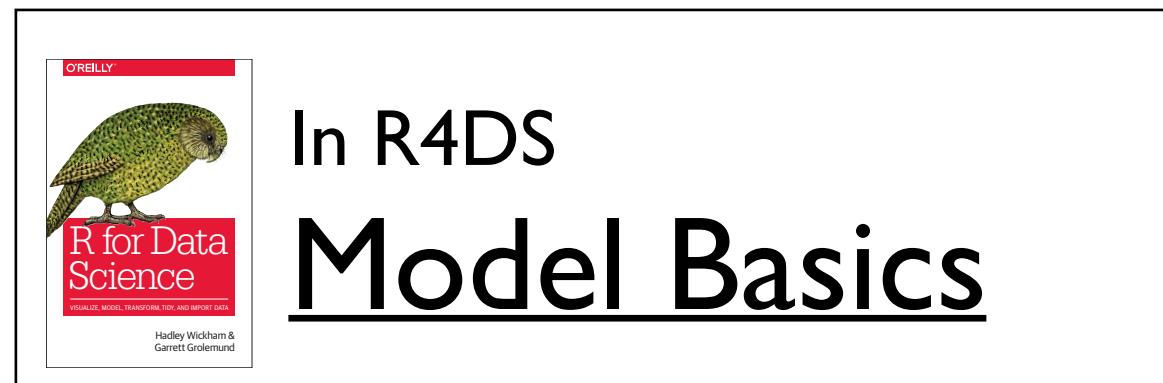
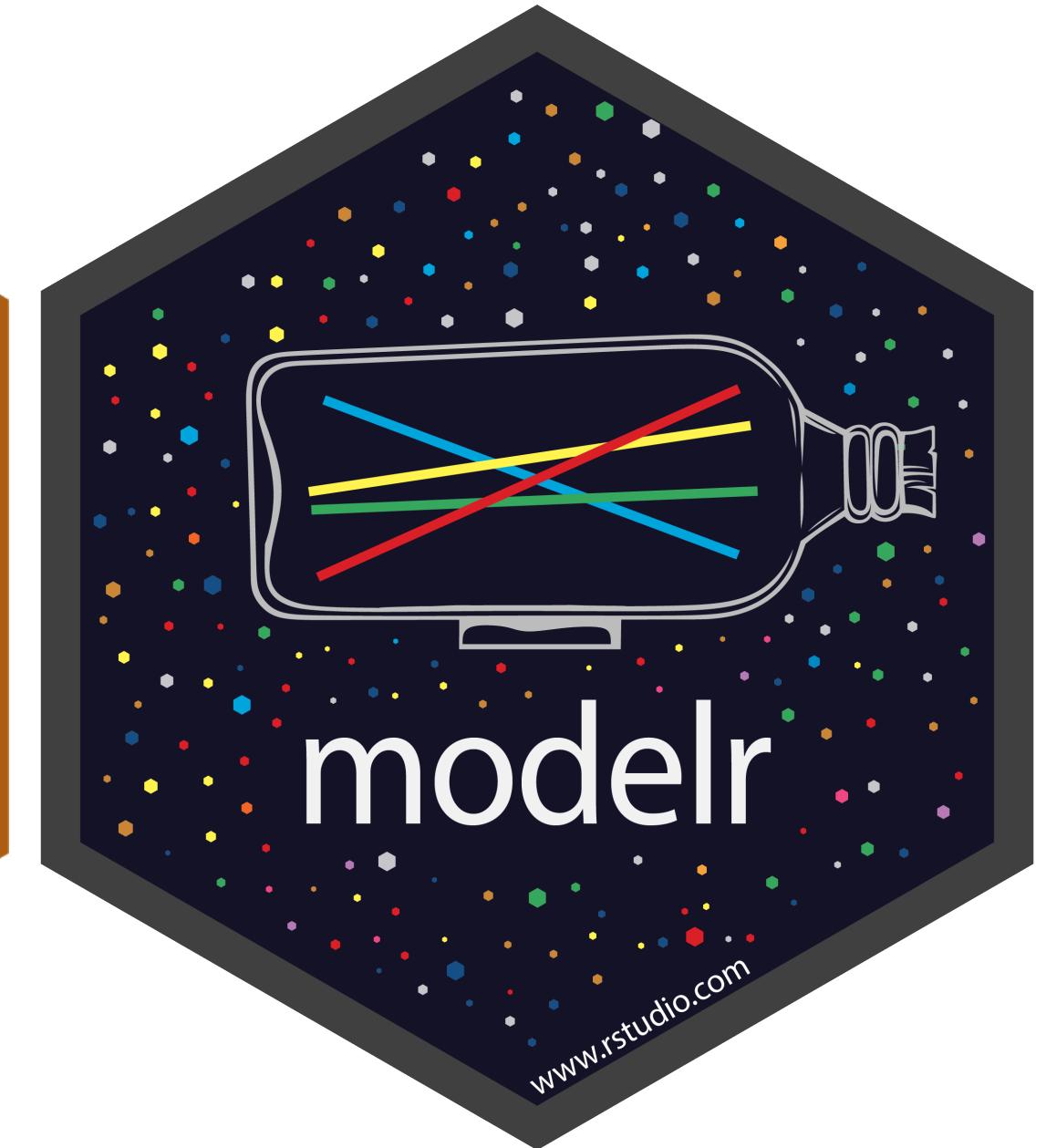
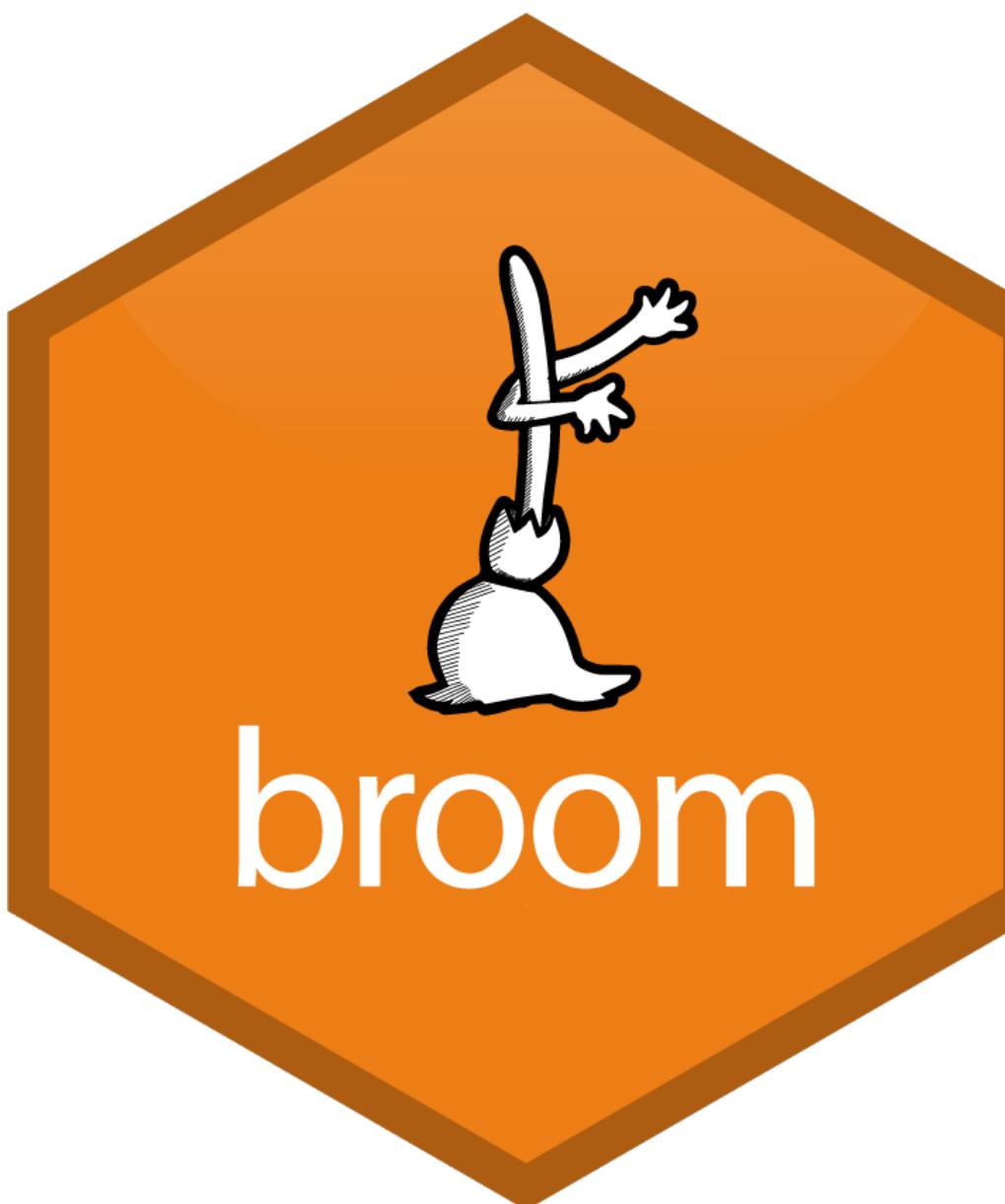


Modeling with



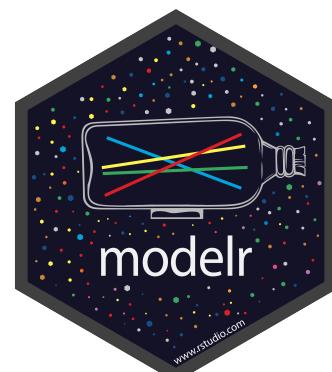
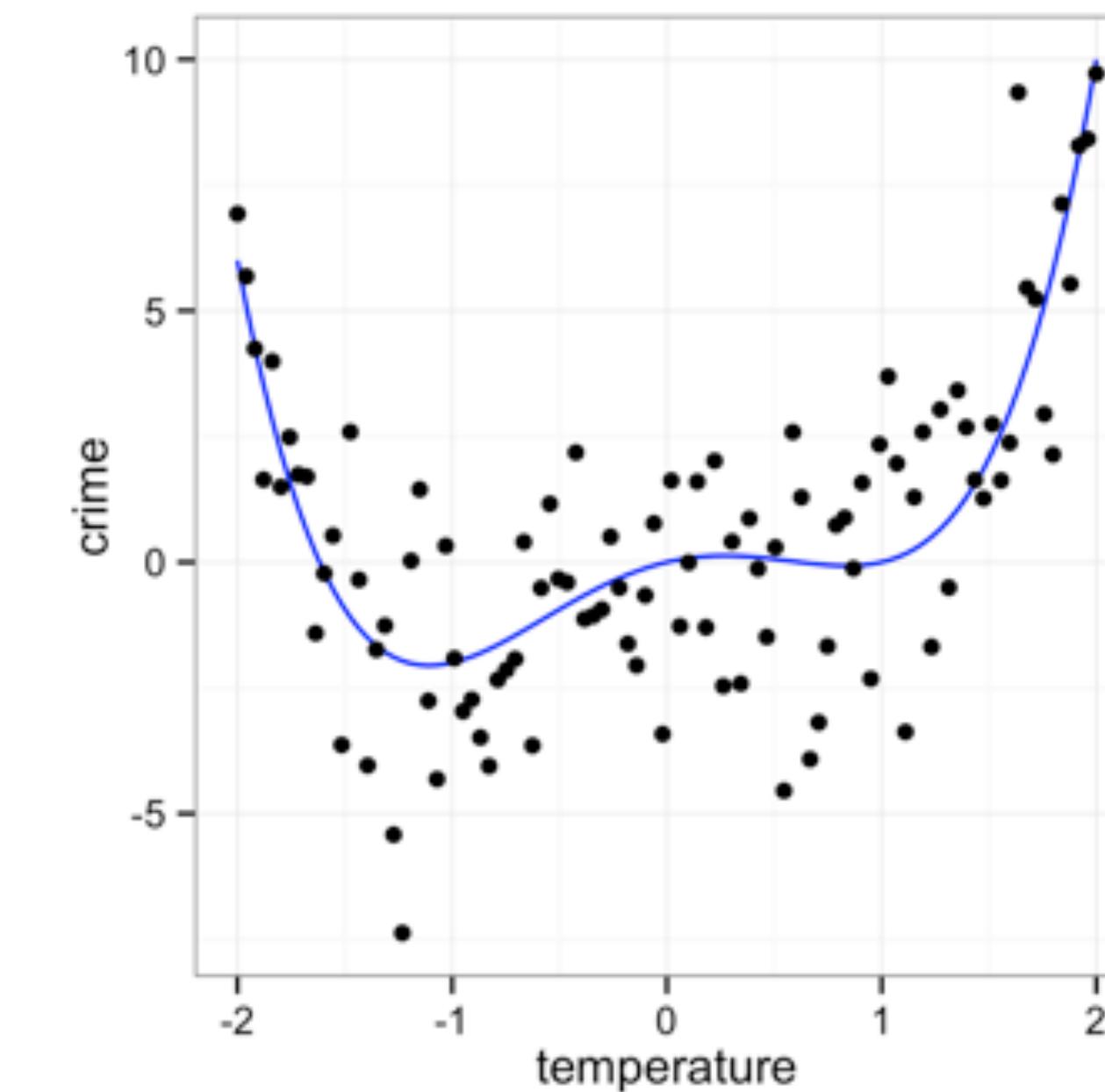
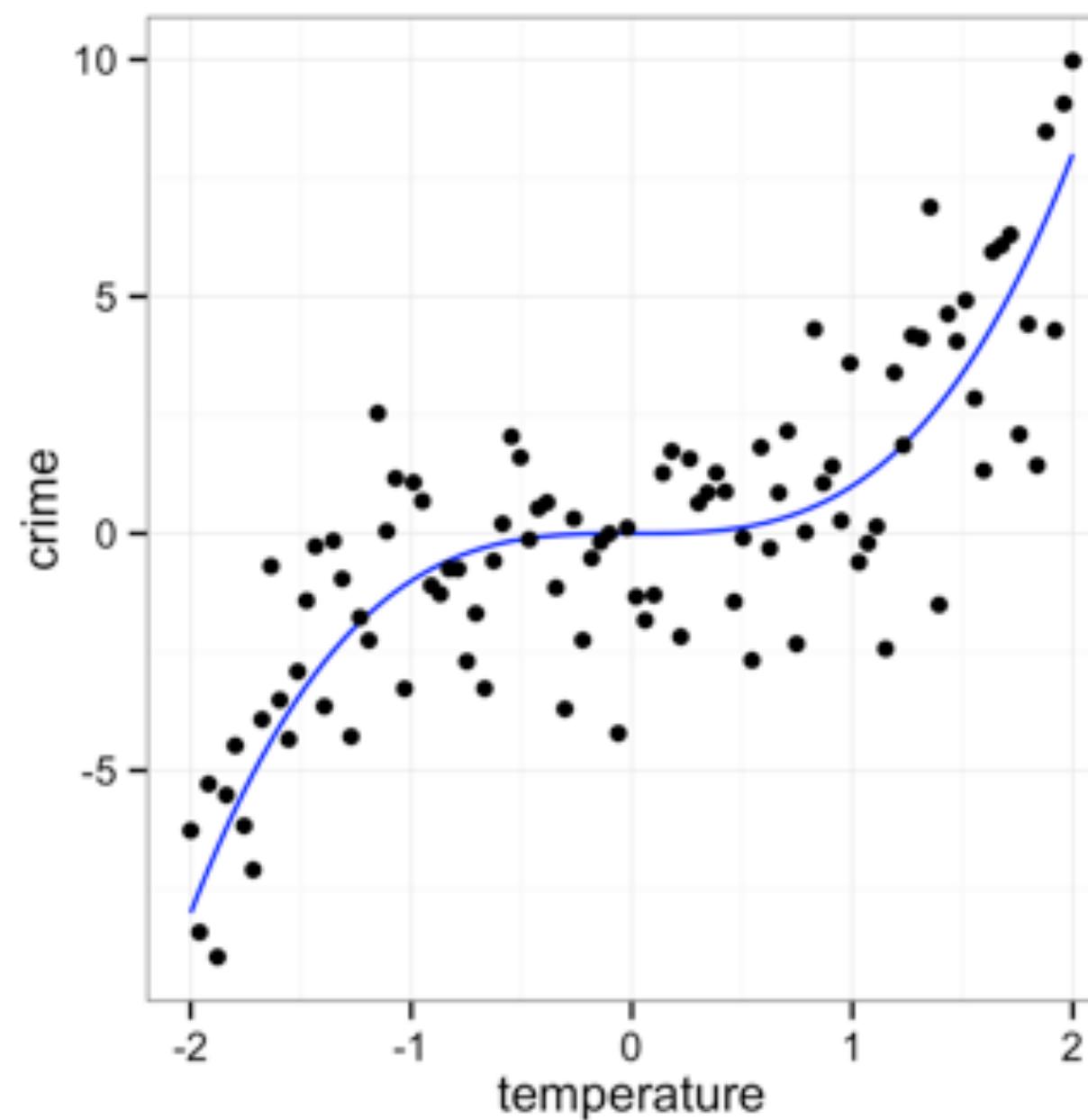
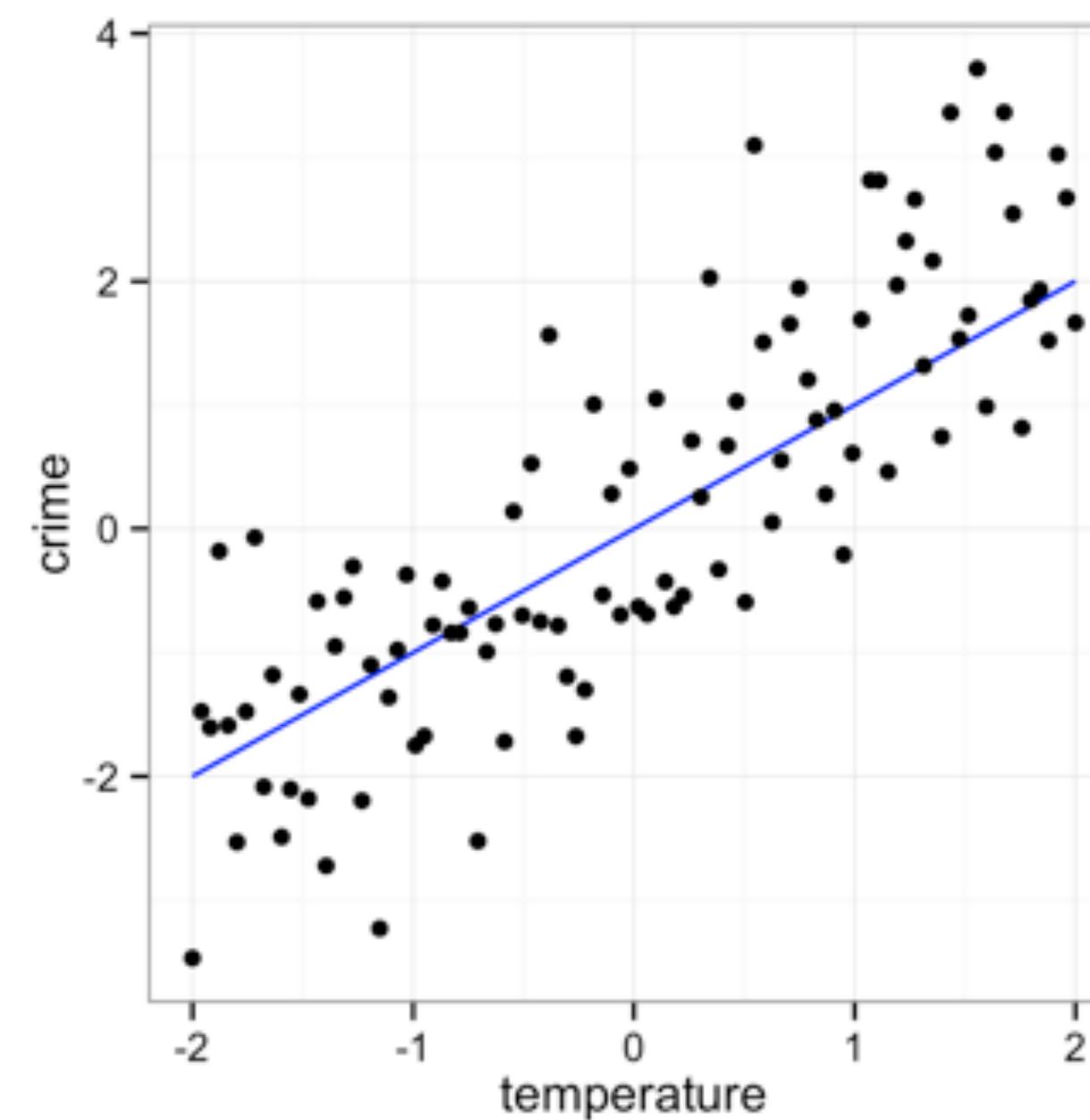
Open 07-Model.rmd

The basics

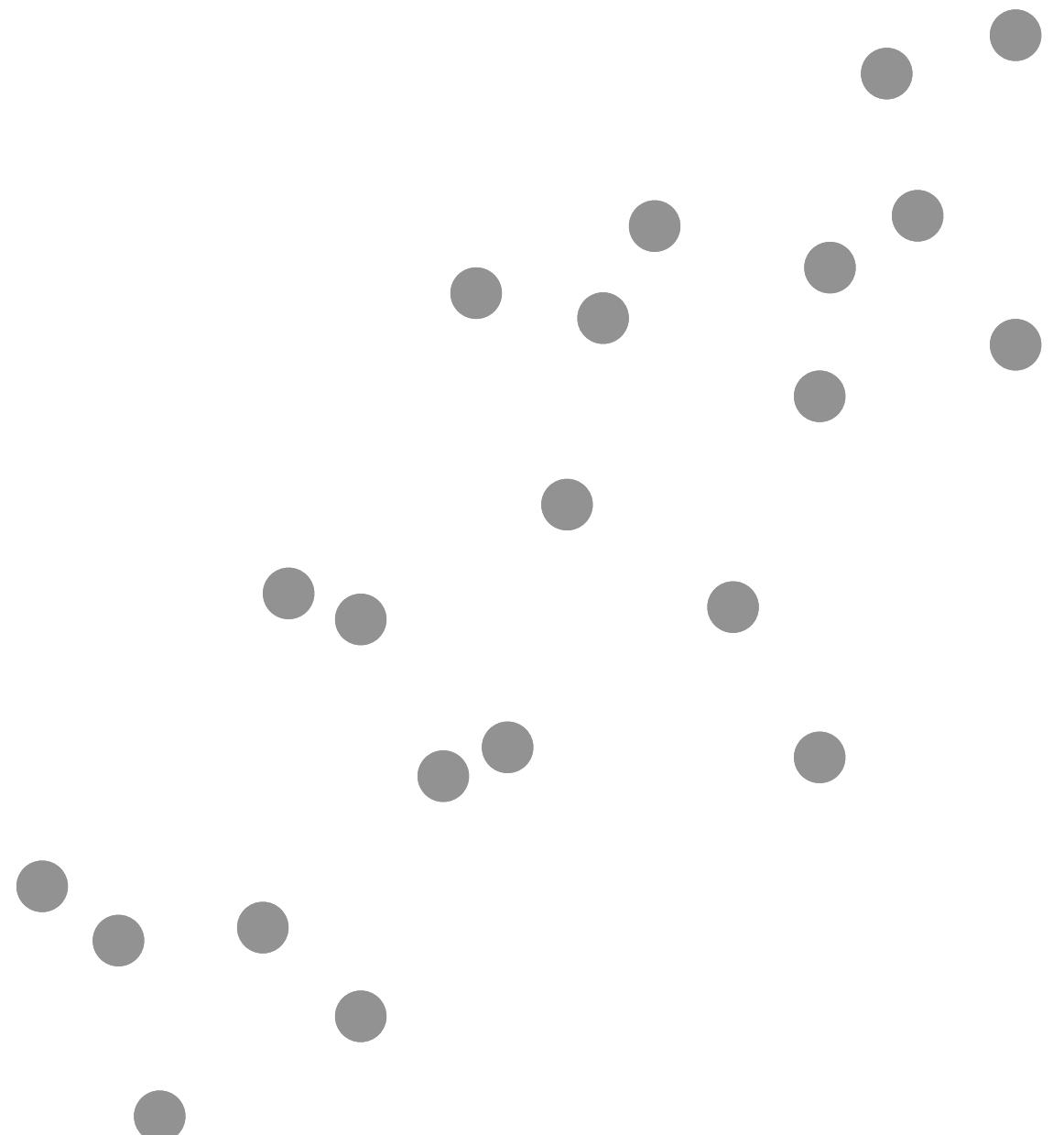
R

Models

A low dimensional description of a higher dimensional data set.



Models



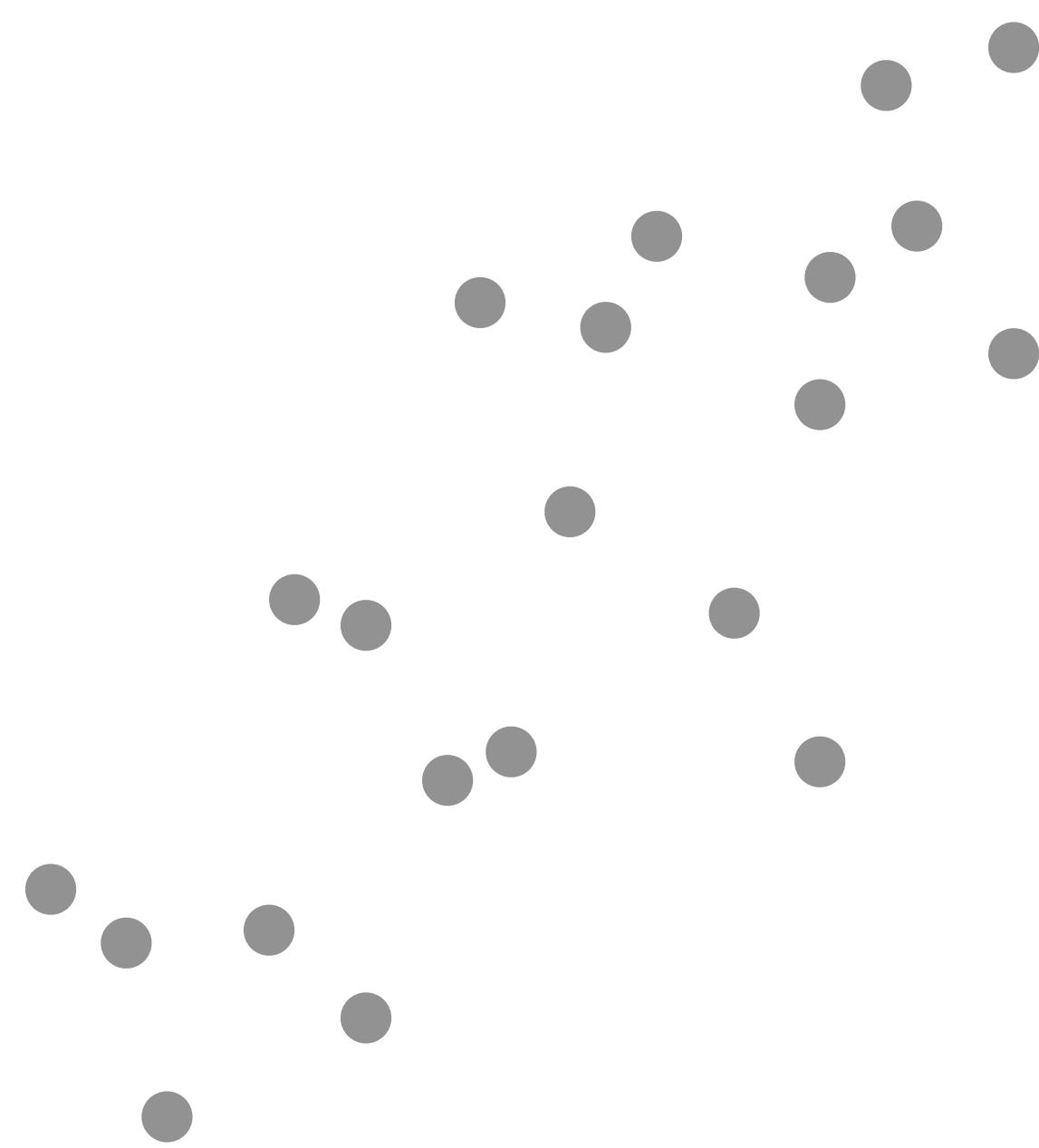
Algorithm

Data

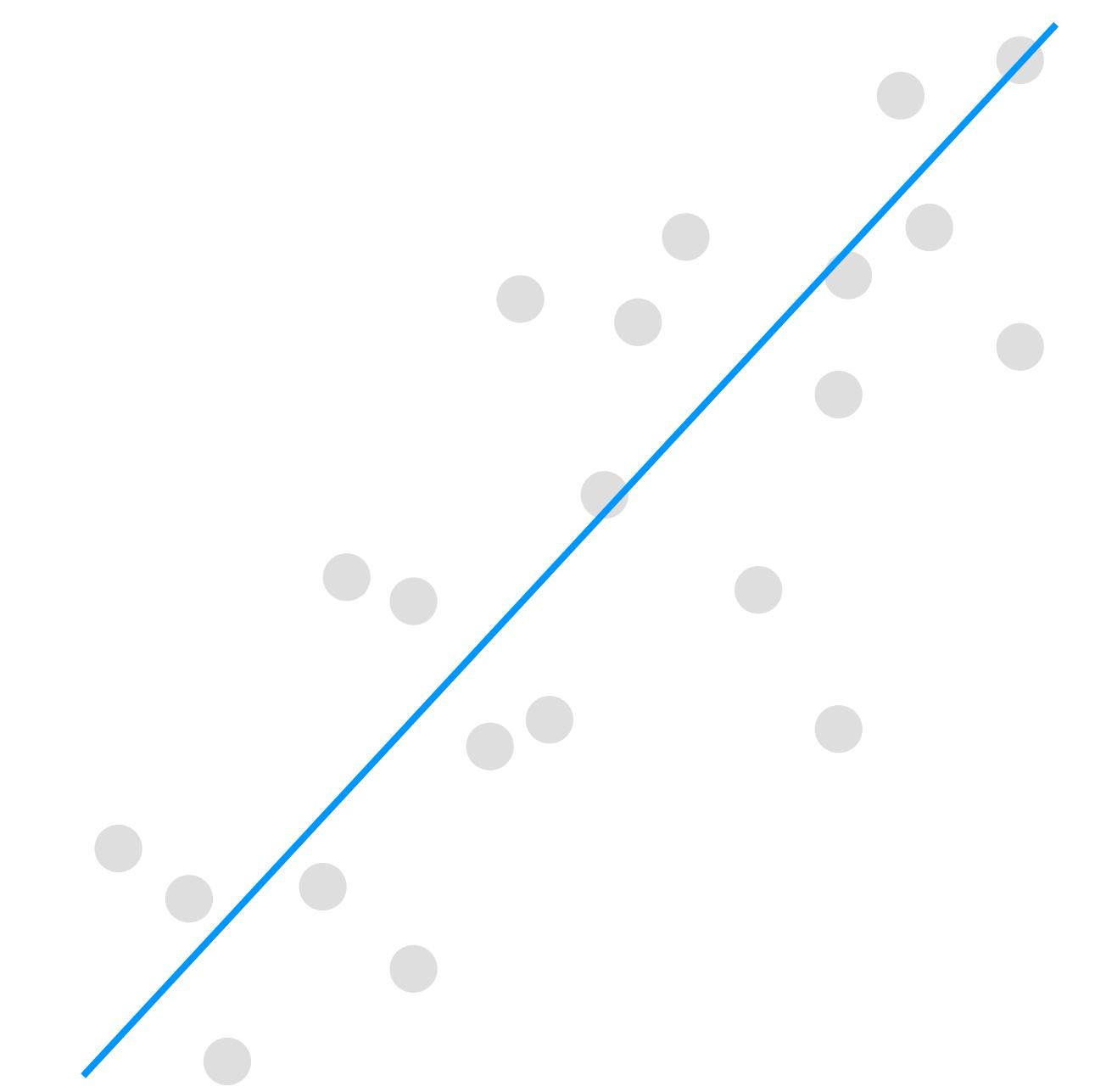
Model Function

Models

What is the **model function**?



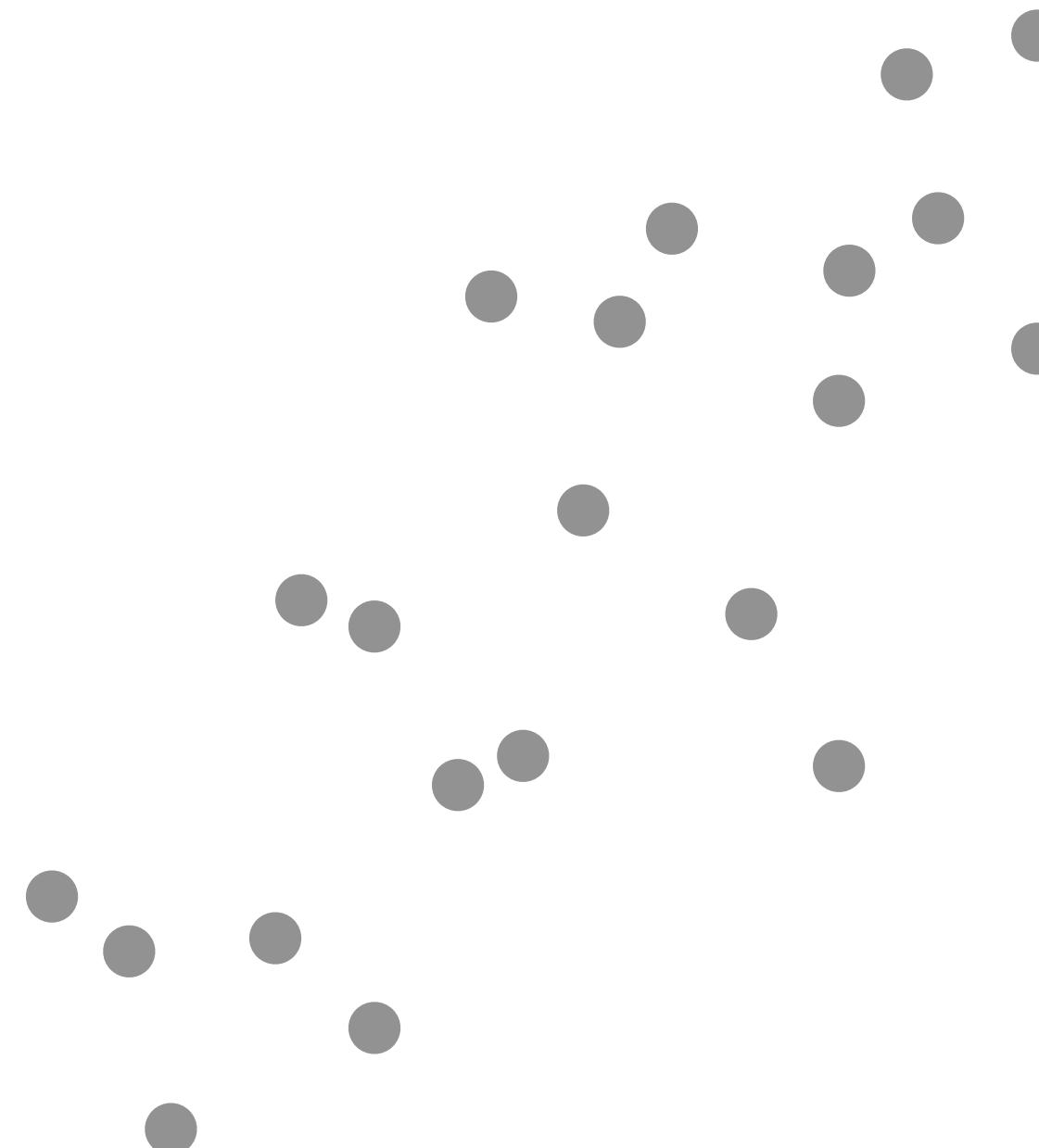
Data



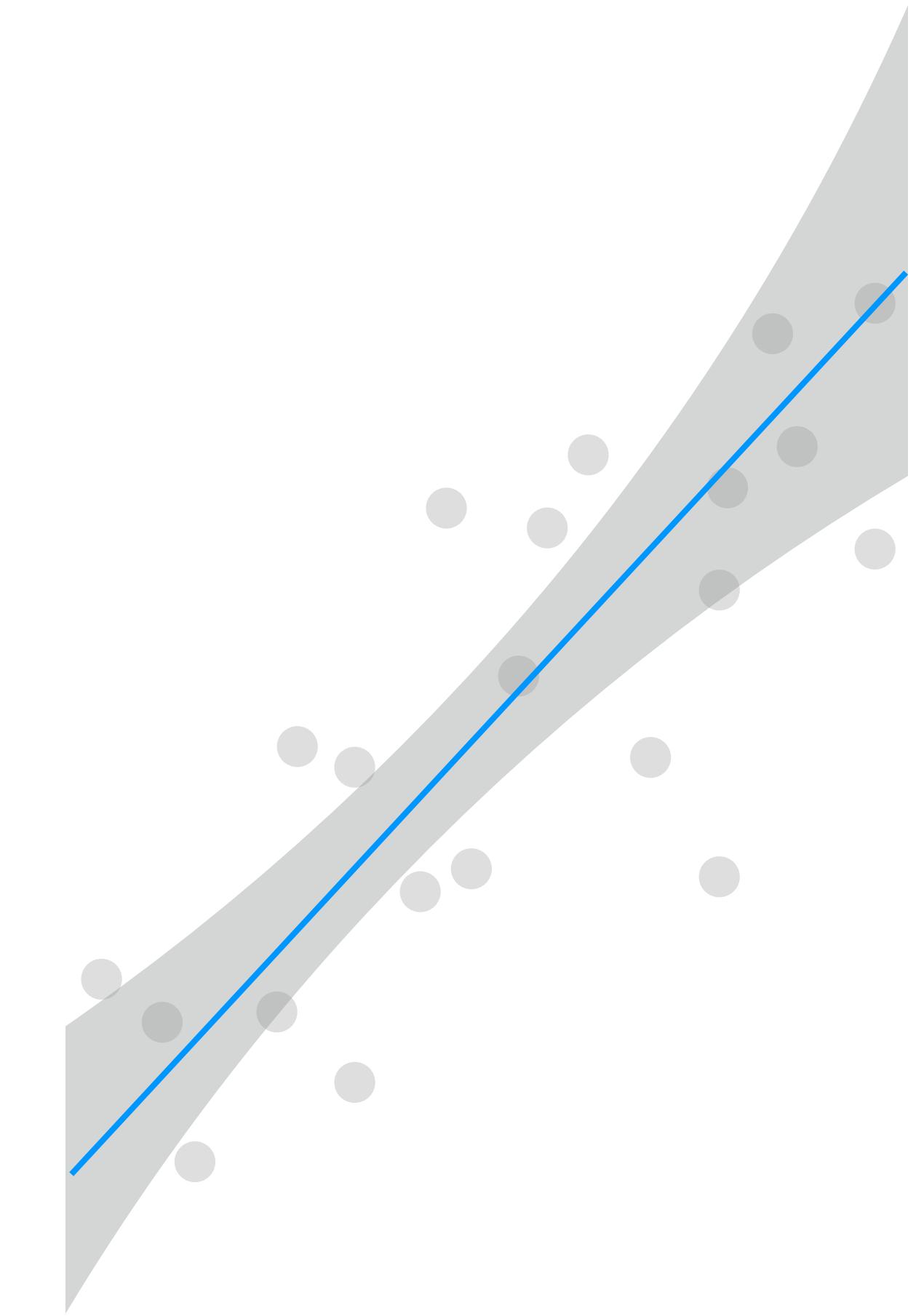
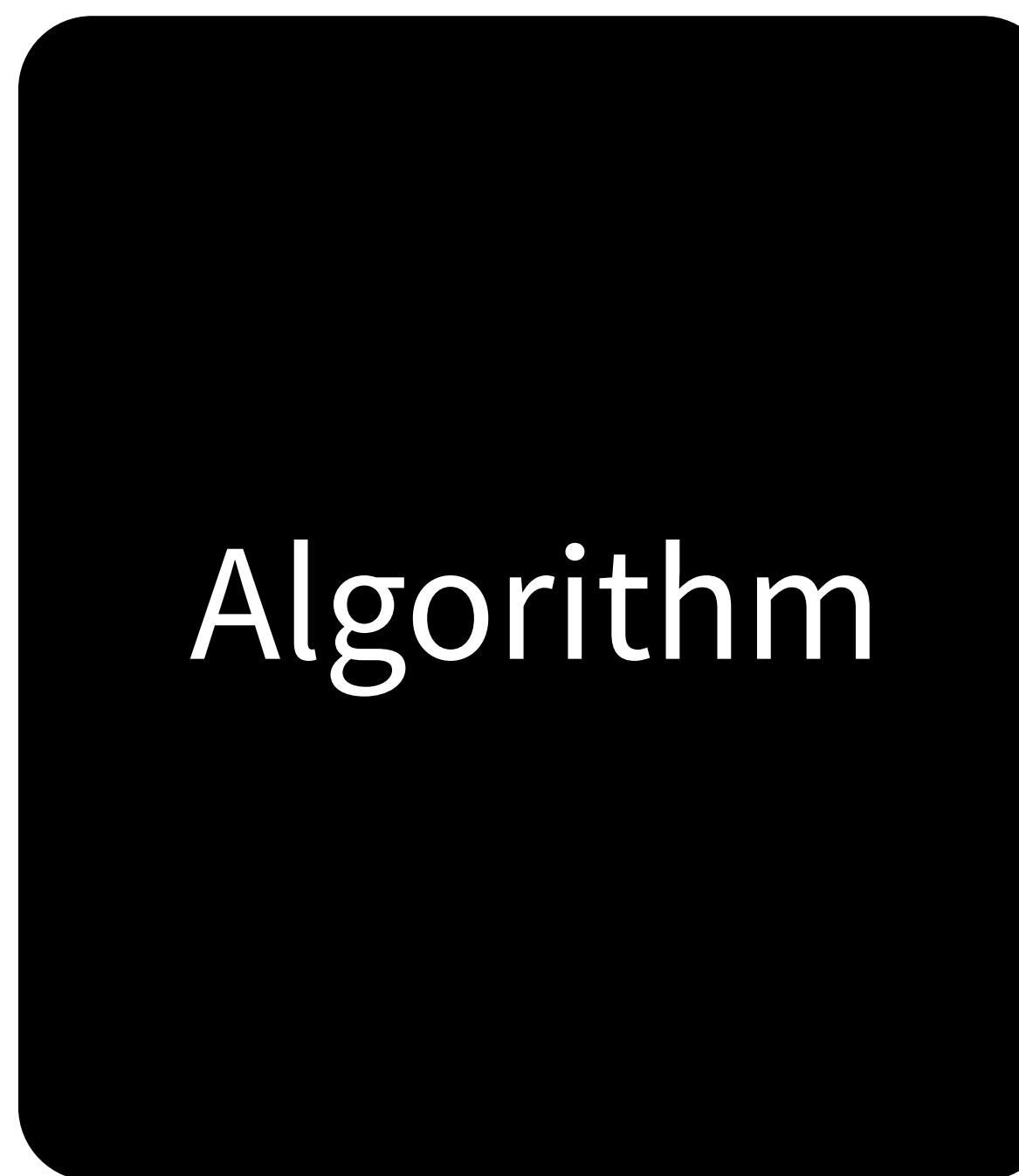
Model Function

Models

What **uncertainty** is associated with it?



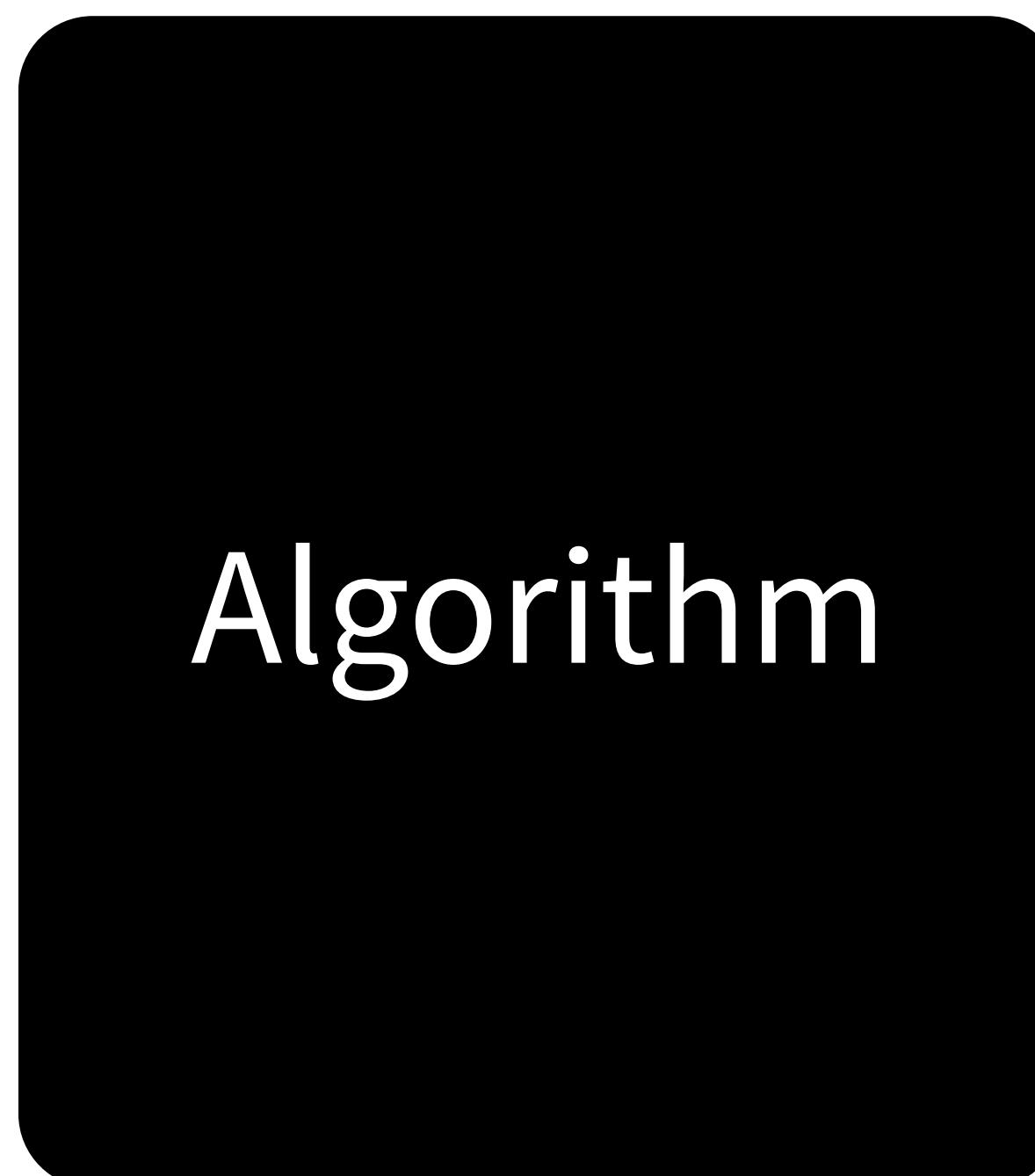
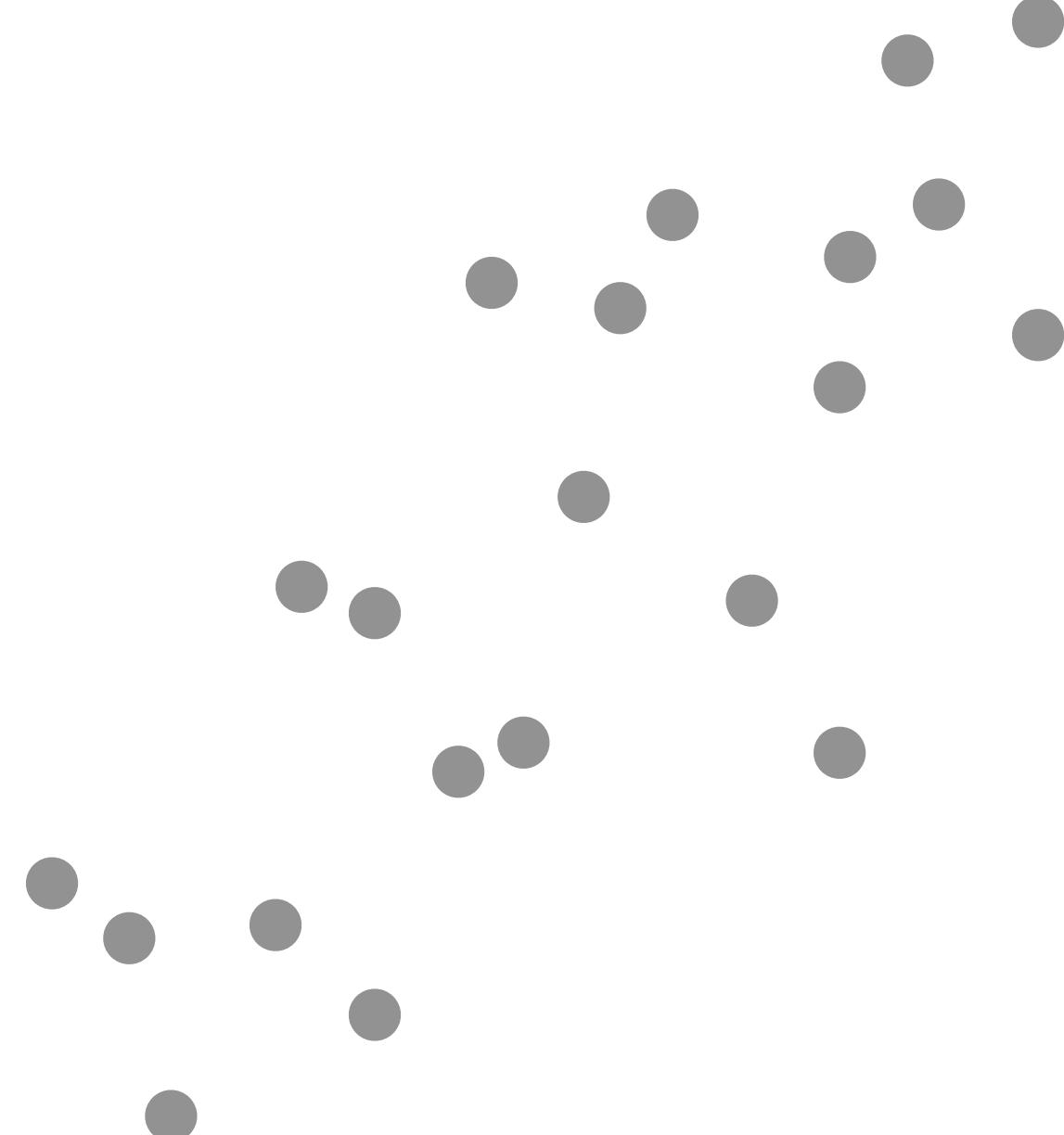
Data



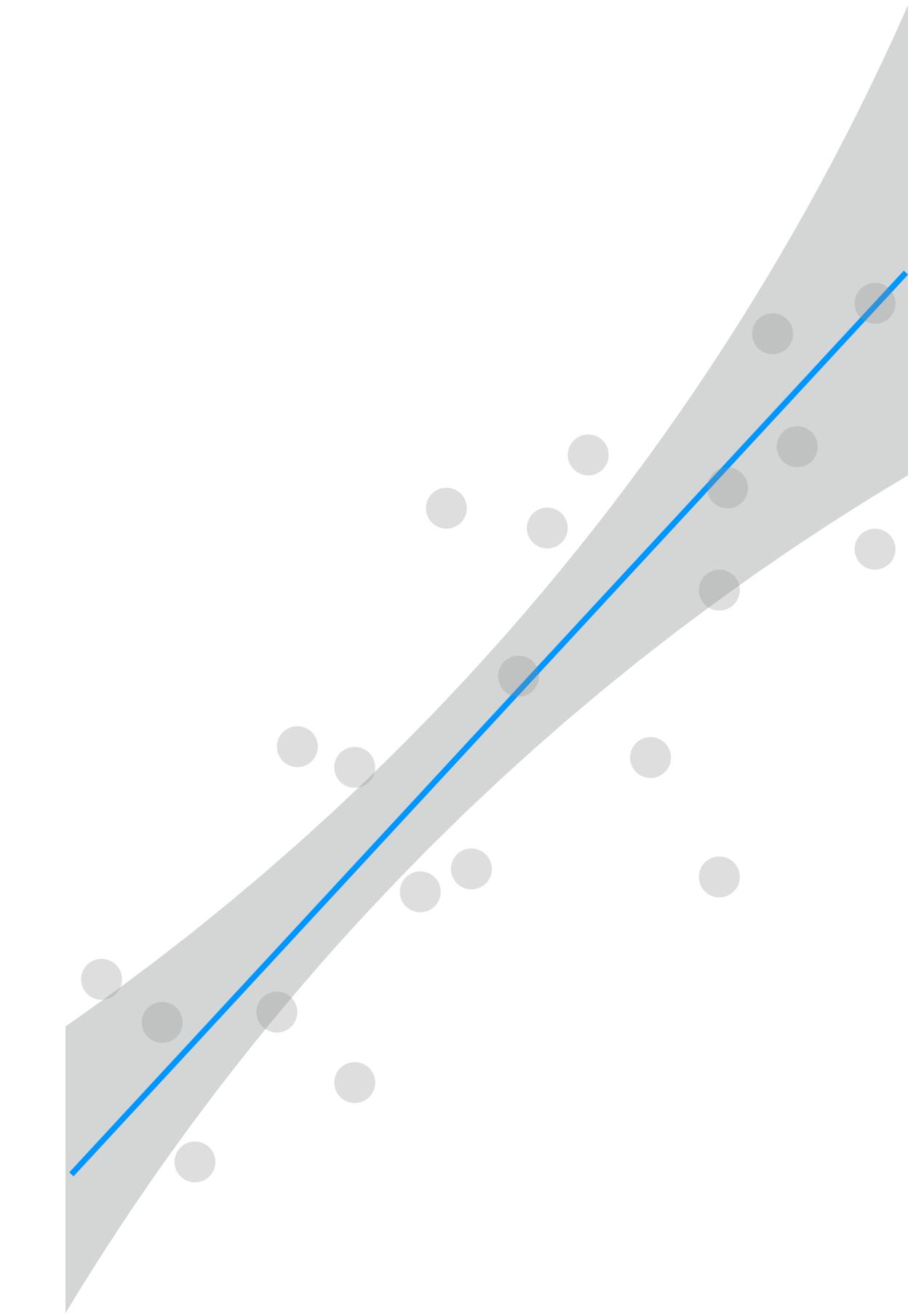
Model Function

Models

How "good" is the model?



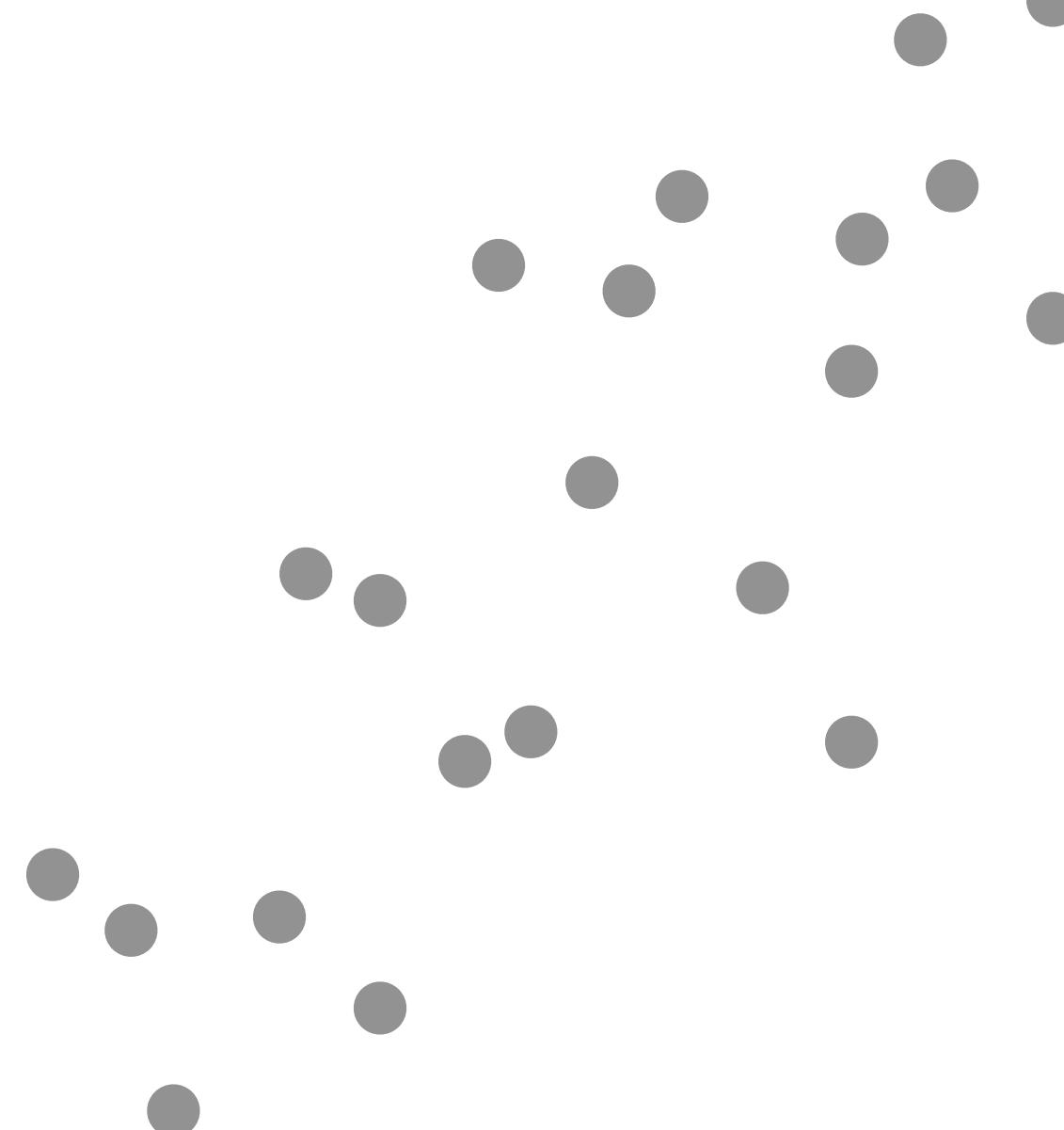
Data



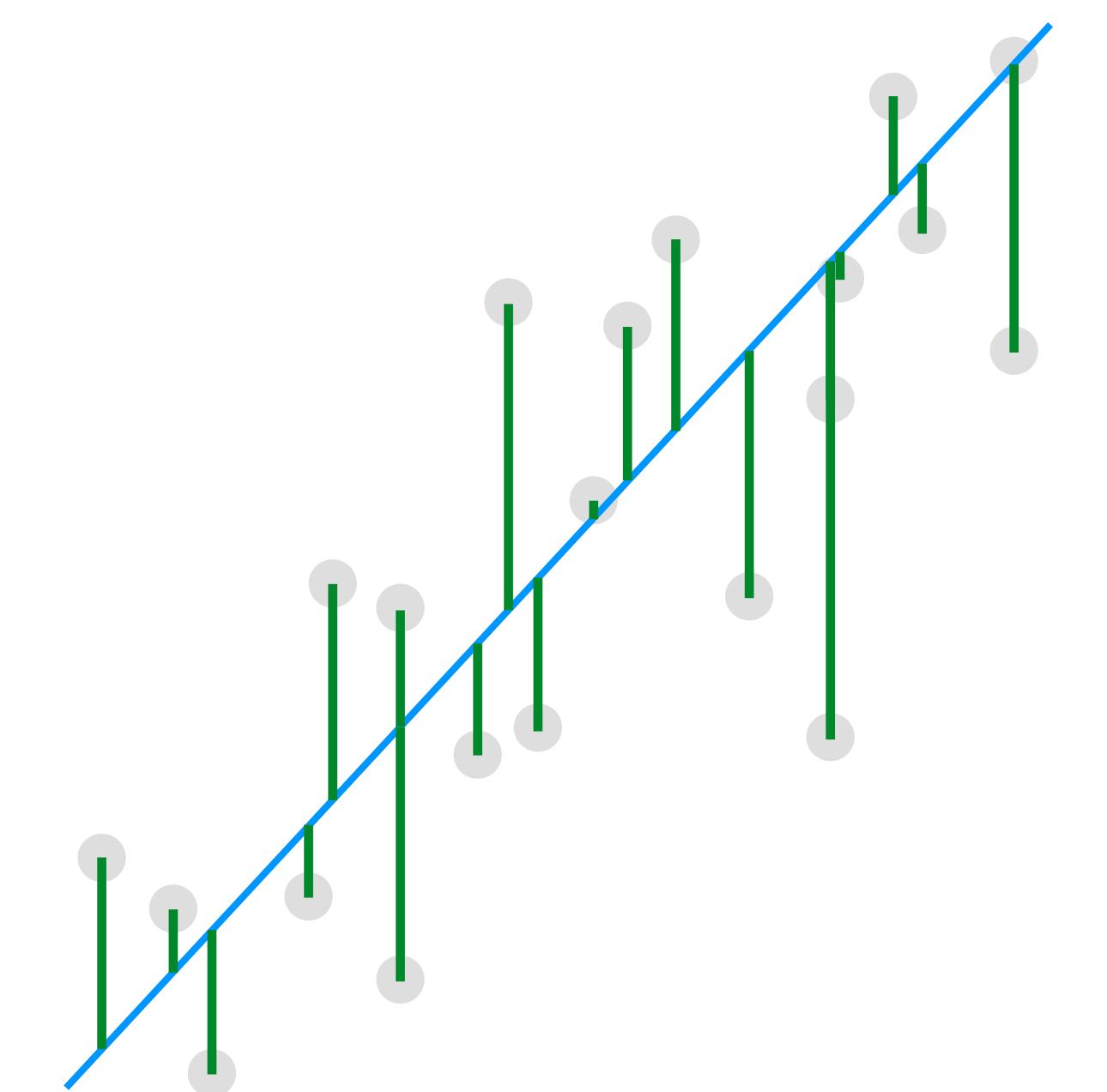
Model Function

Models

What are the **residuals**?



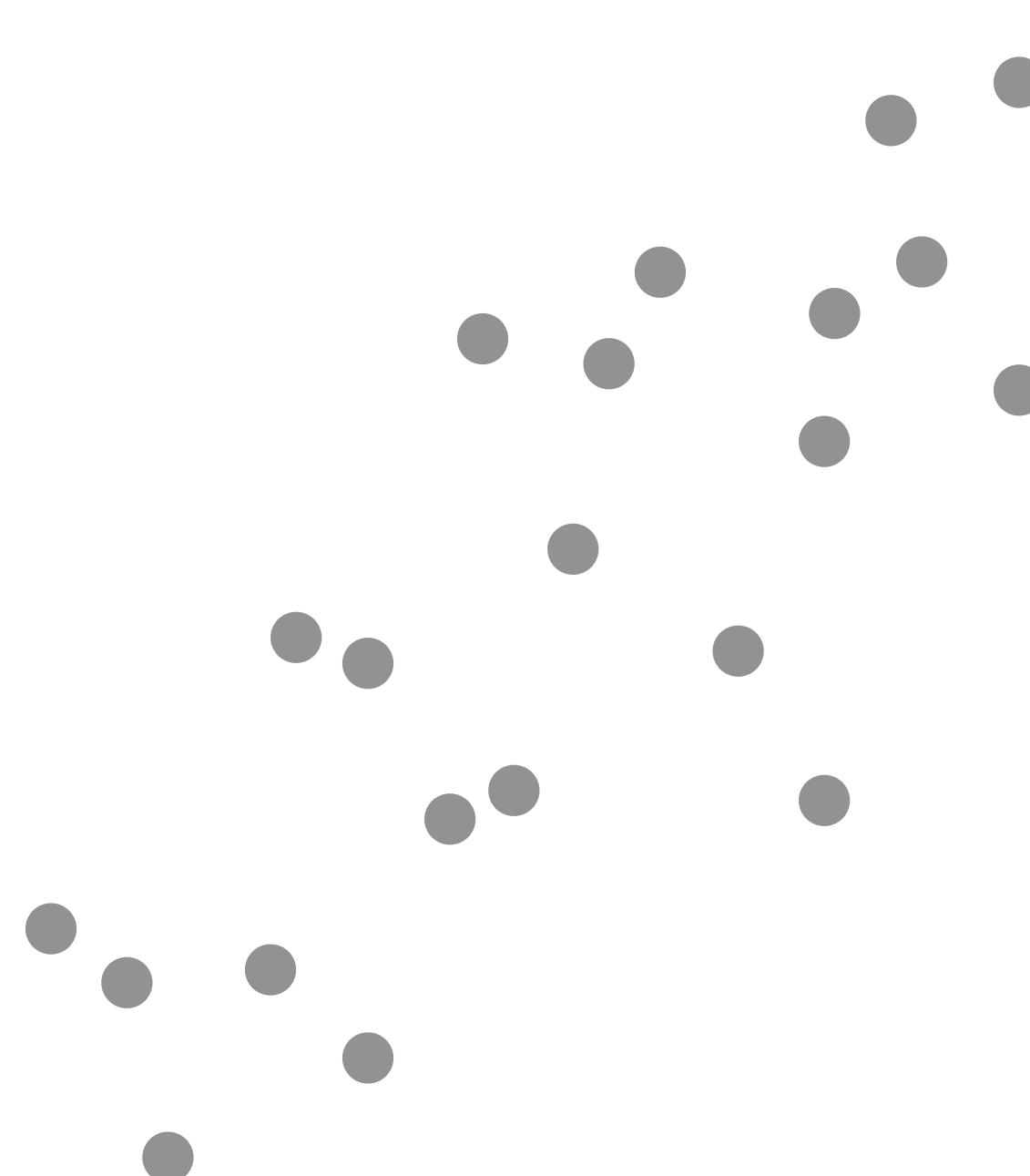
Data



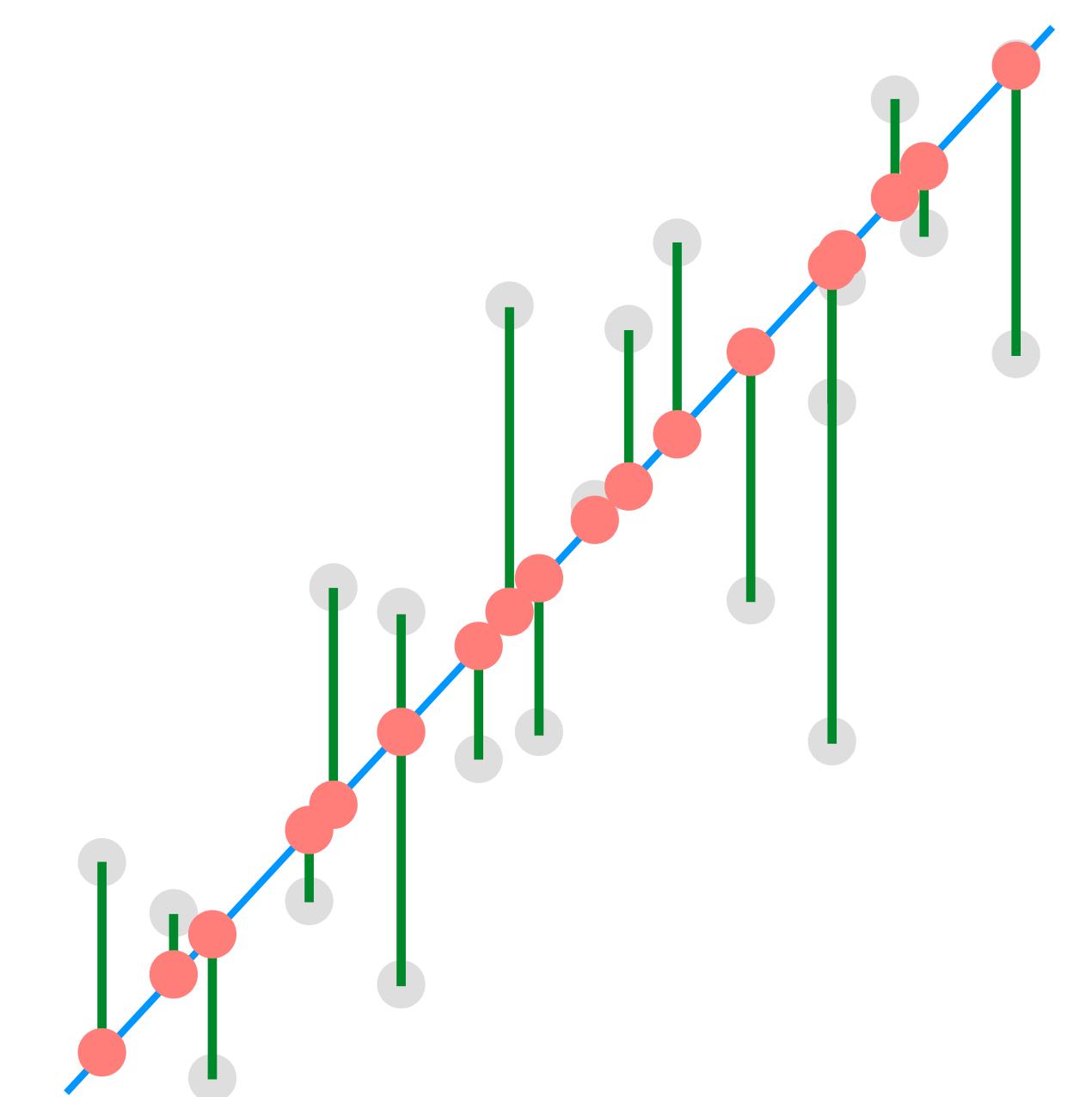
Model Function

Models

What are the **predictions**?



Data

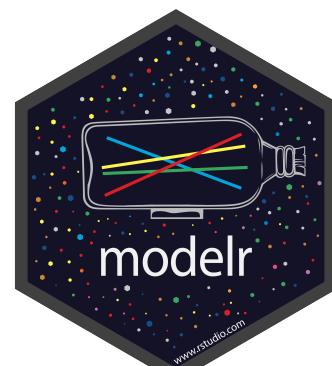


Model Function

Algorithm

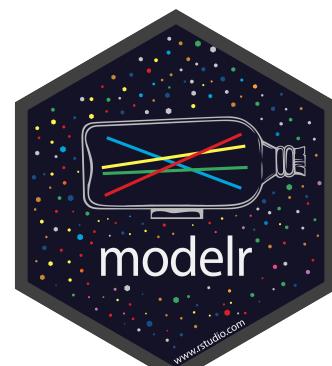
(Popular) modeling functions in R

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

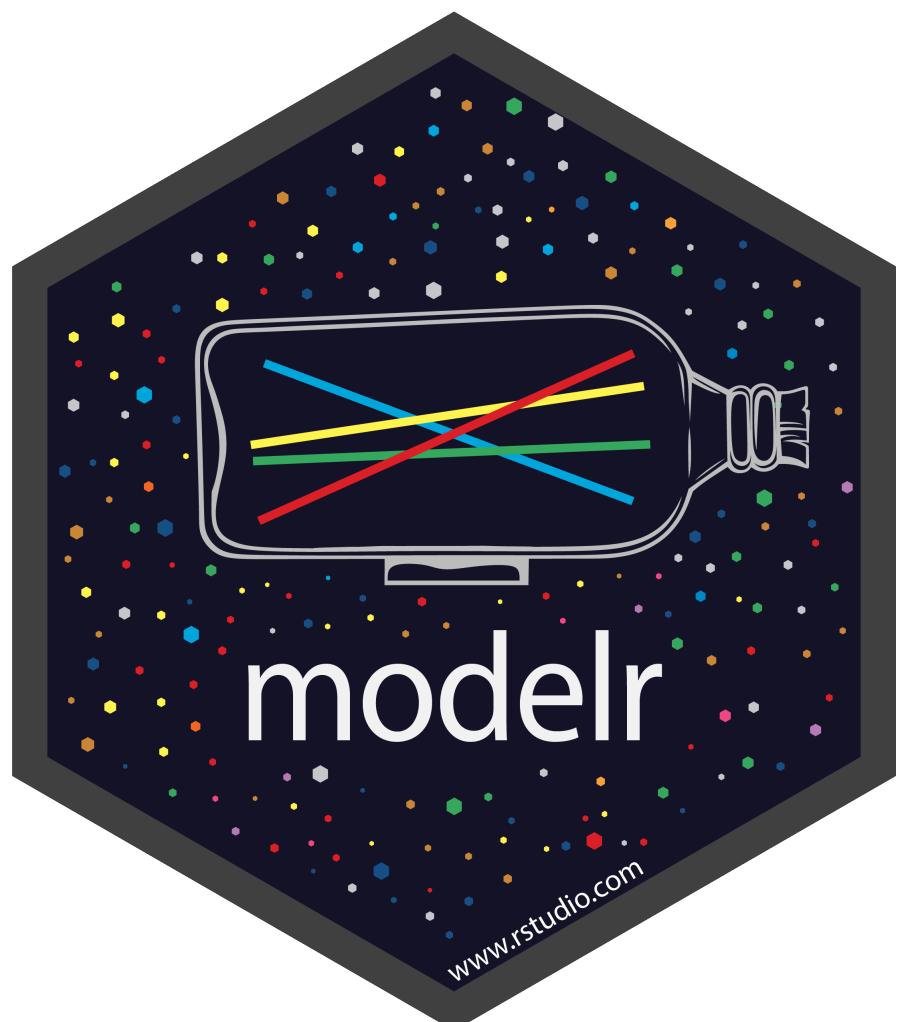


(Popular) modeling functions in R

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

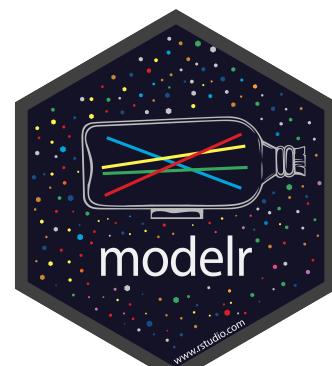


modelr



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

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

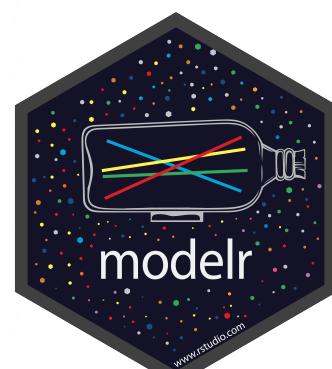


wages

income <int>	weight <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

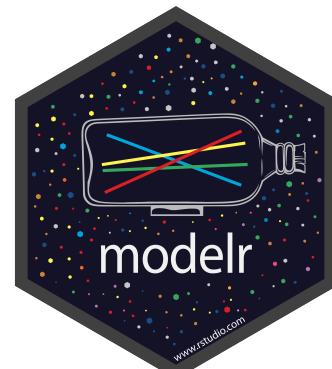
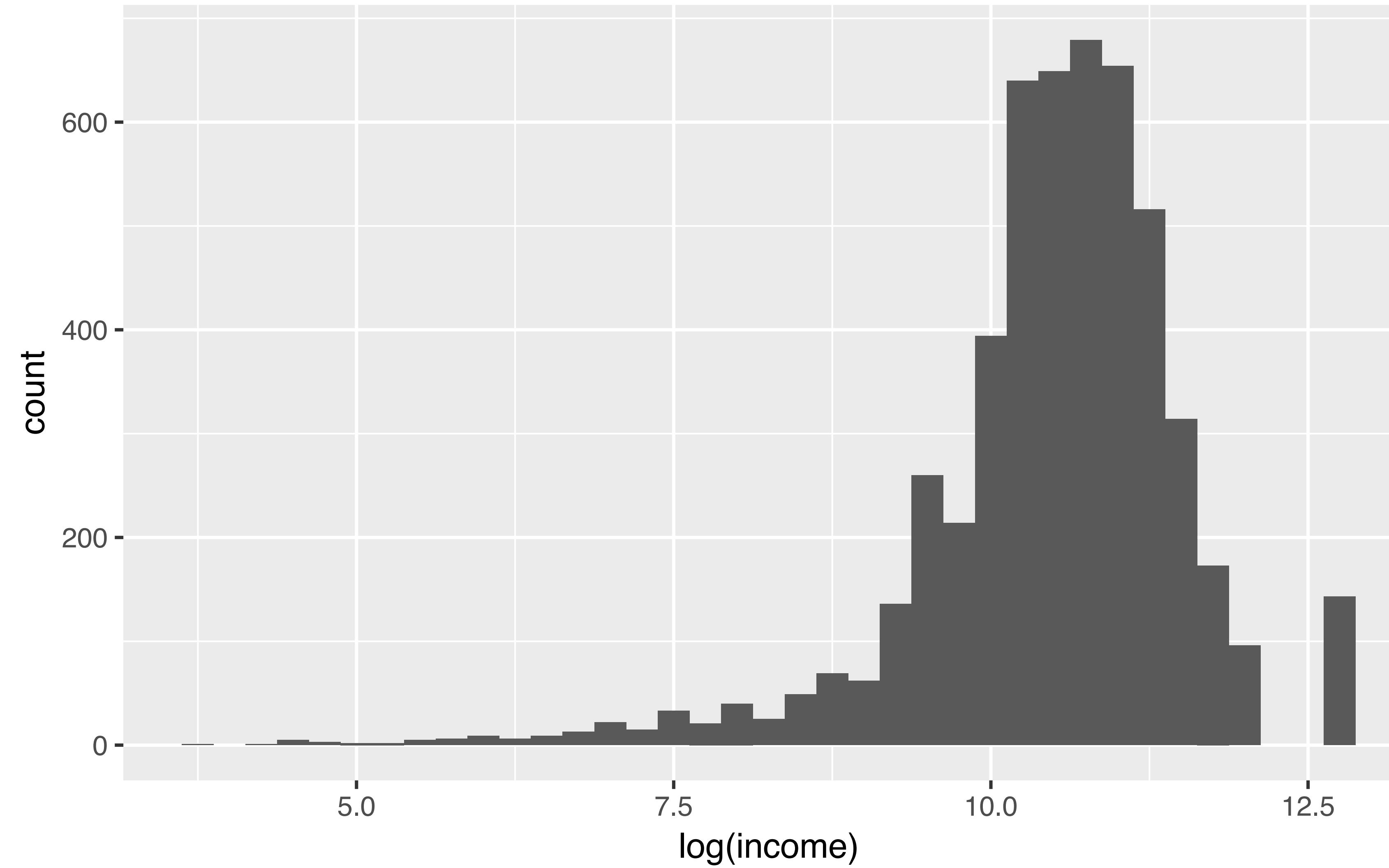
1-10 of 7,006 rows

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



wages %>%

```
ggplot(aes(log(income))) + geom_histogram(binwidth = 0.25)
```



lm()



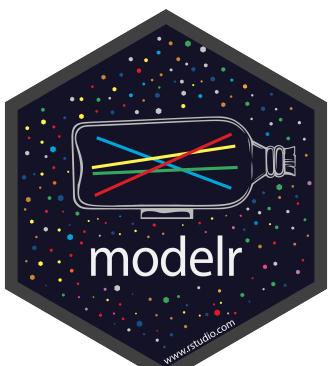
lm()

Fit a linear model to data

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

A formula that describes
the model equation

The data set



formulas

Formula only needs to include the response and predictors

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

$$\mathbf{y} \sim \mathbf{x}$$

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)
```



```
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 :(

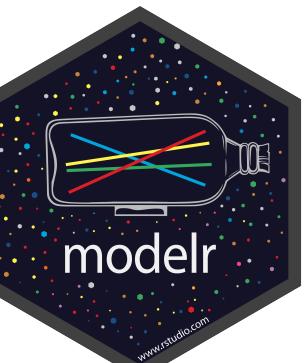
2. Output is not tidy, or even a data frame

•

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

2 rows



glance

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

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

r.squared	adj.r.squared	sigma	statistic	p.value
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
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	log.income.	education	.fitted	.se.fit	.resid	.hat	.sigma	.
<chr>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
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



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



```
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)  
## # A tibble: 2 × 6  
##   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)  
## # A tibble: 2 × 6  
##   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?

multiple regression

R

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

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



Your Turn 3

Model **log(income)** against **education and height**. Then use a broom function to determine:

Do the coefficients change?



```
mod_eh <- wages %>%  
  lm(log(income) ~ education + height, data = .)  
  
mod_eh %>%  
  tidy()  
## # A tibble: 3 × 6  
##   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 4

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



```
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

What does this mean?

Where is sexmale?



##	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:

$$\text{log(income)} = 8.25 + 0.15 * \text{education} + 0 * \text{height}$$

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:

$$\text{log(income)} = 8.25 + 0.15 * \text{education} + 0 * \text{height} - 0.46$$



##	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:

$$\text{log(income)} = 8.25 + 0.15 * \text{education} + 0 * \text{height}$$

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:

$$\text{log(income)} = 8.25 + 0.15 * \text{education} + 0 * \text{height} - 0.46$$



```
##          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

R

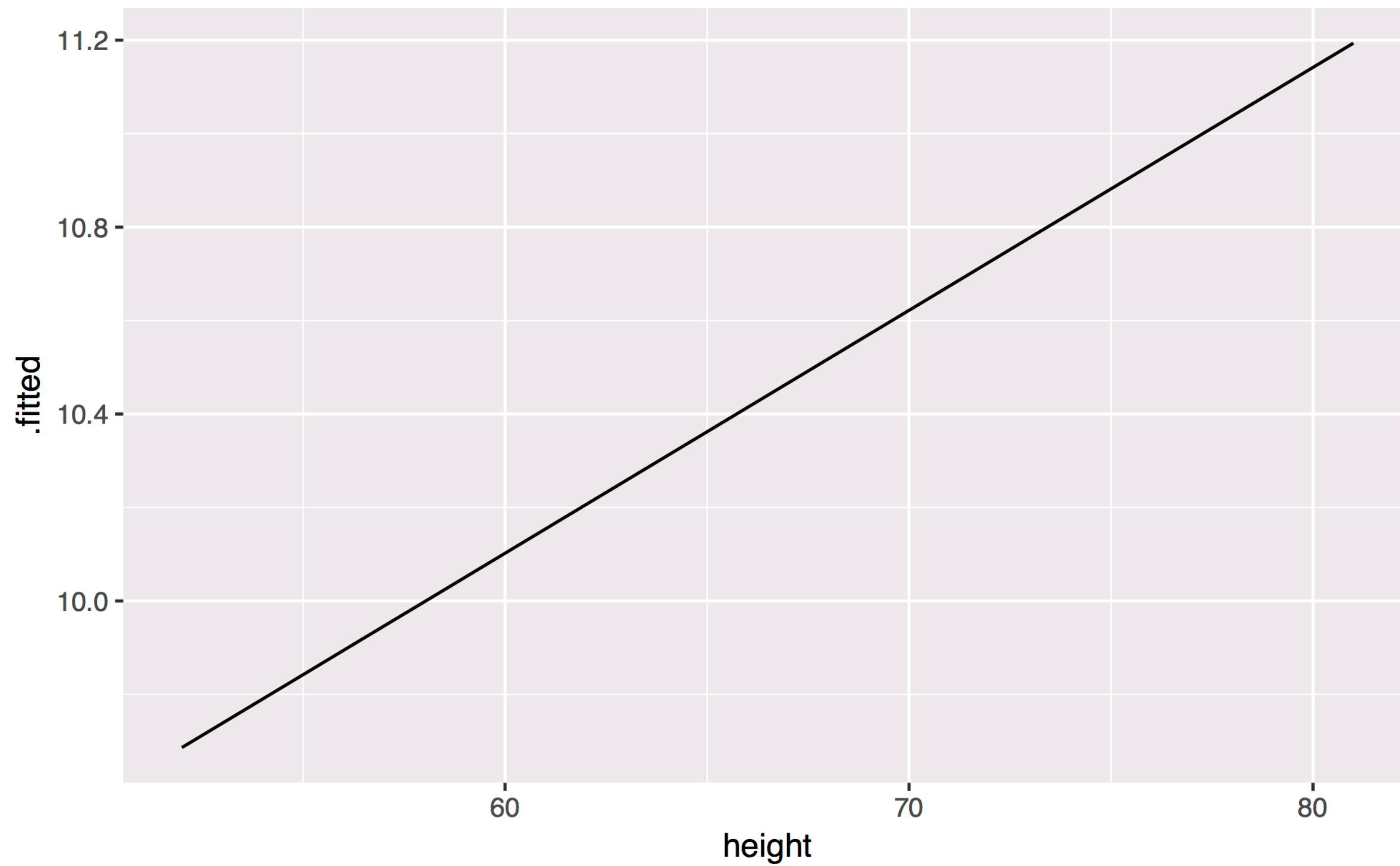
Your Turn 5

Use a broom function and ggplot2 to make a line graph of **height** vs **.fitted** for our heights model, **mod_h**.

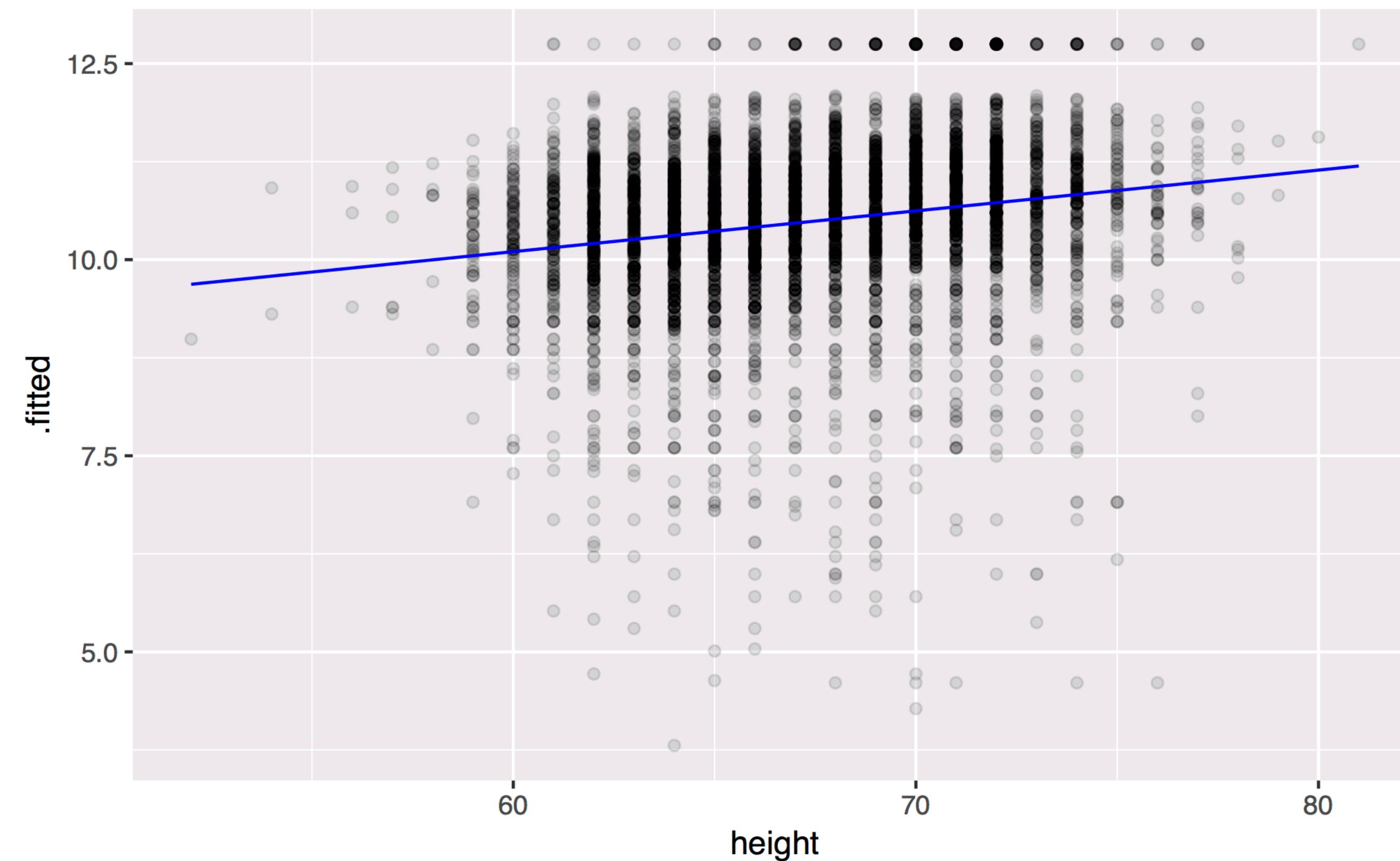
Bonus: Overlay the plot on the original data points.



```
mod_h %>%  
  augment(data = wages) %>%  
  ggplot(mapping = aes(x = height, y = .fitted)) +  
  geom_line()
```



```
mod_h %>%  
  augment(data = wages) %>%  
  ggplot(mapping = aes(x = height, y = .fitted)) +  
    geom_point(mapping = aes(y = log(income)), alpha = 0.1) +  
    geom_line(color = "blue")
```



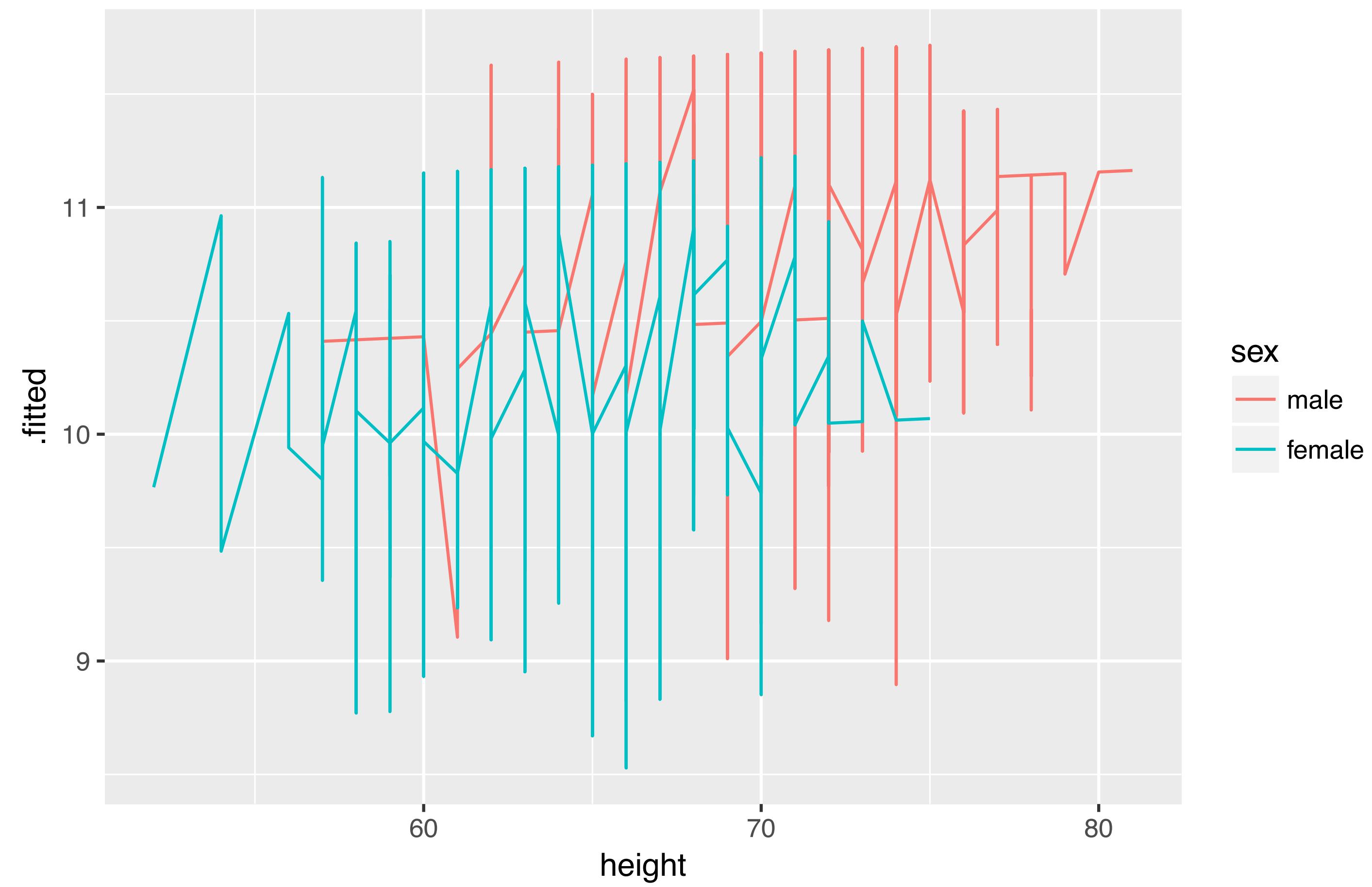
Your Turn 6

Repeat the process to make a line graph of **height** vs **.fitted** colored by **sex** for model **mod_ehs**. Are the results interpretable?

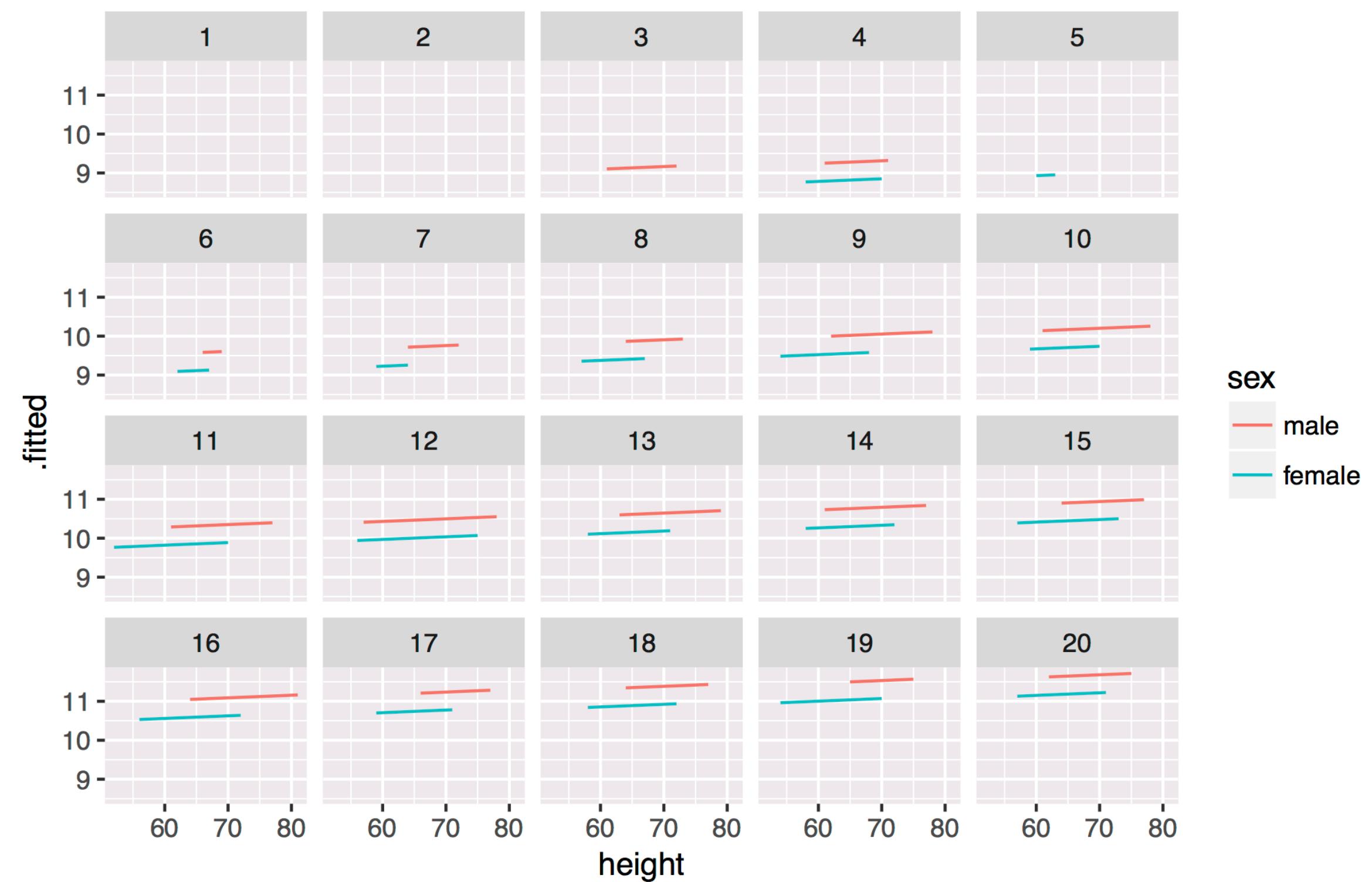
Add **+ facet_wrap(~education)** to the end of your code.
What happens?

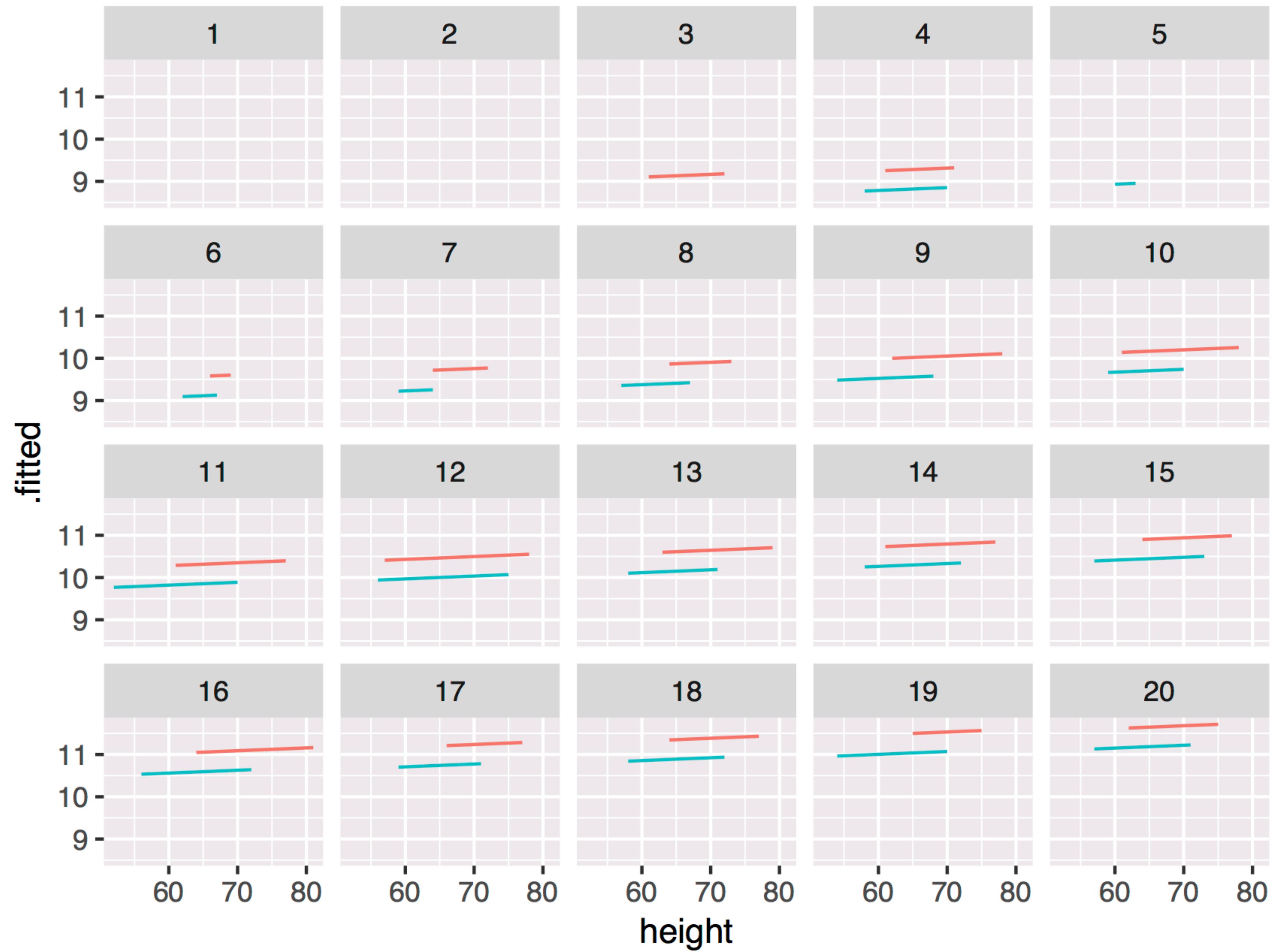


```
mod_ehs %>%  
  augment(data = wages) %>%  
  ggplot(mapping = aes(x = height, y = .fitted, color = sex)) +  
  geom_line()
```



```
mod_ehs %>%  
  augment(data = wages) %>%  
  ggplot(mapping = aes(x = height, y = .fitted, color = sex)) +  
  geom_line() +  
  facet_wrap(~ education)
```





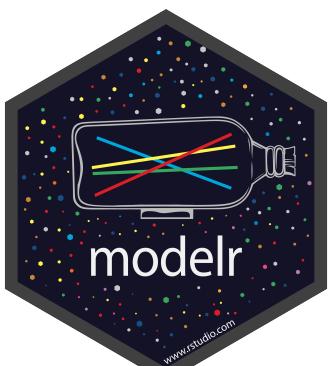
facet_wrap()

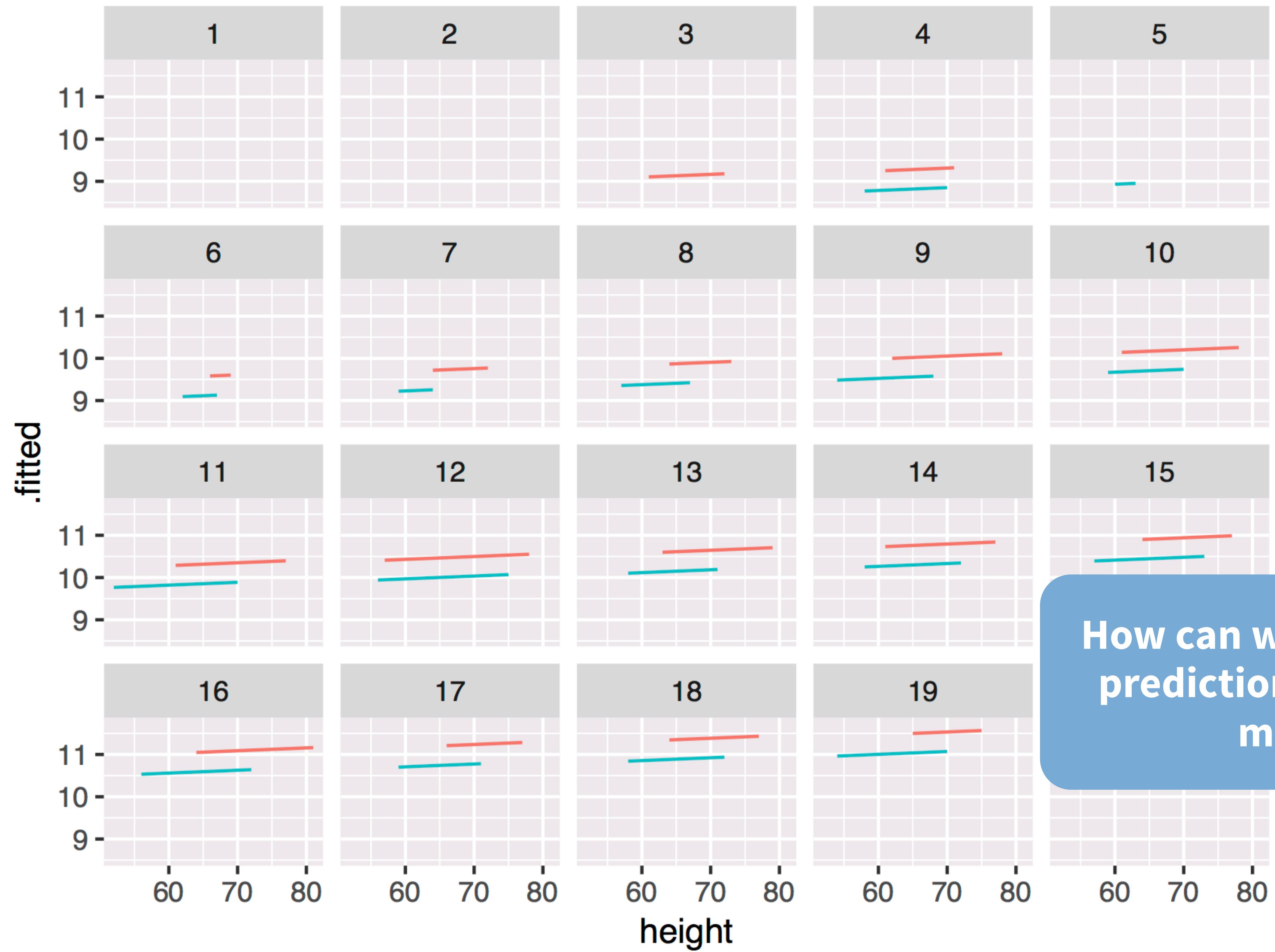
Divides plot into subplots based on a grouping variable.
"Wraps" subplots into rectangular collection.

```
+ facet_wrap(~ var)
```

Always a ~

Name of the grouping
variable. No quotes.





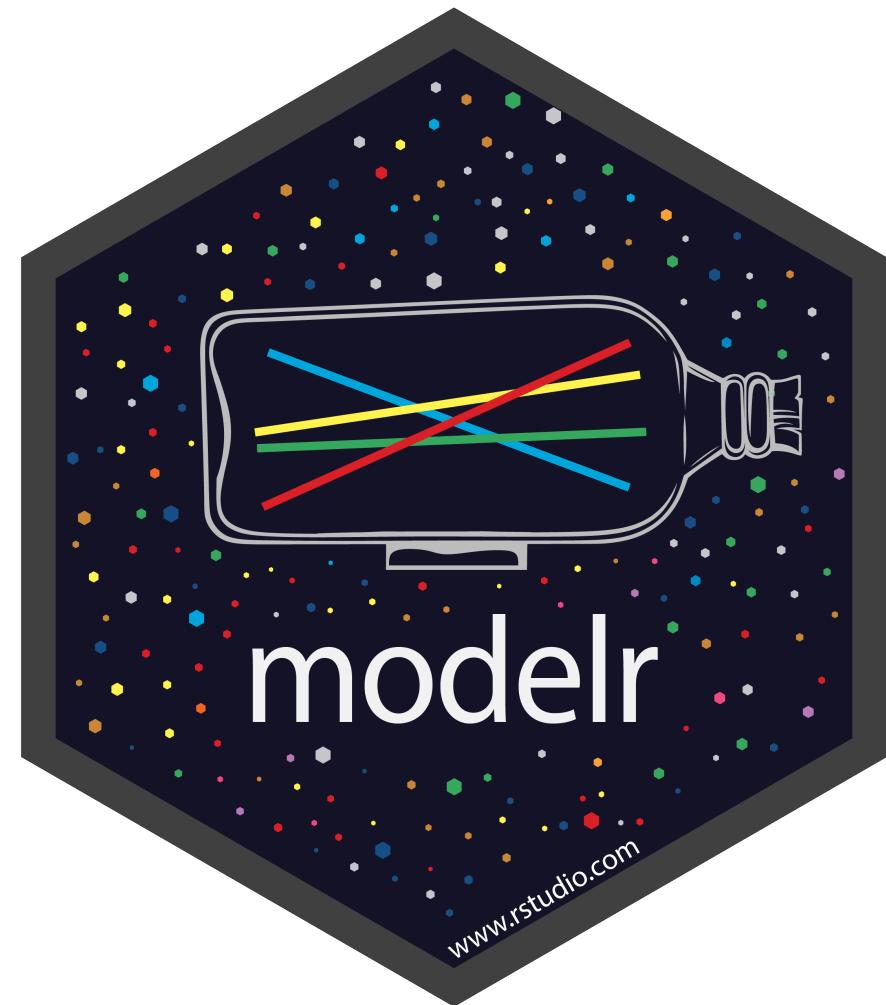
How can we compare the predictions of different models?



visualizing multiple models

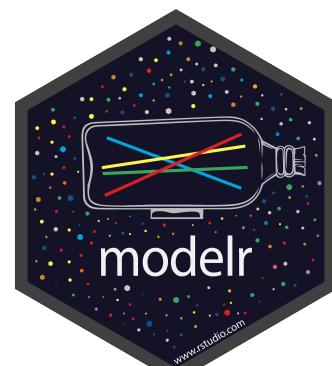
R

modelr



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

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



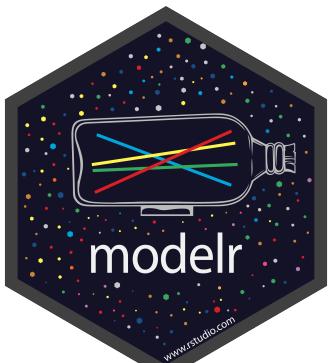
`add_predictions()`

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

```
add_predictions(data, model)
```

Uses this model

To add predictions
to these cases

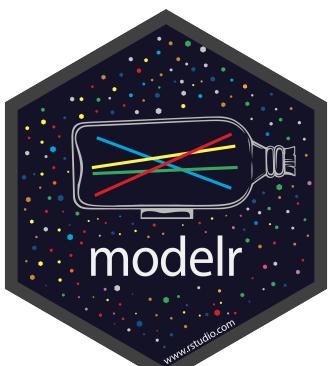


```
wages %>% add_predictions(mod_h)
```

	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



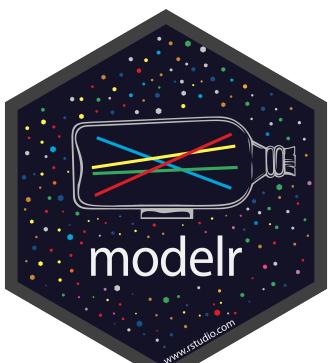
spread_predictions()

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

```
spread_predictions(data, ...)
```

Adds predictions from each of these models

To the cases in this data frame



```
wages %>%
```

```
  spread_predictions(mod_h, mod_eh, mod_ehs)
```

◀ marital <fctr>	sex <fctr>	education <int>	afqt <dbl>	mod_h <dbl>	mod_eh <dbl>	mod_ehs <dbl>
married	female	13	6.841	10.102158	10.050162	10.116052
married	female	10	49.444	10.621947	10.117110	9.739369
married	male	16	99.393	10.362053	10.707844	11.055381
married	female	14	44.022	10.258095	10.333801	10.284215
married	male	14	59.683	10.414032	10.478727	10.766142
divorced	female	18	98.798	10.517989	11.130195	10.909780
divorced	female	12	50.283	10.310074	10.104684	9.994975
divorced	male	12	89.669	10.569968	10.346227	10.490355
divorced	male	13	95.977	10.569968	10.484940	10.638338
married	female	16	67.021	10.310074	10.659535	10.586908

1-10 of 5,266 rows | 5-11 of 11 columns

Previous

1

2

3

4

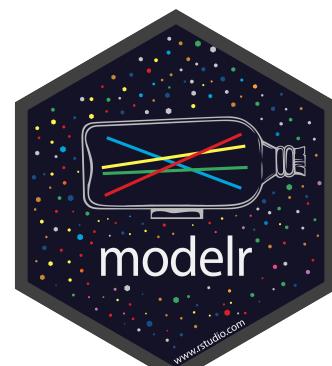
5

6

...

100

Next



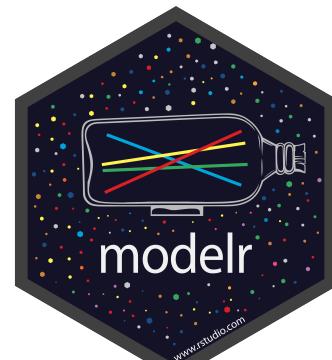
`gather_predictions()`

Adds predictions for multiple models as a pair of key:value columns (model:pred)

```
gather_predictions(data, ...)
```

Adds predictions from each of these models

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



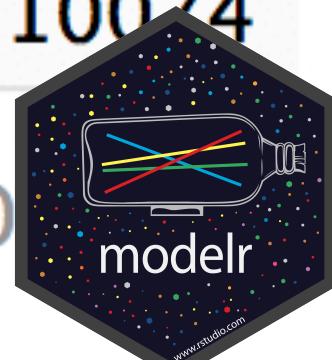
```
wages %>%
```

```
gather_predictions(mod_h, mod_eh, mod_ehs)
```

model <chr>	income <int>	height <dbl>	weight <int>	age <int>	marital <fctr>	sex <fctr>	education <int>	afqt <dbl>	pred <dbl>
mod_h	19000	60	155	53	married	female	13	6.841	10.102158
mod_h	35000	70	156	51	married	female	10	49.444	10.621947
mod_h	105000	65	195	52	married	male	16	99.393	10.362053
mod_h	40000	63	197	54	married	female	14	44.022	10.258095
mod_h	75000	66	190	49	married	male	14	59.683	10.414032
mod_h	102000	68	200	49	divorced	female	18	98.798	10.517989
mod_h	70000	64	160	54	divorced	female	12	50.283	10.310074
mod_h	60000	69	162	55	divorced	male	12	89.669	10.569968
mod_h	150000	69	194	54	divorced	male	13	95.977	10.569968
mod_h	115000	64	145	53	married	female	16	67.021	10.310074

1-10 of 15,798 rows

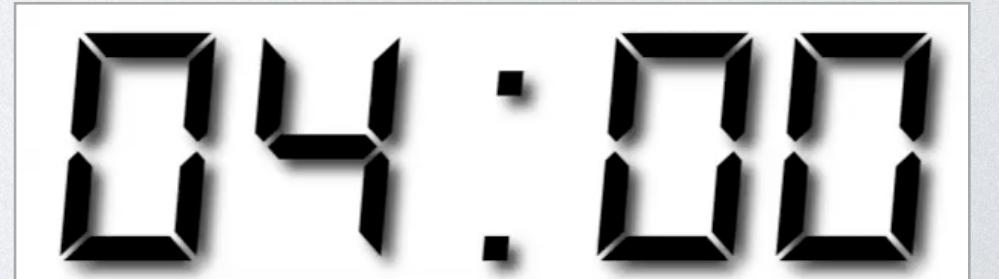
Previous 1 2 3 4 5 6 ... 100



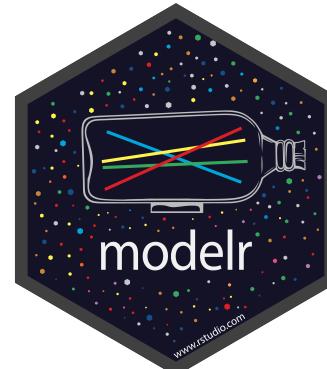
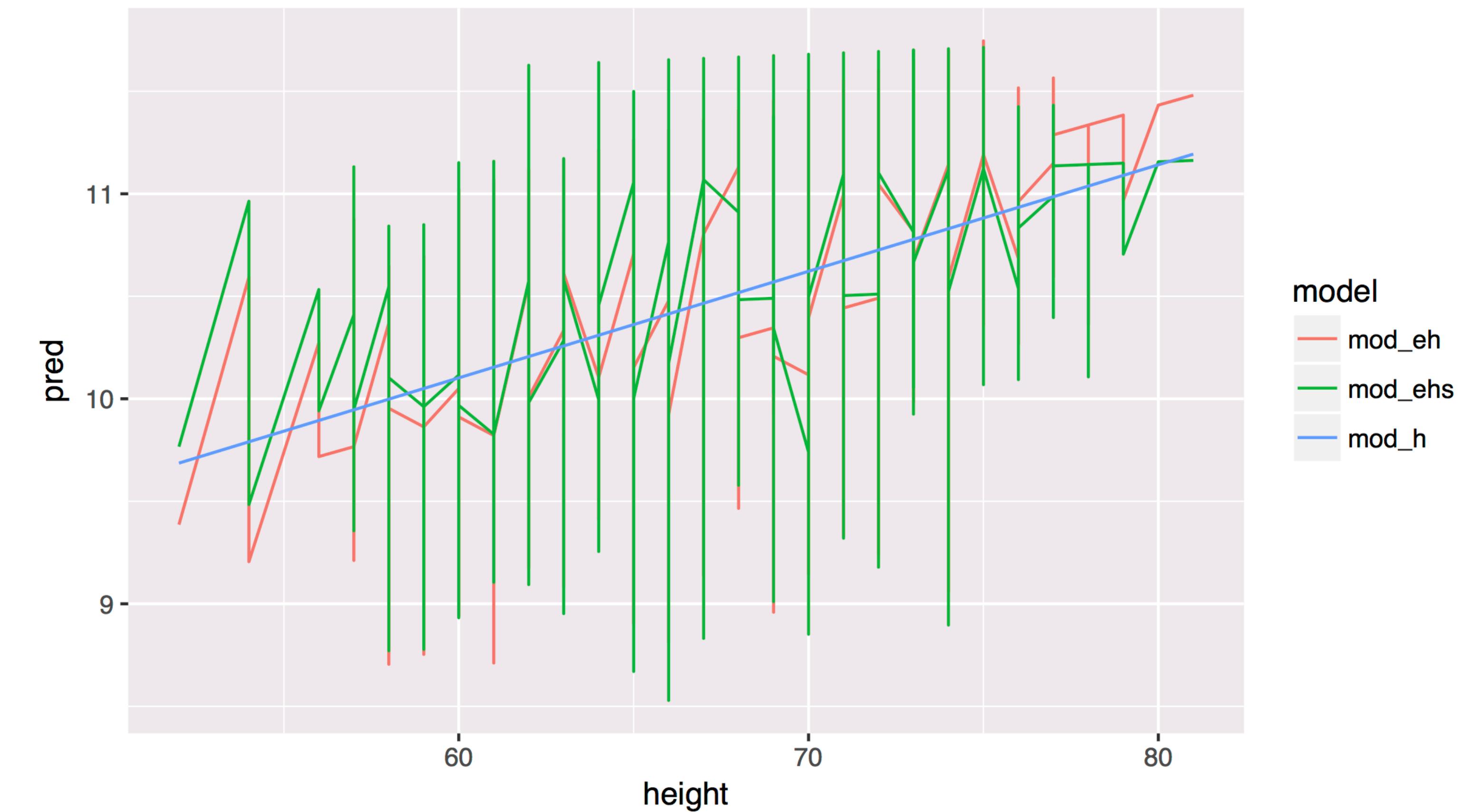
Your Turn 7

Use one of `spread_predictions()` or `gather_predictions()` to make a line graph of `height` vs `pred` colored by `model` for each of `mod_h`, `mod_eh`, and `mod_ehs`. Are the results interpretable?

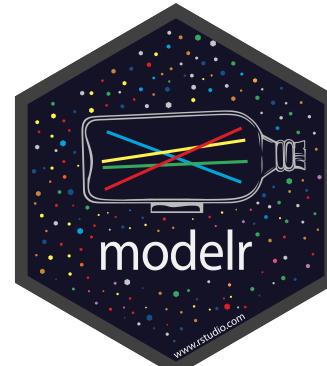
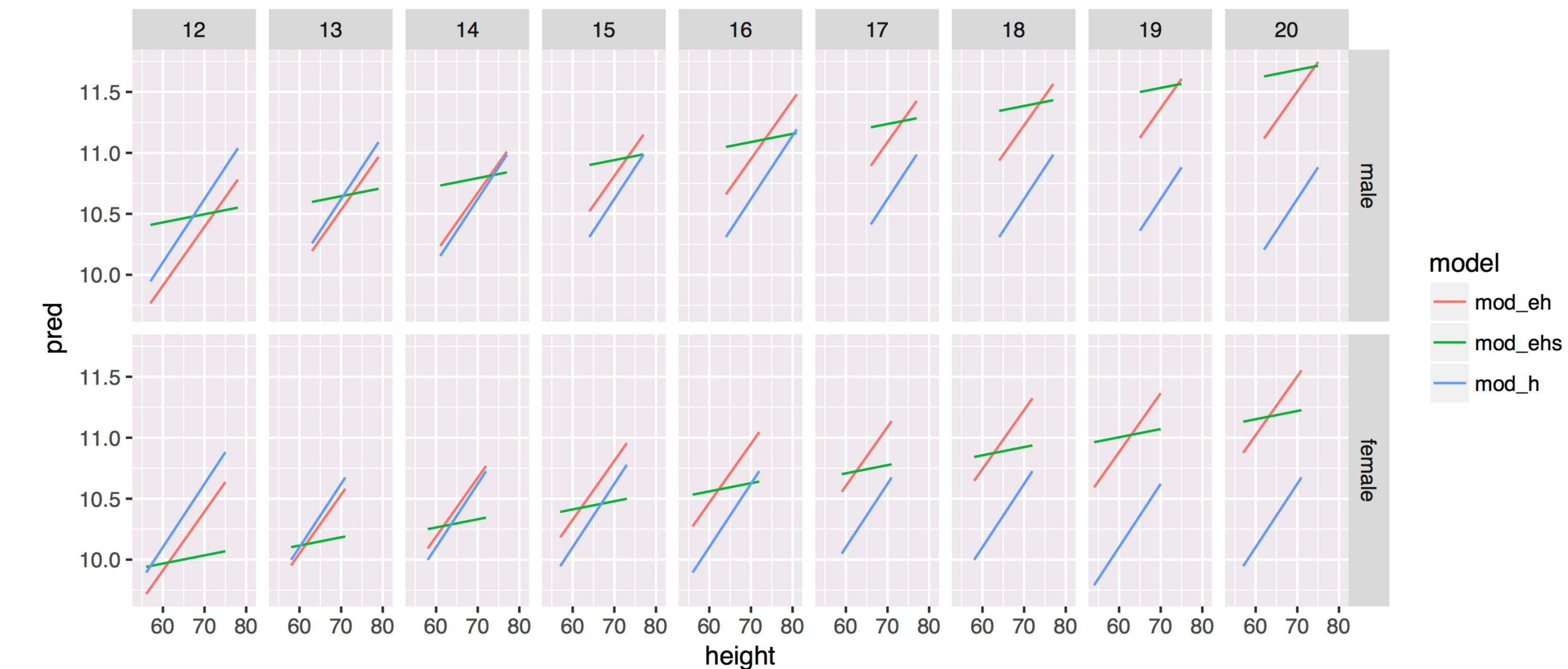
Add `+ facet_grid(sex ~ education)` to the end of your code. What happens?

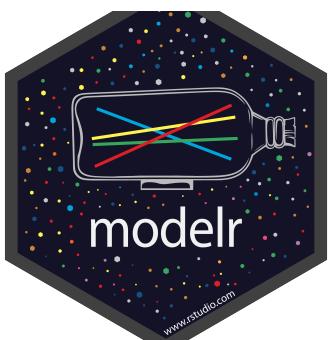
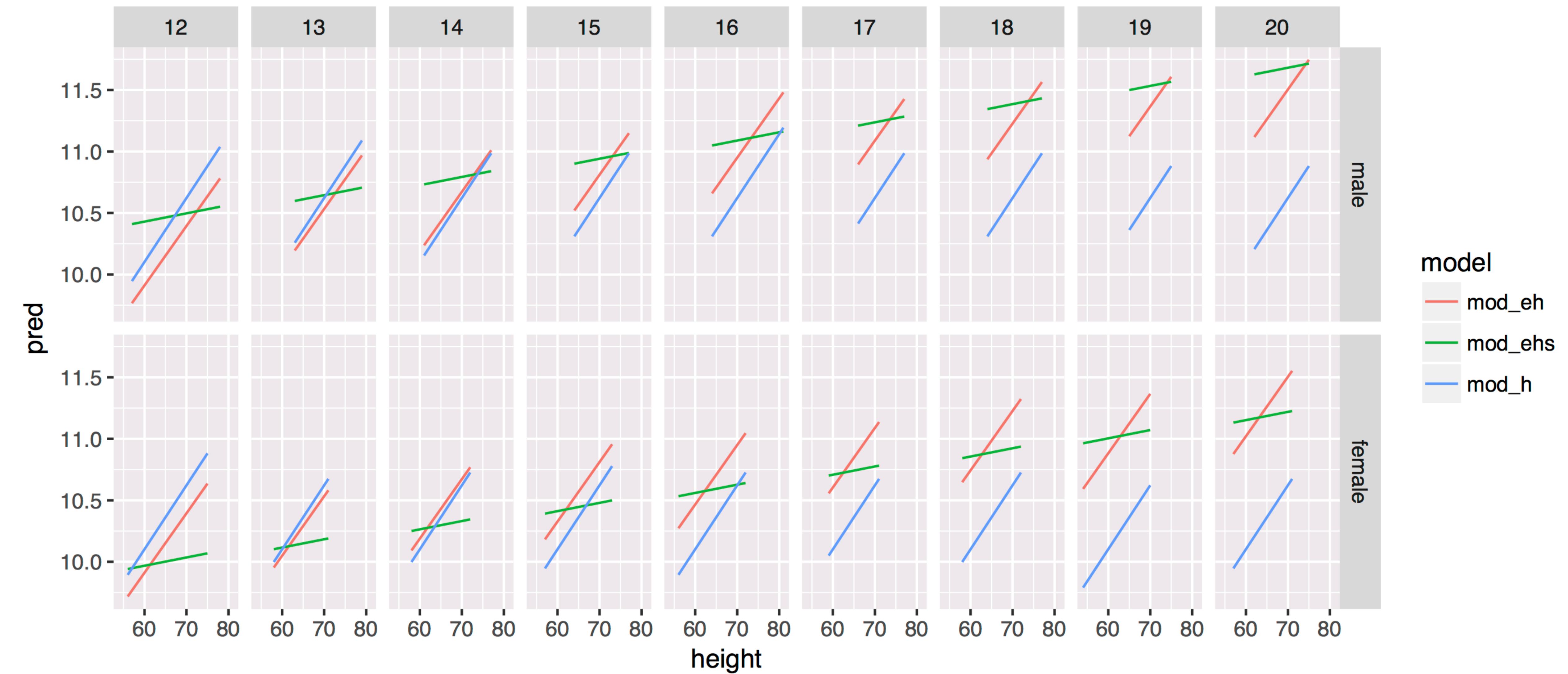


```
wages %>%  
gather_predictions(mod_h, mod_eh, mod_ehs) %>%  
ggplot(mapping = aes(x = height, y = pred, color = model)) +  
geom_line()
```

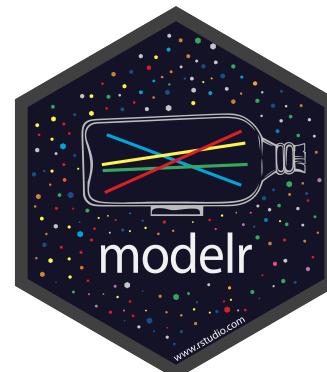
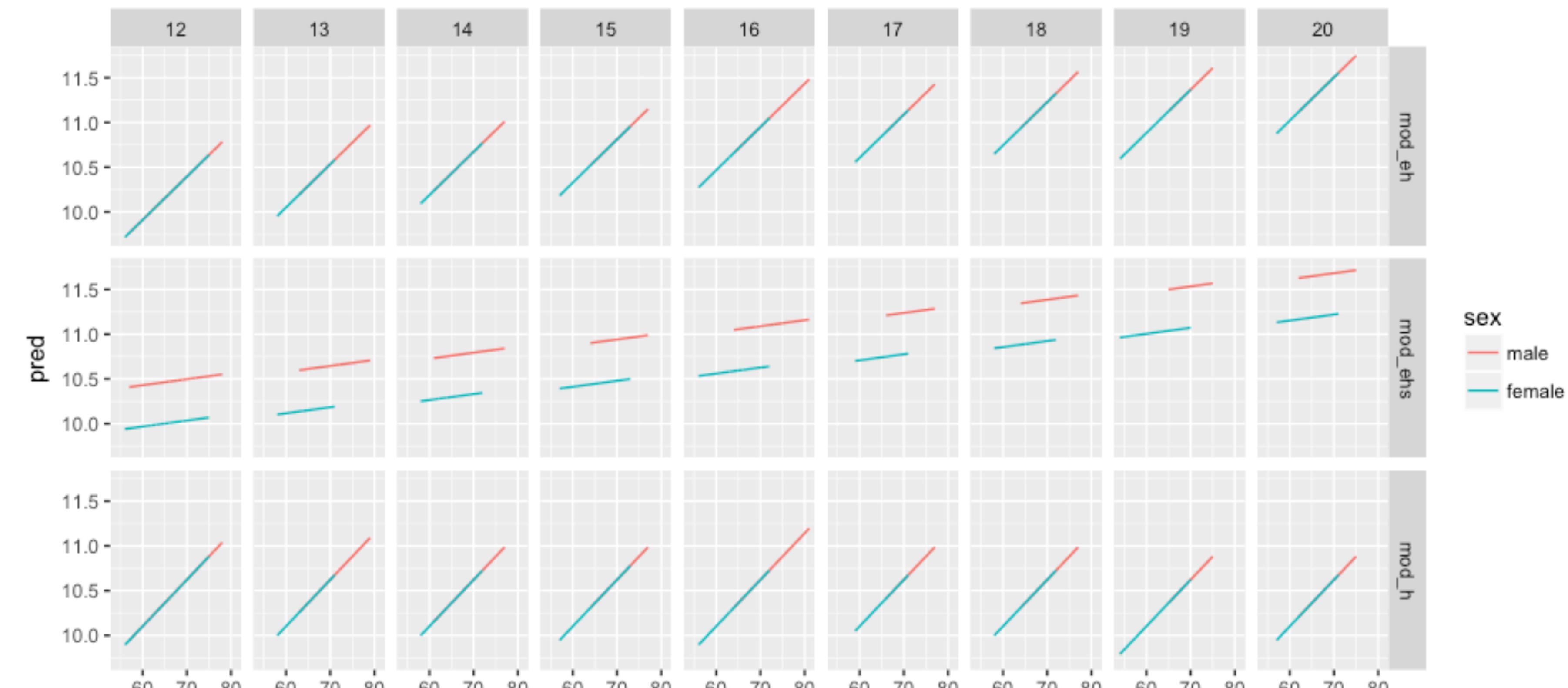


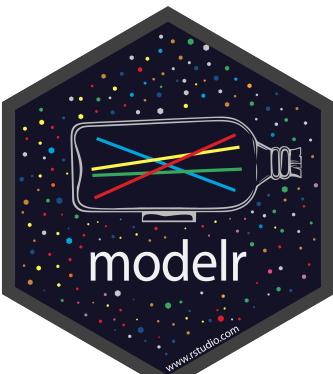
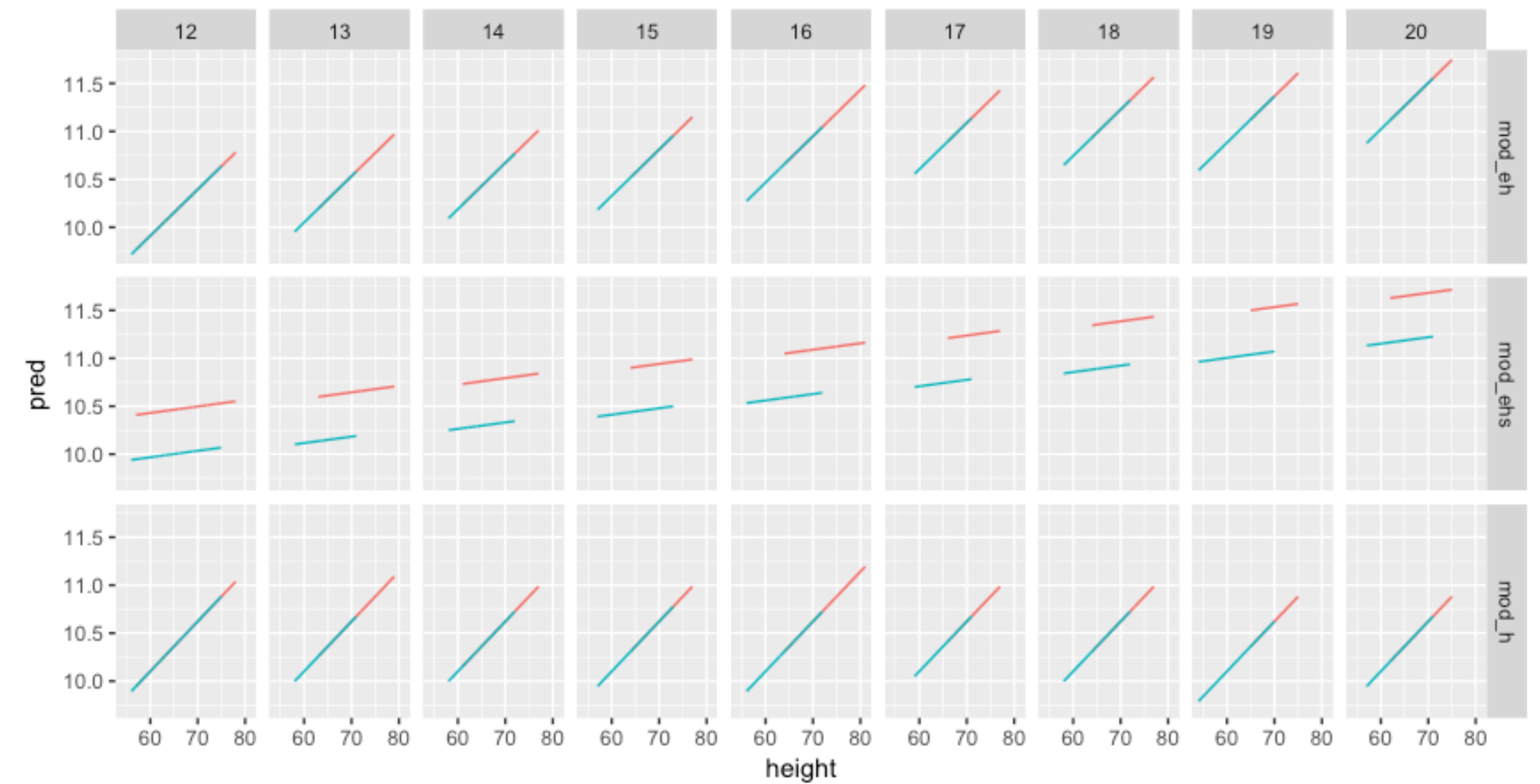
```
wages %>%  
gather_predictions(mod_h, mod_eh, mod_ehs) %>%  
ggplot(mapping = aes(x = height, y = pred, color = model)) +  
geom_line() +  
facet_grid(sex ~ education)
```





```
wages %>%  
gather_predictions(mod_h, mod_eh, mod_ehs) %>%  
filter(education > 11) %>%  
ggplot(mapping = aes(x = height, y = pred, color = sex)) +  
geom_line() +  
facet_grid(model ~ education)
```





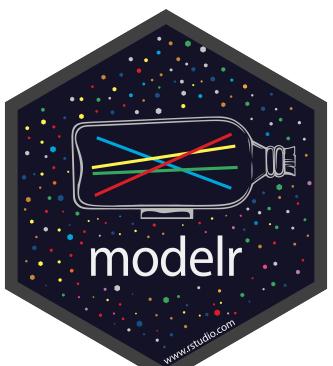
Residuals

Modelr provides the equivalent functions for residuals

`add_predictions()` → `add_residuals()`

`spread_predictions()` → `spread_residuals()`

`gather_predictions()` → `gather_residuals()`



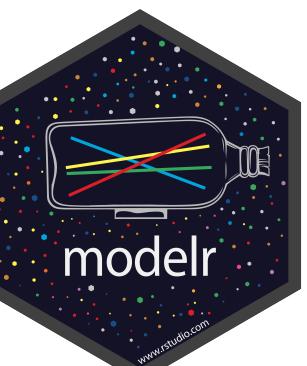
```
wages %>%  
  add_residuals(mod_e)
```

Modelr provides the equivalent functions for residuals

add_predictions() → add_residuals()

spread_predictions() → spread_residuals()

gather_predictions() → gather_residuals()



```
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

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

