Stat405

Modelling

Hadley Wickham

1. Project 3
2. Warmups & goal
3. Fitting models
4. Intepreting models
5. Residuals
6. Predictive ability

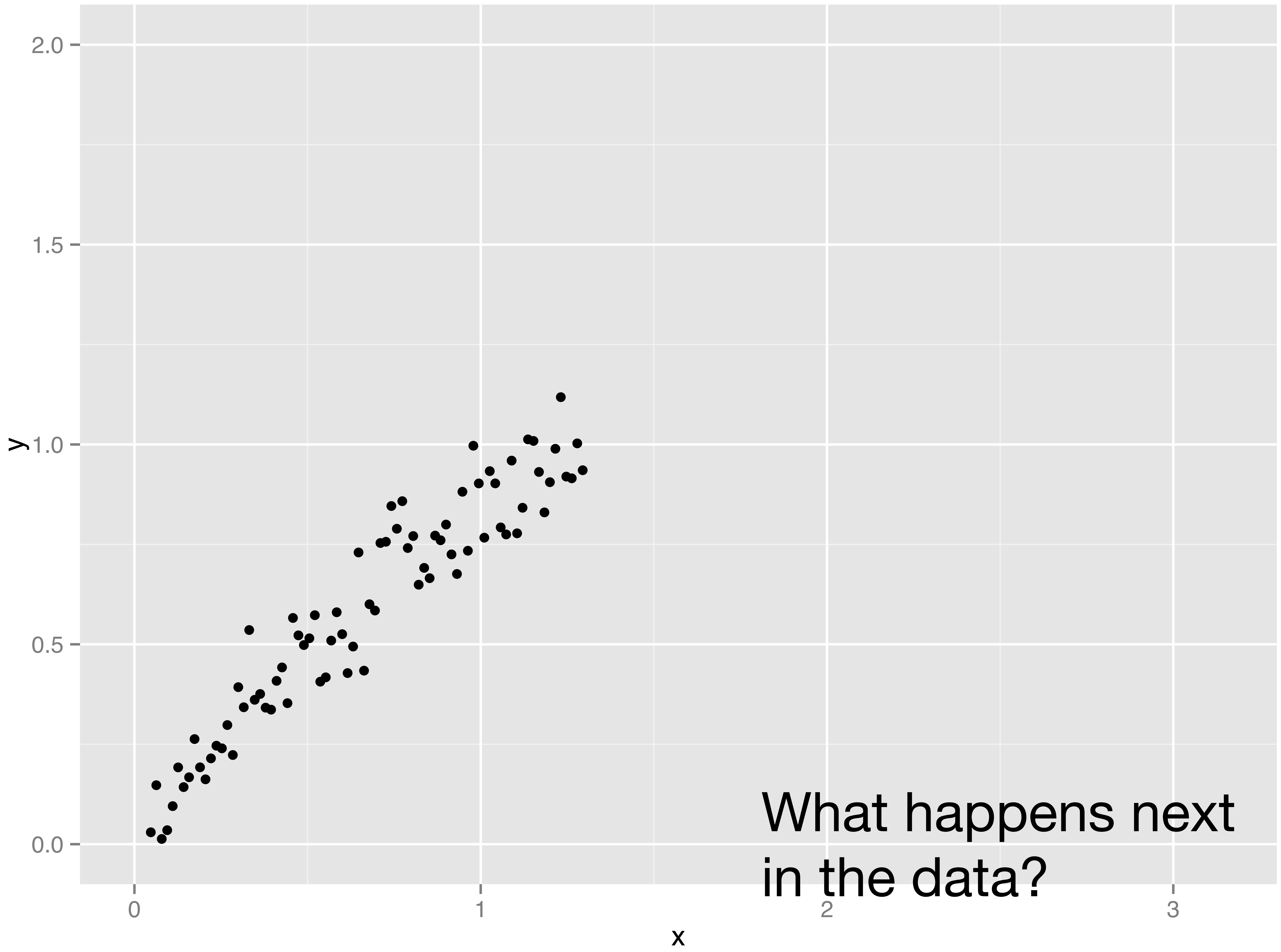
Project 3

- Make sure to read all the info on the website
- Must meet with me, Barret or Yeshaya by Nov 14
- I’ll be in the Pavillion Nov 14 1:30-5 for drop in appointments (about project 3 or anything else)
- Nov 15 = poster training with Tracy Volz
- http://ricestatistics.wufoo.com/forms/end-of-semester-poster-session/

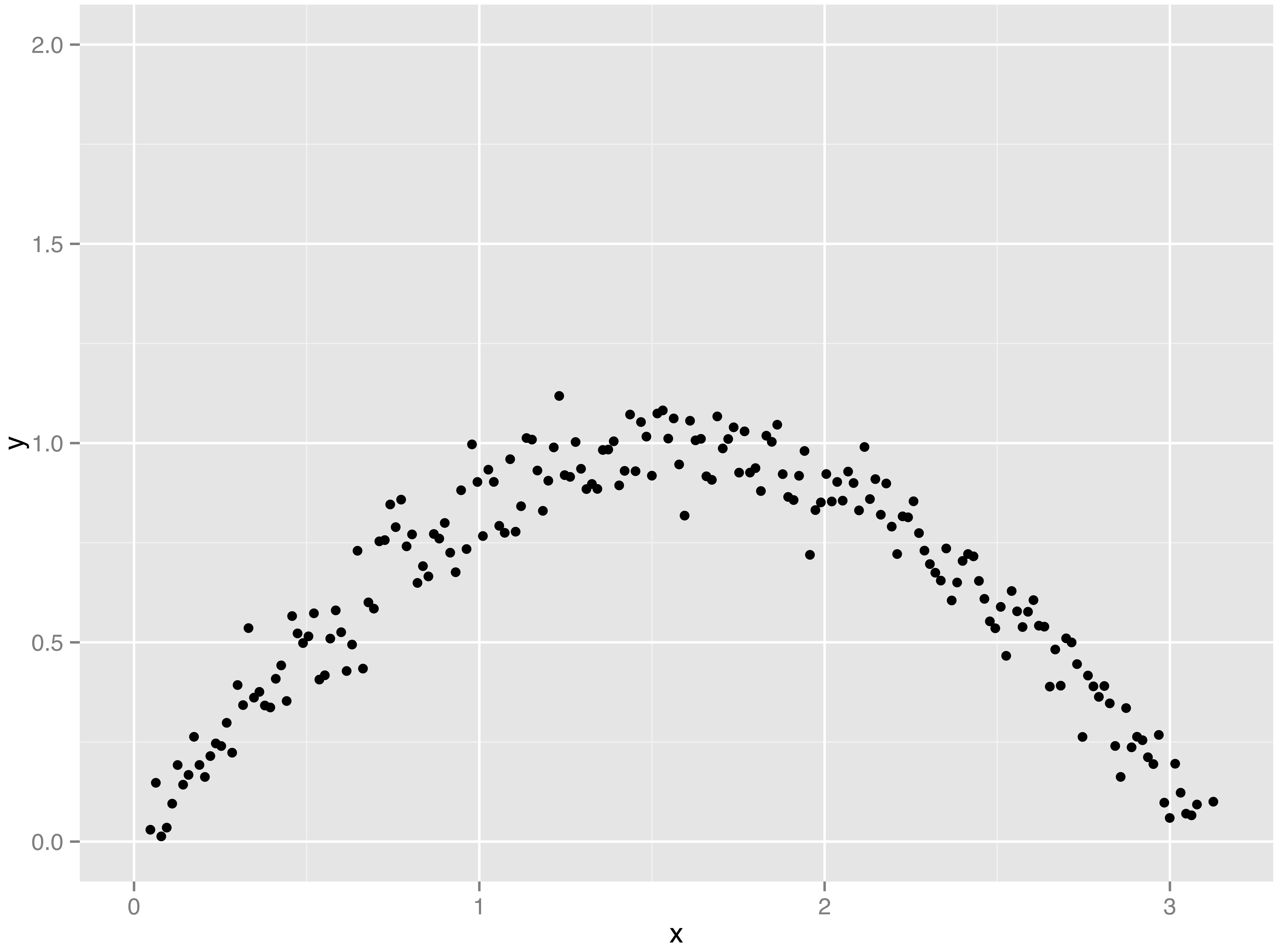
Warmups

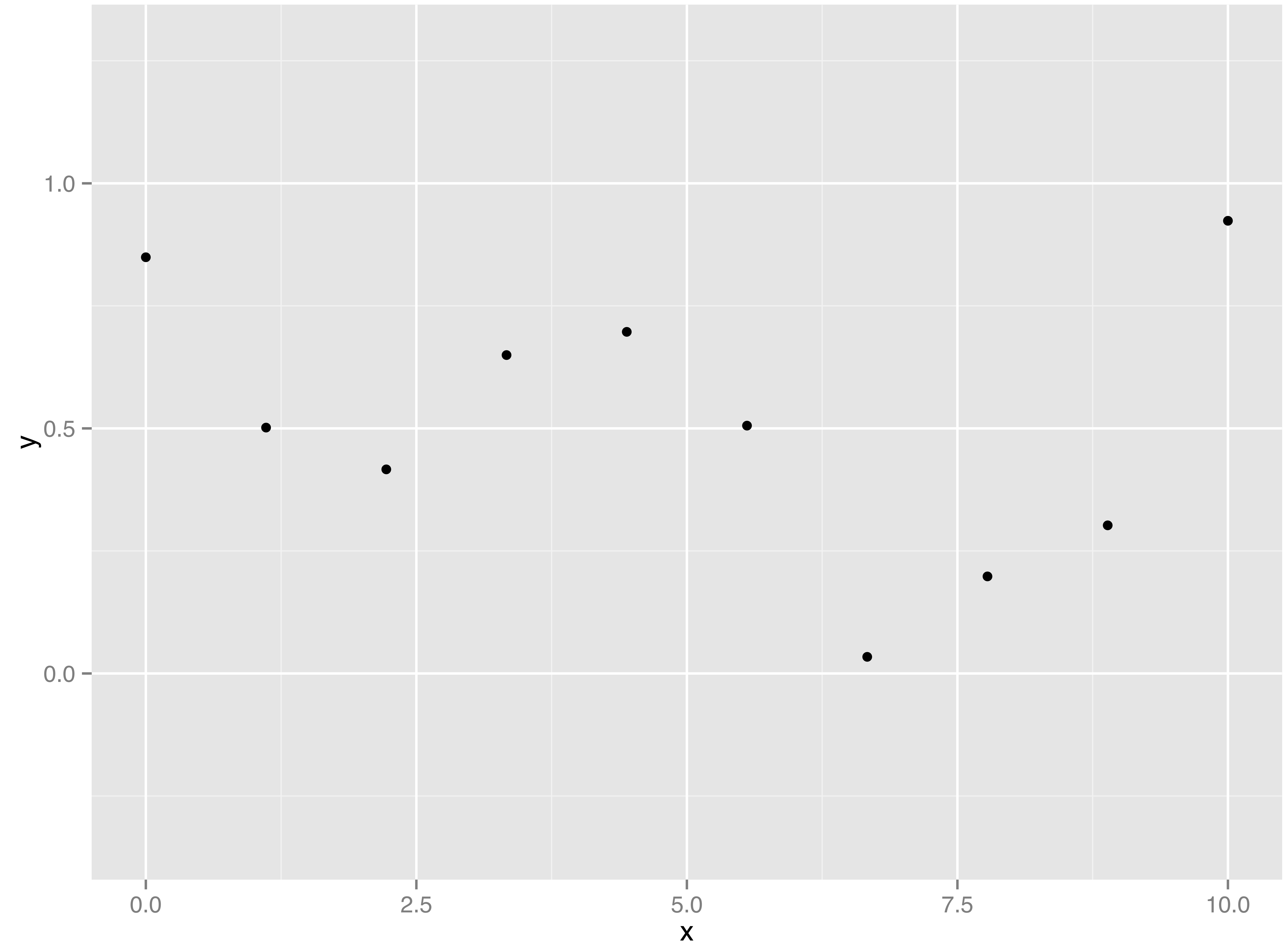
Your turn

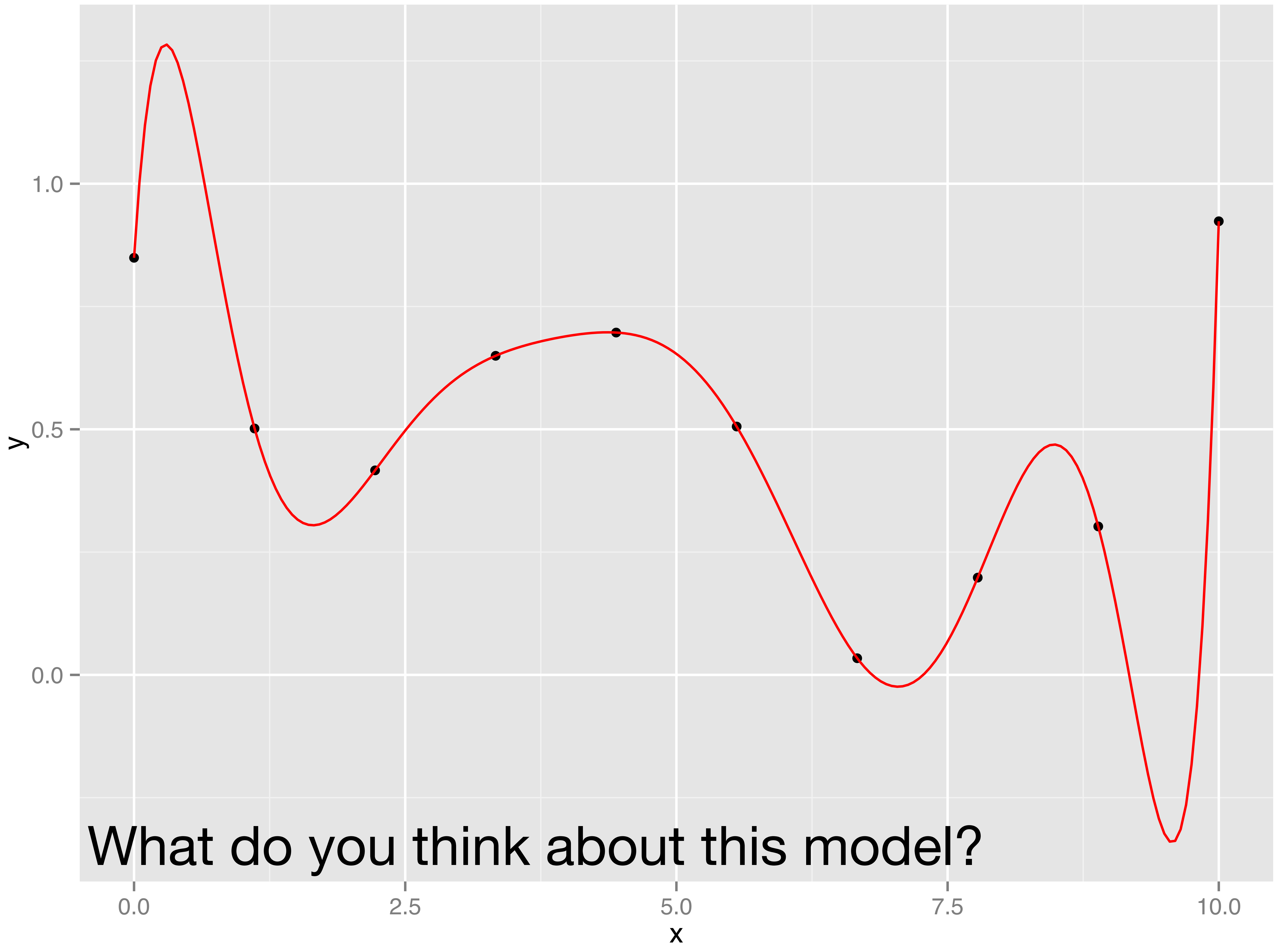
For each of the following slides, you'll get 30s to think about each question. Discuss it with your neighbours and then we'll talk about it as a whole.



What happens next in the data?

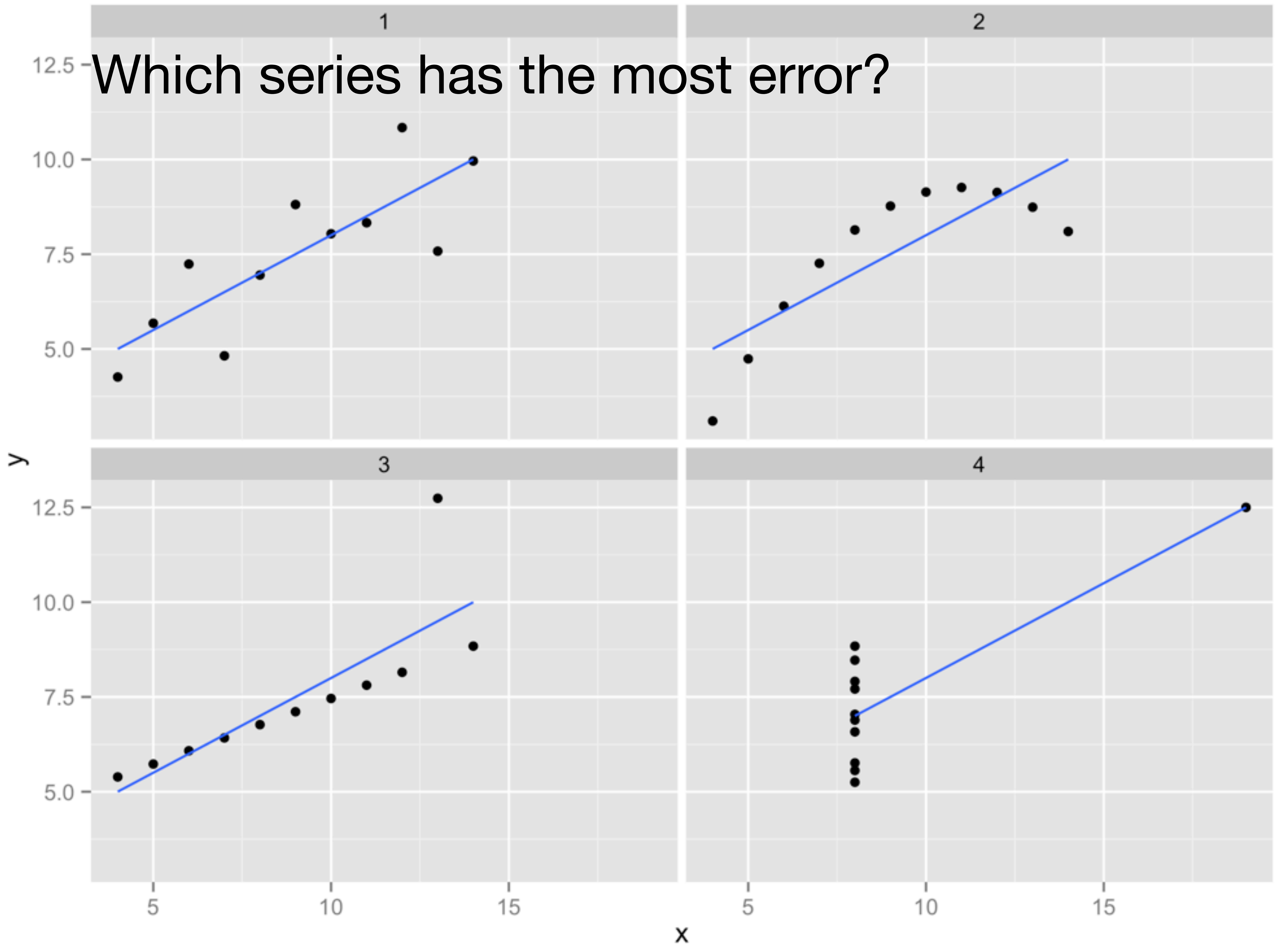

2.0
1.5
1.0
0.5
0.0
y
0
1
2
3
x


1.0
0.5
0.0
y
0.0
2.5
5.0
7.5
10.0
x



What do you think about this model?

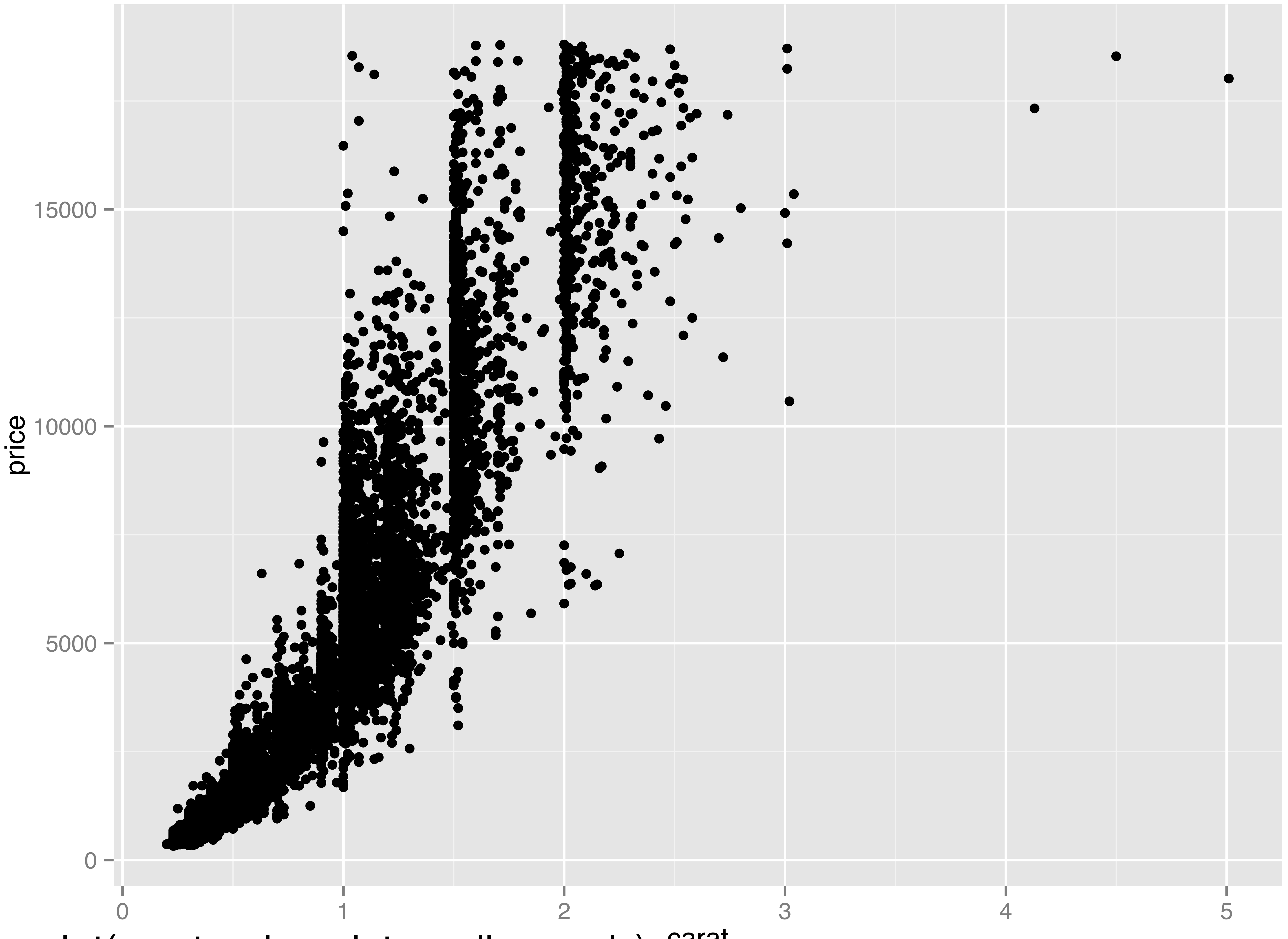
Which series has the most error?



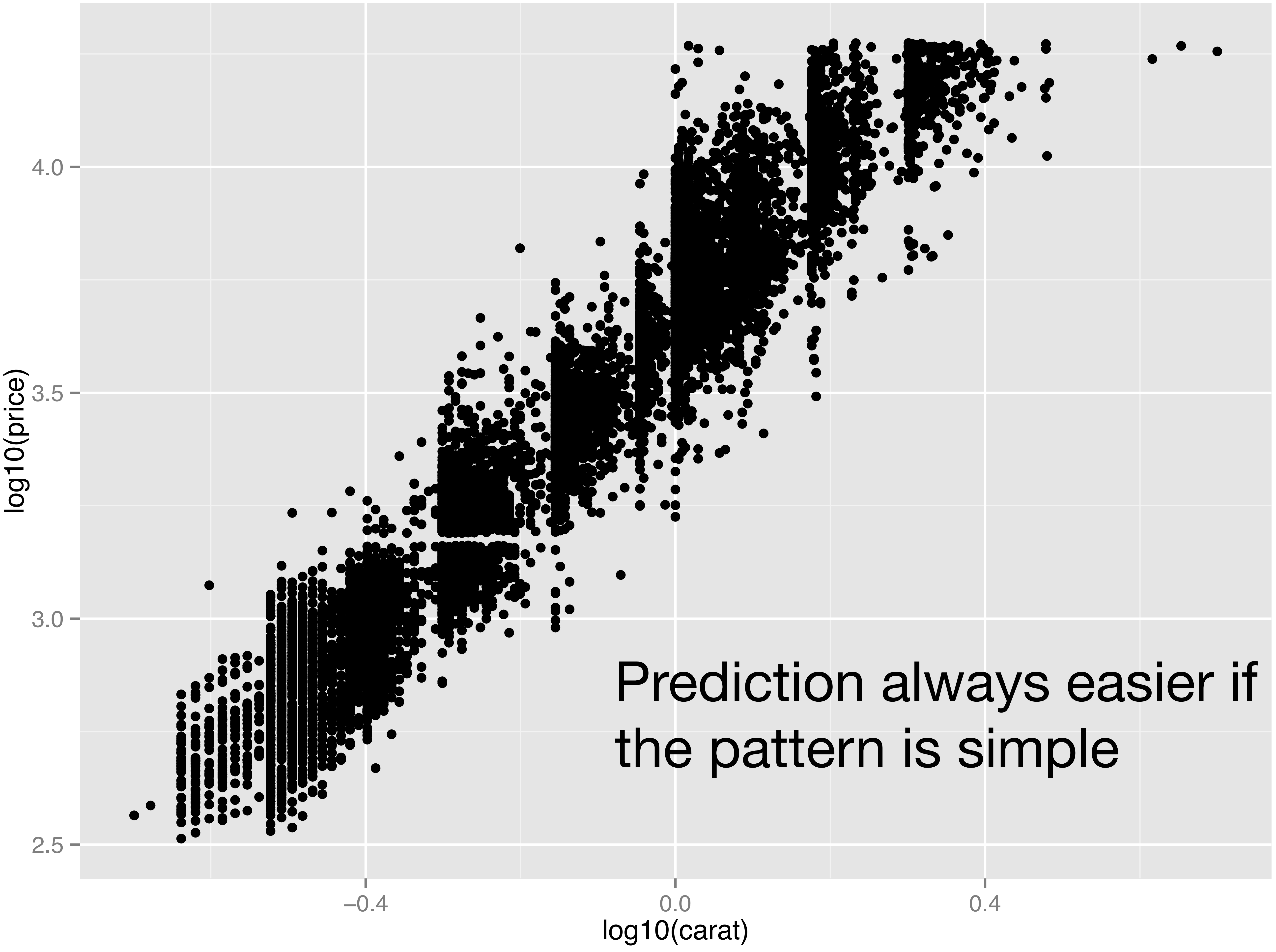
Goal

Motivation

Predict how much a diamond should cost.

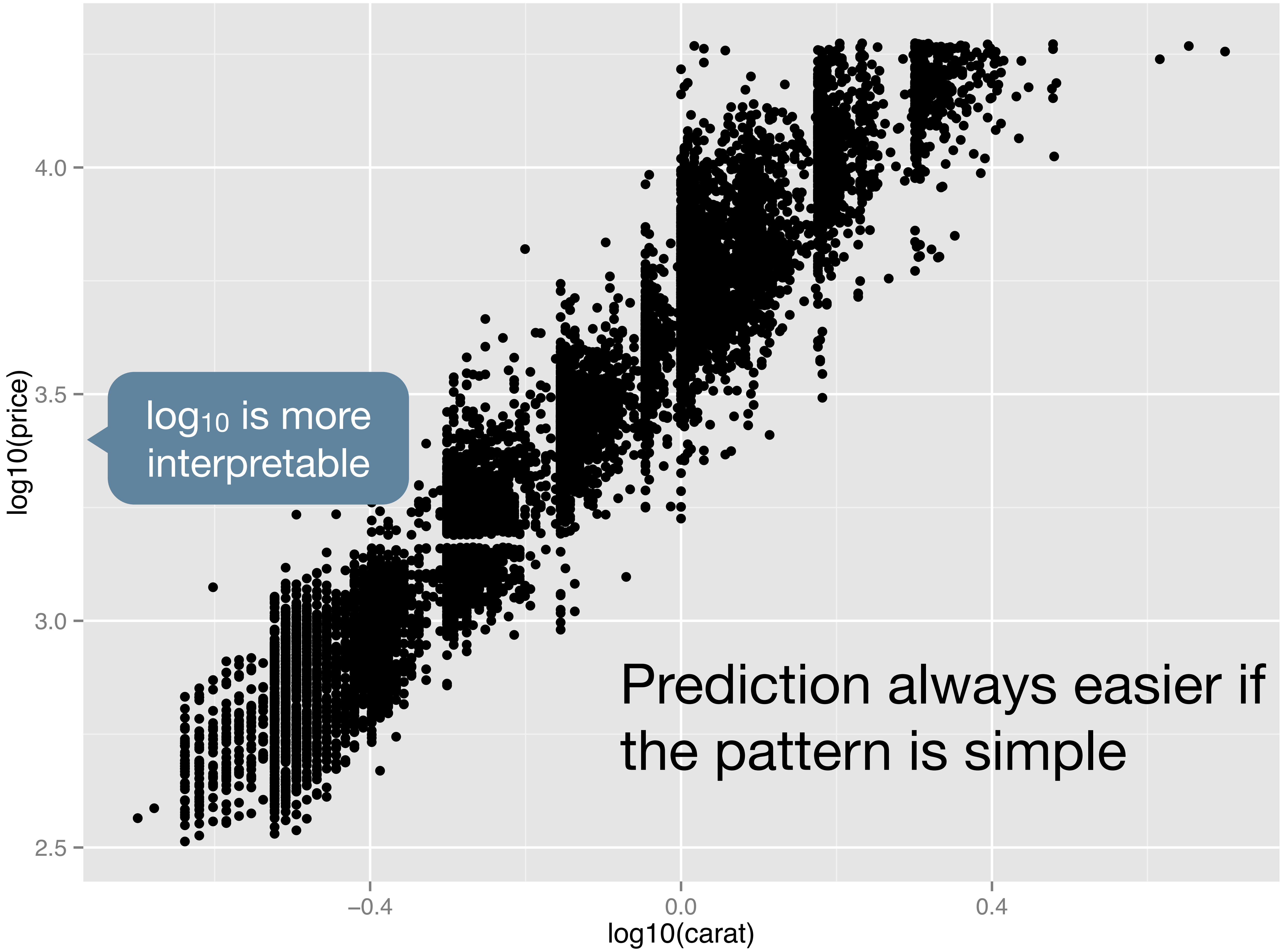


qplot(carat, price, data = diamonds)



Prediction always easier if the pattern is simple

```
qplot(log10(carat), log10(price), data = diamonds)
```



qplot(log10(carat), log10(price), data = diamonds)

```
library(ggplot2)
library(plyr)
source("helpers.r")
diamonds2 <- readRDS("diamonds2.rds")
```

diamonds2 has pre-made lcarat and lprice variables.

It also has fewer observations, which makes it easier to work with.

```
head(diamonds2)
   carat       cut color clarity depth table price    x    y    z     lcarat   lprice
1   0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43 -0.6382722 2.513218
6   0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48 -0.6197888 2.526339
11  0.30      Good     J     SI1  64.0    55   339 4.25 4.28 2.73 -0.5228787 2.530200
16  0.32   Premium     E      I1  60.9    58   345 4.38 4.42 2.68 -0.4948500 2.537819
21  0.30      Good     I     SI2  63.3    56   351 4.26 4.30 2.71 -0.5228787 2.545307
26  0.23 Very Good     G    VVS2  60.4    58   354 3.97 4.01 2.41 -0.6382722 2.549003
```

Fitting models in R

model syntax

```r
mod <- lm(hwy ~ displ, data = mpg, ...)
```

model syntax

Model formula:
response ~ predictor(s)

```
mod <- lm(hwy ~ displ, data = mpg, ...)
```

model syntax

Model formula:
response ~ predictor(s)

data

```
mod <- lm(hwy ~ displ, data = mpg, ...)
```

model syntax

```
mod <-     lm(hwy ~ displ, data = mpg, ...)
```

model syntax

```
mod <-     lm(hwy ~ displ, data = mpg, ...)
```

model syntax

```
mod <-   gam(hwy ~ displ, data = mpg, ...) ...
         glm
```

Your turn

Fit a linear model to the diamonds2 data set. The model should use *log carat* to predict *log price*.

```r
# Always save your model. There's TONS of info in it
mod <- lm(lprice ~ lcarat, data = diamonds2)

# Compare output
lm(lprice ~ lcarat, data = diamonds2)
summary(mod)
```

Coefficients

Coefficients become near impossible to interpret once you stop using simple linear models.

Generally its more useful to examine how predictions behave over the range of the x variables.

Interpreting models

Basic strategy

1. **Generate predictions**
   from the model at evenly spaced x values

2. **Visualize**
   the predictions

3. **Backtransform**
   the predictions if necessary

Extracting info

A common pattern for R models:

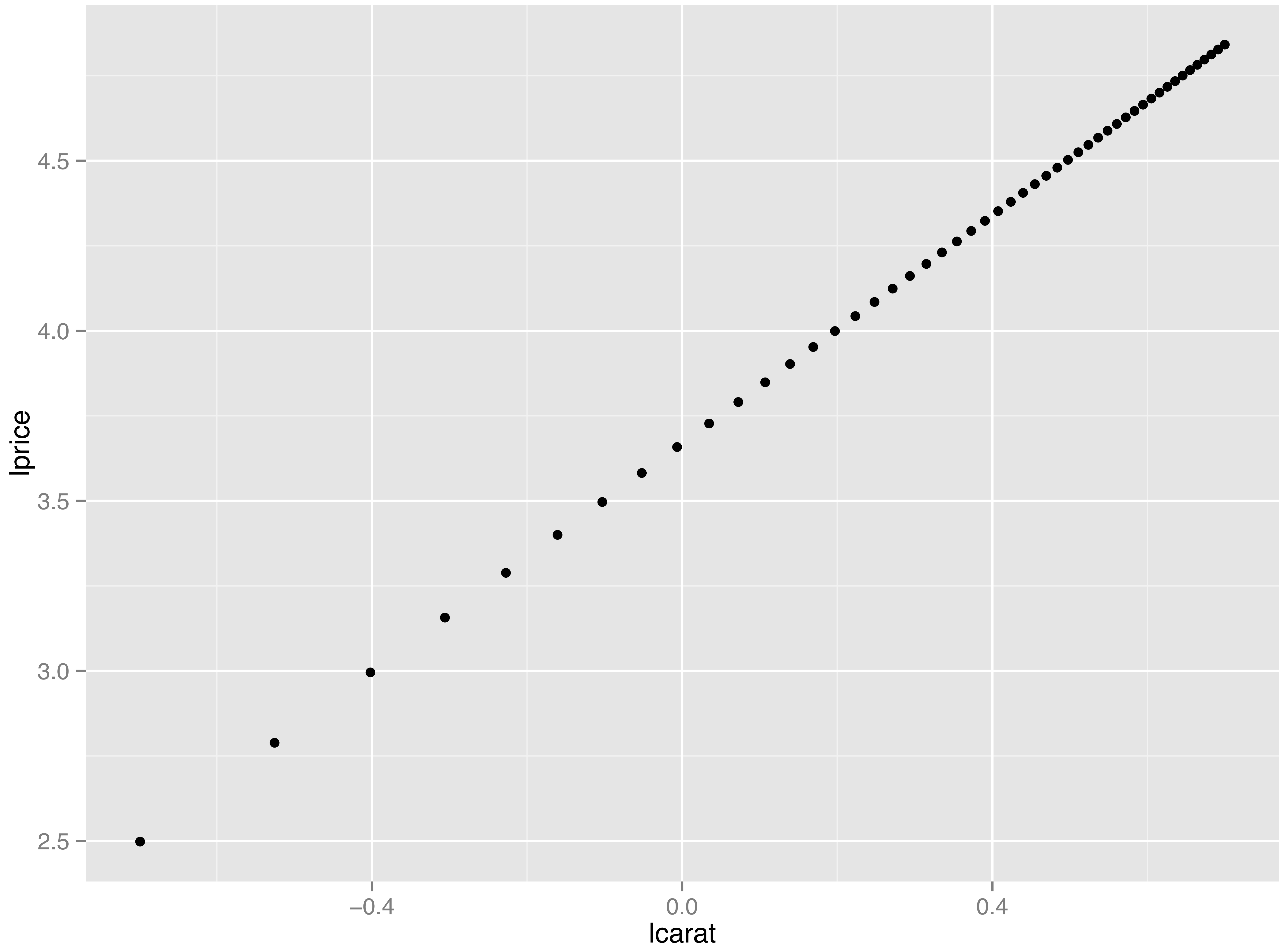
```
resid(mod)

coef(mod)

predict(mod) # predictions at original x values

predict(mod, df) # predictions at new x values
```

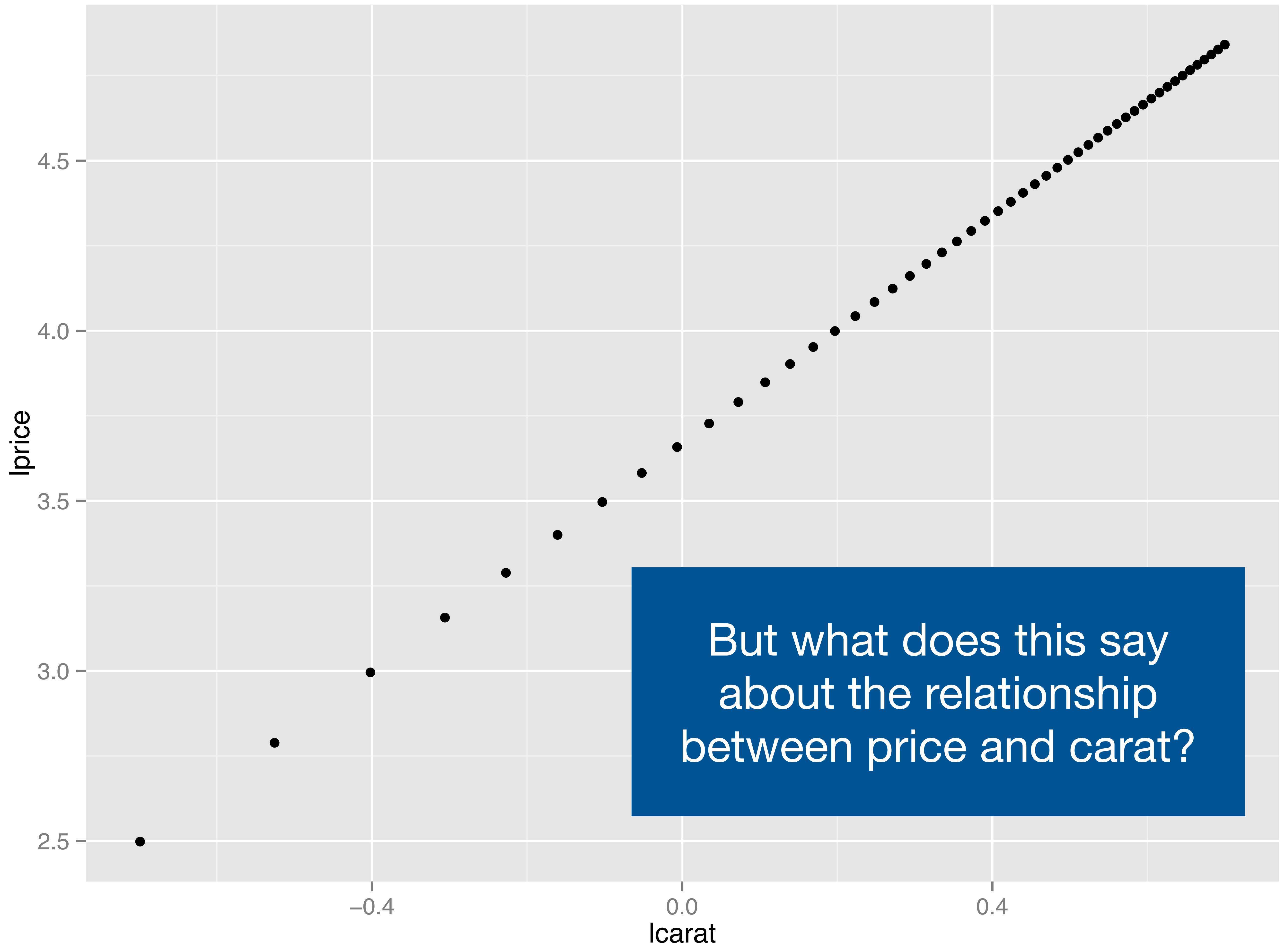
1. Generate predictions

```
# like summarise, but creates all combinations
# mod_grid and seq_range are in helpers.r
grid <- mod_grid(diamonds2,
  lcarat = log10(seq_range(carat, 50))
)

predict(mod, grid)
```

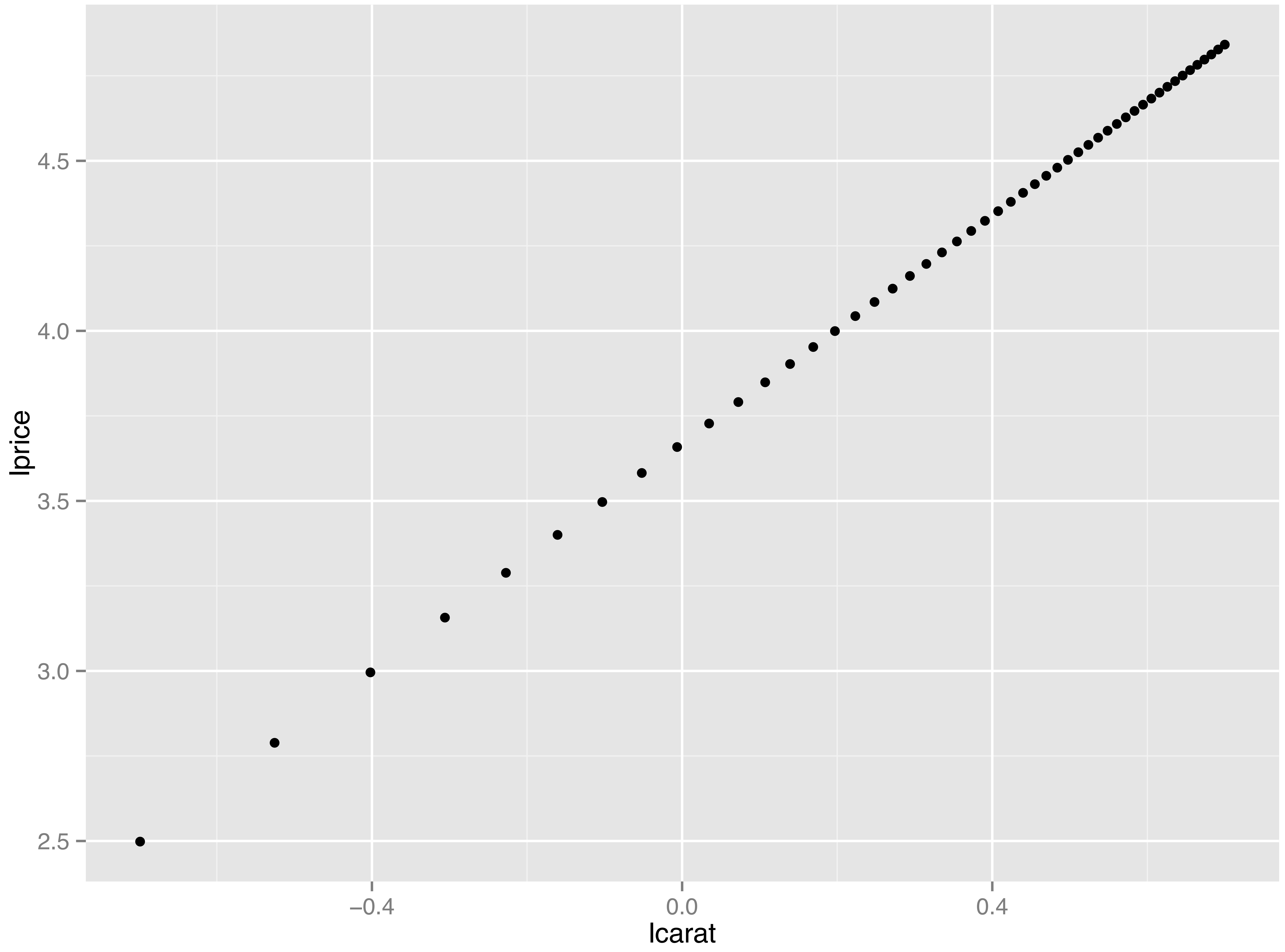
2. Visualise

```
grid$lprice <- predict(mod, grid)
qplot(lcarat, lprice, data = grid)
```


4.5
4.0
3.5
3.0
2.5
lprice
−0.4
0.0
0.4
lcarat



But what does this say about the relationship between price and carat?



lprice
lcarat
4.5
4.0
3.5
3.0
2.5
−0.4
0.0
0.4

Your turn

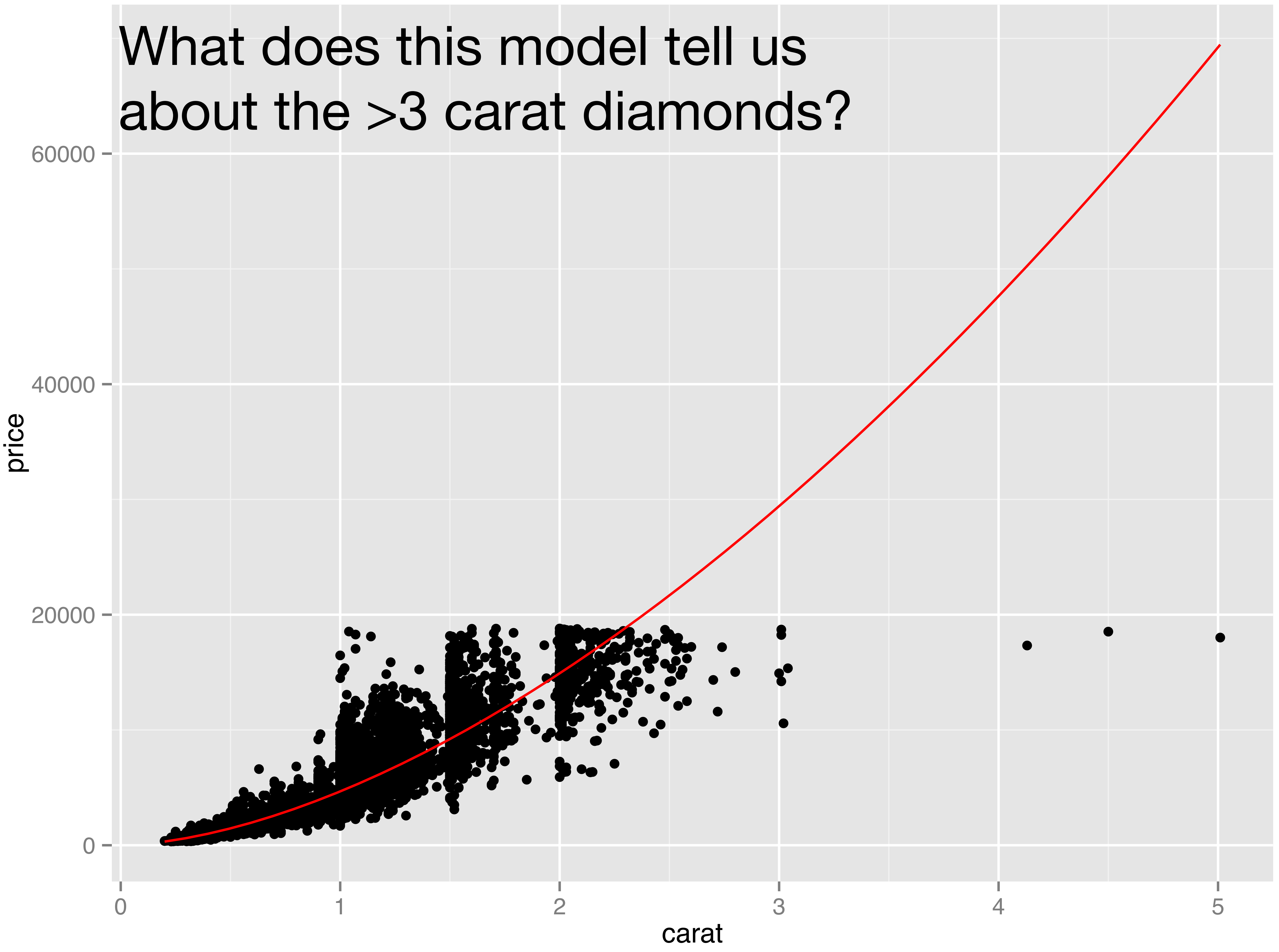
Back-transform the predictions and plot. What does the model say about the relationship between carat and price?

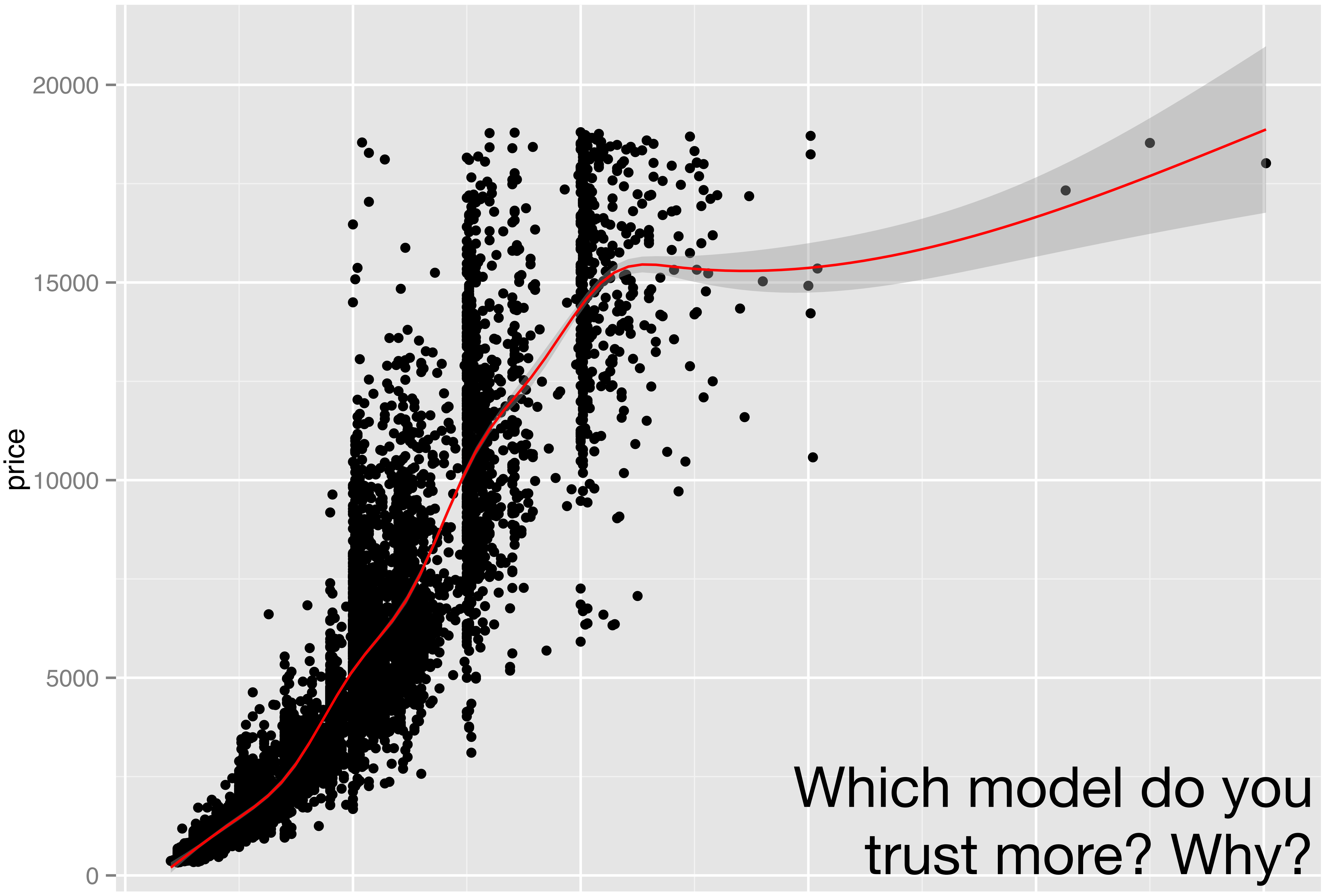
Hint: x = 10 ^ log10(x)

```r
grid <- mutate(grid,
  price = 10 ^ lprice,
  carat = 10 ^ lcarat)

qplot(carat, price, data = grid, geom = "line")
```

```
# To overlay
qplot(carat, price, data = diamonds2) +
  geom_line(data = grid, colour = "red")

# This works because grid and diamonds2 both
# have carat and price variables
```

What does this model tell us about the >3 carat diamonds?





Which model do you trust more? Why?

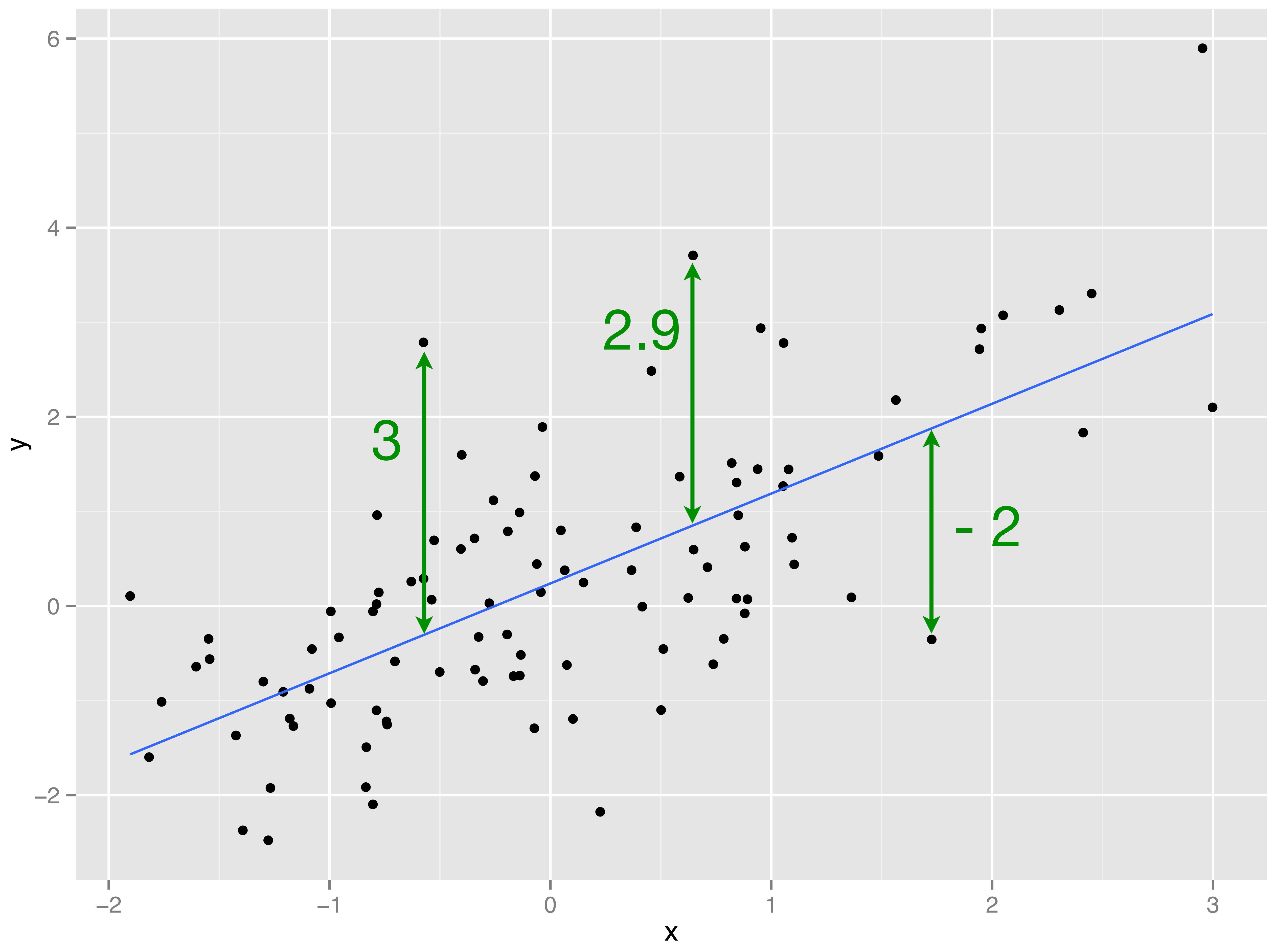
```
qplot(carat, price, data = diamonds2) +
  geom_smooth(colour = "red")
```

Residuals

Residuals

Instead of just looking at predictions, we can look at the difference between predictions and reality: the **residuals**.

Looking at residuals is an important exploratory technique because it allows us to remove strong patterns and look for the more subtle patterns that remain.


6
4
2
0
−2
y
−2
−1
0
1
2
3
x
3
2.9
- 2

```
diamonds2$lprice2 <- diamonds2$lprice -
   predict(mod, diamonds2)
qplot(lcarat, lprice2, data = diamonds2)

# We'll use resid2 (defined in helpers) rather than
# the built-in in resid, because it's more
# consistent across more types of model.

diamonds2$lprice2 <- resid2(mod, diamonds2)
qplot(lcarat, lprice2, data = diamonds2)
```

Your turn

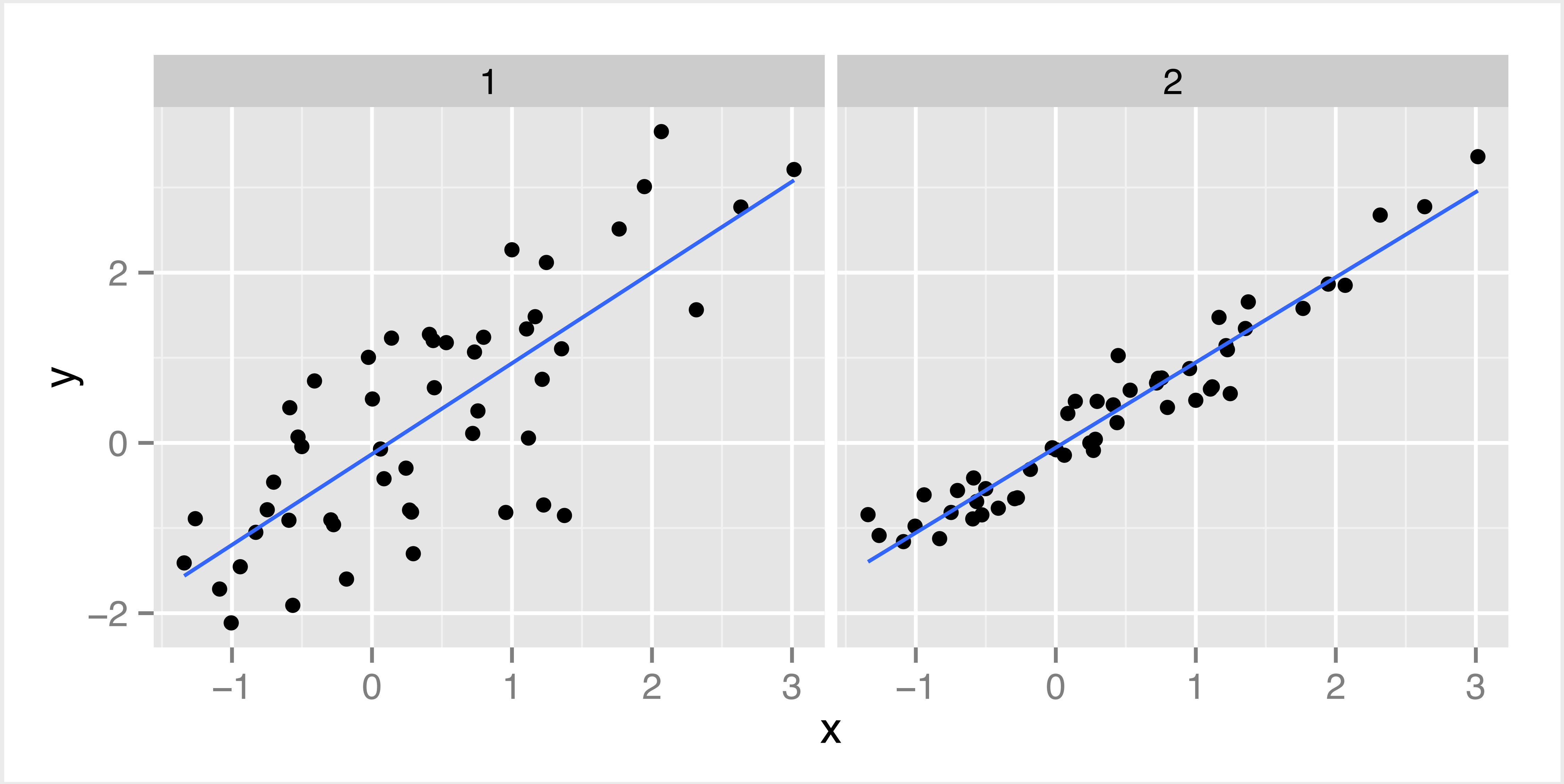
Now that we've removed the effect of size, which of cut, clarity and color is most related to price? Use visualization to explore.

```
qplot(lcarat, lprice2, data = diamonds2,
  colour = cut)
qplot(lcarat, lprice2, data = diamonds2,
  colour = clarity)
qplot(lcarat, lprice2, data = diamonds2,
  colour = color)

# Clarity looks best, but how can we quantify?
```

Predictive ability

Your turn

Debate with your neighbor. Which model will make better predictions? **Why?**



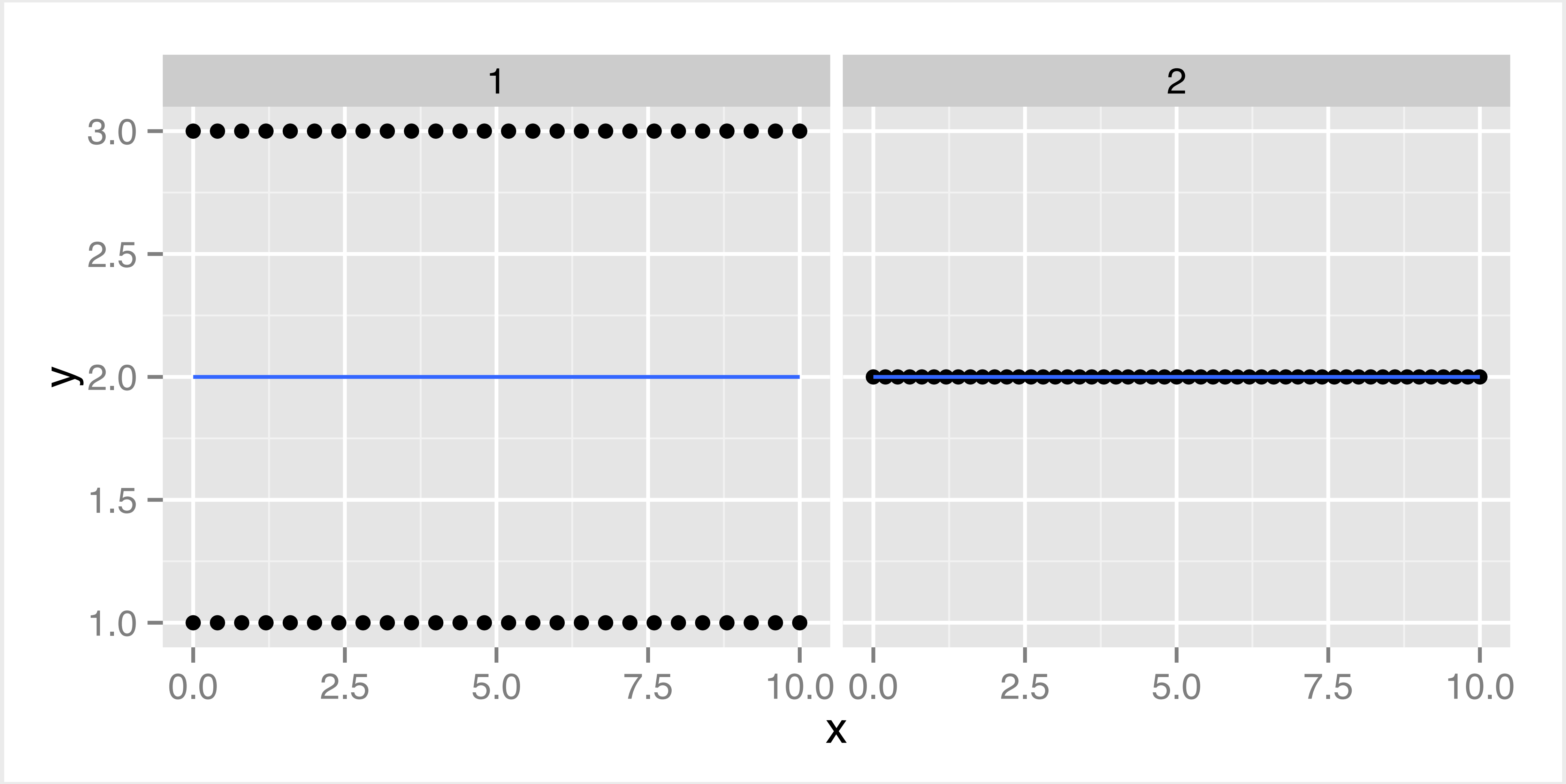
More formally

We need some way of measuring the “quality” of a model. The logic here is very simple:

The larger the average residual, the worse the predictive ability

Your turn

With your neighbor, devise a way to count the average amount of “residual” that can also distinguish between these models:



$$\frac{1}{n}\sum residuals$$

$$\frac{1}{n}\sum residuals$$ 0!

$$\frac{1}{n}\sum residuals$$ 0!

$$\frac{1}{n}\sum |residuals|$$

$\frac{1}{n} \sum residuals$ 0!

$\frac{1}{n} \sum |residuals|$ Mean absolute residual

$$\frac{1}{n}\sum residuals$$ 0!

$$\frac{1}{n}\sum |residuals|$$ Mean absolute residual

$$\frac{1}{n}\sum residuals^2$$

$\frac{1}{n}\sum residuals$	0!
$\frac{1}{n}\sum \lvert residuals \rvert$	Mean absolute residual
$\frac{1}{n}\sum residuals^2$	Mean squared residual

$$\frac{1}{n}\sum residuals$$ 0!

$$\frac{1}{n}\sum |residuals|$$ Mean absolute residual

$$\frac{1}{n}\sum residuals^2$$ Mean squared residual

- but what units is this?

$$\frac{1}{n}\sum residuals$$ 0!

$$\frac{1}{n}\sum |residuals|$$ Mean absolute residual

$$\frac{1}{n}\sum residuals^2$$ Mean squared residual

- but what units is this?

$$\sqrt{\frac{1}{n}\sum residuals^2}$$

$$\frac{1}{n}\sum residuals$$

0!

$$\frac{1}{n}\sum |residuals|$$

Mean absolute residual

$$\frac{1}{n}\sum residuals^2$$

Mean squared residual

- but what units is this?

$$\sqrt{\frac{1}{n}\sum residuals^2}$$

Root mean squared residual

$$\frac{1}{n}\sum residuals$$

0!

$$\frac{1}{n}\sum |residuals|$$

Mean absolute residual

$$\sqrt{\frac{1}{n}\sum residuals^2}$$

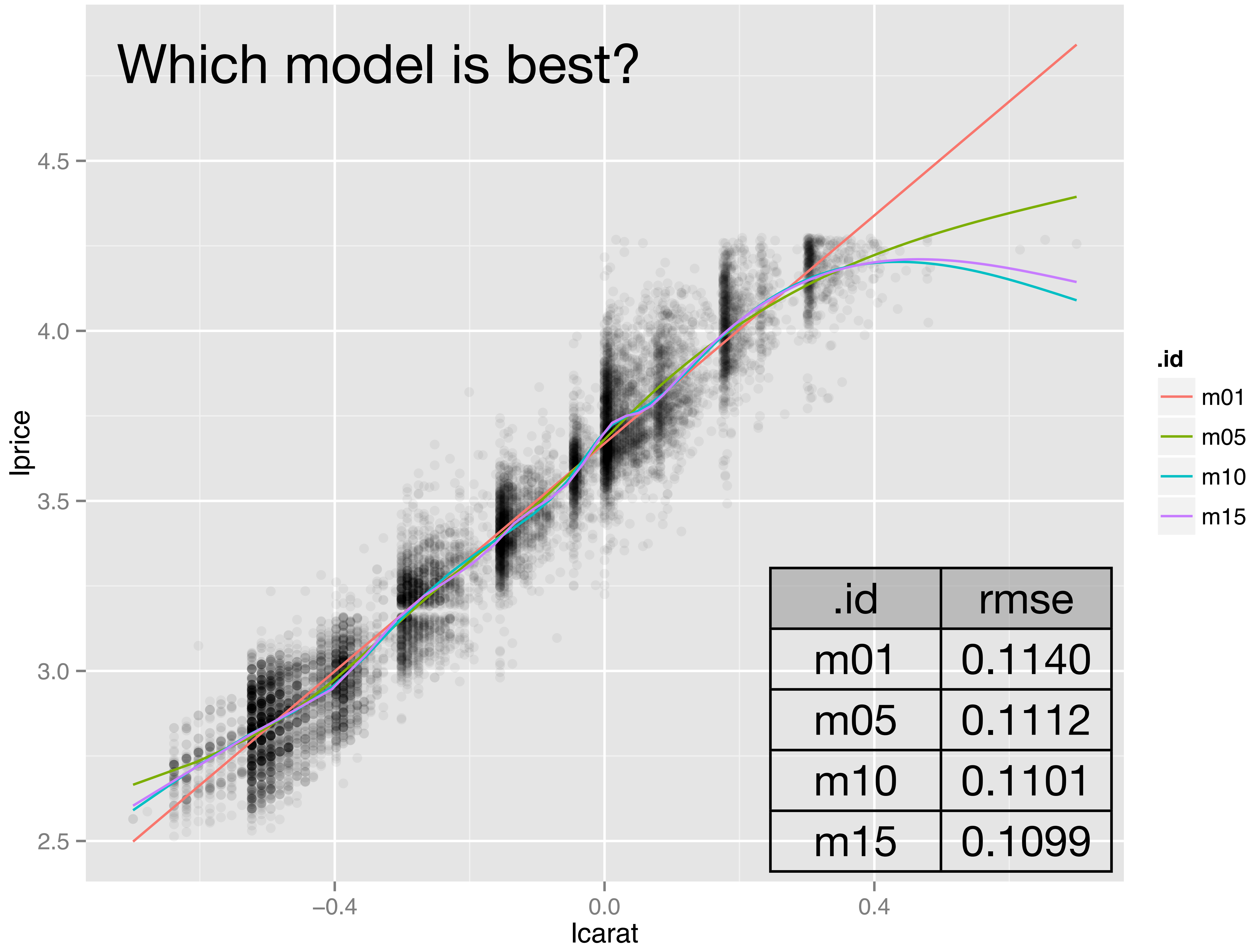
Root mean
squared residual

```r
# These are both defined in helpers.r

rmse <- function(mod, data, r = resid2) {
  sqrt(mean(r(mod, data) ^ 2))
}
rd <- function(mod, data, r = resid2) {
  quantile(abs(r(mod, data)),
    c(0.25, 0.5, 0.75, 0.9))
}

# The best model is the model with the smallest
# prediction errors

rmse(mod, diamonds2)
rd(mod, diamonds2)
```

```r
# Now need models with carat and additional
# variable:

mod_cut <- lm(lprice ~ lcarat + cut,
  data = diamonds2)
mod_clarity <- lm(lprice ~ lcarat + clarity,
  data = diamonds2)
mod_color <- lm(lprice ~ lcarat + color,
  data = diamonds2)

# Which of these models is best?
# rmse = root mean squared error
rmse(mod_cut, diamonds2)
rmse(mod_clarity, diamonds2)
rmse(mod_color, diamonds2)
# rd = (absolute) residual distribution
rd(mod_cut, diamonds2)
rd(mod_clarity, diamonds2)
rd(mod_color, diamonds2)
```

```r
# To make it easier to interpret, we can
# back transform
tresid <- function(mod, data) {
  10 ^ predict(mod, data = data) - 10 ^ data$lprice
}
rmse(mod, diamonds2, tresid)
rd(mod, diamonds2, tresid)

rd(mod_cut, diamonds2, tresid)
rd(mod_clarity, diamonds2, tresid)
rd(mod_color, diamonds2, tresid)
```

```r
# Special case: linear models

anova(lm(lprice ~ lcarat + cut + clarity + color,
  data = diamonds2))
```



.id	rmse
m01	0.1140
m05	0.1112
m10	0.1101
m15	0.1099

Discussion

m15 is implausible, but has the lowest rmse. What gives?

Choosing the model that best fits the **existing** data leads to **overfitting**. We expect it will do worse when predicting new data.

But first we need some new tools...

Lists of models

New tools

When working with multiple models, we need some better tools to reduce duplication.

Lists are an R data structure that can store anything (including other lists).

ldply takes a list as input and returns a data frame as output.

```
mods <- list(
  m01 = lm(lprice ~ lcarat, data = diamonds2),
  m05 = lm(lprice ~ ns(lcarat, 5), data = diamonds2),
  m10 = lm(lprice ~ ns(lcarat, 10), data = diamonds2),
  m15 = lm(lprice ~ ns(lcarat, 15), data = diamonds2)
)

mods
mods$m05  # Extract by name
mods[1]   # Extract by position (list of length 1)
mods[[1]] # Extract by position (model)

# Linear models are also lists!
is.list(mods[[1]])
names(mods[[1]])
mods[[1]]$rank
```

If list `x` is a train carrying objects, then `x[[5]]` is the object in car 5; `x[4:6]` is a train of cars 4-6.

http://twitter.com/#!/RLangTip/status/118339256388304896

```r
grid <- mod_grid(diamonds2, lcarat = seq_range(lcarat, 100))

# ldply works like ddply, but instead of taking a
# data frame as input and breaking it up by variables
# it takes a list and operates on each piece

ldply(mods, function(mod) predict(mod, grid))
grids <- ldply(mods, function(mod) {
  grid$lprice <- predict(mod, grid)
  grid
})
head(grids)
# ldply automatically adds .id variable from names of list

qplot(lcarat, lprice, data = grids, geom = "line",
  colour = .id)

ldply(mods, rmse, data = diamonds2)
```

Your turn

Create a list of models that predict lprice with carat and cut, colour, clarity. (i.e. a length of list 3)

Use `ldply` to predict over a grid of diamond prices for the most common type of diamond (Ideal, E, VS2). Display the predictions on one plot.

Which model is the best?

```r
mods2 <- list(
  cut = lm(lprice ~ lcarat + cut, data = diamonds2),
  clarity = lm(lprice ~ lcarat + clarity, data = diamonds2),
  color = lm(lprice ~ lcarat + color, data = diamonds2)
)
grid <- mod_grid(diamonds2,
  lcarat = seq_range(lcarat, 100),
  cut = "Ideal",
  color = "E",
  clarity = "VS2")
grids <- ldply(mods2, function(mod) {
  grid$lprice <- predict(mod, grid)
  grid
})
qplot(lcarat, lprice, data = grids, geom = "line", colour = .id)
ldply(mods, rmse, data = diamonds2)
```

```r
# When you learn more about R, you’ll be able to
# reduce the duplication even further

fs <- list(
  cut = lprice ~ lcarat + cut,
  clarity = lprice ~ lcarat + clarity,
  color = lprice ~ lcarat + color
)
mods <- lapply(fs, lm, data = diamonds2)
```

Cross-validation

Problem

We’ve been assessing the predictive ability of models based on data they’ve already seen!

Imagine a model that made predictions by finding the closest observation in the original data and using that as prediction. It would be very good at predictions from the original data, but very bad for new data.

Unbiased errors

Solution: Assess the model based on data it hasn't seen.

Problem: where do we get that data from?

Solution: randomly divide the dataset into **training** and **test** datasets

Your turn

Select 90% of the diamonds2 dataset (e.g. `sample(nrow(diamonds2), nrow(diamonds2) * 0.9)`).

Fit the model to the training data.

Calculate the rmse on training data and the test data. How do they compare?

Discuss with your neighbour. What results do they get?

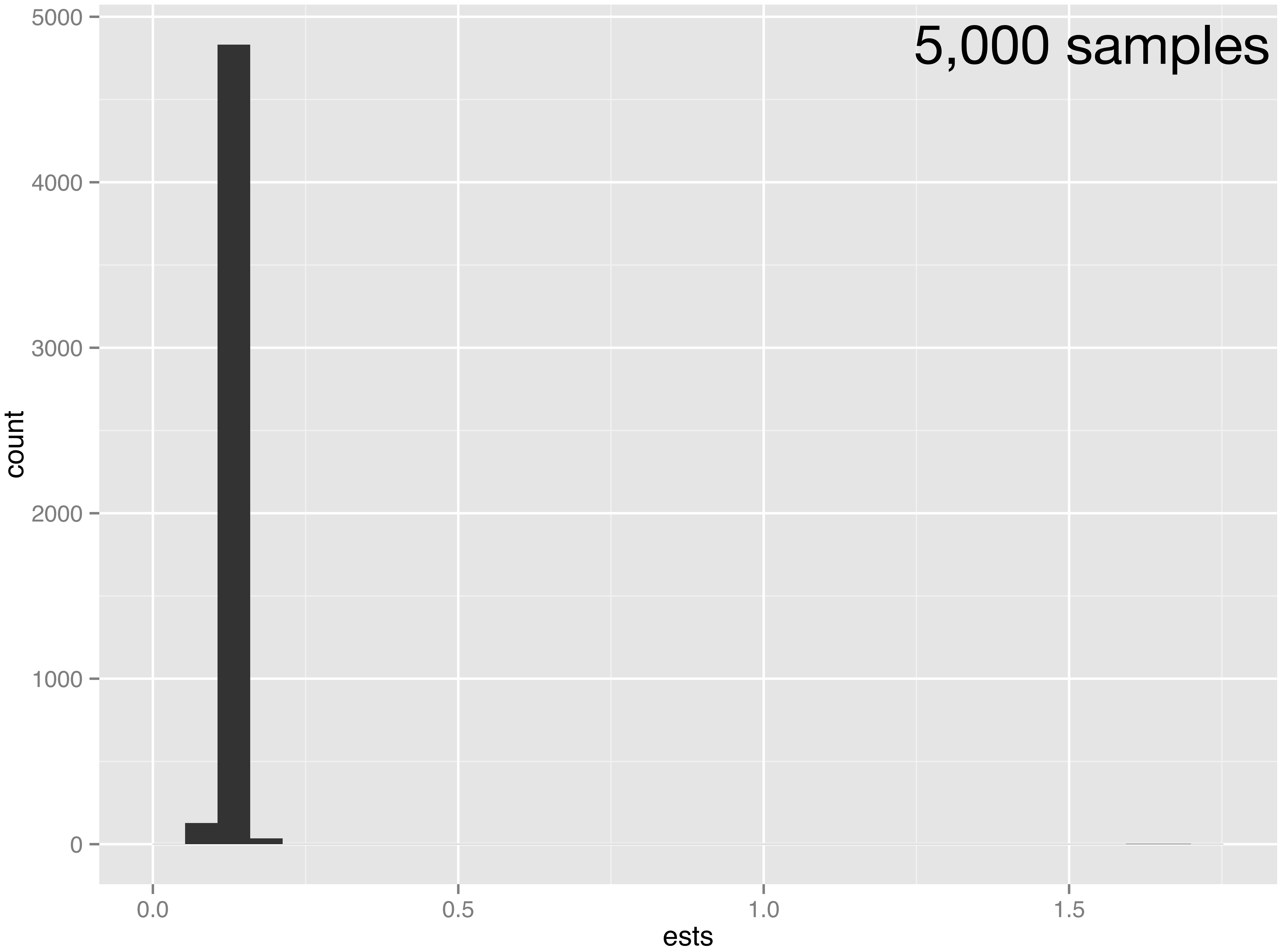
```r
train <- sample(nrow(diamonds2), nrow(diamonds2) * 0.9)
test <- -train

mod15 <- lm(lprice ~ poly(lcarat, 15),
  data = diamonds2[train, ])
rmse(mod15, diamonds2[train, ])
rmse(mod15, diamonds2[test, ])
```

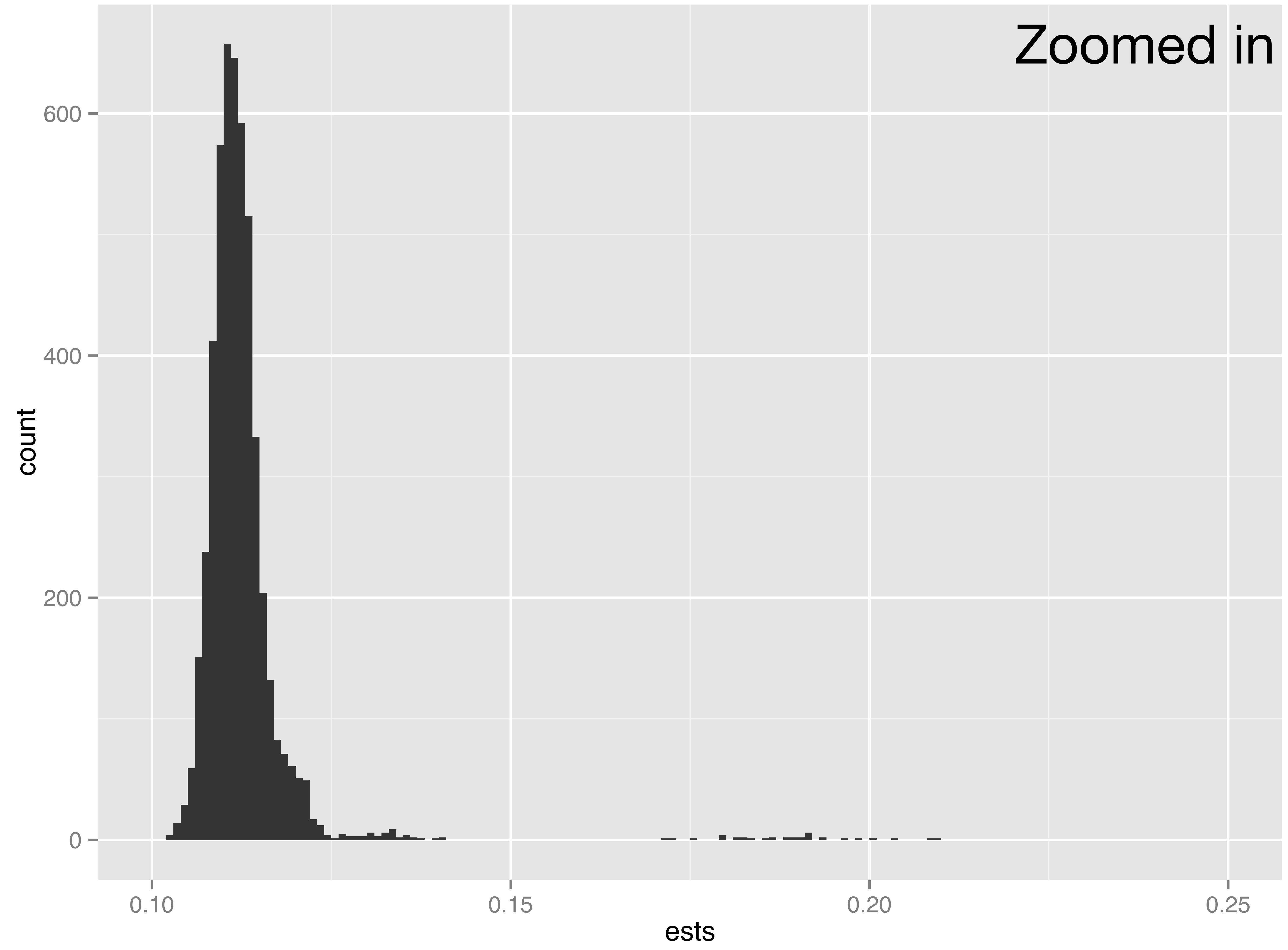
Cross-validation

The subset is random, so you need to repeat it multiple times.

The standard is to do it 10 times, holding out 10% for testing. (This is completely arbitrary just like using a p-value of 0.05)

This technique is called cross-validation.


5,000 samples
count
5000
4000
3000
2000
1000
0
0.0
0.5
1.0
1.5
ests


Zoomed in
count
600
400
200
0
0.10
0.15
0.20
0.25
ests

```
# In helpers.r, I've provided a function that
# does this for you.  Given a model and a data set
# it refits the model to training samples of the
# data and computes the rmse on the test set.

cv_rmse(mod10, diamonds2, n = 20)

ests <- ldply(mods, as.data.frame(cv_rmse),
  data = diamonds2)
qplot(.id, value, data = ests)

# Key take away: rmse will always give you an overly
# optimistic view of your model. DON'T USE IT!
```