Statistical methods in data mining: Project 1

Predicting claim frequencies comparing standard GLM models to some modern machine learning methods.

Carbonera Francesco
Giada Cesaro

06/11/2019

Abstract

In this project we describe, analyze and compare the classical generalized linear model approach (GLM) for claims frequency modeling to some modern machine learning methods, namely boosting and neural network. We train these methods, tune their parameters and test them using the French motor third-party liability insurance data set, which is freely available in the CASdatasets R package.

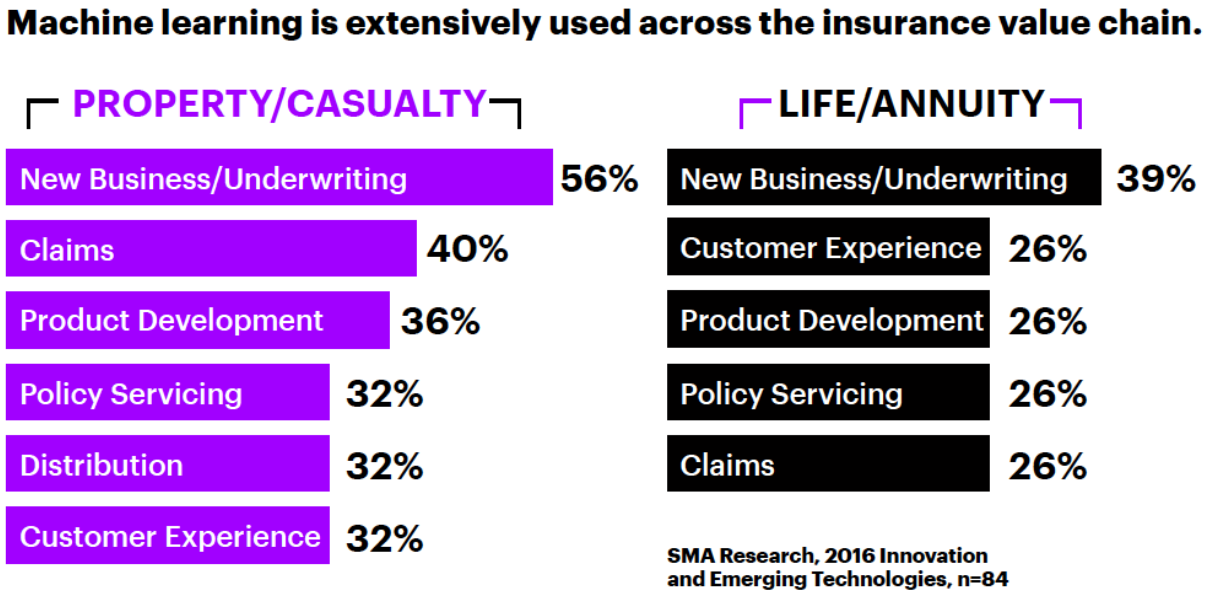
Contents

1 Introduction

"We have created more data in the past two years than the human race has ever created. Most insurers are struggling to maximise the benefits of machine learning to analyze this huge amount of data".[1]

Bernard Marr - Forbes magazine

The use of data mining techniques for predictive modeling problems is growing exponentially, thanks to the huge amount of data at our disposal and the still increasing computational power of machines. All this is affecting the insurance industry, and more and more machine learning applications to non-life pricing have been experimented. Insurers are being forced to explore ways to use predictive modelling and machine learning to maintain their competitive edge, boost business operations and enhance customer satisfaction. The potential applications of machine learning in insurance are numerous: from understanding risk appetite and premium leakage, to expense management, subrogation, litigation and fraud identification.
Below is an example of the proportion of the following macro-fields which is now performed using Machine Learning techniques (non-life business on the left, life business on the right).



As Master's degree students in Actuarial and Statistical Science, in this project we choose to deepen our knowledge on a vast and interesting topic in the industrial industry, namely the modelling of insurance claims. We also thought this could serve us as a starting point for some future development on this current topic, given also our upcoming master's thesis.
Our interest is to compare the common techniques to model the claim frequency (a fundamental component for non-life insurance pricing) to other machine learning methods that will probably be implemented more and more in the future.
The idea is to give an overview on common practices in the insurance industry w.r.t claims frequency modelling, therefore we will present and implement GLM models for count data. We will apply these models using the **French motor third-party liability insurance** dataset, which is freely available in the `CASdatasets` R package, and analyze the results. We will extend our research to other computationally more expensive learning models, namely boosting the and neural network models.

[1] More information is available here: https://www.forbes.com/sites/bernardmarr/

2 Data description

We consider the data `freMTPL2freq` included in the R package `CASdatasets`

We provide a brief description of the data: we have 6780013 individual car insurance policies and for each policy 12 variables are available. The claim counts observed in one accounting year is included among the variables. We briefly describe the variables according to the R documentation for the `CASDatasets` package:

- `IDpol`: policy number (ID);
- `ClaimNb`: number of claims on the policy during the exposure period (our **target** variable);
- `Exposure`: exposure (in years);
- `Area` : area code;
- `VehPower`: power of the car (ordered categorical);
- `VehAge`: age of the car in years;
- `DrivAge`: age of the driver in years (in France, people can drive a car at 18);
- `BonusMalus`: bonus-malus level between 50 and 230, <100 means bonus, >100 means malus;
- `VehBrand`: car brand;
- `VehGas`: car gas, diesel or regular;
- `Density`: density of inhabitants per km^2 in the city driver of the car lives in;
- `Region`: policy regions in France (based on a standard French classification).

We can see the type of data of the variables and the first few rows of the dataset.

```
data("freMTPL2freq")
dati<-freMTPL2freq
n<-nrow(dati)
head(dati)
```

```
  IDpol ClaimNb Exposure Area VehPower VehAge DrivAge BonusMalus VehBrand
1     1       1     0.10    D        5      0      55         50      B12
2     3       1     0.77    D        5      0      55         50      B12
3     5       1     0.75    B        6      2      52         50      B12
4    10       1     0.09    B        7      0      46         50      B12
5    11       1     0.84    B        7      0      46         50      B12
6    13       1     0.52    E        6      2      38         50      B12
   VehGas Density Region
1 Regular    1217    R82
2 Regular    1217    R82
3  Diesel      54    R22
4  Diesel      76    R72
5  Diesel      76    R72
6 Regular    3003    R31
```

It can be verified that these are individual policies by looking if there is any repeated IDpol value:

```
IDunique<- dati%>%group_by(IDpol)%>%summarize(count=n())%>%filter(count>1)
IDunique$count # it accesses the aggregated count for the IDpol which appear more than once (if any)
```

```
integer(0)
```

No `IDpol` matches the query, which means every Policy numbers is unique in the dataset.

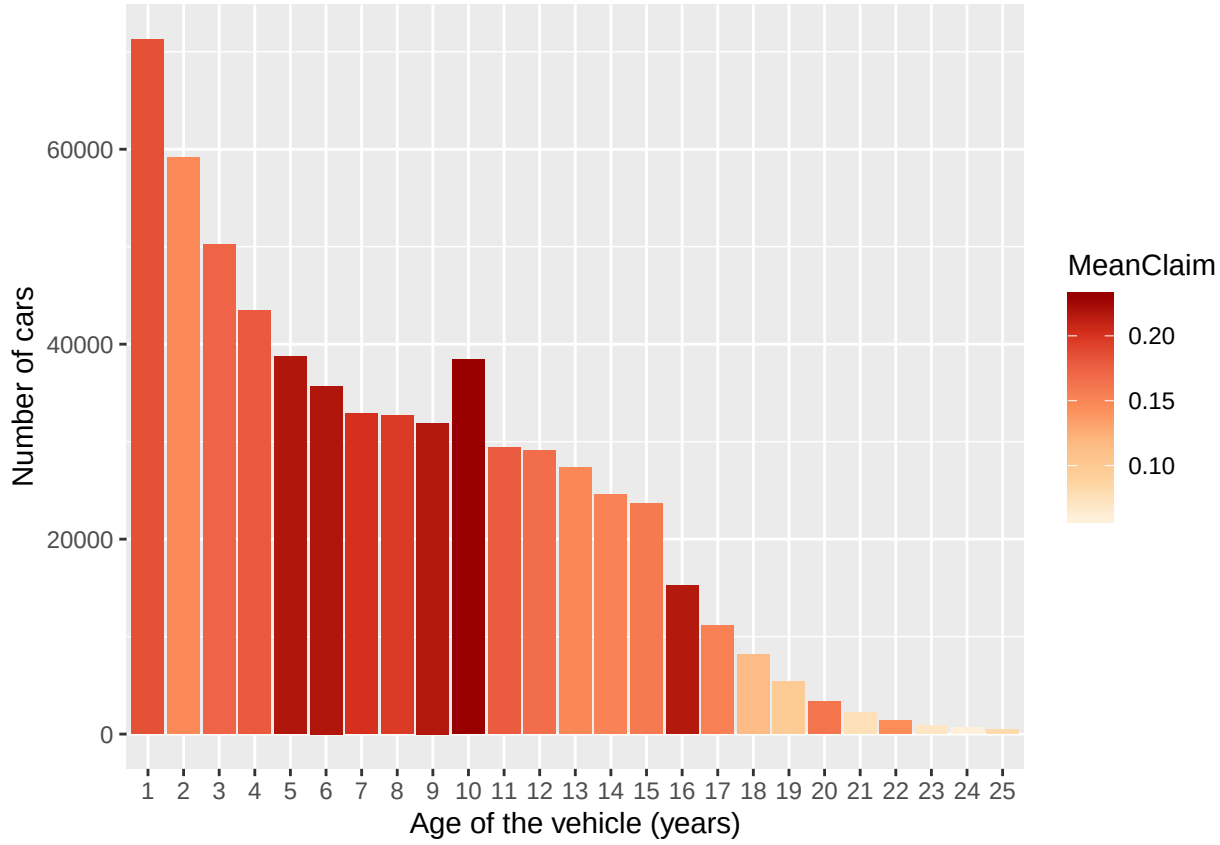
3 Exploratory data analysis

We start by looking at the data available to us to see if some records needs to be filtered out or if there is any data value which is likely to be an error.
First of all, we convert `VehAge` into a factor and we filter out vehicles which are older than 25 years. This is because, when studying pricing models, old vehicles are treated separately and have special pricing rules.

```
dati<- dati %>% mutate(VehAge= as.factor(VehAge))
d<-dati%>% filter (as.numeric(VehAge)>25) %>% summarize(OldVehicles=n())
```

The number of vehicle we are excluding from our analysis is 3114.
Let's see the distribution of the ages of the vehicles for the portfolio being analyzed. The color corresponds to the mean claim frequency[2].

```
dati%>% filter (as.numeric(VehAge)<27 & VehAge!=0) %>% group_by(VehAge)%>%
  summarize(MeanClaim=mean(ClaimNb/Exposure), Count=n()) %>% ggplot(aes(x = VehAge, y=Count)) +
  geom_bar(stat="identity", aes(fill=MeanClaim)) +
  scale_fill_distiller(palette=8, direction = 1)+
  xlab("Age of the vehicle (years)") +
  ylab("Number of cars")
```



We can see that most of the cars are less than 11 years old and that there's no clear pattern which relates the claim frequency to the age of the vehiclem even though it appears that the highest claim frequency is reached for vehicles which are 5 to 10 years old. Let's now focus on the number of claims (`ClaimNb`), which is our target variable.

[2]Vehicles younger than one year have been omitted, since their claim frequency is disproportionately higher than all of the others, hence the color shades would be indistinguishable for all the other vehicle ages. This disproportion can be seen later on as we do some further analysis

```
d<- dati%>% group_by(ClaimNb) %>%  count()
m<- data.frame(ClaimNb=d$ClaimNb, Count=d$n)
m<- t(m)
knitr::kable (m, caption= "Distribution of the number of claims")
```

Table 1: Distribution of the number of claims

ClaimNb	0	1	2	3	4	5	6	8	9	11	16
Count	643953	32178	1784	82	7	2	1	1	1	3	1

As it usually happens in an insurance portfolio, most of the policies do not issue a claim, and those which report claims usually only report one. It can be seen that only few policies have a claim frequency which is equal or higher than 5. We do not consider these policies, since there are probably measurement errors, or they could be non-standard policies (for example a policy for a company vehicle, which is driven by several people and for many kilometers per day).

```
dati<- dati%>% filter(ClaimNb<=5)
dati<- dati%>% filter (as.numeric(VehAge)<27)
```

There are 1211 policies which have a value of Exposure greater than one. We correct for the exposures bigger than one year (by setting them equal to 1) because we believe that this is caused by data errors (since all observations are within one accounting year).

```
dati<- dati %>% mutate(Exposure=pmin(Exposure, 1))
```

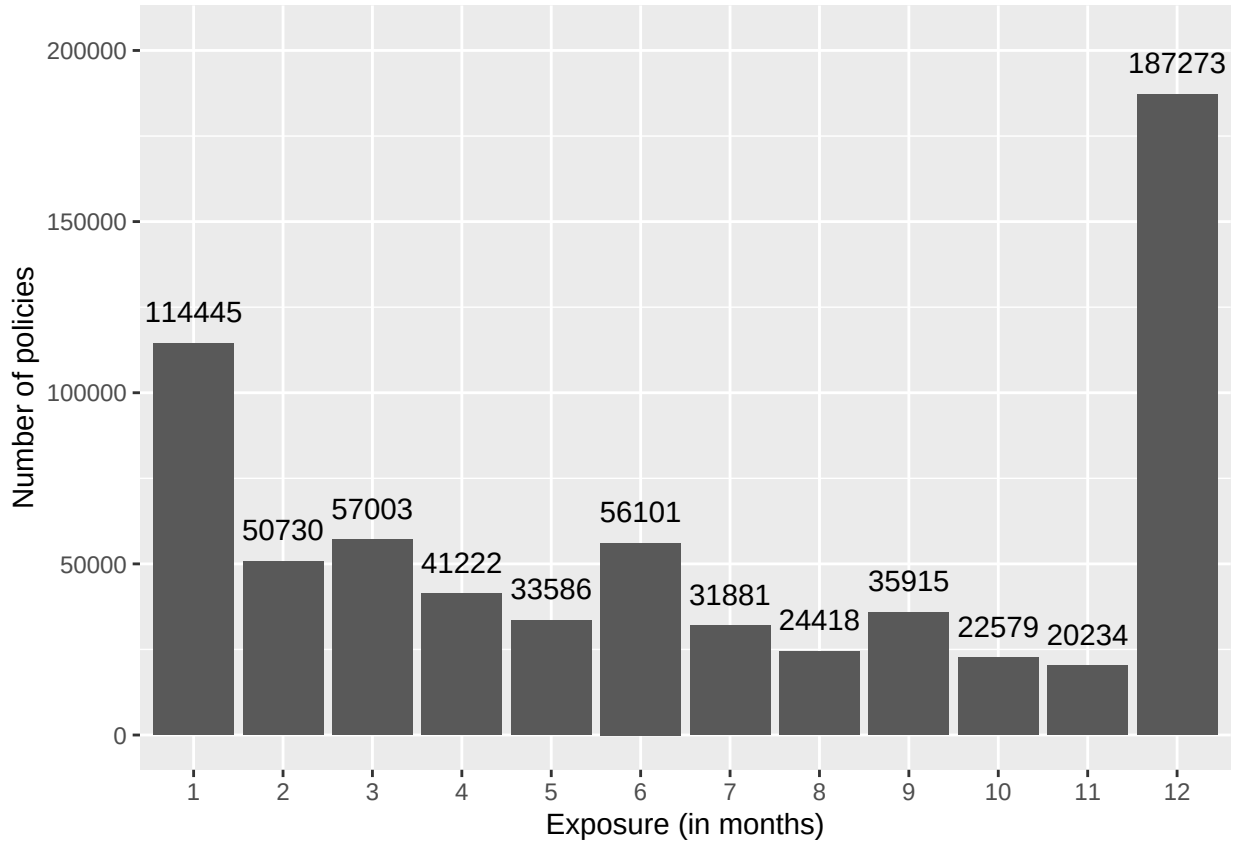
We can compute the average claim frequency in this portfolio, taking into account the different exposures. This is a very important summary measure for the portfolio. Note that in the particular case in which all the policies were observed for the whole year we would divide by the total number of policies, therefore the number below would be the aritmetic mean of `ClaimNb`. However, since many policies are observed for less than a year, when calculating the measurement below it is necessary to "correct" for the exposure, otherwise it will be biased downwards (the numerator is less than n).

```
sum(dati$ClaimNb) /sum(dati$Exposure)
```

```
[1] 0.1007993
```

Now we focus on the exposure of our policies:

```
ExposureF= cut(dati$Exposure, breaks = seq(from = 0, to = 1, by = 1/12))
levels(ExposureF)= c(1:12)
dati <- dati %>% mutate(ExposureF = ExposureF)

dati %>% ggplot(aes(x = ExposureF)) + geom_bar() + xlab("Exposure (in months)")+
  ylab("Number of policies") + geom_text(stat = "count", aes(label = ..count..),
                                         vjust = -1)+ ylim(0,203000)
```



In the barplot above we can see that most of the policies have been observed for (approximately) 12 months, and a considerable amount of policies have been observed for less than a month.
From now on, our variable of interest for the i-th policy will not be $ClaimNb_i$, but it would be $\frac{ClaimNb_i}{Exposure_i}$ instead.

This is because, intuitively, it is not the same thing to have a policy that reports a claim in 2 months or a claim in a whole year of observation. The idea is that if the policy has already reported an accident in two months, its claims rate is expected to be greater than the policy which reported the same number with an observation time 6 times larger. All subsequent analysis will be related to this new measure of interest (that is, claims in relation to the exposure).

Let's now focus on the geographic area and its density.

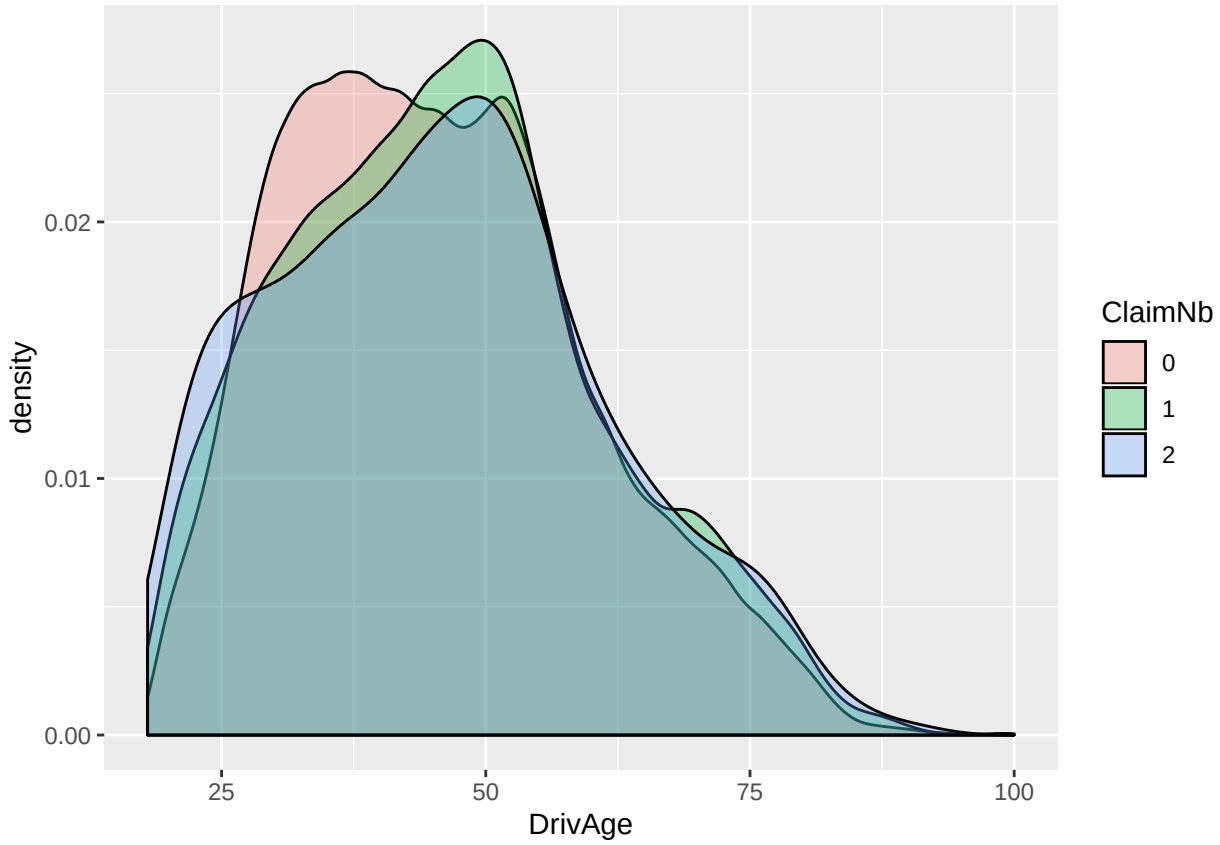
```
dati%>% group_by(Area) %>% summarize(mean_density=mean(Density),
                                     sd_density= sd(Density), number_drivers=n())
```

```
# A tibble: 6 x 4
  Area  mean_density sd_density number_drivers
  <fct>        <dbl>      <dbl>          <int>
1 A             27.5       12.5         103256
2 B             72.6       14.8          75079
3 C            249.       114.          191162
4 D           1076.       399.          151151
5 E           4381.      1953.          136845
6 F          22008.      6391.           17894
```

We can see that the levels of `Area` are (alphabetically) ordered according to the level of `Density`: `A` is the lowest-density area and `F` is the higher density area. Therefore, the two variables kind of give the same information, even though `Density` gives an higher level of detail since it distinguishes units within the same level of `Area`.

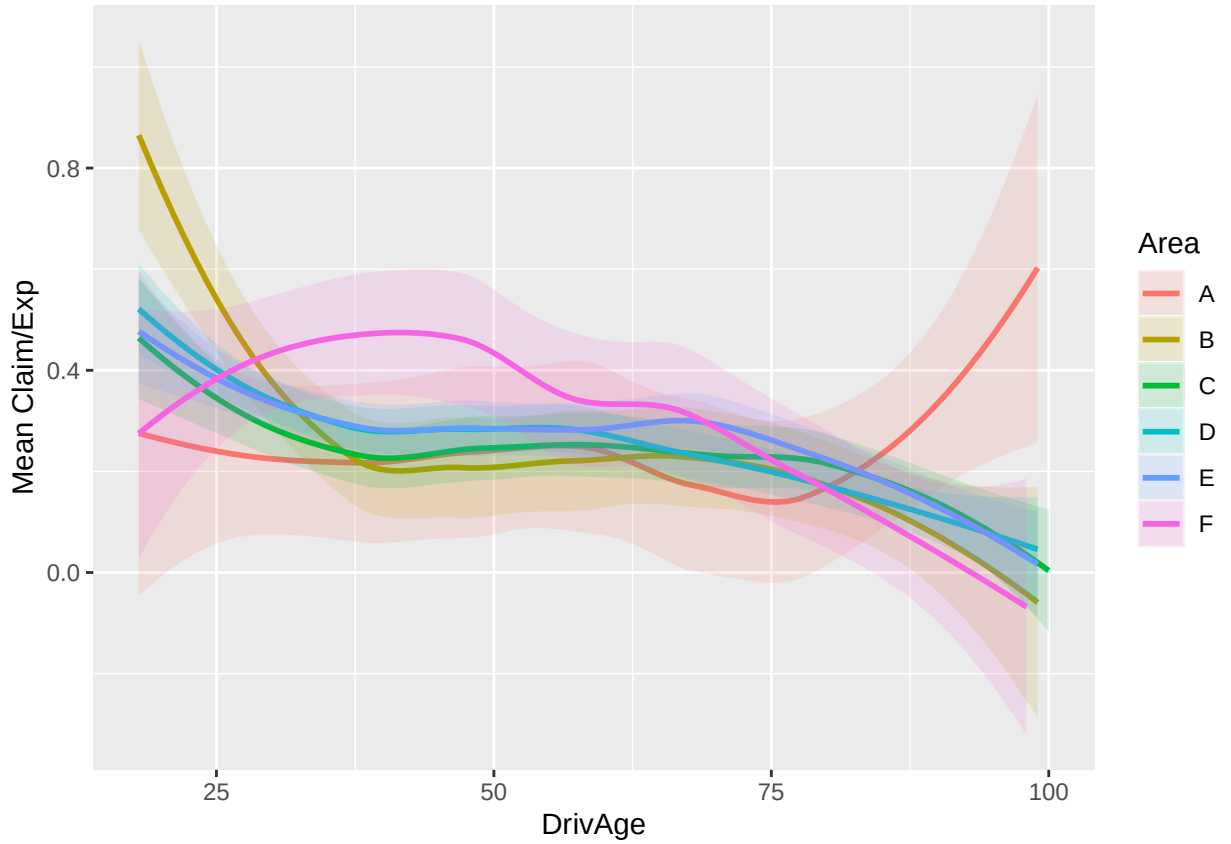
Let's see the relationship between age and claim frequency. The number of claims is filter so that it is not higher than 3, otherwise we would have too little data.

```
dati%>%filter (ClaimNb<3)%>% mutate(ClaimNb=as.factor(ClaimNb))%>%
    ggplot(aes(DrivAge, fill = ClaimNb)) + geom_density(alpha=0.3)
```



It might be interesting to see if the relationship between `DrivAge` and `ClaimNb/Exposure` changes depending on the area in which the driver lives. One can expect, for example, that in areas with greater density the claims rate is higher for each fixed age, but we want to see if the relationship has different forms according to the area.

```
dati %>% group_by(DrivAge, Area) %>% summarize(MeanClaims=(mean(ClaimNb/Exposure))) %>%
  ggplot (aes(x=DrivAge, y=MeanClaims, col=Area))+
   stat_smooth(se=TRUE, aes(fill=Area), alpha=0.15)+ ylab("Mean Claim/Exp")
```



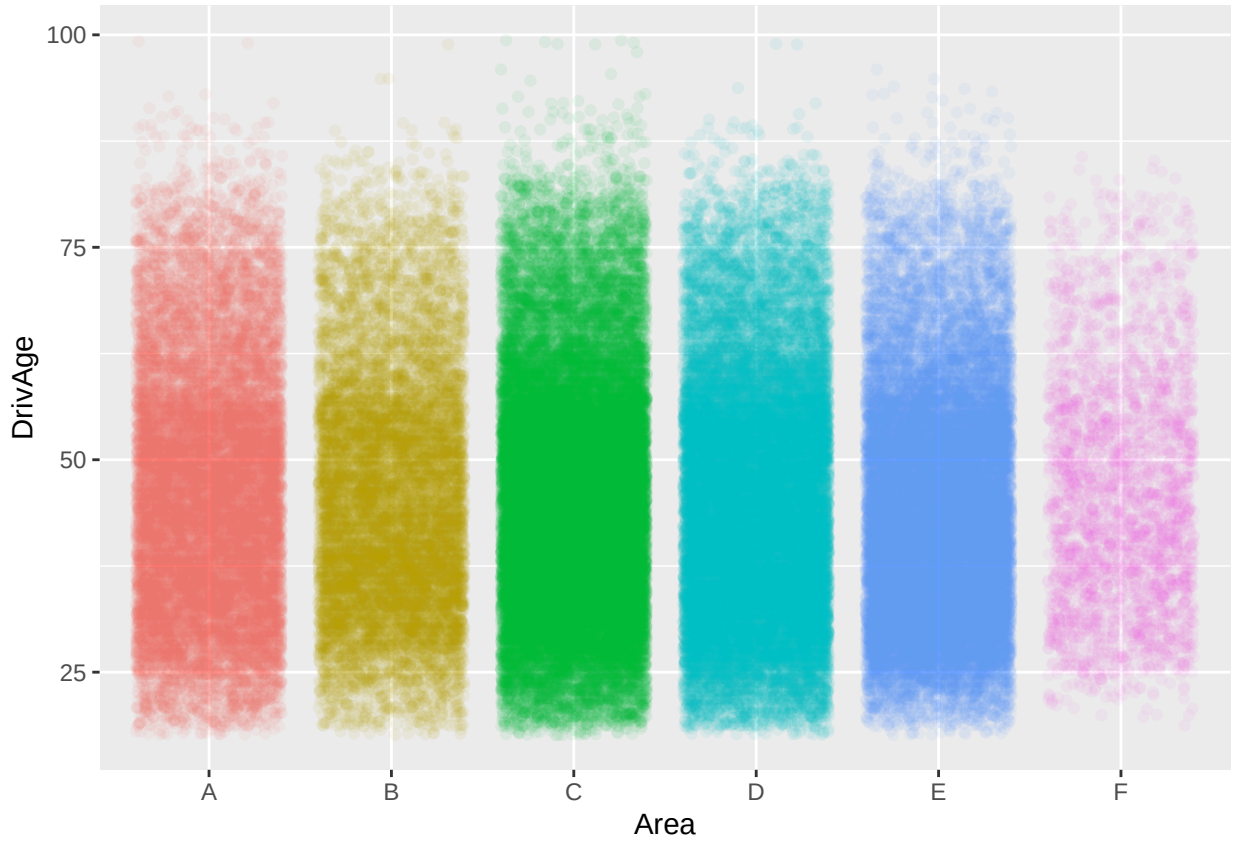
In the graph above we can see that in highly populated areas (D, E, F) the claim rate is generally higher than in low-density areas (A, B, C). The relationship changes at the extremes: it can be seen that, for example, for ages over 77, the average of claims in area F (higher absolute density) is lower than all the other areas, and in particular it is much lower than in area A (low density). This result could be due to the fact that in high-density areas, old people use the car much less than people of the same age who live in sparsely populated areas. A similar relationship occurs in the left end of the graph (age below 25) if we compare area F and area B: in this case we can think that young people living in more populated areas use the car with less frequency than peers living in area B (maybe because of the poorer transport service etc). However, information on the actual use (in terms of km traveled) for each individual policyholder is not available: we only know that there is a policy for that particular policyholder. From now on, we will be working with a subset of records given that processing more than 600,000 policies for the

subsequent graphs becomes computationally expensive (not to mention estimating a model, as we will be doing in later sections). A simple random sample of 100 000 policies will be considered.

```
#random sampling of units (100 000 policies)
set.seed(3)
ind<- sample(c(1:675387), 100000, replace = FALSE)
dati<- dati[ind,]
```

Let's see the distribution of ages for the driver with respect to the area of France where they drive.

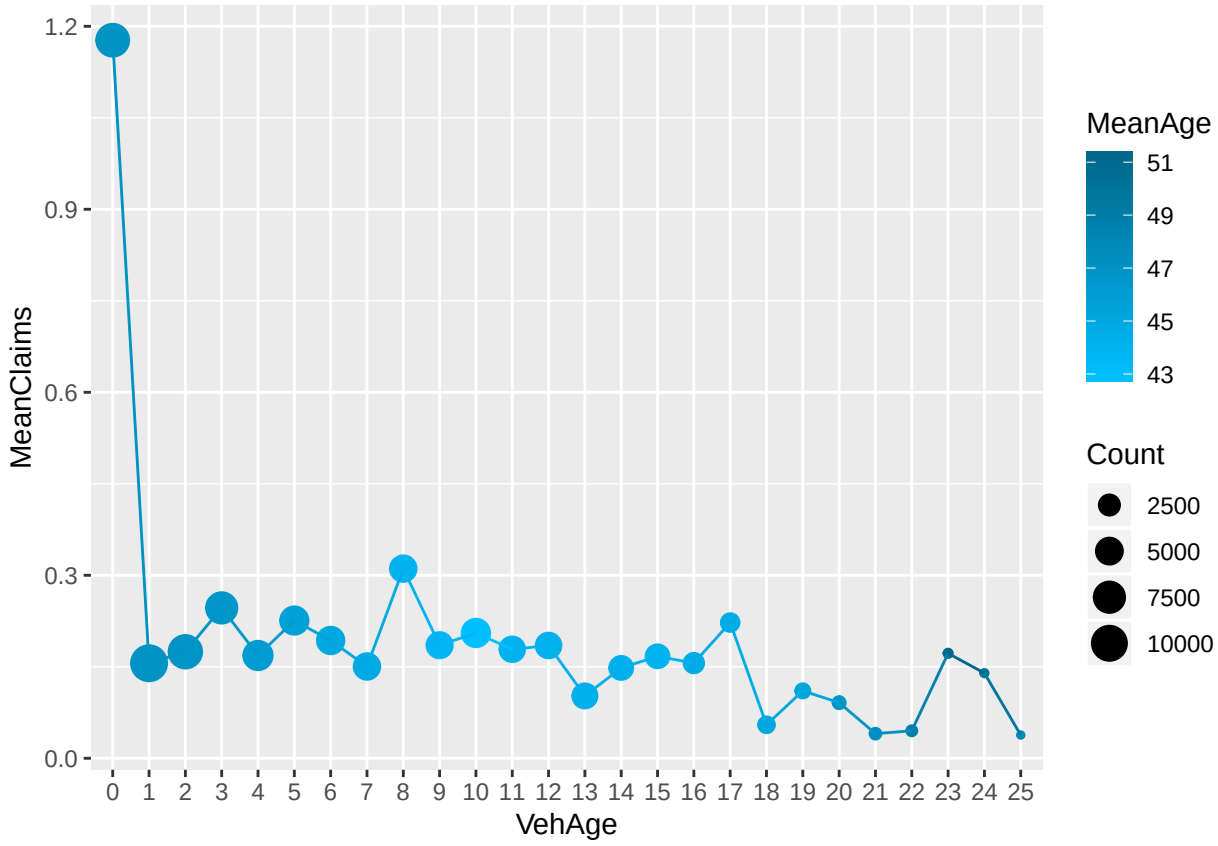
```
dati%>% filter(Area!= "NA")%>%ggplot(aes(x=Area, y=DrivAge))+
  geom_jitter(aes(col=Area), alpha=0.07) +
  theme(legend.title = element_blank())+
  theme(legend.position = "none")
```



It can be seen that most of the observations are concentrated in the non-extreme age groups (from 25 to 60), while for younger and older ages there are fewer policies. This explains the highest standard errors for extreme ages in the previous chart. The area F is where the lowest number of policies is observed.

The following chart aims at understanting the relation among the claim frequency, the vehicle age and the driver's age.

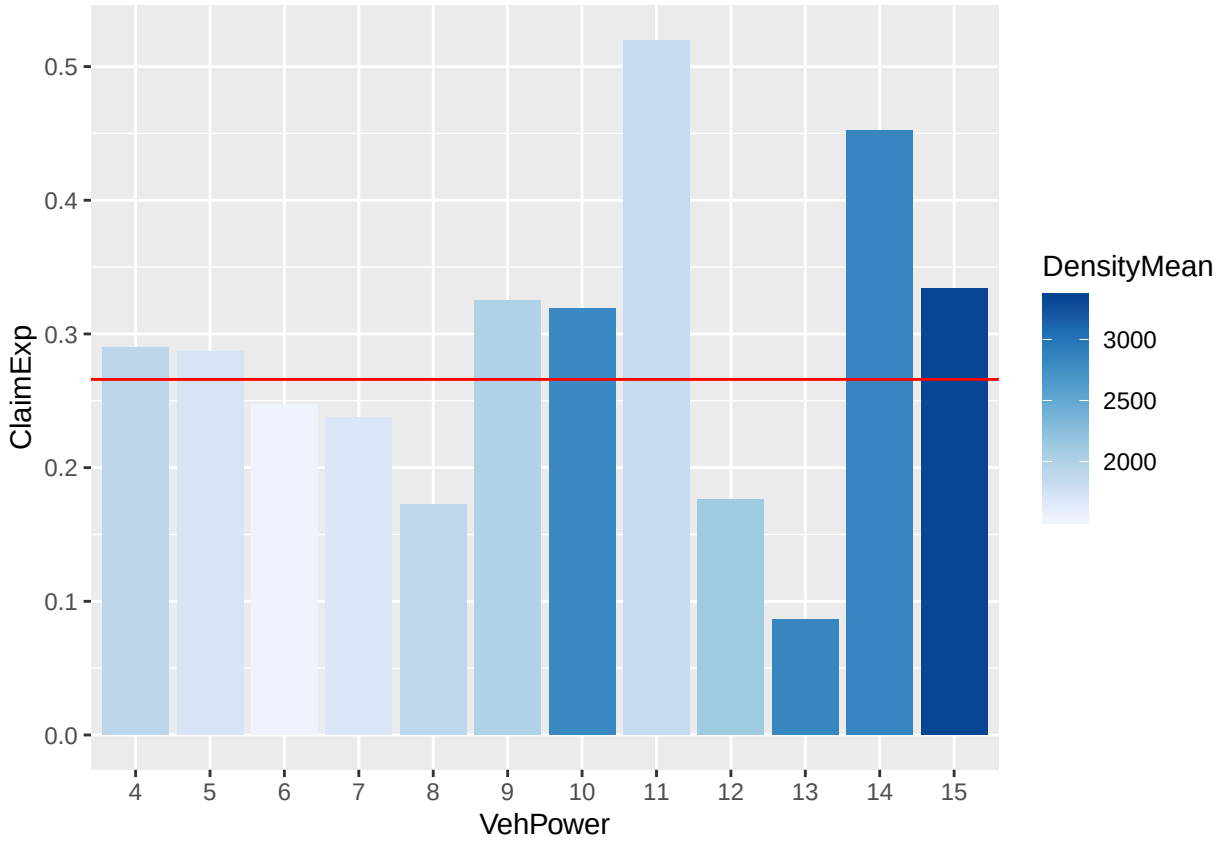
```
dati %>% group_by(VehAge) %>% filter(VehAge!="NA")%>%
  summarize(MeanClaims=mean(ClaimNb/Exposure),
    MeanAge=mean(DrivAge), Count=n()) %>%
  ggplot (aes(x=VehAge, y=MeanClaims, col=MeanAge))+
  geom_point(aes(size=Count)) + geom_line(aes(group=1))+
  scale_colour_gradient(low = "deepskyblue", high = "deepskyblue4")
```



It can be seen that in general older vehicles are owned by drivers which are older on average, and who have on average fewer claims (always in relation to the exposure), while younger drivers (lightblue) have vehicles aged between 7 and 18. It can also be seen that older cars generate on average fewer accidents (probably due to the fact that they are used by more experienced drivers, or they are used for shorter trips). The dot size reflects the number of cars for each given vehicle age. It can therefore be seen that the peak of claims for cars less than one year old is not caused by the fact that we have little data, thus higher variability.

The following graph shows the claim frequency summarized for each Power category. The bars are filled according to the mean of the density where each Power class circulates the most. The horizontal line is the mean of $\sum_{i=1}^{100000}(\frac{ClaimNum_i}{Exp_i})$.

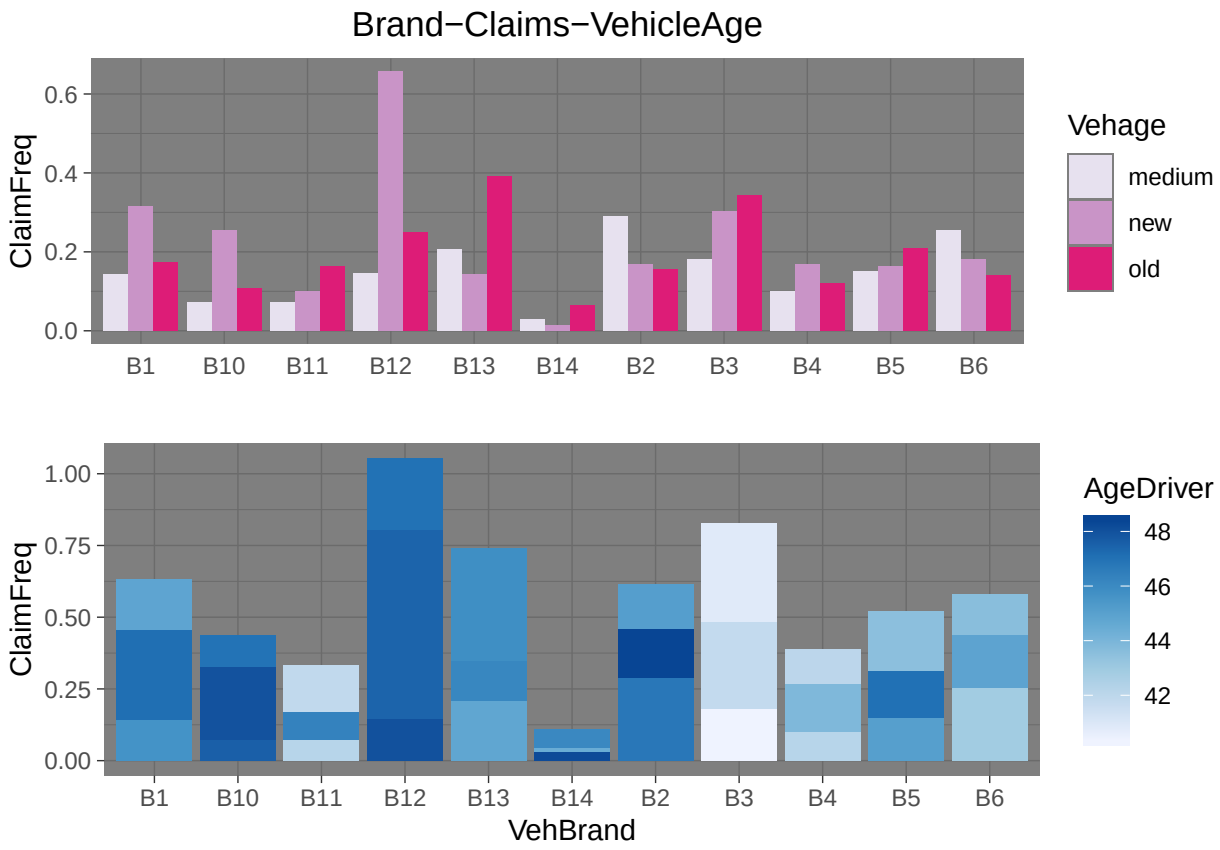
```
dati %>% mutate (VehPower= as.factor(VehPower))%>%group_by(VehPower)%>% filter(VehPower!="NA")%>%
  summarize(ClaimExp= mean(ClaimExp), DensityMean=mean(Density))%>%
  ggplot() + geom_bar(stat="identity", aes(x=VehPower,
                                           y=ClaimExp, fill=DensityMean)) +
geom_hline(aes(yintercept = 0.266), color="red")+
  scale_fill_distiller(palette=1, direction = 1)
```



The most powerful vehicles are those with the higher claim frequency, however it can be seen clearly that they mostly circulate in high-density areas, therefore the higher claims rate could be due to the higher populated area rather than to the power of the car itself. The following plots analyze the relationship:

- between the car brand and the vehicle mean age for each brand (first plot);
- between the car brand and the mean age of the driver. Each bar in the second plot is in turn divided into 3 bands. The three different shades correspond (from the bottom to the top) to the mean age of the driver for middle-aged, new and old cars respectively.

```
d<- dati%>% mutate(VehAge=as.numeric(VehAge), Vehage=ifelse(VehAge<4, "new",
                          ifelse(VehAge>4 & VehAge<9, "medium", "old"))) %>%
group_by(VehBrand, Vehage) %>% filter(VehBrand!="NA")%>%
  summarize(ClaimFreq=mean(ClaimExp), AgeDriver=mean(DrivAge),
                                      AgeVehic=mean(as.numeric(VehAge)))

p1<-d%>%ggplot(aes(x=VehBrand))+
  geom_bar(aes(y=ClaimFreq, fill=Vehage), stat="identity", position="dodge")+
  scale_fill_brewer(palette = 11)+ theme_dark()+
  theme(plot.title = element_text(hjust = 0.5),
 axis.ticks.x=element_blank())+
  ggtitle("Brand-Claims-VehicleAge")+xlab("")


  p2<- d%>% ggplot(aes(x=VehBrand))+
  geom_bar(aes(y=ClaimFreq, fill=AgeDriver), stat="identity")+
  scale_fill_distiller(palette=1, direction = 1)+ theme_dark()

grid.arrange(p1, p2, nrow=2)
```

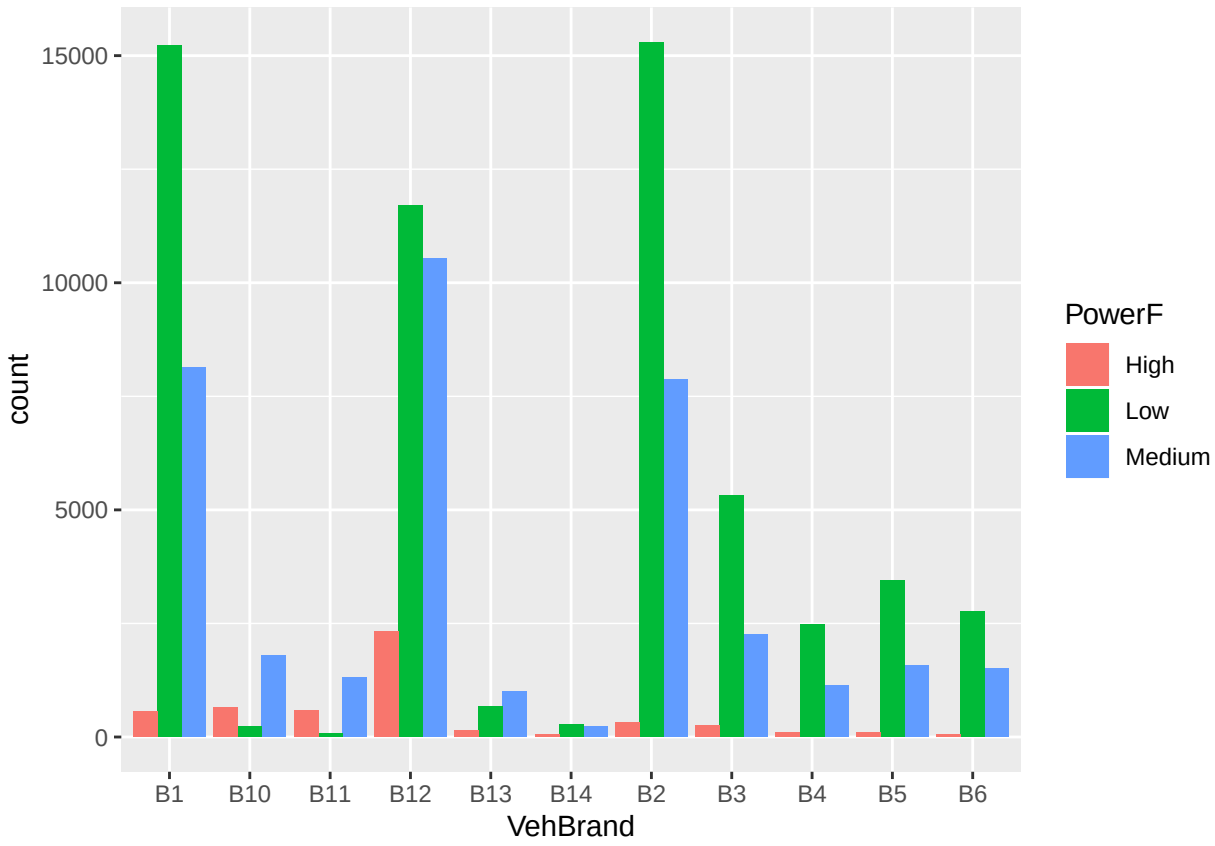


The higher claim frequencies are reached by the car brands `B12`, `B3`, `B13` and `B2` (bar chart 2). There's no clear pattern as to which cars (middle-aged, new or old) are in general more prone to issuing a claim (bar chart 1: the higher bars within the same car brand are sometimes white, sometimes light purple and sometimes pink).
Moreover, we can understand some more things from the second plot:

- some car brands (B11, B3, B4 and B6) are driven by younger people (on average), which suggest they are probably less powerful cars;
- the colour corresponding to the middle bands (= new cars, that is with less than 4 years of age) is in most cases darker than the bands above and below: it means that, given a car brand, the drivers with a newer car of that brand are the oldest, while older cars of every brand are generally driven by less experienced drivers (younger mean age).

We can gain some better insight on the relationship between the vehicle brand and the car power with the following barplot:

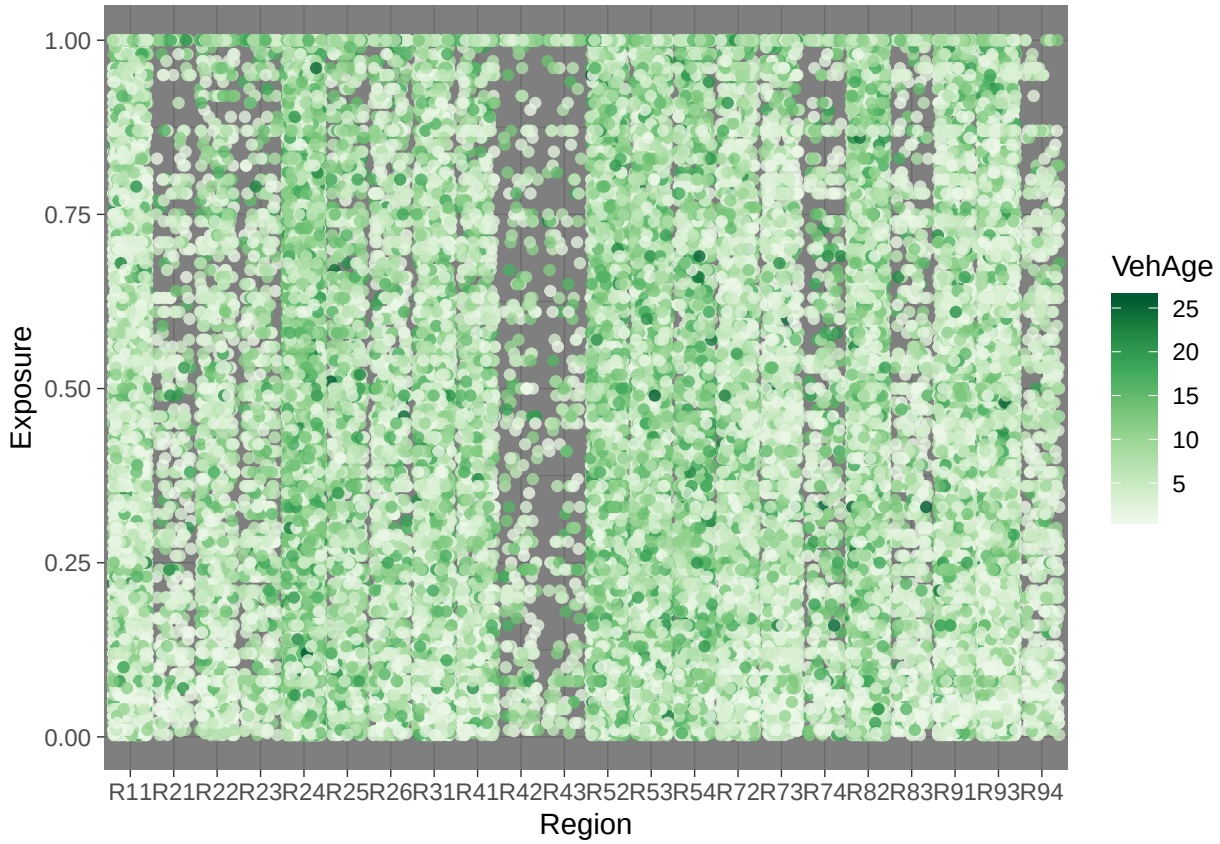
```
dati%>%mutate(PowerF=ifelse(VehPower<7, "Low", ifelse(VehPower>6 & VehPower<11,
                "Medium", "High")))%>% filter(PowerF!="NA")%>%
   ggplot(aes(VehBrand, fill = PowerF)) + geom_bar(position="dodge")
```



It can be seen that, as we guessed, brands `B3`to `B6` are those with the lowest proportion of powerful cars (`PowerF=High` if `VehPower`>=11). However, differently from what we deduced by the previous barplot, car brand `B11` is quite powerful, despite being driven by drivers whose mean age is among the youngest. The following plot allows to see:

- the exposure for each region: some areas have observations which spread across all levels of exposure, while some others have a majority of observations which is more concentrated for low levels of exposure. For some regions (such as B42 and B43) fewer observations are available. It also seems that the oldest cars are in regions R24, R52, R53 and R54 (the table below confirms this fact).
-

```
dati%>% filter(Region!="NA")%>% mutate(VehAge=as.numeric(VehAge))%>%
  ggplot(aes(x=Region, y=Exposure))+
  scale_colour_distiller(palette=5, direction = 1)+
  geom_jitter(aes(col=VehAge), alpha=0.8) + theme_dark()
```



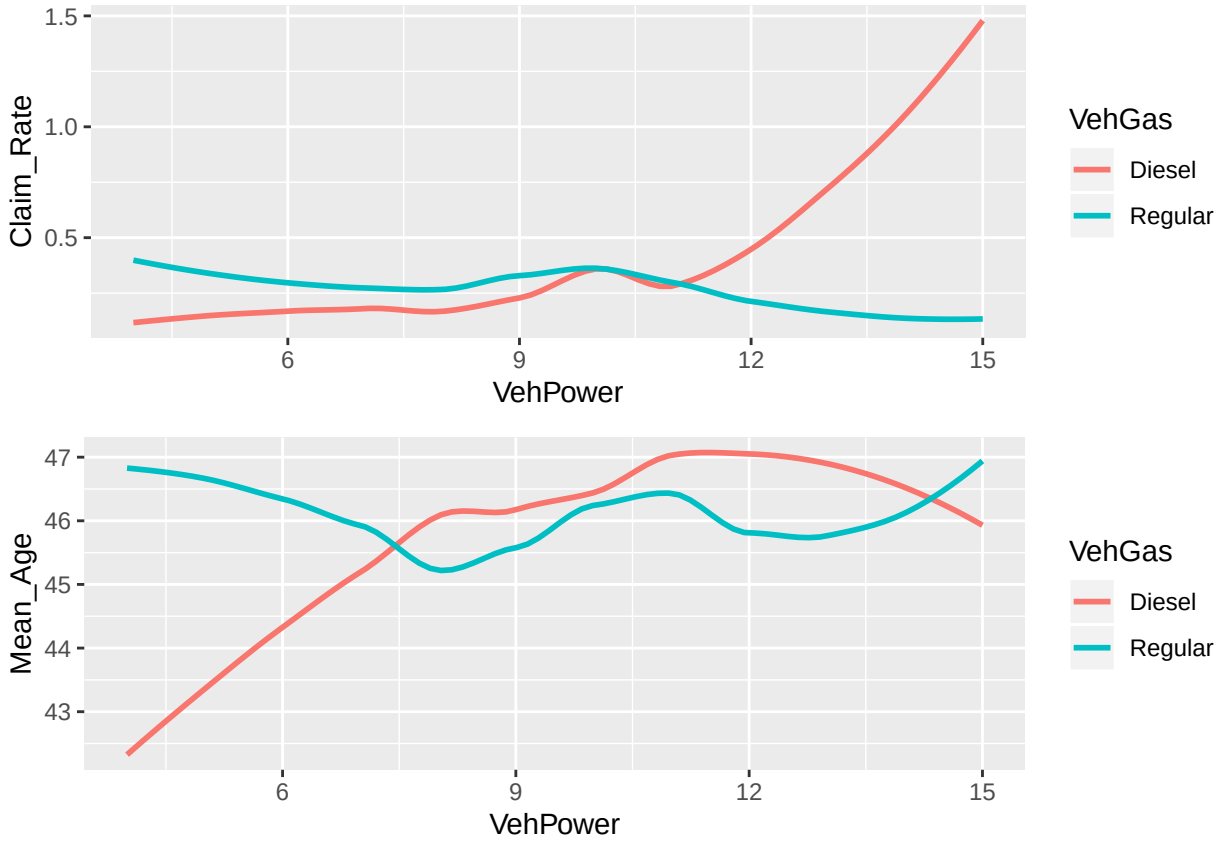

```
age<- dati%>% filter(Region!="NA")%>% mutate(VehAge=as.numeric(VehAge))%>%group_by(Region)%>%
  summarize(MeanVehicleAge=mean(VehAge))%>% arrange(desc(MeanVehicleAge))%>% head(5)

knitr::kable(age)
```

Region	MeanVehicleAge
R24	9.737670
R54	9.511031
R52	9.088973
R53	8.668578
R25	8.193568

Finally, we would like to explore the relationship between the car power and the claims rate in relation to the type of fuel. It seems that the number of claims increases as the power of the vehicle increases but only for cars with Diesel fuel. We want to see therefore if this behaviour was due to some other "hidden" feature of Diesel vehicles: for example, we want to verify if powerful Diesel vehicles are driven by quite old or quite young drivers. It this is true, age could explain the behaviour of Diesel cars described above. However (second chart), both Regular and Diesel cars seems to be driven by drivers with similar ages (age which increases with the power).
This means that age is not the reason for the increasing claims rate of Diesel vehicle, but maybe it is due to the fact that Diesel cars are usually used for people who drive a lot (hence the car fuel would be a proxy variable for "distance in km", which is unknown to us but which would be a relevant indicator for the number of claims issue, since one would expect that the more kilometers a drievr runs, the more likely it is that he has an accident).

```
c<- dati %>%
  group_by(VehGas,VehPower) %>%
  summarize(Claim_Rate=mean(as.double(ClaimNb)/(Exposure))) %>%
  ggplot(aes(x=VehPower, y=Claim_Rate, color=VehGas)) + geom_smooth(se=FALSE)
```

```
c2<- dati%>% group_by(VehPower, VehGas)%>% summarize(M=mean(DrivAge)) %>% ggplot()+
  geom_smooth(aes(x=VehPower, y=M, col=VehGas), se=FALSE)+ ylab("Mean_Age")

grid.arrange(c,c2, nrow=2)
```



4 Generalized linear models for the number of claims

First, we grouped some continuous variables to form factors, as it is usually the case with pricing models in insurance. One of the advantages to grouping data in this way is that is that the model is more stable and able to generalize better. Moreover, insurance companies usually collect information on the future insured person/thing based on factors (age, sex, phisical features, vehicle power, age of the vehicle grouped into classes etc...) because it allows not to change the insurance premium from year to year. The example is an insured person whose age is `x` and whose vehicle age is `xVeh`: each year both `x` and `xVeh` increase by one, therefore if those two variables were treated as continuous variables, the GLM model would cause the premium to (likely) increase year after year. If both variables were categorical, instead, the premium would remain the same for all ages (`x` and `xVeh`) within the same factor level. The variables which are now turned into factors are **Vehpower**, **VehAge**, **DrivAge** y **BonusMalus**.

DrivAge is turned into a factor with levels based on the ones insurance companies usually use: **18-25**, **26-35**, **36-45**, **46-60** and **>60**.

```
DatiGLM<-dati
DatiGLM$age_groups <- ifelse(dati$DrivAge <= 25, "18-25",
                          ifelse(dati$DrivAge <= 35, "26-35",ifelse(dati$DrivAge <= 45,
                           "36-45",ifelse(dati$DrivAge <= 60, "46-60",">60") ) ))
```

As for **VehPower**, we take a look at the quantiles to decide the ranges corresponding to the levels. By doing so we ensure that the levels have the same number of units.

```
quantile(dati$VehPower, prob=c(0.2,0.4,0.6,0.8))
```

```
20% 40% 60% 80%
  5   6   7   8
```

The levels suggested are **<=5**, **5-6**, **6-7**,**7-8** and **>8**.

```
DatiGLM$power_groups <- ifelse(DatiGLM$VehPower <= 5, "<=5",
                           ifelse(DatiGLM$VehPower <= 6, "5-6",
                      ifelse(DatiGLM$VehPower <= 7, "6-7",
                  ifelse(DatiGLM$VehPower <= 8, "7-8",">8") ) ))
```

We did the same for *__VehAge__. The resulting groups are **<=1**, **2-3**, **4-6**,**7-8** y **>8**.

```
DatiGLM<-DatiGLM%>% mutate(VehAge=as.numeric(VehAge))
DatiGLM$age_veh_groups <- ifelse(DatiGLM$VehAge <= 1, "<=1",
                             ifelse(DatiGLM$VehAge <= 3, "2-3",
                             ifelse(DatiGLM$VehAge <= 6,
                   "4-6",ifelse(DatiGLM$VehAge <= 8, "7-8",">8") ) ))
```

The variable **BonusMalus** is however quite different since it has a really wide range. We looked for a split which results in 5 groups as homogeneous as possible, which ensures the levels keep the meaning of the variable, that is the **Bonusmalus** class must reflect the fact that the driver was a good or a bad driver.
The classes are **<=54**, **54-65**, **65-85**, **85-110**, **110-135**. y **>135**.

```
DatiGLM$bonusmalus_groups <- ifelse(dati$BonusMalus <= 54, "<=54",
                              ifelse(dati$BonusMalus <= 65, "54-65",
                              ifelse(dati$BonusMalus <= 85, "65-85",
                              ifelse(dati$BonusMalus <= 110, "85-110",
                        ifelse(dati$BonusMalus <= 135, "110-135",">135" ) ))))
```

All newly created variables are turned into factors (they are originally characters)

```
DatiGLM$VehGas<-as.factor(DatiGLM$VehGas)
DatiGLM$age_groups<-as.factor(DatiGLM$age_groups)
DatiGLM$power_groups<-as.factor(DatiGLM$power_groups)
DatiGLM$age_veh_groups<-as.factor(DatiGLM$age_veh_groups)
DatiGLM$bonusmalus_groups<-as.factor(DatiGLM$bonusmalus_groups)
```

We are using a subset of data (3/4 of the 100000 records we selected) with which we will be working from now on to train the model. The remaining 25000 units will be used as a test set instead. To ensure we are always using the same units to build the models we use `set.seed(3)` so that the random sample of indices always results in the same units being selected.

```
set.seed(3)
indici<-sample(c(1:100000), 75000, replace=FALSE )
trainGLM<- DatiGLM[indici, ]
```

The variables which are not useful for the analysis (such as **PolicyID**) are removed.

```
trainGLM<-trainGLM[,c(2,3,4,9,10, 11, 12, 15,16, 17,18)]
testGLM<-DatiGLM[-indici, ]
testGLM<-testGLM[,c(2,3,4,9,10, 11, 12, 15,16, 17,18)]
```

To analyze the dependent variable, the method we will use is a generalized linear model. It is used to model the parameter (expected value) λ_i of the Poisson variable according to the feature vector $\vec{x}_i$. The Poisson family is part of the so-called Exponential families (a variable which belongs to such family has a probability/density function with the following form $f(x;\theta) = h(x)c(\theta)exp(w(\theta)t(x)))$. The probability function for a Poisson variable is $p(y) = \frac{\lambda^y}{y!}e^{-\lambda} = exp\{-\lambda + ylog(\lambda) - log(y!)\}$, which can be rewritten in order to match the form required by a member of the exponential family for some given functions $h()$, $c()$, $w()$ and $t()$.

A GLM model is estimated to model the mean number of claims per unit time. Defining a GLM involves three steps:

1. Define a probability model for the target variable:

$$y_i \sim Poiss(\lambda_i),\ i \in \{1, ..., n\}, \quad n = 75000$$

2. Define a linear predictor:

$$\eta_i = x_i^T \beta$$

3. Define a function of the linear predictor which is equal to the mean of the target variable, or equivalently (since the function is invertible), define a function of the mean that is equal to the linear predictor. In this case the logaritmic function will be used:

$$\eta_i = log(\lambda_i)$$

In this dataset we have a response variable (a Poisson distributed count variable **ClaimNb**) which is linked to the exposure of the individual insured: if a vehicle is observed for a longer period, then it is more likely that the number of claims is higher. The exposure can be taken into account by adding it to the linear predictor, which will act as a shift. In fact:

$$log(\frac{\lambda_i}{exp_i}) = x_i^T\beta \ \rightarrow\ log(\lambda_i) = x_i^T\beta + log(exp_i)$$

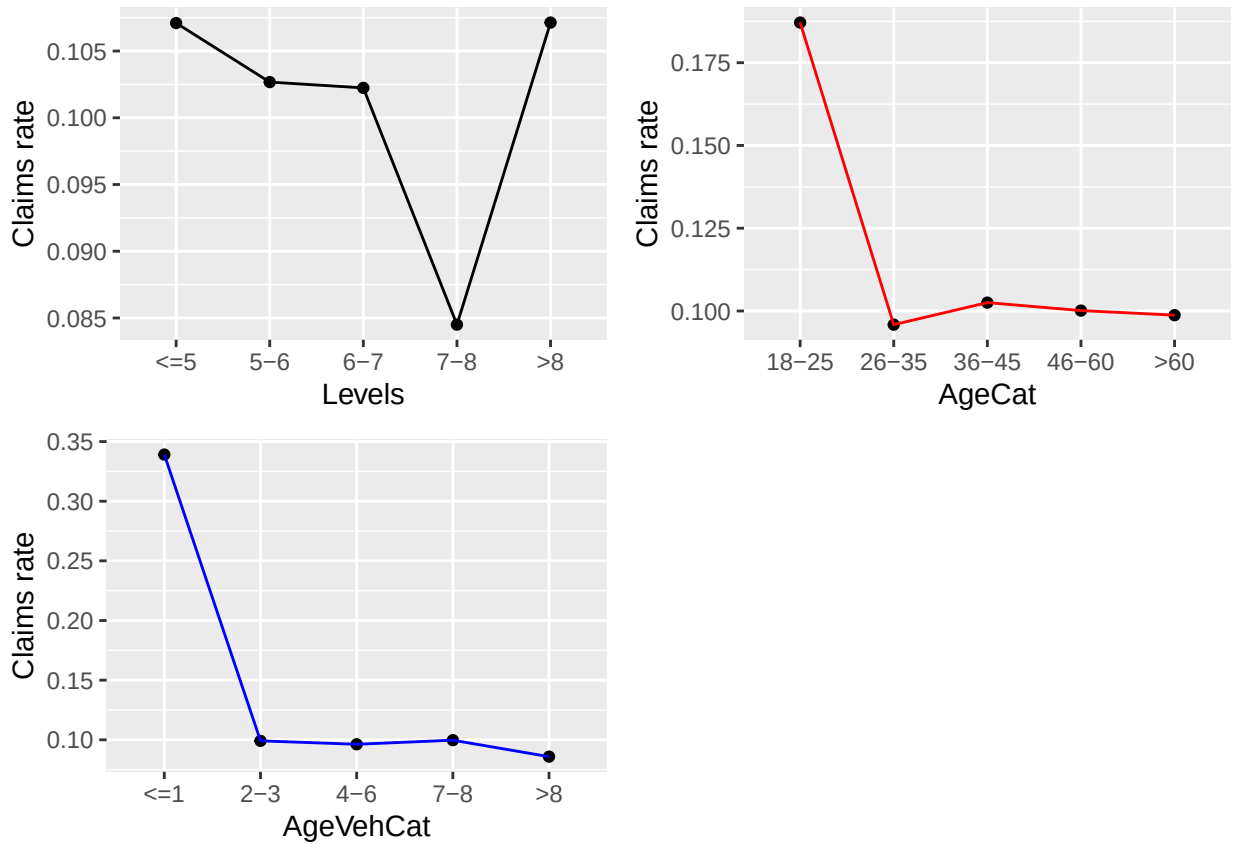
.

We start by fitting simple GLM models which only use one variable, and to do this we choose variables that also in the exploratory analysis seemed the most relevant to explain the number of claims. The first one, `fitPower`, uses the vehicle power classes which result from the previous division.

```
fitPower<-glm(ClaimNb~power_groups, family = poisson, data = trainGLM,
              offset = log(Exposure))
summary(fitPower)
```

```
Call:
glm(formula = ClaimNb ~ power_groups, family = poisson, data = trainGLM,
    offset = log(Exposure))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.4629  -0.4317  -0.3067  -0.1592   7.3286

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -2.23400    0.02574 -86.783   <2e-16 ***
power_groups>8   0.00033    0.04987   0.007   0.9947
power_groups5-6 -0.04219    0.04147  -1.017   0.3089
power_groups6-7 -0.04642    0.04257  -1.091   0.2755
power_groups7-8 -0.23698    0.07260  -3.264   0.0011 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 25321  on 74999  degrees of freedom
Residual deviance: 25309  on 74995  degrees of freedom
AIC: 33181

Number of Fisher Scoring iterations: 6
```

We do the same with **age** and **AgeVehCat**, and then we visualize how the expected frequency of claims varies as **Power**, **Age** and **VehAge** (in groups) vary.



4.1 Model selection with stepAIC

The `step()` command has been used to select the best model according to both the Akaike information criterion (**AIC**). Briefly, the **AIC** is a penalized-likelihood criterion which tries to balance good fit with parsimony[3]. It is therefore, together with the **BIC** measure, a useful mean for model selection: the lower their value the better the model. They differ in the size of the penalty: BIC penalizes model complexity more heavily, therefore we expect the model chosen according to the AIC to be larger then the one selected according to the BIC.
The model of `class glm` which is passed to the `step()` function is a model with all the variables (which is named `fit`): starting from this model, at each iteration a variable is removed or added (`direction="both"`) and the AIC is computed. At the end of the iterative procedure the best model[4] according to the **AIC** is returned.
Another way in which the selection of the model could be carry out is by using the residual deviance to perform appropriate tests to compare models on the base of this quality-of-fit measure.
The `summary(fitSTEP)` command allows to show the selected model.

```
fit<- glm(ClaimNb~., family = poisson, data = trainGLM, offset = log(Exposure))
fitSTEP <- stepAIC(fit, direction = "both")
```

```
summary(fitSTEP)
```

```
Call:
glm(formula = ClaimNb ~ Exposure + VehGas + Region + age_groups +
    power_groups + age_veh_groups + bonusmalus_groups, family = poisson,
    data = trainGLM, offset = log(Exposure))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.2798  -0.3590  -0.3121  -0.2247   7.0364
```

[3] the idea is to use as few predictors as possible to avoid overfitting
[4] However, this is not necessarily the absolute best model because only a subset of possible models are tested.

```
Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -0.610824   0.080886  -7.552 4.30e-14 ***
Exposure                 -1.266241   0.054020 -23.440  < 2e-16 ***
VehGasRegular             0.132395   0.033022   4.009 6.09e-05 ***
RegionR21                 0.621002   0.185921   3.340 0.000837 ***
RegionR22                -0.018787   0.153471  -0.122 0.902571
RegionR23                -0.442274   0.209755  -2.109 0.034985 *
RegionR24                 0.030699   0.059739   0.514 0.607331
RegionR25                 0.190587   0.120181   1.586 0.112779
RegionR26                -0.035033   0.136633  -0.256 0.797639
RegionR31                -0.034820   0.097062  -0.359 0.719791
RegionR41                -0.335069   0.140103  -2.392 0.016776 *
RegionR42                 0.269708   0.228638   1.180 0.238147
RegionR43                -1.062786   0.579367  -1.834 0.066596 .
RegionR52                 0.147422   0.080181   1.839 0.065970 .
RegionR53                 0.250624   0.074436   3.367 0.000760 ***
RegionR54                -0.078995   0.112500  -0.702 0.482569
RegionR72                -0.115147   0.094342  -1.221 0.222263
RegionR73                -0.307970   0.127074  -2.424 0.015370 *
RegionR74                 0.165267   0.185931   0.889 0.374078
RegionR82                 0.118743   0.063766   1.862 0.062579 .
RegionR83                -0.614723   0.247183  -2.487 0.012886 *
RegionR91                -0.056196   0.087708  -0.641 0.521705
RegionR93                 0.042191   0.066062   0.639 0.523053
RegionR94                 0.269943   0.166944   1.617 0.105884
age_groups18-25          -0.208416   0.077473  -2.690 0.007142 **
age_groups26-35          -0.514658   0.057967  -8.879  < 2e-16 ***
age_groups36-45          -0.149225   0.050549  -2.952 0.003156 **
age_groups46-60          -0.056062   0.046991  -1.193 0.232851
power_groups>8           -0.035815   0.050351  -0.711 0.476895
power_groups5-6           0.095461   0.043022   2.219 0.026494 *
power_groups6-7           0.005797   0.042919   0.135 0.892563
power_groups7-8          -0.253783   0.073598  -3.448 0.000564 ***
age_veh_groups>8         -1.121276   0.051022 -21.976  < 2e-16 ***
age_veh_groups2-3        -1.015767   0.056043 -18.125  < 2e-16 ***
age_veh_groups4-6        -0.982398   0.055877 -17.581  < 2e-16 ***
age_veh_groups7-8        -0.928988   0.064916 -14.311  < 2e-16 ***
bonusmalus_groups>135     2.132536   0.238437   8.944  < 2e-16 ***
bonusmalus_groups110-135  1.542451   0.108166  14.260  < 2e-16 ***
bonusmalus_groups54-65    0.557907   0.047523  11.740  < 2e-16 ***
bonusmalus_groups65-85    0.496066   0.049702   9.981  < 2e-16 ***
bonusmalus_groups85-110   0.837989   0.059264  14.140  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 25321  on 74999  degrees of freedom
Residual deviance: 23396  on 74959  degrees of freedom
AIC: 31341

Number of Fisher Scoring iterations: 6
```

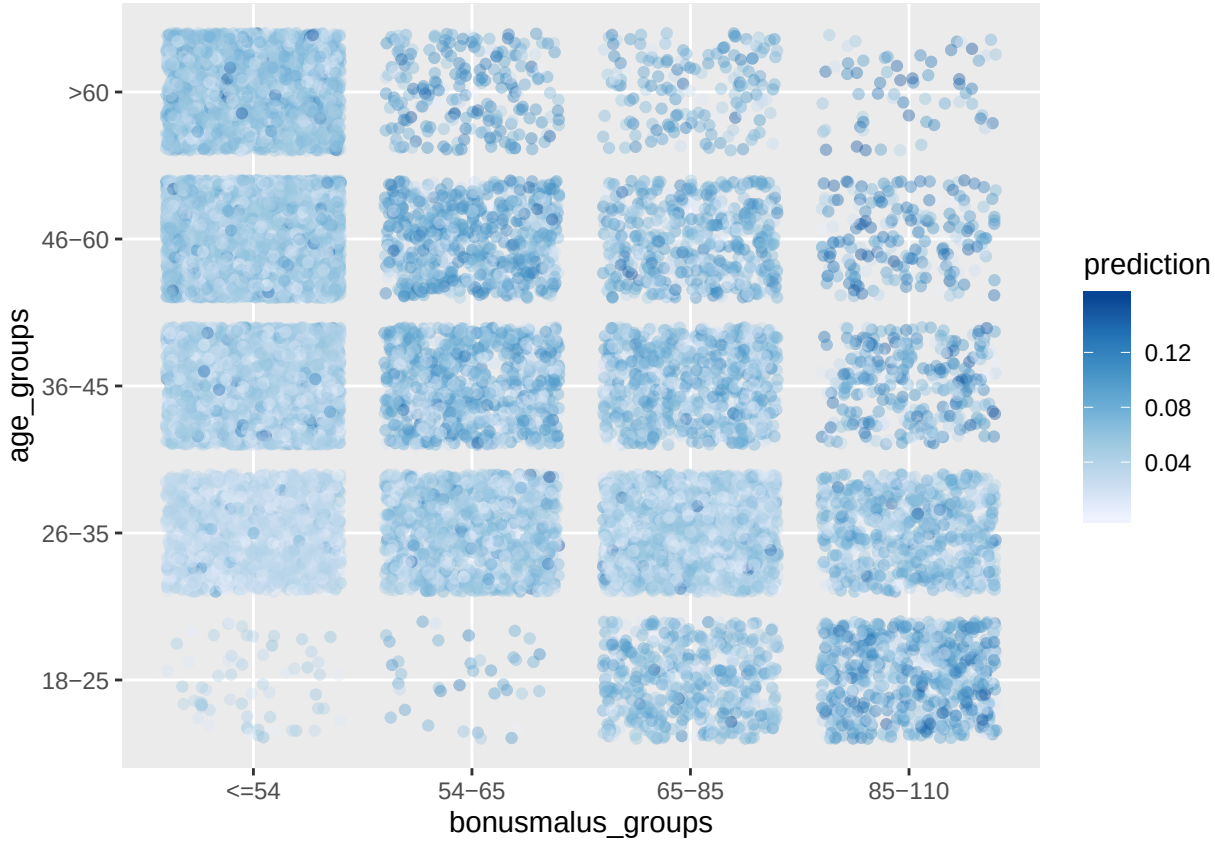
We can see that among the most relevant variables for predictive power are **BonusMalus**, **VehPower**, **VehAge**

and **DrivAge** (all grouped into levels,a s already explained). This is also coherent with what we will find later using **Boosting**, which allows to measure the importance of the variables.

4.2 Prediction

The command `predict(model, type="response")` allows to obtain the prediction for the number of claims on the original scale. We chose two relevant variables for the model to have a 2D representation of `fitSTEP` where the color of the observations correspond to the predicted claim frequency.

```
f<-predict(fitSTEP,testGLM, type = "response" )
testGLM$bonusmalus_groups<- factor(testGLM$bonusmalus_groups,
                          levels = c("<=54", "54-65", "65-85" ,"85-110",">110"))
testGLM$age_groups<-factor(testGLM$age_groups, levels=c("18-25", "26-35", "36-45", "46-60", ">60"))

testGLM2<-testGLM%>%mutate(prediction=f)%>% filter(prediction<0.16)
testGLM2%>% filter(bonusmalus_groups !="NA") %>%
  ggplot(aes(x = bonusmalus_groups, y = age_groups , colour = prediction)) +
  geom_jitter(alpha=.4) +
  scale_colour_distiller(palette=1, direction = 1)
```



We will use, in this example and in the following ones, the **Poisson Deviance as a measure of goodness of fit**. The Poisson Deviance is defined as follows:

$$L(\hat{y}; y) = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i log(\hat{y}_i))$$

where $\hat{y}_i$ is the predicted expected value. Minimizing the Poisson loss is equivalent to maximizing the likelihood of the data under the assumption that the target comes from a Poisson distribution, conditioned on the input

```
yi<- trainGLM$ClaimNb
errorPoissDevGLM<- mean(f-yi*log(f))
errorGLM_MSE<- mean((f-yi)^2)
```

The Poisson Deviance for this model is 0.2268263. We will use this measure to make comparisons with the other methods that will be used later on. In addition to this, we also computed the usual **Mean square error** using the predictions on a `type=response` scale. This measure is 0.0595996 for this model.

4.3 Overdispersed Poisson: the quasipoisson model

A limitation of the Poisson distribution is that the mean is equal to the variance. It may be that the variance is greater than the mean: in this case we say that there is overdispersion of the data with respect to the Poisson model. A **quasi-poisson** model can be used to solve this problem. It allows to estimate the dispersion parameter ϕ, which is introduced to allow the model to have a greater (or possibly lower) variance than the average. In the case of a simple Poisson model it is equal to 1. The coefficients of the model do not change (as it can see by comparing the following model with the `fitAIC` model we estimated before), however the p-values do change, since the variance of the data is now greater.

```
fitsovra<-glm(ClaimNb ~ Exposure + VehGas + Region + age_groups +
    power_groups + age_veh_groups + bonusmalus_groups, family = quasipoisson,
    data = trainGLM, offset = log(Exposure))
summary(fitsovra)
```

```
Call:
glm(formula = ClaimNb ~ Exposure + VehGas + Region + age_groups +
    power_groups + age_veh_groups + bonusmalus_groups, family = quasipoisson,
    data = trainGLM, offset = log(Exposure))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.2798  -0.3590  -0.3121  -0.2247   7.0364

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            -0.610824   0.098446  -6.205 5.51e-10 ***
Exposure               -1.266241   0.065747 -19.259  < 2e-16 ***
VehGasRegular           0.132395   0.040191   3.294 0.000988 ***
RegionR21               0.621002   0.226283   2.744 0.006064 **
RegionR22              -0.018787   0.186789  -0.101 0.919885
RegionR23              -0.442274   0.255290  -1.732 0.083200 .
RegionR24               0.030699   0.072708   0.422 0.672862
RegionR25               0.190587   0.146272   1.303 0.192591
RegionR26              -0.035033   0.166295  -0.211 0.833146
RegionR31              -0.034820   0.118133  -0.295 0.768185
RegionR41              -0.335069   0.170519  -1.965 0.049418 *
RegionR42               0.269708   0.278273   0.969 0.332438
RegionR43              -1.062786   0.705142  -1.507 0.131765
RegionR52               0.147422   0.097587   1.511 0.130876
RegionR53               0.250624   0.090595   2.766 0.005669 **
RegionR54              -0.078995   0.136922  -0.577 0.563988
RegionR72              -0.115147   0.114822  -1.003 0.315947
RegionR73              -0.307970   0.154661  -1.991 0.046456 *
RegionR74               0.165267   0.226295   0.730 0.465200
RegionR82               0.118743   0.077609   1.530 0.126016
RegionR83              -0.614723   0.300845  -2.043 0.041024 *
RegionR91              -0.056196   0.106748  -0.526 0.598587
```

```
RegionR93                 0.042191   0.080404   0.525 0.599771
RegionR94                 0.269943   0.203185   1.329 0.183999
age_groups18-25          -0.208416   0.094292  -2.210 0.027086 *
age_groups26-35          -0.514658   0.070550  -7.295 3.02e-13 ***
age_groups36-45          -0.149225   0.061523  -2.426 0.015289 *
age_groups46-60          -0.056062   0.057192  -0.980 0.326968
power_groups>8           -0.035815   0.061281  -0.584 0.558932
power_groups5-6           0.095461   0.052362   1.823 0.068291 .
power_groups6-7           0.005797   0.052236   0.111 0.911640
power_groups7-8          -0.253783   0.089576  -2.833 0.004610 **
age_veh_groups>8         -1.121276   0.062099 -18.056  < 2e-16 ***
age_veh_groups2-3        -1.015767   0.068209 -14.892  < 2e-16 ***
age_veh_groups4-6        -0.982398   0.068007 -14.445  < 2e-16 ***
age_veh_groups7-8        -0.928988   0.079008 -11.758  < 2e-16 ***
bonusmalus_groups>135     2.132536   0.290200   7.349 2.02e-13 ***
bonusmalus_groups110-135  1.542451   0.131648  11.716  < 2e-16 ***
bonusmalus_groups54-65    0.557907   0.057840   9.646  < 2e-16 ***
bonusmalus_groups65-85    0.496066   0.060492   8.201 2.43e-16 ***
bonusmalus_groups85-110   0.837989   0.072129  11.618  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 1.481309)

    Null deviance: 25321  on 74999  degrees of freedom
Residual deviance: 23396  on 74959  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 6
```

The *dispersion* parameter is $\phi = 1.48$.
In this second case we should proceed by eliminating the variables that are no longer significant, and with the final model obtained we should get the predicted values and compute the two error measures that we have calculated previously. However, we preferred to focus on the next two methods and compare these two with the classic poisson GLM.

5 Ensemble to model the claim frequency: Gradient Boosting

Gradient Boosting methods are proved to be really efficient learning methods which are based on the "strength of weak learners" principles. In fact, *Boosting* combines many **weak learners** (simple methods which can not overfit due to their simplicity) to have a higher level of precision by using them together.
In the following steps we will use both the `base` R package and the the `mlr` package (an alternative to `caret`) to fit a boosting model to predict the number of claims.
The `base` R package allows both to use an offset term to take the different `Exposure` into account and to specify a distribution. We specified `distribution="poisson"` since we are modelling count data, which usually are assumed to be generated by a Poisson distribution.
The aim of the `gbm` algorithm is to find a function

$$\hat{f}(\vec{x}) = \min_{f(\vec{x})} L(y; f(x))$$

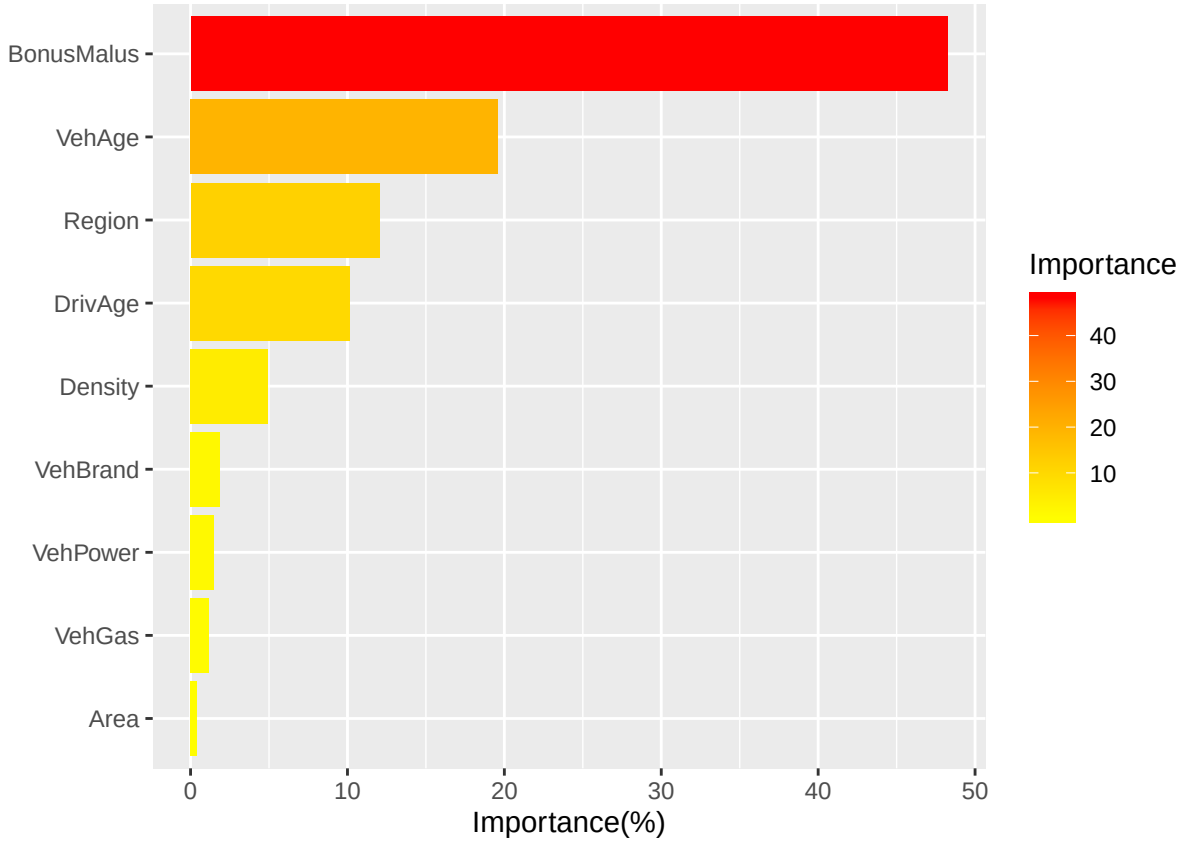
where $\vec{x}$ is the vector of attributes, $\hat{f}(\vec{x})$ is the estimated claim number for a policyholder with attribute vector $\vec{x}$ and L is a specific *loss function*. By specifying `distribution="poisson"` the loss function used will be automatically the poisson loss function, the same we used to evaluate the Poisson GLM.
Initially we set manually the value of the parameters, later on we will do a hiperparameter tuning to choose the best values.
Differently from what has been previously done, we are now using the variables as they originally appear (that is, we are not turning all of the variables into factors with manually-set levels).

5.1 Fitting and visualizing the model

```
library("mboost")
library(gbm)
library(mlr)

dati2<- dati%>% dplyr::select(-c(IDpol, ExposureF, ClaimExp))%>%
  mutate(VehAge=as.numeric(VehAge), VehGas=as.factor(VehGas))

fit1 = formula(ClaimNb ~ Area+VehPower+VehAge+DrivAge+BonusMalus+
                 VehBrand+VehGas+Density+Region+offset(Exposure))

set.seed(3)
num=100000/4*3
num=round(num, digits=0)
indici<-sample(c(1:100000), num, replace=FALSE )
train<- dati2[indici, ]

fitgbm1 = gbm(fit1, distribution = "poisson",
              data = train,
              n.trees = 500,
              shrinkage = 0.1)

a<-summary(fitgbm1)$var
b<-summary(fitgbm1)$rel.inf

ntreeopt<- gbm.perf(fitgbm1, method="OOB")

valtree<- attributes(ntreeopt)$smoother$fitted
optval<- ntreeopt[1]
```

A measure of the importance of each variable is available when the `gbm`algorithm is run. Broadly speaking, the more the use of a variable as split variable reduces a measure of error, the more important is that variable. The following bar chart allows to see the more important attributes for predicting power.

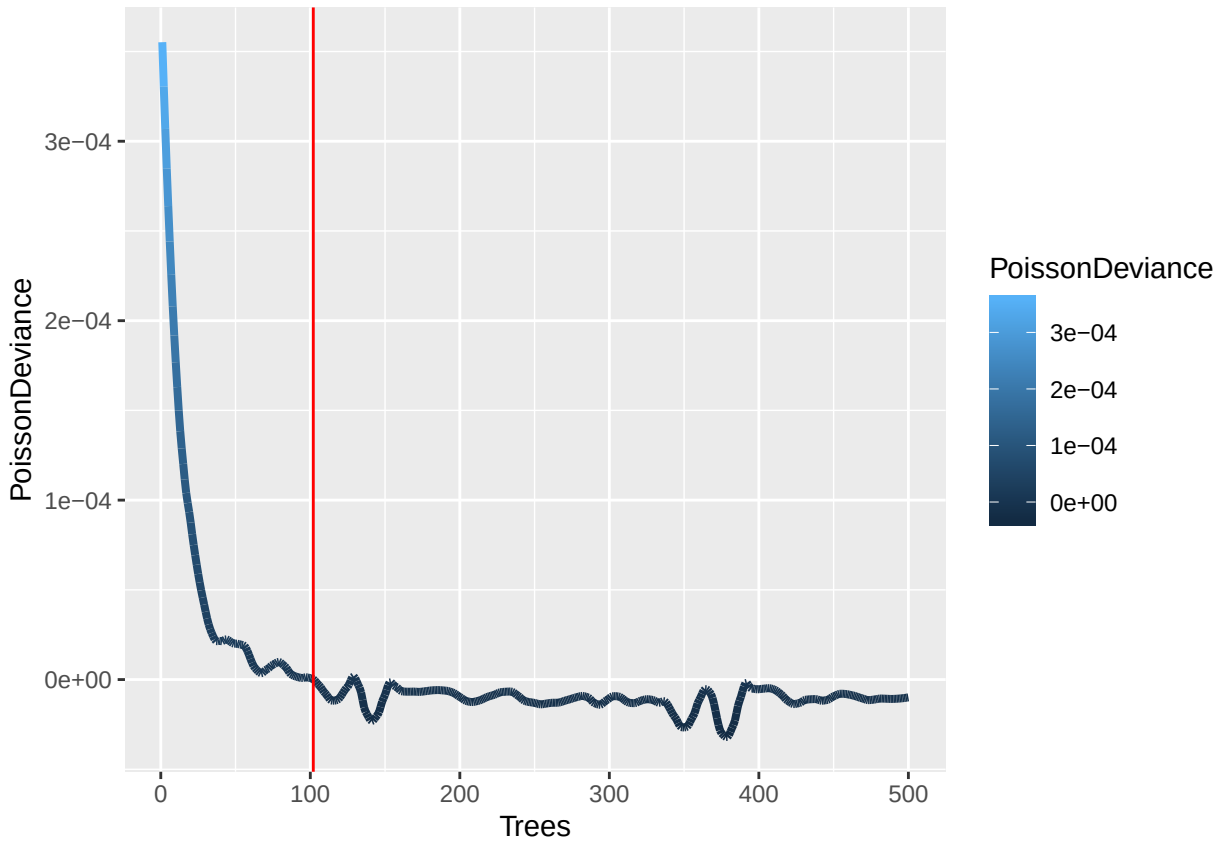
```r
a<-as.character(a)
sum<- sum(b)
dd<- data.frame(Variables=a, Importance = b)
dd$Variables<-factor(dd$Variables, levels=dd$Variables[order(dd$Importance)])

dd%>%ggplot(aes(x=Variables, y=Importance))+ geom_bar(stat="identity",
  aes(fill=Importance))+
  coord_flip()+
  ylab("Importance(%)")+ xlab("")+ scale_fill_gradient2(mid="yellow", high="red")
```



The `gbm` package comes with a default function called `gbm.perf()` to determine the optimum number of iterations. The argument "method" allows the user to specify the technique used ("OOB" or "cv") for deciding the optimum number of trees. In this case the Out-Of-Bag error[5] is used to evaluate how many trees to use in the ensamble. The criterion on which this choice is based is the Poisson Deviance.

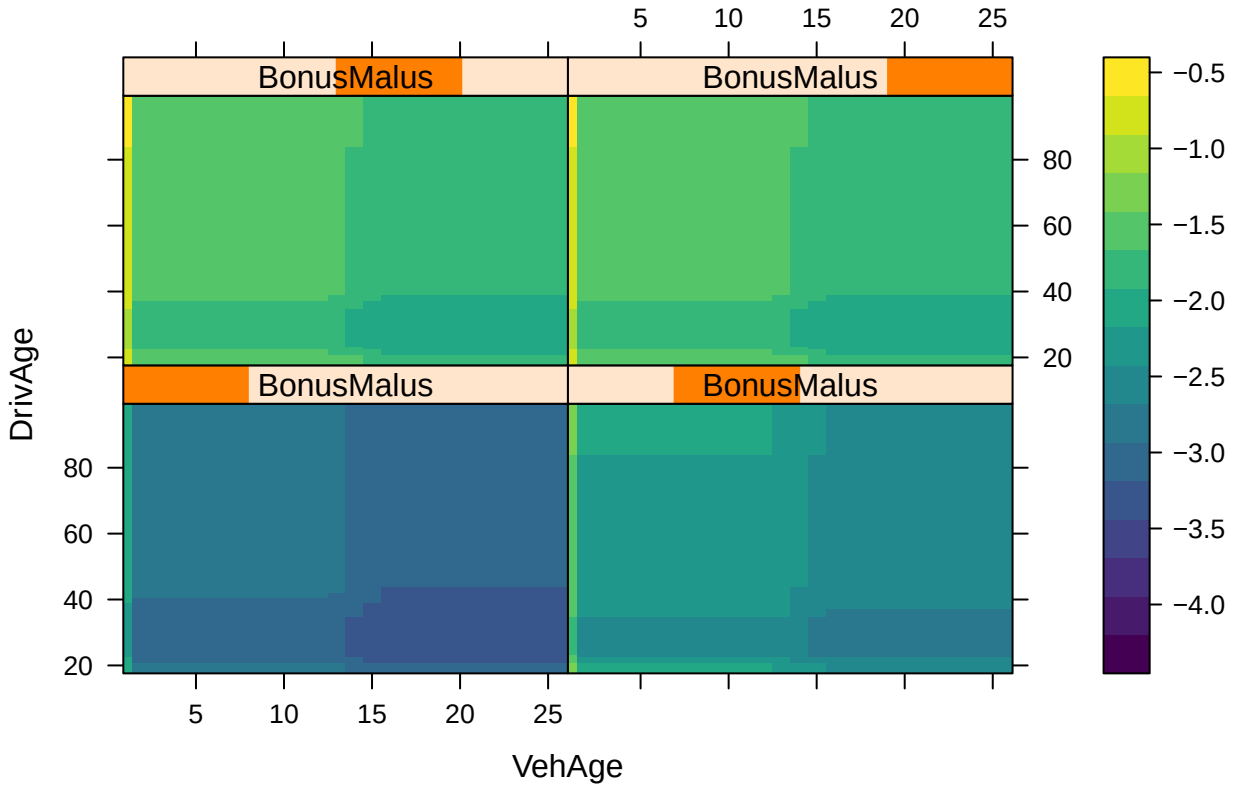
```r
optimV<- data.frame(Trees=c(1:500), PoissonDeviance=valtree)

ggplot(optimV, aes(x=Trees, y=PoissonDeviance))+
  geom_line(aes(col=PoissonDeviance), size=1.4)+
  geom_vline(xintercept = optval, col="red")
```

[5]For the way in which one variant of the Boosting algorithm is built, the OOB error is an error computed using only a subset of observations which were not used to train the model. It represents a kind of freely-available test set



We can visualize the prediction of the model in relation to the three most important variables: `BonusMalus`, `VehAge` and `DrivAge`. The orange bar on the label `BonusMalus` stands for the level of the `Bonusmalus` variable: "low", "medium-low", "medium-high" and "high".

```
plot(fitgbm1, i.var = 3:5 , n.trees = optval)
```



We can visualize above the ensamble created using the first `optval` trees and its predictions. Predictions (which correspond to the different shades of colors) are expressed on a logaritmic scale, in fact the legend ranges from -3.1 (`exp(-3.1)` = 0.0450492) to 1.5 (`exp(1.5)`=4.4816891). It can be clearly seen that lower values of the `BonusMalus` class correspond to lower predicted claim frequency, as one would expect. As the class grows, the expected claim frequency increases.

The prediction error is now obtained: since we previously trained the model using only 3/4 of the observations (random sample), it is now possible to use the test set to have a measure of the goodness of fit. Both the Poisson Deviance and the MSE will be used.

```
test<- dati2[-indici,]
yi<- (test$ClaimNb)
predictions<- predict(fitgbm1, newdata=test, n.trees=optval, type="response")

errorPoissDev<- mean(predictions-yi*log(predictions))
errorMSE<- mean((predictions-yi)^2)
```

The **Poisson Deviance** is 0.2137746.

As previously mentioned, the `mlr` library will now be used. It implements many common machine learning tasks and makes them very simple.
Two data partitions are defined: the first divides the data into train and test (outer partition) and is used for the overall evaluation of the model while the internal one is for tuning the hyperparameters (which in this case are the number of trees and the shrinkage[6]).
The output we get at the end, when we call the `resample(...)` command, is an **estimate of the error of the**

[6]Broadly speaking, it is used to find a good balance between the ability of the model to predict and overfitting: if it is too low the model could end up being too simple and hence lack flexibility. On the other hand if it is too high the model could overly adapt to the data. It is basically the weight which is given by all the subsequent trees which are added, where each new tree correct the error of the precious one, since the updated response variable is the error of the previous model

final model[7] (the final model is the one that is built with all the data, and not the one that is built only with the training set, because the training-test procedure is done only to have an estimate of the ability of the model to generalize). The optimal values of the parameters (`n.tree` and `shrinkage`) are not returned as the optimal values on the training set are not necessarily the same that are obtained when using the entire dataset. In this example we try to tune parameters using 3 discrete values for the number of trees, and a function for the shrinkage values. Using the instruction `makeTuneControlMBO(budget = 2)` allows to do a random search within the parameter space (which is not a Grid Search, thus it is not exhaustive). We have to use the mean square error as a performance measure, since `mlr` does not implement the Poisson Deviance. By adding the offset term, we are modelling directly the expected number of claims (the exposure is added to adjust for the time each policy was under observation).

```
learner_gbm <- makeLearner("regr.gbm")
dati3<-dati2
dati3<- dati3%>% mutate(Exposure=log(Exposure))
task<- makeRegrTask(data=dati3, target="ClaimNb", offset("Exposure"))
set.seed(3)

desc <- makeResampleDesc("Holdout", split=3/4)
particion <- makeResampleInstance(desc, task)


set.seed(3)
desc_outer<- makeResampleDesc("Holdout", split=3/4)
particion_outer <- makeResampleInstance(desc_outer, task)
ps = makeParamSet(
makeDiscreteParam("n.trees", values =c(100,150,200)),
makeNumericParam("shrinkage", lower = -6, upper = -2,
trafo = function(x) 2^x)
)
control_grid <- makeTuneControlMBO(budget = 2)
set.seed(3)
learner_ajuste_gbm <- makeTuneWrapper(learner_gbm,
resampling = desc,
par.set = ps,
control = control_grid,
measures = list(mse))
model_gbm_n <- resample(learner_ajuste_gbm, task, particion, measure= mse)
modelAggr<-model_gbm_n$aggr
```

The second model (the once which does hiperparameters tuning) has an estimated error of 0.0571653, which is lower than the model which does not tune the parameters (0.0581542).
In some cases we can decide to convert the expected claim frequencies into discrete numbers, by fixing some rules. For example one could decide to set `ExpectedClaimNumber`$= 0$ when `predictions[i]`< 0.3 (or any other reasonable number).

[7] the final model is a model which in turn uses cross validation to tune the parameter, therefore the tuning of hiperparameters is seen as part of the model itself

6 Neural Networks

Finally we train a simple neural network using the `nnet` command through the interface offered by `mlr`. `Nnet()` allows to fit single-hidden-layer neural network, possibly with skip-layer connections. We are tuning, for this model, the number of units in the hidden layer, which is the parameter `size`. Also in this case the tuning of the hyperparameter is done through a split of the training set in a validation set and a second training set, and the external evaluation of the model is again done using "Holdout" instead of "Cross Validation", due to the fact that we have quite a big number of units. We tune the model, letting `size` range from 2 to 8, and we compute the mean square error[8]

```
dati2$cl_exp<- (dati2$ClaimNb/dati2$Exposure)
dati2<-dati2[,c(3:12)]
task <- makeRegrTask(data= dati2, target="cl_exp")
learner_nnet <- makeLearner("regr.nnet")

set.seed(3)
particion <- makeResampleInstance(desc, task)

par.set = makeParamSet(makeIntegerLearnerParam(id = "size", lower = 3, upper = 8))
desc_inner <- makeResampleDesc("Holdout", split=3/4)
particion_outer <- makeResampleInstance(desc, task)
control_grid <- makeTuneControlGrid()
set.seed(3)
learner_ajuste_nnet <- makeTuneWrapper(learner_nnet,
                                       resampling = desc_inner,
                                       par.set = par.set,
                                       control = control_grid,
                                       measures = list(mse))
errores_ajuste_nnet <- resample(learner_ajuste_nnet,
                                task,
                                particion_outer,
                                measures = list(mse),
                                extract = getTuneResult)
estimatedMSE<- errores_ajuste_nnet$aggr
```

The estimated Mean square error of the model is 0.0562452450706946. We can finally compare all the errors:

```
             Model        PoissonDev        MSE
1       PoissonGLM 0.226826346742062 0.05959963
2         Boosting 0.213774577470341 0.05815424
3  Boosting_Tuning                   0.05716529
4 NNetwwork_Tuning                   0.05624525
```

[8] since the tuning has been done based on this measure, as we also commented before for the boosting with tuning

7 Resources

http://cas.uqam.ca/pub/R/web/CASdatasets-manual.pdf
https://www.casact.org/education/rpm/2012/handouts/Session_4736_presentation_895_0.pdf
https://mlr.mlr-org.com/articles/tutorial/tune.html
https://ethz.ch/content/dam/ethz/special-interest/math/risklab-dam/documents/walter-saxer-preis/ma-zoechbauer.pdf