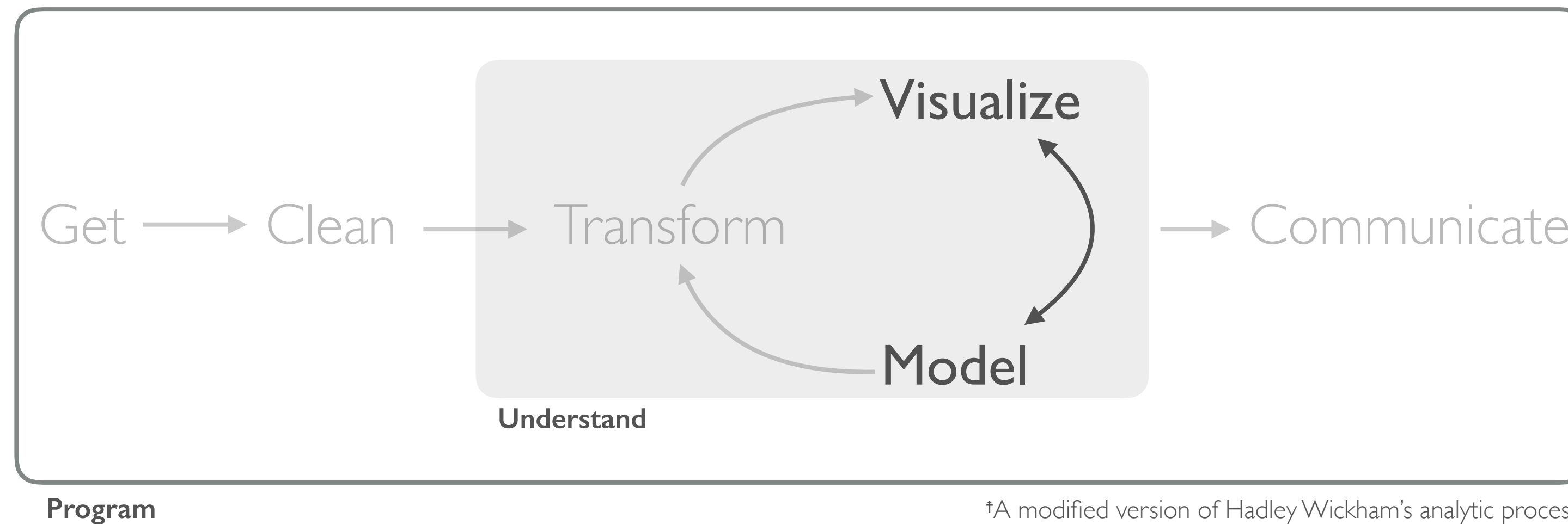


# MODELING BASICS



“All models are wrong, but some are useful.”

– George Box

# INTRODUCTION TO APPLIED MODELING

The next three sections are not going to give you a deep understanding of the mathematical theory that underlies models.

They are meant to build your intuition about how modeling works within R, and to give you a family of useful tools that allow you to use models to better understand your data through:

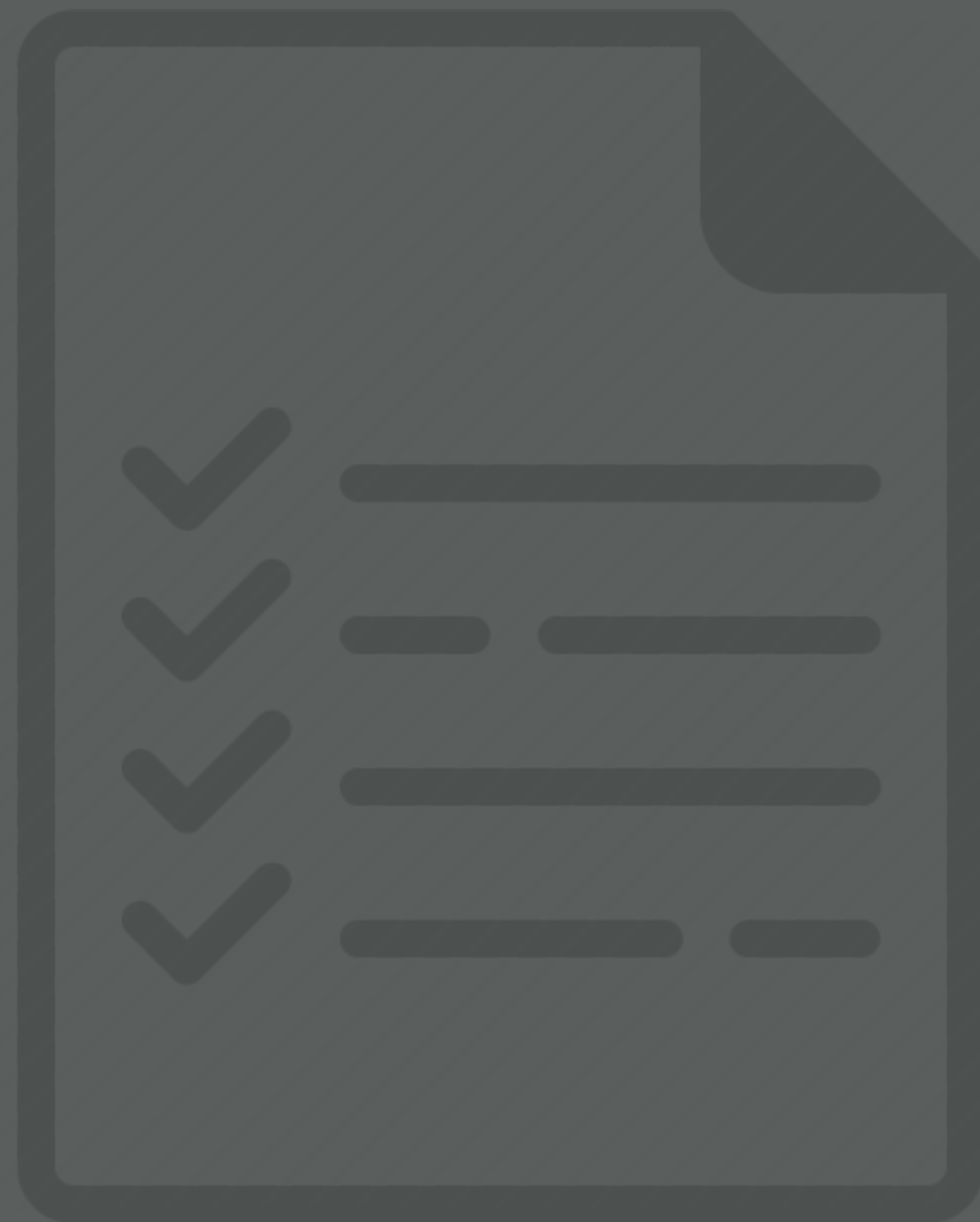
1. modeling basics
2. model building
3. managing many models

# INTRODUCTION TO APPLIED MODELING

These sections may feel overwhelming but they are meant to give you a flavor of what you can do in R and to begin preparing you for the Applied Analytics with R course.

***We will apply purposeful model specifications***

# PREREQUISITES



# PREREQUISITES

```
library(tidyverse)
```

```
library(modelr)
```

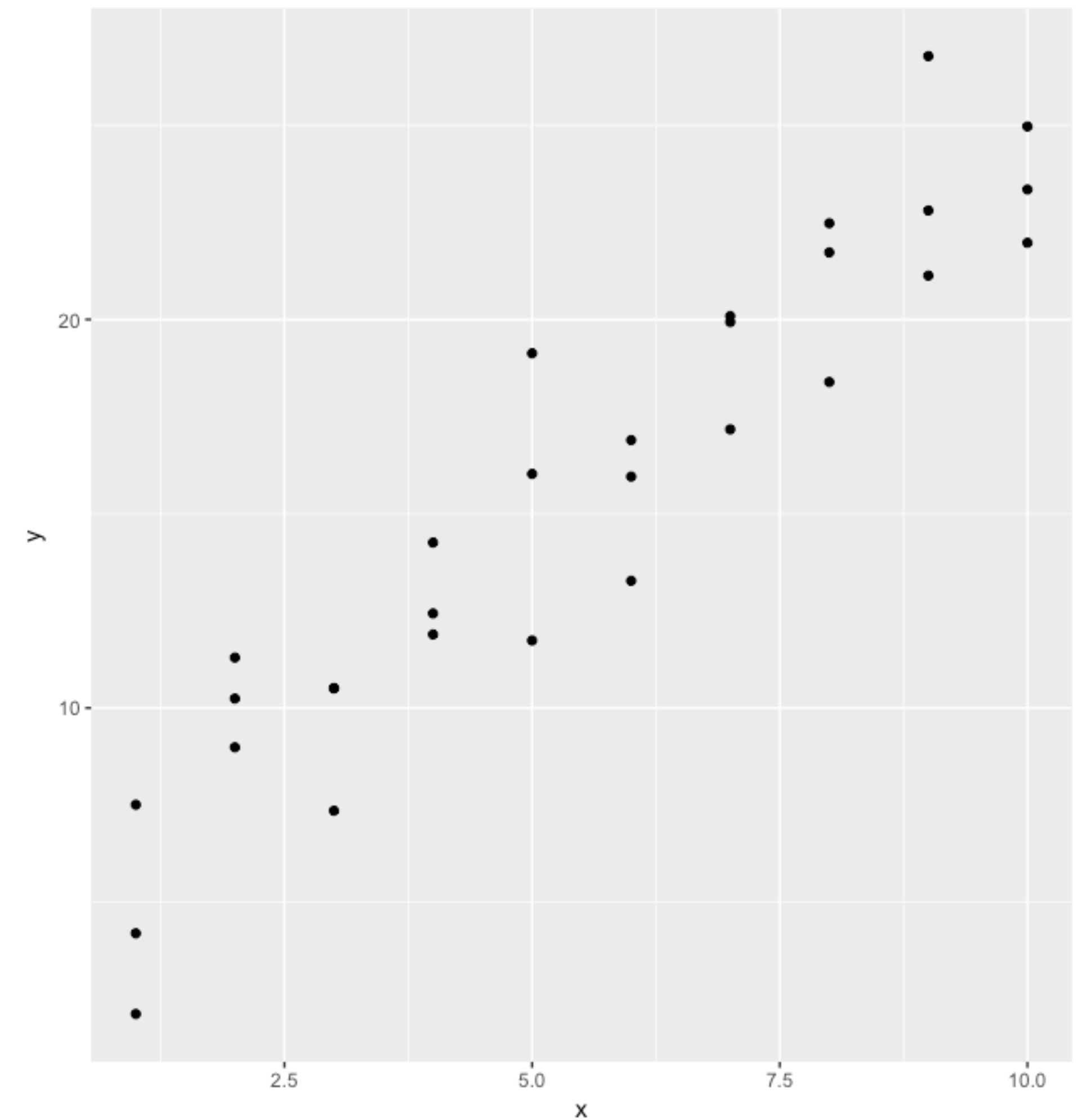
```
options(na.action = na.warn)
```

PRE-MODELING



# IDENTIFYING (LINEAR) RELATIONSHIPS

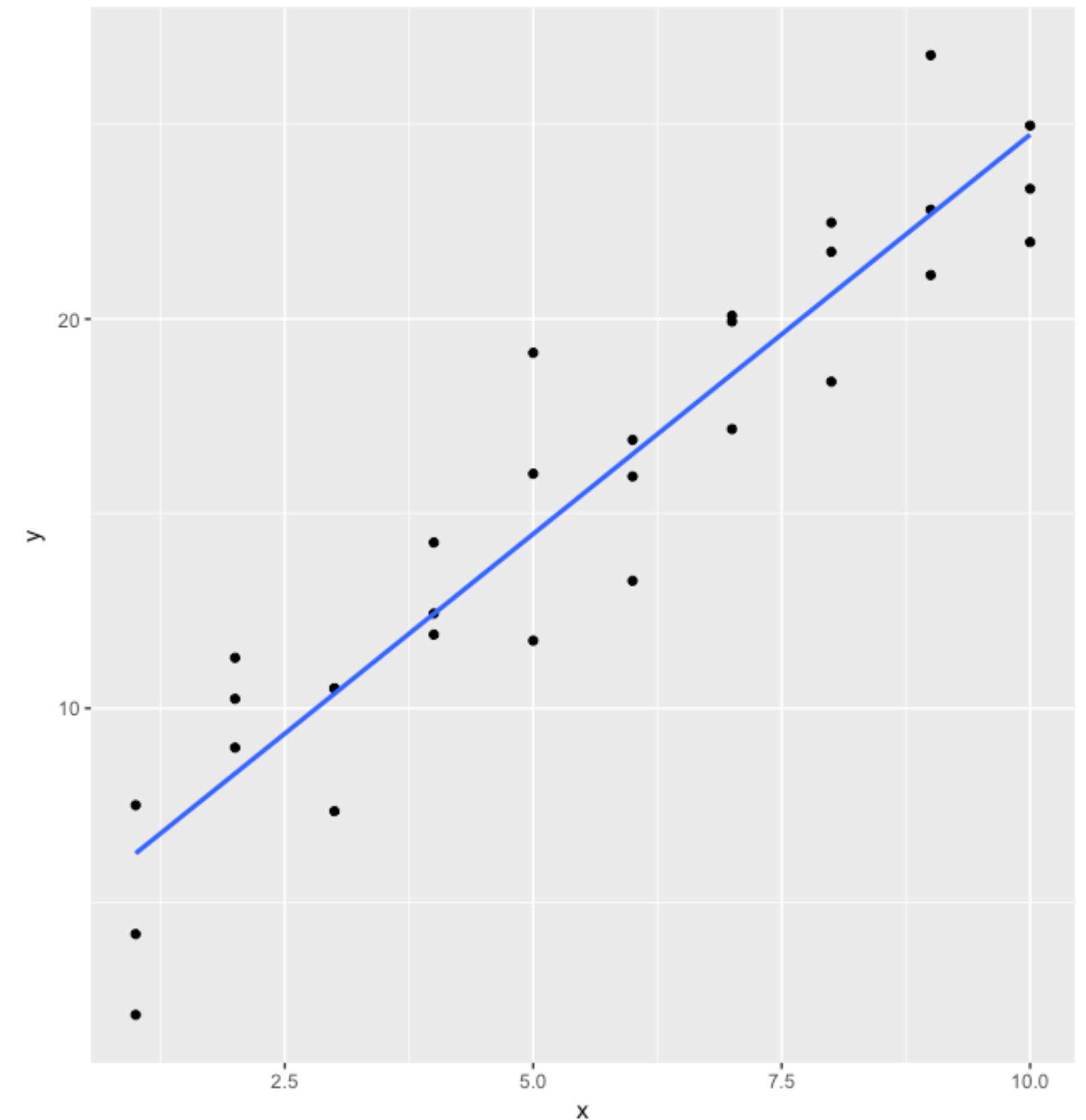
```
ggplot(sim1, aes(x, y)) +  
  geom_point()
```





# IDENTIFYING (LINEAR) RELATIONSHIPS

```
ggplot(sim1, aes(x, y)) +  
  geom_point() +  
  geom_smooth(method = "lm", se = FALSE)
```



# MEASURING (LINEAR) RELATIONSHIPS

```
cor(sim1$x, sim1$y)  
[1] 0.9405384
```

Correlation measures the linear relationship between two variables

Can be an indicator of a predictive relationship to be exploited but does not mean it should be...

nor does it imply a causal relationship

# MEASURING (LINEAR) RELATIONSHIPS

```
cor.test(sim1$x, sim1$y)
Pearson's product-moment correlation

data:  sim1$x and sim1$y
t = 14.651, df = 28, p-value = 1.173e-14
alternative hypothesis: true correlation is not
equal to 0
95 percent confidence interval:
 0.8776625 0.9715879
sample estimates:
      cor
0.9405384
```

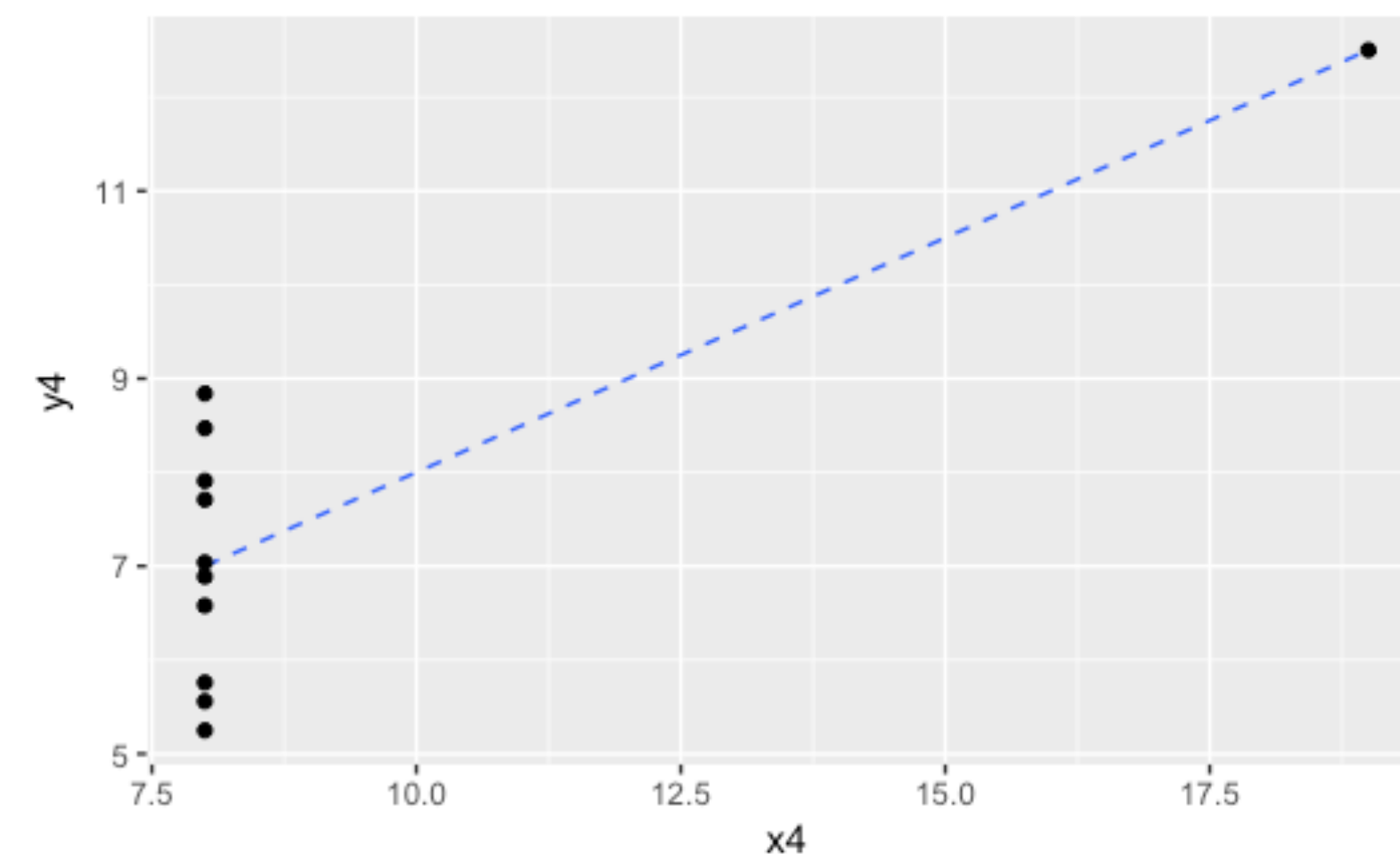
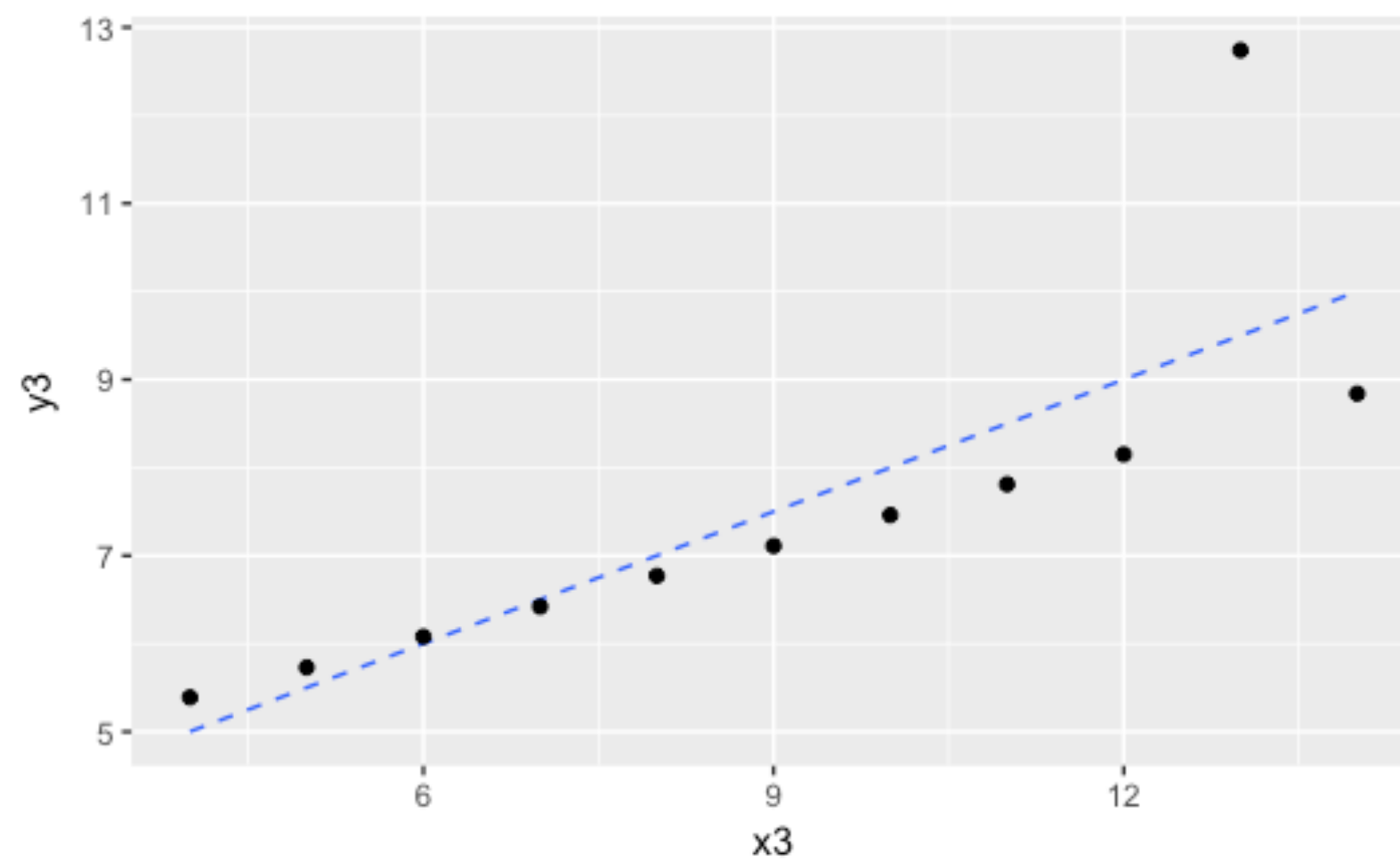
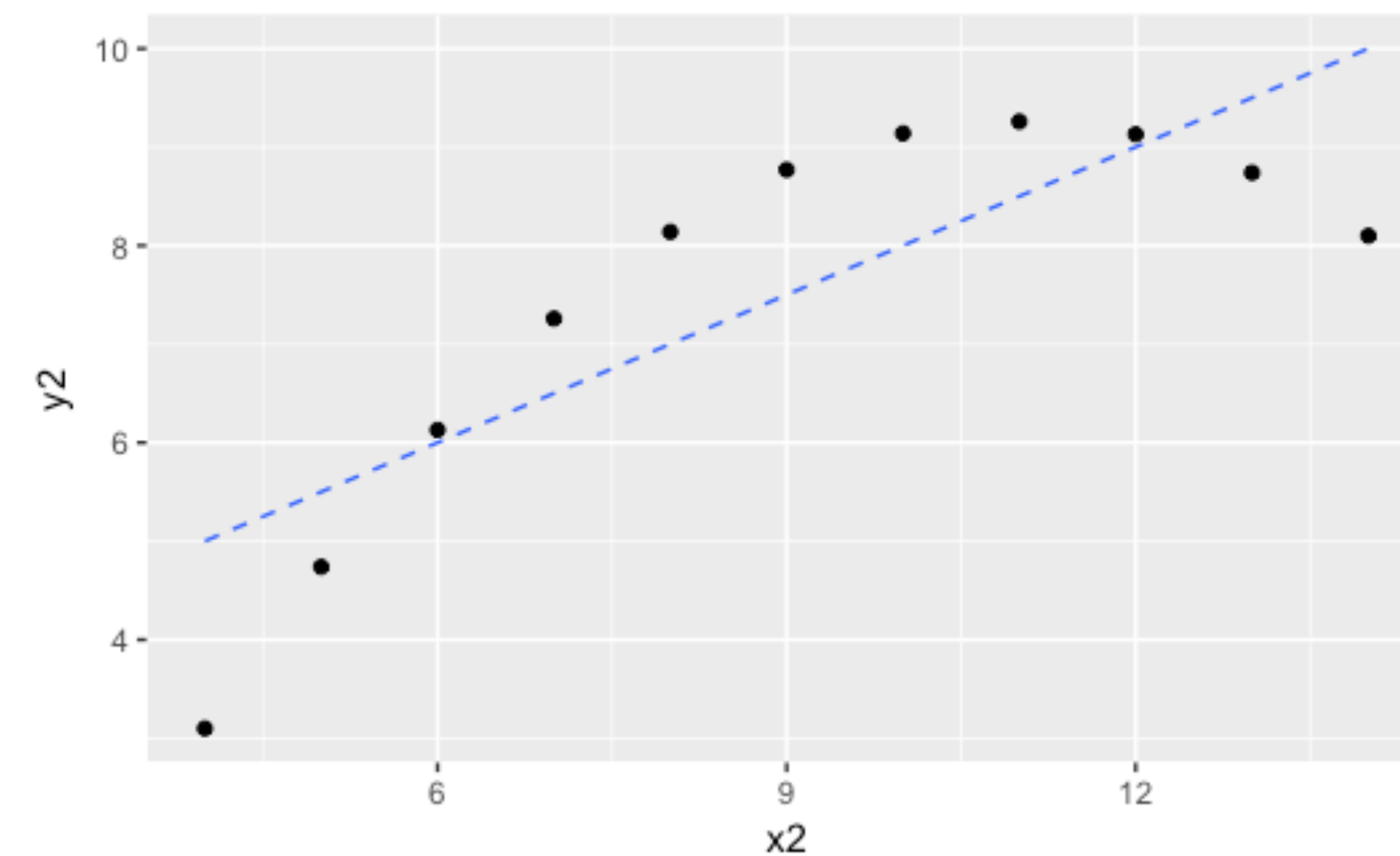
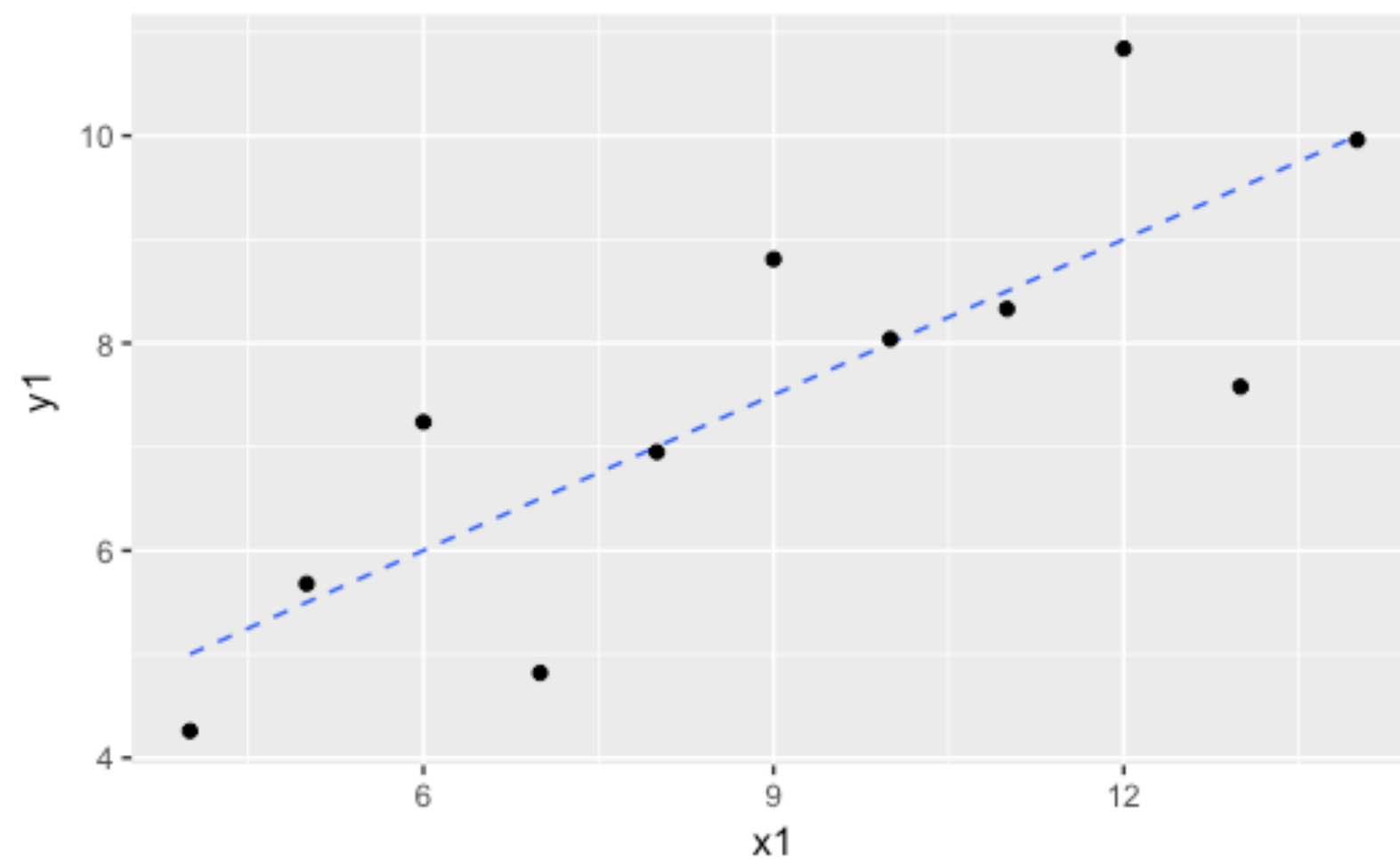
The `cor()` function provides the correlation coefficient...

but `cor.test()` provides additional insights

# WHY IS VISUALIZING RELATIONSHIPS IMPORTANT?

# WHY IS VISUALIZING RELATIONSHIPS IMPORTANT?

Anscombe's Quartet



4 different data sets

4 different relationships

All have  $\text{cor}(x, y) = .816$

***Don't interpret numbers blindly!***

# YOUR TURN!

- *Assess the linear relationship between **mpg** and **wt** in the **mtcars** data set*
- *How could you do this using **dplyr** (hint: **summarise()**)?*
- *How could you visually compare a linear vs. non-linear relationship (hint: **geom\_smooth()**)?*

# YOUR TURN!

- *Assess the linear relationship between **mpg** and **wt** in the **mtcars** data set*
- *How could you do this using **dplyr** (hint: **summarise()**)?*
- *How could you visually compare a linear vs. non-linear relationship (hint: **geom\_smooth()**)?*

# SOLUTION

```
# assess the linear relationship between mpg and wt in the mtcars data set  
cor.test(mtcars$mpg, mtcars$wt)
```

Pearson's product-moment correlation

```
data:  mtcars$mpg and mtcars$wt  
t = -9.559, df = 30, p-value = 1.294e-10  
alternative hypothesis: true correlation is not equal to 0  
95 percent confidence interval:  
 -0.9338264 -0.7440872  
sample estimates:  
      cor  
-0.8676594
```



# YOUR TURN!

- *Assess the linear relationship between **mpg** and **wt** in the **mtcars** data set*
- *How could you do this using **dplyr** (hint: **summarise()**)?*
- *How could you visually compare a linear vs. non-linear relationship (hint: **geom\_smooth()**)?*

# SOLUTION

```
# how could you do this using dplyr (think summarise())?  
mtcars %>%  
  summarise(correlation = cor(mpg, wt),  
            p_value = cor.test(mpg, wt)$p.value)  
correlation      p_value  
1  -0.8676594 1.293959e-10
```

*Why would we do it this way?*

# SOLUTION

```
# how could you do this using dplyr (think summarise())?
```

```
mtcars %>%
```

```
  group_by(cyl) %>%
```

```
  summarise(correlation = cor(mpg, wt),
```

```
            p_value = cor.test(mpg, wt)$p.value)
```

```
# A tibble: 3 × 3
```

	cyl	correlation	p_value
	<dbl>	<dbl>	<dbl>
1	4	-0.7131848	0.01374278
2	6	-0.6815498	0.09175766
3	8	-0.6503580	0.01179281

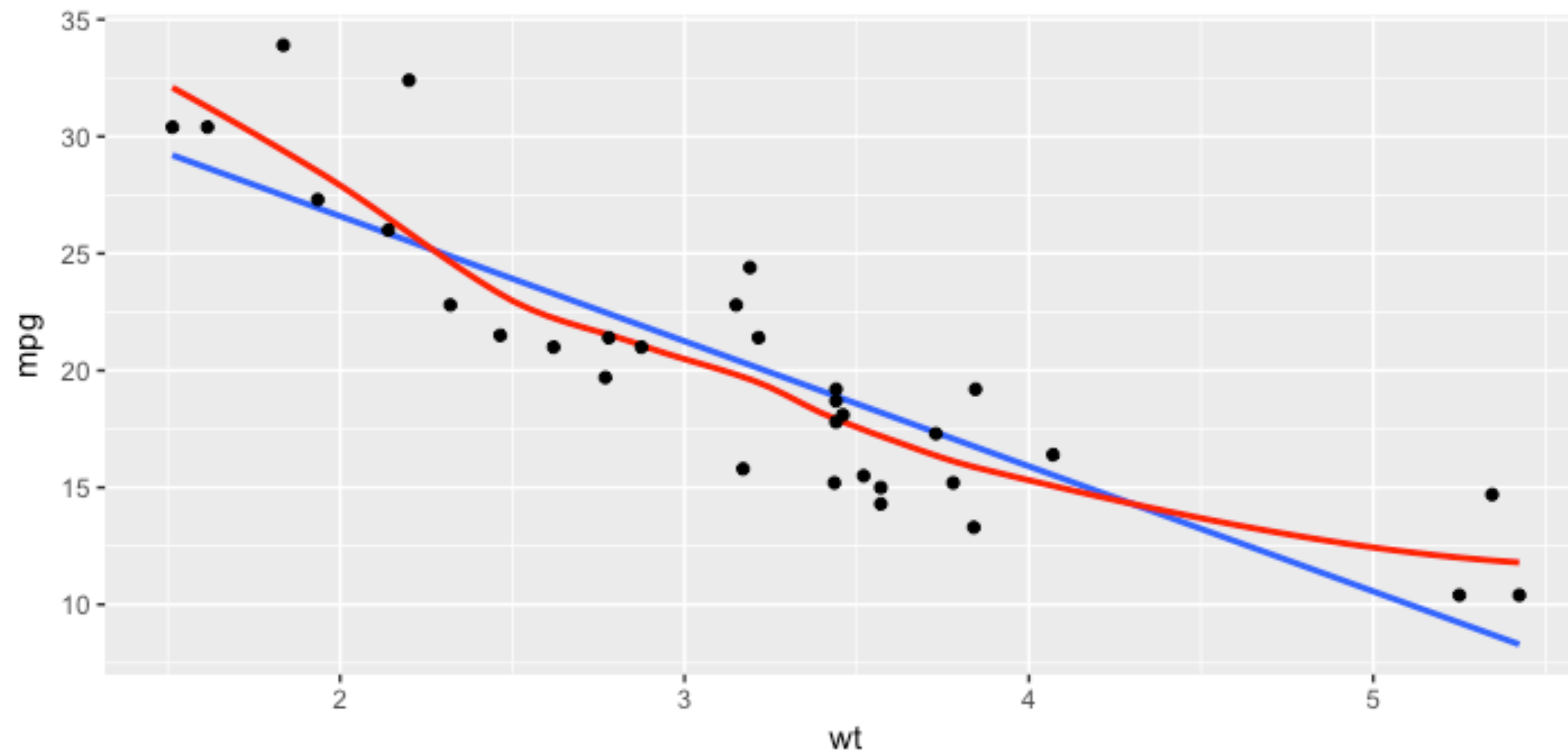
***Allows us to assess correlations at different levels***

# YOUR TURN!

- *Assess the linear relationship between `mpg` and `wt` in the `mtcars` data set*
- *How could you do this using `dplyr` (hint: `summarise()`)?*
- *How could you visually compare a linear vs. non-linear relationship (hint: `geom_smooth()`)?*

# SOLUTION

```
# how could you visually compare a linear vs non-linear relationship (hint: geom_smooth)?  
ggplot(mtcars, aes(wt, mpg)) +  
  geom_smooth(method = "lm", se = FALSE) +  
  geom_smooth(se = "FALSE", color = "red") +  
  geom_point()
```



# MULTIPLE RELATIONSHIPS

```
sim4
# A tibble: 300 × 4
      x1      x2  rep      y
  <dbl> <dbl> <int> <dbl>
1    -1 -1.0000000    1  4.24767769
2    -1 -1.0000000    2  1.20599701
3    -1 -1.0000000    3  0.35347770
4    -1 -0.7777778    1 -0.04665814
5    -1 -0.7777778    2  4.63868987
6    -1 -0.7777778    3  1.37709540
7    -1 -0.5555556    1  0.97522088
8    -1 -0.5555556    2  2.49963753
9    -1 -0.5555556    3  2.70474837
10   -1 -0.3333333    1  0.55751522
# ... with 290 more rows
```

What if we want to look for multiple relationships?

# MULTIPLE RELATIONSHIPS

```
cor(sim4)
```

	x1	x2	rep	y
x1	1.000000e+00	-8.870745e-21	0.000000000	0.38941683
x2	-8.870745e-21	1.000000e+00	0.000000000	-0.59481723
rep	0.000000e+00	0.000000e+00	1.000000000	0.02629568
y	3.894168e-01	-5.948172e-01	0.02629568	1.000000000

```
pairs(sim4)
```

What if we want to look for multiple relationships?

**cor()** will work on a full data frame (assuming all variables are numeric)

**pairs()** will provide x-y scatter plot pairs

# MULTIPLE RELATIONSHIPS

```
sim4 %>%
  gather(var, value, -y) %>%
  group_by(var) %>%
  summarise(corr = cor(y, value),
            p_value = cor.test(y, value)$p.value) %>%
  filter(p_value < 0.05)
# A tibble: 2 × 3
   var      corr      p_value
<chr>   <dbl>   <dbl>
1    x1  0.3894168 2.656616e-12
2    x2 -0.5948172 4.279573e-30
```

...but this can still be difficult to see the strongest patterns

nor do we get the p-values to tell which linear relationships are insignificant

we can get this info by leveraging **tidyr** and **dplyr** functions

***Work through this code line by line to see how it works***



# YOUR TURN!

- *Visualize relationships across all **mtcars** variables; which ones appear to have the strongest relationship to **mpg**?*
- *Quantify the correlations and find the ones that are statistically significant at  $p \leq 0.05$*

# SOLUTION

```
# visualize relationships across all mtcars variables; which ones have appear to
# have the strongest relationship to mpg?
pairs(mtcars)
```

```
# quantify the correlations and find the ones that are statistically significant
mtcars %>%
```

```
  gather(var, value, - mpg) %>%
```

```
  group_by(var) %>%
```

```
  summarise(corr = cor(mpg, value),
```

```
            p_value = cor.test(mpg, value)$p.value) %>%
```

```
  filter(p_value <= 0.05) %>%
```

```
  arrange(desc(abs(corr)))
```

```
# A tibble: 10 × 3
```

	var	corr	p_value
	<chr>	<dbl>	<dbl>
1	wt	-0.8676594	1.293959e-10
2	cyl	-0.8521620	6.112687e-10

# MANY QUESTIONS REMAIN

Many questions remain such as:

- What can we infer from these relationships (other than just strength)
- How should we treat categorical variables
- Are there interactions between variables
- How well do these variables predict an output
- and a host of others

***We can start to answer some of these questions with regression modeling***

# MODELING BASICS



# MODELS AS AN EDA TOOL

Here we are going to use models as a tool for exploratory analysis, not confirmatory analysis

In Applied Analytics with R you will learn how to use regression models for confirmatory analysis which includes:

- Partitioning your data for training, querying, and testing your model
- A more in-depth assessment of model performance
- Correct hypothesis inference

# MODEL FORMULAE IN R

response variable ~ explanatory variable(s)

Read ~ as “is modeled as a function of”

# MODEL FORMULAE IN R

response variable ~ explanatory variable(s)

Read ~ as “is modeled as a function of”

Thus, a simple linear regression of **y** on **x** would be written as:

$$y \sim x$$

# MODEL FORMULAE IN R

response variable ~ explanatory variable(s)

Read ~ as “is modeled as a function of”

Thus, a simple linear regression of y on x would be written as:

$y \sim x$

And a one-way ANOVA where sex is a two-level factor would be written as:

$y \sim \text{sex}$



# MODEL FORMULAE IN R

```
sim1$y ~ sim1$x  
str(sim1$y ~ sim1$x)
```

Execute these in R, what do they do?

# MODEL FORMULAE IN R

```
sim1$y ~ sim1$x  
str(sim1$y ~ sim1$x)  
Class 'formula' language sim1$y ~ sim1$x  
..- attr(*, ".Environment")=<environment:  
R_GlobalEnv>
```

Execute these in R, what do they do?

They simply create a formula object but we need a function to model this formula

# LINEAR MODEL FUNCTION

```
sim1_mod <- lm(y ~ x, data = sim1)
```

In R, `lm()` is the function to perform a *linear model*.

# LINEAR MODEL FUNCTION

```
sim1_mod <- lm(y ~ x, data = sim1)
```

```
sim1_mod
```

```
Call:
```

```
lm(formula = y ~ x, data = sim1)
```

```
Coefficients:
```

```
(Intercept)          x
```

```
4.221          2.052
```

In R, `lm()` is the function to perform a linear model.

This creates an object that contains a lot of results from the `lm()` model

# LINEAR MODEL FUNCTION

```
sim1_mod <- lm(y ~ x, data = sim1)
```

```
sim1_mod
```

```
summary(sim1_mod)
```

```
Call:
```

```
lm(formula = y ~ x, data = sim1)
```

```
Residuals:
```

Min	1Q	Median	3Q	Max
-4.1469	-1.5197	0.1331	1.4670	4.6516

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	4.2208	0.8688	4.858	4.09e-05	***
x	2.0515	0.1400	14.651	1.17e-14	***

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 2.203 on 28 degrees of freedom
```

In R, `lm()` is the function to perform a linear model.

This creates an object that contains a lot of results from the `lm()` model

`summary()` prints off many useful results

# LINEAR MODEL FUNCTION

```
sim1_mod <- lm(y ~ x, data = sim1)
sim1_mod
summary(sim1_mod)
str(sim1_mod)
coef(sim1_mod)
residuals(sim1_mod)
fitted.values(sim1_mod)
```

The `lm()` object is just a list so we can access many results just by knowing their names

***Try these functions***

# YOUR TURN!

1. *Fit a linear model that regresses **mpg** onto **wt** with the **mtcars** data*
2. *How does this model appear to fit (hint: summary)?*
3. *Can you access the fitted (aka predicted) values and residuals?*

# SOLUTION

```
# fit a linear model that regresses mpg onto wt with the mtcars data  
mtcars_mod1 <- lm(mpg ~ wt, data = mtcars)
```

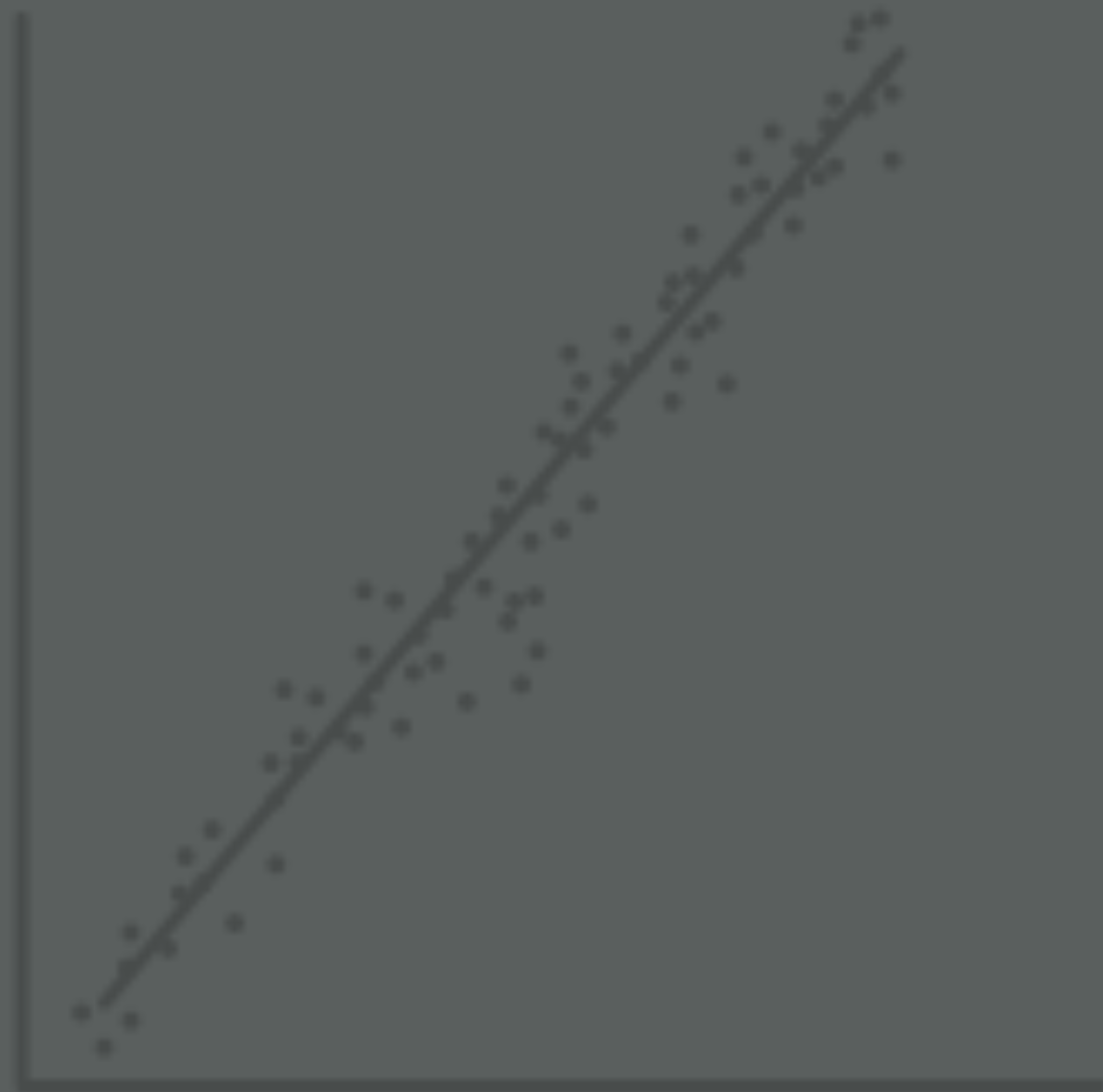
```
# how does this model appear to fit (hint: summary)?  
summary(mtcars_mod1)
```

```
# can you access the fitted values and residuals  
mtcars_mod1$residuals  
mtcars_mod1$fitted.values
```

```
residuals(mtcars_mod1)  
fitted.values(mtcars_mod1)
```



# VISUALIZING MODELS



# DO OUR RELATIONSHIPS HOLD VISUALLY?

```
sim1_mod <- lm(y ~ x, data = sim1)
sim1 %>%
  add_predictions(sim1_mod) %>%
  add_residuals(sim1_mod)
# A tibble: 30 × 4
```

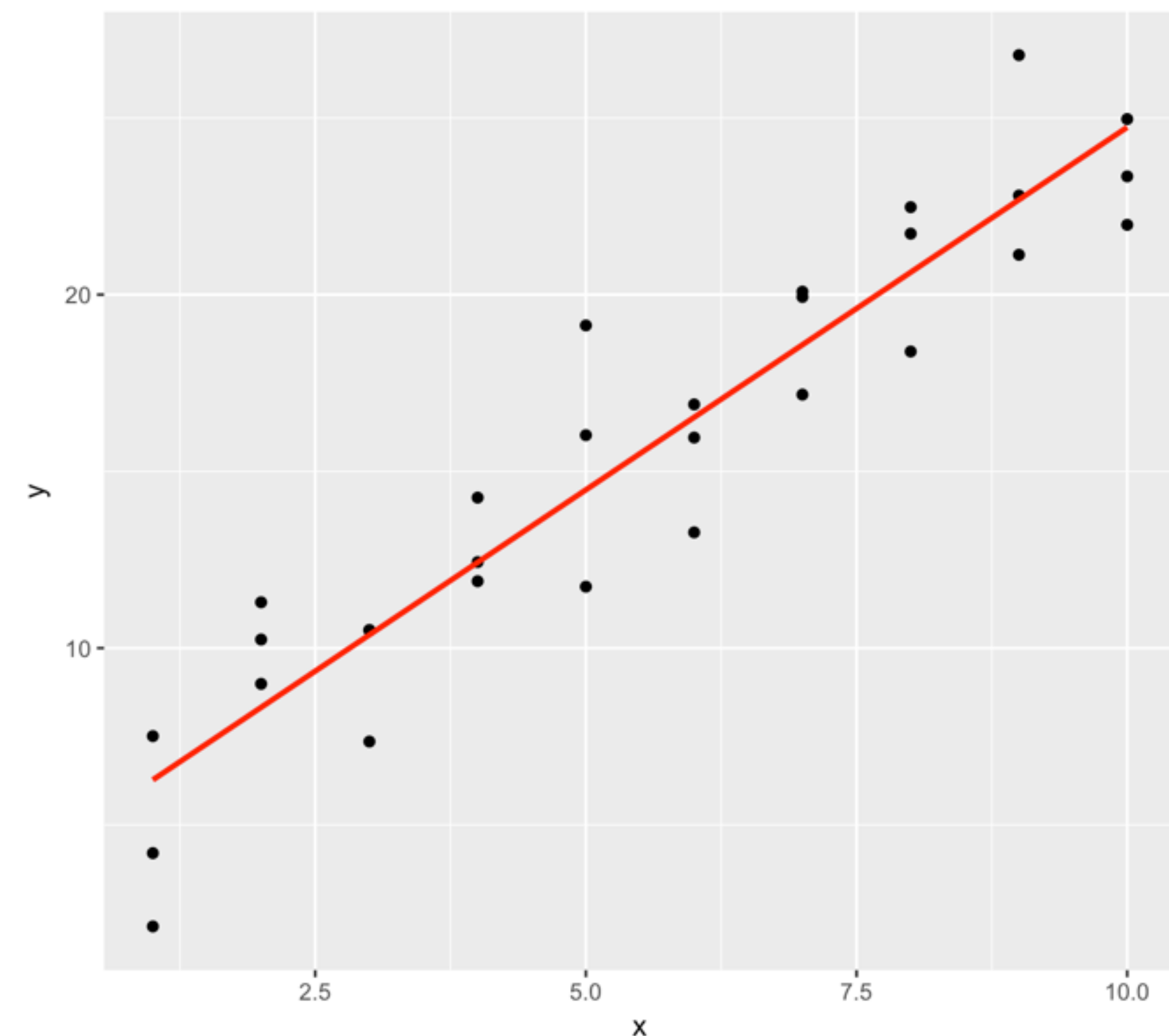
	x	y	pred	resid
	<int>	<dbl>	<dbl>	<dbl>
1	1	4.199913	6.272355	-2.072442018
2	1	7.510634	6.272355	1.238279125
3	1	2.125473	6.272355	-4.146882207
4	2	8.988857	8.323888	0.664969362
5	2	10.243105	8.323888	1.919217378
6	2	11.296823	8.323888	2.972935148
7	3	7.356365	10.375421	-3.019056466
8	3	10.505340	10.375421	0.129928252

We can add the fitted values and residuals to our original `sim1` data frame with `add_predictions()` and `add_residuals()`...

# DO OUR RELATIONSHIPS HOLD VISUALLY?

```
sim1_mod <- lm(y ~ x, data = sim1)
sim1 %>%
  add_predictions(sim1_mod) %>%
  add_residuals(sim1_mod) %>%
  ggplot(aes(x, y)) +
  geom_point() +
  geom_line(aes(y = pred),
            color = "red", size = 1)
```

...which makes it easy to pipe right into a visualization of the fitted values



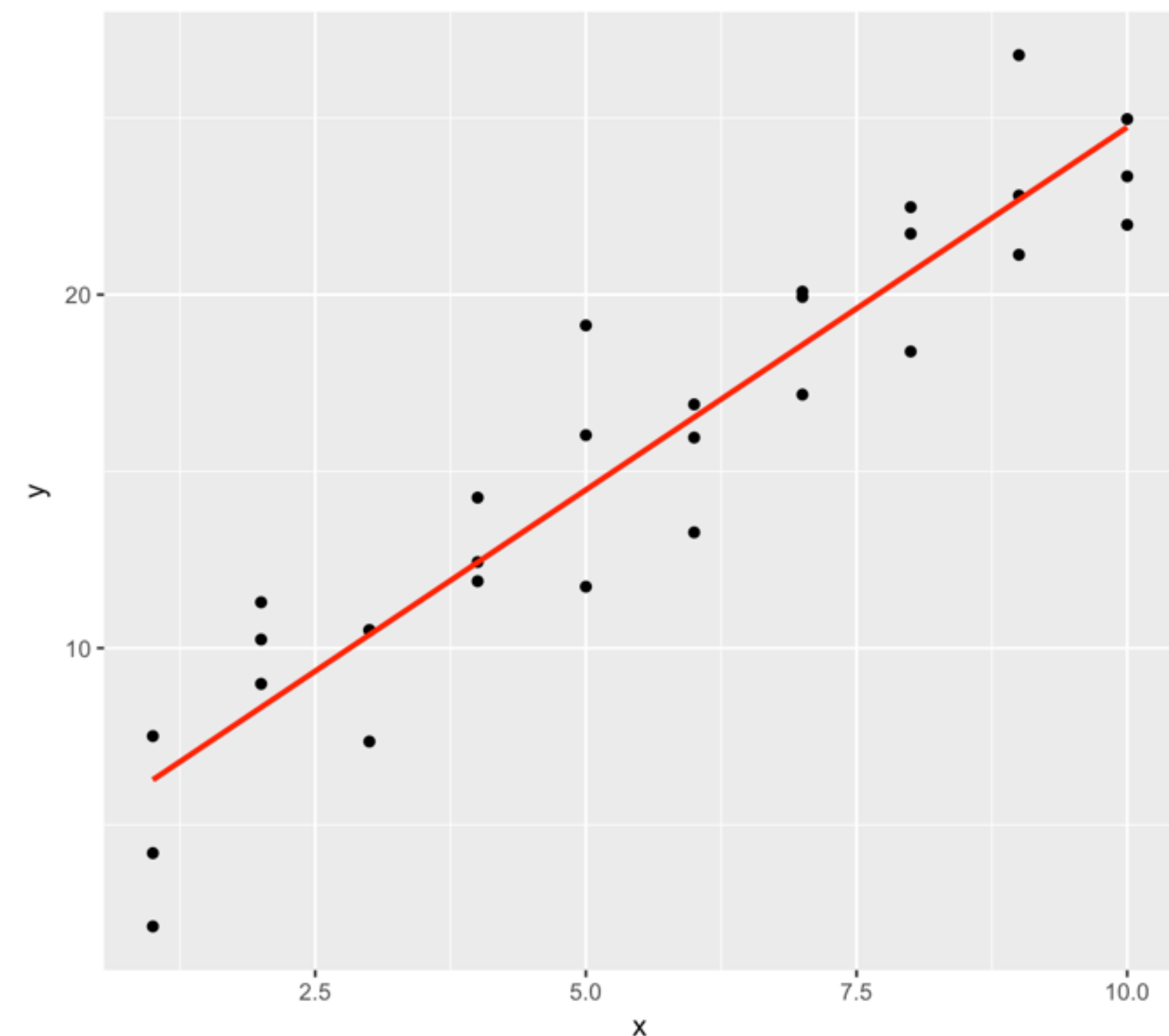
# DO OUR RELATIONSHIPS HOLD VISUALLY?

```
sim1_mod <- lm(y ~ x, data = sim1)
sim1 %>%
  add_predictions(sim1_mod) %>%
  add_residuals(sim1_mod) %>%
  ggplot(aes(x, y)) +
  geom_point() +
  geom_line(aes(y = pred),
            color = "red", size = 1)
```

# do you see how these relate????

```
ggplot(sim1, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm")
```

...which makes it easy to pipe right into a visualization of the fitted values



# DO OUR RELATIONSHIPS HOLD VISUALLY?

```
sim1_mod <- lm(y ~ x, data = sim1)
sim1 %>%
  add_residuals(sim1_mod) %>%
  ggplot(aes(resid)) +
  geom_histogram(binwidth = .5)
```

```
sim1 %>%
  add_residuals(sim1_mod) %>%
  ggplot(aes(x, resid)) +
  geom_ref_line(h = 0) +
  geom_point()
```

...we also want to plot the residuals

***Try these - what do they tell you?***

# DO OUR RELATIONSHIPS HOLD VISUALLY?

```
sim1_mod <- lm(y ~ x, data = sim1)
sim1 %>%
  add_residuals(sim1_mod) %>%
  ggplot(aes(resid)) +
  geom_histogram(binwidth = .5)
```

```
sim1 %>%
  add_residuals(sim1_mod) %>%
  ggplot(aes(x, resid)) +
  geom_ref_line(h = 0) +
  geom_point()
```

...we also want to plot the residuals

***The residuals look like random noise, suggesting the model has done a good job capturing the pattern***

# YOUR TURN!

1. *Fit a linear model that regresses **mpg** onto **wt** with the **mtcars** data*
2. *How does this model appear to fit numerically (hint: summary)?*
3. *Add the predicted and residual values to the **mtcars** data set and plot them? How do they visually fit?*

# YOUR TURN!

1. *Fit a linear model that regresses **mpg** onto **wt** with the **mtcars** data*
2. *How does this model appear to fit numerically (hint: summary)?*
3. *Add the predicted and residual values to the **mtcars** data set and plot them? How do they visually fit?*



# SOLUTION

```
# fit a linear model that regresses mpg onto wt with the mtcars data  
mtcars_mod1 <- lm(mpg ~ wt, data = mtcars)
```

# YOUR TURN!

1. *Fit a linear model that regresses **mpg** onto **wt** with the **mtcars** data*
2. *How does this model appear to fit numerically (hint: **summary**)?*
3. *Add the predicted and residual values to the **mtcars** data set and plot them? How do they visually fit?*

# SOLUTION

```
# how does this model appear to fit (hint: summary)?  
summary(mtcars_mod1)
```

Call:

```
lm(formula = mpg ~ wt, data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-4.5432	-2.3647	-0.1252	1.4096	6.8727

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	37.2851	1.8776	19.858	< 2e-16 ***
wt	-5.3445	0.5591	-9.559	1.29e-10 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.046 on 30 degrees of freedom

Multiple R-squared: 0.7528, Adjusted R-squared: 0.7446

F-statistic: 91.38 on 1 and 30 DF, p-value: 1.294e-10

# YOUR TURN!

1. *Fit a linear model that regresses **mpg** onto **wt** with the **mtcars** data*
2. *How does this model appear to fit numerically (hint: *summary*)?*
3. *Add the predicted and residual values to the **mtcars** data set and plot them? How do they visually fit?*

# SOLUTION

```
# add the predictions and residuals to the mtcars data set and plot them
```

```
mtcars %>%
```

```
  select(mpg, wt) %>%
```

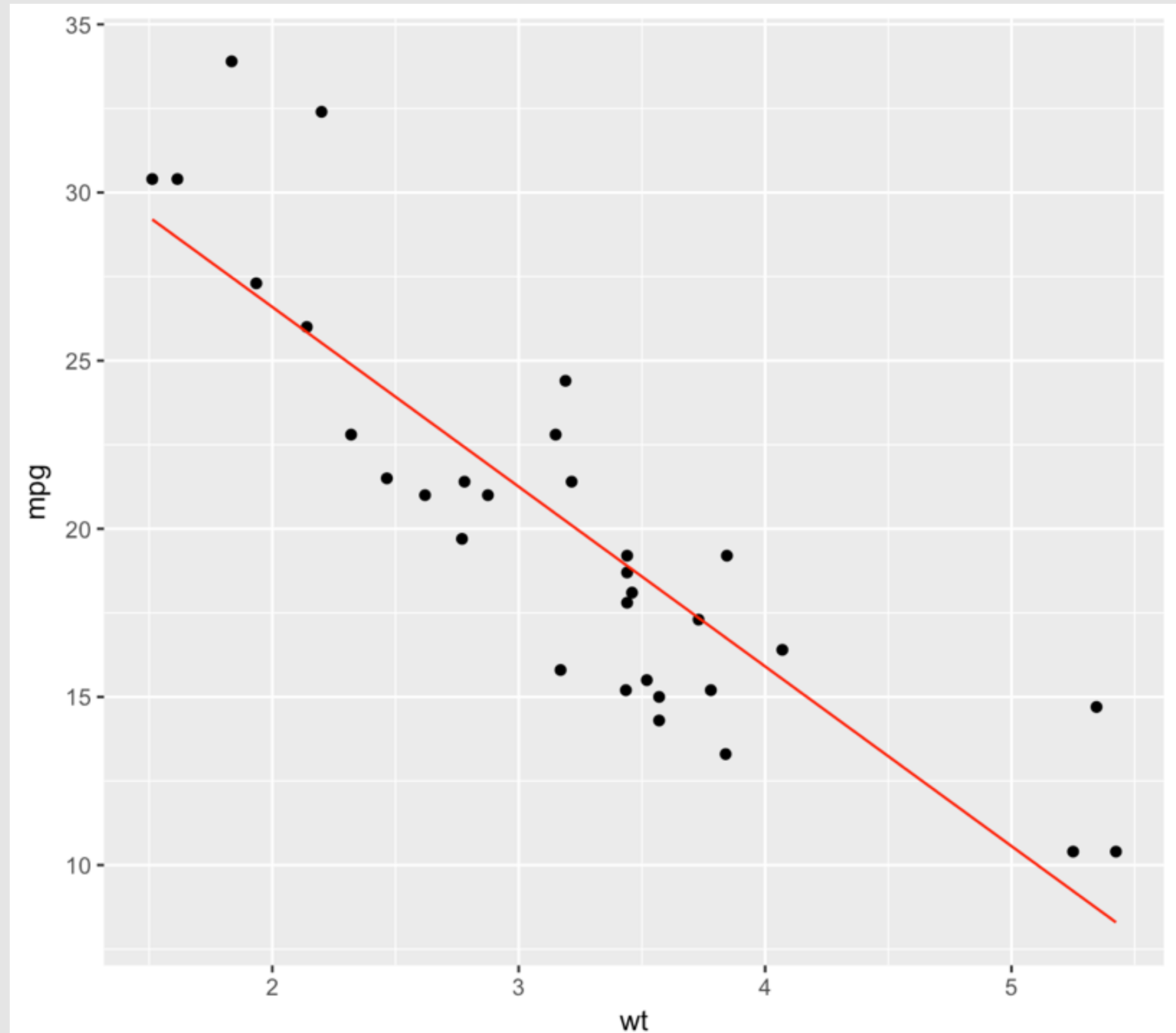
```
  add_predictions(mtcars_mod1) %>%
```

```
  add_residuals(mtcars_mod1) %>%
```

```
  ggplot(aes(wt, mpg)) +
```

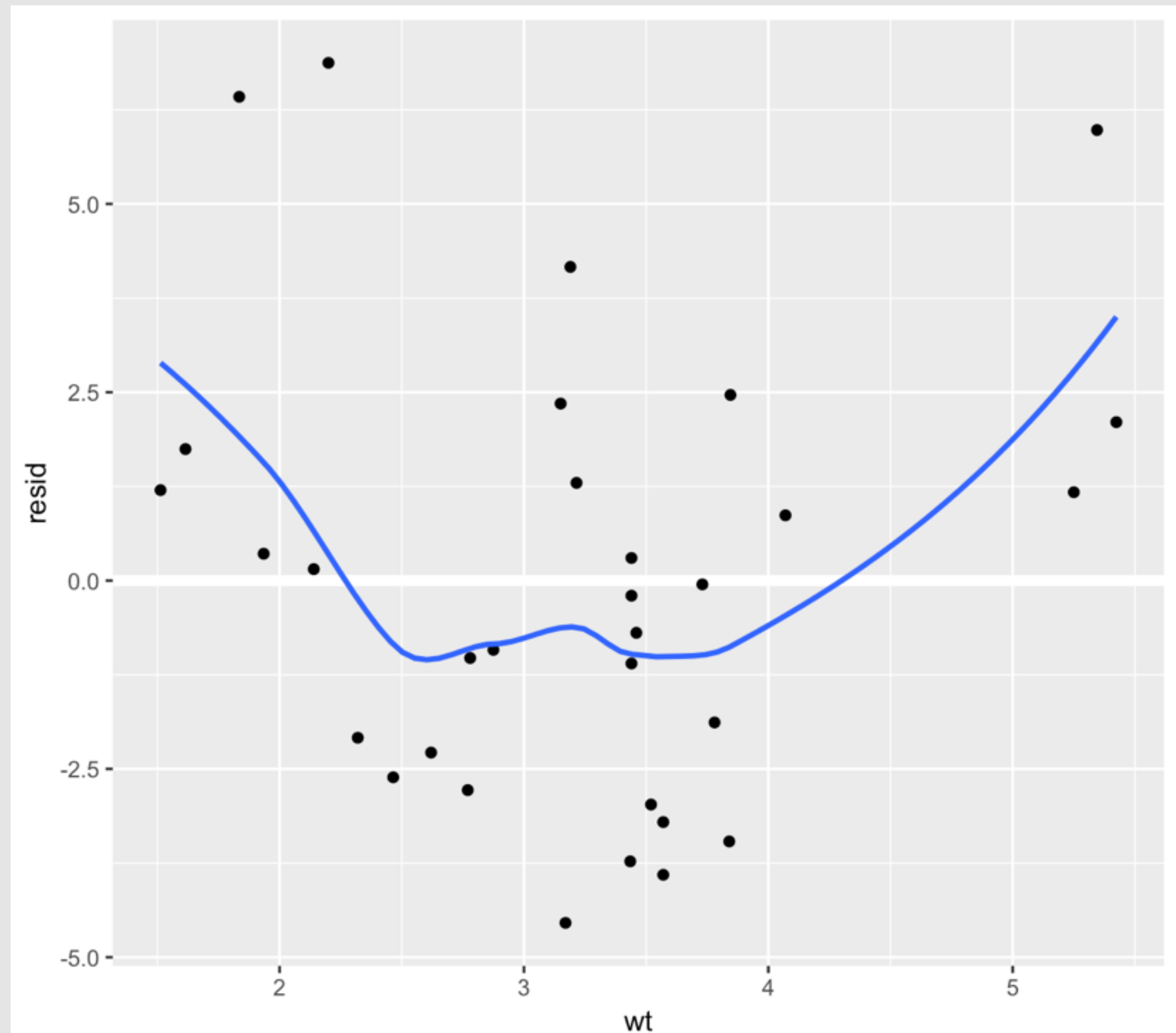
```
  geom_point() +
```

```
  geom_line(aes(y = pred), color = "red")
```

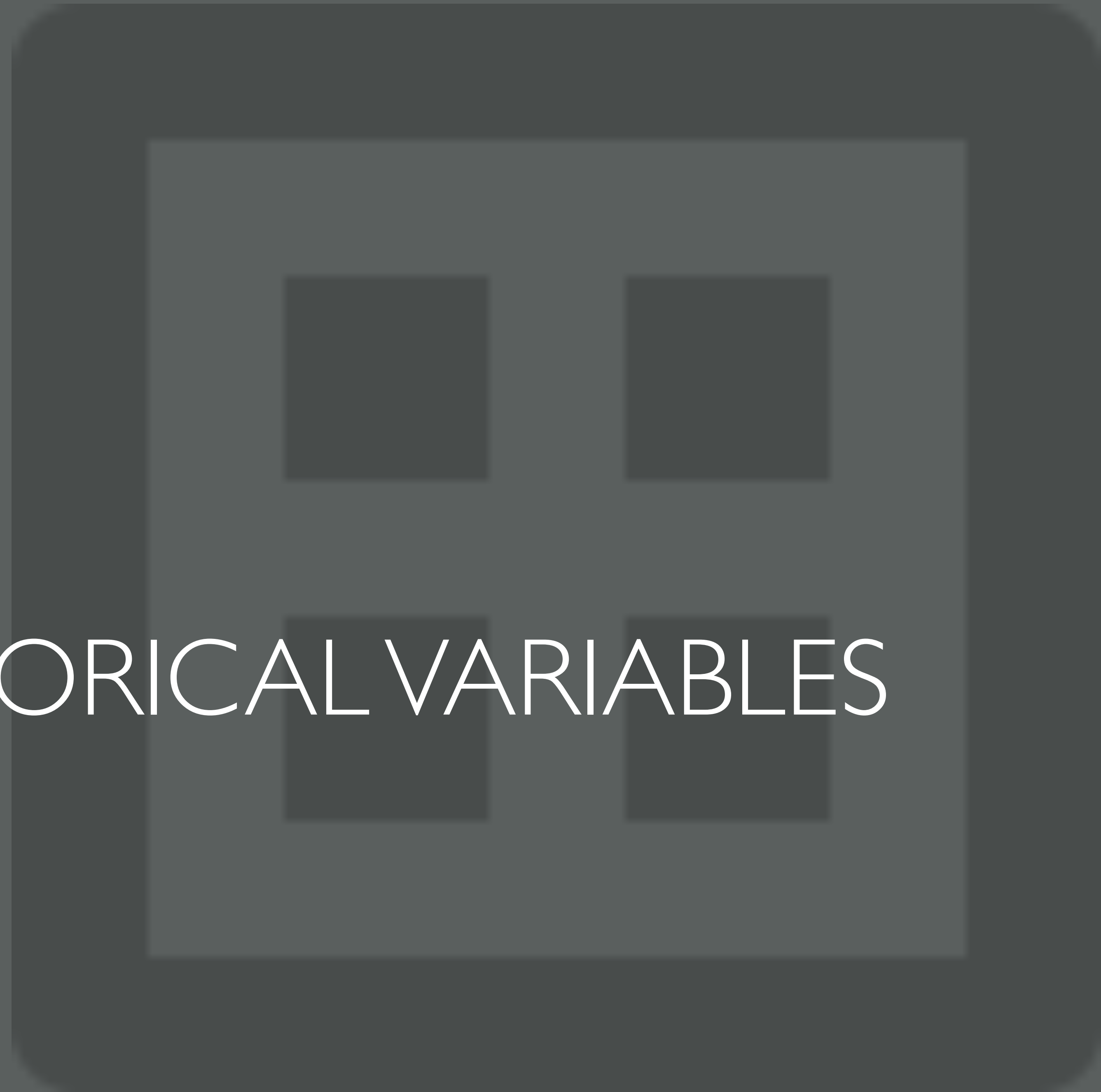


# SOLUTION

```
# add the predictions and residuals to the mtcars data set and plot them
mtcars %>%
  select(mpg, wt) %>%
  add_predictions(mtcars_mod1) %>%
  add_residuals(mtcars_mod1) %>%
  ggplot(aes(wt, resid)) +
  geom_ref_line(h = 0) +
  geom_point() +
  geom_smooth(se = FALSE)
```



# MODELING CATEGORICAL VARIABLES



# CATEGORICAL PREDICTORS

Regression is straight forward when the predictor is continuous, but things get a bit more complicated when the predictor is categorical.

```
sim2
# A tibble: 40 × 2
      x      y
  <chr> <dbl>
1     a 1.9353632
2     a 1.1764886
3     a 1.2436855
4     a 2.6235489
5     a 1.1120381
6     a 0.8660030
7     a -0.9100875
8     a 0.7207628
```



# CATEGORICAL PREDICTORS

Regression is straight forward when the predictor is continuous, but things get a bit more complicated when the predictor is categorical.

```
sim2_mod <- lm(y ~ x, data = sim2)
```

Luckily, the syntax for categorical variables does not change as R takes care of the dirty work...

# CATEGORICAL PREDICTORS

Regression is straight forward when the predictor is continuous, but things get a bit more complicated when the predictor is categorical.

```
sim2_mod <- lm(y ~ x, data = sim2)
```

```
summary(sim2_mod)
```

Call:

```
lm(formula = y ~ x, data = sim2)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.40131	-0.43996	-0.05776	0.49066	2.63938

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	1.1522	0.3475	3.316	0.00209	**
xb	6.9639	0.4914	14.171	2.68e-16	***
xc	4.9750	0.4914	10.124	4.47e-12	***
xd	0.7588	0.4914	1.544	0.13131	

Luckily, the syntax for categorical variables does not change as R takes care of the dirty work...

but how we interpret the results changes slightly

*Who can interpret these results?*

# CATEGORICAL PREDICTORS

Regression is straight forward when the predictor is continuous, but things get a bit more complicated when the predictor is categorical.

```
sim2_mod <- lm(y ~ x, data = sim2)
```

```
summary(sim2_mod)
```

Call:

```
lm(formula = y ~ x, data = sim2)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-2.40131	-0.43996	-0.05776	0.49066	2.63938

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	1.1522	0.3475	3.316	0.00209	**
xb	6.9639	0.4914	14.171	2.68e-16	***
xc	4.9750	0.4914	10.124	4.47e-12	***
xd	0.7588	0.4914	1.544	0.13131	

```
sim2_mod <- lm(y ~ x, data = sim2)
```

```
sim2 %>%
```

```
  data_grid(x) %>%
```

```
  add_predictions(sim2_mod)
```

```
# A tibble: 4 × 2
```

	x	pred
	<chr>	<dbl>

1	a	1.152166
---	---	----------

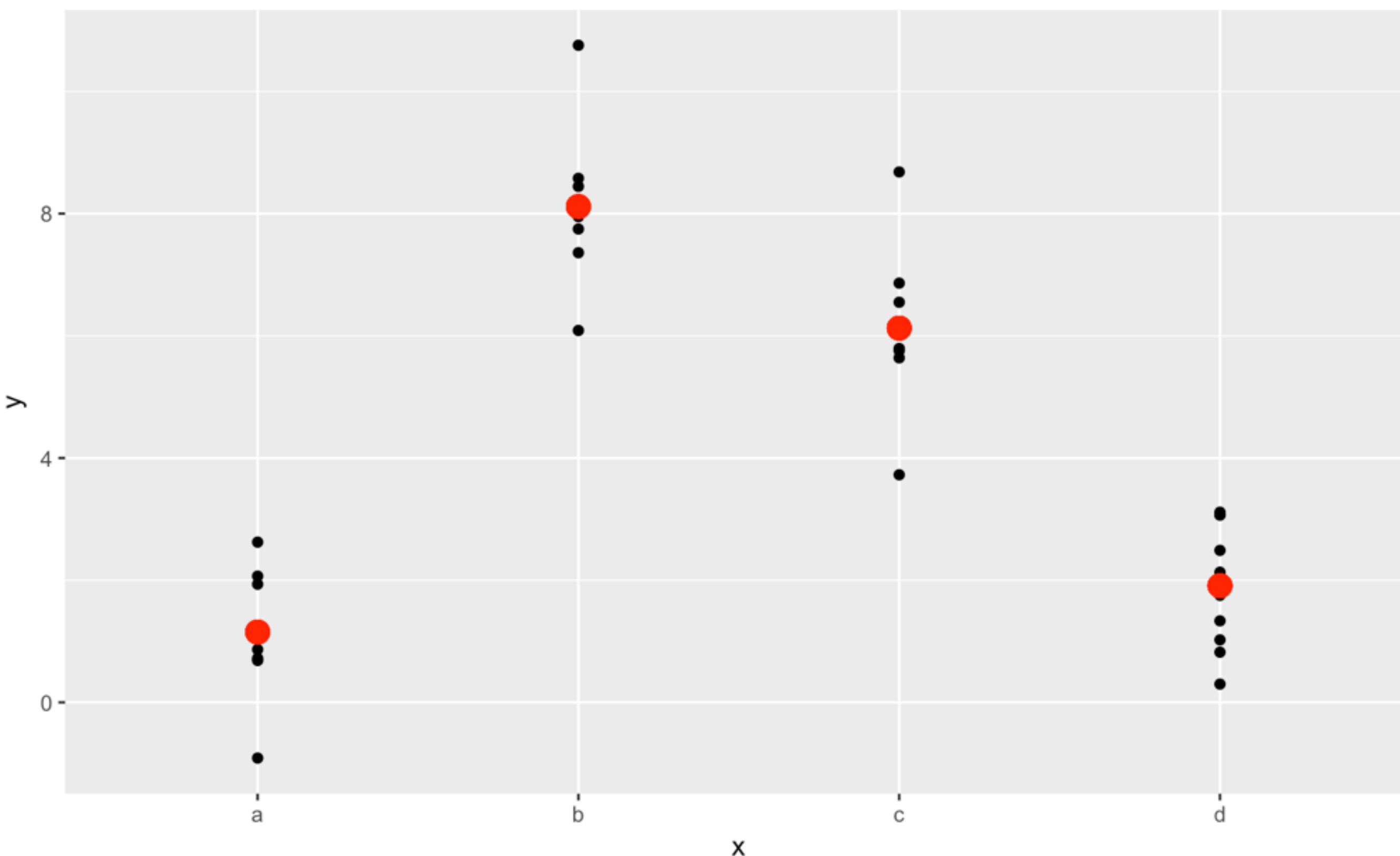
2	b	8.116039
---	---	----------

3	c	6.127191
---	---	----------

4	d	1.910981
---	---	----------

# CATEGORICAL PREDICTORS

Regression is straight forward when the predictor is continuous, but things get a bit more complicated when the predictor is categorical.



```
sim2_mod <- lm(y ~ x, data = sim2)
sim2 %>%
  data_grid(x) %>%
  add_predictions(sim2_mod)
# A tibble: 4 × 2
      x      pred
<chr> <dbl>
1    a  1.152166
2    b  8.116039
3    c  6.127191
4    d  1.910981
```

# CATEGORICAL PREDICTORS

Regression is straight forward when the predictor is continuous, but things get a bit more complicated when the predictor is categorical.

What if we want to change the order of the categorical variables?

# CATEGORICAL PREDICTORS

Regression is straight forward when the predictor is continuous, but things get a bit more complicated when the predictor is categorical.

```
sim2b <- sim2 %>%  
  mutate(x = as.factor(x),  
         x = relevel(x, ref = "b"))  
  
sim2b_mod <- lm(y ~ x, data = sim2b)  
summary(sim2b_mod)  
Coefficients:  
                Estimate Std. Error t value Pr(>|t|)  
(Intercept)    8.1160      0.3475  23.356  < 2e-16 ***  
xa             -6.9639      0.4914 -14.171  2.68e-16 ***  
xc             -1.9888      0.4914  -4.047  0.000263 ***  
xd             -6.2051      0.4914 -12.627  8.67e-15 ***
```

What if we want to change the order of the categorical variables?

This is where understanding factors is handy.

Use `relevel()` to establish a new reference level.

# YOUR TURN!

1. *Fit a model that regresses **mpg** onto **cyl** with the **mtcars** data. Make sure **cyl** is being used as a categorical variable and not a continuous.*
2. *Can you plot the predictions?*
3. *How about plotting the residuals?*

# YOUR TURN!

1. *Fit a model that regresses **mpg** onto **cyl** with the **mtcars** data. Make sure **cyl** is being used as a categorical variable and not a continuous.*
2. *Can you plot the predictions?*
3. *How about plotting the residuals?*



# SOLUTION

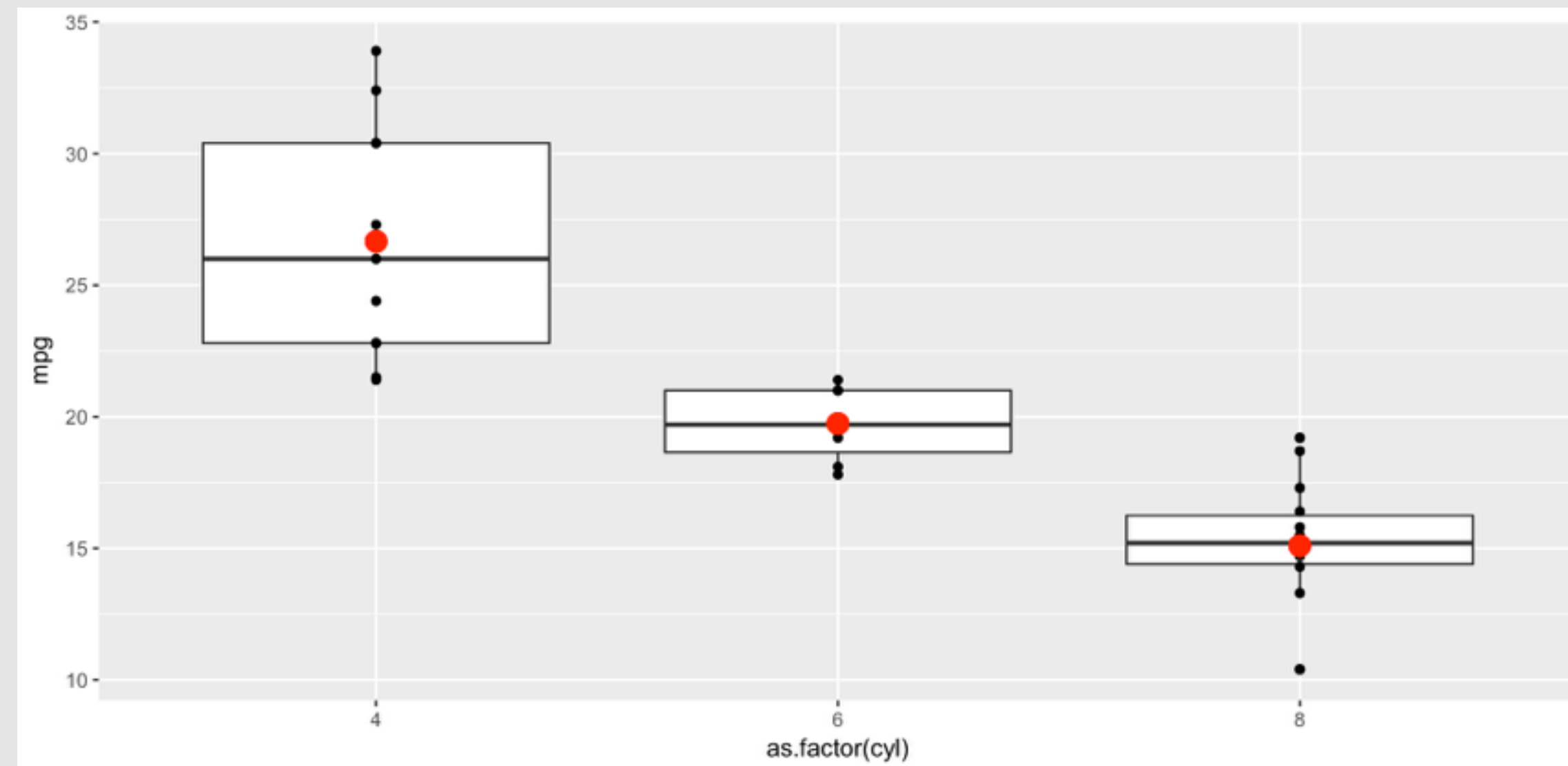
```
# model mpg ~ cyl with cyl being categorical  
mtcars_mod2 <- lm(mpg ~ as.factor(cyl), data = mtcars)
```

# YOUR TURN!

1. *Fit a model that regresses **mpg** onto **cyl** with the **mtcars** data. Make sure **cyl** is being used as a categorical variable and not a continuous.*
2. *Can you plot the predictions?*
3. *How about plotting the residuals?*

# SOLUTION

```
# plot the predictions
mtcars %>%
  add_predictions(mtcars_mod2) %>%
  ggplot(aes(as.factor(cyl), mpg)) +
  geom_boxplot() +
  geom_point() +
  geom_point(aes(y = pred), color = "red", size = 4)
```

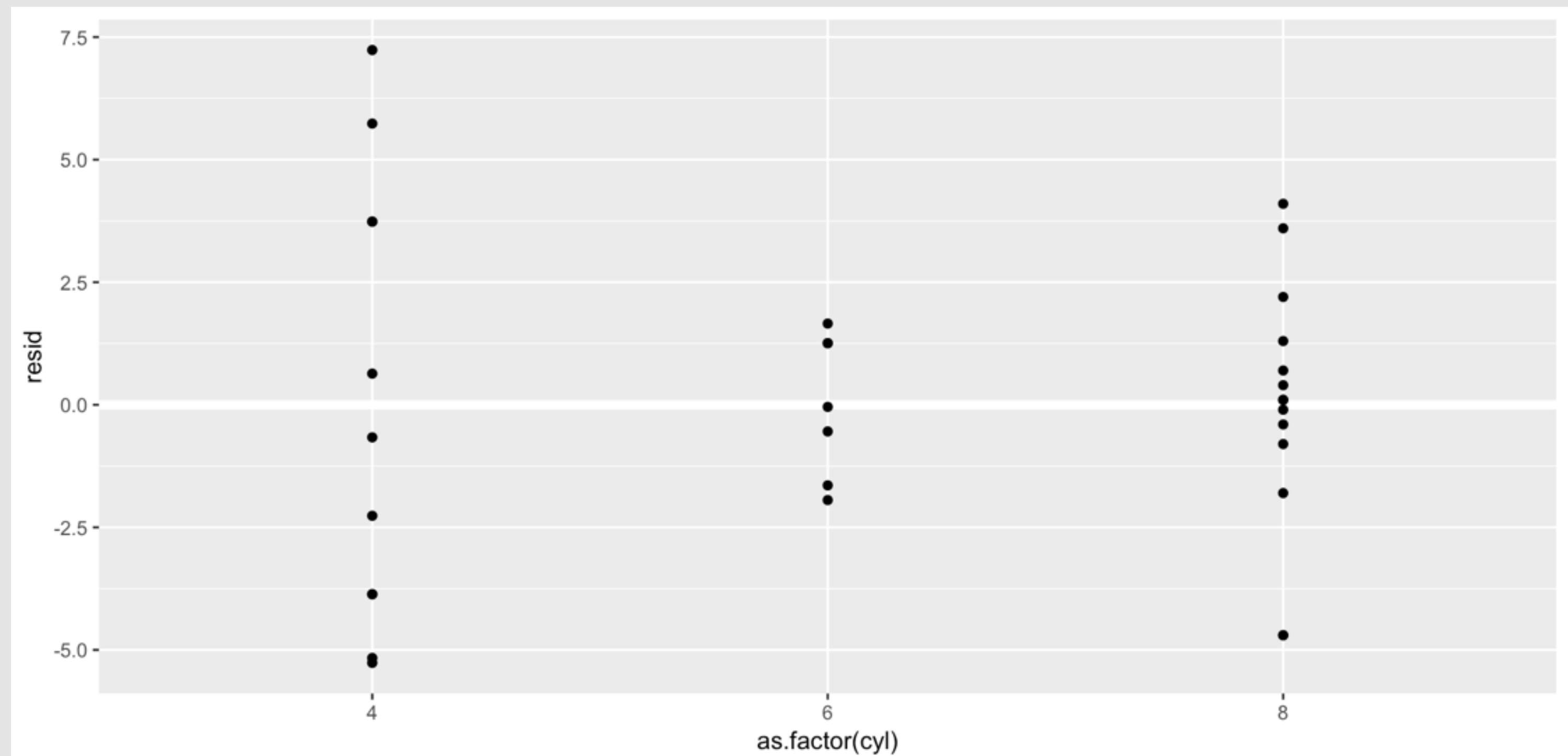


# YOUR TURN!

1. *Fit a model that regresses **mpg** onto **cyl** with the **mtcars** data. Make sure **cyl** is being used as a categorical variable and not a continuous.*
2. *Can you plot the predictions?*
3. *How about plotting the residuals?*

# SOLUTION

```
# plot the residuals  
mtcars %>%  
  add_residuals(mtcars_mod2) %>%  
  ggplot(aes(as.factor(cyl))) +  
  geom_ref_line(h = 0) +  
  geom_point(aes(y = resid))
```



# INTERACTIONS BETWEEN CONTINUOUS AND CATEGORICAL

# INTERACTIONS

What happens when you combine a continuous and a categorical variable? `sim3` contains a categorical predictor and a continuous predictor.

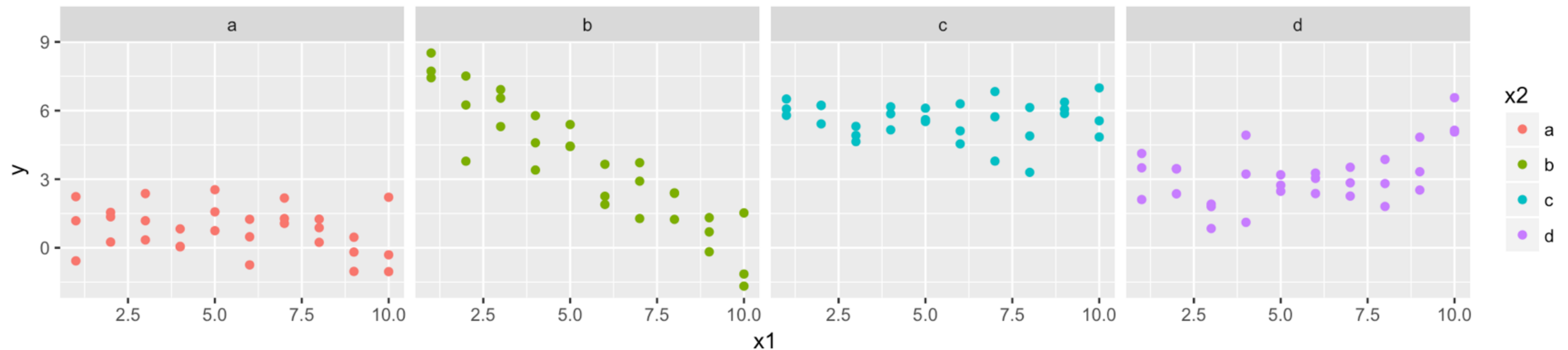
```
sim3
# A tibble: 120 × 5
      x1      x2 rep      y      sd
  <int> <fctr> <int> <dbl> <dbl>
1     1     a     1 -0.5707363     2
2     1     a     2  1.1841503     2
3     1     a     3  2.2373204     2
4     1     b     1  7.4366963     2
5     1     b     2  8.5182934     2
6     1     b     3  7.7239098     2
7     1     c     1  6.5067480     2
8     1     c     2  5.7900643     2
```

# INTERACTIONS

What happens when you combine a continuous and a categorical variable? **sim3** contains a categorical predictor and a continuous predictor.

```
ggplot(sim3, aes(x1, y, color = x2)) +  
  geom_point() +  
  facet_wrap(~ x2)
```

There clearly appears to be different relationships across the categorical variable levels





# INTERACTIONS

```
mod1 <- lm(y ~ x1 + x2, data = sim3)
mod2 <- lm(y ~ x1 * x2, data = sim3)
```

We can model this two different ways:

- **+** will model the variables independent of one another

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

# INTERACTIONS

```
mod1 <- lm(y ~ x1 + x2, data = sim3)
mod2 <- lm(y ~ x1 * x2, data = sim3)
```

We can model this two different ways:

- `+` will model the variables independent of one another
- `*` will model the variables with interactions

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$

# INTERACTIONS

```
mod1 <- lm(y ~ x1 + x2, data = sim3)
mod2 <- lm(y ~ x1 * x2, data = sim3)

# run summaries to compare the difference
summary(mod1)
summary(mod2)
```

We can model this two different ways:

- `+` will model the variables independent of one another
- `*` will model the variables with interactions

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$

# INTERACTIONS

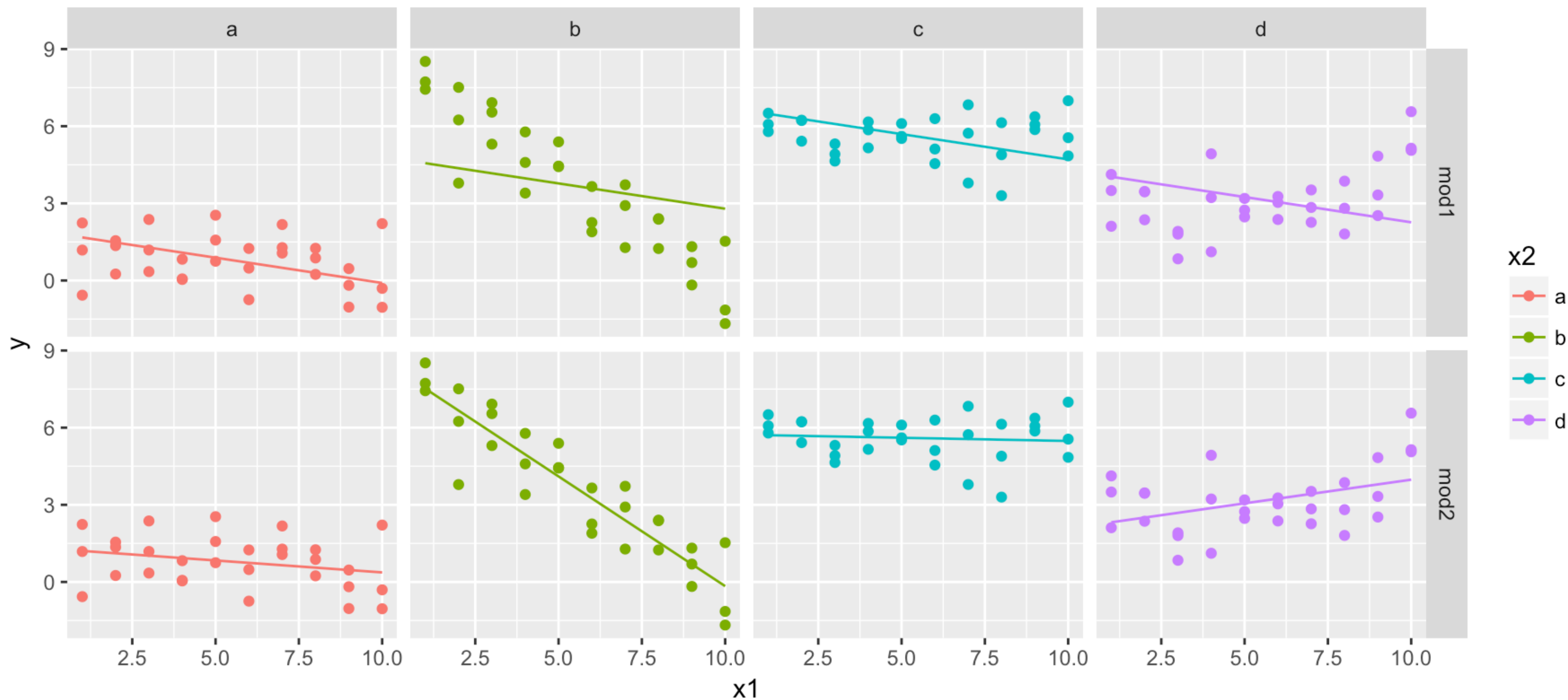
```
sim3 %>%  
  gather_predictions(mod1, mod2) %>%  
  ggplot(aes(x1, y, color = x2)) +  
  geom_point() +  
  geom_line(aes(y = pred)) +  
  facet_grid(model ~ x2)
```

We can compare the difference in these models using a similar process as before...

Except here we use **gather\_predictions** to incorporate results from both models

***Run this code line-by-line to see what is happening? What do the results suggest?***

# INTERACTIONS



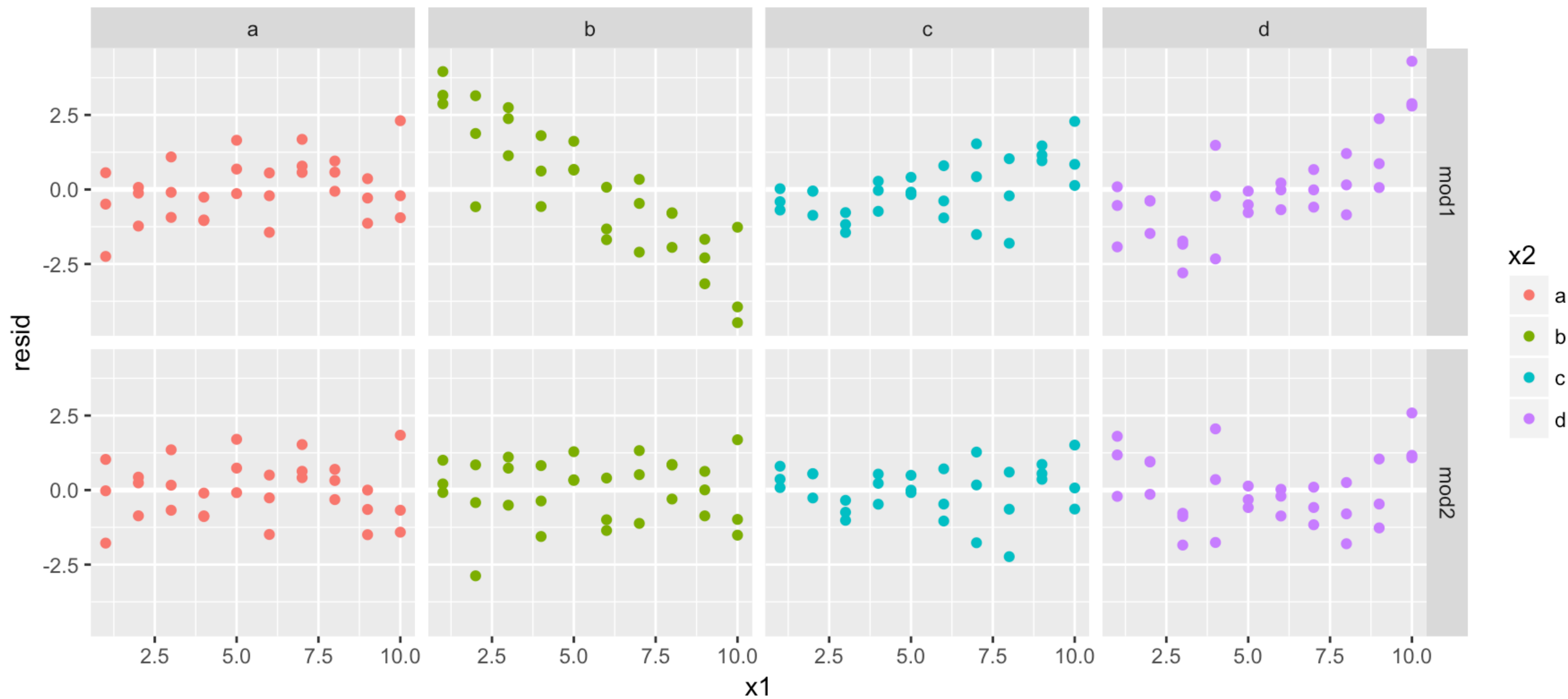
# INTERACTIONS

```
sim3 %>%  
  gather_residuals(mod1, mod2) %>%  
  ggplot(aes(x1, resid, color = x2)) +  
  geom_ref_line(h = 0, size = 1) +  
  geom_point() +  
  facet_grid(model ~ x2)
```

Using a similar process with **gather\_residuals** will allow you to compare residuals in a similar manner

***Run this code line-by-line to see what is happening? What do the results suggest?***

# INTERACTIONS



# YOUR TURN!

*1. compute the following two models for the mtcars data*

```
mtcars_mod3 <- lm(mpg ~ wt + as.factor(cyl), data = mtcars)
```

```
mtcars_mod4 <- lm(mpg ~ wt * as.factor(cyl), data = mtcars)
```

*2. Compare the summaries*

*3. Compare the predicted values*

*4. Compare the residuals*

*5. What are your thoughts?*



# YOUR TURN!

*1. compute the following two models for the mtcars data*

```
mtcars_mod3 <- lm(mpg ~ wt + as.factor(cyl), data = mtcars)
```

```
mtcars_mod4 <- lm(mpg ~ wt * as.factor(cyl), data = mtcars)
```

*2. Compare the summaries*

*3. Compare the predicted values*

*4. Compare the residuals*

*5. What are your thoughts?*

# SOLUTION

```
# compare the following two models for the mtcars data
mtcars_mod3 <- lm(mpg ~ wt + as.factor(cyl), data = mtcars)
mtcars_mod4 <- lm(mpg ~ wt * as.factor(cyl), data = mtcars)

# compare the summaries
summary(mtcars_mod3)
summary(mtcars_mod4)
```

# YOUR TURN!

*1. compute the following two models for the mtcars data*

```
mtcars_mod3 <- lm(mpg ~ wt + as.factor(cyl), data = mtcars)
```

```
mtcars_mod4 <- lm(mpg ~ wt * as.factor(cyl), data = mtcars)
```

*2. Compare the summaries*

*3. Compare the predicted values*

*4. Compare the residuals*

*5. What are your thoughts?*

# SOLUTION

```
# compare predicted values
mtcars %>%
  mutate(cyl = as.factor(cyl)) %>%
  select(mpg, wt, cyl) %>%
  gather_predictions(mtcars_mod3, mtcars_mod4) %>%
  ggplot(aes(wt, mpg, color = cyl)) +
  geom_point() +
  geom_line(aes(y = pred, color = cyl)) +
  facet_grid(model ~ cyl, scales = "free")
```

# YOUR TURN!

*1. compute the following two models for the mtcars data*

```
mtcars_mod3 <- lm(mpg ~ wt + as.factor(cyl), data = mtcars)
```

```
mtcars_mod4 <- lm(mpg ~ wt * as.factor(cyl), data = mtcars)
```

*2. Compare the summaries*

*3. Compare the predicted values*

*4. Compare the residuals*

*5. What are your thoughts?*

# SOLUTION

```
# compare residuals
mtcars %>%
  mutate(cyl = as.factor(cyl)) %>%
  select(mpg, wt, cyl) %>%
  gather_residuals(mtcars_mod3, mtcars_mod4) %>%
  ggplot(aes(wt, resid, color = cyl)) +
  geom_ref_line(h = 0) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  facet_grid(model ~ cyl, scales = "free")
```

# YOUR TURN!

*1. compute the following two models for the mtcars data*

```
mtcars_mod3 <- lm(mpg ~ wt + as.factor(cyl), data = mtcars)
```

```
mtcars_mod4 <- lm(mpg ~ wt * as.factor(cyl), data = mtcars)
```

*2. Compare the summaries*

*3. Compare the predicted values*

*4. Compare the residuals*

*5. What are your thoughts?*

$$\hat{Y} = \alpha + \beta X$$

MODEL SPECIFICATION



# MODEL SPECIFICATION

```
model_matrix(sim1, y ~ x)
model_matrix(sim2, y ~ x)
model_matrix(sim3, y ~ x1 + x2)
model_matrix(sim3, y ~ x1 * x2)
```

As you are learning to specify models in R you may get confused, or forget, just how to interpret the model your specifying.

You can easily assess your model using `model_matrix`

Run these functions and  
observe the difference

# MODEL SPECIFICATION

```
model_matrix(sim1, y ~ x)
model_matrix(sim2, y ~ x)
model_matrix(sim3, y ~ x1 + x2)
model_matrix(sim3, y ~ x1 * x2)
# A tibble: 120 × 5
  `(Intercept)`    x1    x2b    x2c    x2d
      <dbl> <dbl> <dbl> <dbl> <dbl>
1             1     1     0     0     0
2             1     1     0     0     0
3             1     1     0     0     0
4             1     1     1     0     0
5             1     1     1     0     0
6             1     1     1     0     0
7             1     1     0     1     0
8             1     1     0     1     0
```

As you are learning to specify models in R you may get confused, or forget, just how to interpret the model your specifying.

You can easily assess your model using `model_matrix`

Run these functions and observe the difference

# TRANSFORMATIONS

You can also perform transformations inside the model formula

# TRANSFORMATIONS

You can also perform transformations inside the model formula

```
lm(log(y) ~ sqrt(x1) + x2)
```

log and square root transformations

# TRANSFORMATIONS

You can also perform transformations inside the model formula

```
lm(log(y) ~ sqrt(x1) + x2)
```

```
lm(y ~ x + I(x ^ 2))
```

log and square root transformations

transformations involving  $+$ ,  $*$ ,  $^$ , or  $-$ ,  
will need to be wrapped in **I()**

# TRANSFORMATIONS

You can also perform transformations inside the model formula

```
lm(log(y) ~ sqrt(x1) + x2)
```

```
lm(y ~ x + I(x ^ 2))
```

```
model_matrix(sim1, y ~ x + x^2)
```

```
model_matrix(sim1, y ~ x + I(x^2))
```

log and square root transformations

transformations involving  $+$ ,  $*$ ,  $^$ , or  $-$ , will need to be wrapped in `I()`

If you get confused, just use **`model_matrix`**

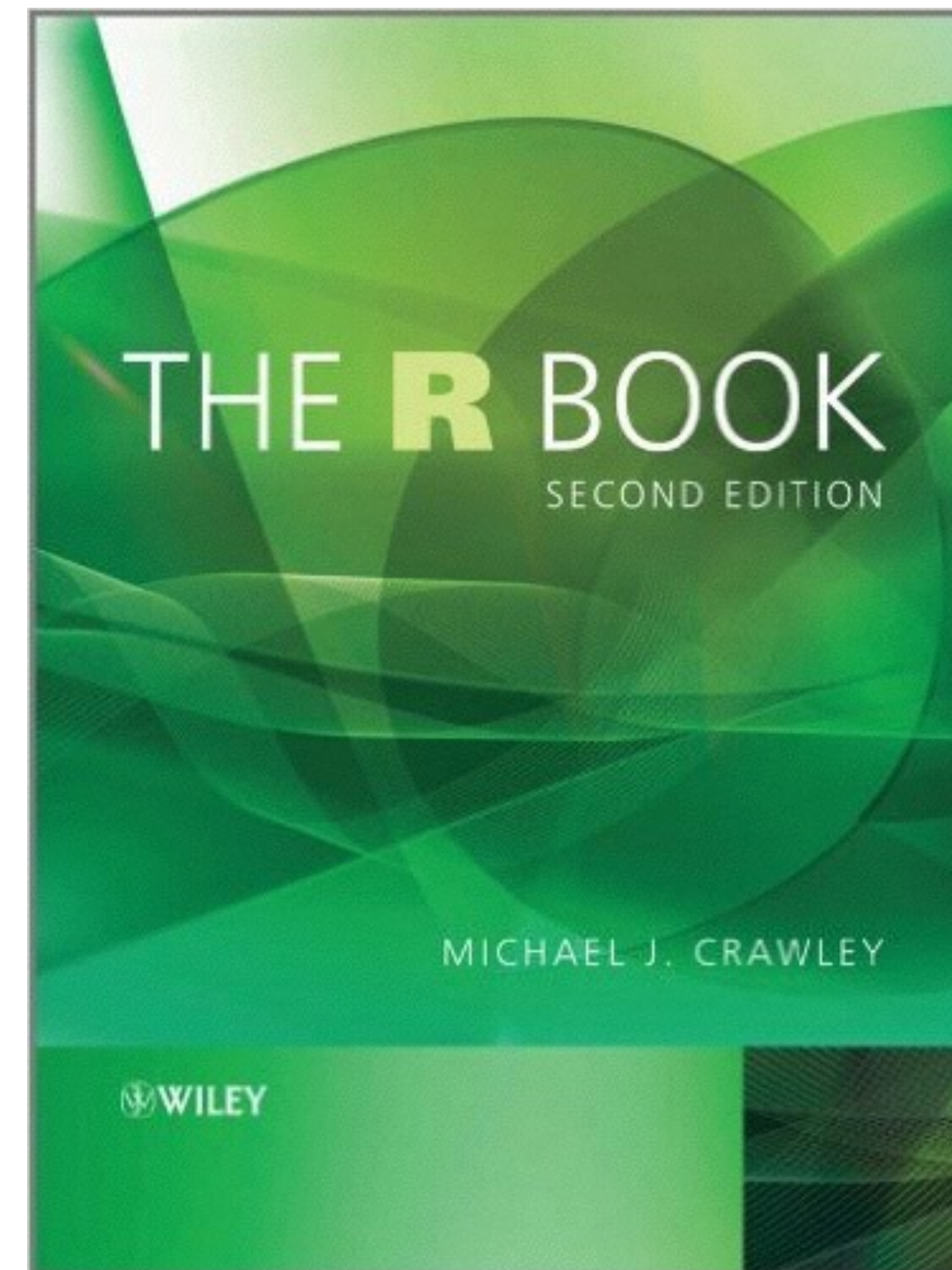
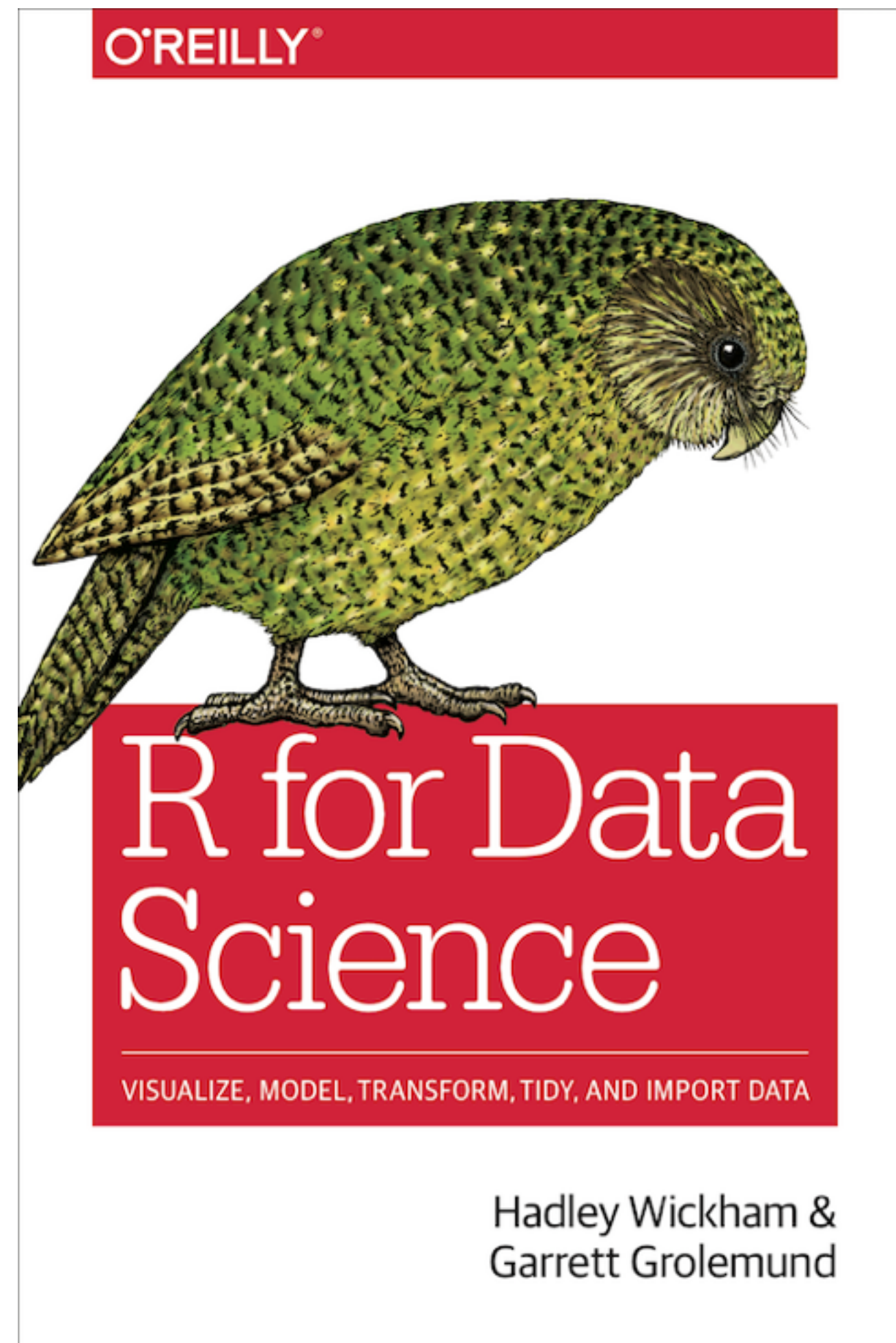


SO LITTLE TIME!





# LEARN MORE





WHAT TO REMEMBER



# FUNCTIONS TO REMEMBER

Operator/Function	Description
<code>cor, cor.test</code>	Compute correlation
<code>pairs, geom_ref_line</code>	Plot pairwise x-y scatterplots, add reference line to ggplot (great for assessing residual)
<code>lm(y ~ x, data = df)</code>	Linear model specification
<code>summary, residuals, fitted.values, coef</code>	Summarize and extract components out of the <code>lm()</code> object
<code>add_predictions, add_residuals, gather_predictions, gather_residuals</code>	Shortcut functions to add predicted values and residuals from an <code>lm()</code> object to a new or existing data frame
<code>model_matrix</code>	assess model specification