(0, \sigma_i^2),$$

where σ_i depends on covariates, e.g. σ_{homozy} and $\sigma_{heterozy}$ for the two genotypes.

Of course, these parameters need priors. Constraining variances to be positive also makes choosing a model difficult in practice.

Robust standard errors (in Bayes)

In linear regression, some robustness to model-misspecification and/or nonconstant variance is available – but it relies on interest in linear 'trends'. Formally, we can define parameter

$$\boldsymbol{\theta} = \operatorname{argmin} \mathbb{E}_x \left[\left(\mathbb{E}_y[y|x] - \mathbf{X}^t \boldsymbol{\theta} \right)^2 \right],$$

i.e. the straight line that best-captures random-sampling, in a least-squares sense.

- This 'trend' can capture important features in how the mean y varies at different x
- Fitting extremely flexible Bayesian models, we get a posterior for $\boldsymbol{\theta}$
- The posterior mean approaches $\hat{\beta}_{ols}$, in large samples
- The posterior variance approaches the 'robust' *sandwich estimate*, in large samples (details in Szpiro et al, 2011)

Robust standard errors (in Bayes)

The OLS estimator can be written as $\hat{\beta}_{ols} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \sum_{i=1}^n c_i y_i$, for appropriate c_i .

True variance $\operatorname{Var}[\widehat{\beta}] = \sum_{i=1}^{n} c_i^2 \operatorname{Var}[Y_i]$ Robust estimate $\widehat{\operatorname{Var}}_R[\widehat{\beta}] = \sum_{i=1}^{n} c_i^2 e_i^2$ Model-based estimate $\widehat{\operatorname{Var}}_M[\widehat{\beta}] = \operatorname{Mean}(e_i^2) \sum_{i=1}^{n} c_i^2$,

where $e_i = y_i - x_i^T \hat{\beta}_{ols}$, the residuals from fitting a linear model.

Non-Bayesian sandwich estimates are available through R's sandwich package – much quicker than Bayes with a very-flexible model. For correlated outcomes, see the GEE package for generalizations.

This is not a big problem for learning about population parameters;

- The variance statements/estimates we just saw don't rely on Normality
- The *central limit theorem* means that $\hat{\beta}$ ends up Normal anyway, in large samples
- In small samples, expect to have limited power to detect non-Normality
- ... except, perhaps, for extreme outliers (data errors?)

For prediction – where we assume that outcomes do follow a Normal distribution – this assumption is more important.