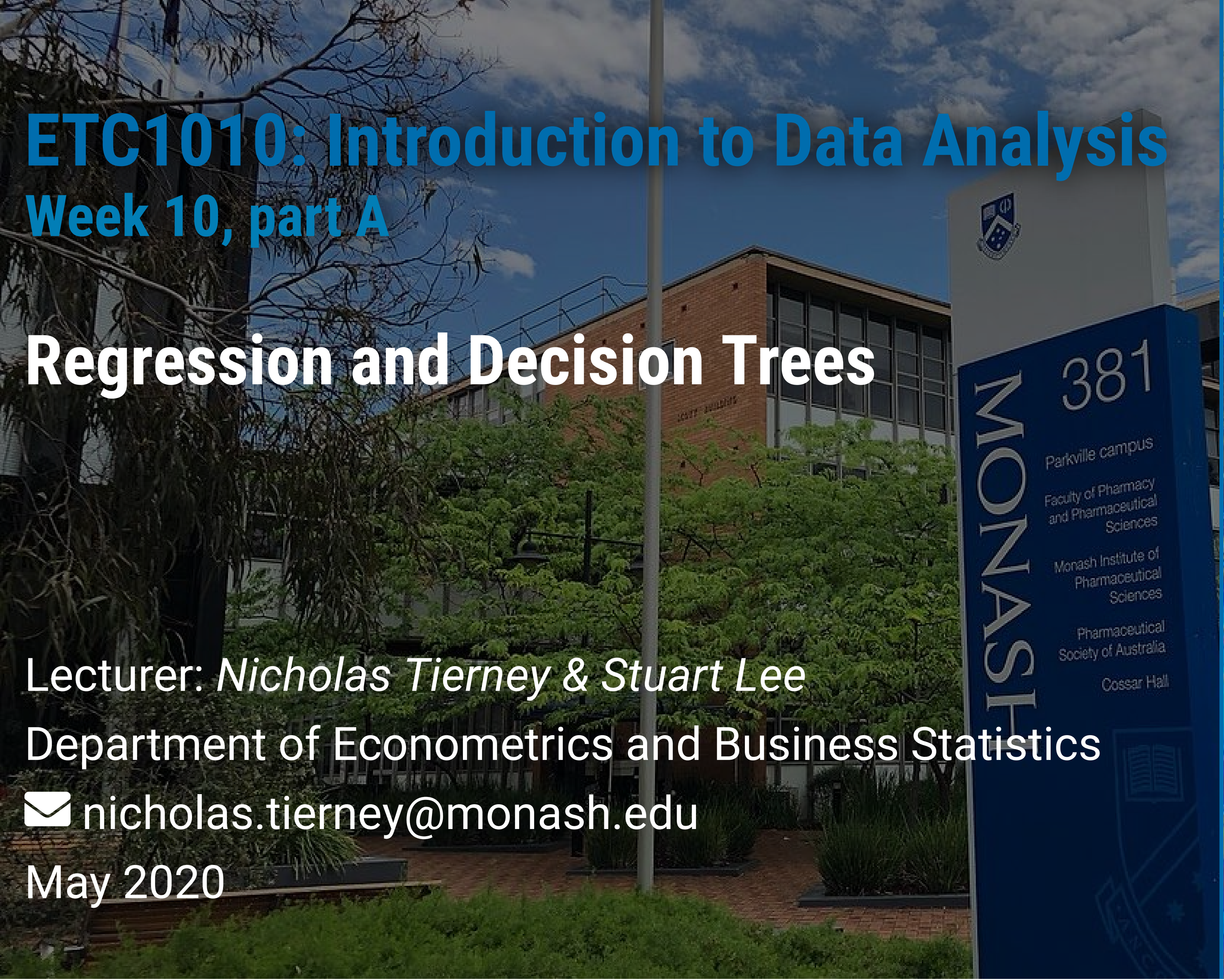

MONASH
381
Parkville campus
Faculty of Pharmacy and Pharmaceutical Sciences
Monash Institute of Pharmaceutical Sciences
Pharmaceutical Society of Australia
Cossar Hall

ETC1010: Introduction to Data Analysis

Week 10, part A

Regression and Decision Trees

Lecturer: *Nicholas Tierney & Stuart Lee*

Department of Econometrics and Business Statistics

nicholas.tierney@monash.edu

May 2020

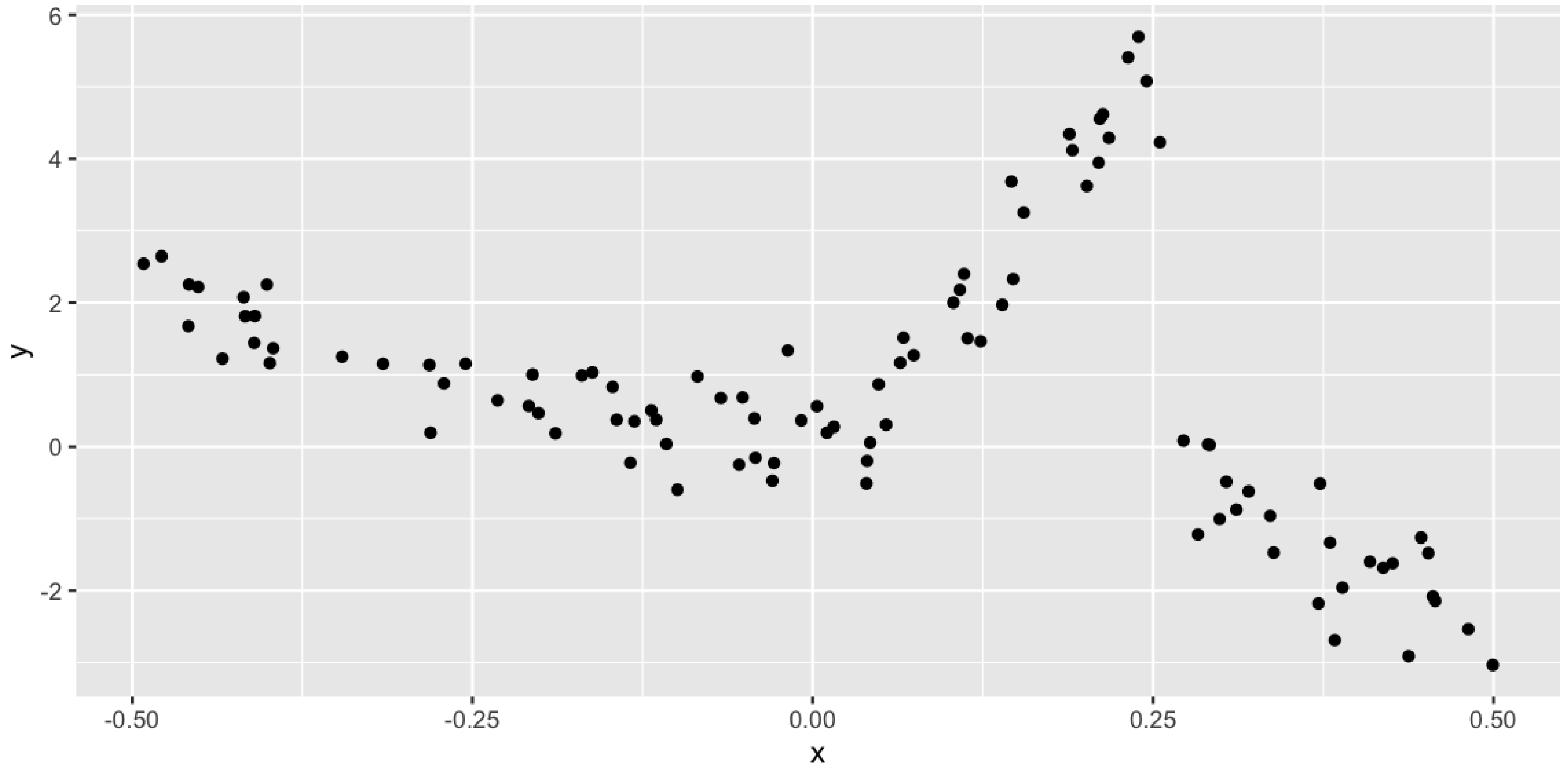
recap

- networks

Overview

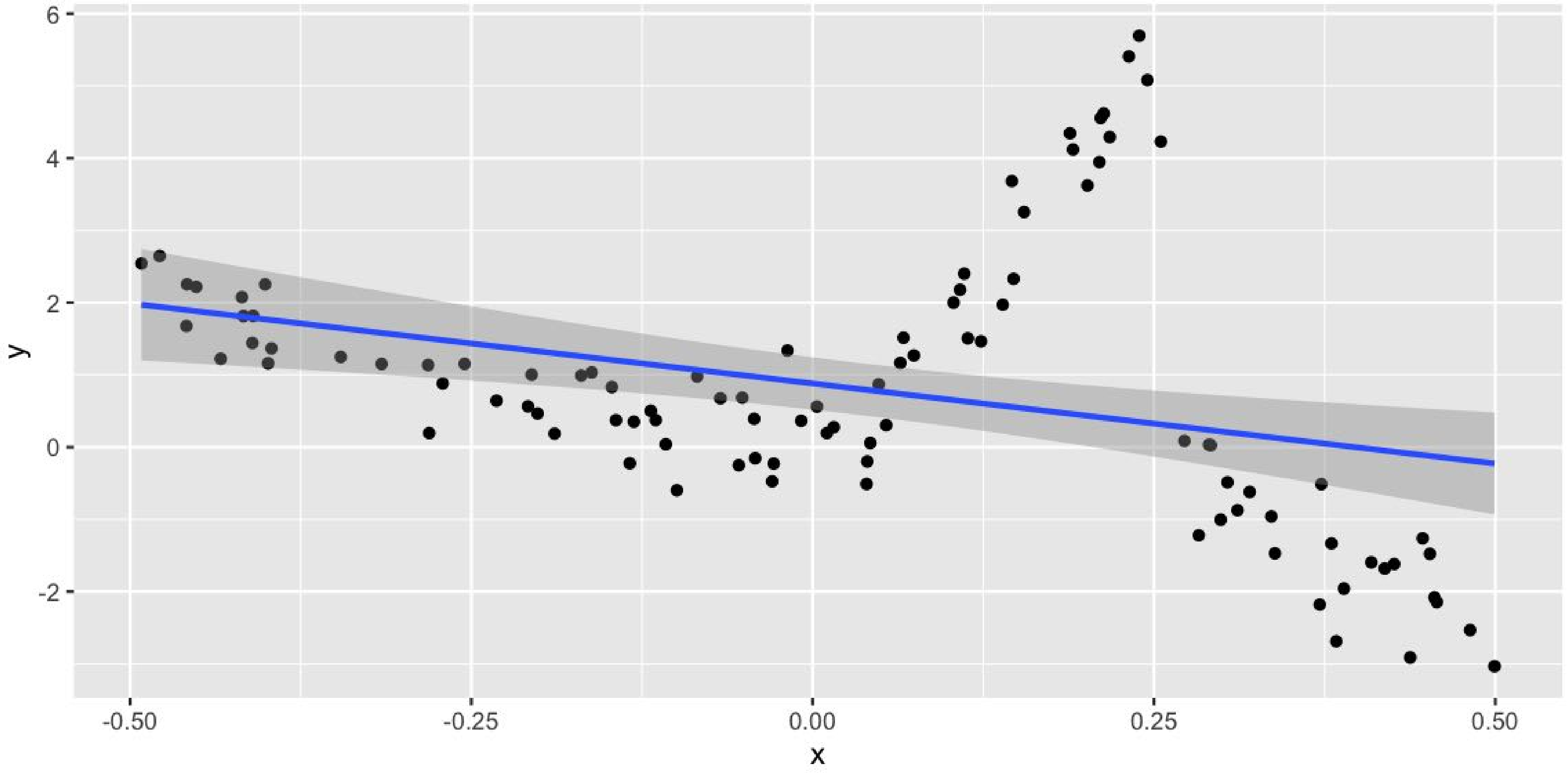
- What is a regression tree?
- How is it computed?
- Deciding when its a good fit
  - rmse
- Comparison with linear models
- Using multiple variables
- Next class:
  - How a classification tree differs from a regression tree?

Example



Let's predict Y using a linear model

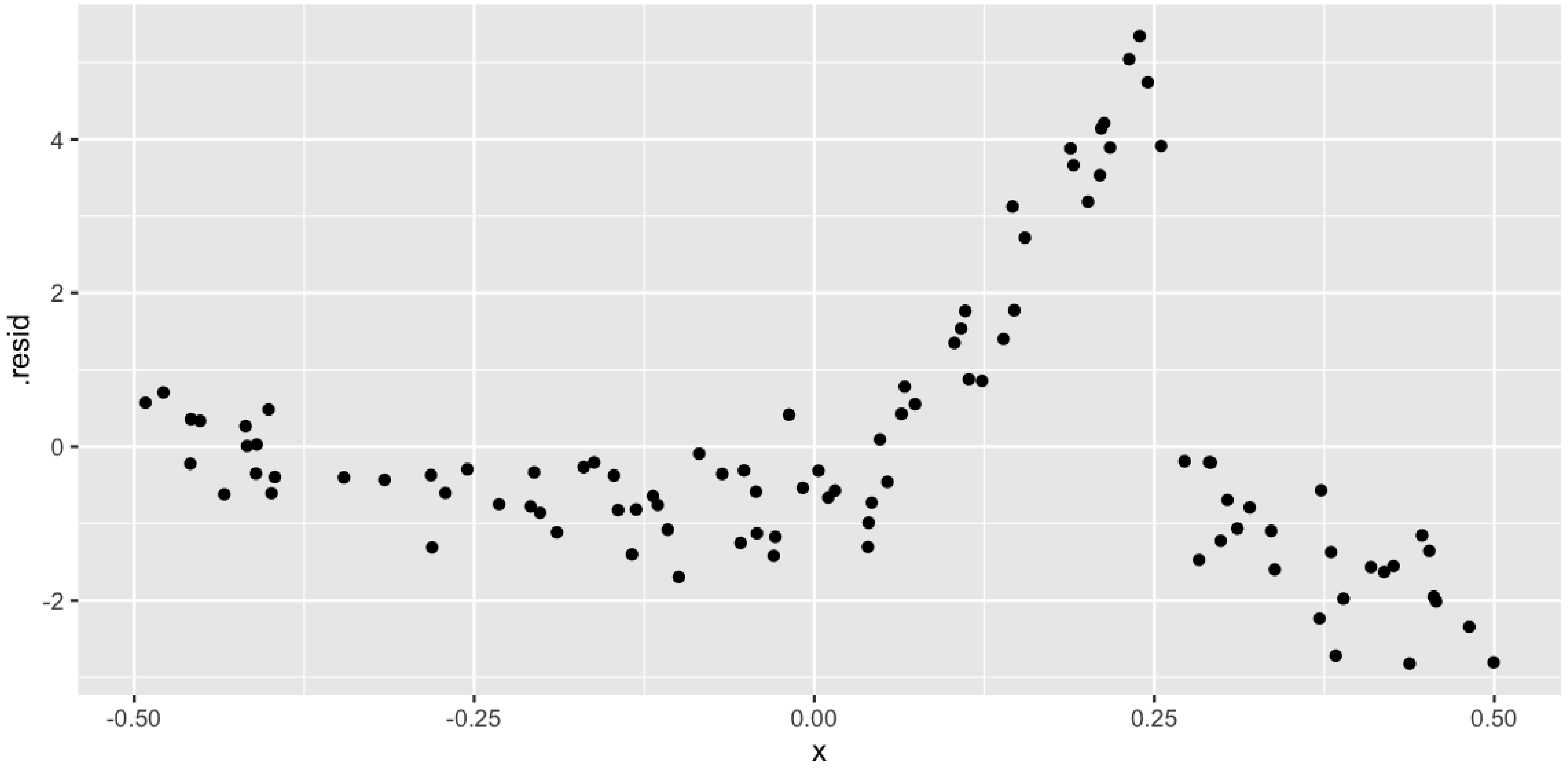
```
df_lm <- lm(y ~ x, df)
```



Assessing model fit

- Look at residuals
- Look at mean square error

Looking at the residuals: this is bad!



It basically looks like the data!

Looking at the Mean square error (MSE)

This is another way to assess a model, it is like taking the average amount of error in the model.

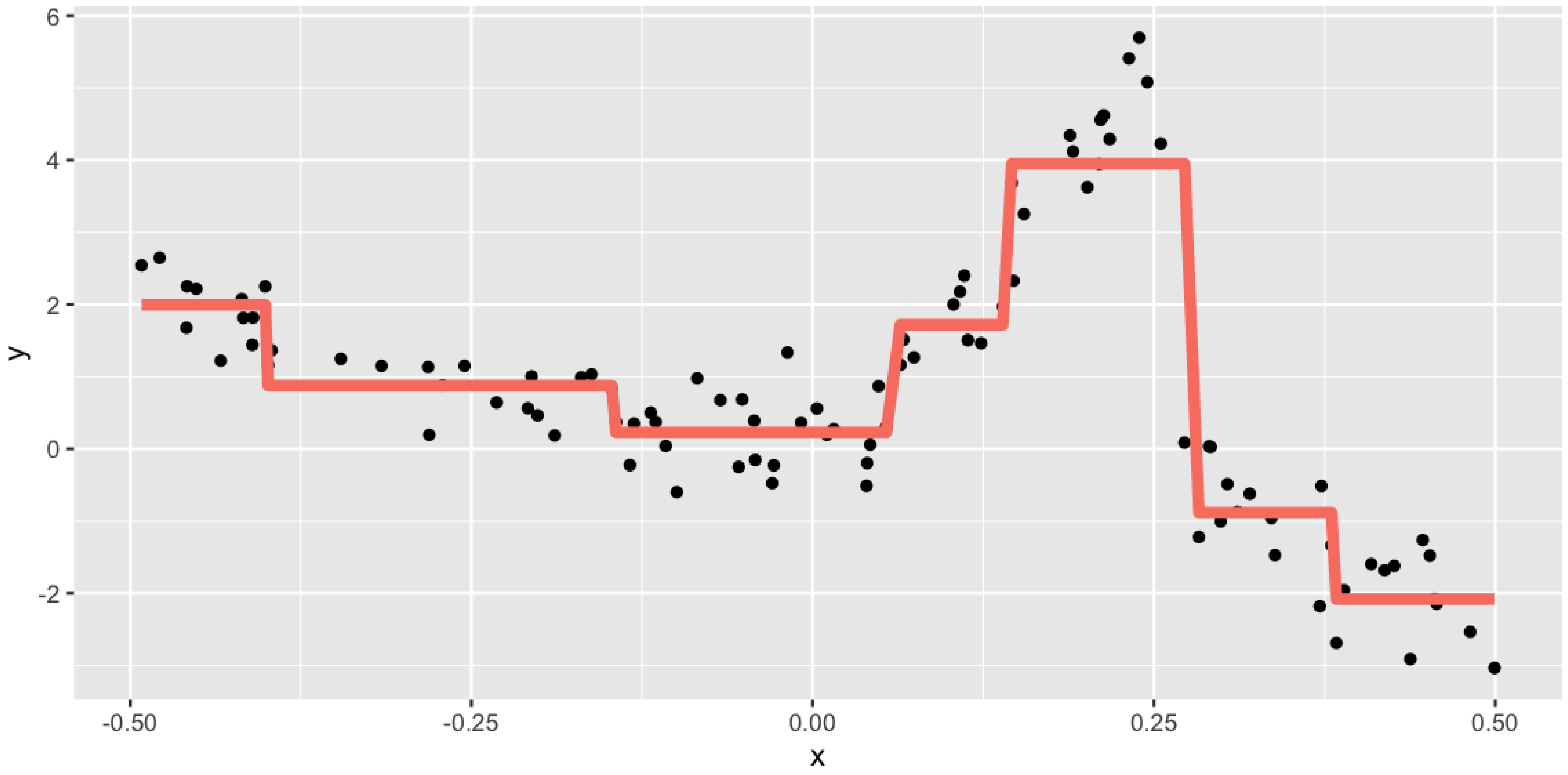
$$MSE(y) = \frac{\sum_{i=1}^{i=N}(y_i - \hat{y}_i)^2}{N}$$

In R code:

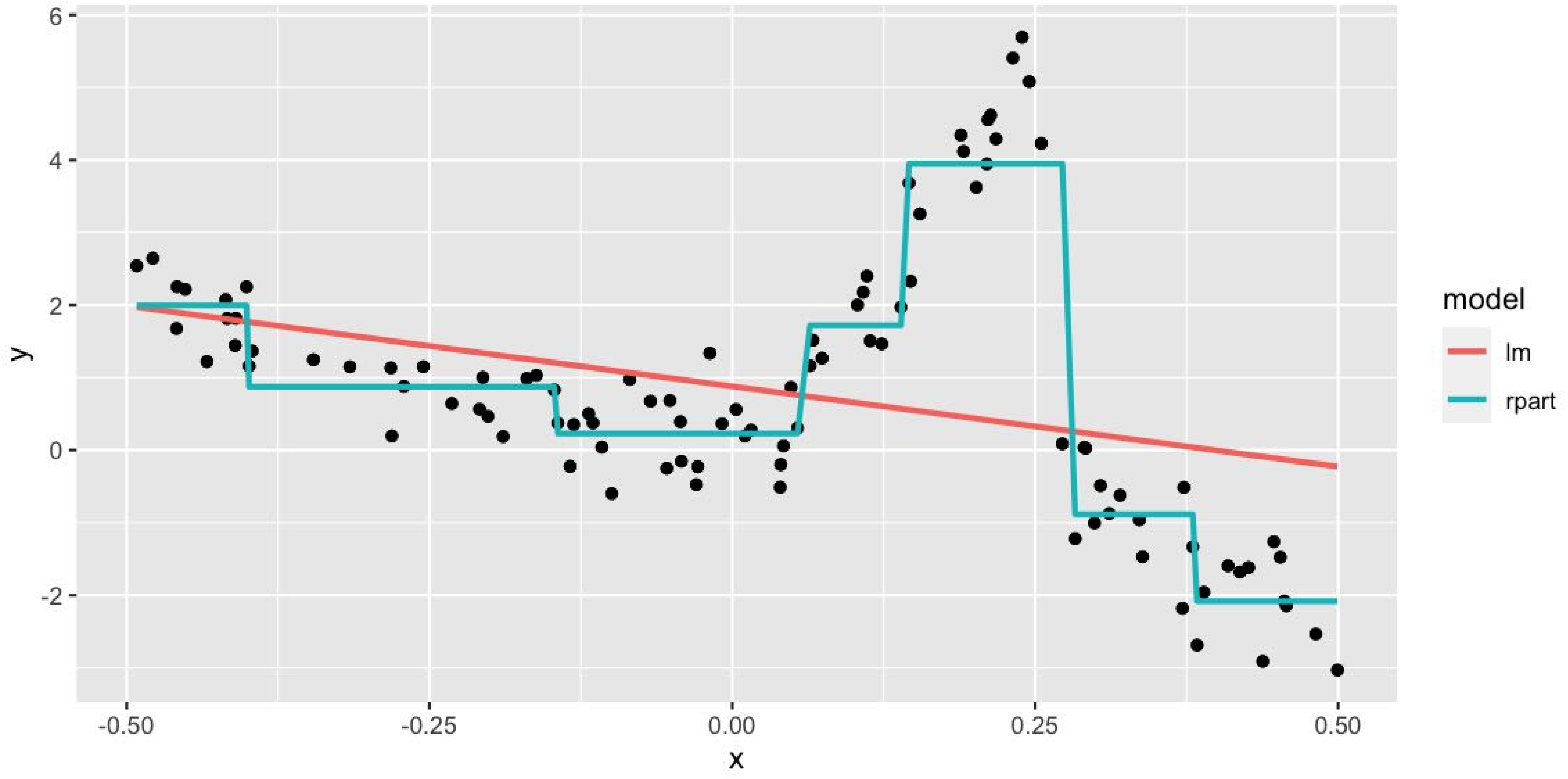
```
mse <- function(model){
  mod_aug <- augment(model)
  mod_aug %>%
    mutate(res_2 = .resid^2) %>%
    summarise(mse = mean(res_2)) %>%
    pull(mse)
}

mse(df_lm)
## [1] 3.216767
```

Let's use a different model: "rpart"

```r
library(rpart)
# df_lm <- lm(y~x, data=df) - similar to lm! But rpart.
df_rp <- rpart(y~x, data=df)
```



Comparing lm vs rpart: Predictions



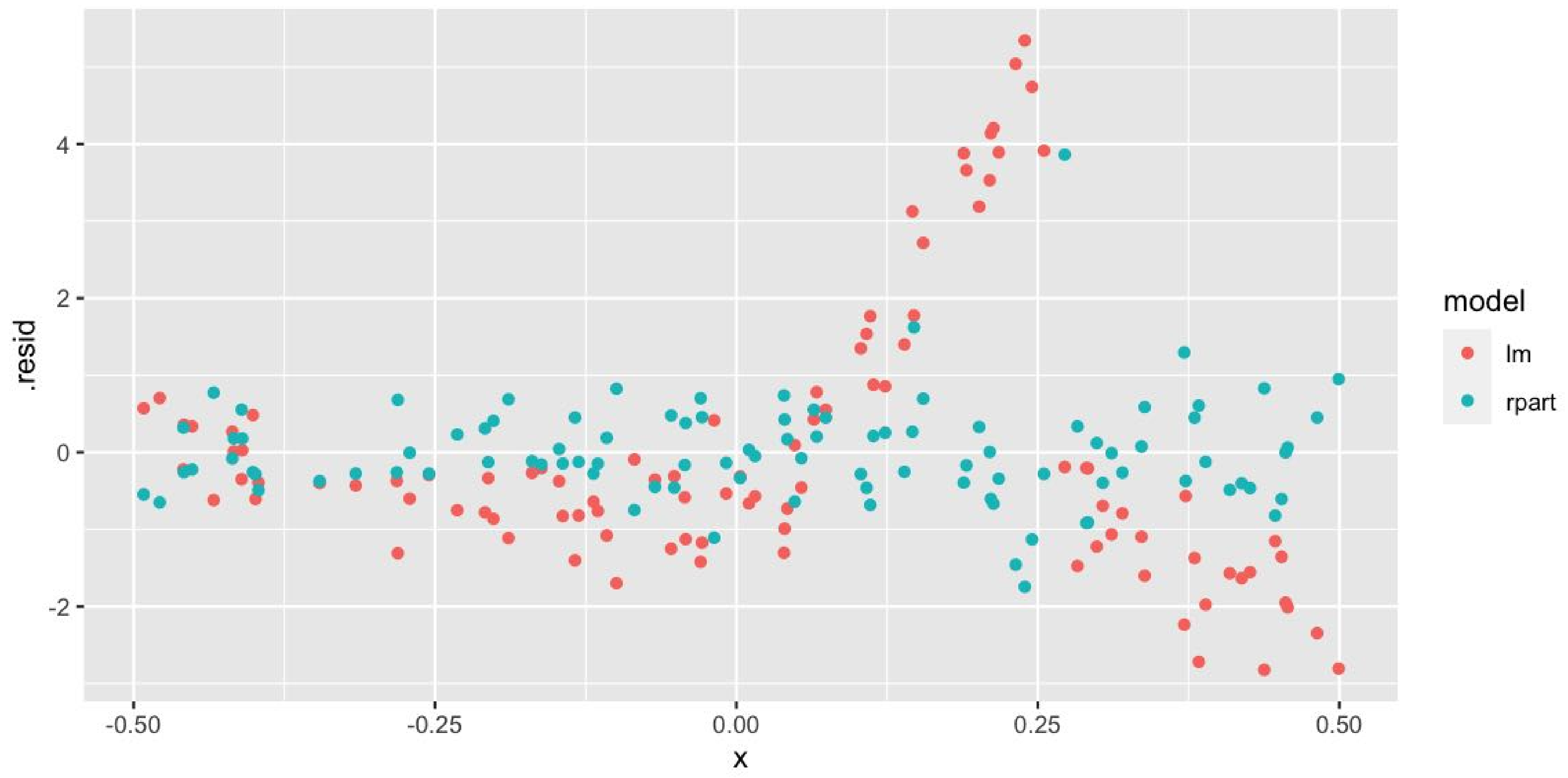
Comparing lm vs rpart: MSE

```
# linear model
mse(df_lm)
## [1] 3.216767

# rpart model
mse(df_rp)
## [1] 0.4517498
```

The rpart model is much lower!

We can look at the residuals plotted against the values of x to get an idea

Comparing lm vs rpart: residuals



Comparing lm vs rpart: output

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Coefficients:
## (Intercept)            x
##      0.8806      -2.2165
```

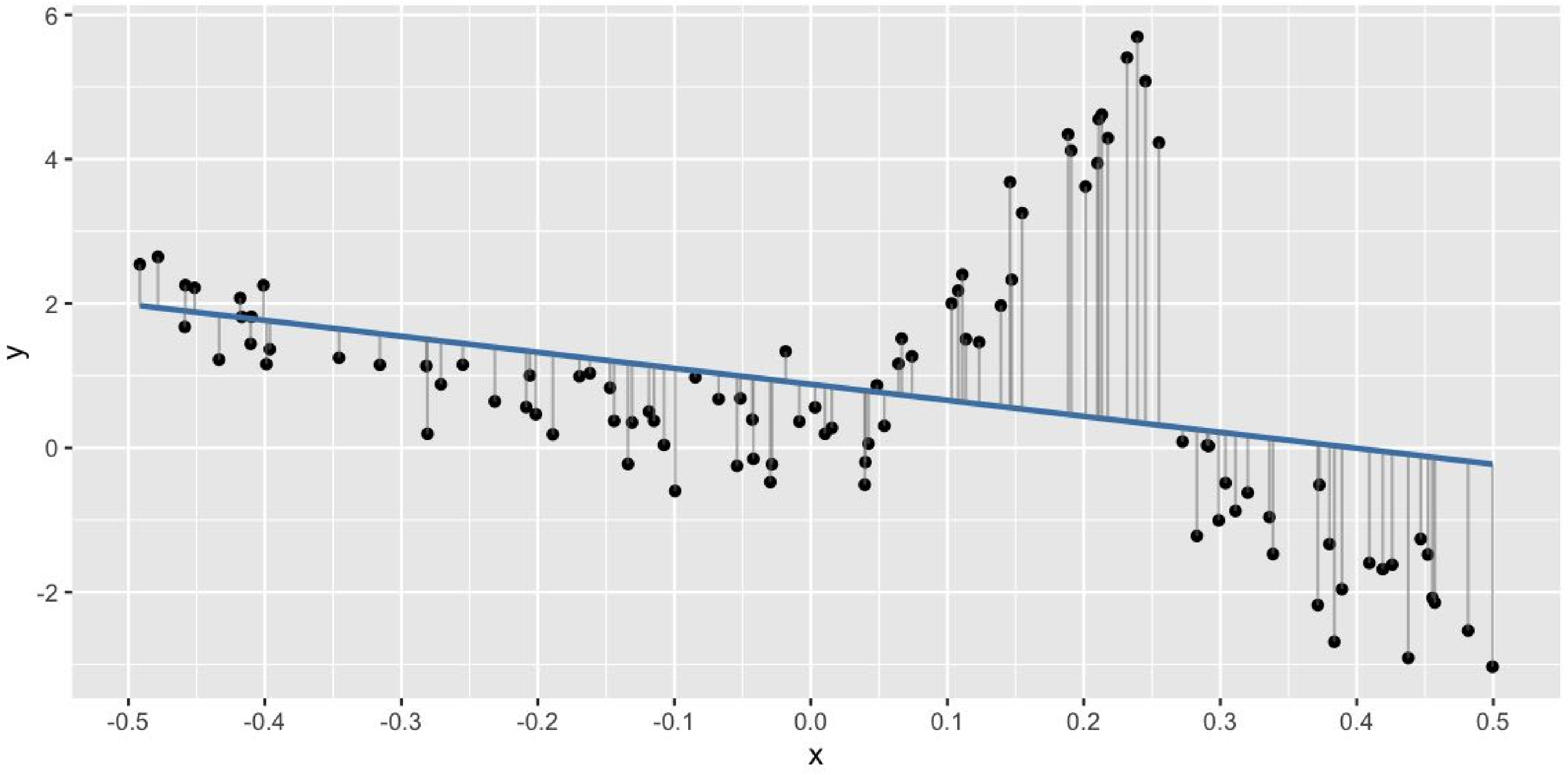
Comparing lm vs rpart: output

```
## n= 100
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 100 359.245100  0.8081071
##    2) x>=0.2775916 24  16.840100 -1.4822830
##      4) x>=0.3817438 12   3.832238 -2.0814410 *
##      5) x< 0.3817438 12   4.392090 -0.8831252 *
##    3) x< 0.2775916 76 176.745400  1.5313880
##      6) x< 0.1426085 61  41.562800  0.9365995
##       12) x>=-0.3999242 50  24.519860  0.7035330
##         24) x< 0.05905847 41  11.729940  0.4807175
##           48) x>=-0.1455513 25   5.653876  0.2281914 *
##           49) x< -0.1455513 16   1.990829  0.8752895 *
##         25) x>=0.05905847 9   1.481498  1.7185820 *
##       13) x< -0.3999242 11   1.981477  1.9959930 *
##      7) x>=0.1426085 15  25.842970  3.9501960 *
```

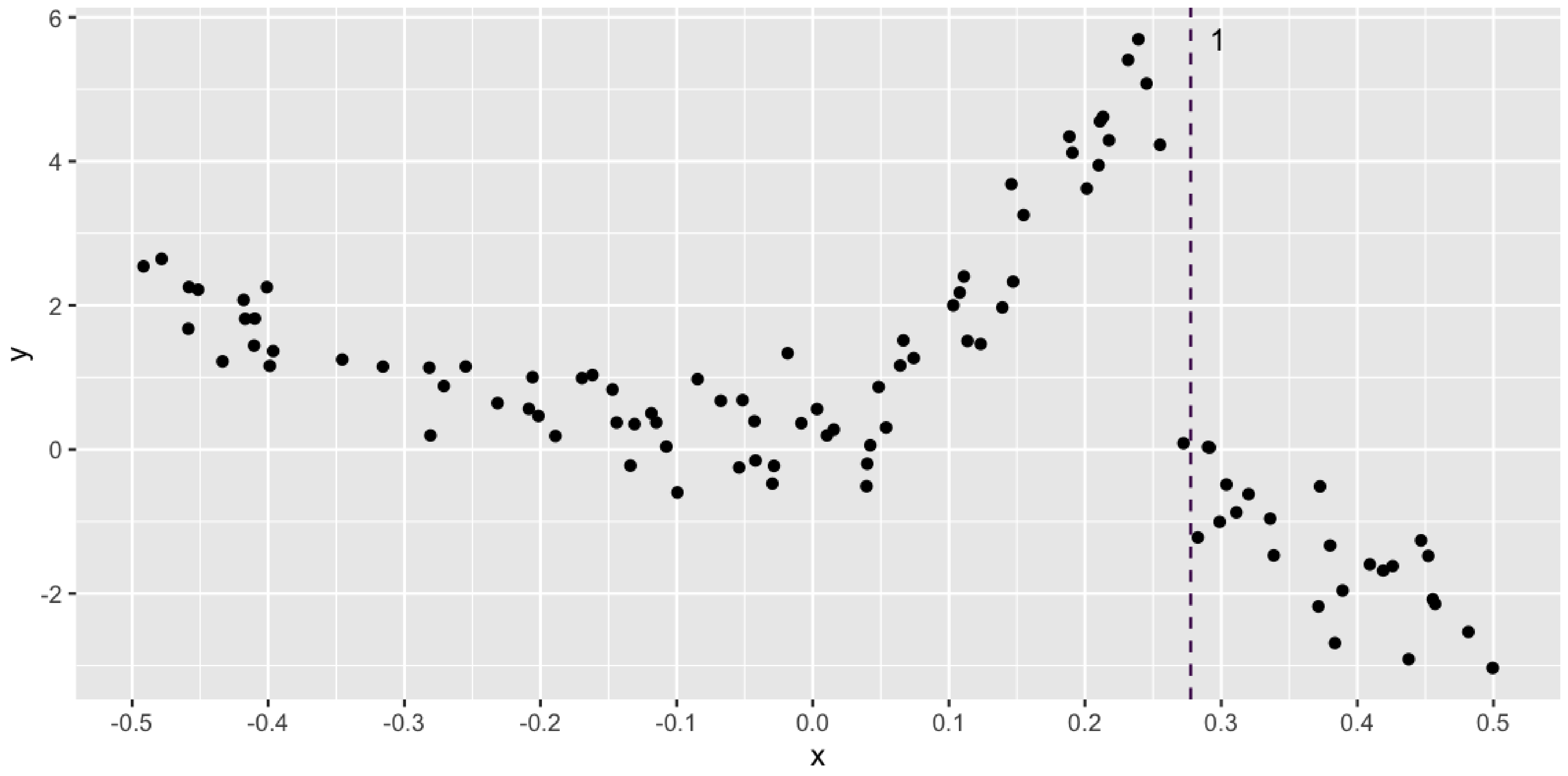
So what is going on?

- A linear model asks "What line fits through these points, to minimise the error"?
- A decision tree model asks "How can I best break the data into segments, to minimize some error?"

A linear model: draws the line of best fit



A regression tree: segments the data to reduce mean error



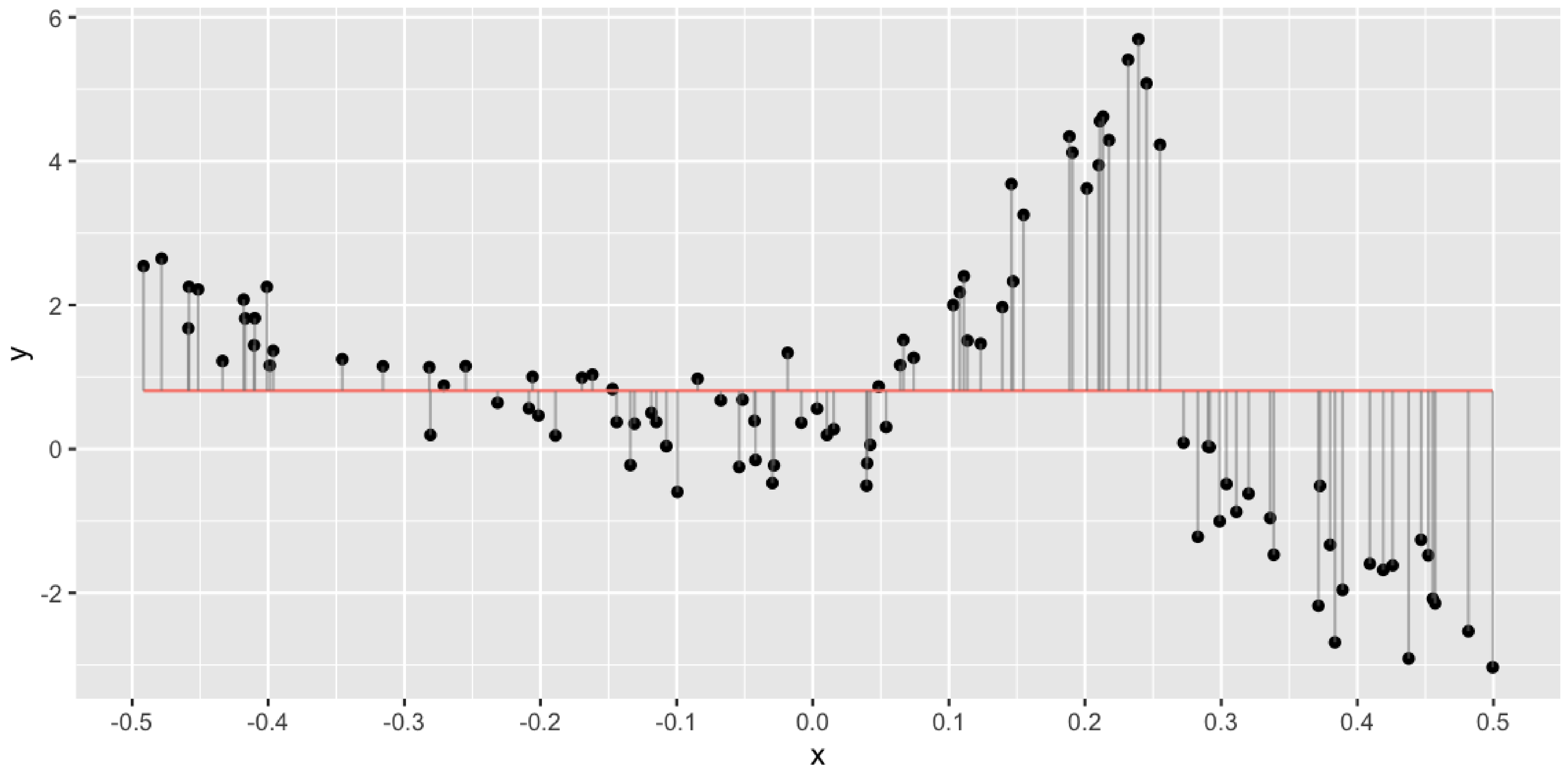
Regression trees

- Regression trees recursively partition the data, and use the average response value of each partition as the model estimate
- It is a computationally intense technique that examines all possible partitions, and choosing the BEST partition by optimizing some criteria
- For regression, with a quantitative response variable, the criteria to maximise is called ANOVA:

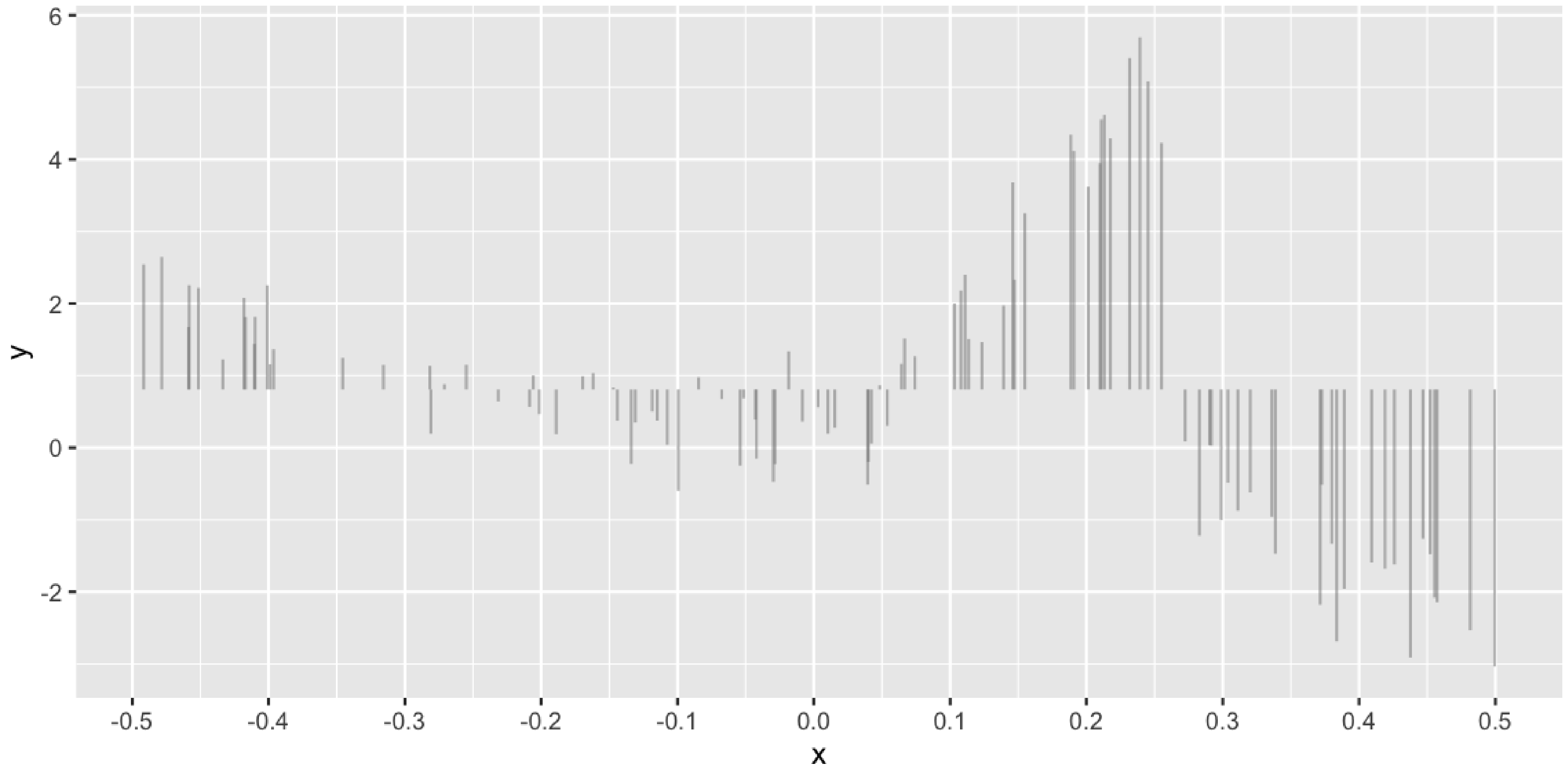
$$SS_T - (SS_L + SS_R)$$

where $SS_T = \sum(y_i - \bar{y})^2$, and SS_L, SS_R are the equivalent values for the two subsets created by partitioning.

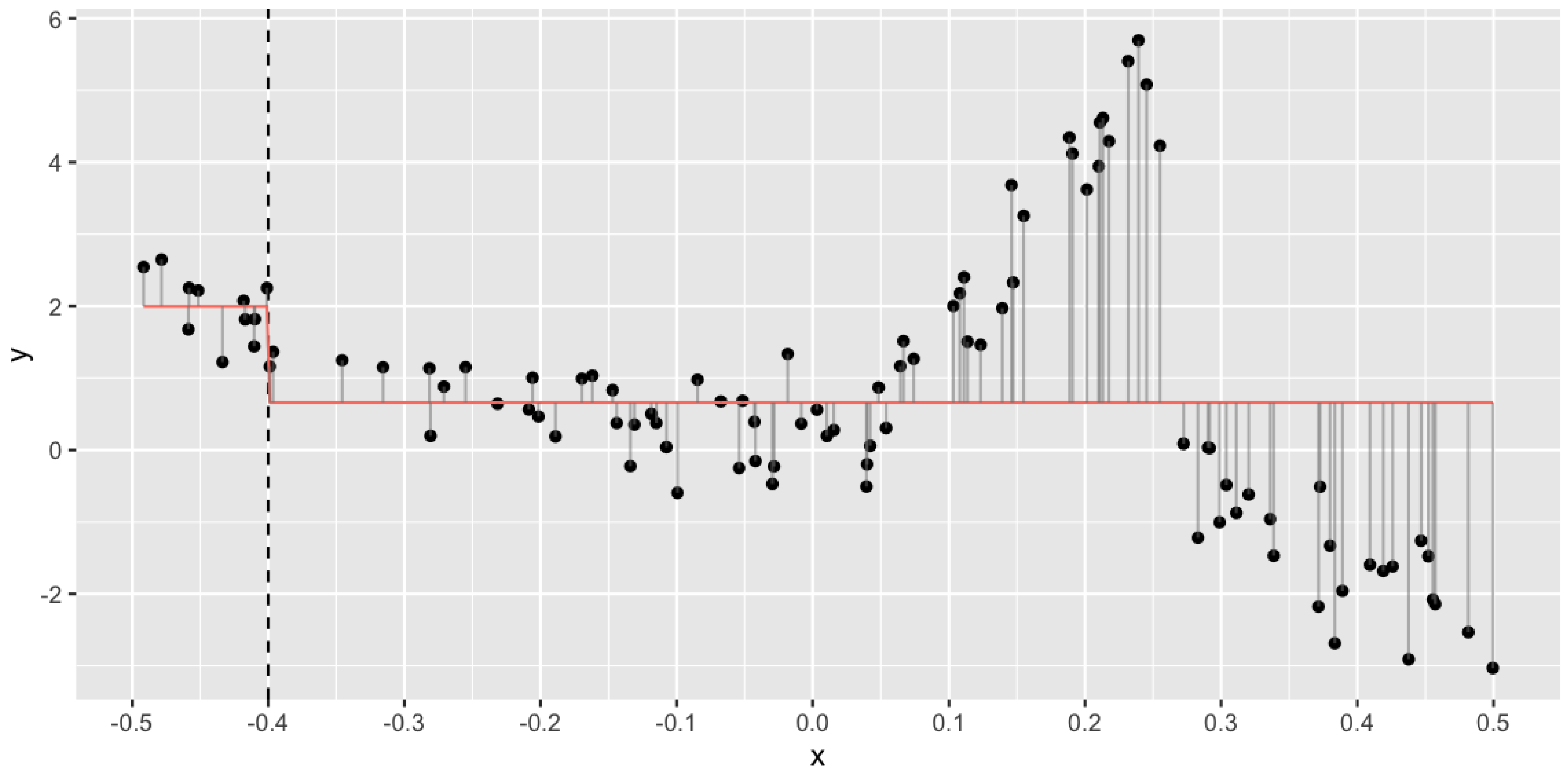
Break down: What is $SS_T = \sum(y_i - \bar{y})^2$?



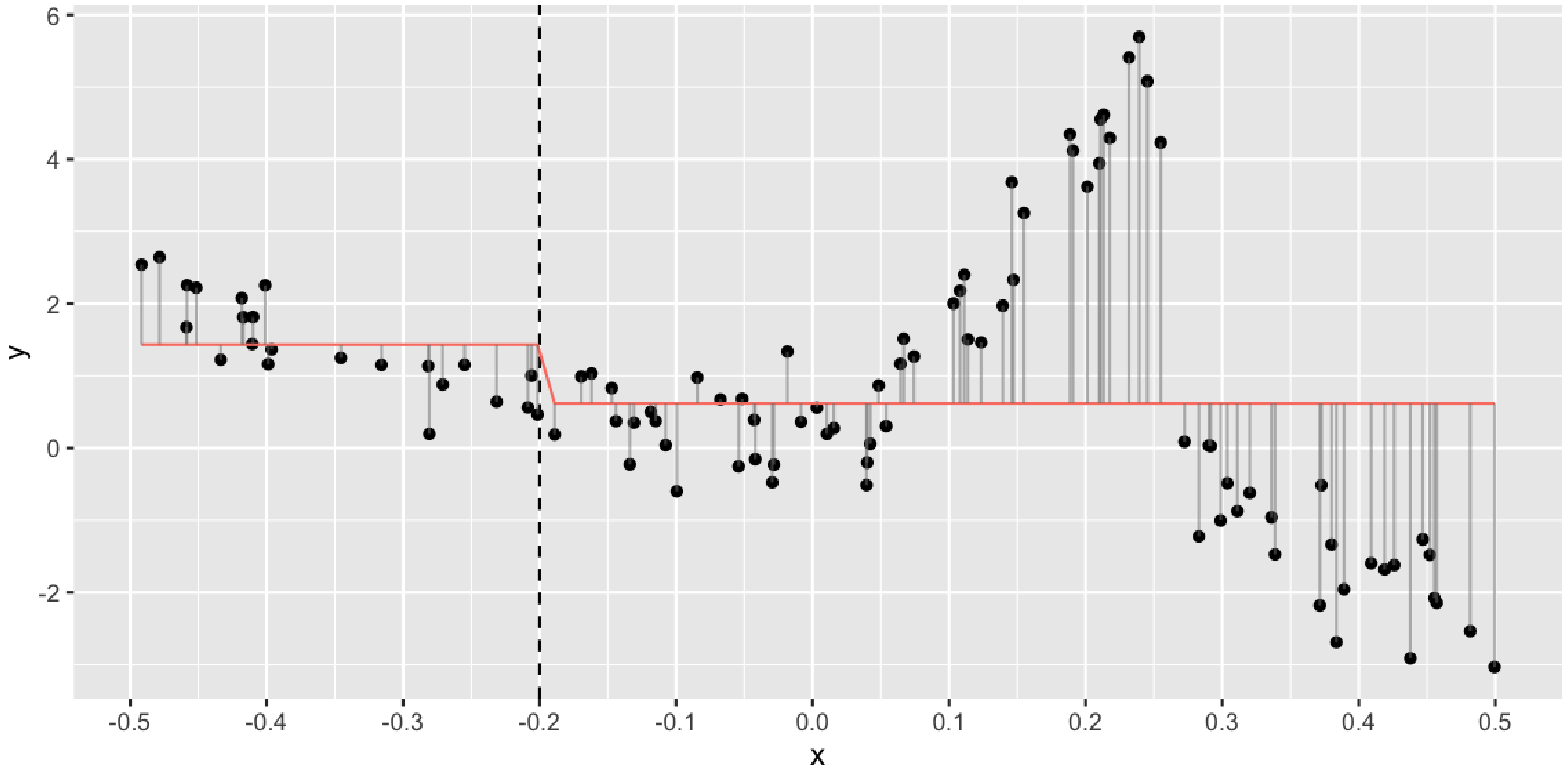
Break down: What is $SS_T = \sum(y_i - \bar{y})^2$?



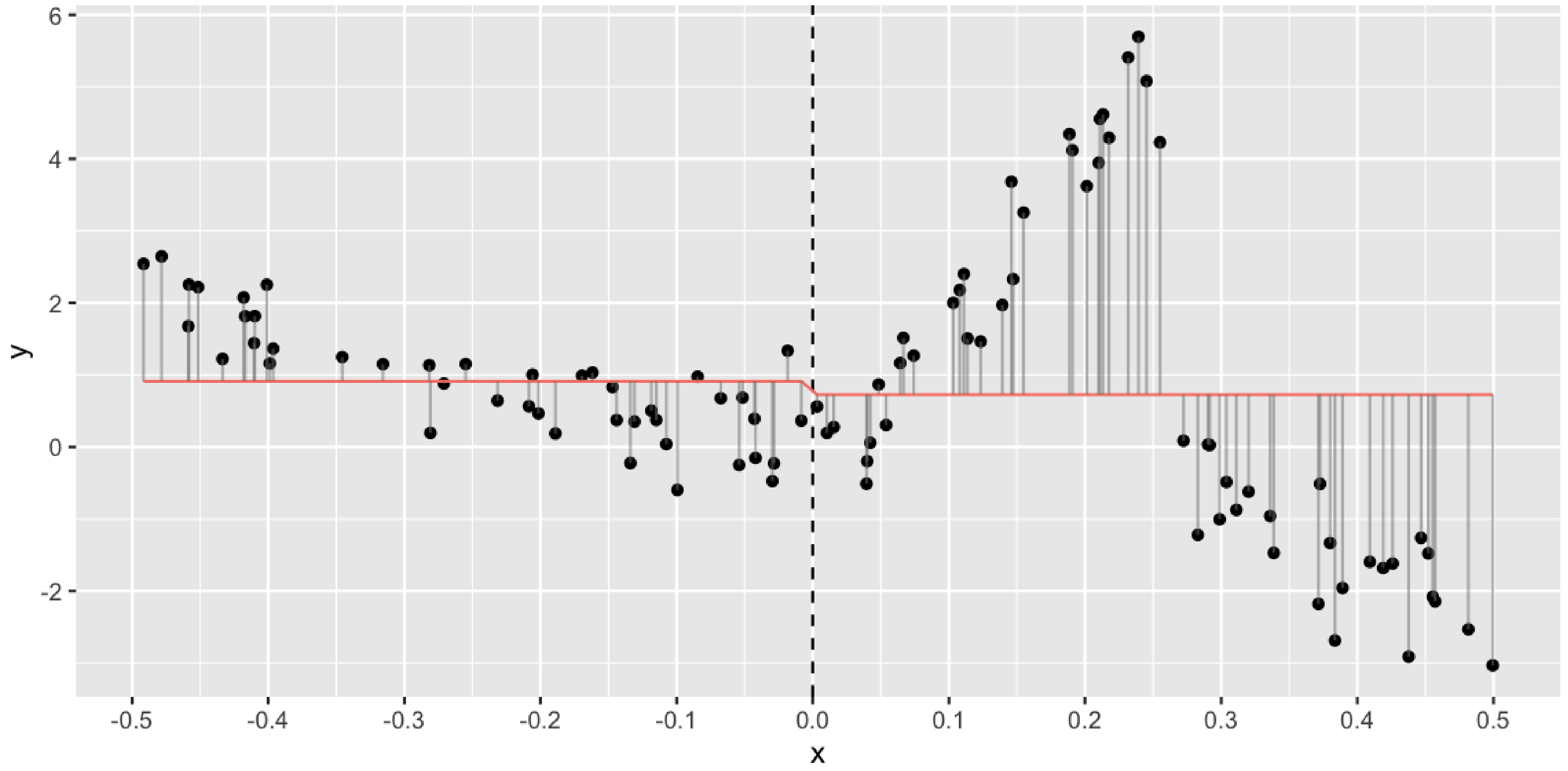
SS_L SS_R? Choose a point, compare the left and right



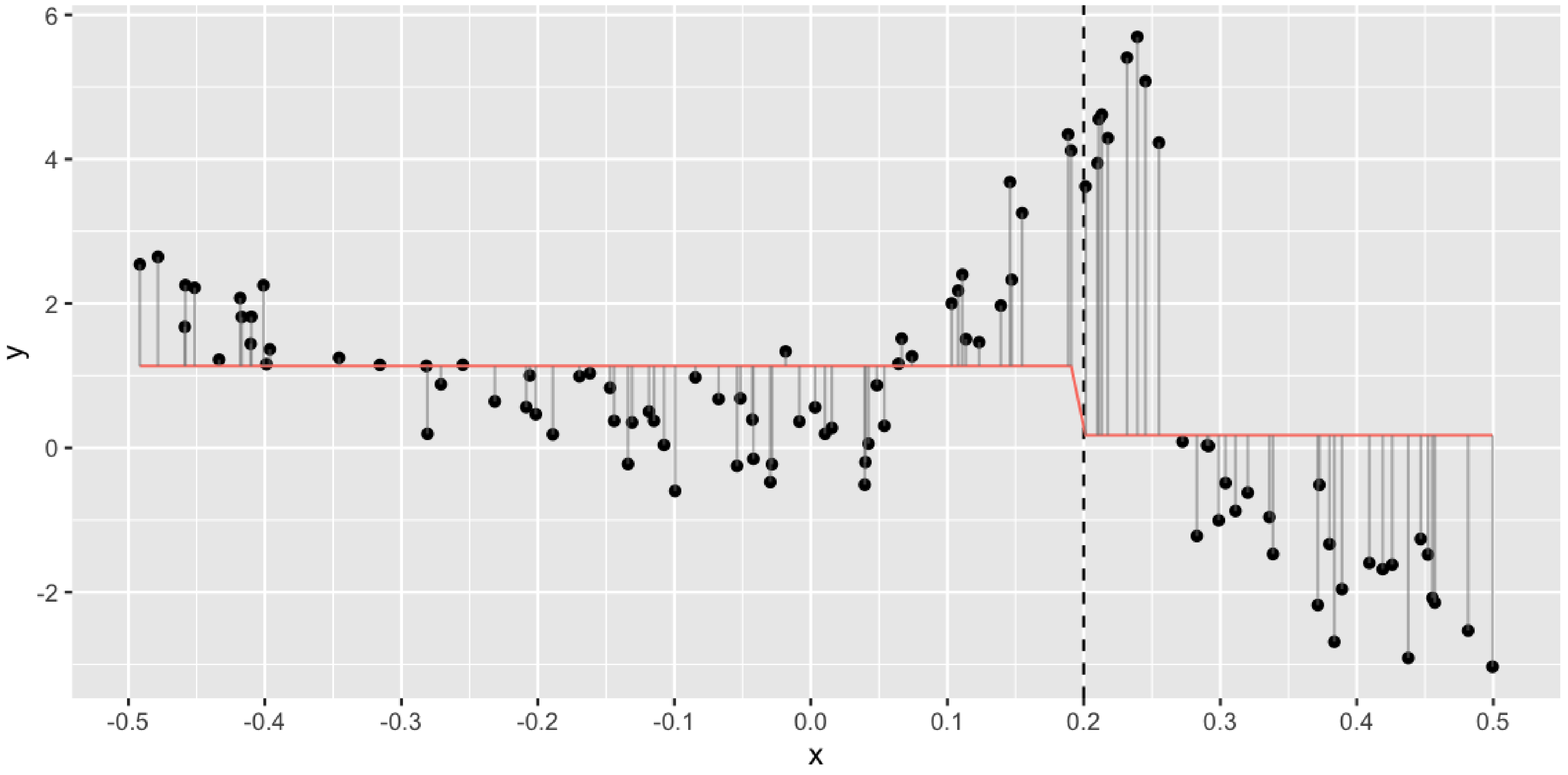
SS_L SS_R? Choose a point, compare the left and right



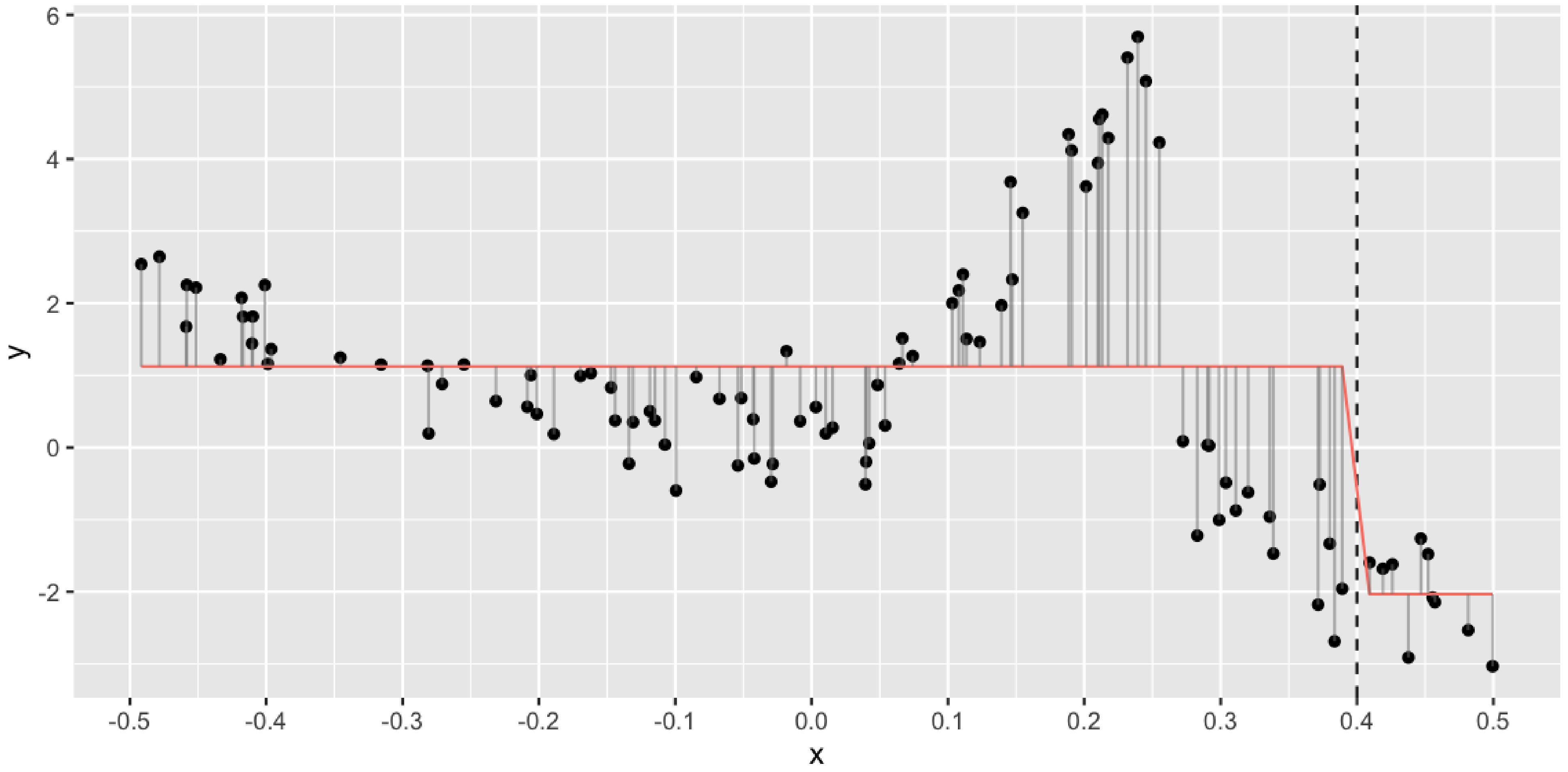
SS_L SS_R? Choose a point, compare the left and right



SS_L SS_R? Choose a point, compare the left and right



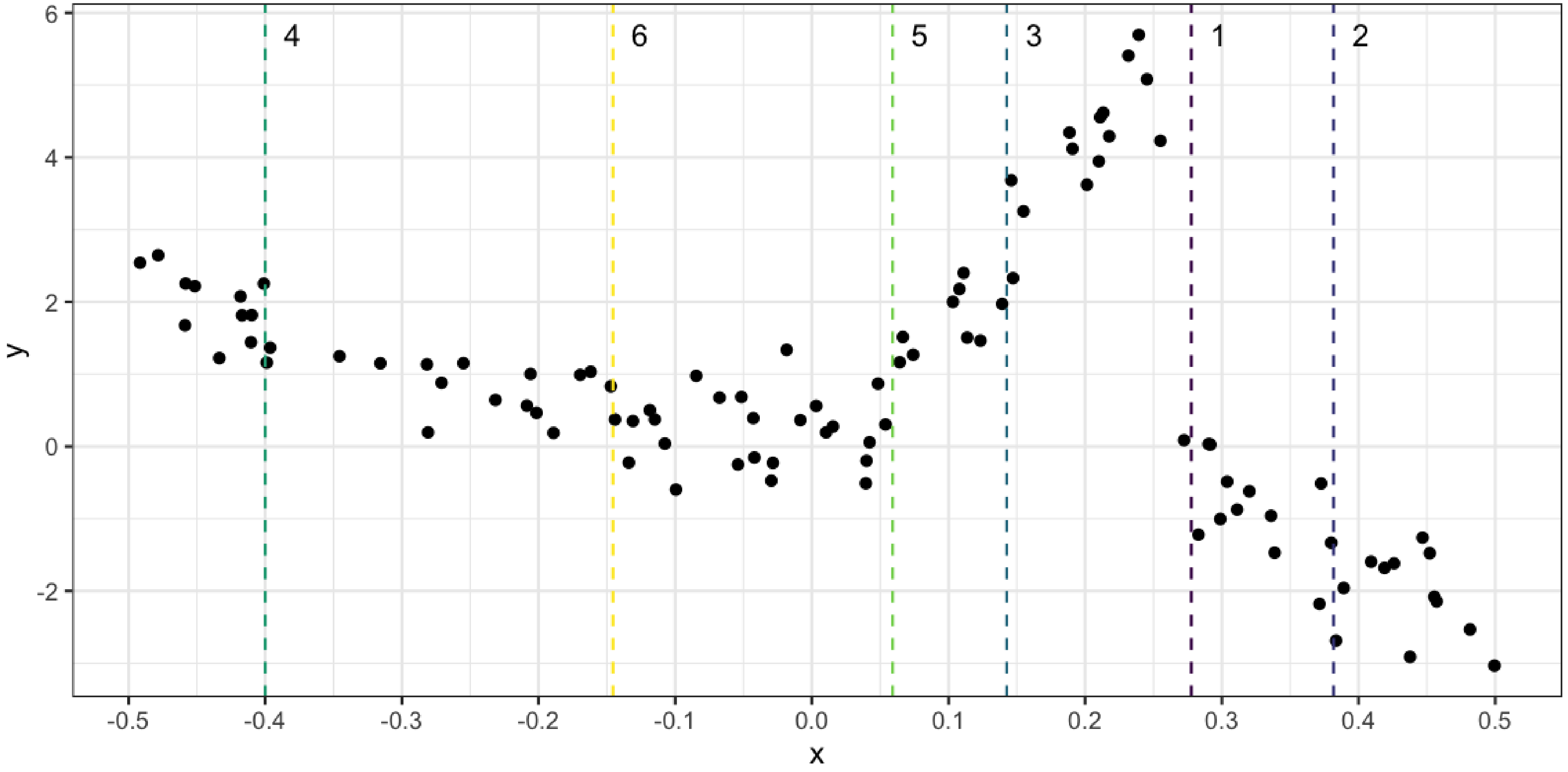
SS_L SS_R? Choose a point, compare the left and right



Across all values of x?

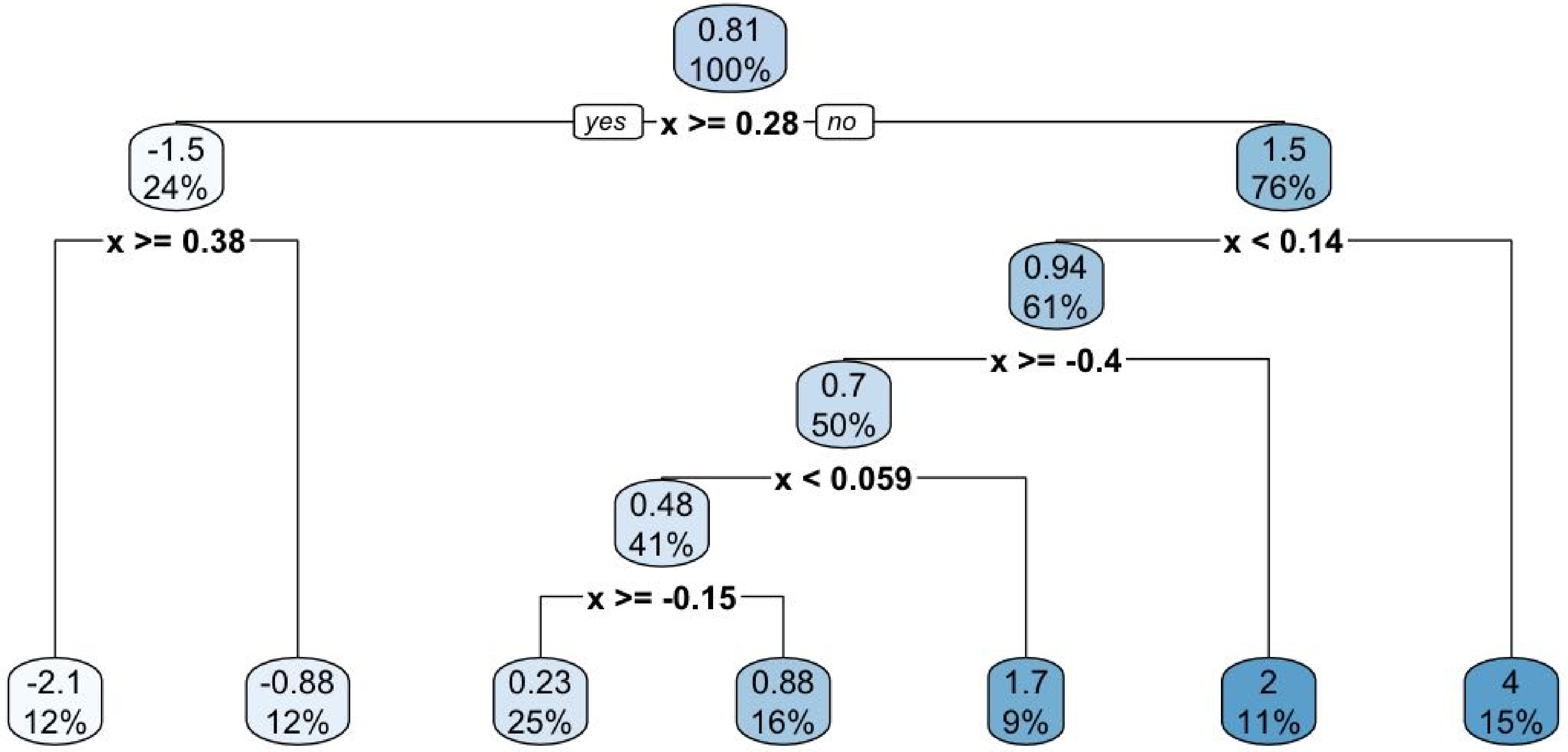
And if we repeated this within each split

This is how the data is split:

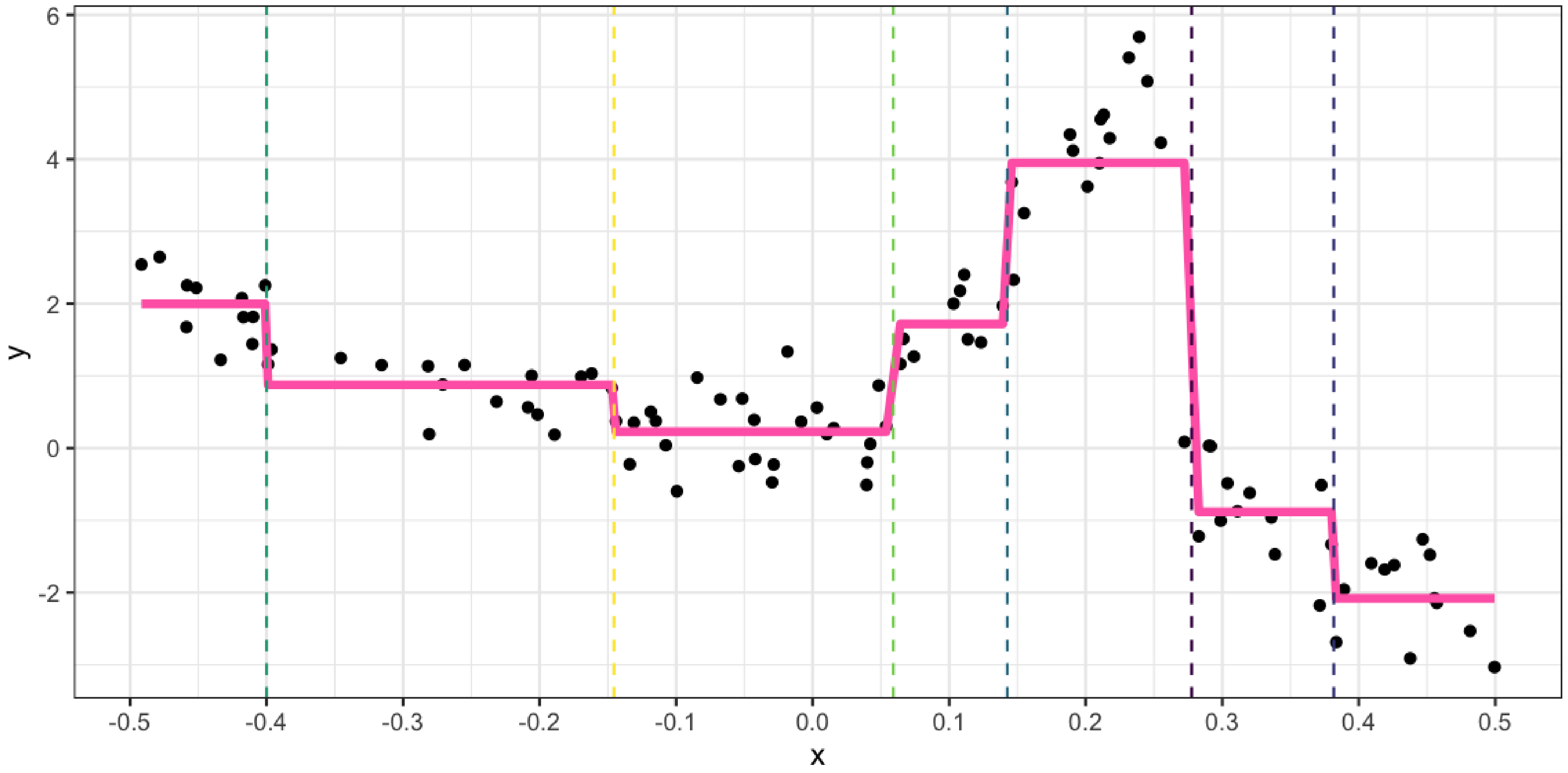


We can represent these splits in a tree format:

```r
library(rpart.plot)
rpart.plot(df_rp)
```



The model predictions with the splits



Your turn: compute a regression tree

Using the small data set, manually compute a regression tree model for the data. Sketch the model.

```r
d <- tibble(x=c(1, 2, 3, 4, 5), y=c(10, 12, 5, 4, 3))
d
ggplot(d, aes(x=x, y=y)) +
  geom_???()
```

Stopping rules

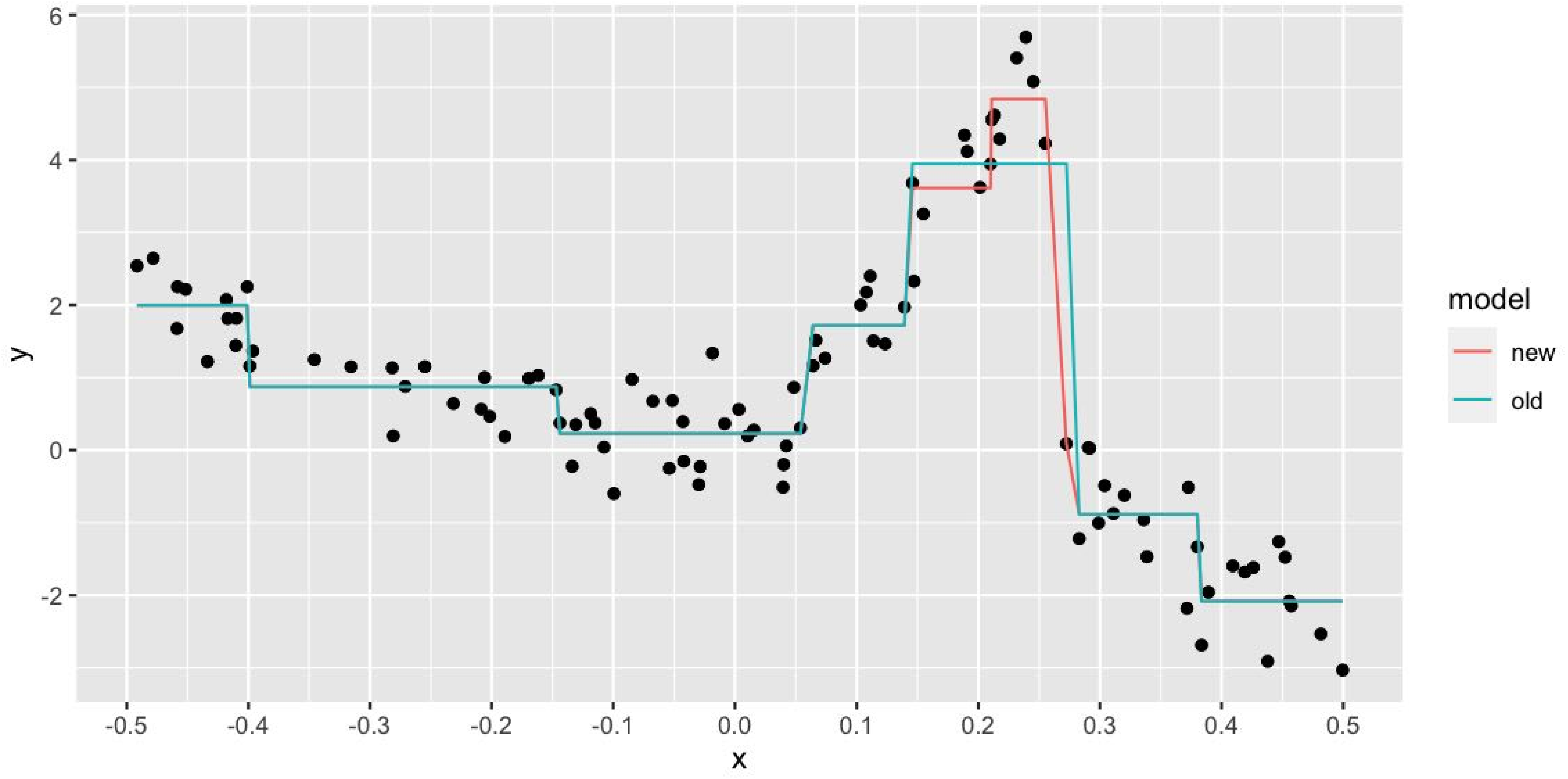
- Its an algorithm, and it has to know when to stop.
- Why did it stop at 7 terminal nodes?
- Stopping rules are needed, else the algorithm will keep fitting until every observation is in its own group.
- Control parameters set stopping points:
  - **minsplit**: minimum number of points in a node that algorithm is allowed to split
  - **minbucket**: minimum number of points in a terminal node
- We can also look at the change in value of $SS_T - (SS_L + SS_R)$ at each split, and if the change is too *small*, stop.

You can change the options to fit a different model

An re-fit, the model will change. Here we reduce the `minbucket` parameter.

```
df_rp_m10 <- rpart(y~x, data=df,
                                control = rpart.control(minsplit = 2))
```

This yields a (slightly) more complex model.



Beyond one variable

- So far we have only considered cases with one explanatory variable:

```
rpart(y ~ x)
```

- When given multiple variables, a decision tree will **only use variables that provide the best splits**
- This means that we can identify variables that are important for predicting an outcome.
- This is called "Variable importance"

Variable importance

- After calculating all the potential splits, each variable is given an importance value, that is typically based on the number of times it was used in splitting, and the order in the splits
- The earlier the split, the more important the variable.
- These "importance values" are usually scaled to sum to 100
- But the numbers themselves are arbitrary
- Let's explore this in the next exercise, "10-exercise-lab-1.Rmd"

Wrapping up

Strengths

- There are no parametric assumptions underlying partitioning methods
- Can handle data of unusual shapes and sizes
- Can identify unusual groups of data
- Provides a tree based graphic that is fun to interpret
- Has an efficient heuristic of handling missing values.
- The method could be influenced by outliers, but it would be isolating them to one partition.

Weaknesses

- Doesn't really handle data that is linear very well
- Can require tuning parameters to get good model fit
- Also means that there is not a nice formula for the model as a result, or inference about populations available
  - E.g., You can't say things like: "For every one unit increase in weight, we expect height to increase by XX amount".

Next class:

- Classification trees