Simple Linear Models

A jack of all trades?

Diogo Melo - 27/01/2025

Why use probabilistic models?

What questions do we want to answer?

- Statistical models should answer questions
  - What is the relation between two variables?
  - What is the difference between two groups?
  - What are the sources of variation in the data?
  - What is the expected result of an intervention?

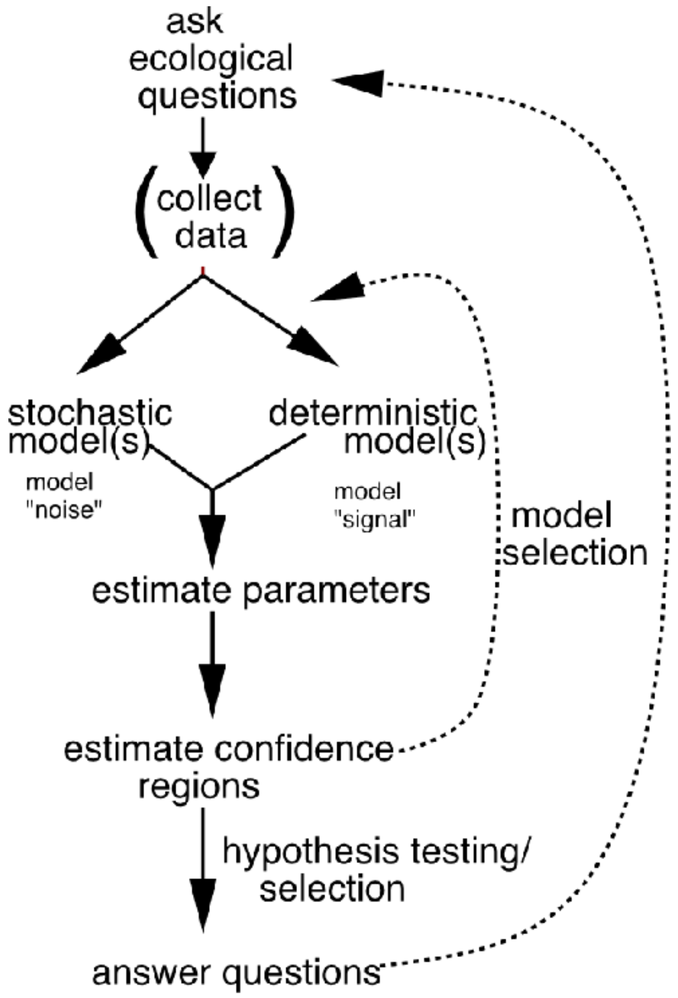
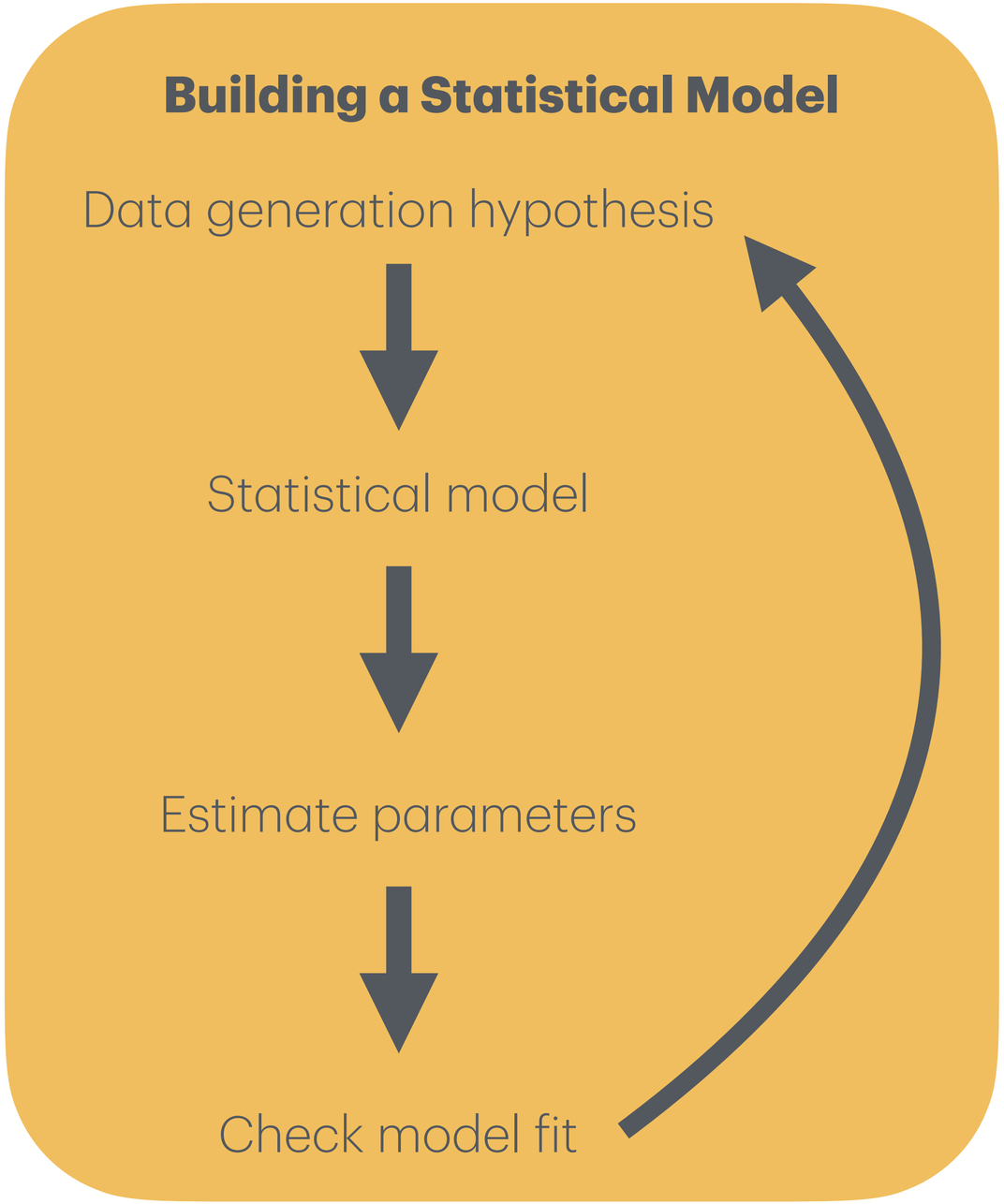


Figure 1.5 Flow of the modeling process.

The basic model

Our hammer

- Our model must:
  - Define a relation between observations and parameters
  - Create a probabilistic description of the system we are studying
  - The description should capture the aspects of the data we are interested in!



Probability definitions

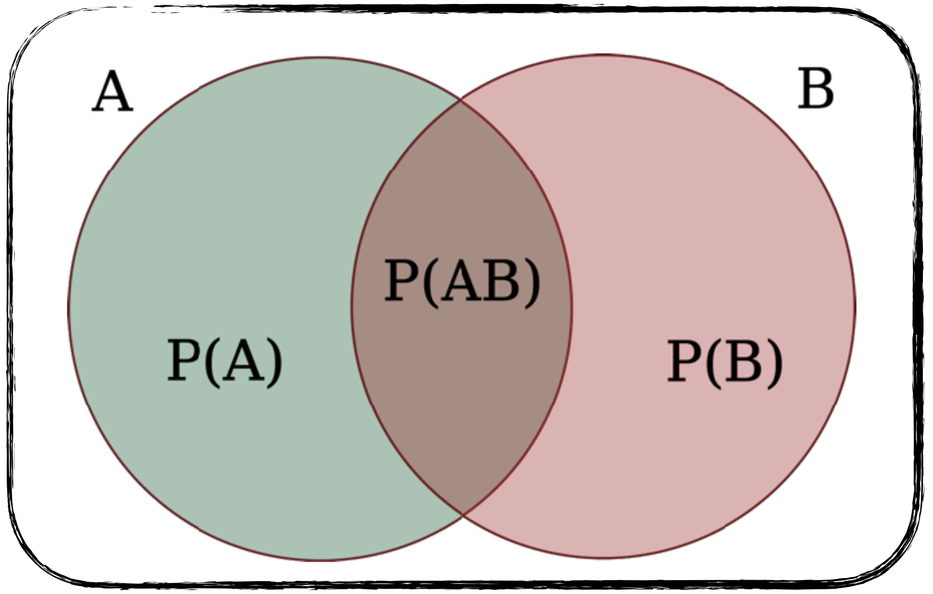
Crash course

- We can express the probability of some variable as:
  - $P(y)$: Read as the “**probability of y**"
- If the distribution of x depends on some parameters, we use the conditional probability:
  - $P(y \mid x)$: Read as the “**probability of y given x**”

Probability rules

- If two events, A and B, are **independent**:
  - $P(A \mid B) = P(A)$: If A and B are independent, the probability of A given B is just the probability of A.
- More generally, we have **the product rule**:
  - $P(AB) = P(A)P(B \mid A) = P(B)P(A \mid B)$

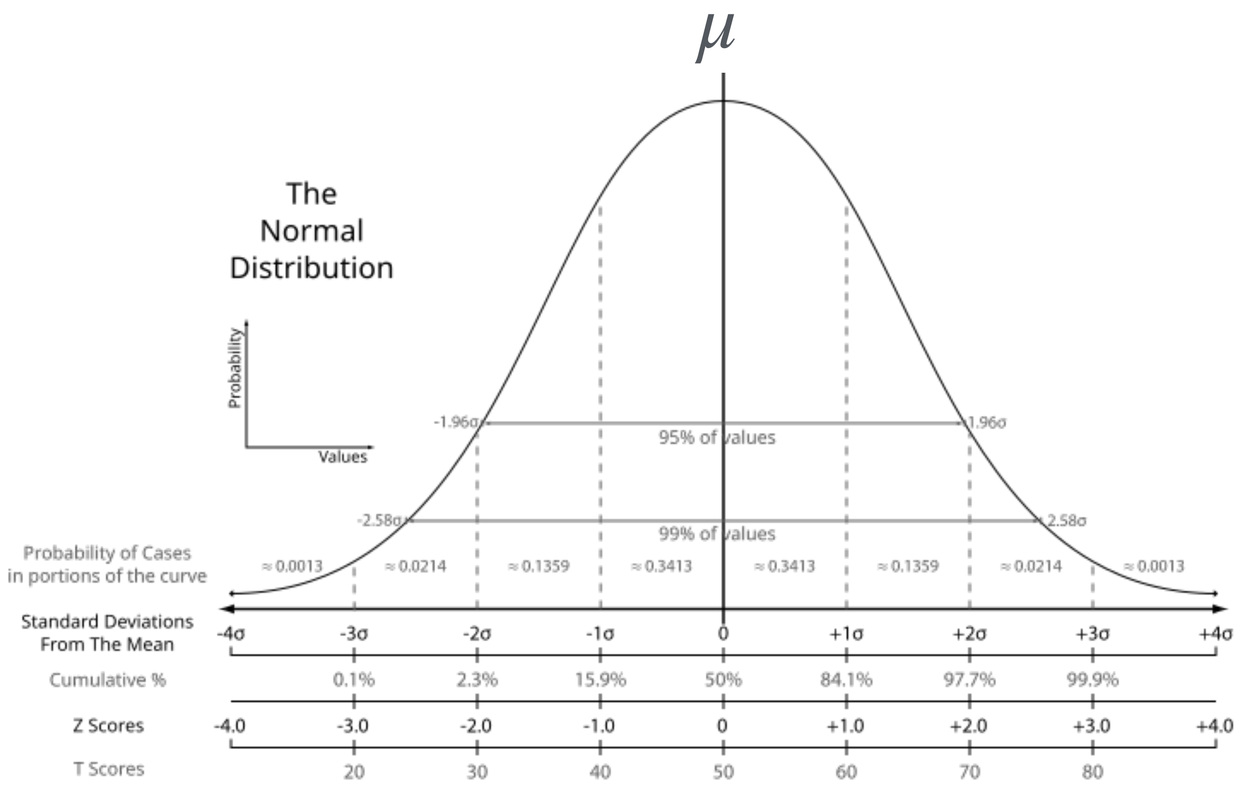
$$P(A \mid B) = \frac{P(A)P(B \mid A)}{P(B)}$$



Probability distribution

Crash course

- We can use standard probability distributions to define these relations
- If a variable y follows a normal distribution:
  - $P(y) = P(y \mid \mu, \sigma) = Normal(y \mid \mu, \sigma)$
  - Where μ and σ are **parameters**
    - μ: is the mean, a location parameter
    - σ: sigma is the standard deviation, a scale parameter

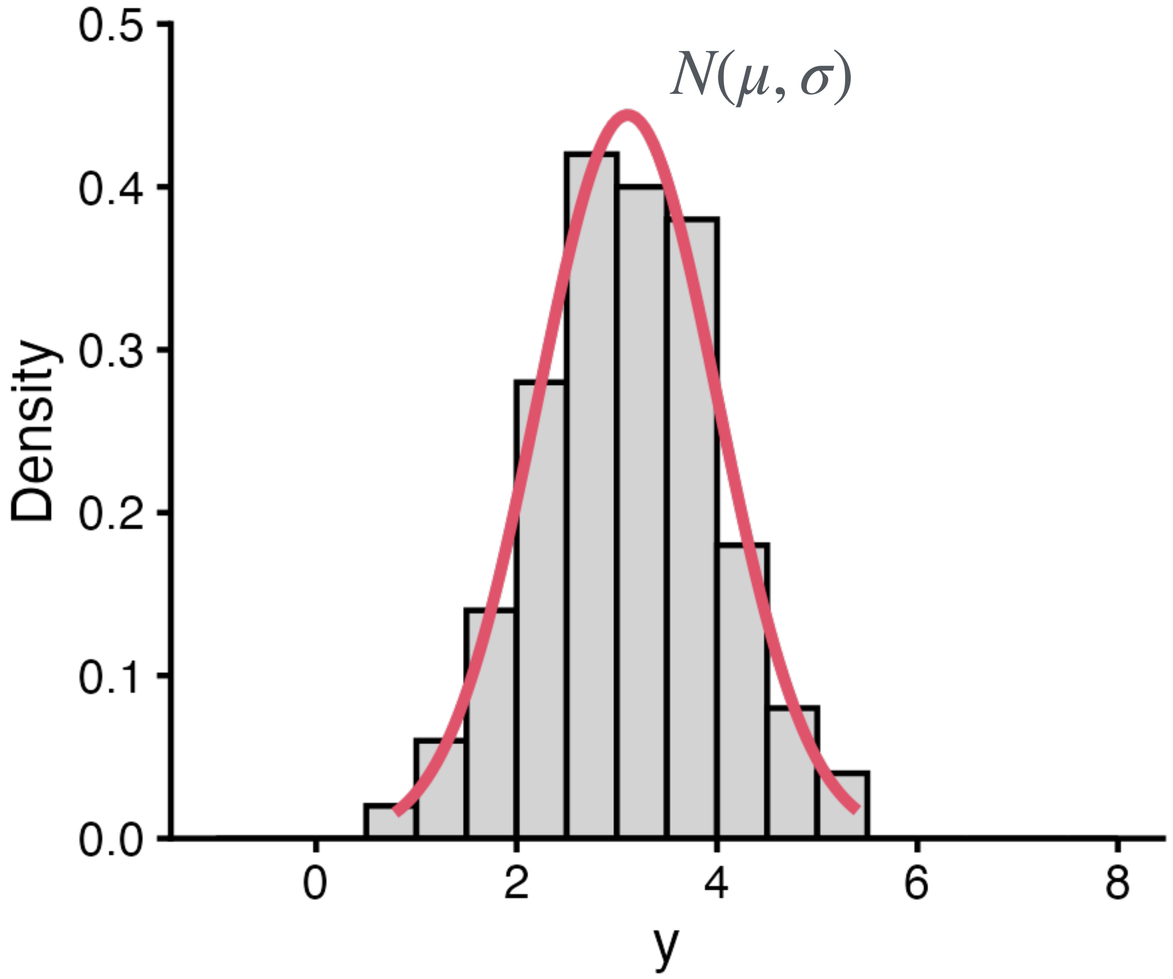


Simplest probabilistic model

Fit parameters to a set of measurements

- Measure a set of y_i values:
- Find the "best fitting" normal distribution by choosing μ and σ such that the N(μ, σ) distribution approximates the histogram of the y_i values

$$y_i \sim N(\mu, \sigma)$$



The likelihood

The probability of each value of y

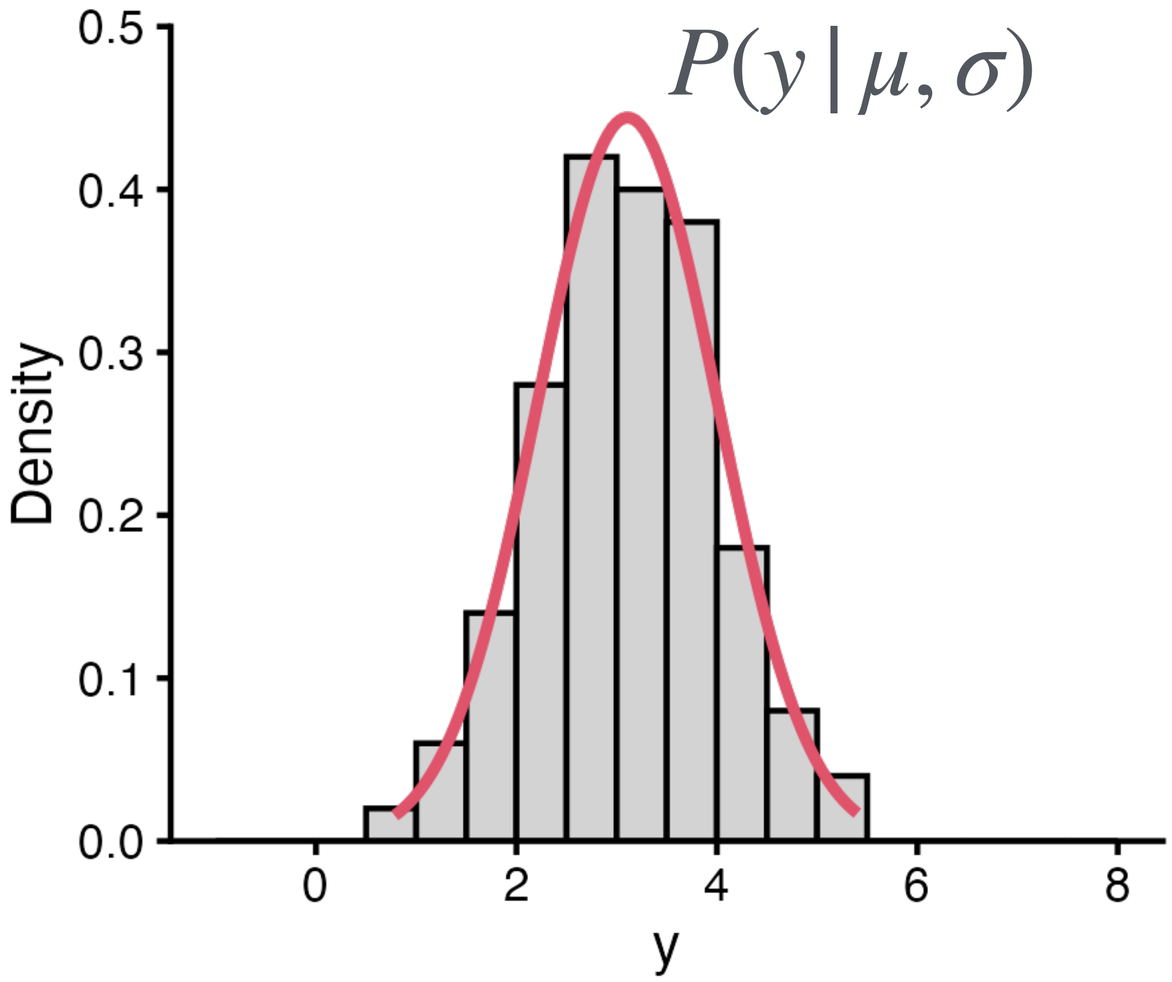
- What does this mean?

$$y_i \sim N(\mu, \sigma)$$

- We can also write this as:

$$P(y \mid \mu, \sigma)$$

The likelihood of y



The likelihood

The probability of each value of y

- What does this mean?

$$y_i \sim N(\mu, \sigma)$$

- We can also write this as:

$$P(y \mid \mu, \sigma)$$

The likelihood of y

- By the product rule:

$$P(\mu, \sigma \mid y) = \frac{P(y \mid \mu, \sigma)P(\mu, \sigma)}{P(y)}$$

- $P(\mu, \sigma) = P(\mu)P(\sigma)$: the prior distribution
- $P(\mu, \sigma \mid y)$: The posterior distribution
- $P(y)$: A constant, the “evidence"

The likelihood and friends

The probability of each value of y

- What does this mean?

$$y_i \sim N(\mu, \sigma)$$

- We can also write this as:

$$P(y \mid \mu, \sigma)$$

The likelihood of y

- By the product rule:

$$P(\mu, \sigma \mid y) \propto P(y \mid \mu, \sigma) P(\mu, \sigma)$$

- $P(\mu, \sigma) = P(\mu)P(\sigma)$: the prior distribution
- $P(\mu, \sigma \mid y)$: The posterior distribution
- ~~$P(y)$: A constant, the "evidence"~~ Not necessary for inference, usually ignored

The posterior distribution

The encoding of our inference

- We can use the posterior distribution to understand what out data says about our parameter values
- By finding the posterior $P(\mu, \sigma \mid y)$, we can infer the most probable values for the parameters.
- So, we just need to define the ingredients:
  - $P(\mu), P(\sigma), P(y_i \mid \mu, \sigma)$

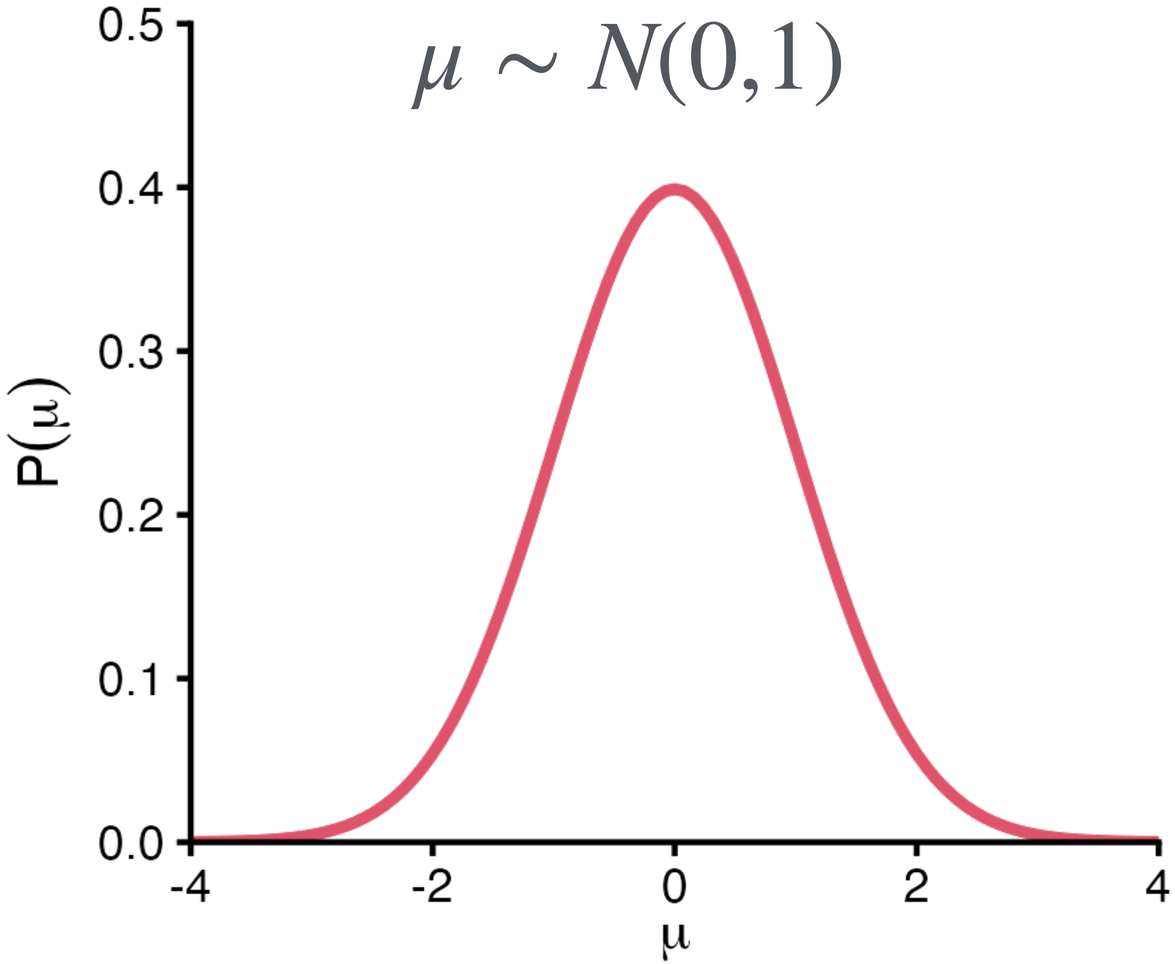
- A full model:

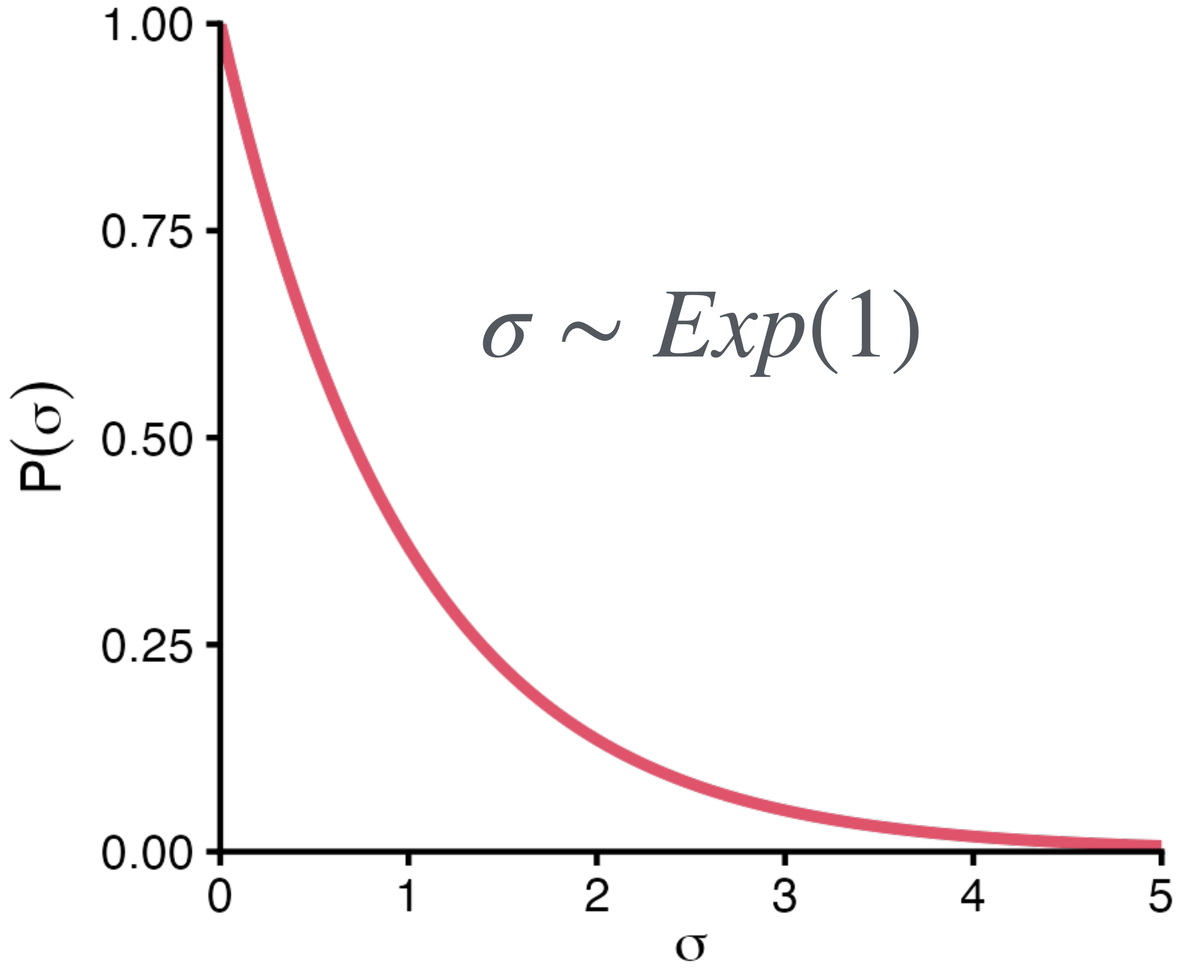
$$y_i \sim N(\mu, \sigma)$$
$$\mu \sim N(0,1)$$
$$\sigma \sim Exp(1)$$

The priors

The encoding of our guesses

$$y_i \sim N(\mu, \sigma)$$
$$\mu \sim N(0,1)$$
$$\sigma \sim Exp(1)$$





Fitting the model

We have pretty good computers

Math

$$y_i \sim N(\mu, \sigma)$$

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

$$\sigma \sim Exp(1)$$

R Code

```
library(rethinking)

df <- data.frame(y = rnorm(100, 3, 1))
fit <- ulam(alist(
          y ~ normal(mu, sigma),
          mu ~ normal(0, 1),
          sigma ~ exponential(1)),
          data = df)
```

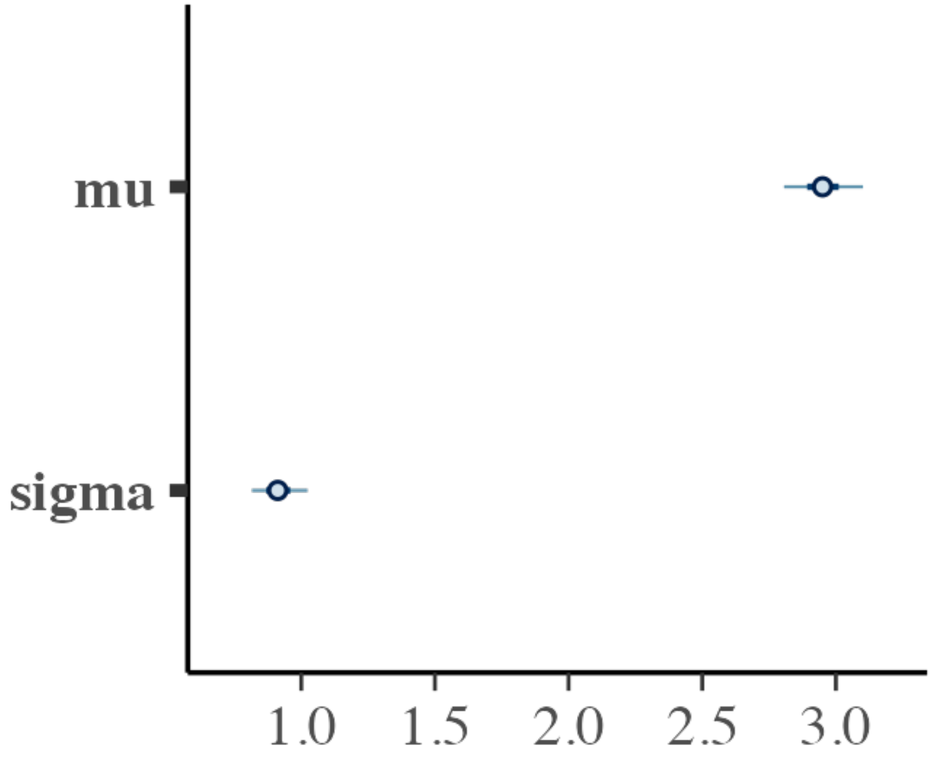
Model fit and parameter estimates

```
> precis(fit, prob = 0.95)
       mean   sd 2.5% 97.5% rhat ess_bulk
mu     3.09 0.09 2.91  3.26    1  1289.83
sigma 0.91 0.06 0.79  1.04    1  1416.83
```

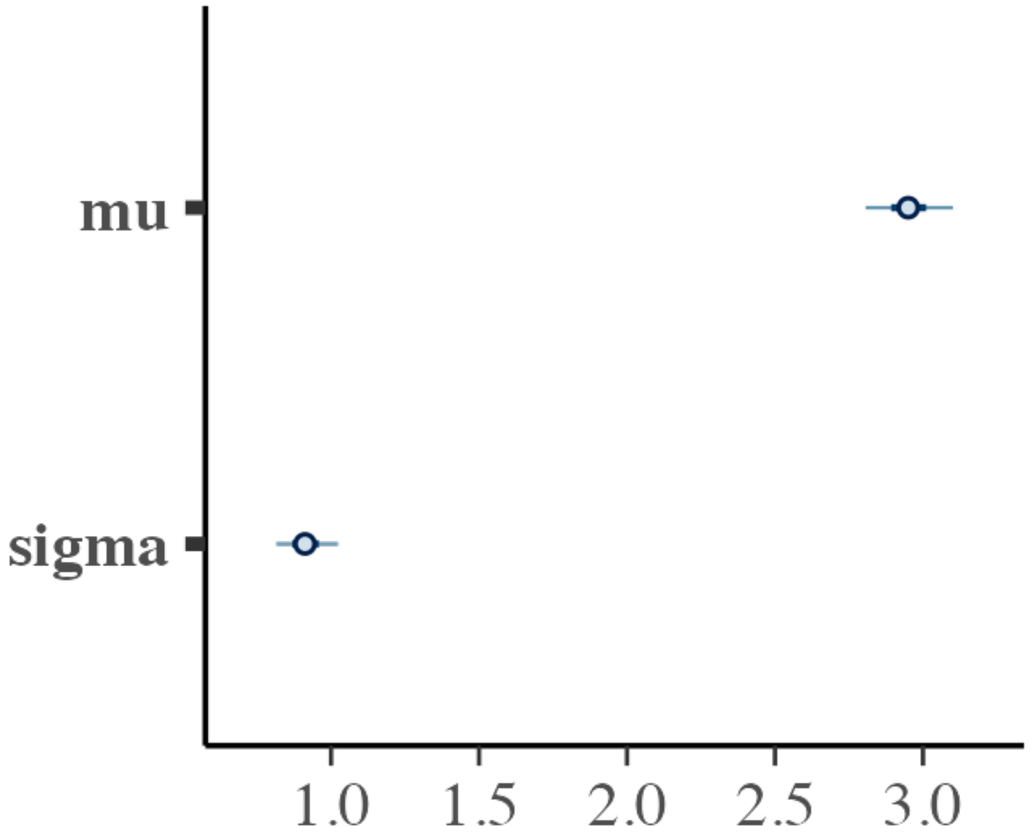
Estimates!

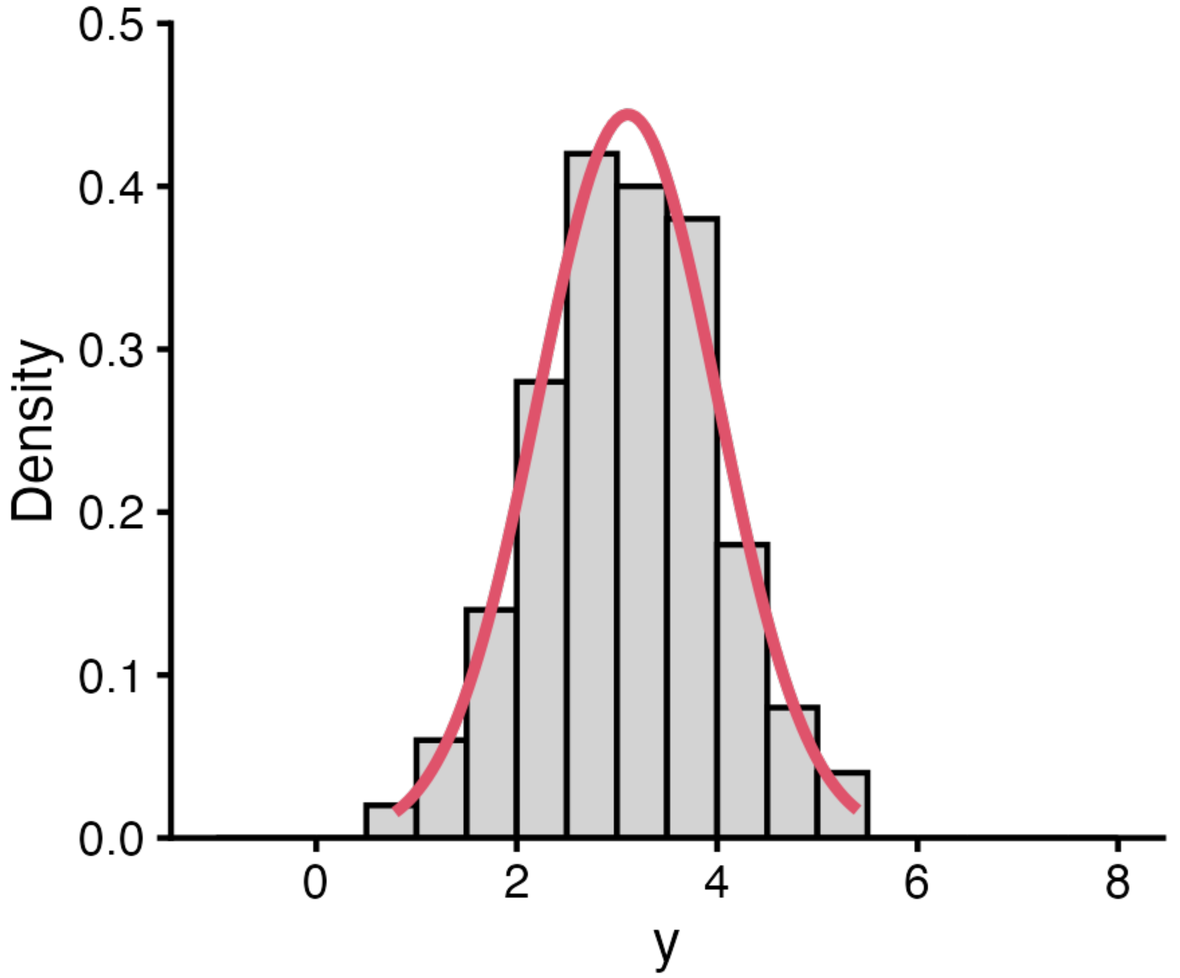
Confidence intervals

Model Diagnostics



Model fit and parameter estimates





The linear model

Adding more variables

- The main strategy for making useful probabilist models is to **allow the parameters to vary**
- If we have two measured variables:
  - y_i: the dependent variable, the outcome, the response, the predicted
  - x_i: the independent variable, the treatment, the control, the predictor

- We make the parameter μ a linear function of the predictor variable:

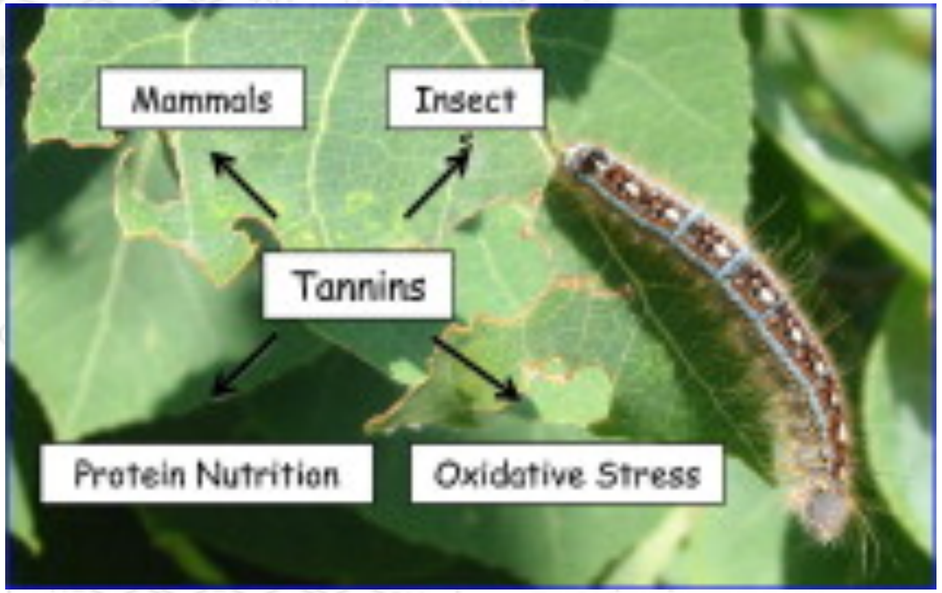
$$y_i \sim N(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta x_i$$

- This replaces μ with two new parameters:
  - α: the **intercept**
  - β: the **slope of x**

An ecological example

Tannins in plant–herbivore interactions

- tannin are the most abundant secondary metabolites in plants
- defense mechanism: defend leaves against herbivores by deterrence or toxicity
- tannins oxidize in insects, leading to oxidative stress



Our variables

- y_i: **caterpillar growth** - response variable
- x_i: **quantity of tannin in the caterpillar diet** - predictor variable,

Our variables

- y_i: **caterpillar growth** - response variable
- x_i: **quantity of tannin in the caterpillar diet** - predictor variable,

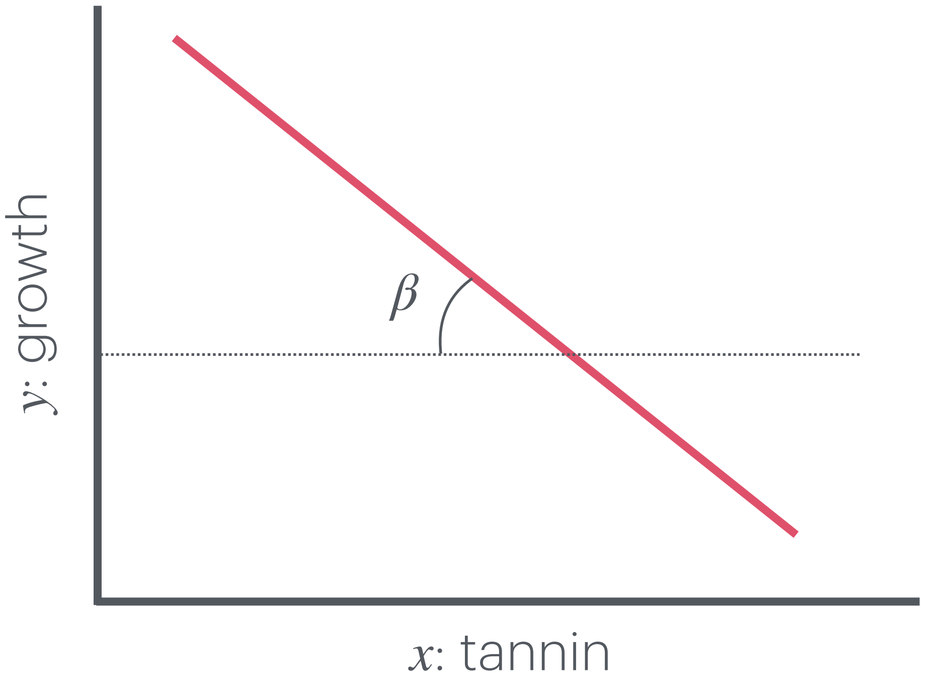
Lets ask a question

Do leaf chemical compounds reduce the growth of caterpillars?

- Our model

$$y_i \sim N(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta x_i$$
$$\alpha \sim N(0,1)$$
$$\beta \sim N(0,1)$$
$$\sigma \sim Exp(1)$$

The relation between growth and tannins is given by the slope parameter β



Model in the computer

Centering both variables is always a good idea

```
df <- data.frame(growth = c(12, 10, 8, 11, 6, 7, 2, 3, 3),
                 tannin = c(0, 1, 2, 3, 4, 5, 6, 7, 8))
df$tannin = scale(df$tannin, scale = FALSE)
df$growth = scale(df$growth, scale = FALSE)
fit  = ulam(alist(growth ~ normal(mu, sigma),
                  mu <- a + b*tannin,
                  a ~ normal(0, 1),
                  b ~ normal(0, 1),
                  sigma ~ exponential(1)),
            data = df)
```

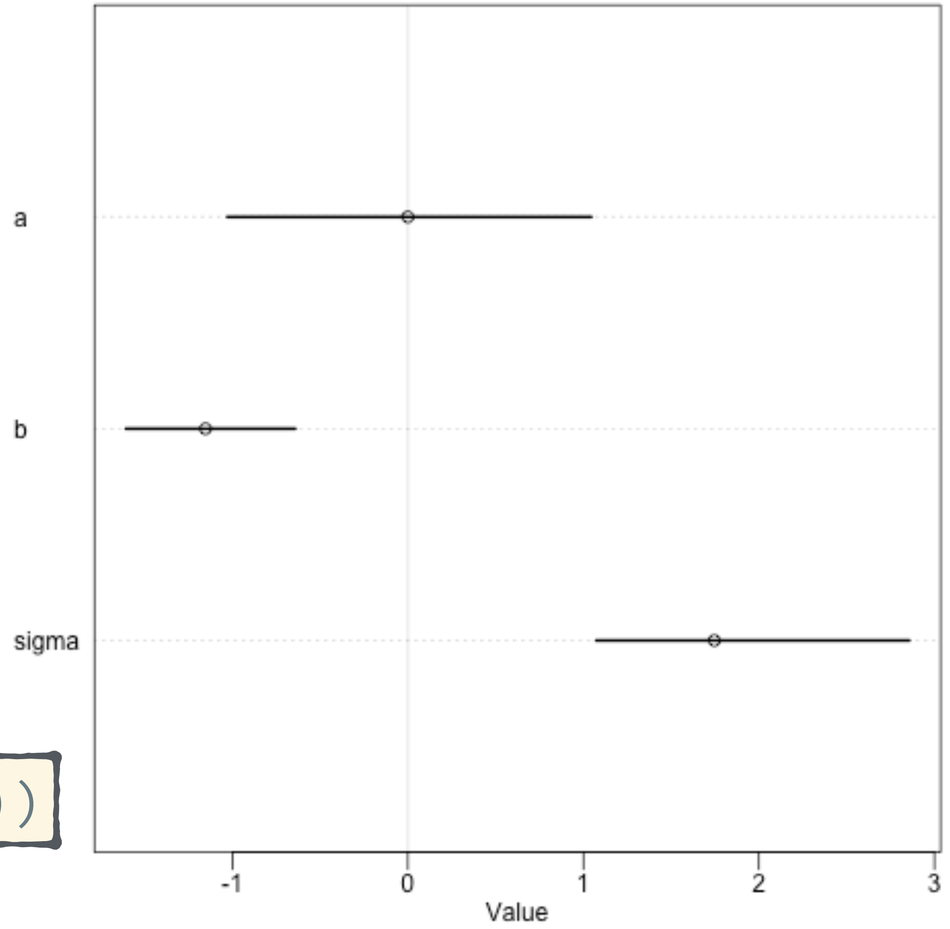
$$
\begin{aligned}
y_i &\sim N(\mu_i, \sigma) \\
\mu_i &= \alpha + \beta x_i \\
\alpha &\sim N(0,1) \\
\beta &\sim N(0,1) \\
\sigma &\sim Exp(1)
\end{aligned}
$$

Model fit

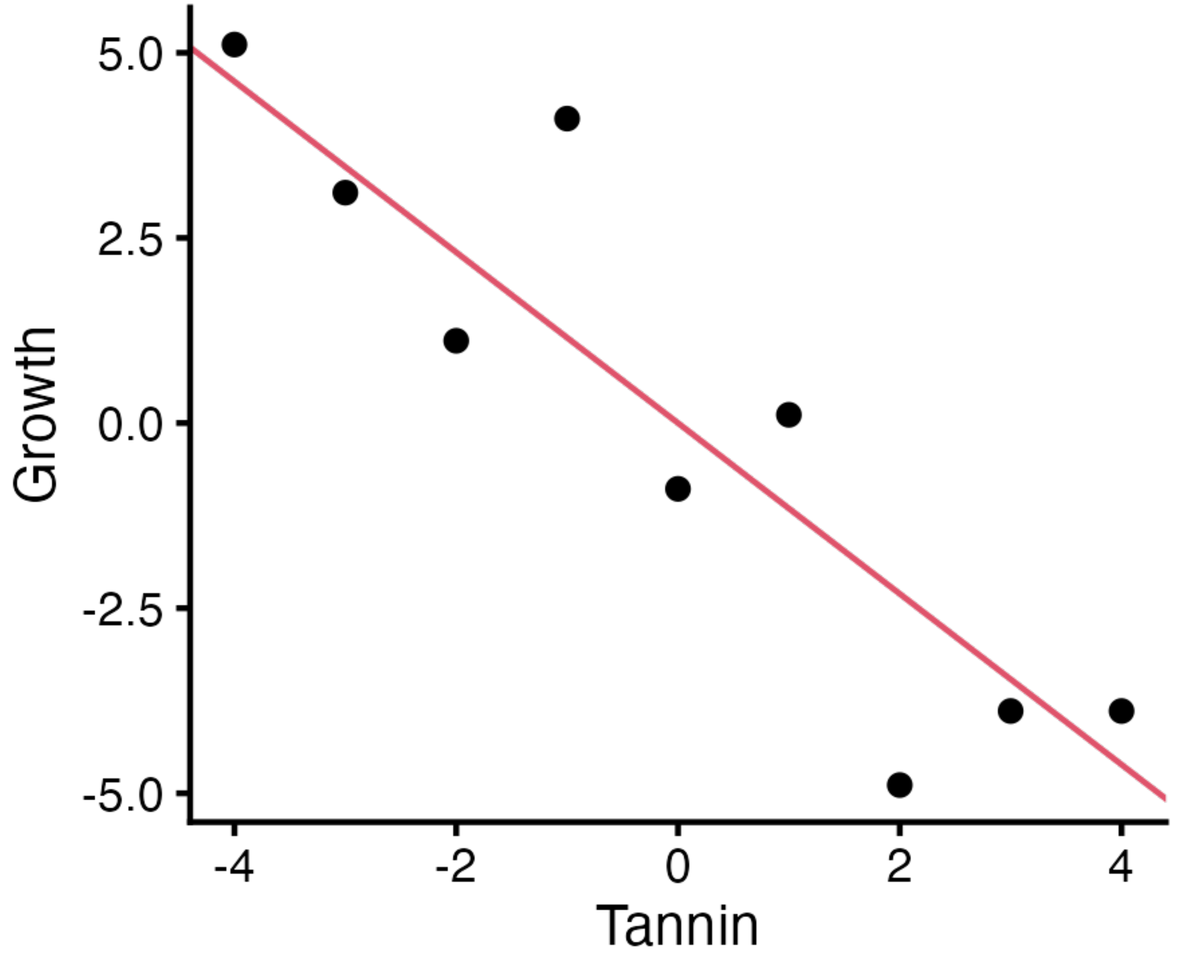
```
> precis(fit, prob = 0.95)
       mean   sd  2.5% 97.5% rhat ess_bulk
a      0.00 0.52 -1.03  1.04 1.00  1468.82
b     -1.15 0.24 -1.61 -0.64 1.01  1056.27
sigma  1.75 0.46  1.07  2.86 1.00  1195.20
```

Estimates!

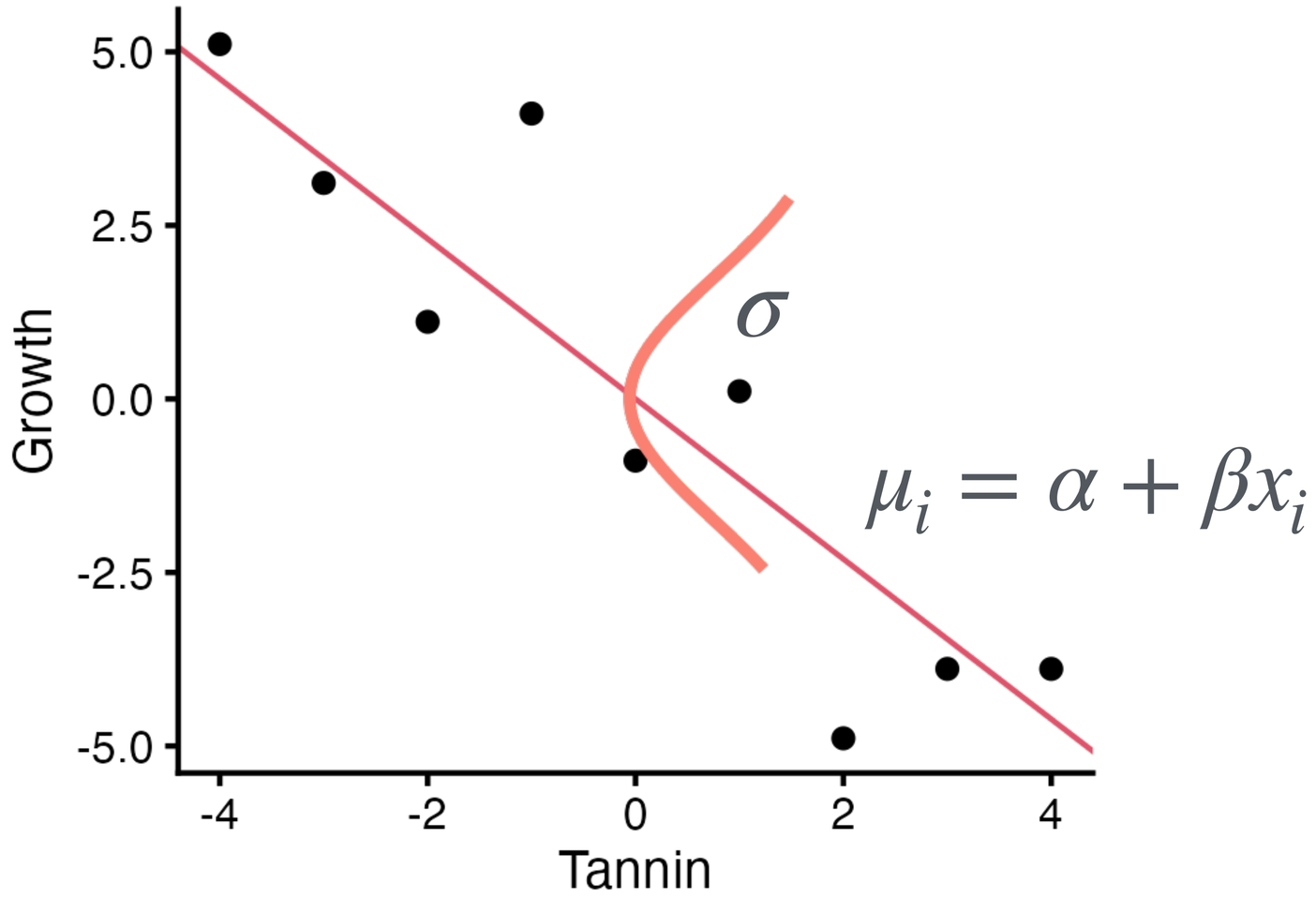
```
plot(precis(fit, prob = 0.95))
```



Model plot



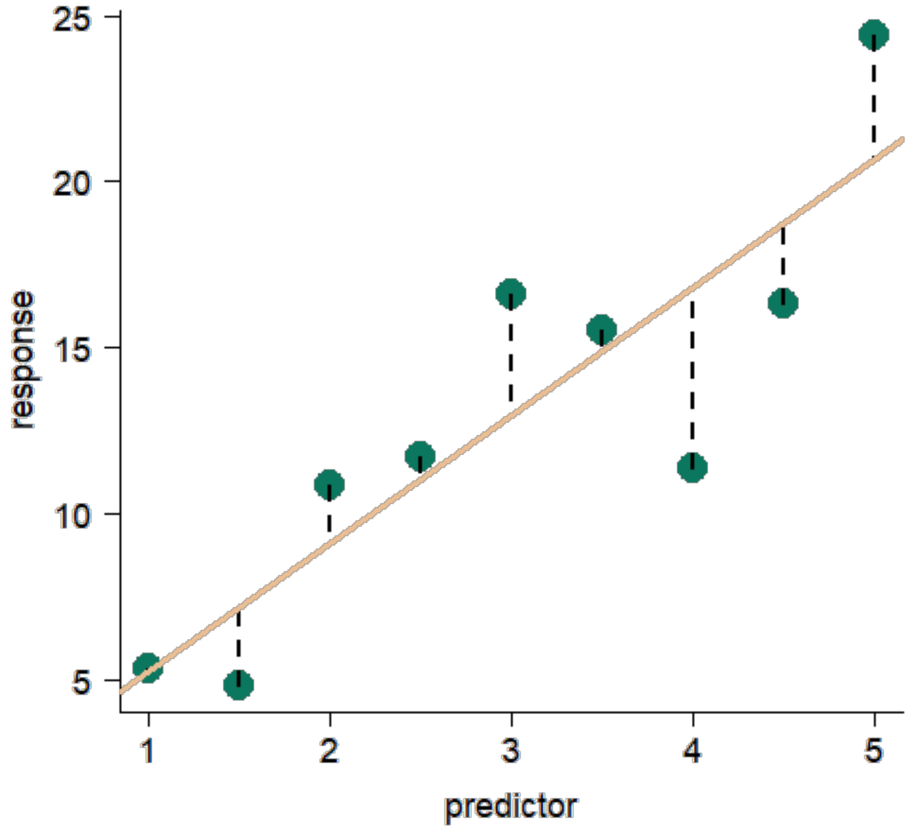
Model plot



Other ways of fitting the linear model

OLS and Maximum Likelihood

- The standard linear model has many flavors and justifications
- Ordinary Least Squares (OLS) is the most common introduction, and consists of minimizing the squared distance between observations and the regression line
- Maximum likelihood (ML) looks for the parameters that maximize the probability of observing y_i:
  - $P(y_i | \theta = \{\alpha, \beta, \sigma\})$
  - ML finds the same solution as OLS under a gaussian model

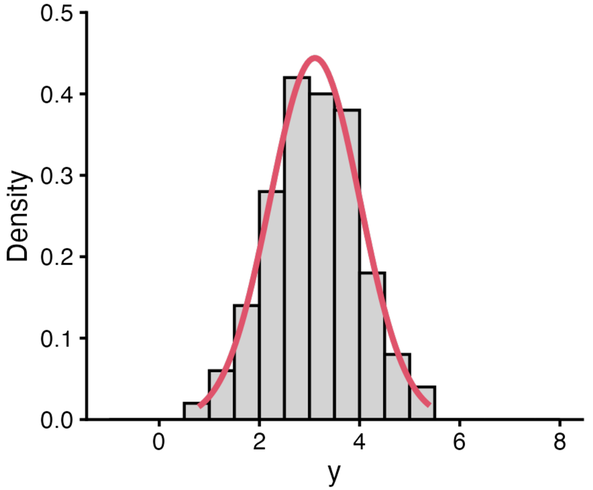


Other ways of fitting the linear model

lm() function for linear models

- The lm() function in R can fit most of the models we saw with OLS using a formula notation
- OLS assumes fixed uniform priors, so we can't change them

$$y_i \sim N(\mu_i, \sigma)$$
$$\mu_i = \alpha$$




```
> df <- data.frame(y = rnorm(100, 3, 1))

# OLS model:
> ols_fit = lm(y ~ 1, data = df)
> precis(ols_fit, prob = 0.95)
             mean  sd 2.5% 97.5%
(Intercept) 3.03 0.1 2.83  3.24
> (summary(ols_fit)$sigma)
[1] 1.037191
```

α

σ

Other ways of fitting the linear model

lm() function for linear models

$$y_i \sim N(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

```
> df <- data.frame(growth = c(12, 10, 8, 11, 6, 7, 2, 3, 3),
                   tannin = c(0, 1, 2, 3, 4, 5, 6, 7, 8))
> df$tannin = scale(df$tannin, scale = FALSE)
> df$growth = scale(df$growth, scale = FALSE)

> ols_fit = lm(growth ~ tannin, data = df)
> precis(ols_fit)
               mean   sd  5.5% 94.5%
(Intercept)    0.00 0.56 -0.90  0.90
tannin        -1.22 0.22 -1.57 -0.87
```

α

β

Other ways of fitting the linear model

stan_glm() function for linear models and priors!

$$y_i \sim N(\mu_i, \sigma)$$

$$\mu_i = \alpha + \beta x_i$$

And some standard priors...

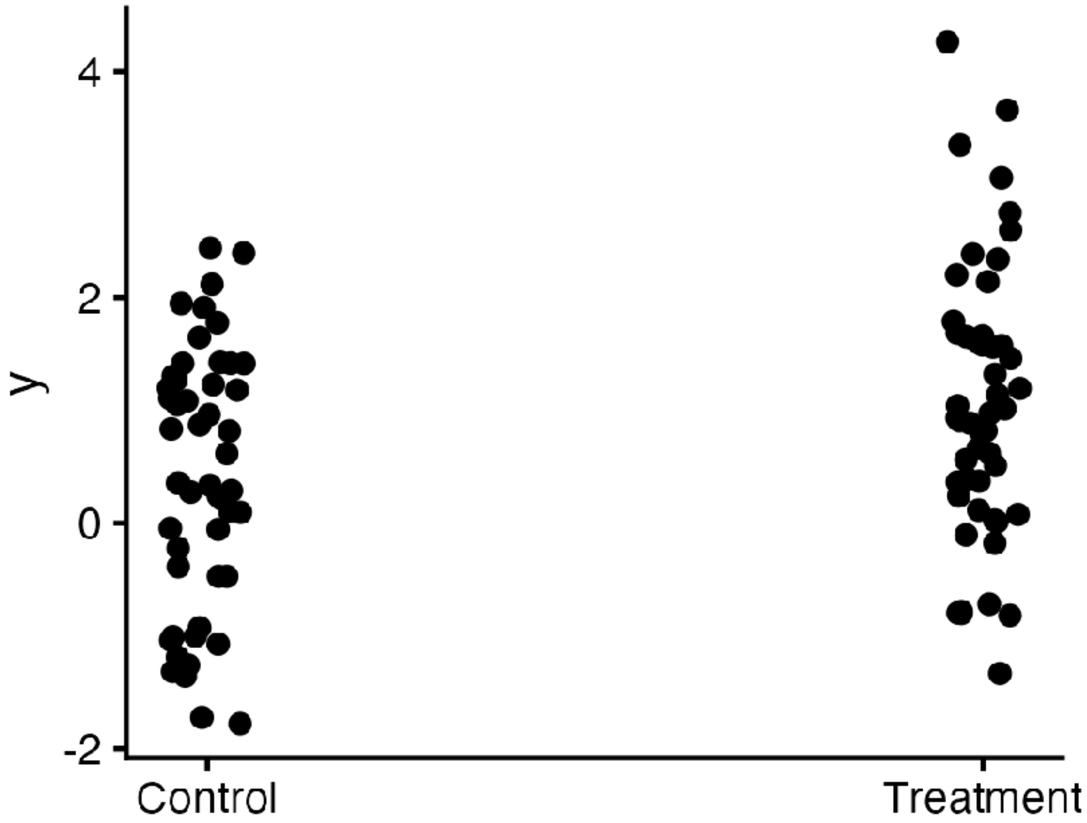
```
> sglm_fit = stan_glm(growth ~ tannin, data = df, cores = 4)

> summary(sglm_fit, probs = c(0.025, 0.975))[, 1:7]
                            mean        mcse        sd       2.5%       97.5% n_eff
(Intercept) α -0.01069275 0.015974907 0.6971944  -1.403377   1.3696905  1905
tannin      β -1.21608408 0.005609107 0.2482728  -1.716820  -0.7229236  1959
sigma          1.98129172 0.016244872 0.6447948   1.145132   3.5859302  1575
mean_PPD      -0.01273309 0.020249189 0.9958192  -2.030730   2.0076369  2418
log-posterior -23.56539992 0.046443876 1.4480299 -27.365501 -21.9264227   972
                   Rhat
(Intercept)    1.000676
tannin         1.001776
sigma          1.000847
mean_PPD       1.000271
log-posterior  1.003540
```

What about categorical predictors?

Linear regression is flexible

- Our questions are frequently based on categories:
  - Is a treatment effective in improving outcomes?
  - Are two geographical regions different in some aspect?
  - Does the diet of a group of species affect their size?



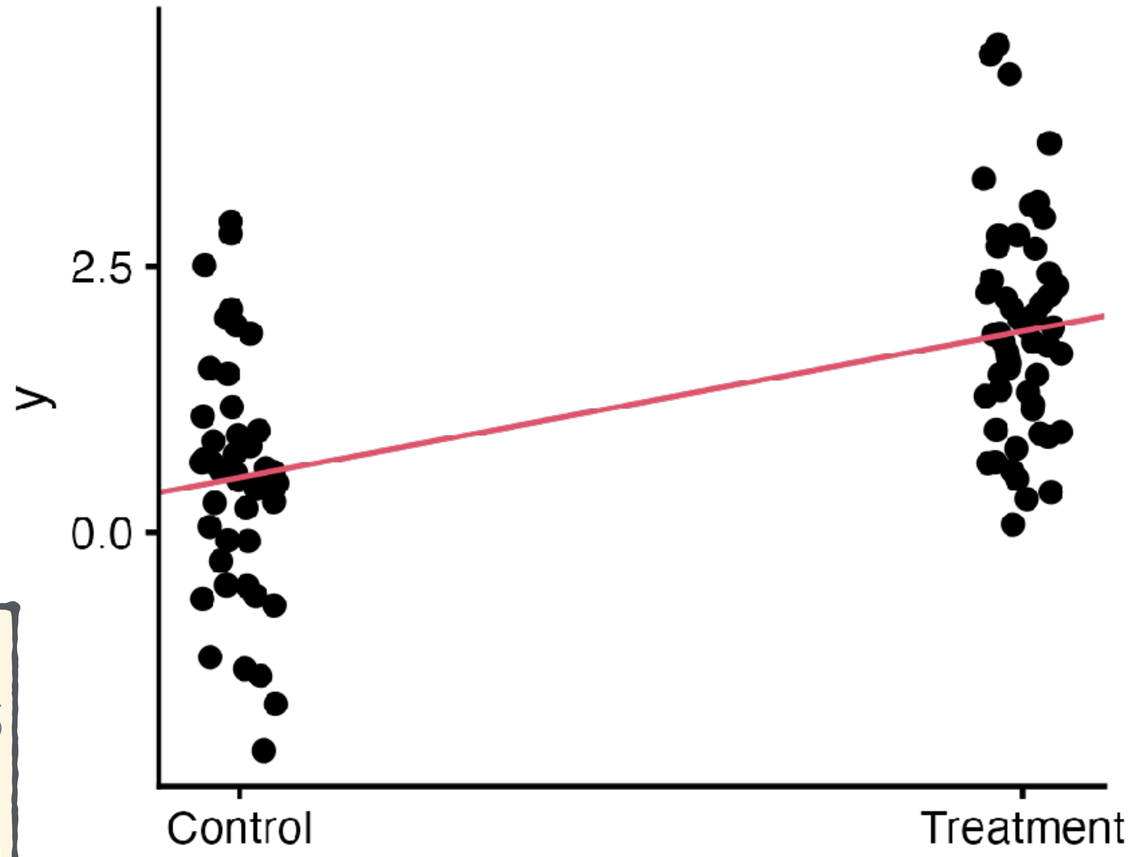
Binary predictor, same model

Control-treatment, two categories...

$$y_i \sim N(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta x_i$$

- x_i: 0 for control, 1 for treatment
- α: the intercept is the **mean of the Control** group
- β: the slope is the **difference in the means across the groups**

```
> precis(fit, prob = 0.95)
      mean   sd 2.5% 97.5%
a     0.52 0.16 0.20  0.82
b     1.37 0.22 0.96  1.79
sigma 1.08 0.08 0.94  1.23
```



How about more categories?

- There are a few ways of modeling predictors with many categories:
  - **Contrasts**: Each category is compared to a **baseline**, and the coefficients are comparisons between baseline and levels
  - **One-hot**: coefficients are means of each level of the predictor
  - **Residuals**: an overall mean is measured, and coefficients are differences between each level and the global mean

```
> x = sample(LETTERS[1:3], 9, replace = TRUE)
> y = 1 + ifelse(x == "A", 0,
           ifelse(x == "B", 1, 2)) + rnorm(9)
> df = tibble(y, x)
> df
# A tibble: 9 × 2
          y x
      <dbl> <chr>
1   1.25    B
2   0.870   A
3  -0.00180 A
4   1.18    B
5   2.03    C
6   2.60    B
7   2.55    B
8   3.92    C
9   4.66    B
```

How about more categories?

Contrasts, the default in most lm() functions

- There are a few ways of modeling predictors with many categories:
  - **Contrasts**: Each category is compared to a **baseline**, and the coefficients are comparisons between baseline and levels
  - **One-hot**: coefficients are means of each level of the predictor
  - **Residuals**: an overall mean is measured, and coefficients are differences between each level and the global mean

```
> fit_contrasts = stan_glm(y ~ x, data = df)
> summary(fit_contrasts)[1:3, 1:3]
             mean mcse   sd
(Intercept) 0.72    0 0.13
xB          1.32    0 0.20
xC          2.20    0 0.186
```

Q: How do we get estimates for the mean in each class?

How about more categories?

One-hot

- There are a few ways of modeling predictors with many categories:
  - **Contrasts**: Each category is compared to a **baseline**, and the coefficients are comparisons between baseline and levels
  - **One-hot**: coefficients are means of each level of the predictor
  - **Residuals**: an overall mean is measured, and coefficients are differences between each level and the global mean

```
> x
[1] "B" "A" "A" "B" "C" "B" "B" "C" "B"
> onehot
  xA xB xC
1  0  1  0
2  1  0  0
3  1  0  0
4  0  1  0
5  0  0  1
6  0  1  0
7  0  1  0
8  0  0  1
9  0  1  0
```

How about more categories?

One-hot

- There are a few ways of modeling predictors with many categories:
  - **Contrasts**: Each category is compared to a **baseline**, and the coefficients are comparisons between baseline and levels
  - **One-hot**: coefficients are means of each level of the predictor
  - **Residuals**: an overall mean is measured, and coefficients are differences between each level and the global mean

```
> onehot = model.matrix(~0+x, data = df)
> fit_onehot = stan_glm(y ~ 0 + onehot, data =
df, cores = 4)
> summary(fit_onehot)[1:3, 1:3]
          mean mcse   sd
onehotxA 0.72    0 0.13
onehotxB 2.04    0 0.14
onehotxC 2.93    0 0.12
```

How about more categories?

One-hot

- There are a few ways of modeling predictors with many categories:
  - **Contrasts**: Each category is compared to a **baseline**, and the coefficients are comparisons between baseline and levels
  - **One-hot**: coefficients are means of each level of the predictor
  - **Residuals**: an overall mean is measured, and coefficients are differences between each level and the global mean

```
> onehot = model.matrix(~0+x, data = df)
> fit_onehot = stan_glm(y ~ 0 + onehot, data =
df, cores = 4)
> summary(fit_onehot)[1:3, 1:3]
          mean mcse   sd
onehotxA 0.72    0 0.13
onehotxB 2.04    0 0.14
onehotxC 2.93    0 0.12
```

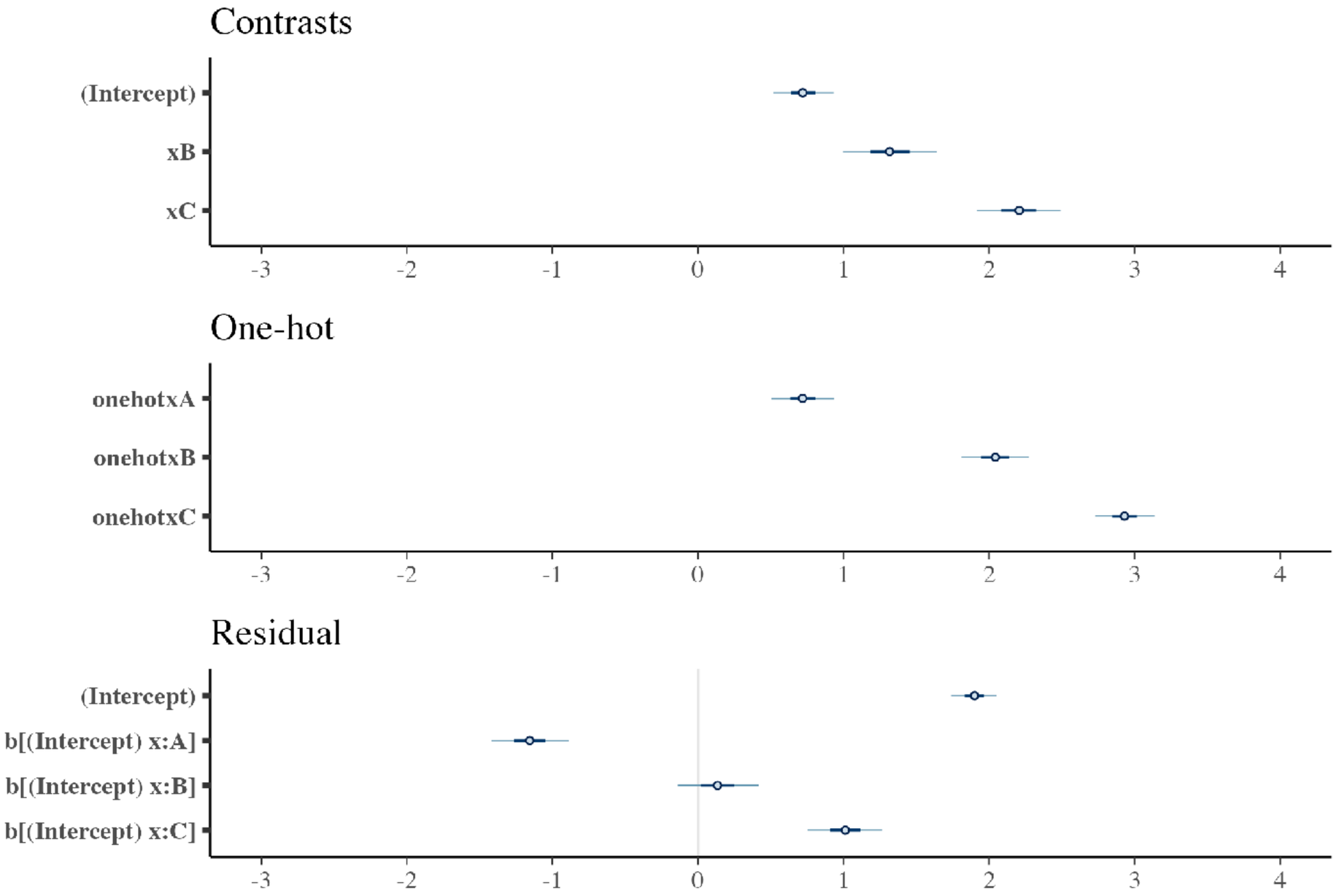
```
> fit_contrasts = stan_glm(y ~ x, data = df)
> summary(fit_contrasts)[1:3, 1:3]
             mean mcse   sd
(Intercept) 0.72    0 0.13
xB          1.32    0 0.20
xC          2.20    0 0.186
```

How about more categories?

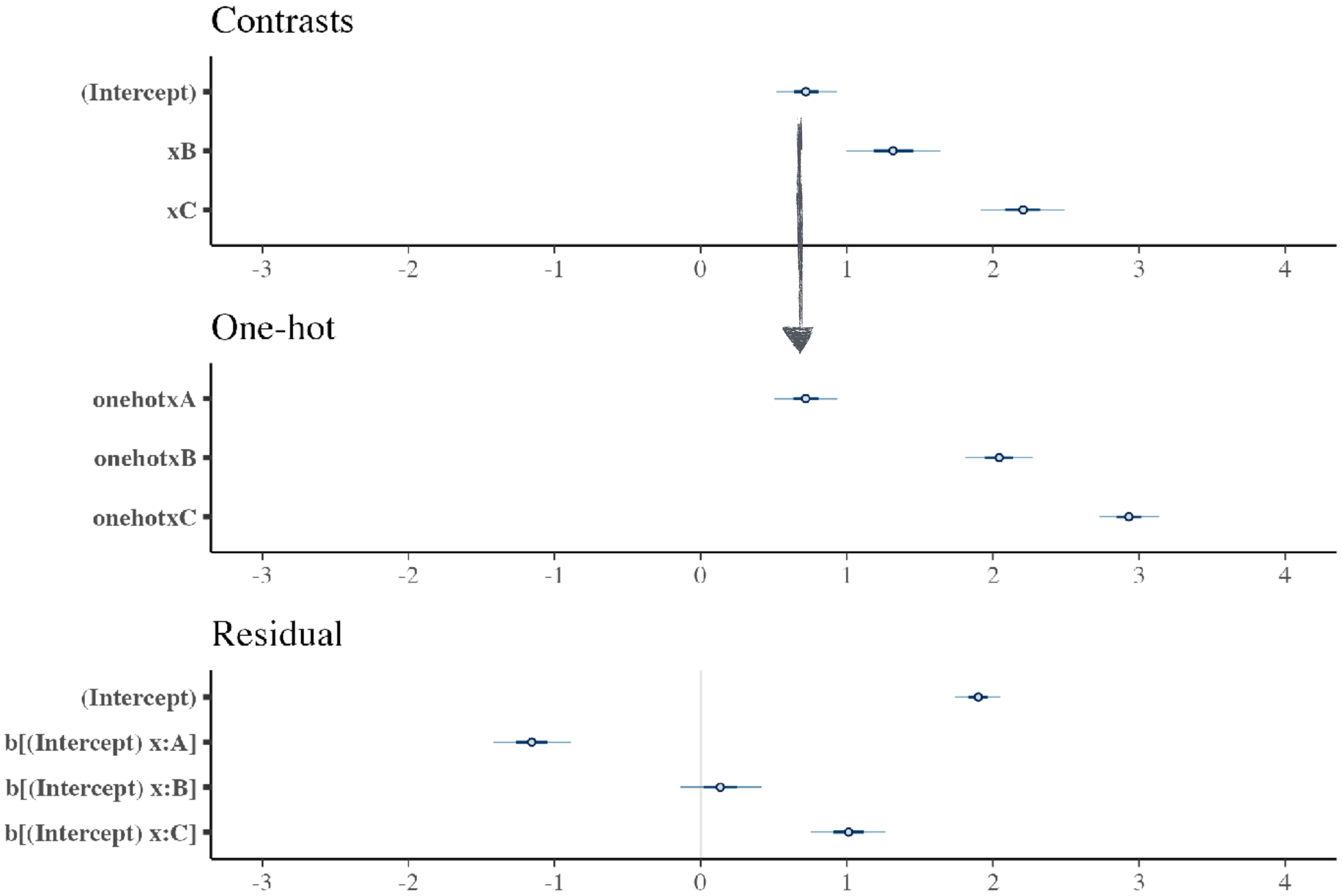
- There are a few ways of modeling predictors with many categories:
  - **Contrasts**: Each category is compared to a **baseline**, and the coefficients are comparisons between baseline and levels
  - **One-hot**: coefficients are means of each level of the predictor
  - **Residuals**: an overall mean is measured, and coefficients are differences between each level and the global mean

```
> fit_residual = stan_glmer(y ~ 1 + (1|x),
data = df, prior_intercept =
normal(mean(df$y), 0.1))
> summary(fit_residual)[1:4, 1:3]
                      mean mcse   sd
(Intercept)           1.90    0 0.10
b[(Intercept) x:A] -1.16    0 0.16
b[(Intercept) x:B]  0.14    0 0.17
b[(Intercept) x:C]  1.01    0 0.15
```

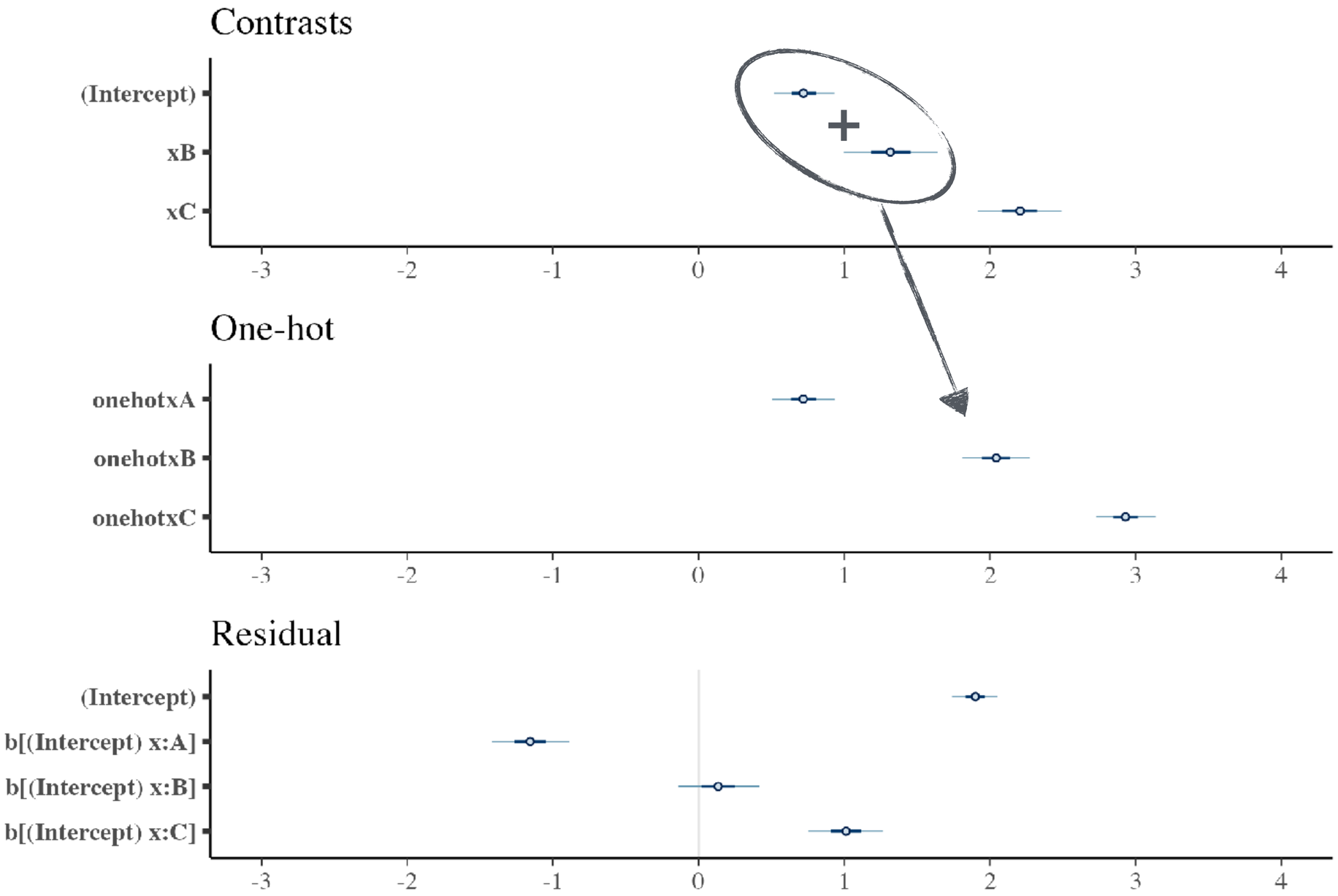
Comparing estimates



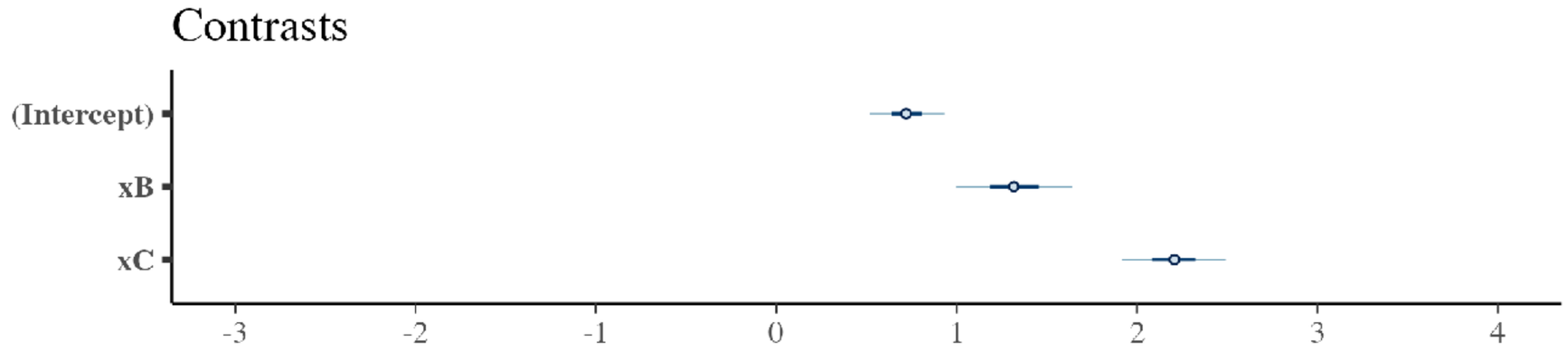
Comparing estimates

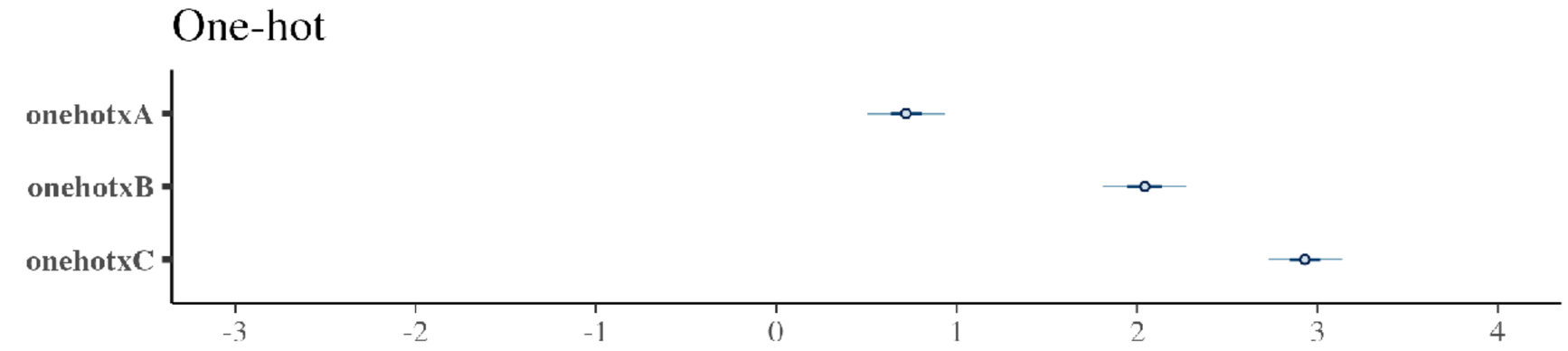


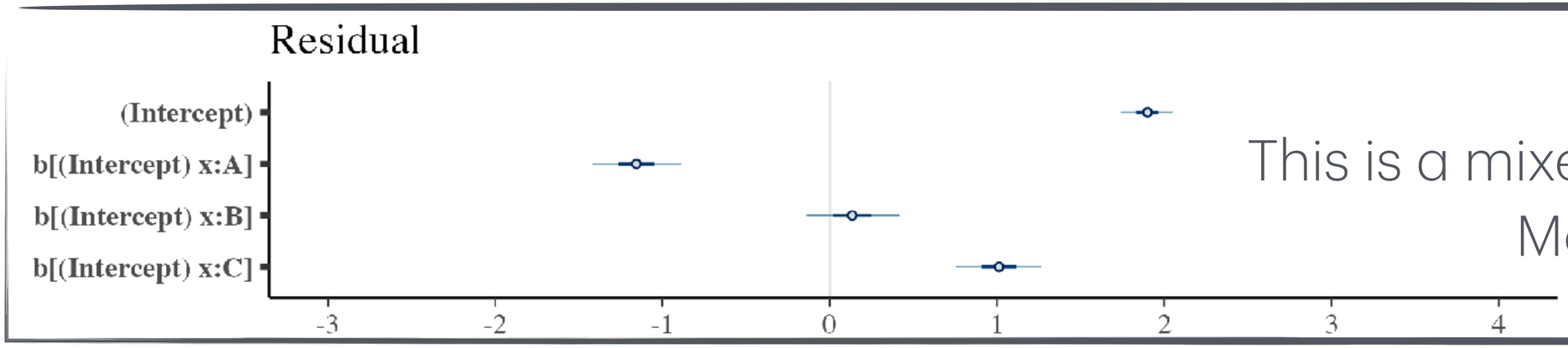
Comparing estimates



Comparing estimates







This is a mixed model!
More later...

Summary

- Linear models help us **answer questions** about differences across groups
- The basic strategy is to formulate a **probabilistic description** of our data, and to establish a **relation between data and parameters** in the statistics model
- In linear models, coefficients are **comparisons across categories**, and we can interpret their estimates

- There are many ways to fit the models
- We like to encode the information the data brings about parameters using the **posterior distribution**, and this requires the definition of **priors**
- We can also fit the models using OLS and Maximum Likelihood, but if we do, we are stuck with default priors that don't bring useful information