Modeling with

broom

modelr

The basics

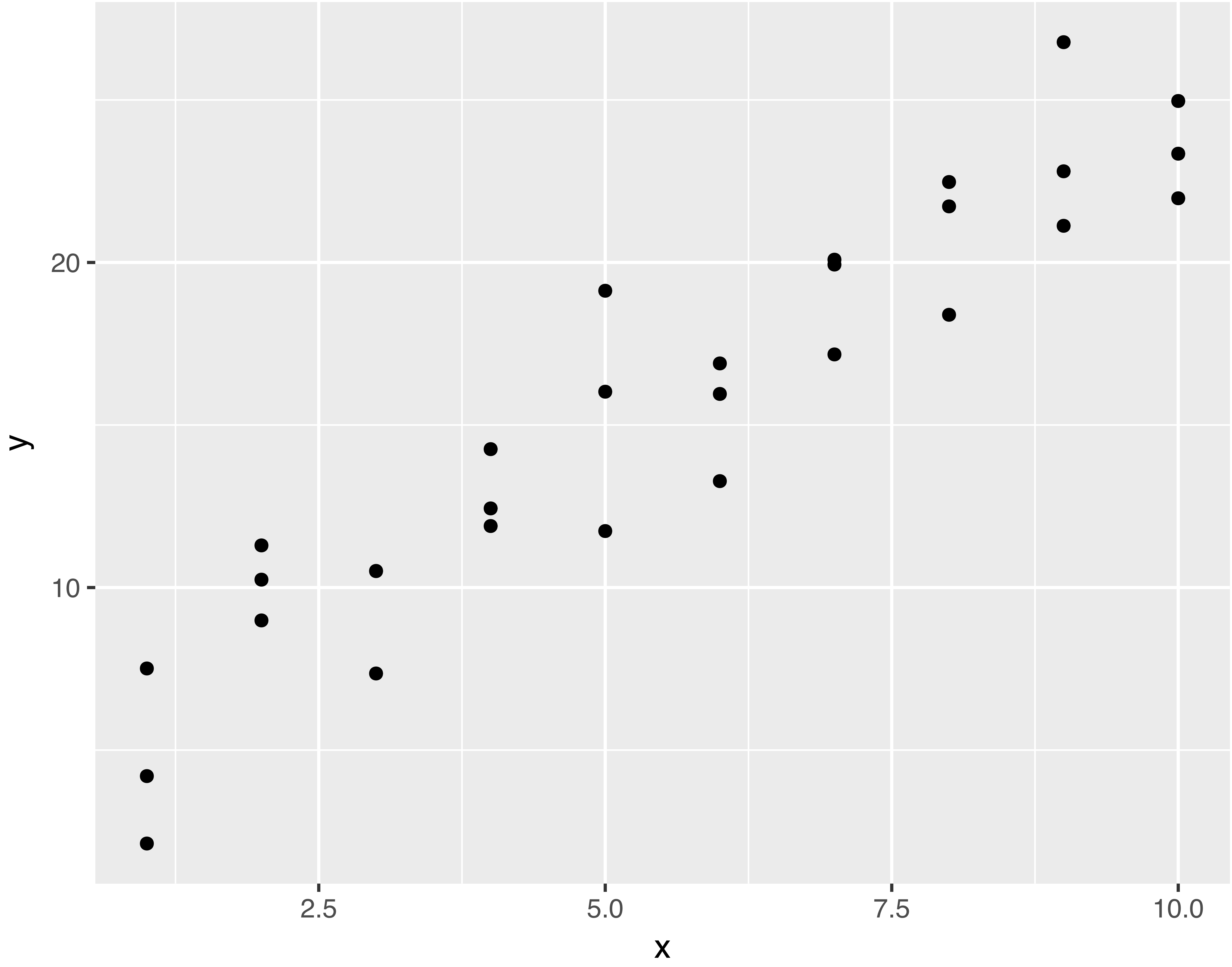
Models

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

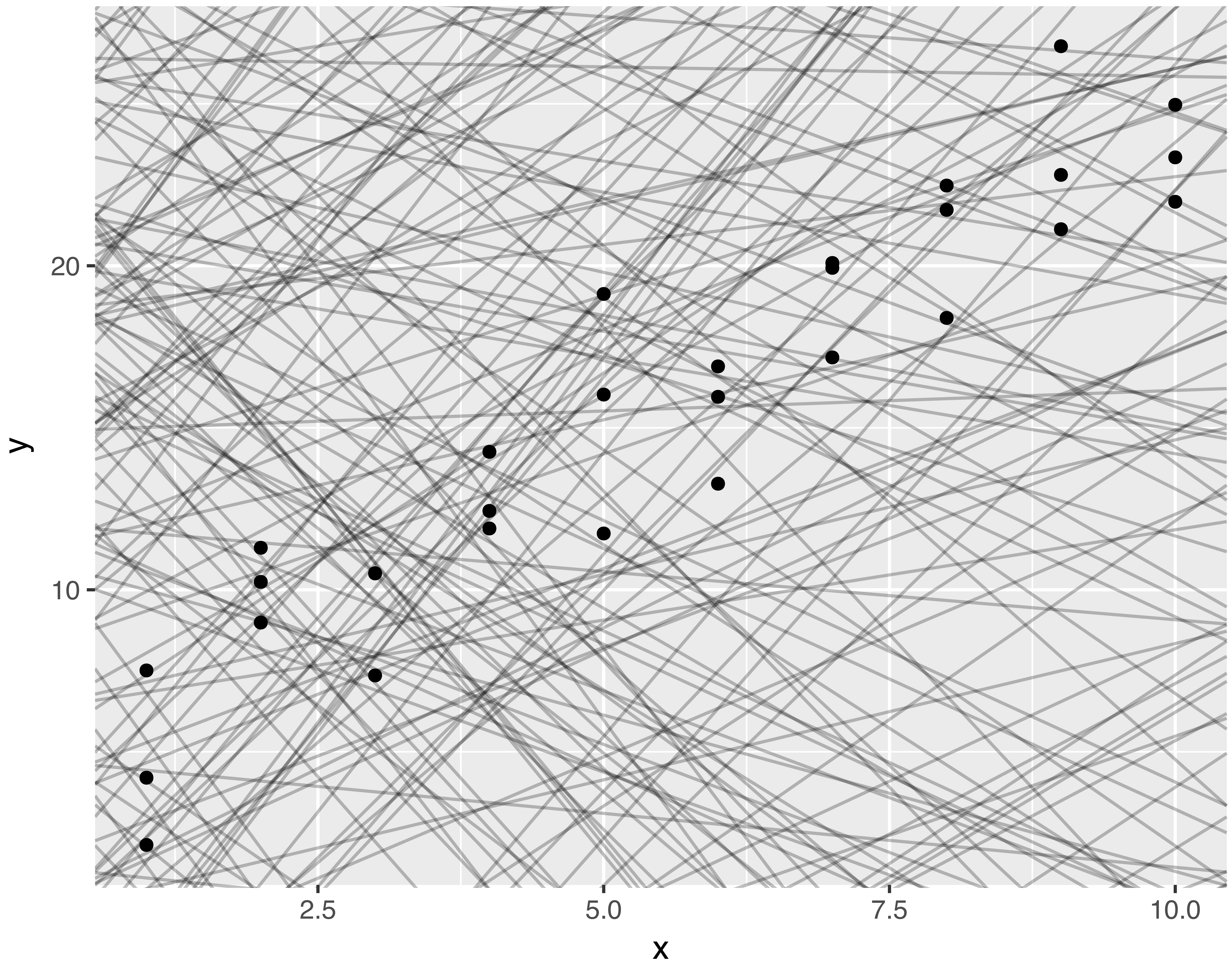
1. A family of functions
2. The function in the family that best approximates the data
3. Residuals

modelr

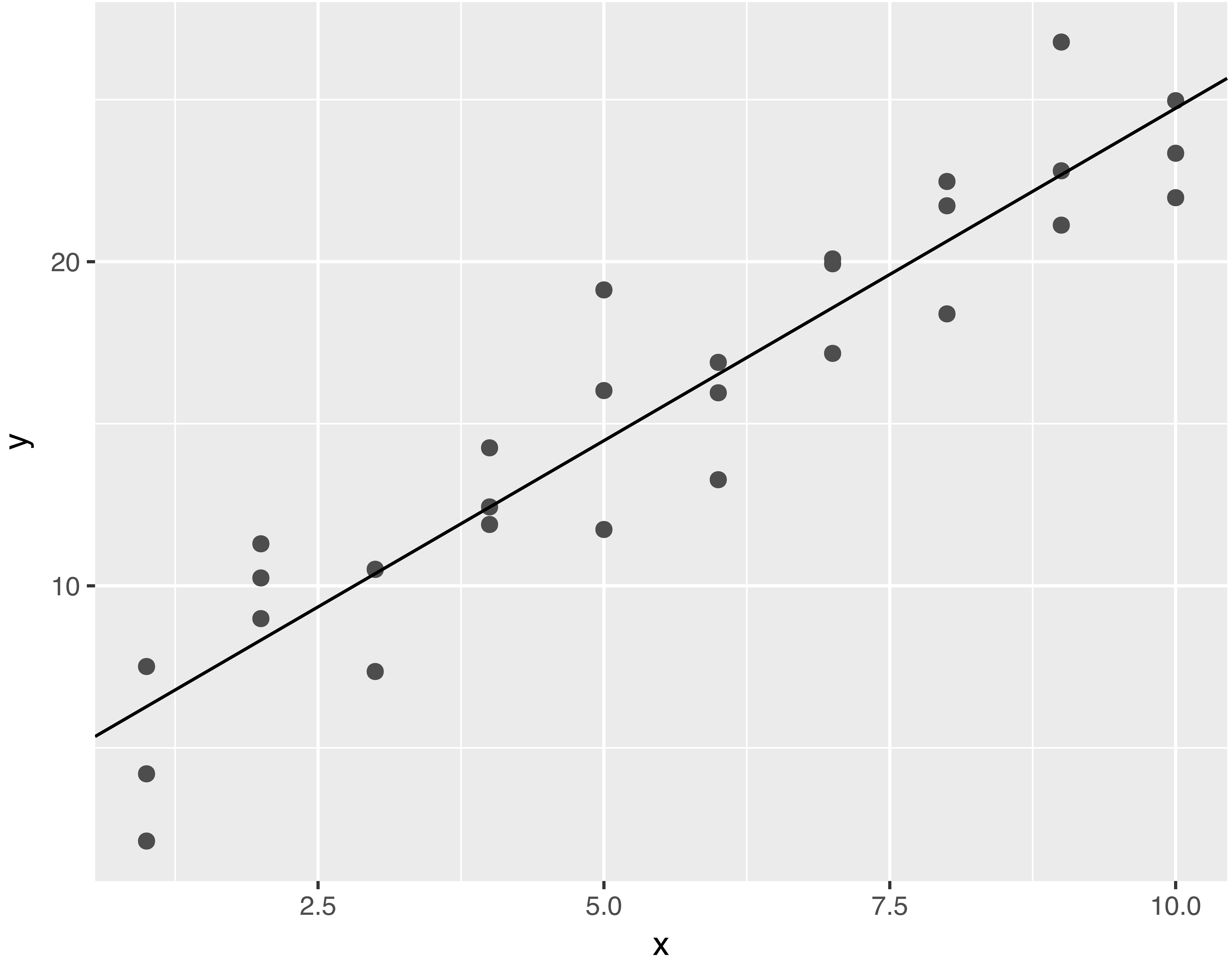
Example

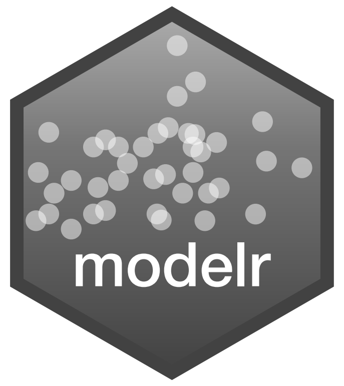


1. A family of functions

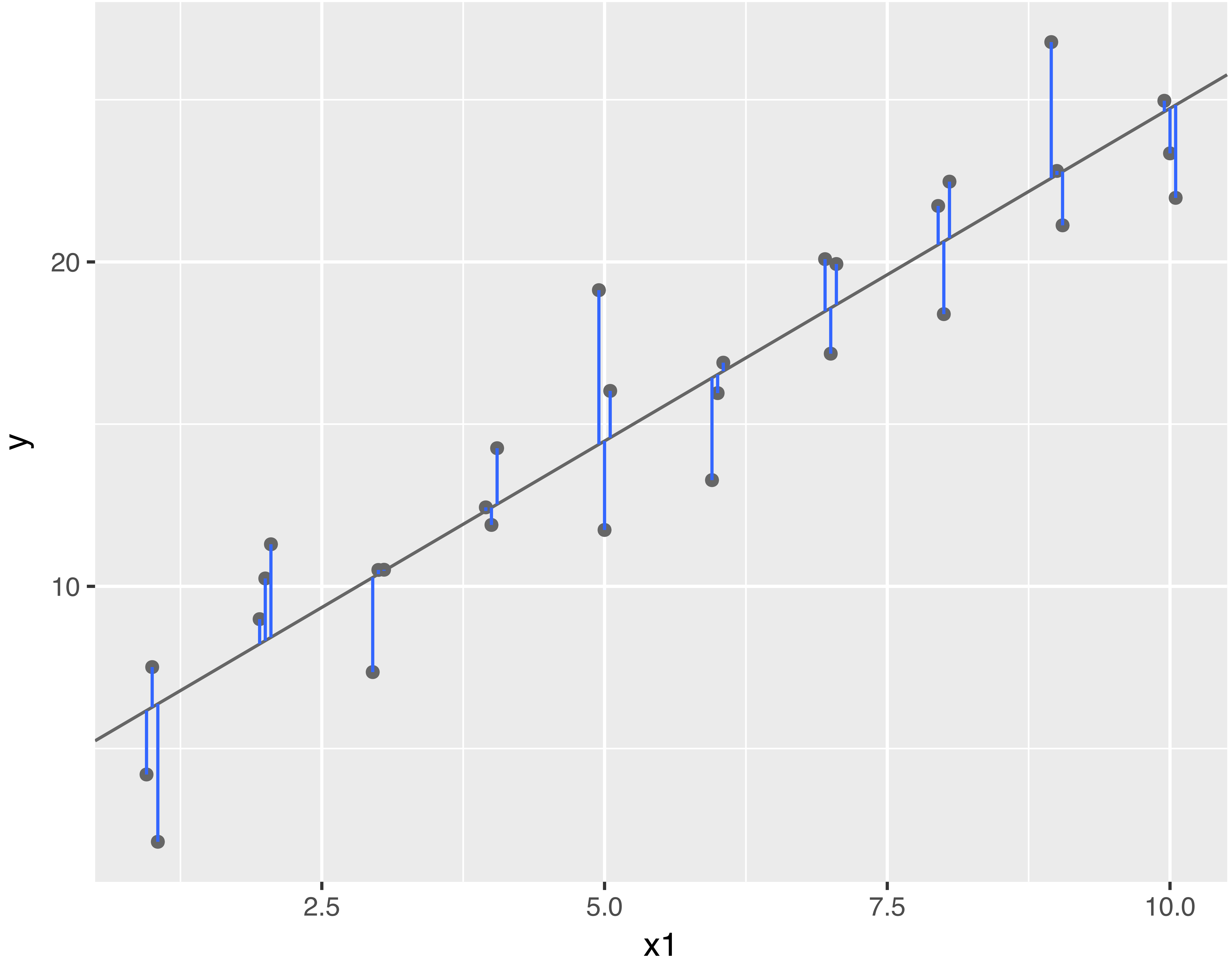


2. The best function of the family





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

modelr

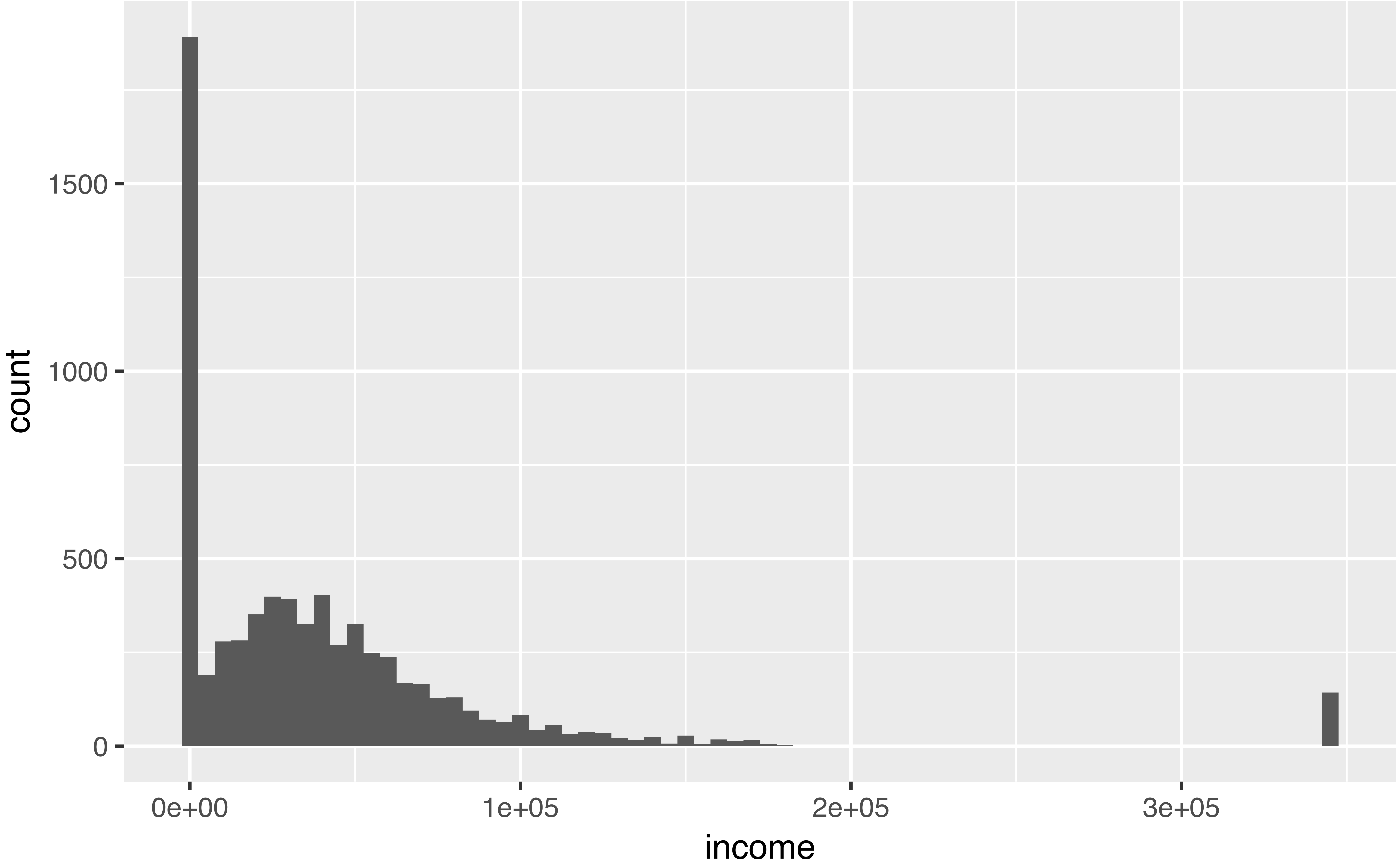
Tidy functions that make it easier to work with models in R

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

```
View(heights)
```

	income	height	weight	age	marital	sex	education	afqt
1	19000	60	155	53	married	female	13	6.841
2	35000	70	156	51	married	female	10	49.444
3	105000	65	195	52	married	male	16	99.393
4	40000	63	197	54	married	female	14	44.022
5	75000	66	190	49	married	male	14	59.683
6	102000	68	200	49	divorced	female	18	98.798
7	0	74	225	48	married	male	16	82.260
8	70000	64	160	54	divorced	female	12	50.283
9	60000	69	162	55	divorced	male	12	89.669
10	150000	69	194	54	divorced	male	13	95.977

```r
heights %>%
  ggplot(aes(income)) + geom_histogram(binwidth = 5000)
```

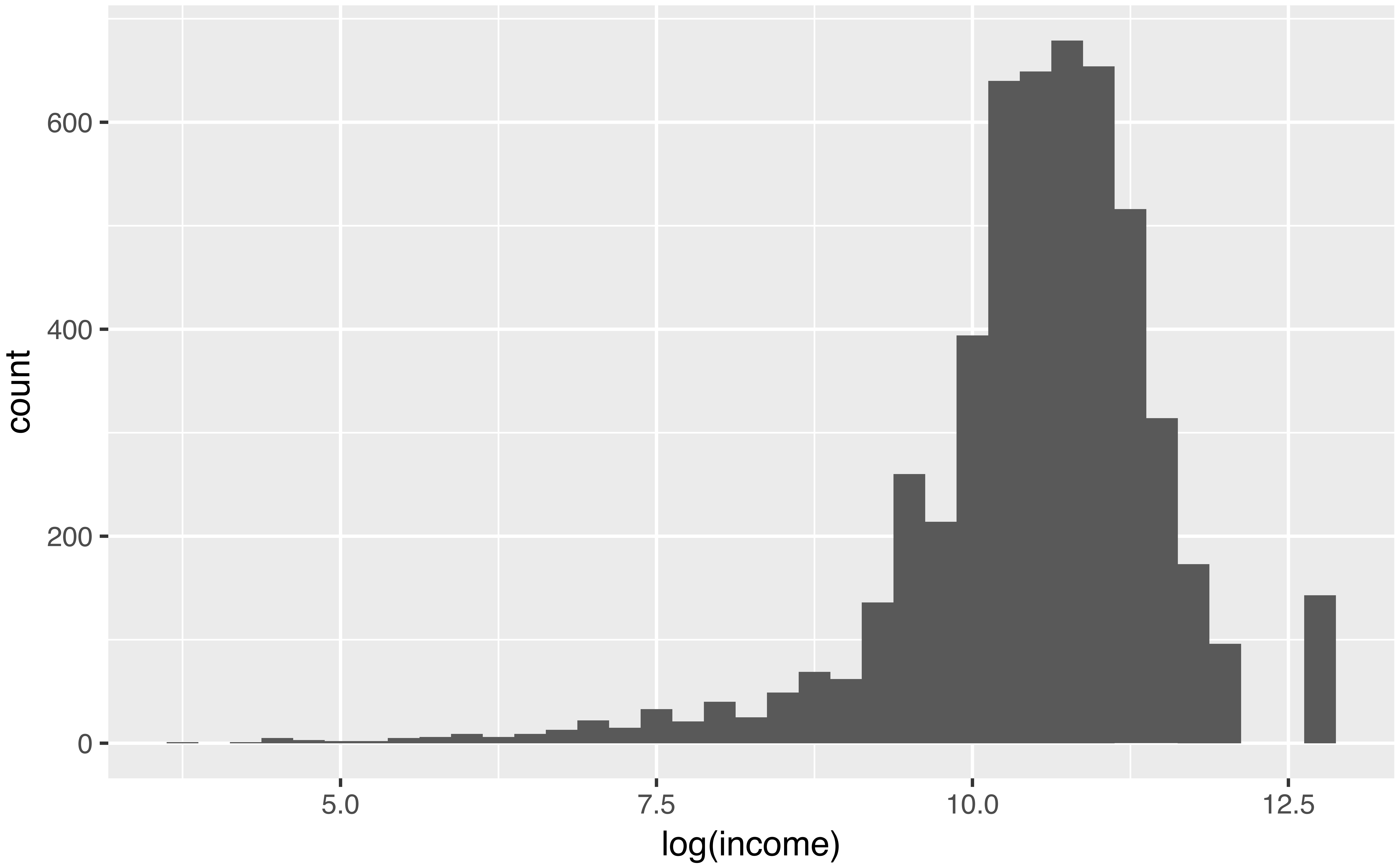


Your Turn

Create your own pre-processed data with

```
heights2 <- heights %>% filter(income > 0)
```

```r
heights2 %>%
  ggplot(aes(log(income)) + geom_histogram(binwidth = 0.25)
```



lm()

lm()

Fit a linear model to data

```
lm(log(income) ~ education, data = heights2)
```

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
```

Your Turn

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

```
mod_e <- lm(log(income) ~ education, data = heights2)
```

01:00

```
mod_e <- lm(log(income) ~ education, data = heights2)
mod_e
## Call:
## lm(formula = log(income) ~ education, data = heights2)
##
## 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 <- heights2 %>%
  lm(log(income) ~ education, data = .)
```

heights2 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() %>% View()
```

	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

broom

glance

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

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

	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual
1	0.1196233	0.119456	0.9923358	714.987	8.408952e-148	2	-7427.793	14861.59	14881.29	5181.651	5262

augment()

Returns **model output related to original data points** as a data frame

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

	.rownames	log.income	education	.fitted	.se.fit	.resid	.hat	.sigma	.cooksd	.std.resid
1	1	9.852194	13	10.401615	0.01400504	-0.549421141	0.0001991827	0.9924012	3.054133e-05	-0.553719667
2	2	10.463103	10	9.976094	0.02335067	0.487009048	0.0005537086	0.9924074	6.675581e-05	0.490906322
3	3	11.561716	16	10.827137	0.01880219	0.734579123	0.0003590043	0.9923784	9.843315e-05	0.740385454
4	4	10.596635	14	10.543456	0.01386811	0.053178965	0.0001953068	0.9924299	2.805560e-07	0.053594919
5	5	11.225243	14	10.543456	0.01386811	0.681787624	0.0001953068	0.9923856	4.611455e-05	0.687120418
6	6	11.532728	18	11.110817	0.02719979	0.421910848	0.0007513008	0.9924131	6.800811e-05	0.425329222
7	7	11.156251	12	10.259775	0.01600734	0.896475490	0.0002602083	0.9923532	1.062372e-04	0.903516852
8	8	11.002100	12	10.259775	0.01600734	0.742324811	0.0002602083	0.9923774	7.284298e-05	0.748155396
9	9	11.918391	13	10.401615	0.01400504	1.516775174	0.0001991827	0.9922098	2.327661e-04	1.528642020
10	10	11.652687	16	10.827137	0.01880219	0.825550901	0.0003590043	0.9923648	1.243231e-04	0.832076300
11	11	12.747903	16	10.827137	0.01880219	1.920766122	0.0003590043	0.9920766	6.729971e-04	1.935948427
12	12	10.596635	16	10.827137	0.01880219	-0.230501773	0.0003590043	0.9924251	9.691986e-06	-0.232323727

broom

augment()

Returns **model output related to original data points** as a data frame

```
mod_e %>% augment(data = heights2) %>% View()
```

Set data = to the original data set to include the full original data in the output.

broom

Your Turn

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 <- heights2 %>%
  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

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

03:00

```r
mod_eh <- heights2 %>%
  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

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

03:00

```
mod_ehs <- heights2 %>%
  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 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

data_grid()

Creates a data frame with useful combinations of values.

```
data_grid(data, …, .model)
```

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

1. Make range of x values

```
heights2 %>% data_grid(height, sex, .model = mod_ehs)
```

```
# A tibble: 56 × 3
   height    sex education
1      52   male        13
2      52 female        13
3      54   male        13
4      54 female        13
5      56   male        13
6      56 female        13
```

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

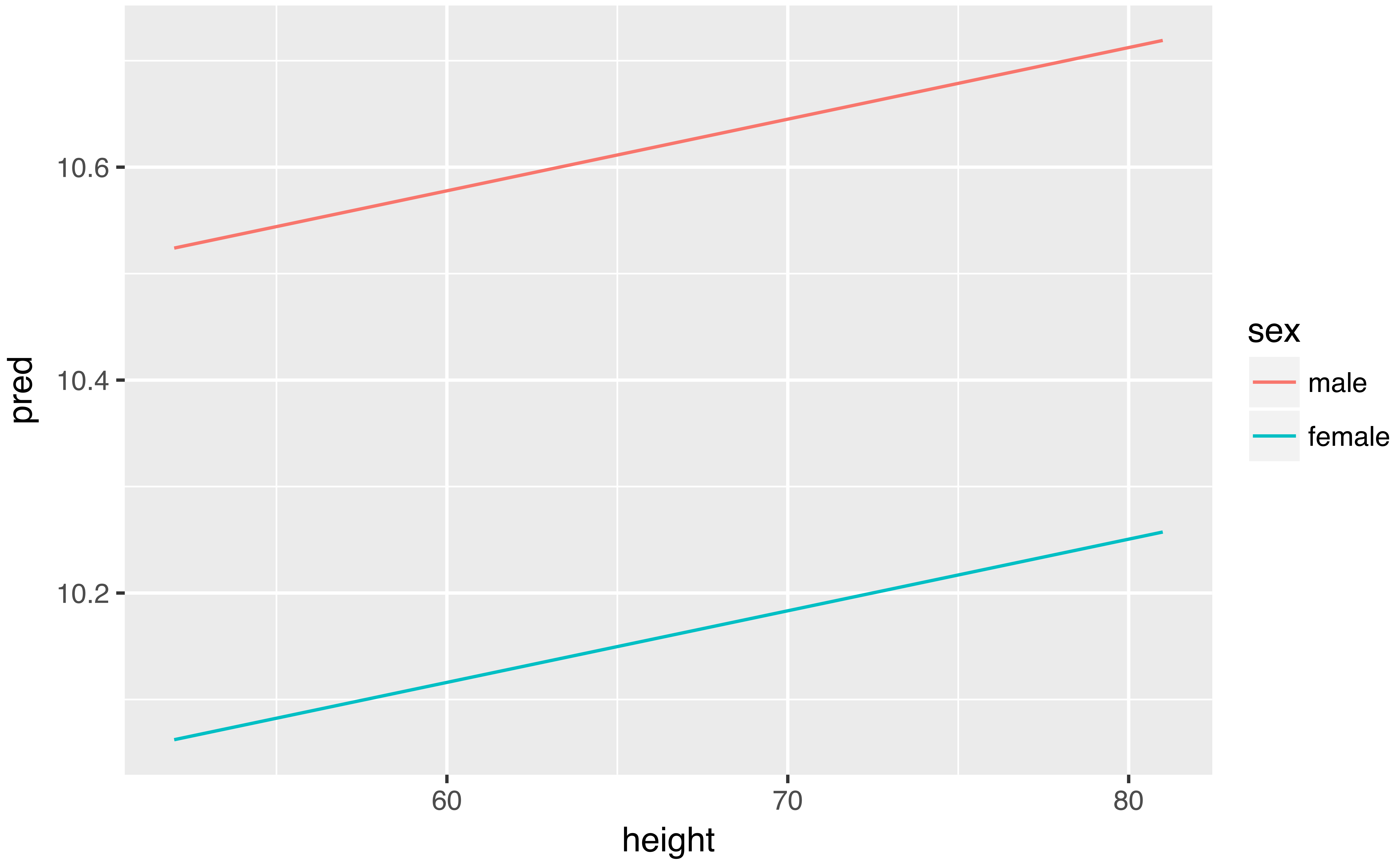
2. Add predictions

```
heights2 %>%
  data_grid(height, sex, .model = mod_ehs) %>%
  add_predictions(mod_ehs)
```

```
# A tibble: 56 × 4
   height    sex education     pred
1      52   male        13 10.52399
2      52 female        13 10.06224
3      54   male        13 10.53744
4      54 female        13 10.07569
5      56   male        13 10.55089
```

3. Plot

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



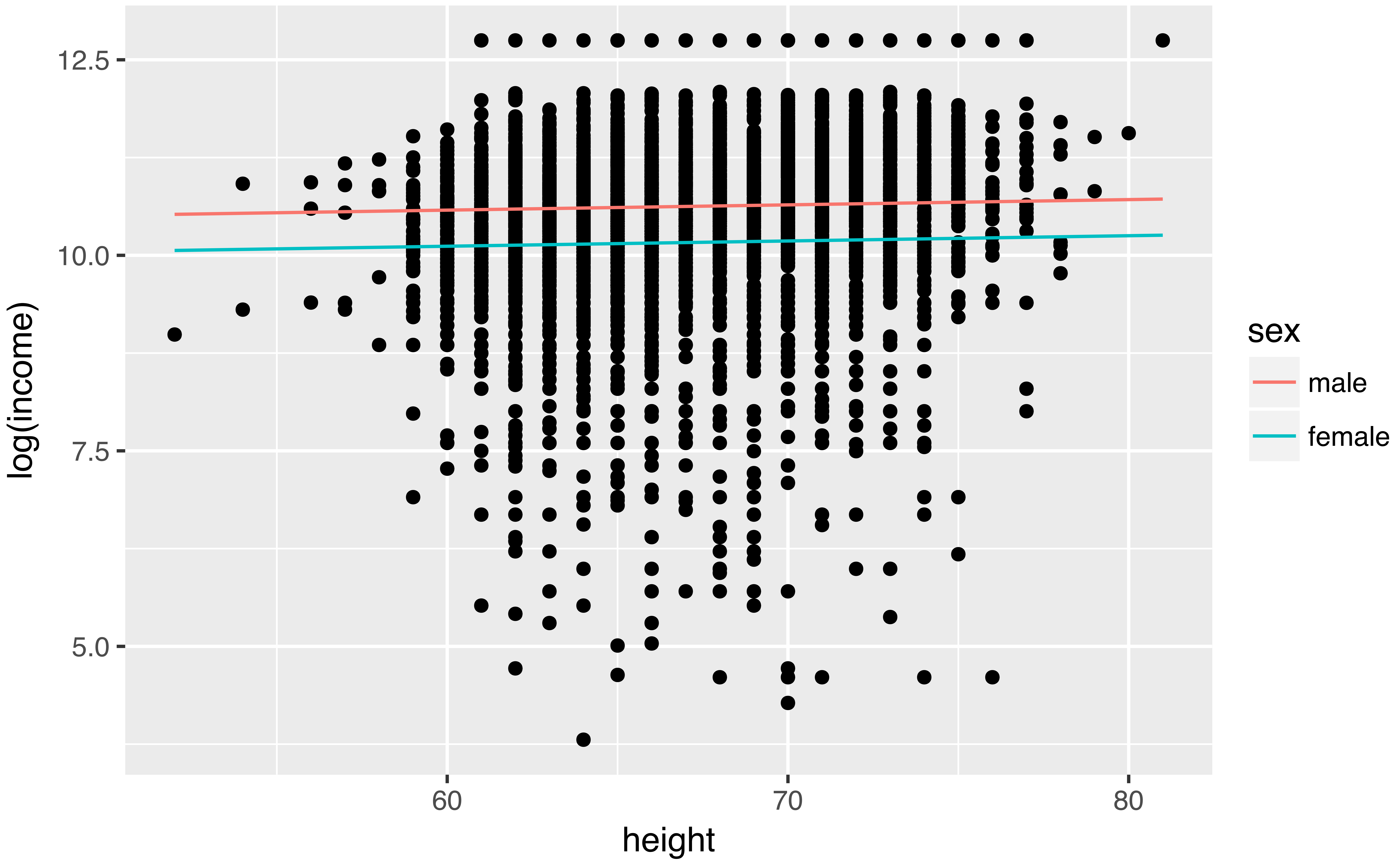
pred
10.6
10.4
10.2
60
70
80
height
sex
male
female

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 = heights2) +
    geom_line(aes(y = pred, color = sex))
```

Adds the original data points

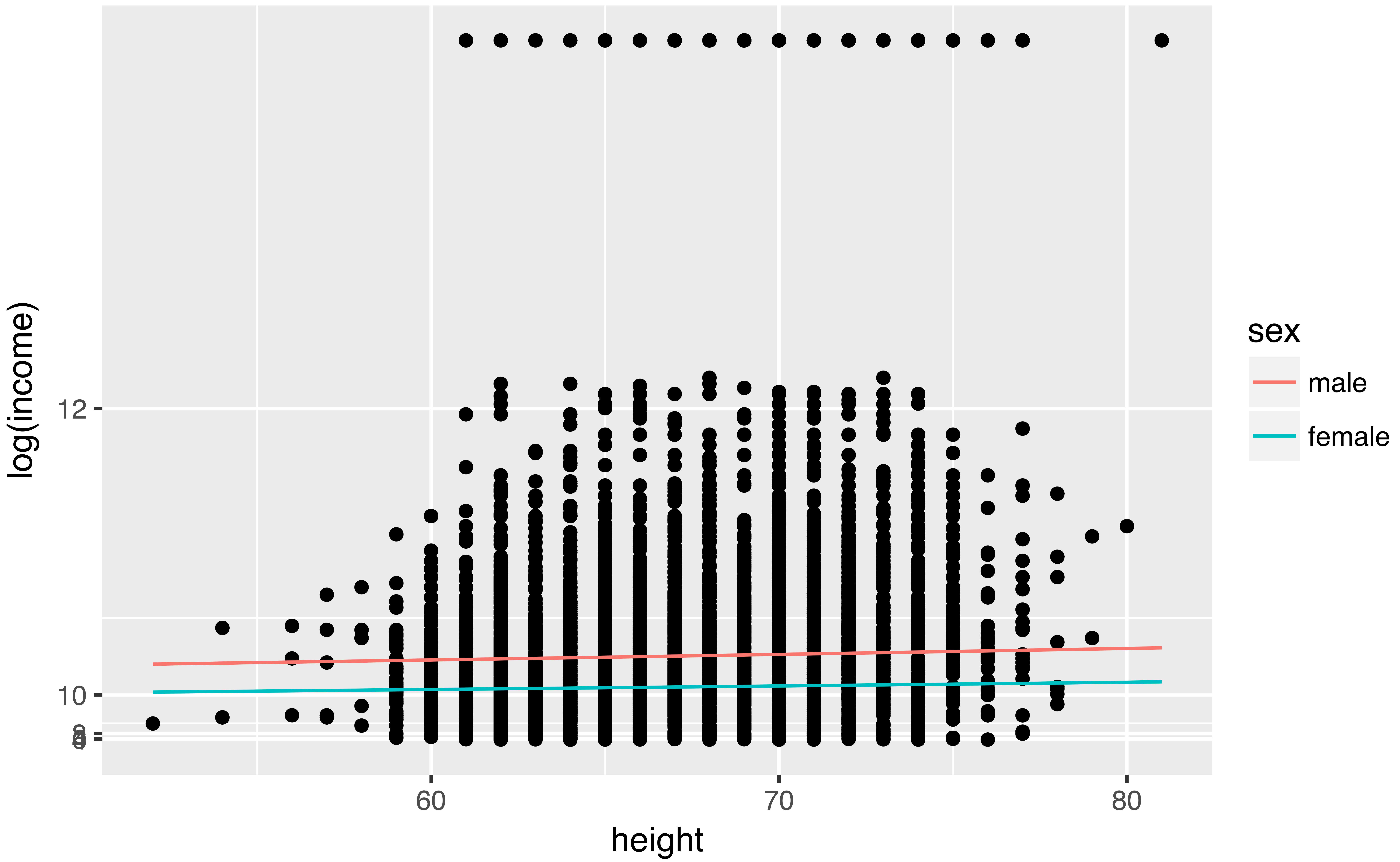
modelr


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 = heights2) +
    geom_line(aes(y = pred, color = sex)) +
    coord_trans(y = "exp")
```

Visually back-transforms the log



modelr

Your Turn

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

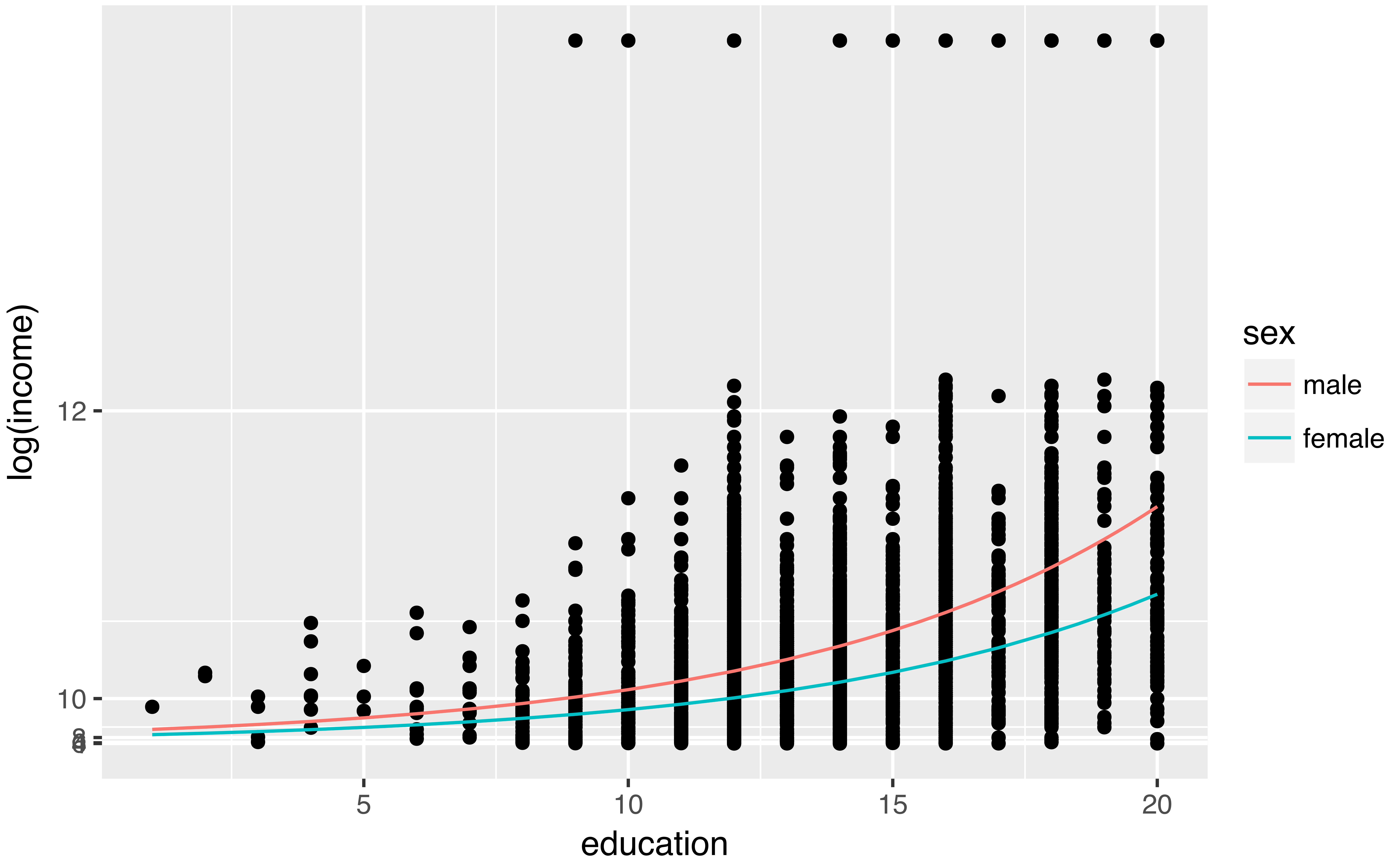
1. Make a range of x values to visualize over

2. Add predictions

3. Plot

06:00

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


log(income)
12
10
5
10
15
20
education
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

```r
heights2 %>%
  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)

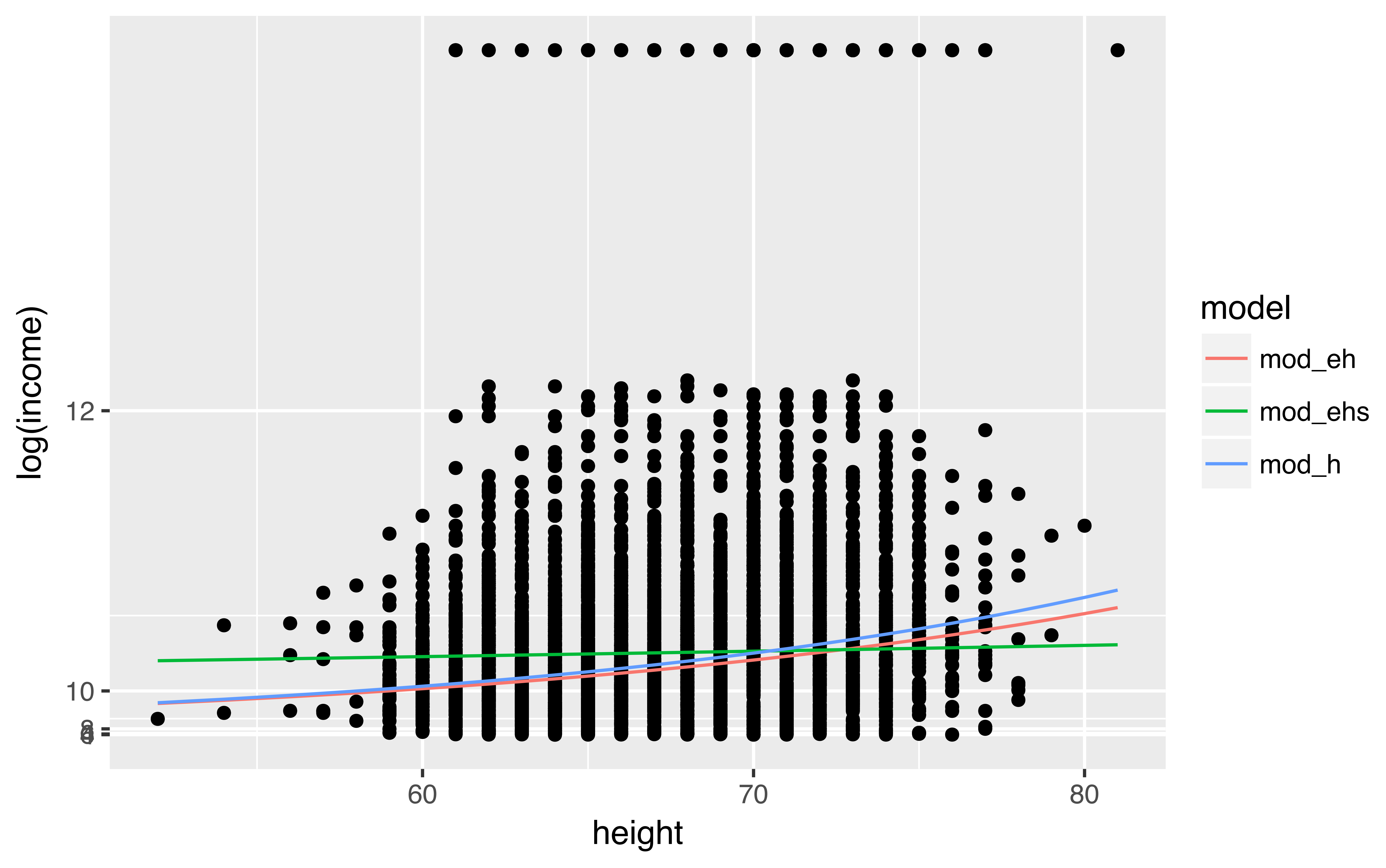
modelr

```
heights2 %>%
  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

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



```
heights2 %>%
  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 = heights2) +
    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.

```
heights2 %>%
  add_residuals(mod_e)
```

Modelr provides the equivalent functions for residuals

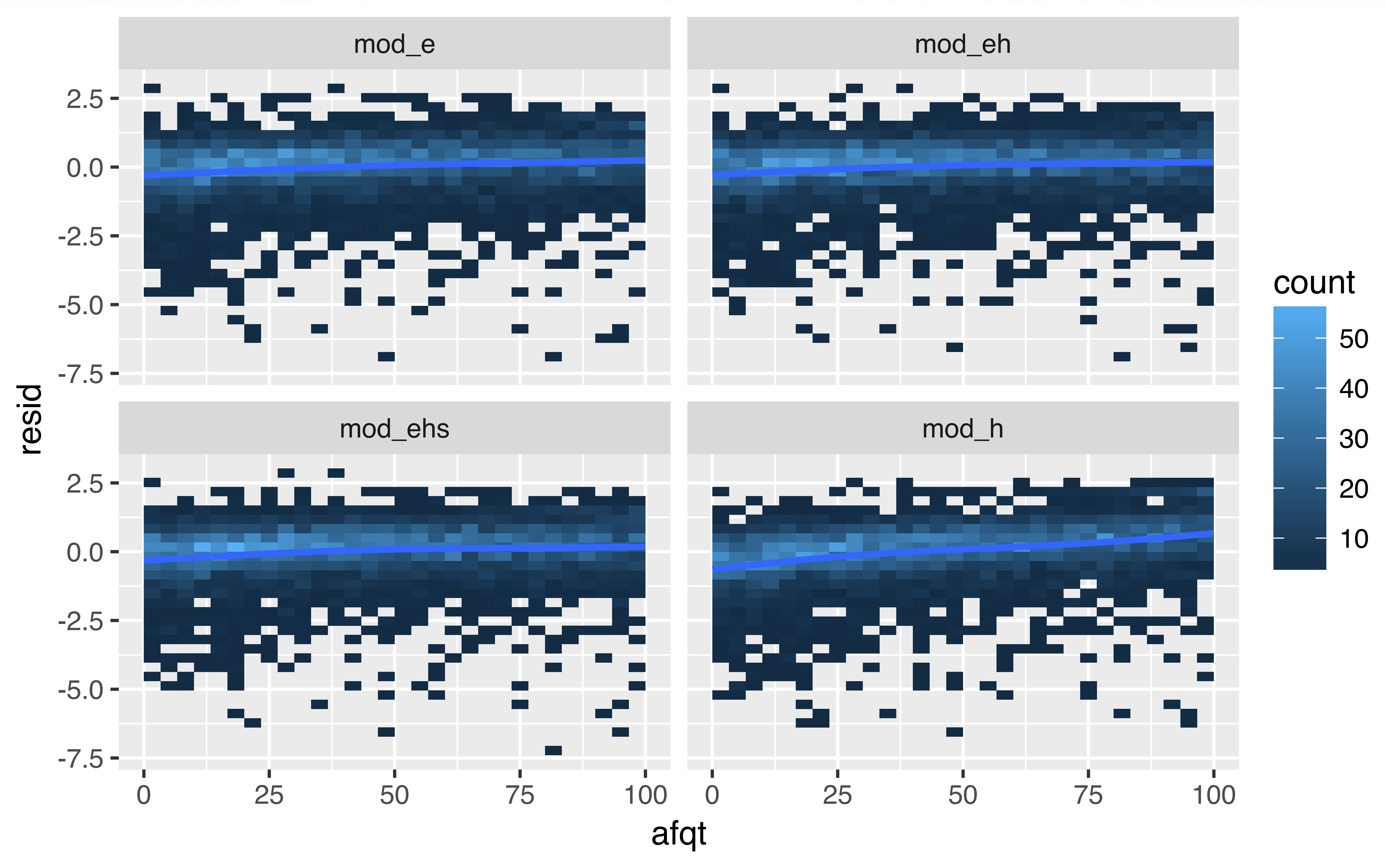
```
add_predictions()    → add_residuals()
spread_predictions() → spread_residuals()
gather_predictions() → gather_residuals()
```

```
heights2 %>%
  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
```

Your Turn

Use a modelr residual function with **geom_bin2d**, **geom_smooth**, and **facetting** to make this plot that compares patterns in the residuals.



```
heights2 %>%
  gather_residuals(mod_e, mod_h, mod_eh, mod_ehs) %>%
  ggplot(aes(afqt, resid)) +
    geom_bin2d() +
    geom_smooth() +
    facet_wrap(~model)
```

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 purrr

Your Turn

Use map() and lm() to fit each of the following formulas to heights2.

```
formulas <- list(
  mod_e = income ~ education,
  mod_eh = income ~ education + height,
  mod_ehs = income ~ education + height + sex
)
```

Use a different map function to apply glance() to each of the models and return the results as a data frame.

Which model has the lowest AIC?

05:00

```r
formulas <- list(
  mod_e = income ~ education,
  mod_eh = income ~ education + height,
  mod_ehs = income ~ education + height + sex
)

formulas %>%
  map(lm, data = heights2) %>%
  map_df(glance)
```

```r
formulas <- list(
  mod_e = income ~ education,
  mod_eh = income ~ education + height,
  mod_ehs = income ~ education + height + sex
)

formulas %>%
  map(lm, data = heights2) %>%
  map_df(glance, .id = "model")
```

Adds the list names as a column named model

gapminder

A subset of the data available at Hans Rosling's gapminder.org

```r
# install.packages("gapminder")
library(gapminder)
```

```
View(gapminder)
```

	country	continent	year	lifeExp	pop	gdpPercap
1	Afghanistan	Asia	1952	28.801	8425333	779.4453
2	Afghanistan	Asia	1957	30.332	9240934	820.8530
3	Afghanistan	Asia	1962	31.997	10267083	853.1007
4	Afghanistan	Asia	1967	34.020	11537966	836.1971
5	Afghanistan	Asia	1972	36.088	13079460	739.9811
6	Afghanistan	Asia	1977	38.438	14880372	786.1134
7	Afghanistan	Asia	1982	39.854	12881816	978.0114
8	Afghanistan	Asia	1987	40.822	13867957	852.3959
9	Afghanistan	Asia	1992	41.674	16317921	649.3414
10	Afghanistan	Asia	1997	41.763	22227415	635.3414
11	Afghanistan	Asia	2002	42.129	25268405	726.7341

```
View(gapminder)
```

	country	continent	year	lifeExp	pop	gdpPercap
1	Afghanistan	Asia	1952	28.801	8425333	779.4453
2	Afghanistan	[illegible]	[illegible]	[illegible]	[illegible]240934	820.8530
3	Afghanistan	[illegible]	[illegible]	[illegible]	[illegible]267083	853.1007
4	Afghanistan	[illegible]	[illegible]	[illegible]	[illegible]537966	836.1971
5	Afghanistan	Asia	1972	36.088	13079460	739.9811
6	Afghanistan	Asia	1977	38.438	14880372	786.1134
7	Afghanistan	Asia	1982	39.854	12881816	978.0114
8	Afghanistan	Asia	1987	40.822	13867957	852.3959
9	Afghanistan	Asia	1992	41.674	16317921	649.3414
10	Afghanistan	Asia	1997	41.763	22227415	635.3414
11	Afghanistan	Asia	2002	42.129	25268405	726.7341
12	Afghanistan	Asia	2007	43.828	31889923	974.5803

Which countries saw the most rapid improvement in life expectancy?

split()

An easy way to break our data frame into a list of smaller data frames

```
split(gapminder, gapminder$country)
```

Splits gapminder...

into one data set for each country

purrr

Your Turn

Run the code below to split gap minder into a list of data frames, one for each country.

```
split(gapminder, gapminder$country)
```

Use map() to apply the the model `lm(lifeExp ~year, data = ______)` to each data frame in the list. Note: you will need to write a function.

Use a map function to apply tidy to each model and combine the results into a data frame.

Filter the results to just rows where term equal "year"

Arrange in descending order of estimate (i.e. slope)

10:00

```r
split(gapminder, gapminder$country) %>%
  map(function(df) lm(lifeExp ~ year, data = df)) %>%
  map_df(tidy, .id = "country") %>%
  filter(term == "year") %>%
  arrange(desc(estimate))
```

List columns

Quiz

How is a data frame/tibble similar to a list?

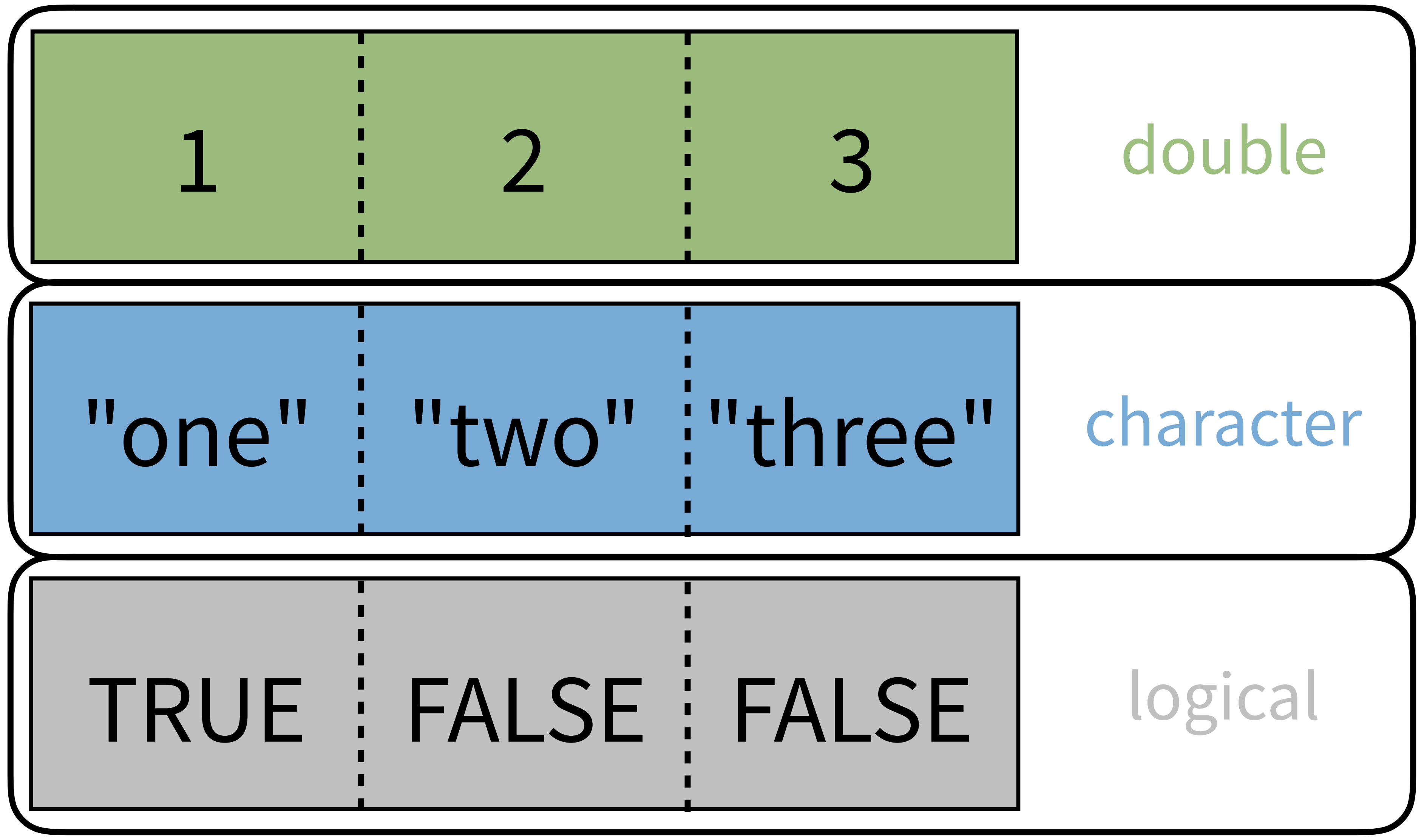
A data frame/tibble is a list!

data frame

num	cha	log
1	"one"	TRUE
2	"two"	FALSE
3	"three"	FALSE

=

List



+ class = "data.frame"

A data frame/tibble is a list!

data frame

num	cha	log
1	"one"	TRUE
2	"two"	FALSE
3	"three"	FALSE

`df["num"]`

num
1
2
3

`df[["num"]]`

`df$num`

`c(1, 2, 3)`

A data frame/tibble is a list!

data frame

num	cha	log
1	"one"	TRUE
2	"two"	FALSE
3	"three"	FALSE

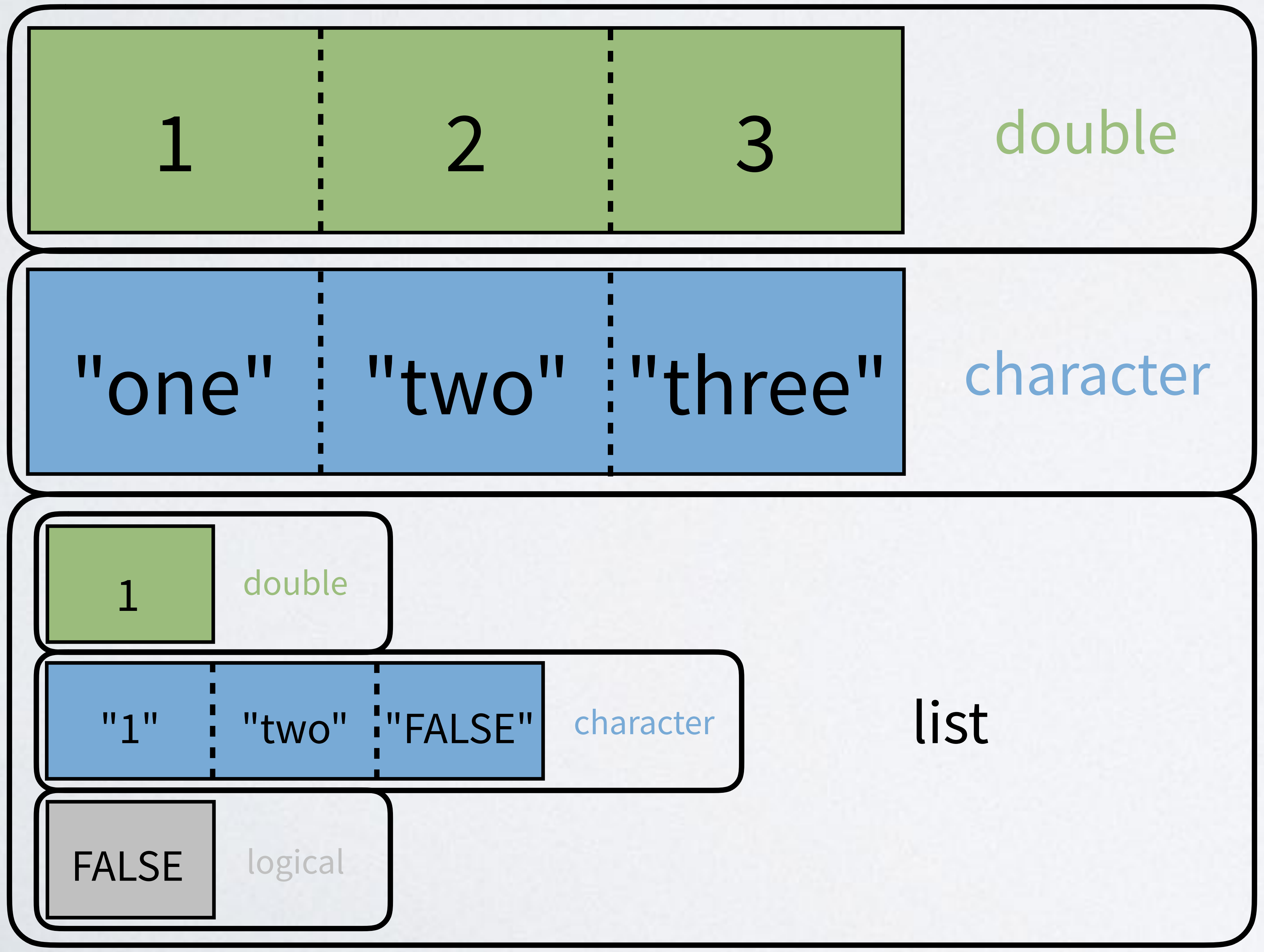
`df %>% select(num)`

num
1
2
3

Quiz

If you one of the elements of a list can be another list, can one of the columns of a data frame be another list?

List



$\stackrel{?}{=}$

data frame

num	cha	listcol
1	"one"	1
2	"two"	c("1", "two", "FALSE")
3	"three"	FALSE

Yes.

```
tibble(
  num = c(1, 2, 3),
  cha = c("one", "two", "three"),
  listcol = list(1, c("1", "two", "FALSE"), FALSE)
)
```

```
# A tibble: 3 × 3
    num   cha   listcol
  <dbl> <chr>    <list>
1     1   one <dbl [1]>
2     2   two <chr [3]>
3     3 three <lgl [1]>
```

nesting

A **nested data frame** stores individual tables within the cells of a larger, organizing table.

nested data frame

Species	data
setosa	<tibble [50 x 4]>
versicolor	<tibble [50 x 4]>
virginica	<tibble [50 x 4]>

n_iris

"cell" contents

Sepal.L	Sepal.W	Petal.L	Petal.W
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

n_iris$data[[1]]

Sepal.L	Sepal.W	Petal.L	Petal.W
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
6.5	2.8	4.6	1.5

n_iris$data[[2]]

Sepal.L	Sepal.W	Petal.L	Petal.W
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8
6.5	3.0	5.8	2.2

n_iris$data[[3]]

Use a nested data frame to:

- preserve relationships between observations and subsets of data
- manipulate many sub-tables at once with the **purrr** functions **map()**, **map2()**, or **pmap()**.

nest()

Places grouped cases into a list column.

```
gapminder %>%
  group_by(country) %>%
  nest()
```

Species	S.L	S.W	P.L	P.W
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
setosa	5.0	3.6	1.4	0.2
versi	7.0	3.2	4.7	1.4
versi	6.4	3.2	4.5	1.5
versi	6.9	3.1	4.9	1.5
versi	5.5	2.3	4.0	1.3
versi	6.5	2.8	4.6	1.5
virgini	6.3	3.3	6.0	2.5
virgini	5.8	2.7	5.1	1.9
virgini	7.1	3.0	5.9	2.1
virgini	6.3	2.9	5.6	1.8
virgini	6.5	3.0	5.8	2.2

→

Species	S.L	S.W	P.L	P.W
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3.0	1.4	0.2
setosa	4.7	3.2	1.3	0.2
setosa	4.6	3.1	1.5	0.2
setosa	5.0	3.6	1.4	0.2
versi	7.0	3.2	4.7	1.4
versi	6.4	3.2	4.5	1.5
versi	6.9	3.1	4.9	1.5
versi	5.5	2.3	4.0	1.3
versi	6.5	2.8	4.6	1.5
virgini	6.3	3.3	6.0	2.5
virgini	5.8	2.7	5.1	1.9
virgini	7.1	3.0	5.9	2.1
virgini	6.3	2.9	5.6	1.8
virgini	6.5	3.0	5.8	2.2

→

Species	data
setos	<tibble [50x4]>
versi	<tibble [50x4]>
virgini	<tibble [50x4]>

S.L	S.W	P.L	P.W
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

S.L	S.W	P.L	P.W
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
6.5	2.8	4.6	1.5

S.L	S.W	P.L	P.W
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8
6.5	3.0	5.8	2.2

*n_iris <- iris %>% **group_by**(Species) %>% **nest**()*

```r
gapminder %>%
  group_by(country) %>%
  nest() %>%

# Apply model to each table in the column
  mutate(model = map(function(df) lm(lifeExp ~ year, data = df)) %>%

# Extract coefficients and save in coefficient column
  mutate(coeffcent = map_dbl(function(df) tidy(df)$estimate[2]) %>%

# Extract adj.r.squares and save in adj_r column
  mutate(adj_r = map_dbl(function(df) glance(df)$adj.r.squared))
```

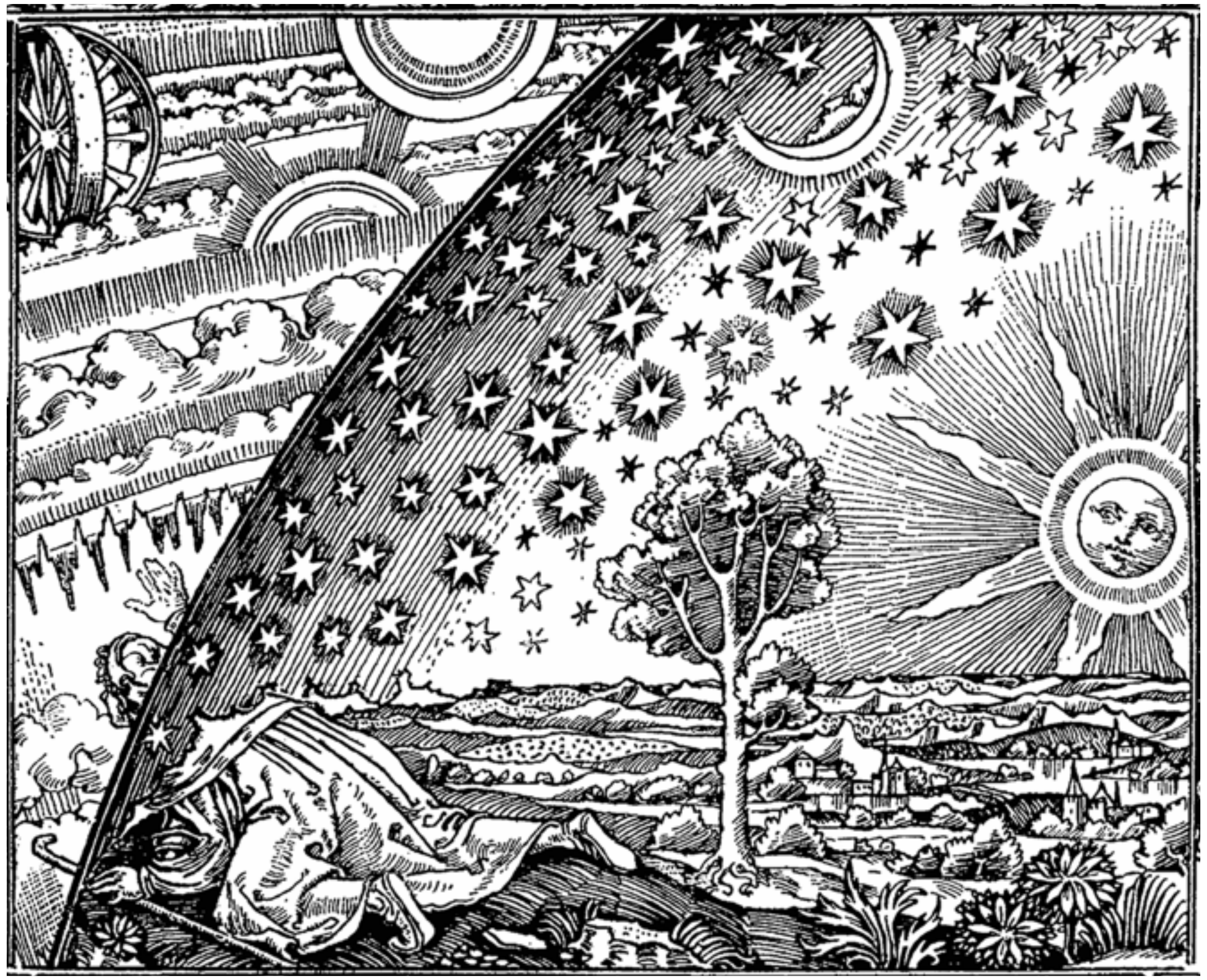
"Better experimental design = simpler statistics.
Better data model = simpler analysis."

- Jenny Bryan (2016)

Modeling with

broom modelr purrr

Thank You



Please take the class survey

www.surveymonkey.com/r/SX9X69R