An Introduction to R - Used Car Prices

Lukas Vogt, Philipp Thienel

28 November 2015

Table of content

1. The problem

2. Reproducing the analysis

2.1 Structure of the documents

2.2 Dependencies & required libraries

3. The dataset

3.1 Reading the data

3.2 Cleaning the data

3.3 Adding features to data

4. Explorative analysis

4.1 Descriptive analysis of the dataset

4.2 Dependent variable - 'preis'

4.3 Numeric covariates

4.4 Categorical variables

5. Regression

5.1 Variable selection

5.2 Constructing dummy variables

5.3 Fitting linear models

5.4 Goodness of fit

5.5 Result

6. Poker Simulation

1. The problem

The following problem was given:

The dataset contains records of a website selling used cars from Jul 2011. Along with the price you find various characteristics of different VW station wagons (Golf, Passat, Bora, Caddy, Multivan).

1. Import data and clean it appropriately
2. Analyze the dataset from a descriptive point of view. What do you observe for prices? How have they potentially been sampled?
3. Find a regression model that has the prices as dependent variable. Look at different models and try to identify variable that you expect to drive the price. Also think about dummies and interaction terms. Construct an additional variable "age", i.e., the difference between the first registration (inverkehrssetzung) of the vehicle and Jul 2011.
4. Which is the most reasonable / best regression model? Does it explain the prices well?
5. Illustrate your findings graphically as well as in tabular and text form.

2. Reproducing the analysis

To reproduce the analysis the reader can just execute all steps as shown by himself using the datasets, scripts and functions that can be found under the following url 'https://github.com/philippthienel/hsg-intro-r-a03'.

The user is advised to download the full repository, set the working directory to the parent folder, and execute the scripts from there.

2.1 Structure of the documents

The repository under this url contains two folders, several R scripts and two markwdown documents.

The data folder contains the dataset that will be used in this analysis under the name 'vw_station_wagons.csv'

The R scripts containing all code used in this analysis are:

- A03_read_clean.R
- A03_descriptive.R
- A03_explorative.R
- A03_linear_model.R
- A03_simulation.R

2.2 Dependencies & required libraries

Following R libraries will be required to run the analysis:

- ggplot2
- knitr
- stringr
- reshape2

3. The dataset

The dataset can be downloaded here: 'https://github.com/philippthienel/hsg-intro-r-a03'. According to the problem the dataset "contains records of a website selling used cars from Jul 2011". The dataset of interest is 'vw_station_wagon.csv'. The file 'variables.csv' provides a list of all variable names in the dataset along with english explanations.

```
variable.names <- data <- read.csv("./data/variables.csv", sep=';')
kable(variable.names)
```

Variable	Description.in.English
modell	model
inverkehrssetzung	first registration of the car Month-Year
fahrzeugart	type of car
aussenfarbe	outside color
kilometer	kilometers
getriebeart	transmission type (manual/automatic)
antrieb	front wheel / rear wheel / 4 wheel
treibstoff	fuel type
tueren	number of doors
sitze	number of seats
innenfarbe	inside color
hubraminccm	engine capacity in ccm
zylinder	cylinder
leistunginps	power in horsepower
leergewicht	curb weight
verbrauch	fuel consumption in liters / 100km
co2.emission	co2 emissions
energieeffizienz	energency efficiency
abmfk	certification of vehicle by authorities, i.e., you can only use a car if you have the MFK
garantie	warranty
preis	price

3.1 Reading and the data

The dataset is contained in the folder 'data'. The file 'A03_read_clean.R' contains all code to read and prepare the data for the analysis.

We will read the dataset and assign the dataframe to 'data':

```
path.data <- './data/vw_station_wagons.csv'
data <- read.csv(path.data, sep=',')
```

3.2 Cleaning the data

We can easily see that the dataset is not in a perfect condition for further analysis, particularly with regards to two points:

1. Some variables include units in the datafield and are encoded as character, while they should be integer or numeric

2. Some variables are encoded as integer but should rather be treated as factor

1. Removing the units

The variables 'kilometer', 'verbrauch', 'leergewicht' 'co2.emission', 'garantie' and 'preis' all include their respective units in the datafield. Naturally the variables are therefor encoded as 'character' or 'factor'. This makes mathematical calculations and transformationson those variables impossible and we have to remove the units and cast those variables as 'numeric'.

```
index <- c('kilometer', 'verbrauch', 'leergewicht', 'co2.emission', 'garantie', 'preis')
kable(head(data[,index]))
```

kilometer	verbrauch	leergewicht	co2.emission	garantie	preis
26'200 km	8.3 l/100km	1570 kg	198 g/km	12 Monate	CHF 21'400.-
101'500 km	10 l/100km	1790 kg	240 g/km	12 Monate	CHF 24'900.-
113'000 km	6.2 l/100km	1613 kg	167 g/km	0 Monate	CHF 22'800.-
166'000 km	11 l/100km	1627 kg	266 g/km	0 Monate	CHF 7'900.-
133'000 km	86 l/100km	1901 kg	80 g/km	0 Monate	CHF 18'000.-
78'000 km	6.3 l/100km	1385 kg	148 g/km	0 Monate	CHF 18'000.-

To remove the units from the data we will define a function that takes a character vector as input, extracts the first numerical sequence in every element of the input vector and returns these tuples in a numerical output vector of the same length as the input.

```
GetValue <- function(x) {
  require(stringr)
  x <- gsub("'","",x)
  x <- str_extract(x, '[0-9]+\\.*[0-9]*')
  return(as.numeric(x))
}
```

Then we will call this function on all variables of the dataset, that included units, replacing the old values.

```
data <- within(data,{
  kilometer <- GetValue(kilometer)
  leergewicht <- GetValue(leergewicht)
  verbrauch <- GetValue(verbrauch)
  co2.emission <- GetValue(co2.emission)
  garantie <- GetValue(garantie)
  preis <- GetValue(preis)
})
```

As we can see, the variables are now numerical and do not include the units any longer.

```
index <- c('kilometer', 'verbrauch', 'leergewicht', 'co2.emission', 'garantie', 'preis')
kable(head(data[,index]))
```

kilometer	verbrauch	leergewicht	co2.emission	garantie	preis
26200	8.3	1570	198	12	21400
101500	10.0	1790	240	12	24900

kilometer	verbrauch	leergewicht	co2.emission	garantie	preis
113000	6.2	1613	167	0	22800
166000	11.0	1627	266	0	7900
133000	86.0	1901	80	0	18000
78000	6.3	1385	148	0	18000

2. Convert integer variables to factors

The variables 'tueren', 'sitze' and 'zylinder' are encoded as integer. They take on only a few values and are better interpreted as factors. So we will want to convert them.

```
kable(sapply(data[,c("tueren","sitze","zylinder")], class))
```

tueren	integer
sitze	integer
zylinder	integer

To convert them we can simply cast the respective variables as factors.

```
data <- within(data,{
  tueren <- as.factor(tueren)
  sitze <- as.factor(sitze)
  zylinder <- as.factor(zylinder)
})
```

3.3 Adding features to the data

Apart from the variables that are explicitly provided in the dataset, there are further variables that can be useful for the regression modeling later on.

Three additional features that can easily be extracted from the existing variables:

1. plattform of the model (Golf, Bora, Passat etc.) - can be extracted from the 'modell' variable
2. age of the car in months - we will just take the time difference between first registration and July 2011 (when the data was sampled) as an approximation
3. displacement in litres - can be optained by dividing 'hubrauminccm' by 1000

1. Extracting the 'plattform' from the 'model' variable.

The plattform is always the second word in the 'modell' variable. So we need to extract only the second word of every element of that variable.

Noting that the first word is always 'VW', we can define a function 'GetPlattform' that takes a character vector as input, removes any string 'VW' from every element, and then extracts always the first character sequence of length >= 1 until the first character that is not a letter.

```
GetPlattform <- function(x) {
  require(stringr)
  x <- gsub('VW','',x)
  plattform <- str_extract(x, '[a-zA-Z]+')
  return(plattform)
}
```

We call this function on the variable 'modell' to extract the plattform and assign it to the variable 'plattform'

```
data$plattform <- as.factor(GetPlattform(data$modell))
```

2. Calculating the age of the vehicle

We can get the month and year of the first registration from the variable 'inverkehrssetzung'. This variable is of type factor and has the structure: 'month-year', so for example '01-2011'.

To calculate the age we define a function that takes a factor variable as input, splits the factor by '-' in two columns (month and year), and then calculates the difference between the month of the first registration and July 2011.

```
GetAge <- function(x, month=7, year=2011) {
  require(reshape2)
  df <- colsplit(x,"-",names=c("month","year"))
  age <- (year - df$year)*12 + (month - df$month)
  return(age)
}
```

We call that function on the 'inverkehrssetzung' variable to calculate theage of the vehicle in months and assign the values to the new variable 'age'.

```
data$age <- GetAge(data$inverkehrssetzung)
```

3. Calculating displacement in liters

This is just the simple measure of scaling the variable 'hubrauminccm' down by factor 1000. We will keep one decimalpoint however, since it is customary to indicate the size of the engine like that.

```
data$hubraum.liter <- round(data$hubraum/1000,1)
```

4. Explorative analysis

All relevant code can be found in 'A03_descriptive.R' and 'A03_explorative.R'.

In the following we will conduct an explorative analysis of the variables in the dataset. This will provide important insights on which we will base the development of a linear regression model later.

The exploratorive analysis is structured as follows:

1. Descriptive analysis of dataset
2. Dependent variable - 'preis'
3. Numeric covariates
4. Factor variables

4.1 Descriptive analysis of the dataset

First we will provide some basic facts about the dataset that we are dealing with. We are primarily interested in the following metrics:

1. Dimensions of the dataset: number of observations and variables

2. Types of variables (numeric, factors etc.)
3. Missing data - how many observations have missing values, how are they distributed

1. Number of observations and variables

We have a total of 1170 observations on 24 variables. Three of those variables were added by us manually ('age', 'plattform', 'hubraum.liter').

```
dimensions <- dim(data)
names(dimensions) <- c("observations","variables")
kable(dimensions)
```

observations	1170
variables	24

2. Types of variables

We can easily determine the class of every variable qnd save the result in a named vector 'variables'. For this vector the name of every element will correspond to the name of the variable.

```
variables <- sapply(data, class)
```

Following variables are 'numeric' or 'integer' in the dataset.

```
numerical.variables <- variables[variables %in% c("integer","numeric")]
kable(numerical.variables , col.names="class")
```

	class
kilometer	numeric
hubrauminccm	integer
leistunginps	integer
leergewicht	numeric
verbrauch	numeric
co2.emission	numeric
garantie	numeric
preis	numeric
age	numeric
hubraum.liter	numeric

The following variables are either 'factors' or 'logical'. We will call them *categorical* to denote that they describe categories rather than numerical values (even though the categories can be numbers, like the number of doors for example). Also note that we do not have any variabls of type 'character', which is why we did not list any.

```
categorical.variables <- variables[variables %in% c("factor","logical")]
kable(categorical.variables , col.names="class")
```

	class
modell	factor

	class
inverkehrssetzung	factor
fahrzeugart	factor
aussenfarbe	factor
getriebeart	factor
antrieb	factor
treibstoff	factor
tueren	factor
sitze	factor
innenfarbe	factor
zylinder	factor
energieeffizienz	factor
abmfk	logical
plattform	factor

3. Missing values

Missing values are encoded as NA in the dataset. We are primirly interested in how many observations are afected by missing values overall, how those missing values are distributed over the variables (are there variables that suffer more strongly from missing values than others?), and if there is possibly a systematic component as to why values are missing.

In the whole dataset we only have 616 observations where no values are missing in any of the variables.

```
sum(complete.cases(data))
```

```
## [1] 616
```

To display the missing values per variable we define a short function 'CountNA'. The function takes a data frame as input and returns a data frame with the numer of missing values per variable (only if count >0).

```
CountNA <- function(df){
  count <- sapply(df, FUN = function(x) sum(is.na(x)))
  count <- data.frame(variable = names(count), count.na = count,
                      row.names=NULL)
  count <- count[count$count.na>0,]
  index <- order(count$count.na, decreasing=T)
  return(count[index,])
}
```

As we can see especially the variables 'energieeffizienz, 'co2.emission' and 'innenfarbe' suffer strongly from missing values.

```
count.na <- CountNA(data)
kable(count.na)
```

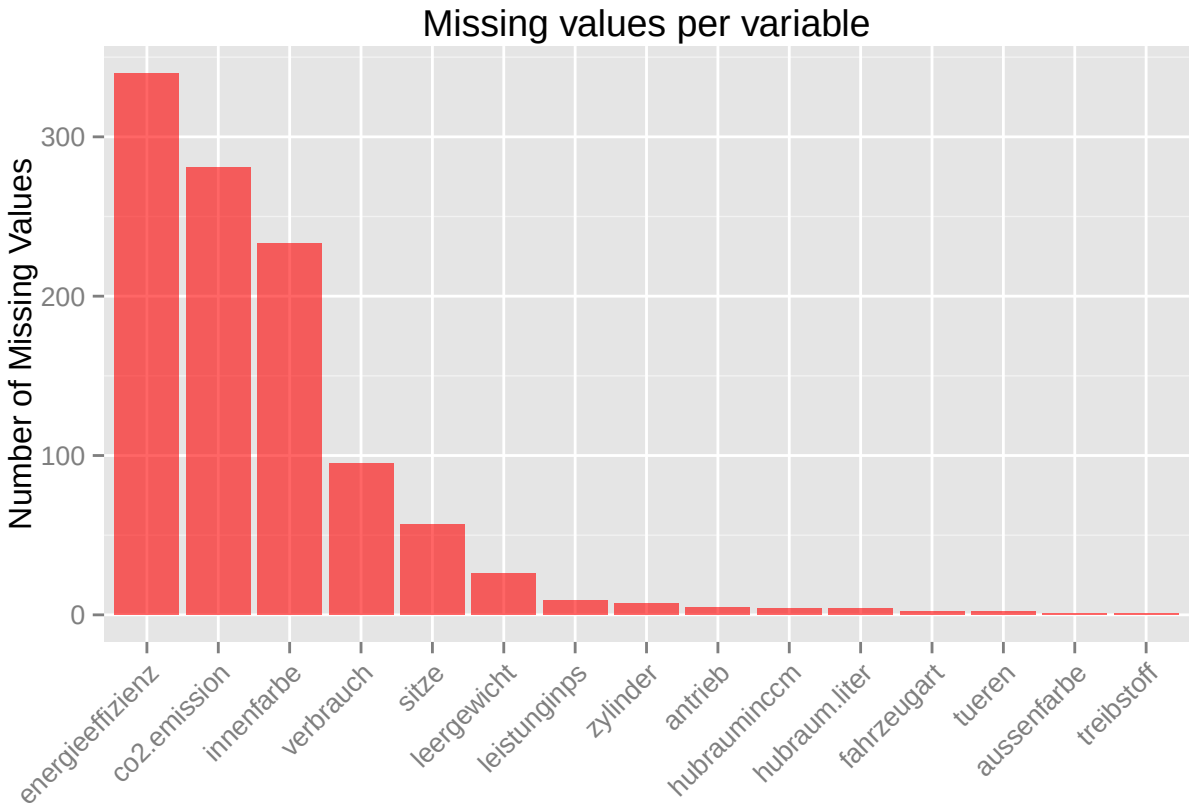
	variable	count.na
18	energieeffizienz	340
17	co2.emission	281
11	innenfarbe	233
16	verbrauch	95

	variable	count.na
10	sitze	57
15	leergewicht	26
14	leistunginps	9
13	zylinder	7
7	antrieb	5
12	hubraumincm	4
24	hubraum.liter	4
3	fahrzeugart	2
9	tueren	2
4	aussenfarbe	1
8	treibstoff	1

For a graphical representation we call the function 'naBar' which is contained in the file 'naBar.R' in the functions folder. This function basically does tha same as the 'CountNA' function, but instead of a dataframe it returns a barplot (using ggplot2) counting the number of missing values per variable. For further information the reader can refer to the documentation of the function itself.

```
source("./functions/naBar.R")
```

```
naBar(data)
```



Bias in missing values

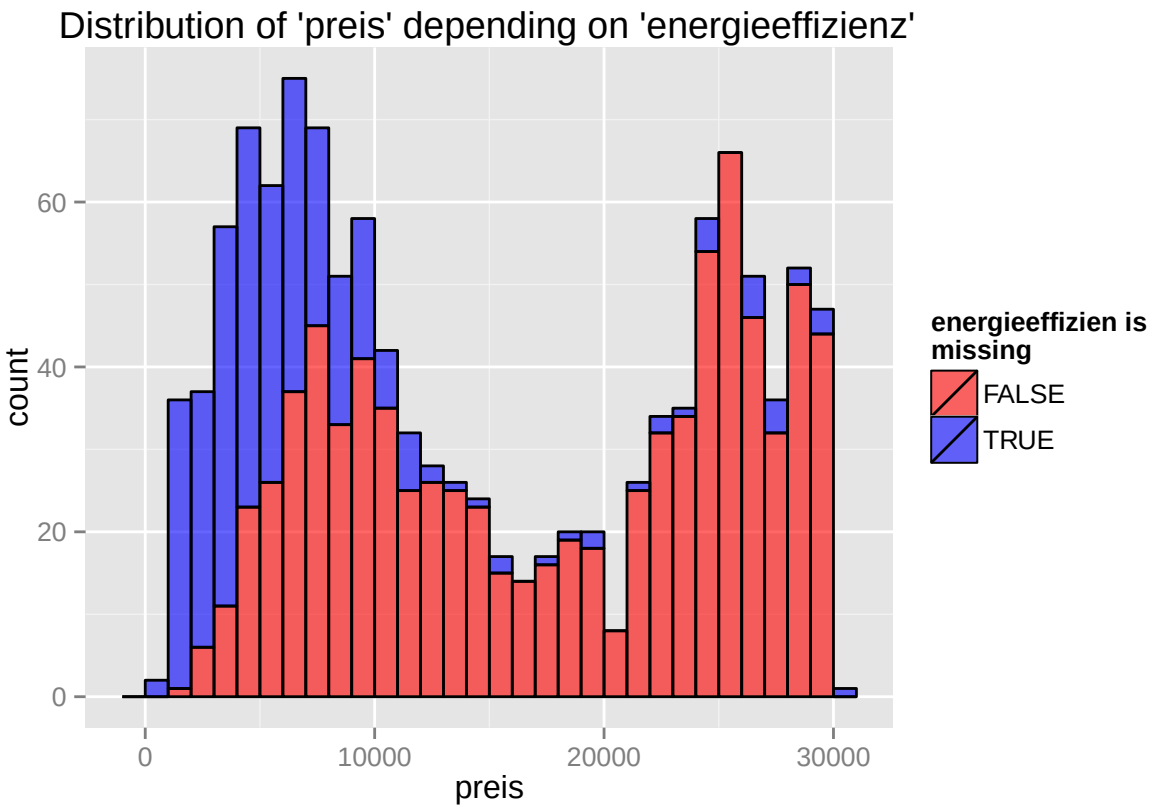
Now that we could observe a high number of missing values especially in the 'energieeffizienz' variable the question poses itself how to deal with those missing values.

Would we introduce a selection bias if we exclude the observations where the variable 'energieeffizienz' is missing?

In short: Yes, we would.

It is easy to see, that 'energieeffizienz' is missing disproportionally in lower ranges of the dependent variable 'preis'.

```
plot <- ggplot(data, aes(x=preis, fill = is.na(energieeffizienz)))
plot <- plot + geom_histogram(alpha = 0.6, binwidth=1000, colour="black")
plot <- plot + scale_fill_manual(values=c("red","blue"))
plot + labs(fill = 'energieeffizien is\nmissing',
            title="Distribution of 'preis' depending on 'energieeffizienz'")
```



If we wanted to use that variable in our regression model, we would have to pay special attention on how to treat missing variables. Since we have no information on the underlying reasons for the missing values, we would have a hard time getting rid of the bias. We therefore refrain from using the that variable.

4.2 Dependent variable - 'preis'

In preperation for developing a regression model explaining (or predicting) the price of the cars, we will have a closer look at the 'preis' variable.

A first look on some numerical distribution measures lead to the following observations:

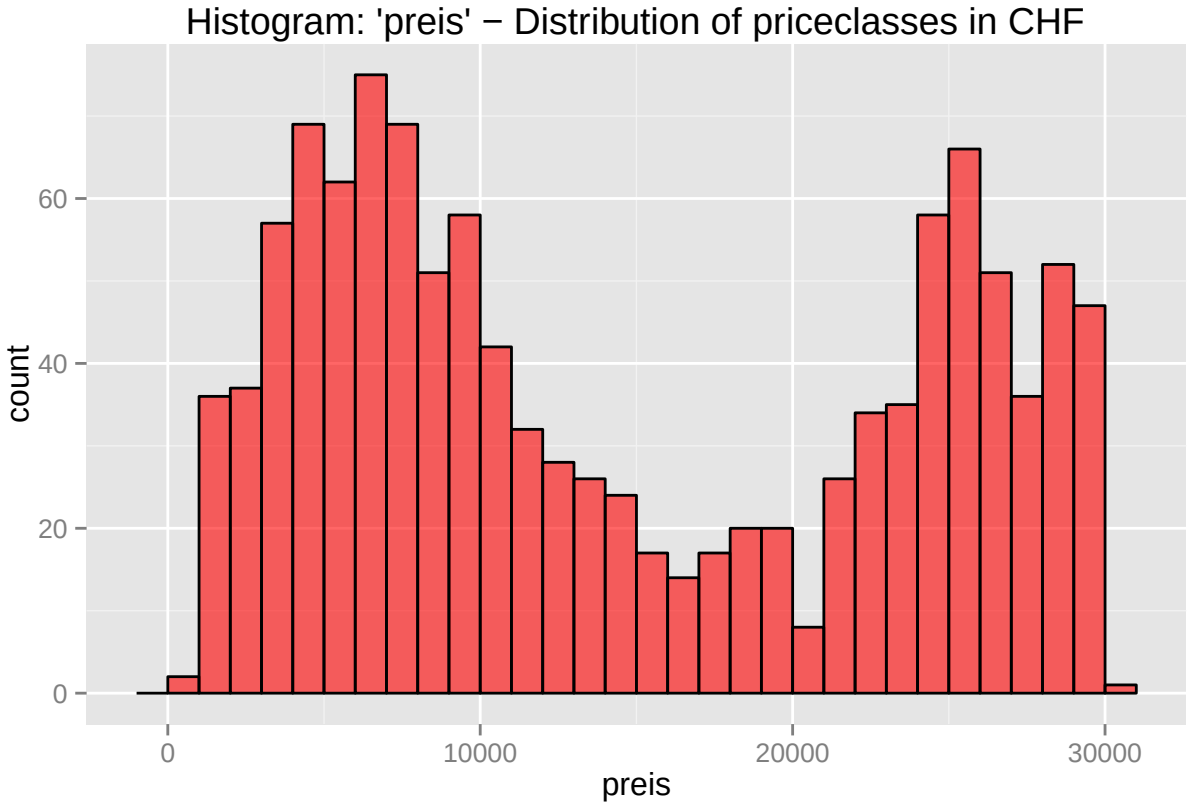
- the price ranges from CHF 700 to CHF 30.000
- CHF 30.000 seems very high for a used car. We will have a look at outliers in paragraph 4.5
- median and mean are relatively close together - so that we would assume the distribution to be relatively symmetrical

```
kable(t(as.matrix(summary(data$preis))))
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
700	6800	11900	14790	24740	30000

A graphical representation of the distribution of 'preis' helps to substantiate some of the earlier observations. What is particularly interesting is that while the distribution seems indeed almost symmetrical, it is not concentrated around the mean or median. Instead we find a bimodal distribution. With two distinct *peaks* around CHF 5000 and CHF 25.000.

```
plot <- ggplot(data, aes(x=preis))
plot <- plot + geom_histogram(binwidth=1000, fill="red", alpha=0.6, color="black")
plot+ labs(title="Histogram: 'preis' - Distribution of priceclasses in CHF")
```

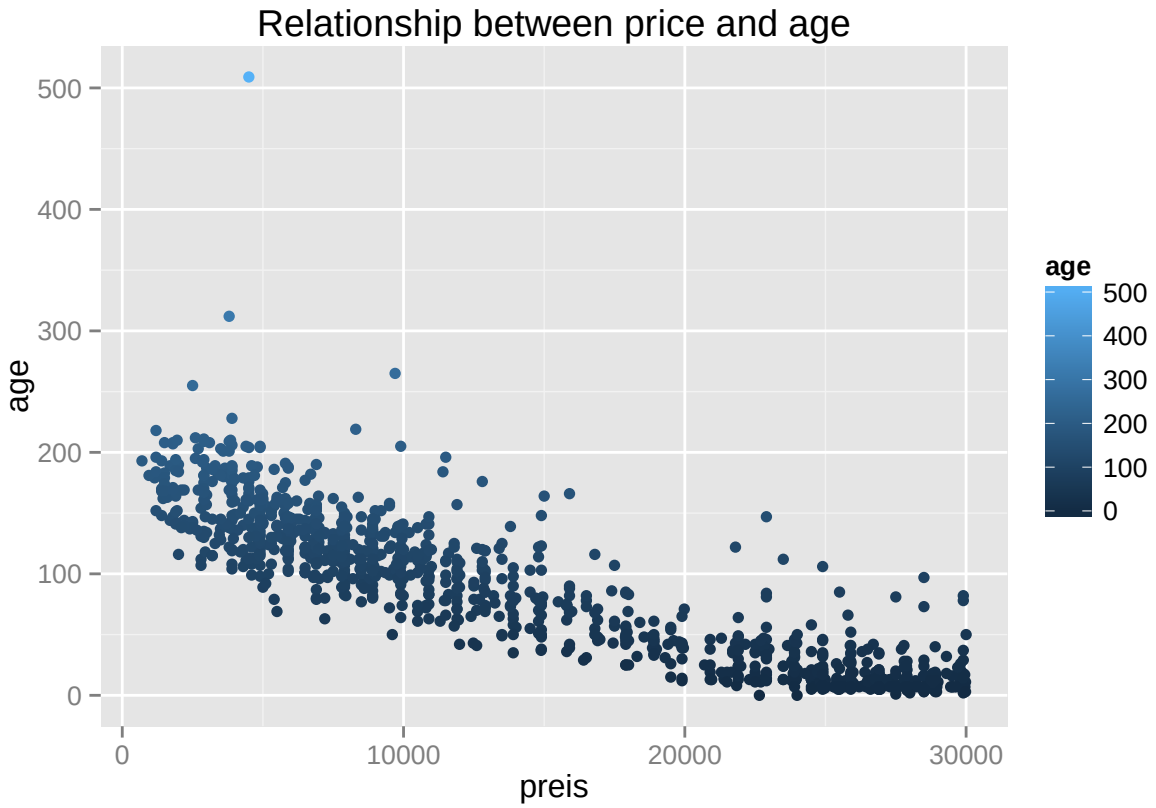


This behavior could potentially stem from sampling issues. If the sampling contains a strong selection bias then we might end up sampling only 2 subgroups (2 price-classes) of the underlying population.

Another explanation would be that the underlying population of used cars just exhibits naturally such a price structure. A reason for this might be that car owners either sell their car when it is still relatively new, or when it is already pretty old. And the prices just reflect this natural selling behavior.

Looking at the relationship between 'preis' and 'age' we find that it almost follows a strong linear trend.

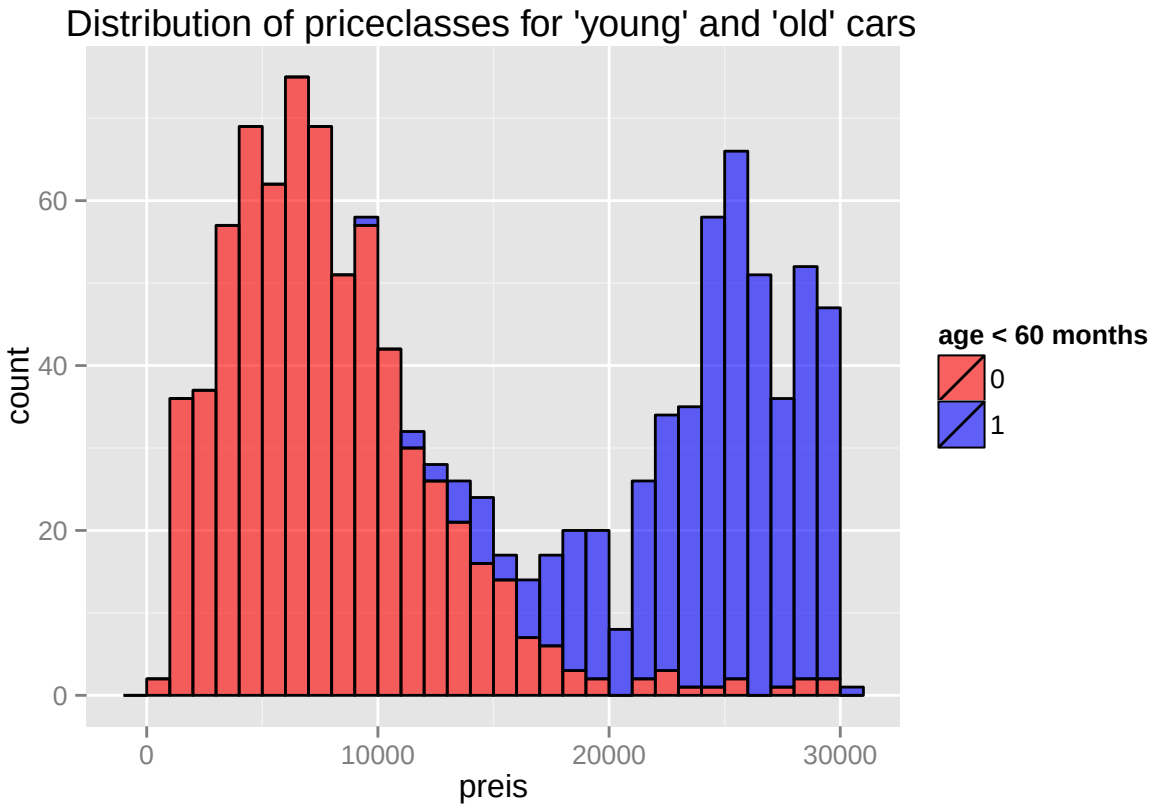
```
plot <- ggplot(data, aes(x=preis, y=age, colour = age)) + geom_point()
plot + labs(title="Relationship between price and age")
```



If we introduce a variable 'young' (age < 60 months) and differentiate the 'preis' histogram by this variable, we find indeed two subgroups that differ strongly in price distribution and age.

```
# add binary variable 'young' if age < 60 months (5years)
data$young <- ifelse(data$age <60, 1, 0)
data$young <- as.factor(data$young)

# graphical distribution differenciated by 'young'
plot <- ggplot(data, aes(x=preis, fill= young))
plot <- plot + geom_histogram(binwidth=1000, alpha=0.6, colour="black")
plot <- plot + scale_fill_manual(values=c("red","blue"))
plot + labs(title="Distribution of priceclasses for 'young' and 'old' cars", fill = "age < 60 months")
```



4.3 Numeric covariates

As a preparation for the regression model later on it is valuable to gain some insights into the distribution of the covariates.

We are mainly interested in two aspects:

1. Dispersion of covariates
2. Correlation between covariates

1. Dispersion To get a better comparaility of dispersion some normalization or is useful. We will rescale all variables to the interval [0,1]. Where 0 is given by the minimum of the respective variable and 1 by the maximum. This allows for an easier graphical comparison of the dispersion of the variables.

First we define a short function to rescale a numerical vector:

```
ReScaling <- function(x){
  x.min <- min(x, na.rm = TRUE)
  x.max <- max(x, na.rm = TRUE)
  y <- (x - x.min)/(x.max - x.min)
  return(y)
}
```

Then we apply this function to all numerical or integer variables and save the rescald variables in a new dataframe 'dataScaled'.

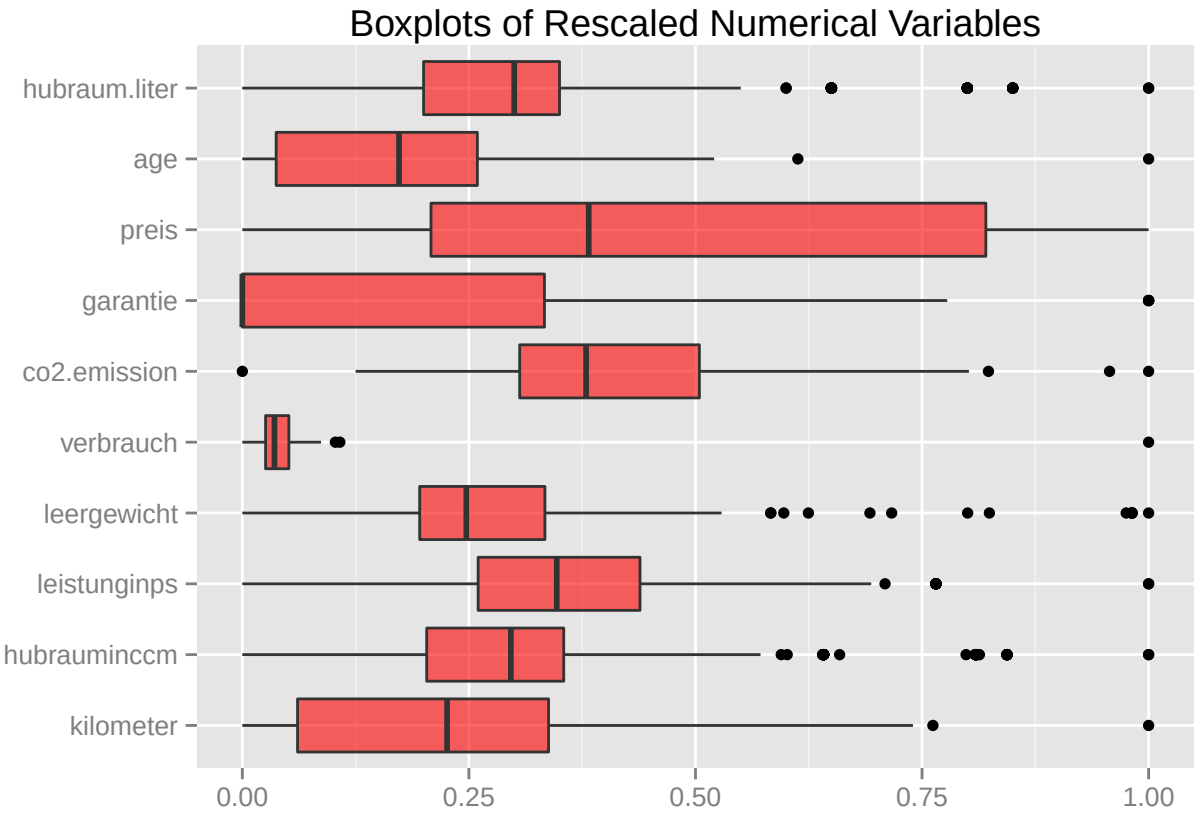
```
numeric.covariates <- names(data)[sapply(data, class) %in% c("integer","numeric")]

dataScaled <- data.frame(sapply(data[,numeric.covariates],FUN=ReScaling))
```

We use boxplots to visualize the dispersion of all variables. We can observe the following:

- the variable 'verbrauch' seems to be dominated by one extreme outlier at the upper end, while the rest of the values is relatively close together
- 'age', 'kilometer' and 'garantie' show the highest (relative) spread
- most other variables show a relatively low spread, where 50% of their values cover less than 25% of the range

```
dataScaledMelt <- melt(dataScaled)
plt <- ggplot(data=dataScaledMelt, aes(x=variable, y= value))
plt <- plt + geom_boxplot(fill="red", alpha=0.6) + coord_flip()
plt + labs(title="Boxplots of Rescaled Numerical Variables", x=NULL, y=NULL)
```



2. Correlation between covariates

Correlations between the covariates and the dependent variable and among the covariates can give valueable input for the variable selection in the regression model.

Correlation Matrix:

```
# create correlation matrix
cor.matrix <- cor(data[,numeric.covariates], use="pairwise.complete.obs")
kable(cor.matrix[,1:5])
```

	kilometer	hubrauminccm	leistunginps	leergewicht	verbrauch
kilometer	1.0000000	0.4421278	-0.0326063	-0.1006008	0.1860555
hubrauminccm	0.4421278	1.0000000	0.3942003	0.3817734	0.3036200
leistunginps	-0.0326063	0.3942003	1.0000000	0.3680665	0.2953052
leergewicht	-0.1006008	0.3817734	0.3680665	1.0000000	0.0801746
verbrauch	0.1860555	0.3036200	0.2953052	0.0801746	1.0000000
co2.emission	0.3538358	0.5560042	0.4686437	0.4427186	0.3594363
garantie	-0.4530271	-0.2431581	0.0373400	0.1071301	-0.1297943
preis	-0.8605651	-0.3806537	0.1354277	0.2982627	-0.2302126
age	0.7749900	0.4261774	-0.1052117	-0.3100865	0.2883474
hubraum.liter	0.4464731	0.9994005	0.3948409	0.3792670	0.3082417

Correlation Matrix (continued):

```
kable(cor.matrix[,6:10])
```

	co2.emission	garantie	preis	age	hubraum.liter
kilometer	0.3538358	-0.4530271	-0.8605651	0.7749900	0.4464731
hubrauminccm	0.5560042	-0.2431581	-0.3806537	0.4261774	0.9994005
leistunginps	0.4686437	0.0373400	0.1354277	-0.1052117	0.3948409
leergewicht	0.4427186	0.1071301	0.2982627	-0.3100865	0.3792670
verbrauch	0.3594363	-0.1297943	-0.2302126	0.2883474	0.3082417
co2.emission	1.0000000	-0.1669743	-0.4144796	0.5315443	0.5621505
garantie	-0.1669743	1.0000000	0.4822365	-0.4597750	-0.2455172
preis	-0.4144796	0.4822365	1.0000000	-0.8962758	-0.3844113
age	0.5315443	-0.4597750	-0.8962758	1.0000000	0.4342286
hubraum.liter	0.5621505	-0.2455172	-0.3844113	0.4342286	1.0000000

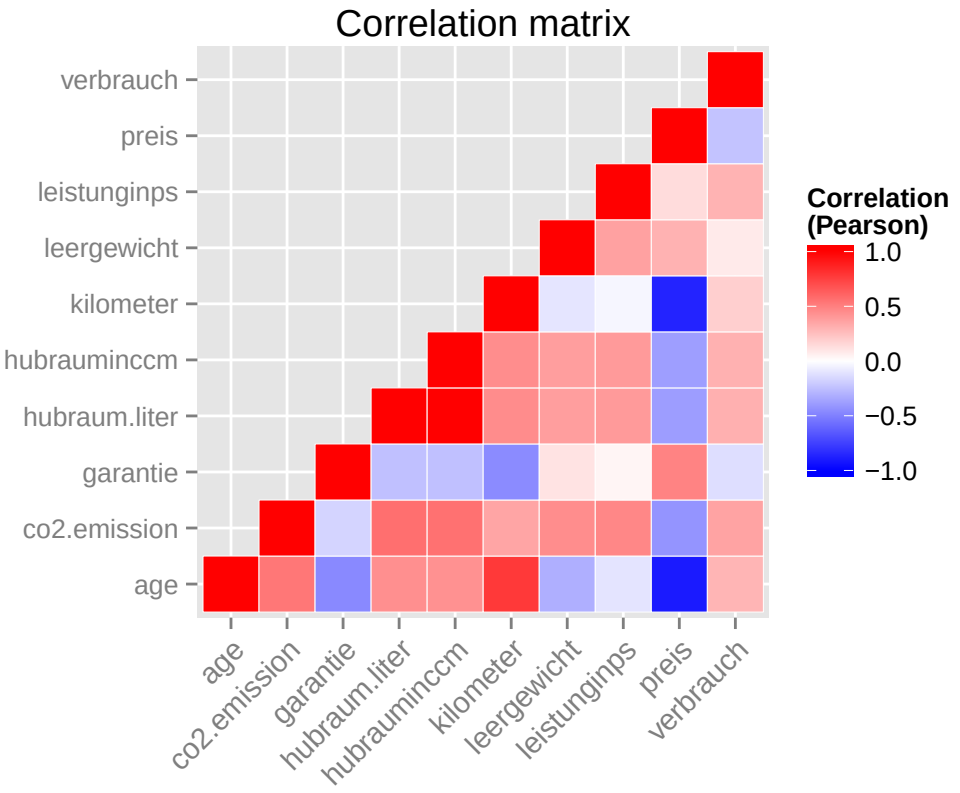
Since the numerical correlation matrix is somewhat unintuitive at first glance, will represent this matrix as a tile plot, where the colour gradient indicates the degree of correlation. More detaled explanation of the function 'corTile' can be found in the file 'corTile.R' in the functions folder.

With respect to correlations we can observe some very useful facts:

- there is a strong negative correlation between 'preis' and 'age' and 'preis' and 'kilometer'
- unfortunately 'age' and 'kilometer' are strongly correlated themselves

```
source("./functions/corTiles.R")
```

```
corTile(data[,numeric.covariates], use="pairwise.complete.obs")
```



4.4 Categorical variables

Frequency tables

Of main interest are the categorical variables that allow for easy implementation of binary (dummy) variables for the regression modeling later on.

The variable 'getriebeart' (gear shift) can take the values 'Automat' and 'Handschaltung' (automativ and manual). While 'Handschaltung' dominates, the proportion with an automatic gear shift is not insignificant.

```
kable(t(as.matrix(with(data, table(getriebeart)))))
```

Automat	Handschaltung
214	956

'Antrieb' indicates wether a car has a four-wheel, front-wheel or back-wheel-drive. The 'Vorderradantrieb' (front-wheel-drive) clearly dominates, although there is again a significant number of cars with 'Allrad' (four-wheel-drive).

```
kable(t(as.matrix(with(data, table(antrieb)))))
```

Allrad	Hinterradantrieb	Vorderradantrieb
174	4	987

'Treibstoff' indicates what kind of fuel a car uses. 'Benzin' (standard gas) is in the majority, although a significant portion uses 'diesel'.

```
kable(t(as.matrix(with(data, table(treibstoff)))))
```

Benzin	Diesel	Gas (LNG, LPG, GPL, CNG)
776	388	5

'abmfk' indicates wether a car has a certain emission certification with respect to exhaust fumes. This variable is almost evenly split.

```
kable(t(as.matrix(with(data, table(abmfk)))))
```

FALSE	TRUE
652	518

5. Regression

5.1 Variable selection

We can base the decision on which variables to include in our regression on both economic reasoning and on the statistical evidence we found so far in the explorative analysis.

The four categorical variables we have discussed earlier are all economically likely to have an impact on the sales price of a car:

- automatic shift: higher convenience (higher price)
- four-wheel-drive: higher versatility (higher price)
- uses diesel: lower fuel cost (higher price)
- 'abmfk': has impact on where the car can be driven and on taxation (lower price if not present)

From the correlation analysis we are most interested in the variables 'age' and 'kilometer' as they both correlated strongly with 'preis'.

5.2 Constructing dummy variables

As we saw in the analysis of the *categorical* variables, there are mainly 4 variables that allow for an easy implemetation of dummy variables: 'getriebeart', 'treibstoff', 'antrieb' and 'abmfk'.

We will therefore implement the 4 dummies as follows:

- 'automatic' - 1: if the car has automatic gear shift (getriebeart == Automat)
- 'diesel' - 1: if the engine is a diesel (treibstoff == Diesel)
- 'by4' - 1: if the car is four wheel drive (antrieb == Allrad)
- 'abmfk' - True: if the car has 'abmfk' (is already given in dataset)

```
data$automatic <- ifelse(data$getriebeart == "Automat", 1, 0)
data$diesel <- ifelse(data$treibstoff == "Diesel", 1, 0)
data$by4 <- ifelse(data$antrieb == "Allrad", 1, 0)
```

5.3 Fitting linear models

We will now fit four different linear regression models and estimate their coeficients.

1. A naive model

The first model is very simple in it's structure. We estimate a linear dependency of 'preis' on 'age' and 'kilometer'. Interestingly both coefficients are statistically significant on 99.9% confidence.

```
formula1 <- 'preis ~ age + kilometer'
fit1 <- lm(formula1, data)
kable(summary(fit1)$coefficients)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	27631.0481236	174.3930851	158.44119	0
age	-86.4997690	2.4936103	-34.68857	0
kilometer	-0.0506342	0.0020172	-25.10144	0

2. A linear model with level effects in age (decreasing)

Instead of including age linearly in the model we take now the square root of age. The economic intuition behind this is, that the (negative) impact of an additional unit of age decreases with the overall age of a car.

```
formula2 <- 'preis ~ sqrt(age) + kilometer'
fit2 <- lm(formula2, data)
kable(summary(fit2)$coefficients)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	32674.6508666	224.1265363	145.78662	0
sqrt(age)	-1632.5775884	40.9327372	-39.88440	0
kilometer	-0.0385959	0.0020389	-18.92934	0

3. A linear model with level effects in age (decreasing) and dummies

We now add three dummy variables to the model. We tested using 'abmfk' as well, but could not find a significant effect of this variable.

```
formula3 <- 'preis ~ sqrt(age) + kilometer + diesel + by4 + automatic'
fit3 <- lm(formula3, data)
kable(summary(fit3)$coefficients)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	32044.5096598	238.9506062	134.105162	0e+00
sqrt(age)	-1619.1538289	41.5575916	-38.961686	0e+00

	Estimate	Std. Error	t value	Pr(>|t|)
kilometer	-0.0412678	0.0020368	-20.260745	0e+00
diesel	900.1757104	191.6637011	4.696642	3e-06
by4	1699.2944817	250.4700818	6.784421	0e+00
automatic	1390.4879199	226.4838142	6.139458	0e+00

4. A linear model with maximum fit (with significant coefficients)

The last model is our best fit model. We added more numeric covariates, such as:

- 'leistunginps' - the horsepower
- 'garantie' - months in warranty
- 'leergewicht' - weight when empty

```
formula4 <- 'preis ~ sqrt(age) + kilometer + leistunginps + garantie +
             leergewicht + diesel + by4 + automatic'
fit4 <- lm(formula4, data)
kable(summary(fit4)$coefficients)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	21805.8051531	919.6837199	23.710113	0.0000000
sqrt(age)	-1540.5577248	40.0680327	-38.448549	0.0000000
kilometer	-0.0424727	0.0019018	-22.333204	0.0000000
leistunginps	25.7827023	3.7704325	6.838129	0.0000000
garantie	33.8425052	13.6472498	2.479804	0.0132900
leergewicht	4.4889445	0.6743046	6.657147	0.0000000
diesel	693.6746097	211.2882301	3.283073	0.0010582
by4	1055.4720415	233.8242236	4.513955	0.0000070
automatic	616.1787567	213.5599536	2.885273	0.0039848

5.4 Goodness of fit

A standard measure to evaluate if a model fits well is the so called *R Squared*. It measures how much of the total variation in the dependent variable can be explained by the model. We can simply acess the R Squared from our regression analysis and compare them. By this measure the last (and biggest) model clearly outperforms the other models.

```
r.sq <- data.frame(name = c("fit1","fit2","fit3","fit4"),
                   r.squared = c(summary(fit1)$r.squared,
                                 summary(fit2)$r.squared,
                                 summary(fit3)$r.squared,
                                 summary(fit4)$r.squared))
kable(r.sq)
```

name	r.squared
fit1	0.8722725
fit2	0.8902183
fit3	0.9003334
fit4	0.9210243

However, only looking at the R Squared might not be a good idea. R Squared measures only the goodness of fit in the data on which the model was optimized on. But what would happen if we needed to predict the 'preis' based on our model but on new data?

A common measure for the predictive quality of a model is the Root Mean Squared Error.

To get an idea what the predictive quality of our four models would be like we chose the following approach:

1. We split the existing dataset randonly into a training and a test dataset
2. We optimize (train) our models on the training dataset
3. Then we use the models to predict the 'preis' on the test dataset
4. We compare the prediction with the actual values and calculate the Root Mean Squared Error

All this handles the function 'RMSEPredict'. For further information on the function just access the file directly. It is located in the functions folder.

Just calculating the RMSE once for the four models might still not be enough. Since we randomly split the dataset into test and trainig sets, we could just be unlucky in this assignment and not come to a fair comparison.

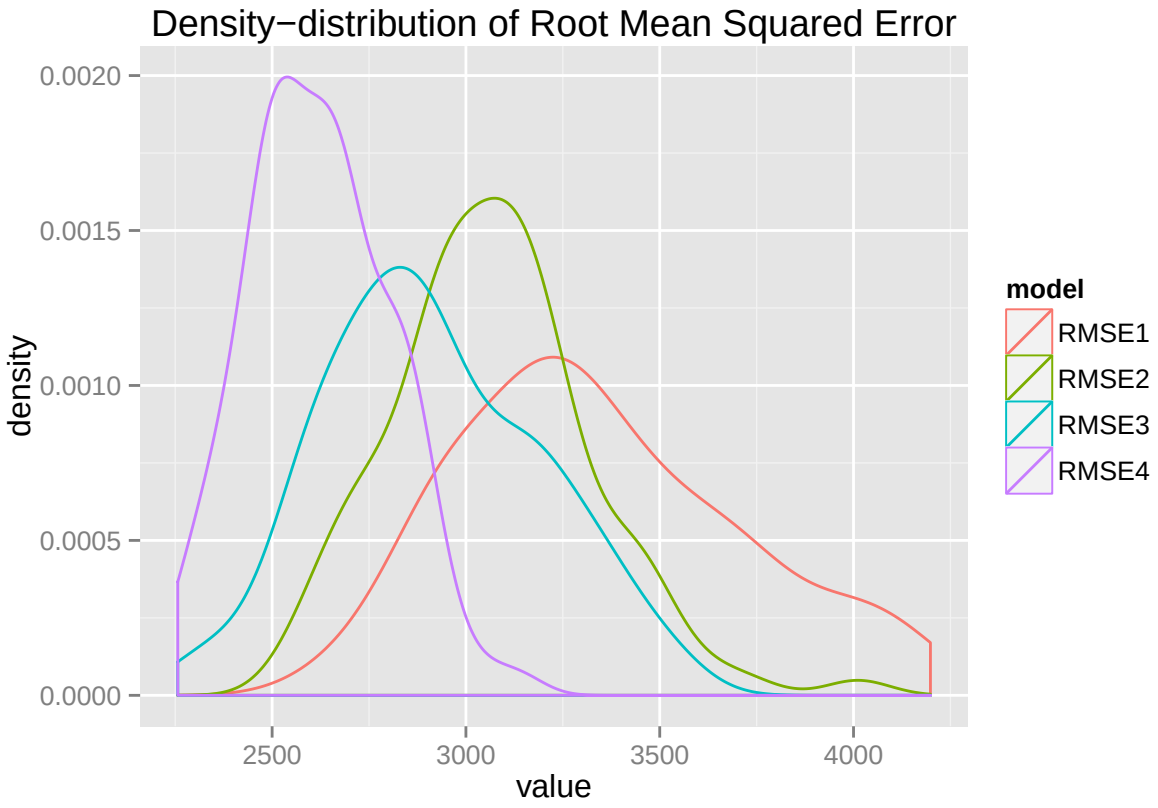
Therefore we will calculate the RMSE 300 times per model and compare the densitiy distributions of the RSME's, rather than just one RMSE each.

```
source("./functions/RMSEPredict.R")
```

```
# calculate RMSE's
RMSE1 <- replicate(100, RMSEPredict(formula1, data, alpha=0.8))
RMSE2 <- replicate(100, RMSEPredict(formula2, data, alpha=0.8))
RMSE3 <- replicate(100, RMSEPredict(formula3, data, alpha=0.8))
RMSE4 <- replicate(100, RMSEPredict(formula4, data, alpha=0.8))
```

```
# create data frame
RMSE.compare <- data.frame(RMSE1, RMSE2, RMSE3, RMSE4)
RMSE.compare <- melt(RMSE.compare)
```

```
plot <- ggplot(RMSE.compare, aes(x=value, colour=variable))
plot <- plot + geom_density()
plot + labs(title="Density-distribution of Root Mean Squared Error",
            colour="model")
```



5.5 Result

As we can see from the density plot the last model (fit4) also exhibits the better RMSE. We can conclude that from our models so far, this model is indeed the best model. It has the highest fit to the data and also the lowest expected RMSE, when predicting new data.

6. Poker Simulation

6.1 The problem

The following problem was given:

Suppose you play cards with friends. You have a deck of 52 cards with 13 ranks and 4 suits (clubs, diamonds, spades, and hearts). Think of the deck as a typical poker deck. Suppose you are 6 players and each is dealt 5 cards. Find by means of a simulation (i.e., dealing cards, for 100, 10000, 100000 times) the following (approximate) probabilities:

1. The probability that any two players are dealt a Full House?
2. Player 1 is dealt a Royal Flush (a Royal Flush is hand consisting of the cards 9,10,11,12,13 of a single suit)

6.2 The simulation

The file containing the full script to run the simulation is saved as 'A03_simulation.R'.

The script contains four functions:

1. DrawHands: Deal hand to each player
2. CheckFH: Check if one player has a full house
3. ProbFH: Compute probability of any two players having a full house in one game
4. ProbRF: Compute probability that player 1 has a royal flush

ProbFH

Compute probability of any two players having a full house in one game

Args:

- deck: Card deck
- nPL: Number of players
- nDrws: Number of cards drawn
- nSim: Number of simulations

Returns:

- Number of simulations, number of occurences of two players having a full house, probability of two players having a full house

ProbRF

Compute probability that player 1 has a royal flush

Args:

- deck: Card deck
- nSim: Number of simulations
- nDraws: Number of cards drawn

Returns:

- Number of simulations, number of occurences of a royal flush, probability of a royal flush