ASTA

The ASTA team

Contents

1 Resampling techniques

Topics:

- Overfitting (the model fits too well to the observed data)
- Generalisation (how well a model performs on a new sample)
- Cross-validation (estimate out-of-sample prediction error)
- Bootstrap (estimate standard errors)

2 Model complexity

2.1 A linear model for tree data

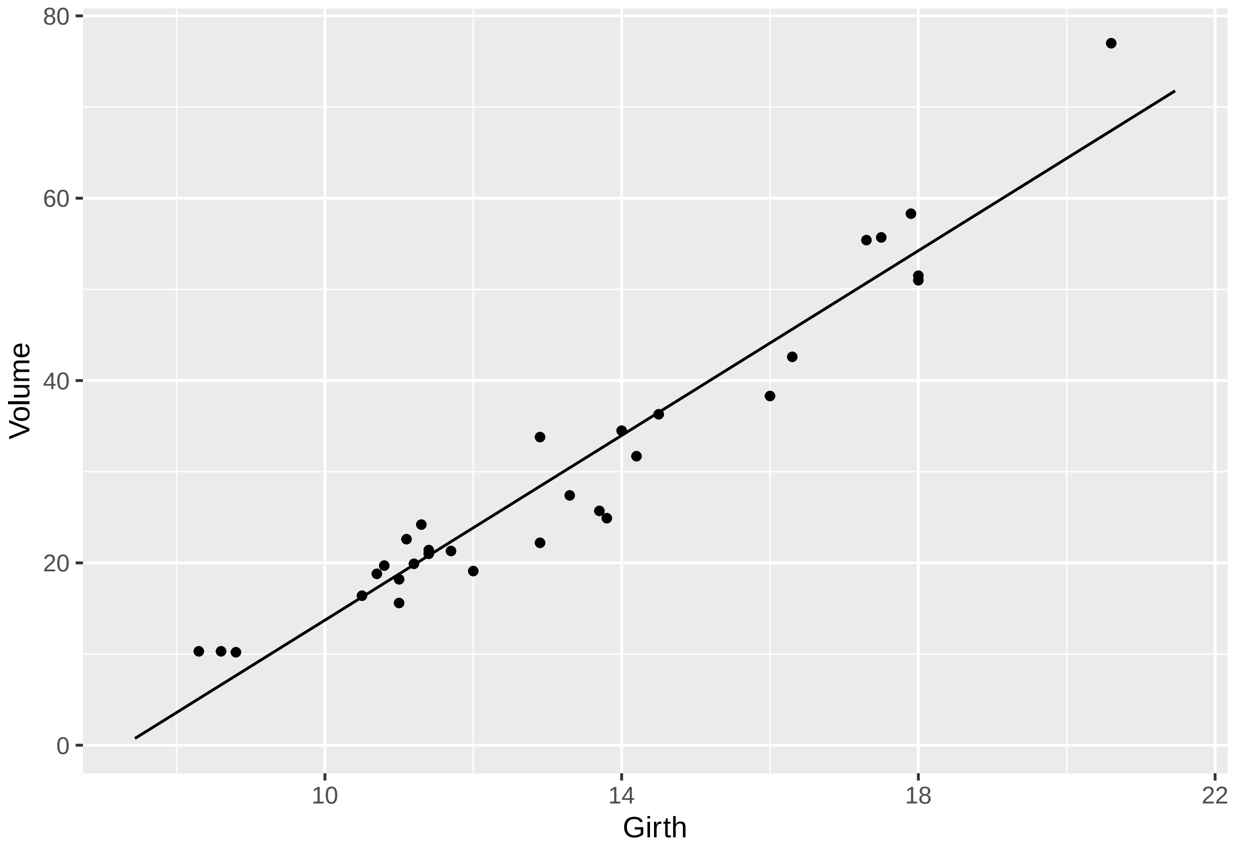
```
trees <- read.delim("https://asta.math.aau.dk/datasets?file=trees.txt")
head(trees)
```

```
##   Girth Height Volume
## 1   8.3     70     10
## 2   8.6     65     10
## 3   8.8     63     10
## 4  10.5     72     16
## 5  10.7     81     19
## 6  10.8     83     20
```

- We consider the dataset `trees` containing the following observations on 31 trees:
  - Response: Volume - timber volume
  - Predictor: Girth - the tree diameter
- We consider a linear model.

$$Y = \alpha + \beta \cdot x + \varepsilon$$

```
m0 <- lm(Volume ~ Girth, data = trees)
plotModel(m0)
```



```r
summary(m0)
```

```
##
## Call:
## lm(formula = Volume ~ Girth, data = trees)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.065 -3.107  0.152  3.495  9.587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -36.943      3.365   -11.0  7.6e-12 ***
## Girth          5.066      0.247    20.5  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.2 on 29 degrees of freedom
## Multiple R-squared:  0.935,  Adjusted R-squared:  0.933
## F-statistic:  419 on 1 and 29 DF,  p-value: <2e-16
```

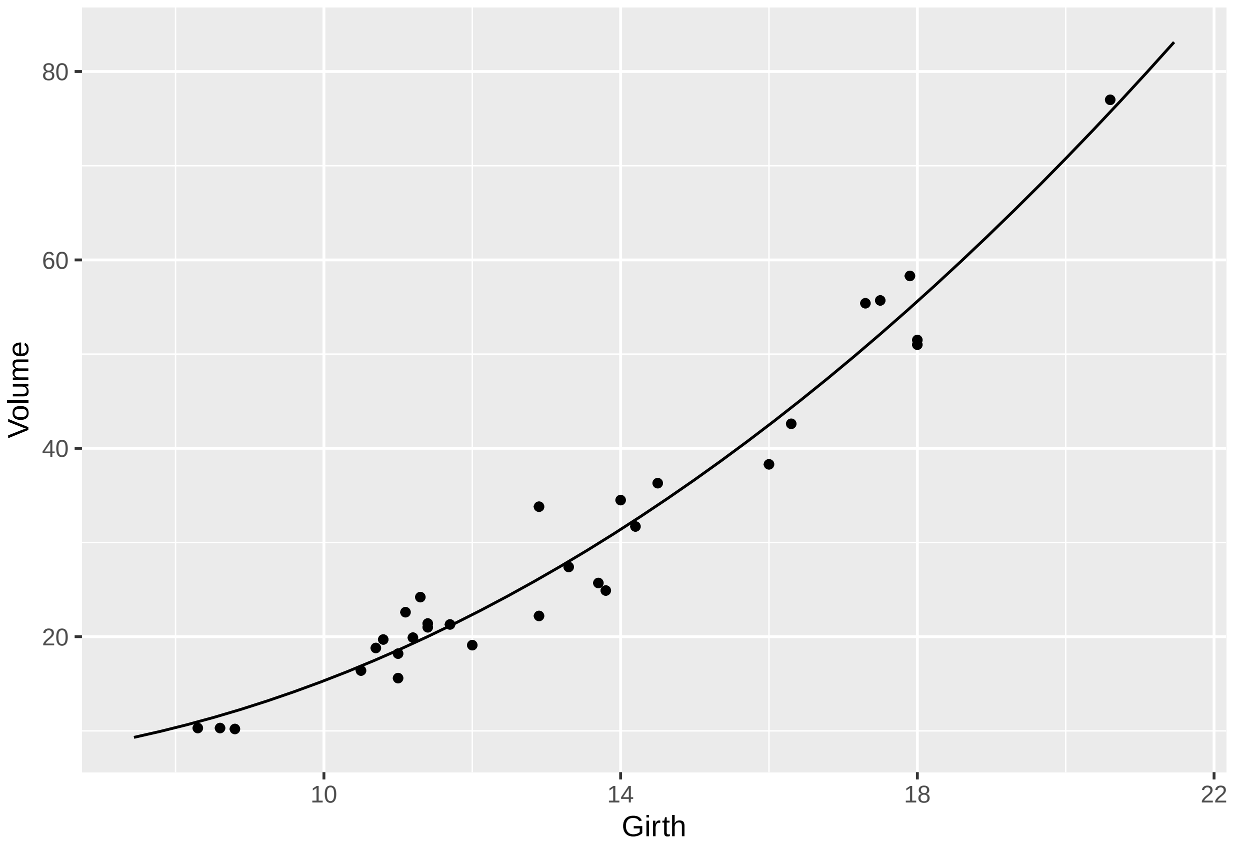
- We obtain the prediction equation
$$\hat{y} = -36.9 + 5.07 \cdot x$$
- The model has $R^2 = 0.935$

2.2 A polynomial model

- We can also try to fit a second degree polynomial to the data

$$Y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon$$

```
m1 <- lm(Volume ~ poly(Girth, 2), data = trees)
plotModel(m1)
```



```
summary(m1)
```

```
##
## Call:
## lm(formula = Volume ~ poly(Girth, 2), data = trees)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.489 -2.429 -0.372  2.076  7.645
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         30.171      0.599   50.37  < 2e-16 ***
## poly(Girth, 2)1     87.073      3.335   26.11  < 2e-16 ***
## poly(Girth, 2)2     14.592      3.335    4.38  0.00015 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.3 on 28 degrees of freedom
## Multiple R-squared:  0.962,  Adjusted R-squared:  0.959
## F-statistic:  350 on 2 and 28 DF,  p-value: <2e-16
```

- Prediction equation

$$\hat{y} = 30.2 + 87.1x + 14.6x^2$$

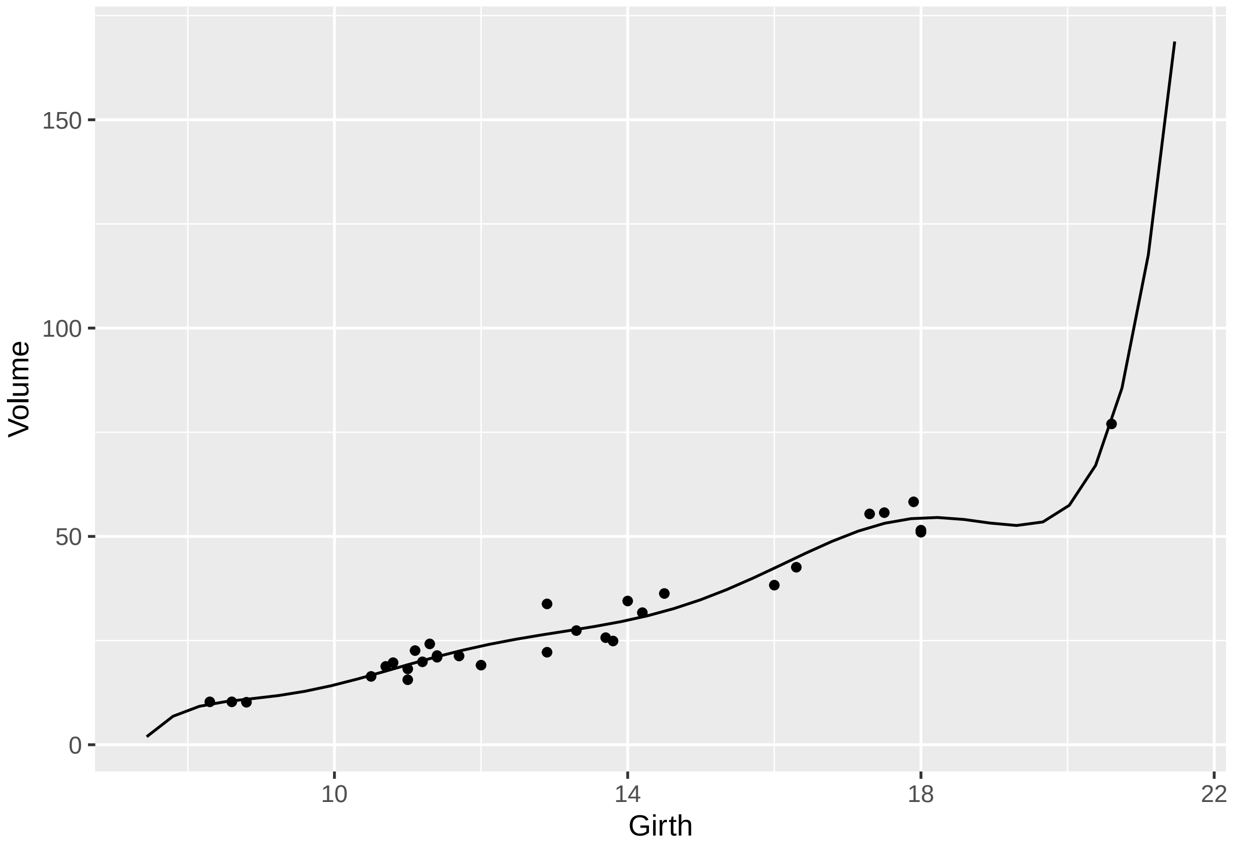
- $R^2 = 0.962$

2.3 Another polynomial model

- Or a polynomial of degree 7

$$Y = \alpha + \beta_1 x + \beta_2 x^2 + \cdots + \beta_7 x^7 + \varepsilon$$

```
m1_bad <- lm(Volume ~ poly(Girth, 7), data = trees)
plotModel(m1_bad)
```

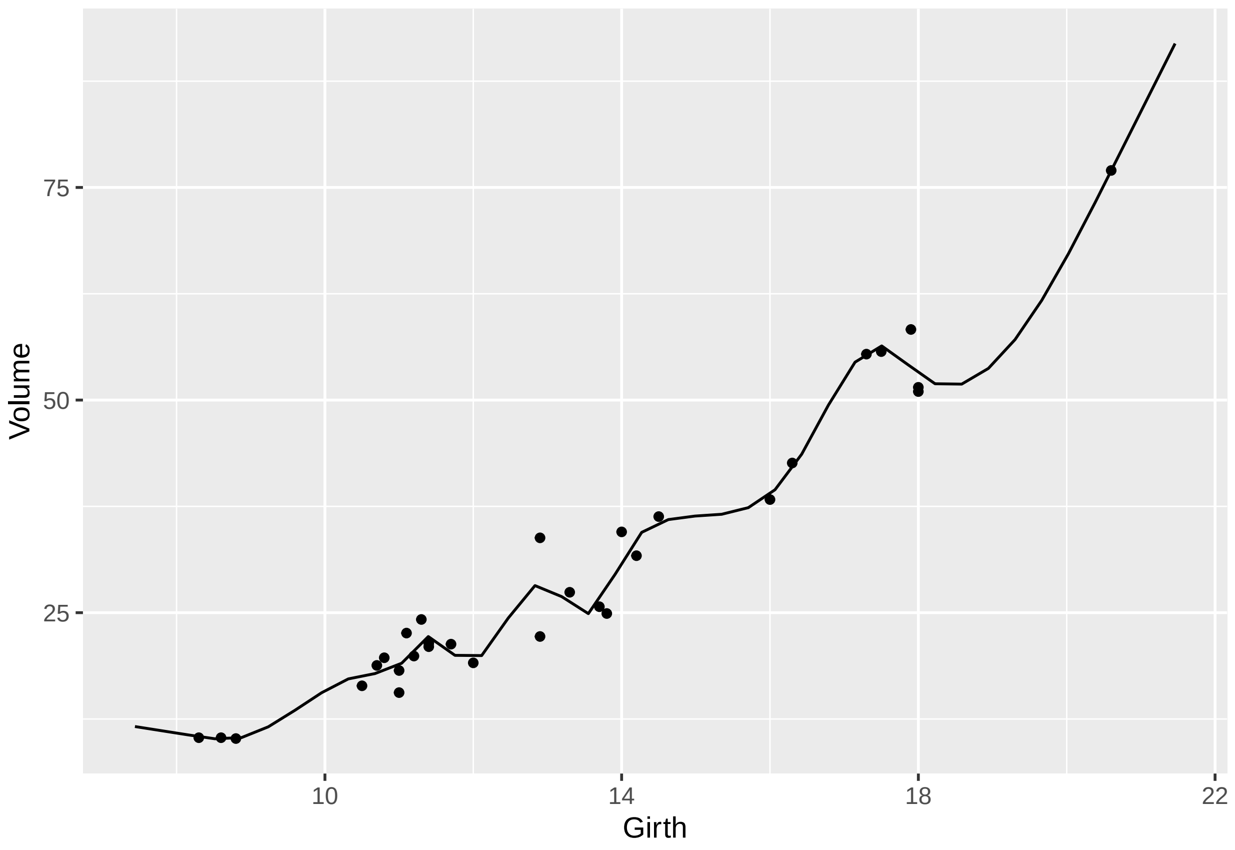


- Polynomials tend to behave wildly at the ends.

2.4 A natural spline

- A natural spline is a piecewise third degree polynomial with smooth overlaps which is linear at the ends. It can also be fitted to the data.

```
library(splines)
m2 <- lm(Volume ~ ns(Girth, 15), data = trees)
plotModel(m2)
```



- The model is very “wiggly” to get near the data points.
- How to compare this model with the others?

3 Measures of fit

3.1 R^2 and correlation

- We can compare the models using R^2

```
summary(m0)$r.squared
```

```
## [1] 0.94
```

```
summary(m1)$r.squared
```

```
## [1] 0.96
```

```
summary(m2)$r.squared
```

```
## [1] 0.98
```

- R^2 is always higher for more complex models
- Note that R^2 is the squared correlation between the observed response values y_i and the values predicted by the prediction equation $\hat{y}_i$

$$R^2 = cor(y_i, \hat{y}_i)^2$$

```r
# or: m0$fitted
cor(trees$Volume,predict(m0, newdata = trees))^2
```

```
## [1] 0.94
```

```r
cor(trees$Volume,predict(m1, newdata = trees))^2
```

```
## [1] 0.96
```

```r
cor(trees$Volume,predict(m2, newdata = trees))^2
```

```
## [1] 0.98
```

3.2 Mean squared error

- We can also compare the models using mean squared errors (MSE)

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

```r
mean((trees$Volume - predict(m0, newdata = trees))^2)
```

```
## [1] 17
```

```r
mean((trees$Volume - predict(m1, newdata = trees))^2)
```

```
## [1] 10
```

```r
mean((trees$Volume - predict(m2, newdata = trees))^2)
```

```
## [1] 4.9
```

- The more complicated model has lowest MSE
- The model is fitted using least squares, i.e. minimising MSE.
- The model is trained to predict the datapoints well.
  - Would it also predict well on new data points?

4 Out-of-sample error

4.1 Reproducibility and random number generation

- The code below generates three random numbers three times.

```r
rnorm(3)
```

```
## [1] -0.35  1.53  1.04
```

```r
rnorm(3)
```

```
## [1]  0.072 -1.505  0.885
```

```r
rnorm(3)
```

```
## [1] -0.62 -1.12 -1.17
```

- We get a new sample in each try

- If we want to be sure we always get the same, we can set a seed.

```
set.seed(1)
rnorm(3)
```

```
## [1] -0.63  0.18 -0.84
```

```
set.seed(1)
rnorm(3)
```

```
## [1] -0.63  0.18 -0.84
```

4.2 Out-of-sample error

- Our dataset contains $n = 31$ observations
- We now split the dataset in two:
  - A **training dataset** consisting of 20 observations
  - A **test dataset** consisting of 11 observations

```
set.seed(1)
train_idx <- sample(x = seq_len(nrow(trees)), size = 20, replace = FALSE)
trees_train <- trees[train_idx, ]
nrow(trees_train)
```

```
## [1] 20
```

```
trees_test <- trees[-train_idx, ]
nrow(trees_test)
```

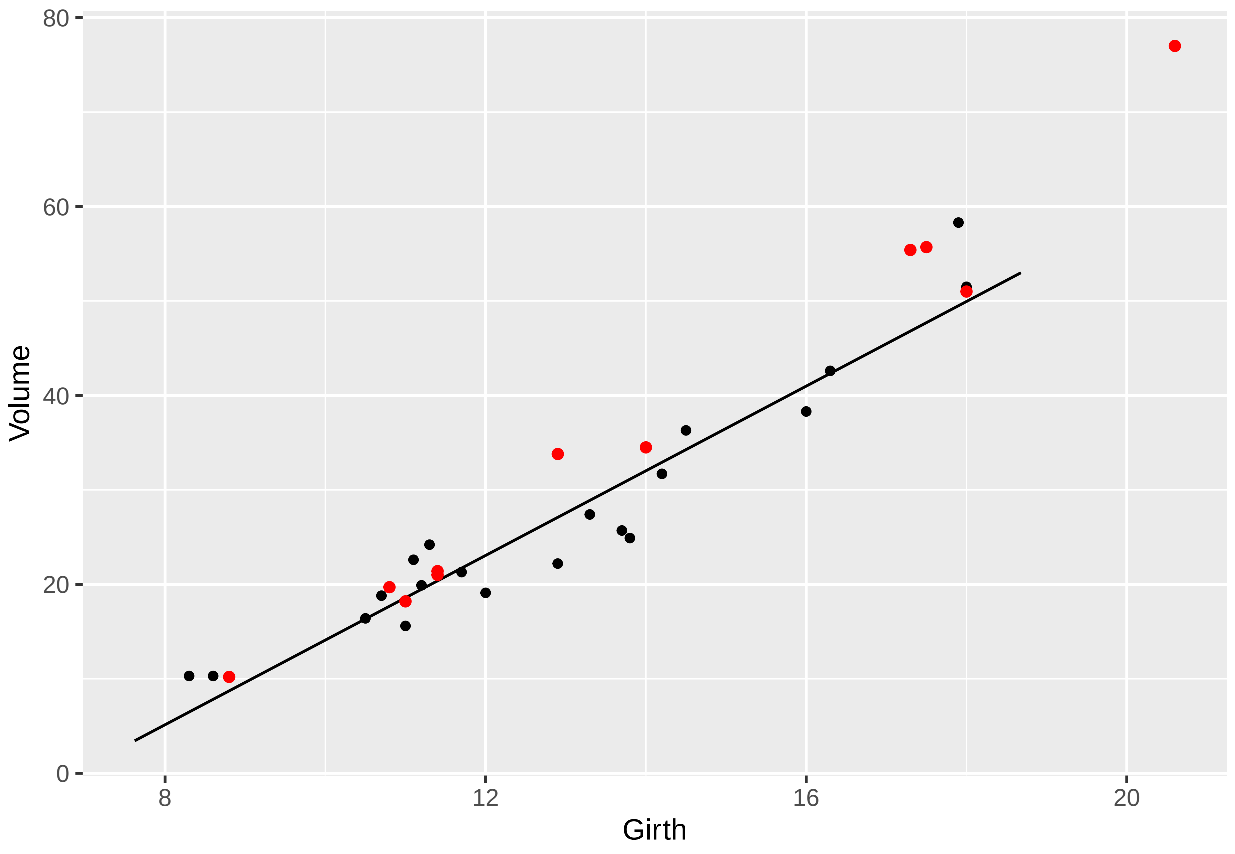
```
## [1] 11
```

4.3 Training the models

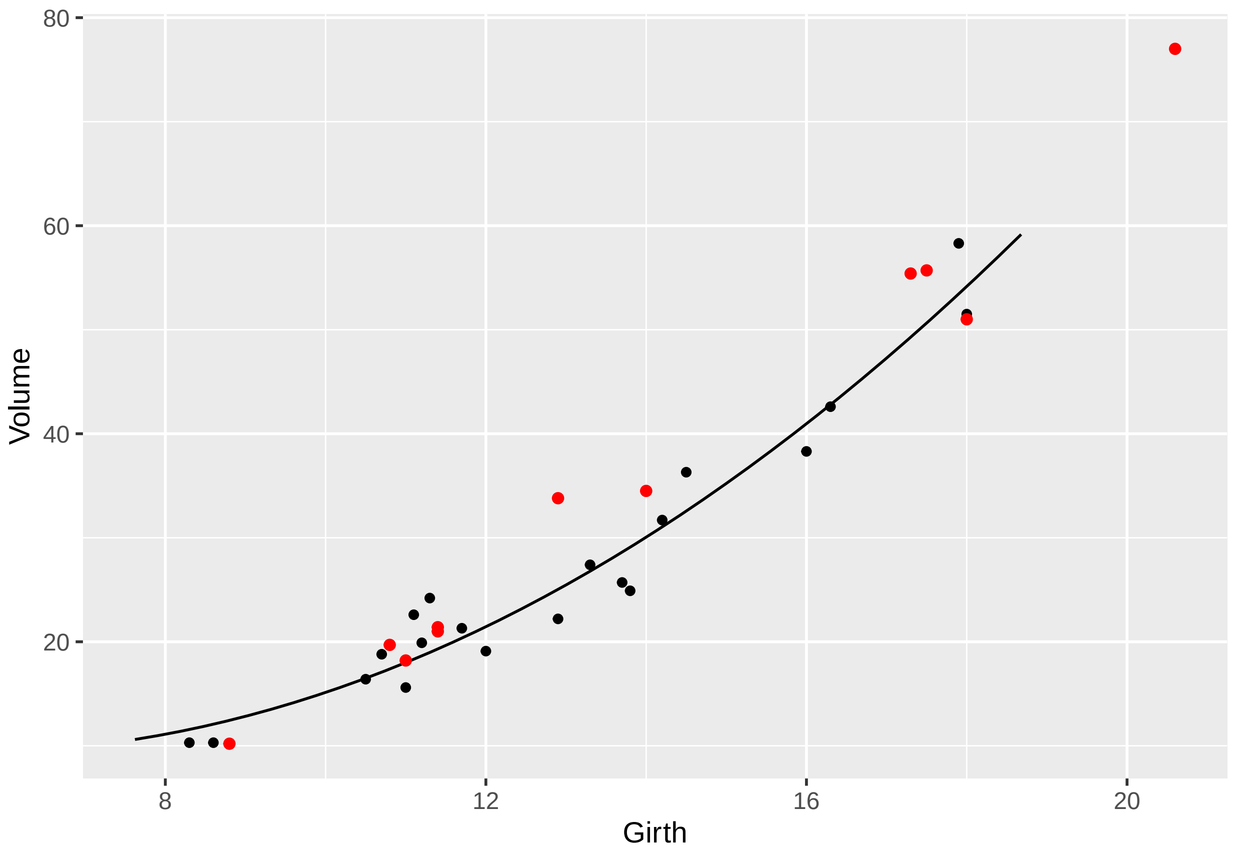
- We use only the training data for fitting the models

```
m0_train <- lm(Volume ~ Girth, data = trees_train)
m1_train <- lm(Volume ~ poly(Girth, 2), data = trees_train)
m2_train <- lm(Volume ~ ns(Girth, 15), data = trees_train)
```

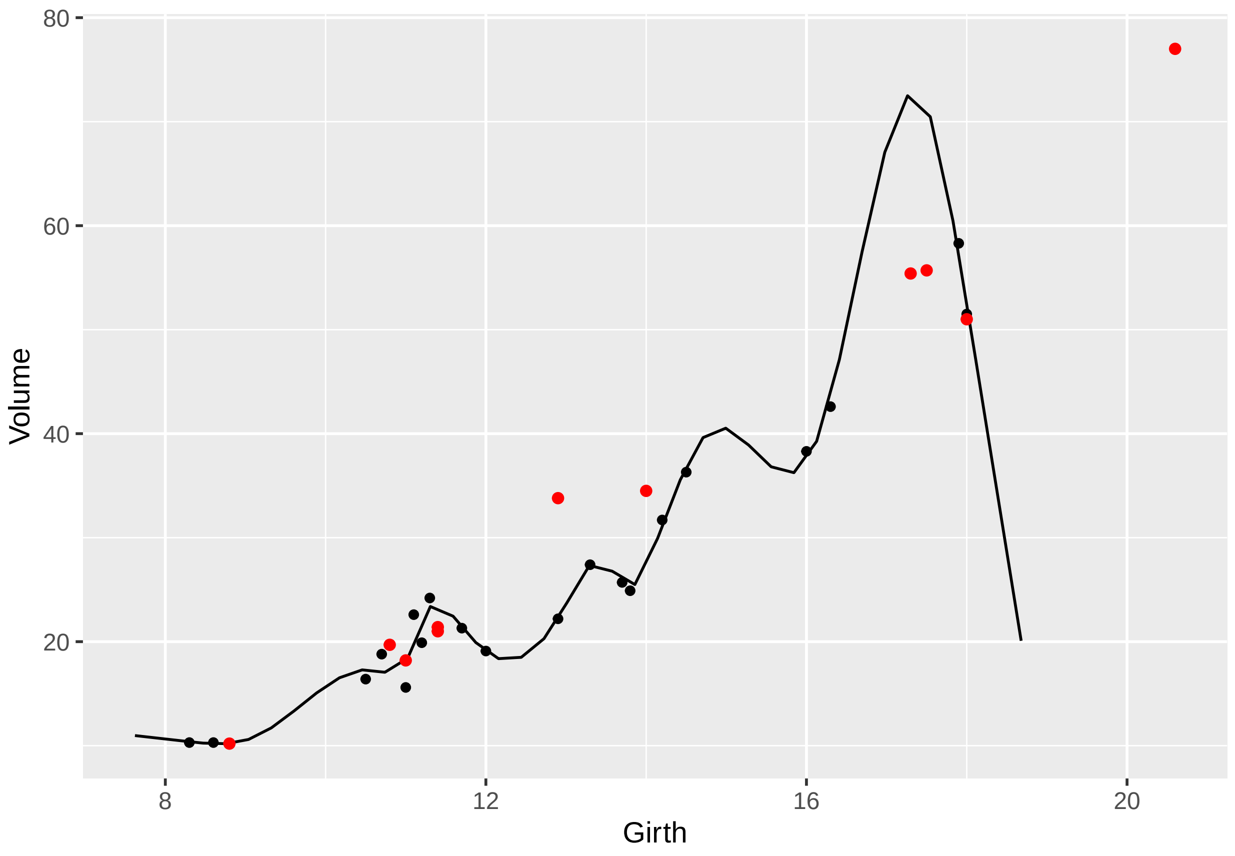
```
plotModel(m0_train) +
  geom_point(aes(Girth, Volume), data = trees_test, color = "red")
```



```r
plotModel(m1_train) +
  geom_point(aes(Girth, Volume), data = trees_test, color = "red")
```



```r
plotModel(m2_train) +
  geom_point(aes(Girth, Volume), data = trees_test, color = "red")
```



4.4 Testing the models

- We now use the test dataset to test the models
  - We predict the response in the test dataset
  - We then compare the predictions to the observed response

```
cor(predict(m0_train, newdata = trees_test), trees_test$Volume)^2
```

```
## [1] 0.98
```

```
cor(predict(m1_train, newdata = trees_test), trees_test$Volume)^2
```

```
## [1] 0.97
```

```
cor(predict(m2_train, newdata = trees_test), trees_test$Volume)^2
```

```
## [1] 0.016
```

- The linear model has the highest correlation between observations and predictions

```
mean((predict(m0_train, newdata = trees_test) - trees_test$Volume)^2)
```

```
## [1] 40
```

```
mean((predict(m1_train, newdata = trees_test) - trees_test$Volume)^2)
```

```
## [1] 17
```

```
mean((predict(m2_train, newdata = trees_test) - trees_test$Volume)^2)
```

```
## [1] 2074
```

- The quadratic polynomial has the smallest MSE

4.5 Summary

- When we test on the same data as we train the model on, we get lower MSE and higher $cor(y_i, \hat{y}_i)$ for the more complex model
- When we test on new data, the more complicated model does not predict well
- **Overfitting:** A complex model tends to fit too well to the training data, but does not fit well to new data.

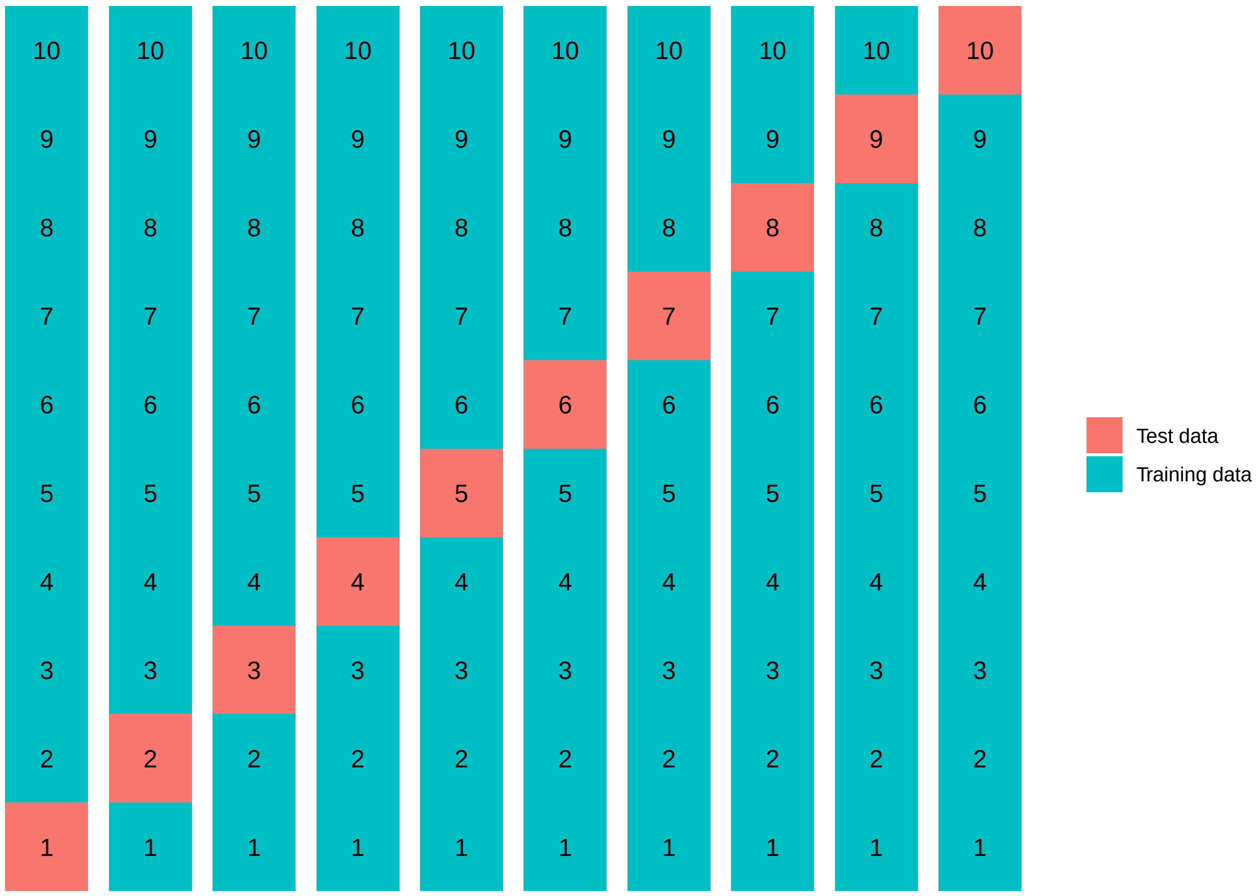
5 Cross-validation

5.1 Testing predictive ability

- Ideally, we should test a models predictive ability on new data that was not used to fit the model
- Typically, only one dataset is available
  - A solution could be to split the dataset in test and training data
  - Waste of data
- Solution: repeat the splitting of data multiple times

5.2 Cross-validation

- Cross-validation provides a clever way of repeating the training and test of a model
- Divide data into k folds
- In each iteration, fit the model on $k-1$ folds, test on the last fold
- E.g. k-fold cross validation for $k = 10$:



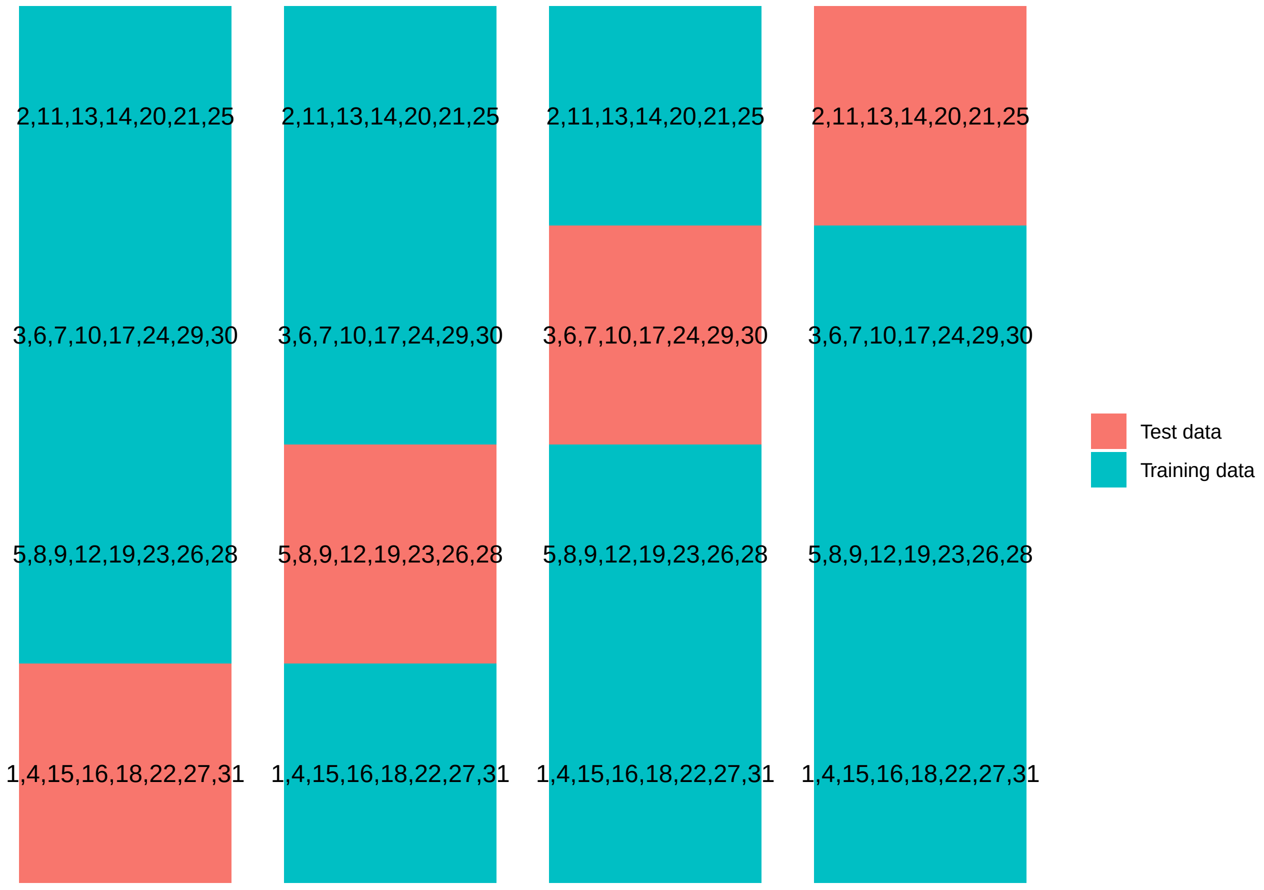
- Benefits:
  - We use most of the data for fitting the model
  - Each observation is used once for testing

5.3 Example

- Number of folds depends on size of dataset (often $k = 5$ or $k = 10$)
- We have 31 observations
- 4-fold cross validation seems suitable
- Divide data into 4 fold

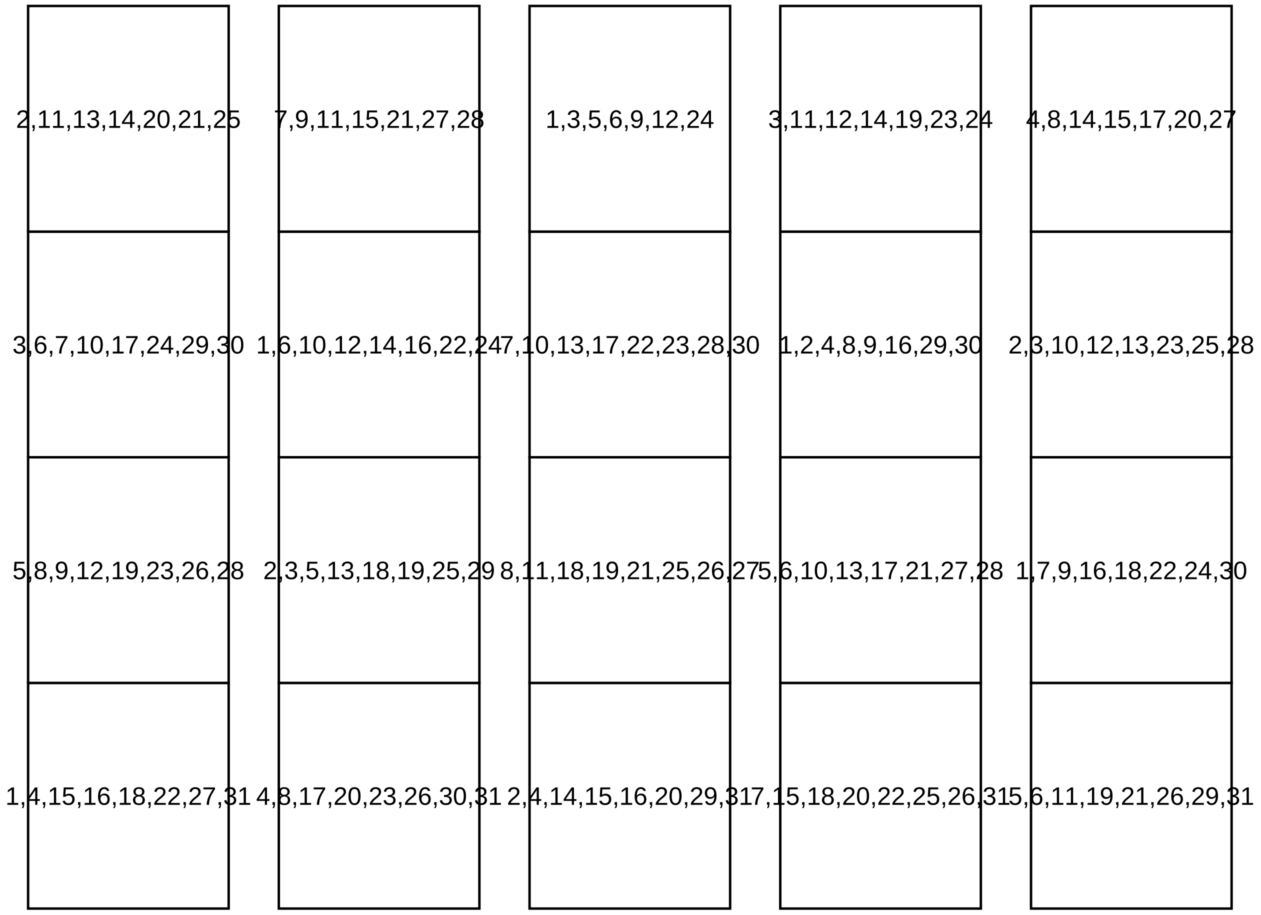
2,11,13,14,20,21,25
3,6,7,10,17,24,29,30
5,8,9,12,19,23,26,28
1,4,15,16,18,22,27,31

- “Rotate” which folds are training data and which one is test data:



5.4 Repeated CV

- Cross-validation may be repeated several times



5.5 Cross-validation in R

- The caret package can be use for cross-validatin in R

```
library(caret)
```

https://cran.r-project.org/package=caret

https://topepo.github.io/caret/

- We first set up the cross-validation

```
train_control <- trainControl(method = "repeatedcv",
                              # k-fold CV
                              number = 4,
                              # repeated five times
                              repeats = 5)
```

- Then we carry out the cross-validation

```
set.seed(1)
m0_cv <- train(Volume ~ Girth, data = trees, trControl = train_control, method = "lm")
m1_cv <- train(Volume ~ poly(Girth, 2), data = trees, trControl = train_control, method = "lm")
m2_cv <- train(Volume ~ ns(Girth, 15), data = trees, trControl = train_control, method = "lm")
```

5.6 Result of cross-validation

```
m0_cv
```

```
## Linear Regression
##
## 31 samples
##  1 predictor
##
## No pre-processing
## Resampling: Cross-Validated (4 fold, repeated 5 times)
## Summary of sample sizes: 24, 23, 23, 23, 23, 23, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   4.5   0.95      3.7
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

- **RMSE** is root mean squared error

5.7 More on RMSE

- Here is the resulting RMSE for all folds and all repetitions:

```
m0_cv$resample
```

```
##    RMSE Rsquared MAE   Resample
## 1   4.7     0.91 4.2 Fold1.Rep1
## 2   4.2     0.91 3.6 Fold2.Rep1
## 3   5.3     0.99 3.3 Fold3.Rep1
## 4   4.7     0.96 4.2 Fold4.Rep1
## 5   3.8     0.93 3.2 Fold1.Rep2
## 6   5.6     0.93 4.6 Fold2.Rep2
## 7   2.7     0.97 2.0 Fold3.Rep2
## 8   5.0     0.95 4.7 Fold4.Rep2
## 9   3.5     0.97 2.7 Fold1.Rep3
## 10  4.7     0.93 4.2 Fold2.Rep3
## 11  4.6     0.96 3.6 Fold3.Rep3
## 12  5.2     0.96 4.2 Fold4.Rep3
## 13  4.4     0.93 3.6 Fold1.Rep4
## 14  5.3     0.95 4.2 Fold2.Rep4
## 15  3.5     0.96 2.7 Fold3.Rep4
## 16  4.1     0.95 3.4 Fold4.Rep4
## 17  3.7     0.98 3.0 Fold1.Rep5
## 18  4.4     0.92 3.7 Fold2.Rep5
## 19  6.2     0.98 4.6 Fold3.Rep5
## 20  4.8     0.92 4.2 Fold4.Rep5
```

- The average of these RMSE is the total RMSE

```
mean(m0_cv$resample$RMSE)
```

```
## [1] 4.5
```

- This can also be obtained directly via the code

```
m0_cv$results$RMSE
```

```
## [1] 4.5
```

5.8 Model comparison

- We obtain the model **RMSE** for the three models

```
m0_cv$results$RMSE
```

```
## [1] 4.5
```

```
m1_cv$results$RMSE
```

```
## [1] 3.5
```

```
m2_cv$results$RMSE
```

```
## [1] 86
```

- The quadratic model has the lowest **RMSE** and hence the best predictive power

6 Non-parametric bootstrap

6.1 Sampling variability

- When we estimate a parameter from a sample, there is some uncertainty due to the fact that the sample is random
- A new sample would result in new estimates
- The standard error is the standard deviation of the estimate when we repeat the sampling many times
  - Measures the uncertainty of the estimate
- However, we only have one sample available

6.2 Bootstrap principle

- Idea: Create new samples by resampling n observations from original data with replacement (the same observation may be sampled several times)
- Mimic new samples
- **Example:** Data indices:

```
index<-c(1,2,3,4,5)
```

- Bootstrap sample indices:

```
set.seed(1)
boot_index<-sample(index, replace = TRUE)
boot_index
```

```
## [1] 1 4 1 2 5
```

- Observation 1 appears 2 times in Bootstrap sample

6.3 Bootstrap data example

- We want to fit a linear model on the tree data. Coefficients of the linear model can be extracted by

```
m0 <- lm(Volume ~ Girth, data = trees)
coef(m0)
```

```
## (Intercept)       Girth
##       -36.9         5.1
```

- We want to estimate their standard errors using bootstrap.
- To prepare for the bootstrap, we define a function that takes as input a vector of indices of the bootstrap observations and does linear regression and extracts coefficients:

```
model_coef <- function(index){
  coef(lm(Volume ~ Girth, data = trees, subset = index))
}
model_coef(1:nrow(trees))
```

```
## (Intercept)       Girth
##       -36.9         5.1
```

```
set.seed(1)
model_coef(sample(1:nrow(trees), replace = TRUE))
```

```
## (Intercept)       Girth
##       -29.8         4.5
```

```
model_coef(sample(1:nrow(trees), replace = TRUE))
```

```
## (Intercept)       Girth
##       -34.4         4.8
```

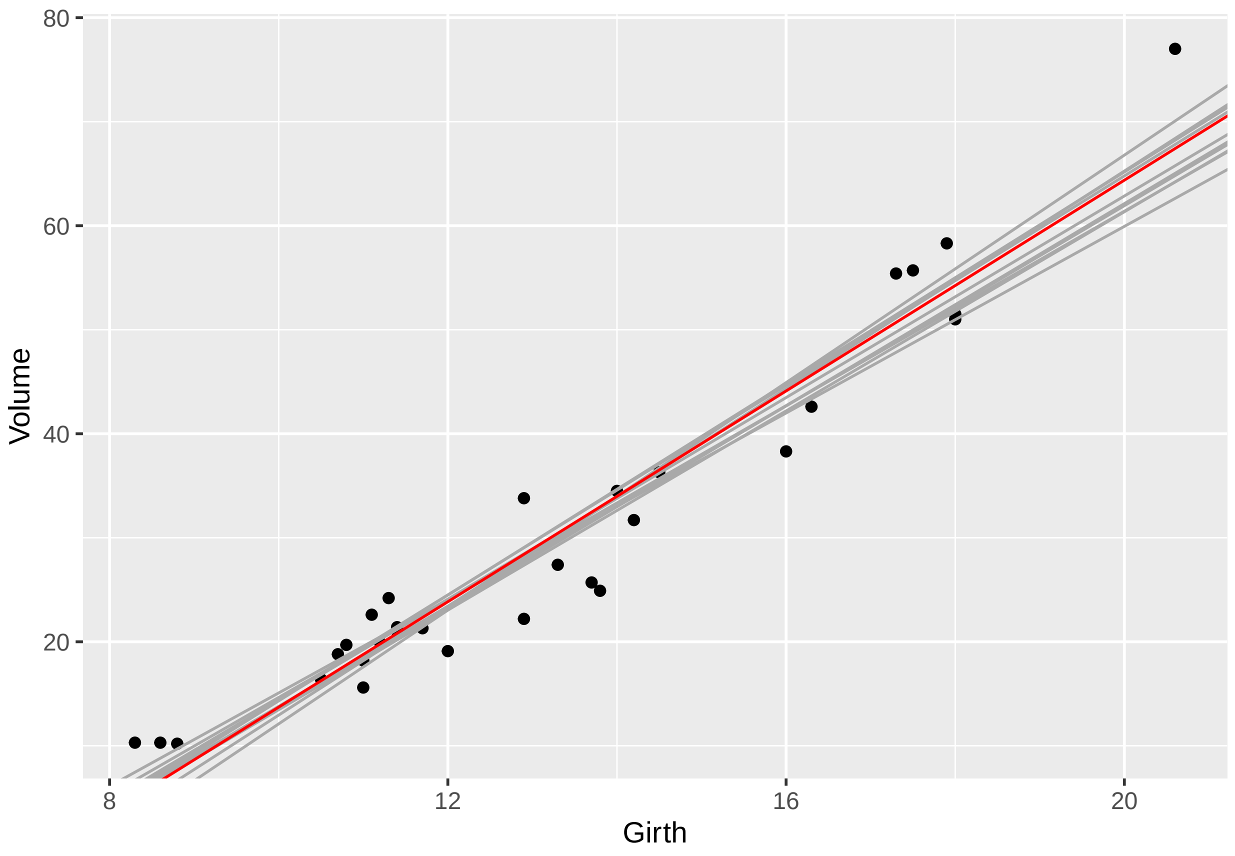
6.4 Bootstrap data example - continued

- We now create 1000 bootstrap samples and estimate the linear regression coefficients for each
- We view the first ten results

```
set.seed(1)
bootstrap_coefs <- replicate(1000, {
  model_coef(sample(1:nrow(trees), replace = TRUE))
})
bootstrap_coefs[, 1:10]
```

```
##              [,1]  [,2]  [,3]  [,4] [,5]  [,6]  [,7]  [,8]  [,9] [,10]
## (Intercept) -29.8 -34.4 -34.3 -42.6  -36 -35.0 -36.5 -39.4 -34.0 -32.1
## Girth         4.5   4.8   4.8   5.5    5   4.9   5.1   5.2   4.8   4.7
```

- Below we plot the regression lines for the original data (red) and for the first ten bootstrap samples (black)



6.5 Bootstrap estimates for the standard error

- We estimate the standard error by taking the standard deviation of the 1000 parameter estimates

```
apply(bootstrap_coefs, 1, sd) # applies the function sd to each row in the matrix bootstrap_coefs
```

```
## (Intercept)       Girth
##        3.98        0.32
```

- This can be compared to the standard errors found by lm() using theoretical formulas

```
summary(m0)
```

```
##
## Call:
## lm(formula = Volume ~ Girth, data = trees)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.065 -3.107  0.152  3.495  9.587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -36.943      3.365   -11.0  7.6e-12 ***
## Girth          5.066      0.247    20.5  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 4.2 on 29 degrees of freedom
## Multiple R-squared:  0.935,  Adjusted R-squared:  0.933
## F-statistic:  419 on 1 and 29 DF,  p-value: <2e-16
```

6.6 The boot package

- Bootstrapping can be done automatically using the `boot` package in R: https://cran.r-project.org/package=boot

```r
library(boot)
```

- We now need a function of both the dataset and an index vector that returns the linear regression coefficients.

```r
model_coef_boot <- function(data, index){
  coef(lm(Volume ~ Girth, data = data, subset = index))
}
```

- Then the bootstrap is carried out as follows

```r
set.seed(1)
b <- boot(trees, model_coef_boot, R = 1000)
b
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = trees, statistic = model_coef_boot, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    -36.9   0.372        4.05
## t2*      5.1  -0.038        0.33
```

```r
coef(summary(m0))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -36.9       3.37     -11  7.6e-12
## Girth            5.1       0.25      20  8.6e-19
```

7 Bootstrap by resampling residuals

- Idea:
  - First fit regression line
  - Compute residuals $\hat{\varepsilon}_i = y_i - \hat{y}_i$
  - Create new dataset by replacing y_i by $\hat{y}_i + \hat{\varepsilon}_{i,new}$, where $\hat{\varepsilon}_{i,new}$ is randomly sampled from the residuals $\hat{\varepsilon}_j$
- Can be used if residuals are not normally distributed
- First fit the model

```r
m0 <- lm(Volume ~ Girth, data = trees)
```

- Contruct 1000 new samples with resampled residuals.

```r
set.seed(1)
res_bootstrap_coefs <- replicate(1000, {
  new_y <- m0$fitted.values + sample(m0$residuals, replace = TRUE)
  coef(lm(new_y ~ trees$Girth))
})
```

- Compute the regression parameters for each sample and find standard deviation

```r
res_bootstrap_coefs[, 1:10]
```

```
##              [,1]  [,2]  [,3] [,4] [,5]  [,6]  [,7]  [,8]  [,9] [,10]
## (Intercept) -38.4 -38.5 -34.0  -36  -35 -31.7 -38.6 -40.8 -30.5 -38.8
## trees$Girth   5.2   5.1   4.8    5    5   4.6   5.2   5.3   4.6   5.2
```

```r
apply(res_bootstrap_coefs, 1, sd)
```

```
## (Intercept) trees$Girth
##        3.31        0.24
```

- Compare with ordinary bootstrap

```r
apply(bootstrap_coefs, 1, sd)
```

```
## (Intercept)       Girth
##        3.98        0.32
```

- Compare with lm()

```r
coef(summary(m0))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -36.9       3.37     -11  7.6e-12
## Girth            5.1       0.25      20  8.6e-19
```

8 Maximum likelihood estimation of a probability

8.1 Estimating a probability

- Assume that we want to estimate a probability p of a certain event, e.g.
  - the probability that a bank customer will default their loan
  - the probability that a customer will buy a certain product
- We take a sample of n observations $Y_1, \dots, Y_n$, where
  - $Y_i = 1$ if the event happens,
  - $Y_i = 0$ if the event does not happen,
  - The Y_i, $i = 1, \dots, n$, are independent random variables with $P(Y_i = 1) = p$.
- Let $X = \sum_i Y_i$ be the number of ones in our sample. The natural estimate for p is

$$\hat{P} = \frac{X}{n}.$$

  - Theoretical justification?

8.2 The likelihood function

- Idea: choose $\hat{P}$ to be the value of p that makes our observations *as likely as possible.*
- Suppose we have observed $Y_1 = y_1, \dots, Y_n = y_n$. The probability of observing this is

$$P(Y_1 = y_1, \dots, Y_n = y_n) = P(Y_1 = y_1) \cdot \dots \cdot P(Y_n = y_n).$$

- Note that

$$P(Y_i = y_i) = \begin{cases} p, & y_i = 1, \\ (1-p), & y_i = 0. \end{cases}$$

- Therefore, if we let $x = \sum_i y_i$ be the number of 1's in our sample,

$$P(Y_1 = y_1, \dots, Y_n = y_n) = p^x \cdot (1-p)^{(n-x)}.$$

- This probability depends on the value of p. We may think of it as a function

$$L(p) = P(Y_1 = y_1, \dots, Y_n = y_n) = p^x \cdot (1-p)^{(n-x)}.$$

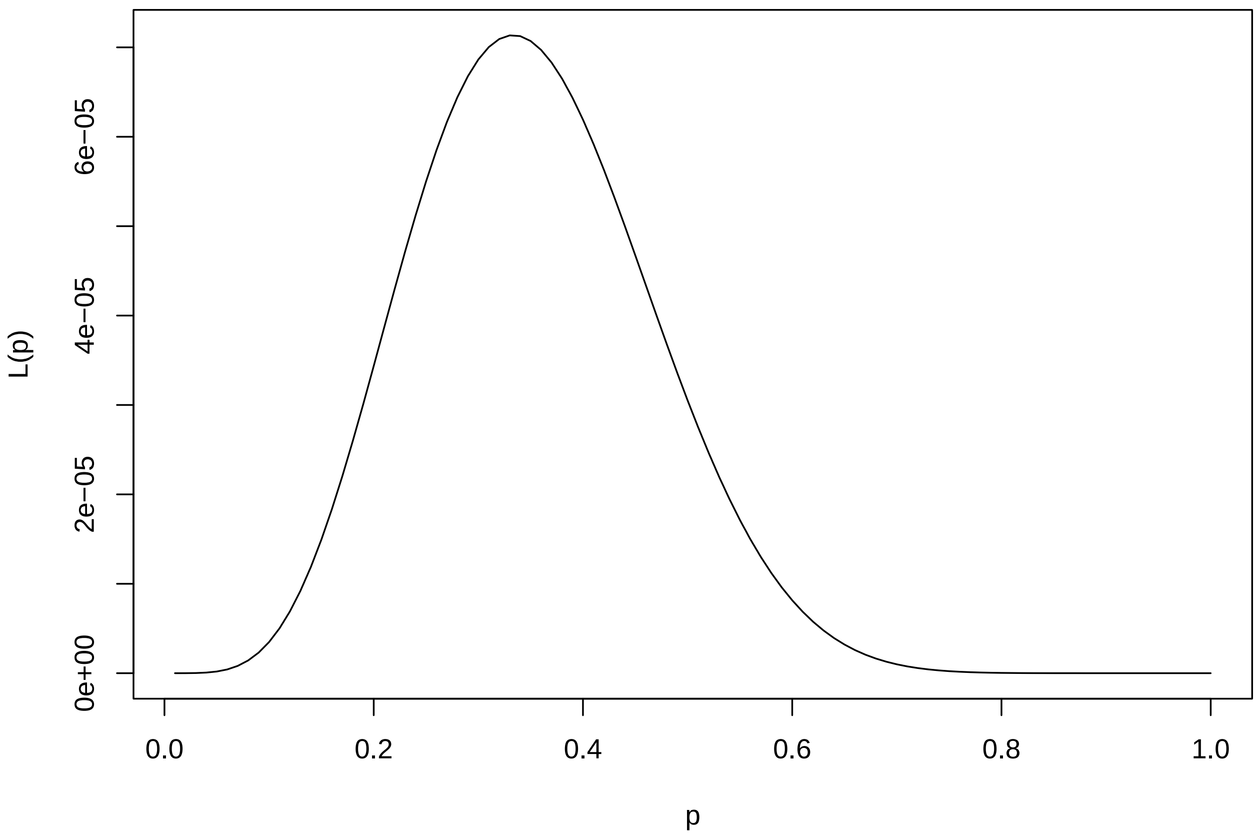
  – This is called the **likelihood function**.
- The **maximum likelihood estimate** $\hat{p}$ is the value of p that maximizes the likelihood function.

8.3 Likelihood function - example

- **Example:** Suppose we take a sample of $n = 15$ observations. We observe 5 ones and 10 zeros. The likelihoodfunction becomes

$$L(p) = p^5(1-p)^{10}$$

- We plot the graph of $L(p)$:



- The probability of our observations seems to be largest when p is around 1/3.

8.4 The log-likelihood function

- We seek the value of p that maximizes the likelihood function
$$L(p) = p^x \cdot (1-p)^{(n-x)}.$$
- Recall that $\ln(x)$ is a strictly increasing function.
- The value of p that maximizes $L(p)$ also maximizes $\ln(L(p))$.
- This is the **log-likelihood function**
$$l(p) = \ln(L(p)) = x\ln(p) + (n-x)\ln(1-p).$$
- It is often easier to maximize the log-likelihood function.

8.5 Maximum likelihood estimation

- In order to maximize
$$l(p) = x\ln(p) + (n-x)\ln(1-p),$$
we differentiate
$$l'(p) = \frac{x}{p} - \frac{n-x}{1-p}.$$

- The maximum must be found in a point with $l'(p) = 0$. Thus, we solve

$$l'(p) = \frac{x}{p} - \frac{n-x}{1-p} = 0.$$

- Multiply by $p(1-p)$ to get

$$\begin{aligned} x(1-p) - (n-x)p &= 0 \\ x - xp - np + xp &= 0 \\ x &= np \\ p &= \frac{x}{n}. \end{aligned}$$

- Note that this must indeed be a maximum point since

$$\lim_{p \to 0} l(p) = \lim_{p \to 1} l(p) = -\infty.$$

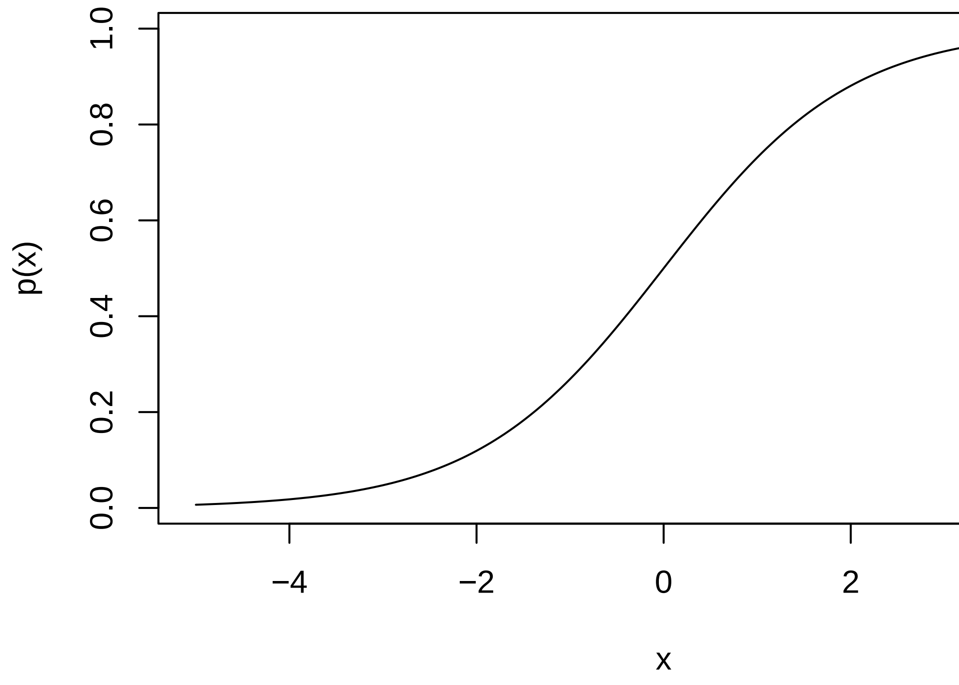
- Our **maximum likelihood estimate** of p is $\hat{p} = \frac{x}{n}$.

9 Maximum likelihood for logistic regression

9.1 The logistic regression model

- Estimation of a probability was a simple use of maximum likelihood estimation, which could easily have been treated by more direct methods.
- **Logistic regression** is a more complex case, where we want to model a probability $p(x)$ that depends on a predictor variable x.
  - E.g. the probability of a customer buying a certain product as a function of their monthly income.
- In logistic regression, $p(x)$ is modelled by a logistic function

$$p(x) = \frac{1}{1 + e^{-(\alpha + \beta x)}}.$$



- Graph of $p(x)$ when $\alpha = 0$ and $\beta = 1$.
  - α determines how steep the graph is.
  - β shifts the graph along the x-axis.
- How to estimate α and β?

9.2 Maximum likelihood estimation for logistic regression

- A sample consists of $(x_1, y_1), \dots, (x_n, y_n)$, where x_i is the predictor and y_i is the response, which is either 0 or 1.
- The probability of our observations is

$$P(Y_1 = y_1, \dots, Y_n = y_n) = \prod_i p(Y_i = y_i) = \prod_i p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

  since

$$p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} = \begin{cases} p(x_i), & y_i = 1, \\ 1 - p(x_i), & y_i = 0. \end{cases}$$

- Inserting what $p(x_i)$ is, we obtain a function of the unknown parameters α and β:

$$L(\alpha, \beta) = P(Y_1 = y_1, \dots, Y_n = y_n) = \prod_i \Big(\frac{1}{1 + e^{-(\alpha + \beta x_i)}}\Big)^{y_i} \Big(1 - \frac{1}{1 + e^{-(\alpha + \beta x_i)}}\Big)^{1-y_i}$$

- Again, it is easier to maximize the log-likelihood

$$l(\alpha, \beta) = \sum_i \Big(y_i \ln(p(x_i)) + (1 - y_i)\ln(1 - p(x_i))\Big).$$

- However, this maximum can only be found using numerical methods.

9.3 Logistic regression - example

- We consider a dataset from the ISLR package on whether or not 10000 bank costumers will default their loans.
  - Response: default (1=yes, 0=no)
  - Predictor: income

```
library(ISLR)
x<-Default$income/10000 # Annual income in 10000 dollars
y<-as.numeric(Default$default=="Yes") # Loan default, 1 means "Yes"
```

- We want to model the probability of default as a logistic function of income

$$p(x) = \frac{1}{1+e^{-(\alpha+\beta x)}}.$$

9.4 Logistic regression - example continued

- We make a function in R that computes the log-likelihood function as a function of the vector $\theta = (\alpha, \beta)^T$.
  - We first compute a vector px that contains all the probabilities $p(x_i)$.
  - Then we compute the vector logpy which contains all the $\ln(P(Y_i = y_i)) = y_i \ln(p(x_i)) + (1 - y_i)\ln(1 - p(x_i))$.
  - Finally, we compute the log-likelihood with the formula

$$l(\alpha, \beta) = \sum_i \Big(y_i \ln(p(x_i)) + (1 - y_i)\ln(1 - p(x_i)) \Big)$$

```
loglik <- function(theta) {
alpha=theta[1]
beta=theta[2]
px<-1/(1+exp(-alpha-beta*x))
logpy<-y*log(px) + (1-y)*log(1-px)
sum(logpy)
}
loglik(c(2,2))
```

```
## [1] -84250
```

- We maximize the log-likelihood using the `optim()` function in R.
  - It needs an initial guess of θ. Here we use `c(2,2)`.
  - The option `control=list(fnscale=-1)` ensures that we maximize rather than minimize.

```
optim(c(2,2),loglik,control=list(fnscale=-1))
```

```
## $par
## [1] -3.099 -0.081
##
## $value
## [1] -1458
##
## $counts
## function gradient
##       69       NA
##
## $convergence
## [1] 0
##
```

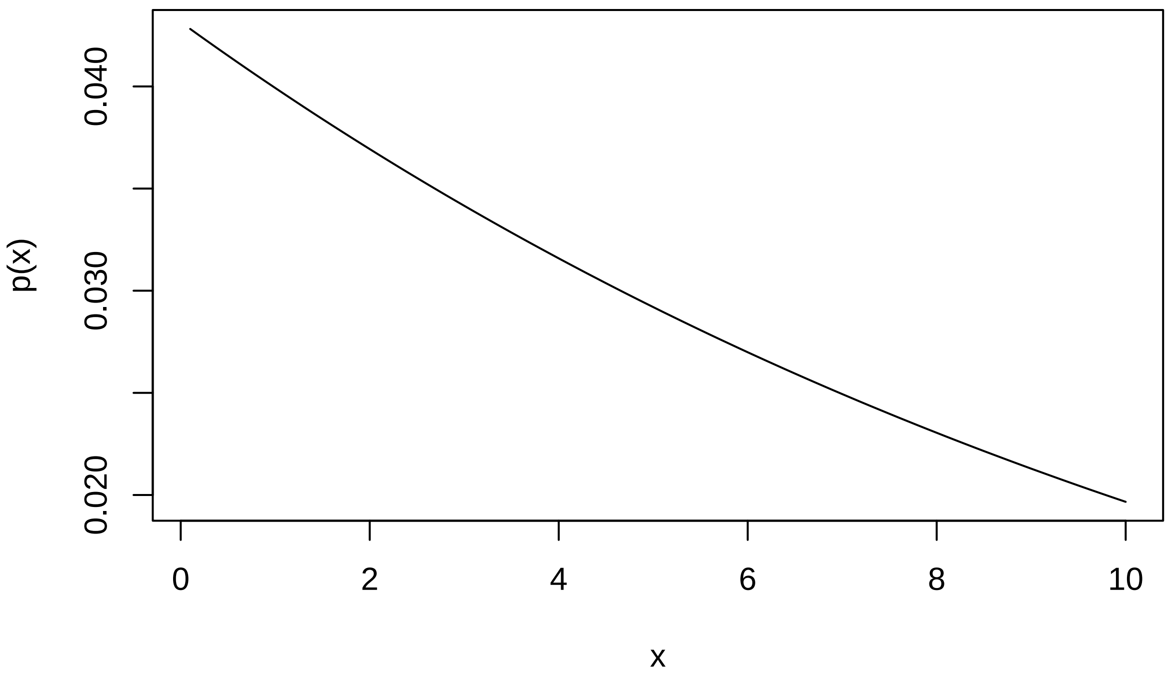
```
## $message
## NULL
```

- We obtain the maximum likelihood estimates $\hat{\alpha} = -3.099$ and $\hat{\beta} = -0.081$.

9.5 Logistic regression - example continued

- We can plot the estimated logistic function

$$\hat{p}(x) = \frac{1}{1 + e^{3.099 + 0.081x}}.$$



- The maximum likelihood estimates of α and β can be found directly using R:

```
model<-glm(y~x,family="binomial")
summary(model)
```

```
##
## Call:
## glm(formula = y ~ x, family = "binomial")
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.0941     0.1463  -21.16   <2e-16 ***
## x            -0.0835     0.0421   -1.99    0.047 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 2916.7  on 9998  degrees of freedom
## AIC: 2921
```

```
##
## Number of Fisher Scoring iterations: 6
```

10 Maximum likelihood estimation with continuous variables

10.1 The probability density function

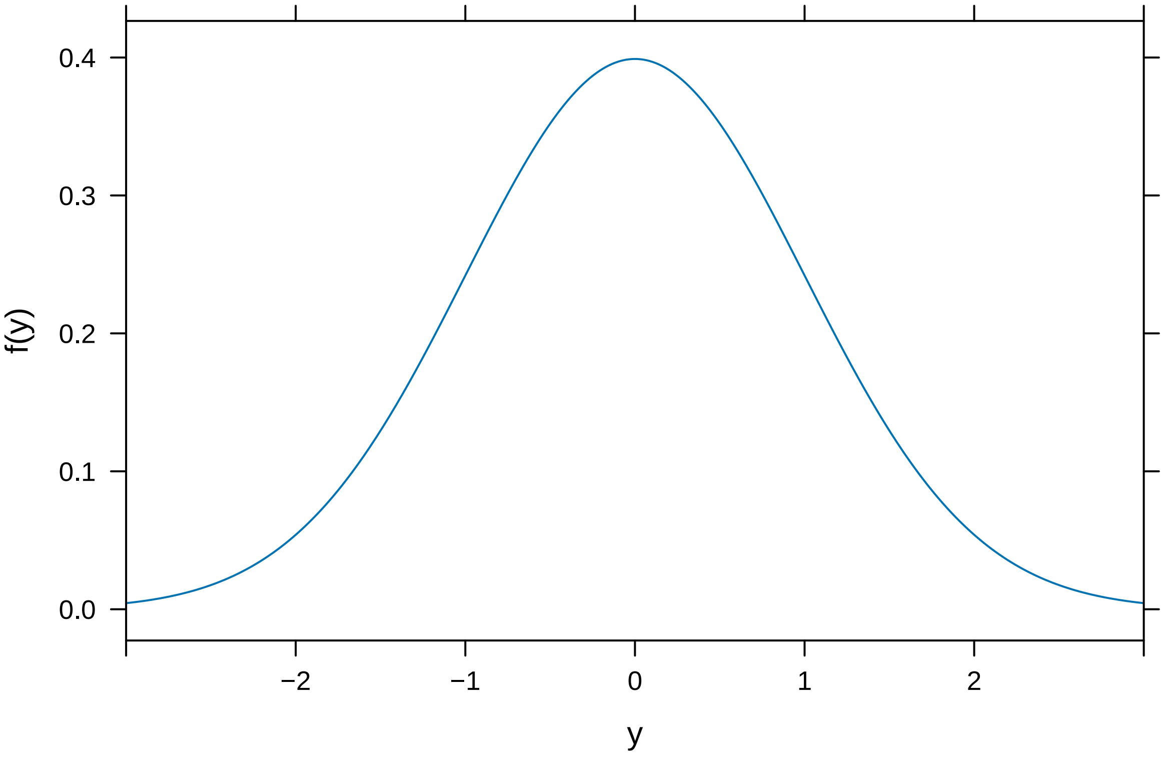
- Suppose we have a sample $Y_1, \dots, Y_n$ of independent variables with

$$Y_i \sim N(\mu, \sigma)$$

- We would like to estimate the unknown parameters μ and σ.
- For a continuous variable Y we have $P(Y = y) = 0$ for all y.
  - We cannot use the probability of observing a given outcome to define the likelihood function.
- Instead we consider the probability density function

$$f(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right]$$

- E.g. for $\mu = 0$ and $\sigma = 1$:



- The most likely values are the ones where $f(y)$ is large.
- Thus we will use $f(y)$ as a measure of how likely it is to observe $Y = y$.

10.2 The likelihood function for n observations

- Since $Y_1, \dots, Y_n$ are independent observations, the joint density function becomes a product of marginal densities:
$$f_{(Y_1,\dots,Y_n)}(y_1,\dots,y_n) = \prod_i f_{Y_i}(y_i).$$
- If we have observed a sample $Y_1 = y_1, \dots, Y_n = y_n$, our likelihood function is defined as

$$L(\mu,\sigma) = f_{(Y_1,\dots,Y_n)}(y_1,\dots,y_n) = \prod_i f_{Y_i}(y_i) = \prod_i \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-\mu)^2}{2\sigma^2}}. \tag{1}$$

- The maximum likelihood estimate $(\hat{\mu}, \hat{\sigma})$ is the value of (μ, σ) that maximizes the likelihood function.

10.3 Log-likelihood function in the normal case

- We found the log-likelihood function

$$L(\mu,\sigma) = \prod_i \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-\mu)^2}{2\sigma^2}}. \tag{2}$$

- Again it is easier to maximize the log-likelihood function.

$$l(\mu,\sigma) = \ln(L(\mu,\sigma)) = \sum_i \ln\Big(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i-\mu)^2}{2\sigma^2}}\Big)$$
$$= \sum_i \Big(-\ln(\sigma\sqrt{2\pi}) - \frac{(y_i-\mu)^2}{2\sigma^2}\Big) = -n\ln(\sigma\sqrt{2\pi}) - \sum_i \frac{(y_i-\mu)^2}{2\sigma^2}$$

- We find the partial derivatives and set them equal to 0. First with respect to μ:

$$\frac{\partial}{\partial\mu} l(\mu,\sigma) = \sum_i \frac{2(y_i-\mu)}{2\sigma^2} = \frac{1}{\sigma^2}\sum_i (y_i-\mu) = \frac{1}{\sigma^2}\Big(\sum_i y_i - n\mu\Big) = 0$$

- Vi får at $n\mu = \sum_i y_i$, så $\mu = \frac{1}{n}\sum_i y_i = \bar{y}$.
- Then with respect to σ:

$$\frac{\partial}{\partial\sigma} l(\mu,\sigma) = -\frac{n}{\sigma} + \sum_i \frac{(y_i-\mu)^2}{\sigma^3} = 0$$

- We multiply by σ and insert $\mu = \bar{y}$:

$$-n + \sum_i \frac{(y_i-\bar{y})^2}{\sigma^2} = 0$$
$$n = \frac{1}{\sigma^2}\sum_i (y_i-\bar{y})^2$$
$$\sigma^2 = \frac{1}{n}\sum_i (y_i-\bar{y})^2$$

- In total we get the maximum likelihood estimates:

$$\hat{\mu} = \frac{1}{n}\sum_i y_i = \bar{y}$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_i (y_i - \bar{y})^2$$

10.4 Numerical solution - normal distribution

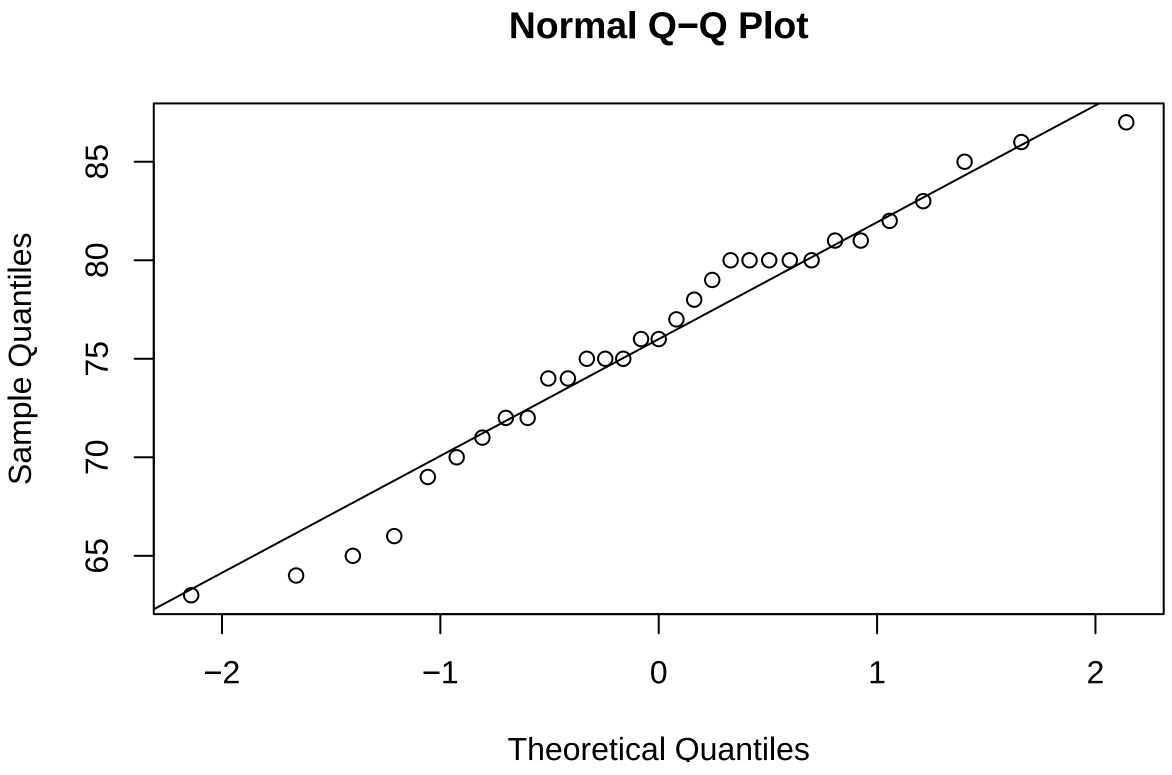
- The maximum likelihood estimates can also be found numerically. We consider again the `trees` data.

```
trees <- read.delim("https://asta.math.aau.dk/datasets?file=trees.txt")
head(trees)
```

```
##   Girth Height Volume
## 1   8.3     70     10
## 2   8.6     65     10
## 3   8.8     63     10
## 4  10.5     72     16
## 5  10.7     81     19
## 6  10.8     83     20
```

- We will assume that the variable `Height` is normally distributed.

```
qqnorm(trees$Height)
qqline(trees$Height)
```



10.5 Numerical solution - normal distribution

- We define the log-likelihood as a function of the parameter vector $\theta = (\mu, \sigma)^T$.

  - `dnorm(y, mean = mu, sd = sigma)` gives the normal density $f(y)$ with mean μ and standard deviation σ evaluated at y.

```r
loglik_normal <- function(theta) {
  mu <- theta[1]
  sigma <- theta[2]
  y<-trees$Height
  fy<-dnorm(y , mean = mu, sd = sigma)
  sum(log(fy))
}
loglik_normal(c(1,5))
```

```
## [1] -3590
```

- We maximize again using `optim()`:

```r
optim(c(1, 5), loglik_normal,control=list(fnscale=-1))
```

```
## $par
## [1] 76.0  6.3
##
## $value
## [1] -101
##
## $counts
## function gradient
##      103       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

- We can compare this to the theoretical formulas for the maximum likelihood estimates:

$$\hat{\mu} = \bar{y},$$

$$\hat{\sigma} = \sqrt{\frac{1}{n}\sum_i (y_i - \bar{y})^2} = \sqrt{\frac{n-1}{n}}s.$$

```r
mean(trees$Height)
```

```
## [1] 76
```

```r
sd(trees$Height)
```

```
## [1] 6.4
```

```r
n <- length(trees$Height)
sd(trees$Height)*sqrt((n-1)/n)
```

```
## [1] 6.3
```

11 Properties of maximum likelihood estimators

- Suppose $\theta \in \mathbb{R}$ is a parameter that we estimate by $\hat{\theta}$ using maximum likelihood estimation. Then (under suitable conditions) one may show the following mathematically.

- **Consistency:** For all $\varepsilon > 0$,
$$\lim_{n\to\infty} P(|\theta - \hat{\theta}| > \varepsilon) = 0$$
- **Central limit theorem:** When $n \to \infty$,
$$\sqrt{n}(\hat{\theta} - \theta) \to N(0, \sigma_\theta^2).$$
That is, for large n,
$$\sqrt{n}(\hat{\theta} - \theta) \approx N(0, \sigma_\theta^2),$$
or equivalently,
$$\hat{\theta} \approx N\Big(\theta, \frac{\sigma_\theta^2}{n}\Big).$$