SciencesPo
DEPARTMENT OF ECONOMICS

ScPoEconometrics: Advanced

Intro to Statistical Learning

Bluebery Planterose
SciencesPo Paris
2023-04-11

Intro to Statistical Learning: ISLR

- This set of slides is based on the amazing book *An introduction to statistical learning* by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani.

- I'll freely use some of their plots. They say that is ok if I put:

> Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani

- Thanks so much for putting that resource online **for free**.

- We will try to look at their material with our econometrics background. It's going to be fun!

What is Statistical Learning?

- We want to learn the relationship `Y ~ X`, where `X` has p components.
- We assume a general form like

$$Y = f(X) + \epsilon$$

- f is a fixed function, but we don't know what it looks like.
- We want an **estimate** $\hat{f}$ for it.

What is Statistical Learning?

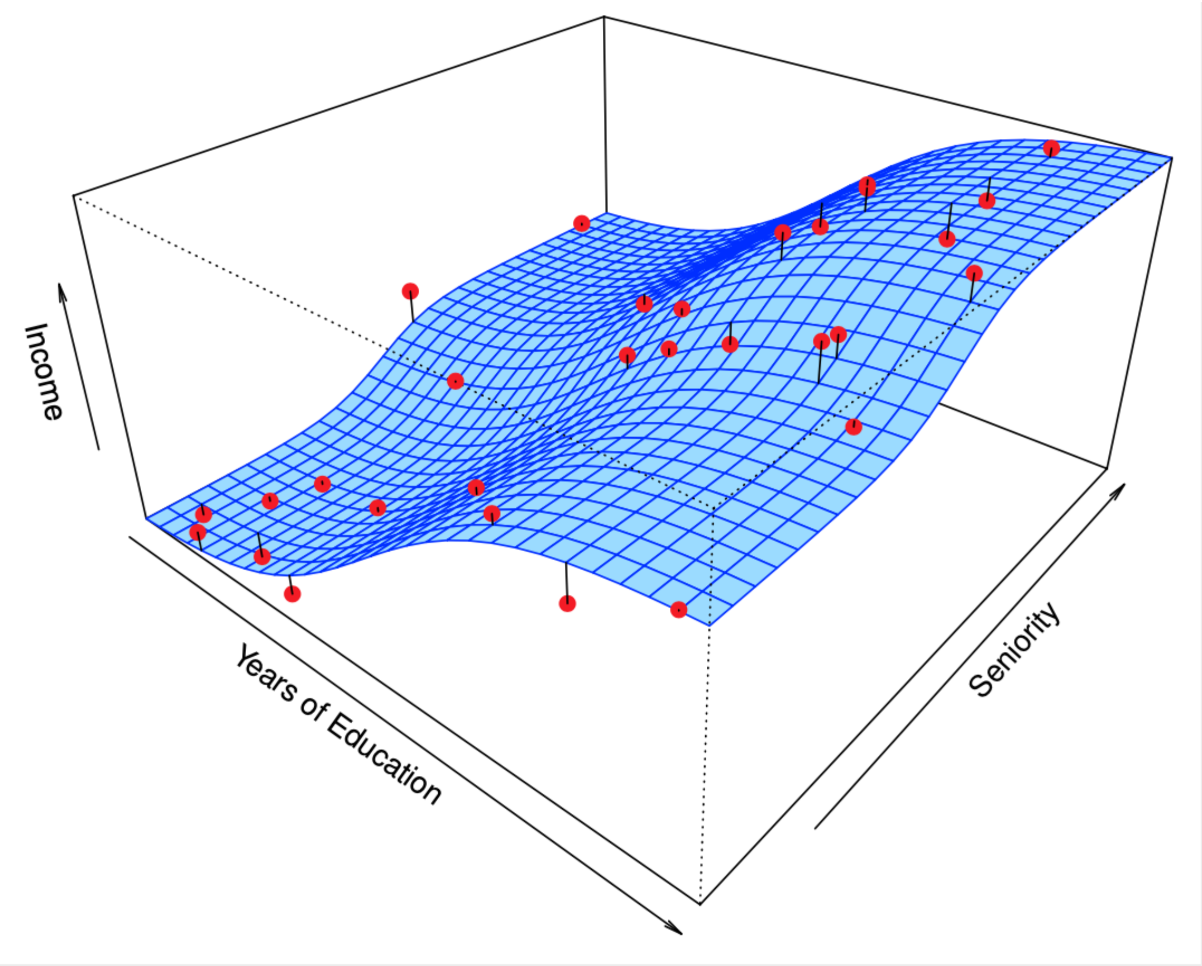
- We want to learn the relationship `Y ~ X`, where `X` has p components.
- We assume a general form like

$$Y = f(X) + \epsilon$$

- f is a fixed function, but we don't know what it looks like.
- We want an **estimate** $\hat{f}$ for it.

- Assume $E[\epsilon|x] = 0$!
- I.e. we assume to have an **identified** model
- We have done this 👈 many times before already.
- But we restricted ourselves to OLS estimation. There are so many ways to estimate f!

An Example of f



- The **blue** shape is true relationship f
- Red dots are observed data: Y
- Red dots are off **blue** shape because of ϵ

What Do You Want To Do with your $\hat{f}$?

Fundamental Difference: (🚨 slight exaggerations ahead!)

Prediction (Machine Learning, AI)

- generate $\hat{Y} = \hat{f}(X)$
- $\hat{f}$ is a **black box**
- We don't know or care *why* it works as long as the prediction is good

What Do You Want To Do with your $\hat{f}$?

Fundamental Difference: (🚨 slight exaggerations ahead!)

Prediction (Machine Learning, AI)

- generate $\hat{Y} = \hat{f}(X)$
- $\hat{f}$ is a **black box**
- We don't know or care *why* it works as long as the prediction is good

Inference (ECON)

- *Why* does Y respond to X? (Causality)
- *How* does Y respond to X_p? Interpret parameter estimates
- $\hat{f}$ is not a **black box**.
- (Out of sample) Prediction often secondary concern.

What makes a Good prediction?

Remember the data generating process (DGP):

$$Y = f(X) + \epsilon$$

- There are two (!) **Errors**:

  1. Reducible error $\hat{f}$
  2. Irredicuble error ϵ

- We can work to improve the Reducible error

- The Irreducible error is a feature of the DGP, hence, nature. Life. Karma. Measurement incurs error.

What makes a Good prediction?

Remember the data generating process (DGP):

$$Y = f(X) + \epsilon$$

- There are two (!) **Errors**:

  1. Reducible error $\hat{f}$
  2. Irredicuble error ϵ

- We can work to improve the Reducible error

- The Irreducible error is a feature of the DGP, hence, nature. Life. Karma. Measurement incurs error.

- The squared error for a given estimate $\hat{f}$ is $E[Y - \hat{Y}]^2$: Similar to *mean squared residuals*!

- One can easily show that that this factors as

$$E[f(X) + \epsilon - \hat{f}(X)]^2 = \underbrace{[f(X) - \hat{f}(X)]^2}_{\text{Reducible}} + \underbrace{Var(\epsilon)}_{\text{Irreducible}}$$

First Classification of Estimators

In general:

Nonlinear Models

- More nonlinear models are able to get closer to the data.
- Hence, they are good predictors
- But hard to **interpret**

Linear Models

- Easy to Interpret
- Less tight fit to data
- worse Prediction

How to Estimate an f?

Training Data

1. n data points $i = 1, \ldots, n$
2. y_i is i's response
3. $X_i = (x_{i1}, \ldots, x_{ip})$ are predictors
4. Data: $(X_1, y_1), \ldots, (X_n, y_n)$

(Up until now, *training* data was the only data we have encountered!)

Estimate $\hat{f}$ = Learn $\hat{f}$

There are two broad classes of learning $\hat{f}$:

1. Parametric Learning
2. Non-Parametric Learning

Parametric Methods

Parametrics Methods

Procedure

1. We make a **parametric assumption**, i.e. we write down how we think f looks like. E.g.

   $$Y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

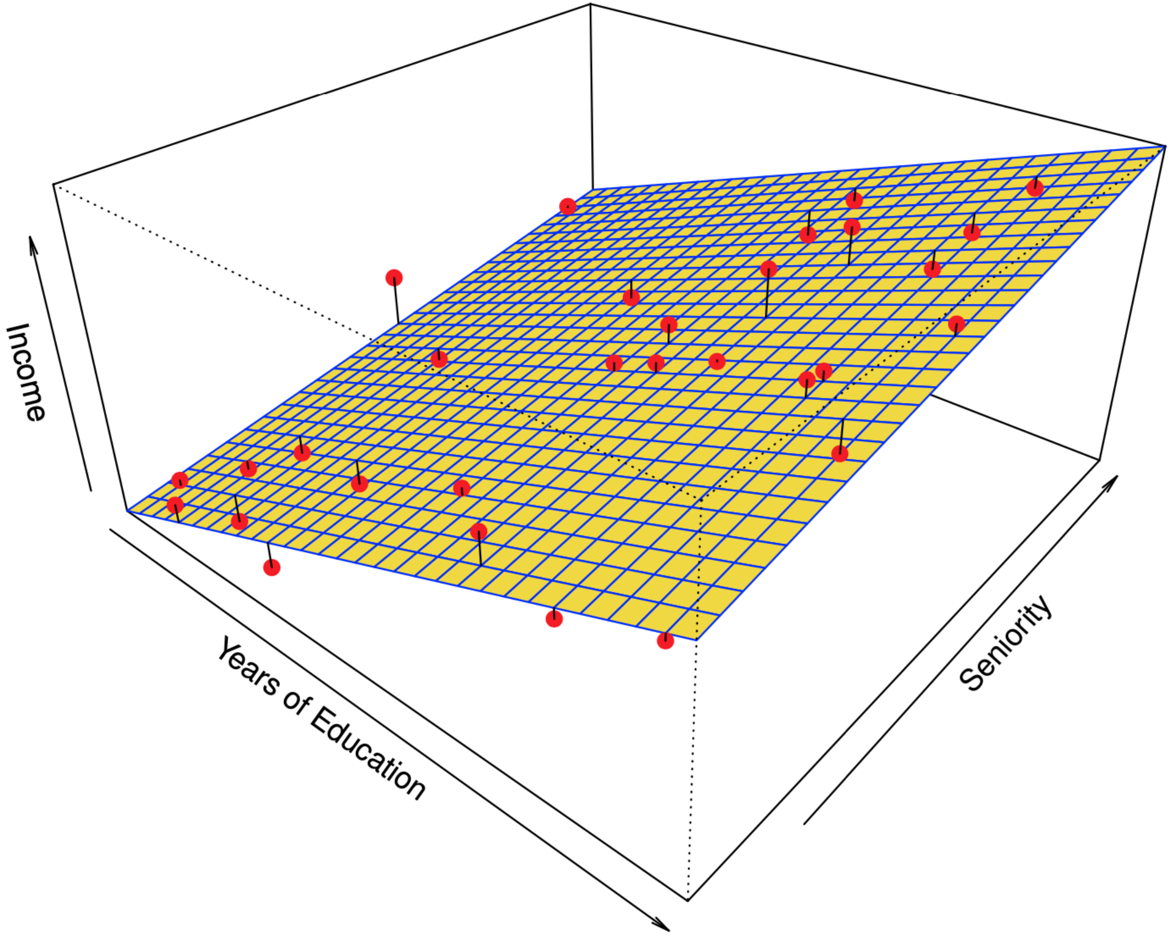
   Here we only have to find $p+1$ numbers!

2. We *train* the model, i.e. we choose the β 's. We are pretty good at that -> OLS ✌️

Potential Issues

- Typically, our model is **not** the true DGP. Why we want a model in the first place.
- If our parametric assumption is a poor model of the true DGP, we will be far away from the truth. Kind of…logical.

A Parametric Model for f



- The **yellow** plane is $\hat{f}$:

  $y = \beta_0 + \beta_1 \text{educ} + \beta_2 \text{sen}$

- It's easy to interpret (need only 3 β's to draw this!)

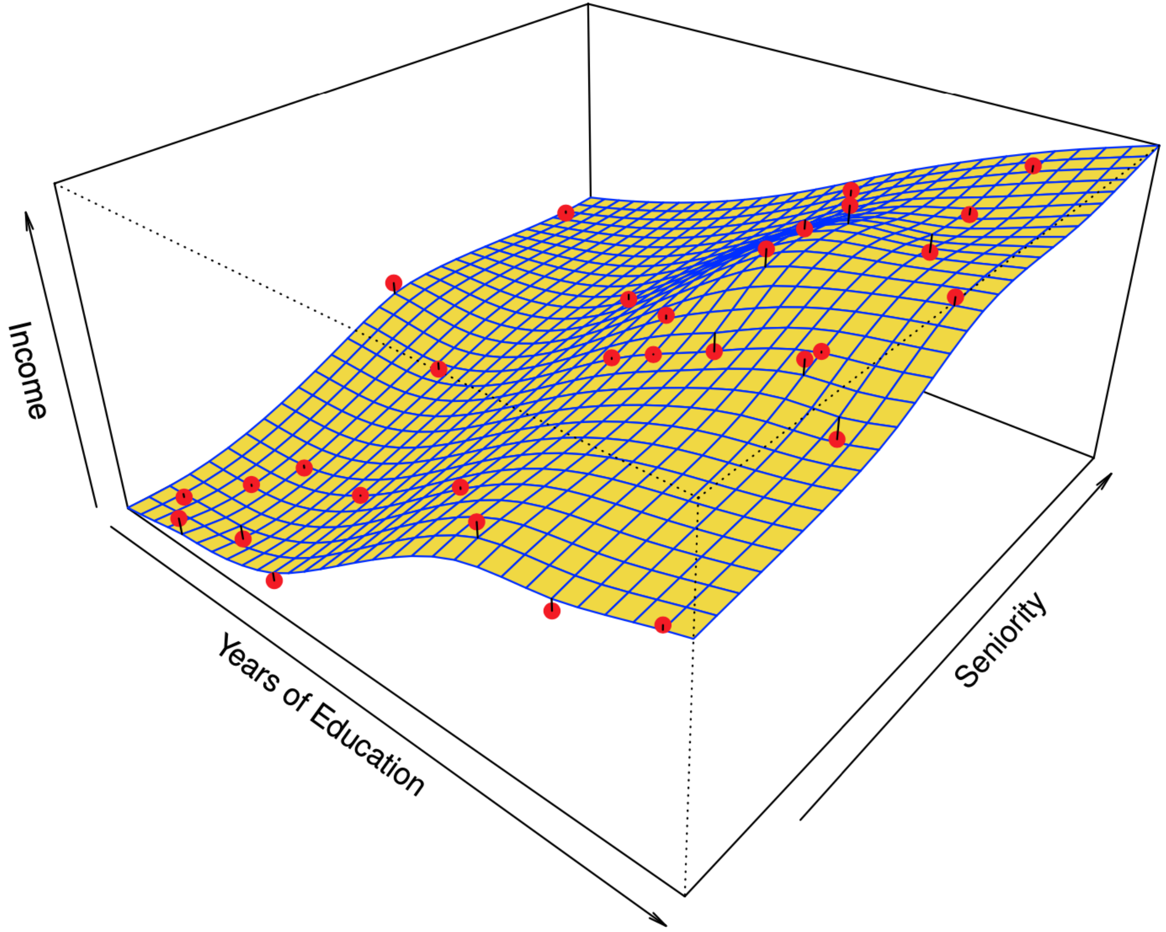
- Incurs substantial training error because it's a rigid plane (go back to blue shape to check true f).

Non-Parametric Methods

- We make a no explicit assumption about functional form.
- We try to get *as close as possible* to the data points.
- We try to do that under some contraints like:
  - Not too rough
  - Not too wiggly

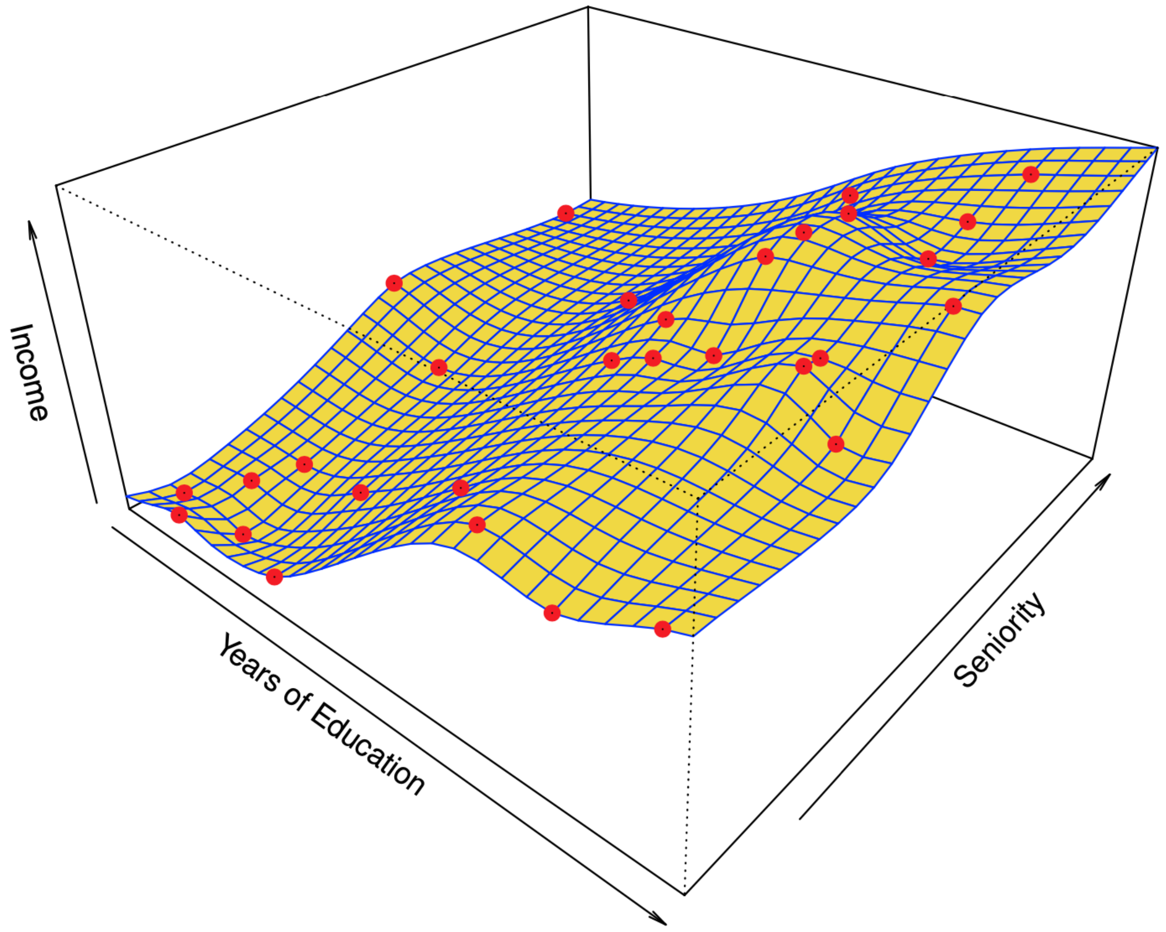
- Usually provides a good fit to the training data.
- **But** it does *not* reduce the number of parameters!
- Quite the contrary. The number of parameters increases so fast that those methods quickly run into feasibility issues (your computer can't run the model!)

A Non-Parametric Model for f



- The yellow plane is a thin-plate spline
- This clearly captures the shape of the true f (**the blue one**) better: Smaller Training Error.
- But it's harder to interpret. Is `income` increasing with `Seniority`?

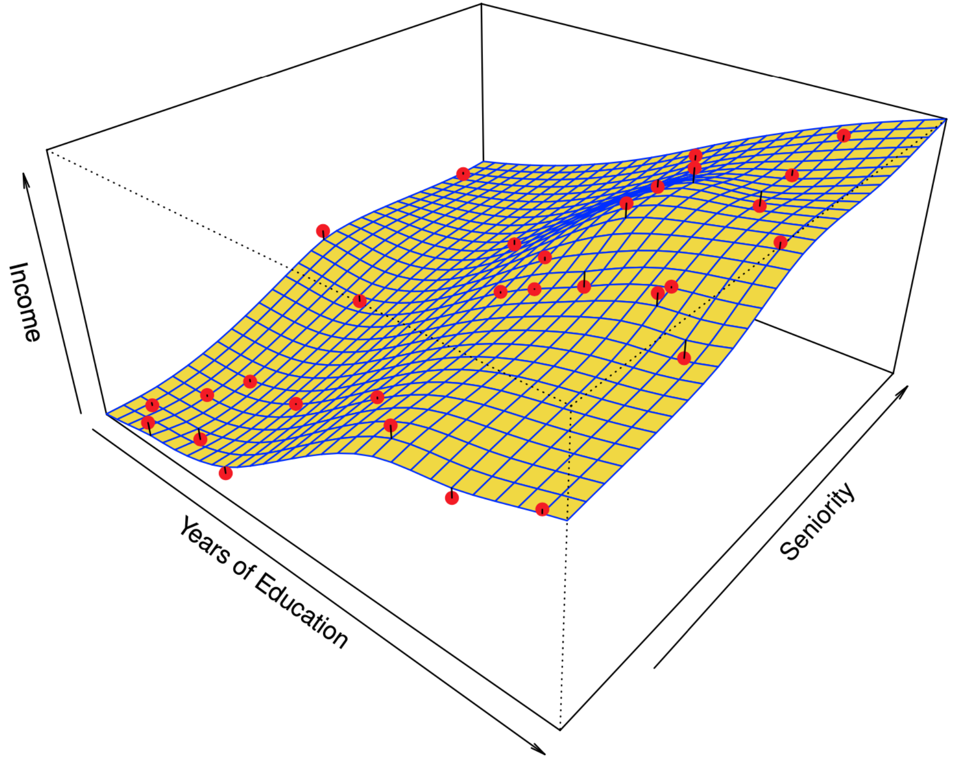
Overfitting: Choosing *Smoothness*



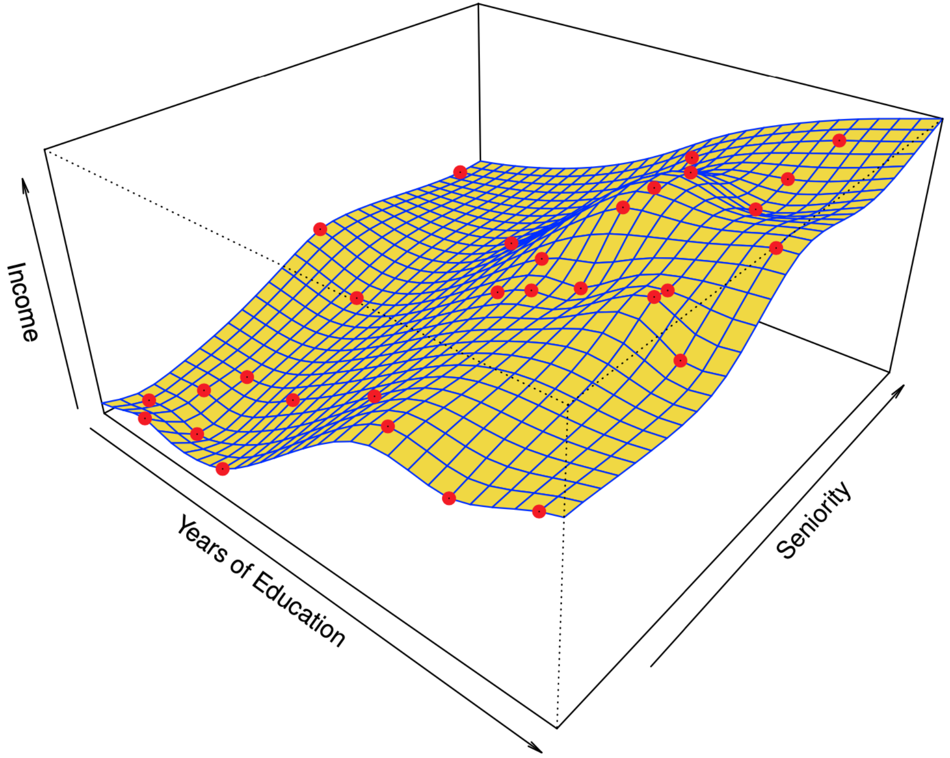
- We can choose the degree of flexibility or *smoothness* of our spline surface.
- Here we increased flexibility so much that there is **zero** training error: spline goes through all points!
- But it's a much wigglier surface now than before! Even harder to interpret.

Overfitting: Choosing *Smoothness*

Smooth, not wiggly

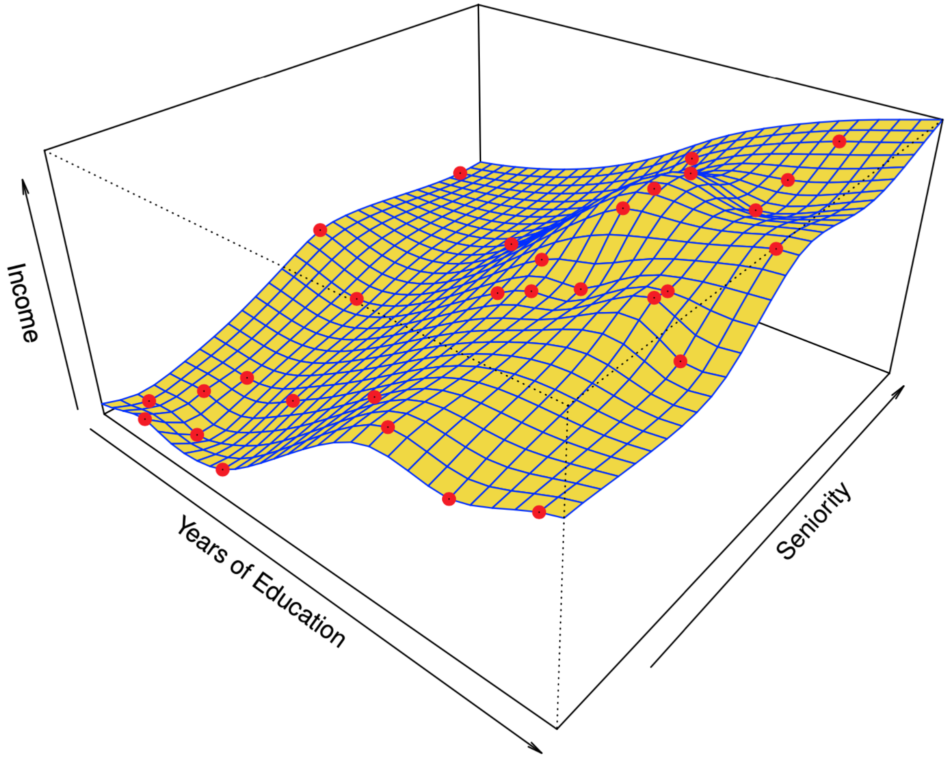


Smooth but high variance (wiggly!)



Overfitting: Over-doing it

- You can see that the researcher has an active choice to make here: *how smooth*?
- Parameters which guide choices like that are called **tuning parameters**.
- As $\hat{f}$ becomes too variable, we say there is **overfitting**: The model tries too hard to fit patterns in the data, which are **not** part of the true f!



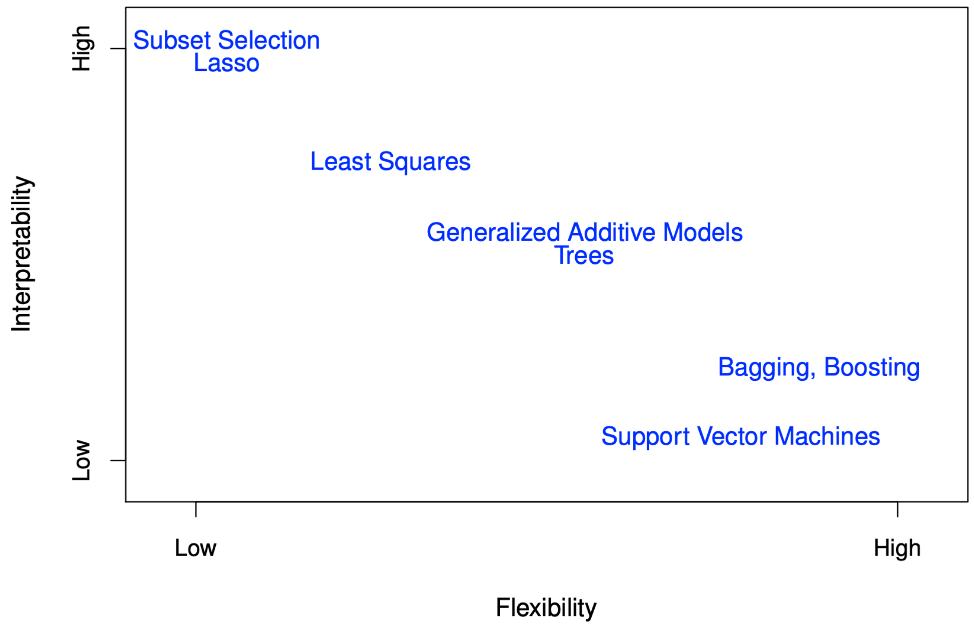
What Method To Aim For?

> Why would we not always want the most flexible method available?

- that's a reasonable question to ask.
- The previous slide already gave a partial answer: more flexbility generally leads to more variability.
- If we want to use our model outside of our training data set, that's an issue.

Classifying Methods 1: **flexibility** vs **interpretability**

- This graph offers a nice classification of statistical learning methods in **flexibility** vs **interpretability** space.
- Sometimes it's obvious what the right choice is for your application.
- But often it's not. It's a more complicated tradeoff than the picture suggests.
- (It's a very helpful picture!)
- We will only be touching upon a small number of those. They are all nicely treated in the ISLR book though!



Classifying Methods 2: Supervised vs Unsupervised Learning

Supervised Learning

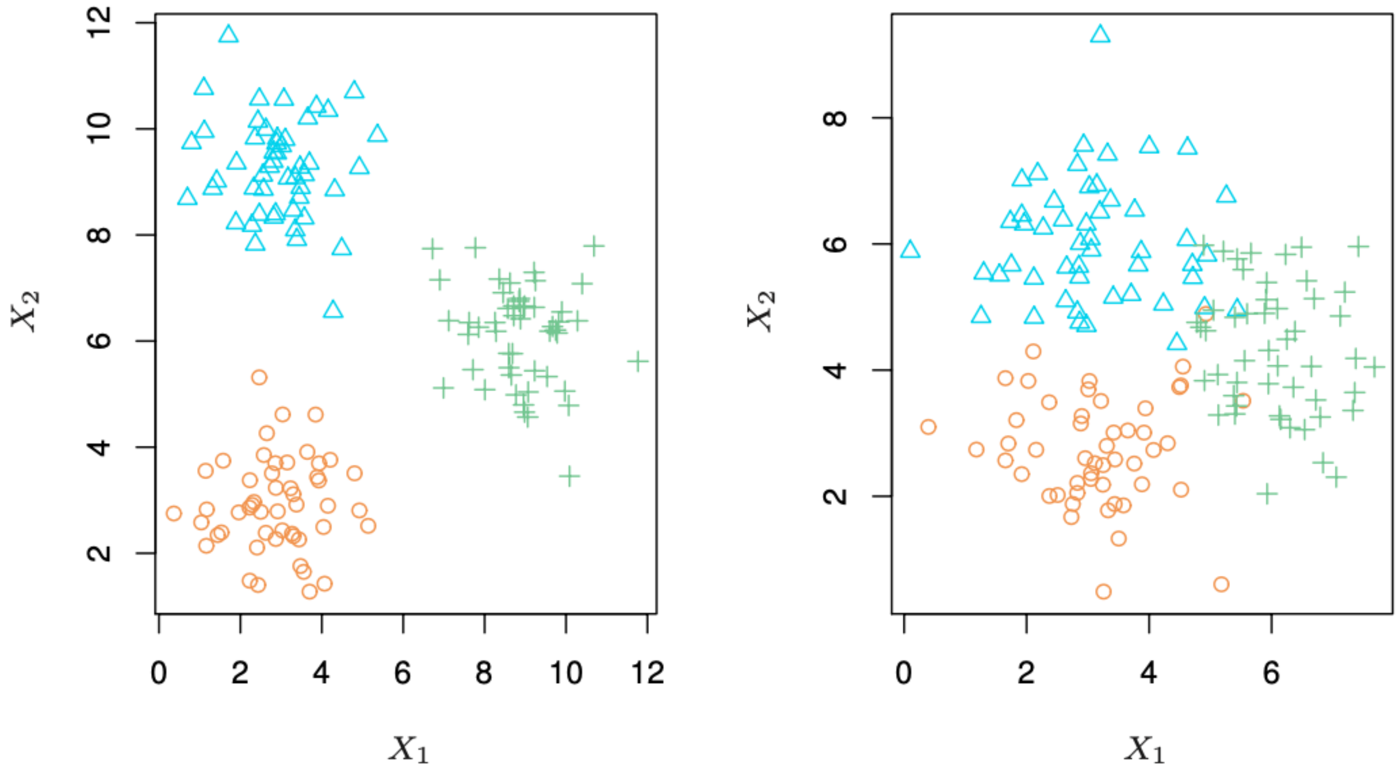
- We have measures of input x and output y
- We could *predict* new y's
- Or infer things about `Y ~ X`
- Regression or Classification are typical tasks

Unsupervised Learning

- We have **no** measure of output y!
- Only a bunch of x's
- We are interested in *grouping* of those x (**cluster analysis**)

Clustering Example

- Sometimes clustering is easy: in the left panel the data fall naturally into groups.
- When data overlap, it's harder: right panel



Assessing Model Accuracy

What is a good model?

Quality of Fit: the Mean Squard Error

- We know the **mean squared error (MSE)** already:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

- We encountered the closely related **sum of squared residuals (SSR)**:

$$SSR = \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

- As we know, OLS minimizes the SSR. (minimizing SSR or MSE yields the same OLS estimates.)

Quality of Fit: the Mean Squard Error

- We know the **mean squared error (MSE)** already:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

- We encountered the closely related **sum of squared residuals (SSR)**:

$$SSR = \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

- As we know, OLS minimizes the SSR. (minimizing SSR or MSE yields the same OLS estimates.)

- However, what MSE 👈 really is: it's the **training MSE**! It's computed using the *same* data we used to compute $\hat{f}$!

- Suppose we used data on last 6 months of stock market prices and we want to predict future prices. *We don't really care how well we can predict the past prices*.

- In general, we care about how $\hat{f}$ will perform on **unseen** data. We call this **test data**.

Training MSE vs Test MSE

Training

- We have a *training data set*

$$\{(y_1, x_1), \ldots, (y_n, x_n)\}$$

- we use those n observations to find the function q that minimizes the **Training MSE**:

$$\hat{f} = \arg\min_q MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - q(x_i))^2$$

Training MSE vs Test MSE

Training

- We have a *training data set*

$$\{(y_1, x_1), \ldots, (y_n, x_n)\}$$

- we use those n observations to find the function q that minimizes the **Training MSE**:

$$\hat{f} = \arg\min_q MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - q(x_i))^2$$

Testing

- We want to know whether $\hat{f}$ will perform well on *new* data.
- Suppose (y_0, x_0) is unseen data - in particular, we haven't used it to train our model!
- We want to know the magnitude of the **test MSE**:

$$E[(y_0 - \hat{f}(x_0))^2]$$

A Problem of MSEs

- In many cases we don't have a true test data set at hand.
- Most methods therefore try to minimize the training MSE. (OLS does!)
- At first sight this seems really reasonable.

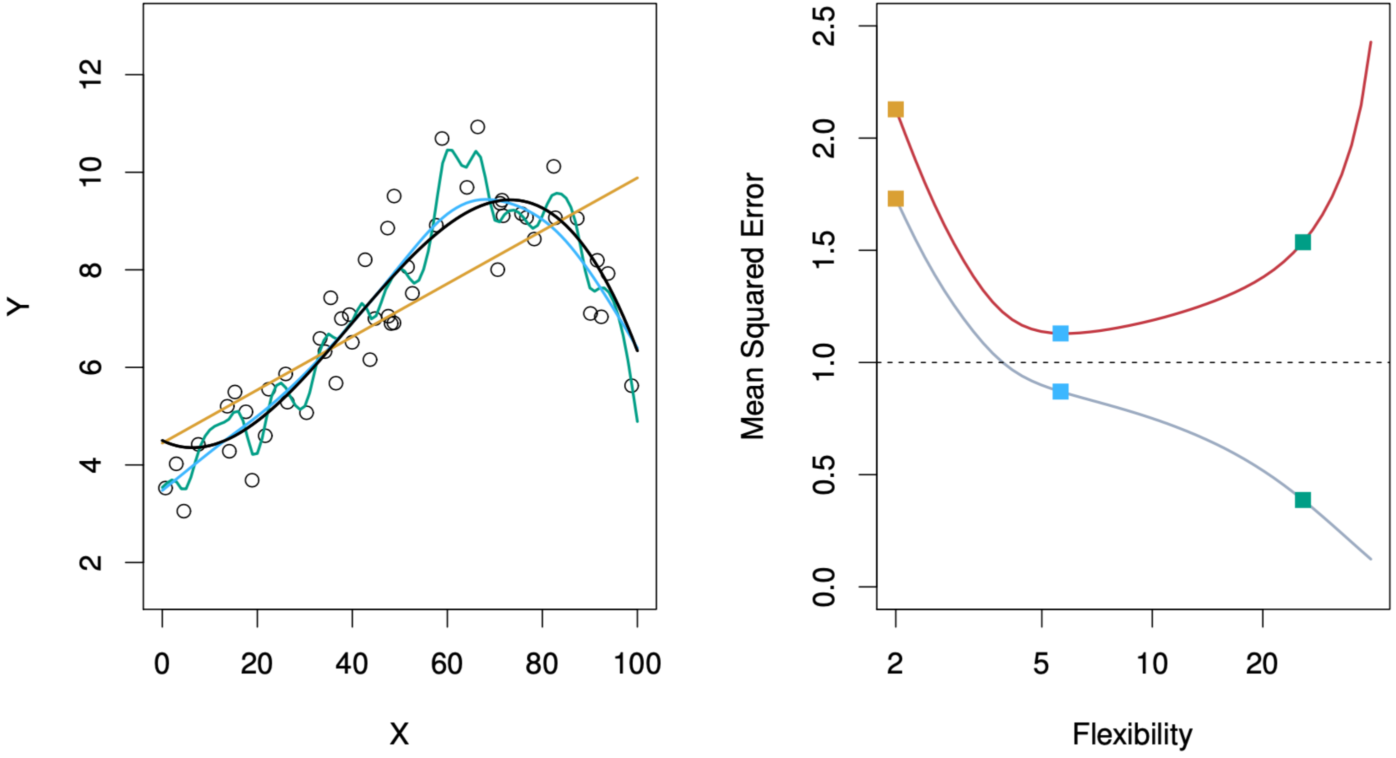
A Problem of MSEs

- In many cases we don't have a true test data set at hand.
- Most methods therefore try to minimize the training MSE. (OLS does!)
- At first sight this seems really reasonable.

- The problem is that test and training MSE are less closely related than one might think!
- Very small training MSEs might go together with pretty big test MSEs!
- That is, most methods are *really* good at fitting the training data, but they fail to generalize to outside of that set of point!

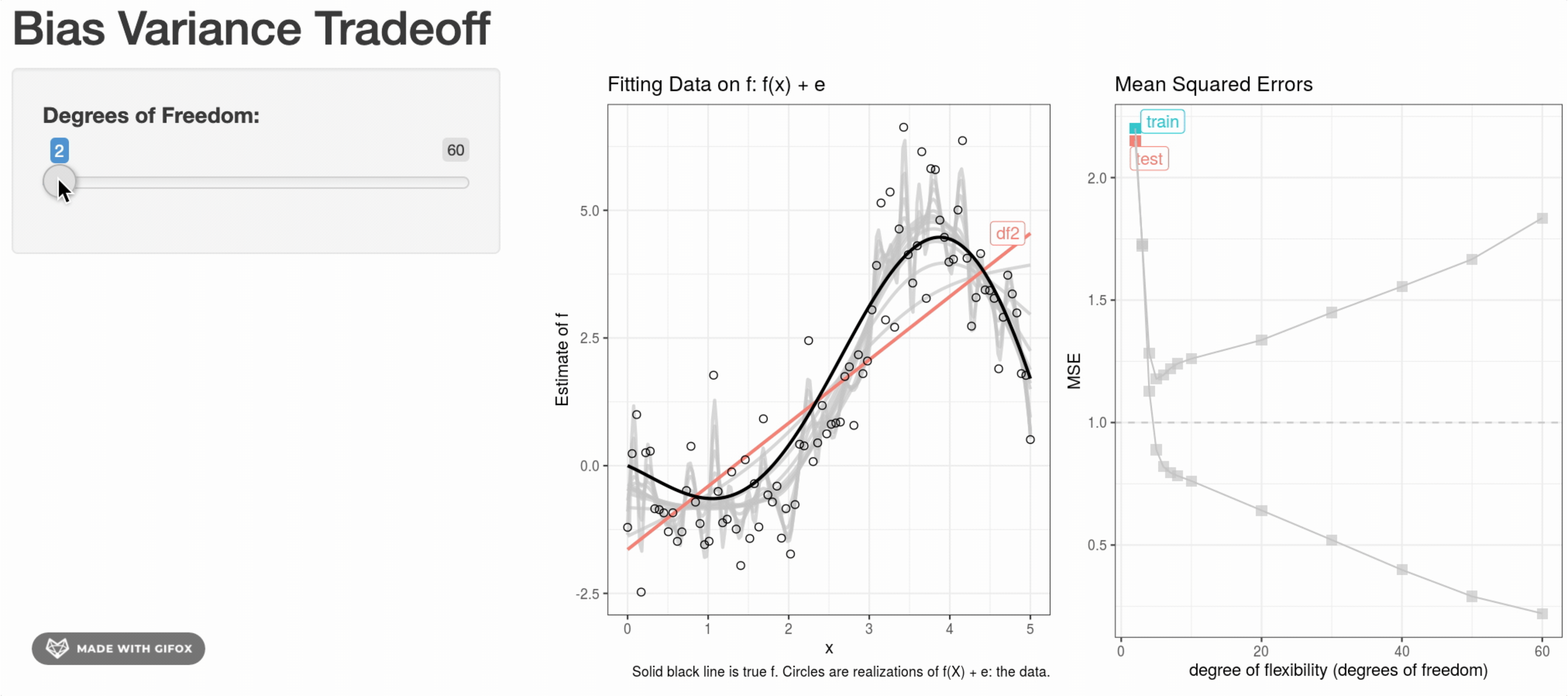
Simulation: We *know* the test data!

- In an artifical setting we now the test data because we know the true f.
- Here Solid black line. 👉
- Increasing flexibility mechanically reduces training error (grey curve in right panel.)
- However not the test MSE, in general (red curve!)



Simulation: App!

- Let's look at our app online or `ScPoApps::launchApp("bias_variance_tradeoff")`

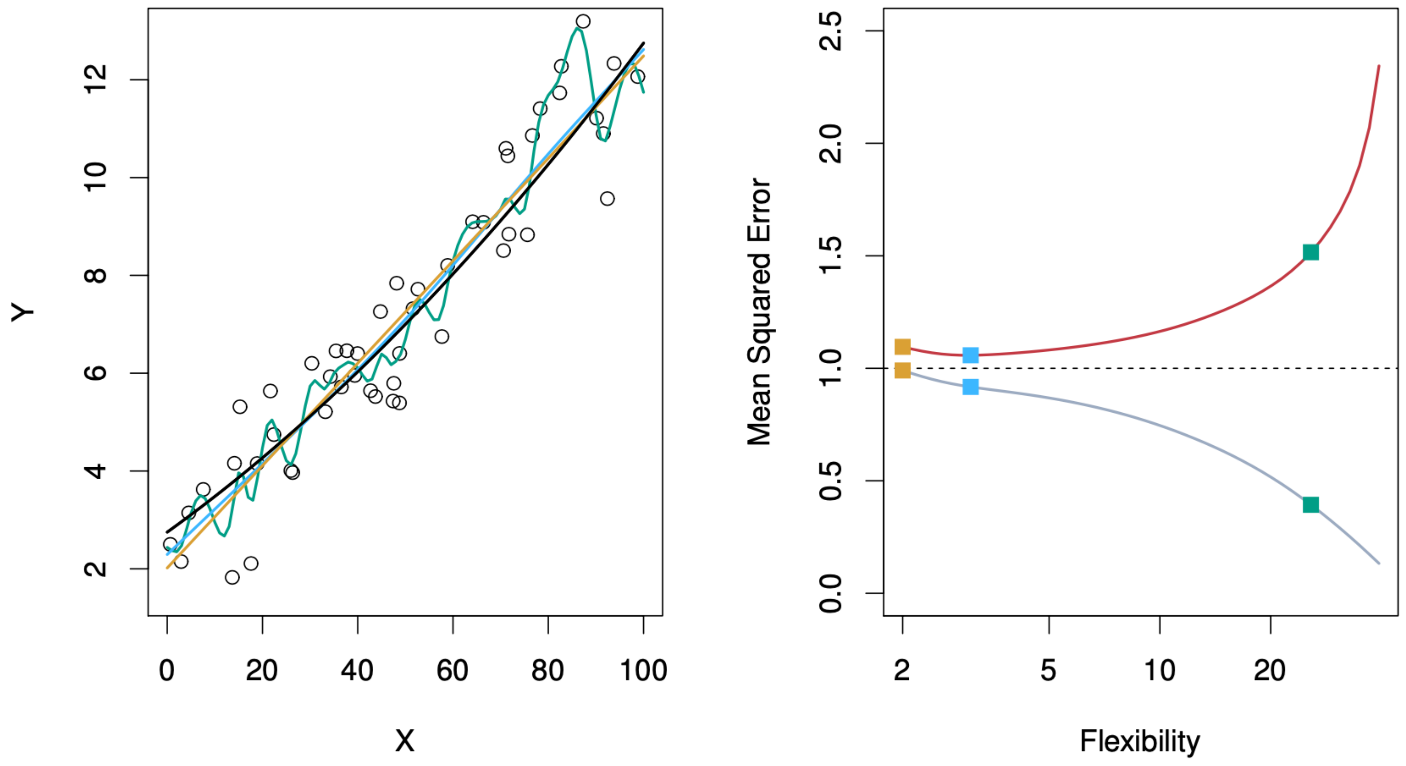


So! A Tradeoff at Last!

- What's going on here?
- Initially, increasing flexibility provides a better fit to the observed data points, decreasing the training error.
- That means that also the test error decreases for a while.
- As soon as we start **overfitting** the data points, though, the test error starts to increase again!
- At very high flexibility, our method tries to fit patterns in the data which are *not* part of the true f (the black line)!
- To make matters worse, the extent of this phenomenon will depend on the shape of the underlying **true** f!

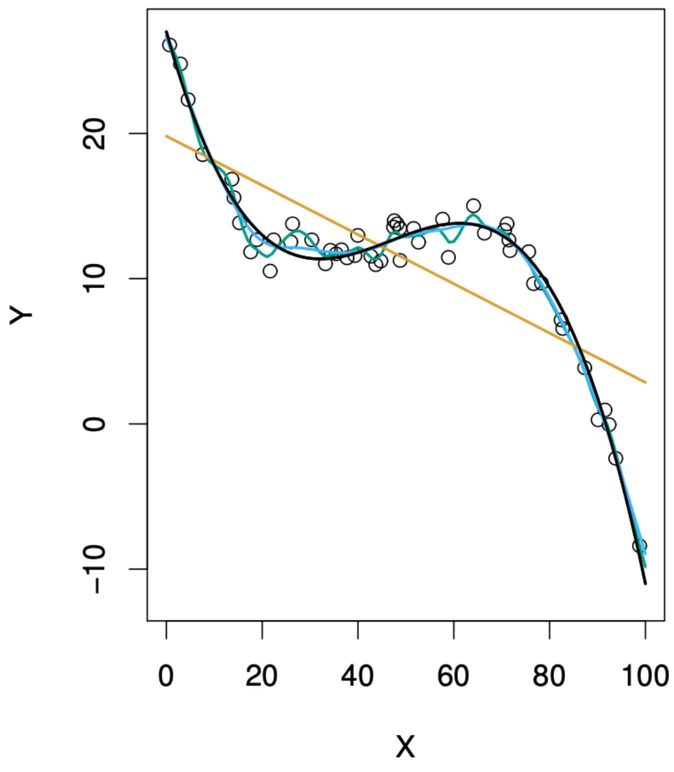
Almost linear f

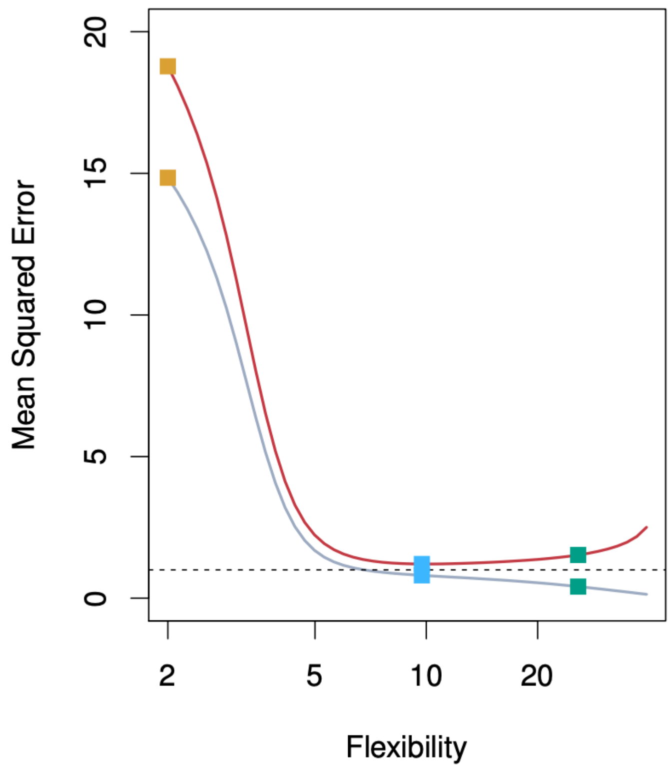
- In this example, the true f is almost linear.
- The inflexible method does well!
- Increasing flexibility incurs large testing MSE.



Highly Non-linear f

- In this example, the true f is very non linear.
- The inflexible method does very poorly in both trainign and testing MSE.
- the model at 10 degrees of freedom performs best here.
- 👉 You can see that the best model is not obvious to choose!





Formalizing the Bias-Variance-Tradeoff

- We can decompose the expected test MSE as follows:

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + \left[\text{Bias}(\hat{f}(x_0))\right]^2 + Var(\epsilon)$$

- From this we can see that we have to minimize **both** variance and bias when chooseing a suitable method.
- We have seen before that those are competing forces in some situations.
- Notice that the best we could achieve is $Var(\epsilon) > 0$ since that is a feature of our DGP.

Bias-Variance-Tradeoff: What are Bias and Variance?

Variance

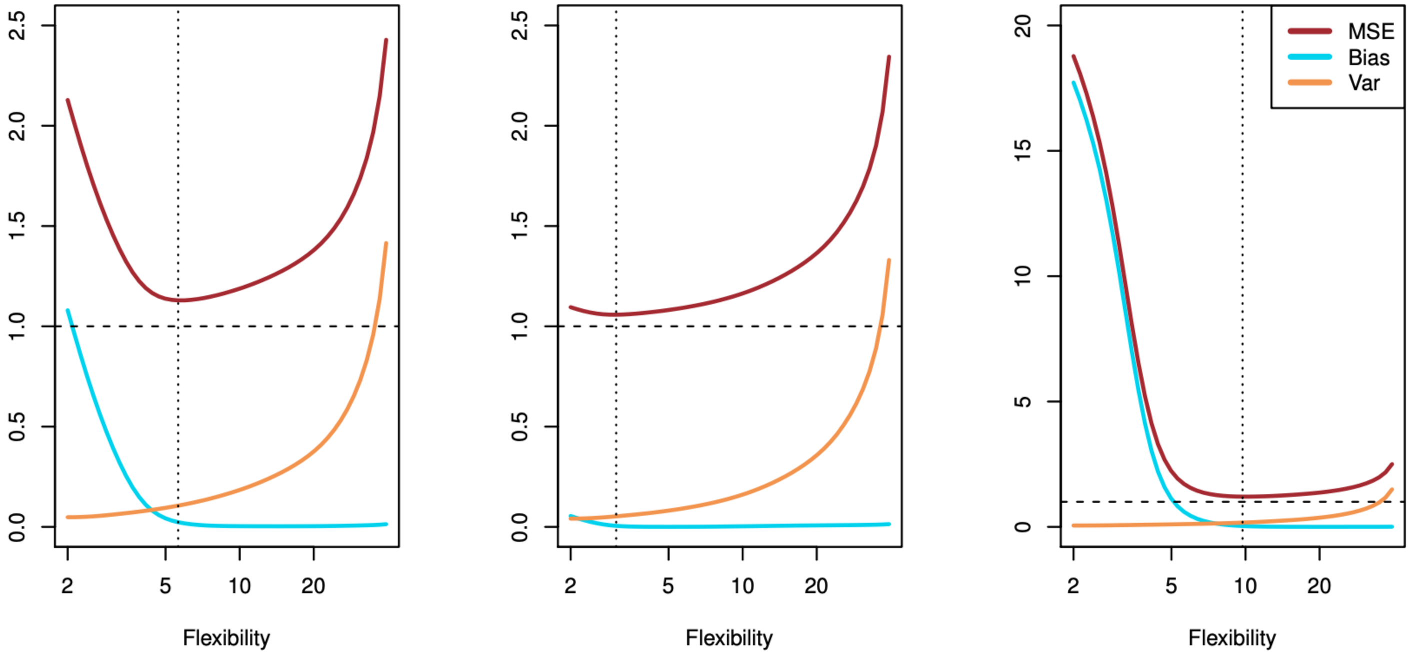
- How much would $\hat{f}$ change if we estimated it using a **different** data set?
- Clearly we expect some variation when using different samples (sampling variation), but not too much.
- Flexibel models: moving just a single data point will result in a large change in $\hat{f}$.

Bias

- The difference between $\hat{f}$ and f (notice the missing ϵ).
- We approximate a potentially very complex real phenomenon by a *simple* model, e.g. linear model.
- If true model highly non-linear, linear model will be biased.
- General: more flexible, lower bias but higher variance.

Bias Variance Tradeoff vs Flexibility

- Here 👉 we illustrate for preceding 3 true f's
- Precise Tradeoff depends on f's shape.
- Bias declines with flexibility.
- Test MSE is U-shaped, Var increasing.



SciencesPo
DEPARTMENT OF ECONOMICS

END

	bluebery.planterose@sciencespo.fr
	Original Slides from Florian Oswald
	Book
	@ScPoEcon
	@ScPoEcon