Machine Learning

Mohamad Ghassany

2018-09-26

Contents

Welcome **5**
Course Overview . . . 5

Introduction **7**
What is Machine Learning ? . . . 7
Supervised Learning . . . 7
Unsupervised Learning . . . 13

I Supervised Learning **17**

Regression **21**

1 Linear Regression **21**
1.1 Notation . . . 21
1.2 Model Representation . . . 22
1.3 Why Estimate f ? . . . 24
Prediction . . . 24
Inference . . . 25
1.4 Simple Linear Regression Model . . . 25
1.5 Estimating the Coefficients . . . 25
1.6 Assessing the Accuracy of the Coefficient Estimates . . . 26
Hypothesis testing . . . 27
1.7 ANOVA and model fit . . . 28
1.7.1 ANOVA . . . 28
1.7.2 The R^2 Statistic . . . 30

PW 1 **33**
1.8 Some `R` basics . . . 33
1.8.1 Basic Commands . . . 33
1.8.2 Vectors . . . 34
1.8.3 Matrices, data frames and lists . . . 34
1.8.4 Graphics . . . 37
1.8.5 Distributions . . . 38
1.8.6 Working directory . . . 40
1.8.7 Loading Data . . . 40
1.9 Regression . . . 41
1.9.1 The `lm` function . . . 41
1.9.2 Predicting House Value: Boston dataset . . . 44

2 Multiple Linear Regression **49**

2.1 The Model . . . 49
2.2 Estimating the Regression Coefficients . . . 50
2.3 Some important questions . . . 51
2.3.1 Other Considerations in Regression Model . . . 54
2.4 How to select the best performing model . . . 54
Use the Adjusted R^2_{adj} for univariate models . . . 54
Have a look at the residuals or error terms . . . 55
Histogram of residuals . . . 57

PW 2 **59**
Multiple Linear Regression . . . 59
Reporting . . . 60

Classification **65**

3 Logistic Regression **65**
3.1 Introduction . . . 65
3.2 Logistic Regression . . . 65
3.2.1 The Logistic Model . . . 65
3.2.2 Estimating the Regression Coefficients . . . 67
3.2.3 Prediction . . . 67
3.3 Multiple Logistic Regression . . . 68
3.4 Example . . . 69
3.4.1 Case study: *The Challenger disaster* . . . 69

PW 3 **77**
Social Networks Ads . . . 77

Appendix **82**

A Introduction to `RStudio` **83**

B Review on hypothesis testing **85**

C Use of qualitative predictors **87**

D Model Selection **89**
Linear Model Selection and Best Subset Selection . . . 89
Forward Stepwise Selection . . . 89
Backward Stepwise Selection . . . 89
Estimating Test Error Using Mallow's Cp, AIC, BIC, Adjusted R-squared . . . 89
Estimating Test Error Using Cross-Validation . . . 89
Examples . . . 89
Best Subset Selection . . . 89
Forward Stepwise Selection and Model Selection Using Validation Set . . . 89
Model Selection Using Cross-Validation . . . 89

E References **91**

Welcome

Welcome! In this course you will learn about the state of the art of Machine Learning and also gain practice implementing and deploying machine learning algorithms.

The aim of Machine Learning is to build computer systems that can adapt to their environments and learn form experience. Learning techniques and methods from this field are successfully applied to a variety of learning tasks in a broad range of areas, including, for example, spam recognition, text classification, gene discovery, financial forecasting. The course will give an overview of many concepts, techniques, and algorithms in machine learning, beginning with topics such as linear regression and classification and ending up with topics such as kmeans and Expectation Maximization. The course will give you the basic ideas and intuition behind these methods, as well as a more formal statistical and computational understanding. You will have an opportunity to experiment with machine learning techniques in R and apply them to a selected problem.

Course Overview

Introduction

What is Machine Learning ?

What is Machine Learning?

Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

Tom Mitchell provides a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Machine Learning is also called Statistical Learning.

Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

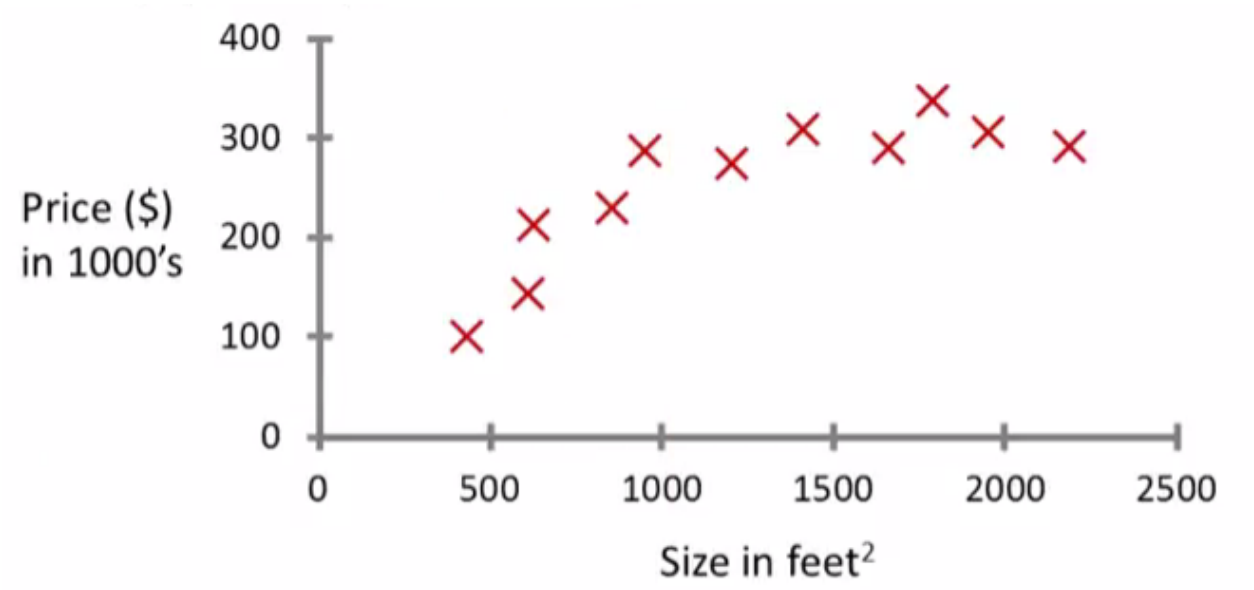
P = the probability that the program will win the next game.

In general, any machine learning problem can be assigned to one of two broad classifications:

Supervised learning and Unsupervised learning.

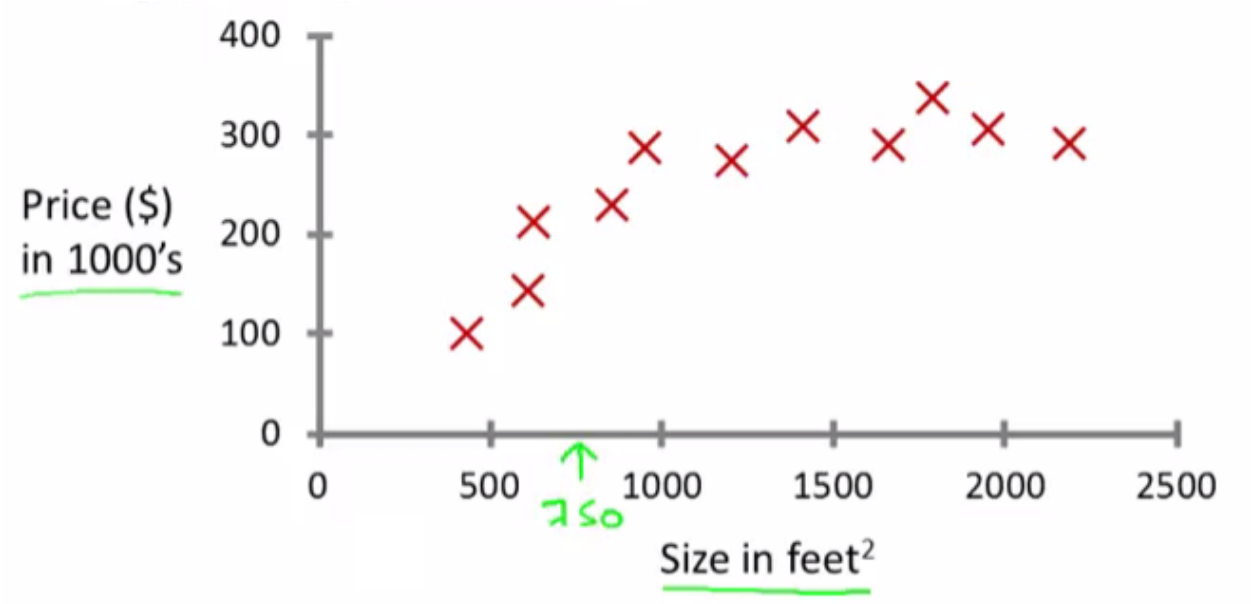
Supervised Learning

Supervised Learning is probably the most common type of machine learning problem. Let's start with an example of what is it. Let's say we want to predict housing prices. We plot a data set and it looks like this.



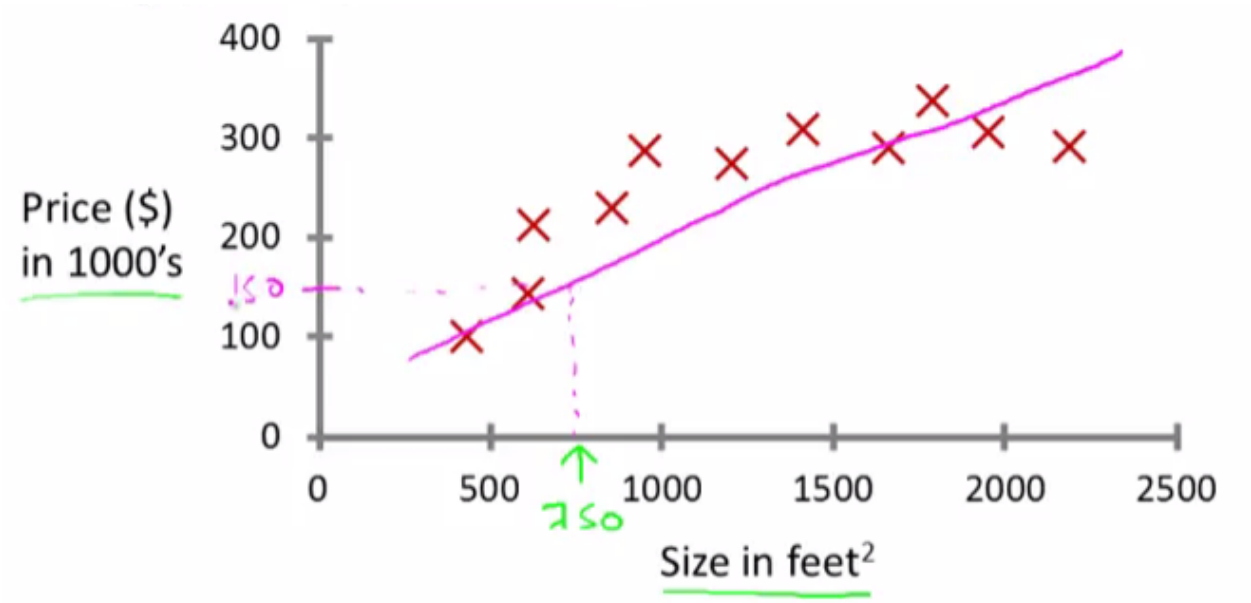
Here on the horizontal axis, the size of different houses in square feet, and on the vertical axis, the price of different houses in thousands of dollars.

So. Given this data, let's say we own a house that is, say 750 square feet and hoping to sell the house and we want to know how much we can get for the house.

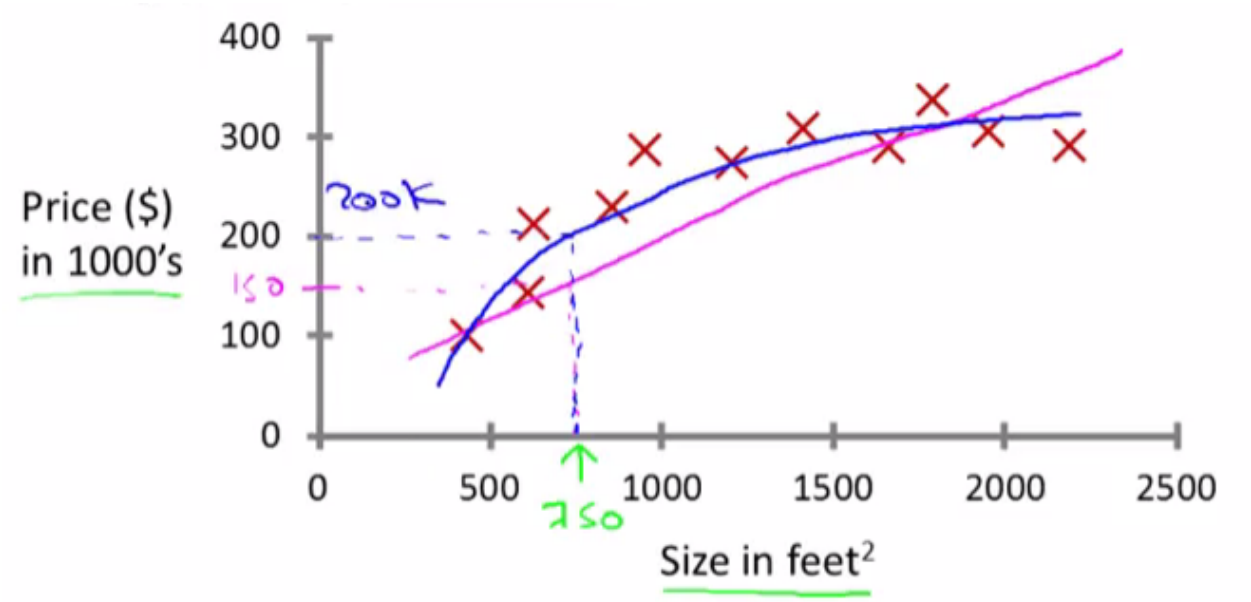


So how can the learning algorithm help?

One thing a learning algorithm might be able to do is put a straight line through the data or to **"fit"** a straight line to the data and, based on that, it looks like maybe the house can be sold for maybe about $150,000.



But maybe this isn't the only learning algorithm we can use. There might be a better one. For example, instead of sending a straight line to the data, we might decide that it's better to fit a *quadratic function* or a *second-order polynomial* to this data.



If we do that, and make a prediction here, then it looks like, well, maybe we can sell the house for closer to $200,000.

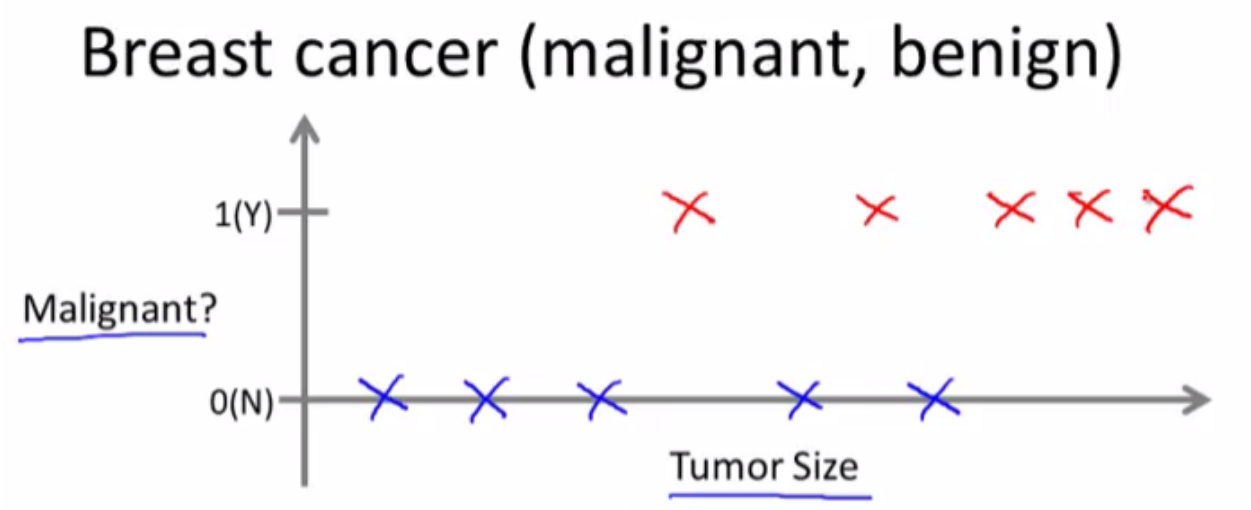
This is an example of a supervised learning algorithm.

The term supervised learning refers to the fact that we gave the algorithm a data set in which the **"right answers"** were given.

The example above is also called a regression problem. A regression problem is when we try to predict a **continuous** value output. Namely the price in the example.

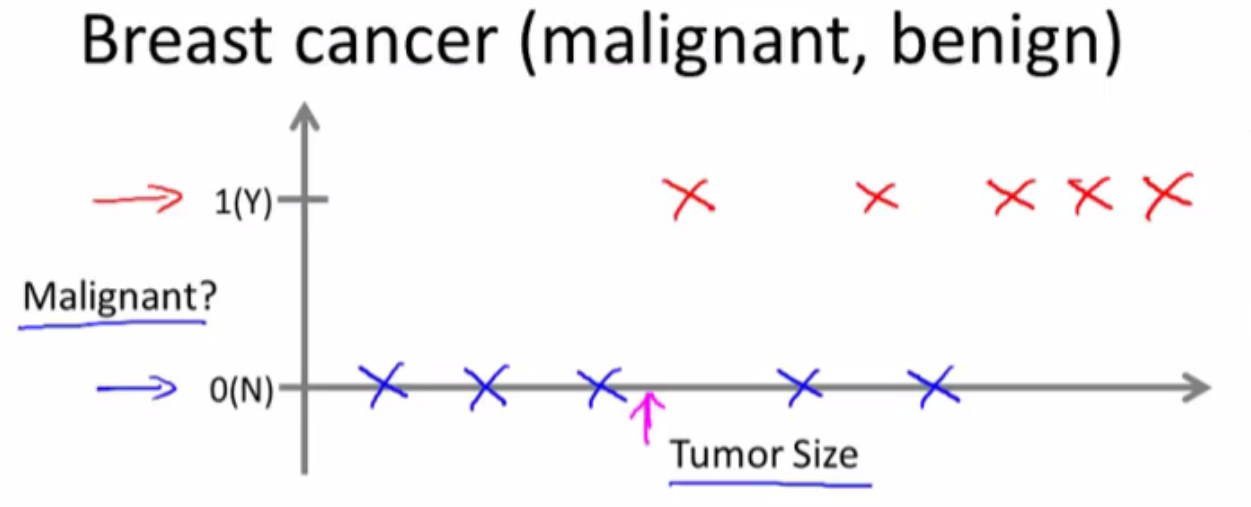
Here's another supervised learning example. Let's say we want to look at medical records and try to predict

of a breast cancer as malignant or benign. If someone discovers a breast tumor, a lump in their breast, a malignant tumor is a tumor that is harmful and dangerous and a benign tumor is a tumor that is harmless. Let's see a collected data set and suppose in the data set we have the size of the tumor on the horizontal axis and on the vertical axis we plot one or zero, yes or no, whether or not these are examples of tumors we've seen before are malignant (which is one) or zero if not malignant or benign.



In this data set we have five examples of benign tumors, and five examples of malignant tumors.

Let's say a person who tragically has a breast tumor, and let's say her breast tumor size is known (rose arrow in the following figure).

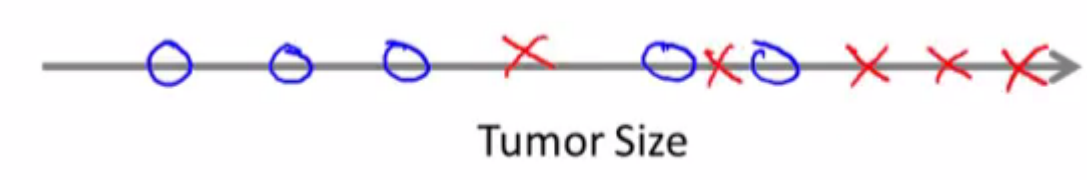


The machine learning question is, can you estimate what is the probability that a tumor is malignant versus benign? To introduce a bit more terminology this is an example of a ***classification*** problem.

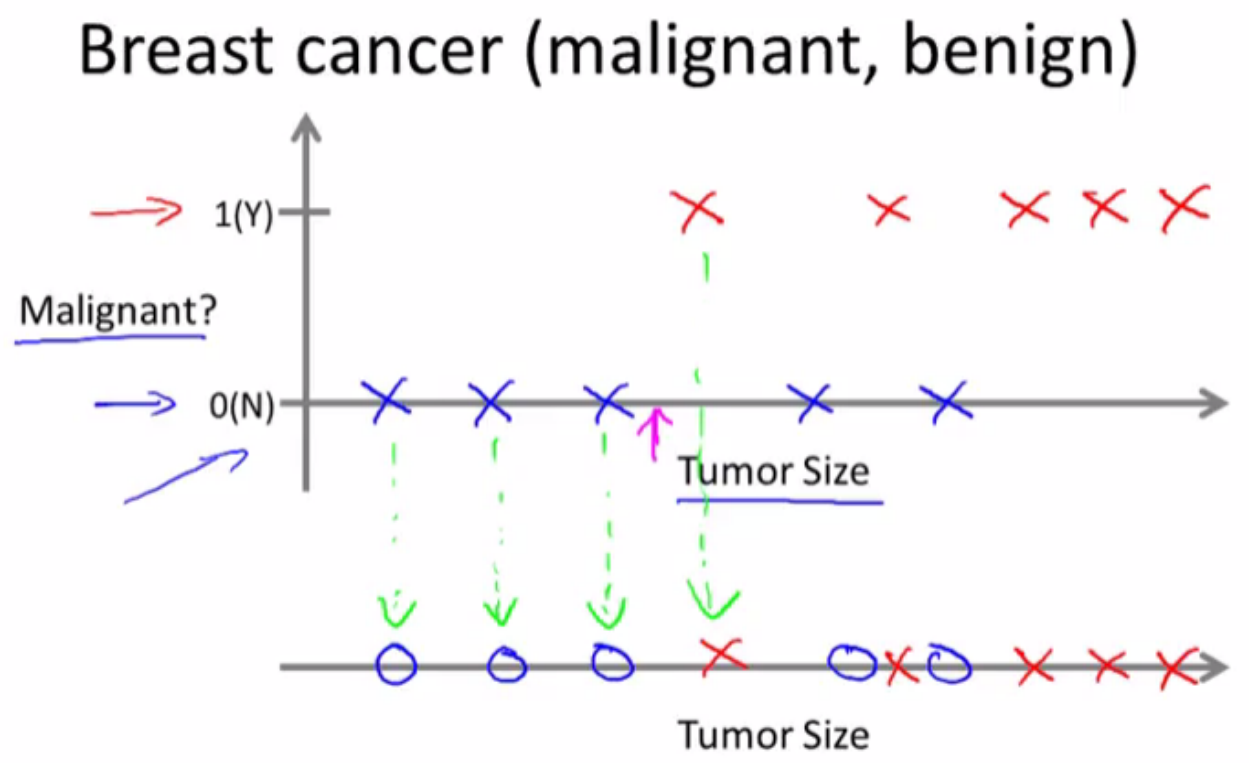
The term classification refers to the fact that here we're trying to predict a **discrete** value output: zero or one, malignant or benign. And it turns out that in classification problems sometimes you can have more than two values for the two possible values for the output.

In classification problems there is another way to plot this data. Let's use a slightly different set of symbols to plot this data. So if tumor size is going to be the attribute that we are going to use to predict malignancy

or benignness, we can also draw the data like this.



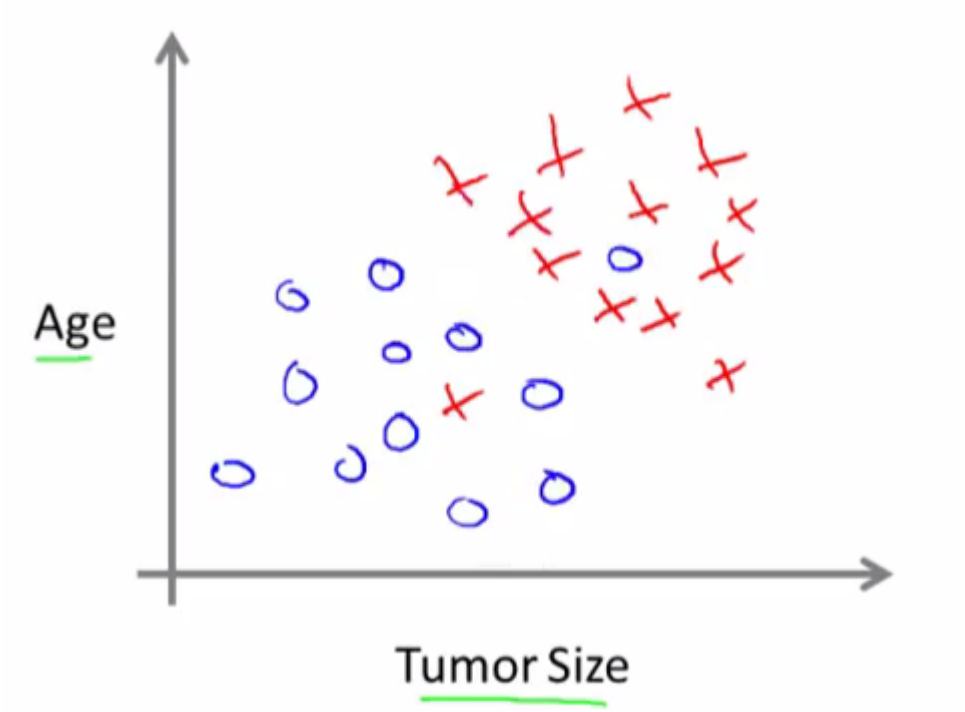
All we did was we took the data set on top and just mapped it down using different symbols. So instead of drawing crosses, we are now going to draw `O`'s for the benign tumors.



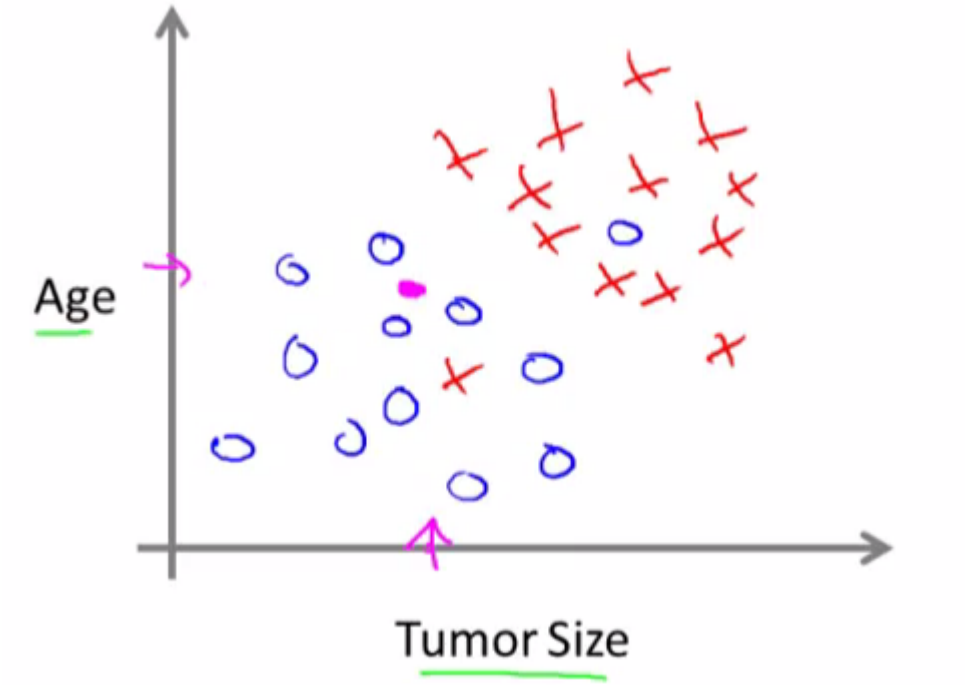
Now, in this example we use only one **feature** or one attribute, mainly, the *tumor size* in order to predict whether the tumor is malignant or benign.

In other machine learning problems we may have more than one feature.

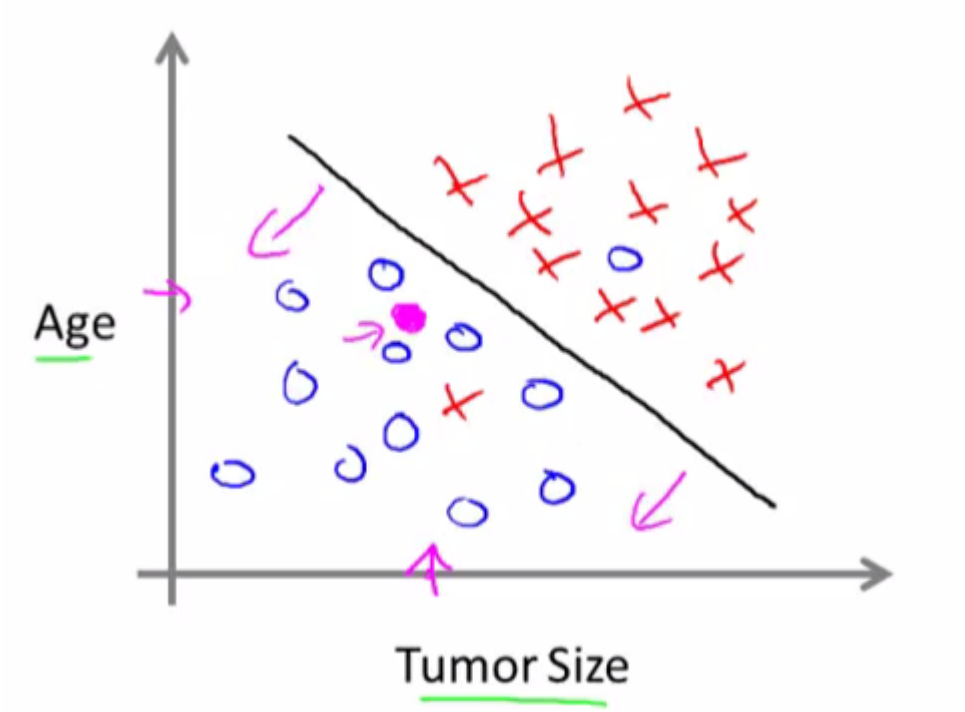
Here's an example. Let's say that instead of just knowing the tumor size, we know both the age of the patients and the tumor size. In that case maybe the data set will look like this.



So, let's say a person who tragically has a tumor. And maybe, their tumor size and age falls around there (rose point):



So given a data set like this, what the learning algorithm might do is throw a straight line through the data to try to separate out the malignant tumors from the benign ones. And with this, hopefully we can decide that the person's tumor falls on this benign side and is therefore more likely to be benign than malignant.



In this example we had **two features**, namely, the age of the patient and the size of the tumor. In other machine learning problems we will often have more features.

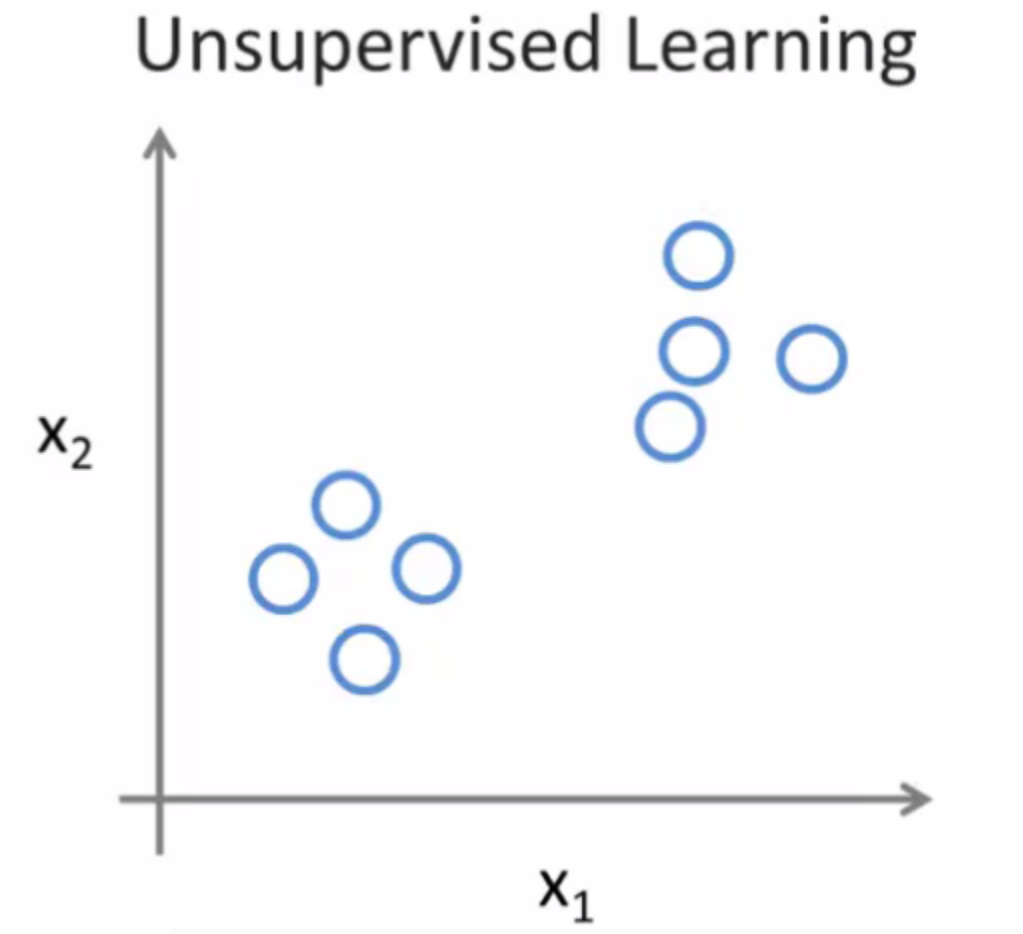
Most interesting learning algorithms is a learning algorithm that can deal with, not just two or three or five features, but an **infinite number of features**. So how do you deal with an infinite number of features. How do you even store an infinite number of things on the computer when your computer is gonna run out of memory.

Unsupervised Learning

The second major type of machine learning problem is called Unsupervised Learning.

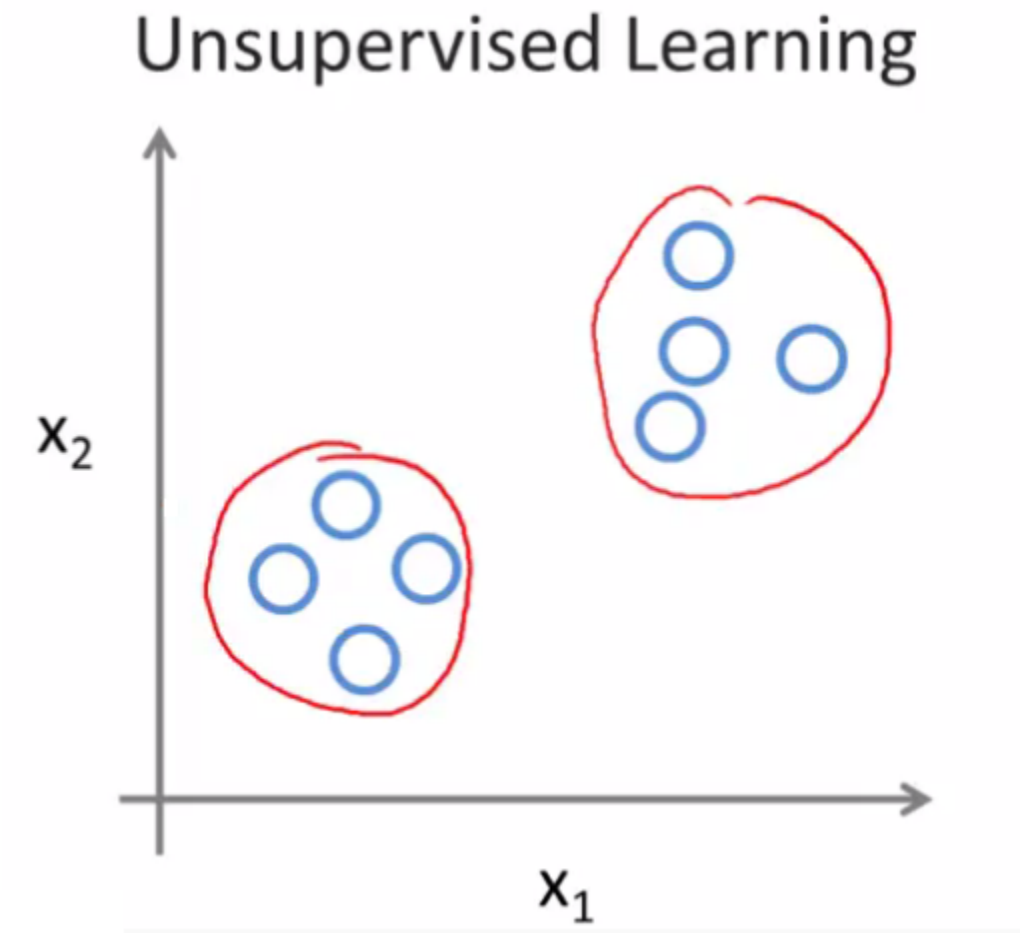
The difference between Unsupervised Learning and Supervised Learning is that in Supervised Learning we are told explicitly what is the so-called right answers (data are labeled).

In Unsupervised Learning, we're given data that doesn't have any labels or that all has the same label or really no labels. Like in this example:



So we're given the data set and we're not told what to do with it and we're not told what each data point is. Instead we're just told, here is a data set. Can you find some structure in the data?

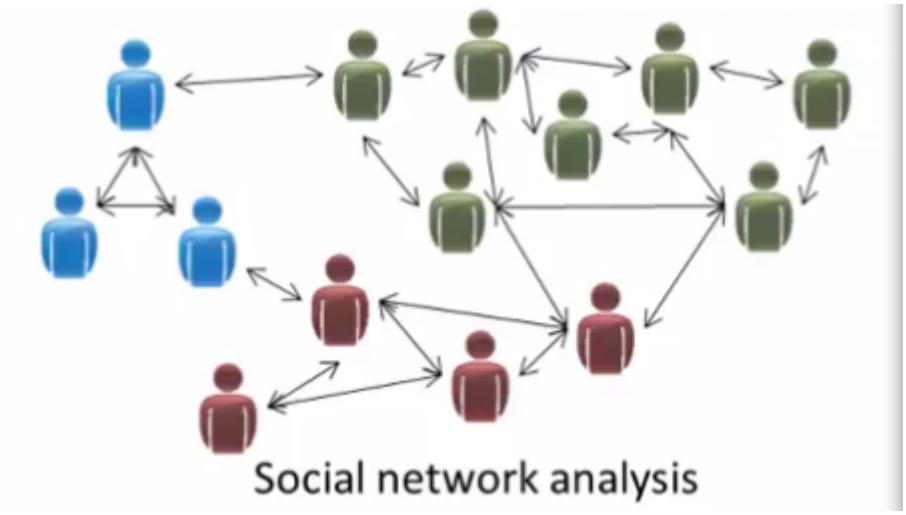
Given this data set, an Unsupervised Learning algorithm might decide that the data lives in two different clusters.



This is called a **clustering** algorithm.

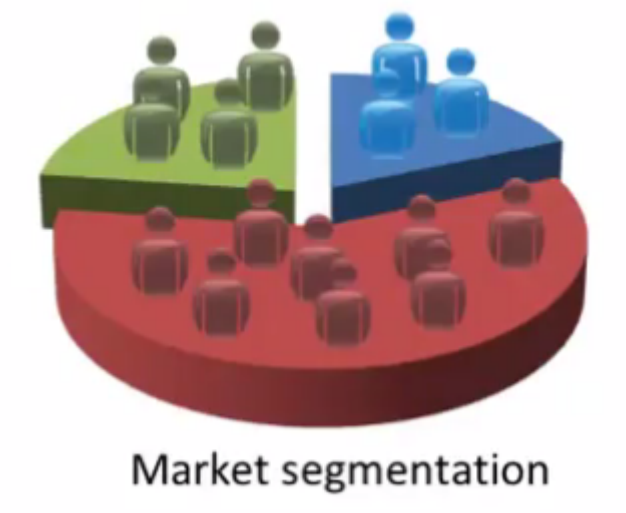
Here are two examples where Unsupervised Learning or clustering is used.

Social network analysis:



So given knowledge about which friends you email the most or given your Facebook friends or your Google+ circles, can we automatically identify which are cohesive groups of friends, also which are groups of people that all know each other?

Market segmentation:



Many companies have huge databases of customer information. So, can you look at this customer data set and automatically discover market segments and automatically group your customers into different market segments so that you can automatically and more efficiently sell or market your different market segments together?

This is Unsupervised Learning because we have all this customer data, but we don't know in advance what are the market segments and for the customers in our data set, we don't know in advance who is in market segment one, who is in market segment two, and so on. But we have to let the algorithm discover all this just from the data.

Part I

Supervised Learning

Regression

Chapter 1

Linear Regression

1.1 Notation

In general, we will let x_{ij} represent the value of the jth variable for the ith observation, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, p$. We will use i to index the samples or observations (from 1 tp n) and j will be used to index the variables (or features) (from 1 to p). We let $\mathbf{X}$ denote a $n \times p$ matrix whose (i, j)th element is x_{ij}. That is,

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \ldots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \ldots & x_{np} \end{pmatrix}$$

Note that it is useful to visualize $\mathbf{X}$ as a spreadsheet of numbers with n rows and p columns. We will write the rows of $\mathbf{X}$ as $x_1, x_2, \ldots, x_n$. Here x_i is a vector of length p, containing the p variable measurements for the ith observation. That is,

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}$$

(Vectors are by default represented as columns.)

We will write the columns of $\mathbf{X}$ as $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$. Each is a vector of length n. That is,

$$\mathbf{x}_j = \begin{pmatrix} \mathbf{x}_{1j} \\ \mathbf{x}_{2j} \\ \vdots \\ \mathbf{x}_{nj} \end{pmatrix}$$

Using this notation, the matrix $\mathbf{X}$ can be written as

$$\mathbf{X} = (\mathbf{x}_1 \mathbf{x}_2 \ldots \mathbf{x}_p)$$

or

$$\mathbf{X} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix}$$

The ^{T} notation denotes the transpose of a matrix or vector.

We use y_i to denote the ith observation of the variable on which we wish to make predictions. We write the set of all n observations in vector form as

$$\mathbf{y} = \begin{pmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{pmatrix}$$

Then the observed data consists of $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where each x_i is a vector of length p. (If $p = 1$, then x_i is simply a scalar).

1.2 Model Representation

Let's consider the example about predicting housing prices. We're going to use this data set as an example,

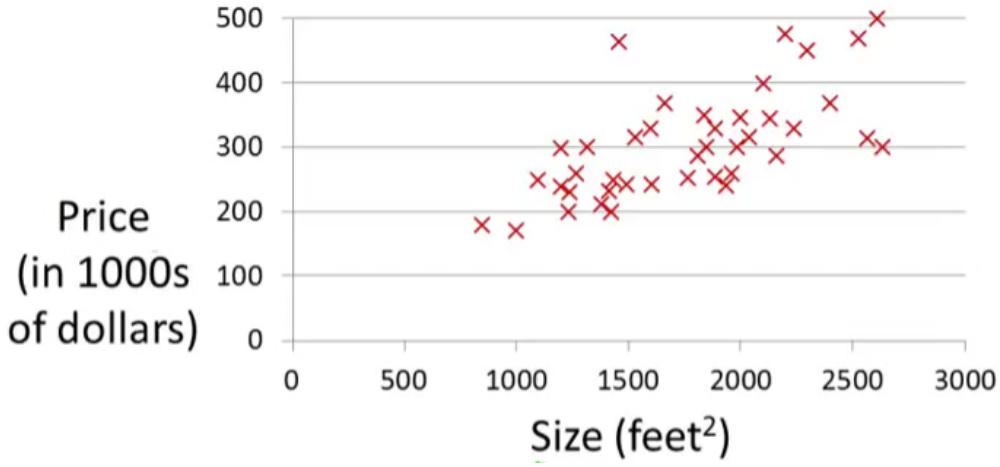


Figure 1.1:

Suppose that there is a person trying to sell a house of size 1250 square feet and he wants to know how much he might be able to sell the house for. One thing we could do is fit a model. Maybe fit a straight line to this data. Looks something like this,

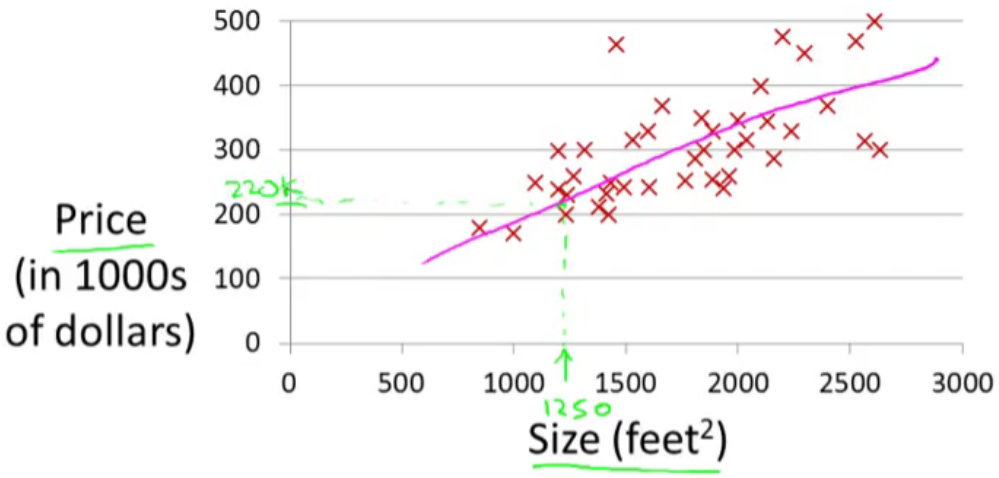


Figure 1.2:

and based on that, maybe he can sell the house for around $220,000. Recall that this is an example of a supervised learning algorithm. And it's supervised learning because we're given the "right answer" for each of our examples. More precisely, this is an example of a regression problem where the term regression refers to the fact that we are predicting a real-valued output namely the price.

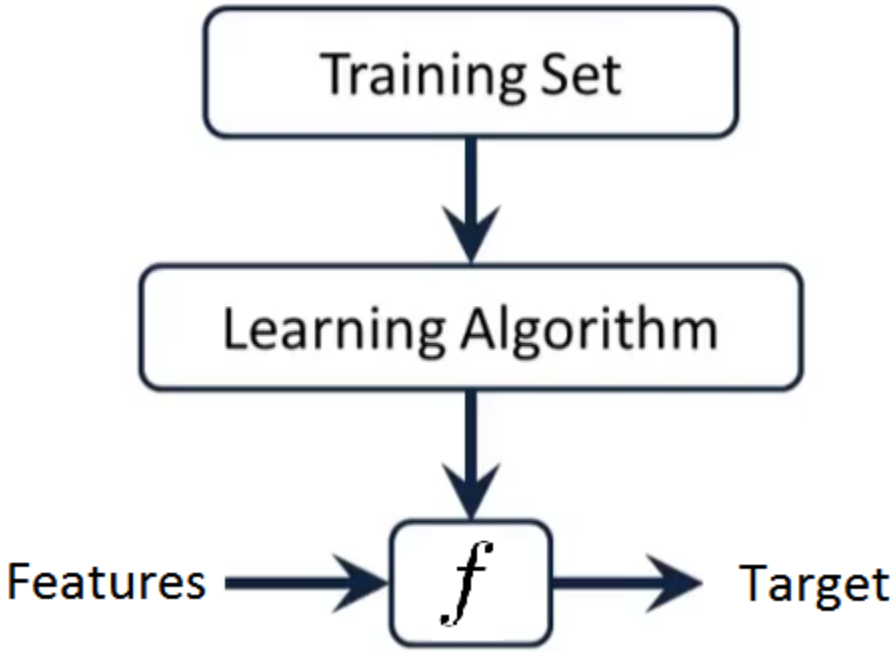
More formally, in supervised learning, we have a data set and this data set is called a **training set**. So for housing prices example, we have a training set of different housing prices and our job is to learn from this data how to predict prices of the houses.

Let's define some notation from this data set:

- The size of the house is the input variable.
- The house price is the output variable.
- The input variables are typically denoted using the variable symbol X,
- The inputs go by different names, such as *predictors*, *independent variables*, *features*, or sometimes just *variables*.
- The output variable is often called the *response*, *dependent variable* or *target*, and is typically denoted using the symbol Y.
- (x_i, y_i) is the ith training example.
- The set of $\{(x_i, y_i)\}$ is the training set.
- n is the number of training examples.

So here's how this supervised learning algorithm works. Suppose that we observe a quantitative response Y and p different predictors, $X_1, X_2, \ldots, X_p$. We assume that there is some relationship between Y and $X = (X_1, X_2, \ldots, X_p)$, which can be written in the very general form

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



Here f is some fixed but unknown function of $X_1, X_2, \ldots, X_p$, and ϵ is a random error term, which is independent of X and has mean zero. The f function is also called *hypothesis* in Machine Learning. In general, the function f may involve more than one input variable. In essence, Supervised Learning refers to a set of approaches for estimating f.

1.3 Why Estimate f ?

There are two main reasons that we may wish to estimate f: *prediction* and *inference*.

Prediction

In many situations, a set of inputs X are readily available, but the output Y cannot be easily obtained. In this setting, since the error term averages to zero, we can predict Y using

$$\hat{Y} = \hat{f}(X)$$

where $\hat{f}$ represents our estimate for f, and $\hat{Y}$ represents the resulting prediction for Y. Like in the example above about predicting housing prices.

We can measure the accuracy of $\hat{Y}$ by using a **cost function**. In the regression models, the most commonly-used measure is the *mean squared error* (MSE), given by

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

Inference

We are often interested in understanding the way that Y is affected as $X_1, X_2, \ldots, X_p$ change. In this situation we wish to estimate f , but our goal is not necessarily to make predictions for Y. We instead want to understand the relationship between X and Y, or more specifically, to understand how Y changes as a function of $X_1, X_2, \ldots, X_p$. In this case, one may be interested in answering the following questions:

- Which predictors are associated with the response?
- What is the relationship between the response and each predictor?
- Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?

1.4 Simple Linear Regression Model

Simple linear regression is a very straightforward approach for predicting a quantitative response Y on the basis of a single predictor variable X. It assumes that there is approximately a linear relationship between X and Y. Mathematically, we can write this linear relationship as

$$Y = \beta_0 + \beta_1 X + \epsilon$$
$$Y \approx \beta_0 + \beta_1 X$$

where β_0 and β_1 are two unknown constants that represent the *intercept* and *slope*, also known as ***coefficients*** or *parameters*, and ϵ is the error term.

Given some estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ for the model coefficients, we predict future inputs x using

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

where $\hat{y}$ indicates a prediction of Y on the basis of $X = x$. The *hat* symbol, $\hat{}$, denotes an estimated value.

1.5 Estimating the Coefficients

Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ be the prediction for Y based on the ith value of X. Then $e_i = y_i - \hat{y}_i$ represents the ith ***residual***.

We define the ***Residual Sum of Squares*** (**RSS**)[1] as

$$\begin{aligned} RSS &= e_1^2 + e_2^2 + \ldots + e_n^2 \\ &= \sum_{i=1}^{n} e_i^2 \end{aligned}$$

or equivantly as

$$\begin{aligned} RSS &= (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \end{aligned}$$

[1] Also known as **SSE**: Sum of Squared Errors.

The *least squares* approach chooses $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the RSS. The minimizing values can be shown to be[2]

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{s_{xy}}{s_x^2} = (s_x^2)^{-1} s_{xy}$$

and

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where:

- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the *sample mean*.
- $s_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$ is the *sample variance*. The sample standard deviation is $s_x = \sqrt{s_x^2}$.
- $s_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$ is the *sample covariance*. It measures the degree of linear association between $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$. Once scaled by $s_x s_y$, it gives the *sample correlation coefficient*, $r_{xy} = \frac{s_{xy}}{s_x s_y}$.

1- To find the optimal estimates for β_0 and β_1 we need a choice-criterion. In the case of the *least squares* approach (more precisely, the *ordinary least squares OLS*) this criterion is the residual sum of squares *RSS*: we calculate β_0 and β_1 that minimise the *RSS*.

2- Minimizing the *RSS* function requires to calculate the first order derivatives with respect to β_0 and β_1 and set them to zero.

3- Click here and watch the video to understand more about the residuals and least squares.

4- Click here to see the influence of the distance employed in the sum of squares. Try to minimize the sum of squares for the different datasets. The choices of intercept and slope that minimize the sum of squared distances for a kind of distance are not the optimal for a different kind of distance.

1.6 Assessing the Accuracy of the Coefficient Estimates

The standard error of an estimator reflects how it varies under repeated sampling. We have

$$\text{SE}(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\text{SE}(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\right]$$

where $\sigma^2 = Var(\epsilon)$

In general, σ^2 is unknown, but can be estimated from the data. The estimate of σ is known as the *residual standard error*, and is given by

$$\text{RSE} = \sqrt{\frac{\text{RSS}}{(n-2)}}$$

These standard errors can be used to compute *confidence intervals*. A 95% confidence interval is defined as a range of values such that with 95% probability, the range will contain the true unknown value of the parameter. It has the form

[2] They are unique and always exist. They can be obtained by solving $\frac{\partial}{\partial \beta_0}\text{RSS}(\beta_0, \beta_1) = 0$ and $\frac{\partial}{\partial \beta_1}\text{RSS}(\beta_0, \beta_1) = 0$.

$$\hat{\beta}_1 \pm 2 \cdot \text{SE}(\hat{\beta}_1)$$

That is, there is approximately a 95% chance that the interval

$$\left[\hat{\beta}_1 - 2 \cdot \text{SE}(\hat{\beta}_1), \hat{\beta}_1 + 2 \cdot \text{SE}(\hat{\beta}_1)\right]$$

will contain the true value of β_1. Similarly, a confidence interval for β_0 approximately takes the form

$$\hat{\beta}_0 \pm 2 \cdot \text{SE}(\hat{\beta}_0)$$

Hypothesis testing

Standard errors can also be used to perform *hypothesis tests* on the coefficients. The most common hypothesis test involves testing the *null hypothesis* of

$$H_0 : \text{There is no relationship between } X \text{ and } Y$$

versus the *alternative hypothesis*

$$H_1 : \text{There is some relationship between } X \text{ and } Y$$

Mathematically, this corresponds to testing

$$H_0 : \beta_1 = 0$$

versus

$$H_1 : \beta_1 \neq 0$$

since if $\beta_1 = 0$ then the simple linear regression model reduces to $Y = \beta_0 + \epsilon$, and X is not associated with Y.

To test the null hypothesis H_0, we compute a ***t-statistic***, given by

$$t = \frac{\hat{\beta}_1 - 0}{\text{SE}(\hat{\beta}_1)}$$

This will have a t-distribution (*Student*) with $n - 2$ degrees of freedom, assuming $\beta_1 = 0$.

Using statistical software, it is easy to compute the probability of observing any value equal to $|t|$ or larger. We call this probability the ***p-value***.

If p-value is small enough (typically under 0.01 (1% error) or 0.05 (5% error)) we reject the null hypothesis, that is we declare a relationship to exist between X and Y.

1.7 ANOVA and model fit

1.7.1 ANOVA

In this section we will see how the variance of Y is decomposed into two parts, each one corresponding to the regression and to the error, respectively. This decomposition is called the *ANalysis Of VAriance* (ANOVA).

Before explaining ANOVA, it is important to recall an interesting result: *the mean of the fitted values* $\hat{Y}_1, \ldots, \hat{Y}_n$ *is the mean of* $Y_1, \ldots, Y_n$. This is easily seen if we plug-in the expression of $\hat{\beta}_0$:

$$\frac{1}{n}\sum_{i=1}^{n}\hat{Y}_i = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\beta}_0 + \hat{\beta}_1 X_i\right) = \hat{\beta}_0 + \hat{\beta}_1\bar{X} = \left(\bar{Y} - \hat{\beta}_1\bar{X}\right) + \hat{\beta}_1\bar{X} = \bar{Y}.$$

The ANOVA decomposition considers the following measures of variation related with the response:

- SST = TSS = $\sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2$, the **Total Sum of Squares**. This is the *total variation* of $Y_1, \ldots, Y_n$, since SST = ns_y^2, where s_y^2 is the sample variance of $Y_1, \ldots, Y_n$.
- SSR = ESS = $\sum_{i=1}^{n}\left(\hat{Y}_i - \bar{Y}\right)^2$, the **Regression Sum of Squares** or **Explained Sum of Squares**[3]. This is the variation explained by the regression line, that is, *the variation from* $\bar{Y}$ *that is explained by the estimated conditional mean* $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$. SSR = $ns_{\hat{y}}^2$, where $s_{\hat{y}}^2$ is the sample variance of $\hat{Y}_1, \ldots, \hat{Y}_n$.
- SSE = RSS = $\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$, the **Sum of Squared Errors** or **Residual Sum of Squares**[4]. Is the variation around the conditional mean. Recall that SSE = $\sum_{i=1}^{n}\hat{\varepsilon}_i^2 = (n-2)\hat{\sigma}^2$, where $\hat{\sigma}^2$ is the sample variance of $\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_n$.

The ANOVA decomposition is

$$\underbrace{\text{SST}}_{\text{Variation of } Y_i's} = \underbrace{\text{SSR}}_{\text{Variation of } \hat{Y}_i's} + \underbrace{\text{SSE}}_{\text{Variation of } \hat{\varepsilon}_i's}$$

The graphical interpretation of this equation is shown in the following figures.

[3] Recall that SSR is different from RSS (Residual Sum of Squares)

[4] Recall that SSE and RSS (for $(\hat{\beta}_0, \hat{\beta}_1)$) are just different names for referring to the same quantity: SSE = $\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 = \sum_{i=1}^{n}\left(Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i\right)^2 = \text{RSS}\left(\hat{\beta}_0, \hat{\beta}_1\right)$.

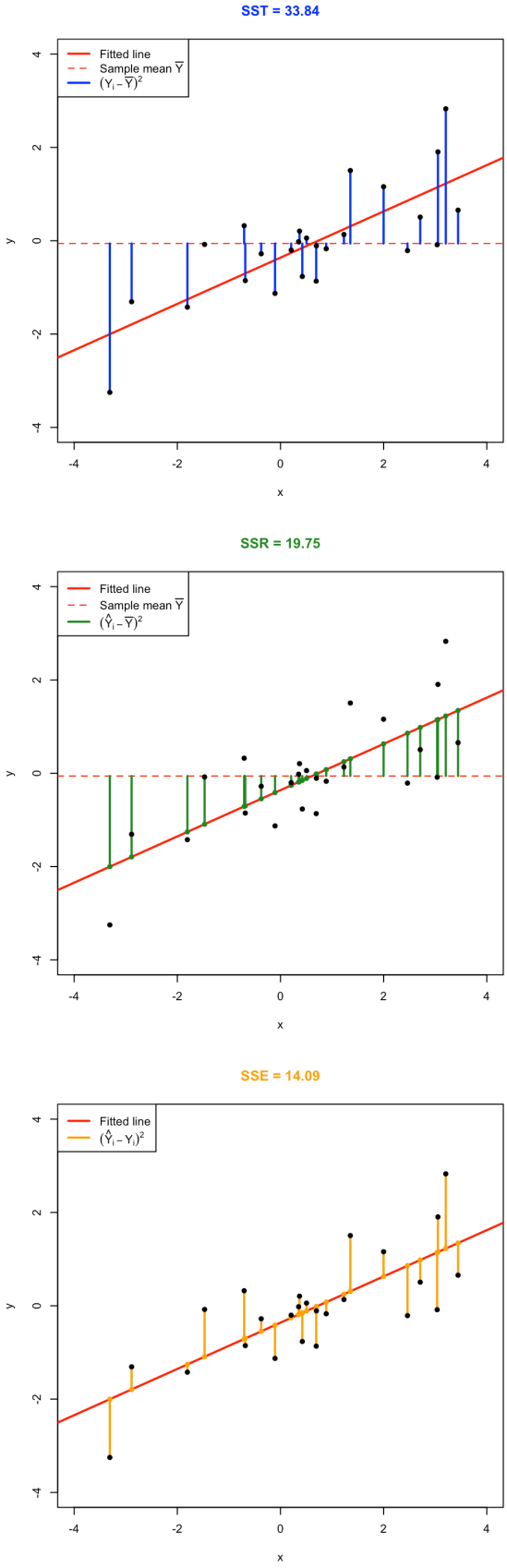


Figure 1.3: Visualization of the ANOVA decomposition. SST measures the variation of $Y_1, \ldots, Y_n$ with respect to $\bar{Y}$. SST measures the variation with respect to the conditional means, $\hat{\beta}_0 + \hat{\beta}_1 X_i$. SSE collects the variation of the residuals.

Below the ANOVA decomposition and its dependence on σ^2 and $\hat{\sigma}^2$. Application is also available here.

Note that the **animation** will not be displayed the first time it is browsed (The reason is because it is hosted at `https` websites with auto-signed SSL certificates). **To see it**, click on the link above. You will get a warning from your browser saying that *"Your connection is not private"*. Click in *"Advanced"* and **allow an exception** in your browser. The next time the animation will show up correctly.

The ANOVA table summarizes the decomposition of the variance. Here is given in the layout employed by `R`.

	Degrees of freedom	Sum Squares	Mean Squares	F-value	p-value
Predictor	1	SSR $= \sum_{i=1}^{n}\left(\hat{Y}_i - \bar{Y}\right)^2$	MSR $= \frac{\text{SSR}}{1}$	$\frac{\text{SSR}/1}{\text{SSE}/(n-2)}$	p
Residuals	$n-2$	SSE $= \sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$	MSE $= \frac{\text{SSE}}{n-2}$		
Total	$n-1$	SST $= \sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2$			

The `anova` function in `R` takes a model as an input and returns the ANOVA table.

The "F-value" of the ANOVA table represents the value of the F-statistic $\frac{\text{SSR}/1}{\text{SSE}/(n-2)}$. This statistic is employed to test

$$H_0 : \beta_1 = 0 \quad \text{vs.} \quad H_1 : \beta_1 \neq 0,$$

that is, the hypothesis of no linear dependence of Y on X. The result of this test is completely equivalent to the t-test for β_1 that we saw previously in the Hypothesis testing (this is something *specific for simple linear regression* – the F-test will not be equivalent to the t-test for β_1 in the Mulitple Linear Regression).

It happens that

$$F = \frac{\text{SSR}/1}{\text{SSE}/(n-2)} \overset{H_0}{\sim} F_{1,n-2},$$

where $F_{1,n-2}$ is the *Snedecor's F distribution*[5] with 1 and $n-2$ degrees of freedom.

If H_0 is true, then F is expected to be *small* since SSR will be close to zero. The p-value of this test is the same as the p-value of the t-test for $H_0 : \beta_1 = 0$.

1.7.2 The R^2 Statistic

To calculate R^2, we use the formula

[5] The $F_{n,m}$ distribution arises as the quotient of two independent random variables χ^2_n and χ^2_m, $\frac{\chi^2_n/n}{\chi^2_m/m}$.

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where $\text{TSS} = \sum(y_i - \bar{y})^2$ is the *total sum of squares*.

R^2 measures the *proportion of variability in Y that can be explained using X*. An R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression. A number near 0 indicates that the regression did not explain much of the variability in the response; this might occur because the linear model is wrong, or the inherent error σ^2 is high, or both.

It can be shown that in this simple linear linear regression setting that $R^2 = r^2$, where r is the correlation between X and Y:

$$r = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

R^2 does not measure the correctness of a linear model but its **usefulness** (for prediction, for *explaining the variance* of Y), assuming the model is correct.

Trusting blindly the R^2 can lead to catastrophic conclusions, since the model may not be correct.

So remember:

A large R^2 means *nothing* if the **assumptions of the model do not hold**. R^2 is the proportion of variance of Y explained by X, but, of course, *only when the linear model is correct*.

PW 1

1.8 Some R basics

1.8.1 Basic Commands

R uses functions to perform operations. To run a function called funcname,we type funcname(input1, input2) , where the inputs (or arguments) input1 and input2 tell R how to run the function. A function can have any number of inputs. For example, to create a vector of numbers, we use the function c() (for *concatenate*).

```
x <- c(1,3,2,5)
x
#ans> [1] 1 3 2 5
```

Note that the > is not part of the command; rather, it is printed by R to indicate that it is ready for another command to be entered. We can also save things using = rather than <-. Note that the answer in the code above is followed by #ans> while in the R console it is not.

```
x = c(1,6,2)
x
#ans> [1] 1 6 2
y = c(1,4,3)
length(x)
#ans> [1] 3
length(y)
#ans> [1] 3
x+y
#ans> [1]  2 10  5
```

Hitting the *up arrow* multiple times will display the previous commands, which can then be edited. This is useful since one often wishes to repeat a similar command.

The ls() function allows us to look at a list of all of the objects, such ls() as data and functions, that we have saved so far. The rm() function can be used to delete any object that we don't want.

```
ls()
#ans> [1] "x" "y"
rm(x)
ls()
#ans> [1] "y"
```

1.8.2 Vectors

```
# A handy way of creating sequences is the operator :
# Sequence from 1 to 5
1:5
#ans> [1] 1 2 3 4 5

# Storing some vectors
vec <- c(-4.12, 0, 1.1, 1, 3, 4)
vec
#ans> [1] -4.12  0.00  1.10  1.00  3.00  4.00

# Entry-wise operations
vec + 1
#ans> [1] -3.12  1.00  2.10  2.00  4.00  5.00
vec^2
#ans> [1] 16.97  0.00  1.21  1.00  9.00 16.00

# If you want to access a position of a vector, use [position]
vec[6]
#ans> [1] 4

# You also can change elements
vec[2] <- -1
vec
#ans> [1] -4.12 -1.00  1.10  1.00  3.00  4.00

# If you want to access all the elements except a position, use [-position]
vec[-2]
#ans> [1] -4.12  1.10  1.00  3.00  4.00

# Also with vectors as indexes
vec[1:2]
#ans> [1] -4.12 -1.00

# And also
vec[-c(1, 2)]
#ans> [1] 1.1 1.0 3.0 4.0
```

Do the following:

- Create the vector $x = (1, 7, 3, 4)$.
- Create the vector $y = (100, 99, 98, ..., 2, 1)$.
- Compute $x_3 + y_4$ and $\cos(x_3) + \sin(x_2)e^{-y_2}$. (Answers: `100`, `-0.9899925`)
- Set $x_3 = 0$ and $y_2 = -1$. Recompute the previous expressions. (Answers: `97`, `2.785875`)
- Index y by $x + 1$ and store it as `z`. What is the output? (Answer: `z` is `c(-1, 93, 100, 96)`)

1.8.3 Matrices, data frames and lists

```
# A matrix is an array of vectors
A <- matrix(1:4, nrow = 2, ncol = 2)
```

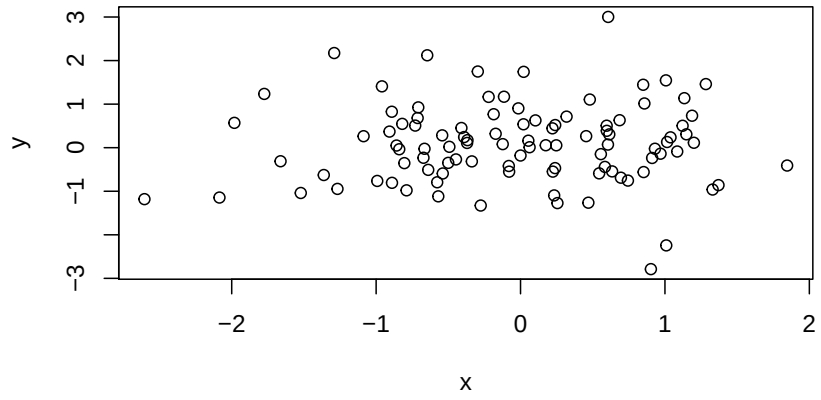
```
A
#ans>      [,1] [,2]
#ans> [1,]    1    3
#ans> [2,]    2    4

# Another matrix
B <- matrix(1:4, nrow = 2, ncol = 2, byrow = TRUE)
B
#ans>      [,1] [,2]
#ans> [1,]    1    2
#ans> [2,]    3    4

# Binding by rows or columns
rbind(1:3, 4:6)
#ans>      [,1] [,2] [,3]
#ans> [1,]    1    2    3
#ans> [2,]    4    5    6
cbind(1:3, 4:6)
#ans>      [,1] [,2]
#ans> [1,]    1    4
#ans> [2,]    2    5
#ans> [3,]    3    6

# Entry-wise operations
A + 1
#ans>      [,1] [,2]
#ans> [1,]    2    4
#ans> [2,]    3    5
A * B
#ans>      [,1] [,2]
#ans> [1,]    1    6
#ans> [2,]    6   16

# Accessing elements
A[2, 1] # Element (2, 1)
#ans> [1] 2
A[1, ] # First row
#ans> [1] 1 3
A[, 2] # Second column
#ans> [1] 3 4

# A data frame is a matrix with column names
# Useful when you have multiple variables
myDf <- data.frame(var1 = 1:2, var2 = 3:4)
myDf
#ans>   var1 var2
#ans> 1    1    3
#ans> 2    2    4

# You can change names
names(myDf) <- c("newname1", "newname2")
myDf
#ans>   newname1 newname2
```

```
#ans> 1        1        3
#ans> 2        2        4

# The nice thing is that you can access variables by its name with the $ operator
myDf$newname1
#ans> [1] 1 2

# And create new variables also (it has to be of the same
# length as the rest of variables)
myDf$myNewVariable <- c(0, 1)
myDf
#ans>   newname1 newname2 myNewVariable
#ans> 1        1        3             0
#ans> 2        2        4             1

# A list is a collection of arbitrary variables
myList <- list(vec = vec, A = A, myDf = myDf)

# Access elements by names
myList$vec
#ans> [1] -4.12 -1.00  1.10  1.00  3.00  4.00
myList$A
#ans>      [,1] [,2]
#ans> [1,]    1    3
#ans> [2,]    2    4
myList$myDf
#ans>   newname1 newname2 myNewVariable
#ans> 1        1        3             0
#ans> 2        2        4             1

# Reveal the structure of an object
str(myList)
#ans> List of 3
#ans>  $ vec : num [1:6] -4.12 -1 1.1 1 3 4
#ans>  $ A   : int [1:2, 1:2] 1 2 3 4
#ans>  $ myDf:'data.frame': 2 obs. of  3 variables:
#ans>   ..$ newname1     : int [1:2] 1 2
#ans>   ..$ newname2     : int [1:2] 3 4
#ans>   ..$ myNewVariable: num [1:2] 0 1
str(myDf)
#ans> 'data.frame': 2 obs. of  3 variables:
#ans>  $ newname1     : int  1 2
#ans>  $ newname2     : int  3 4
#ans>  $ myNewVariable: num  0 1

# A less lengthy output
names(myList)
#ans> [1] "vec"  "A"    "myDf"
```

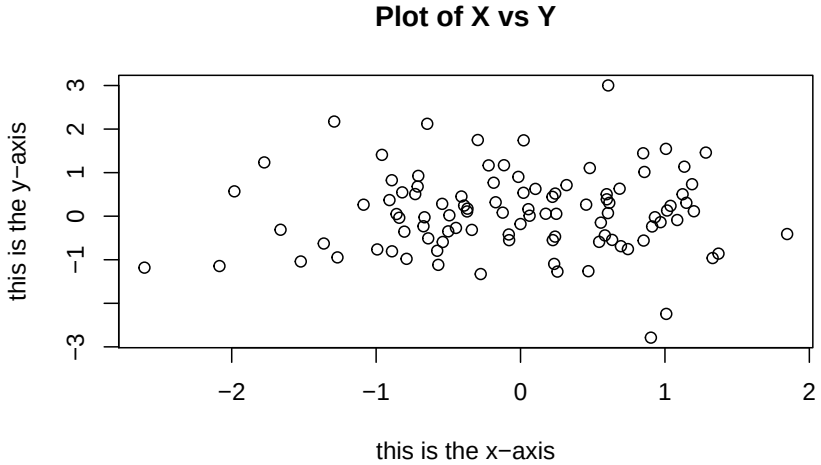
1.8.4 Graphics

The `plot()` function is the primary way to plot data in `R` . For instance, `plot(x,y)` produces a scatterplot of the numbers in `x` versus the numbers in `y`. There are many additional options that can be passed in to the `plot()` function. For example, passing in the argument `xlab` will result in a label on the `x-axis`. To find out more information about the `plot()` function, type `?plot`.

```
x=rnorm(100)
# The rnorm() function generates a vector of random normal variables,
# rnorm() with first argument n the sample size. Each time we call this
# function, we will get a different answer.
y=rnorm(100)
plot(x,y)
```

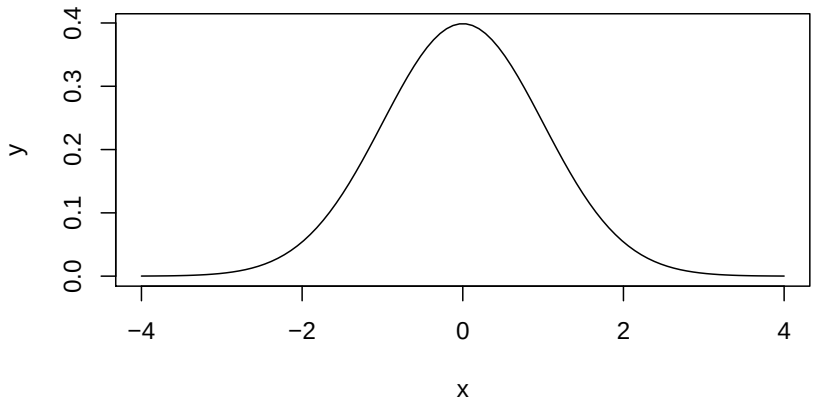



```
# with titles
plot(x,y,xlab="this is the x-axis",ylab="this is the y-axis",
main="Plot of X vs Y")
```



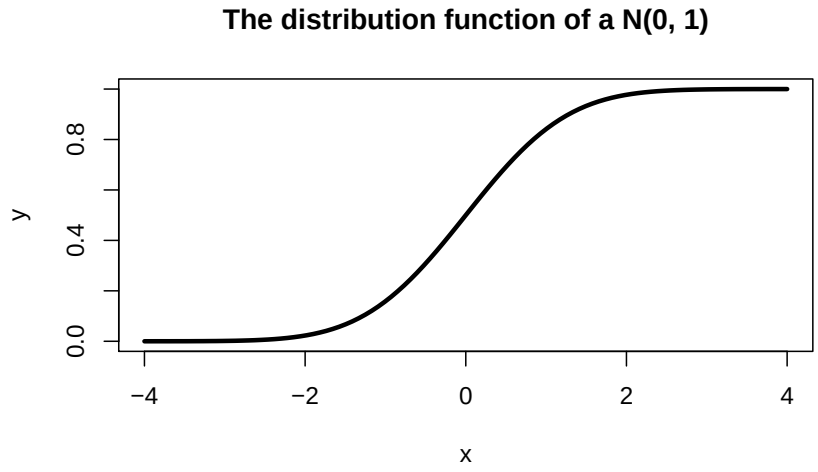
1.8.5 Distributions

```
# R allows to sample [r], compute density/probability mass [d],
# compute distribution function [p] and compute quantiles [q] for several
# continuous and discrete distributions. The format employed is [rdpq]name,
# where name stands for:
# - norm -> Normal
# - unif -> Uniform
# - exp -> Exponential
# - t -> Student's t
# - f -> Snedecor's F (Fisher)
# - chisq -> Chi squared
# - pois -> Poisson
# - binom -> Binomial
# More distributions: ?Distributions


# Sampling from a Normal - 100 random points from a N(0, 1)
rnorm(n = 10, mean = 0, sd = 1)
#ans>  [1] -0.02861 -0.35848  0.89207  0.00277  0.21199 -1.73837 -0.32862
#ans>  [8] -1.45277  0.32805  0.30720

# If you want to have always the same result, set the seed of the random number
# generator
set.seed(45678)
rnorm(n = 10, mean = 0, sd = 1)
#ans>  [1]  1.440 -0.720  0.671 -0.422  0.378 -1.667 -0.508  0.443 -1.799 -0.618

# Plotting the density of a N(0, 1) - the Gauss bell
x <- seq(-4, 4, l = 100)
y <- dnorm(x = x, mean = 0, sd = 1)
plot(x, y, type = "l")
```

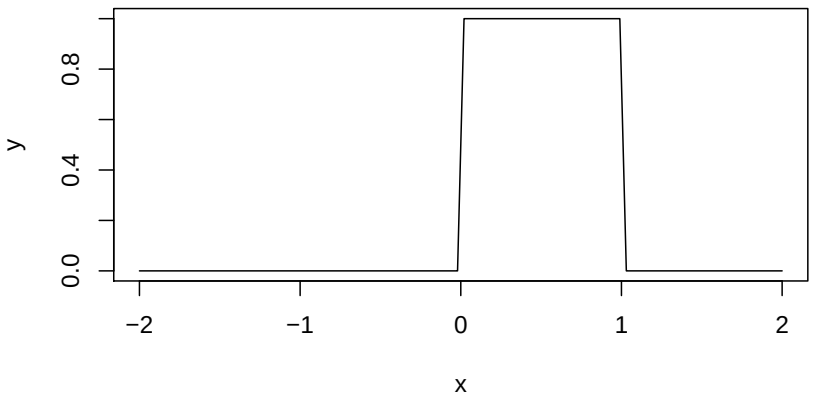



```
# Plotting the distribution function of a N(0, 1)
x <- seq(-4, 4, l = 100)
y <- pnorm(q = x, mean = 0, sd = 1)
```

```r
plot(x, y, type = "l", lwd = 3, main="The distribution function of a N(0, 1)")
```




```r
# Computing the 95% quantile for a N(0, 1)
qnorm(p = 0.95, mean = 0, sd = 1)
#ans> [1] 1.64

# All distributions have the same syntax: rname(n,...), dname(x,...), dname(p,...)
# and qname(p,...), but the parameters in ... change. Look them in ?Distributions
# For example, here is que same for the uniform distribution

# Sampling from a U(0, 1)
set.seed(45678)
runif(n = 10, min = 0, max = 1)
#ans>  [1] 0.9251 0.3340 0.2359 0.3366 0.7489 0.9327 0.3365 0.2246 0.6474 0.0808

# Plotting the density of a U(0, 1)
x <- seq(-2, 2, l = 100)
y <- dunif(x = x, min = 0, max = 1)
plot(x, y, type = "l")
```




```
# Computing the 95% quantile for a U(0, 1)
qunif(p = 0.95, min = 0, max = 1)
#ans> [1] 0.95
```

Do the following:

- Compute the 90%, 95% and 99% quantiles of a F distribution with `df1 = 1` and `df2 = 5`. (Answer: `c(4.060420, 6.607891, 16.258177)`)
- Sample 100 points from a Poisson with `lambda = 5`.
- Plot the density of a t distribution with `df = 1` (use a sequence spanning from `-4` to `4`). Add lines of different colors with the densities for `df = 5`, `df = 10`, `df = 50` and `df = 100`.

1.8.6 Working directory

Your *working directory* is the folder on your computer in which you are currently working. When you ask R to open a certain file, it will look in the working directory for this file, and when you tell R to save a data file or figure, it will save it in the working directory.

To set your working directory within RStudio you can go to `Tools / Set working directory`, or use the command `setwd()`, we put the complete path of the directory between the brackets, do not forget to put the path into quotation marks `""`.

To know the actual working directory we use `getwd()`.

1.8.7 Loading Data

The `read.table()` function is one of the primary ways to import a data set into `R`. The help file `?read.table()` contains details about how to use this function. We can use the function `write.table()` to export data.

Next we will show how to load the data set `Auto.data` (Download it from ▦ here).

```
Auto=read.table("Auto.data",header=T,na.strings ="?")
# For this file we needed to tell R that the first row is the
# names of the variables.
```

```
# na.strings tells R that any time it sees a particular character
# or set of characters (such as a question mark), it should be
# treated as a missing element of the data matrix.
```

- If the file is of csv format, we use `read.csv`.
- Always try to look to the file before importing it to `R` (Open it in a text editor. See for example if the first row containes the variables names, if the columns are separated by `,` or `;` or ..
- For text editors, I suggest `Sublime Text` or `Atom`.

```
dim(Auto) # To see the dimensions of the data set
#ans> [1] 397   9
nrow(Auto) # To see the number of rows
#ans> [1] 397
ncol(Auto) # To see the number of columns
#ans> [1] 9
Auto[1:4,] # The first 4 rows of the data set
#ans>   mpg cylinders displacement horsepower weight acceleration year origin
#ans> 1  18         8          307        130   3504         12.0   70      1
#ans> 2  15         8          350        165   3693         11.5   70      1
#ans> 3  18         8          318        150   3436         11.0   70      1
#ans> 4  16         8          304        150   3433         12.0   70      1
#ans>                        name
#ans> 1 chevrolet chevelle malibu
#ans> 2         buick skylark 320
#ans> 3        plymouth satellite
#ans> 4             amc rebel sst
```

```
# Once the data are loaded correctly, we can use names()
# to check the variable names.
names(Auto)
#ans> [1] "mpg"          "cylinders"    "displacement" "horsepower"
#ans> [5] "weight"       "acceleration" "year"         "origin"
#ans> [9] "name"
```

Take a look at this (very) short introduction to R. It can be useful.

1.9 Regression

1.9.1 The `lm` function

We are going to employ the `EU` dataset. The `EU` dataset contains 28 rows with the member states of the European Union (Country), the number of seats assigned under different years (Seats2011, Seats2014), the Cambridge Compromise apportionment (CamCom2011), and the countries population (Population2010,Population2013). Click here to download the `EU` dataset.

```
# Load the dataset, when we load an .RData using load()
# function we do not attribute it to a name like we did
# when we used read.table() or when we use read.csv()
```

```
load("EU.RData")
```

There is two ways to tell R where is the file you want to load/use/import or where to save a file when you write/export/save :

1. write the complete path of the files.
2. set a working directory and put the files in it.

```
# lm (for linear model) has the syntax:
# lm(formula = response ~ predictor, data = data)
# The response is the y in the model. The predictor is x.
# For example (after loading the EU dataset)
mod <- lm(formula = Seats2011 ~ Population2010, data = EU)

# We have saved the linear model into mod, which now contains all the output of lm
# You can see it by typing
mod
#ans>
#ans> Call:
#ans> lm(formula = Seats2011 ~ Population2010, data = EU)
#ans>
#ans> Coefficients:
#ans>    (Intercept)  Population2010
#ans>       7.91e+00        1.08e-06

# mod is indeed a list of objects whose names are
names(mod)
#ans>  [1] "coefficients"  "residuals"     "effects"       "rank"
#ans>  [5] "fitted.values" "assign"        "qr"            "df.residual"
#ans>  [9] "na.action"     "xlevels"       "call"          "terms"
#ans> [13] "model"

# We can access these elements by $
# For example
mod$coefficients
#ans>    (Intercept) Population2010
#ans>       7.91e+00       1.08e-06

# The residuals
mod$residuals
#ans>        Germany         France United Kingdom          Italy          Spain
#ans>         2.8675        -3.7031        -1.7847         0.0139        -3.5084
#ans>         Poland        Romania    Netherlands         Greece        Belgium
#ans>         1.9272         1.9434         0.2142         1.8977         2.3994
#ans>       Portugal Czech Republic        Hungary         Sweden        Austria
#ans>         2.6175         2.7587         3.2898         2.0163         2.0575
#ans>       Bulgaria        Denmark       Slovakia        Finland        Ireland
#ans>         1.9328        -0.8790        -0.7606        -0.6813        -0.7284
#ans>      Lithuania         Latvia       Slovenia        Estonia         Cyprus
#ans>         0.4998        -1.3347        -2.1175        -3.3552        -2.7761
#ans>     Luxembourg          Malta
#ans>        -2.4514        -2.3553
```

```
# The fitted values
mod$fitted.values
#ans>        Germany         France United Kingdom          Italy          Spain
#ans>          96.13          77.70          74.78          72.99          57.51
#ans>         Poland        Romania    Netherlands         Greece        Belgium
#ans>          49.07          31.06          25.79          20.10          19.60
#ans>       Portugal Czech Republic        Hungary         Sweden        Austria
#ans>          19.38          19.24          18.71          17.98          16.94
#ans>       Bulgaria        Denmark       Slovakia        Finland        Ireland
#ans>          16.07          13.88          13.76          13.68          12.73
#ans>      Lithuania         Latvia       Slovenia        Estonia         Cyprus
#ans>          11.50          10.33          10.12           9.36           8.78
#ans>     Luxembourg          Malta
#ans>           8.45           8.36

# Summary of the model
sumMod <- summary(mod)
sumMod
#ans>
#ans> Call:
#ans> lm(formula = Seats2011 ~ Population2010, data = EU)
#ans>
#ans> Residuals:
#ans>    Min     1Q Median     3Q    Max
#ans> -3.703 -1.951  0.014  1.980  3.290
#ans>
#ans> Coefficients:
#ans>                Estimate Std. Error t value Pr(>|t|)
#ans> (Intercept)    7.91e+00   5.66e-01    14.0  2.6e-13 ***
#ans> Population2010 1.08e-06   1.92e-08    56.3  < 2e-16 ***
#ans> ---
#ans> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#ans>
#ans> Residual standard error: 2.29 on 25 degrees of freedom
#ans>   (1 observation deleted due to missingness)
#ans> Multiple R-squared:  0.992,   Adjusted R-squared:  0.992
#ans> F-statistic: 3.17e+03 on 1 and 25 DF,  p-value: <2e-16
```

The following table contains a handy cheat sheet of equivalences between R code and some of the statistical concepts associated to linear regression.

R	Statistical concept
`x`	Predictor $X_1, \ldots, X_n$
`y`	Response $Y_1, \ldots, Y_n$
`data <- data.frame(x = x, y = y)`	Sample $(X_1, Y_1), \ldots, (X_n, Y_n)$
`model <- lm(y ~ x, data = data)`	Fitted linear model
`model$coefficients`	Fitted coefficients $\hat{\beta}_0, \hat{\beta}_1$
`model$residuals`	Fitted residuals $\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_n$
`model$fitted.values`	Fitted values $\hat{Y}_1, \ldots, \hat{Y}_n$
`model$df.residual`	Degrees of freedom $n - 2$
`summaryModel <- summary(model)`	Summary of the fitted linear model
`summaryModel$sigma`	Fitted residual standard deviation $\hat{\sigma}$
`summaryModel$r.squared`	Coefficient of determination R^2

R	Statistical concept
`summaryModel$fstatistic`	F-test
`anova(model)`	ANOVA table

Do the following:

- Download The 'EU' dataset from here as an `.RData` file and load it using the function `load`.
- Compute the regression of `CamCom2011` into `Population2010`. Save that model as the variable `myModel`.
- Access the objects `residuals` and `coefficients` of `myModel`.
- Compute the summary of `myModel` and store it as the variable `summaryMyModel`.
- Access the object `sigma` of `myModel`.

1.9.2 Predicting House Value: Boston dataset

We are going to use a dataset called Boston which is part of the `MASS` package. It recordes the median value of houses for 506 neighborhoods around Boston. Our task is to predict the median house value (`medv`) using only one predictor (`lstat`: percent of households with low socioeconomic status).

```
# First, install the MASS package using the command: install.packages("MASS")

# load MASS package
library(MASS)

# Check the dimensions of the Boston dataset
dim(Boston)
#ans> [1] 506  14
```

STEP 1: Split the dataset

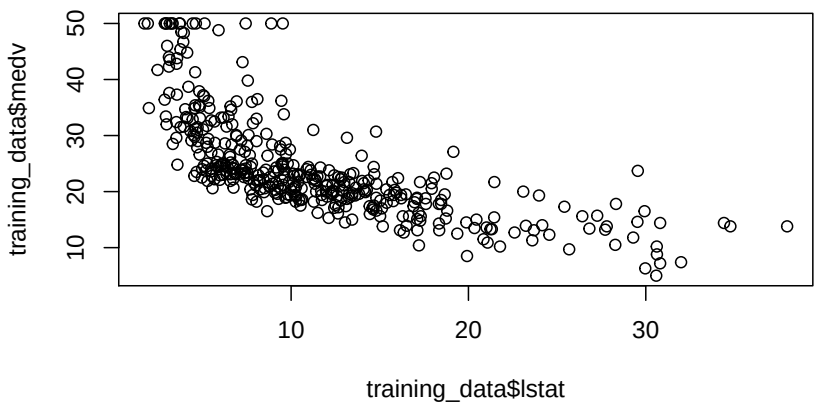
```
# Split the data by using the first 400 observations as the training
# data and the remaining as the testing data
train = 1:400
test = -train

# Speficy that we are going to use only two variables (lstat and medv)
variables = which(names(Boston) ==c("lstat", "medv"))
training_data = Boston[train, variables]
testing_data = Boston[test, variables]

# Check the dimensions of the new dataset
dim(training_data)
#ans> [1] 400   2
```

STEP 2: Check for Linearity

In order to perfom linear regression in `R`, we will use the function `lm()`to fit a simple linear regression with `medv` as the response (dependent variable) and `lstat` as the predictor or independent variable, and then save it in `model`.

But before we run our model, let's visually check if the relationship between x and y is linear.

```
# Scatterplot of lstat vs. medv
plot(training_data$lstat, training_data$medv)
```

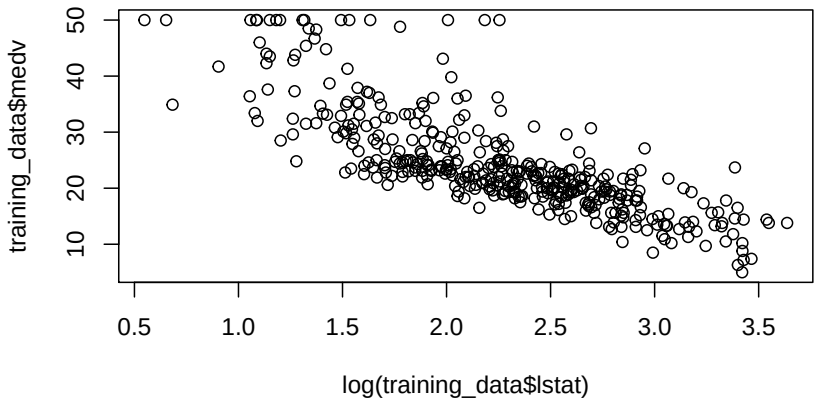


On the figure above, modify the following:

- Figure title.
- Axis titles.
- Shape of observations and their colors.
- Sizes of the chosen shape.

According to the plot, we see that the relationship is not linear. Let's try a transformation of our explanatory variable `lstat`.

```
# Scatterplot of log(lstat) vs. medv
plot(log(training_data$lstat), training_data$medv)
```



Look at the plot, it is more linear, so we can proceed and perform `lm()`:

STEP 3: Run the linear regression model

```
model = lm(medv ~ log(lstat), data = training_data)
model
#ans>
#ans> Call:
#ans> lm(formula = medv ~ log(lstat), data = training_data)
#ans>
#ans> Coefficients:
#ans> (Intercept)   log(lstat)
#ans>        51.8        -12.2
```

Notice that basic information when we print `model`. This only give us the slope (-12.2) and the intercept (51.8) of the linear model. Note that here we are looking at `log(lstat)` and not `lstat` anymore. So for every one unit increase in `lstat`, the median value of the house will decrease by $e^{12.2}$. For more detailed information, we can use the `summary()` function:

```
summary(model)
#ans>
#ans> Call:
#ans> lm(formula = medv ~ log(lstat), data = training_data)
#ans>
#ans> Residuals:
#ans>     Min      1Q  Median      3Q     Max
#ans> -11.385  -3.908  -0.779   2.245  25.728
#ans>
#ans> Coefficients:
#ans>             Estimate Std. Error t value Pr(>|t|)
#ans> (Intercept)   51.783      1.097    47.2   <2e-16 ***
#ans> log(lstat)   -12.203      0.472   -25.9   <2e-16 ***
#ans> ---
#ans> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#ans>
#ans> Residual standard error: 5.6 on 398 degrees of freedom
#ans> Multiple R-squared:  0.627,   Adjusted R-squared:  0.626
#ans> F-statistic:  669 on 1 and 398 DF,  p-value: <2e-16
```

Now, we have access to p-values and standard errors for the coefficients, as well as the R^2.

- The output states that the slope is statistically significant and different from 0 and with a t-value$=-25.9$ (p-value < 0.05), which means that there is a significant relationship between the percentage of households with low socioeconomic income and the median house value.
- This relationship is negative. That is as the percantage of household with low socioeconomic income increases, the median house value decreases.
- Looking at R^2, we can deduce that 62.7% of the model variation is being explained by the predictor `log(lstat)`. This is probably low, but indeed it would increase if we had more independent (explanatory) variables. We can use the `names()` function to see what other pieces of information are stored in our linear model (`model`).

```
names(model)
#ans>  [1] "coefficients"  "residuals"     "effects"       "rank"
#ans>  [5] "fitted.values" "assign"        "qr"            "df.residual"
#ans>  [9] "xlevels"       "call"          "terms"         "model"
```

```
model$coefficients
#ans> (Intercept)   log(lstat)
#ans>        51.8        -12.2
```

To obtain the confidence intervel for the linear model (`model`), we can use the `confint()` function:

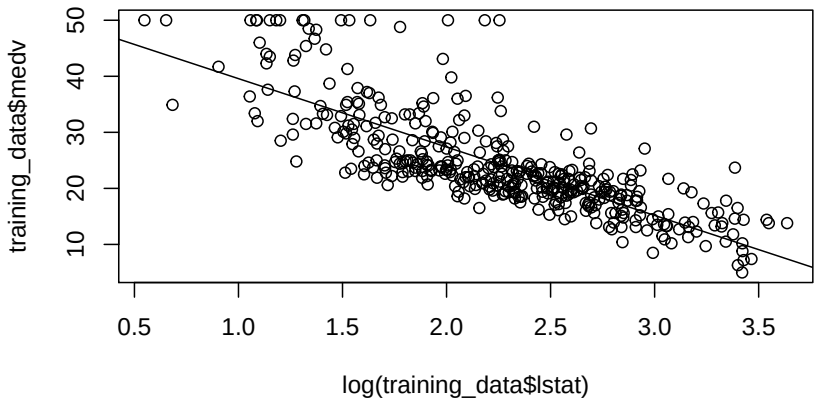
```
confint(model, level = 0.95)
#ans>              2.5 % 97.5 %
#ans> (Intercept)  49.6   53.9
#ans> log(lstat)  -13.1  -11.3
```

So, a 95% confidence interval for the slope of `log(lstat)` is $(-13.13, -11.28)$. Notice that this confidence interval gives us the same result as the hypothesis test performed earlier, by stating that we are 95% confident that the slope of `lstat` is not zero (in fact it is less than zero, which means that the relationship is negative.)

STEP 4: Plot the regression model

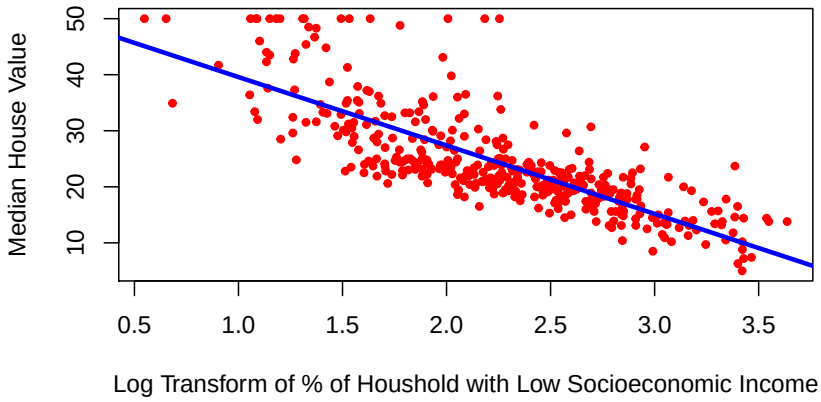
Now, let's plot our regression line on top of our data.

```
# Scatterplot of lstat vs. medv
plot(log(training_data$lstat), training_data$medv)

# Add the regression line to the existing scatterplot
abline(model)
```



Let's play with the look of the plot, and makes it perttier!

```
# Scatterplot of lstat vs. medv
plot(log(training_data$lstat), training_data$medv,
xlab = "Log Transform of % of Houshold with Low Socioeconomic Income",
ylab = "Median House Value",
col = "red",
pch = 20)

# Make the line color blue, and the line's width =3 (play with the width!)
abline(model, col = "blue", lwd =3)
```



STEP 5: Assess the model

Final thing we will do is to predict using our fitted model. We can use the `predict()` function for this purpose:

```
# Predict what is the median value of the house with lstat= 5%
predict(model, data.frame(lstat = c(5)))
#ans>    1
#ans> 32.1
```

```
# Predict what is the median values of houses with lstat= 5%, 10%, and 15%
predict(model, data.frame(lstat = c(5,10,15), interval = "prediction"))
#ans>    1    2    3
#ans> 32.1 23.7 18.7
```

Now let's assess our model, by computing th mean squared error (MSE). To assess the model we created, then we will be using the test data!

```
# Save the testing median values for houses (testing y) in y
y = testing_data$medv

# Compute the predicted value for this y (y hat)
y_hat = predict(model, data.frame(lstat = testing_data$lstat))

# Now we have both y and y_hat for our testing data.
# let's find the mean square error
error = y-y_hat
error_squared = error^2
MSE = mean(error_squared)
MSE
#ans> [1] 17.7
```

Chapter 2

Multiple Linear Regression

Simple linear regression is a useful approach for predicting a response on the basis of a single predictor variable. However, in practice we often have more than one predictor. In the previous chapter, we took for example the prediction of housing prices considering we had the size of each house. We had a single feature X, the size of the house. But now imagine if we had not only the size of the house as a feature but we also knew the number of bedrooms, the number of flours and the age of the house in years. It seems like this would give us a lot more information with which to predict the price.

2.1 The Model

In general, suppose that we have p distinct predictors. Then the multiple linear regression model takes the form

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

where X_j represents the jth predictor and β_j quantifies the association between that variable and the response. We interpret β_j as the average effect on Y of a one unit increase in X_j, *holding all other predictors fixed*.

In matrix terms, supposing we have n observations and p variables, we need to define the following matrices:

$$\mathbf{Y}_{n\times 1} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} \qquad \mathbf{X}_{n\times(p+1)} = \begin{pmatrix} 1 & X_{11} & X_{12} & \dots & X_{1p} \\ 1 & X_{21} & X_{22} & \dots & X_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & X_{n2} & \dots & X_{np} \end{pmatrix} \tag{2.1}$$

$$\beta_{(p+1)\times 1} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \qquad \epsilon_{n\times 1} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix} \tag{2.2}$$

In matrix terms, the general linear regression model is

$$\mathbf{Y}_{n\times 1} = \mathbf{X}_{n\times(p+1)}\beta_{(p+1)\times 1} + \epsilon_{n\times 1}$$

where,

- $\mathbf{Y}$ is a vector of responses.
- β is a vector of parameters.
- $\mathbf{X}$ is a matrix of constants.
- ϵ is a vector of independent *normal* (Gaussian) random variables.

2.2 Estimating the Regression Coefficients

As was the case in the simple linear regression setting, the regression coefficients $\beta_0, \beta_1, \ldots, \beta_p$ are unknown, and must be estimated. Given estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$, we can make predictions using the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots, \hat{\beta}_p x_p$$

We choose $\beta_0, \beta_1, \ldots, \beta_p$ to minimize the residual sum of squares

$$\begin{aligned} RSS &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_1 - \hat{\beta}_0 - \hat{\beta}_1 \hat{x}_{i1} - \hat{\beta}_2 \hat{x}_{i2} - \ldots - \hat{\beta}_p \hat{x}_{ip})^2 \end{aligned}$$

The values $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ that minimize the RSS are the multiple least squares regression coefficient estimates, they are calculated using this formula (in matrix terms):

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

To obtain $\hat{\beta}$, we can write the residual sum of squares as

$$RSS = (\mathbf{Y} - \mathbf{X}\beta)^T (\mathbf{Y} - \mathbf{X}\beta)$$

This is a quadratic function in the $p + 1$ parameters. Differentiating with respect to β we obtain

$$\begin{aligned} \frac{\partial RSS}{\partial \beta} &= -2\mathbf{X}^T (\mathbf{Y} - \mathbf{X}\beta) \\ \frac{\partial^2 RSS}{\partial \beta \partial \beta^T} &= 2\mathbf{X}^T \mathbf{X}. \end{aligned}$$

Assuming (for the moment) that $\mathbf{X}$ has full column rank, and hence $\mathbf{X}^T \mathbf{X}$ is positive definite[1], we set the first derivative to zero

$$\mathbf{X}^T (\mathbf{Y} - \mathbf{X}\beta) = 0$$

to obtain the unique solution

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

Note 1:

[1] Important to be sure that $\hat{\beta}$ is minimising RSS.

It is a remarkable property of matrix algebra that the results for the general linear regression model in matrix notation appear exactly as those for the simple linear regression model. Only the degrees of freedom and other constants related to the number of X variables and the dimensions of some matrices are different. Which means there are some similarities between $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ and $\hat{\beta}_1 = (s_x^2)^{-1}s_{xy}$ from the simple linear model: both are related to the covariance between $\mathbf{X}$ and $\mathbf{Y}$ weighted by the variance of $\mathbf{X}$.

Note 2:

If $\mathbf{X}^T\mathbf{X}$ is noninvertible, the common causes might be having:

- Redundant features, where two features are very closely related (i.e. they are linearly dependent)
- Too many features (e.g. $p \geq n$). In this case, we delete some features or we use "regularization" (to be, maybe, explained in a later lesson).

2.3 Some important questions

When we perform multiple linear regression, we usually are interested in answering a few important questions.

1. Is at least one of the predictors $X_1, X_2, \ldots, X_p$ useful in predicting the response?
2. Do all the predictors help to explain Y, or is only a subset of the predictors useful?
3. How well does the model fit the data?
4. Given a set of predictor values, what response value should we predict, and how accurate is our prediction?

Relationship Between the Response and Predictors?

F-Statistic

Recall that in the simple linear regression setting, in order to determine whether there is a relationship between the response and the predictor we can simply check whether $\beta_1 = 0$. In the multiple regression setting with p predictors, we need to ask whether all of the regression coefficients are zero, i.e. whether $\beta_1 = \beta_2 = \ldots = \beta_p = 0$. As in the simple linear regression setting, we use a hypothesis test to answer this question. We test the null hypothesis,

$$H_0 : \beta_1 = \beta_2 = \ldots = \beta_p = 0$$

versus the alternative hypothesis

$$H_1 : \text{at least one } \beta_j \text{ is non-zero}$$

This hypothesis test is performed by computing the F-statistic (*Fisher*):

$$F = \frac{(\text{TSS} - \text{RSS})/p}{\text{RSS}/(n-p-1)} \sim F_{p,n-p-1}$$

where, as with simple linear regression, $\text{TSS} = \sum(y_i - \bar{y})^2$ and $\text{RSS} = \sum(y_i - \hat{y}_i)^2$.

Note that $F_{p,n-p-1}$ represents the *Fisher-Snedecor's F distribution* with p and $n-p-1$ degrees of freedom. If H_0 is true, then F is expected to be *small* since ESS[2] will be close to zero (little variation is explained by the regression model since $\hat{\boldsymbol{\beta}} \approx \mathbf{0}$).

So the question we ask here: *Is the whole regression explaining anything at all?* The answer comes from the F-test in the ANOVA (ANalysis Of VAriance) table. This is what we get in an ANOVA table:

Source	df	SS	MS	F	p-value
Factor (Explained)	p	ESS=SSR	MSR=ESS/(p)	F=MSR/MSE	p-value
Error (Unexplained)	$n-p-1$	RSS=SSE	MSE=RSS/($n-p-1$)		
Total	$n-1$	TSS=SST			

The ANOVA table has many pieces of information. What we care about is the F Ratio and the corresponding p-value. We compare the F Ratio with $F_{(p,n-p-1)}$ and a corresponding α value (error).

The "ANOVA table" is a broad concept in statistics, with different variants. Here we are only covering the basic ANOVA table from the relation SST = SSR + SSE. However, further sophistications are possible when SSR is decomposed into the variations contributed by *each* predictor. In particular, for multiple linear regression `R`'s `anova` implements a *sequential (type I) ANOVA table*, which is **not** the previous table!

The `anova` function in `R` takes a model as an input and returns the following *sequential* ANOVA table[3]:

	Degrees of freedom	Sum Squares	Mean Squares	F-value	p-value
Predictor 1	1	ESS_1	$\frac{\text{ESS}_1}{1}$	$\frac{\text{ESS}_1/1}{\text{RSS}/(n-p-1)}$	p_1
Predictor 2	1	ESS_2	$\frac{\text{ESS}_2}{1}$	$\frac{\text{ESS}_2/1}{\text{RSS}/(n-p-1)}$	p_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Predictor p	1	ESS_p	$\frac{\text{ESS}_p}{1}$	$\frac{\text{ESS}_p/1}{\text{RSS}/(n-p-1)}$	p_p
Residuals	$n-p-1$	RSS	$\frac{\text{RSS}}{n-p-1}$		

Here the ESS_j represents the explained sum of squares (same as regression sum of squares) associated to the inclusion of X_j in the model with predictors $X_1, \ldots, X_{j-1}$, this is:

$$\text{ESS}_j = \text{ESS}(X_1, \ldots, X_j) - \text{ESS}(X_1, \ldots, X_{j-1}).$$

The p-values $p_1, \ldots, p_p$ correspond to the testing of the hypotheses

$$H_0 : \beta_j = 0 \quad \text{vs.} \quad H_1 : \beta_j \neq 0,$$

carried out *inside the linear model* $Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_j X_j + \varepsilon$. This is like the t-test for β_j for the model with predictors $X_1, \ldots, X_j$. Recall that there is no F-test in this version of the ANOVA table.

p-values

The p-values provide information about whether each individual predictor is related to the response, after adjusting for the other predictors. Let's look at the following table we obtain in general using a statistical software for example

[2]Recal that ESS is the explained sum of squares, ESS = TSS - RSS.
[3]More complex – included here just for clarification of the `anova`'s output.

	Coefficient	Std. error	t-statistic	p-value
Constant	2.939	0.3119	9.42	<0.0001
X_1	0.046	0.0014	32.81	<0.0001
X_2	0.189	0.0086	21.89	<0.0001
X_3	-0.001	0.0059	-0.18	0.8599

In this table we have the following model

$$Y = 2.939 + 0.046X_1 + 0.189X_2 - 0.001X_3$$

Note that for each individual predictor a t-statistic and a p-value were reported. These p-values indicate that X_1 and X_2 are related to Y, but that there is no evidence that X_3 is associated with Y, in the presence of these two.

Deciding on Important Variables

The most direct approach is called *all subsets* or *best subsets* regression: we compute the least squares fit for all possible subsets and then choose between them based on some criterion that balances training error with model size.

However we often can't examine all possible models, since they are 2^p of them; for example when $p = 40$ there are over a billion models! Instead we need an automated approach that searches through a subset of them. Here are two commonly use approaches:

Forward selection:

- Begin with the *null model* — a model that contains an intercept (constant) but no predictors.
- Fit p simple linear regressions and add to the null model the variable that results in the lowest RSS.
- Add to that model the variable that results in the lowest RSS amongst all two-variable models.
- Continue until some stopping rule is satisfied, for example when all remaining variables have a p-value above some threshold.

Backward selection:

- Start with all variables in the model.
- Remove the variable with the largest p-value — that is, the variable that is the least statistically significant.
- The new ($p1$)-variable model is fit, and the variable with the largest p-value is removed.
- Continue until a stopping rule is reached. For instance, we may stop when all remaining variables have a significant p-value defined by some significance threshold.

There are more systematic criteria for choosing an "optimal" member in the path of models produced by forward or backward stepwise selection. These include *Mallow's* C_p , *Akaike information criterion (AIC)*, *Bayesian information criterion (BIC)*, *adjusted* R^2 and *Cross-validation (CV)*.

Model Fit

Two of the most common numerical measures of model fit are the RSE and R^2, the fraction of variance explained. These quantities are computed and interpreted in the same fashion as for simple linear regression. Recall that in simple regression, R^2 is the square of the correlation of the response and the variable. In multiple linear regression, it turns out that it equals $Cor(Y, \hat{Y})^2$, the square of the correlation between the response and the fitted linear model; in fact one property of the fitted linear model is that it maximizes this correlation among all possible linear models. An R^2 value close to 1 indicates that the model explains a large portion of the variance in the response variable.

In general RSE is defined as

$$\text{RSE} = \sqrt{\frac{1}{n-p-1}\text{RSS}}$$

2.3.1 Other Considerations in Regression Model

Qualitative Predictors

- If we have a categorial (qualitative) variable (feature), how do we fit into a regression equation?
- For example, if X_1 is the gender (male or female).
- We can code, for example, male = 0 and female = 1.
- Suppose X_2 is a quantitative variable, the regression equation becomes:

$$Y_i \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 = \begin{cases} \beta_0 + \beta_2 X_2 & \text{if male} \\ \beta_0 + \beta_1 X_1 + \beta_2 X_2 & \text{if female} \end{cases}$$

- Another possible coding scheme is to let male = -1 and female = 1, the regression equation is then:

$$Y_i \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 = \begin{cases} \beta_0 - \beta_1 X_1 + \beta_2 X_2 & \text{if male} \\ \beta_0 + \beta_1 X_1 + \beta_2 X_2 & \text{if female} \end{cases}$$

Interaction Terms

- When the effect on Y of increasing X_1 depends on another X_2.
- We may in this case try the model

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$$

- $X_1 X_2$ is the Interaction term.

2.4 How to select the best performing model

After trying different linear models, you need to make a choice which model you want to use. More specifically, the questions that one can ask: "How to determine which model suits best to my data? Do I just look at the R square, SSE, etc.?" and "As the interpretation of that model (quadratic, root, etc.) will be very different, won't it be an issue?"

The second question can be answered easily. First, find a model that best suits to your data and then interpret its results. It is good if you have ideas how your data might be explained. However, interpret the best model, only. Now we will address the first question. Note that there are multiple ways to select a best model. In addition, this approach only applies to univariate models (simple models) whith just one input variable.

Use the following interactive application and play around with different datasets and models. Notice how parameters change and become more confident with assessing simple linear models.

Use the Adjusted R^2_{adj} for univariate models

If you only use one input variable, the adjusted R^2_{adj} value gives you a good indication of how well your model performs. It illustrates how much variation is explained by your model.

In contrast to the simple R^2[4], the adjusted adjusted R^2_{adj}[5] takes the number of input factors into account. It penalizes too many input factors and favors parsimonious models.

The adjusted R^2_{adj} is sensitive to the amount of noise in the data. As such, only compare this indicator of models for the same dataset than comparing it across different datasets.

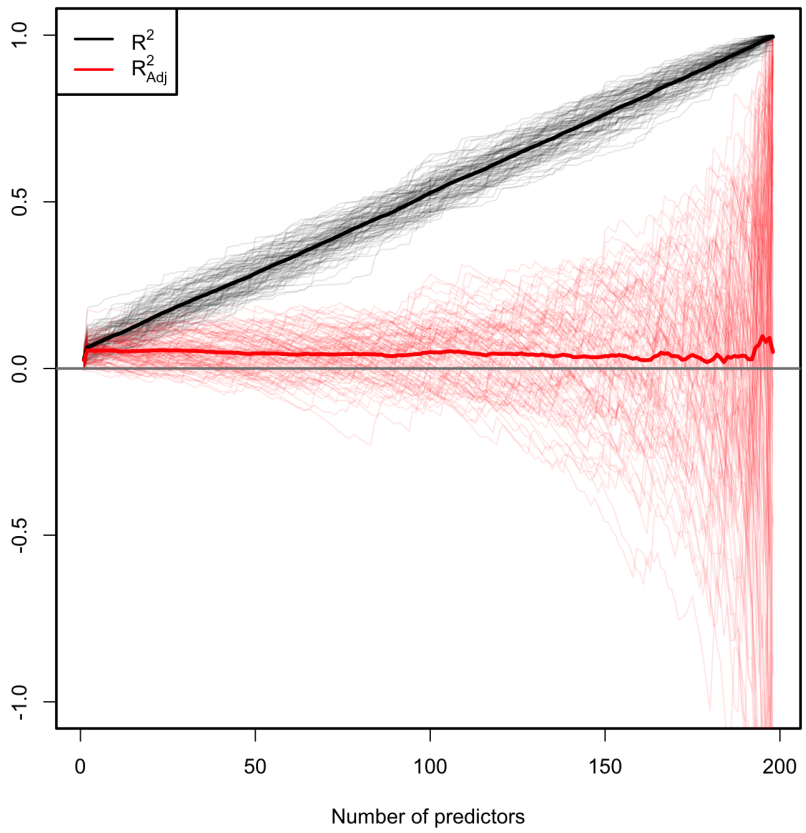


Figure 2.1: Comparison of R^2 and R^2_{adj} for $n = 200$ and p ranging from 1 to 198. $M = 100$ datasets were simulated with **only the first two** predictors being significant. The thicker curves are the mean of each color's curves.

Figure 2.1 contains the results of an experiment where 100 datasets were simulated with **only the first two** predictors being significant. As you can see R^2 increases linearly with the number of predictors considered, although only the first two ones were important! On the contrary, R^2_{adj} only increases in the first two variables and then is flat on average, but it has a huge variability when p approaches $n-2$. The experiment evidences that **R^2_{adj} is more adequate than the R^2 for evaluating the fit of a multiple linear regression**.

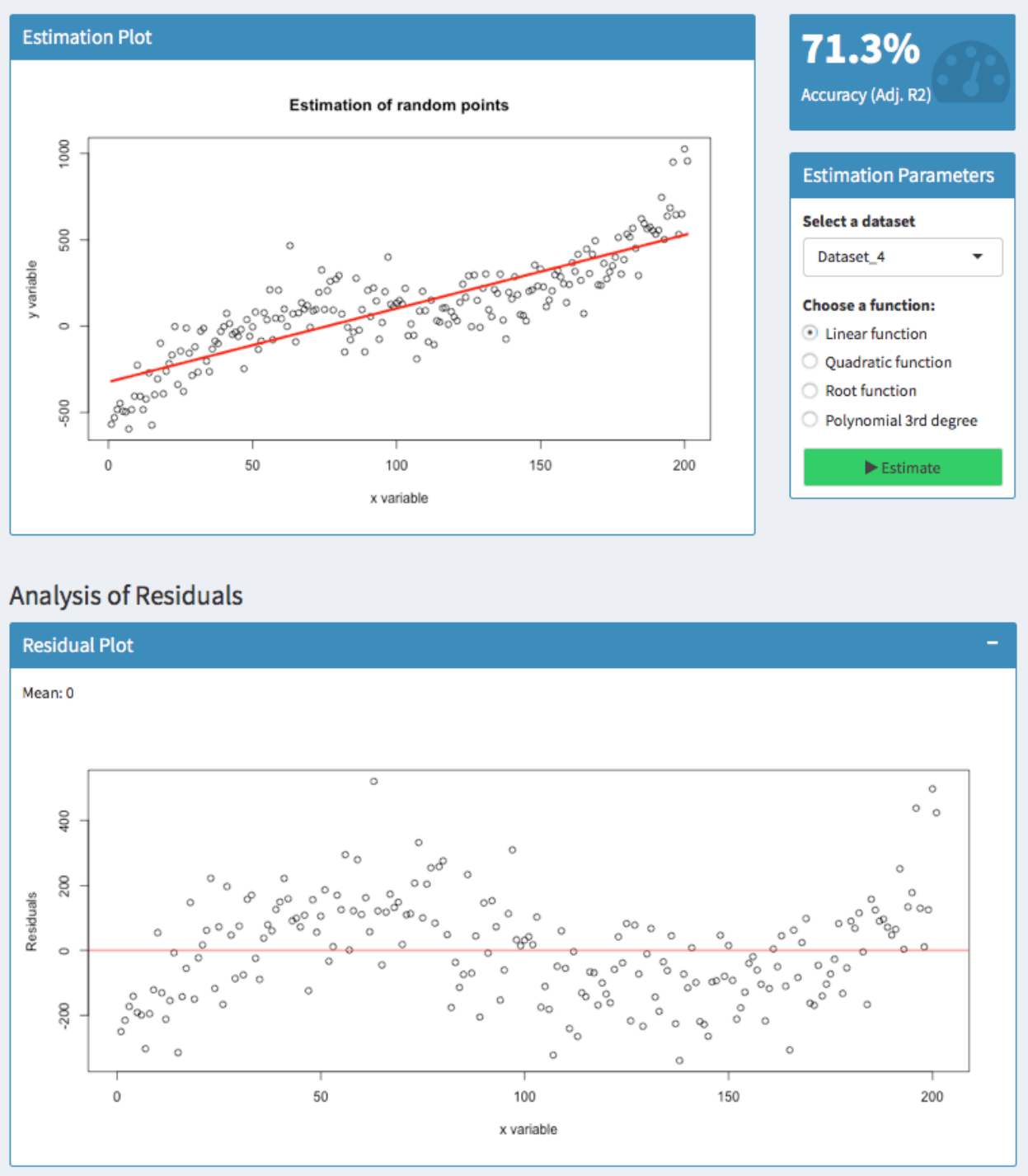
Have a look at the residuals or error terms

What is often ignored are error terms or so-called residuals. They often tell you more than what you might think. The residuals are the difference between your predicted values and the actual values. Their benefit is that they can show both the magnitude as well as the direction of the errors.

Let's have a look at an example:

[4]Recall that $R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$

[5]It is defined as $R^2_{adj} = 1 - \frac{\text{RSS}/(n-p-1)}{\text{TSS}/(n-1)} = 1 - \frac{\text{RSS}}{\text{TSS}} \times \frac{n-1}{n-p-1}$



Here, we try to predict a polynomial dataset with a linear function. Analyzing the residuals shows that there are areas where the model has an upward or downward bias.

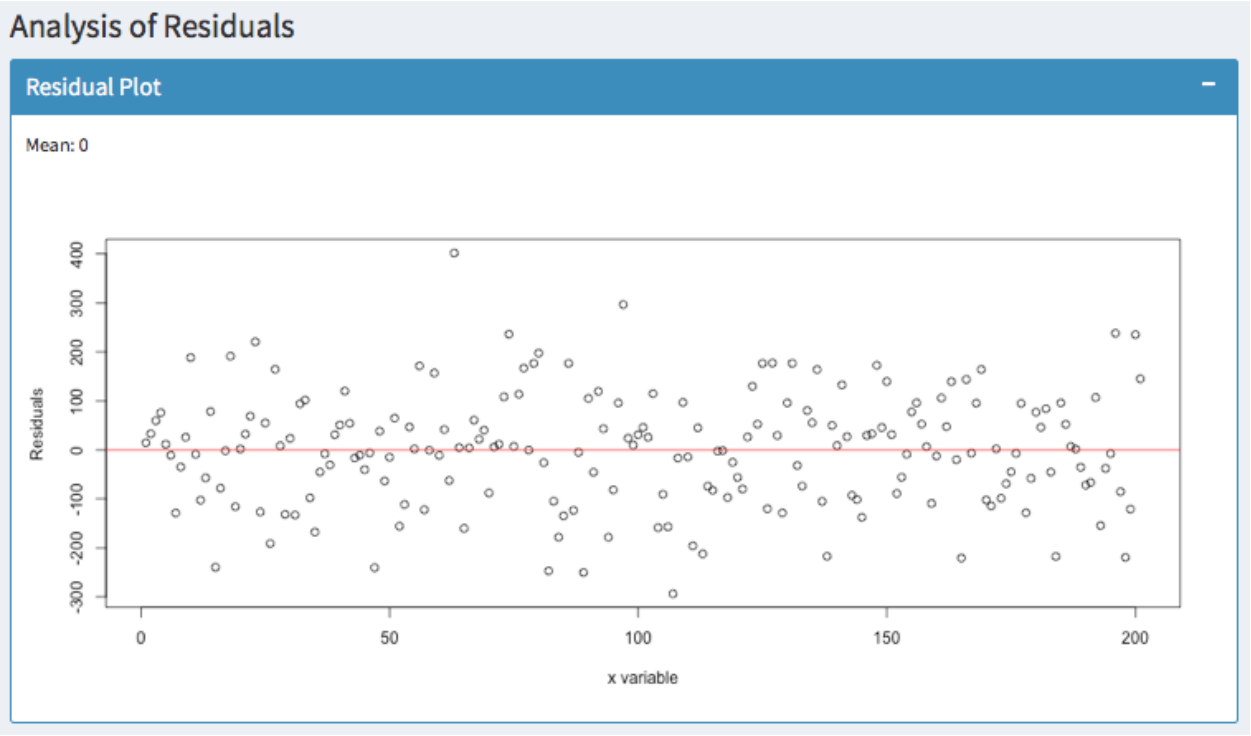
For `50 < x < 100`, the residuals are above zero. So in this area, the actual values have been higher than the predicted values—our model has a downward bias.

For `100 < x < 150`, however, the residuals are below zero. Thus, the actual values have been lower than the predicted values—the model has an upward bias.

It is always good to know, whether your model suggests too high or too low values. But you usually do not

want to have patterns like this.

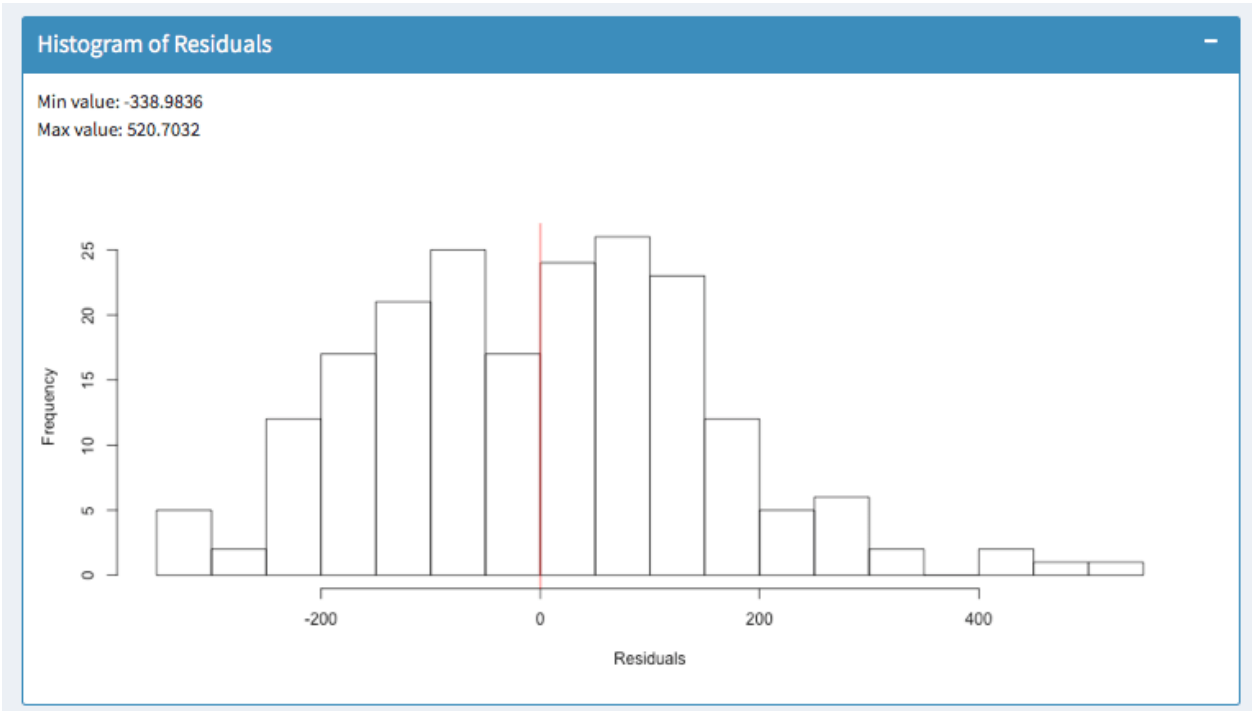
The residuals should be zero on average (as indicated by the mean) and they should be equally distributed. Predicting the same dataset with a polynomial function of `3 degrees` suggests a much better fit:

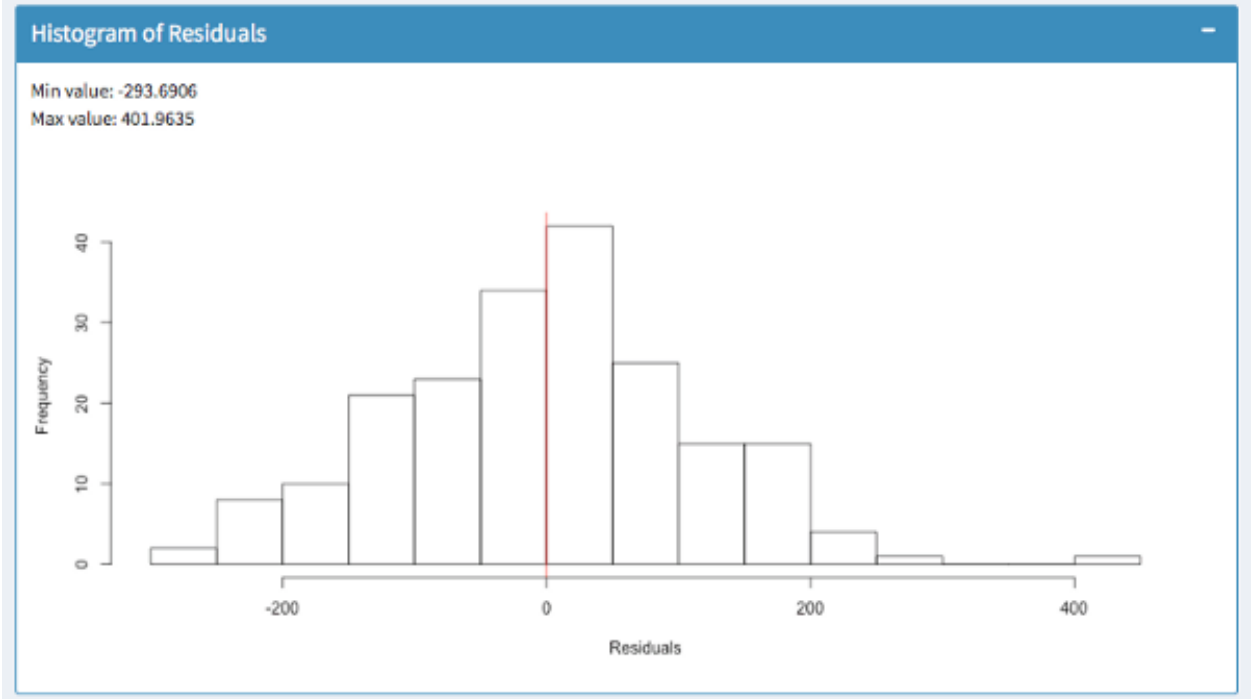


In addition, you can observe whether the variance of your errors increases. In statistics, this is called Heteroscedasticity. You can fix this easily with robust standard errors. Otherwise, your hypothesis tests are likely to be wrong.

Histogram of residuals

Finally, the histogram summarizes the magnitude of your error terms. It provides information about the bandwidth of errors and indicates how often which errors occurred.





The above screenshots show two models for the same dataset. In the first histogram, errors occur within a range of -338 and 520. In the second histogram, errors occur within -293 and 401. So the outliers are much lower. Furthermore, most errors in the model of the second histogram are closer to zero. So we would favor the second model.

PW 2

Multiple Linear Regression

In this practical work, we will continue the analysis of the Boston data set that we started last week (section 1.9.2). Recall that this dataset records the median value of houses for 506 neighborhoods around Boston. Our task is to predict the median house value (`medv`).

1. Load the Boston dataset from `MASS` package.

2. Split the dataset into traning set and testing set. (keep all the variables of the Boston data set)

3. Check if there is a linear relationship between the variables `medv` and `age`. (use `cor()` function).

4. Fit a model of housing prices in function of age and plot the observations and the regression line.

5. Train a regression model using both `lstat` and `age` as predictors of median house value. (Remember that we transformed `lstat`, use the same transformation here). What is the obtained model?

6. Print the summary of the obtained regression model.

7. Are the predictors significant ?

8. Is the model as a whole significant? Answer on this question must be detailed.

9. Train a new model using all the variables of the dataset. (We can use `.` as a short cut instead of writing down all the variables names)

10. When using all the variables as predictors, we didn't transform `lstat`. Re train the model using `log(lstat)` instead of `lstat`.

11. Did R^2 improve ?

12. To see if there is correlated variables print the correlation matrix using the `cor()` function (round the correlations with 2 digits).

13. Visualize the correlations using the `corrplot` package. To do so, install the `corrplot` package, load it, then use the function `corrplot.mixed()`. See this link for examples and to understand how to use it.

14. What is the correlation between `tax` and `rad`?

15. Run the model again without `tax`. What happens to the R^2 ? and for the F-statistic?

Of course R^2 should go a little lower because we deleted one of the variables. But check for the model significance (F-statistic) gets higher, which means the p-values gets lower and thus the model is more significant without `rad`.

16. Calculate the mean squared error (MSE) for the last model.

Anova

Next we will apply an analysis of variances (ANOVA) in order to test if there is a significant difference of means between two groups i and j (Consider group i is the suburbs bounding the river and j the suburbs which not). The hypotheses are

$$H_0 : \mu_i = \mu_j$$

$$H_1 : \mu_i \neq \mu_j$$

Where μ_i is the mean of `medv` in group i.

17. In the Boston data set there is a categorical variable `chas` which corresponds to Charles River (= 1 if a suburb bounds the river; 0 otherwise). Use command `str()` to see how this variable is present in the dataset. How many of the suburbs in this data set bound the Charles river?

18. Create Boxplots of the median value of houses with respect to the variable `chas`. Do we observe some difference between the median value of houses with respect to the neighborhood to Charles River?

19. Calculate μ_i and μ_j (in one line using the function `aggregate()`).

20. Apply an ANOVA test of `medv` whith respect to `chas` (use the function `aov()`). Print the result and the summary of it. what do you conclude ?

Qualitative predictors

⚠ **Before starting the next question, please read section 2.3.1 and Appendix C about using qualitative predictors in regression**.

We are going to use the categorical variable `chas` which corresponds to Charles River (= 1 if a suburb bounds the river; 0 otherwise). Using the `str()` command you can notice that this variable is not codified as a factor, but it has values 0 or 1, so it is already dummyfied.

21. Fit a new model where the predictors are the Charles River and the Crime Rate. Interpret the coefficients of this model and conclude if the presence of the river adds a valuable information for explaining the house price.

22. Is `chas` is significant as well in the presence of more predictors?

Interaction terms

As you saw in section 2.3.1 we may sometimes try models with intercation terms. Let's say we have two predictors X_1 and X_2, the way of adding these interactions in `lm` is through `:` and `*`. The operator `:` only adds the term X_1X_2 and `*` adds X_1, X_2, and X_1X_2.

23. Fit a model whith first order interaction term where predictors are `lstat` and `age`. Print its summary.

24. Fit a model with all the first order interaction terms.

Reporting

In `R` there is some packages to make it **easy to create reproducible web-based reports**. To do so, click on `File -> Knit document` or `File -> Compile report...` The output is a html report containing the results of your `R` codes. If your file is named `report.R`, your report is named `report.html`.

25. Compile a report based on your script.

- Make sure to have the latest version of `Rstudio`.
- If you have problems with compiling (problem in installing packages, etc..) close your `Rstudio` and reopen it with administrative tools and retry.
- Be ready to submit your report (your `.html` file) at the end of each class.
- You report must be named: `YouLastName_YourFirstName_WeekNumber.html`

Classification

Chapter 3

Logistic Regression

3.1 Introduction

In the previous chapters we discussed the linear regression model, which assumes that the response variable Y is quantitative. But in many situations, the response variable is instead qualitative (categorical). For example, eye color is qualitative, taking on values blue, brown, or green.

The process for predicting qualitative responses is known as ***classification***.

Given a feature vector X and a qualitative response Y taking values in the set $\mathcal{C}$, the classification task is to build a function $C(X)$ that takes as input the feature vector X and predicts its value for Y; i.e. $C(X) \in \mathcal{C}$. We are often more interested in estimating the probabilities that X belongs to each category in $\mathcal{C}$.

If c is a category ($c \in \mathcal{C}$), by the probability that X belongs to c we mean $p(X \in c) = \mathbb{P}(Y = c|X)$.

In the binomial or binary logistic regression, the outcome can have only two possible types of values (e.g. "Yes" or "No", "Success" or "Failure"). Multinomial logistic refers to cases where the outcome can have three or more possible types of values (e.g., "good" vs. "very good" vs. "best"). Generally outcome is coded as "0" and "1" in binary logistic regression.

3.2 Logistic Regression

Consider a data set where the response falls into one of two categories, Yes or No. Rather than modeling the response Y directly, logistic regression models the *probability* that Y belongs to a particular category.

3.2.1 The Logistic Model

Let us suppose the response has two categories and we use the generic 0/1 coding for the response. How should we model the relationship between $p(X) = \mathbb{P}(Y = 1|X)$ and X?

The simplest situation is when Y is *binary*: it can only take two values, codified for convenience as 1 (success) and 0 (failure).

More formally, a binary variable is known as a *Bernoulli variable*, which is the simplest non-trivial random variable. We say that $Y \sim \text{Ber}(p)$, $0 \leq p \leq 1$, if

$$Y = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } 1 - p, \end{cases}$$

or, equivalently, if $\mathbb{P}[Y=1]=p$ and $\mathbb{P}[Y=0]=1-p$, which can be written compactly as

$$\mathbb{P}[Y=y]=p^y(1-p)^{1-y}, \quad y=0,1.$$

Recall that a *binomial variable with size n and probability p*, $\text{Bi}(n,p)$, was obtained by adding n independent $\text{Ber}(p)$ (so $\text{Ber}(p)$ is the same as $\text{Bi}(1,p)$).

A Bernoulli variable Y is completely determined by p. **So its mean and variance**:

- $\mathbb{E}[Y]=p\times 1+(1-p)\times 0=p$
- $\mathbb{V}\text{ar}[Y]=p(1-p)$.

In particular, recall that $\mathbb{P}[Y=1]=\mathbb{E}[Y]=p$.

Assume then that Y is a binary/Bernoulli variable and that X are predictors associated to them (no particular assumptions on them). The purpose in *logistic regression* is to estimate

$$p(x)=\mathbb{P}[Y=1|X=x]=\mathbb{E}[Y|X=x],$$

this is, how the probability of $Y=1$ is changing according to particular values, denoted by x, of the random variables X.

Why not linear regression? A tempting possibility is to consider the model

$$p(x)=\beta_0+\beta_1 x.$$

However, such a model will run into problems inevitably: negative probabilities and probabilities larger than one ($p(x)<0$ for some values of X and $p(X)>1$ for others). To avoid this problem, the solution is to consider a function to encapsulate the value of $z=\beta_0+\beta_1 x$, in $\mathbb{R}$, and map it to $[0,1]$. There are several alternatives to do so, based on distribution functions $F:\mathbb{R}\longrightarrow[0,1]$ that deliver $y=F(z)\in[0,1]$. Many functions meet this description. In logistic regression, we use the *logistic function*,

$$p(X)=\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

- No matter what values β_0, β_1 or X take, $p(X)$ will have values between 0 and 1.
- The logistic function will always produce an *S-shaped* curve.
- The logistic *distribution* function is:

$$F(z)=\text{logistic}(z)=\frac{e^z}{1+e^z}=\frac{1}{1+e^{-z}}.$$

After a bit of manipulation of the previous equation, we find that

$$\frac{p(X)}{1-p(X)}=e^{\beta_0+\beta_1 X}$$

The quantity $p(X)/[1p(X)]$ is called the *odds*, and can take on any value between 0 and ∞.

By taking the logarithm of both sides of the equation, we arrive at

$$\log\left(\frac{p(X)}{1-p(X)}\right)=\beta_0+\beta_1 X$$

The left-hand side is called the *log-odds* or *logit*. We see that the logistic regression model has a logit that is linear in X.

3.2.2 Estimating the Regression Coefficients

We estimate β_0 and β_1 using the *Maximum Likelihood Estimation* method (MLE). The basic intuition behind using maximum likelihood to fit a logistic regression model is as follows: we seek estimates for β_0 and β_1 such that the predicted probability $\hat{p}(x_i)$ of the response for each individual, corresponds as closely as possible to the individual's observed response status (recall that the response Y is categorical). The *likelihood function* is

$$l(\beta_0, \beta_1) = \prod_{i=1}^{n} p(x_i)^{Y_i}(1 - p(x_i))^{1-Y_i}.$$

This likelihood is **the probability of the data based on the model**. It gives the probability of the observed zeros and ones in the data. The estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are chosen to *maximize* this likelihood function. The interpretation of the likelihood function is the following:

- $\prod_{i=1}^{n}$ appears because the sample elements are assumed to be independent and we are computing the probability of observing the whole sample $(x_1, y_1), \ldots, (x_n, y_n)$. This probability is equal to the product of the *probabilities of observing each* (x_i, y_i).
- $p(x_i)^{Y_i}(1 - p(x_i))^{1-Y_i}$ is the probability of observing (x_i, Y_i).

In the linear regression setting, the least squares approach is a special case of maximum likelihood.

We will not give mathematical details about the maximum likelihood and how to estimate the parameters. We will use R to fit the logistic regression models (using `glm` function).

Use the following application (also available here) to see how the log-likelihood changes with respect to the values for (β_0, β_1) in three data patterns. The logistic regression fit and its dependence on β_0 (horizontal displacement) and β_1 (steepness of the curve). Recall the effect of the sign of β_1 in the curve: if positive, the logistic curve has an s form; if negative, the form is a reflected s.

```
#ans> PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pl
```

Note that the **animation** will not be displayed the first time it is browsed (The reason is because it is hosted at `https` websites with auto-signed SSL certificates). **To see it**, click on the link above. You will get a warning from your browser saying that *"Your connection is not private"*. Click in *"Advanced"* and **allow an exception** in your browser. The next time the animation will show up correctly.

3.2.3 Prediction

Example

	Coefficient	Std. error	Z-statistic	p-value
Constant	-10.6513	0.3612	-29.5	<0.0001
X	0.0055	0.0002	24.9	<0.0001

In this example, $\hat{\beta}_0 = -10.6513$ and $\hat{\beta}_1 = 0.0055$. It produces the blue curve that separates that data in the

following figure,

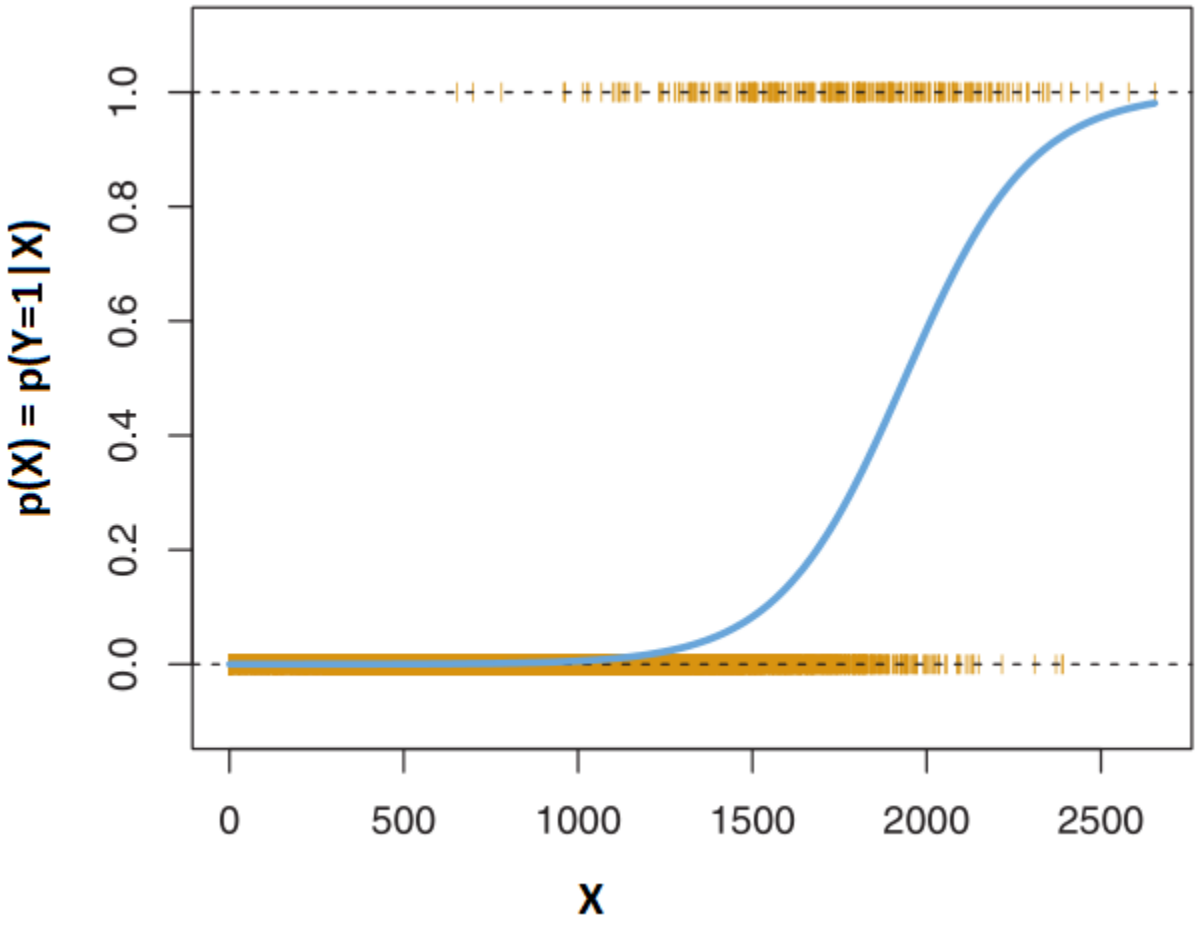


Figure 3.1:

As for prediction, we use the model built with the estimated parameters to predict probabilities. For example,

If $X = 1000$,

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1000}}{1+e^{-10.6513+0.0055\times 1000}} = 0.006$$

If $X = 2000$,

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 2000}}{1+e^{-10.6513+0.0055\times 2000}} = 0.586$$

3.3 Multiple Logistic Regression

We now consider the problem of predicting a binary response using multiple predictors. By analogy with the extension from simple to multiple linear regression in the previous chapters, we can generalize the simple

logistic regression equation as follows:

$$\log(\frac{p(X)}{1-p(X)}) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$$

where $X = (X_1, \ldots, X_p)$ are p predictors. The equation above can be rewritten as

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_p X_p}}$$

Just as in the simple logistic regression we use the maximum likelihood method to estimate $\beta_0, \beta_1, \ldots, \beta_p$.

3.4 Example

3.4.1 Case study: *The Challenger disaster*

The *Challenger* disaster occurred on the 28th January of 1986, when the NASA Space Shuttle orbiter *Challenger* broke apart and disintegrated at 73 seconds into its flight, leading to the deaths of its seven crew members. The accident deeply shocked the US society, in part due to the attention the mission had received because of the presence of Christa McAuliffe, who would have been the first astronaut-teacher. Because of this, NASA TV broadcasted live the launch to US public schools, which resulted in millions of school children witnessing the accident. The accident had serious consequences for the NASA credibility and resulted in an interruption of 32 months in the shuttle program. The Presidential *Rogers Commission* (formed by astronaut Neil A. Armstrong and Nobel laureate Richard P. Feynman, among others) was created to investigate the disaster.

Challenger launch and posterior explosion, as broadcasted live by NBC in 28/01/1986.

The Rogers Commission elaborated a report (Presidential Commission on the Space Shuttle Challenger Accident, 1986) with all the findings. The commission determined that the disintegration began with the **failure of an O-ring seal in the solid rocket motor due to the unusual cold temperatures (-0.6 Celsius degrees)** during the launch. This failure produced a breach of burning gas through the solid rocket motor that compromised the whole shuttle structure, resulting in its disintegration due to the extreme aerodynamic forces. The **problematic with O-rings was something known**: the night before the launch, there was a three-hour teleconference between motor engineers and NASA management, discussing the effect of low temperature forecasted for the launch on the O-ring performance. The conclusion, influenced by Figure 3.2a, was:

> **"Temperature data [are] not conclusive on predicting primary O-ring blowby."**

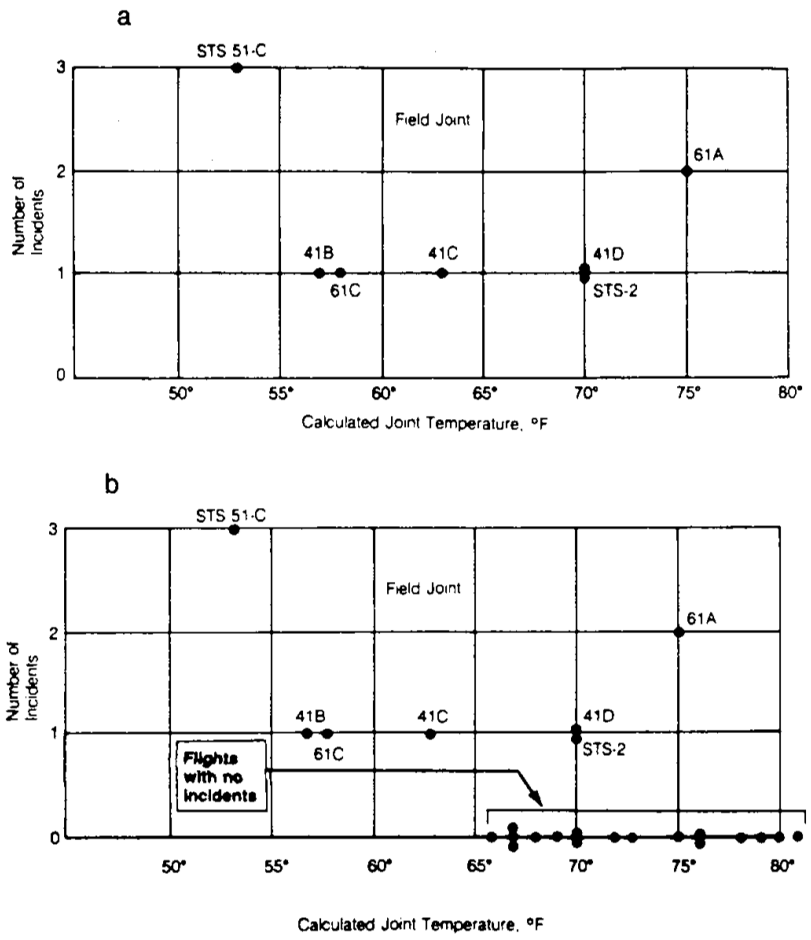


Figure 3.2: Number of incidents in the O-rings (filed joints) versus temperatures. Panel *a* includes only flights with incidents. Panel *b* contains all flights (with and without incidents).

The Rogers Commission noted a major flaw in Figure 3.2a: the **flights with zero incidents were excluded** from the plot because *it was felt* that **these flights did not contribute any information about the temperature effect** (Figure 3.2b). The Rogers Commission concluded:

> **"A careful analysis of the flight history of O-ring performance would have revealed the correlation of O-ring damage in low temperature"**.

The purpose of this case study, inspired by Dalal et al. (1989), is to quantify what was the influence of the temperature in the probability of having at least one incident related with the O-rings. Specifically, we want to address the following questions:

- Q1. *Is the temperature associated with O-ring incidents?*
- Q2. *In which way was the temperature affecting the probability of O-ring incidents?*
- Q3. *What was the predicted probability of an incidient in an O-ring for the temperature of the launch day?*

To try to answer these questions we have the `challenger` (▦ dataset). The dataset contains (as shown in the table below) information regarding the state of the solid rocket boosters after launch[1] for 23 flights. Each row has, among others, the following variables:

[1] After the shuttle exits the atmosphere, the solid rocket boosters separate and descend to land using a parachute where they are carefully analyzed.

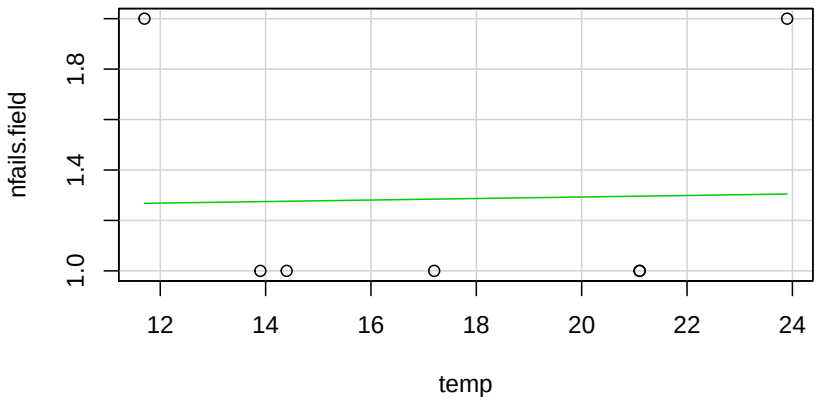
- `fail.field`, `fail.nozzle`: binary variables indicating whether there was an incident with the O-rings in the field joints or in the nozzles of the solid rocket boosters. `1` codifies an incident and `0` its absence. On the analysis, we focus on the O-rings of the field joint as being the most determinants for the accident.
- `temp`: temperature in the day of launch. Measured in Celsius degrees.
- `pres.field`, `pres.nozzle`: leak-check pressure tests of the O-rings. These tests assured that the rings would seal the joint.

Table 3.2: The `challenger` dataset.

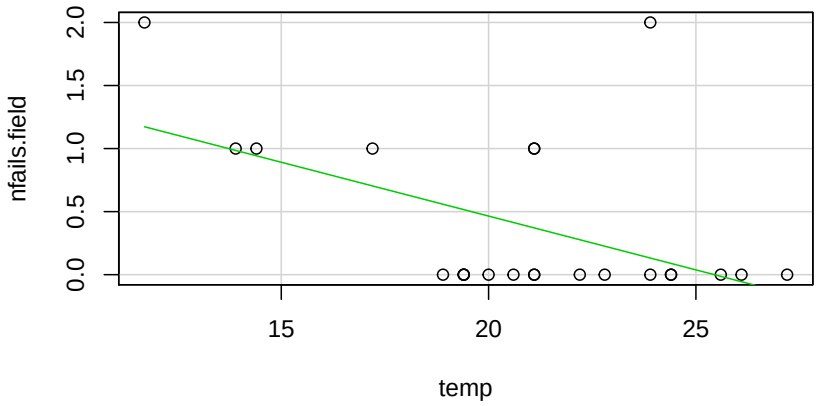
flight	date	fail.field	fail.nozzle	temp
1	12/04/81	0	0	18.9
2	12/11/81	1	0	21.1
3	22/03/82	0	0	20.6
5	11/11/82	0	0	20.0
6	04/04/83	0	1	19.4
7	18/06/83	0	0	22.2
8	30/08/83	0	0	22.8
9	28/11/83	0	0	21.1
41-B	03/02/84	1	1	13.9
41-C	06/04/84	1	1	17.2
41-D	30/08/84	1	1	21.1
41-G	05/10/84	0	0	25.6
51-A	08/11/84	0	0	19.4
51-C	24/01/85	1	1	11.7
51-D	12/04/85	0	1	19.4
51-B	29/04/85	0	1	23.9
51-G	17/06/85	0	1	21.1
51-F	29/07/85	0	0	27.2
51-I	27/08/85	0	0	24.4
51-J	03/10/85	0	0	26.1
61-A	30/10/85	1	0	23.9
61-B	26/11/85	0	1	24.4
61-C	12/01/86	1	1	14.4

Let's begin the analysis by replicating Figures 3.2a and 3.2b and checking that linear regression is not the right tool for answering Q1–Q3. For that, we make two scatterplots of `nfails.field` (number of total incidents in the field joints) in function of `temp`, the first one excluding the launches without incidents (`subset = nfails.field > 0`) and the second one for all the data.

```
require(car)
#ans> Loading required package: car
scatterplot(nfails.field ~ temp, reg.line = lm, smooth = FALSE, spread = FALSE,
            boxplots = FALSE, data = challenger, subset = nfails.field > 0)
```

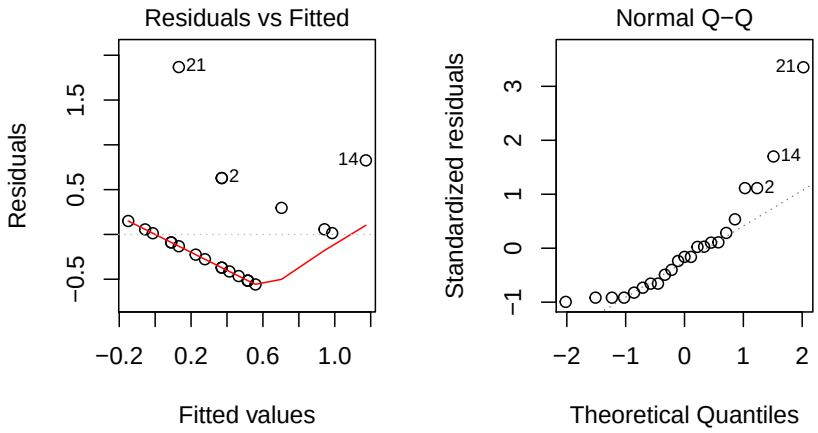



```
scatterplot(nfails.field ~ temp, reg.line = lm, smooth = FALSE, spread = FALSE,
            boxplots = FALSE, data = challenger)
```



There is a fundamental problem in using linear regression for this data: **the response is not continuous**. As a consequence, there is no linearity and the errors around the mean are not normal (indeed, they are strongly non normal). We can check this with the corresponding diagnostic plots:

```
mod <- lm(nfails.field ~ temp, data = challenger)
par(mfrow = 1:2)
plot(mod, 1)
plot(mod, 2)
```



Although linear regression is not the adequate tool for this data, it is able to detect the obvious difference between the two plots:

1. **The trend for launches with incidents is flat, hence suggesting there is no dependence on the temperature** (Figure 3.2a). This was one of the arguments behind NASA's decision of launching the rocket at a temperature of -0.6 degrees.
2. However, **the trend for *all* launches indicates a clear negative dependence between temperature and number of incidents!** (Figure 3.2b). Think about it in this way: the minimum temperature for a launch without incidents ever recorded was above 18 degrees, and the Challenger was launched at -0.6 without clearly knowing the effects of such low temperatures.

Instead of trying to predict the number of incidents, we will concentrate on modeling the *probability of expecting at least one incident given the temperature*, a simpler but also revealing approach. In other words, we look to estimate the following curve:

$$p(x) = \mathbb{P}(\text{incident} = 1 | \text{temperature} = x)$$

from `fail.field` and `temp`. This probability can not be properly modeled as a linear function like $\beta_0 + \beta_1 x$, since inevitably will fall outside $[0, 1]$ for some value of x (some will have negative probabilities or probabilities larger than one). The technique that solves this problem is the **logistic regression**. The logistic model in this case is

$$\mathbb{P}(\text{incident} = 1 | \text{temperature} = x) = \text{logistic}\left(\beta_0 + \beta_1 x\right) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}},$$

with β_0 and β_1 unknown.

Let's fit the model to the data by estimating β_0 and β_1.

```
nasa <- glm(fail.field ~ temp, family = "binomial", data = challenger)
summary(nasa)
#ans>
#ans> Call:
#ans> glm(formula = fail.field ~ temp, family = "binomial", data = challenger)
#ans>
#ans> Deviance Residuals:
#ans>    Min      1Q  Median      3Q     Max
#ans> -1.057  -0.757  -0.382   0.457   2.220
#ans>
#ans> Coefficients:
#ans>             Estimate Std. Error z value Pr(>|z|)
```

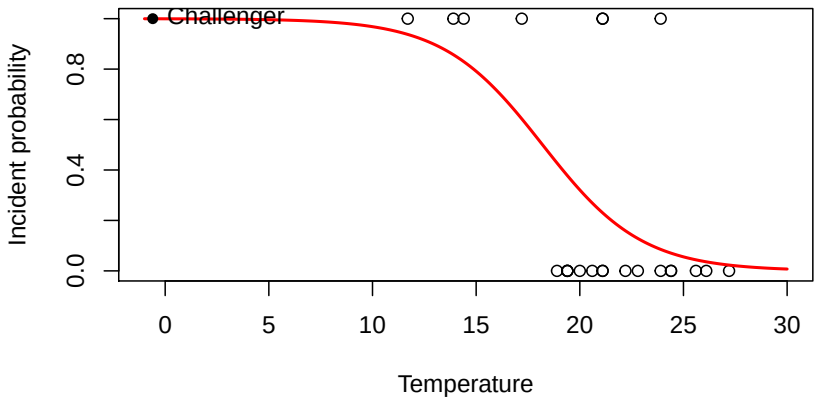
```
#ans> (Intercept)    7.584      3.915    1.94    0.053 .
#ans> temp          -0.417      0.194   -2.15    0.032 *
#ans> ---
#ans> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#ans>
#ans> (Dispersion parameter for binomial family taken to be 1)
#ans>
#ans>     Null deviance: 28.267  on 22  degrees of freedom
#ans> Residual deviance: 20.335  on 21  degrees of freedom
#ans> AIC: 24.33
#ans>
#ans> Number of Fisher Scoring iterations: 5
exp(coef(nasa)) # Exponentiated coefficients ("odds ratios")
#ans> (Intercept)        temp
#ans>    1965.974       0.659
```

The `glm()` function fits **g**eneralized **l**inear **m**odels, a class of models that includes logistic regression. The syntax of the `glm()` function is similar to that of `lm()`, except that we must pass in the argument `family=binomial` in order to tell `R` to run a logistic regression rather than some other type of generalized linear model.

The summary of the logistic model is notably different from the linear regression, as the methodology behind is quite different. Nevertheless, we have tests for the significance of each coefficient. Here we obtain that `temp` is significantly different from zero, at least at a level $\alpha = 0.05$. Therefore we can conclude that **the temperature is indeed affecting the probability of an incident with the O-rings** (answers Q1).

The coefficient of `temp`, $\hat{\beta}_1$, can be regarded the "correlation between the temperature and the probability of having at least one incident". This correlation, as evidenced by the sign of $\hat{\beta}_1$, is negative. Let's plot the fitted logistic curve to see that indeed the probability of incident and temperature are negatively correlated:

```
# Plot data
plot(challenger$temp, challenger$fail.field, xlim = c(-1, 30), xlab = "Temperature",
     ylab = "Incident probability")

# Draw the fitted logistic curve
x <- seq(-1, 30, l = 200)
y <- exp(-(nasa$coefficients[1] + nasa$coefficients[2] * x))
y <- 1 / (1 + y)
lines(x, y, col = 2, lwd = 2)

# The Challenger
points(-0.6, 1, pch = 16)
text(-0.6, 1, labels = "Challenger", pos = 4)
```



At the sight of this curve and the summary of the model we can conclude that **the temperature was increasing the probability of an O-ring incident** (Q2). Indeed, the confidence intervals for the coefficients show a significative negative correlation at level $\alpha = 0.05$:

```
confint(nasa, level = 0.95)
#ans>                2.5 % 97.5 %
#ans> (Intercept)   1.336 17.783
#ans> temp         -0.924 -0.109
```

Finally, **the probability of having at least one incident with the O-rings in the launch day was** 0.9996 **according to the fitted logistic model** (Q3). This is easily obtained:

```
predict(nasa, newdata = data.frame(temp = -0.6), type = "response")
#ans> 1
#ans> 1
```

Be aware that `type = "response"` has a different meaning in logistic regression. In linear models it returns a CI for the prediction. But, `type = "response"` means that the *probability* should be returned, instead of the value of the link function, which is returned with `type = "link"` (the default).

Recall that there is a serious problem of **extrapolation** in the prediction, which makes it less precise (or more variable). But this extrapolation, together with the evidences raised by a simple analysis like we did, should have been strong arguments for postponing the launch.

PW 3

Social Networks Ads

In this PW we are going to analyse the `Social_Network_Ads` dataset . This dataset contains informations of users of a social network. The social network has several business clients and its business clients put ads on the social network for marketing compaigns purposes. For this dataset, a company has put ads for one of its new products and the social network gathered some informations about wich users responded positively to the ad by buying the product and those who responded negatively by not buying the product.

1. Download the `Social_Network_Ads` dataset and import it into `R`.

2. **Explore** and **Describe** the dataset (you can use `str()` and `summary()` functions, you can calculate and visualize the correlations, show some histograms, scatterplots, pie charts, etc..).

We will consider the variables `Age` and `EstimatedSalary` as input variables (features) to see the correlations between them and the decision of the user to buy (or not) the product.

3. Now we are going to split the dataset into training set and test set. Last week we did it manually. From now on we will split it randomly, you can use this code (after undestanding it of course):

```
library(caTools) # install it first in the console
set.seed(123)
# we use the function set.seed() with the same seed number
# to randomly generate the same values, you already know that right?
#and you know why we want to generate the same values, am I wrong?
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
# here we chose the SplitRatio to 75% of the dataset,
# and 25% for the test set.
training_set = subset(dataset, split == TRUE)
# we use subset to split the dataset
test_set = subset(dataset, split == FALSE)
```

4. Scale the input variables in both training set and test set. Do you know what is scaling? Explain it one sentence.

5. Now fit a **simple** logistic regression model of `Purchased` in function of `Age`.

6. As you saw in the Logistic Regression chapter and in the previous question, we choose argument `family` to be `binomial` when we use the function `glm`. Explain why!

7. Write in a equation the model you obtained in question **5**? (Read the following note concerning this question).

Note that there is three different ways to express the equation of a Logistic Regression model.

8. Is the feature `Age` significant to the model? Justify your answer.

9. What is the value of AIC of the model?

The **AIC** is the **Akaike Information Criterion**. You will use this while comparing multiple models. The model with lower value of AIC is better. Suppose that we have a statistical model of some data. Let $\hat{L}$ be the maximum value of the likelihood function for the model; let k be the number of estimated parameters in the model. Then the AIC value of the model is the following.

$$\text{AIC} = 2k - 2\ln(\hat{L})$$

where

- $\hat{L}$ = the maximized value of the likelihood function of the model M, i.e. $\hat{L} = p(x|\hat{\beta}, M)$, where $\hat{\beta}$ are the parameter values that maximize the likelihood function.
- x = the observed data.
- k = the number of free parameters to be estimated. If the model under consideration is a linear regression, k is the number of regressors, including the intercept.

10. Plot `Purchased` in function of `Age` and add the curve of the obtained logistic regression model.

(**Hints**: One way to do it is to first plot the observations, then use the `curve()` function with option `add=TRUE` in order to add the curve to the plot. Pay attention that the argument "type" of the function `predict()` must be `reponse`)

You must obtain something like

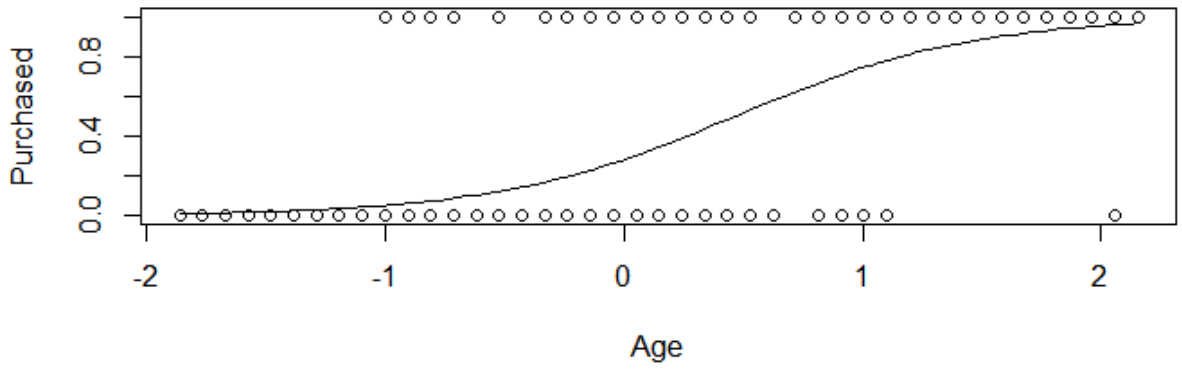


Figure 3.3:

Extra: A great R package for visualization if `ggplot2`. Take a look on this link for some examples. With this library we obtain the following plot for our model,

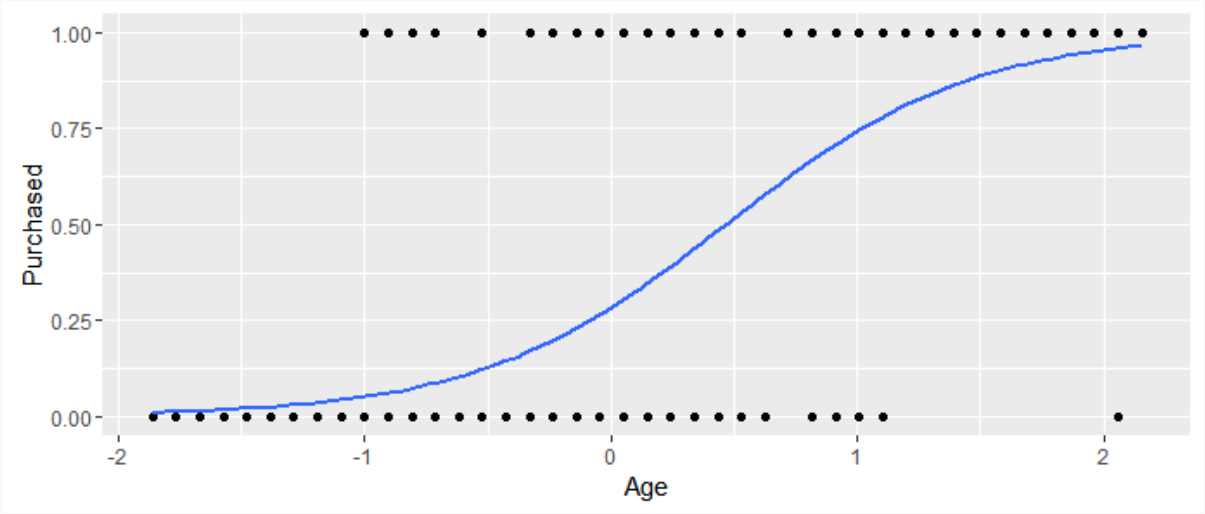


Figure 3.4:

I obtained the last figure with these lines of code,

```
library(ggplot2)
ggplot(training_set, aes(x=Age, y=Purchased)) +
  geom_point() +
  stat_smooth(method="glm", method.args=list(family="binomial"), se=FALSE)
```

11. Now let us take another feature into account in the model. Fit a logistic regression model of purchasing the product in function of the age of a user and its salary.

12. Are the predictors significant to the new model?

13. Do we obtain a better model by adding the estimated salary?

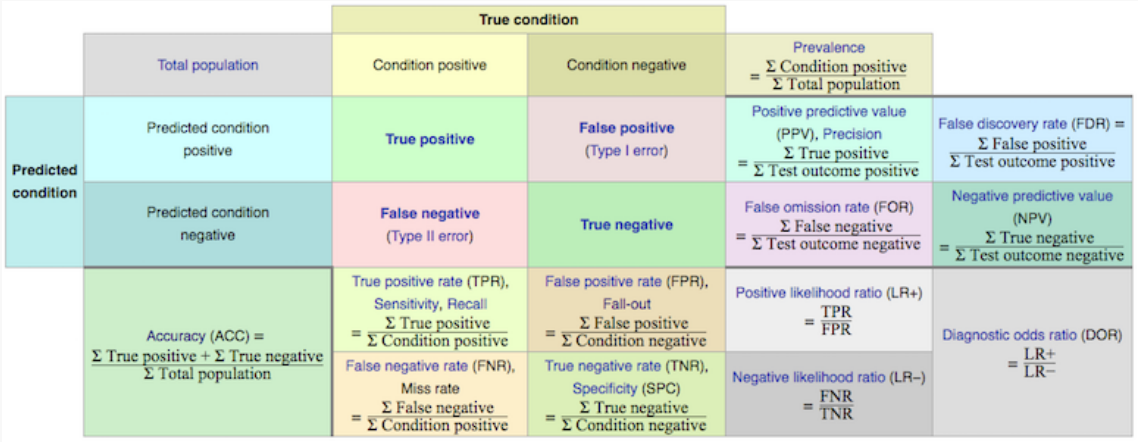
14. Predictions: On the test set, predict the probability of purchasing the product by the users using the obtained model.

15. Take a look on your predicted values for the variable `Purchased`. We predicted the probability that the user will purchase the product right? Now in order to compare your results with the real answers, transform the predicted values to 0 or 1 (1 if >0.5).

Hint: You can easily do it with the `ifelse()` function.

Now read the following about the evaluation of a classifier (of a classification model).

Confusion matrix: is a tabular representation of Actual vs Predicted values. This helps us to find the accuracy of the model. The different results from a binary classifier are true positives, true negatives, false positives, and false negatives. This is how the confusion matrix looks like:



(Image source: Wikipedia)

You can calculate the **accuracy** of your model with:

$$\frac{\text{True Positive + True Negatives}}{\text{True Positive + True Negatives + False Positives + False Negatives}}$$

Figure 3.5:

Accuracy is a key measure of performance, and is more specifically the rate at which the model is able to predict the correct value (classification or regression) for a given data point or observation. In other words, accuracy is the proportion of correct predictions out of all predictions made.

The other two metrics from the confusion matrix worth discussing are **Precision** and **Recall**.

Precision (positive predictive value) is the ratio of true positives to the total amount of positive predictions made (i.e., true or false). Said another way, precision measures the proportion of accurate positive predictions out of all positive predictions made.

Recall on the other hand, or true positive rate, is the ratio of true positives to the total amount of actual positives, whether predicted correctly or not. So in other words, recall measures the proportion of accurate positive predictions out of all actual positive observations.

A metric that is associated with precision and recall is called the F-score (also called F1 score), which combines them mathematically, and somewhat like a weighted average, in order to produce a single measure of performance based on the simultaneous values of both. Its values range from 0 (worst) to 1 (best).

Another important concept to know about is the ***Receiver Operating Characteristic***, which when plotted, results in what's known as an **ROC** curve.

ROC Curve: An ROC curve is a two-dimensional plot of *sensitivity* (recall, or true positive rate) vs *1- specificity* (false positive rate). The area under the curve is referred to as the ***AUC***, and is a numeric metric used to represent the quality and performance of the classifier (model).

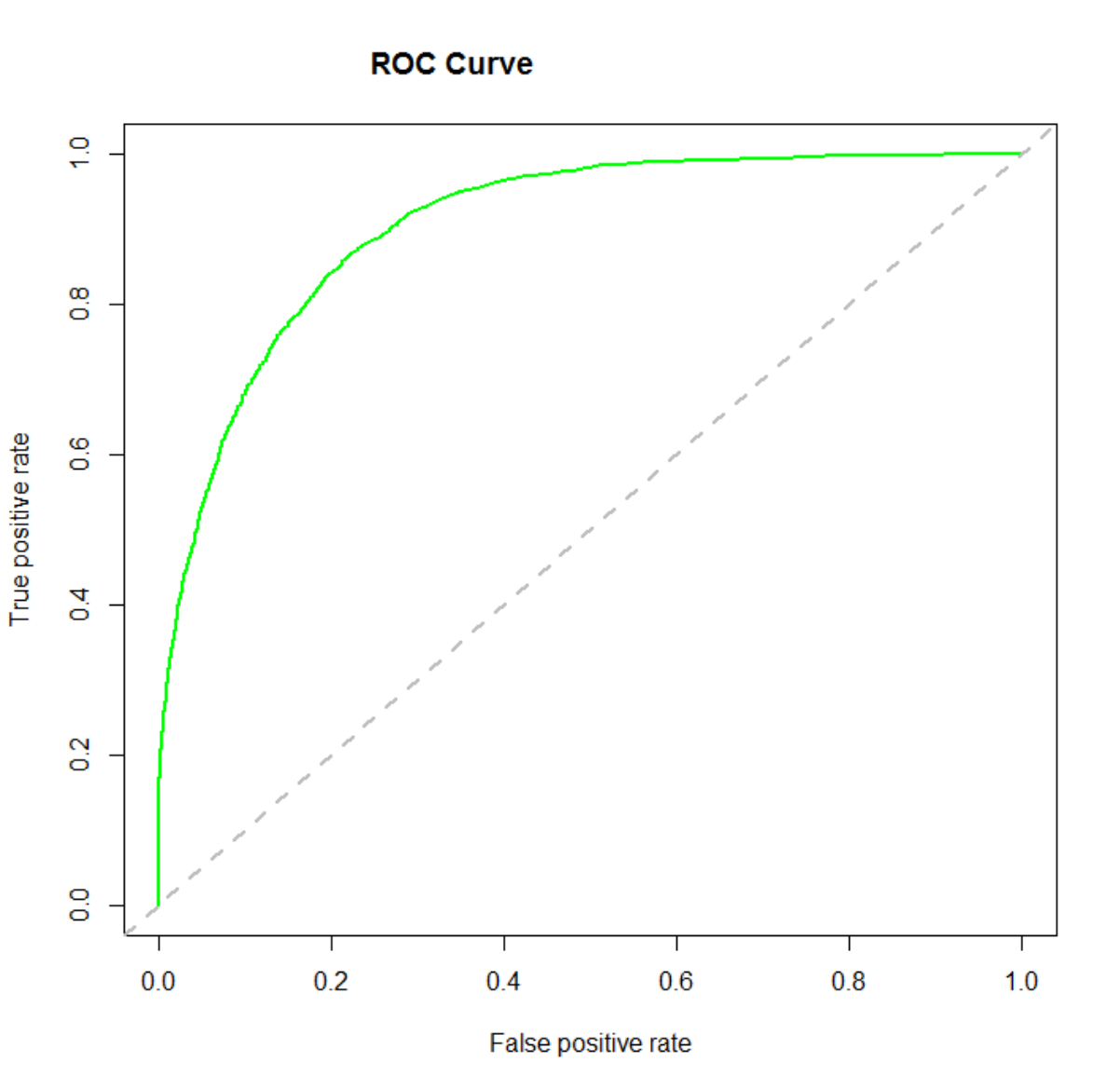


Figure 3.6:

An AUC of 0.5 is essentially the same as random guessing without a model, whereas an AUC of 1.0 is considered a perfect classifier. Generally, the higher the AUC value the better, and an AUC above 0.8 is considered quite good.

The higher the AUC value, the closer the curve gets to the upper left corner of the plot. One can easily see from the ROC curves then that the goal is to find and tune a model that maximizes the true positive rate, while simultaneously minimizing the false positive rate. Said another way, the goal as shown by the ROC curve is to correctly predict as many of the actual positives as possible, while also predicting as many of the actual negatives as possible, and therefore minimize errors (incorrect classifications) for both.

16. Now in order to evaluate the model and its predictions, compute the confusion matrix. What do you

obtain ?

(**Hint**: you can use the `table()` function).

17. Calculate the accuracy, specificity, sensitivity and the precision of the model.

(**Note**: You can create a function that takes a confusion matrix in input and returns the needed metrics)

18. Plot the ROC curve and calculate AUC value.

Hints: to plot it, install the `ROCR` package. Load and use the functions:

- `prediction()` to calculate the elements of the confusion matrix.
- `performance()` to calculate the AUC.
- `plot()` to plot the ROC curve, you can plot the performance calculated before.
- `abline()` to plot a line of equation `y=x`.

19. Compare the AUC of the two models you fitted (one with only age and one with age and estimated salary) and plot their ROC curves in the same figure.

Appendix A

Introduction to `RStudio`

`RStudio` is the most employed Integrated Development Environment (IDE) for `R` nowadays. When you start `RStudio` you will see a window similar to Figure A.1. There are a lot of items in the GUI, most of them described in the `RStudio` IDE Cheat Sheet. The most important things to keep in mind are:

1. The code is written in scripts in the *source panel* (upper-right panel in Figure A.1);
2. for running a line or code selection from the script in the *console* (first tab in the lower-right panel in Figure A.1), you do it with the keyboard shortcut `'Ctrl+Enter'` (Windows and Linux) or `'Cmd+Enter'` (Mac OS X).

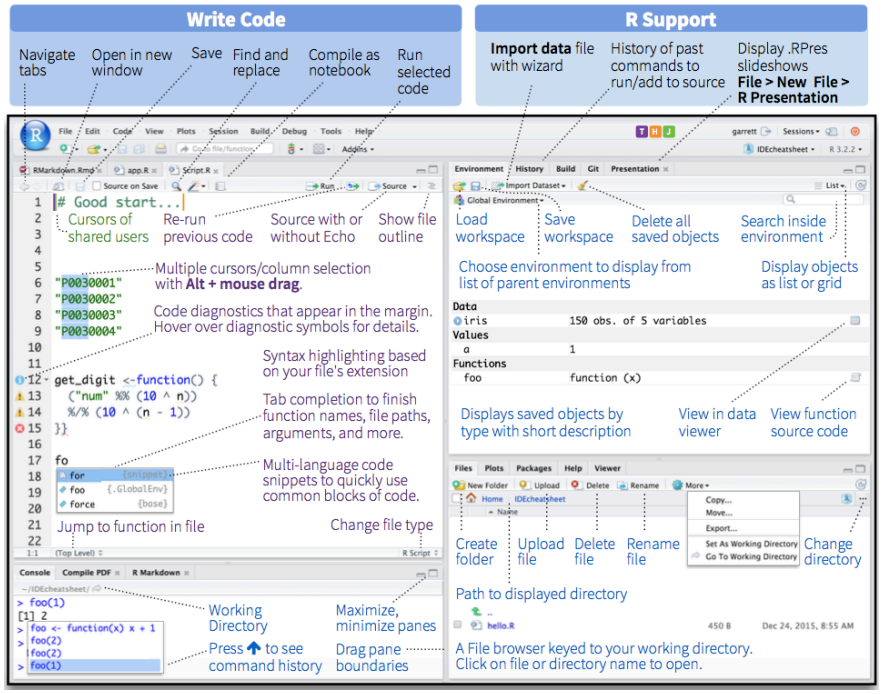


Figure A.1: Main window of 'RStudio'. The red shows the code panel and the yellow shows the console output. Extracted from [here](https://www.rstudio.com/wp-content/uploads/2016/01/rstudio-IDE-cheatsheet.pdf).

Appendix B

Review on hypothesis testing

The process of hypothesis testing has an interesting analogy with a trial that helps on understanding the elements present in a formal hypothesis test in an intuitive way.

Hypothesis testing	Trial
Null hypothesis H_0	**Accused of comitting a crime**. It has the "presumption of innocence", which means that it is *not guilty* until there is enough evidence to supporting its guilt
Sample $X_1, \ldots, X_n$	Collection of small **evidences supporting innocence and guilt**. These evidences contain a certain degree of uncontrollable randomness because of how they were collected and the context regarding the case
Statistic T_n	**Summary of the evicences presented** by the prosecutor and defense lawyer
Distribution of T_n under H_0	The **judge** conducting the trial. Evaluates the evidence presented by both sides and presents a verdict for H_0
Significance level α	$1-\alpha$ is the **strength of evidences required by the judge for condemning** H_0. The judge allows evidences that on average condemn $100\alpha\%$ of the innocents, due to the randomness inherent to the evidence collection process. $\alpha = 0.05$ is considered a reasonable level
p-value	**Decision** of the judge that measures the degree of compatibility, in a scale 0–1, of the presumption of innocence with the summary of the evidences presented. If p-value$< \alpha$, H_0 is declared guilty. Otherwise, is declared not guilty
H_0 is rejected	H_0 is declared guilty: there are **strong evidences supporting its guilt**
H_0 is not rejected	H_0 is declared not guilty: either is **innocent or there are no enough evidences supporting its guilt**

More formally, the p-value of an hypothesis test about $H0$ is defined as:

> The p-value is the probability of obtaining a statistic more unfavourable to H_0 than the observed, assuming that H_0 is true.

Therefore, **if the p-value is small** (smaller than the chosen level α), **it is unlikely that the evidence against H_0 is due to randomness. As a consequence, H_0 is rejected**. If the p-value is large (larger

than α), then it is more possible that the evidences against H_0 are merely due to the randomness of the data. In this case, we do not reject H_0.

If H_0 holds, then the p-value (which is a random variable) is distributed uniformly in $(0, 1)$. If H_0 does not hold, then the distribution of the p-value is not uniform but concentrated at 0 (where the rejections of H_0 take place).

Appendix C

Use of qualitative predictors

An important situation is how to deal with *qualitative*, and not *quantitative*, predictors when we fit a regression model. Qualitative predictors, also known as *categorical* variables or, in `R`'s terminology, *factors*, are very common, for example in social sciences. Dealing with them requires some care and proper understanding of how these variables are represented.

The simplest case is the situation with **two levels**. A binary variable C with two levels (for example, a and b) can be represented as

$$D = \begin{cases} 1, & \text{if } C = b, \\ 0, & \text{if } C = a. \end{cases}$$

D now is a *dummy variable*: it codifies with zeros and ones the two possible levels of the categorical variable. An example of C could be *gender*, which has levels *male* and *female*. The dummy variable associated is $D = 0$ if the gender is male and $D = 1$ if the gender is female.

The advantage of this *dummification* is its interpretability in regression models. Since level a corresponds to 0, it can be seen as the *reference level* to which level b is compared. This is the key point in dummification: **set one level as the reference and codify the rest as departures from it** with ones.

`R` does the dummification automatically (translates a categorical variable C into its dummy version D) if it detects that a factor variable is present in the regression model.

Let's see now the case with **more than two levels**, for example, a categorical variable C with levels a, b, and c. If we take a as the reference level, this variable can be represented by *two* dummy variables:

$$D_1 = \begin{cases} 1, & \text{if } C = b, \\ 0, & \text{if } C \neq b \end{cases}$$

and

$$D_2 = \begin{cases} 1, & \text{if } C = c, \\ 0, & \text{if } C \neq c. \end{cases}$$

Then $C = a$ is represented by $D_1 = D_2 = 0$, $C = b$ is represented by $D_1 = 1, D_2 = 0$ and $C = c$ is represented by $D_1 = 0, D_2 = 1$.

In general, if we have a categorical variable with J levels, then the number of dummy variables required is $J - 1$. Again, `R` does the dummification automatically for you if it detects that a factor variable is present in the regression model.

It may happen that one dummy variable, say D_1 is not significant, while other dummy variables, say D_2, are significant.

Do not codify a categorical variable as a discrete variable. This constitutes a major methodological fail that will flaw the subsequent statistical analysis.

For example if you have a categorical variable `party` with levels `partyA`, `partyB`, and `partyC`, do not encode it as a discrete variable taking the values `1`, `2`, and `3`, respectively. If you do so:

- You assume implicitly an order in the levels of `party`, since `partyA` is closer to `partyB` than to `partyC`.
- You assume implicitly that `partyC` is three times larger than `partyA`.
- The codification is completely arbitrary – why not considering `1`, `1.5`, and `1.75` instead of?

The right way of dealing with categorical variables in regression is to set the variable as a factor and let `R` do internally the dummification.

Appendix D

Model Selection

Linear Model Selection and Best Subset Selection

Forward Stepwise Selection

Backward Stepwise Selection

Estimating Test Error Using Mallow's Cp, AIC, BIC, Adjusted R-squared

Estimating Test Error Using Cross-Validation

Examples

Best Subset Selection

Forward Stepwise Selection and Model Selection Using Validation Set

Model Selection Using Cross-Validation

Appendix E

References

Bibliography

Dalal, S. R., Fowlkes, E. B., and Hoadley, B. (1989). Risk analysis of the space shuttle: Pre-challenger prediction of failure. *Journal of the American Statistical Association*, 84(408):945–957.

Presidential Commission on the Space Shuttle Challenger Accident (1986). *Report of the Presidential Commission on the Space Shuttle Challenger Accident (Vols. 1 & 2)*. Washington, DC.