Using splines in regression

Author: Nicholas G Reich, Andrea Foulkes

This material is part of the **statsTeachR** *project*

Today's Lecture

- Assessing model accuracy
- Overfitting
- Fitting smooth curves to data

More info:

- *ISL* Chapter 7

Assessing model accuracy: Regression setting

- The mean squared error (MSE) is the most commonly used measure of the performance of a statistical learning method in the regression setting:

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

- The MSE will be small if the observed and predicted responses are close to one another, and it will be large if these differ substantially, in at least some instances.

Robust studies replicate results on 'external data'

- A good model should have a robust "validation" of the model.
- Ideally, this means replicating results on a new dataset.
- Example 1: a study done by investigators at Institution A shows that in a sample of 100 cancer patients, a specific set of commonly collected biomarkers accurately classified cancer patients based on severity of outcome at 1 year follow-up. What would be a good replication study?
- Example 2: using photos from a common social media platform, authors claimed that their face recognition algorithm could accurately predict the sexual orientation of individuals based solely on the photo. What would be a good replication study?

Often, an ‘internal‘ test sample is used

- In general, we are interested in the accuracy of the predictions that we obtain when we apply our method to previously unseen data $\rightarrow$ called the test sample.
- Often, obtaining an external validation or testing dataset is infeasible or expensive.
- In these cases, investigators may set aside a portion of the observations from the original dataset as a test sample.
- Test samples can be particularly useful in diagnosing whether your model was “overfit“ to the particular features of the training dataset.

Hands-on example: FEV dataset

The FEV dataset describes a sample of 654 youths, aged 3 to 19, in the area of East Boston during middle to late 1970's. Data includes the following variables (among others)

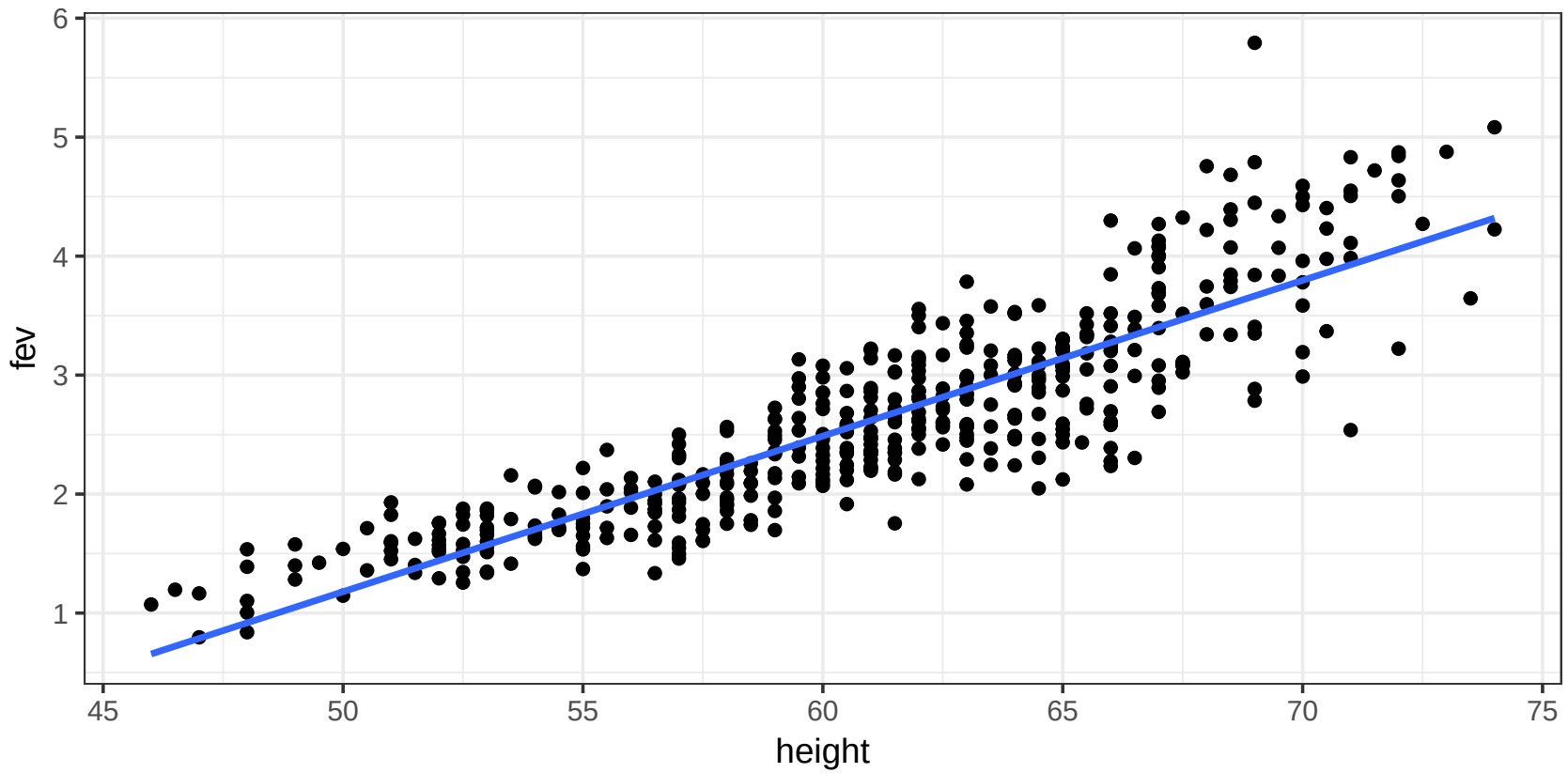
- fev - forced expiratory volume, a measure of lung capacity and strength (in liters)
- height (in inches)

```
library("Hmisc")
getHdata(FEV)
head(FEV)
```

```
##     id age   fev height    sex              smoke
## 1  301   9 1.708   57.0 female non-current smoker
## 2  451   8 1.724   67.5 female non-current smoker
## 3  501   7 1.720   54.5 female non-current smoker
## 4  642   9 1.558   53.0   male non-current smoker
## 5  901   9 1.895   57.0   male non-current smoker
## 6 1701   8 2.336   61.0 female non-current smoker
```

Creating an FEV test sample

```
set.seed(756)  ## so we all get the same result
idx_test <- sample(1:nrow(FEV), size=200) ## 200 test observations
fev_train <- FEV[-idx_test,]
fev_test <- FEV[idx_test,]
ggplot(fev_train, aes(height, fev)) + geom_point() +
  geom_smooth(method="lm", se=FALSE)
```



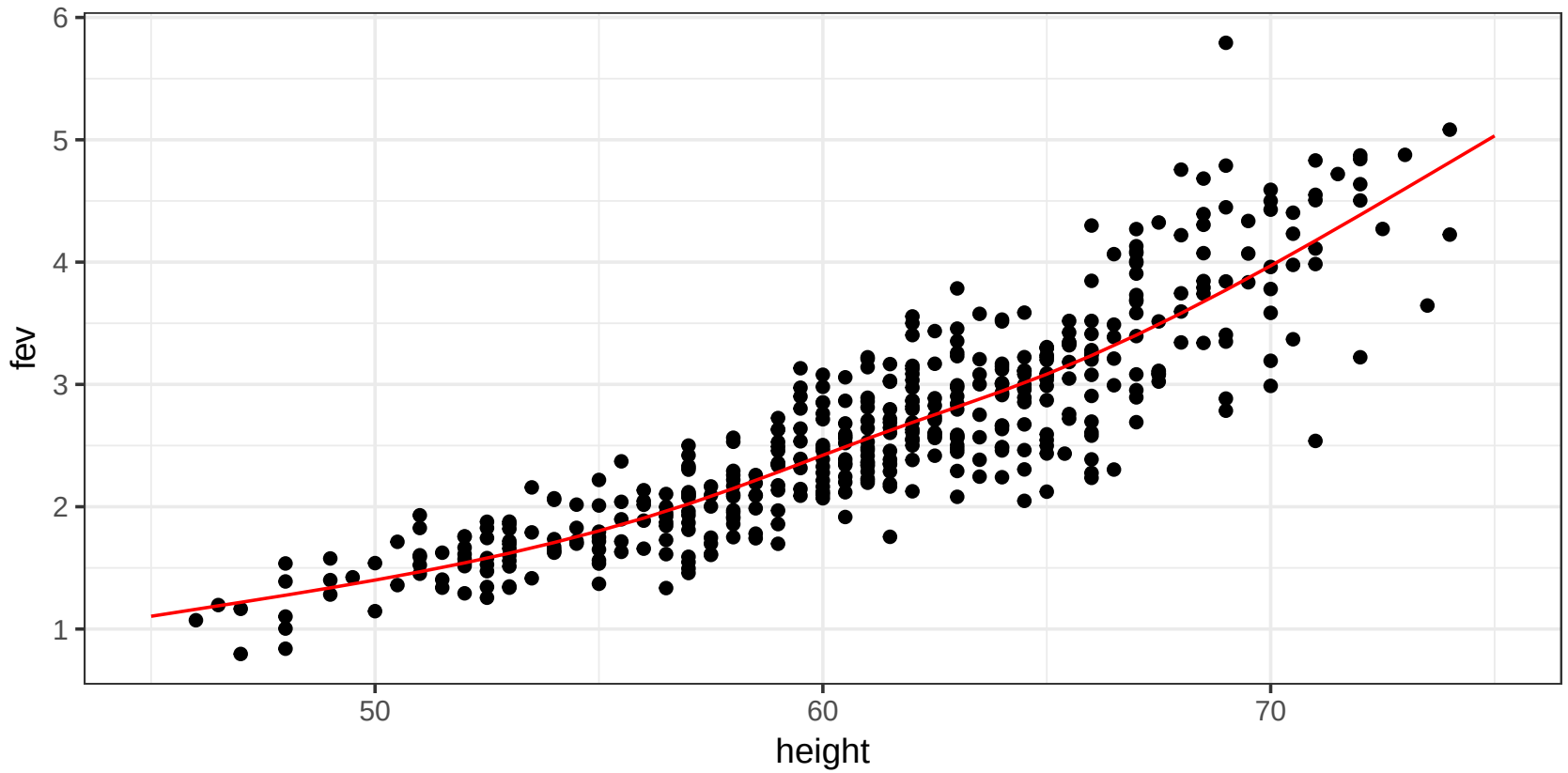
Using a smooth model for *fev* $\sim$ *height*

Let's fit a model to the data that can capture some of what appears to be a non-linear relationship between height and fev. The `ns()` function in the splines package uses a "natural spline" to model a flexible, non-linear relationship bewteen a covariate and an outcome by splicing together polynomial functions. The larger the `df` parameter is, the more functions it pieces together and therefore the more wiggly the fitted model becomes.

```
library(splines)
spline_mdl <- lm(fev ~ ns(height, df = 4), data=fev_train)
```

Visualizing the smooth model

```
spline_x_vals <- seq(45, 75, by=1)
spline_y_vals <- predict(spline_mdl,
  newdata = data.frame(height=spline_x_vals))
ggplot() +
  geom_point(aes(x=height, y=fev), data=fev_train) +
  geom_line(aes(x = spline_x_vals, y=spline_y_vals), color="red")
```



Find the Goldilocks zone!

That model looks good, but how to we know what degree of smoothness is just the right amount? We can compare MSE between our training and our test set for different levels of smoothness.

```
fev_train$spline_preds <- predict(spline_mdl)
fev_test$spline_preds <- predict(spline_mdl, newdata = fev_test)
( mse_train <- mean((fev_train$fev - fev_train$spline_preds)^2) )

## [1] 0.1578606

( mse_test <- mean((fev_test$fev - fev_test$spline_preds)^2) )

## [1] 0.1934739
```

With your group

- At your tables, assign each individual one value of `df` to test. Work together to make sure that at least one `df` is run on everyone's computer.
- Compile a dataset (on the whiteboard and/or on one computer for the table) with the `df` value and the associated MSEs for both the training and testing samples.
- Discuss your results as a table and decide on the optimal `df` to choose.

Using a test sample avoids overfitting

- The mean squared error will always get lower with the training sample as we add more features to our model.
- However, we want to find the model that minimizes the test sample MSE.
- Overfitting refers to the situation in which a less complex model would result in a smaller test sample MSE.

MSE: learning versus test sample

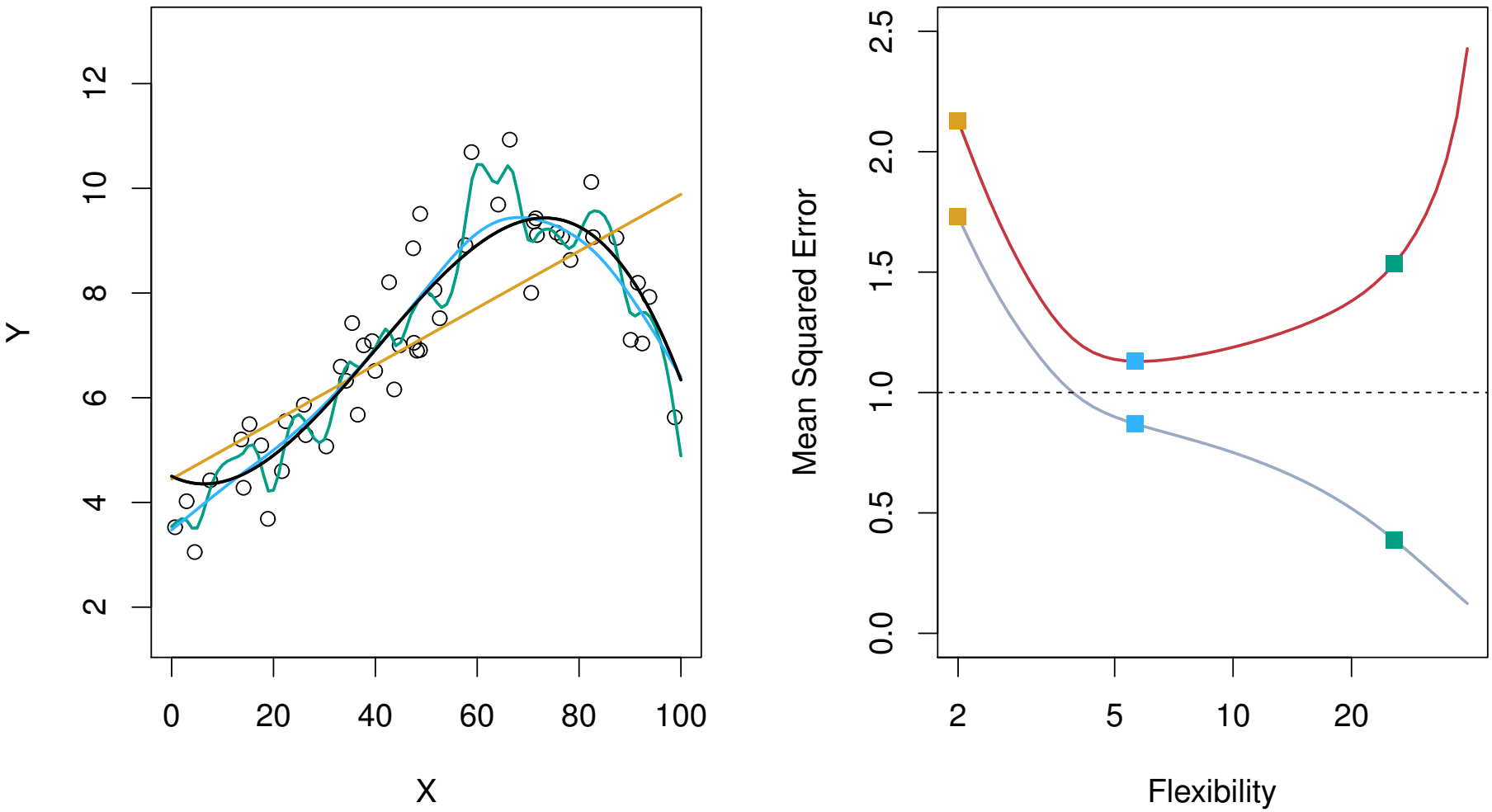


Figure 2.9 from ISL. Black line is the true relationship from which data are simulated

MSE: learning versus test sample

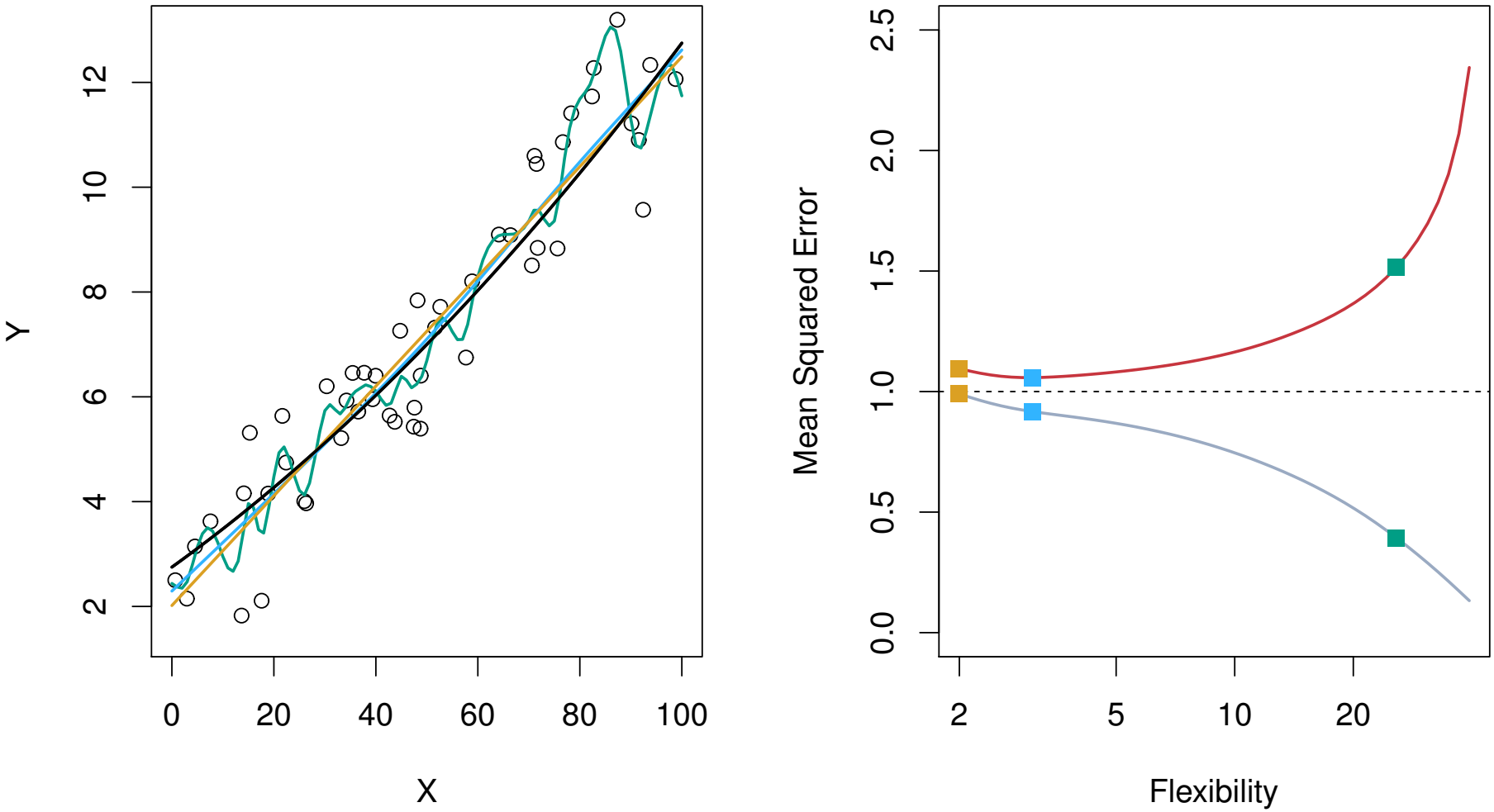


Figure 2.10 from ISL.