Modeling with

broom modelr

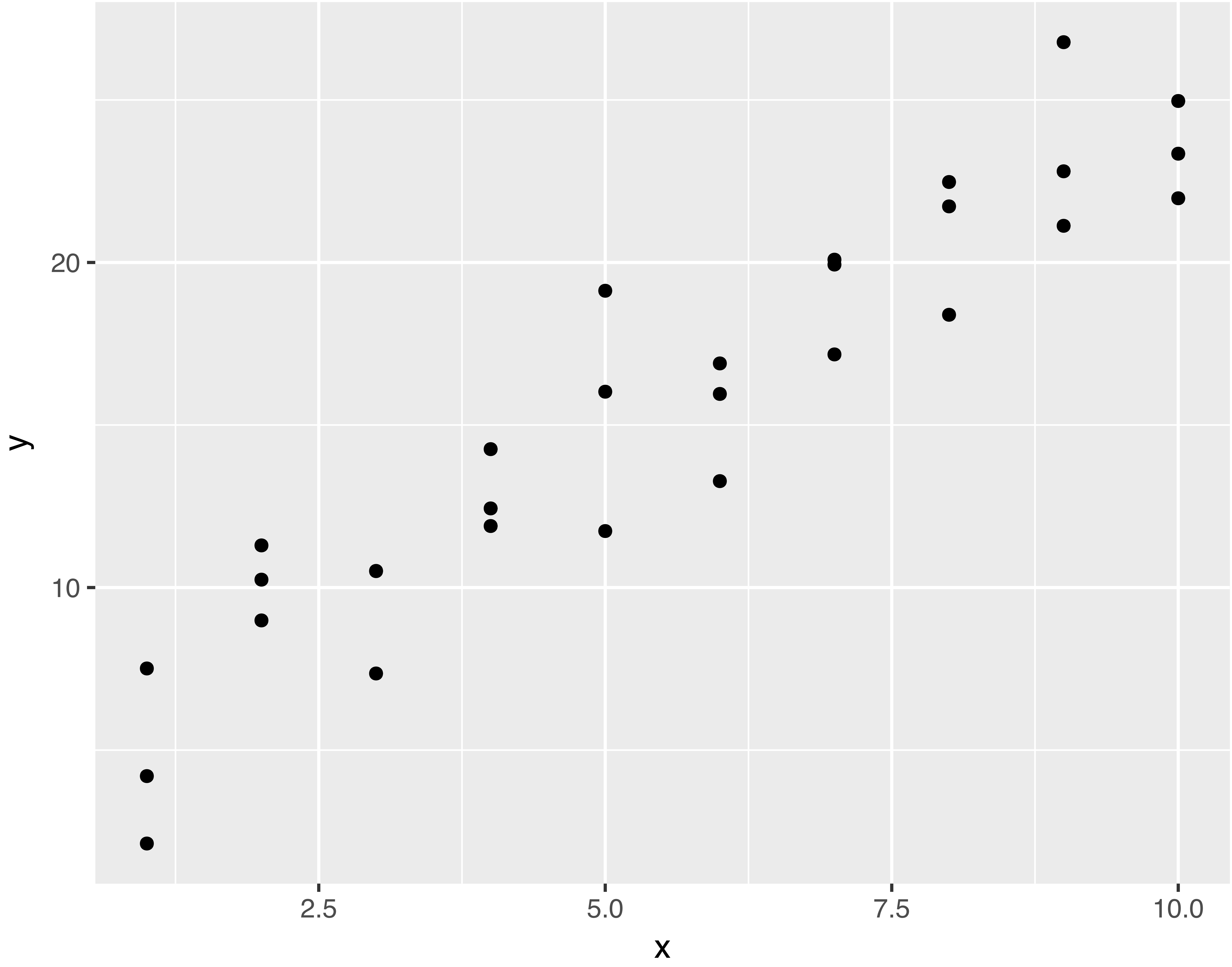
The basics

Models

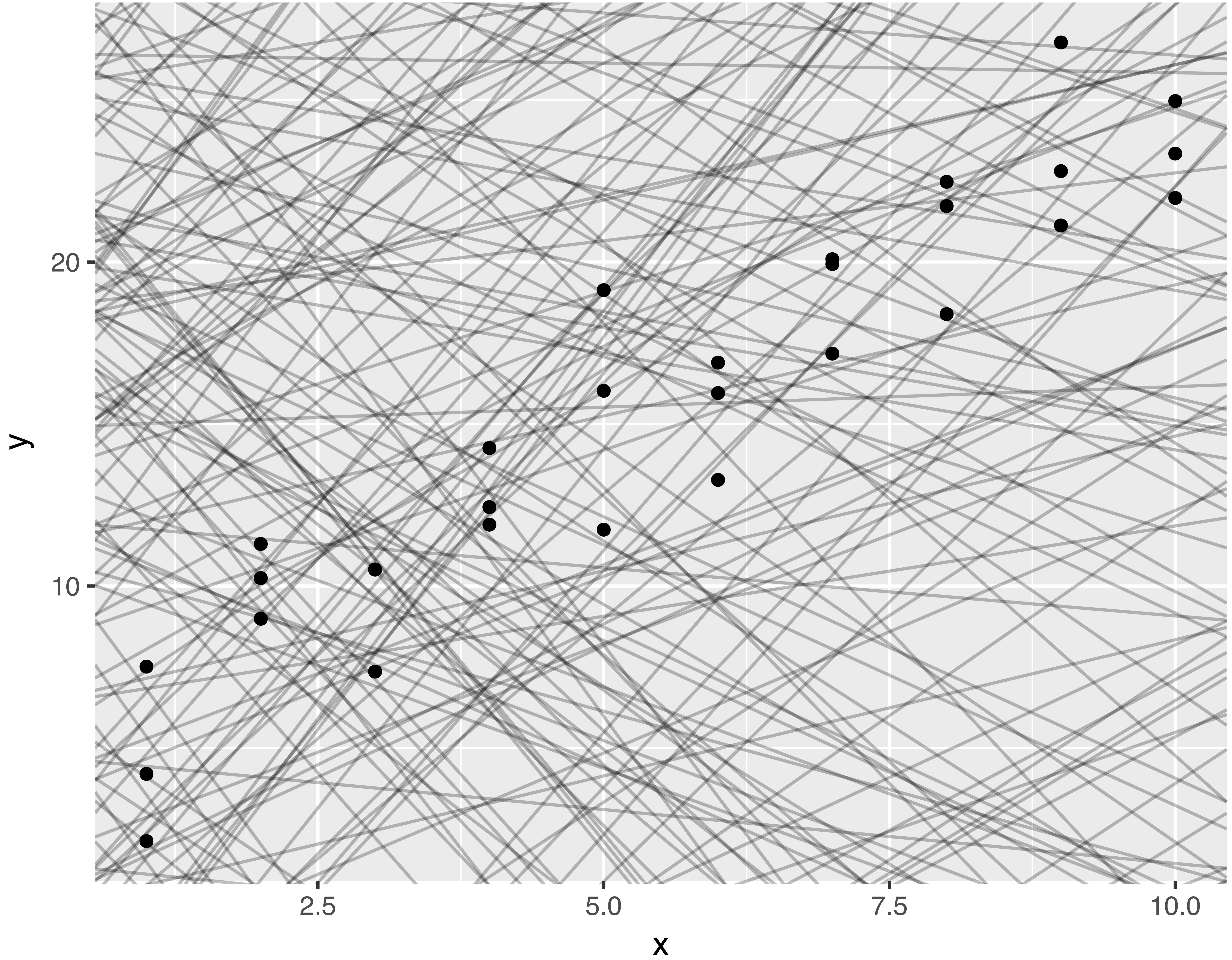
A low dimensional description of a higher dimensional data set.
Consists of three parts:

1. A family of functions
2. The function in the family that best approximates the data
3. Residuals

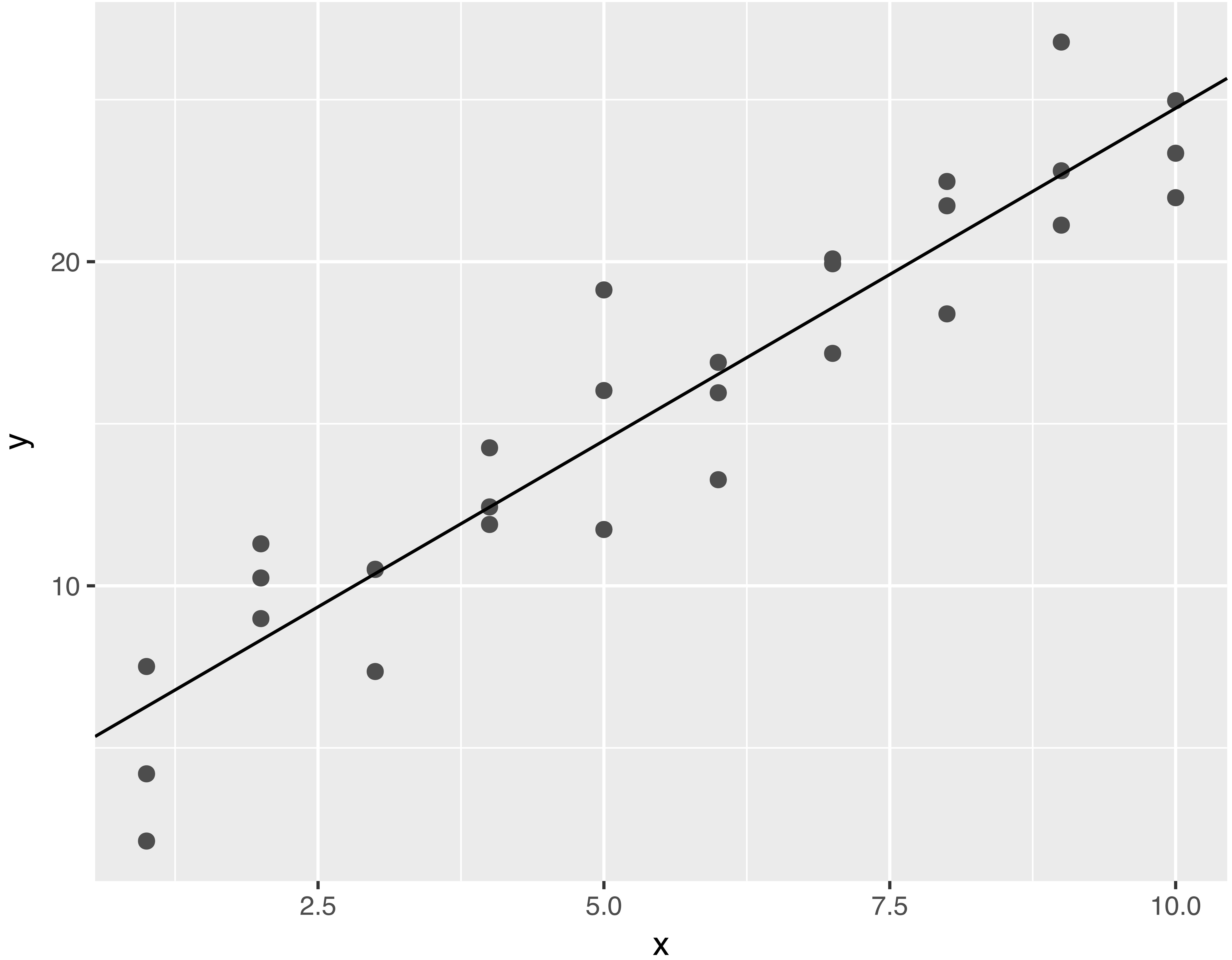
Example



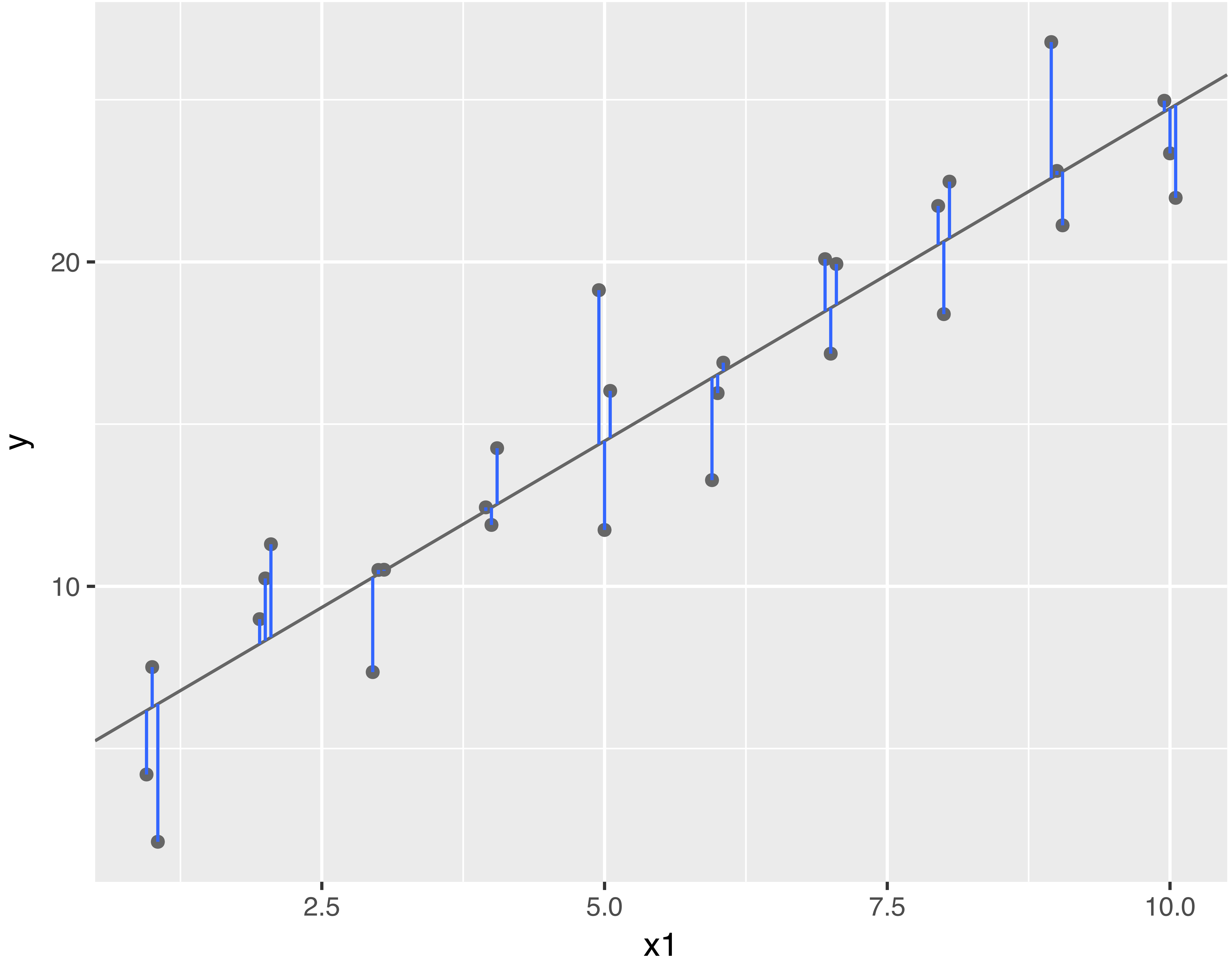
1. A family of functions



2. The best function of the family



3. The residuals



(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

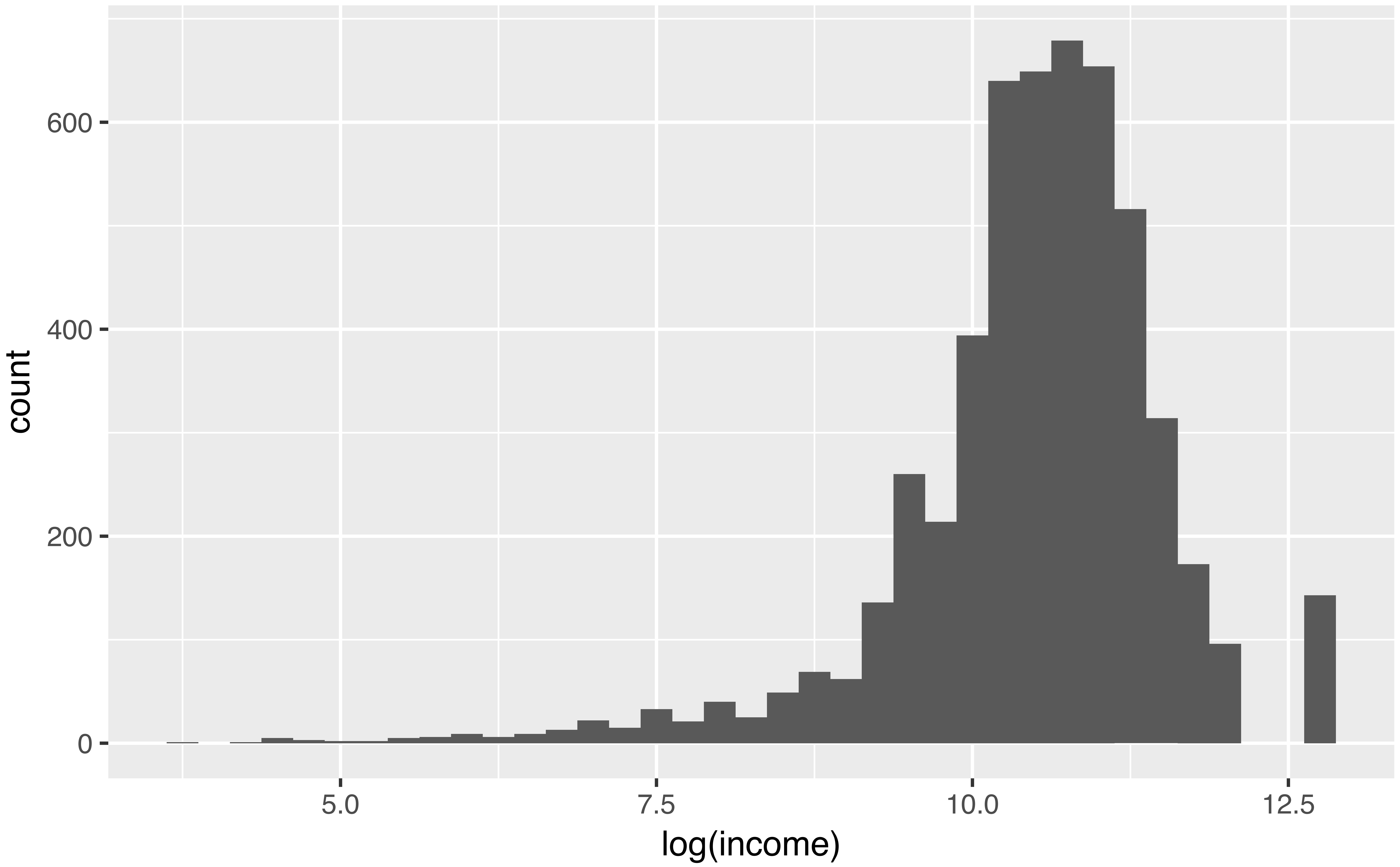
```
# install.packages("tidyverse")
library(modelr)
wages <- heights %>% filter(income > 0)
```

```
wages
```

income <int>	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

```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

formulas

Formula only needs to include the response and predictors

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

```
y ~ x
```


modelr

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	estimate	std.error	statistic	p.value
<chr>	<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

```r
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

broom

augment()

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

```r
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

```r
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
```

Your Turn 3

Model **log(income)** against **education** *and* **height**. Do the coefficients change?

```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 4

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

03:00

```r
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:

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

```
##            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

```
##            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

modelr

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

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

Visualize predictions

To visualize model predictions:

1. Make a range of x (and y) values to visualize with **data_grid()**
2. Add predictions with **add_predictions()**
3. **Plot**

data_grid()

Creates a data frame with useful combinations of values.

```
data_grid(data, var)
```

Generates range of evenly spaced values for this variable

...from this data set

data_grid()

Creates a data frame with useful combinations of values.

```
data_grid(data, var1, var2)
```

Generates every combination of values in the ranges of these variables

...from this data set

Your Turn 5

Test your understanding. Try to use **data_grid()** to generate this data frame of values.

height <dbl>	education <int>	sex <fctr>
52	1	male
52	1	female
52	2	male
52	2	female
52	3	male
52	3	female
52	4	male
52	4	female
52	5	male
52	5	female

02:00

```
wages %>% data_grid(height, education, sex)
```

height <dbl>	education <int>	sex <fctr>
52	1	male
52	1	female
52	2	male
52	2	female
52	3	male
52	3	female
52	4	male
52	4	female
52	5	male
52	5	female

add_predictions()

Uses the values in a data frame to generate a prediction for each case.

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

Uses this model

To add predictions to these cases

modelr

Your Turn 6

Use **add_predictions()** to add predictions to your results from above.

height <dbl>	education <int>	sex <fctr>
52	1	male
52	1	female
52	2	male
52	2	female
52	3	male
52	3	female
52	4	male
52	4	female
52	5	male
52	5	female

02:00

```
wages %>%
  data_grid(height, education, sex) %>%
  add_predictions(mod_ehs)
```

height <dbl>	education <int>	sex <fctr>	pred <dbl>
52	1	male	8.748189
52	1	female	8.286442
52	2	male	8.896172
52	2	female	8.434425
52	3	male	9.044155
52	3	female	8.582408
52	4	male	9.192138

But...
it is difficult to visualize four variables

height	education	sex	pred
<dbl>	<int>	<fctr>	<dbl>
52	1	male	8.748189
52	1	female	8.286442
52	2	male	8.896172
52	2	female	8.434425
52	3	male	9.044155
52	3	female	8.582408
52	4	male	9.192138

data_grid()

Creates a data frame with useful combinations of values.

```
data_grid(wages, height, sex, .model = model_ehs)
```

Generates every combination of values in the ranges of these variables

...from this data set

and repeats a typical value for every other variable in this model

modelr

```
wages %>% data_grid(height, sex, .model = mod_ehs)
```

height	sex	education
<dbl>	<fctr>	<dbl>
52	male	13
52	female	13
54	male	13
54	female	13
56	male	13
56	female	13
57	male	13
57	female	13
58	male	13
58	female	13

1-10 of 56 rows   Previous 1 2 3 ... 6 Next

median of education

modelr

Your Turn 7

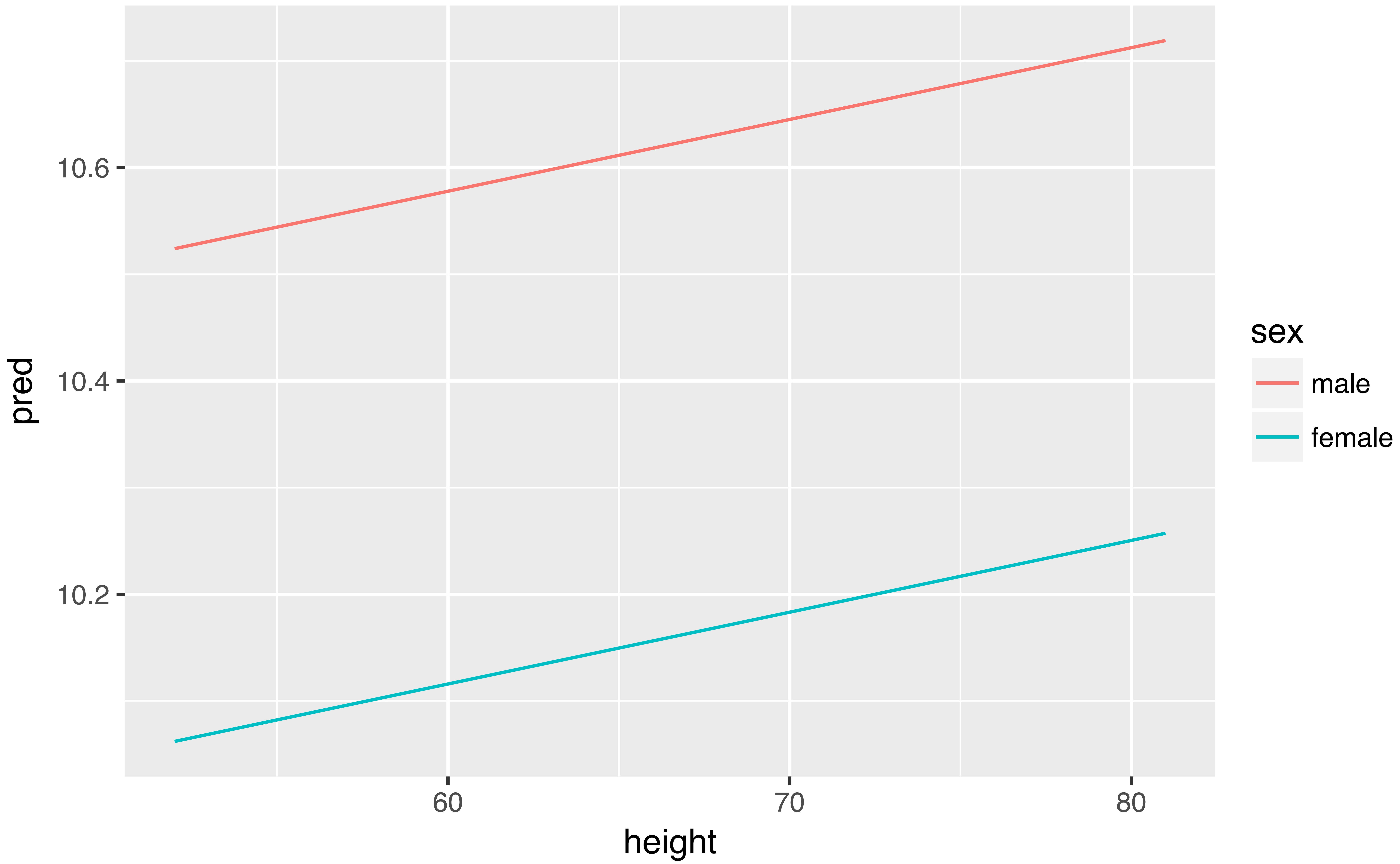
Adjust your code to standardise on the median of education. Then plot a line graph of height vs. predictions, colored by sex.

Bonus: overlay the results on the original data points.

04:00

3. Plot

```
wages %>%
  data_grid(height, sex, .model = mod_ehs) %>%
  add_predictions(mod_ehs) %>%
  ggplot() +
    geom_line(aes(x = heigh, y = pred, color = sex))
```



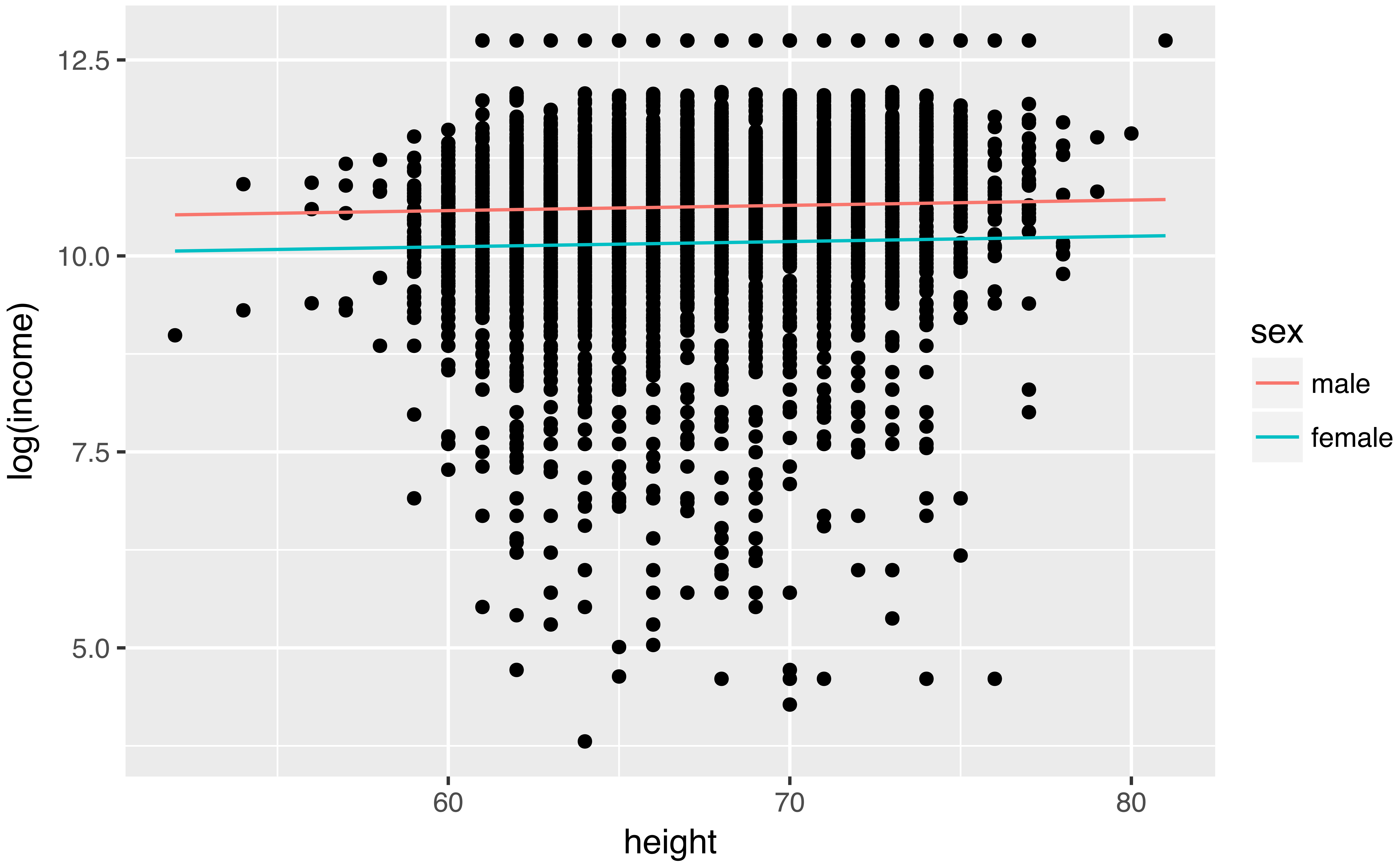
pred
10.6
10.4
10.2
60
70
80
height
sex
male
female

3. Plot

```
heights2 %>%
  data_grid(height, sex, .model = mod_ehs) %>%
  add_predictions(mod_ehs) %>%
  ggplot(aes(x = height)) +
    geom_point(aes(y = log(income)), data = wages) +
    geom_line(aes(y = pred, color = sex))
```

Adds the original data points

modelr



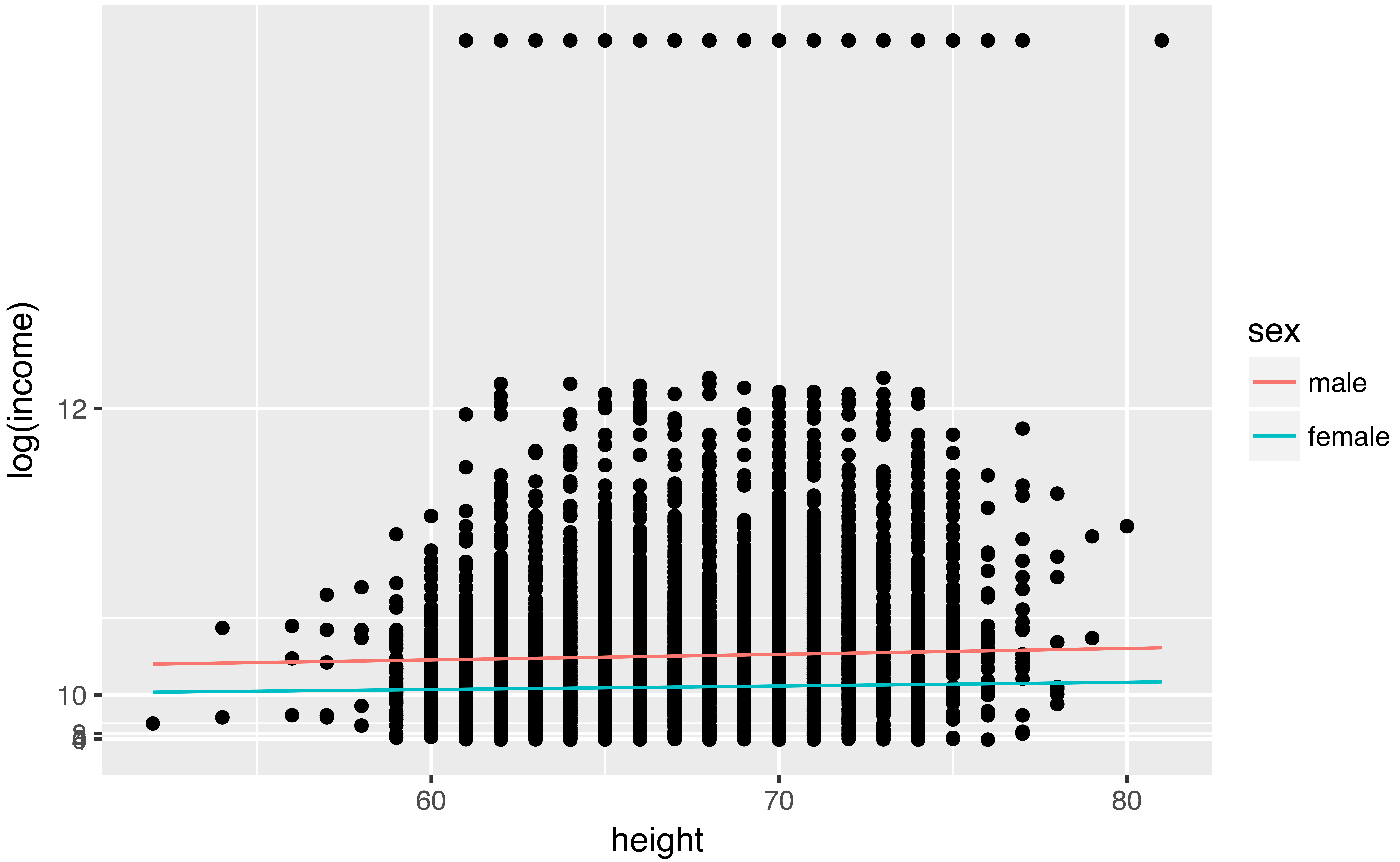
12.5
10.0
7.5
5.0
log(income)
60
70
80
height
sex
male
female

3. Plot

```
heights2 %>%
  data_grid(height, sex, .model = mod_ehs) %>%
  add_predictions(mod_ehs) %>%
  ggplot(aes(x = height)) +
    geom_point(aes(y = log(income)), data = wages) +
    geom_line(aes(y = pred, color = sex)) +
    coord_trans(y = "exp")
```

Visually back-transforms the log

modelr


log(income)
12
10
60
70
80
height
sex
male
female

Your Turn 8

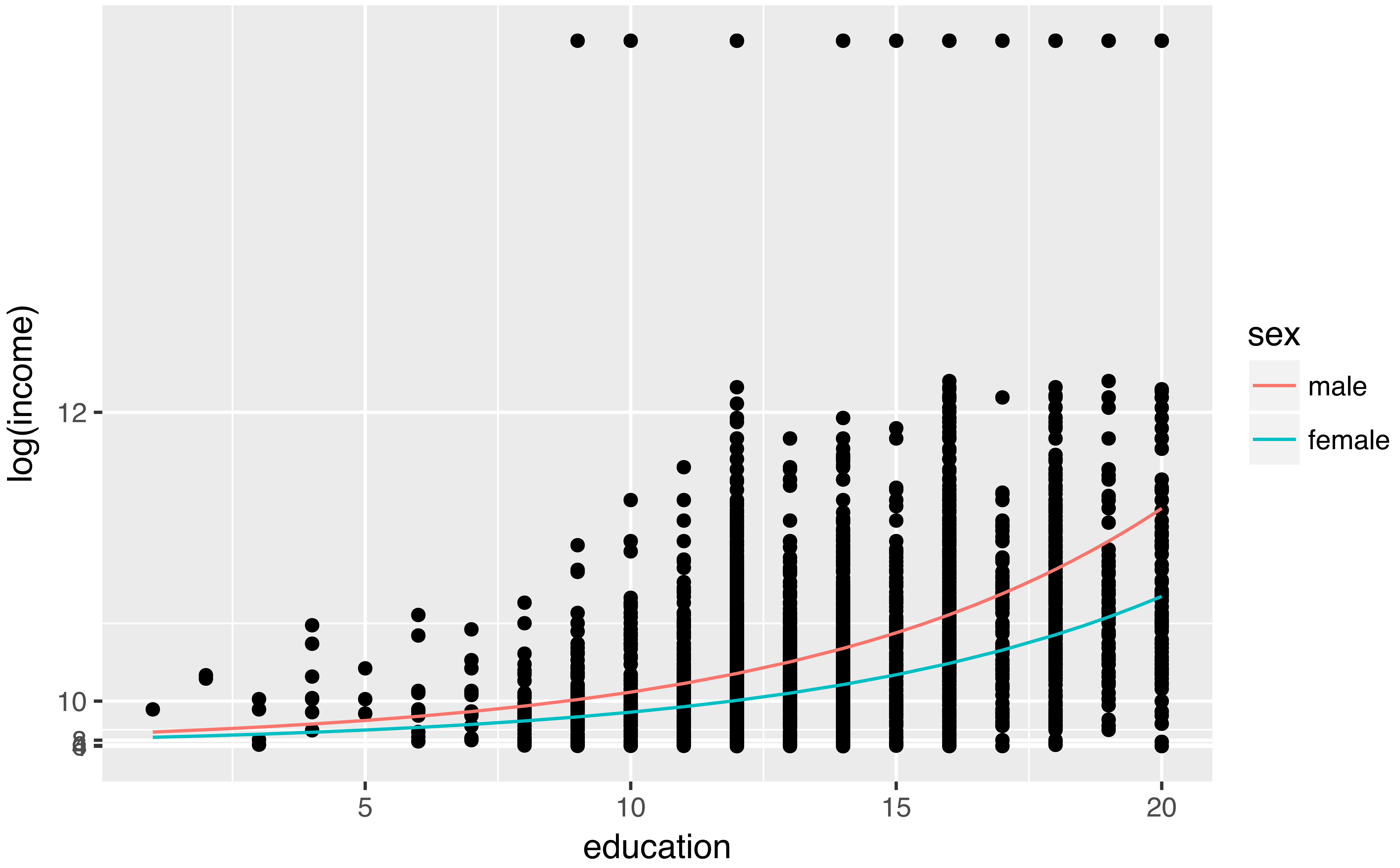
Plot the predictions of model_ehs against **education** and **sex** for a reasonable value of **height**:

1. Make a range of x values to visualize over

2. Add predictions

3. Plot

```r
wages %>%
  data_grid(education, sex, .model = mod_ehs) %>%
  add_predictions(mod_ehs) %>%
  ggplot(aes(x = education)) +
    geom_point(aes(y = log(income)), data = wages) +
    geom_line(aes(y = pred, color = sex)) +
    coord_trans(y = "exp")
```


log(income)
12
10
8
6
4
education
5
10
15
20
sex
male
female

visualizing multiple models

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

modelr

```
wages %>%
  data_grid(height, .model = mod_ehs) %>%
  spread_predictions(mod_h, mod_eh, mod_ehs)
```

```
# A tibble: 28 × 6
   height education    sex    mod_h    mod_eh  mod_ehs
    <dbl>     <dbl> <chr>     <dbl>     <dbl>    <dbl>
1      52        13   male  9.686327  9.663693 10.52399
2      54        13   male  9.790285  9.760310 10.53744
3      56        13   male  9.894243  9.856927 10.55089
4      57        13   male  9.946222  9.905236 10.55762
5      58        13   male  9.998201  9.953545 10.56435
6      59        13   male 10.050179 10.001853 10.57107
```

gather_predictions()

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

```
gather_predictions(data, …)
```

Adds predictions from each of these models

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

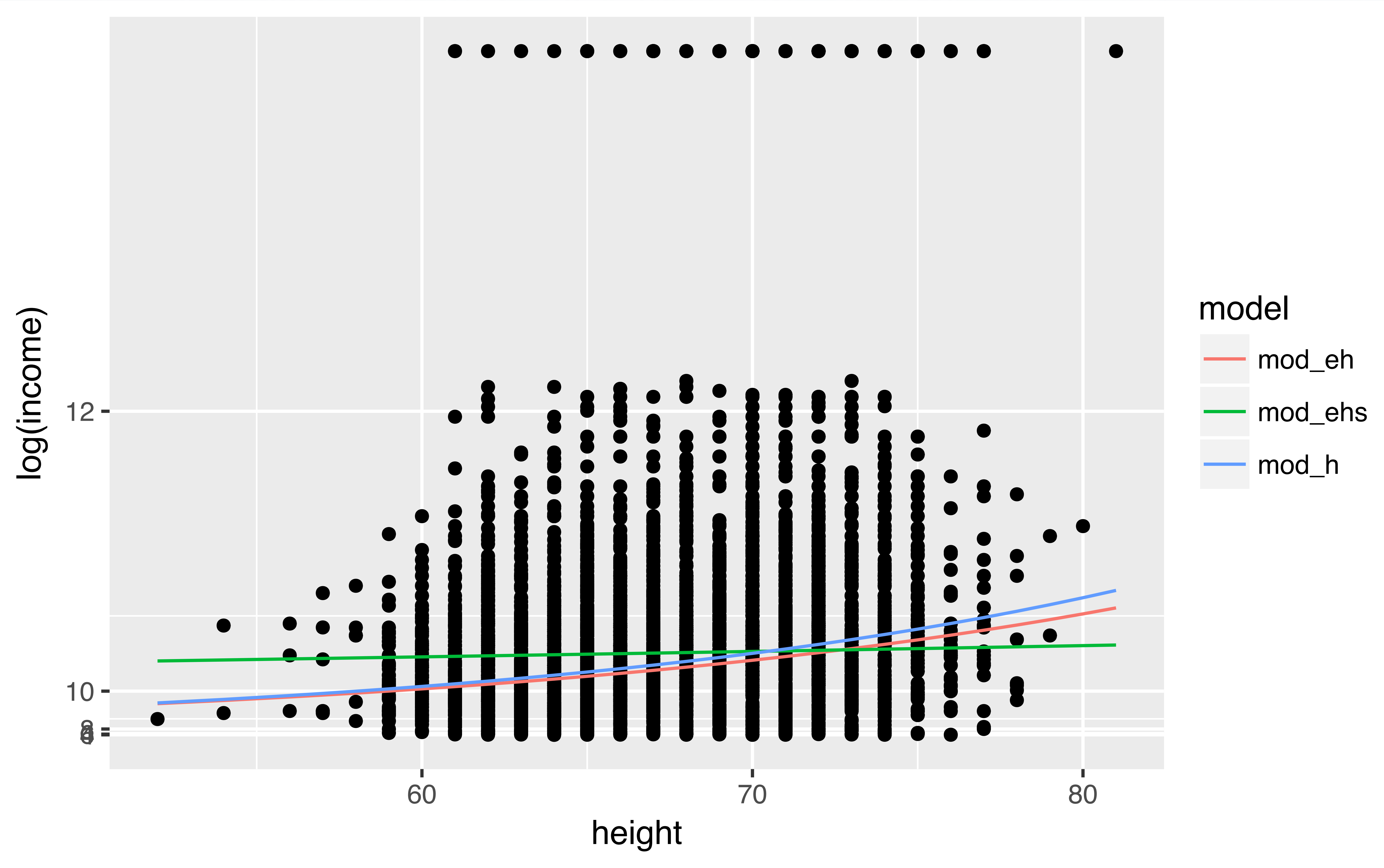
modelr

```r
wages %>%
  data_grid(height, .model = mod_ehs) %>%
  gather_predictions(mod_h, mod_eh, mod_ehs)
```

```
# A tibble: 84 × 5
   model height education    sex      pred
   <chr>  <dbl>     <dbl> <chr>     <dbl>
1  mod_h     52        13   male  9.686327
2  mod_h     54        13   male  9.790285
3  mod_h     56        13   male  9.894243
4  mod_h     57        13   male  9.946222
5  mod_h     58        13   male  9.998201
6  mod_h     59        13   male 10.050179
```

Your Turn 9

Use **data_grid()** and one of **gather_predictions()** or **spread_predictions()** to make the plot below. (Hint: only one works easily)



```
wages %>%
  data_grid(height, .model = mod_ehs) %>%
  gather_predictions(mod_h, mod_eh, mod_ehs) %>%
  ggplot(aes(x = height)) +
    geom_point(aes(y = log(income)), data = wages) +
    geom_line(aes(y = pred, color = model)) +
    coord_trans(y = "exp")
```

Residuals

Modelr provides the equivalent functions for residuals

```
add_predictions()     → add_residuals()
spread_predictions()  → spread_residuals()
gather_predictions()  → gather_residuals()
```

Instead of adding residuals to a data grid, you add them to the original data.

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

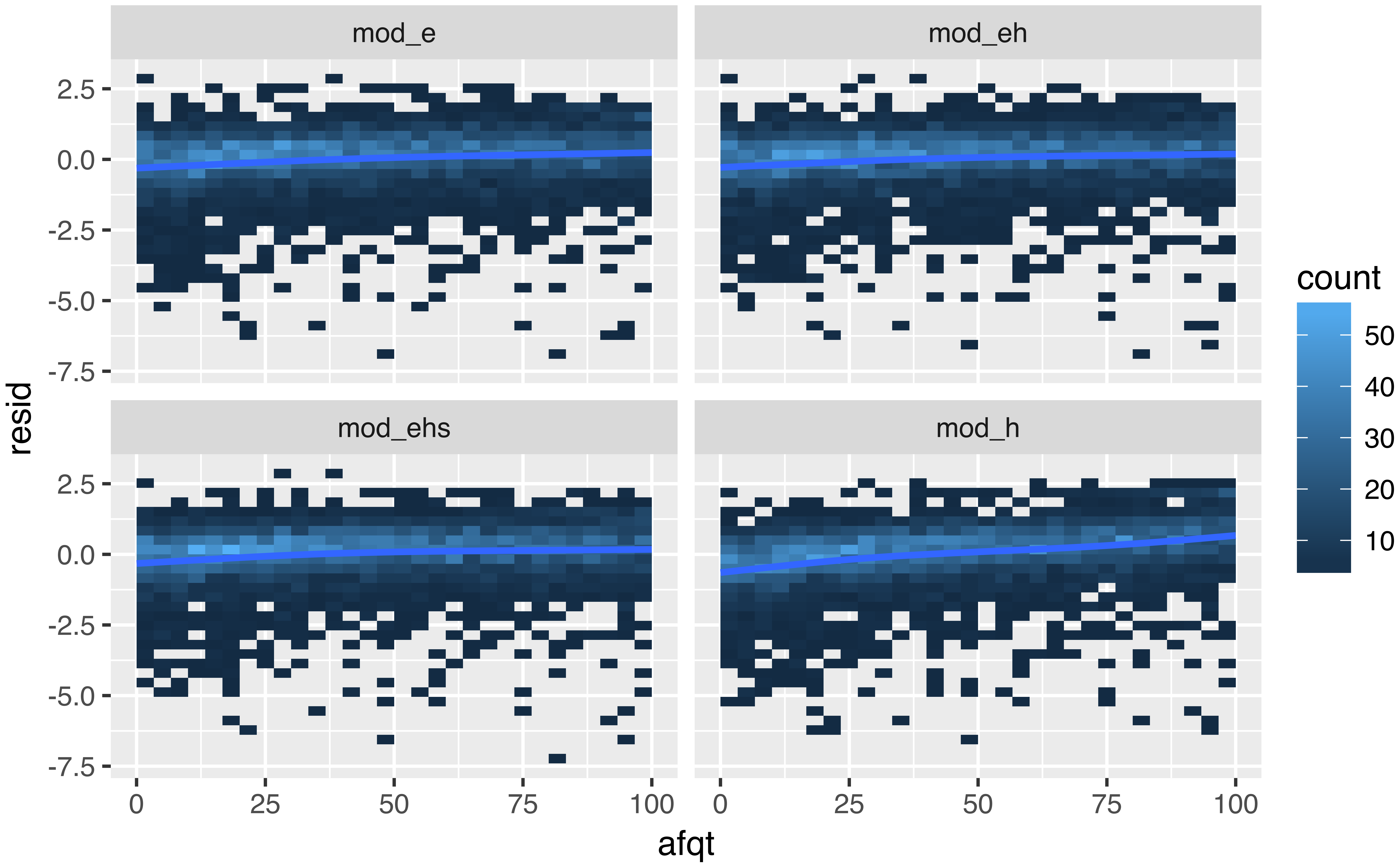
Modelr provides the equivalent functions for residuals

```
add_predictions()    →  add_residuals()
spread_predictions() →  spread_residuals()
gather_predictions() →  gather_residuals()
```

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

```
# A tibble: 5,266 × 9
   income height weight   age  marital    sex education   afqt       resid
    <int>  <dbl>  <int> <int>   <fctr> <fctr>     <int>  <dbl>       <dbl>
1   19000     60    155    53  married female        13  6.841 -0.54942114
2   35000     70    156    51  married female        10 49.444  0.48700905
3  105000     65    195    52  married   male        16 99.393  0.73457912
4   40000     63    197    54  married female        14 44.022  0.05317896
5   75000     66    190    49  married   male        14 59.683  0.68178762
6  102000     68    200    49 divorced female        18 98.798  0.42191085
7   70000     64    160    54 divorced female        12 50.283  0.89647549
8   60000     69    162    55 divorced   male        12 89.669  0.74232481
9  150000     69    194    54 divorced   male        13 95.977  1.51677517
```

```
wages %>%
  gather_residuals(mod_e, mod_h, mod_eh, mod_ehs) %>%
  ggplot(aes(afqt, resid)) +
    geom_bin2d() +
    geom_smooth() +
    facet_wrap(~model)
```


mod_e
mod_eh
mod_ehs
mod_h
resid
afqt
count
2.5
0.0
-2.5
-5.0
-7.5
0
25
50
75
100
10
20
30
40
50

Recap

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

Use **data_grid()** and **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