The dataset

We'll be working on the *Titanic dataset*. The dataset (training) is a collection of data about some of the passengers (889 to be precise), and the goal is to predict the survival (either 1 if the passenger survived or 0 if they did not) based on some features such as the *class of service*, the *sex*, the *age* etc. As you can see, we are going to use both categorical and continuous variables.

The data cleaning process

When working with a real dataset we need to take into account the fact that some data might be missing or corrupted, therefore we need to prepare the dataset for our analysis. As a first step we load the csv data using the `read.csv()` function.
Make sure that the parameter `na.strings` is equal to `c("")` so that each missing value is coded as a `NA`. This will help us in the next steps.

```
training.data.raw <- read.csv('train.csv',header=T,na.strings=c(""))
```

Now we need to check for missing values and look how many unique values there are for each variable using the `sapply()` function which applies the function passed as argument to each column of the dataframe.

```
sapply(training.data.raw,function(x) sum(is.na(x)))
```

```
PassengerId    Survived      Pclass        Name         Sex
          0           0           0           0           0
        Age       SibSp       Parch      Ticket        Fare
        177           0           0           0           0
      Cabin    Embarked
        687           2
```

The variable cabin has too many missing values, we will not use it. We will also drop PassengerId since it is only an index and Ticket.
Using the `subset()` function we subset the original dataset selecting the relevant columns only.

```
data <- subset(training.data.raw,select=c(2,3,5,6,7,8,10,12))
```

Taking care of the missing values

Now we need to account for the other missing values. R can easily deal with them when fitting a generalized linear model by setting a parameter inside the fitting function. However, personally I prefer to replace the `NAs` "by hand", when is possible. There are different ways to do this, a typical approach is to replace the missing values with the average, the median or the mode of the existing one. I'll be using the average.

```
data$Age[is.na(data$Age)] <- mean(data$Age,na.rm=T)
```

As far as categorical variables are concerned, using the `read.table()` or `read.csv()` by default will encode the categorical variables as factors. A factor is how R deals categorical variables.

As for the missing values in Embarked, since there are only two, we will discard those two rows (we could also have replaced the missing values with the mode and keep the datapoints).

```
data <- data[!is.na(data$Embarked),]
rownames(data) <- NULL
```

Model fitting

We split the data into two chunks: training and testing set. The training set will be used to fit our model which we will be testing over the testing set.

```
train <- data[1:800,]
test <- data[801:889,]
```

Now, let's fit the model. Be sure to specify the parameter `family=binomial` in the `glm()` function.

```
model <- glm(Survived ~.,family=binomial(link='logit'),data=train)
```

By using function `summary()` we obtain the results of our model:

```
summary(model)
```

```
Call:
glm(formula = Survived ~ ., family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6064  -0.5954  -0.4254   0.6220   2.4165

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.137627   0.594998   8.635  < 2e-16 ***
Pclass      -1.087156   0.151168  -7.192 6.40e-13 ***
Sexmale     -2.756819   0.212026 -13.002  < 2e-16 ***
Age         -0.037267   0.008195  -4.547 5.43e-06 ***
SibSp       -0.292920   0.114642  -2.555   0.0106 *
Parch       -0.116576   0.128127  -0.910   0.3629
Fare         0.001528   0.002353   0.649   0.5160
EmbarkedQ   -0.002656   0.400882  -0.007   0.9947
EmbarkedS   -0.318786   0.252960  -1.260   0.2076
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1065.39  on 799  degrees of freedom
Residual deviance:  709.39  on 791  degrees of freedom
AIC: 727.39

Number of Fisher Scoring iterations: 5
```

Assessing the predictive ability of the model

In the steps above, we briefly evaluated the fitting of the model, now we would like to see how the model is doing when predicting *y* on a new set of data. By setting the parameter `type='response'`, R will output probabilities in the form of P(y=1|X). Our decision boundary will be 0.5. If P(y=1|X) > 0.5 then y = 1 otherwise y=0. Note that for some applications different thresholds could be a better option.

```
fitted.results <-
predict(model,newdata=subset(test,select=c(2,3,4,5,6,7,8)),type='respon
se')
fitted.results <- ifelse(fitted.results > 0.5,1,0)

misClasificError <- mean(fitted.results != test$Survived)
print(paste('Accuracy',1-misClasificError))
```

"Accuracy 0.842696629213483"

The 0.84 accuracy on the test set is quite a good result. However, keep in mind that this result is somewhat dependent on the manual split of the data that I made earlier, therefore if you wish for a more precise score, you would be better off running some kind of cross validation such as k-fold cross validation.