Module 5: Resampling

TMA4268 Statistical Learning V2020

Stefanie Muff, Department of Mathematical Sciences, NTNU

February xx, 2020

Introduction

Learning material for this module

- James et al (2013): An Introduction to Statistical Learning. Chapter 5.
- Classnotes 04.02.2019

Additional material for the interested reader: Chapter 7 (in particular 7.10) in Friedman et al (2001): Elements of Statistical learning.

Some of the figures and slides in this presentation are taken (or are inspired) 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.

What will you learn?

- What is model assessment and model selection?
- Ideal solution in a data rich situation.
- Cross-validation and what is best:
  - validation set
  - leave-one-out cross-validation (LOOCV)
  - k-fold CV
- Bootstrapping - how and why.
- Summing up
- The plan for the interactive lesson.

Performance of a learning method

Our models are "good" when they can generalize → We want a learning method to perform well on new data.

Why?

- Prediction capacity on independent test data
- Inference and understanding the true pattern (in contrast to overfitting)

This is important both for

Model selection

Estimate the *performance* of different models (often different order of complexity within one model class) to *choose the best model*.

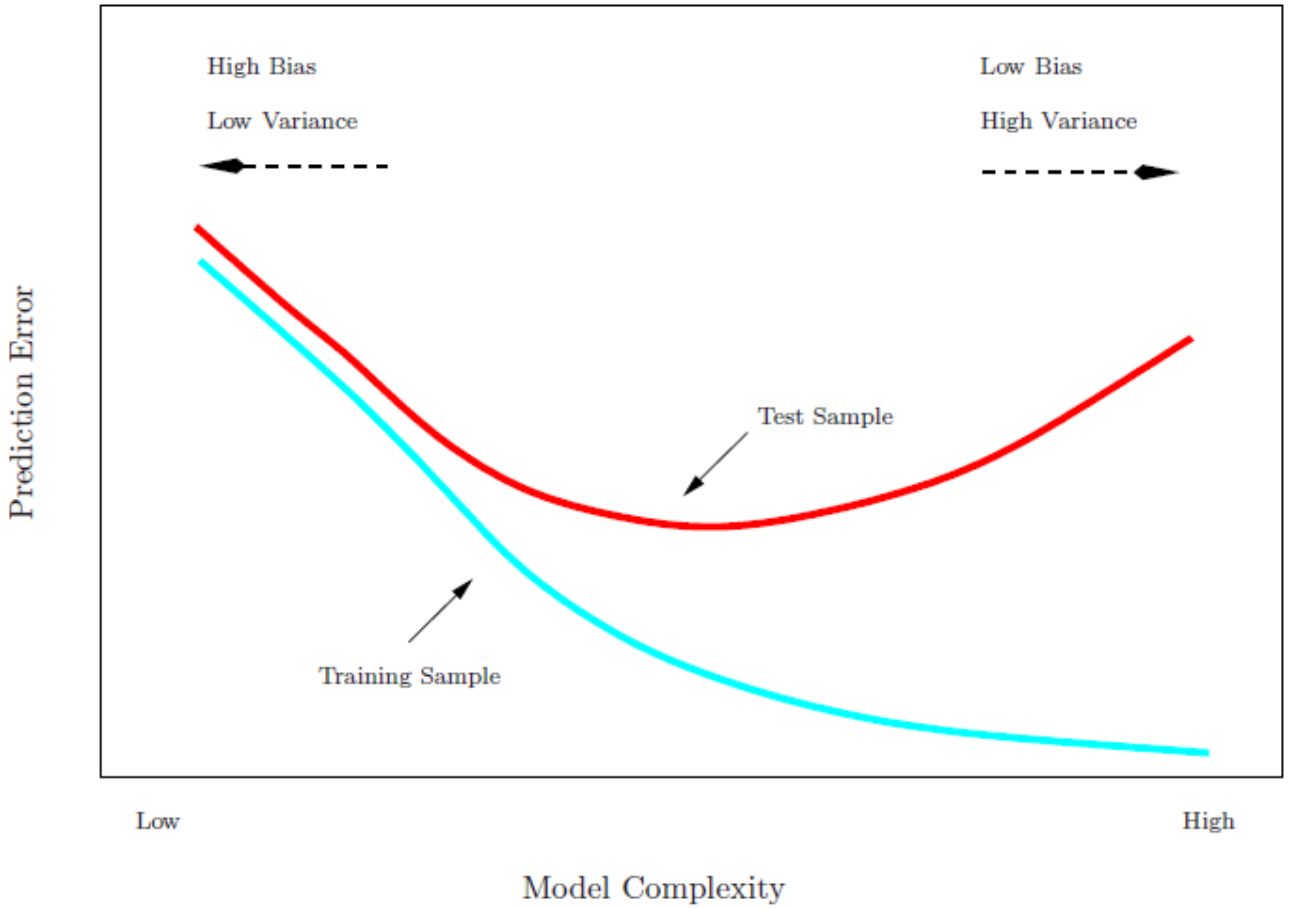
Model assessment

Estimating the performance (prediction error) of the final model, on new data.

Training vs Test Error

Recall:

- The *test error* is the average error that results from using a statistical learning method to predict the response on a new observation, one that was not used in training the method.
- The *training error* can be easily calculated by applying the statistical learning method to the observations used in its training.
- The training error rate often is quite different from the test error rate.
- **The training error can dramatically underestimate the test error**.


High Bias
Low Variance
Low Bias
High Variance
Test Sample
Training Sample
Prediction Error
Low
High
Model Complexity

Loss functions

In order to define how we measure error, we must first decide for a **loss function**. Here we use:

- *Mean squared error* (quadratic loss) for regression problems (continuous outcomes) $Y_i = f(\boldsymbol{x}_i) + \varepsilon_i$, $i = 1, \ldots, n$:

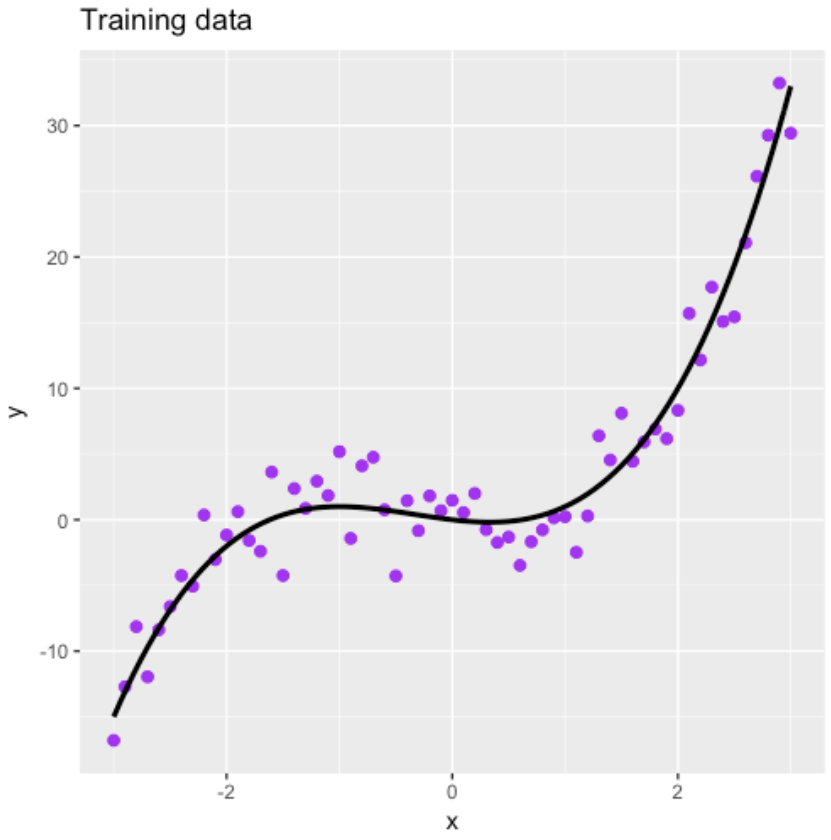
$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(\boldsymbol{x}_i))^2 \ .$$

- *Misclassification rate* (0/1 loss) for classification problems where we classify to the class with the highest probability $P(Y = j \mid \boldsymbol{x}_0)$ for $j = 1, \ldots, K$:

$$\frac{1}{n} \sum_{i=1}^{n} \text{I}(y_i \neq \hat{y}_i) \ .$$

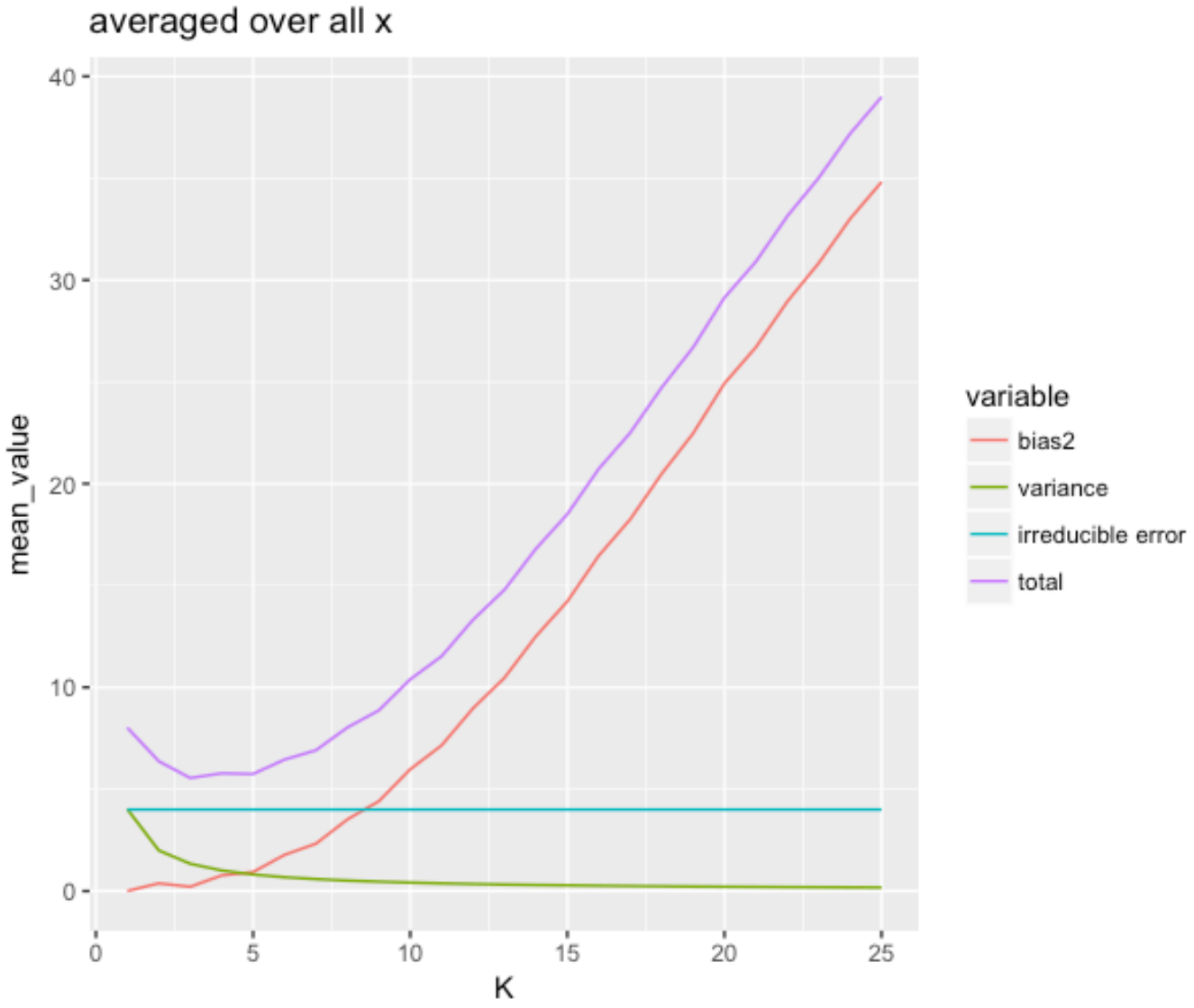
Example (from recommended exercises 2)

We aim to do *model selection* in KNN-regression, where true curve is $f(x) = -x + x^2 + x^3$ with $x \in [-3, 3]$. $n = 61$ for the training data.



Remember: The bias-variance trade-off

For KNN: K small = high complexity; K large = low complexity.



The challenge

- In the above examples we knew the truth, so we could assess training and test error.
- In reality this is of course not the case.
- We need approaches that work with real data!

The data-rich situation (often unrealistic)

If we had a large amount of data we could divide our data into three parts:

- **Training set**: to fit the model
- **Validation set**: to select the best model (*model selection*)
- **Test set**: to assess how well the model fits on new independent data (*model assessment*)

Q: Before we had just training and test. Why do we need the additional validation set?
A: We have not discussed model selection before.
Q: Why can't we just use the training set for training, and then the test set both for model selection and for model evaluation?
A: We will be too optimistic if we report the error on the test set when we have already used the test set to choose the best model.

- If you have a lot of data – great – then you do not need Module 5.
- But, this is very seldom the case – so we will study other solutions based on efficient sample reuse with *resampling* data.
- An alternative strategy for model selection (using methods penalizing model complexity, e.g. AIC or lasso) is covered in Module 6.

We will look at *cross-validation* and the *bootstrap*.

Cross-validation (CV)

Consider the following "model selection" situation: We assume that test data is available (and has been put aside), and we want to use the rest of our data to both fit the data and to find the best model.

This can be done by:

- the validation set approach (not strictly a *cross*-validation approach)
- leave one out cross-validation (LOOCV)
- k-fold cross-validation (CV), typically $k = 5$ or 10

We will also discuss that there is a "right and a wrong way" to to CV

- selection bias - all elements of a model selection strategy need to be within the CV-loop

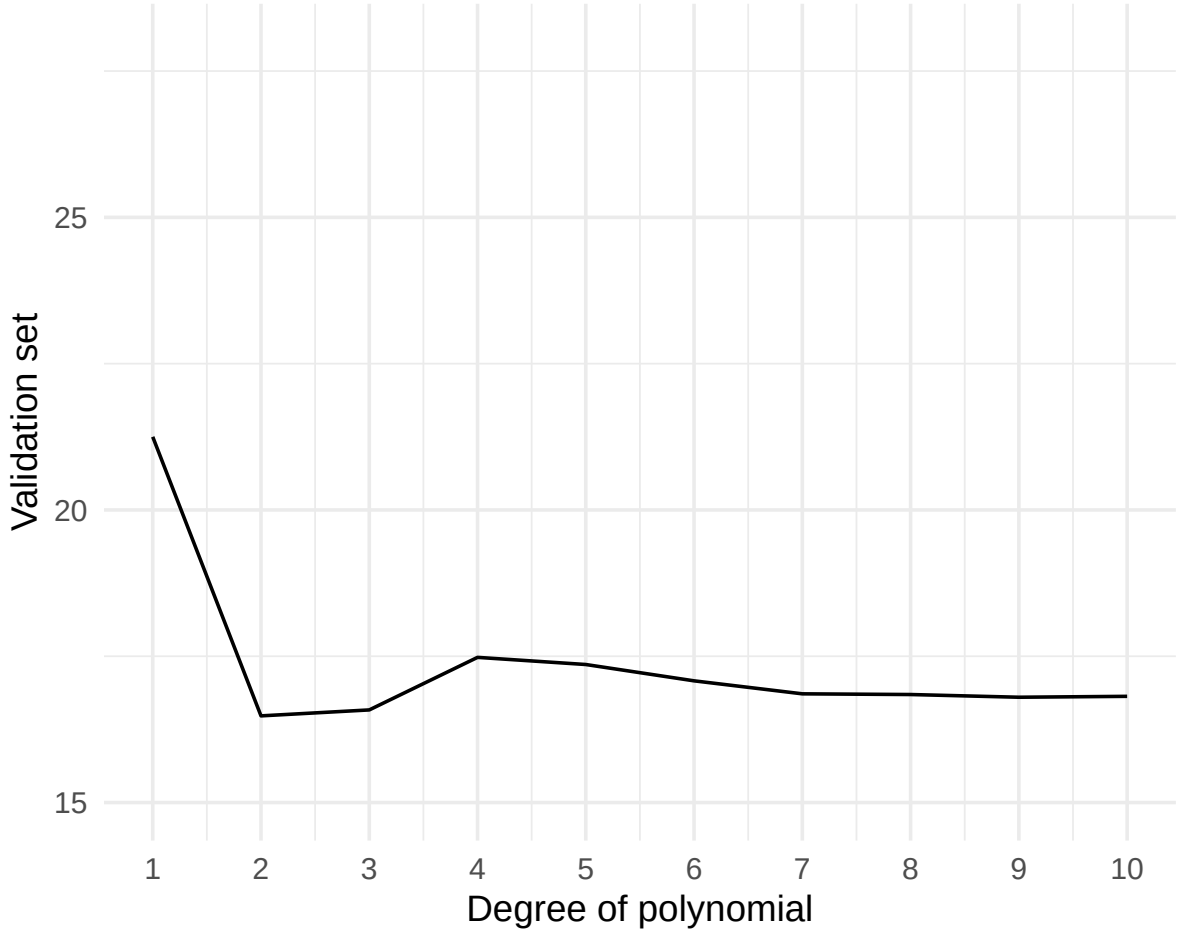
The validation set approach

- Consider the case when you have a data set consisting of n observations.
- To fit a model and to evaluate its predictive performance you randomly divide the data set into two parts ($n/2$ sample size each):
  - a *training set* (to fit the model) and
  - a *validation set* (to make predictions of the response variable for the observations in the validation set)

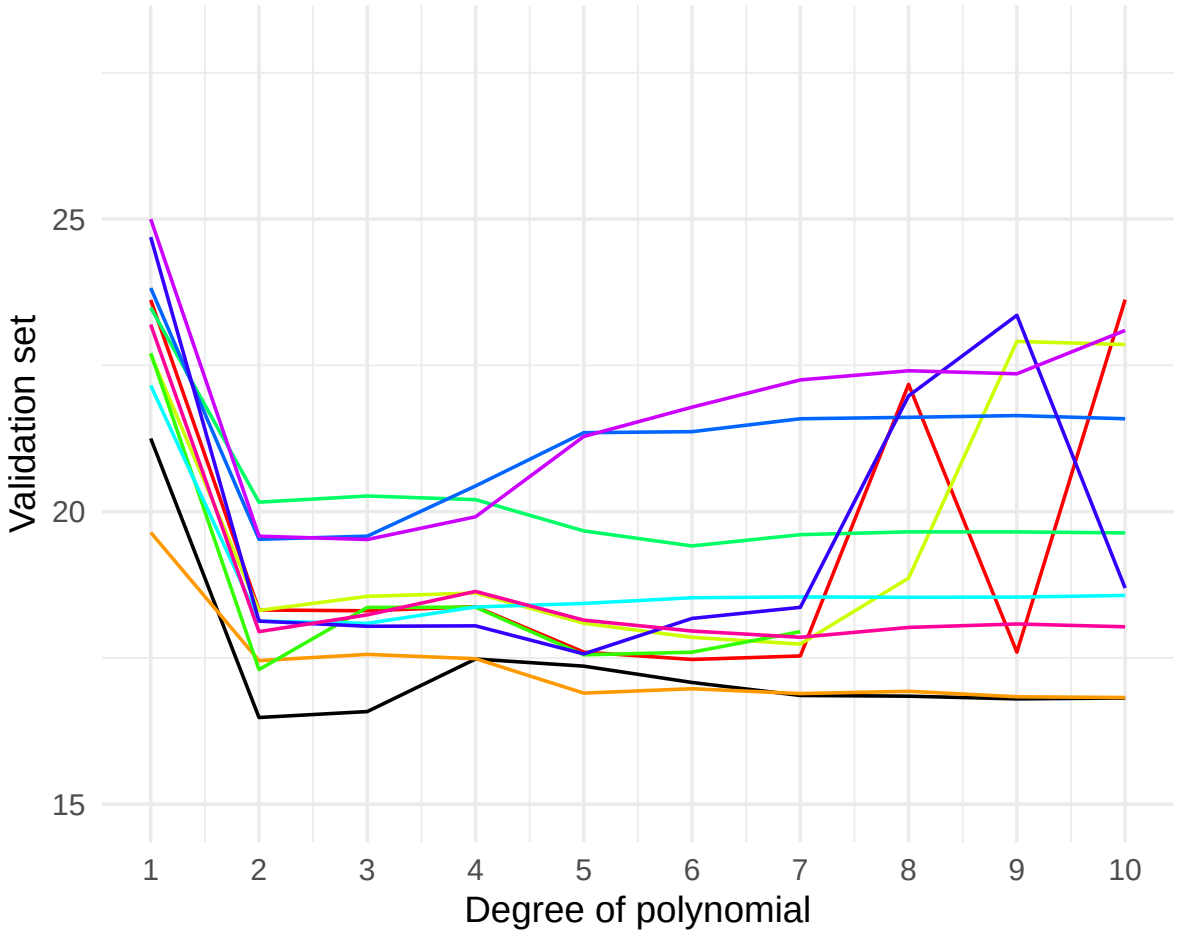
Remember: The focus is on model selection (finding the model that performs "best", that is, with lowest test error).

Example of validation set approach

Auto data set (library `ISLR`): predict `mpg` (miles pr gallon) using polynomial function of `horsepower` (of engine), $n = 392$. What do you see?



But what if we select another split into two parts? Let's do this for many splits:



$\rightarrow$ No consensus which model really gives the lowest validation set MSE.

Drawbacks with the validation set approach

- High variability of validation set error due to dependency on the set of observation included in the training and validation set
- Smaller sample size for model fit, as only half of the observations are in the training set. Therefore, the validation set error may tend to overestimate the error rate on new observations for a model that is fit on the full data set (the more data, the lower the error).

Better ideas?

Leave-one-out cross-validation (LOOCV)

Leave-one-out cross-validation (LOOCV) addresses the limitations of the validation set approach.

Idea:

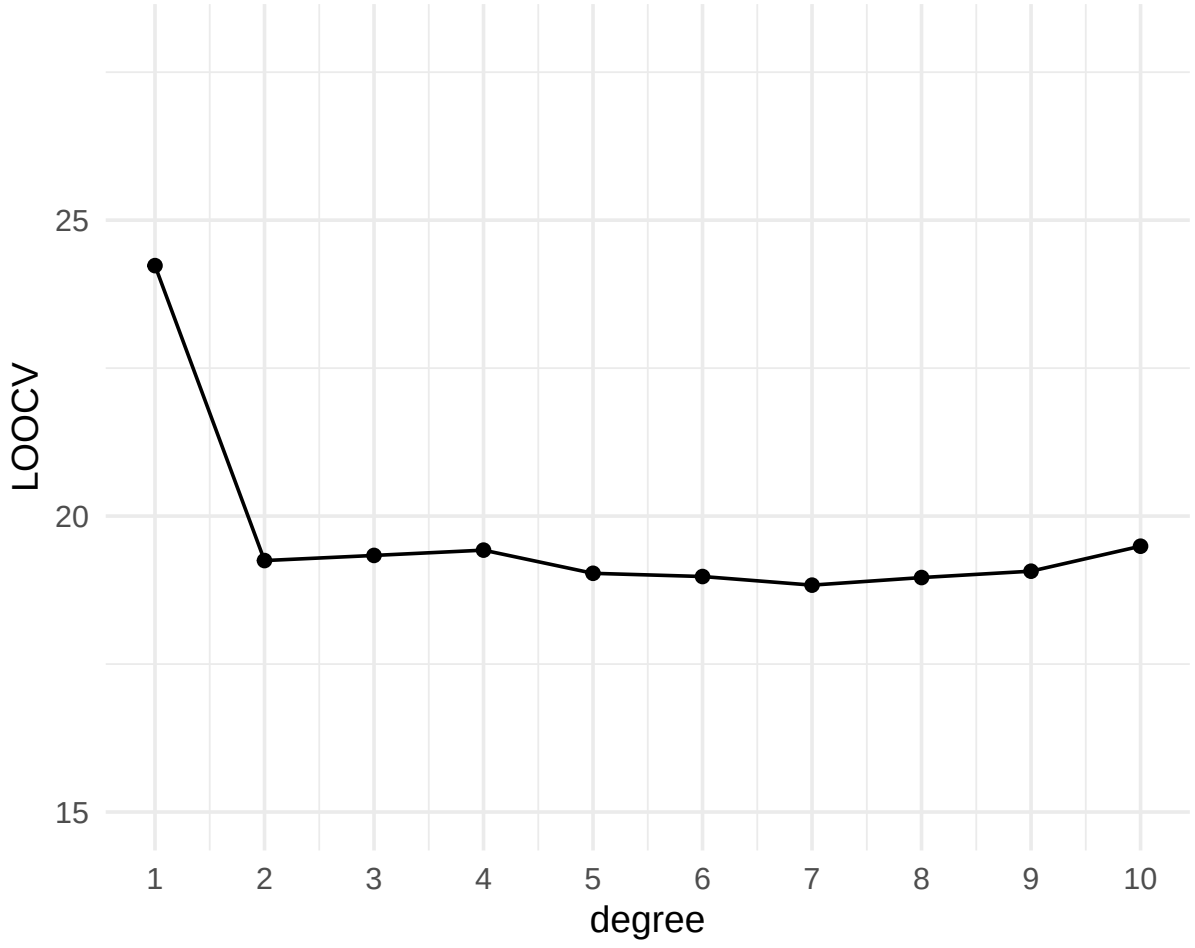
- Only **one observation at a time** is left out and makes up the new observations (test set).
- The remaining $n - 1$ observations make up the training set.
- The procedure of model fitting is repeated n times, such that each of the n observations is left out once. In each step, we calculate the MSE as

$$\text{MSE}_i = (y_i - \hat{y}_i)^2 \ .$$

- The **total prediction error** is the mean across these n models

$$\text{CV}_n = \frac{1}{n} \sum_{i=1}^{n} \text{MSE}_i \ .$$

Regression example: LOOCV



```r
library(ISLR)  #for Auto data set
library(boot)  #for cv.glm
library(ggplot2)  #for plotting
set.seed(123)
n = dim(Auto)[1]
testMSEvec = NULL
start = Sys.time()
for (polydeg in 1:10) {
    glm.fit = glm(mpg ~ poly(horsepower, polydeg), data = Auto)
    glm.cv1 = cv.glm(Auto, glm.fit, K = n)
    testMSEvec = c(testMSEvec, glm.cv1$delta[1])
}
stopp = Sys.time()
yrange = c(15, 28)
plotdf = data.frame(testMSE = testMSEvec, degree = 1:10)
g0 = ggplot(plotdf, aes(x = degree, y = testMSE)) + geom_line() + geom_point() +
    scale_y_continuous(limits = yrange) + scale_x_continuous(breaks = 1:10) +
    labs(y = "LOOCV")
g0 + theme_minimal()
```

Issues with leave-one-out cross-validation

- Pros:
  - No randomness in training/validation splits!
  - Little bias, since nearly the whole data set used for training (compared to half for validation set approach).
- Cons:
  - Expensive to implement – need to fit n different models.
  - High variance since: two training sets only differ by one observation - which makes estimates from each fold highly correlated and this can lead to that their average can have high variance*.

* Recall that

$$\mathrm{Var}\Big(\sum_{i=1}^{n} a_i X_i + b\Big) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \mathrm{Cov}(X_i, X_j)$$

$$= \sum_{i=1}^{n} a_i^2 \mathrm{Var}(X_i) + 2\sum_{i=2}^{n}\sum_{j=1}^{i-1} a_i a_j \mathrm{Cov}(X_i, X_j).$$

LOOCV for multiple linear regression

There is a nice shortcut for LOOCV in the case of linear regression:

$$\mathrm{CV}_n = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_{ii}}\right)^2 \quad ,$$

where h_i is the ith diagonal element (leverage) of the hat matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, and $\hat{y}_i$ is the ith fitted value from the original least squares fit.

$\rightarrow$ Need to fit the model only once!

k-fold cross-validation

To address the drawbacks of LOOCV, we can leave out not just one single observation in each iteration, but $1/k$-th of all data.

Procedure:

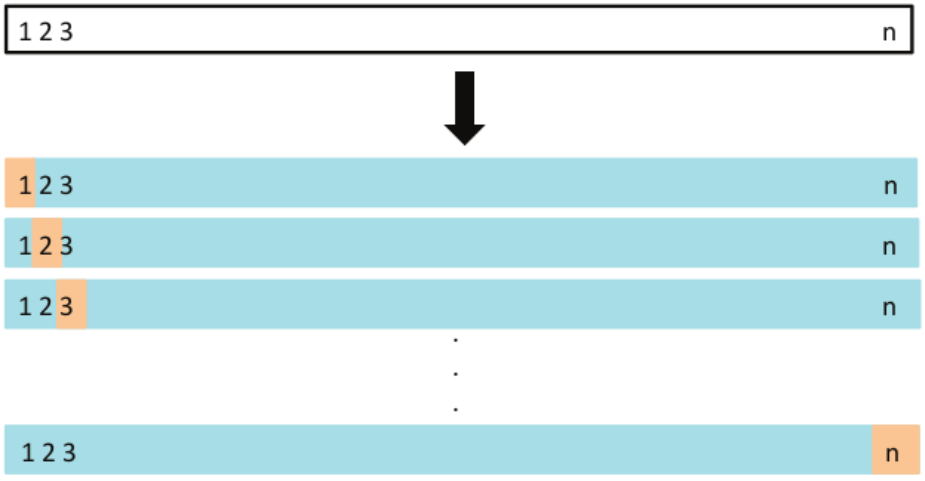
- Split the data into k (more or less) equal parts.
- Use $k-1$ parts to fit and the kth part to validate.
- Do this k times and leave out another part in each round.

The MSE is then estimated in each of the k iterations $(\text{MSE}_1, \ldots, \text{MSE}_k)$, and the the k-fold CV is

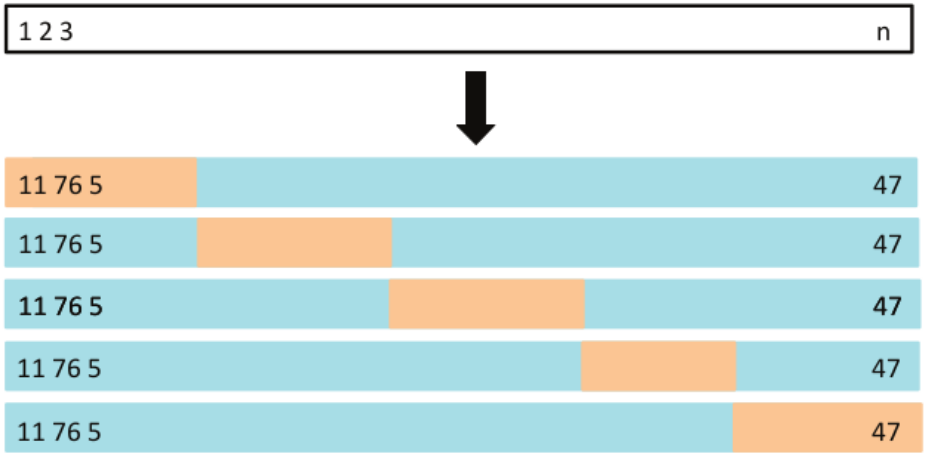
$$\text{CV}_k = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i \ .$$

Comparison of LOOCV and k-fold CV:

LOOCV:



k-fold:



Formally

- Indices of observations - divided into k folds: $C_1, C_2, \ldots, C_k$.
- n_k elements in each fold, if n is a multiple of k then $n_k = n/k$.

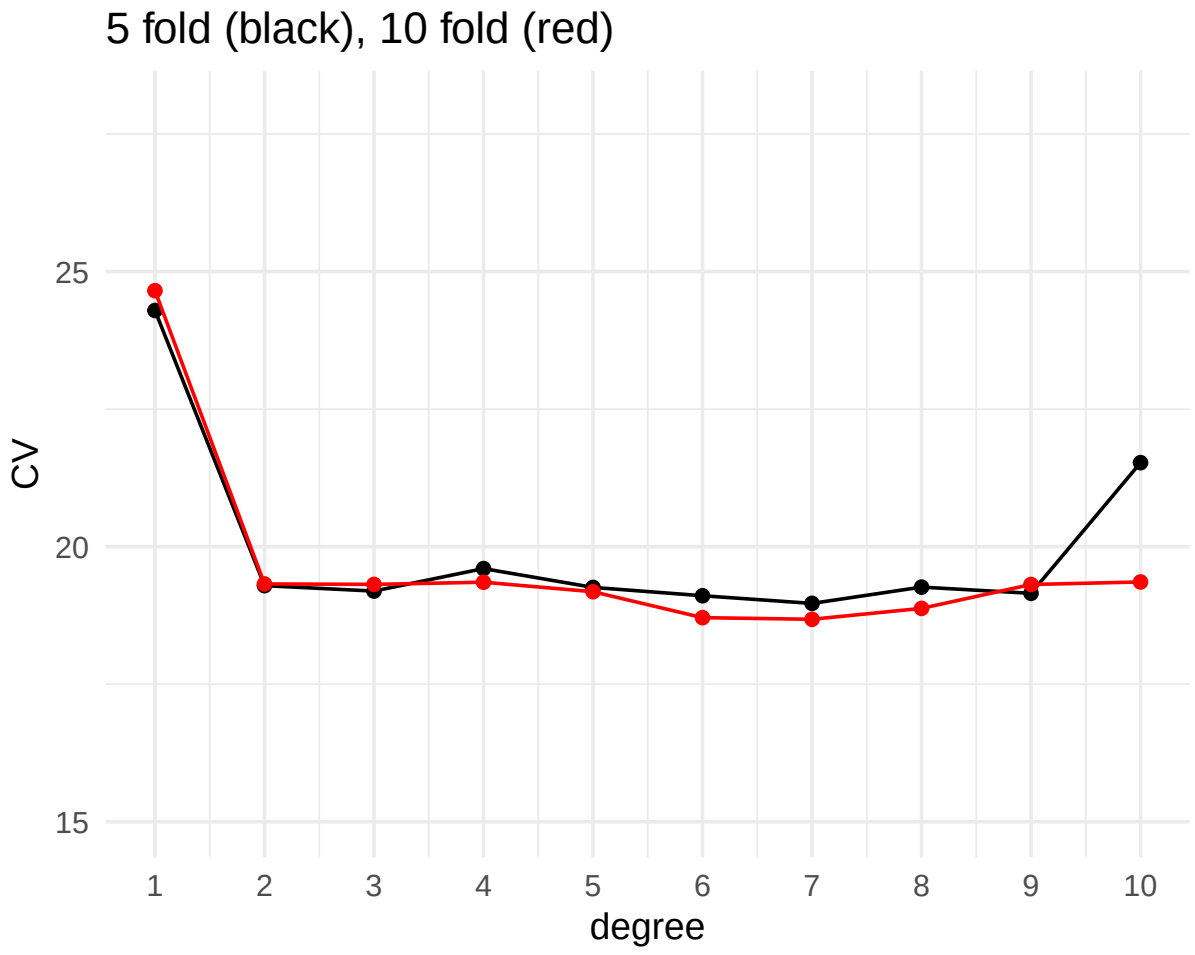
$$\text{MSE}_k = \frac{1}{n_k} \sum_{i \in C_k} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i$ is the fit for observation i obtained from the data with part k removed.

$$\text{CV}_k = \frac{1}{n} \sum_{j=1}^{k} n_j \text{MSE}_j$$

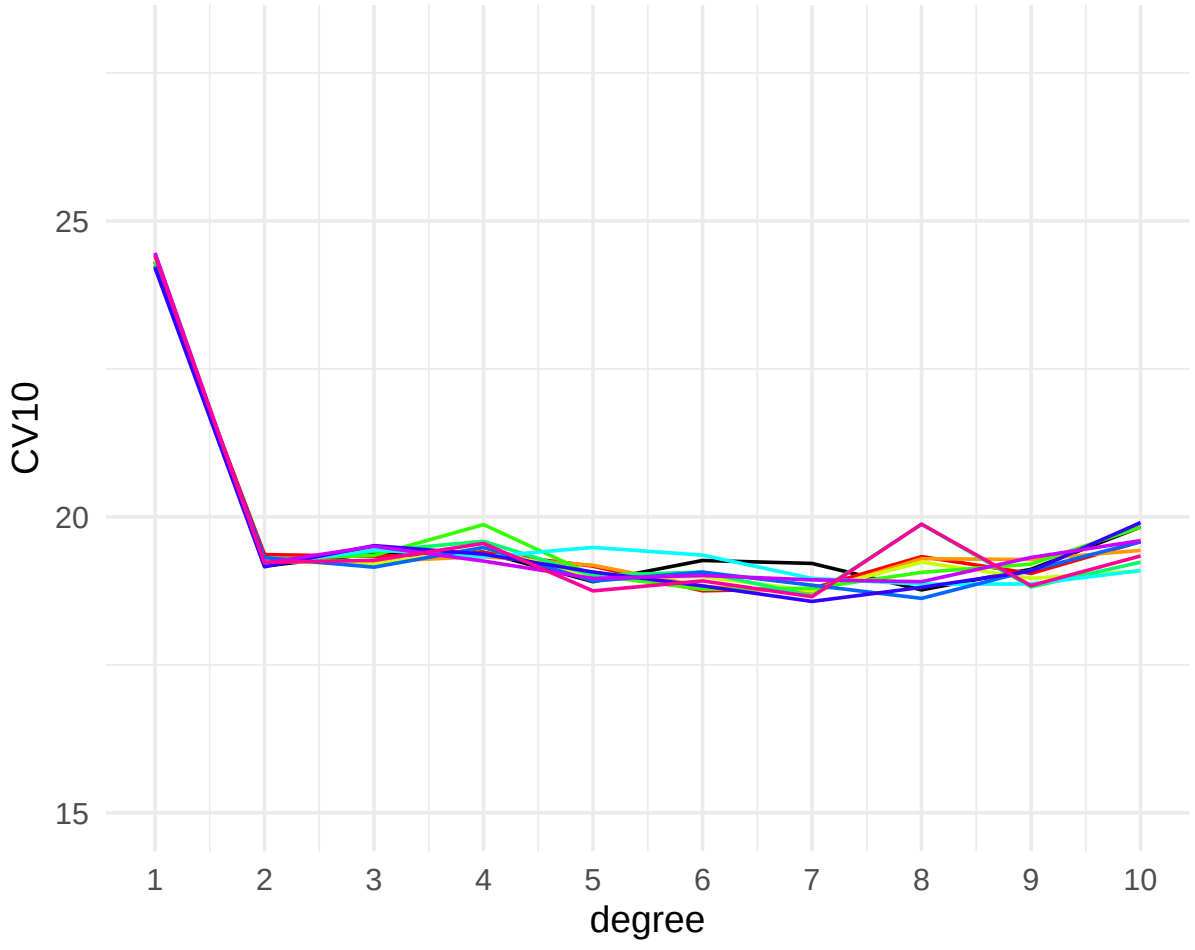
Observe: setting $k = n$ gives LOOCV.

Regression example: 5 and 10-fold cross-validation



```r
library(ISLR)
library(boot)
library(ggplot2)
set.seed(123)
n = dim(Auto)[1]
testMSEvec5 = NULL
testMSEvec10 = NULL
start = Sys.time()
for (polydeg in 1:10) {
    glm.fit = glm(mpg ~ poly(horsepower, polydeg), data = Auto)
    glm.cv5 = cv.glm(Auto, glm.fit, K = 5)
    glm.cv10 = cv.glm(Auto, glm.fit, K = 10)
    testMSEvec5 = c(testMSEvec5, glm.cv5$delta[1])
    testMSEvec10 = c(testMSEvec10, glm.cv10$delta[1])
}
stopp = Sys.time()
yrange = c(15, 28)
plotdf = data.frame(testMSE5 = testMSEvec5, degree = 1:10)
g0 = ggplot(plotdf, aes(x = degree, y = testMSE5)) + geom_line() + geom_point() +
    scale_y_continuous(limits = yrange) + scale_x_continuous(breaks = 1:10) +
    labs(y = "CV") + ggtitle("5 and 10 fold CV")
g0 + geom_line(aes(y = testMSEvec10), colour = "red") + geom_point(aes(y = testMSEvec10),
    colour = "red") + ggtitle("5 fold (black), 10 fold (red)") + theme_minimal()
```

10 reruns (different splits) of the 10-CV method - to see variability:



Issues with k-fold cross-validation

1. As for the validation set, the result may vary according to how the folds are made, but the variation is in general lower than for the validation set approach.
2. Computational issues: less work with $k = 5$ or 10 than LOOCV.
3. The training set is $(k-1)/k$ of the original data set - the estimate of the prediction error is biased upwards.
4. This bias is the smallest when $k = n$ (LOOCV), but we know that LOOCV has high variance.
5. Therefore, often $k = 5$ or $k = 10$ is used as a compromise.

Choosing the best model

- Remember that we *randomly divide* the data into k folds and then perform the CV for all possible model that we want to choose between.
- We have not directly indicated that there is a model parameter (maybe K in KNN or the degree of the polynomial), say θ, involved to calculate CV_j, $j = 1, \ldots, k$

Smallest CV-error:

- Based on the CV-plot we may choose the model with the smallest CV_k as our best model (with corresponding θ).
- We then fit this model using the whole data set (not the test part, that is still kept away), and evaluate the performance on the test set.

One standard error rule:

Denote by $\mathrm{MSE}_j(\theta)$, $j = 1, \ldots, k$ the k parts of the MSE that together give the CV_k.

We can compute the sample standard deviation (standard error) of all $\mathrm{MSE}_j(\theta)$, $j = 1, \ldots, k$

$$\hat{\mathrm{SE}}(\mathrm{CV}_k(\theta)) = \sqrt{\sum_{j=1}^{k} (\mathrm{MSE}_j(\theta) - \overline{\mathrm{MSE}}(\theta))/(k-1)}$$

for each value of the complexity parameter θ.

The *one standard error rule* is to choose the simplest model (*e.g.*, with lowest polynomial degree) within one standard error of the minimal error.

k-fold cross-validation in classification

What do we need to change from our regression set-up?

- For LOOCV $\hat{y}_i$ is the fit for observation i obtained from the data with observations i removed, and $\text{Err}_i = I(y_i \neq \hat{y}_i)$. LOOCV is then

$$\text{CV}_n = \frac{1}{n} \sum_{i=1}^{n} \text{Err}_i$$

- The k-fold CV is defined analogously.

Can we use CV for model assessment?

Assume that we have a method where we not need to perform model selection (maybe this is done using the methods from Module 6, that is AIC or lasso), but want to perform model assessment based on all our data.
Then we can use CV with all data (then the validation part is really the test part) and report on the model performance using the validation parts of the data as above.

Can we use CV both for model selection and model assessment?

Then you can use two layers of CV - also called *nested CV*. See drawing in class.

The right and the wrong way to do cross-validation

ISL book slides, page 17: model assessment.

- We have a two-class problem and would like to use a simple classification method, however,
- we have many possible predictors $p = 5000$ and not so big sample size $n = 50$.

We use this strategy to produce a classifier:

1. We calculate the correlation between the class label and each of the p predictors, and choose the $d = 25$ predictors that have the highest (absolute value) correlation with the class label. (We need to have $d < n$ to fit the logistic regression uniquely.)
2. Then we fit our classifier (here: logistic regression) using only the $d = 25$ predictors.

How can we use cross-validation to produce an estimate of the performance of this classifier?
Q: Can we apply cross-validation only to step 2? Why or why not?

Can we apply cross-validation only to step 2?

A: No, step 1 is part of the training procedure (the class labels are used) and must be part of the CV to give an honest estimate of the performance of the classifier.

- Wrong: Apply cross-validation in step 2.
- Right: Apply cross-validation to steps 1 and 2.

We will see in the Recommended Exercises that doing the wrong thing can give a misclassification error approximately 0 - even if the "true" rate is 50%.

Selection bias in gene extraction on the basis of microarray gene-expression data

Article by Christophe Ambroise and Geoffrey J. McLachlan, PNAS 2002: Direct quotation from the abstract of the article follows.

- In the context of cancer diagnosis and treatment, we consider the problem of constructing an accurate prediction rule on the basis of a relatively small number of tumor tissue samples of known type containing the expression data on very many (possibly thousands) genes.
- Recently, results have been presented in the literature suggesting that it is possible to construct a prediction rule from only a few genes such that it has a negligible prediction error rate.
- However, in these results the test error or the leave-one-out cross-validated error is calculated without allowance for the selection bias.

- There is no allowance because the rule is either tested on tissue samples that were used in the first instance to select the genes being used in the rule or because the cross-validation of the rule is not external to the selection process; that is, gene selection is not performed in training the rule at each stage of the crossvalidation process.
- We describe how in practice the selection bias can be assessed and corrected for by either performing a crossvalidation or applying the bootstrap external to the selection process.
- We recommend using 10-fold rather than leave-one-out cross-validation, and concerning the bootstrap, we suggest using the so-called .632 bootstrap error estimate designed to handle overfitted prediction rules.
- Using two published data sets, we demonstrate that when correction is made for the selection bias, the cross-validated error is no longer zero for a subset of only a few genes.

The Bootstrap

- flexible and powerful statistical tool that can be used to quantify *uncertainty* associated with an estimator or statistical learning method
- we will look at getting an estimate for the standard error of a sample median and of a regression coefficient
- in Module 8 - bootstrapping is the core of the *ensemble method* referred to at *bagging*=bootstrap aggregation,
- in TMA4300 Computation statistics - a lot more theory on the bootstrap.

The inventor: Bradley Efron in 1979 - see interview.

The name? *To pull oneself up by one's bootstraps* from "The Surprising Adventures of Baron Munchausen" by Rudolph Erich Raspe:

The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.

Idea: Use the data itself to get more information about a statistic (an estimator).

Example: the standard deviation of the sample median?

Assume that we observe a random sample $X_1, X_2, \ldots, X_n$ from an unknown probability distribution f. We are interesting in saying something about the population median, and to do that we calculate the sample median $\tilde{X}$. But, how accurate is $\tilde{X}$ as an estimator?

The bootstrap was introduced as a computer-based method to estimate the standard deviation of an estimator, for example our estimator $\tilde{X}$.

But, before we look at the bootstrap method, first we assume that we know F and can sample from F, and use simulations to answer our question.

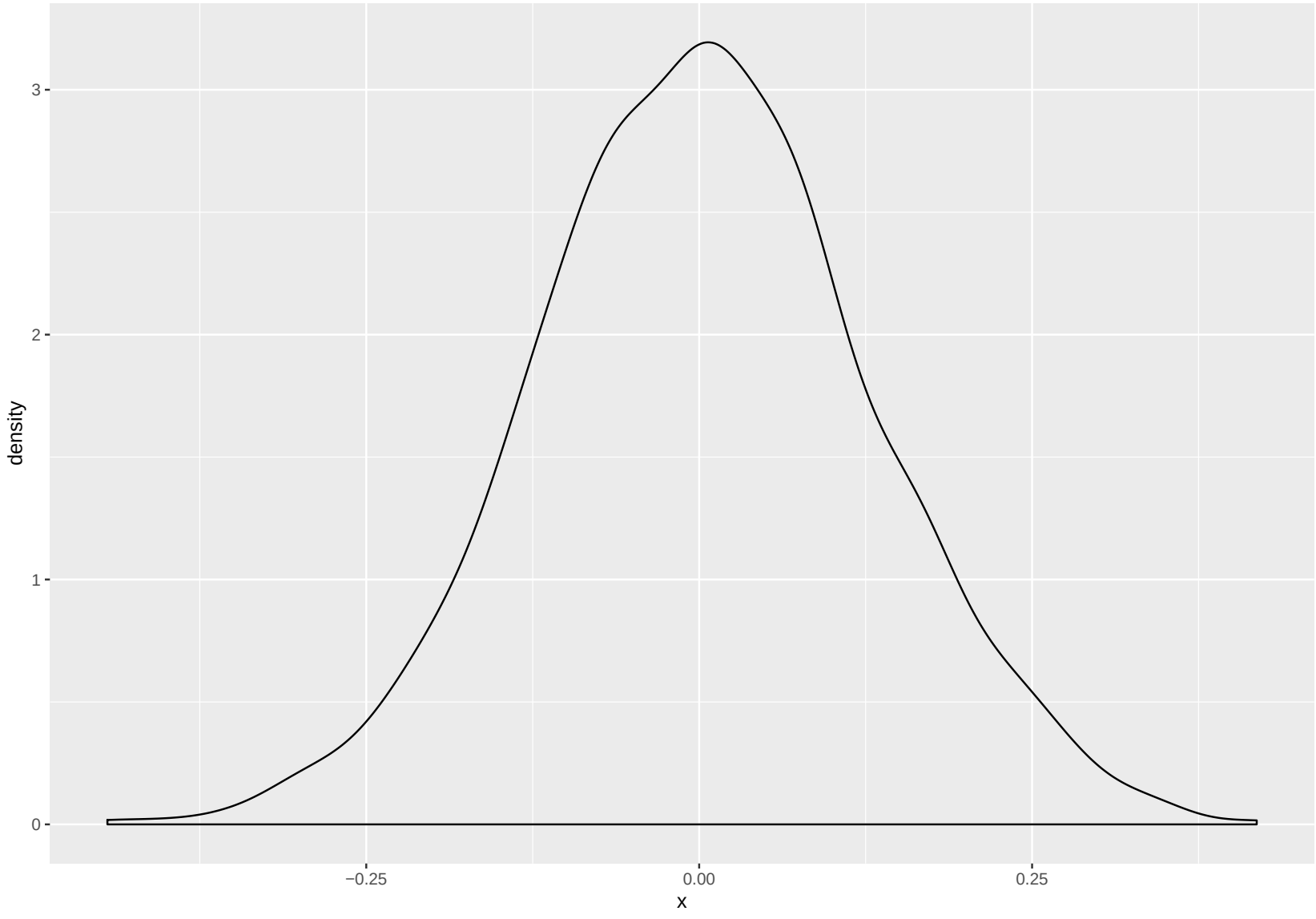
```r
set.seed(123)
n = 101
B = 1000
estimator = rep(NA, B)
for (b in 1:B) {
    xs = rnorm(n)
    estimator[b] = median(xs)
}
sd(estimator)
```

```
## [1] 0.1259035
```

```r
# approximation for large samples (sd of median of standard norm
1.253 * 1/sqrt(n)
```

```
## [1] 0.1246782
```

```
ggplot(data = data.frame(x = estimator), aes(x = x)) + geom_dens
```



Moving from simulation to bootstrapping

The bootstrap method is using the observed data to estimate the *empirical distribution* $\hat{f}$, that is each observed value of x is given probability $1/n$.

A *bootstrap sample* $X_1^*, X_2^*, \ldots, X_n^*$ is a random sample drawn from $\hat{f}$.

A simple way to obtain the bootstrap sample is to *draw with replacement from* $X_1, X_2, \ldots, X_n$.

This means that our bootstrap sample consists of n members of $X_1, X_2, \ldots, X_n$ - some appearing 0 times, some 1, some 2, etc.

```r
set.seed(123)
n = 101
original = rnorm(n)
median(original)
```

```
## [1] 0.05300423
```

```r
boot1 = sample(x = original, size = n, replace = TRUE)
table(table(boot1))
```

```
##
##  1  2  3  4
## 34 22  5  2
```

```r
# how many observations have not been selected in our boot1 samp
n - sum(table(table(boot1)))
```

```
## [1] 38
```

```r
median(boot1)
```

```
## [1] -0.08854676
```

The bootstrap algorithm for estimating standard errors

1. B bootstrap samples: drawn with replacement from the original data
2. Evaluate statistic: on each of the B bootstrap samples to get $\tilde{X}_b^*$ for the bth bootstrap sample.
3. Estimate squared standard error by:

$$\frac{1}{B-1}\sum_{b=1}^{B}(\tilde{X}_b^* - \frac{1}{B}\sum_{b=1}^{B}\tilde{X}_b^*)^2$$

with for-loop in R

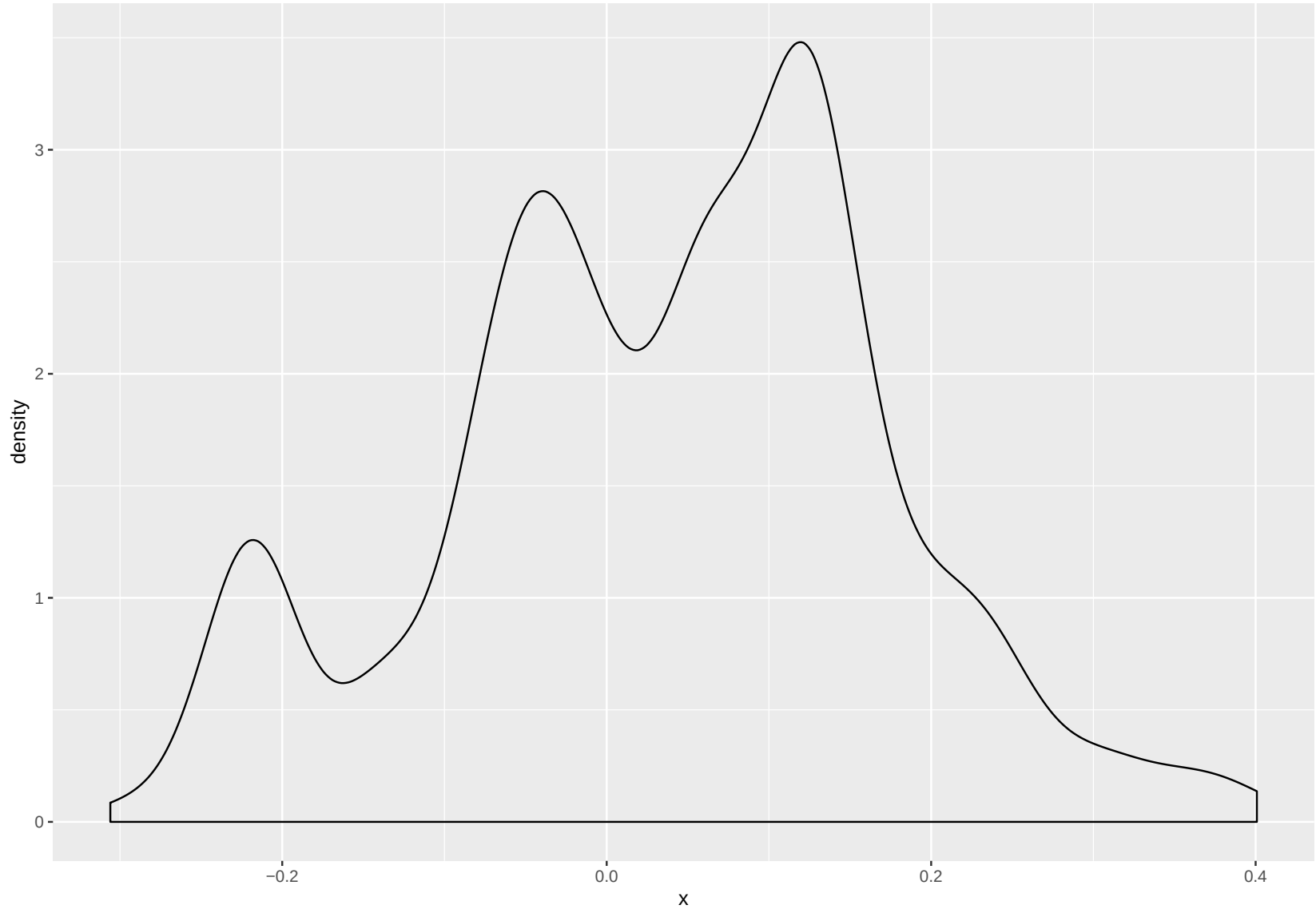
```
set.seed(123)
n = 101
original = rnorm(n)
median(original)
```

```
## [1] 0.05300423
```

```
B = 1000
estimator = rep(NA, B)
for (b in 1:B) {
    thisboot = sample(x = original, size = n, replace = TRUE)
    estimator[b] = median(thisboot)
}
sd(estimator)
```

```
## [1] 0.1365448
```

```
ggplot(data = data.frame(x = estimator), aes(x = x)) + geom_dens
```



using built in `boot` function from library `boot`

```
library(boot)
set.seed(123)
n = 101
original = rnorm(n)
median(original)
```

```
## [1] 0.05300423
```

```
summary(original)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.30917 -0.50232  0.05300  0.08248  0.68864  2.18733
```

```r
boot.median = function(data, index) return(median(data[index]))
B = 1000
boot(original, boot.median, R = B)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = original, statistic = boot.median, R = B)
##
##
## Bootstrap Statistics :
##       original       bias    std. error
## t1* 0.05300423 -0.01577692   0.1290482
```

With or without replacement?

In bootstrapping we sample *with replacement* from our observations.
Q: What if we instead sample *without replacement*?
A: Then we would always get the same sample - given that the order of the sample points is not important for our estimator.
(In permutation testing we sample without replacement to get samples under the null hypothesis - a separate field of research.)

Example: multiple linear regression

We assume, for observation i:

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \varepsilon_i,$$

where $i = 1, 2, ..., n$. The model can be written in matrix form:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

The least squares estimator: $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ has
$\text{Cov}(\boldsymbol{\beta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$.
We will in the recommended exercises look at how to use bootstrapping to estimate the covariance of the estimator. Why is that "needed" if we already know the mathematical formula for the standard deviation? Answer: not needed - but OK to look at an example?
We will not do this here - but our bootstrap samples can also be used to make confidence intervals for the regression coefficients or prediction intervals for new observations. This means that we do not have to rely on assuming that the error terms are normally distributed!

Bagging

Bagging is a special case of *ensemble methods*.
In Module 8 we will look at bagging, which is built on bootstrapping the the fact that it is possible to reduce the variance of a prediction by taking the average of many model fits.

- Assume that we have B different predictors $X_1, X_2, \ldots, X_B$. We have built them on B different bootstrap samples.
- All are predictors for some parameter μ and that all have some unknown variance σ^2.
- We then decide that we want to use all the predictors together - equally weighted - and make $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, which we often use to predict μ.

We can therefore obtain a new model (our average of the individual models) that has a smaller variance than each of the individual model because

$$\text{Var}(\bar{X}) = \frac{\sigma^2}{n}$$

Since this averaged predictor has smaller variance than each of the predictors we would assum that this is a more accurate prediction. However, the interpretation of this bagged prediction might be harder then for the separate predictors.

Models that have poor prediction ability (as we may see can happen with regression and classification trees) might benefit greatly from bagging. More in Module 8.

Summing up

Take home messages

- Use $k = 5$ or 10 fold cross-validation for model selection or assessment.
- Use bootstrapping to estimate the standard deviation of an estimator, and understand how it is performed before module 8 on trees.

Further reading

- Videoes on YouTube by the authors of ISL, Chapter 5, and corresponding slides
- Solutions to exercises in the book, chapter 5