

Lecture 2: First steps with a dataset

Isabel Casas
icasas@deusto.es

Class content

- Using MPG data:
 - ▶ Data cleaning, missing values
 - ▶ Data description and visualisation
 - ▶ Use linear regression and regression trees to estimate and predict

We will use the following R packages: `car`, `rpart`, `rpart.plot`. Install them once if you have not done so.

```
install.packages (c("car", "rpart", "rpart.plot"))
```

Section 1

Data collection and handling

Activity 1 in groups of 3 or 4 (10 minutes)

- Working the MPG data downloaded last week, answer the following questions:
 - 1 How many files are in the .zip files?
 - 2 What are the type of files?
 - 3 How do you open each?
 - 4 Open *auto-mpg.data-original*
 - ▶ Does it have a header (name of variables in first row)?
 - ▶ Which are the data variable names?
 - ▶ Does it have missing values?
 - ▶ How many rows? and columns?

Load the data

We load the data into a data.frame called *mydata* and look at the first row

Because it does not have a header, we change the variable names from V1, ..., V9 to meaningful names

```
mydata <- read.table("./data/auto-mpg.data-original", header = FALSE)
#print the first row, to see the variables names
mydata[1,]
```

```
##      V1 V2  V3  V4   V5 V6 V7 V8                V9
## 1  18   8 307 130 3504 12 70   1 chevrolet chevelle malibu
names(mydata)
```

```
## [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8" "V9"
names(mydata) <- c("mpg", "cylinders", "displacement", "horsepower",
                  "weight", "acceleration", "model_year", "origin", "car_name")
names(mydata)
```

```
## [1] "mpg"           "cylinders"      "displacement"  "horsepower"    "weight"
## [6] "acceleration"  "model_year"    "origin"        "car_name"
```

Missing values

- What are missing values?
- Does our dataset has missing values? How many? Which positions? Which variables?

```
anyNA(mydata)
```

```
## [1] TRUE
```

```
sum(is.na(mydata))
```

```
## [1] 14
```

```
which(is.na(mydata))
```

```
## [1] 11 12 13 14 15 18 40 368 1257 1352 1556
```

Data visualisation and summarization

Let us take a glimpse at the data format of *mydata*. NA in my data?
Where?

```
summary(mydata)
```

```
##      mpg      cylinders      displacement      horsepower
## Min.   : 9.00    Min.   :3.000    Min.   : 68.0    Min.   : 46.00
## 1st Qu.:17.50    1st Qu.:4.000    1st Qu.:105.0    1st Qu.: 75.75
## Median :23.00    Median :4.000    Median :151.0    Median : 95.00
## Mean   :23.51    Mean   :5.475    Mean   :194.8    Mean   :105.08
## 3rd Qu.:29.00    3rd Qu.:8.000    3rd Qu.:302.0    3rd Qu.:130.00
## Max.   :46.60    Max.   :8.000    Max.   :455.0    Max.   :230.00
## NA's   :8
##      weight      acceleration      model_year      origin
## Min.   :1613    Min.   : 8.00    Min.   :70.00    Min.   :1.000
## 1st Qu.:2226    1st Qu.:13.70    1st Qu.:73.00    1st Qu.:1.000
## Median :2822    Median :15.50    Median :76.00    Median :1.000
## Mean   :2979    Mean   :15.52    Mean   :75.92    Mean   :1.569
## 3rd Qu.:3618    3rd Qu.:17.18    3rd Qu.:79.00    3rd Qu.:2.000
## Max.   :5140    Max.   :24.80    Max.   :82.00    Max.   :3.000
##
##      car_name
## Length:406
## Class :character
## Mode  :character
##
##
##
##
```

Data visualisation and summarization

Summary of variables *mpg* (first) and *horsepower* (fourth)

```
summary(mydata[, c(1,4)])
```

| ## | mpg | horsepower |
|----|---------------|----------------|
| ## | Min. : 9.00 | Min. : 46.00 |
| ## | 1st Qu.:17.50 | 1st Qu.: 75.75 |
| ## | Median :23.00 | Median : 95.00 |
| ## | Mean :23.51 | Mean :105.08 |
| ## | 3rd Qu.:29.00 | 3rd Qu.:130.00 |
| ## | Max. :46.60 | Max. :230.00 |
| ## | NA's :8 | NA's :6 |

mpg has 8 missing values and *horsepower* 6, a total of 14 missing values.

Data visualisation (histogram)

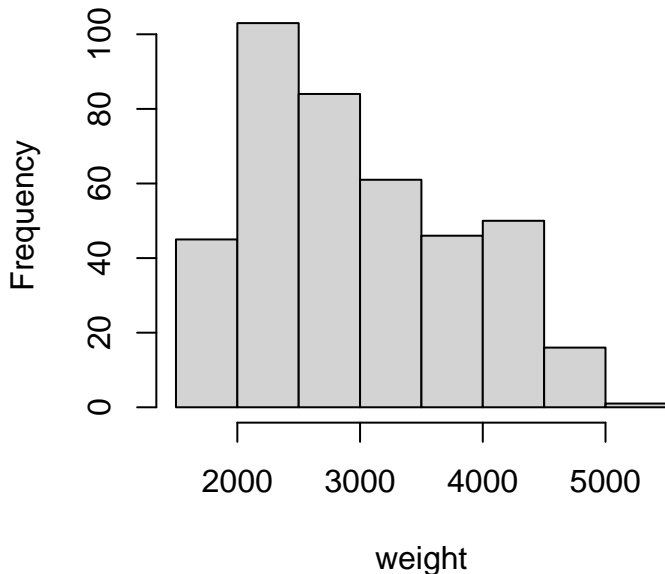
We can visualise the distribution of a continuous variable using a histogram, which in R is plotted using function *hist()*

Below, we plot the histogram of variable *weight*.

```
hist(mydata$weight,  
     main = "Hist of weight", xlab = "weight")
```

Hist of weight

Data visualisation (histogram)



Test of normality

Does the histogram of weight look normal?

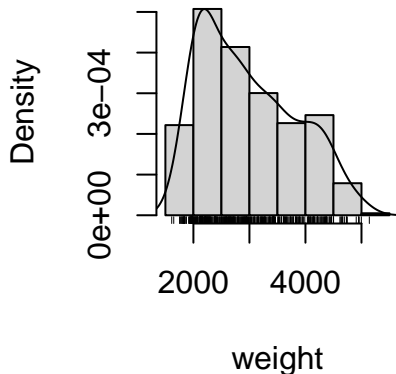
- We can run some statistical checks to confirm this hypothesis.
- We can visualise the QQ-plot of this variable and see if a line describes the data points

Test of normality: QQ-plot

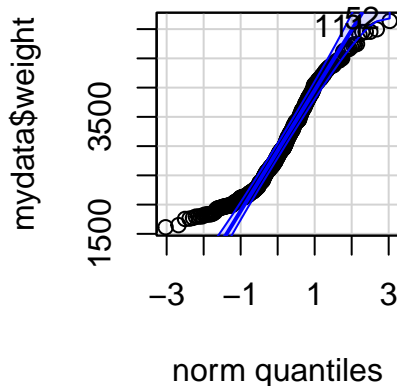
```
library(car)
par(mfrow=c(1,2))
hist(mydata$weight, freq =FALSE,
     main = "Histogram of car weight", xlab = "weight")
lines(density(mydata$weight, na.rm =TRUE))
rug(jitter(mydata$weight))
qqPlot(mydata$weight, main = "QQ plot of weight")
par(mfrow = c(1,1))
```

Test of normality: QQ-plot

Hist of weight



QQ plot of weight

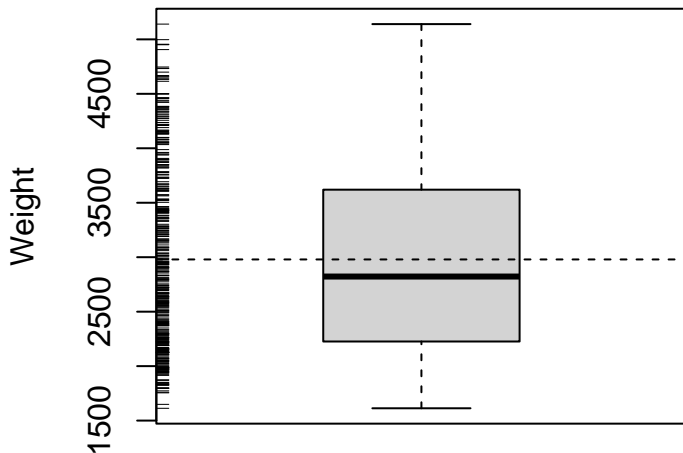


```
## [1] 52 111
```

Distribution symmetry and outliers: boxplot

```
boxplot(mydata$weight, ylab = "Weight")  
rug(jitter(mydata$weight), side = 2)  
abline(h = mean (mydata$weight, na.rm = T), lty = 2)
```

Distribution symmetry and outliers: boxplot



Activity 2 in groups

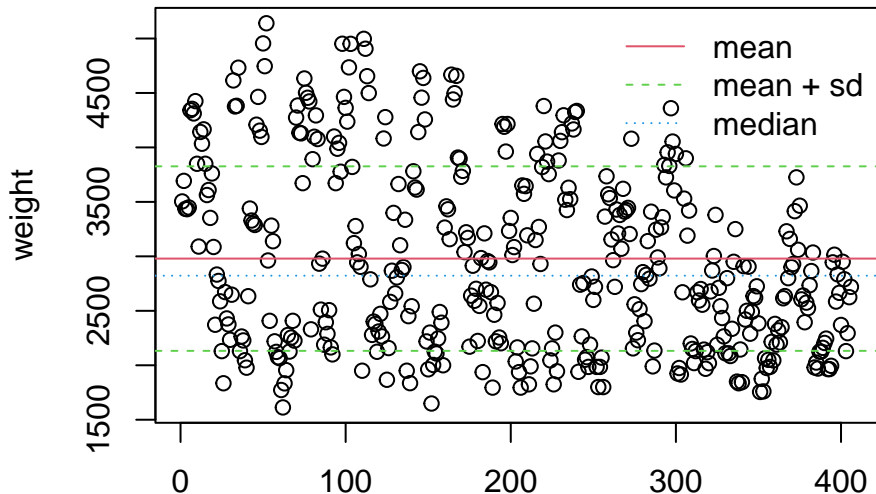
Summarise variable *mpg* from the *mydata*, plot its histogram and density function, check its normality of its distribution with a QQ-plot and plot its boxplot?

- Is the *mpg* variable normally distributed?
- Is the *mpg* distribution symmetric?
- Have you found any possible outliers?
- Plot variable *mpg*

Plotting a variable, its mean and standard deviation (sd)

```
plot(mydata$weight, xlab="", ylab="weight")
abline(h = mean(mydata$weight, na.rm = TRUE), lty = 1, col=2)
abline(h = mean(mydata$weight, na.rm = TRUE) +
      sd(mydata$weight, na.rm = T), lty = 2, col= 3)
abline(h = mean(mydata$weight, na.rm = TRUE) -
      sd(mydata$weight, na.rm = T), lty = 2, col= 3)
abline(h = median(mydata$weight, na.rm = TRUE),
      lty = 3, col= 4)
legend("topright", c("mean", "mean + sd", "median"),
      col = 2:4, lty =1:3, bty = "n")
```

Plotting a variable, its mean and standard deviation (sd)



Section 2

Missing values

What can we do with the missing (NA) values?

- We can remove every row in the data that has a missing value – **problem:** data lost
- We can do imputations of these values, i.e. fill-in those missing values with a “good” estimate

Removing the missing values

It makes sense if the number of missing values is small with respect to the size of your dataset.

There are only missing values in *mpg* and *horsepower*. Because *mpg* is the response/dependent variable, we will remove lines with NA in the *mpg*

```
nrow(mydata)
```

```
## [1] 406
```

```
#number of rows with NA in mpg
```

```
index <- which(is.na(mydata$mpg))
```

```
index
```

```
## [1] 11 12 13 14 15 18 40 368
```

```
# I create a new data.frame mydata2 without the rows with the  
#missing values of mpg
```

```
mydata2 <- mydata[-index, ]
```

```
nrow(mydata2)
```

```
## [1] 398
```

Filling-in the missing values with a central value

There were only missing values in *mpg* and *horsepower*. After removing the ones in the *mpg*, are there any other missing values?

```
index <- which(is.na(mydata2$horsepower))  
index
```

```
## [1] 33 127 331 337 355 375
```

```
hp.mean <- mean(mydata2$horsepower, na.rm = TRUE)  
hp.mean
```

```
## [1] 104.4694
```

```
mydata2$horsepower[index] <- hp.mean
```

Save your dataset

- We want to save the new version of our *mydata* in our computer to work on it in the future
- To save it in the working directory use function *write.table()*:

```
write.table(mydata2[, c(1:8)], "./data/mpg_new.csv", sep = ";",  
            row.names = FALSE, quote = FALSE,  
            col.names = TRUE)
```

Section 3

Fitting models

We have learnt to visualise the *MPG* data and clean its missing values. We will learn:

- How to use R to fit a linear regression and regression tree to the MPG data (estimation)

Training two models with MPG

- We fit two models to explain variable *mpg*.
 - ▶ Model 1: multiple linear regression (supervised, regression, continuous response)
 - ▶ Model 2: regression tree (supervised, regression and continuous response)
- *mpg* represents the miles per gallon consumed by the car in 100 miles.
- The rest of the variables are: “cylinders”, “displacement”, “horsepower”, “weight”, “acceleration”, “model_year”, “origin”, “car_name”

Question: which variables can we use as predictors?

Multiple linear regression (a bit of theory)

You are probably familiar with the formula of a linear regression

$$y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \dots + \beta_p X_{pi} + \epsilon_i \quad i = 1, 2, \dots, n$$

Also written like:

$$Y = X'\beta + \epsilon$$

In our case:

- y_i are each value of the response variable, *mpg*
- X is a matrix with all the predictors
- $\beta = (\beta_0, \beta_1, \beta_2, \dots, \beta_p)^T$ are the model coefficients, what we are interested in estimating

Multiple linear regression

Fitting a linear regression or estimating the coefficients of a linear regression are expressions referring to **estimating** (or guess/approximate) the real values of coefficient β (unknown) using our sample (known)".

- We find *estimates*, $\hat{\beta}$, by using the ordinary least squares (OLS - Minimos cuadrados ordinarios) method.
- The fitted values $\hat{Y} = X'\hat{\beta}$ are not Y , but their estimation. Note we do not using β but $\hat{\beta}$ in this formula
- \hat{Y} are the fitted values and $Y - \hat{Y}$ are the residuals, $\hat{\epsilon}$ (estimation error)

Multiple linear regression

Summarizing,

- We obtain the estimates $\hat{\beta}$ of the model coefficients β
- We calculate the fitted values \hat{Y} which are an estimation of Y
- We calculate the residuals $\hat{\epsilon} = Y - \hat{Y}$ which are an estimation of the model error, ϵ

Multiple linear regression

The estimates $\hat{\beta}$ are reliable if the following conditions are satisfied:

- 1 The residuals $\hat{\epsilon}$ have mean zero, i.e., they look around the zero line.
- 2 The residuals are normally distributed (check qqplot).
- 3 The error term is uncorrelated with X (there are no confounding variables, important in social science studies).
- 4 There is no pattern between $\hat{\epsilon}$ and \hat{Y} .

Video: [Linear regression in R](#)

Video: [An interesting application of regression](#)

Multiple linear regression for the *MPG* problem

The two commands below do the same thing, why?

```
lm.mpg1 <- lm (mpg ~ 1 + cylinders + displacement  
               + horsepower+weight+acceleration,  
               data = mydata2)
```

```
lm.mpg2 <- lm (mpg ~ ., data = mydata2[, 1:6])
```

```
summary(lm.mpg1)
```

```
##
## Call:
## lm(formula = mpg ~ 1 + cylinders + displacement + horsepower +
##     weight + acceleration, data = mydata2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.6878  -2.8456  -0.3652   2.2818  16.2732
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  45.8649597   2.6351102   17.405 < 2e-16 ***
## cylinders    -0.3587105   0.4100050    -0.875  0.3822
## displacement -0.0013923   0.0091024    -0.153  0.8785
## horsepower   -0.0390278   0.0161164    -2.422  0.0159 *
## weight       -0.0053656   0.0008064   -6.654 9.64e-11 ***
## acceleration -0.0070042   0.1227492    -0.057  0.9545
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.271 on 392 degrees of freedom
## Multiple R-squared:  0.7051, Adjusted R-squared:  0.7013
## F-statistic: 187.5 on 5 and 392 DF,  p-value: < 2.2e-16
```


Improving model fitness

The Adjusted-R² is 70% (not bad!) and the F-test discards the fact that all variables are statistically non-significant

- There are many variables in *lm.mpg1* that are statistically non-significant (large p-values in the t-tests)

Improving model fitness

- Let us remove some variables from the models using backward elimination with the *step* function

```
lm.mpg4 <- step(lm.mpg1)
summary(lm.mpg4)
```

Improving model fitness

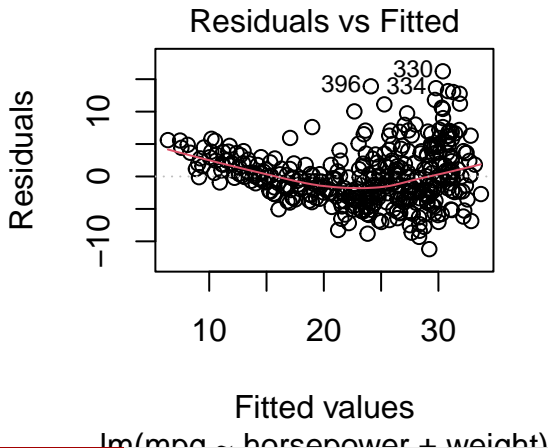
```
summary(lm.mpg4)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower + weight, data = mydata2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.2050  -2.7525  -0.3381   2.1622  16.2138
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 45.8415178  0.7897255   58.05 < 2e-16 ***
## horsepower  -0.0439036  0.0110021   -3.99 7.86e-05 ***
## weight      -0.0059723  0.0004963  -12.03 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.265 on 395 degrees of freedom
## Multiple R-squared:  0.7037, Adjusted R-squared:  0.7022
## F-statistic: 469.1 on 2 and 395 DF,  p-value: < 2.2e-16
```

- The Adjusted R2 has not improved a lot, but we have only the variables that are significant to explain/model the variability in *mpg* (horsepower and weight)

Check the model conditions

```
plot(lm.mpg4)
```



Regression trees

- Regression trees are a type of decision trees with a continuous response variable
- We use it to explain *mpg*
 - ▶ These model can handle rows with a few NAs, so we remove only the two rows whose most variables are NAs
- Several packages in R handle regression trees, e.g., *rpart*, *party* and *randomForest*

Regression trees

These videos will help you to understand the algorithm behind trees algorithm.

Video: [Regression trees explained](#)

Video: [Decision trees explained](#)

Fitting a regression trees

- *rpart* function uses also R formula

```
library(rpart)
rt.mpg <- rpart(mpg ~ ., data = mydata2[, 1:6])
```

Plotting a regression tree

```
library(rpart.plot)
rpart.plot(rt.mpg, roundint = FALSE)
```


Plotting a regression tree

