R Bootcamp Assignment: Prediction of the price of second hand cars

Goal

The aim of this project was to build a model that can predict the price of the cars based on certain parameters as precisely as possible.

Authors

Bodo Gruetter, Malik Sogukoglu

Source of dataset

https://www.kaggle.com/orgesleka/used-cars-database (https://www.kaggle.com/orgesleka/used-cars-database)

Description of variables in the dataset

- dateCrawled: Date when data record was crawled
- name: name of the car
- seller: information whether seller was “gewerblich”(commercial) or “privat”(private)
- offerType: information if car was searched (“Gesuch”) or offered (“Angebot”)
- price: Price of car
- abtest: information if auto has been tested (“test”) or the test is still pending (“control”).
- vehicleType: information about the class of the car
- yearOfRegistration: car’s year of registration
- gearbox: information whether the gearbox is automatic or manual.
- powerPS: horsepower of the car
- model: model of the car
- kilometer: kilometers covered by the car
- monthOfRegistration: month of registration of the car
- fuelType: information whether fuel type is “diesel” or “benzin”.
- brand: brand of the car
- notRepairedDamage: information whether car has not repaired damage or not
- dateCreated; information when data was crawled
- nrOfPictures: unknown
- postalCode: information where car is at the moment
- lastSeen: information when car was last seen.

0. Import Packages and prepare environment

```
library(igraph)
library(mgcv)
library(tidyverse)
library(igraphdata)
library(corrplot)
library("ggplot2")
```

Prepare Environment

Set seed for reproducability:

```
set.seed(123)
```

Clear environment:

```
rm(list = ls())
```

Set plotting window to default:

```
par(mfrow = c(1, 1))
```

1. Prepare Data

CSV is read whereby zero values are explicitly marked as NAs:

```
df.cars <- read.csv2(file="autos.csv", sep = ",", header = T, na.strings = c("", " ", "NA"))
```

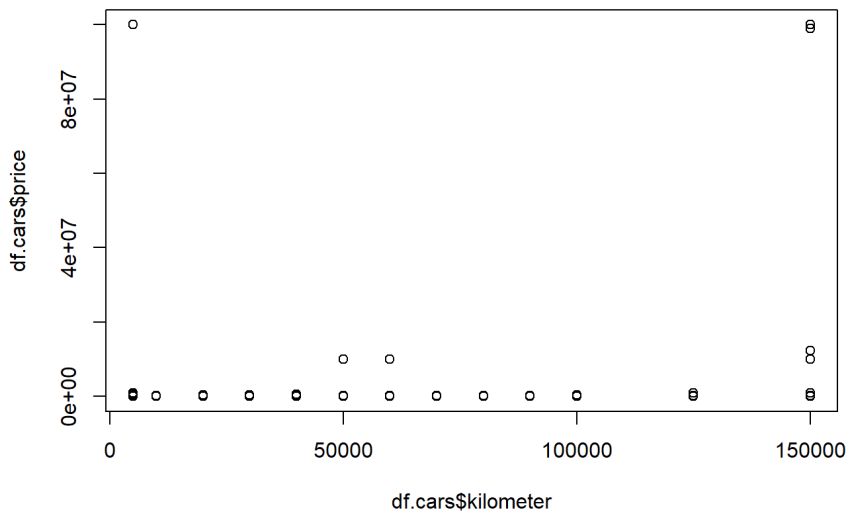
remove NAs:

```
df.cars <- na.omit(df.cars)
```

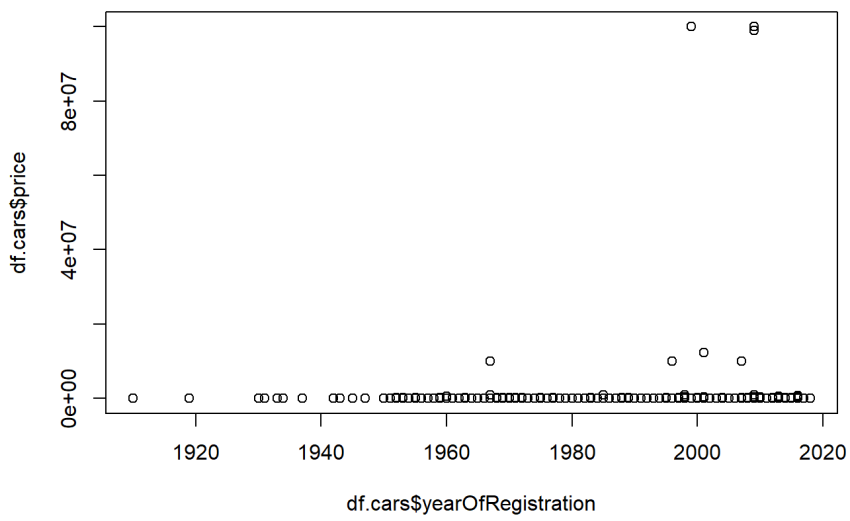
To predict price of cars, it is useful to select “price” as the target variable and “kilometres” as well as “yearOfRegistration” could be good independent variables.

With the following plots it is shown how “kilometres” resp. “yearOfRegistration” stand to the variable “price”.

```
plot(df.cars$price ~df.cars$kilometer)
```




```
plot(df.cars$price ~df.cars$yearOfRegistration)
```



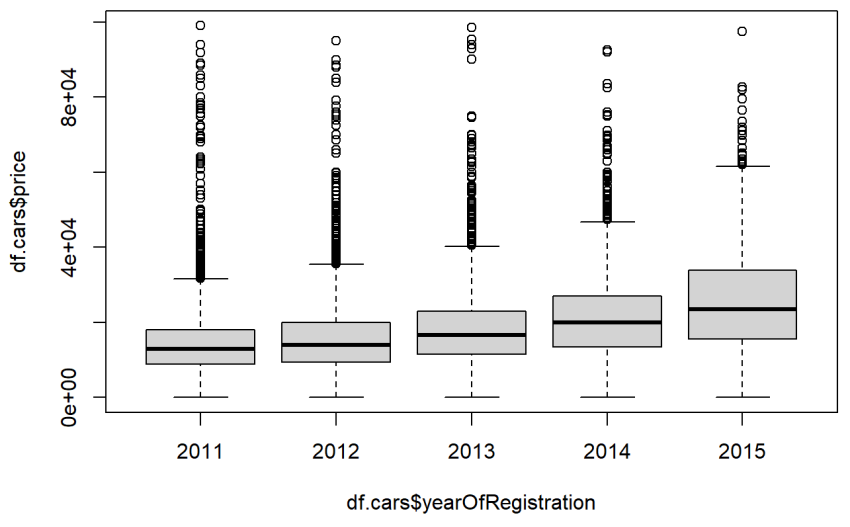
The unrealistic “price” and “yearOfRegistration” data in the plots above clearly show that there are outliers which should be filtered. These unrealistic values are probably caused by failures during crawling.

Filter outliers: - To get rid of these unrealistic values, only cars with a value up to 100000 Euros and only the models between 2010 and 2016 are considered.

```
df.cars <- df.cars %>% filter(yearOfRegistration > 2010,
                              yearOfRegistration < 2016,
                              price < 100000)
```

The following example-boxplot shows that by filtering to the years 2010 - 2016 the graphic became much more visible

```
boxplot(df.cars$price ~df.cars$yearOfRegistration)
```



Columns “lastSeen|nrOfPictures|dateCreated|dateCrawled” are removed since they obviously do not provide useful information. Even if it would influence the price, it would probably be a false correlation.

```
df.cars = subset(df.cars, select = -c(lastSeen,nrOfPictures,dateCreated,dateCrawled) )
```

The “age” of the car would be an important value to calculate the price. Since it was not included in the data set, it is inserted here as a new column: (Note: 2016 is the year of data collection)

```
df.cars$age <- 2016 - df.cars$yearOfRegistration
```

See if/how many “NA”s exist...

```
anyNA(df.cars)
```

```
## [1] FALSE
```

```
apply(df.cars, MARGIN = 2,
      FUN = function(x) {sum(is.na(x))})
```

```
##                name              seller           offerType               price
##                   0                   0                   0                   0
##             abtest         vehicleType  yearOfRegistration             gearbox
##                   0                   0                   0                   0
##            powerPS               model           kilometer monthOfRegistration
##                   0                   0                   0                   0
##           fuelType               brand   notRepairedDamage          postalCode
##                   0                   0                   0                   0
##                age
##                   0
```

...seems there aren't any which is good.

2. Graphical Analysis

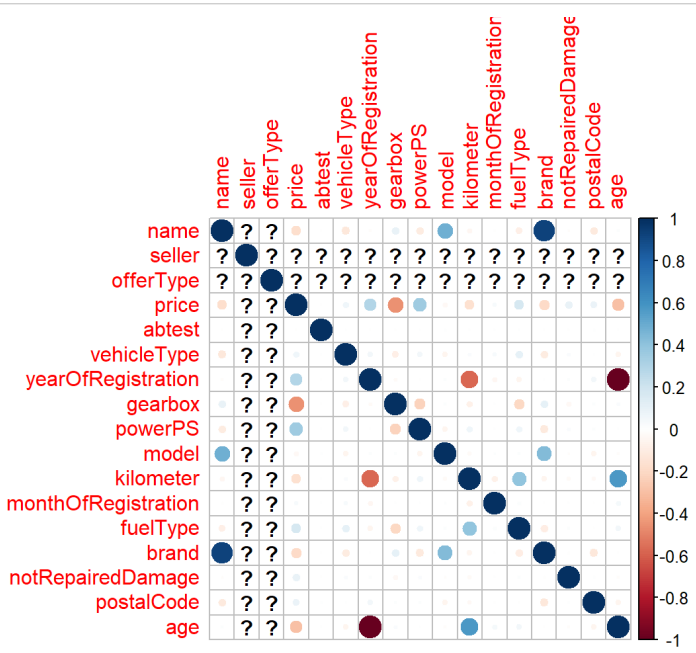
We assumed that "yearOfRegistration" and "kilometer" strongly influence the "price". With the following correlation analyis we want to see which variables actually influence the price.

2.1. Correlation analysis

The following correlation plot shows which variables influence price:

```
predictors = c(colnames(df.cars))
corrplot(cor(data.matrix(df.cars[predictors]) ))
```

```
## Warning in cor(data.matrix(df.cars[predictors])): Standardabweichung ist Null
```



Above, We see that - yearOfRegistration - powerPS - age (calculated from yearOfRegistration) - kilometer - name - brand - fuel type - gearbox - vehicle Type - notRepairedDamage - postalCode have a certain influence on the price. So, there are also variables that we did not think as having a strong effect previously.

In the following analysis we will take a closer look at the effects of the categorical and numerical variables that influence the price according to the correlation plot.

2.2. Categorical variables

...lets start with the categorical values. Those are best explaint with boxplots and Anova-tests.

lets see which are the categorical:

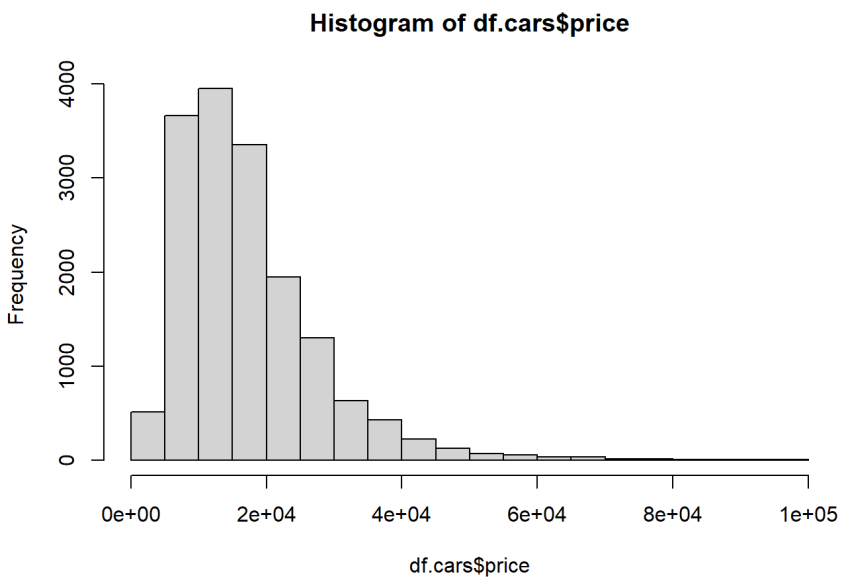
```
sapply(df.cars, class)
```

```
##                name              seller           offerType               price
##         "character"         "character"         "character"           "integer"
##             abtest         vehicleType  yearOfRegistration             gearbox
##         "character"         "character"           "integer"         "character"
##            powerPS               model           kilometer monthOfRegistration
##           "integer"         "character"           "integer"           "integer"
##           fuelType               brand   notRepairedDamage          postalCode
##         "character"         "character"         "character"           "integer"
##                age
##           "numeric"
```

...the independent categorical variables to be considered are - yearOfRegistration - name - brand - fuel type - gearbox - vehicle Type - notRepairedDamage - postalCode - age

Now that all necessary independent variables have been determined, it is important to ensure that the dependent variable "price" follows a normal distribution:

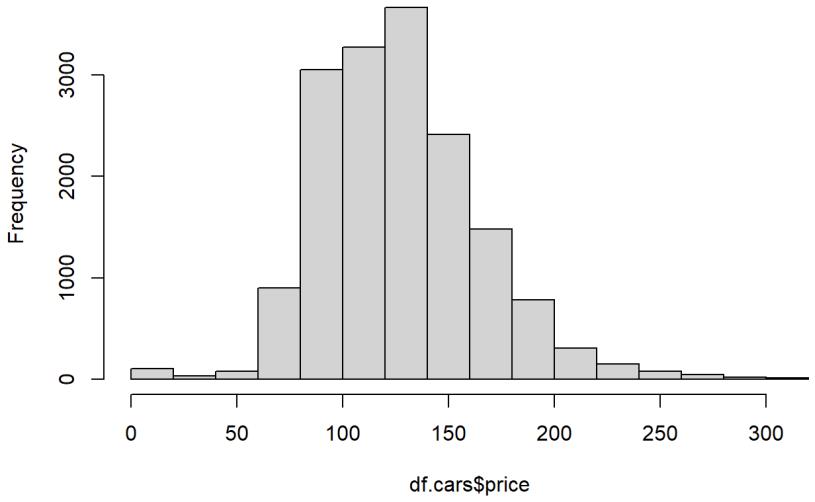
```
hist(df.cars$price)
```



.....and it does not, so the values in

this column are to be squered squared:

```
df.cars$price <- sqrt(df.cars$price)
hist(df.cars$price)
```



... after squaring, price is now

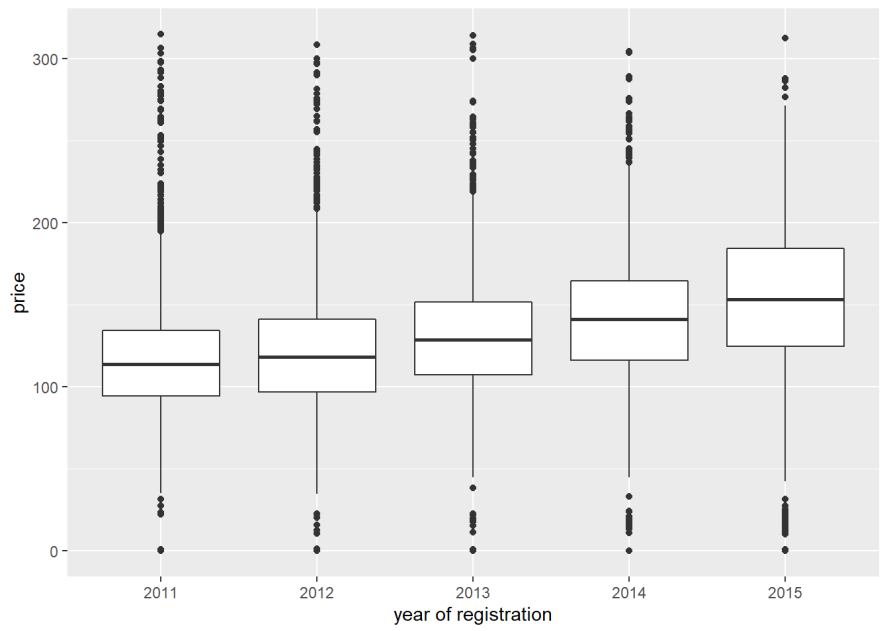
normally distributed and thus suitable for analysis.

lm.cars below is the dummy model that contains every independent variable. This will be used for comparison with other models:

```
lm.cars <- lm(price ~ 1, data = df.cars)
```

year of registration

```
ggplot(data = df.cars, aes(group=yearOfRegistration, y = price, x = as.factor(yearOfRegistration))) +
   geom_boxplot() + xlab("year of registration") + ylab("price")
```




```
lm.yearOfRegistration <- lm(price ~ as.factor(yearOfRegistration), data = df.cars)
anova(lm.cars, lm.yearOfRegistration)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ 1
## Model 2: price ~ as.factor(yearOfRegistration)
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1  16408 23809728
## 2  16404 21643166  4   2166562 410.53 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

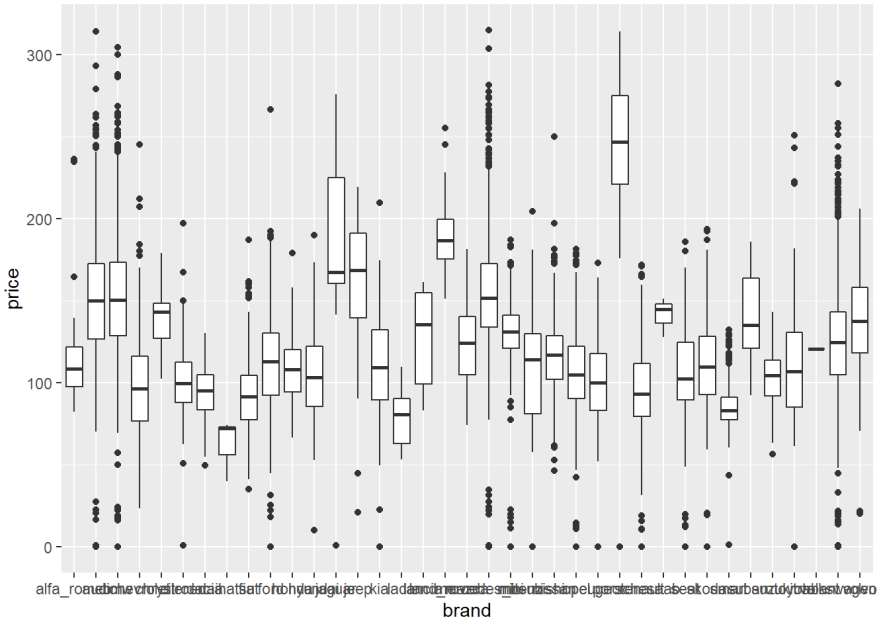
- The boxplot above clearly shows that the year of registration influences the price, because newer cars cost more.
- The anova test shows that there is a strong evidence that at least one year-level influences the price (see p-value) .
- The anova test shows also that the model only having yearOfRegistration as independent variable has smaller residuals than the dummy model (see RSS).

name

Because name has many levels it is not suitable for plotting and Anova-test.

brand

```
ggplot(data = df.cars, aes(group=brand, y = price, x = brand)) +
   geom_boxplot() + xlab("brand") + ylab("price")
```



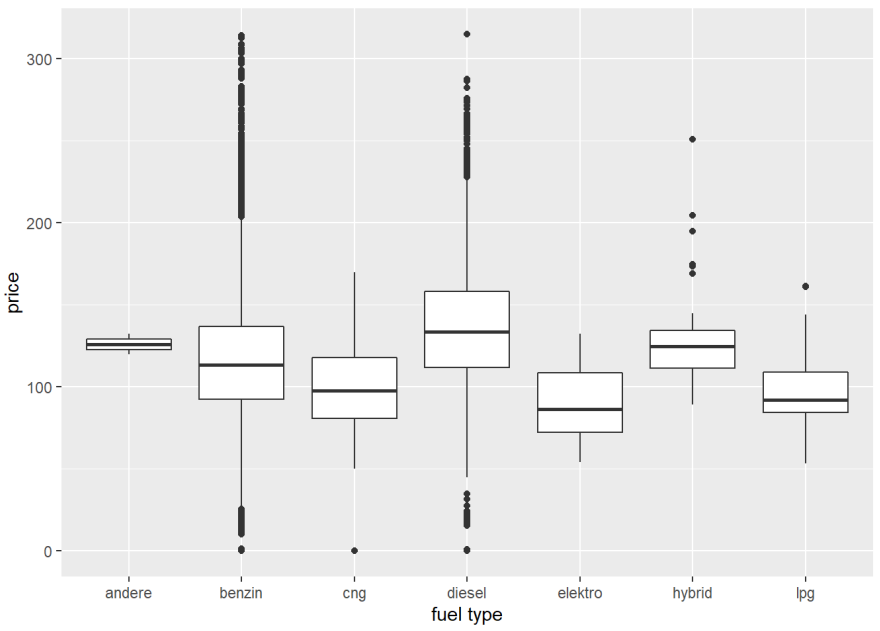

```
lm.brand <- lm(price ~ brand, data = df.cars)
anova(lm.cars, lm.brand)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ 1
## Model 2: price ~ brand
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1  16408 23809728
## 2  16372 14895909 36   8913819 272.14 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The boxplot above clearly shows that the brand influences the price.
- The anova test shows that there is a strong evidence that at least one brand influences the price (see p-value) .
- the anova test shows also that the model only having brand as independent variable has smaller residuals than the dummy model (see RSS).

fuel type

```
ggplot(data = df.cars, aes(group = fuelType, y = price, x = fuelType)) +
   geom_boxplot() + xlab("fuel type") + ylab("price")
```



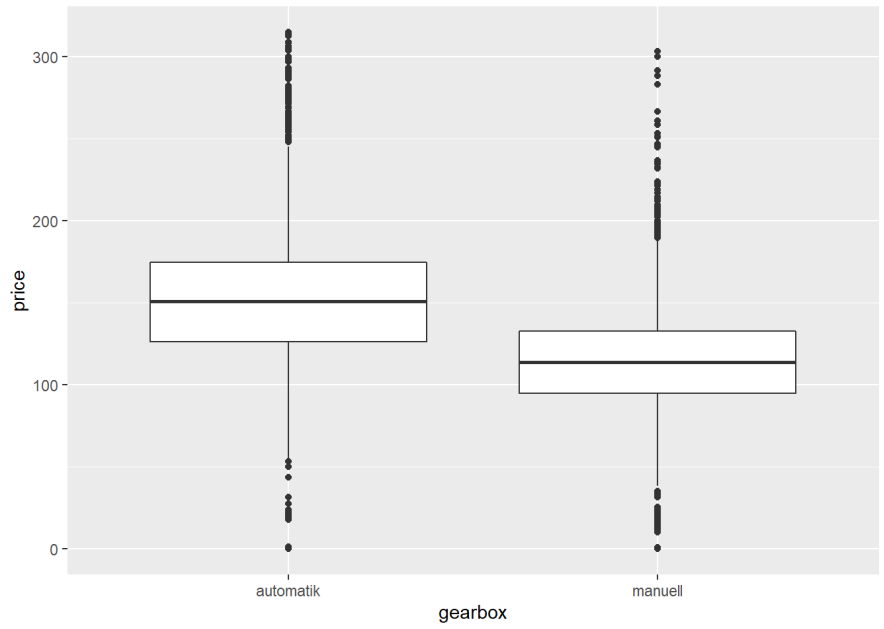

```
lm.fuelType <- lm(price ~ fuelType, data = df.cars)
anova(lm.cars, lm.fuelType)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ 1
## Model 2: price ~ fuelType
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1  16408 23809728
## 2  16402 22495659  6   1314069 159.69 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The boxplot above certainly shows that the fuel type influences the price.
- The anova test shows that there is a strong evidence that at least one fuel type-level influences the price (see p-value) .
- The anova test shows also that the model only having fuelType as independent variable has smaller residuals than the dummy model (see RSS).

gearbox

```
ggplot(data = df.cars, aes(group = gearbox, y = price, x = gearbox)) +
   geom_boxplot() + xlab("gearbox") + ylab("price")
```



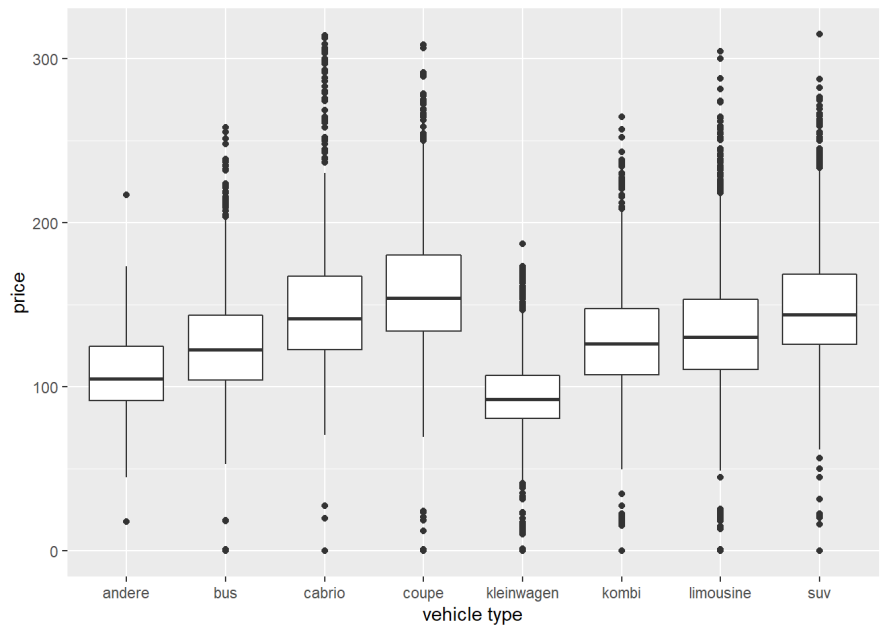

```
lm.gearbox <- lm(price ~ gearbox, data = df.cars)
anova(lm.cars, lm.gearbox)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ 1
## Model 2: price ~ gearbox
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1  16408 23809728
## 2  16407 18907629  1   4902099 4253.8 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- the boxplot above clearly shows that gearbox influences the price.
- the anova test shows that there is a strong evidence that at least one gearbox-type influences the price (see p-value) .
- the anova test shows also that the model only having gearbox as independent variable has smaller residuals than the dummy model (see RSS).

vehicleType

```
ggplot(data = df.cars, aes(y = price, x = vehicleType)) +
   geom_boxplot() + xlab("vehicle type") + ylab("price")
```



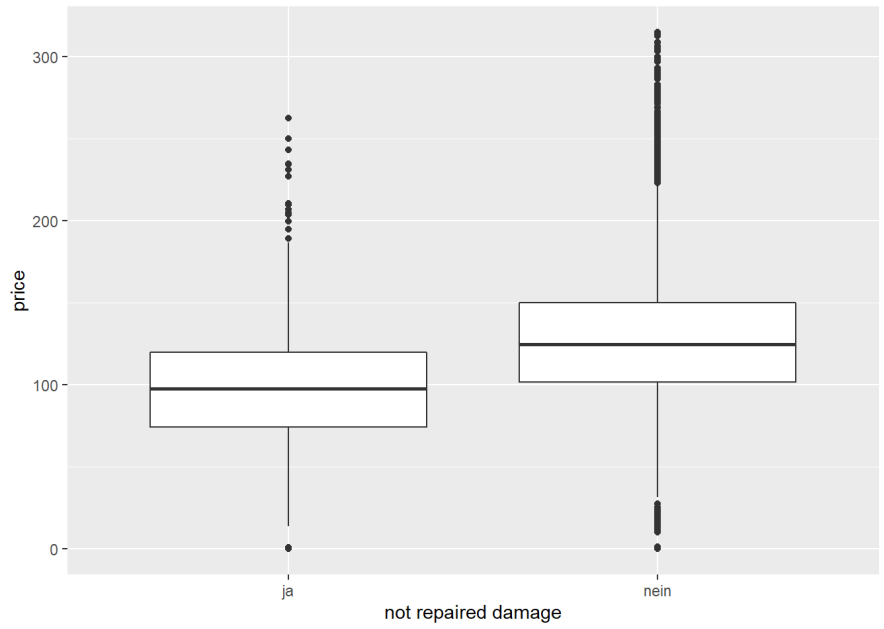

```
lm.vehicleType <- lm(price ~ vehicleType, data = df.cars)
anova(lm.cars, lm.vehicleType)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ 1
## Model 2: price ~ vehicleType
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1  16408 23809728
## 2  16401 17921955  7   5887773 769.73 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- the boxplot above clearly shows that the vehicle type influences the price.
- the anova test shows that there is a strong evidence that at least one vehicle-type influences the price (see p-value) .
- the anova test shows also that the model only having vehicleType as independent variable has smaller residuals than the dummy model (see RSS).

notRepairedDamage

```
ggplot(data = df.cars, aes(y = price, x = notRepairedDamage)) +
   geom_boxplot() + xlab("not repaired damage") + ylab("price")
```




```
lm.notRepairedDamage <- lm(price ~ notRepairedDamage, data = df.cars)
anova(lm.cars, lm.notRepairedDamage)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ 1
## Model 2: price ~ notRepairedDamage
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1  16408 23809728
## 2  16407 23441981  1    367748 257.39 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

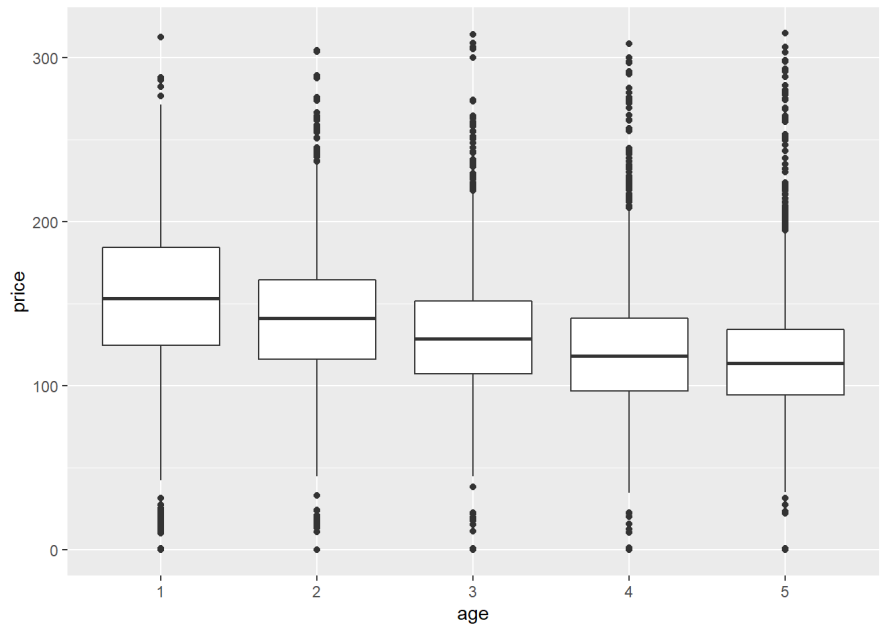
- the boxplot above clearly shows that the information whether car has a non-repaired demage or not influences the price.
- the anova test shows that there is a strong evidence that this information influences the price (see p-value) .
- the anova test shows also that the model only having this information as independent variable has smaller residuals than the dummy model (see RSS).

postalCode

Because postalCode has many levels it is not suitable for plotting and anova test

age

```
ggplot(data = df.cars, aes(y = price, x = as.factor(age))) +
   geom_boxplot() + xlab("age") + ylab("price")
```




```
lm.age <- lm(price ~ as.factor(age), data = df.cars)
anova(lm.cars, lm.age)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ 1
## Model 2: price ~ as.factor(age)
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1  16408 23809728
## 2  16404 21643166  4   2166562 410.53 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The above plot and model (see p-value) show that age has a linear influence on price.

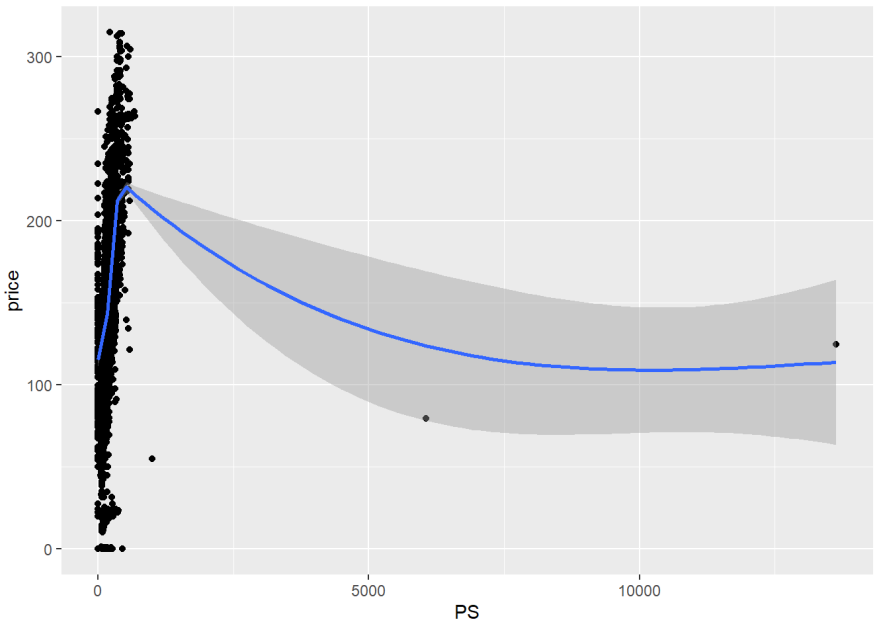
2.3. Numerical variables

Lets go on with the numeric variables which are the following: - powerPS - kilometer Numeric variables are best explained with a classical plot.

powerPS

```
ggplot(data = df.cars, mapping = aes(y = price, x = powerPS)) + geom_point(alpha = 0.4) +
   geom_point()  + xlab("PS") +ylab("price") +geom_smooth(method = "gam")
```

```
## `geom_smooth()` using formula 'y ~ s(x, bs = "cs")'
```



...there is little to recognise.

therefore the x axis is logaritised in the following plot:

```
ggplot(data = df.cars, mapping = aes(y = price, x = powerPS)) + geom_point(alpha = 0.4) +
   geom_point()  + xlab("PS") +ylab("price") + scale_x_log10() +geom_smooth(method = "gam")
```

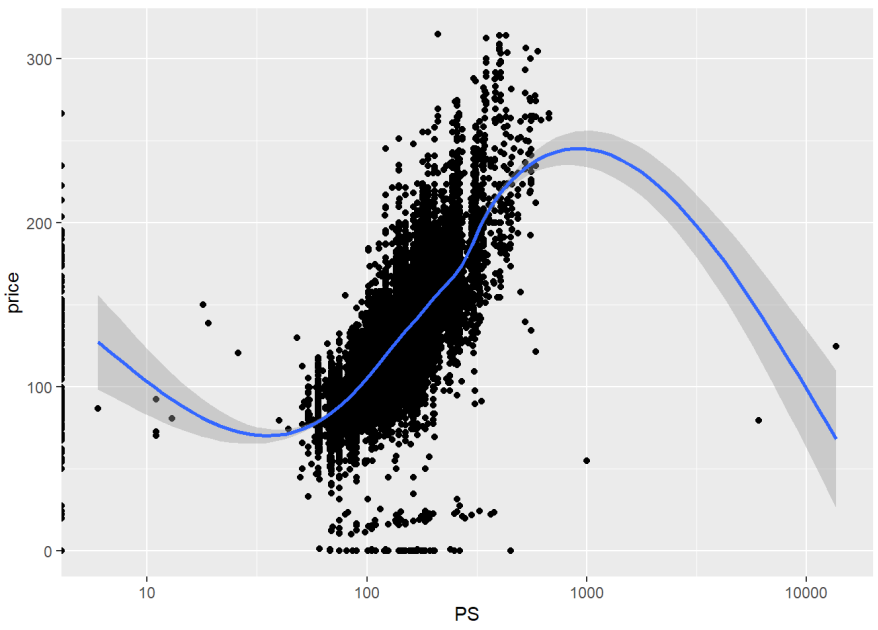
```
## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## `geom_smooth()` using formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 178 rows containing non-finite values (stat_smooth).
```



In the plot above, one can see that

“powerPS” does not show a linear but a quadratic relationship to the “price” ....following GAM-model confirms that where the p-value us significant:

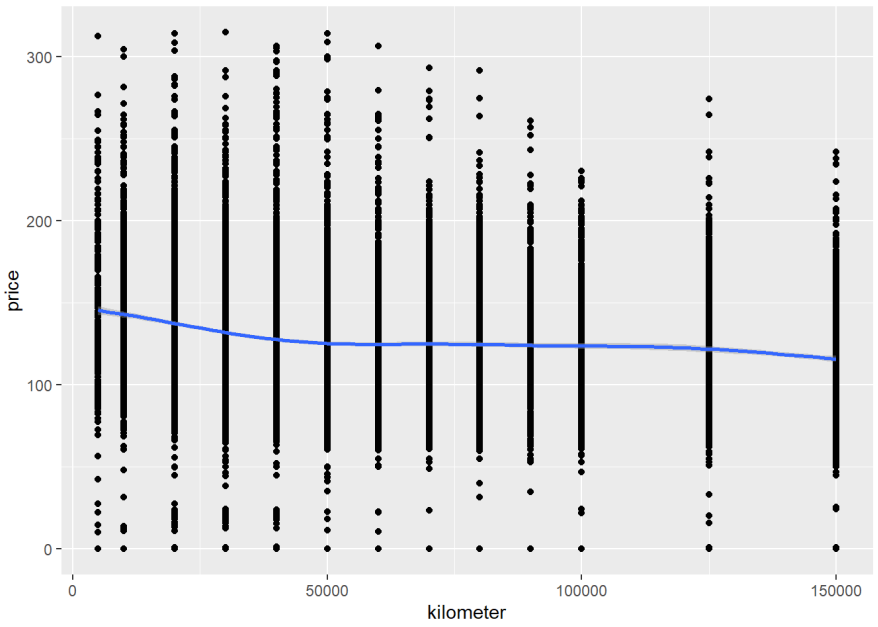
```
gam.powerPS <- gam(price ~ s(powerPS), data = df.cars)
summary(gam.powerPS)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## price ~ s(powerPS)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 127.2022     0.2039   623.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df    F p-value
## s(powerPS) 8.989      9 2055  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.53   Deviance explained =   53%
## GCV = 682.83  Scale est. = 682.42    n = 16409
```

kilometer

```
ggplot(data = df.cars, mapping = aes(y = price, x = kilometer)) + geom_point(alpha = 0.4) +
   geom_point()  + xlab("kilometer") +ylab("price") + geom_smooth(method = "gam")
```

```
## `geom_smooth()` using formula 'y ~ s(x, bs = "cs")'
```



In plot above, one can see that kilometres seems to have a linear effect on price. ...following model should confirm that:

```
lm.kilometer <- lm(price ~ kilometer, data = df.cars)
summary(lm.kilometer)
```

```
##
## Call:
## lm(formula = price ~ kilometer, data = df.cars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -136.002  -25.975   -2.647   21.446  184.464
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.367e+02  5.473e-01  249.84   <2e-16 ***
## kilometer   -1.471e-04  7.125e-06  -20.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37.61 on 16407 degrees of freedom
## Multiple R-squared:  0.02532,    Adjusted R-squared:  0.02526
## F-statistic: 426.2 on 1 and 16407 DF,  p-value: < 2.2e-16
```

...and indeed the p-value shows that “kilometer” has a linear effect on the price.

2.3. Interaction analysis

In this section, some analyses regarding possible interactions between an independent categorical variables and a numerical variable are processed.

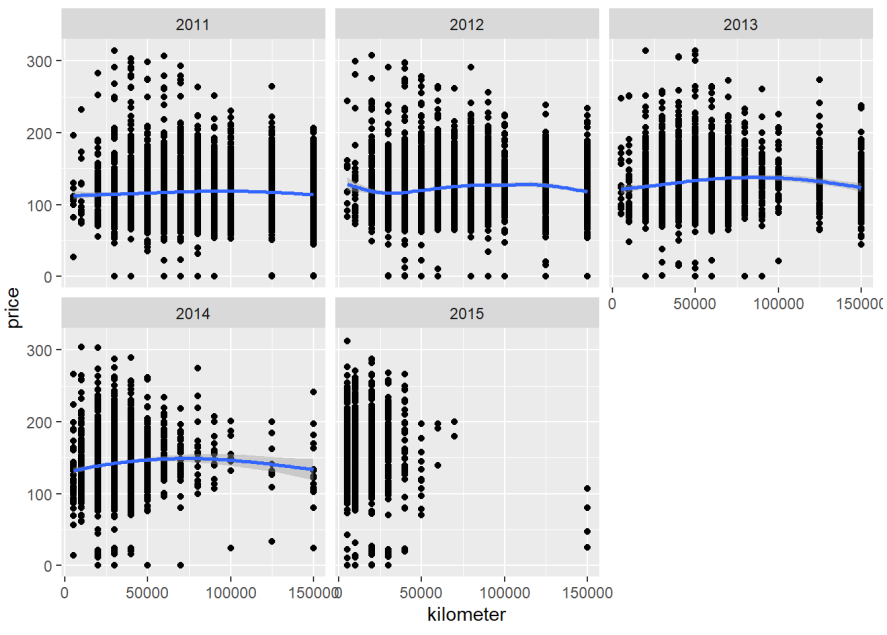
Based on information from the correlation plot above, there could be interactions between the following variables: - factor(yearOfRegistration) : kilometer - kilometer : fuelType - kilometer : gearbox

Example plot to see interaction “yearOfRegistration : kilometer”

```
qplot(y = price, x = kilometer, data = df.cars, facets = ~ as.factor(yearOfRegistration)) + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```



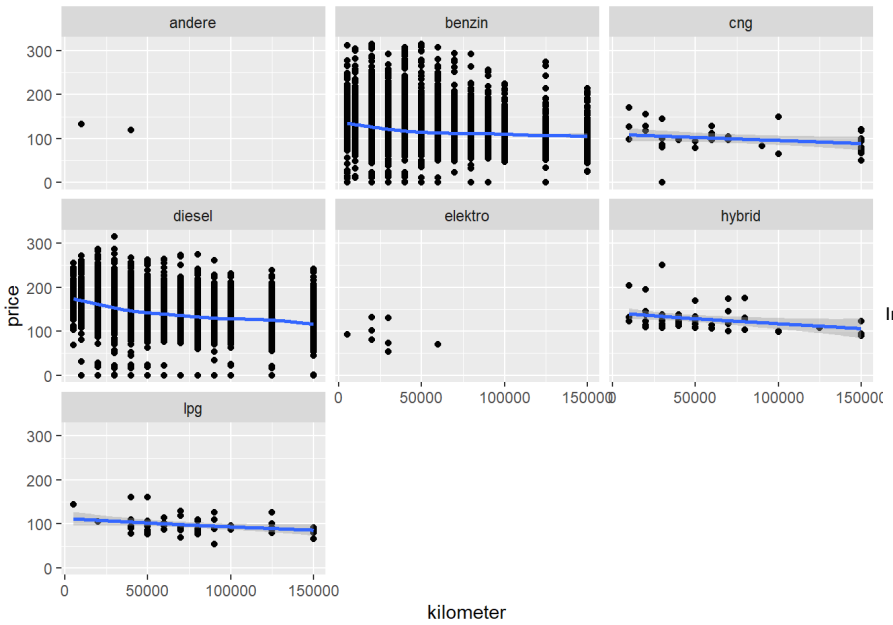
Example plot to see interaction “kilometer : fuelType”

```
qplot(y = price, x = kilometer, data = df.cars, facets = ~ fuelType) + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.

## Warning: Computation failed in `stat_smooth()`:
## x has insufficient unique values to support 10 knots: reduce k.
```

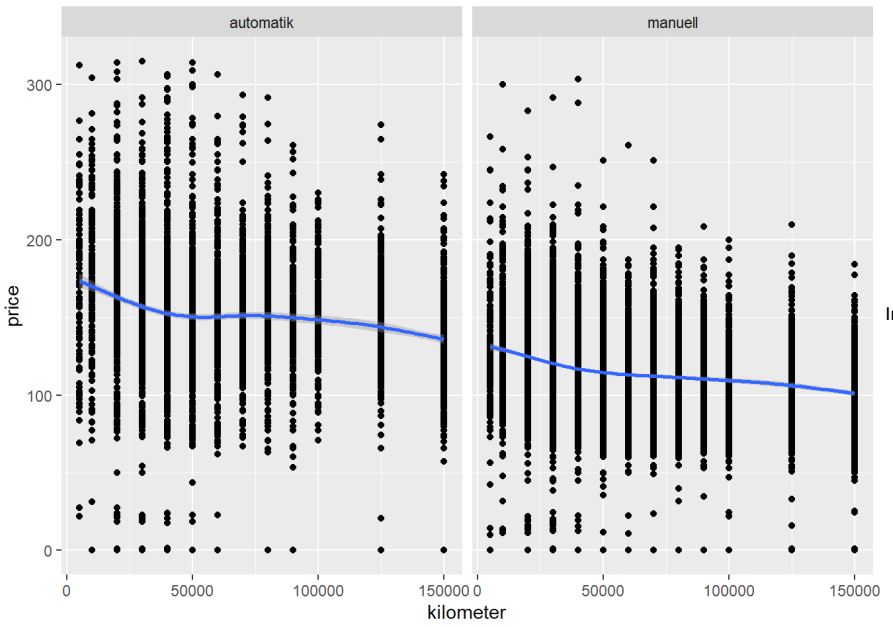


In both graphs above, a clear interaction is to be recognized, since between the categories clear differences are to be determined.

Example plot to see interaction “kilometer : gearbox”

```
qplot(y = price, x = kilometer, data = df.cars, facets = ~ gearbox) + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



In the graph just above, one can see a slight interaction effect between kilometer and gearbox, since there are no recognizable differences between categories of gearbox.

Now let’s look at the interactions with the help of linear models...

Example linear model for interaction “yearOfRegistration : kilometer”

```
lm.cars.interaction.1 <- lm(df.cars$price ~ df.cars$kilometer * as.factor(df.cars$yearOfRegistration))
summary(lm.cars.interaction.1)
```

```
##
## Call:
## lm(formula = df.cars$price ~ df.cars$kilometer * as.factor(df.cars$yearOfRegistration))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -155.340  -24.551   -2.736   20.118  197.219
##
## Coefficients:
##                                                            Estimate
## (Intercept)                                               1.177e+02
## df.cars$kilometer                                        -9.637e-06
## as.factor(df.cars$yearOfRegistration)2012                 1.780e+00
## as.factor(df.cars$yearOfRegistration)2013                 1.147e+01
## as.factor(df.cars$yearOfRegistration)2014                 2.006e+01
## as.factor(df.cars$yearOfRegistration)2015                 3.829e+01
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2012  4.335e-05
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2013  5.577e-05
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2014  1.320e-04
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2015 -1.239e-04
##                                                           Std. Error t value
## (Intercept)                                                1.222e+00  96.311
## df.cars$kilometer                                          1.255e-05  -0.768
## as.factor(df.cars$yearOfRegistration)2012                  1.700e+00   1.047
## as.factor(df.cars$yearOfRegistration)2013                  1.815e+00   6.319
## as.factor(df.cars$yearOfRegistration)2014                  1.903e+00  10.543
## as.factor(df.cars$yearOfRegistration)2015                  2.118e+00  18.080
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2012  1.913e-05   2.267
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2013  2.513e-05   2.220
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2014  4.040e-05   3.269
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2015  8.313e-05  -1.491
##                                                           Pr(>|t|)
## (Intercept)                                                < 2e-16 ***
## df.cars$kilometer                                          0.44252
## as.factor(df.cars$yearOfRegistration)2012                  0.29495
## as.factor(df.cars$yearOfRegistration)2013                  2.7e-10 ***
## as.factor(df.cars$yearOfRegistration)2014                  < 2e-16 ***
## as.factor(df.cars$yearOfRegistration)2015                  < 2e-16 ***
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2012  0.02343 *
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2013  0.02647 *
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2014  0.00108 **
## df.cars$kilometer:as.factor(df.cars$yearOfRegistration)2015  0.13602
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.3 on 16399 degrees of freedom
## Multiple R-squared:  0.09229,    Adjusted R-squared:  0.09179
## F-statistic: 185.3 on 9 and 16399 DF,  p-value: < 2.2e-16
```

The linear model above confirms a strong interaction between kilometers and the year 2014.

Example linear model for interaction “fuelType : kilometer”

```
lm.cars.interaction.2 <- lm(df.cars$price ~ df.cars$kilometer * df.cars$fuelType)
summary(lm.cars.interaction.2)
```

```
##
## Call:
## lm(formula = df.cars$price ~ df.cars$kilometer * df.cars$fuelType)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -157.861  -23.038   -3.625   18.524  196.031
##
## Coefficients:
##                                         Estimate Std. Error t value
## (Intercept)                             1.365e+02  4.874e+01   2.801
## df.cars$kilometer                      -4.235e-04  1.672e-03  -0.253
## df.cars$fuelTypebenzin                 -7.826e+00  4.875e+01  -0.161
## df.cars$fuelTypecng                    -2.288e+01  4.984e+01  -0.459
## df.cars$fuelTypediesel                  2.442e+01  4.875e+01   0.501
## df.cars$fuelTypeelektro                -2.878e+01  5.526e+01  -0.521
## df.cars$fuelTypehybrid                  7.236e+00  4.952e+01   0.146
## df.cars$fuelTypelpg                    -2.237e+01  5.017e+01  -0.446
## df.cars$kilometer:df.cars$fuelTypebenzin  2.059e-04  1.672e-03   0.123
## df.cars$kilometer:df.cars$fuelTypecng     2.353e-04  1.676e-03   0.140
## df.cars$kilometer:df.cars$fuelTypediesel  1.153e-04  1.672e-03   0.069
## df.cars$kilometer:df.cars$fuelTypeelektro -1.735e-04  1.875e-03  -0.093
## df.cars$kilometer:df.cars$fuelTypehybrid  1.345e-04  1.678e-03   0.080
## df.cars$kilometer:df.cars$fuelTypelpg     2.146e-04  1.677e-03   0.128
##                                        Pr(>|t|)
## (Intercept)                              0.0051 **
## df.cars$kilometer                        0.8000
## df.cars$fuelTypebenzin                   0.8725
## df.cars$fuelTypecng                      0.6462
## df.cars$fuelTypediesel                   0.6164
## df.cars$fuelTypeelektro                  0.6025
## df.cars$fuelTypehybrid                   0.8838
## df.cars$fuelTypelpg                      0.6557
## df.cars$kilometer:df.cars$fuelTypebenzin  0.9020
## df.cars$kilometer:df.cars$fuelTypecng     0.8883
## df.cars$kilometer:df.cars$fuelTypediesel  0.9450
## df.cars$kilometer:df.cars$fuelTypeelektro  0.9263
## df.cars$kilometer:df.cars$fuelTypehybrid  0.9361
## df.cars$kilometer:df.cars$fuelTypelpg     0.8982
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35.47 on 16395 degrees of freedom
## Multiple R-squared:  0.1339, Adjusted R-squared:  0.1332
## F-statistic: 194.9 on 13 and 16395 DF,  p-value: < 2.2e-16
```

Between kilometer and fuelType there is actually no interaction according to the linear model just above.

Example linear model for interaction “gearbox : kilometer”

```
lm.cars.interaction.3 <- lm(df.cars$price ~ df.cars$kilometer * df.cars$gearbox)
summary(lm.cars.interaction.3)
```

```
##
## Call:
## lm(formula = df.cars$price ~ df.cars$kilometer * df.cars$gearbox)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -162.212  -20.368   -0.292   19.070  184.317
##
## Coefficients:
##                                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                               1.640e+02  8.344e-01 196.601   <2e-16
## df.cars$kilometer                        -1.831e-04  1.024e-05 -17.883   <2e-16
## df.cars$gearboxmanuell                   -3.824e+01  1.024e+00 -37.350   <2e-16
## df.cars$kilometer:df.cars$gearboxmanuell  8.952e-06  1.299e-05   0.689    0.491
##
## (Intercept)                              ***
## df.cars$kilometer                        ***
## df.cars$gearboxmanuell                   ***
## df.cars$kilometer:df.cars$gearboxmanuell
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33.16 on 16405 degrees of freedom
## Multiple R-squared:  0.2426, Adjusted R-squared:  0.2425
## F-statistic:  1752 on 3 and 16405 DF,  p-value: < 2.2e-16
```

According to the linear model just above, between kilometer and gearbox there is no interaction.

3. Modelling

3.1 The calculation of R-Squared

For the comparison of our models we will use the r-squared value. The R-squared value describes how much of the variation of price is explained by the modelled predictors.

Because it is not possible to access this value using the GAM function in the Summary, we have written a function which calculates and returns the R-Squared. So we don’t always have to output the whole content of Summary.

```
getR2 <- function(gam){
   R2 <- 1-((sum(residuals(gam)^2))/
               (sum((gam$y - mean(gam$y))^2)))
   return(paste("R-Squared: ", R2))
}
```

3.2 The starting model

We have now examined a wide variety of variables and their effects on the target variable *price* using a variety of graphs. Now we want to develop a good model from these findings and compare it with two models of different degrees of complexity. To achieve this we must first define a starting model that we can further develop.

In the graphical analysis the following categorical variables have shown significant effects on the price of the vehicles: yearOfRegistration, age, brand, fuelType, gearbox, vehicleType and notRepairedDamage. The notRepairedDamage improves the model only minimally. Therefore this factor is not considered in our starting model.

The numeric variables: powerPS and kilometers show significant effects on the price. While the effects of kilometer and age can be considered linear, PS shows a cubic effect.

This results in the following starting model:

price ~ age + brand + fuelType + gearbox + vehicleType + s(powerPS) + kilometer

In R we build the following model:

```
starting.model.1 <- price ~ as.factor(age) + brand + fuelType + gearbox + vehicleType +
   s(powerPS) + kilometer

gam.starting.model.1 <- gam(starting.model.1, data = df.cars)

summary(gam.starting.model.1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## price ~ as.factor(age) + brand + fuelType + gearbox + vehicleType +
##     s(powerPS) + kilometer
##
## Parametric coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1.567e+02  1.406e+01  11.149  < 2e-16 ***
## as.factor(age)2       -6.449e+00  6.709e-01  -9.612  < 2e-16 ***
## as.factor(age)3       -1.031e+01  6.661e-01 -15.473  < 2e-16 ***
## as.factor(age)4       -1.626e+01  6.667e-01 -24.384  < 2e-16 ***
## as.factor(age)5       -2.157e+01  6.903e-01 -31.250  < 2e-16 ***
## brandaudi              1.596e+01  2.646e+00   6.034 1.63e-09 ***
## brandbmw               1.190e+01  2.655e+00   4.481 7.50e-06 ***
## brandchevrolet        -2.033e+01  3.068e+00  -6.626 3.55e-11 ***
## brandchrysler         -1.486e+01  8.242e+00  -1.803 0.071394 .
## brandcitroen          -1.070e+01  2.933e+00  -3.646 0.000267 ***
## branddacia            -1.763e+01  3.029e+00  -5.822 5.93e-09 ***
## branddaihatsu         -2.178e+01  1.135e+01  -1.920 0.054875 .
## brandfiat             -1.041e+01  2.796e+00  -3.724 0.000197 ***
## brandford             -7.539e+00  2.685e+00  -2.808 0.004994 **
## brandhonda            -4.205e+00  3.386e+00  -1.242 0.214312
## brandhyundai          -1.060e+01  2.781e+00  -3.811 0.000139 ***
## brandjaguar            9.934e+00  5.034e+00   1.973 0.048463 *
## brandjeep              1.066e+00  3.624e+00   0.294 0.768531
## brandkia              -1.076e+01  2.876e+00  -3.743 0.000183 ***
## brandlada             -3.411e+01  6.934e+00  -4.920 8.75e-07 ***
## brandlancia           -9.731e+00  8.245e+00  -1.180 0.237915
## brandland_rover        3.725e+01  3.648e+00  10.212  < 2e-16 ***
## brandmazda            -2.702e+00  2.977e+00  -0.907 0.364187
## brandmercedes_benz     1.518e+01  2.667e+00   5.692 1.28e-08 ***
## brandmini              7.158e+00  2.795e+00   2.561 0.010448 *
## brandmitsubishi       -1.063e+01  3.553e+00  -2.992 0.002780 **
## brandnissan           -2.364e+00  2.846e+00  -0.831 0.406064
## brandopel             -8.106e+00  2.673e+00  -3.033 0.002427 **
## brandpeugeot          -1.042e+01  2.814e+00  -3.704 0.000213 ***
## brandporsche           6.200e+01  3.217e+00  19.272  < 2e-16 ***
## brandrenault          -1.341e+01  2.756e+00  -4.867 1.14e-06 ***
## brandsaab              1.742e+01  1.134e+01   1.536 0.124641
## brandseat             -5.756e+00  2.754e+00  -2.090 0.036645 *
## brandskoda            -3.301e+00  2.728e+00  -1.210 0.226369
## brandsmart            -1.726e+01  2.900e+00  -5.952 2.70e-09 ***
## brandsubaru            3.394e+00  5.210e+00   0.651 0.514777
## brandsuzuki           -9.079e+00  3.169e+00  -2.865 0.004178 **
## brandtoyota           -6.013e+00  2.922e+00  -2.058 0.039615 *
## brandtrabant          -7.098e+00  1.933e+01  -0.367 0.713428
## brandvolkswagen        4.387e+00  2.629e+00   1.669 0.095220 .
## brandvolvo             5.205e+00  3.147e+00   1.654 0.098089 .
## fuelTypebenzin        -8.356e+00  1.353e+01  -0.617 0.536956
## fuelTypecng           -1.522e+01  1.390e+01  -1.095 0.273571
## fuelTypediesel         1.030e+00  1.354e+01   0.076 0.939330
## fuelTypeelektro       -8.965e+00  1.516e+01  -0.591 0.554297
## fuelTypehybrid         1.006e+01  1.383e+01   0.727 0.466980
## fuelTypelpg           -2.722e+00  1.380e+01  -0.197 0.843602
## gearboxmanuell        -5.599e+00  4.045e-01 -13.843  < 2e-16 ***
## vehicleTypebus         9.594e+00  2.696e+00   3.559 0.000373 ***
## vehicleTypecabrio      1.194e+01  2.737e+00   4.363 1.29e-05 ***
## vehicleTypecoupe       5.797e+00  2.754e+00   2.105 0.035341 *
## vehicleTypekleinwagen -6.778e+00  2.689e+00  -2.521 0.011712 *
## vehicleTypekombi       1.199e+00  2.686e+00   0.446 0.655336
## vehicleTypelimousine  -2.544e-01  2.681e+00  -0.095 0.924418
## vehicleTypesuv         1.340e+01  2.712e+00   4.942 7.80e-07 ***
## kilometer             -1.977e-04  5.139e-06 -38.463  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df    F p-value
## s(powerPS) 6.909  6.996 1060  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 63/65
## R-sq.(adj) =  0.748   Deviance explained = 74.9%
## GCV = 366.78  Scale est. = 365.37    n = 16409
```

The R-squared-value is with over 70% quite high. Our starting model seems to be a relatively good model. When we look at the individual predictors, something catches our eye relatively quickly. In this constellation, no level of fuelType has a significant influence on price. Since our goal is to develop a model that is as good as possible so that all variables have a significant influence on price, we decide to remove this factor from the model.

For reasons of reading flow, we will from now on output only the p-values and the R-squared value.

```
gam.starting.model.2 <- update(gam.starting.model.1, . ~ . - fuelType)

summary(gam.starting.model.2 )$p.table[,4]
```

```
##           (Intercept)       as.factor(age)2       as.factor(age)3
##          0.000000e+00          7.805554e-23          1.492040e-61
##       as.factor(age)4       as.factor(age)5             brandaudi
##         1.388304e-149         2.428615e-245         1.923218e-10
##             brandbmw        brandchevrolet         brandchrysler
##          5.672087e-07          6.127468e-11          7.913045e-02
##          brandcitroen            branddacia         branddaihatsu
##          4.889334e-04          3.995743e-09          4.801660e-02
##             brandfiat             brandford            brandhonda
##          1.372117e-04          5.457458e-03          1.525921e-01
##          brandhyundai           brandjaguar             brandjeep
##          6.071354e-05          1.036304e-02          5.033711e-01
##             brandkia             brandlada           brandlancia
##          1.300258e-04          1.004800e-07          2.308065e-01
##       brandland_rover            brandmazda    brandmercedes_benz
##          1.420456e-26          2.629343e-01          1.196437e-08
##             brandmini       brandmitsubishi           brandnissan
##          1.929788e-02          5.003837e-03          3.821869e-01
##             brandopel          brandpeugeot          brandporsche
##          1.584304e-03          3.158300e-04          5.632798e-82
##          brandrenault             brandsaab             brandseat
##          5.768648e-07          7.936349e-02          2.287768e-02
##            brandskoda            brandsmart           brandsubaru
##          1.485908e-01          3.751839e-10          6.620556e-01
##           brandsuzuki           brandtoyota          brandtrabant
##          7.672216e-04          9.253507e-02          5.144075e-01
##       brandvolkswagen             brandvolvo        gearboxmanuell
##          8.875430e-02          2.618103e-02          9.363580e-64
##         vehicleTypebus     vehicleTypecabrio      vehicleTypecoupe
##          1.294274e-04          1.130035e-03          1.479571e-01
## vehicleTypekleinwagen      vehicleTypekombi  vehicleTypelimousine
##          5.801421e-04          5.387738e-01          6.621036e-01
##         vehicleTypesuv             kilometer
##          2.054401e-07         1.009787e-212
```

```
summary(gam.starting.model.2 )$s.table
```

```
##                edf   Ref.df        F p-value
## s(powerPS) 6.918584 6.996992 977.2541       0
```

```
getR2(gam.starting.model.2)
```

```
## [1] "R-Squared:  0.739694698030594"
```

The result looks much better. If we now look at the R-squared-value, we can see that it gets only slightly worse by omitting fuelType. If we look now at vehicleType, we see that only three levels have a highly significant effect on the price. So we check if omitting this variable changes the R-Squared strongly.

```
gam.starting.model.3 <- update(gam.starting.model.2, . ~ . - vehicleType)

summary(gam.starting.model.3 )$p.table[,4]
```

```
##         (Intercept)    as.factor(age)2    as.factor(age)3    as.factor(age)4
##        0.000000e+00       3.908663e-22       1.249426e-61      8.073163e-148
##    as.factor(age)5          brandaudi           brandbmw     brandchevrolet
##       5.261804e-240       1.924011e-12       6.568418e-09       9.472524e-07
##      brandchrysler       brandcitroen         branddacia      branddaihatsu
##       4.280561e-01       3.135119e-01       1.840633e-01       7.176574e-02
##          brandfiat          brandford         brandhonda       brandhyundai
##       1.499675e-01       6.486125e-01       7.655526e-01       1.708622e-01
##        brandjaguar          brandjeep           brandkia          brandlada
##       5.996353e-03       2.134398e-06       1.831257e-01       1.640477e-02
##        brandlancia    brandland_rover         brandmazda brandmercedes_benz
##       6.391022e-01       1.657505e-47       1.918723e-01       5.273682e-14
##          brandmini    brandmitsubishi        brandnissan          brandopel
##       5.074599e-04       2.875623e-01       8.458290e-02       1.627687e-01
##       brandpeugeot       brandporsche       brandrenault          brandsaab
##       2.164887e-01      9.393617e-101       5.550295e-03       1.005167e-01
##          brandseat          brandskoda         brandsmart         brandsubaru
##       2.499488e-01       4.979178e-01       1.599162e-05       1.933330e-01
##        brandsuzuki         brandtoyota       brandtrabant    brandvolkswagen
##       3.674711e-01       9.598692e-01       4.986855e-01       9.469319e-05
##         brandvolvo     gearboxmanuell          kilometer
##       3.526859e-05       2.248434e-70     2.829885e-179
```

```
summary(gam.starting.model.3 )$s.table
```

```
##                edf   Ref.df        F p-value
## s(powerPS) 6.930767 6.997835 1357.711       0
```

```
getR2(gam.starting.model.3)
```

```
## [1] "R-Squared:  0.71165472170807"
```

We see that the R-Squared again deteriorates not strongly. What we continue to notice, however, is the strong change in the “brand” factor. One example is the Peugeot brand. This no longer has any significant effect. Besides this brand, the significance values of many other brands have also changed. In the next step we will investigate the effect of omitting this factor on the result of the model hypothesis test.

```
gam.starting.model.4 <- update(gam.starting.model.3, . ~ . - brand)

summary(gam.starting.model.4 )$p.table[,4]
```

```
##     (Intercept) as.factor(age)2 as.factor(age)3 as.factor(age)4 as.factor(age)5
##    0.000000e+00    3.624958e-18    1.089312e-60   5.491963e-139   7.900871e-225
##  gearboxmanuell       kilometer
##   7.532157e-103   5.692991e-106
```

```
summary(gam.starting.model.4 )$s.table
```

```
##                edf   Ref.df        F p-value
## s(powerPS) 6.949206 6.998828 2239.498       0
```

```
getR2(gam.starting.model.4)
```

```
## [1] "R-Squared:  0.63802957205337"
```

```
starting.model <- price ~ as.factor(age) + gearbox +
   s(powerPS) + kilometer
```

The result looks good. All factors show: There is strong evidence that these have an effect on price. The R-Squared is reduced by almost 10%. Since we want to have a model that is as good as possible but also as simple as possible, this is a good compromise for us. The following model results from our model development:

price ~ age + gearbox + s(powerPS) + kilometer

3.3 Two more models

Our starting model is already relatively complex. We have already mapped a polynomial effect of powerPS. In order to be able to make a meaningful comparison, we will create two models in this section, which differ significantly from our starting model. First, we model a very simple linear model with only two predictors and a more complex model with even more factors than in the starting model.

A simple model is quickly built. We simply consider which one factor could have an influence on the price, purely logically without including the results of the graphical analysis. We simply assume that a Porsche brand car is more expensive than a Skoda.

Based on these consideration, we set up the following simple model:

price ~ brand

In R we build the following model:

```
simple.model <- price ~ brand

lm.simple.model <- lm(simple.model, data = df.cars)

summary(lm.simple.model)$r.squared
```

```
## [1] 0.3743772
```

Not surprisingly, this model has a lower R-square value than the more complex initial model.

Now, we would like to setting up a model that is more complex than our final starting model To do this, we simply take our initial starting model, which we had formed using findings from exploratory graphical analysis. Since the graphical analysis did not show any interactions, it makes no sense to implement any. The following model results:

price ~ age + brand + fuelType + vehicleType + s(powerPS) + gearbox + kilometer

In R we build the following model:

```
complex.model <- price ~ as.factor(age) + brand + fuelType + vehicleType +
s(powerPS) + gearbox + kilometer

gam.complex.model <- gam(complex.model, data = df.cars)

getR2(gam.complex.model)
```

```
## [1] "R-Squared:  0.749161792623369"
```

The R-Squared is now correspondingly higher. We are curious if this automatically leads to a better result in the prediction of price.

3.4 A comparison of the models

To compare our models we use the 10-fold cross validation concept. To obtain a meaningful value for the accuracy of the prediction, we split our dataset df.cars into two datasets. A training data set, which contains about 75% of all data and a test data set, which contains the remaining 25% of the data. This ensures that we can train our model on the training data set and then let it make a prediction based on a new, unknown test data set. We train all three models ten times and then make predictions. We use R-Squared as an accuracy indicator, which we have already used to evaluate the models. From the 10 runs we finally take the arithmetic mean.

In our case an error occurred at first. Because certain levels of the categories brand and fuelType have only one value, it can happen that a model is trained on the basis of factors with fewer levels than it is finally predicted. We solve this problem by deleting those levels of the affected factors that occur only once.

```
df.cars <- droplevels(df.cars[!df.cars$brand == 'trabant',])
df.cars <- droplevels(df.cars[!df.cars$brand == 'daihatsu',])
df.cars <- droplevels(df.cars[!df.cars$brand == 'saab',])
df.cars <- droplevels(df.cars[!df.cars$fuelType == 'andere',])

for(i in 1:10){
   df.cars.train.id <- sample(seq_len(nrow(df.cars)),size = floor(0.75*nrow(df.cars)))
   df.cars.train <- df.cars[df.cars.train.id,]
   df.cars.test <- df.cars[-df.cars.train.id,]

   #predict data with starting model
   gam.starting.model.train <- gam(starting.model, data = df.cars.train)
   predicted.starting.model.test <- predict(gam.starting.model.train,
                                            newdata = df.cars.test)
   r.squared.starting.model <- cor(predicted.starting.model.test, df.cars.test$price)^2

   #predict data with simple model
   lm.simple.model.train <- gam(simple.model, data = df.cars.train)
   predicted.simple.model.test <- predict(lm.simple.model.train,
                                          newdata = df.cars.test)
   r.squared.simple.model <- cor(predicted.simple.model.test, df.cars.test$price)^2

   #predict data with complex model
   gam.complex.model.train <- gam(complex.model, data = df.cars.train)
   predicted.complex.model.test <- predict(gam.complex.model.train,
                                          newdata = df.cars.test)
   r.squared.complex.model <- cor(predicted.complex.model.test, df.cars.test$price)^2
}
```

```
mean(r.squared.starting.model)
```

```
## [1] 0.6381819
```

```
mean(r.squared.simple.model)
```

```
## [1] 0.3624076
```

```
mean(r.squared.complex.model)
```

```
## [1] 0.7395839
```

We get good values for all models. The complex model concludes best with an accuracy of almost 78%. Our starting model follows with about 13% less accuracy than the more complex model. If we look at the simple model, we can see how strongly the brand factor influences the price of the vehicles. Based on the results, the decision is still made between our starting model and the complex model. If we only consider the accuracy, we would undoubtedly choose the more complex model. If we also look at the performance, we might also choose the simpler starting model.

4. Further experimential analyses with cars dataset

The analysis has now been completed. In this section we want to show further analyses with this used cars-example, using functions that were demonstrated in the R-Bootcamp course. Also other functions we found in the internet are used here.

4.1. Aggregation functions using streams

Calculate average price for cars of different classes:

```
dfPrizeMean <- df.cars %>%
   group_by(vehicleType) %>%
   summarize(meanPrice = mean(price))
dfPrizeMean
```

```
## # A tibble: 8 x 2
##   vehicleType meanPrice
##   <chr>           <dbl>
## 1 andere          109.
## 2 bus             127.
## 3 cabrio          149.
## 4 coupe           158.
## 5 kleinwagen       94.3
## 6 kombi           128.
## 7 limousine       133.
## 8 suv             149.
```

Calculate average price for cars of different brands:

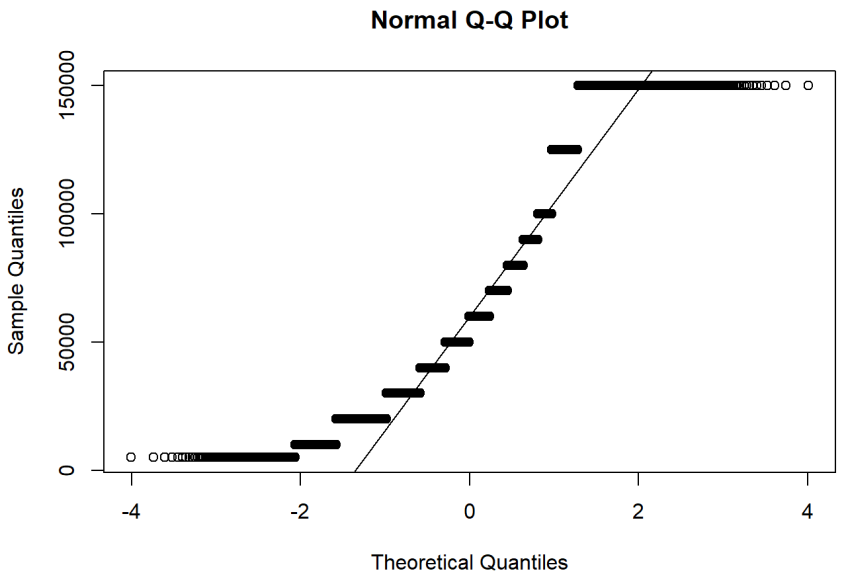
```
dfPrizeMean <- df.cars %>%
   group_by(brand) %>%
   summarize(meanPrice = mean(price))
dfPrizeMean
```

```
## # A tibble: 34 x 2
##    brand      meanPrice
##    <chr>          <dbl>
##  1 alfa_romeo     114.
##  2 audi           151.
##  3 bmw            152.
##  4 chevrolet      101.
##  5 chrysler       140.
##  6 citroen        101.
##  7 dacia           94.1
##  8 fiat            92.3
##  9 ford           113.
## 10 honda          110.
## # ... with 24 more rows
```

4.2. check for normal distribution

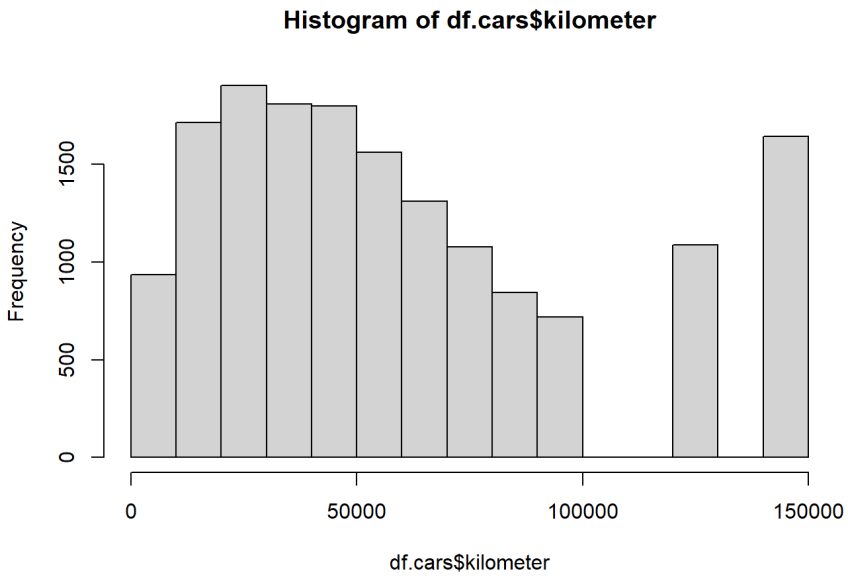
Check normal distribution of residuals of kilometer:

```
qqnorm(df.cars$kilometer)
qqline(df.cars$kilometer)
```



Check normal distribtuion of kilometer:

```
hist(df.cars$kilometer)
```



...the residuals of kilometer seem to be normal distributen whereas kilometer it self seem to be not so.

4.3. Colored plots

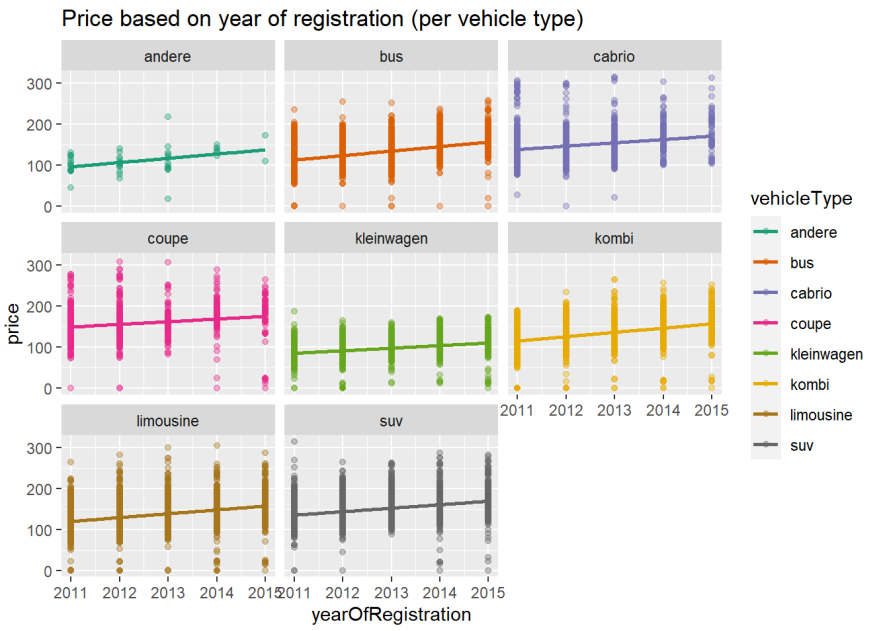
Colored plot: “Price based on Kilometers (per vehicle type)”

```
plot_cars <- ggplot(
   data = df.cars,
   mapping = aes(x = kilometer,
                 y = price,
                 colour = vehicleType)
)  + geom_point(alpha = 0.4) +
   geom_smooth(method = "lm", se=FALSE) +
   scale_color_brewer(type = "qual", palette = "Dark2") +
   facet_wrap( ~ vehicleType) +
   scale_x_log10() +
   ggtitle("Price based on Kilometers (per vehicle type)")
```

Colored plot: “Price based on year of registration (per vehicle type)”

```
plot_cars <- ggplot(
   data = df.cars,
   mapping = aes(x = yearOfRegistration,
                 y = price,
                 colour = vehicleType)
)  + geom_point(alpha = 0.4) +
   geom_smooth(method = "lm", se=FALSE) +
   scale_color_brewer(type = "qual", palette = "Dark2") +
   facet_wrap( ~ vehicleType) +
   scale_x_log10() +
   ggtitle("Price based on year of registration (per vehicle type)")

plot_cars
```



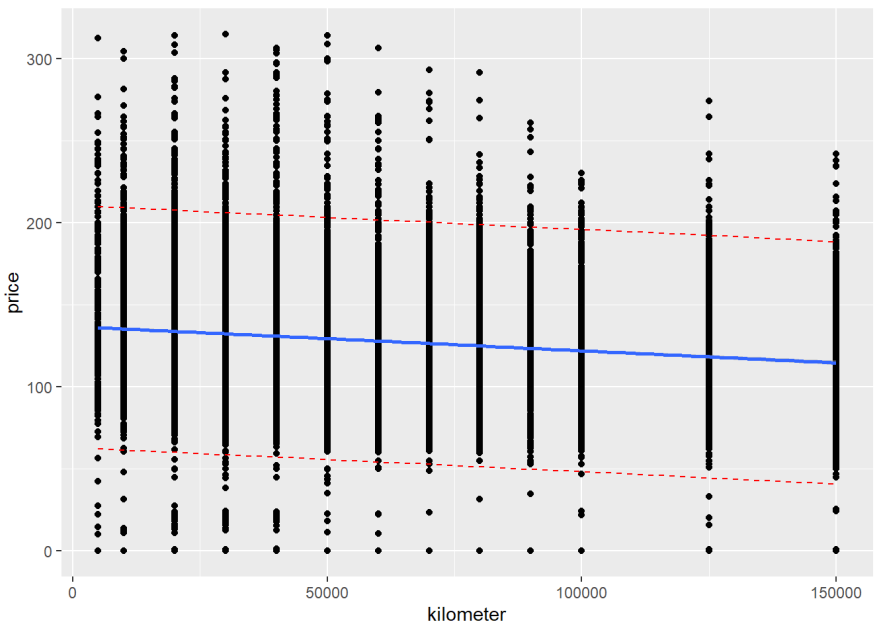
4.4. Predictions

prediction: kilometer predicts price

```
lm.cars.kilometer <- lm(df.cars$price ~df.cars$kilometer)
pred.int <- predict(lm.cars.kilometer, interval = "prediction")
```

```
## Warning in predict.lm(lm.cars.kilometer, interval = "prediction"): predictions on current data refer to _future_ response
s
```

```
mydata <- cbind(df.cars, pred.int)
p <- ggplot(mydata, aes(kilometer, price)) +
   geom_point() +
   stat_smooth(method = lm)
p + geom_line(aes(y = lwr), color = "red", linetype = "dashed")+
   geom_line(aes(y = upr), color = "red", linetype = "dashed")
```

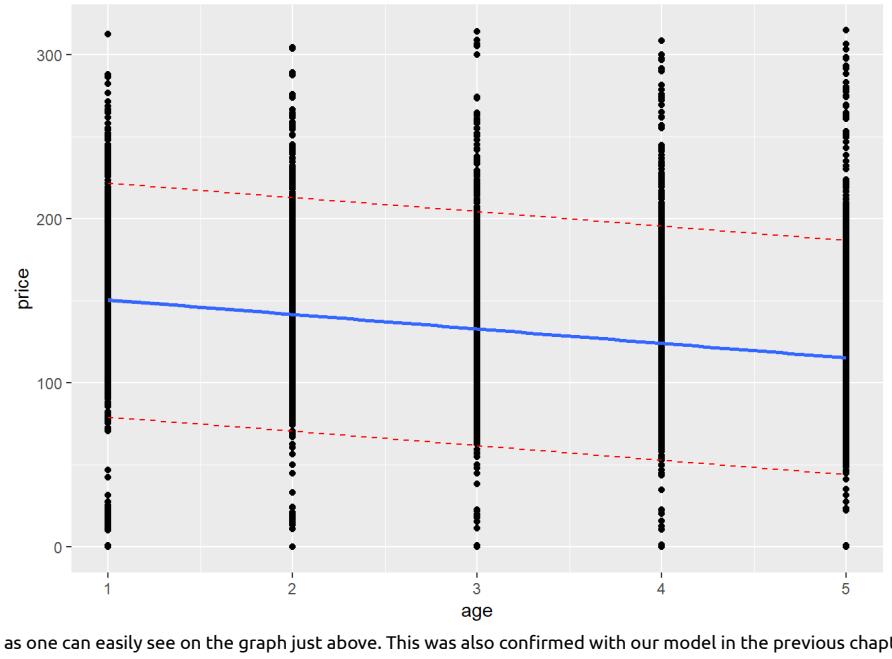


prediction: age predicts price

```
lm.cars.age <- lm(df.cars$price ~df.cars$age)
pred.int <- predict(lm.cars.age, interval = "prediction")
```

```
## Warning in predict.lm(lm.cars.age, interval = "prediction"): predictions on current data refer to _future_ responses
```

```
mydata <- cbind(df.cars, pred.int)
p <- ggplot(mydata, aes(age, price)) +
   geom_point() +
   stat_smooth(method = lm)
p + geom_line(aes(y = lwr), color = "red", linetype = "dashed")+
   geom_line(aes(y = upr), color = "red", linetype = "dashed")
```



Age and kilometers predicted price

as one can easily see on the graph just above. This was also confirmed with our model in the previous chapter (chap. 3.3).

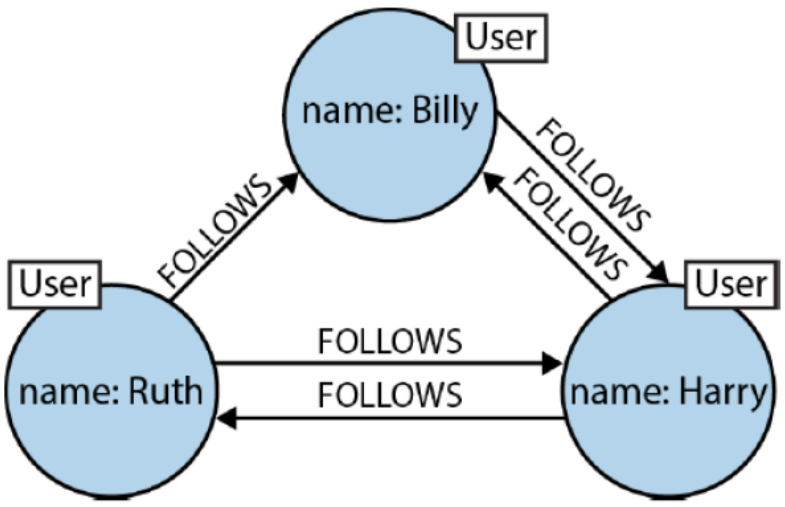
5. The igraph-Package

5.1 An introduction into the graph theory

Using the igraph library, graph algorithms can be easily implemented in R, large graphs can be handled and visualized quickly. Before we can do this, some terms in the context of graphs must be defined. The theoretical basis of graphs is graph theory. It is a branch of mathematics.

A *graph* is a network of relationships between different entities from the real world. These entities are represented in the graph as *nodes*. The nodes have *properties* that describe the corresponding object in the real world. They make it an individual. Nodes are related to each other. They are connected by *edges*.

A simple example Graphs are suitable for displaying data from social networks like Facebook and Twitter. An introductory example will go into this in more detail. The following figure shows a very simple graph with three nodes, five edges and a property name - represented as a key-value construct - in each node. The graph shows a small social network between three users. Billy follows Harry and Harry follows Billy. Ruth and Harry also follow each other. Ruth follows Billy, but Billy does not follow Ruth.



5.2 Graphs in igraph

In igraph graphs are represented with the class “igraph”. Graphs can be directed (D) as seen in the previous example, or they can be undirected (U). Furthermore, graphs can be named (N), weighted (W) - that means that the edges have different weights - and bipartite (B). A bipartite graph has relations between elements of two sets. In contrast to the conventional graph, the relations do not exist from node to node, but from element in set A to element in set B.

In igraph, typically two parameters indicating the size of the graph are specified to form a graph. The number of vertices indicates how many nodes the graph contains and the number of edges indicates how many edges the graph conains. **Attention: in igraph we do not speak of nodes but of vertices!**

A simple example Now we want to recreate the simple example of the small social network between Billy, Ruth and Harry. For this we use the “igraph” package, which must first be imported. In this example we will build a graph from a data.frame. So first we have to save the data in data.frames before we can form the graph. These are two data.frames. “user” includes the users Billy, Ruth and Harry. “relations” contains the relations between the three users.

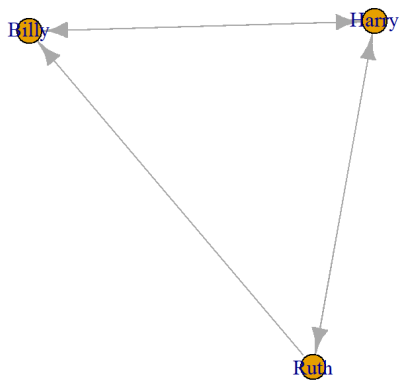
```
users <- data.frame(name=c("Billy", "Harry", "Ruth"))
relations <- data.frame(from=c("Billy", "Harry", "Harry", "Ruth", "Ruth"), to=c("Harry", "Billy", "Ruth", "Harry", "Billy"))

g <- graph_from_data_frame(relations, directed=TRUE, vertices=users)
print(g)
```

```
## IGRAPH a7bc104 DN-- 3 5 --
## + attr: name (v/c)
## + edges from a7bc104 (vertex names):
## [1] Billy->Harry Harry->Billy Harry->Ruth  Ruth ->Harry Ruth ->Billy
```

As the output shows, it is a directed, named graph with 3 nodes and 5 edges. The nodes each have an attribute: name. In the last line, the relations representing the same image as in the figure of the last section are given. Now we ask ourselves the question how we can display the graph visually so that it looks similar to the figure just mentioned.

```
plot(g)
```

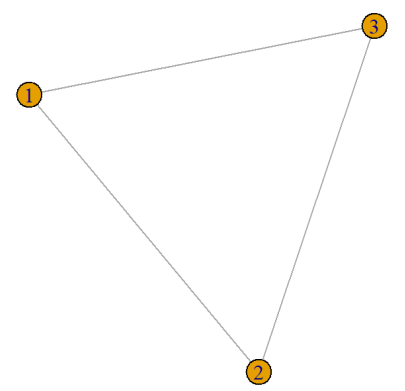


We use a simple function that we have often used to create charts: plot. As we see, the graphic corresponds to our little social network between Billy, Ruth and Harry.

5.3 Some further examples

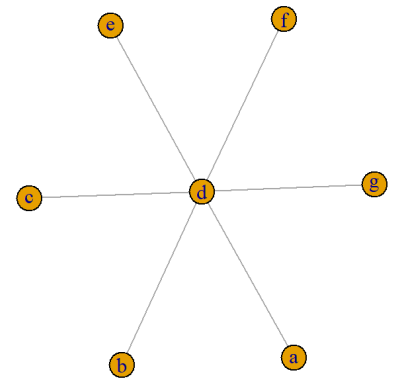
Create your own graph Now we know how easy it is to create a graph from a data frame. But now we want to create our own graph. The principle is the same: Now we use the graph function instead of the graph_from_data_frame() function. The parameters remain the same. First we specify the edges. But now we have to determine the size of the graph ourselves. We pass the parameter n the size 3 and set the parameter directed to FALSE.

```
g.triangle <- graph( edges=c(1,2, 2,3, 3, 1), n=3, directed=F )
plot(g.triangle)
```



Create your own graph with different edges Now we want to create a more complex graph. We want to define a set of nodes that are connected to a certain node. For example, a star network can be formed. This is made possible by the : operator. This is made possible by the : operator. In the following code snippet we connect the nodes a,b and c as well as e, f and g with the node d.

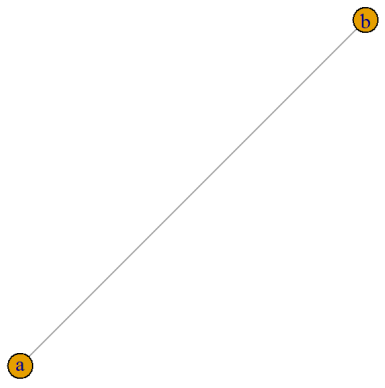
```
g.star <- graph_from_literal(a:b:c--d--e:f:g)
plot(g.star)
```



With – we create an indirect edge, with +- or -+ we can create a directed edge to the left or right. ++ creates a symmetrical edge.

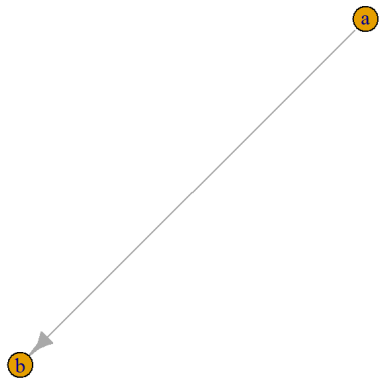
indirected edge

```
g.indirected <- graph_from_literal(a--b)
plot(g.indirected)
```



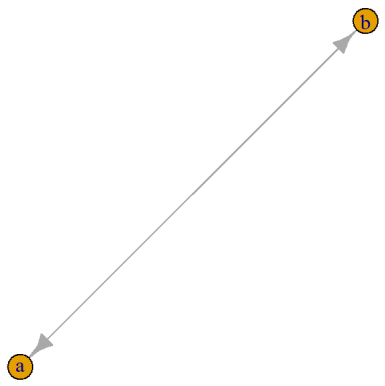
directed edge

```
g.directed <- graph_from_literal(a-+b)
plot(g.directed)
```



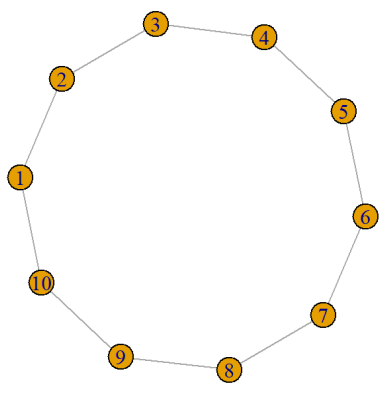
symmetrical edge

```
g.symmetrical <- graph_from_literal(a++b)
plot(g.symmetrical)
```



Ring If you want to make a graph in the form of a ring, you can use the function make_ring(). You only have to specify the number of nodes and a simple ring graph is created.

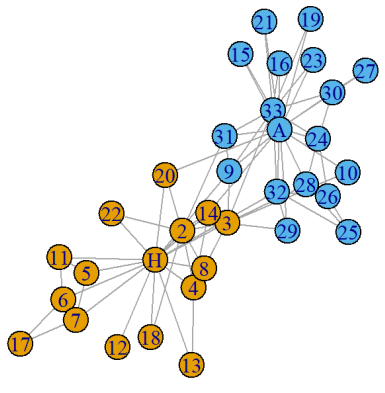
```
g.ring <- make_ring(10)
plot(g.ring)
```



5.4 Use predefined graphs

In another example we use a pre-defined data set *karate*, which is delivered with the package “igraphdata”. The data set already belongs to the igraph class and can therefore already be visualized with the plot() function.

```
data("karate")
plot(karate)
```



5.5 A short closing word on graphs and further interesting graph-packages

Graphs provide a structure to store data. In contrast to relational databases not only the entities but also the relations are given importance. So graphs are a powerful tool for data storage and allow strong queries, which would not be possible in relational environments at all or can only be reached very laboriously via many joins.

A typical application example is social media. But there are also several other possibilities to use graphs. Very interesting are, for example, the interests of website visitors, which can be easily connected by means of graphs, for example to be able to design and deliver advertising very individually.

Other exciting packages for graph modeling and analysis are:

- *tidygraph*: With tidygraph, network objects like graphs and edges are stored similar to tibbles and data.frames. Thus the data preparation and transformation functions of dplyr can be used.
- *ggraph*: As the name of this package suggests, ggraph brings the visualization of graphs into the same form as ggplot2.
- *visnetwork and network3D*: visNetwork and network3D allow to create interactive graphs.