Modeling with

broom modelr

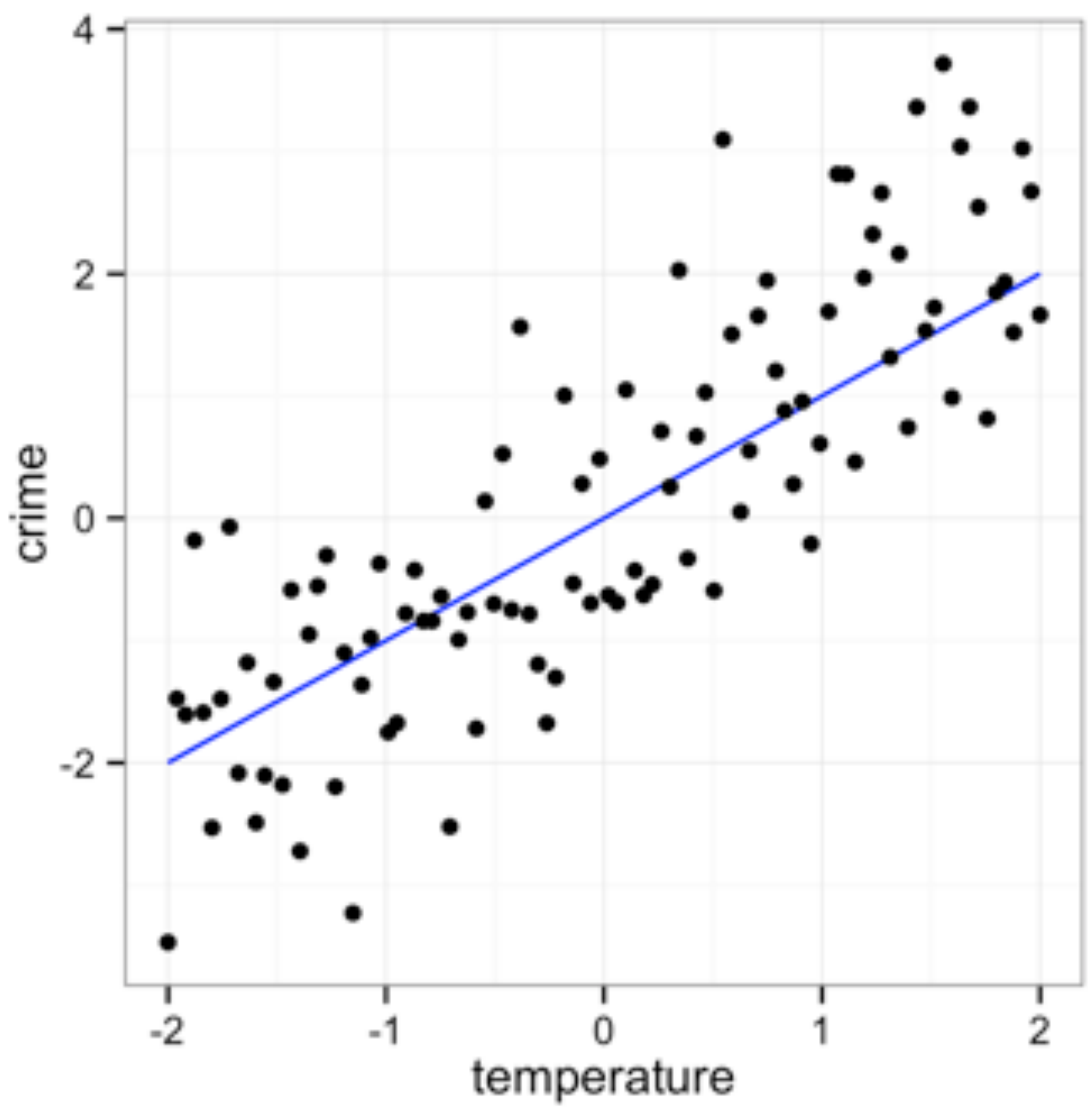
Navigate to the **04-Model** folder.

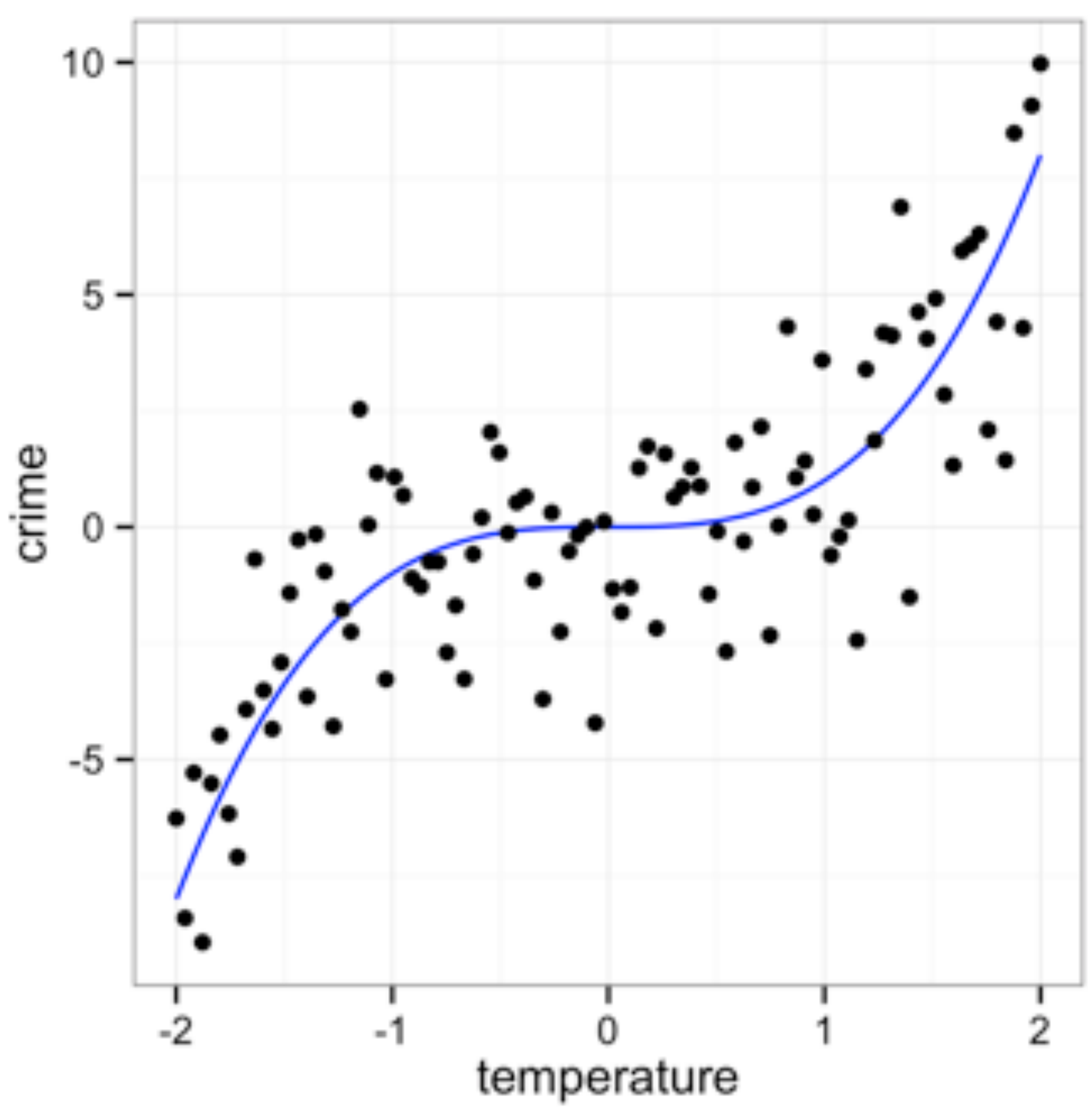
Open **04-Model-Exercises.Rmd**

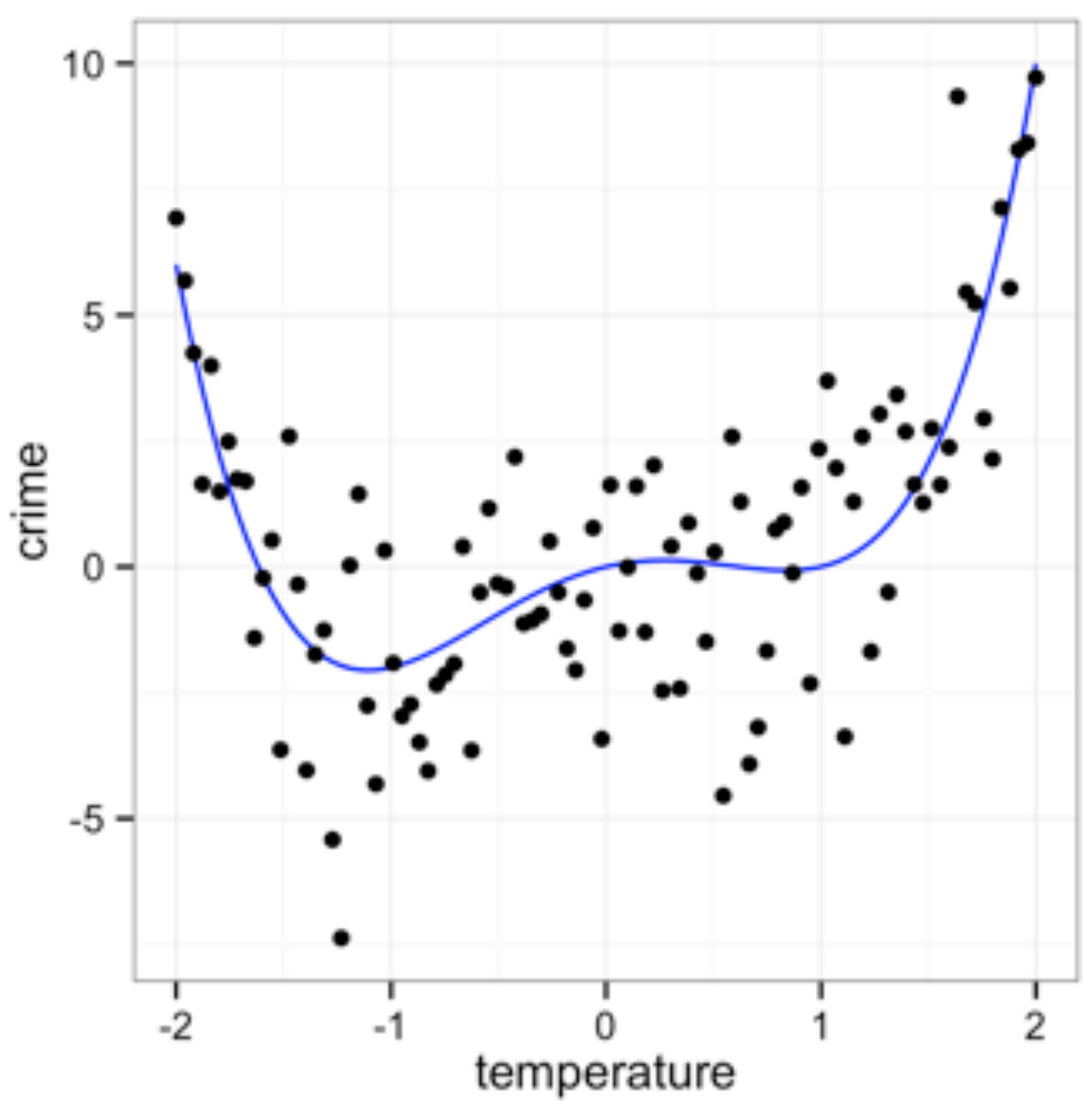
The basics

Models

A low dimensional description of a higher dimensional data set.

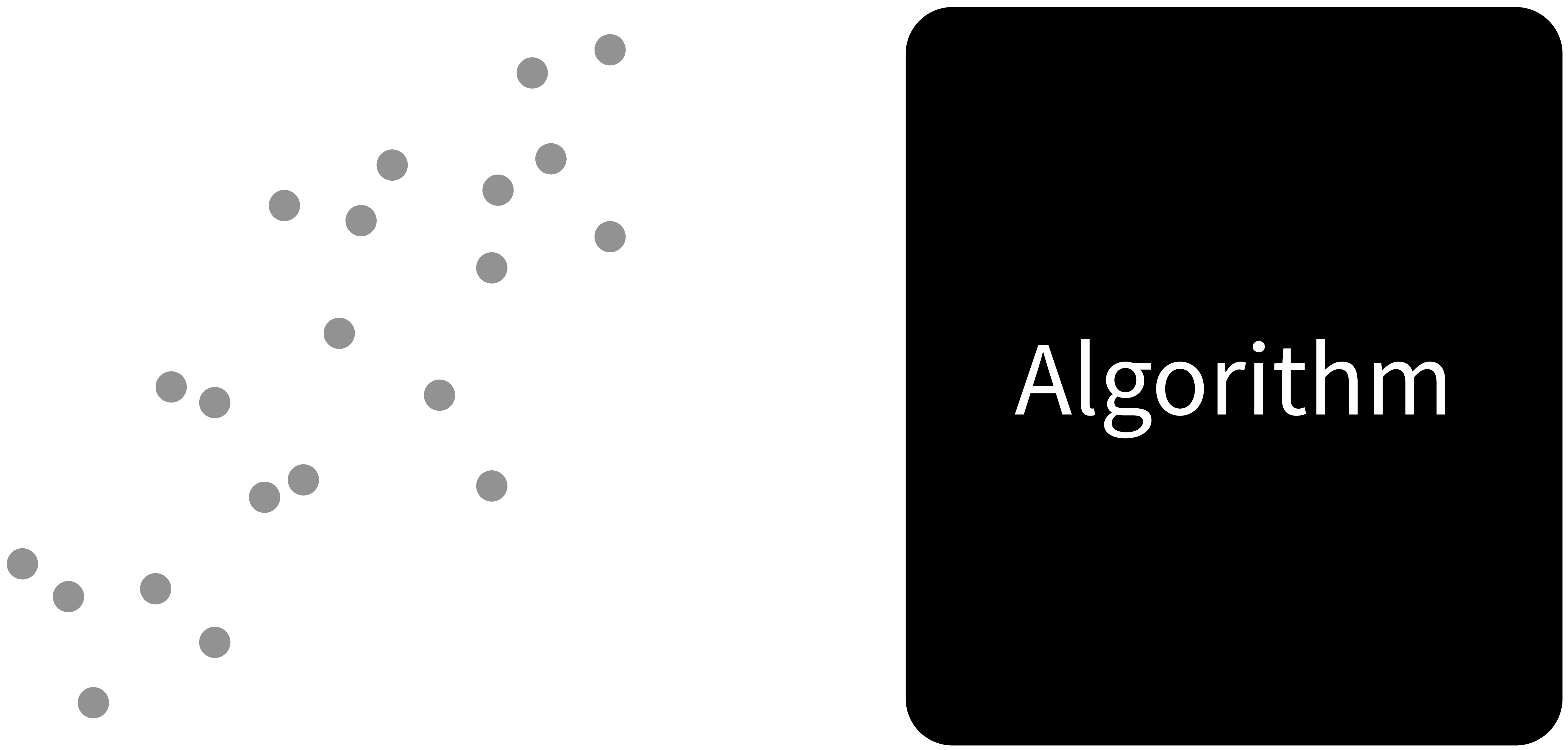






modelr

Models

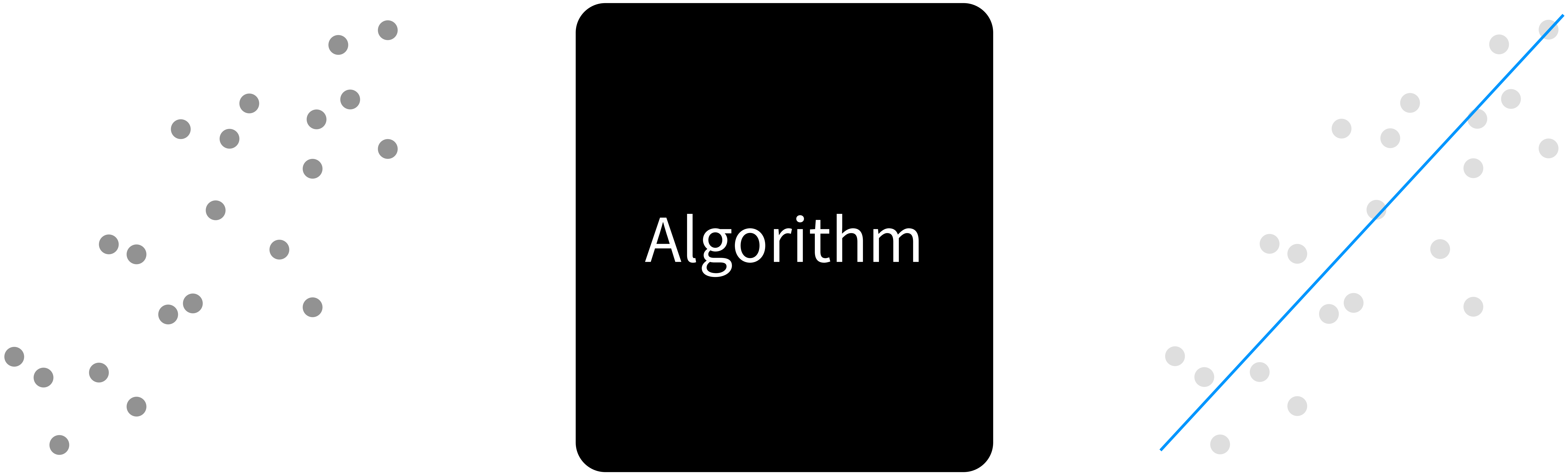


Data

Model Function

Models

What is the **model function**?

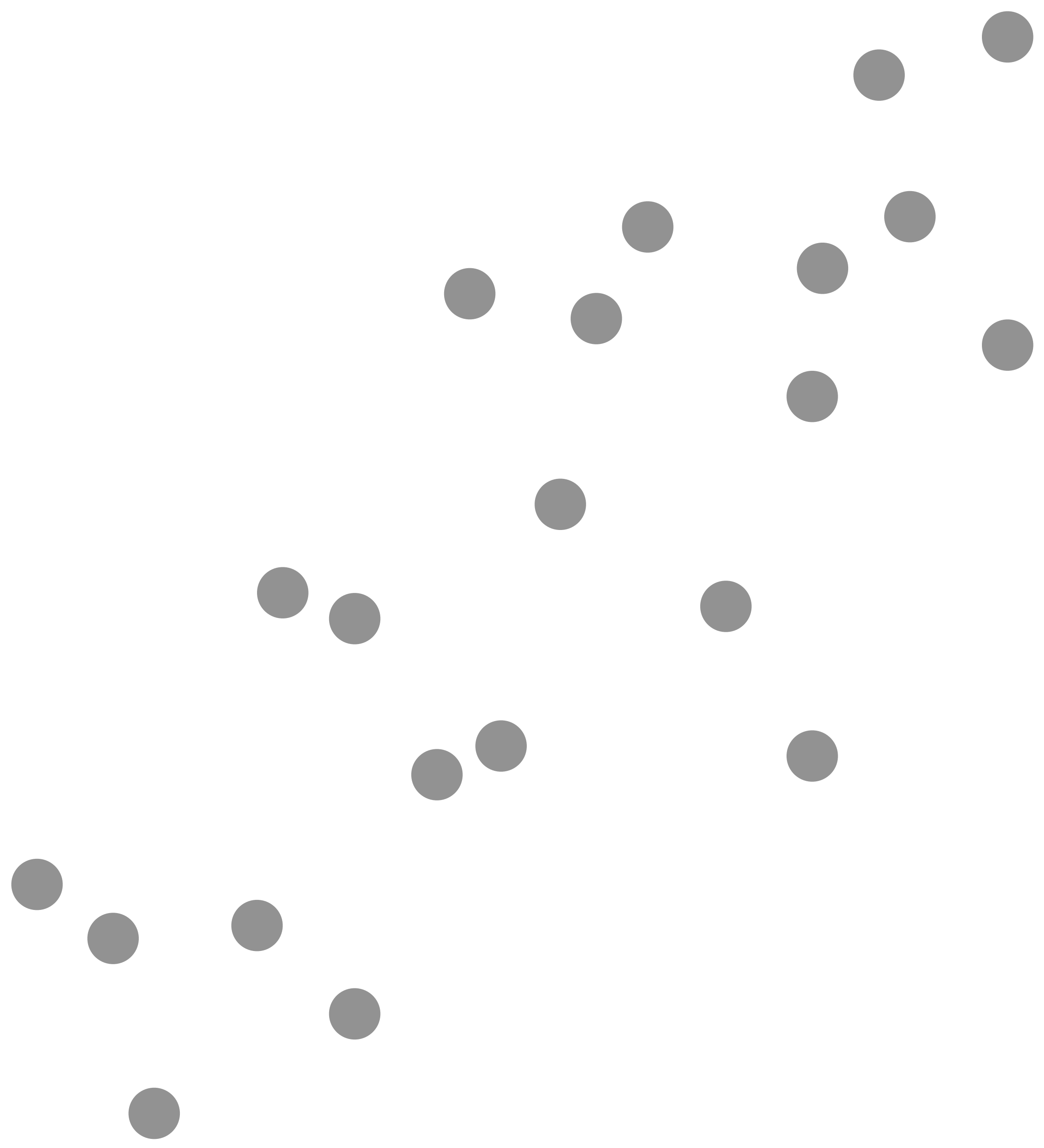


Data

Model Function

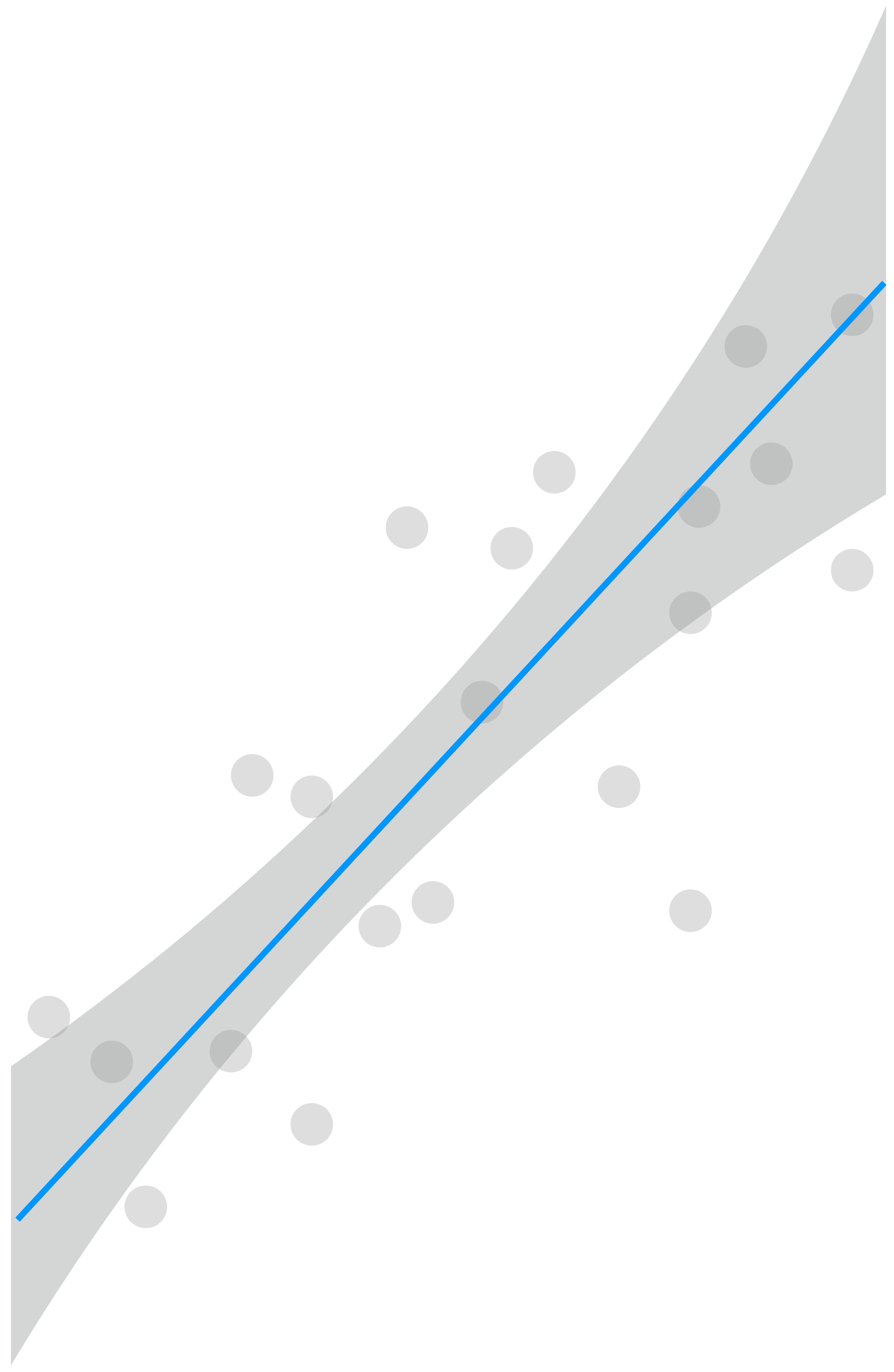
Models

What **uncertainty** is associated with it?



Data

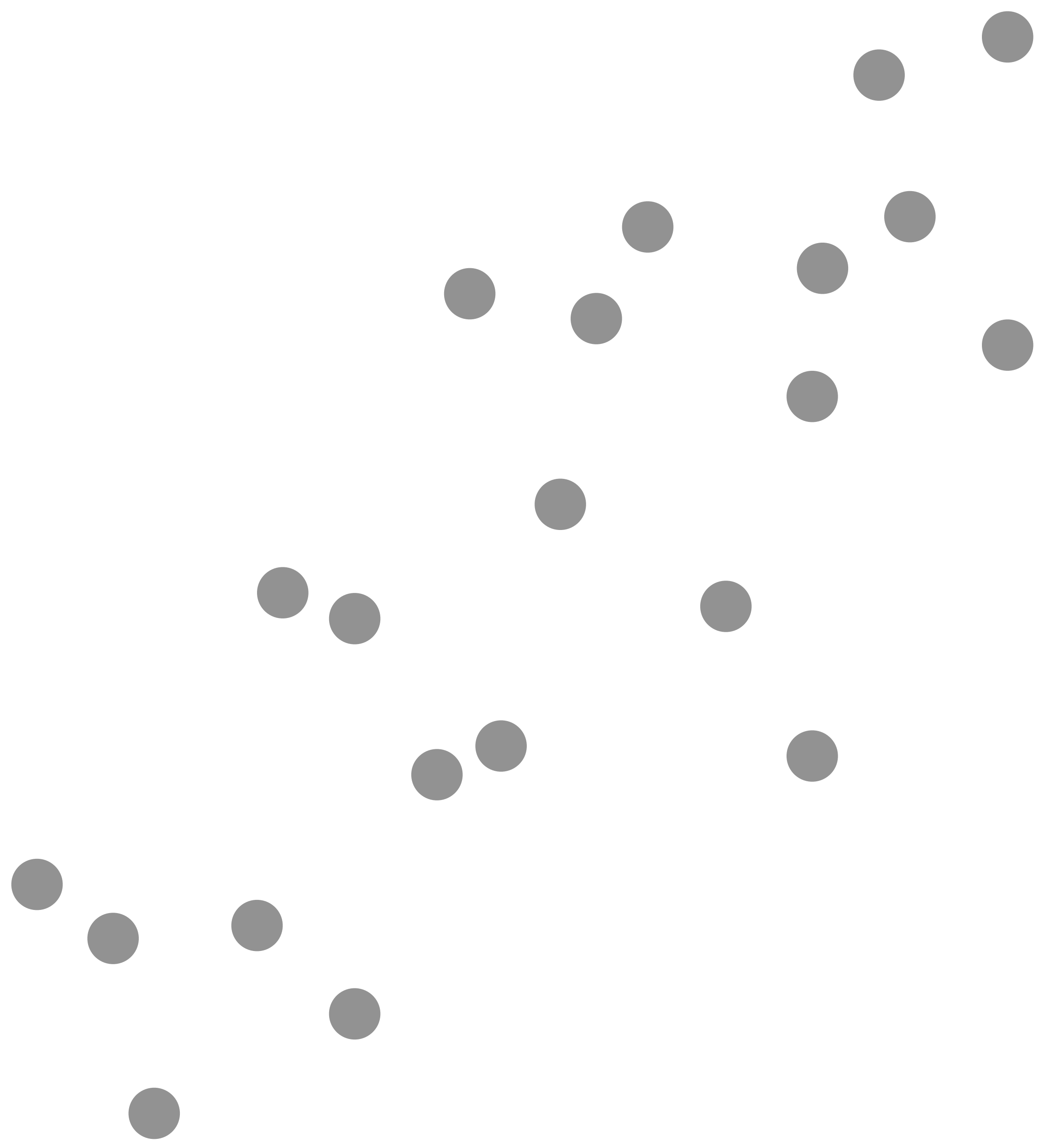
Algorithm



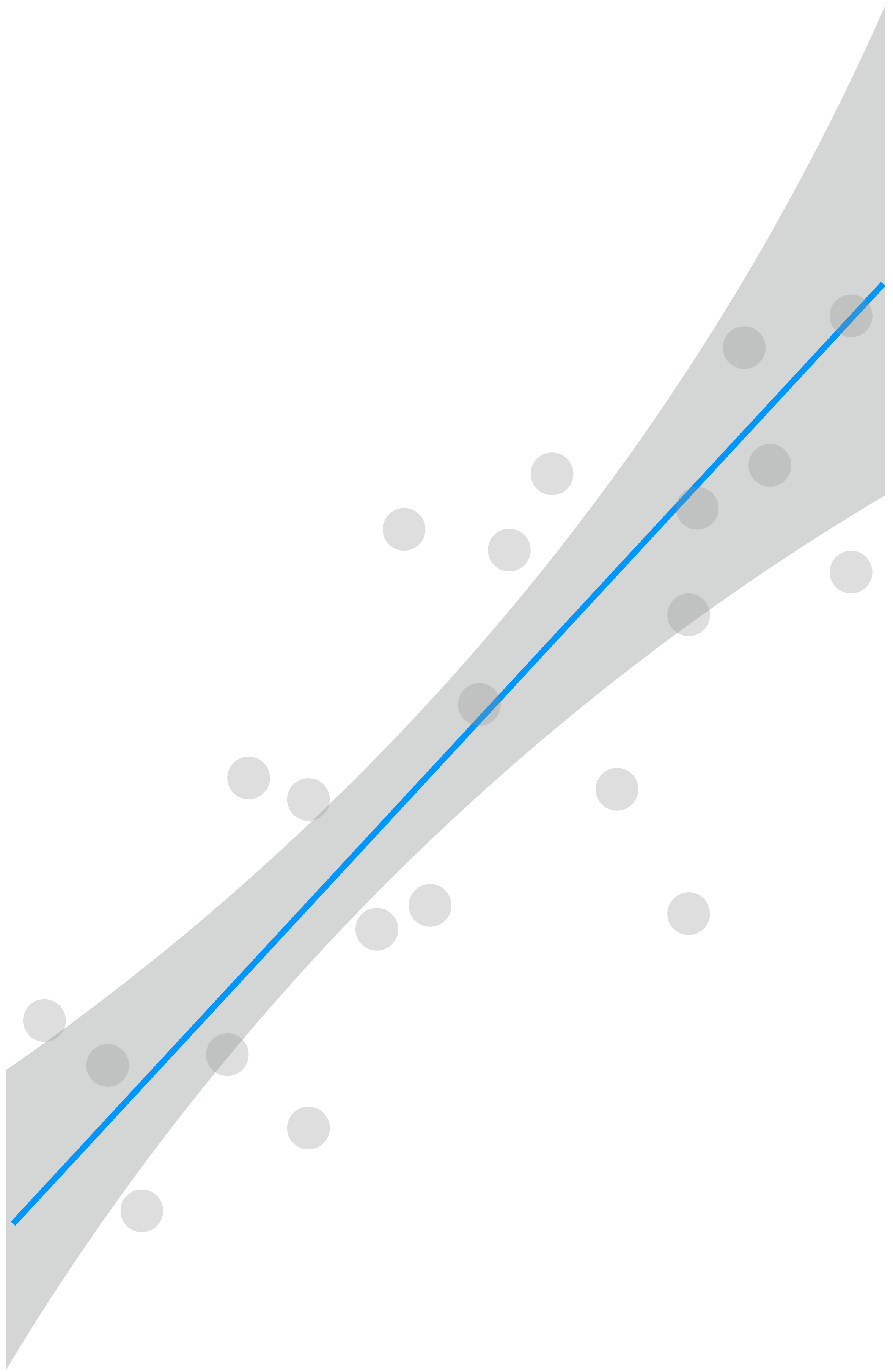
Model Function

Models

How "good" is the model?



Algorithm

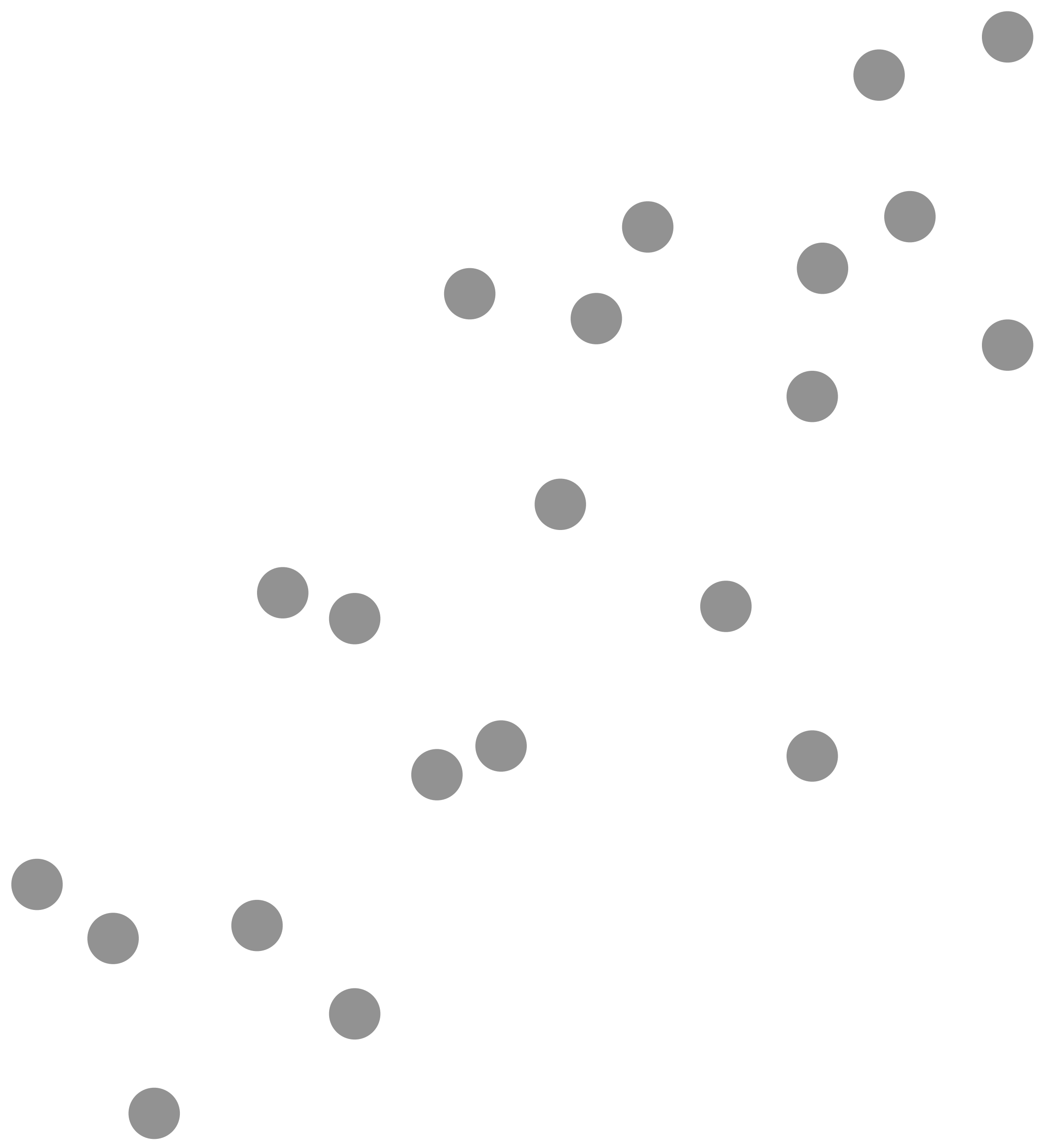


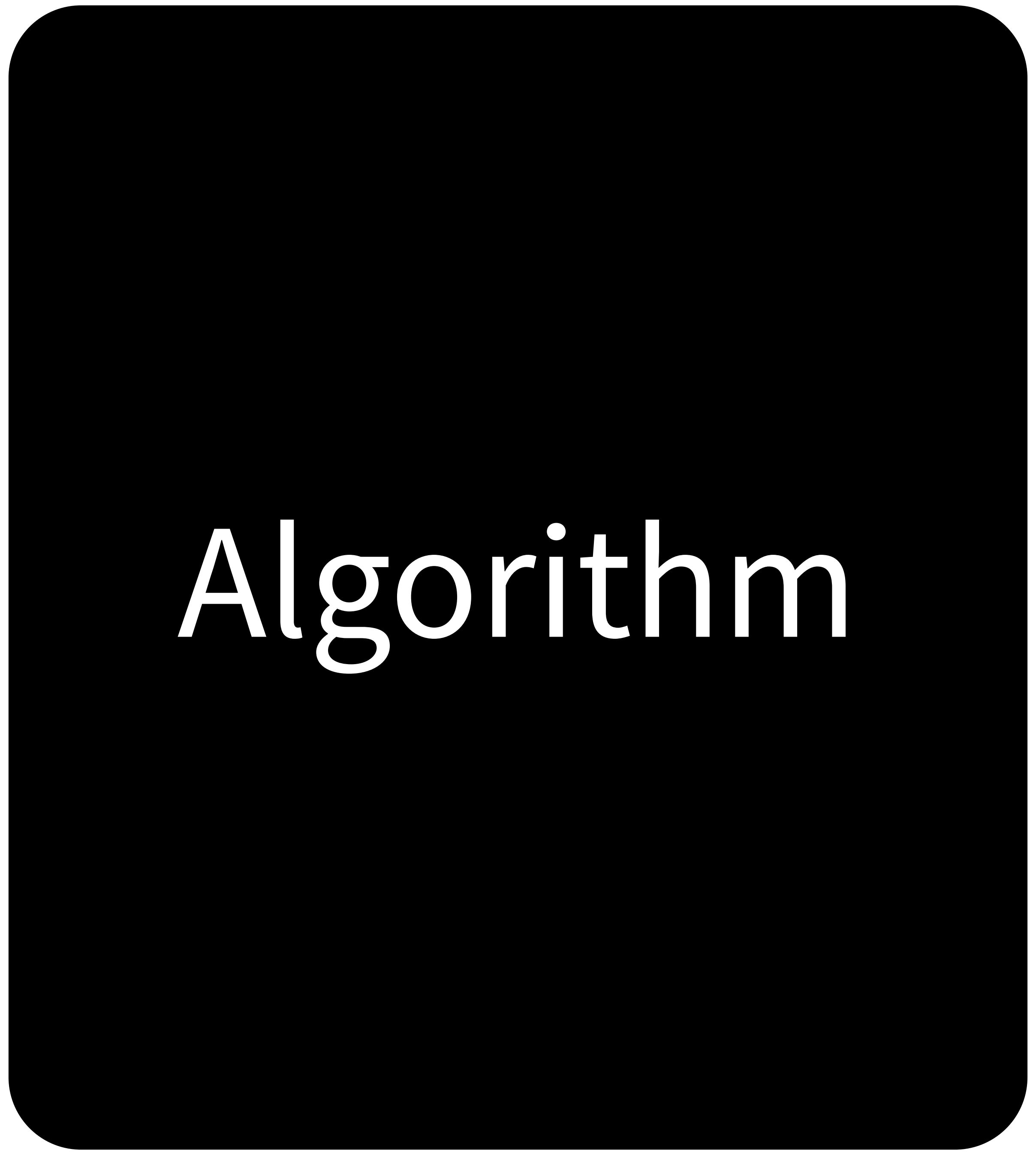
Data

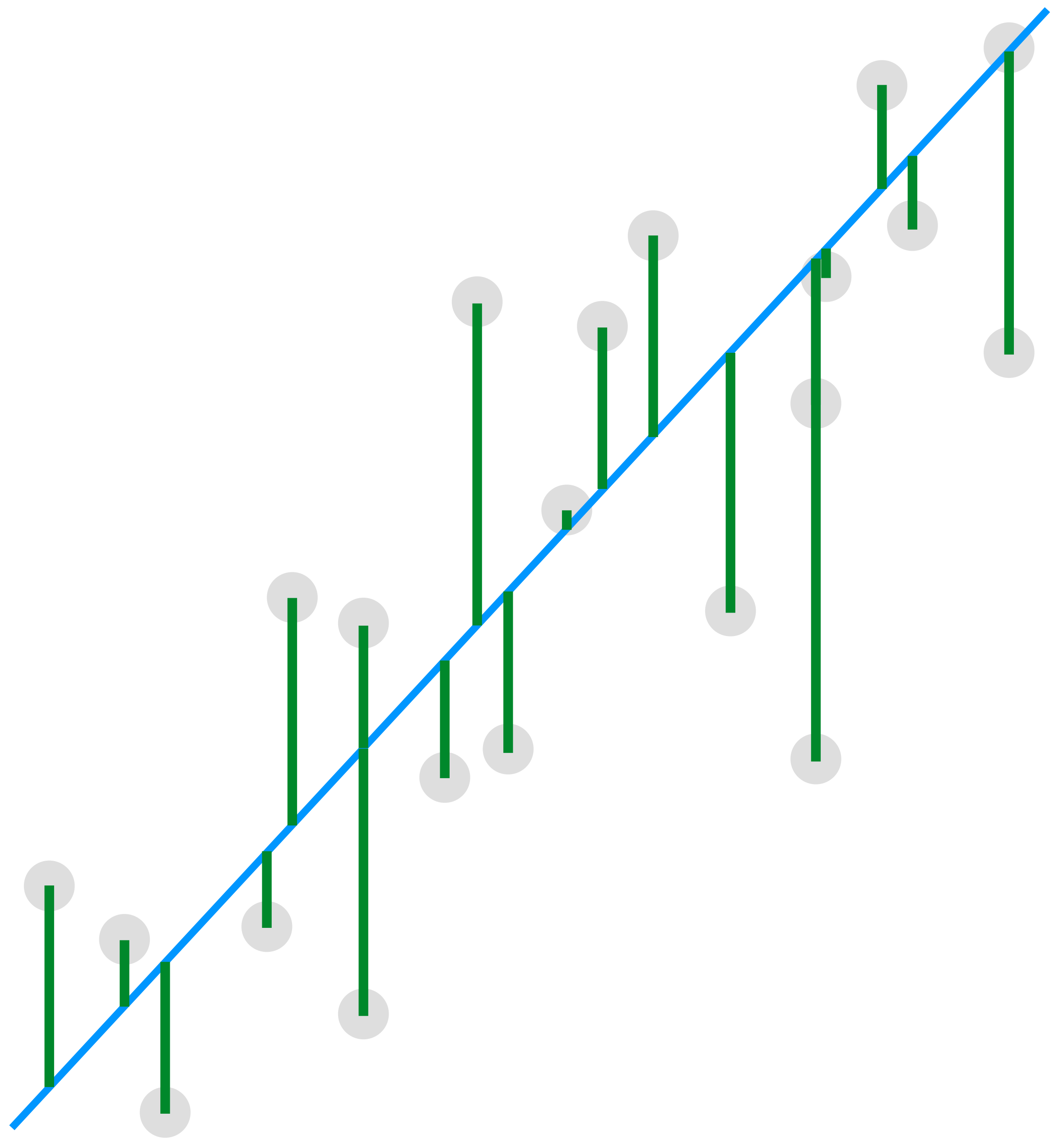
Model Function

Models

What are the **residuals**?





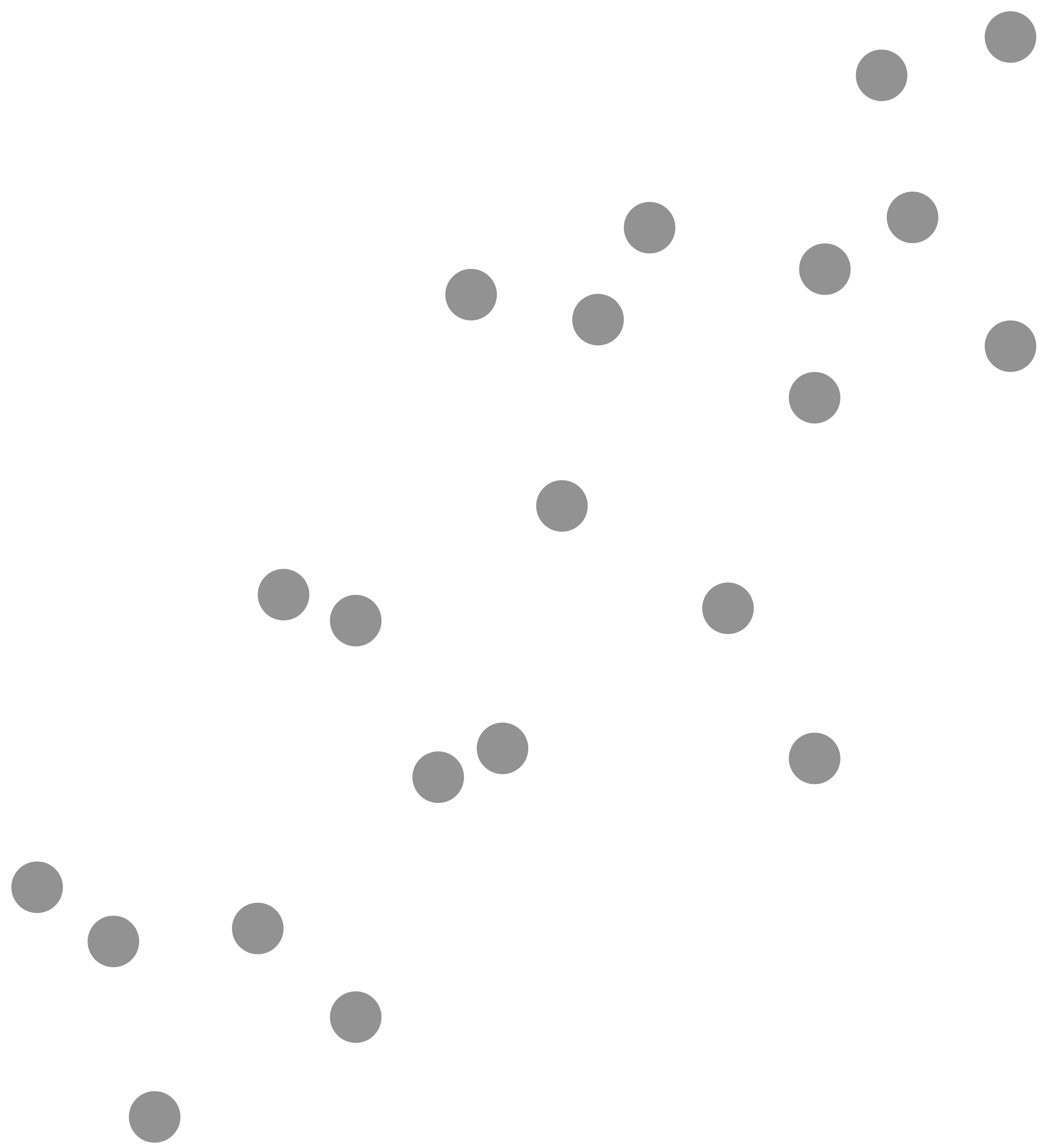


Data

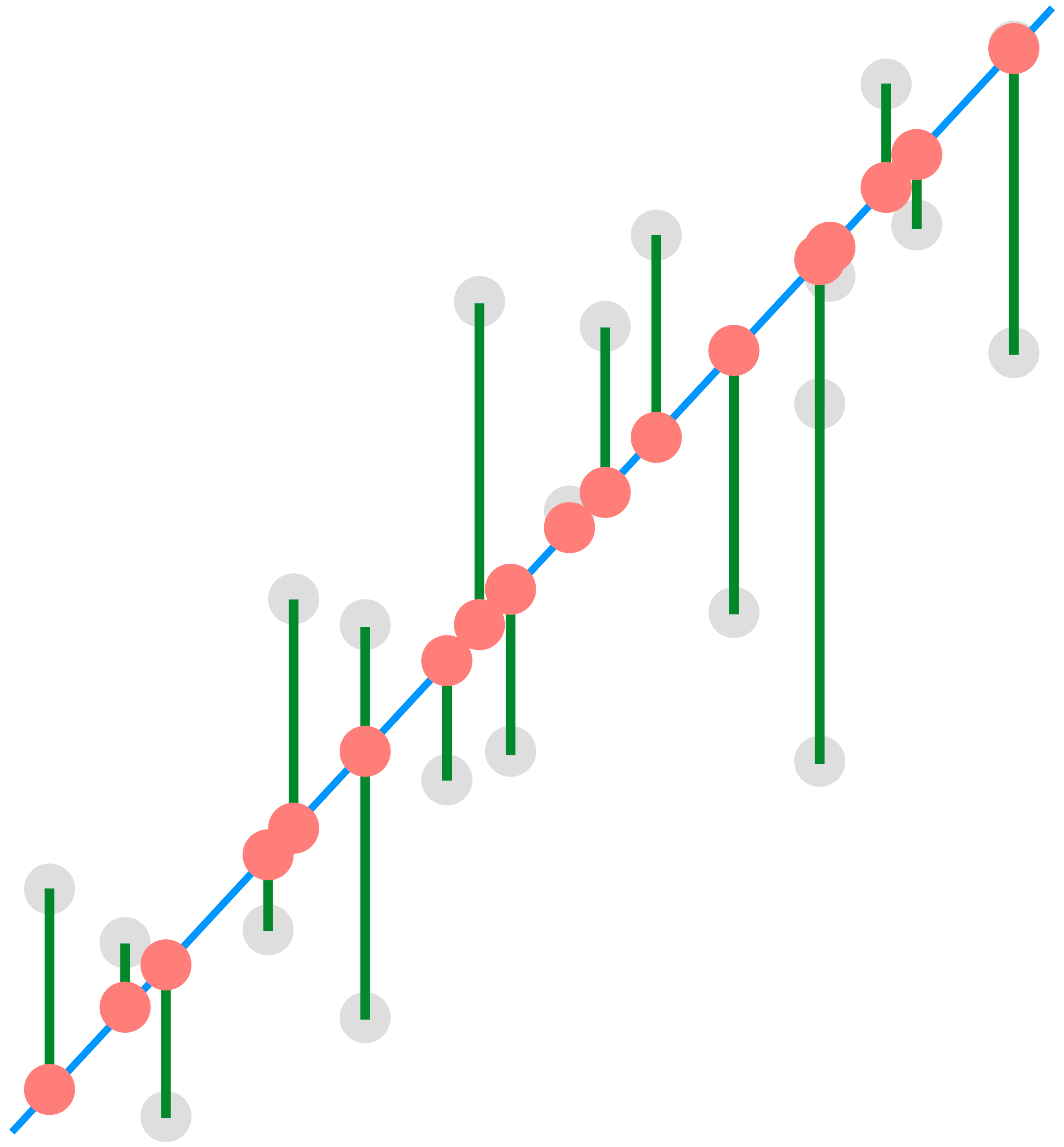
Model Function

Models

What are the **predictions**?



Algorithm



Data

Model Function

Algorithm

(Popular) **modeling functions in R**

function	package	fits
lm()	stats	linear models
glm()	stats	generalized linear models
gam()	mgcv	generalized additive models
glmnet()	glmnet	penalized linear models
rlm()	MASS	robust linear models
rpart()	rpart	trees
randomForest()	randomForest	random forests
xgboost()	xgboost	gradient boosting machines

(Popular) **modeling functions in R**

function	package	fits
lm()	**stats**	**linear models**
glm()	stats	generalized linear models
gam()	mgcv	generalized additive models
glmnet()	glmnet	penalized linear models
rlm()	MASS	robust linear models
rpart()	rpart	trees
randomForest()	randomForest	random forests
xgboost()	xgboost	gradient boosting machines

modelr

Tidy functions that make it easier to work with models in R

```
# install.packages("tidyverse")
library(modelr)
wages <- heights %>% filter(income > 0)
```

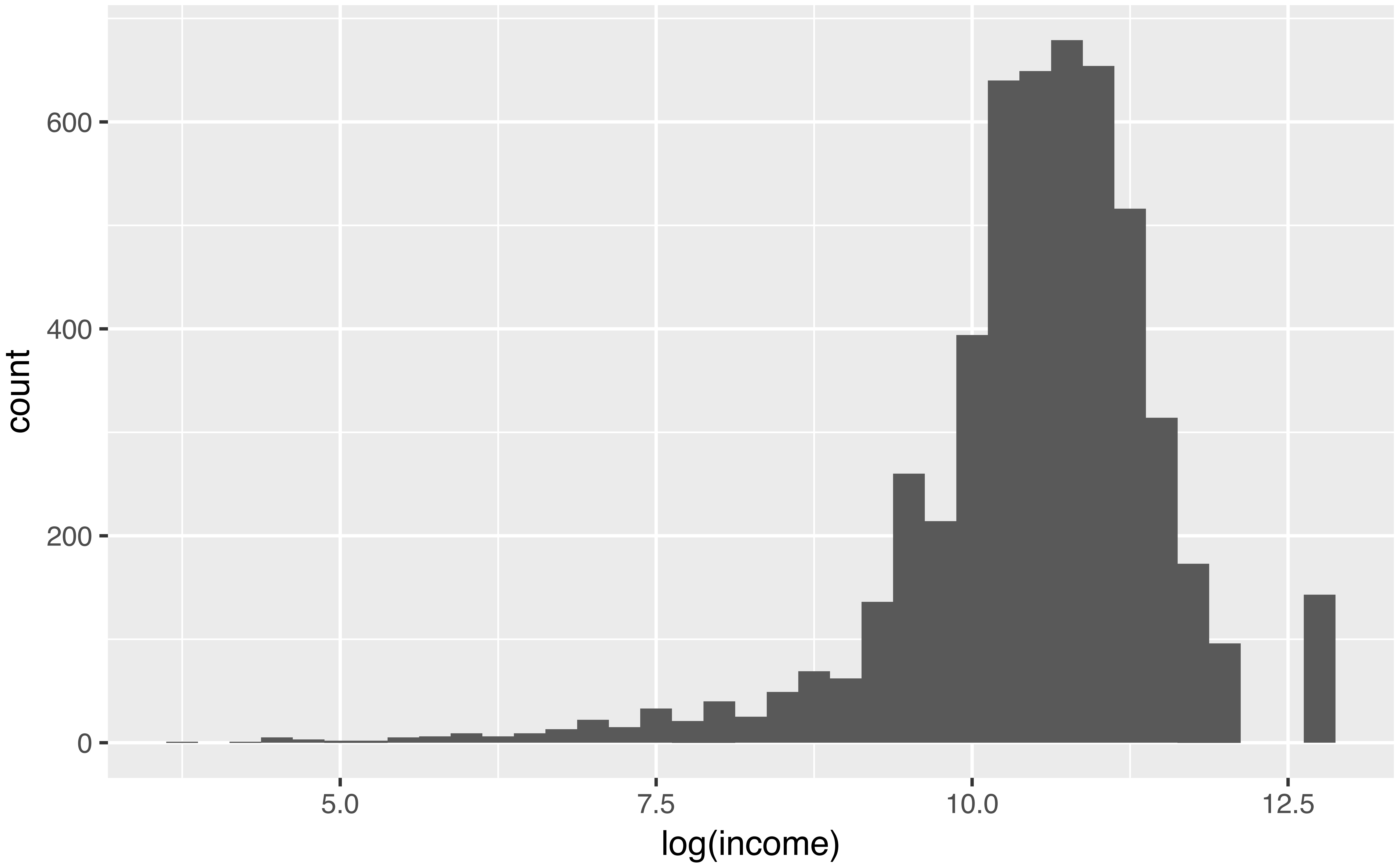
```
wages
```

income <int>	eight <dbl>	weight <int>	age <int>	marital <fctr>	sex <fctr>	education <int>	afqt <dbl>
19000	60	155	53	married	female	13	6.841
35000	70	156	51	married	female	10	49.444
105000	65	195	52	married	male	16	99.393
40000	63	197	54	married	female	14	44.022
75000	66	190	49	married	male	14	59.683
102000	68	200	49	divorced	female	18	98.798
0	74	225	48	married	male	16	82.260
70000	64	160	54	divorced	female	12	50.283
60000	69	162	55	divorced	male	12	89.669
150000	69	194	54	divorced	male	13	95.977

1–10 of 7,006 rows Previous 1 2 3 4 5 6 ... 100 Next

modelr

```r
wages %>%
  ggplot(aes(log(income))) + geom_histogram(binwidth = 0.25)
```



lm()

lm()

Fit a linear model to data

```
lm(log(income) ~ education, data = wages)
```

A formula that describes the model equation

The data set

modelr

formulas

Formula only needs to include the response and predictors

$$y = \alpha + \beta x + \epsilon$$

```
y ~ x
```

Response

A tilde

The predictors

Your Turn 1

Fit the model below and then examine the output. What does it look like?

```
mod_e <- lm(log(income) ~ education, data = wages)
```

02:00

```
mod_e <- lm(log(income) ~ education, data = wages)
mod_e
## Call:
## lm(formula = log(income) ~ education, data = wages)
##
## Coefficients:
## (Intercept)    education
##      8.5577       0.1418
class(mod_e)
## "lm"
```

1. Not pipe friendly to have data as second argument :(

2. Output is not tidy, or even a data frame

.

Use "." to pipe input to somewhere other than the first argument

```
mod_e <- wages %>%
  lm(log(income) ~ education, data = .)
```

wages will be passed to here

broom

broom

Turns model output into data frames

```
# install.packages("tidyverse")
library(broom)
```

broom

Broom includes three functions which work for most types of models (and can be extended to more):

1. **tidy()** - returns model coefficients, stats
2. **glance()** - returns model diagnostics
3. **augment()** - returns predictions, residuals, and other raw values

tidy()

Returns useful **model output** as a data frame

```
mod_e %>% tidy()
```

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	8.5576906	0.073259622	116.81320	0.000000e+00
education	0.1418404	0.005304577	26.73924	8.408952e-148

2 rows

broom

glance

Returns common **model diagnostics** as a data frame

```
mod_e %>% glance()
```

r.squared <dbl>	adj.r.squared <dbl>	sigma <dbl>	statistic <dbl>	p.valu <db
0.1196233	0.119456	0.9923358	714.987	8.408952e-14

1 row | 1-10 of 11 columns

augment()

Returns data frame of **model output related to original data points**

```
mod_e %>% augment()
```

.rownames <chr>	log.income. <dbl>	education <int>	.fitted <dbl>	.se.fit <dbl>	.resid <dbl>	.hat <dbl>	.sigma <dbl>	.c
1	9.852194	13	10.401615	0.01400504	-0.549421141	0.0001991827	0.9924012	3.05413
2	10.463103	10	9.976094	0.02335067	0.487009048	0.0005537086	0.9924074	6.67558
3	11.561716	16	10.827137	0.01880219	0.734579123	0.0003590043	0.9923784	9.84331
4	10.596635	14	10.543456	0.01386811	0.053178965	0.0001953068	0.9924299	2.80556
5	11.225243	14	10.543456	0.01386811	0.681787624	0.0001953068	0.9923856	4.61145
6	11.532728	18	11.110817	0.02719979	0.421910848	0.0007513008	0.9924131	6.80081
7	11.156251	12	10.259775	0.01600734	0.896475490	0.0002602083	0.9923532	1.06237
8	11.002100	12	10.259775	0.01600734	0.742324811	0.0002602083	0.9923774	7.28429
9	11.918391	13	10.401615	0.01400504	1.516775174	0.0001991827	0.9922098	2.32766
10	11.652687	16	10.827137	0.01880219	0.825550901	0.0003590043	0.9923648	1.24323

augment()

Returns data frame of **model output related to original data points**

```
mod_e %>% augment(data = wages)
```

Adds the original wages data set to the output

broom

Your Turn 2

Use a pipe to model **log(income)** against **height**. Then use broom and dplyr functions to extract:

1. The coefficient estimates and their related statistics
2. The adj.r.squared and p.value for the overall model

05:00

```
mod_h <- wages %>% lm(log(income) ~ height, data = .)
mod_h %>%
  tidy()
##          term   estimate   std.error statistic       p.value
## 1 (Intercept) 6.98342583 0.237484827  29.40578 4.129821e-176
## 2      height 0.05197888 0.003521666  14.75974  2.436945e-48
mod_h %>%
  glance() %>%
  select(adj.r.squared, p.value)
##   adj.r.squared      p.value
## 1    0.03955779 2.436945e-48
```

```
mod_h %>%
  tidy() %>% filter(p.value < 0.05)
##          term   estimate   std.error statistic       p.value
## 1 (Intercept) 6.98342583 0.237484827  29.40578 4.129821e-176
## 2      height 0.05197888 0.003521666  14.75974  2.436945e-48
mod_e %>%
  tidy() %>% filter(p.value < 0.05)
##          term  estimate   std.error statistic       p.value
## 1 (Intercept) 8.5576906 0.073259622 116.81320  0.000000e+00
## 2   education 0.1418404 0.005304577  26.73924 8.408952e-148
```

so which determines income?

multivariate regression

To fit multiple predictors,
add multiple variables to the formula:

```
log(income) ~ education + height
```

broom

```r
mod_eh <- wages %>%
  lm(log(income) ~ education + height, data = .)

mod_eh %>%
  tidy()
##          term   estimate   std.error statistic       p.value
## 1 (Intercept) 5.34837618 0.231320415  23.12107 1.002503e-112
## 2   education 0.13871285 0.005205245  26.64867 7.120134e-147
## 3      height 0.04830864 0.003309870  14.59533  2.504935e-47
```

Your Turn 3

Model **log(income)** against **education** and **height** and **sex**. Can you interpret the coefficients?

03:00

```
mod_ehs <- wages %>%
  lm(log(income) ~ education + height + sex, data = .)

mod_ehs %>%
  tidy()
##          term     estimate   std.error  statistic       p.value
## 1 (Intercept)  8.250422260 0.334703051  24.649976 4.681336e-127
## 2   education  0.147983063 0.005196676  28.476486 5.164290e-166
## 3      height  0.006726614 0.004792698   1.403513  1.605229e-01
## 4   sexfemale -0.461747002 0.038941592 -11.857425  5.022841e-32
```

Where is sexmale?

What does this mean?

broom

```
wages
```

FACTOR

income <int>	height <dbl>	weight <int>	age <int>	marital <fctr>	sex <fctr>	education <int>	afqt <dbl>
19000	60	155	53	married	female	13	6.841
35000	70	156	51	married	female	10	49.444
105000	65	195	52	married	male	16	99.393
40000	63	197	54	married	female	14	44.022
75000	66	190	49	married	male	14	59.683
102000	68	200	49	divorced	female	18	98.798
0	74	225	48	married	male	16	82.260
70000	64	160	54	divorced	female	12	50.283
60000	69	162	55	divorced	male	12	89.669
150000	69	194	54	divorced	male	13	95.977

1-10 of 7,006 rows Previous 1 2 3 4 5 6 ... 100 Next

modelr

factors

R’s representation of categorical data. Consists of:

1. A set of **values**
2. An ordered set of **valid levels**

YOU CAN MAKE WITH FACTOR()

```
sexes <- factor(x = c("male", "female", "male"),
                levels = c("male", "female", "other"))
sexes
## male   female male
## Levels: male female other
```

factors

Stored as an integer vector with a levels attribute

```
unclass(sexes)
## 1 2 1
## attr(,"levels")
## "male"   "female" "other"
```

```
##             term     estimate   std.error  statistic       p.value
## 1 (Intercept)  8.250422260 0.334703051  24.649976 4.681336e-127
## 2   education  0.147983063 0.005196676  28.476486 5.164290e-166
## 3      height  0.006726614 0.004792698   1.403513  1.605229e-01
## 4   sexfemale -0.461747002 0.038941592 -11.857425  5.022841e-32
```

For factors, R treats the first level as the baseline level, e.g. the mean log(income) for a male is:

log(income) = 8.25 + 0.15 * education + 0 * height

Each additional level gets a coefficient that acts as an *adjustment* between the baseline level and the additional level, e.g. the mean income for a female is:

log(income) = 8.25 + 0.15 * education + 0 * height - 0.46

To change the levels, refactor.

```
sexes <- factor(sexes, levels = c("female", "male", "other"))

sexes
## male   female male
Levels: female male other

unclass(sexes)
## 2 1 2
## attr(,"levels")
## "female" "male"   "other"
```

NEW LEVEL ORDER

To change the levels, refactor.

```
sexes <- factor(sexes, levels = c("female", "male", "other"))


wages <-
  wages %>%
  mutate(sex = factor(sex, levels = c("female", "male"))
```

```
##          term     estimate   std.error  statistic       p.value
## 1 (Intercept)  8.250422260 0.334703051  24.649976 4.681336e-127
## 2   education  0.147983063 0.005196676  28.476486 5.164290e-166
## 3      height  0.006726614 0.004792698   1.403513  1.605229e-01
## 4   sexfemale -0.461747002 0.038941592 -11.857425  5.022841e-32
```

But what does all of this look like?

model visualization

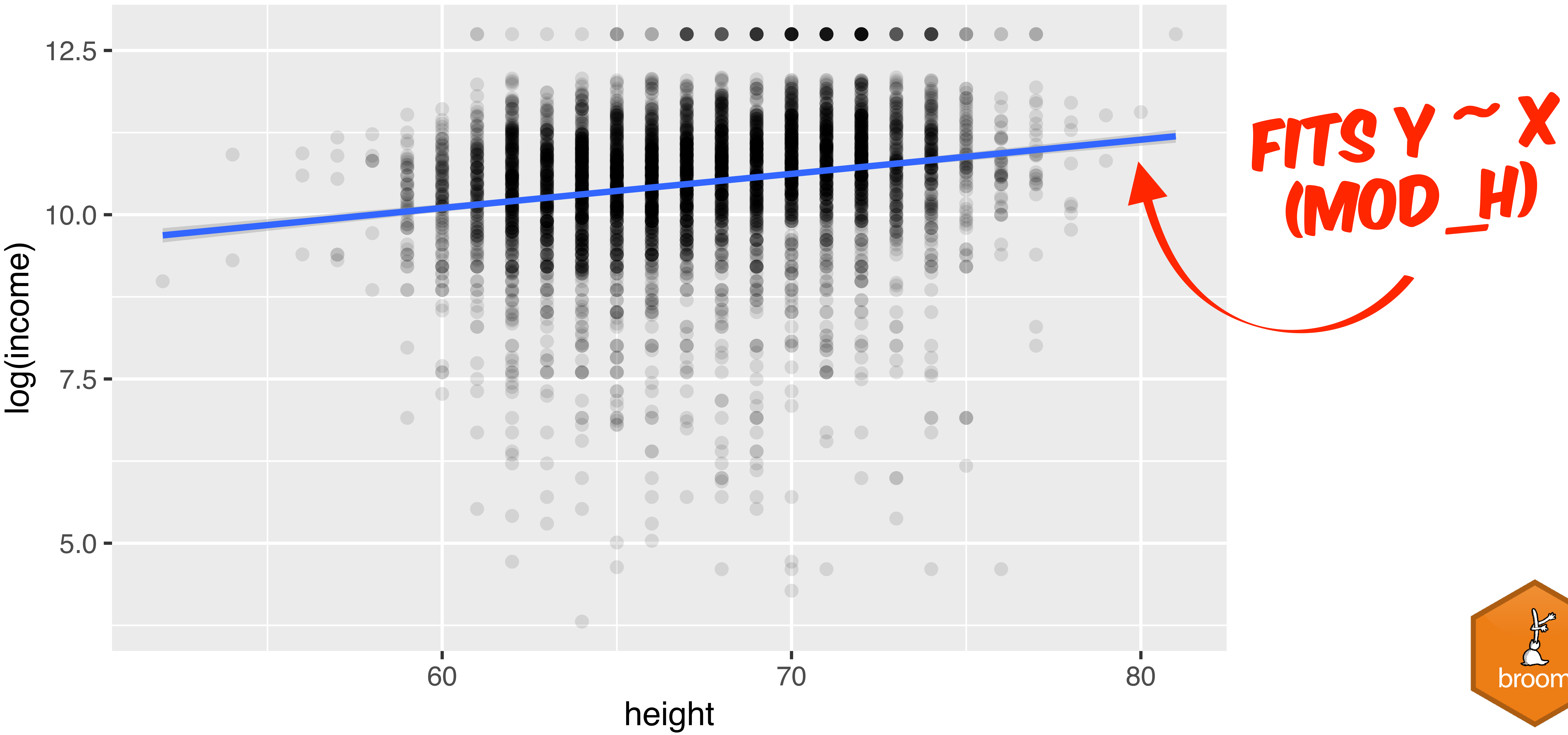
Your Turn 4

Add **+ geom_smooth(model = lm)** to the code below. What happens?

```
wages %>%
  ggplot(aes(x = height, y = log(income))) +
  geom_point(alpha = 0.1)
```

01:00

```
wages %>%
  ggplot(aes(x = height, y = log(income))) +
    geom_point(alpha = 0.1) +
    geom_smooth(method = lm)
```



geom_smooth()

Adds model line for predicting y ~ x (default).

```
p + geom_smooth(method = lm, se = TRUE, …)
```

An R modeling function to use to generate the line

Include standard error bars?

Other args to pass to modeling function

What about complex models, or residuals?

broom

modelr

Tidy functions that make it easier to work with models in R

```
# install.packages("tidyverse")
library(modelr)
```

add_predictions()

Uses the values in a data frame to generate a prediction for each case. Overlaps with augment()*

```
add_predictions(data, model, var = "pred")
```

Uses this model

To add predictions to these cases

Gives new column this name

modelr

```
wages %>% add_predictions(mod_h)
```

height <dbl>	weight <int>	age <int>	marital <fctr>	sex <fctr>	education <int>	afqt <dbl>	pred <dbl>
60	155	53	married	female	13	6.841	10.102158
70	156	51	married	female	10	49.444	10.621947
65	195	52	married	male	16	99.393	10.362053
63	197	54	married	female	14	44.022	10.258095
66	190	49	married	male	14	59.683	10.414032
68	200	49	divorced	female	18	98.798	10.517989
64	160	54	divorced	female	12	50.283	10.310074
69	162	55	divorced	male	12	89.669	10.569968
69	194	54	divorced	male	13	95.977	10.569968
64	145	53	married	female	16	67.021	10.310074

1-10 of 5,266 rows | 2-9 of 9 columns   Previous 1 2 3 4 5 6 ... 100 Next

modelr

add_residuals()

Uses the values in a data frame to generate a residual for each case. Overlaps with augment()*

```
add_residuals(data, model, var = "resid")
```

Uses this model

To add residuals to these cases

Gives new column this name

```r
wages %>% add_residuals(mod_h)
```

income <int>	height <dbl>	weight <int>	age <int>	marital <fctr>	sex <fctr>	education <int>	afqt <dbl>	resid <dbl>
19000	60	155	53	married	female	13	6.841	-0.2499641042
35000	70	156	51	married	female	10	49.444	-0.1588437767
105000	65	195	52	married	male	16	99.393	1.1996628894
40000	63	197	54	married	female	14	44.022	0.3385397443
75000	66	190	49	married	male	14	59.683	0.8112117773
102000	68	200	49	divorced	female	18	98.798	1.0147387260
70000	64	160	54	divorced	female	12	50.283	0.8461766568
60000	69	162	55	divorced	male	12	89.669	0.4321315995
150000	69	194	54	divorced	male	13	95.977	1.3484223314
115000	64	145	53	married	female	16	67.021	1.3426135431

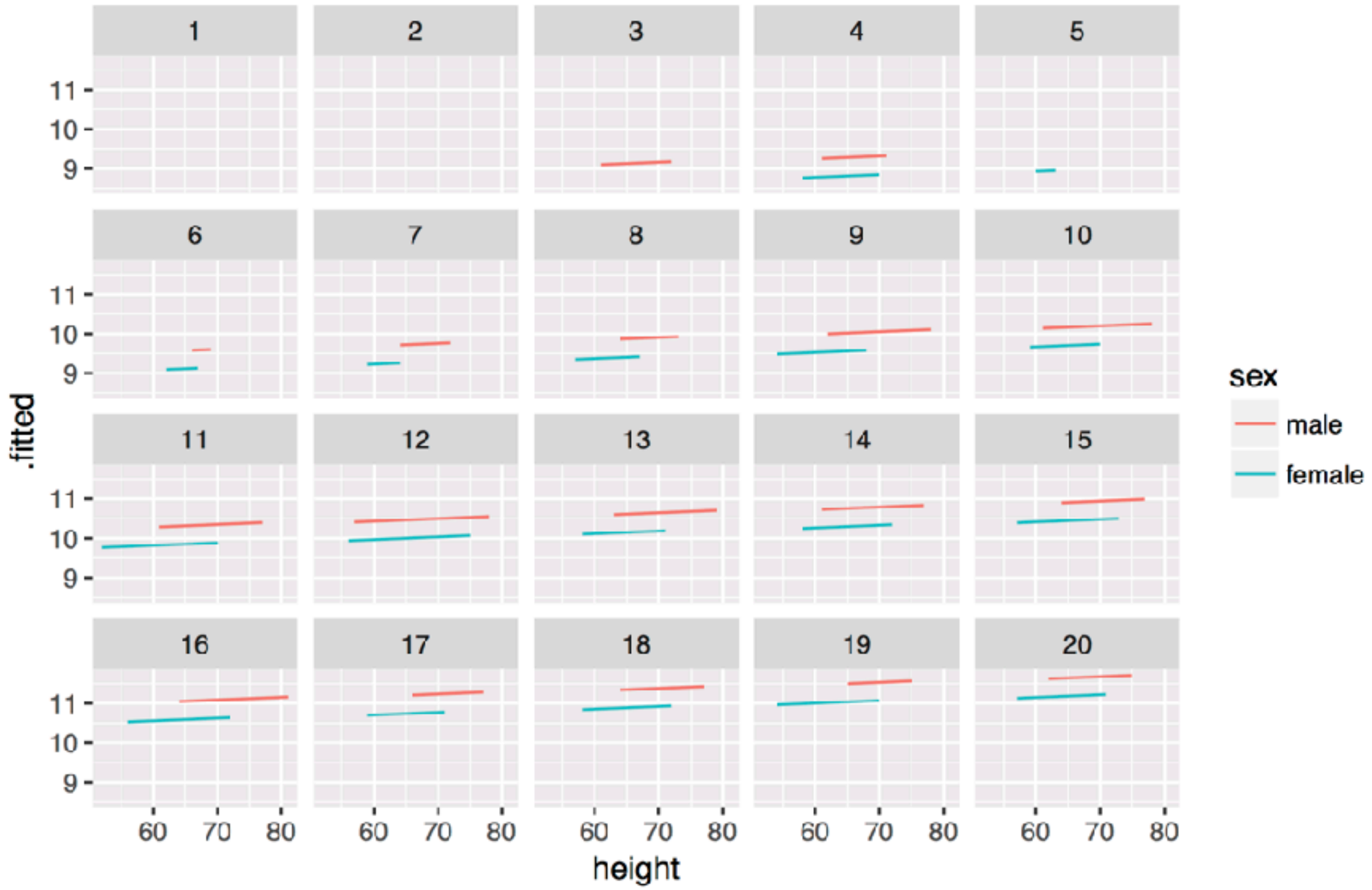
1–10 of 5,266 rows

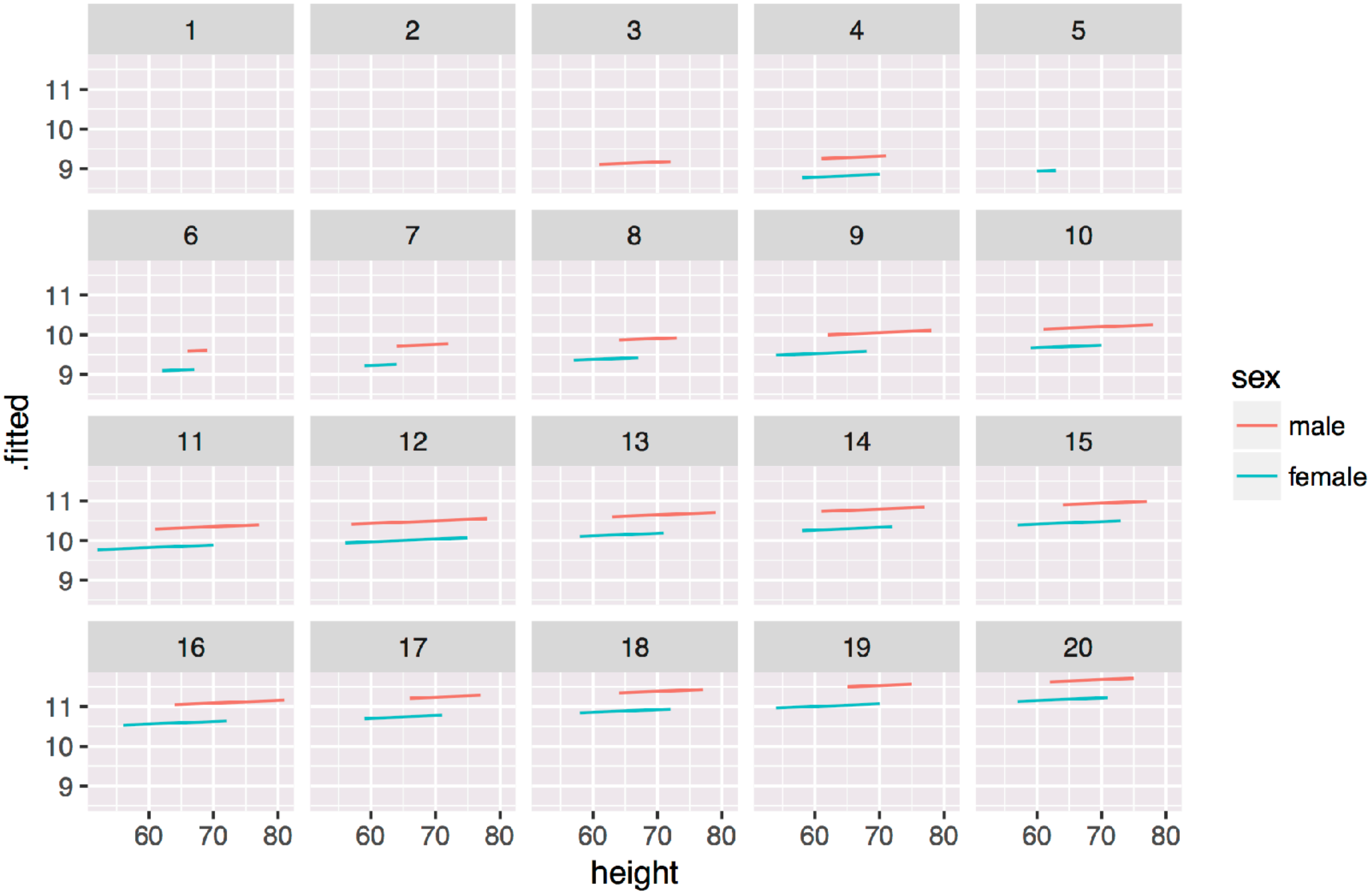
Previous 1 2 3 4 5 6 ... 100

Help Me

How can we use **add_predictions** to visualize the predictions of model_ehs?

```
wages %>%
  add_predictions(mod_ehs) %>%
  ggplot(mapping = aes(x = height, y = pred, color = sex)) +
    geom_line() +
    facet_wrap(~ education)
```

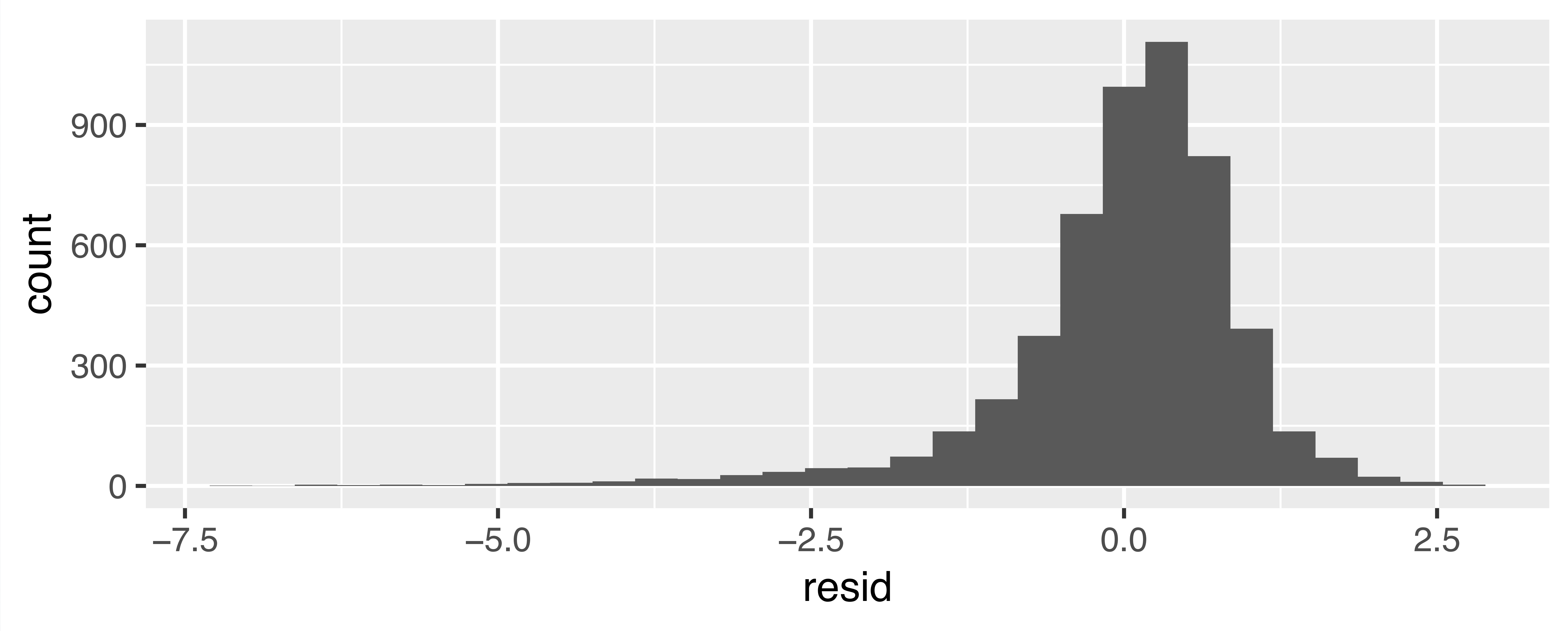



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
11
10
9
60
70
80
.fitted
height
sex
male
female

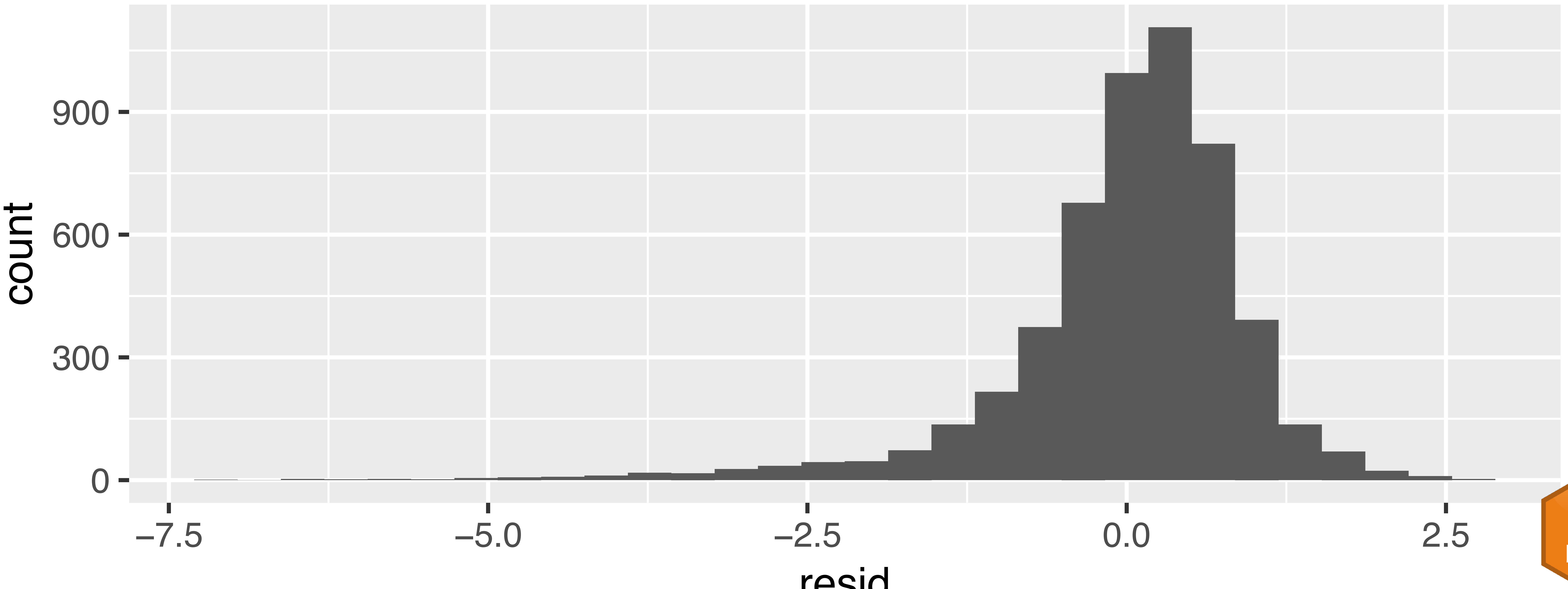
broom

Your Turn 5

Use **add_residuals** to make this histogram of the residuals of **mod_ehs**.



```r
wages %>%
  add_residuals(mod_ehs) %>%
  ggplot() +
    geom_histogram(aes(resid))
```



Strategy

To compare models,

1. Use add_residuals repeatedly to add residuals from each model

```
wages %>% add_residual(mod_h, var = "mod_h")
```

income <int>	height <dbl>	sex <fctr>	education <int>	mod_h <dbl>
19000	60	female	13	-0.2499641042
35000	70	female	10	-0.1588437767
105000	65	male	16	1.1996628894
40000	63	female	14	0.3385397443
75000	66	male	14	0.8112117773
102000	68	female	18	1.0147387260
70000	64	female	12	0.8461766568
60000	69	male	12	0.4321315995
150000	69	male	13	1.3484223314
115000	64	female	16	1.3426135431

Strategy

To compare models,

1. Use add_residuals repeatedly to add residuals from each model

```
… %>% add_residual(mod_eh, var = "mod_eh")
```

income <int>	height <dbl>	sex <fctr>	education <int>	mod_h <dbl>	mod_eh <dbl>
19000	60	female	13	-0.2499641042	-0.197967581
35000	70	female	10	-0.1588437767	0.345993609
105000	65	male	16	1.1996628894	0.853872025
40000	63	female	14	0.3385397443	0.262834114
75000	66	male	14	0.8112117773	0.746516842
102000	68	female	18	1.0147387260	0.402532860
70000	64	female	12	0.8461766568	1.051566955
60000	69	male	12	0.4321315995	0.655873056
150000	69	male	13	1.3484223314	1.433450940
115000	64	female	16	1.3426135431	0.993152447

Strategy

To compare models,

1. Use add_residuals repeatedly to add residuals from each model

```
… %>% add_residual(mod_ehs, var = "mod_ehs")
```

income <int>	height <dbl>	sex <fctr>	education <int>	mod_h <dbl>	mod_eh <dbl>	mod_ehs <dbl>
19000	60	female	13	-0.2499641042	-0.197967581	-0.2638576582
35000	70	female	10	-0.1588437767	0.345993609	0.7237344726
105000	65	male	16	1.1996628894	0.853872025	0.5063344524
40000	63	female	14	0.3385397443	0.262834114	0.3124199119
75000	66	male	14	0.8112117773	0.746516842	0.4591017275
102000	68	female	18	1.0147387260	0.402532860	0.6229479494
70000	64	female	12	0.8461766568	1.051566955	1.1612752115
60000	69	male	12	0.4321315995	0.655873056	0.5117444599
150000	69	male	13	1.3484223314	1.433450940	1.2800521289
115000	64	female	16	1.3426135431	0.993152447	1.0657798463

Strategy

To compare models,

1. Use add_residuals repeatedly to add residuals from each model
2. Gather the residuals into model and resid columns

```
… %>% gather(key = "model", value = "resid", …)
```

income <int>	height <dbl>	sex <fctr>	education <int>	model <chr>	resid <dbl>
19000	60	female	13	mod_h	-0.2499641042
19000	60	female	13	mod_eh	-0.1979675815
19000	60	female	13	mod_ehs	-0.2638576582
115000	64	female	16	mod_h	1.3426135431
115000	64	female	16	mod_eh	0.9931524472
115000	64	female	16	mod_ehs	1.0657798463

Strategy

To compare models,

1. Use add_residuals repeatedly to add residuals from each model
2. Gather the residuals into model and resid columns
3. Plot resid and facet by model

```
… %>%
  ggplot() +
    geom_histogram(aes(resid)) +
    facet_grid(model ~ .)
```

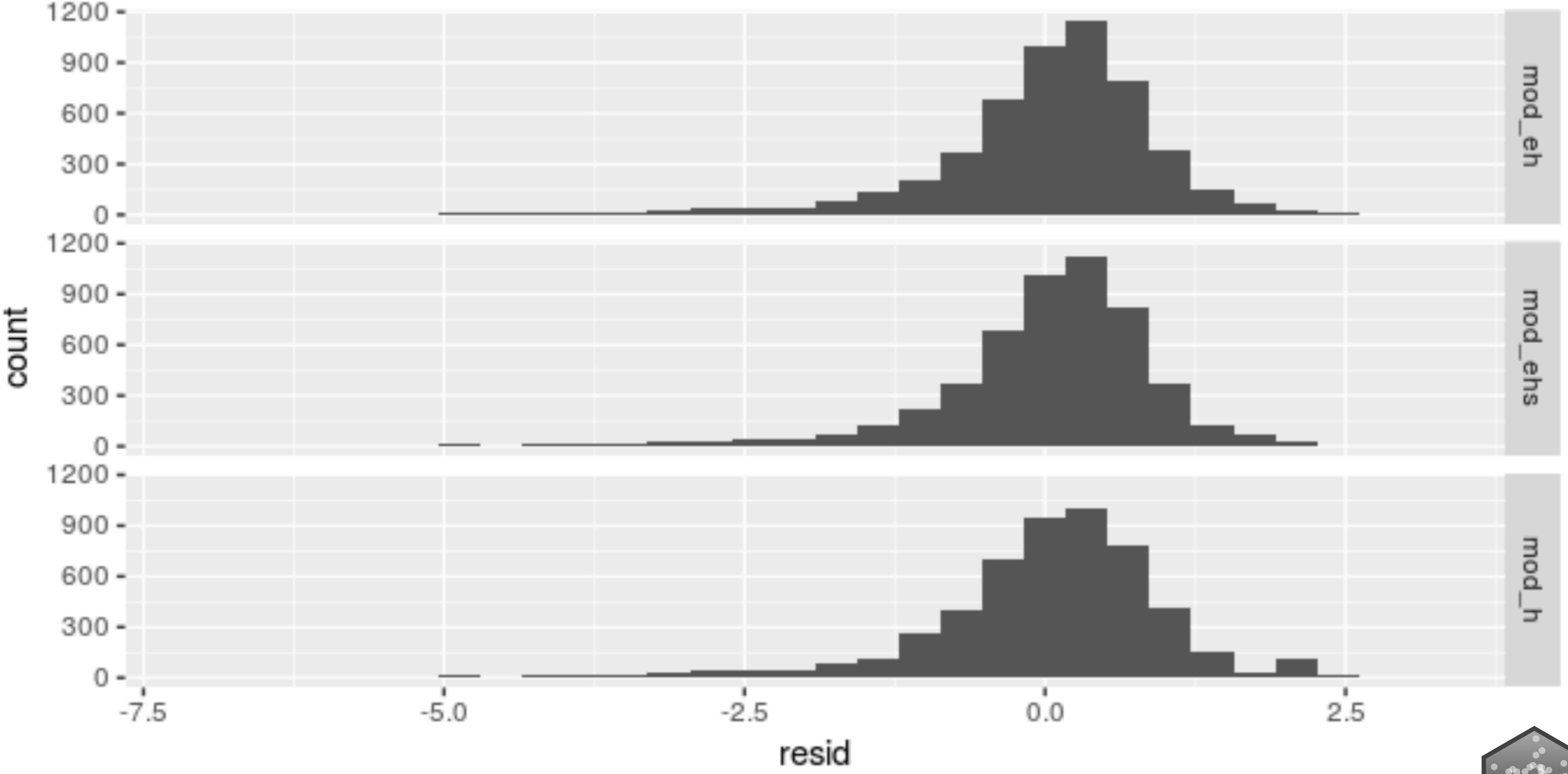
Your Turn 6

Work together to complete the code and make a plot that compares the residuals of all three models.

05:00

```r
wages %>%
  add_residuals(mod_h, var = "mod_h") %>%
  add_residuals(mod_eh, var = "mod_eh") %>%
  add_residuals(mod_ehs, var = "mod_ehs") %>%
  gather(key = "model", value = "resid",
    mod_h, mod_eh, mod_ehs) %>%
  ggplot() +
    geom_histogram(aes(resid)) +
    facet_grid(model ~ .)
```

modelr



1200
900
600
300
0
mod_eh
mod_ehs
mod_h
count
-7.5
-5.0
-2.5
0.0
2.5
resid

spread_residuals()

Adds residuals for multiple models, each in their own column.

```
spread_residuals(data, ...)
```

Adds residuals from each of these models

To the cases in this data frame

```r
wages %>%
  spread_residuals(mod_h, mod_eh, mod_ehs)
```

income <int>	height <dbl>	sex <fctr>	education <int>	mod_h <dbl>	mod_eh <dbl>	mod_ehs <dbl>
19000	60	female	13	-0.2499641042	-0.197967581	-0.2638576582
35000	70	female	10	-0.1588437767	0.345993609	0.7237344726
105000	65	male	16	1.1996628894	0.853872025	0.5063344524
40000	63	female	14	0.3385397443	0.262834114	0.3124199119
75000	66	male	14	0.8112117773	0.746516842	0.4591017275
102000	68	female	18	1.0147387260	0.402532860	0.6229479494
70000	64	female	12	0.8461766568	1.051566955	1.1612752115
60000	69	male	12	0.4321315995	0.655873056	0.5117444599
150000	69	male	13	1.3484223314	1.433450940	1.2800521289
115000	64	female	16	1.3426135431	0.993152447	1.0657798463

1-10 of 5,266 rows

Previous 1 2 3 4 5 6 ... 100 Next

gather_residuals()

Adds residuals for multiple models as a pair of key:value columns (model:resid)

```
gather_residuals(data, ...)
```

Adds residuals from each of these models

To the cases in this data frame (duplicating rows as necessary)

modelr

```r
wages %>%
  gather_predictions(mod_h, mod_eh, mod_ehs)
```

income	height	sex	education	model	resid
<int>	<dbl>	<fctr>	<int>	<chr>	<dbl>
19000	60	female	13	mod_h	-0.2499641042
19000	60	female	13	mod_eh	-0.1979675815
19000	60	female	13	mod_ehs	-0.2638576582
115000	64	female	16	mod_h	1.3426135431
115000	64	female	16	mod_eh	0.9931524472
115000	64	female	16	mod_ehs	1.0657798463
41000	65	female	16	mod_h	0.2592746059
41000	65	female	16	mod_eh	-0.0865162582
41000	65	female	16	mod_ehs	0.0276931707
19000	63	female	12	mod_h	-0.4059007306

```r
wages %>%
  add_residuals(mod_h, var = "mod_h") %>%
  add_residuals(mod_eh, var = "mod_eh") %>%
  add_residuals(mod_ehs, var = "mod_ehs") %>%
  gather(key = "model", value = "resid",
    mod_h, mod_eh, mod_ehs) %>%
  ggplot() +
    geom_histogram(aes(resid)) +
    facet_grid(model ~ .)
```

```
wages %>%
  spread_residuals(mod_h, mod_eh, mod_ehs) %>%
  gather(key = "model", value = "resid",
    mod_h, mod_eh, mod_ehs) %>%
  ggplot() +
    geom_histogram(aes(resid)) +
    facet_grid(model ~ .)
```

modelr

```r
wages %>%
  spread_residuals(mod_h, mod_eh, mod_ehs) %>%
  gather(key = "model", value = "resid",
    mod_h, mod_eh, mod_ehs) %>%
  ggplot() +
    geom_histogram(aes(resid)) +
    facet_grid(model ~ .)
```

modelr

```
wages %>%
  gather_residuals(mod_h, mod_eh, mod_ehs) %>%
  ggplot() +
    geom_histogram(aes(resid)) +
    facet_grid(model ~ .)
```

Residuals

Modelr provides the equivalent functions for residuals

`add_predictions()` ⟷ `add_residuals()`

`spread_predictions()` ⟷ `spread_residuals()`

`gather_predictions()` ⟷ `gather_residuals()`

Recap

Use **glance()**, **tidy()**, and **augment()** to return model values in a data frame.

Use **add_predictions()** or **gather_predictions()** or **spread_predictions()** to visualize predictions.

Use **add_residuals()** or **gather_residuals()** or **spread_residuals()** to visualize residuals.

Modeling with

broom

modelr