

# Diagnostics and Model Evaluation

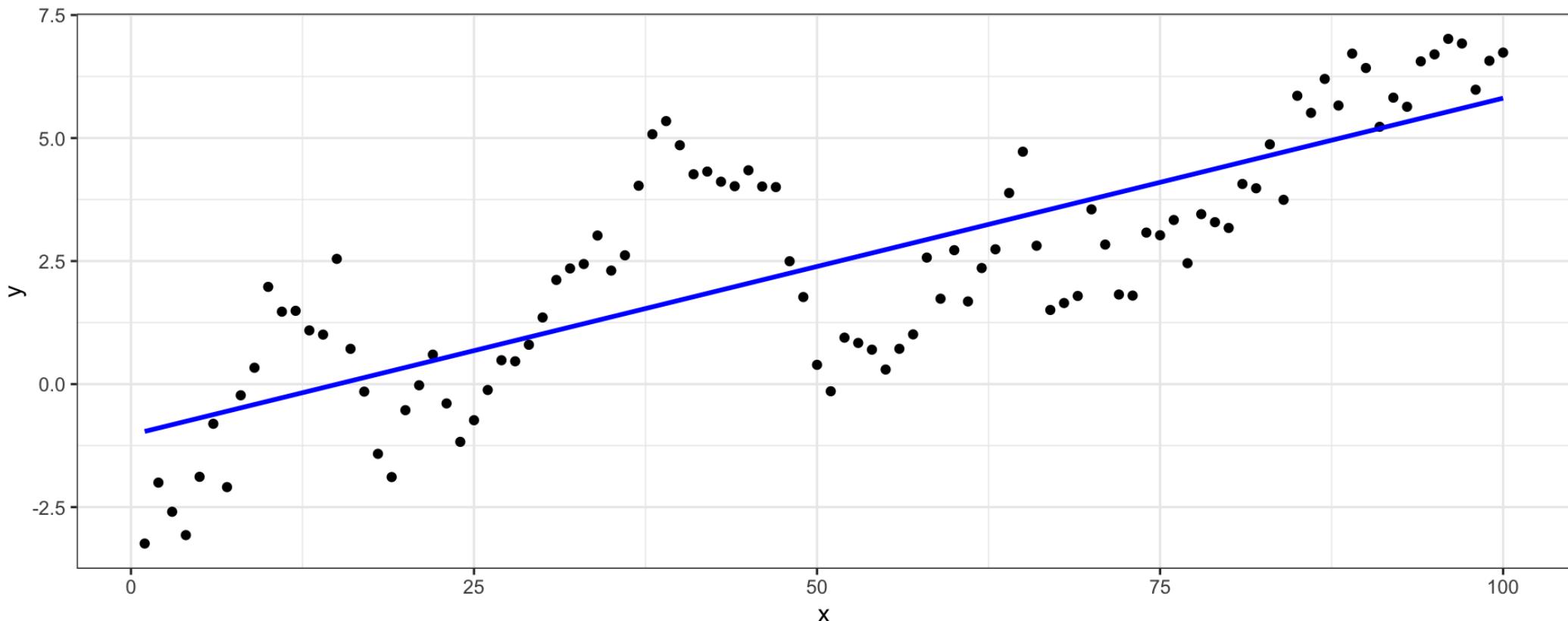
Lecture 03

Dr. Colin Rundel

# Some more linear models

# Linear model and data

```
1 ggplot(d, aes(x=x,y=y)) +  
2   geom_point() +  
3   geom_smooth(method="lm", color="blue", se = FALSE)
```



# Linear model

```
1 l = lm(y ~ x, data=d)
2 summary(l)
```

Call:

```
lm(formula = y ~ x, data = d)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.6041	-1.2142	-0.1973	1.1969	3.7072

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )							
(Intercept)	-1.030315	0.310326	-3.32	0.00126 **							
x	0.068409	0.005335	12.82	< 2e-16 ***							
---											
Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'..'	0.1	' '	1

Residual standard error: 1.54 on 98 degrees of freedom

Multiple R-squared: 0.6266, Adjusted R-squared: 0.6227

F-statistic: 164.4 on 1 and 98 DF, p-value: < 2.2e-16

# Bayesian model (brms)

```
1 ( b = brms:::brm(  
2     y ~ x, data=d,  
3     prior = c(  
4         brms:::prior(normal(0, 100), class = Intercept),  
5         brms:::prior(normal(0, 10),  class = b),  
6         brms:::prior(cauchy(0, 2),   class = sigma)  
7     ),  
8     silent = 2, refresh = 0  
9 )  
10 )
```

# Bayesian model (brms)

Family: gaussian

Links: mu = identity; sigma = identity

Formula: y ~ x

Data: d (Number of observations: 100)

Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;  
total post-warmup draws = 4000

Population-Level Effects:

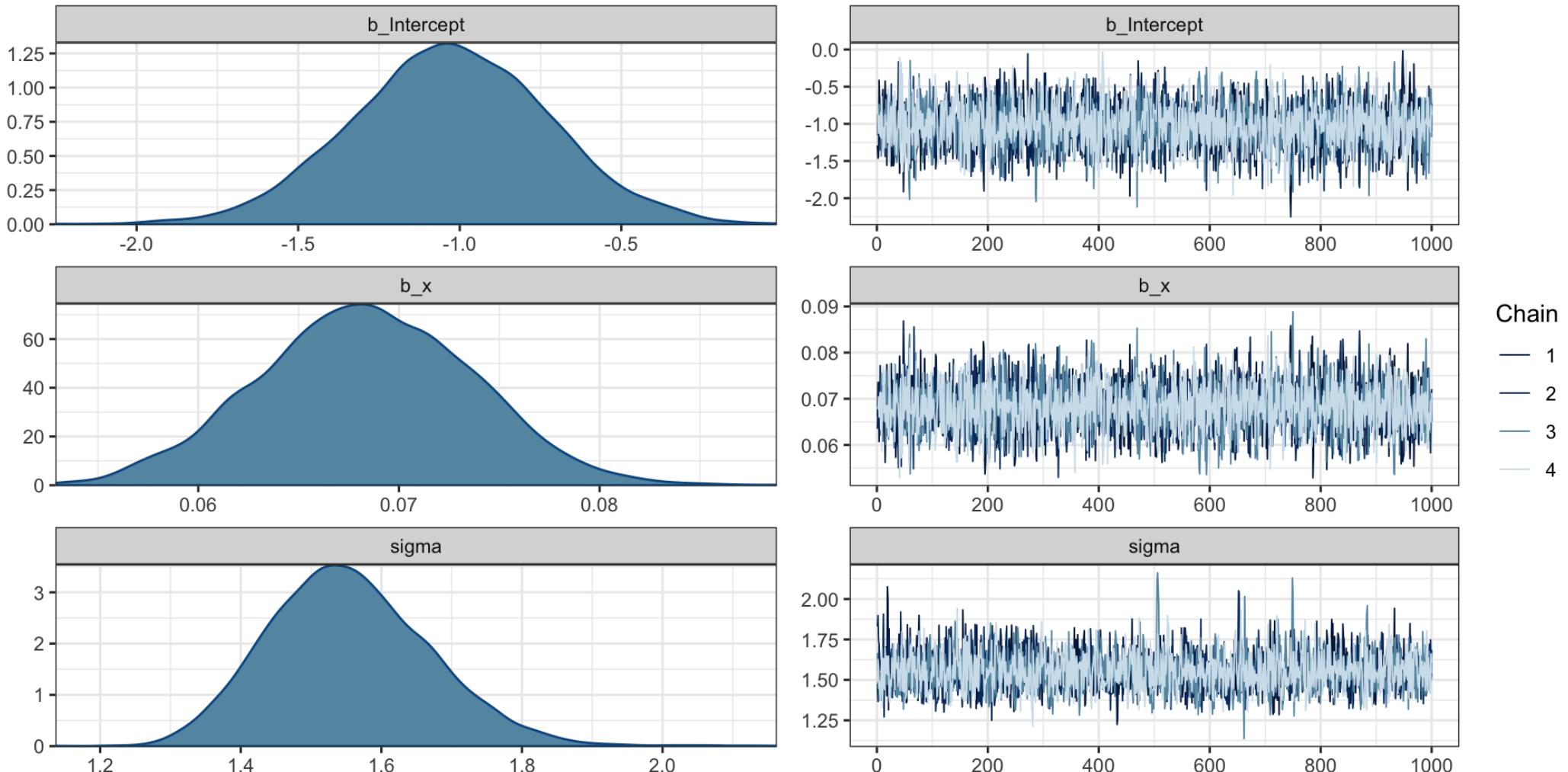
	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
Intercept	-1.03	0.30	-1.63	-0.42	1.00	4020	2851
x	0.07	0.01	0.06	0.08	1.00	4387	2662

Family Specific Parameters:

	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
sigma	1.55	0.12	1.35	1.80	1.00	3103	2316

# Parameter estimates

```
1 plot(b)
```



# tidybayes - gather\_draws (long)

```
1 (b_post = b |>
2   tidybayes::gather_draws(b_Intercept, b_x, sigma))
```

# A tibble: 12,000 × 5  
# Groups: .variable [3]  
 .chain .iteration .draw .variable .value  
 <int> <int> <int> <chr> <dbl>  
1 1 1 1 b\_Intercept -1.29  
2 1 2 2 b\_Intercept -1.46  
3 1 3 3 b\_Intercept -1.44  
4 1 4 4 b\_Intercept -0.414  
5 1 5 5 b\_Intercept -0.658  
6 1 6 6 b\_Intercept -1.04  
7 1 7 7 b\_Intercept -0.919  
8 1 8 8 b\_Intercept -1.06  
9 1 9 9 b\_Intercept -1.15  
10 1 10 10 b\_Intercept -0.634  
# i 11,990 more rows

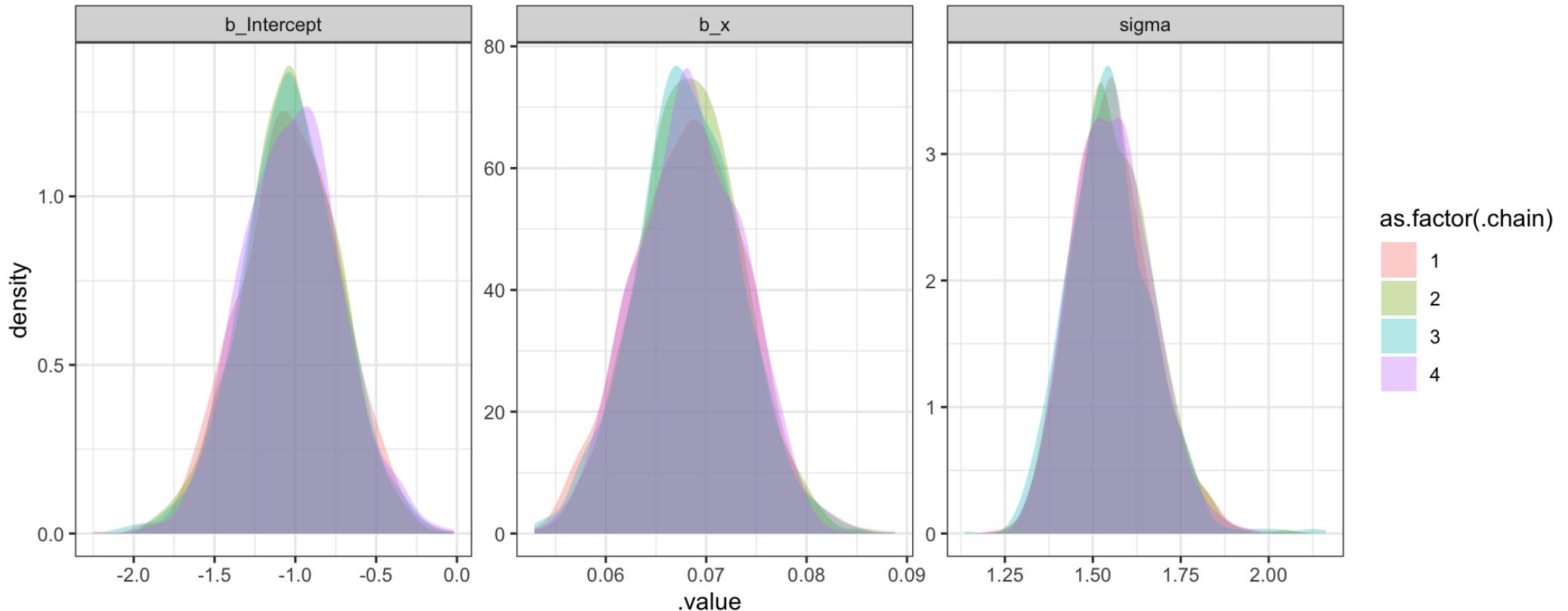
# tidybayes - spread\_draws (wide)

```
1 (b_post_wide = b |>
2   tidybayes::spread_draws(b_Intercept, b_x, sigma))
```

```
# A tibble: 4,000 × 6
  .chain .iteration .draw b_Intercept     b_x   sigma
  <int>      <int> <int>      <dbl>    <dbl>   <dbl>
1     1          1     1      -1.29  0.0662  1.48
2     1          2     2      -1.46  0.0736  1.37
3     1          3     3      -1.44  0.0735  1.40
4     1          4     4      -0.414 0.0607  1.72
5     1          5     5      -0.658 0.0664  1.62
6     1          6     6      -1.04  0.0723  1.60
7     1          7     7      -0.919 0.0656  1.56
8     1          8     8      -1.06  0.0711  1.62
9     1          9     9      -1.15  0.0727  1.54
10    1         10    10      -0.634 0.0666  1.65
# i 3,990 more rows
```

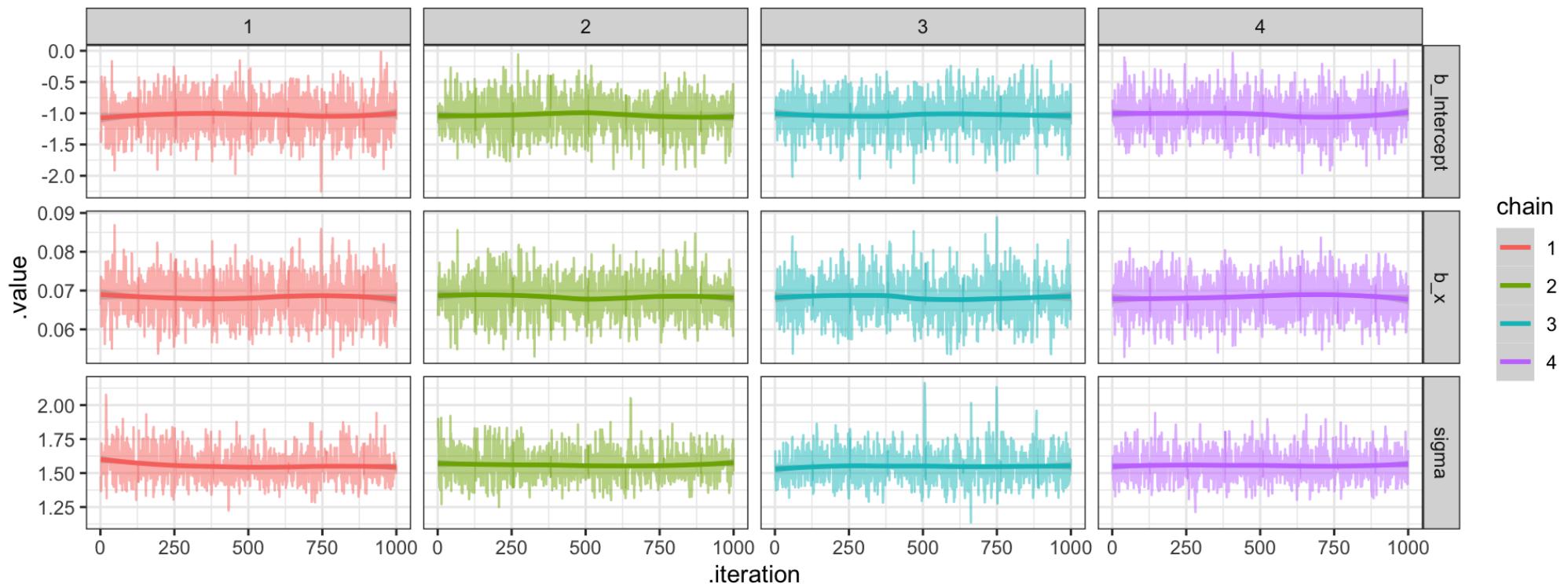
# Posterior plots

```
1 b_post |>
2   ggplot(aes(fill=as.factor(.chain), group=.chain, x=.value)) +
3   geom_density(alpha=0.33, color=NA) +
4   facet_wrap(~.variable, scales = "free")
```



# Trace plots

```
1 b_post |>
2   ggplot(aes(x=.iteration, y=.value, color=as.factor(.chain))) +
3   geom_line(alpha=0.5) +
4   facet_grid(.variable~.chain, scale="free_y") +
5   geom_smooth(method="loess") + labs(color="chain")
```



# Credible Intervals

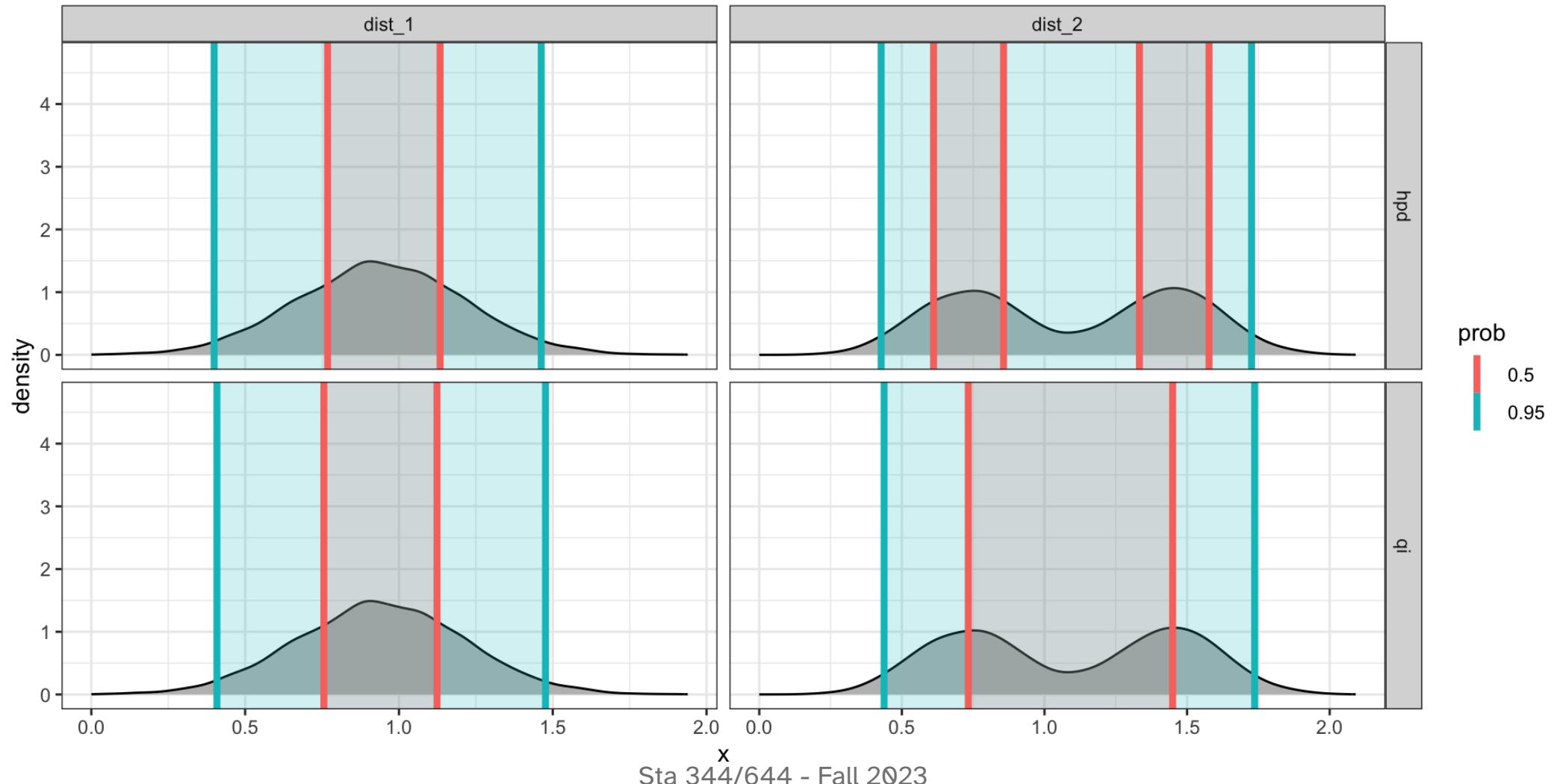
```
1 ( b_ci = b_post |>
2   group_by(.chain, .variable) |>
3   ggdist::mean_hdi(.value, .width=c(0.95))
4 )
```

# A tibble: 12 × 8

	.chain	.variable	.value	.lower	.upper	.width	.point	.interval
	<int>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>
1	1	b_Intercept	-1.03	-1.61	-0.411	0.95	mean	hdi
2	1	b_x	0.0683	0.0570	0.0783	0.95	mean	hdi
3	1	sigma	1.55	1.34	1.77	0.95	mean	hdi
4	2	b_Intercept	-1.03	-1.64	-0.460	0.95	mean	hdi
5	2	b_x	0.0685	0.0594	0.0796	0.95	mean	hdi
6	2	sigma	1.56	1.35	1.78	0.95	mean	hdi
7	3	b_Intercept	-1.03	-1.60	-0.400	0.95	mean	hdi
8	3	b_x	0.0683	0.0567	0.0775	0.95	mean	hdi
9	3	sigma	1.55	1.32	1.76	0.95	mean	hdi
10	4	b_Intercept	-1.02	-1.57	-0.376	0.95	mean	hdi
11	4	b_x	0.0684	0.0587	0.0781	0.95	mean	hdi
12	4	sigma	1.56	1.36	1.78	0.95	mean	hdi

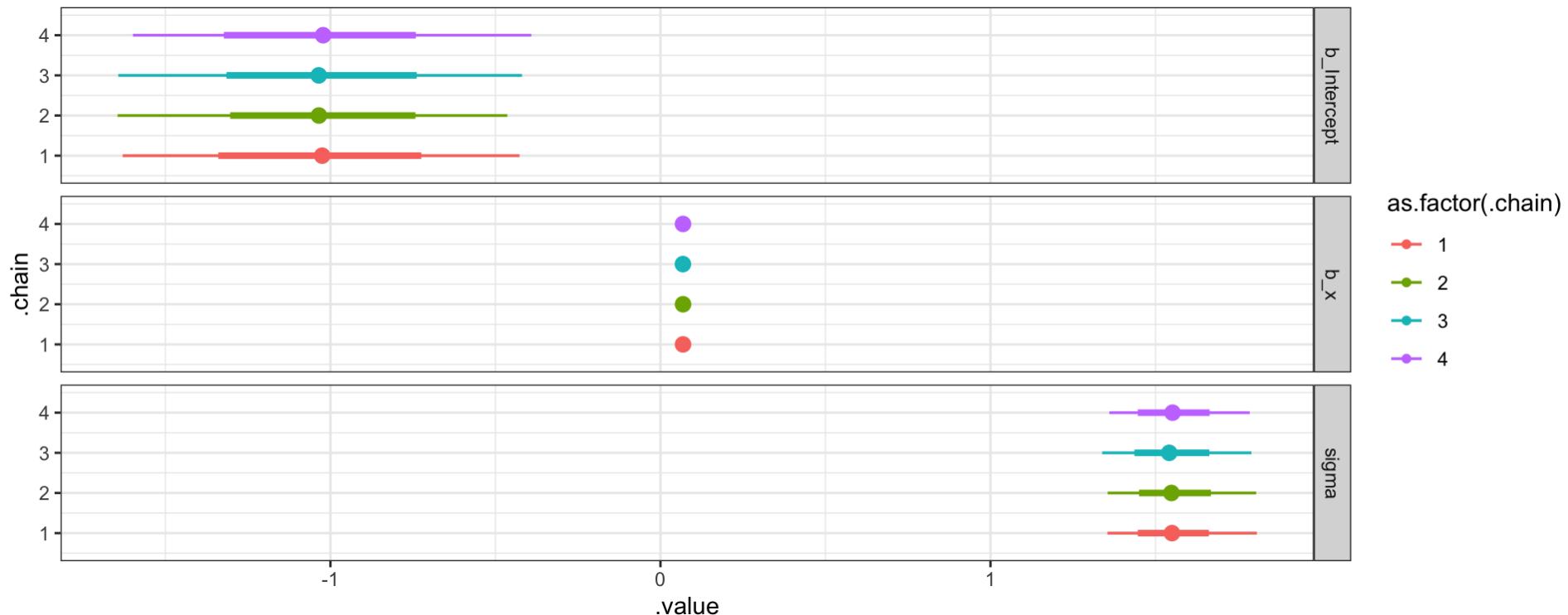
# `mean_qi()` vs `mean_hdi()`

These differ in the use of the quantile interval vs. the highest-density interval.



# Caterpillar Plots

```
1 b_post |>
2   ggplot(aes(x=.value, y=.chain, color=as.factor(.chain))) +
3   facet_grid(.variable ~ .) +
4   ggdist::stat_pointinterval() +
5   ylim(0.5, 4.5)
```



# Predictions

# lm predictions

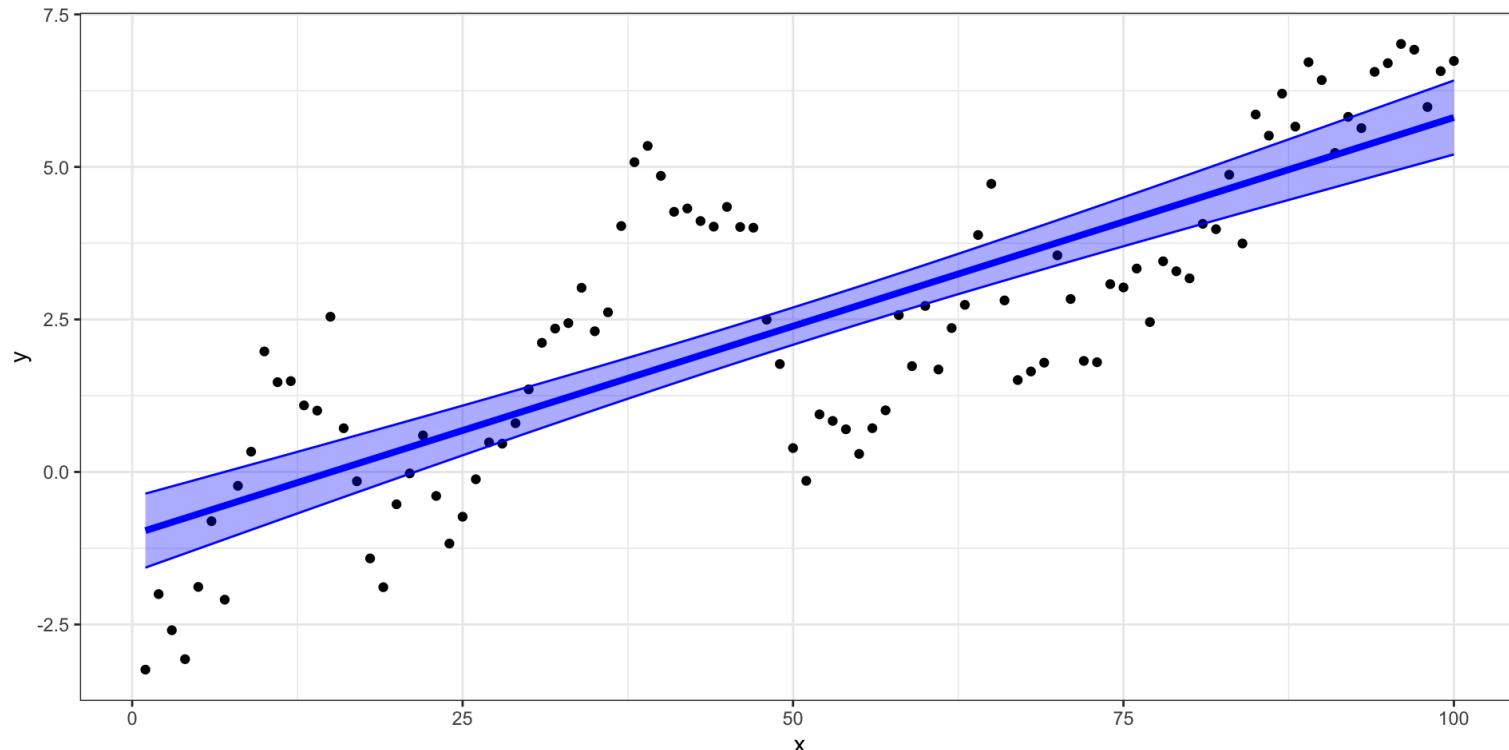
```
1 l_pred = broom::augment(l, interval="confidence"))

# A tibble: 100 × 10
      y     x .fitted .lower   .upper .resid   .hat .sigma .cooksdi .std.resid
  <dbl> <int>   <dbl> <dbl>    <dbl>   <dbl>   <dbl>   <dbl>    <dbl>       <dbl>
1 -3.24     1   -0.962 -1.57   -0.355  -2.28  0.0394  1.53  0.0467    -1.51
2 -2.00     2   -0.893 -1.49   -0.296  -1.11  0.0382  1.54  0.0107    -0.734
3 -2.59     3   -0.825 -1.41   -0.237  -1.77  0.0371  1.54  0.0264    -1.17
4 -3.07     4   -0.757 -1.34   -0.177  -2.31  0.0359  1.53  0.0436    -1.53
5 -1.88     5   -0.688 -1.26   -0.118  -1.20  0.0348  1.54  0.0113    -0.791
6 -0.807    6   -0.620 -1.18   -0.0583  -0.187 0.0338  1.55  0.000266   -0.123
7 -2.09     7   -0.551 -1.10   0.00127  -1.54  0.0327  1.54  0.0175    -1.02
8 -0.227    8   -0.483 -1.03   0.0609   0.256  0.0317  1.55  0.000466   0.169
9  0.333    9   -0.415 -0.950  0.121   0.747  0.0307  1.55  0.00384   0.493
10 1.98     10  -0.346 -0.873  0.180   2.32   0.0297  1.53  0.0358    1.53
# i 90 more rows
```

These are values you would get from `predict(l, se.fit = TRUE)` added to the data frame `d`

# Confidence interval

```
1 l_pred |>
2   ggplot(aes(x=x, y=y)) +
3     geom_point() +
4     geom_line(aes(y=.fitted), col="blue", size=1.5) +
5     geom_ribbon(aes(ymin=.lower, ymax=.upper), col="blue", fill="blue", a
```



# brms predictions

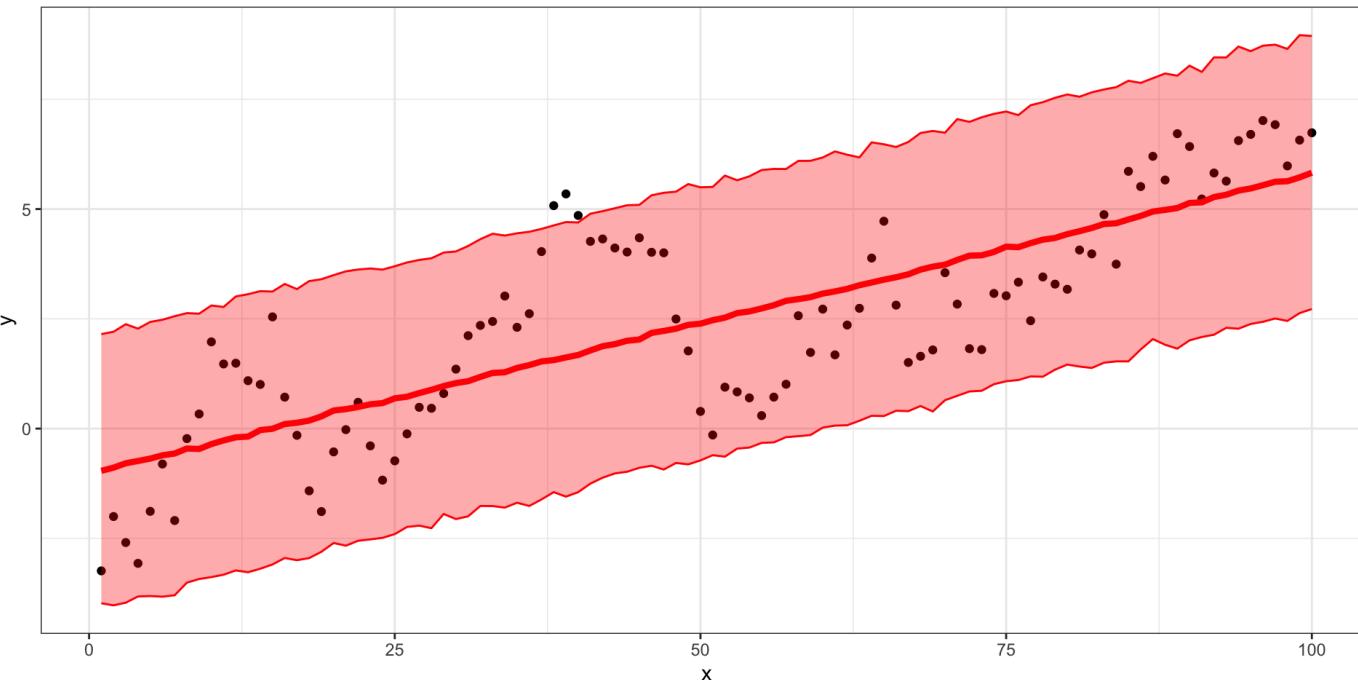
```
1 (b_pred = predict(b)) |> as_tibble()
```

```
# A tibble: 100 × 4
```

	Estimate	Est.Error	Q2.5	Q97.5
	<dbl>	<dbl>	<dbl>	<dbl>
1	-0.962	1.56	-3.98	2.15
2	-0.889	1.60	-4.03	2.21
3	-0.790	1.60	-3.96	2.38
4	-0.738	1.59	-3.82	2.28
5	-0.683	1.58	-3.81	2.43
6	-0.608	1.61	-3.83	2.48
7	-0.568	1.59	-3.80	2.56
8	-0.454	1.58	-3.51	2.63
9	-0.464	1.57	-3.43	2.62
10	-0.353	1.56	-3.38	2.80
# i 90 more rows				

# Credible interval

```
1 d |>
2   bind_cols(b_pred) |>
3   ggplot(aes(x=x,y=y)) +
4     geom_point() +
5     geom_line(aes(y=Estimate), col="red", size=1.5) +
6     geom_ribbon(aes(ymin=Q2.5, ymax=Q97.5), col='red', fill='red', alpha=0.2)
```



# Why are the intervals different?

# Raw predictions

```
1 dim( brms::posterior_predict(b) )
```

```
[1] 4000 100
```

```
1 dim( brms::posterior_epred(b) )
```

```
[1] 4000 100
```

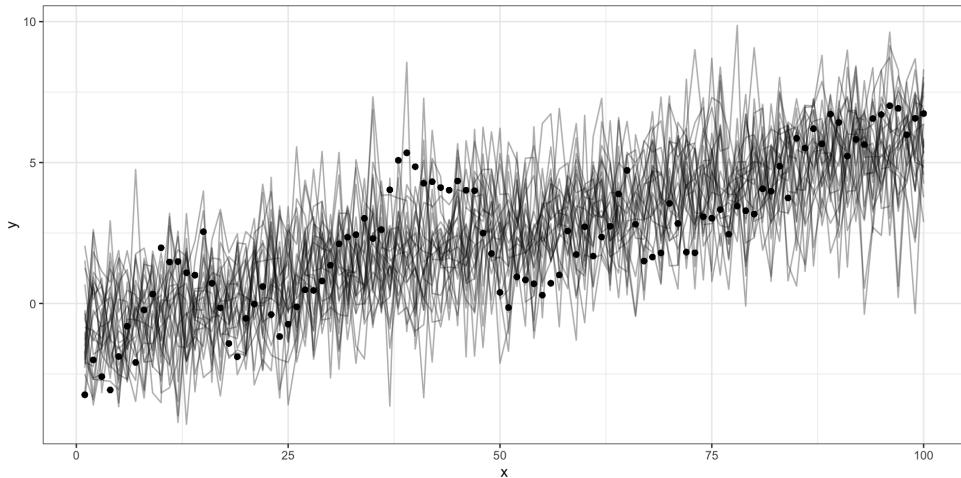
# Tidy raw predictions

```
1 ( b_post_pred = tidybayes::predicted_draws(  
2   b, newdata=d  
3 ) )  
  
# A tibble: 400,000 × 7  
# Groups:   x, y, .row [100]  
  x     y   .row .chain .iteration .draw  
  <int> <dbl> <int> <int>      <int> <int>  
1 1 -3.24 1 NA NA 1  
2 1 -3.24 1 NA NA 2  
3 1 -3.24 1 NA NA 3  
4 1 -3.24 1 NA NA 4  
5 1 -3.24 1 NA NA 5  
6 1 -3.24 1 NA NA 6  
7 1 -3.24 1 NA NA 7  
8 1 -3.24 1 NA NA 8  
9 1 -3.24 1 NA NA 9  
10 1 -3.24 1 NA NA 10  
# i 399,990 more rows  
# i 1 more variable: .prediction <dbl>
```

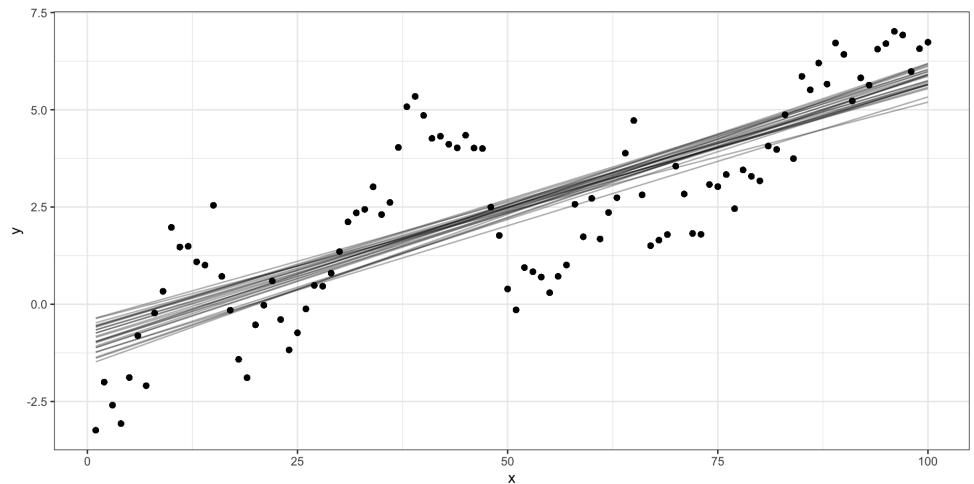
```
1 ( b_post_epred = tidybayes::epred_draws(  
2   b, newdata=d  
3 ) )  
  
# A tibble: 400,000 × 7  
# Groups:   x, y, .row [100]  
  x     y   .row .chain .iteration .draw  
  <int> <dbl> <int> <int>      <int> <int>  
1 1 -3.24 1 NA NA 1  
2 1 -3.24 1 NA NA 2  
3 1 -3.24 1 NA NA 3  
4 1 -3.24 1 NA NA 4  
5 1 -3.24 1 NA NA 5  
6 1 -3.24 1 NA NA 6  
7 1 -3.24 1 NA NA 7  
8 1 -3.24 1 NA NA 8  
9 1 -3.24 1 NA NA 9  
10 1 -3.24 1 NA NA 10  
# i 399,990 more rows  
# i 1 more variable: .epred <dbl>
```

# Posterior predictions vs Expected posterior predictions

```
1 b_post_pred |>
2   filter(.draw <= 25) |>
3   ggplot(aes(x=x,y=y)) +
4     geom_point() +
5     geom_line(
6       aes(y=.prediction, group=.draw),
7       alpha=0.33
8     )
```



```
1 b_post_epred |>
2   filter(.draw <= 25) |>
3   ggplot(aes(x=x,y=y)) +
4     geom_point() +
5     geom_line(
6       aes(y=.epred, group=.draw),
7       alpha=0.33
8     )
```



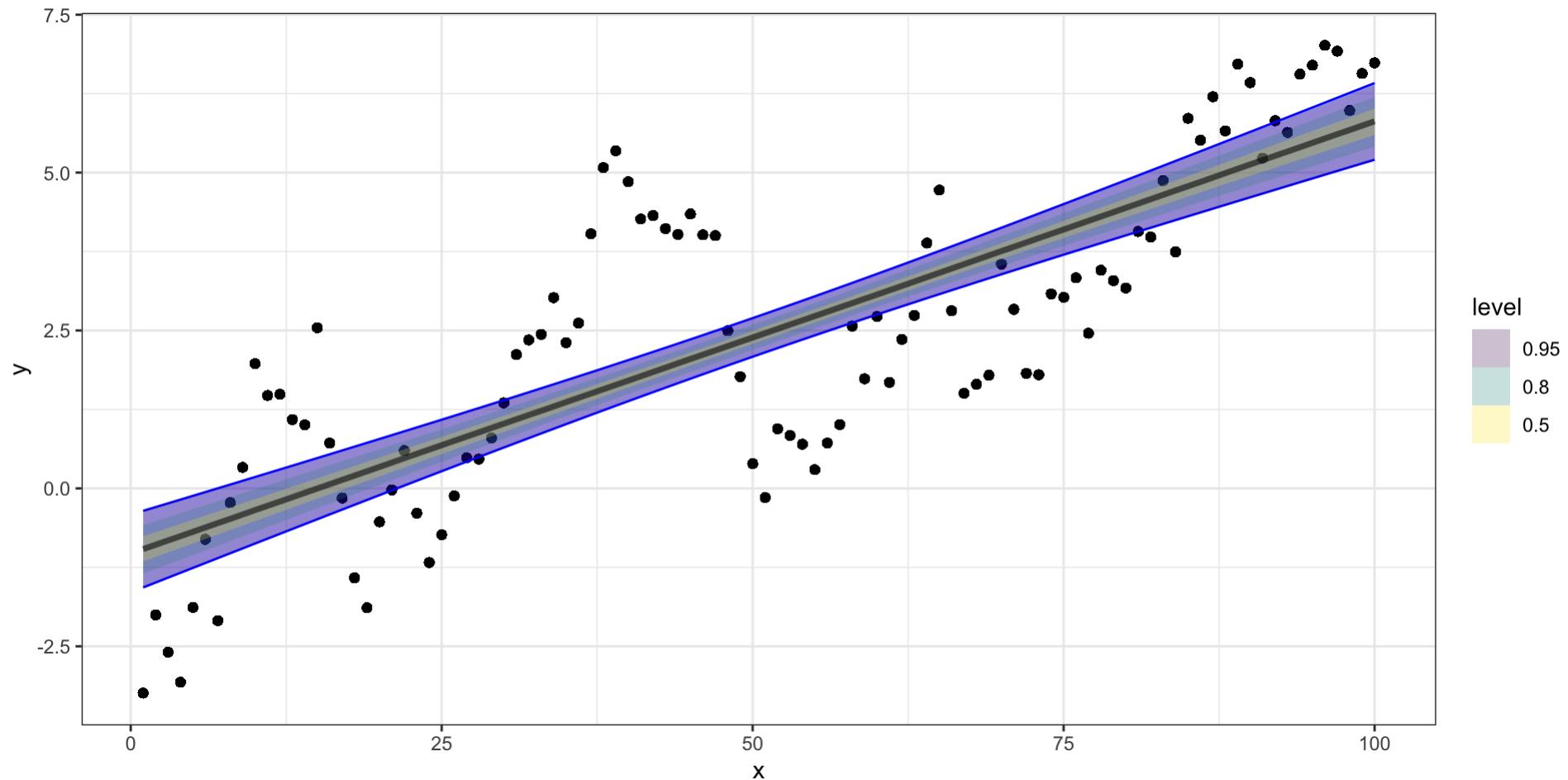
# Credible interval - predicted

```
1 ggplot(b_post_pred, aes(x=x, y=y)) +  
2   geom_point() +  
3   ggdist::stat_lineribbon(  
4     aes(y=.prediction), alpha=0.25  
5   )
```

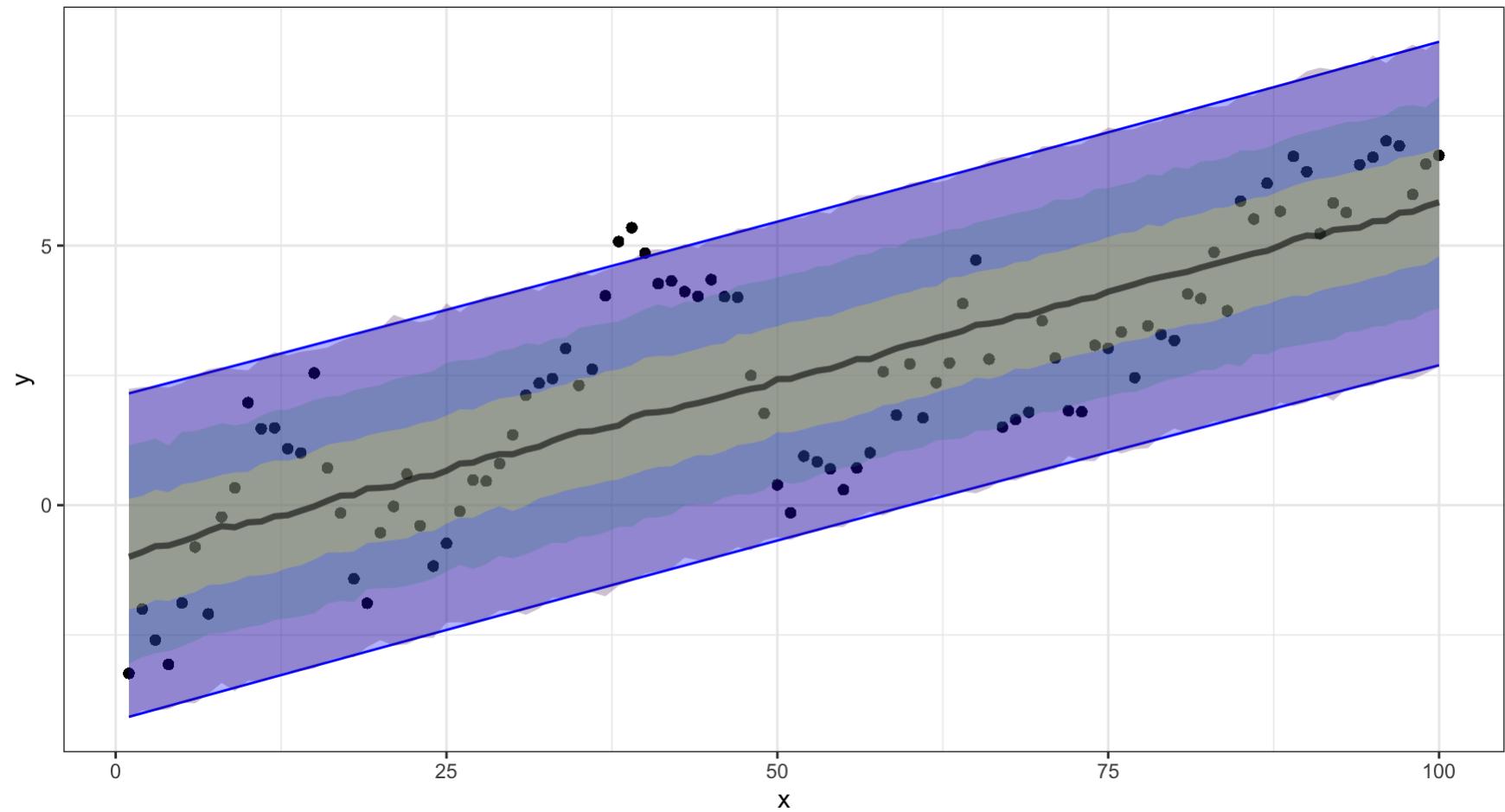
# Credible interval - epred

```
1 ggplot(b_post_epred, aes(x=x, y=y)) +
2   geom_point() +
3   ggdist::stat_lineribbon(
4     aes(y=.epred), alpha=0.25
5   )
```

# Confidence interval (frequentist)



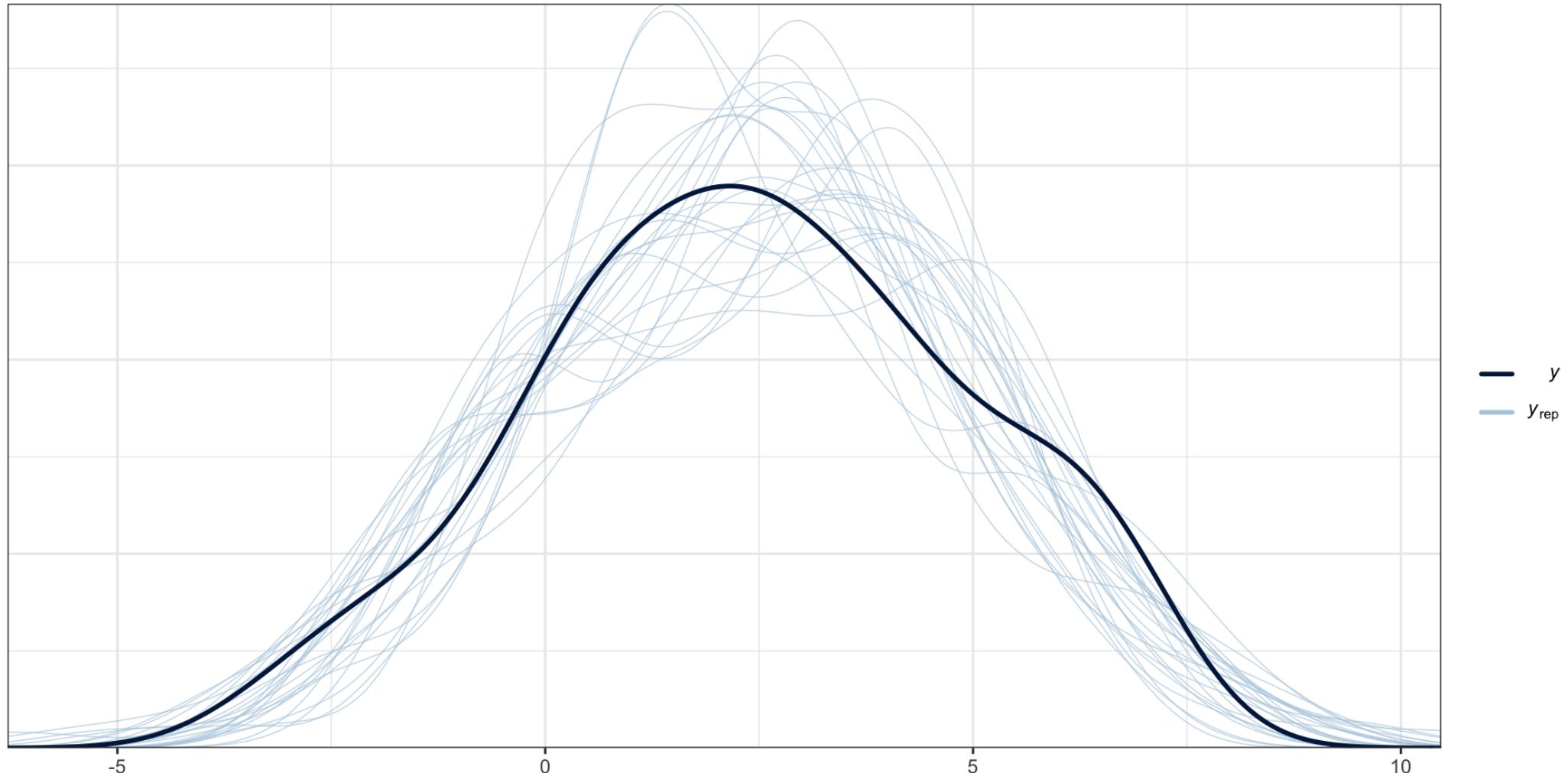
# Prediction interval (frequentist)



# Other useful plots

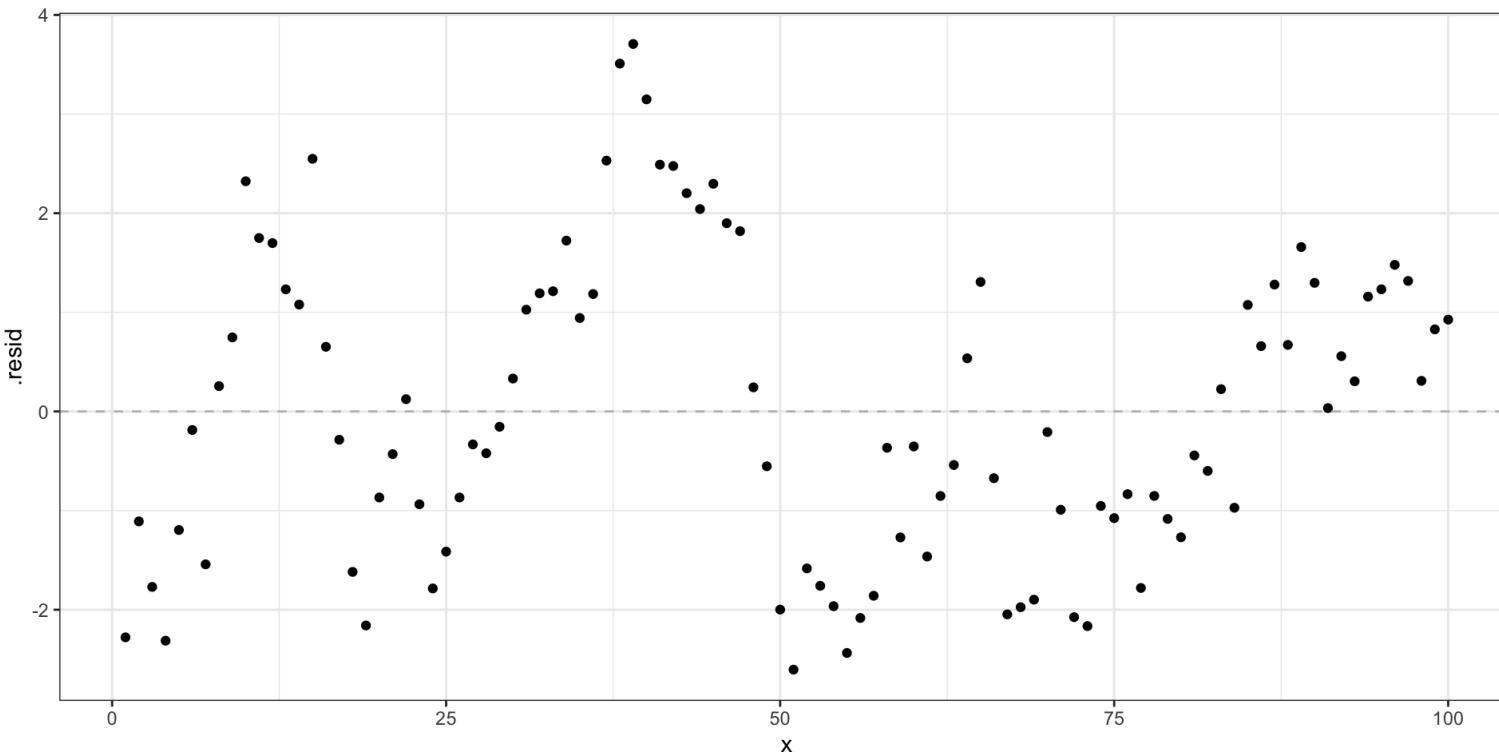
# Posterior predictive checks

```
1 brms:::pp_check(b, ndraws = 25)
```



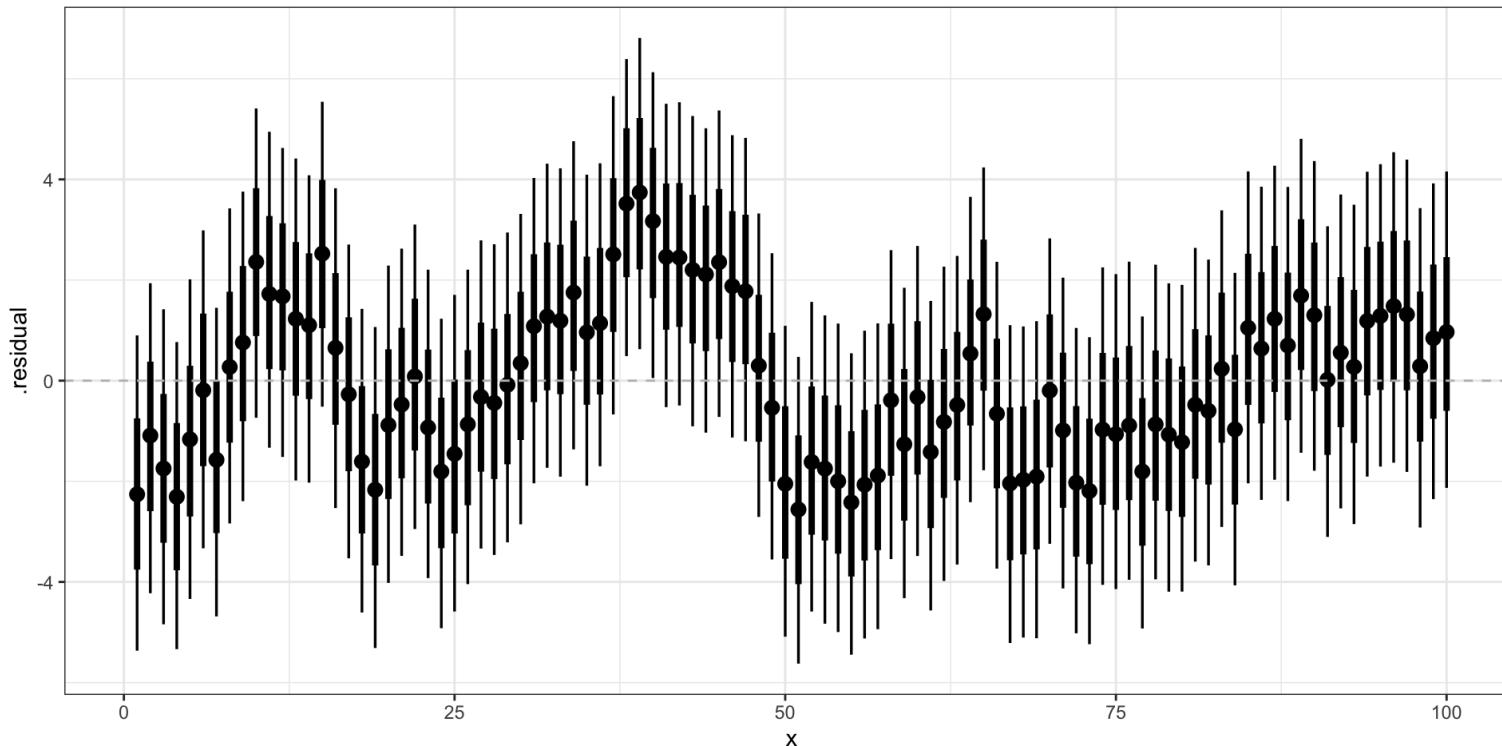
# Residuals - lm

```
1 1 |>
2 broom::augment() |>
3 ggplot(aes(x=x, y=.resid)) +
4 geom_point() +
5 geom_hline(yintercept=0, color='grey', linetype=2)
```



# Residual posteriors - brms

```
1 b |>
2 tidybayes::residual_draws(newdata=d) |>
3 ggplot(aes(x=x, y=.residual, group=x)) +
4 ggdist::stat_pointinterval() +
5 geom_hline(yintercept = 0, color='grey', linetype=2)
```



# Model Evaluation

# Model assessment

If we remember back to our first regression class, one common option is  $R^2$  which gives us the variability in  $y$  explained by our model.

Quick review:

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Total                    Model                    Error

$$R^2 = \frac{SS_{\text{model}}}{SS_{\text{total}}} = \frac{\sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2} = \frac{\text{Var}(\hat{Y})}{\text{Var}(Y)} = \text{Cor}(Y, \hat{Y})^2$$

# Some data prep

```
1 ( b_post_full = b |>
  2   tidybayes::spread_draws(b_Intercept, b_x, sigma) |>
  3   tidyr::expand_grid(d) |>
  4   mutate(
  5     y_hat = b_Intercept + b_x * x,
  6     resid = y - y_hat
  7   )
  8 )
```

```
# A tibble: 400,000 × 10
  .chain .iteration .draw b_Intercept     b_x sigma      x      y    y_hat    resid
  <int>     <int> <int>     <dbl>   <dbl> <dbl> <int>   <dbl>   <dbl>   <dbl>
1     1         1     1     -1.29  0.0662  1.48     1 -3.24  -1.23  -2.01
2     1         1     1     -1.29  0.0662  1.48     2 -2.00  -1.16  -0.842
3     1         1     1     -1.29  0.0662  1.48     3 -2.59  -1.09  -1.50
4     1         1     1     -1.29  0.0662  1.48     4 -3.07  -1.03  -2.04
5     1         1     1     -1.29  0.0662  1.48     5 -1.88  -0.962 -0.923
6     1         1     1     -1.29  0.0662  1.48     6 -0.807 -0.895  0.0887
7     1         1     1     -1.29  0.0662  1.48     7 -2.09  -0.829 -1.26
8     1         1     1     -1.29  0.0662  1.48     8 -0.227 -0.763  0.536
9     1         1     1     -1.29  0.0662  1.48     9  0.333  -0.697  1.03
```

```
10      1      1      -1.29  0.0662  1.48      10  1.98  -0.630  2.61
# i 399,990 more rows
```

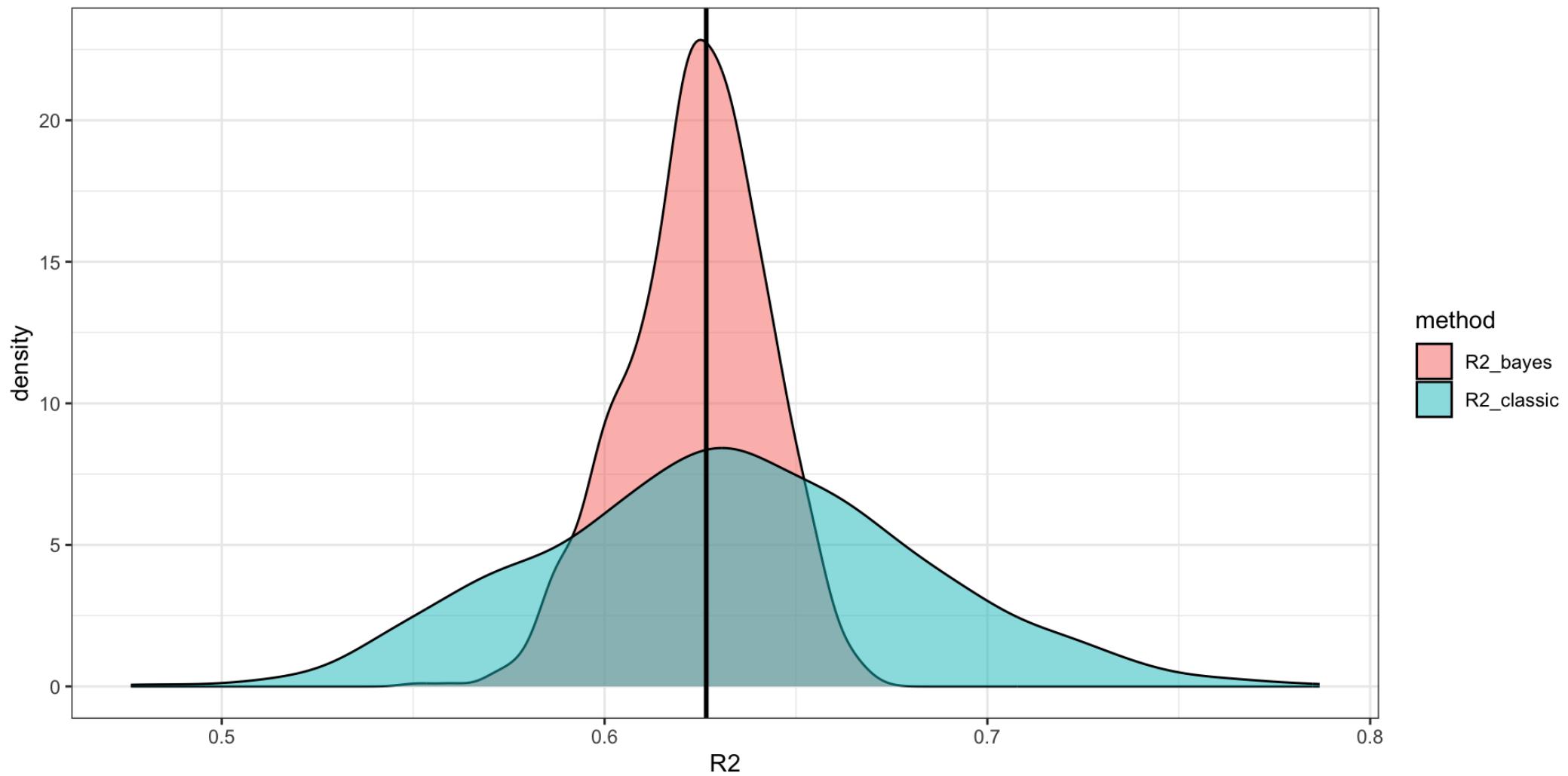
# Bayesian R<sup>2</sup>

When we compute any statistic for our model we want to do so at each iteration so that we can obtain the posterior distribution of that particular statistic (e.g. the posterior distribution of R<sup>2</sup> in this case).

```
1 ( b_R2 = b_post_full |>
  2   group_by(.iteration) |>
  3   summarize(
  4     R2_classic = var(y_hat) / var(y),
  5     R2_bayes   = var(y_hat) /
  6                   (var(y_hat) + var(resid))
  7   )
  8 )
```

```
# A tibble: 1,000 × 3
  .iteration R2_classic R2_bayes
      <int>      <dbl>      <dbl>
1          1      0.602     0.612
2          2      0.651     0.634
3          3      0.666     0.639
4          4      0.614     0.619
5          5      0.621     0.619
6          6      0.707     0.652
7          7      0.571     0.602
8          8      0.708     0.652
9          9      0.580     0.605
10         10      0.629     0.626
# i 990 more rows
```

# Uh oh ...



# Frequentist sanity check

```
1 l_pred = broom::augment(l)
```

```
# A tibble: 100 × 8
```

	y	x	.fitted	.resid	.hat	.sigma
	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
1	-3.24	1	-0.962	-2.28	0.0394	1.53
2	-2.00	2	-0.893	-1.11	0.0382	1.54
3	-2.59	3	-0.825	-1.77	0.0371	1.54
4	-3.07	4	-0.757	-2.31	0.0359	1.53
5	-1.88	5	-0.688	-1.20	0.0348	1.54
6	-0.807	6	-0.620	-0.187	0.0338	1.55
7	-2.09	7	-0.551	-1.54	0.0327	1.54
8	-0.227	8	-0.483	0.256	0.0317	1.55
9	0.333	9	-0.415	0.747	0.0307	1.55
10	1.98	10	-0.346	2.32	0.0297	1.53

```
# i 90 more rows
```

```
# i 2 more variables: .cooksdi <dbl>,
#   .std.resid <dbl>
```

```
1 broom::glance(l)$r.squared
```

```
[1] 0.6265565
```

```
1 var(l_pred$.fitted) / var(l_pred$y)
```

```
[1] 0.6265565
```

```
1 var(l_pred$.fitted) / (var(l_pred$.fitted) +
```

```
[1] 0.6265565
```

# What if we collapsed first?

Here we calculate the posterior mean of  $\hat{y}$  and use that to estimate  $R^2$ ,

```
1 b_post_full |>
2   group_by(x) |>
3   summarize(
4     y_hat = mean(y_hat),
5     y = mean(y),
6     resid = mean(y - y_hat),
7     .groups = "drop"
8   ) |>
9   summarize(
10    R2_classic = var(y_hat) / var(y),
11    R2_bayes   = var(y_hat) / (var(y_hat) + var(resid))
12  )
```

```
# A tibble: 1 × 2
R2_classic R2_bayes
      <dbl>     <dbl>
1     0.626     0.626
```

# Some problems with $R^2$

Some new issues,

- $R^2$  doesn't really make sense in the Bayesian context
  - multiple possible definitions with different properties
  - fundamental equality doesn't hold anymore
  - Possible to have  $R^2 > 1$

Some old issues,

- $R^2$  always increases (or stays the same) when adding a predictor
- $R^2$  is highly susceptible to over fitting
- $R^2$  is sensitive to outliers
- $R^2$  depends heavily on values of  $y$  (can differ for two equivalent models)

# Some Other Metrics

# Root Mean Square Error

The traditional definition of rmse is as follows

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

In the bayesian context, we have posterior samples from each parameter / prediction of interest so we can express this as

$$\frac{1}{m} \sum_{s=1}^m \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_{i,s})^2}$$

where  $m$  is the number of iterations and  $\hat{Y}_i^s$  is the prediction for  $Y_i$  at iteration  $s$ .

# Continuous Rank Probability Score

Another approach is the continuous rank probability score which comes from the probabilistic forecasting literature, it compares the full posterior predictive distribution to the observation / truth.

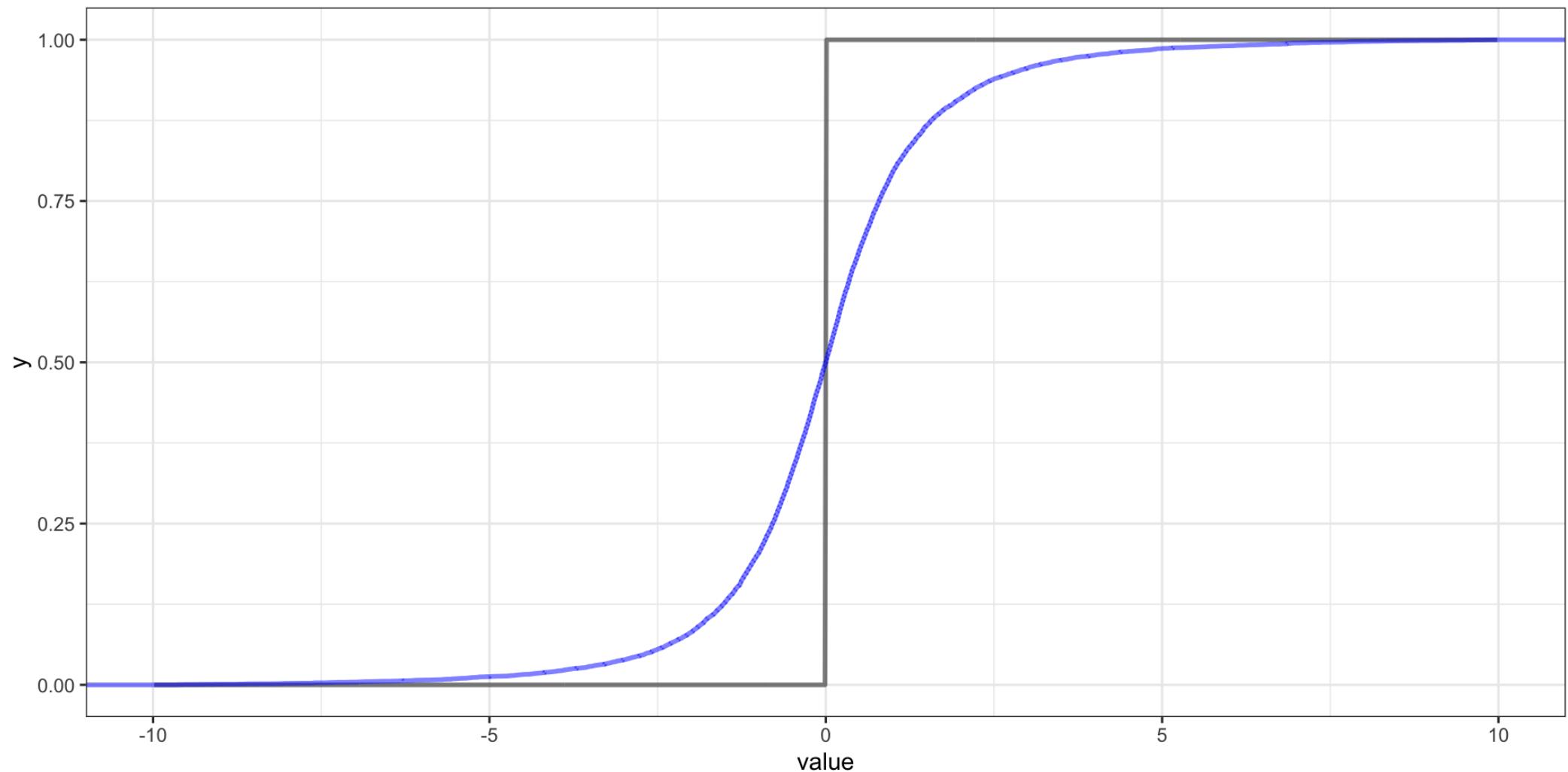
$$\text{CRPS} = \int_{-\infty}^{\infty} (F_{\hat{y}}(z) - 1_{z \geq y})^2 dz$$

where  $F_{\hat{y}}$  is the CDF of  $\hat{y}$  (the posterior predictive distribution for  $y$ ) and  $1_{z \geq Y}$  is an indicator function which equals 1 when  $z \geq y$ , the true/observed value of  $y$ .

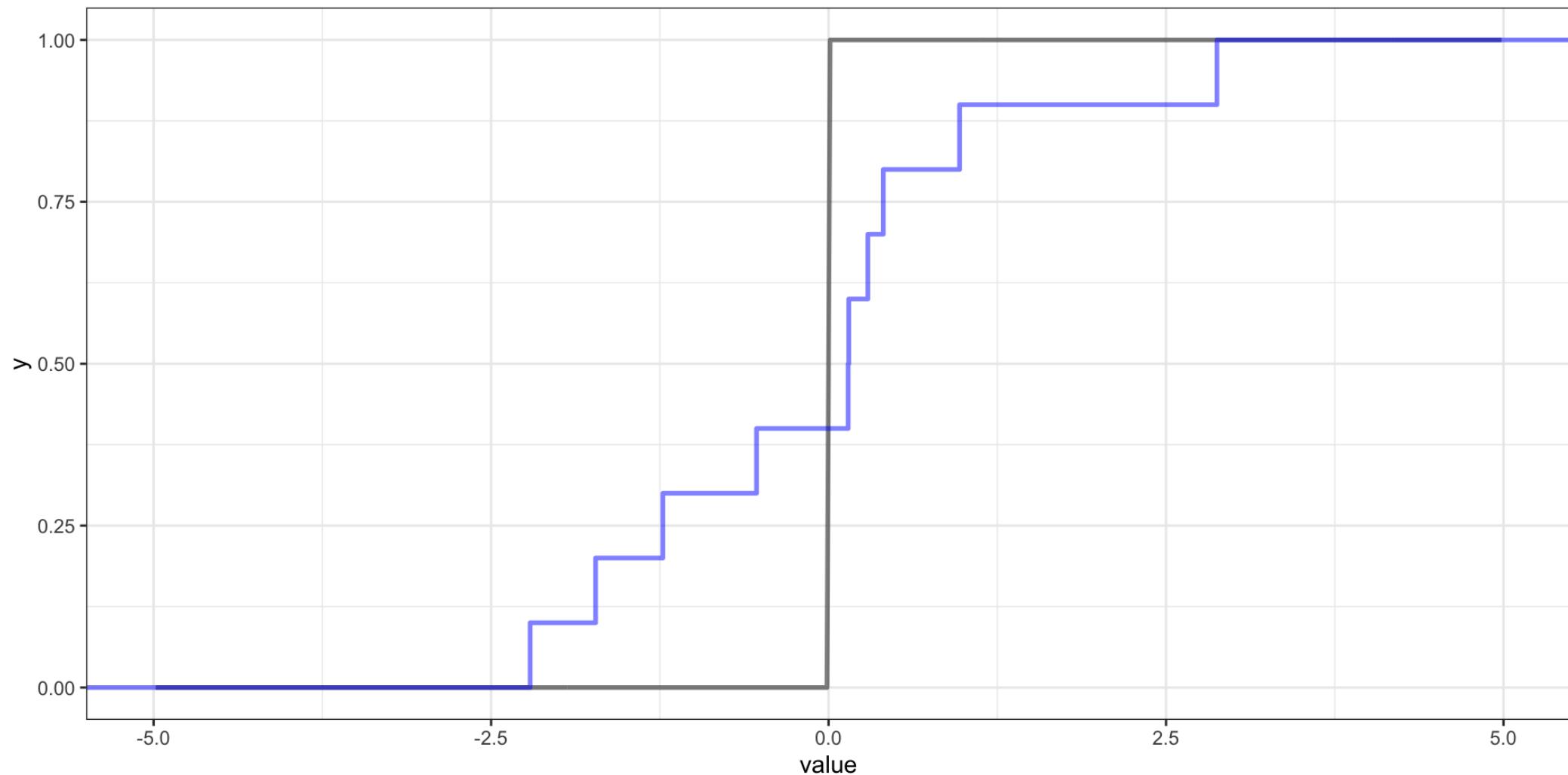
Since this calculates a score for a single probabilistic prediction we can naturally extend it to multiple predictions by calculating an average CRPS

$$\frac{1}{n} \sum_{i=1}^n \int_{-\infty}^{\infty} (F_{\hat{y}_i}(z) - 1_{z \geq y_i})^2 dz$$

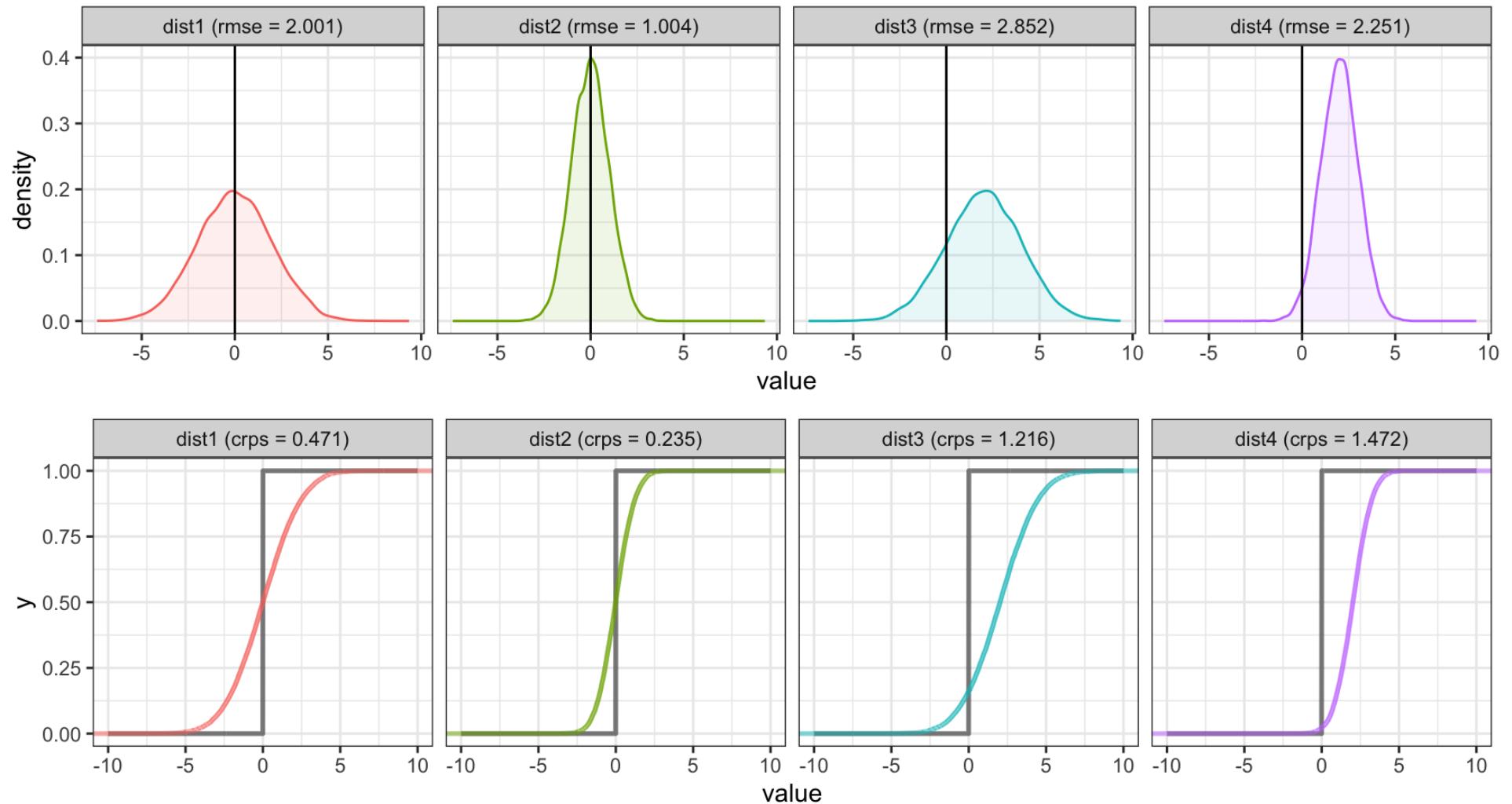
# CDF vs Indicator



# Empirical CDF vs Indicator



# Accuracy vs. Precision



# Empirical Coverage

One final method, which assesses model calibration is to examine how well credible intervals, derived from the posterior predictive distributions of the  $y_s$ , capture the true/observed values.



# Back to our example

# RMSE

```
1 broom::augment(l) |>
2   yardstick::rmse(y, .fitted)

# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>        <dbl>
1 rmse    standard     1.52
```

```
1 ( b_rmse = b_post_full |>
2   group_by(.chain, .iteration) |>
3   summarize(
4     rmse = sqrt( sum( (y - y_hat)^2 ) / n()
5   )
6 )

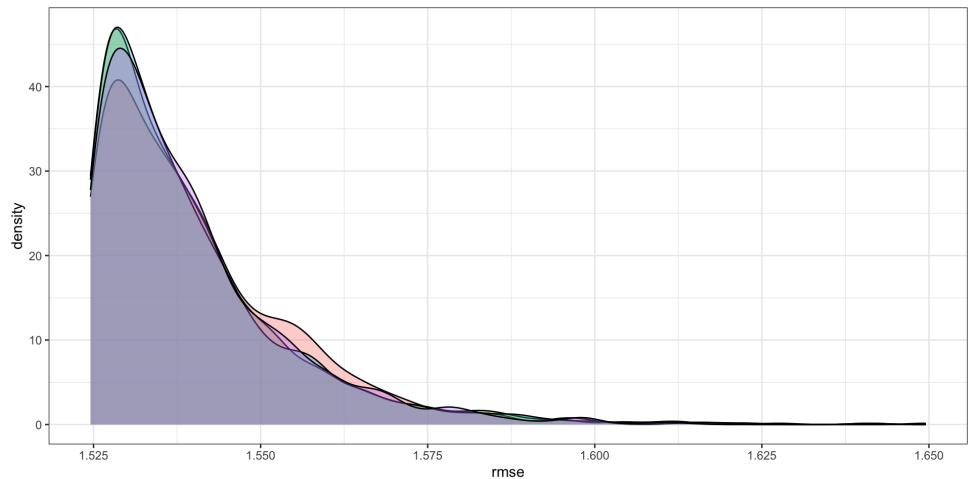
# A tibble: 4,000 × 3
# Groups:   .chain [4]
  .chain .iteration   rmse
  <int>      <int> <dbl>
1       1          1  1.57
2       1          2  1.54
3       1          3  1.54
4       1          4  1.56
5       1          5  1.55
6       1          6  1.54
7       1          7  1.53
8       1          8  1.53
9       1          9  1.53
10      1         10  1.56
# i 3,990 more rows
```

# RMSE (cont)

```
1 b_rmse |>
2   group_by(.chain) |>
3   summarize(post_mean = mean(rmse))

# A tibble: 4 × 2
  .chain post_mean
  <int>     <dbl>
1     1     1.54
2     2     1.54
3     3     1.54
4     4     1.54
```

```
1 ggplot(b_rmse) +
2   geom_density(
3     aes(x=rmse, fill=as.factor(.chain))
4     alpha=0.33
5   ) +
6   guides(fill="none")
```



# CRPS

# Empirical Coverage

```
1 ( b_cover = b_post_full |>
  2   group_by(x, y) |>
  3   tidybayes::mean_hdi(
  4     y_hat, .prob = c(0.5,0.9,0.95)
  5   )
  6 )
```

```
# A tibble: 300 × 8
      x     y   y_hat .lower .upper .width .point .interval
  <int> <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
1     1 -3.24 -0.960 -1.13 -0.736  0.5 mean   hdi
2     2 -2.00 -0.892 -1.07 -0.676  0.5 mean   hdi
3     3 -2.59 -0.823 -0.988 -0.602  0.5 mean   hdi
4     4 -3.07 -0.755 -0.938 -0.558  0.5 mean   hdi
5     5 -1.88 -0.687 -0.868 -0.493  0.5 mean   hdi
6     6 -0.807 -0.618 -0.778 -0.408  0.5 mean   hdi
7     7 -2.09 -0.550 -0.730 -0.367  0.5 mean   hdi
8     8 -0.227 -0.482 -0.644 -0.285  0.5 mean   hdi
9     9  0.333 -0.413 -0.578 -0.226  0.5 mean   hdi
10    10  1.98 -0.345 -0.505 -0.157  0.5 mean   hdi
# i 290 more rows
```

# Empirical Coverage - $\hat{y}$ - Results

```
1 b_cover |>
2   mutate(contains = y >= .lower & y <= .upper) |>
3   group_by(prob = .width) |>
4   summarize(
5     emp_cov = sum(contains)/n()
6   )
```

```
# A tibble: 3 × 2
```

	prob	emp_cov
1	0.5	0.02
2	0.9	0.11
3	0.95	0.14

# What went wrong now?

