Linear Models

Lecture 02

Dr. Colin Rundel

Linear Models Basics

Pretty much everything we a going to see in this course will fall under the umbrella of either linear or generalized linear models.

In previous classes most of your time has likely been spent with the simple iid case,

$$y_i = \beta_0 + \beta_1\, x_{i1} + \cdots + \beta_p\, x_{ip} + \epsilon_i$$

$$\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$$

these models can also be expressed simply as,

$$y_i \overset{iid}{\sim} N(\beta_0 + \beta_1\, x_{i1} + \cdots + \beta_p\, x_{ip},\ \sigma^2)$$

Some notes on notation

- Observed values and scalars will usually be lower case letters, e.g. x_i, y_i, z_{ij}.
- Parameters will usually be greek symbols, e.g. μ, σ, ρ.
- Vectors and matrices will be shown in bold, e.g. $\boldsymbol{\mu}$, $\boldsymbol{X}$, $\boldsymbol{\Sigma}$.
- Elements of a matrix (or vector) will be referenced with {}s, e.g. $\{\boldsymbol{Y}\}_i, \{\boldsymbol{\Sigma}\}_{ij}$
- Random variables will be indicated by ~, e.g. $x \sim \text{Norm}(0, 1), z \sim \text{Gamma}(1, 1)$
- Matrix / vector transposes will be indicated with $'$, e.g. $\boldsymbol{A}', (1 - \boldsymbol{B})'$

Linear model - matrix notation

We can also express a linear model using matrix notation as follows,

$$\underset{n\times 1}{\boldsymbol{Y}} = \underset{n\times p}{\boldsymbol{X}}\,\underset{p\times 1}{\boldsymbol{\beta}} + \underset{n\times 1}{\boldsymbol{\epsilon}}$$

$$\underset{n\times 1}{\boldsymbol{\epsilon}} \sim N(\underset{n\times 1}{\boldsymbol{0}},\; \sigma^2 \underset{n\times n}{\mathbb{1}_n})$$

or alternative as,

$$\underset{n\times 1}{\boldsymbol{Y}} \sim N\left(\underset{n\times p}{\boldsymbol{X}}\,\underset{p\times 1}{\boldsymbol{\beta}},\; \sigma^2 \underset{n\times n}{\mathbb{1}_n}\right)$$

Where possible I will include the dimensions of matrices and vectors as these provide a useful sanity check that the dimensions conform.

Multivariate Normal Distribution - Review

For an n-dimension multivate normal distribution with covariance $\boldsymbol{\Sigma}$ (positive semidefinite) can be written as

$$\underset{n\times 1}{\boldsymbol{Y}} \sim N(\underset{n\times 1}{\boldsymbol{\mu}}, \underset{n\times n}{\boldsymbol{\Sigma}})$$

where $\{\boldsymbol{\Sigma}\}_{ij} = \rho_{ij}\sigma_i\sigma_j$

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix}, \begin{pmatrix} \rho_{11}\sigma_1\sigma_1 & \rho_{12}\sigma_1\sigma_2 & \cdots & \rho_{1n}\sigma_1\sigma_n \\ \rho_{21}\sigma_2\sigma_1 & \rho_{22}\sigma_2\sigma_2 & \cdots & \rho_{2n}\sigma_2\sigma_n \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{n1}\sigma_n\sigma_1 & \rho_{n2}\sigma_n\sigma_2 & \cdots & \rho_{nn}\sigma_n\sigma_n \end{pmatrix} \right)$$

Multivariate Normal Distribution - Density

For the n dimensional multivate normal given on the last slide, its density is given by

$$f\big(\boldsymbol{Y}|\boldsymbol{\mu},\boldsymbol{\Sigma}\big) = (2\pi)^{-n/2}\;\det(\boldsymbol{\Sigma})^{-1/2}\;\exp\left(-\frac{1}{2}\underset{1\times n}{(\boldsymbol{Y}-\boldsymbol{\mu})'}\underset{n\times n}{\boldsymbol{\Sigma}^{-1}}\underset{n\times 1}{(\boldsymbol{Y}-\boldsymbol{\mu})}\right)$$

and its log density is given by

$$\log f\big(\boldsymbol{Y}|\boldsymbol{\mu},\boldsymbol{\Sigma}\big) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log\det(\boldsymbol{\Sigma}) - \frac{1}{2}\underset{1\times n}{(\boldsymbol{Y}-\boldsymbol{\mu})'}\underset{n\times n}{\boldsymbol{\Sigma}^{-1}}\underset{n\times 1}{(\boldsymbol{Y}-\boldsymbol{\mu})}$$

Some useful identities

The following come from the Matrix Cookbook Chapters 1 & 2.

$$
\begin{aligned}
(\boldsymbol{A}\boldsymbol{B})' &= \boldsymbol{B}'\boldsymbol{A}' \\
(\boldsymbol{A}+\boldsymbol{B})' &= \boldsymbol{A}'+\boldsymbol{B}' \\
(\boldsymbol{A}')^{-1} &= (\boldsymbol{A}^{-1})' \\
(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}\ldots)^{-1} &= \ldots\boldsymbol{C}^{-1}\boldsymbol{B}^{-1}\boldsymbol{A}^{-1}
\end{aligned}
$$

$$
\begin{aligned}
\det(\boldsymbol{A}') &= \det(\boldsymbol{A}) \\
\det(\boldsymbol{A}\boldsymbol{B}) &= \det(\boldsymbol{A})\det(\boldsymbol{B}) \\
\det(c\boldsymbol{A}) &= c^n\det(\boldsymbol{A}) \\
\det(\boldsymbol{A}^n) &= \det(\boldsymbol{A})^n
\end{aligned}
$$

$$
\begin{aligned}
\partial\boldsymbol{A} &= 0 && (\text{where }\boldsymbol{A}\text{ is} \\
\partial(a\boldsymbol{X}) &= a(\partial\boldsymbol{A}) \\
\partial(\boldsymbol{X}+\boldsymbol{Y}) &= \partial\boldsymbol{X}+\partial\boldsymbol{Y} \\
\partial(\boldsymbol{X}\boldsymbol{Y}) &= (\partial\boldsymbol{X})\boldsymbol{Y}+\boldsymbol{X}(\partial\boldsymbol{Y}) \\
\partial(\boldsymbol{X}') &= (\partial\boldsymbol{X})'
\end{aligned}
$$

Maximum Likelihood - β (iid)

Maximum Likelihood - σ^2 (iid)

A Quick Example

Parameters -> Synthetic Data

Lets generate some simulated data where the underlying model is known and see how various regression procedures function.

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \epsilon_i$$

$$\beta_0 = 0.7, \quad \beta_1 = 1.5, \quad \beta_2 = -2.2, \quad \beta_3 = 0.1$$

$$\epsilon_i \sim N(0,1)$$

```r
set.seed(1234)
n = 100
beta = c(0.7, 1.5, -2.2, 0.1)
sigma = 1
eps = rnorm(n, 0, sd = sigma)

d = data_frame(
  X1 = rt(n,df=5),
  X2 = rt(n,df=5),
```

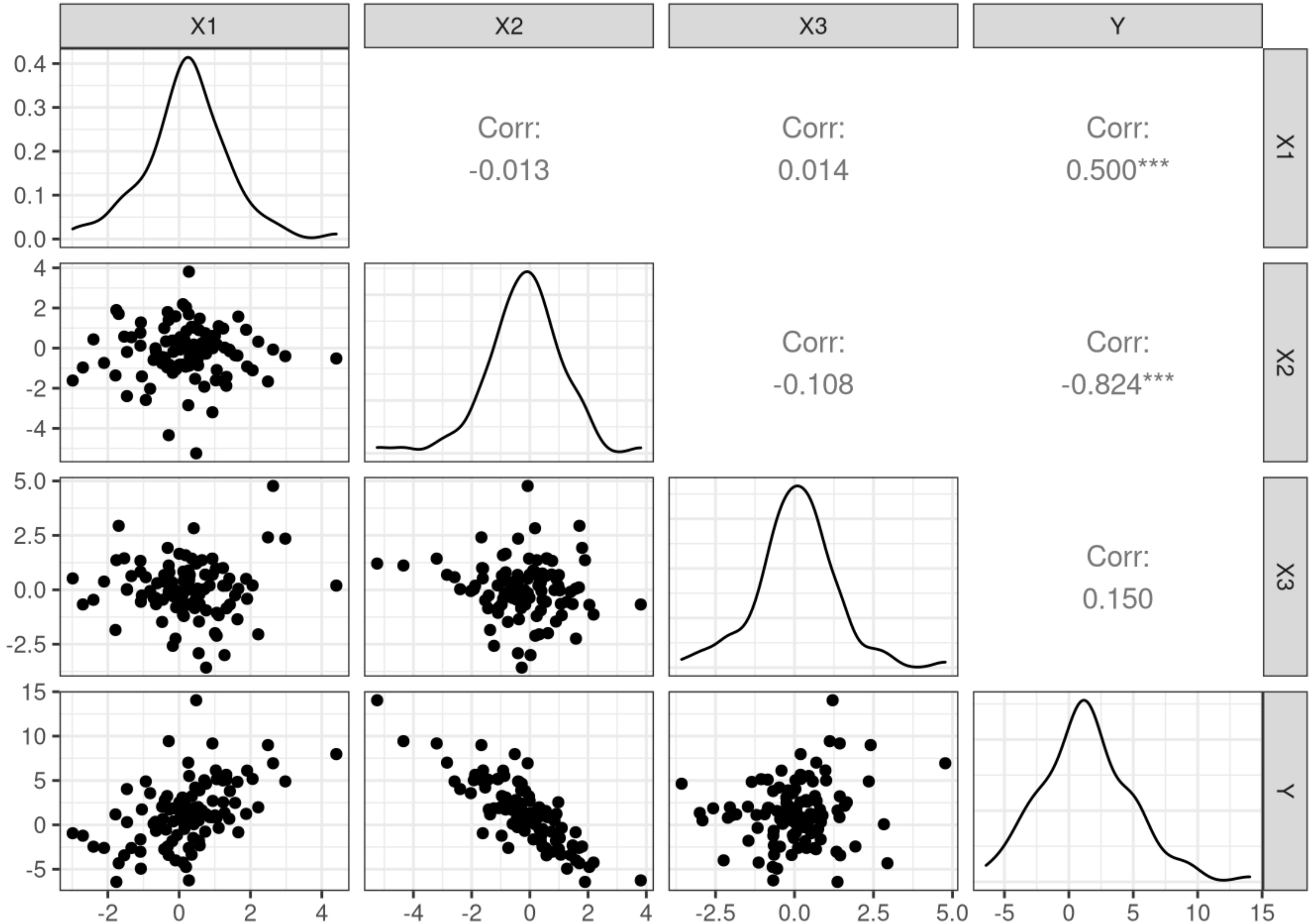
Model Matrix

```r
X = model.matrix(~X1+X2+X3, d)
tbl_df(X)
```

```
# A tibble: 100 × 4
   `(Intercept)`      X1       X2      X3
           <dbl>   <dbl>    <dbl>   <dbl>
 1             1   0.557   0.897   -1.46
 2             1   0.758   0.375   -0.945
 3             1   0.273   3.81    -0.675
 4             1   1.41   -0.0745   0.514
 5             1   1.01    0.623   -1.99
 6             1   0.942  -0.00618  0.700
 7             1   1.66    1.57     0.0478
 8             1  -1.09    0.766    1.33
 9             1  -0.296   1.40    -0.0914
```

Pairs plot

```r
GGally::ggpairs(d, progress = FALSE)
```



Least squares fit

Let $\hat{\boldsymbol{Y}}$ be our estimate for $\boldsymbol{Y}$ based on our estimate of $\boldsymbol{\beta}$,

$$\hat{\boldsymbol{Y}} = \hat{\beta}_0 + \hat{\beta}_1\,\boldsymbol{X}_1 + \hat{\beta}_2\,\boldsymbol{X}_2 + \hat{\beta}_3\,\boldsymbol{X}_3 = \boldsymbol{X}\,\hat{\boldsymbol{\beta}}$$

The least squares estimate, $\hat{\boldsymbol{\beta}}_{ls}$, is given by

$$\underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} (Y_i - \boldsymbol{X}_{i\cdot}\boldsymbol{\beta})^2$$

Previously we showed that,

$$\hat{\boldsymbol{\beta}}_{ls} = (\boldsymbol{X}'\,\boldsymbol{X})^{-1}\,\boldsymbol{X}'\,\boldsymbol{Y}$$

Frequentist Fit

```
(beta_hat = solve(t(X) %*% X, t(X)) %*% d$Y)
```

```
                  [,1]
(Intercept)  0.5522298
X1           1.4708769
X2          -2.1761159
X3           0.1535830
```

```
l = lm(Y ~ X1 + X2 + X3, data=d)
l$coefficients
```

```
(Intercept)          X1          X2          X3 
  0.5522298   1.4708769  -2.1761159   0.1535830 
```

Bayesian regression model

Basics of Bayes

brms

We will be using a package called brms for most of our Bayesian model fitting

- it has a convenient model specification syntax
- mostly sensible prior defaults
- supports most of the model types we will be exploring
- uses Stan behind the scenes

brms + linear regression

```r
b = brms::brm(Y ~ X1 + X2 + X3, data=d, chains = 2)
```

```
Running /usr/lib64/R/bin/R CMD SHLIB foo.c
gcc -m64 -I"/usr/include/R" -DNDEBUG   -
I"/usr/lib64/R/library/Rcpp/include/"  -
I"/usr/lib64/R/library/RcppEigen/include/"  -
I"/usr/lib64/R/library/RcppEigen/include/unsupported"  -
I"/usr/lib64/R/library/BH/include" -
I"/usr/lib64/R/library/StanHeaders/include/src/"  -
I"/usr/lib64/R/library/StanHeaders/include/"  -
I"/usr/lib64/R/library/RcppParallel/include/"  -
I"/usr/lib64/R/library/rstan/include" -DEIGEN_NO_DEBUG  -
DBOOST_DISABLE_ASSERTS  -DBOOST_PENDING_INTEGER_LOG2_HPP  -DSTAN_THREADS  -
DBOOST_NO_AUTO_PTR  -include
```

Model results

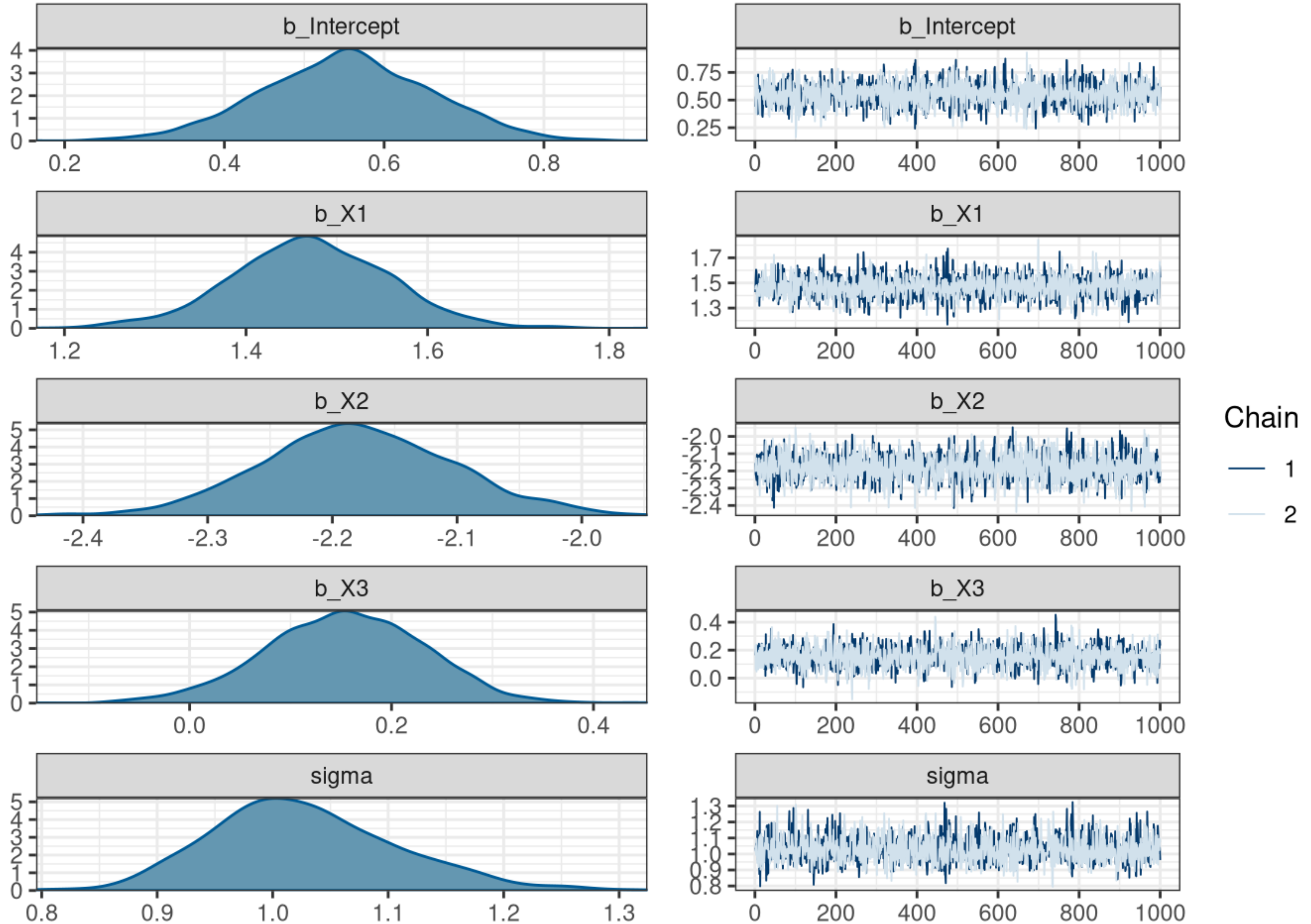
```
b
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: Y ~ X1 + X2 + X3
   Data: d (Number of observations: 100)
  Draws: 1 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 1000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept     0.55      0.11     0.34     0.76 1.00     1269      627
X1            1.47      0.09     1.29     1.64 1.00     1273      723
X2           -2.18      0.08    -2.32    -2.02 1.00     1178      663
```

Model visual summary

```r
plot(b)
```



What about the priors?

```r
brms::prior_summary(b)
```

```
                 prior     class
coef group resp dpar nlpar lb ub
source
                (flat)         b
default
                (flat)         b
X1
(vectorized)
                (flat)         b
X2
(vectorized)
                (flat)         b
```

```r
brms::get_prior("Y ~ X1 + X2 + X3
```

```
                 prior     class
coef group resp dpar nlpar lb ub
source
                (flat)         b
default
                (flat)         b
X1
(vectorized)
                (flat)         b
X2
(vectorized)
                (flat)         b
```

tidybayes

```r
(post = b %>%
  tidybayes::gather_draws(b_Intercept, b_X1, b_X2, b_X3, sigma)
)
```

```
# A tibble: 10,000 × 5
# Groups:   .variable [5]
   .chain .iteration .draw .variable   .value
    <int>      <int> <int> <chr>        <dbl>
 1      1          1     1 b_Intercept  0.440
 2      1          2     2 b_Intercept  0.524
 3      1          3     3 b_Intercept  0.544
 4      1          4     4 b_Intercept  0.502
 5      1          5     5 b_Intercept  0.567
 6      1          6     6 b_Intercept  0.569
 7      1          7     7 b_Intercept  0.373
 8      1          8     8 b_Intercept  0.731
```

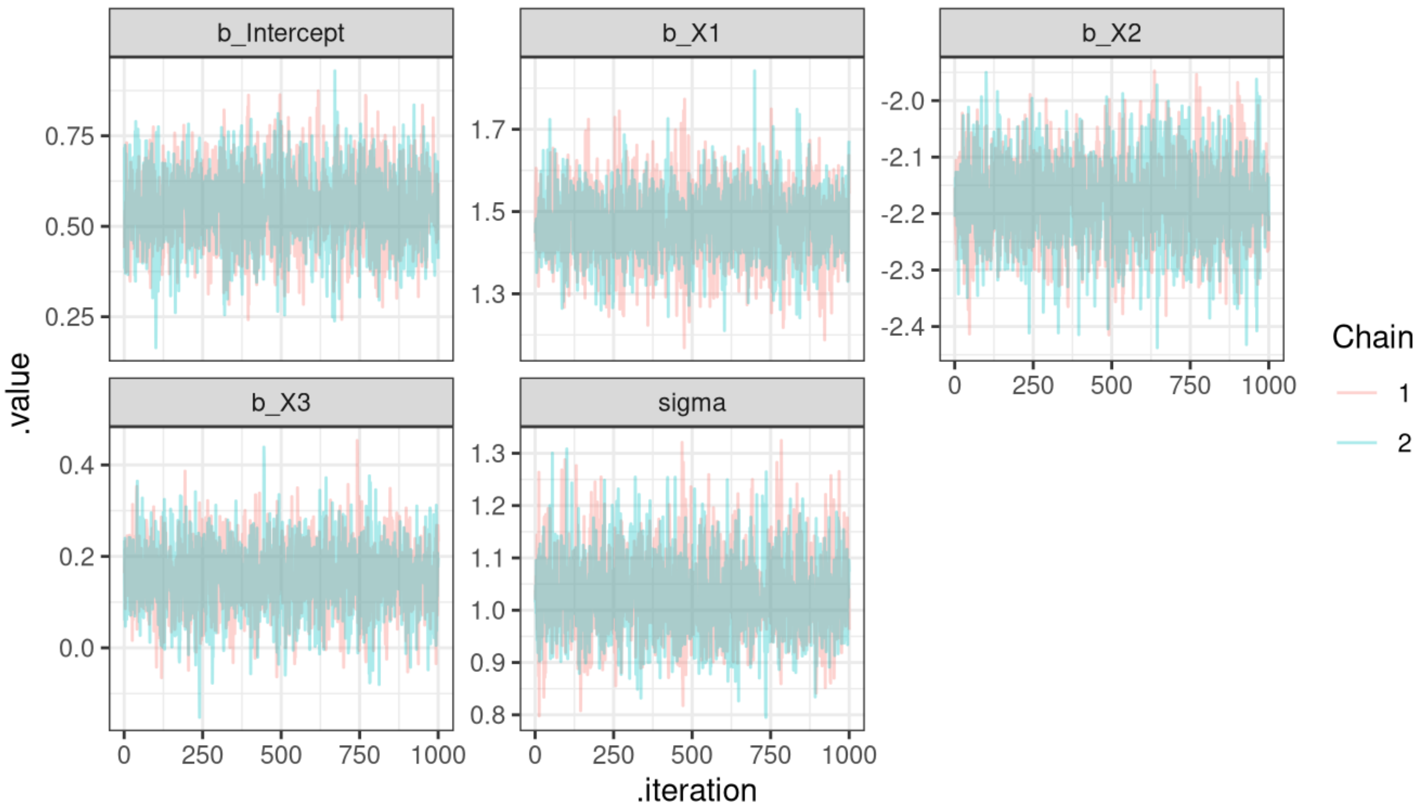
tidybayes + posterior summaries

```r
(post_sum = post %>%
  group_by(.variable, .chain) %>%
  summarize(
    post_mean = mean(.value),
    post_median = median(.value)
  )
)
```

```
# A tibble: 10 × 4
# Groups:   .variable [5]
   .variable   .chain post_mean post_median
   <chr>        <int>     <dbl>       <dbl>
 1 b_Intercept      1     0.556       0.555
 2 b_Intercept      2     0.550       0.553
 3 b_X1             1     1.47        1.47
 4 b_X1             2     1.47        1.47
 5 b_X2             1    -2.18       -2.18
 6 b_X2             2    -2.18       -2.18
 7 b_X3             1     0.157       0.157
```

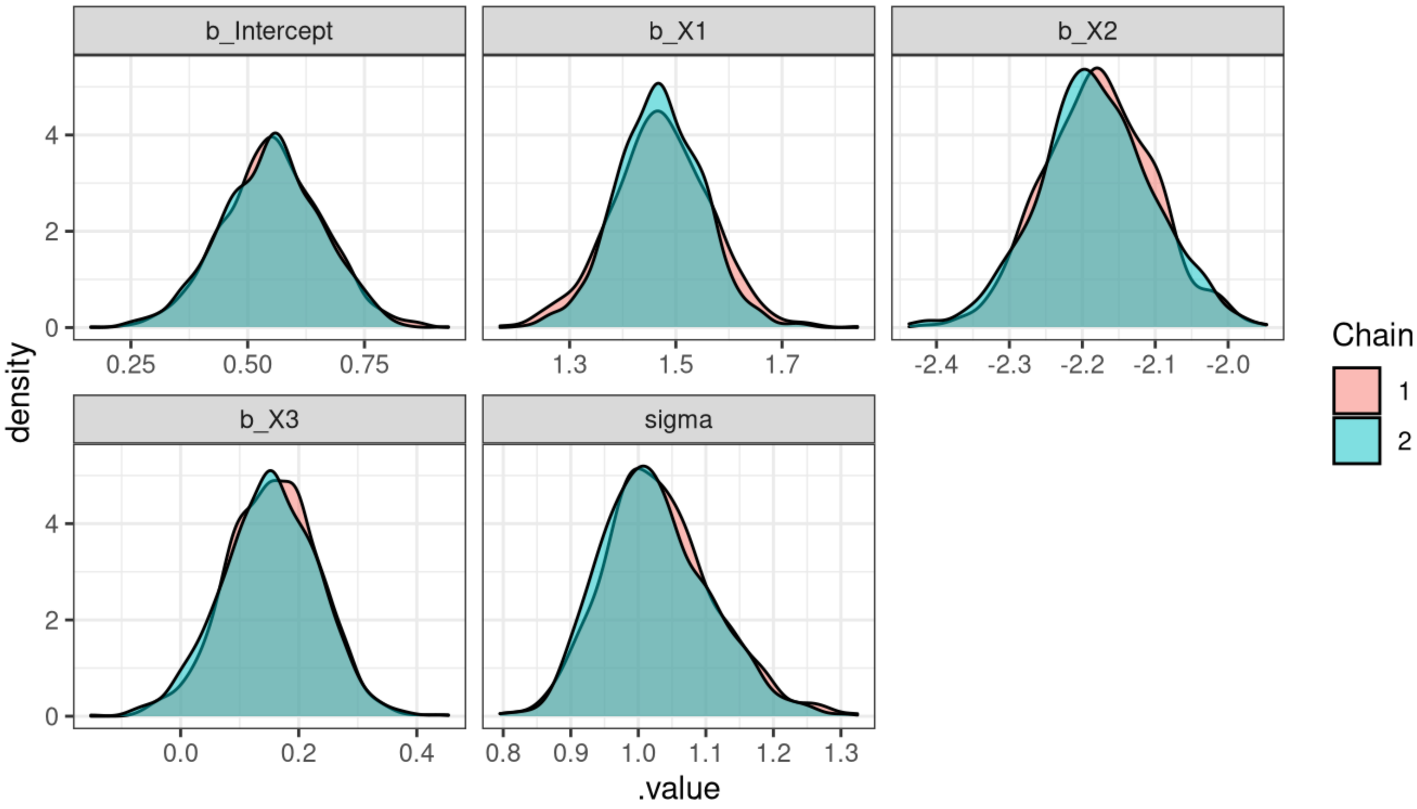
tidybayes + ggplot - traceplot

```
post %>%
  ggplot(aes(x=.iteration, y=.value, color=as.character(.chain))) +
  geom_line(alpha=0.33) +
  facet_wrap(~.variable, scale="free_y") +
  labs(color="Chain")
```



Tidy Bayes + ggplot - Density plot

```
post %>%
  ggplot(aes(x=.value, fill=as.character(.chain))) +
  geom_density(alpha=0.5) +
  facet_wrap(~.variable, scale="free_x") +
  labs(fill="Chain")
```



Comparing Approaches

```r
(pt_est = post_sum %>%
  filter(.chain == 1) %>%
  ungroup() %>%
  mutate(
    truth = c(beta, sigma),
    ols   = c(l$coefficients, sd(l$residuals))
  ) %>%
  select(.variable, truth, ols, post_mean)
)
```

```
# A tibble: 5 × 4
  .variable   truth    ols post_mean
  <chr>       <dbl>  <dbl>     <dbl>
1 b_Intercept   0.7  0.552     0.556
2 b_X1          1.5  1.47      1.47 
3 b_X2         -2.2 -2.18     -2.18 
4 b_X3          0.1  0.154     0.157
5 sigma         1    1.00      1.03 
```

Comparing Approaches - code

```
post %>%
  filter(.chain == 1) %>%
  ggplot(aes(x=.value)) +
  geom_density(alpha=0.5, fill="lightblue") +
  facet_wrap(~.variable, scale="free_x") +
  geom_vline(
    data = pt_est %>% tidyr::pivot_longer(cols = truth:post_mean, names_
    aes(xintercept = value, color=pt_est),
    alpha = 0.5, size=2
  )
```

Comparing Approaches - plot