APSTA-GE 2352

Statistical Computing: Lecture 8

Klint Kanopka

New York University

Table of Contents

Announcements

- PS4 due the day before Halloween
  - Very spooky
  - How’s it going?
- If you haven’t already, please do the mid-semester survey
  - https://forms.gle/Ljh25bvYeoAL5dEU8

Check-In

- PollEv.com/klintkanopka

Vector Norms

- We often want to talk about the *length* of a vector, but how do we measure length?
- Conceptually, think of the length of a vector as the distance it spans from the origin
- We'll talk about two types of norms today

The Euclidean Norm

- The Euclidean distance a vector extends in space from the origin
- Often called the L^2 norm and abbreviated $||\vec{x}||_2$

$$||\vec{x}||_2 = \sqrt{\vec{x}^2}$$

$$||\vec{x}||_2 = \sqrt{\vec{x} \cdot \vec{x}}$$

$$||\vec{x}||_2 = \sqrt{\sum_{k=1}^{K} x_k^2}$$

$$||\vec{x}||_2 = \sqrt{x_1^2 + \cdots + x_k^2}$$

The Manhattan Norm

- Sometimes called the *Taxicab Norm*
- The Manhattan distance a vector extends in space from the origin
- Often called the L^1 norm and abbreviated $||\vec{x}||_1$

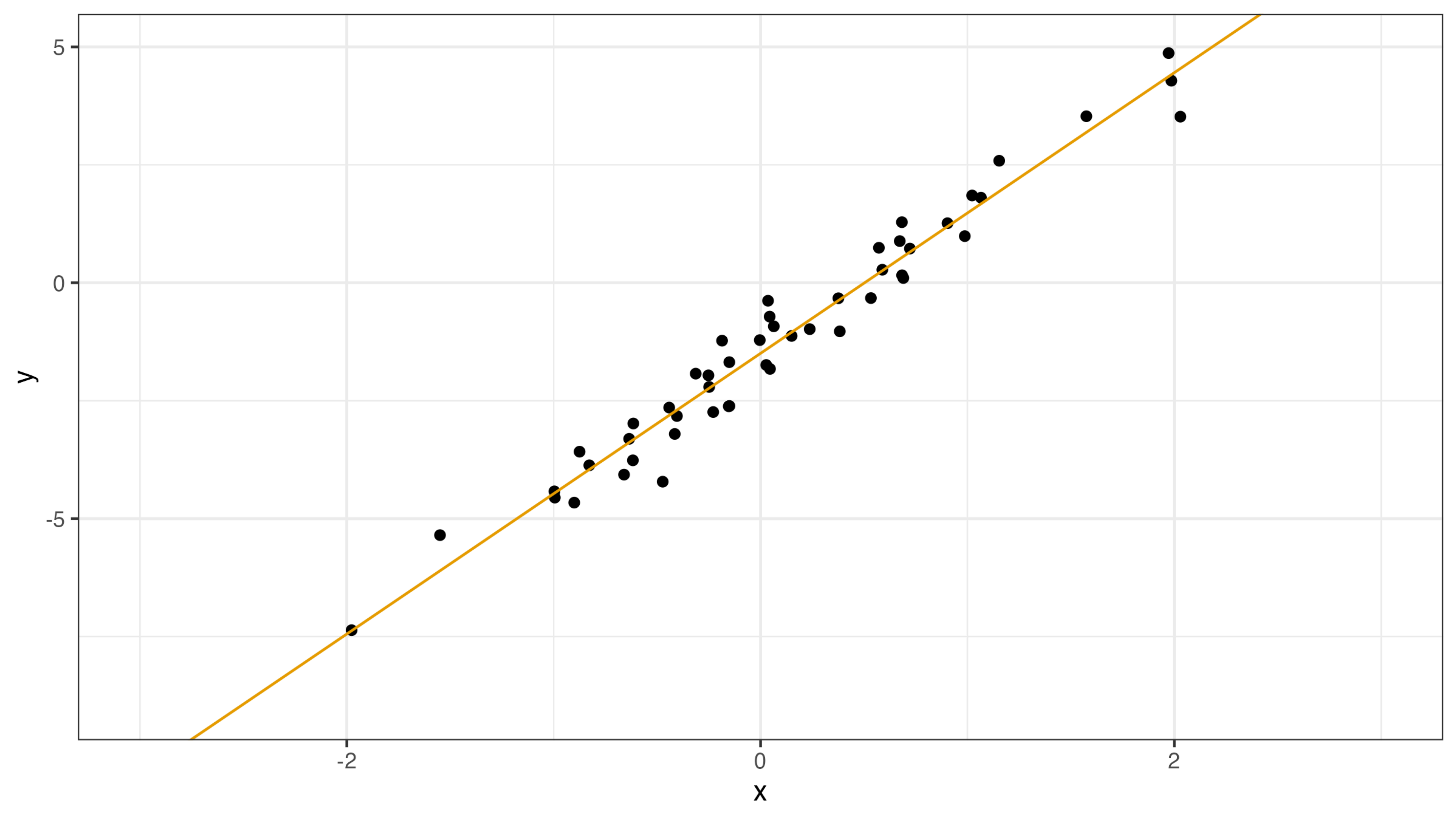
$$||\vec{x}||_1 = \sum_{k=1}^{K} |x_k|$$

$$||\vec{x}||_1 = |x_1| + \cdots + |x_k|$$

Motivating Problem

Overfitting

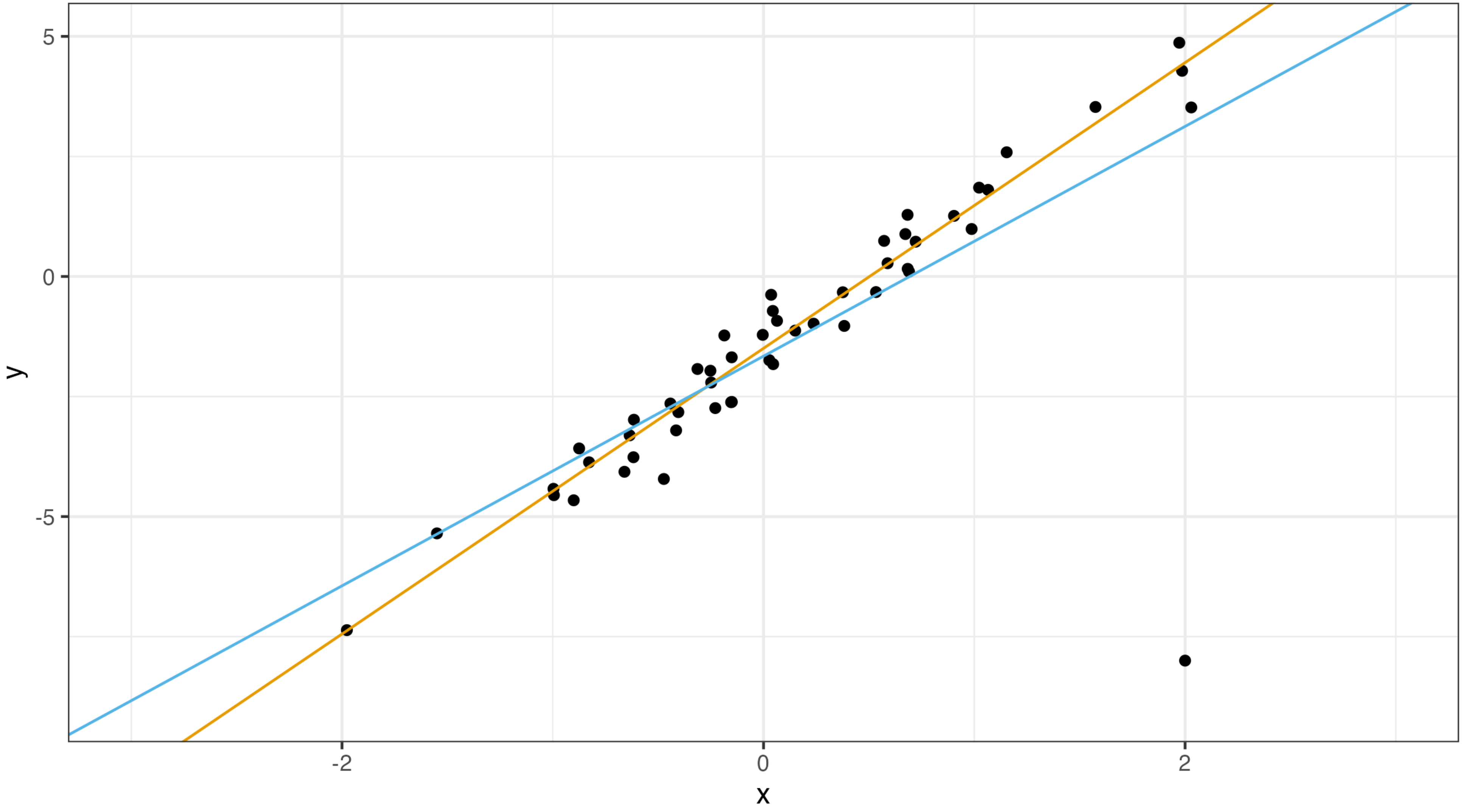
```
d <- data.frame(x = rnorm(50))
d$y <- -1.5 + 3*d$x + rnorm(50, sd=0.5)

m <- lm(y~x, d)
coef(m)

# (Intercept)           x
#  -1.425421    2.883035
```



5
0
-5
y
-2
0
2
x

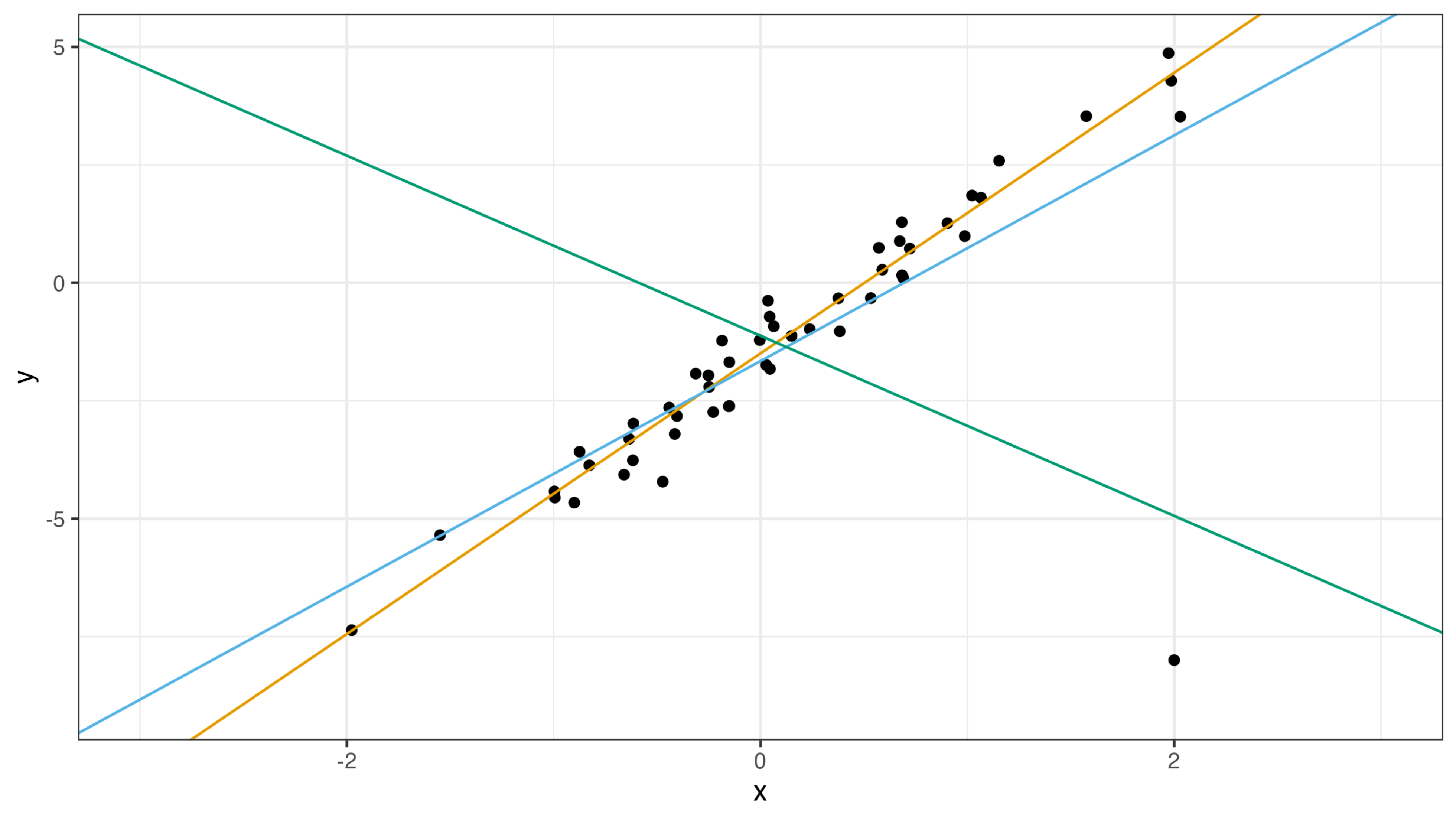
Overfitting

```r
d2 <- rbind(d, data.frame(x=2, y=-8))

m2 <- lm(y~x, d2)
coef(m2)

# (Intercept)           x
#   -1.659093    2.391916
```


5
0
-5
-2
0
2
x
y

Overfitting

```
d3 <- rbind(d2, data.frame(x=50, y=-100))
m3 <- lm(y~x, d3)
coef(m3)

# (Intercept)           x
#   -1.124650   -1.908941
```



5
0
-5
y
-2
0
2
x

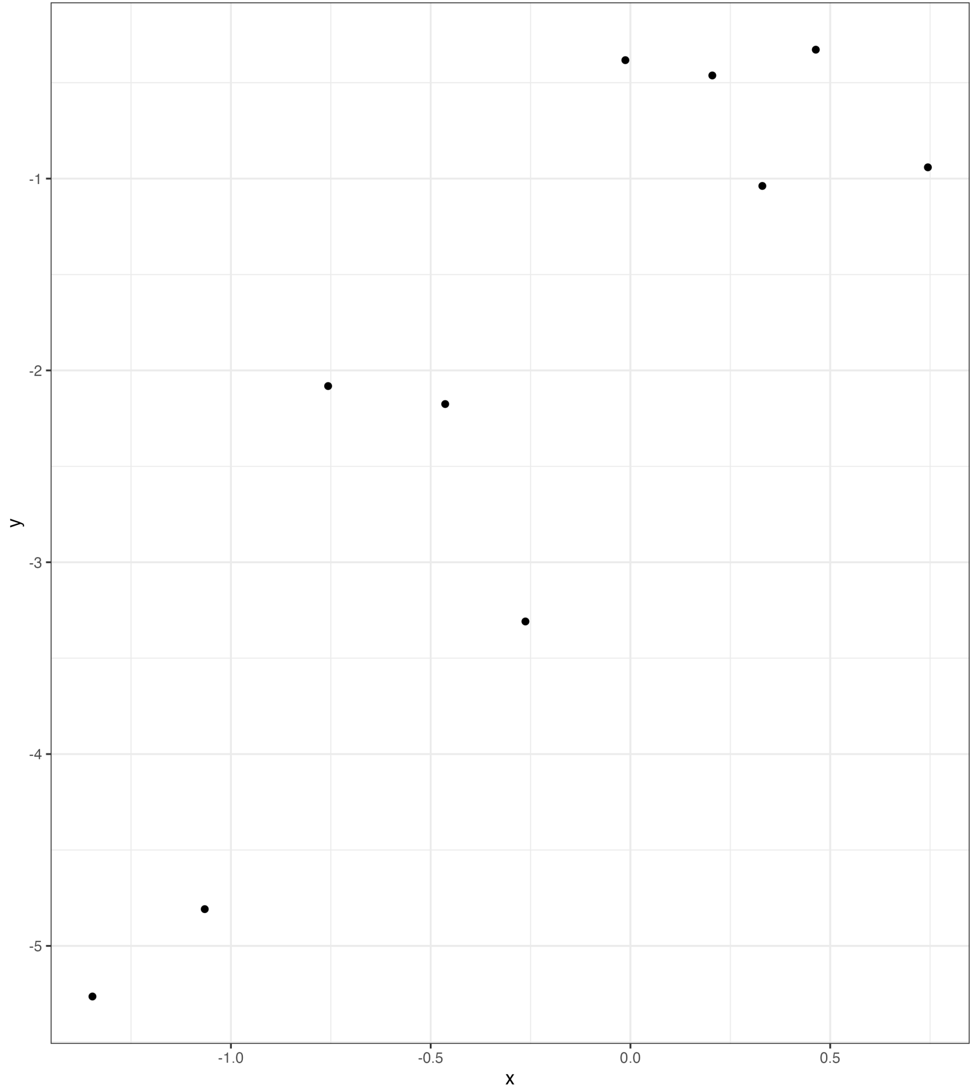
Overfitting

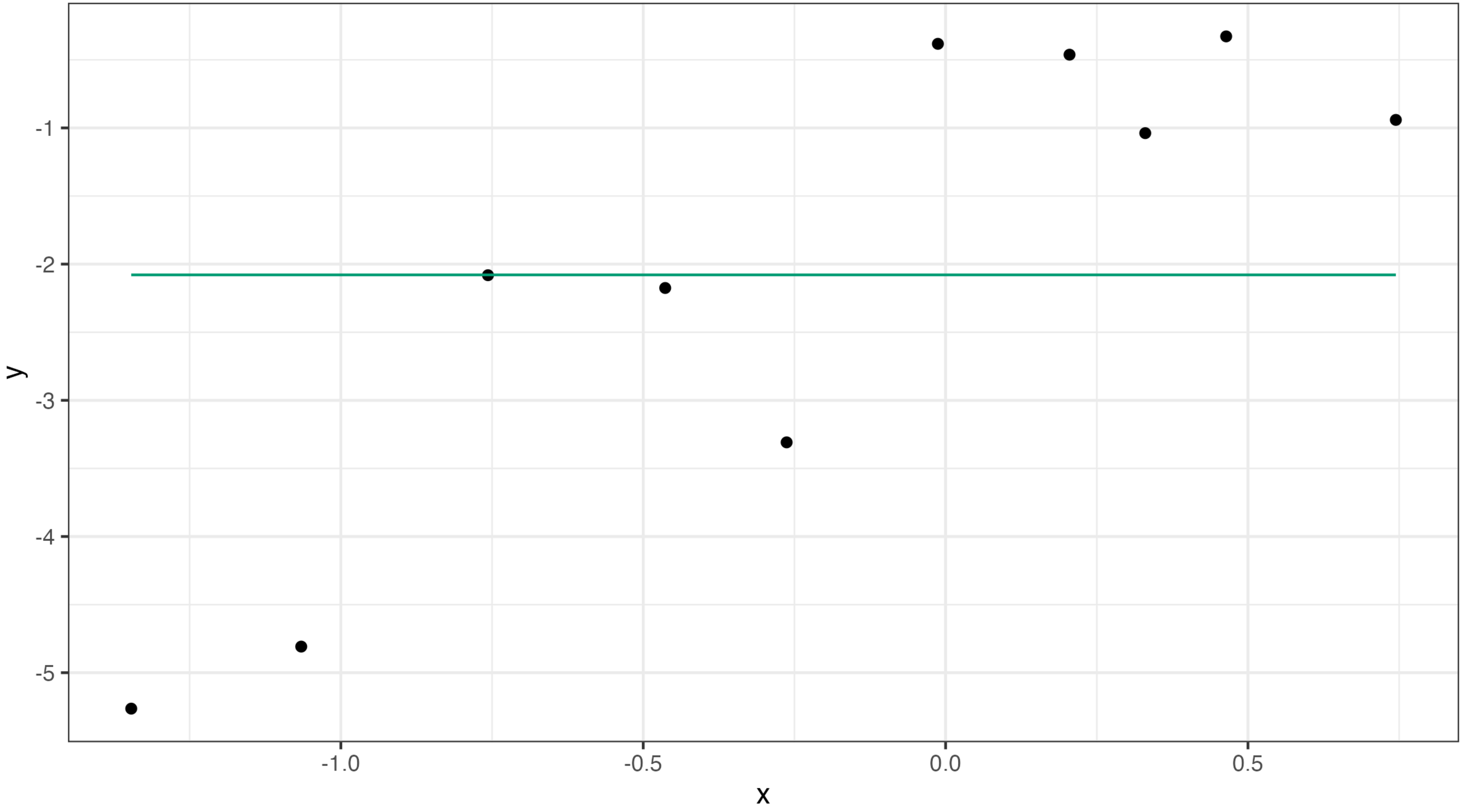
- *Overfitting* occurs when the model you are estimating has too much flexibility and can make large adjustments for small variations in the data it is fit to (or *trained on*)
- Overfitting hurts the ability of your model to *generalize*, or make reasonable predictions with data it hasn't seen yet (we call these predictions *out of sample*, or OOS)
- Overfitting often occurs under one of two conditions:
  1. The individual parameter estimates grow too large
  2. Your model has a large number of estimable parameters relative to the amount of data

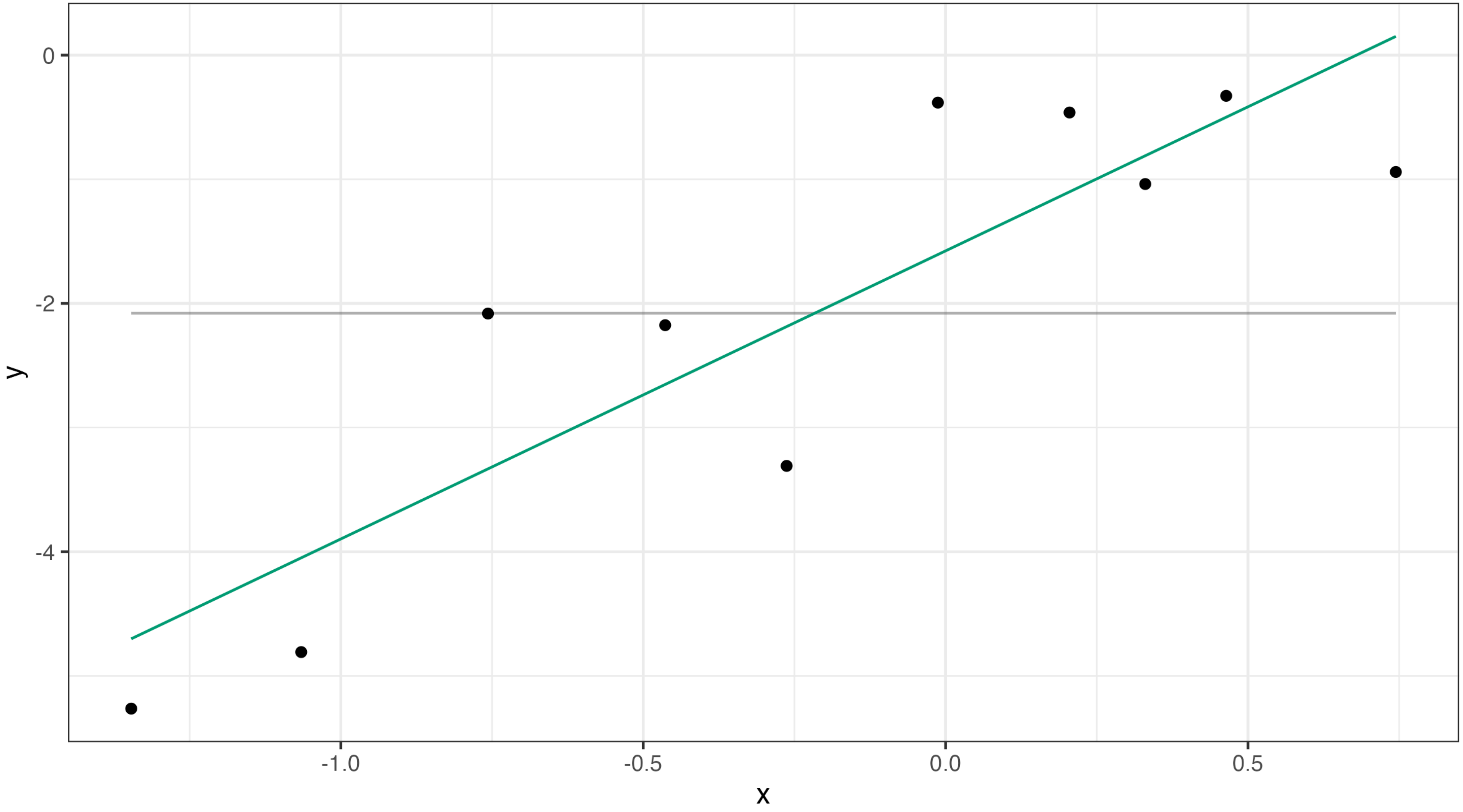
Overfitting

- What happens when functions become too flexible?
- We'll plot ten data points and then a polynomial function of best fit

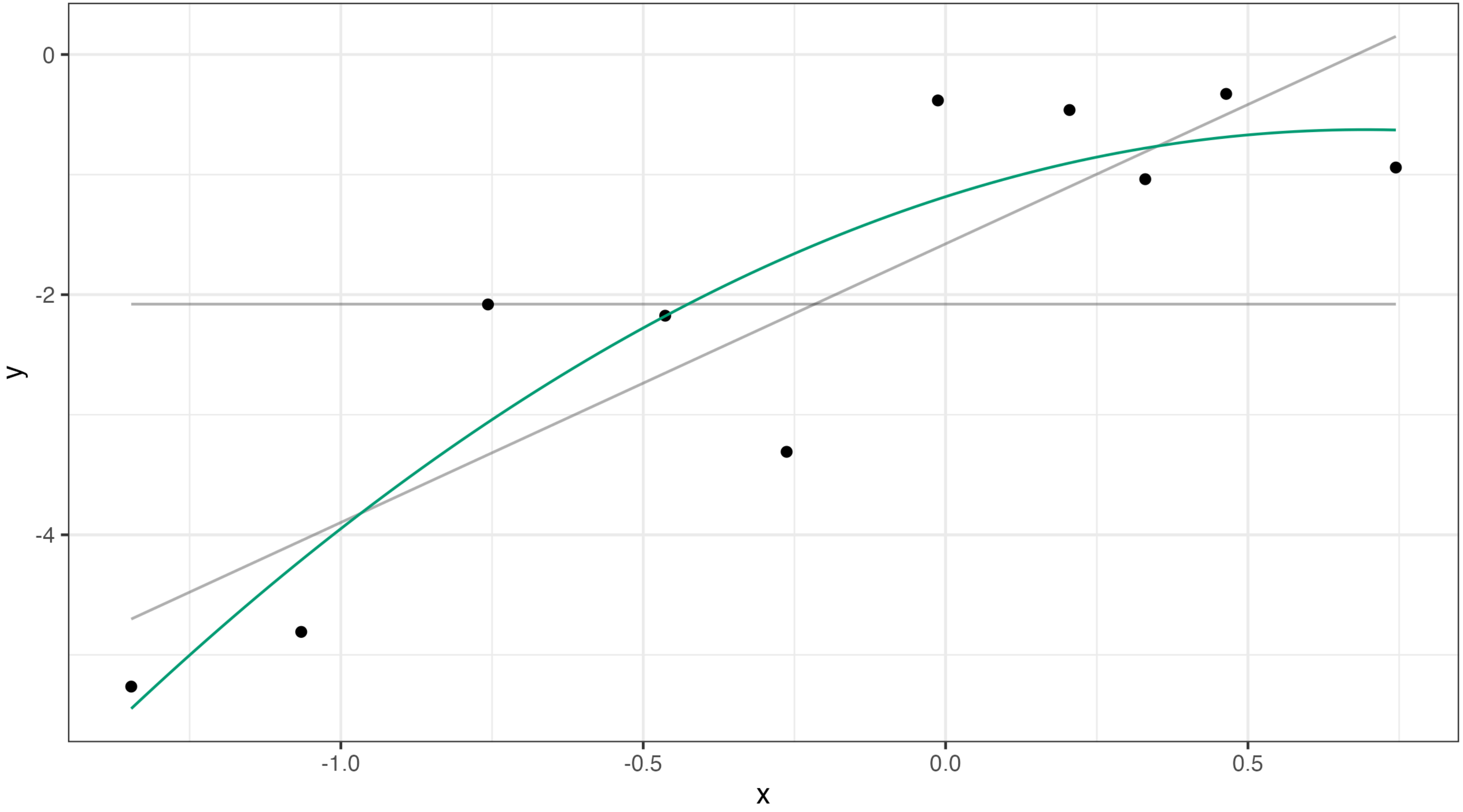
```r
d <- data.frame(x = rnorm(10))
d$y <- -1.5 + 3*d$x + rnorm(10)

ggplot(d, aes(x=x, y=y)) +
  geom_point() +
  theme_minimal()
```



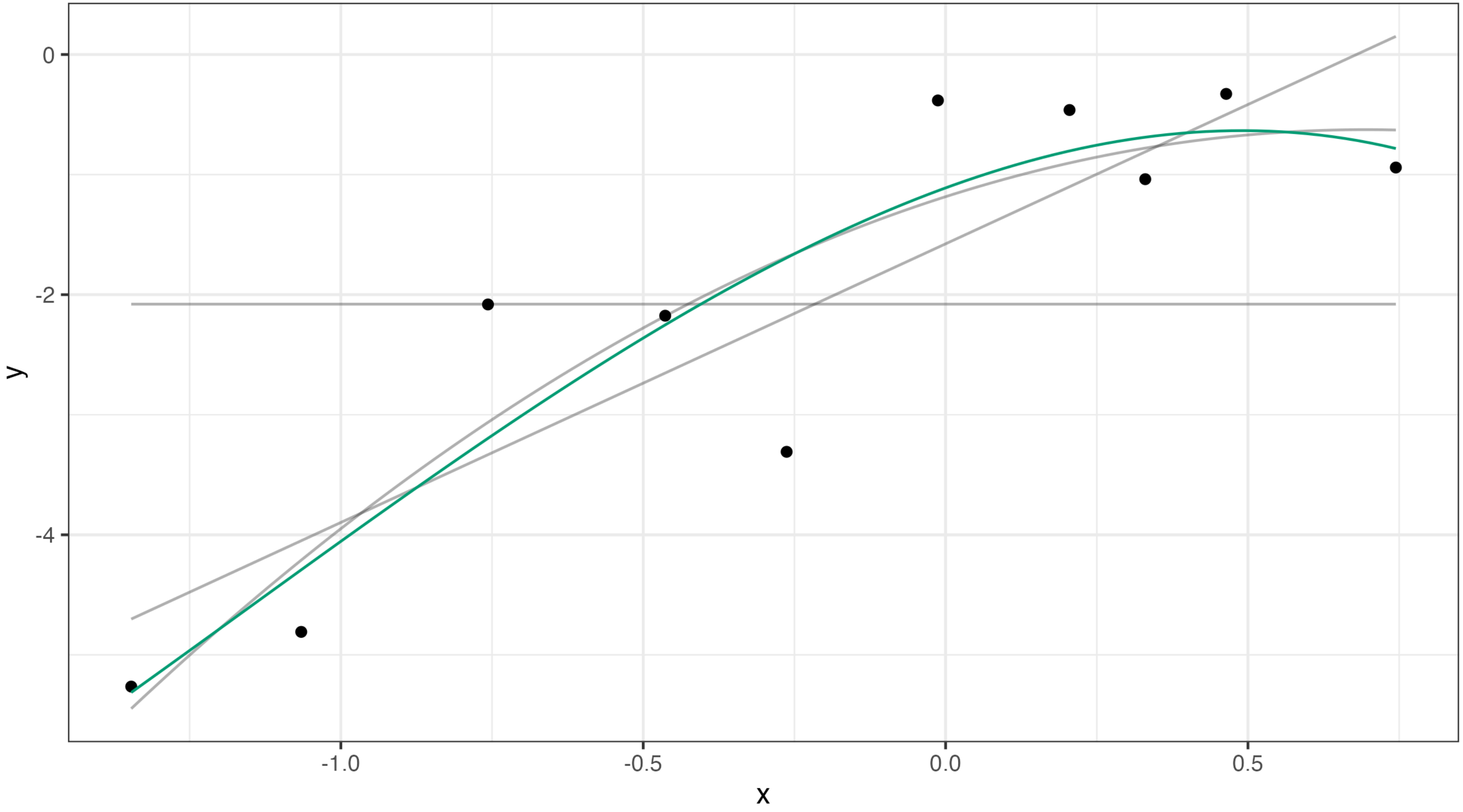

-1
-2
-3
-4
-5
y
-1.0
-0.5
0.0
0.5
x

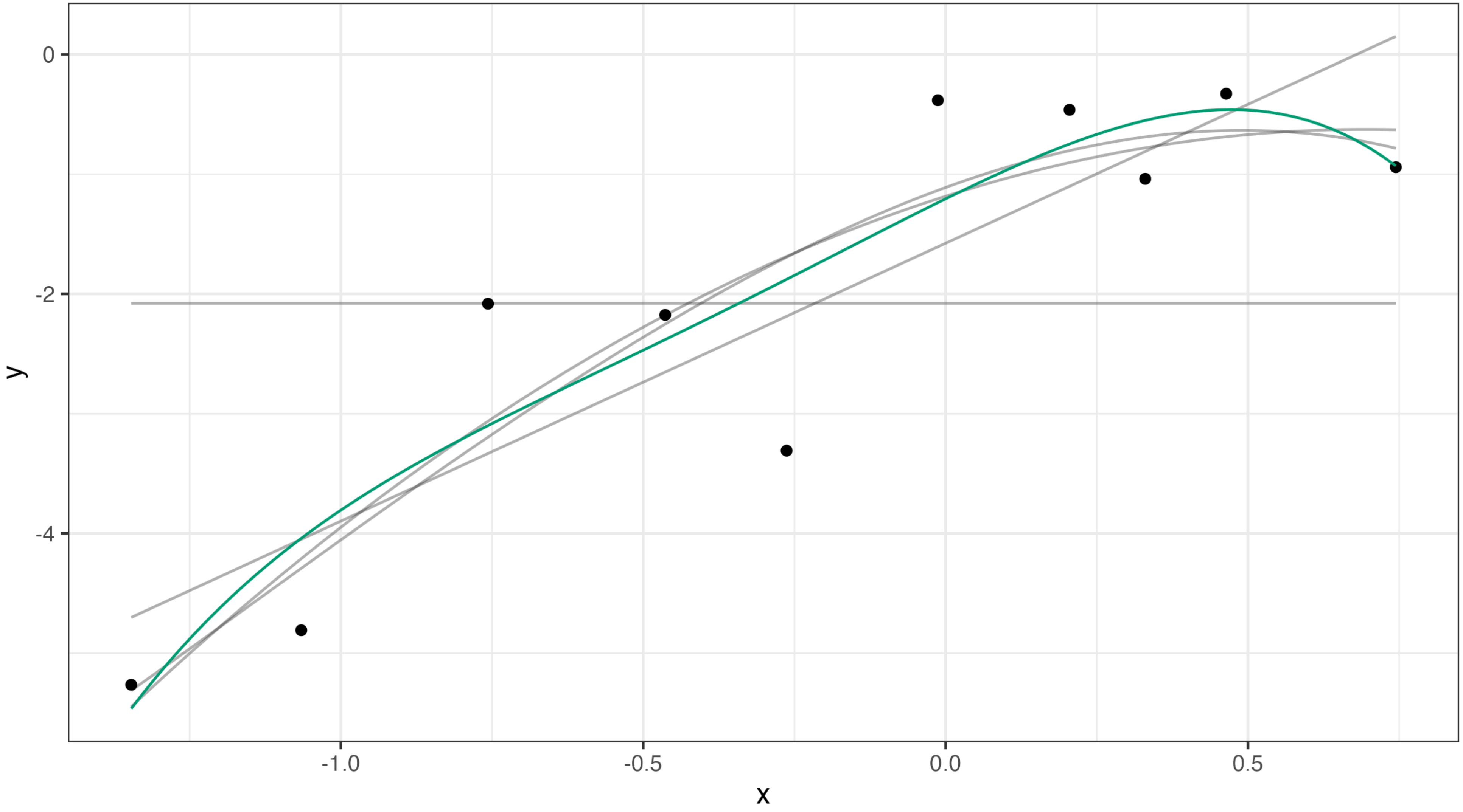


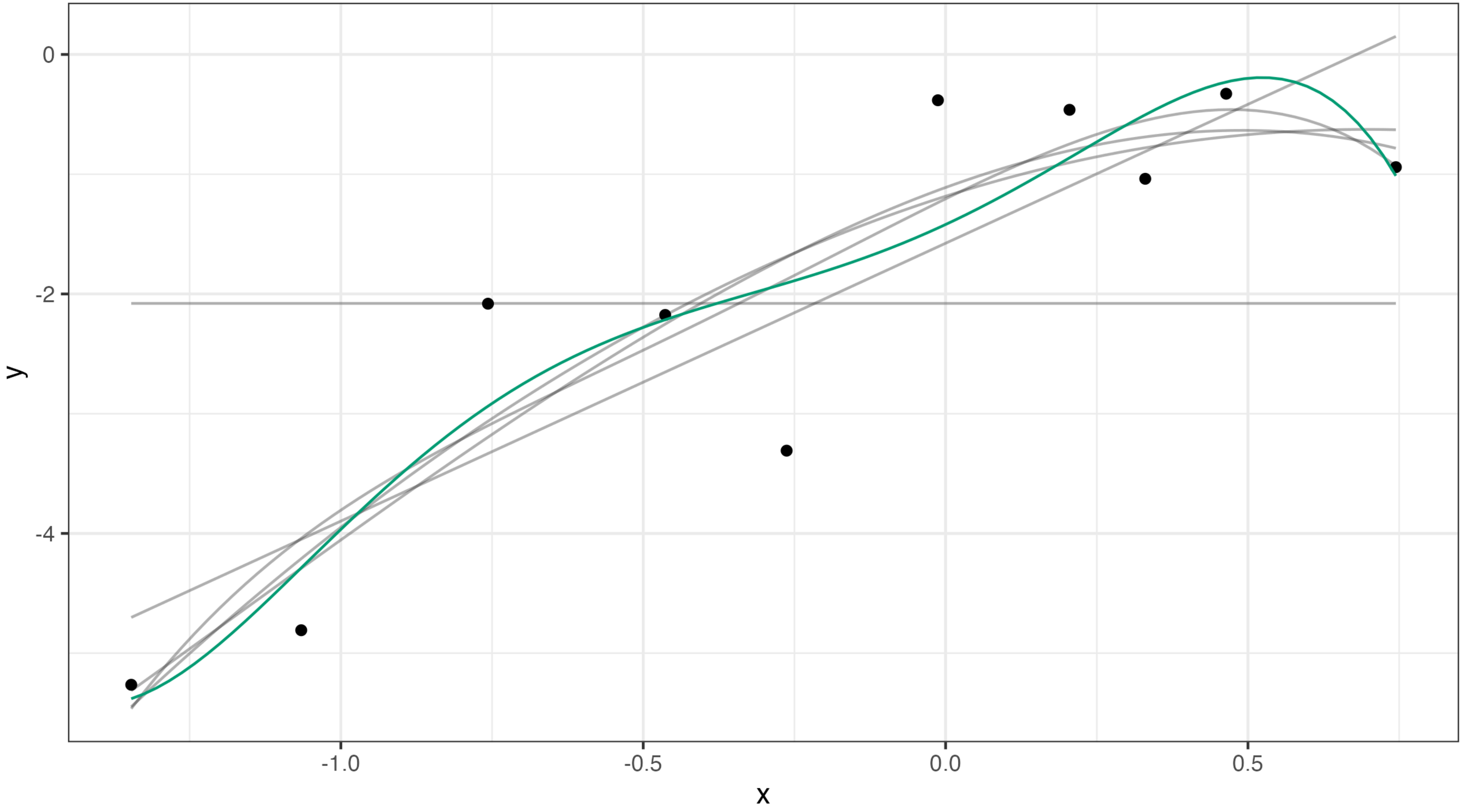
0
-2
-4
y
-1.0
-0.5
0.0
0.5
x



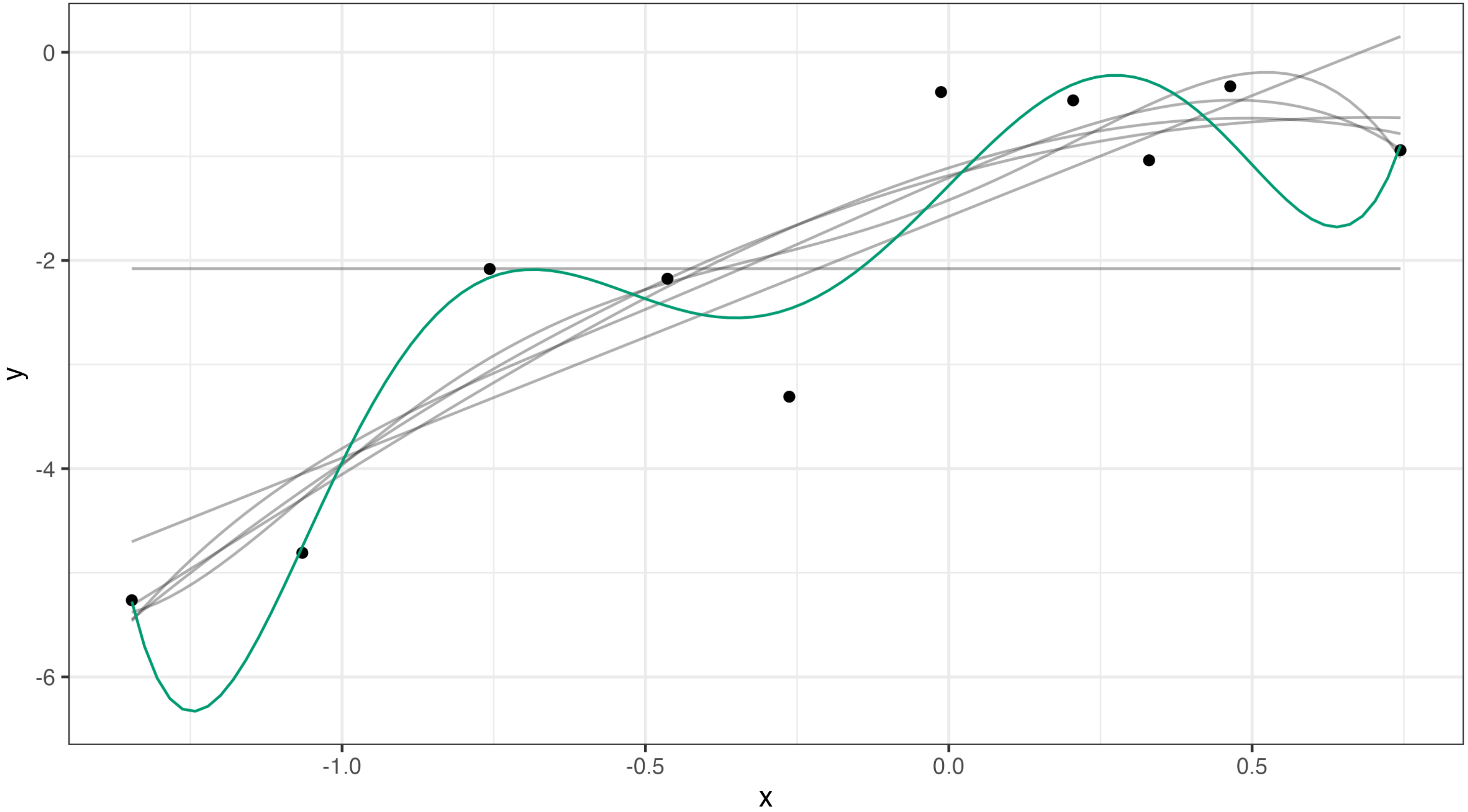
0
-2
-4
y
-1.0
-0.5
0.0
0.5
x


0
-2
-4
y
-1.0
-0.5
0.0
0.5
x

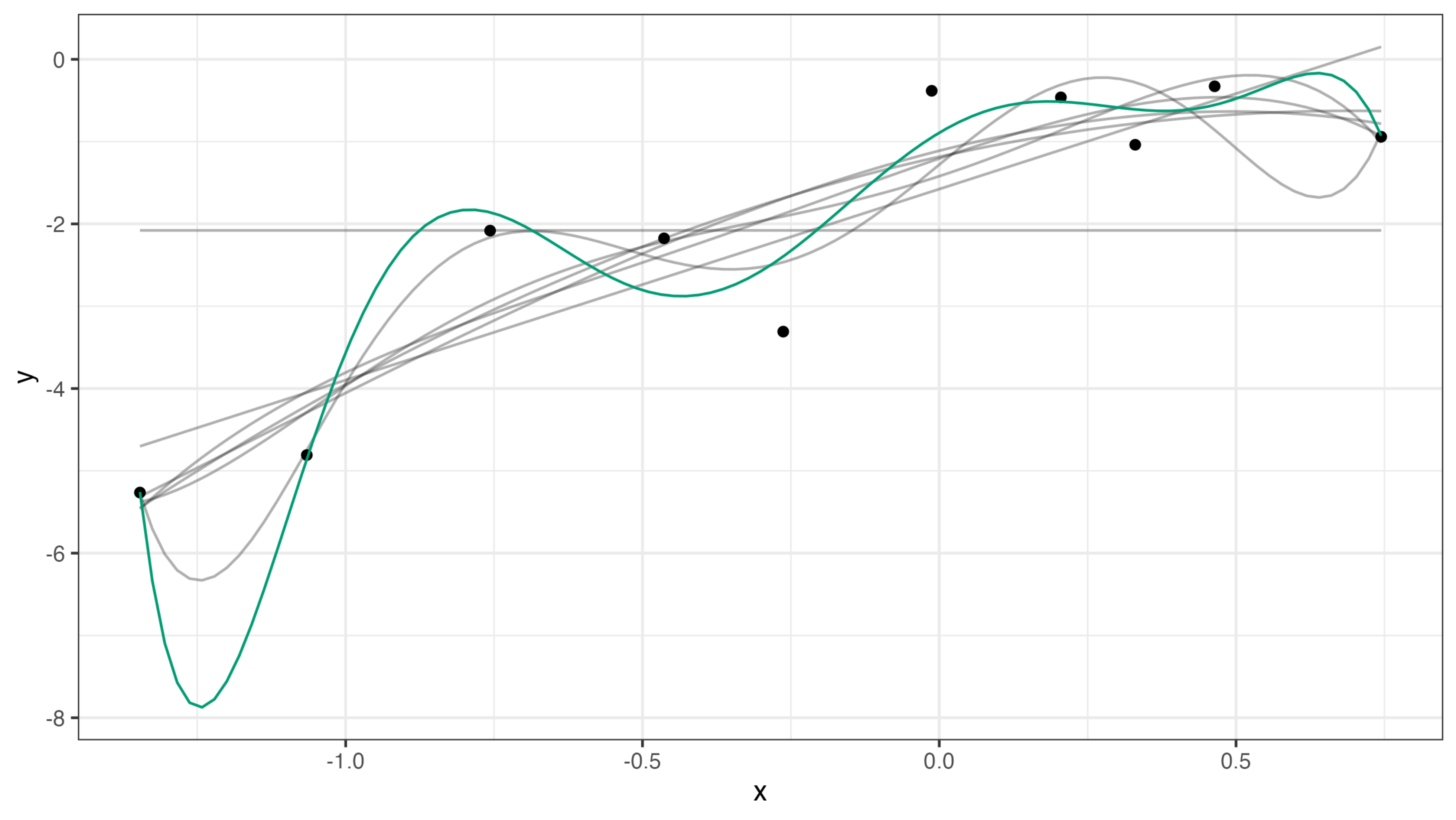

0
-2
-4
y
-1.0
-0.5
0.0
0.5
x



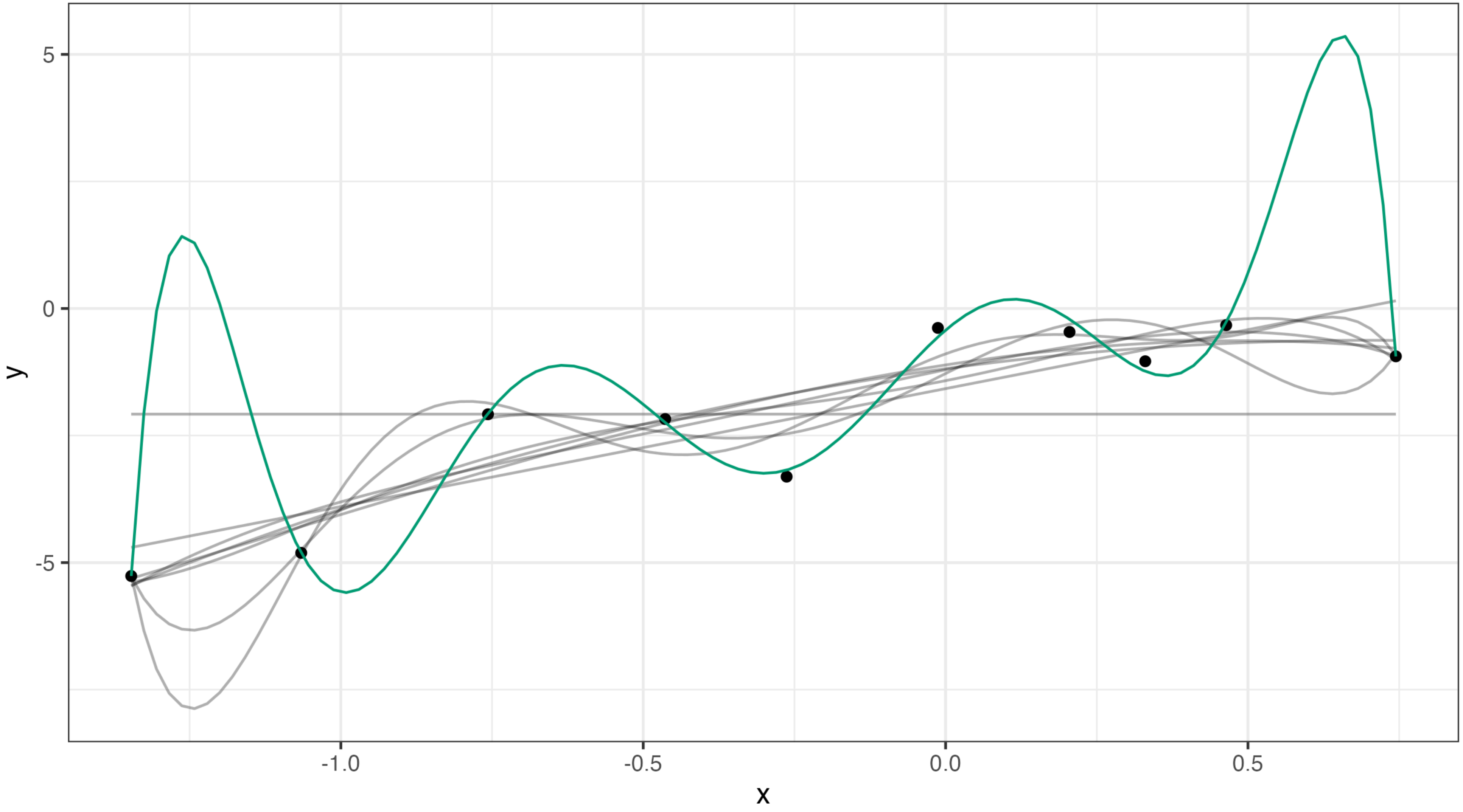
0
-2
-4
y
-1.0
-0.5
0.0
0.5
x



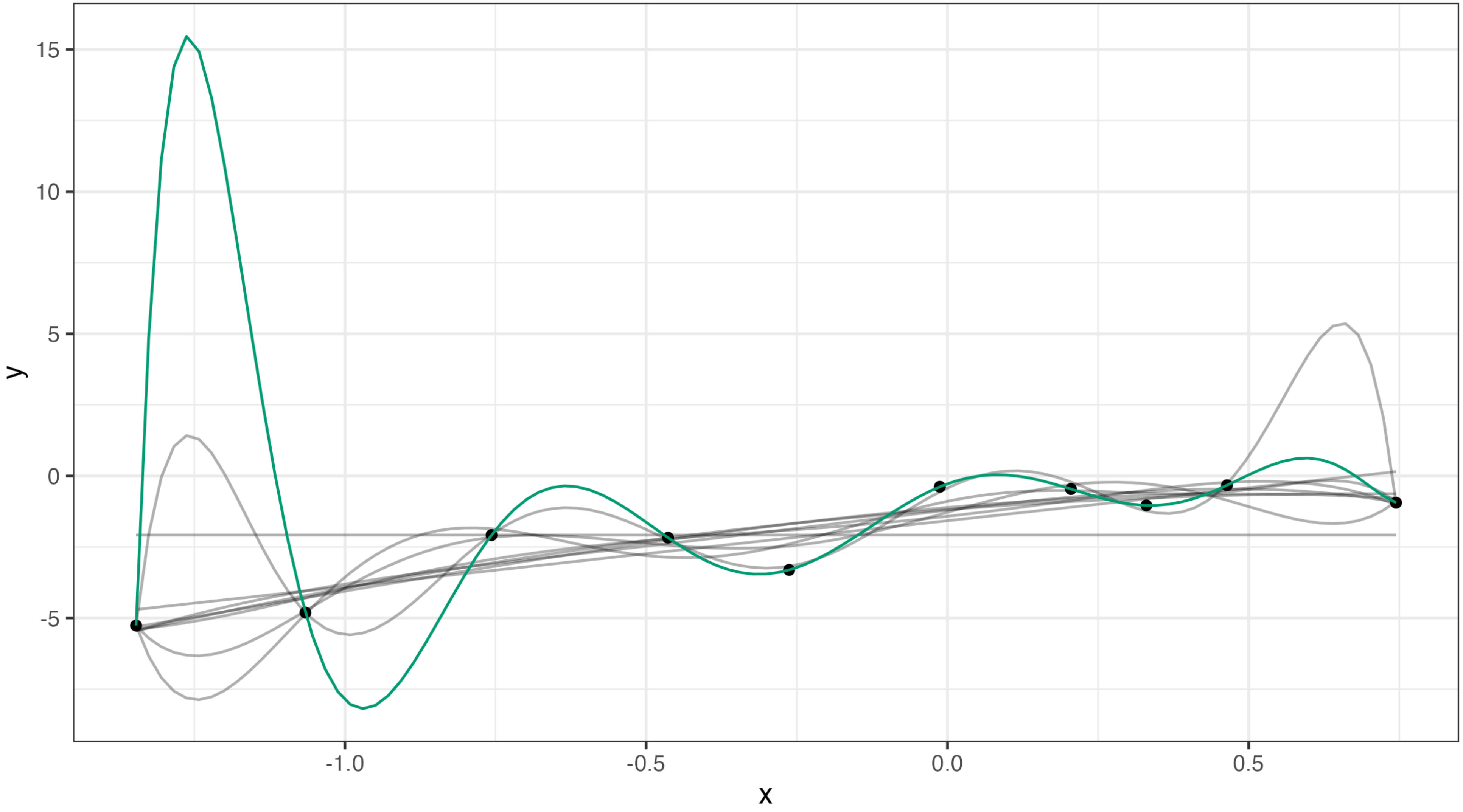
0
-2
-4
-6
y
-1.0
-0.5
0.0
0.5
x



0
-2
-4
-6
-8
y
-1.0
-0.5
0.0
0.5
x



5
0
-5
y
-1.0
-0.5
0.0
0.5
x


15
10
5
0
-5
y
-1.0
-0.5
0.0
0.5
x

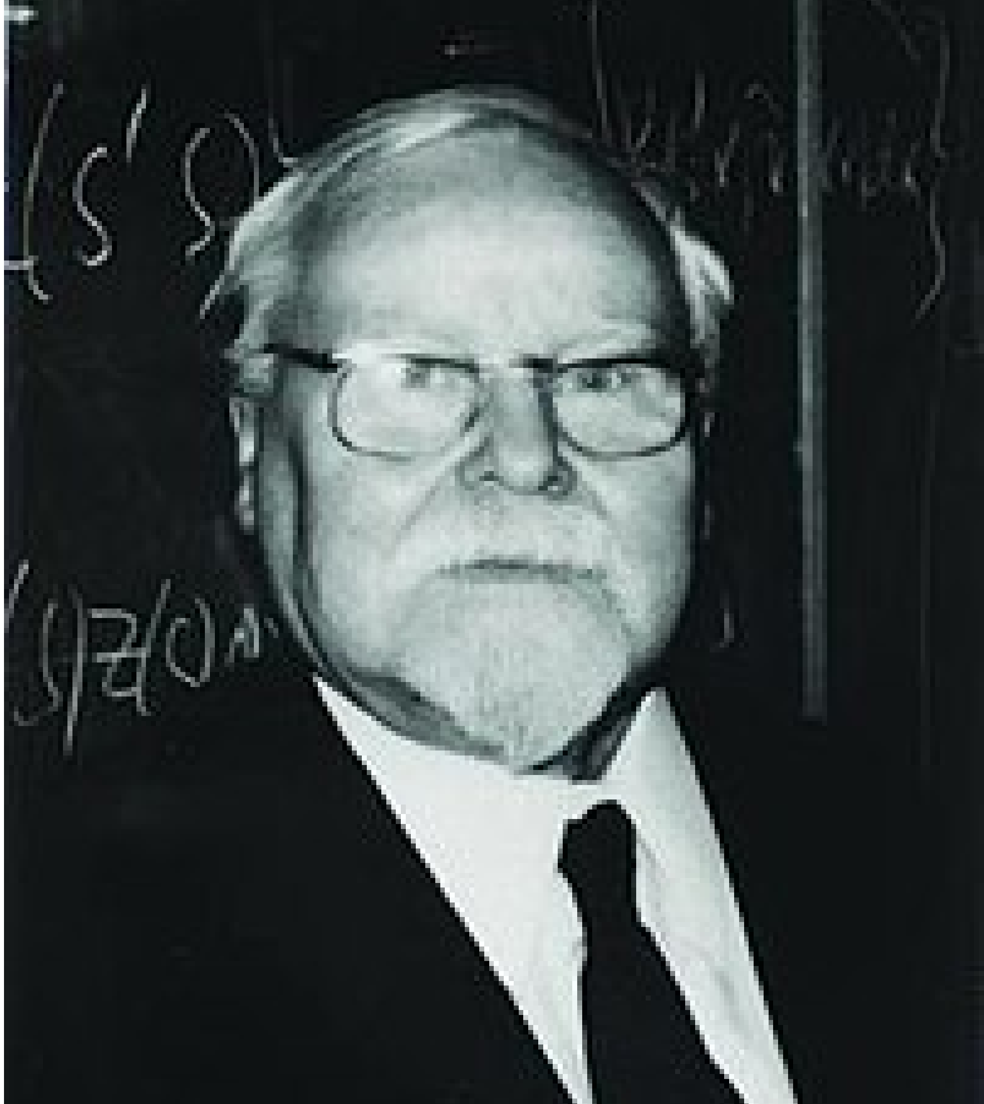
Dealing with Overfitting

Regularization

- **Core Idea:** Regularization adds a penalty to your loss function (or likelihood) when you go to estimate your parameters that helps to fight against overfitting
- Downside: Induces some bias in your parameter estimates
  - This means you technically get "the wrong answer" when you go to interpret coefficients
  - You can control how much bias you introduce, however!
- Upsides: Lots!
  - Helps models generalize to OOS data
  - Useful when you have multicolinearity
  - Can allow you to estimate models that would be otherwise unidentified
  - Easy to implement
  - Works with (almost) any type of model

Andrey Tikhonov (1906-1993)

- Soviet mathematician
- Active in topology and mathematical physics (among other things)
- Recipient of the Lenin Prize (1966)
- Twice named Hero of Socialist Labor (1954, 1986)



Ridge Regression

- **Core Idea:** We will conduct a linear regression, but add a penalty proportional to the L^2 norm of the vector of coefficients, β
- This type of penalty is called the Ridge Penalty, Ridge Regularization, or L^2 Regularization
- Not just useful for linear regressions!
- Recall the OLS loss function that minimizes the squared error:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{\mathbf{x}_i, y_i \in \mathbf{X}} \left(y_i - \mathbf{x}_i \beta\right)^2$$

- Now we add the Ridge penalty:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{\mathbf{x}_i, y_i \in \mathbf{X}} \left(y_i - \mathbf{x}_i \beta\right)^2 \underbrace{+\lambda ||\beta||_2^2}_{\text{Ridge Penalty}}$$

- Note that minimizing the SSE is the same as minimizing the MSE!

Ridge Regression

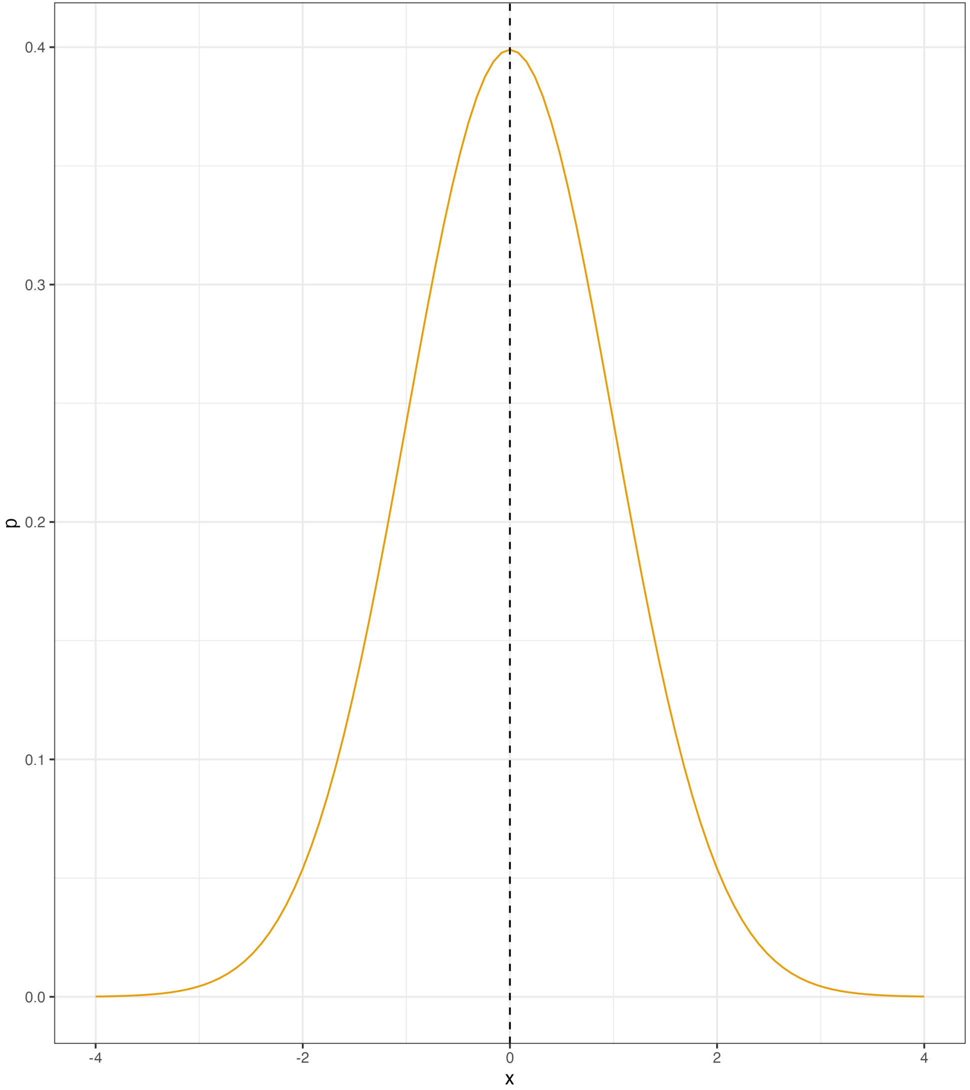
- Recall the loss function:

$$\hat{\beta} = \underset{\beta}{\mathbf{argmin}} \sum_{\mathbf{x}_i, y_i \in \mathbf{X}} \left(y_i - \mathbf{x}_i \beta\right)^2 \underbrace{+\lambda ||\beta||_2^2}_{\text{Ridge Penalty}}$$

- So what happens here?
  - The left part of the argmin tries to minimize the MSE
  - The right part tries to minimize the length of the vector β
- So who wins?
  - It depends on λ, often called the *regularization parameter*
  - If $\lambda = 0$, there is no regularization, and we recover OLS exactly
  - For $\lambda > 0$, the optimization problem has to consider the size of β
  - As $\lambda \to \infty$, the only thing that matters is reducing the size of β until $||\beta||_2 = 0$
  - The larger λ is, the more β is shrunk toward zero
  - This is how you control how much regularization (and how much bias) you induce

The Ridge Penalty Puts a Normal Prior on β

- We think that the elements of β should be normally distributed
- They are *most likely* to be *nearly* zero
- The value of λ controls how much shrinkage toward zero is experienced
- Another way to think about λ is that it controls the *variance* of the normal prior
- When you get to multilevel models, think of random effects as similar to regularization!
- Typically, we pick λ to maximize OOS predictive performance, either by holding out a *test set* or through cross-validation



One Quick Note on Maximizing a Penalized Likelihood

- This is how we minimize squared error:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_i \left(y_i - \mathbf{x}_i\beta\right)^2 + \lambda||\beta||_2^2$$

- This is how we maximize the log likelihood:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} \ -\sum_i \left(y_i - \mathbf{x}_i\beta\right)^2 - \lambda||\beta||_2^2$$

What about other types of regularization?

- Remember that we said overfitting occurs if your parameters are too big or if you have too many
- Ridge only solves the first problem. What about the second?
- Turns out, we can do this using Least Absolute Shrinkage and Selection Operator (LASSO)
- This is implemented through another penalized loss function

Robert Tibshirani (born 1956)

- Professor of Statistics at Stanford University
- Student of Bradley Efron
- Invented Generalized Additive Models (GAMs)
- Wrote *The Elements of Statistical Learning*
- Wrote *An Introduction to Statistical Learning*

LASSO Regression

- **Core Idea:** We will conduct a linear regression, but add a penalty proportional to the L^1 norm of the vector of coefficients, β
- This type of penalty is called the LASSO Penalty, LASSO Regularization, or L^1 Regularization
- Still not just useful for linear regressions!
- Recall the OLS loss function that minimizes the squared error:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{\mathbf{x}_i, y_i \in \mathbf{X}} \left(y_i - \mathbf{x}_i \beta\right)^2$$

- Now we add the LASSO penalty:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{\mathbf{x}_i, y_i \in \mathbf{X}} \left(y_i - \mathbf{x}_i \beta\right)^2 \underbrace{+\lambda ||\beta||_1}_{\text{LASSO Penalty}}$$

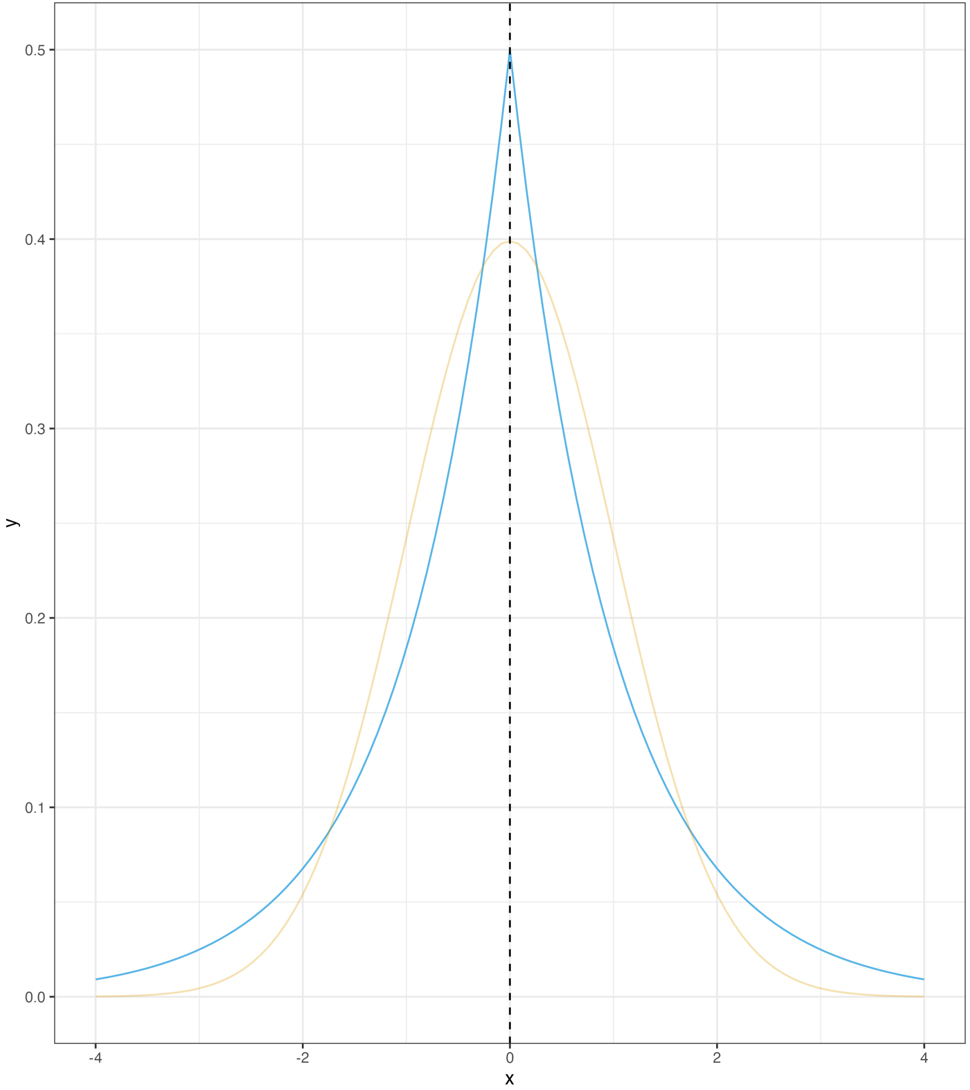
LASSO Regression

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{\mathbf{x}_i, y_i \in \mathbf{X}} \left(y_i - \mathbf{x}_i\beta\right)^2 \underbrace{+\lambda||\beta||_1}_{\text{LASSO Penalty}}$$

- So what happens here?
  - The left part of the argmin tries to minimize the MSE
  - The right part tries to minimize the length of the vector β, but in kind of a different way from the Ridge penalty!
- So what changes?
  - Note that as $\beta, \epsilon \to 0$, $|\beta + \epsilon| - |\beta| > (\beta + \epsilon)^2 - \beta^2$
  - This gives a loss function with a LASSO penalty extra incentive to force some coefficients to be zero!
  - The Ridge penalty will uniformly shrink coefficients toward zero, while the LASSO penalty will force some number of coefficients to be *precisely* zero
  - This effectively drops variables with less predictive value from your model (which is the "selection" part of LASSO)
  - Higher values of λ will drop more variables

LASSO Penalty Puts a Laplace Prior on β

- This means we think that the elements of β should be Laplace distributed
- This means they are *most likely* to be *exactly* zero, but with enough evidence they can take on any value
- The value of λ controls how aggressively coefficients are forced to be zero
- Again, we pick the λ value to maximize OOS predictive performance, either by holding out a *test set* of some data or through cross-validation



More Resampling Methods

The Central Limit Theorem

- Core to statistics, because it says "normal distributions are worth understanding" even though most data is not actually normally distributed!
- $\{X_1, X_2, \ldots, X_n\}$ are i.i.d. random variables with:
  - $\mathrm{E}[X_i] = \mu$
  - $\mathrm{Var}[X_i] = \sigma^2$
- We define the *sample mean*: $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$

The Central Limit Theorem

- As n grows large, the *distribution of sample means* converges to:

$$\bar{X}_i \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$$

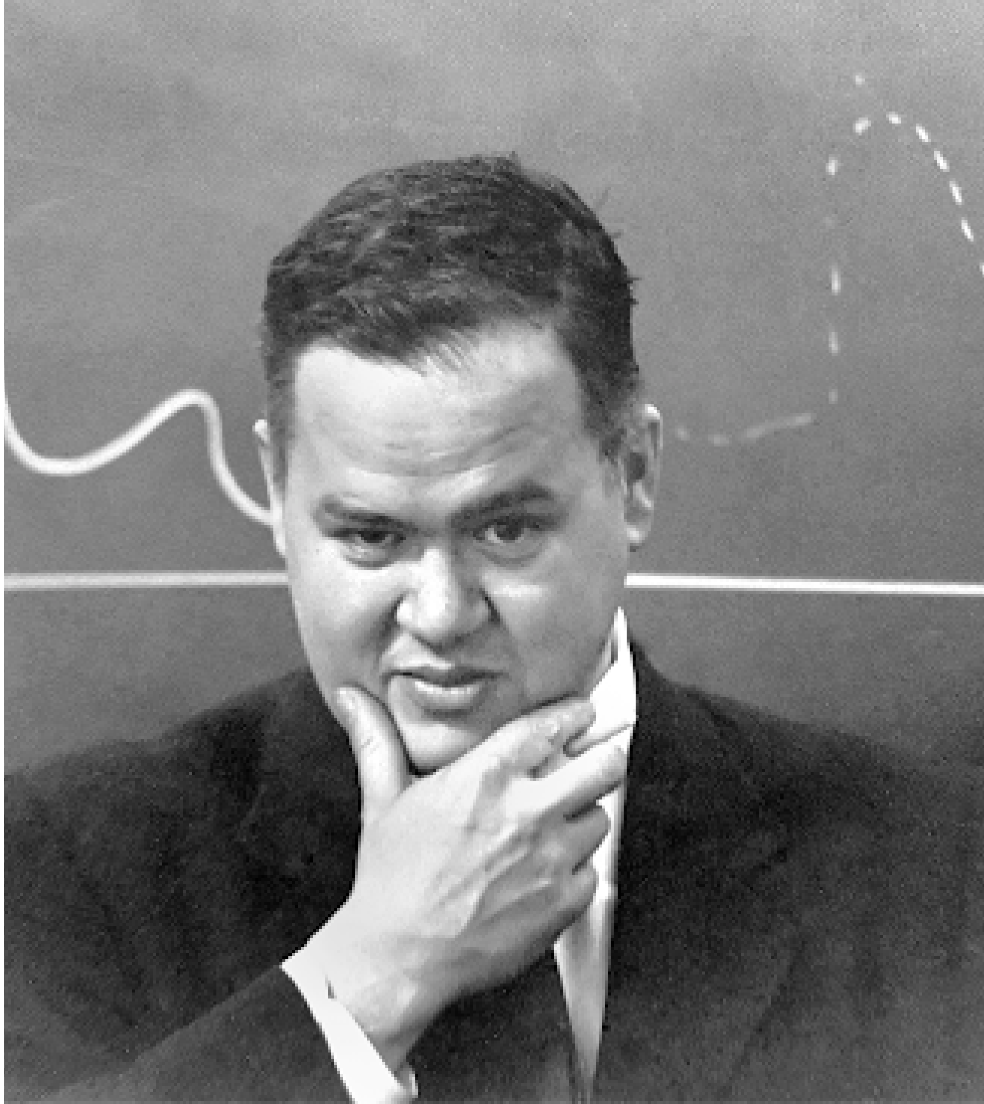
- This holds even if X_i is not normally distributed!
- This only applies to sample means!

Resampling Methods

- We want to reason about populations from samples
- The CLT helps a lot, but it doesn't do everything!
- **Core idea:** The distribution of the data you have is your best guess at the distribution of the population
- But what do we do with this?

John Tukey (1915-2000)

- His ideas about “exploratory data analysis” are the foundations of modern data science
- His book “Exploratory Data Analysis” is still one of the best sources for data visualization and exploratory analysis
- Invented the term “bit” at Bell Labs
- Invented the box plot
- Invented the Fast Fourier Transform
- Certified badass



What did Tukey give us today?

The Jackknife

- Thanks Tukey!
- "Rough and ready" tool to solve statistical problems when the exact solution might be hard or unknown
- **Core idea:** We take our data, make new datasets by dropping single observations, compute our statistic of interest, and then look at the distribution

The Jackknife

- We have $\{X_1, X_2, \ldots, X_n\}$, which are i.i.d. samples from some population
- We care about some estimator or statistic in the population, $f(X_i)$ We often think that: $\mathrm{E}[f(X_i)] = f\left(\{X_1, X_2, \ldots, X_n\}\right)$
- But this tells us nothing about the bias or variance of that estimator!

The Jackknife

- From your dataset of length n, create n new resampled datasets of length $n - 1$, each missing one observation
- Compute your estimator or statistic on each resampled dataset
- Aggregate these statistics to get your jackknife estimate
  - The mean of the estimator in jackknife samples can be used to understand bias -The variance of the estimator in jackknife samples can be used to quantify uncertainty
  - Does not require the CLT to apply to the estimator!

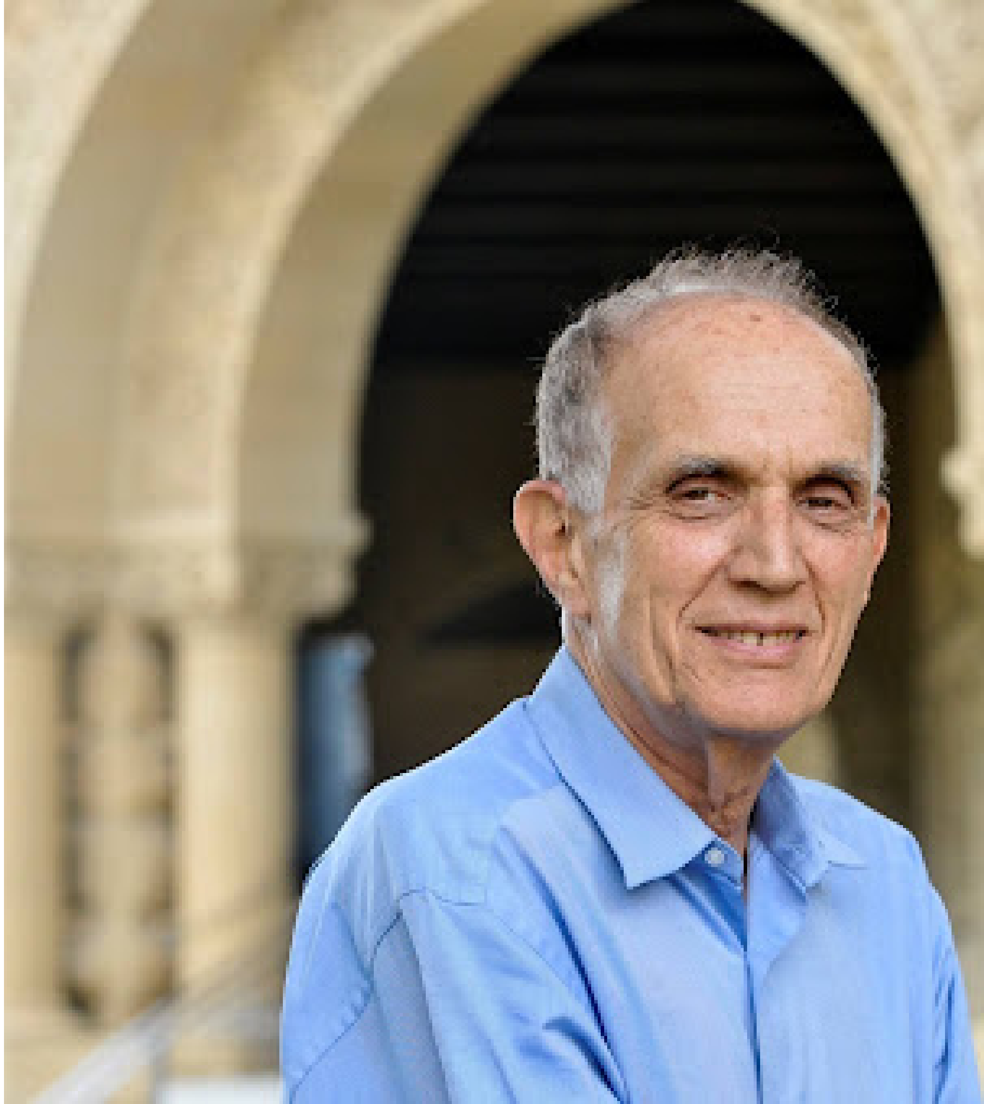
The Bootstrap

The Bootstrap

- **Resampling's Core Idea:** The distribution of the data you have is your best guess at the distribution of the population
- **Jackknife's Core Idea:** We take our data, make new datasets by dropping single observations, compute our statistic of interest, and then look at the distribution
- **Bootstrap's Core Idea:** We take our data and make new datasets by resampling *with replacement* from our original dataset to approximate the population. Then we compute our statistic of interest on each resampled dataset and look at the distribution

Bradley Efron (born 1938)

- Invented the bootstrap in 1982
- One of the first major computationally intensive statistical techniques
- His list of awards is insane
  - MacArthur Prize Fellowship
  - Membership in the National Academy of Sciences and the American Academy of Arts and Sciences
  - Fellowship in the Institute of Mathematical Statistics (IMS) and the American Statistical Association (ASA)
  - The Lester R. Ford Award, the Wilks Medal, the Parzen Prize, and the Rao Prize



The Bootstrap

- It works for basically anything!
- Your data describes an *empirical distribution function* - this is an approximation of the cumulative distribution function (CDF)
- The empirical distribution function (eCDF) converges to the CDF
- The bootstrap uses resampling with replacement to build new eCDFs
- Because we resample with replacement, we can have many more eCDFs *with the same sample size as the original data* than the jackknife
- Data are guaranteed to be drawn from the population of interest!
- Often used for bias, variance, and constructing confidence intervals
- Does not require the CLT to apply to your estimator!

How to Bootstrap

1. Compute your estimate in the full sample with N observations
2. Draw M bootstrap samples, each of length N, from the original dataset *with replacement*
3. Compute your estimate on each bootstrap sample
4. Use the distribution of bootstrap estimates to:

- Check the original estimate for bias (using the mean value)
- Find the variance of a standard error of the estimate (using the variance or sd)
- Construct a confidence interval (by looking at the end points of the middle 95% of the distribution of bootstrap estimates)

Applied Bootstrapping

```
d <- data.frame(x = rnorm(1e2))
d$y <- -1.5 + 3*d$x + rnorm(1e2)

m <- lm(y~x, d)


summary(m)$coefficients

#              Estimate Std. Error   t value     Pr(>|t|)
# (Intercept) -1.486122 0.09547956 -15.56482 3.078629e-28
# x            3.101451 0.09144205  33.91712 5.913505e-56

coef(m)[2]

#        x
# 3.101451
```

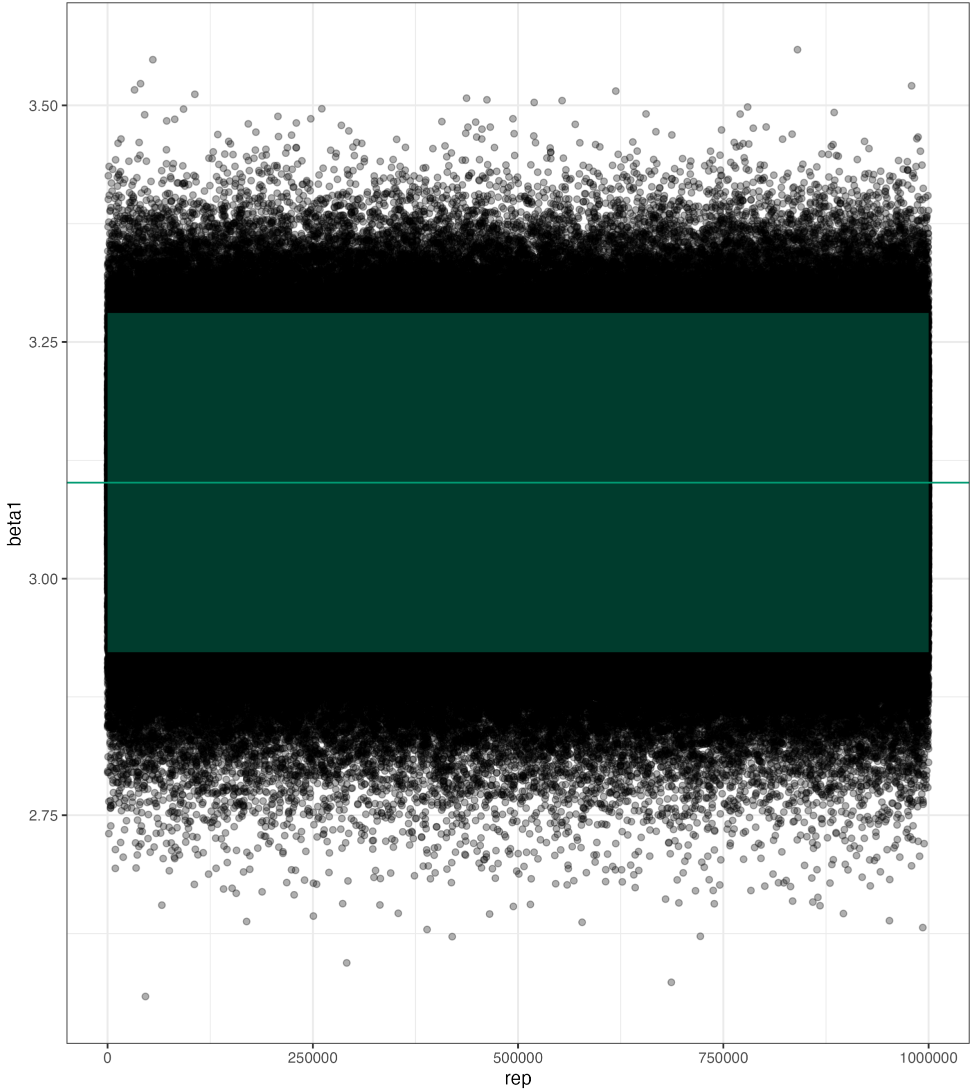
Applied Bootstrapping

1. Write a function that extracts the coefficient we care about
2. Then we add resampling
3. Run it a bunch of times

```r
Beta1Boot <- function(d){
  idx <- sample(nrow(d), replace=TRUE)
  m <- lm(y~x, d[idx,])
  beta1 <- unname(coef(m)[2])
  return(beta1)
}

replicate(5, Beta1Boot(d))

# [1] 2.930935 3.264935 3.177743 3.027659 2.913476

N_boot <- 1e6
boot <- data.frame(rep = 1:N_boot,
                   beta1 = replicate(N_boot, Beta1Boot(d)))
```

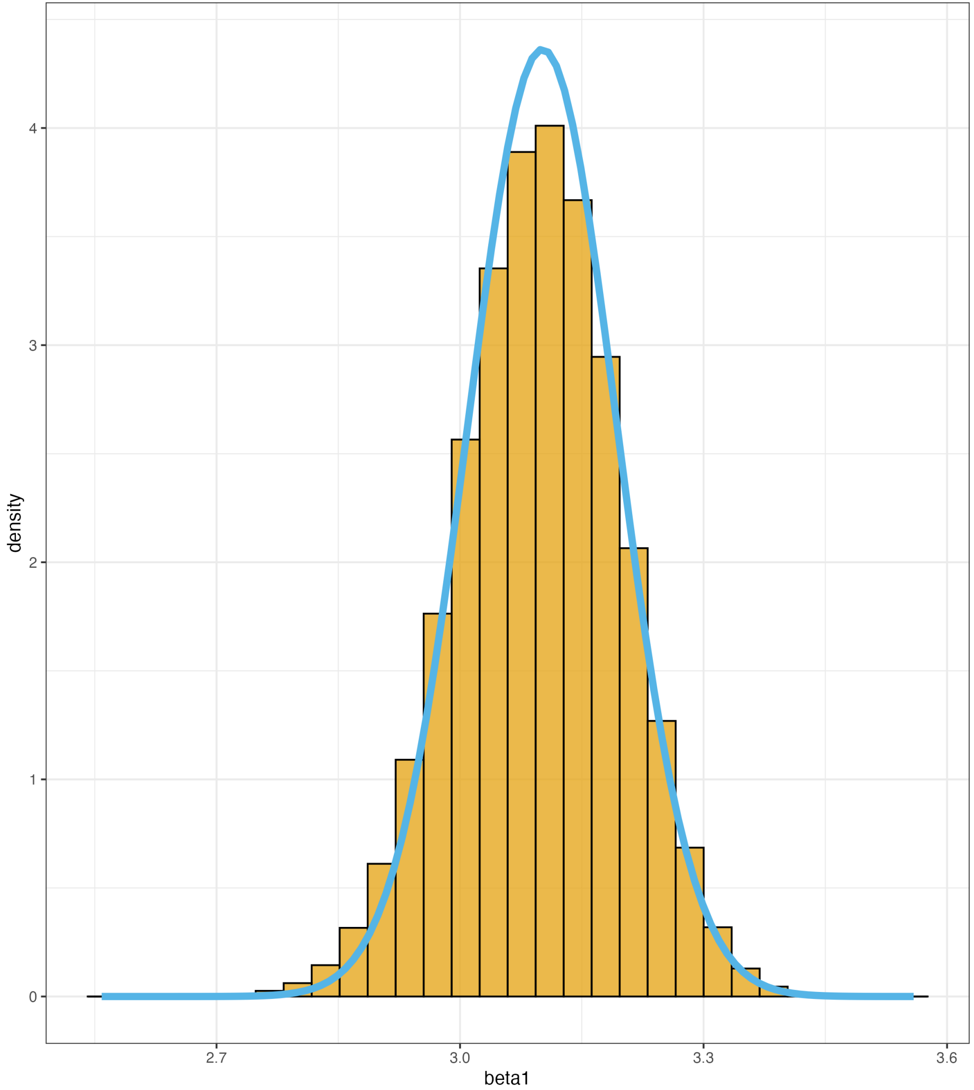
Applied Bootstrapping

```r
ggplot(boot, aes(x = rep, y = beta1)) +
  geom_point(alpha = 0.3) +
  geom_hline(
    aes(
      yintercept =
        summary(m)$coefficients[2, 1]),
      color = okabeito_colors(3)
  ) +
  geom_ribbon(
    aes(
      ymin =
        summary(m)$coefficients[2, 1] -
        1.96*summary(m)$coefficients[2, 2],
      ymax =
        summary(m)$coefficients[2, 1] +
        1.96*summary(m)$coefficients[2, 2]
    ),
    fill = okabeito_colors(3),
    alpha = 0.4
  ) +
  theme_bw()
```



Applied Bootstrapping

```r
ggplot(boot, aes(x = beta1)) +
  geom_histogram(
    aes(y = after_stat(density)),
    bins = 30,
    color = 'black',
    fill = okabeito_colors(1),
    alpha = 0.7
  ) +
  geom_function(
    fun = dnorm,
    size = 2,
    color = okabeito_colors(2),
    args = list(
      mean = summary(m)$coefficients[2, 1],
      sd = summary(m)$coefficients[2, 2]
    )
  ) +
  theme_bw()
```



Applied Boostrapping

- `lm()` estimate, SE, and 95% CI:

```r
summary(m)$coefficients[2,1:2]

#   Estimate Std. Error
# 3.10145067 0.09144205

c(summary(m)$coefficients[2,1] - 1.96*summary(m)$coefficients[2,2],
  summary(m)$coefficients[2,1] + 1.96*summary(m)$coefficients[2,2])

# [1] 2.922224 3.280677
```

- Bootstrapped estimate, SE, and 95% CI:

```r
c(mean(boot$beta1), sd(boot$beta1))

# [1] 3.09708749 0.09906309

quantile(boot$beta1, c(0.025, 0.975))

#     2.5%    97.5%
# 2.897704 3.286801
```

Wrap Up

Recap

- Regularization puts a penalty on the likelihood proportional to the length of the parameter vector
  - This shrinks parameters toward zero
  - Ridge regression shrinks them slightly
  - LASSO regression tends to force parameters to be zero
- Resampling is a way to estimate things that are difficult to know analytically
  - The jackknife drops single observations to create new datasets of approximately the same size
  - The bootstrap resamples *with replacement* to create new datasets the same size as your old data
- Regularization and the bootstrap are probably the two most generalizable applied techniques you'll learn here!

Final Thoughts

- PollEv.com/klintkanopka