8. Assumptions and checks: linear regression

Frank Edwards

School of Criminal Justice, Rutgers - Newark

Today's example

Do tall people make more money than short people?

```
# earnings and height data
dat <- read_csv("./data/earnings.csv")

# regression for the day
m0 <- lm(earn ~ height, data = dat)

## output
summary(m0)
```

```
##
## Call:
## lm(formula = earn ~ height, data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -31405 -12456  -3645   6570 370190
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -85027.3     8860.7  -9.596   <2e-16 ***
## height        1595.0      132.9  12.003   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21690 on 1814 degrees of freedom
## Multiple R-squared:  0.07357,    Adjusted R-squared:  0.07306
## F-statistic: 144.1 on 1 and 1814 DF,  p-value: < 2.2e-16
```

Regression assumptions: 1. Validity

Your domain expertise is key here!

- Does your theoretical construct map onto your measures?
- Have you included important covariates / predictors?

Regression assumptions: 1. Validity

Your domain expertise is key here!

- Does your theoretical construct map onto your measures?
- Have you included important covariates / predictors?

Tough to comment on validity for this example...

Regression assumptions: 2 Representativeness

Your domain expertise is key here!

- Does your sample generalize to the population of interest?
- Carefully consider the structure of your inference

Regression assumptions: 2 Representativeness

Your domain expertise is key here!

- Does your sample generalize to the population of interest?
- Carefully consider the structure of your inference

What would we need to have this analysis generalize?

Regression assumptions: 3 Correct functional form

In linear regression, we assume that x predicts y through an additive linear functional form.

Regression assumptions: 3 Correct functional form

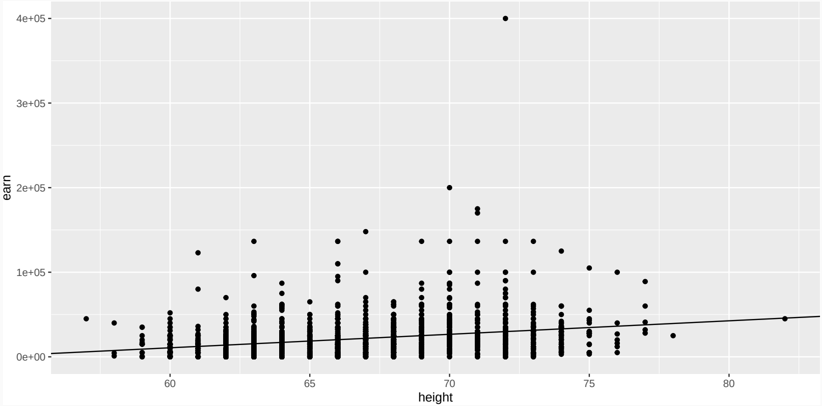
In linear regression, we assume that x predicts y through an additive linear functional form.

What does this mean in our example?

$$earn_i = -85000 + 1595 height_i + \varepsilon_i$$

Checking linearity assumptions: visual inspection

```
betas <- coef(m0)
ggplot(dat, aes(x = height, y = earn)) + geom_point() + geom_abline(intercept = betas[1],
    slope = betas[2])
```



Regression assumptions: 4 - 6 iid Normal errors

We assume that the error terms are *iid*, *independent* and *identically distributed*.

We also assume that they are Normally distributed: $\varepsilon \sim N(0, \sigma^2)$

Independence of errors

We assume that error terms are uncorrelated with each other.

This is nearly always violated when:

1. Individuals are measured multiple times (longitudinal data)
2. Individuals are clustered in groups (multilevel data)
3. Measurements are grouped by place (spatial data)

Identically distributed errors

We assume that all errors follow the same distribution, which means constant location and constant variance. Also known as homoskedasticity.

We assume that $\varepsilon \sim N(0, \sigma^2)$

Identically distributed errors

We assume that all errors follow the same distribution, which means constant location and constant variance. Also known as homoskedasticity.

We assume that $\varepsilon \sim N(0, \sigma^2)$

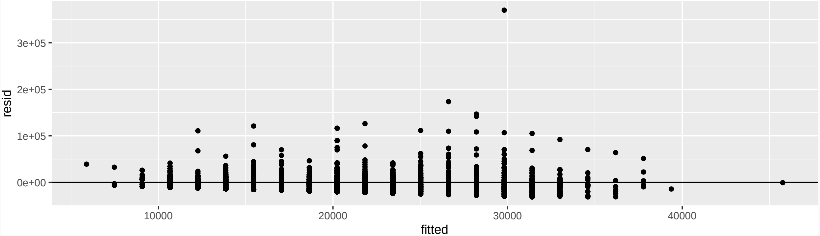
This assumption about the *stochastic* component of the model impacts the posterior predictive distribution, but has little impact on the *deterministic* component of our model.

Checking for heteroskedasticity and Normality

Here, a residuals vs fitted plot is a perfect test

```
plot_dat <- data.frame(resid = residuals(m0), fitted = fitted(m0))

ggplot(plot_dat, aes(x = fitted, y = resid)) + geom_point() + geom_abline(intercept = 0,
    slope = 0)
```



Let's evaluate model fit

$$R^2 = 1 - (\sigma^2 / s_y^2)$$

```r
sigma <- summary(m0)$sigma
s_y <- sd(dat$earn)

1 - sigma^2/s_y^2
```

```
## [1] 0.07306251
```

Let's evaluate model fit

$$R^2 = 1 - (\sigma^2 / s_y^2)$$

```
sigma <- summary(m0)$sigma
s_y <- sd(dat$earn)

1 - sigma^2/s_y^2
```

```
## [1] 0.07306251
```

Or we could just use `summary()`

Another route: Posterior predictive checks

Now that we are using Bayesian models, we can directly simulate model predictions.

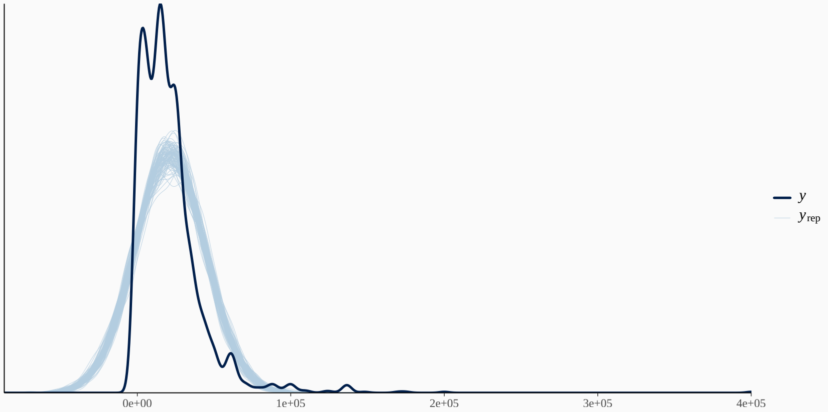
We use the observed data, then simulate predictions using the observed data. This helps us evaluate if the model produces predictions that look similar to the observed data.

```
# Use refresh=0 to suppress the sampling messages
m0_b <- stan_glm(earn ~ height, data = dat, refresh = 0)
m0_preds <- posterior_predict(m0_b)
```

Evaluating the data against the simulations

What do you think?

```r
library(bayesplot)
# ppc_dens_overlay from bayesplot is nice!  the first 100 sims will be adequate
# here
ppc_dens_overlay(dat$earn, m0_preds[1:100, ])
```

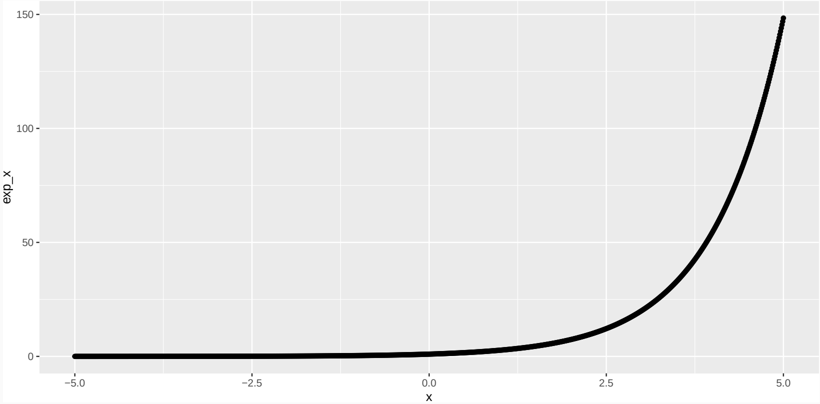


Problems

1. Our model predicts negative incomes. This never occurs in the data
2. Our model doesn't predict very high incomes nearly as often as they occur in the data

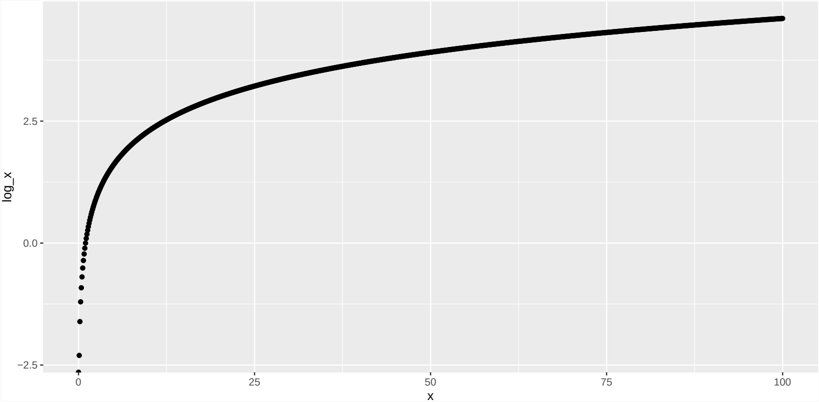
Exponentials

```
plot_dat <- data.frame(x = seq(-5, 5, by = 0.01), exp_x = exp(seq(-5, 5, by = 0.01)))
ggplot(plot_dat, aes(x = x, y = exp_x)) + geom_point()
```



Logarithms

```r
plot_dat <- data.frame(x = seq(0, 100, by = 0.1), log_x = log(seq(0, 100, by = 0.1)))
ggplot(plot_dat, aes(x = x, y = log_x)) + geom_point()
```



Log transformations for regression

- Do you have strictly postive data?
- Do you have a distribution with some very extreme values?
- Do you suspect linearity is not reasonable?

- Do you have strictly postive data?
- Do you have a distribution with some very extreme values?
- Do you suspect linearity is not reasonable?

Consider a transformation!

Log transforming the outcome in a regression

We take the log of the left-hand side

$$log(earn_i) = \beta_0 + \beta_1 height_i + \varepsilon_i$$

Log transforming the outcome in a regression

We take the log of the left-hand side

$$log(earn_i) = \beta_0 + \beta_1 height_i + \varepsilon_i$$

To interpret on the original scale, we exponentiate both sides

$$earn_i = e^{\beta_0 + \beta_1 height_i + \varepsilon_i}$$

Log transforming the outcome in a regression

We take the log of the left-hand side

$$log(earn_i) = \beta_0 + \beta_1 height_i + \varepsilon_i$$

To interpret on the original scale, we exponentiate both sides

$$earn_i = e^{\beta_0 + \beta_1 height_i + \varepsilon_i}$$

Which transforms our additive equation into a multiplicative equation

$$earn_i = e^{\beta_0} e^{\beta_1 x_i} e^{\varepsilon_i}$$

Estimating the model

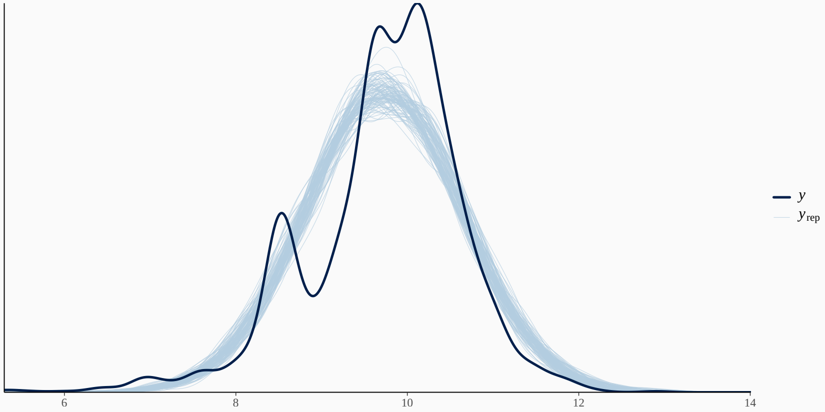
R can handle transformations within the formula, but we've got some zeroes that need to go. (try `log(0)` and see what happens)

```
m1_b <- stan_glm(log(earn) ~ height, data = dat %>%
    filter(earn > 0), refresh = 0)
m1_b
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      log(earn) ~ height
##  observations: 1629
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 5.9    0.4
## height      0.1    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.9    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

So what impacts did it have?

```
earn_nozeroes <- dat %>%
    filter(earn > 0)
m1_preds <- posterior_predict(m1_b)
ppc_dens_overlay(log(earn_nozeroes$earn), m1_preds[1:100, ])
```



What does the intercept mean?

```
m1_b
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      log(earn) ~ height
##  observations: 1629
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 5.9    0.4   
## height      0.1    0.0   
## 
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.9    0.0   
## 
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

Expected log earnings when height is zero. That's not helpful...

Centering

We can *center* a variable to improve interpretation

Centering

We can *center* a variable to improve interpretation

Let's center height at the mean

```
# I() forces R to evaluate a math expression in a formula
m2_b <- stan_glm(log(earn) ~ I(height - mean(dat$height)), data = dat %>%
    filter(earn > 0), refresh = 0)
m2_b
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      log(earn) ~ I(height - mean(dat$height))
##  observations: 1629
##  predictors:   2
## ------
##                              Median MAD_SD
## (Intercept)                  9.7    0.0
## I(height - mean(dat$height)) 0.1    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.9    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

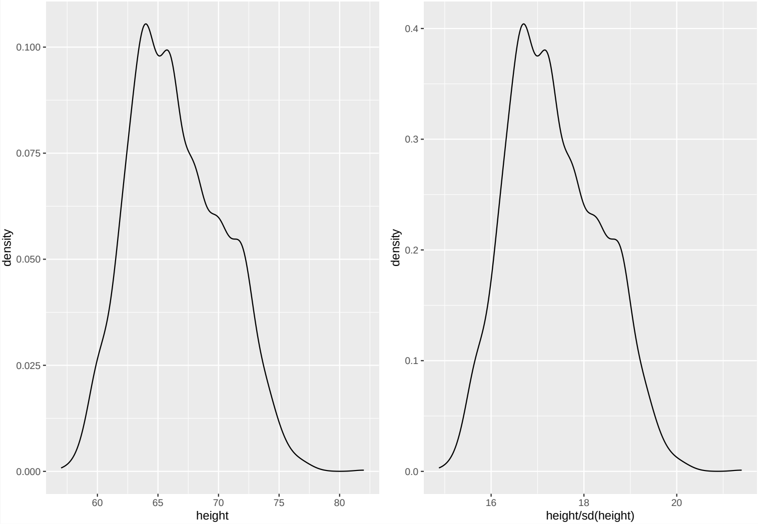
Scaling

We can *scale* a variable to improve interpretation when units aren't easy to interpret. A z-score transformation is convenient.

$Z(x) = x/s_x$

The z distribution of a variable has the same shape as the untransformed variable

A linear transformation



In a regression

The `scale()` function will by default mean center and *z* transform a variable.

```
m3_b <- stan_glm(log(earn) ~ scale(height), data = dat %>%
    filter(earn > 0), refresh = 0)
m3_b
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      log(earn) ~ scale(height)
##  observations: 1629
##  predictors:   2
## ------
##                Median MAD_SD
## (Intercept)    9.7    0.0
## scale(height) 0.2    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.9    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

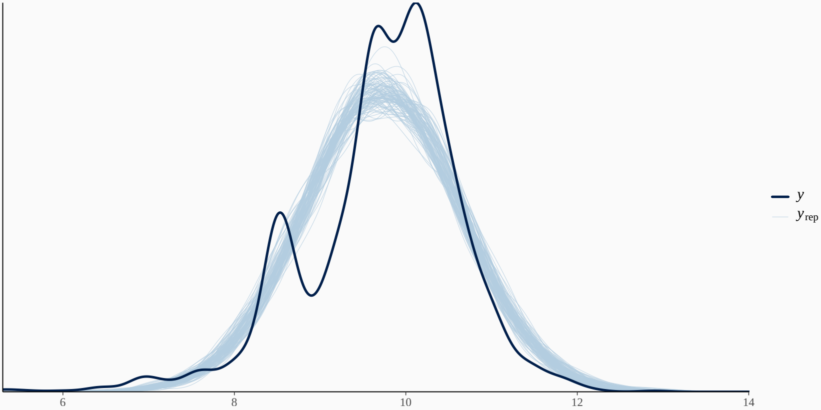
Guidance on scale()

Use `scale()` when a standard deviation unit is helpful, or when we've got predictors on very different measurement scales

Back to the fit

We've got multiple modes in the observed that aren't being reflected in the simulations

```
ppc_dens_overlay(log(earn_nozeroes$earn), m1_preds[1:100, ])
```



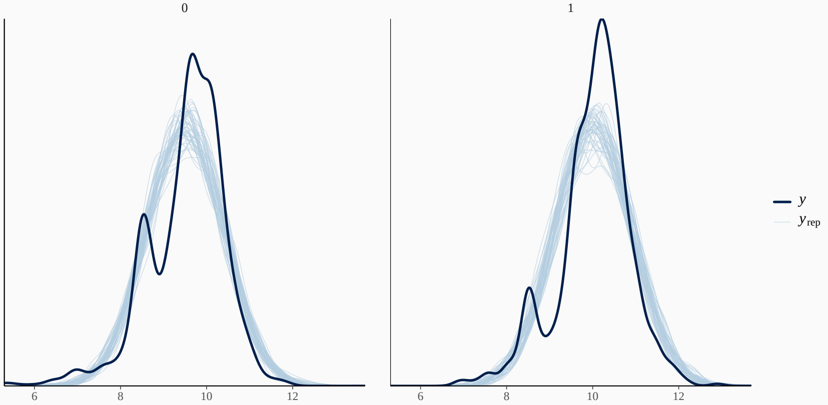
Revise the model

```r
m3_b <- stan_glm(log(earn) ~ scale(height) + male, data = dat %>%
    filter(earn > 0), refresh = 0)
m3_b
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      log(earn) ~ scale(height) + male
##  observations: 1629
##  predictors:   3
## ------
##               Median MAD_SD
## (Intercept)   9.6    0.0
## scale(height) 0.1    0.0
## male          0.4    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.9    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

Let's check the ppd (posterior predictive distribution) again

```
ppc_dens_overlay_grouped(log(earn_nozeroes$earn), posterior_predict(m3_b, draws = 50),
    group = earn_nozeroes$male)
```



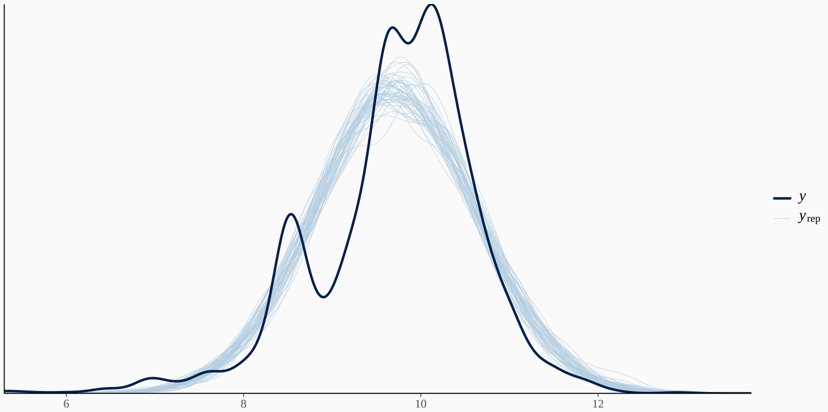
Range looks ok, maybe an interaction will help?

```r
m4_b <- stan_glm(log(earn) ~ scale(height) * male, data = dat %>%
    filter(earn > 0), refresh = 0)
m4_b
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      log(earn) ~ scale(height) * male
##  observations: 1629
##  predictors:   4
## ------
##                     Median MAD_SD
## (Intercept)         9.5    0.0
## scale(height)       0.1    0.0
## male                0.4    0.1
## scale(height):male 0.1    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.9    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

ppd check

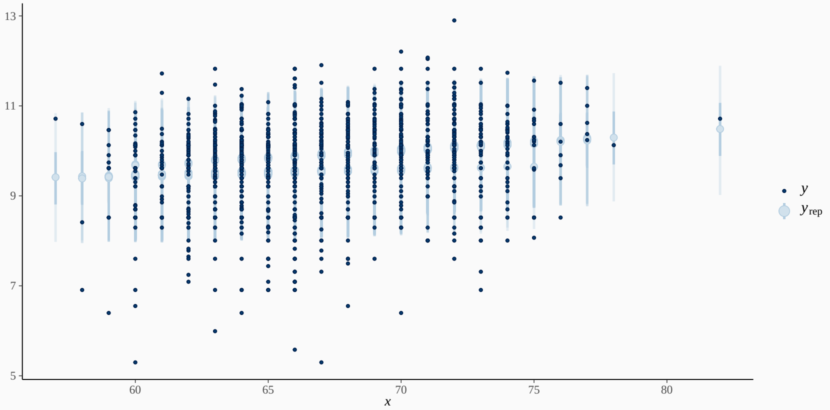
```r
ppc_dens_overlay(log(earn_nozeroes$earn), posterior_predict(m4_b, draws = 50))
```



Looking better there

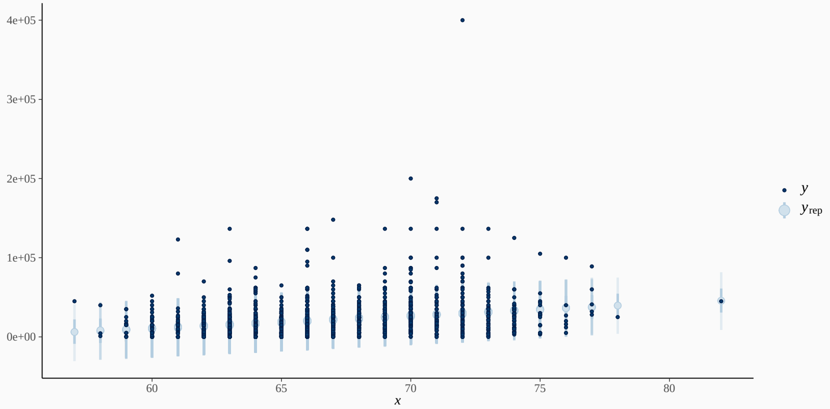
Let's check what posterior predictions look like relative to the predictors

```
ppc_intervals(y = log(earn_nozeroes$earn), yrep = posterior_predict(m4_b), x = earn_nozeroes$height)
```



And compare this to the untransformed model

```
ppc_intervals(y = dat$earn, yrep = posterior_predict(m0_b), x = dat$height)
```



One last visual: Model 4 on the original scale

```
ppc_intervals(y = earn_nozeroes$earn, yrep = exp(posterior_predict(m4_b)), x = earn_nozeroes$height)
```