Descriptive Statistics

French Motor Third-Party Liability Claims

Daniel Meier and Jürg Schelldorfer, with support from Christian Lorentzen, Friedrich Loser, Michael Mayer, Mario V. Wüthrich and Mirai Solutions GmbH.

2021-15-10

Introduction

This notebook was created for the course "Deep Learning with Actuarial Applications in R" of the Swiss Association of Actuaries (https://www.actuaries.ch/).

This notebook serves as companion to the tutorial "Case Study: French Motor Third-Party Liability Claims", available on SSRN.

The code is similar to the code used in above tutorial and combines the raw R code in the scripts, available on GitHub along with some more comments. Please refer to the tutorial for explanations.

Note that the results might vary depending on the R and Python package versions, see last section for the result of sessionInfo() and corresponding info on the Python setup.

Data Preparation

The tutorial uses the French MTPL data set available on openML (ID 41214).

Load packages and data

```
library(rgdal)
```

```
## Loading required package: sp
```

```
## Please note that rgdal will be retired by the end of 2023,
## plan transition to sf/stars/terra functions using GDAL and PROJ
## at your earliest convenience.
##
## rgdal: version: 1.5-27, (SVN revision 1148)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.0.4, released 2020/01/28
## Path to GDAL shared files: /usr/share/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 6.3.1, February 10th, 2020, [PJ_VERSION: 631]
## Path to PROJ shared files: /usr/share/proj
## Linking to sp version:1.4-5
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading sp or rgdal.
```

```
# library(rgeos)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(corrplot)
```

```
## corrplot 0.90 loaded
```

```
# plotting parameters in R Markdown notebook
knitr::opts_chunk$set(fig.width = 9, fig.height = 9)
# plotting parameters in Jupyter notebook
library(repr)  # only needed for Jupyter notebook
options(repr.plot.width = 9, repr.plot.height = 9)
```

Set global parameters

```
options(encoding = 'UTF-8')
```

Helper functions

Subsequently, for ease of reading, we provide all the helper functions which are used in this tutorial in this section.

```
summarize <- function(...) suppressMessages(dplyr::summarize(...))
```

```
load_data <- function(file) {
  load(file.path("../0_data/", file), envir = parent.frame(1))
}
```

```
runMultiPlot <- function(dat, VarName) {
  dat <- rename(dat, "VarName" = all_of(VarName))
  out_sum <- dat %>%
    group_by(VarName) %>%
    summarize(NrObs = length(Exposure),
              Exp = sum(Exposure),
              Nr.Claims = sum(ClaimNb),
              Freq = sum(ClaimNb) / sum(Exposure),
              StDev = sqrt(sum(ClaimNb)) / sum(Exposure))
  # Plot 1
  p1 <- ggplot(out_sum, aes(x = VarName, y = Exp, fill = VarName)) +
    geom_bar(stat = "identity") +
```

```
    geom_text(stat = 'identity', aes(label = round(Exp, 0), color = VarName), vjust = -0.5, size = 2.5)
    labs(x = VarName, y = "Exposure in years", title = "exposure") + theme(legend.position = "none")

  # Plot 2
  p2 <- ggplot(out_sum, aes(x = VarName, group = 1)) + geom_point(aes(y = Freq, colour = "observed")) +
    geom_line(aes(y = Freq, colour = "observed"), linetype = "dashed") +
    geom_line(aes(x = as.numeric(VarName), y = pf_freq), color = "red") +
    geom_line(aes(x = as.numeric(VarName), y = Freq + 2 * StDev), color = "red", linetype = "dotted") +
    geom_line(aes(x = as.numeric(VarName), y = Freq - 2 * StDev), color = "red", linetype = "dotted") +
    ylim(0, 0.35) +
    labs(x = paste(VarName, "groups"), y = "frequency", title = "observed frequency") + theme(legend.pos

  # Plot 3
  p3 <- ggplot(out_sum) + geom_bar(stat = "identity", aes(x = VarName, y = Freq, fill = VarName)) +
    geom_line(aes(x = as.numeric(VarName), y = pf_freq), color = "red") + guides(fill = FALSE) +
    labs(x = paste(VarName, "groups"),  y = "frequency", title = "observed frequency") + theme(legend.p

  grid.arrange(p1, p2, p3, ncol = 2)
}

plot_2dim_contour <- function(data, VarX, VarY, LabelX, LabelY) {
  data <- rename(data, "VarX" = all_of(VarX), "VarY" = all_of(VarY))
  df_plt <- data %>%
    group_by(VarX, VarY) %>%
    summarize(Exp = sum(Exposure),
              Freq = sum(ClaimNb) / sum(Exposure),
              Pol = n())
  p <- ggplot(df_plt, aes(
    x = as.numeric(VarX),
    y = as.numeric(VarY),
    z = Exp
  )) + geom_contour_filled() + labs(x = LabelX, y = LabelY)
}

plotMap <- function(area_points, Var, label, clow, chigh) {
  area_points <- rename(area_points, "Var" = all_of(Var))
  ggplot(area_points, aes(long, lat, group=group)) +
    ggtitle(paste(label, "by region", sep = " ")) +
    geom_polygon(aes(fill = Var)) +
    scale_fill_gradient(low = clow, high = chigh, name = label) +
    xlab("Longitude") + ylab("Latitude")
}
```

Load data

We consider the data `freMTPL2freq` included in the R package `CASdatasets` for claim frequency modeling. This data comprises a French motor third-party liability (MTPL) insurance portfolio with corresponding claim counts observed in one accounting year. We do not incorporate claim sizes which would also be available through freMTPL2sev.

As the current package version provides a slightly amended dataset, we use an older dataset available on openML (ID 41214). Before we can use this data set we need to do some data cleaning. It has been pointed out by F. Loser that some claim counts do not seem to be correct. Hence, we use the pre-processing of the data described in the book "Statistical Foundations of Actuarial Learning and its Applications" in Appendix

A.1. This pre-processed data can be downloaded from the course GitHub page here.

```
load_data("freMTPL2freq.RData")
```

Inspect the raw dataset

```
str(freMTPL2freq)
```

```
## 'data.frame':    678007 obs. of  13 variables:
##  $ IDpol     : num  1 3 5 10 11 13 15 17 18 21 ...
##  $ Exposure  : num  0.1 0.77 0.75 0.09 0.84 0.52 0.45 0.27 0.71 0.15 ...
##  $ Area      : Factor w/ 6 levels "A","B","C","D",..: 4 4 2 2 2 5 5 3 3 2 ...
##  $ VehPower  : num  5 5 6 7 7 6 6 7 7 7 ...
##  $ VehAge    : num  0 0 2 0 0 2 2 0 0 0 ...
##  $ DrivAge   : num  55 55 52 46 46 38 38 33 33 41 ...
##  $ BonusMalus: num  50 50 50 50 50 50 50 68 68 50 ...
##  $ VehBrand  : Factor w/ 11 levels "B1","B2","B3",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ VehGas    : Factor w/ 2 levels "Diesel","Regular": 2 2 1 1 1 2 2 1 1 1 ...
##  $ Density   : num  1217 1217 54 76 76 ...
##  $ Region    : Factor w/ 22 levels "R11","R21","R22",..: 18 18 3 15 15 8 8 20 20 12 ...
##  $ ClaimTotal: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ ClaimNb   : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
knitr::kable(head(freMTPL2freq))
```

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

We briefly describe this data. See the `CASdatasets` reference manual here for a description of the variables of the `freMTPLfre` dataset.

We have 6'780'013 individual car insurance policies and for each policy we have 12 variables:

- `IDpol`: policy number (unique identifier)
- `ClaimNb`: number of claims on the given policy
- `Exposure`: total exposure in yearly units
- `Area`: area code (categorical, ordinal)
- `VehPower`: power of the car (categorical, ordinal)
- `VehAge`: age of the car in years
- `DrivAge`: age of the (most common) driver in years
- `BonusMalus`: bonus-malus level between 50 and 230 (with reference level 100)
- `VehBrand`: car brand (categorical, nominal)
- `VehGas`: diesel or regular fuel car (binary)
- `Density`: density of inhabitants per km2 in the city of the living place of the driver
- `Region`: regions in France (prior to 2016), as illustrated below

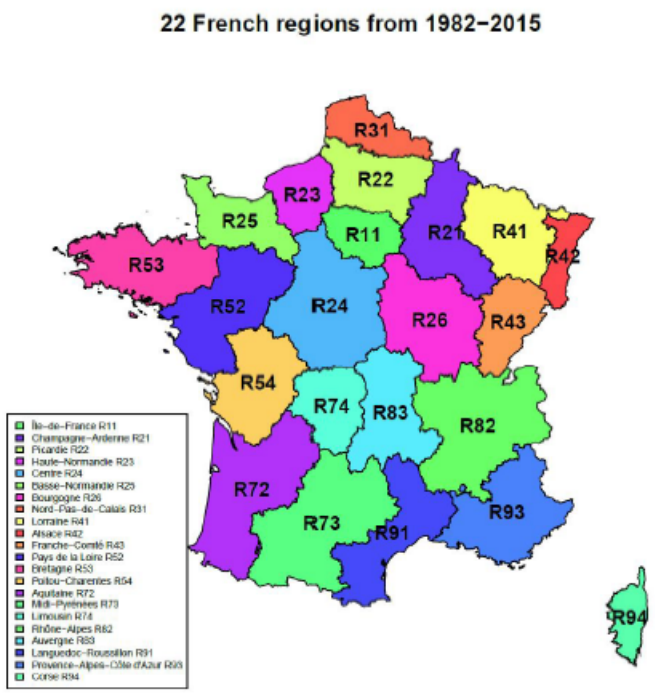


Figure A.1: The 22 regions in France between 1982 and 2015.

Please find some more charts on the raw data in the tutorial.

Data preprocessing

Data preprocessing includes a couple of transformations. We ensure that ClaimNb is an integer, VehAge, DrivAge and BonusMalus have been capped for the plots at age 20, age 90 and bonus-malus level 150, respectively, to improve visualization. Density is logarithmized and VehGas is a categorical variable.

```
dat <- freMTPL2freq %>%
  mutate(ClaimNb = as.integer(ClaimNb),
         VehAge = pmin(VehAge, 20),
         DrivAge = pmin(DrivAge, 90),
         BonusMalus = round(pmin(BonusMalus, 150) / 10, 0) * 10,
         Density = round(log(Density), 0),
         VehGas = factor(VehGas))
```

Descriptive Analysis

In order to get used to the dataset, we start with a descriptive analysis.

Inspect the prepared dataset

```
knitr::kable(head(dat))
```

IDpol	Exposure	Area	VehPower	VehAge	DrivAge	BonusMalus	VehBrand	VehGas	Density	Region	ClaimTotal	ClaimNb
1	0.10	D	5	0	55	50	B12	Regular	7	R82	0	0

IDpol	Exposure	Area	VehPower	VehAge	DrivAge	BonusMalus	VehBrand	VehGas	Density	Region	ClaimTotal	ClaimNb
3	0.77	D	5	0	55	50	B12	Regular	7	R82	0	0
5	0.75	B	6	2	52	50	B12	Diesel	4	R22	0	0
10	0.09	B	7	0	46	50	B12	Diesel	4	R72	0	0
11	0.84	B	7	0	46	50	B12	Diesel	4	R72	0	0
13	0.52	E	6	2	38	50	B12	Regular	8	R31	0	0

```
str(dat)
```

```
## 'data.frame':    678007 obs. of  13 variables:
##  $ IDpol     : num  1 3 5 10 11 13 15 17 18 21 ...
##  $ Exposure  : num  0.1 0.77 0.75 0.09 0.84 0.52 0.45 0.27 0.71 0.15 ...
##  $ Area      : Factor w/ 6 levels "A","B","C","D",..: 4 4 2 2 2 5 5 3 3 2 ...
##  $ VehPower  : num  5 5 6 7 7 6 6 7 7 7 ...
##  $ VehAge    : num  0 0 2 0 0 2 2 0 0 0 ...
##  $ DrivAge   : num  55 55 52 46 46 38 38 33 33 41 ...
##  $ BonusMalus: num  50 50 50 50 50 50 50 70 70 50 ...
##  $ VehBrand  : Factor w/ 11 levels "B1","B2","B3",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ VehGas    : Factor w/ 2 levels "Diesel","Regular": 2 2 1 1 1 2 2 1 1 1 ...
##  $ Density   : num  7 7 4 4 4 8 8 5 5 4 ...
##  $ Region    : Factor w/ 22 levels "R11","R21","R22",..: 18 18 3 15 15 8 8 20 20 12 ...
##  $ ClaimTotal: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ ClaimNb   : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
summary(dat)
```

```
##      IDpol            Exposure        Area          VehPower
##  Min.   :      1   Min.   :0.002732   A:103957   Min.   : 4.000
##  1st Qu.:1157948   1st Qu.:0.180000   B: 75459   1st Qu.: 5.000
##  Median :2272153   Median :0.490000   C:191880   Median : 6.000
##  Mean   :2621857   Mean   :0.528547   D:151590   Mean   : 6.455
##  3rd Qu.:4046278   3rd Qu.:0.990000   E:137167   3rd Qu.: 7.000
##  Max.   :6114330   Max.   :1.000000   F: 17954   Max.   :15.000
##
##      VehAge          DrivAge       BonusMalus        VehBrand
##  Min.   : 0.000   Min.   :18.0   Min.   : 50.00   B12    :166024
##  1st Qu.: 2.000   1st Qu.:34.0   1st Qu.: 50.00   B1     :162730
##  Median : 6.000   Median :44.0   Median : 50.00   B2     :159861
##  Mean   : 6.976   Mean   :45.5   Mean   : 59.74   B3     : 53395
##  3rd Qu.:11.000   3rd Qu.:55.0   3rd Qu.: 60.00   B5     : 34753
##  Max.   :20.000   Max.   :90.0   Max.   :150.00   B6     : 28548
##                                                   (Other): 72696
##      VehGas          Density           Region         ClaimTotal
##  Diesel :332136   Min.   : 0.000   R24    :160601   Min.   :      0
##  Regular:345871   1st Qu.: 5.000   R82    : 84752   1st Qu.:      0
##                   Median : 6.000   R93    : 79315   Median :      0
##                   Mean   : 5.956   R11    : 69791   Mean   :     88
##                   3rd Qu.: 7.000   R53    : 42122   3rd Qu.:      0
##                   Max.   :10.000   R52    : 38751   Max.   :4075401
##                                    (Other):202675
##     ClaimNb
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
```

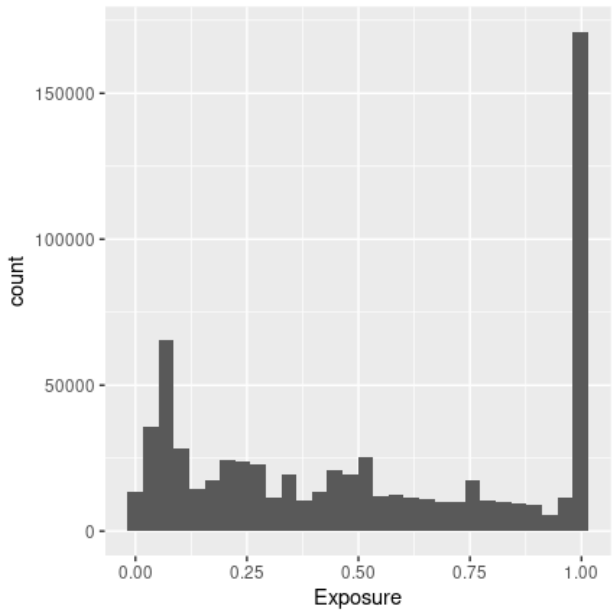
```
##  Mean   :0.03891
##  3rd Qu.:0.00000
##  Max.   :5.00000
##
```

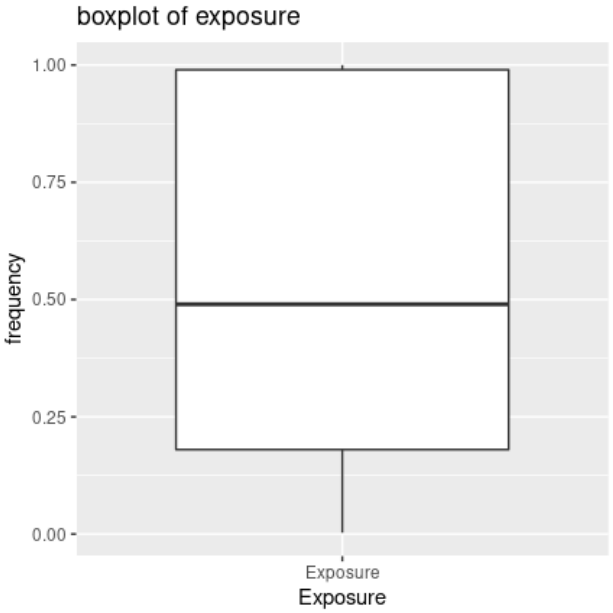
Portfolio Structure

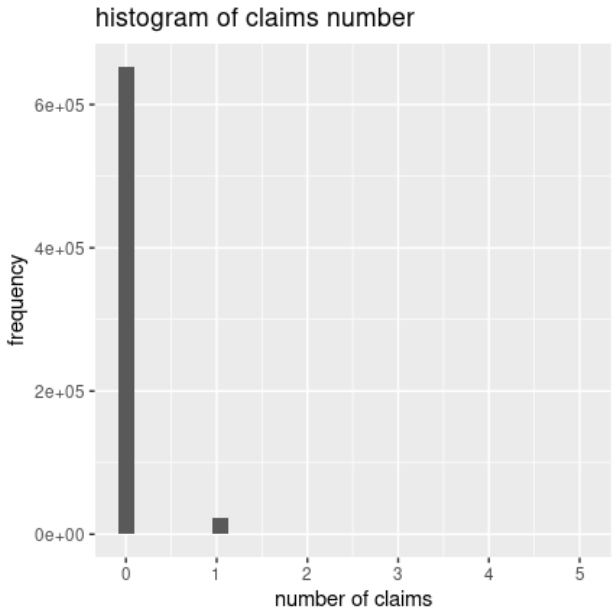
We start by providing descriptive and exploratory statistics of the data. This comprises first the portfolio structure in terms of volumes and key statistics.

```
p1 <- ggplot(dat, aes(Exposure)) + geom_histogram()
p2 <- ggplot(dat, aes(x = "Exposure", y = Exposure)) + geom_boxplot() +
      labs(x = "Exposure", y = "frequency", title = "boxplot of exposure")
p3 <- ggplot(dat, aes(ClaimNb)) + geom_histogram() +
      labs(x = "number of claims", y = "frequency", title = "histogram of claims number")
grid.arrange(p1, p2, p3, ncol = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```







We start by describing the `Exposure`. The `Exposure` measures the duration of an insurance policy in yearly units; sometimes it is also called years-at-risk. The shortest exposure in our data set is 0.0027 which corresponds to 1 day, and the longest exposure is 1 which corresponds to 1 accounting year. The figure shows a histogram and a boxplot of these exposures. In view of the histogram we conclude that roughly 1/4 of all policies have a full exposure of 1 accounting year, and all other policies are only partly exposed during the accounting year. From a practical point of view this high ratio of partly exposed policies seems rather unusual. A further inspection of the data indicates that policy renewals during the year are accounted for two separate records in the data set. Of course, such split policies should be merged to one yearly policy. Unfortunately, we do not have the necessary information to perform this merger, therefore, we need to work with the data as it is.

On 653'069 insurance policies (amounting to a total exposure of 341'090 years-at-risk) we do not have any claim, and on the remaining 24'938 policies (17'269 years-at-risk) we have at least one claim.

```
dat %>%
  group_by(ClaimNb) %>%
  summarize(n = n(), Exposure = round(sum(Exposure), 0))
```

```
## # A tibble: 6 × 3
##   ClaimNb      n Exposure
##     <int>  <int>    <dbl>
## 1       0 653069   341090
## 2       1  23571    16315
## 3       2   1298      909
## 4       3     62       42
## 5       4      5        2
## 6       5      2        1
```

Before fitting any model later, let us see what the overall observed claim frequency in the data is.

```
# calculate portfolio claims frequency
pf_freq <- sum(dat$ClaimNb) / sum(dat$Exposure)

# portfolio claims frequency (homogeneous estimator)
sprintf("Portfolio claim frequency:  %s", round(pf_freq, 4))
```

```
## [1] "Portfolio claim frequency:  0.0736"
```

Observed (marginal) frequencies

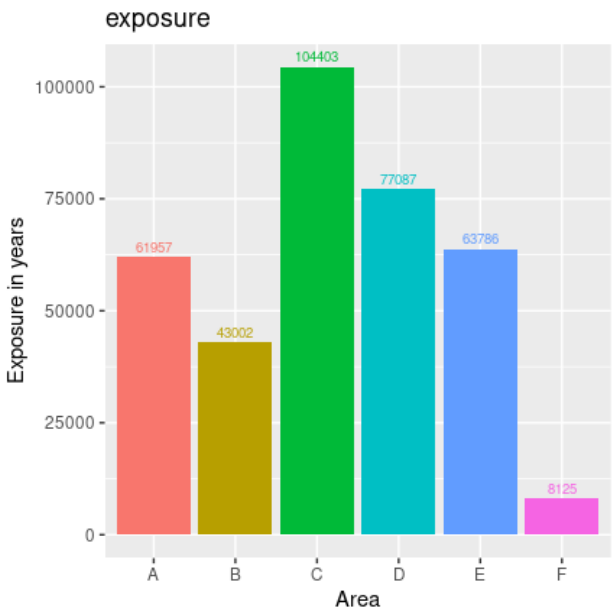
Describing the observed (marginal) frequencies build the foundations for fitting any frequency model to the data. Below, we provide exposures per variable on the top left, the marginal empirical frequency per variable (w.r.t. `Exposure`) (bottom left) and the marginal empirical frequencies including confidence intervals.

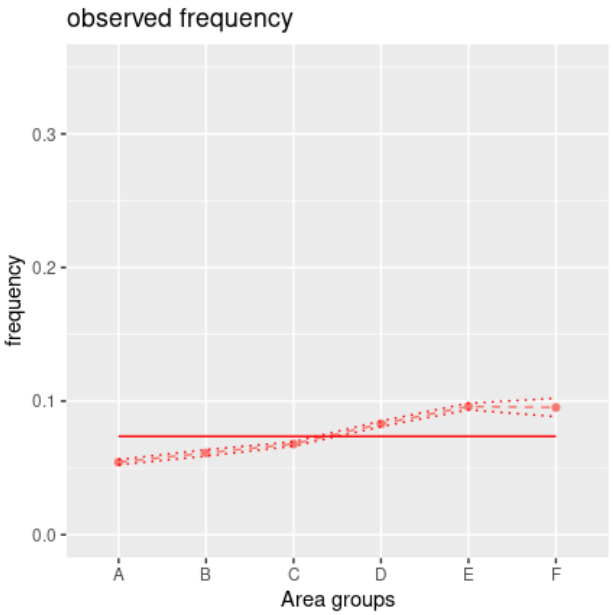
The frequencies are complemented by confidence bounds of two standard deviations (dotted lines). These confidence bounds correspond to twice the estimated standard deviations, see Appendix A.1 in the book for the mathematical definition. We note that in all frequency plots the y-axis ranges from 0% to 35%. From these plots we conclude that some labels have only a small underlying `Exposure`, `BonusMalus` leads to the highest variability in frequencies followed by `DrivAge`; and there is quite some heterogeneity in feature values across the different French regions.

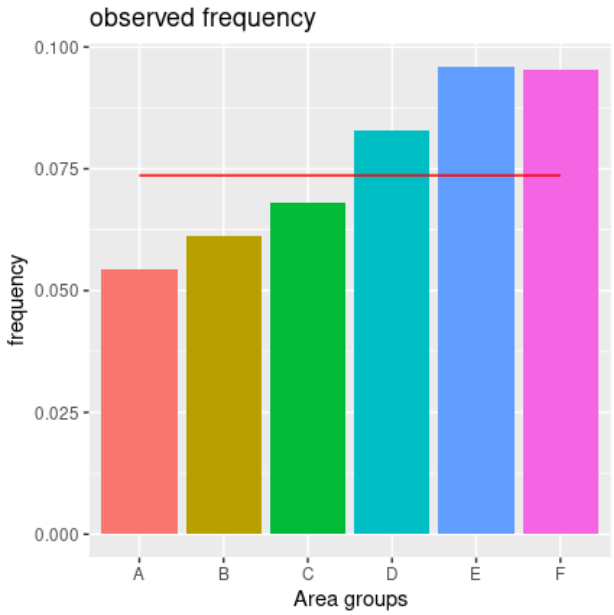
Area

```
runMultiPlot(dat, VarName = "Area")
```

```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.
```



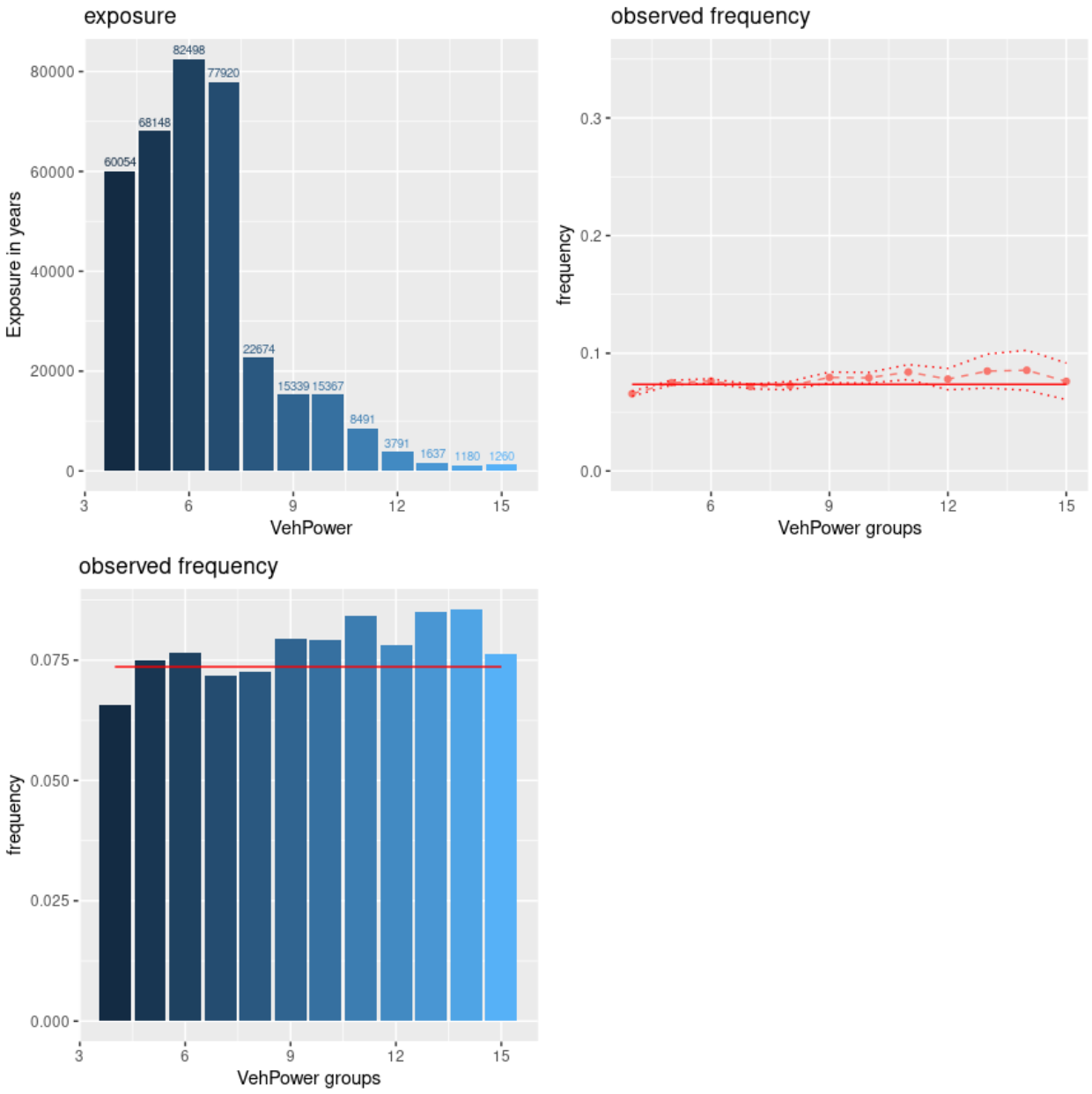




VehPower

```
runMultiPlot(dat, VarName = "VehPower")
```

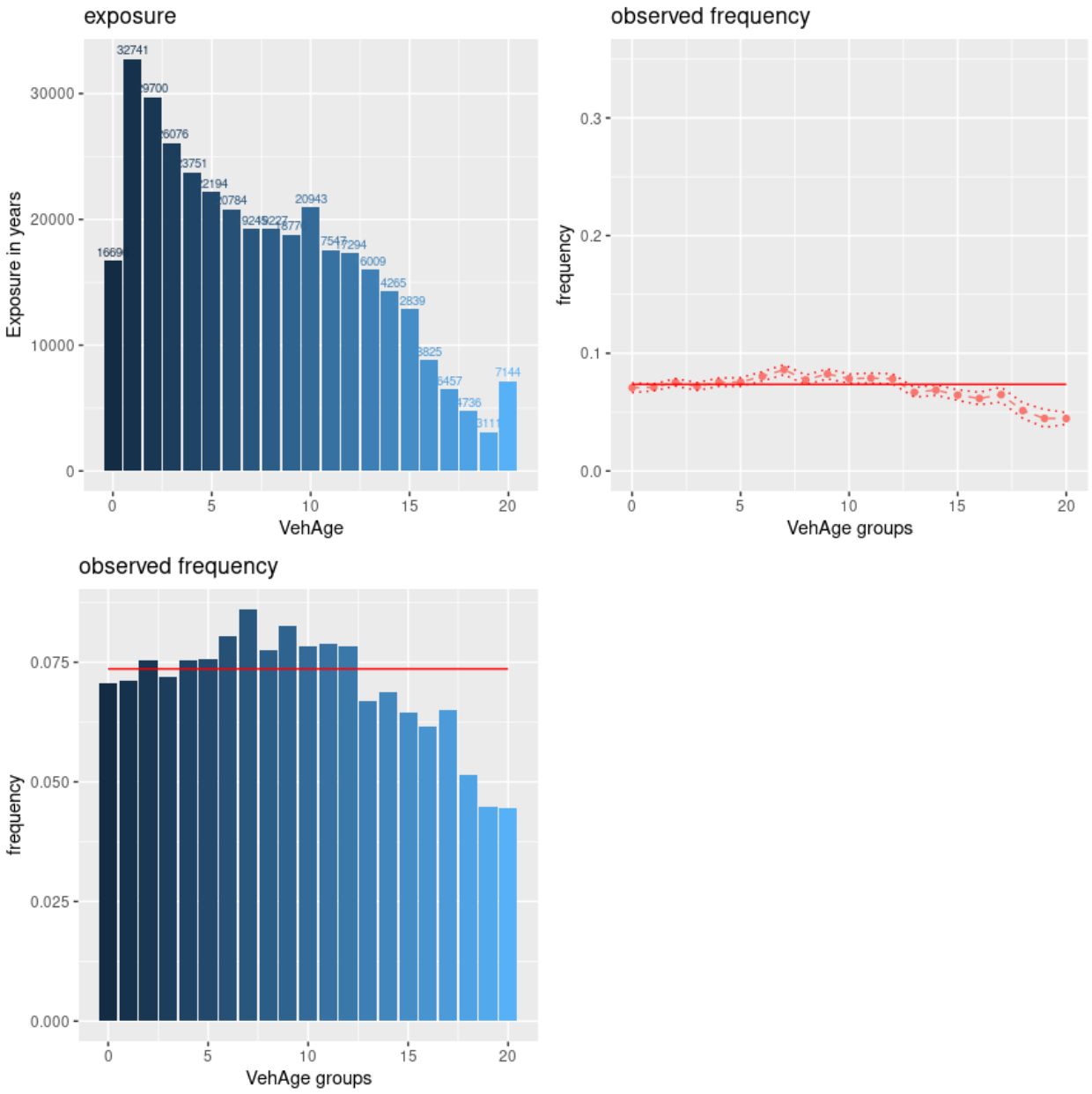
```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



VehAge

```
runMultiPlot(dat, VarName = "VehAge")
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

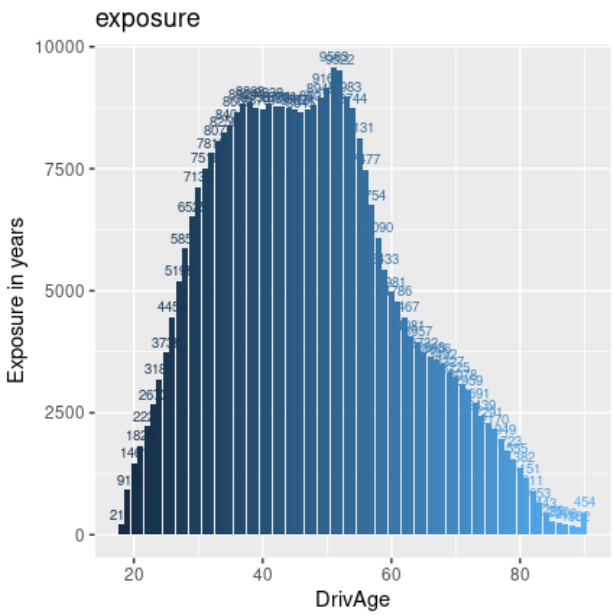


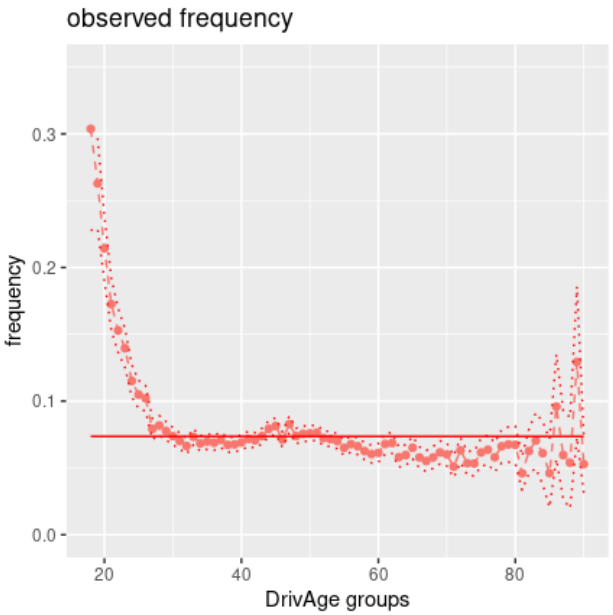
DrivAge

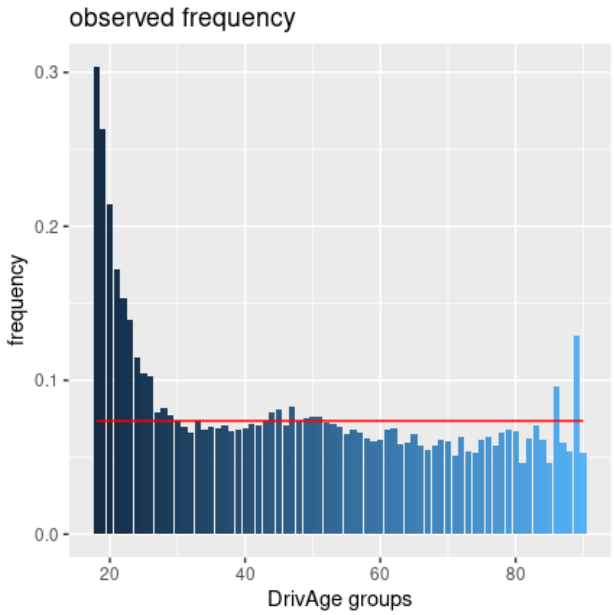
```
runMultiPlot(dat, VarName = "DrivAge")
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```







BonusMalus

```
runMultiPlot(dat, VarName = "BonusMalus")
```

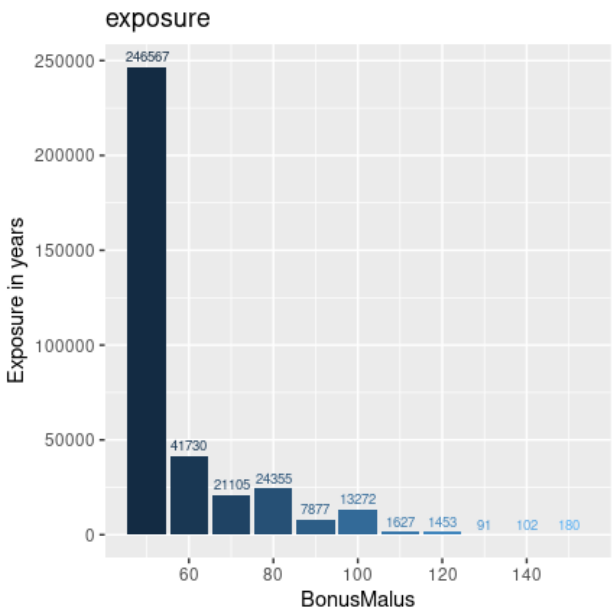
```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.
```

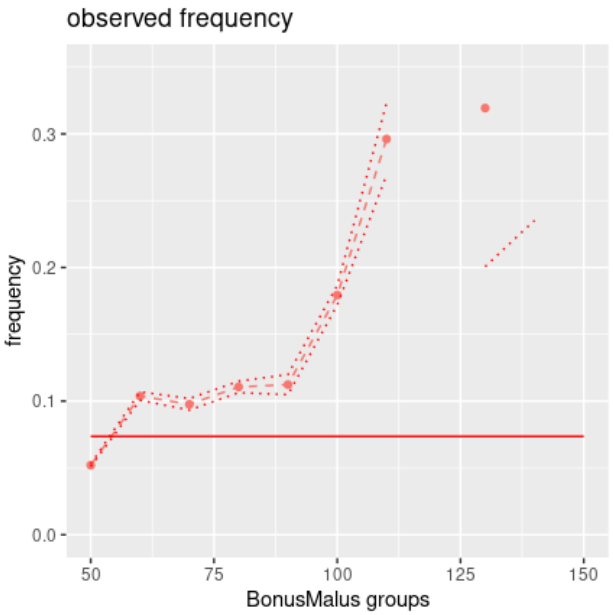
```
## Warning: Removed 3 rows containing missing values (geom_point).
```

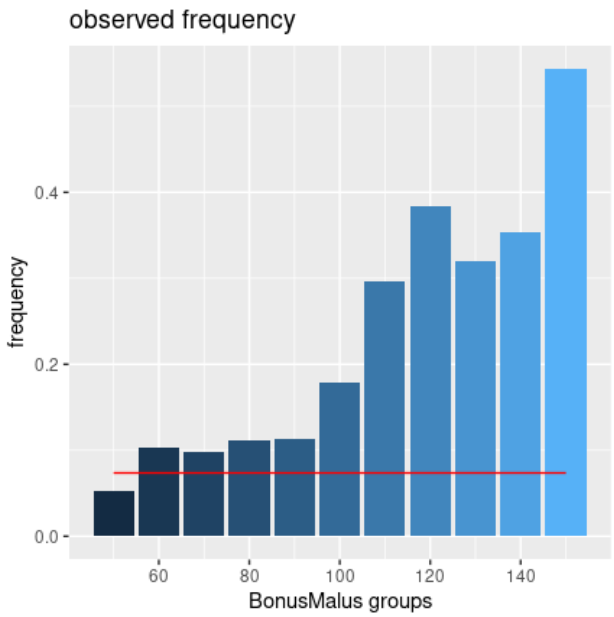
```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 4 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



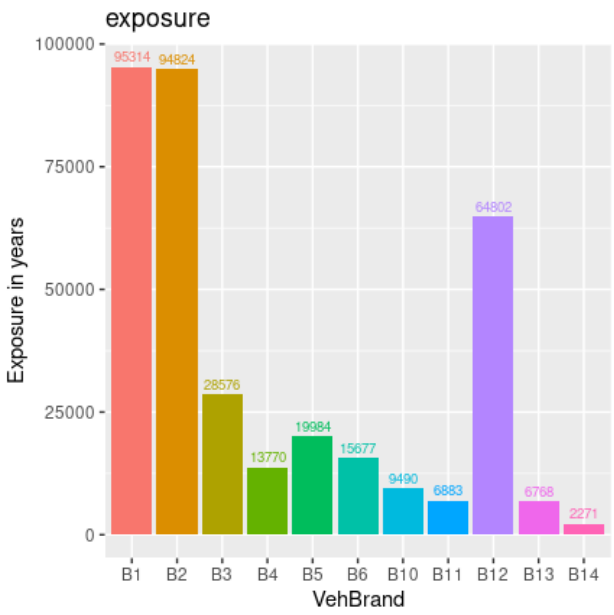


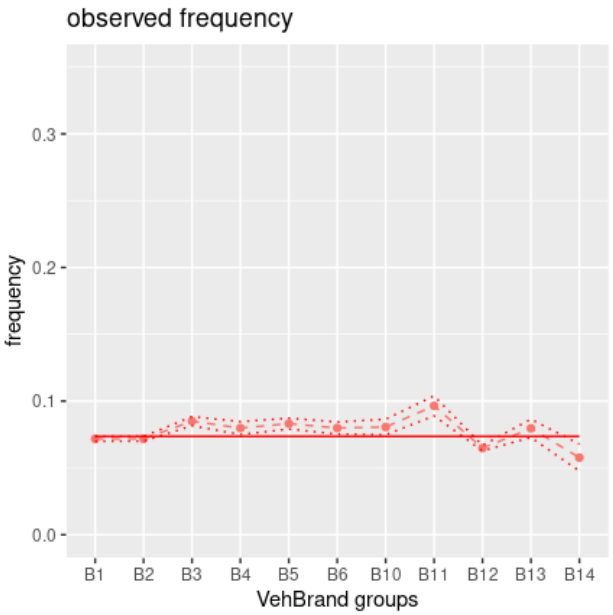


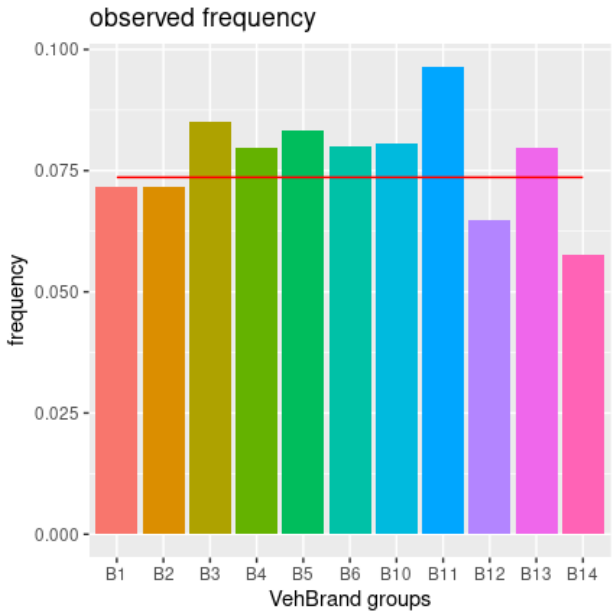
VehBrand

```
runMultiPlot(dat, VarName = "VehBrand")
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



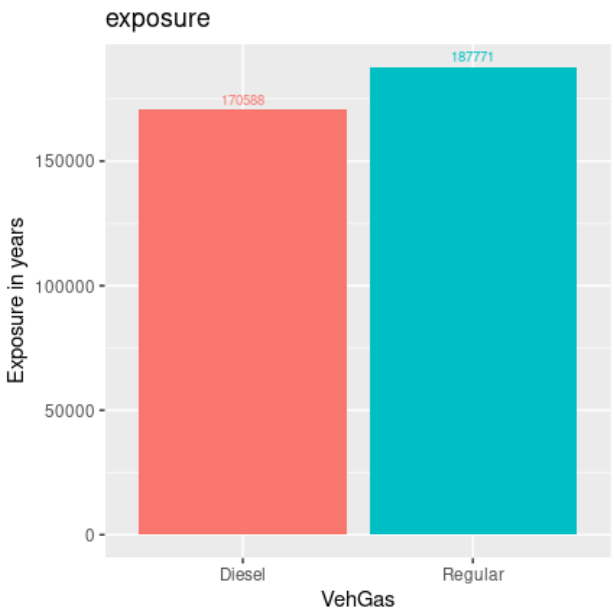


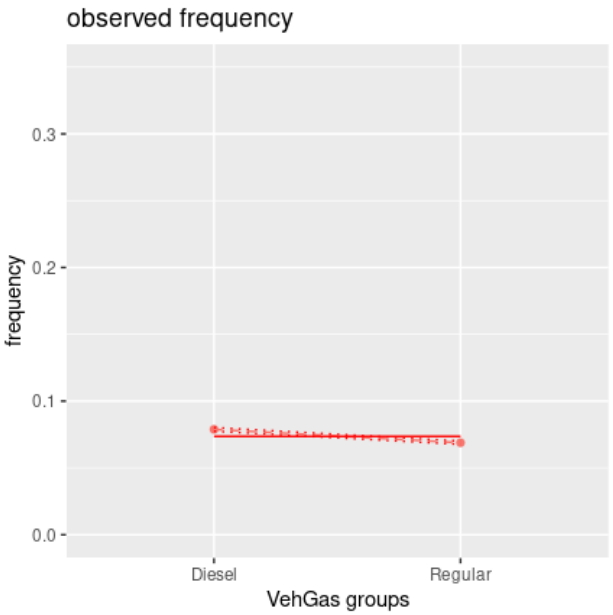


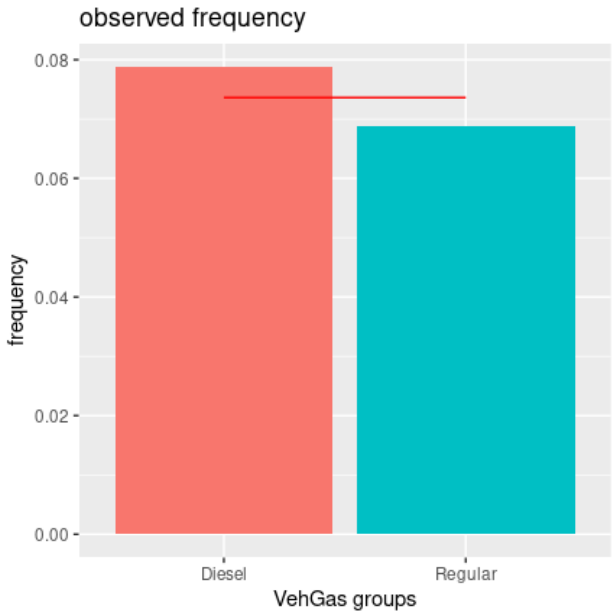
VehGas

```
runMultiPlot(dat, VarName = "VehGas")
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



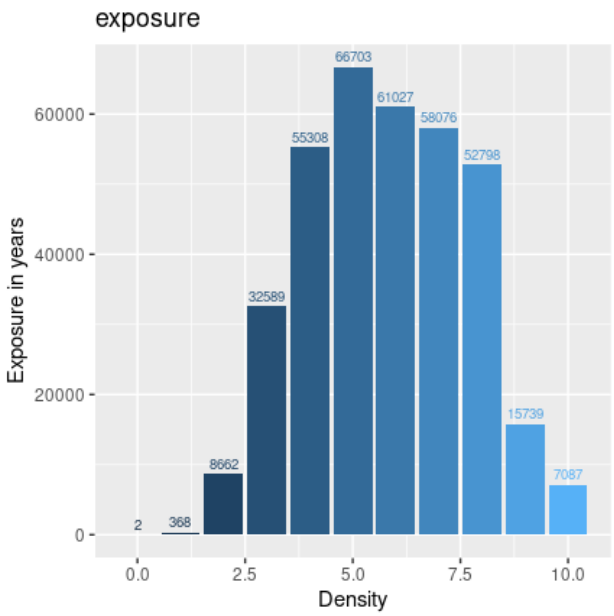


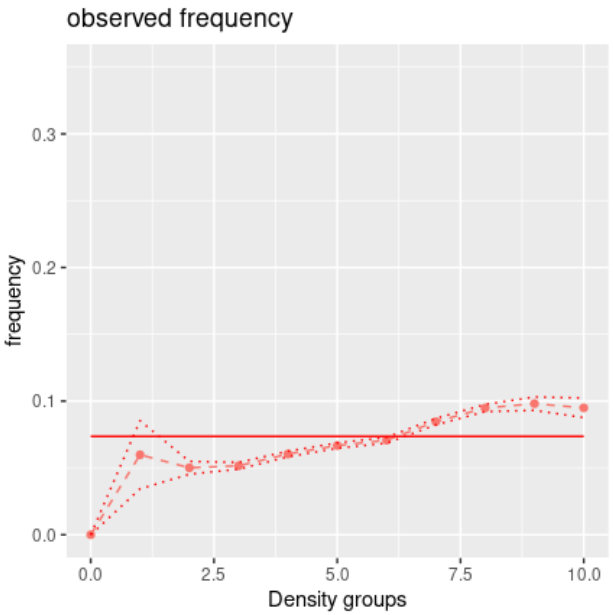


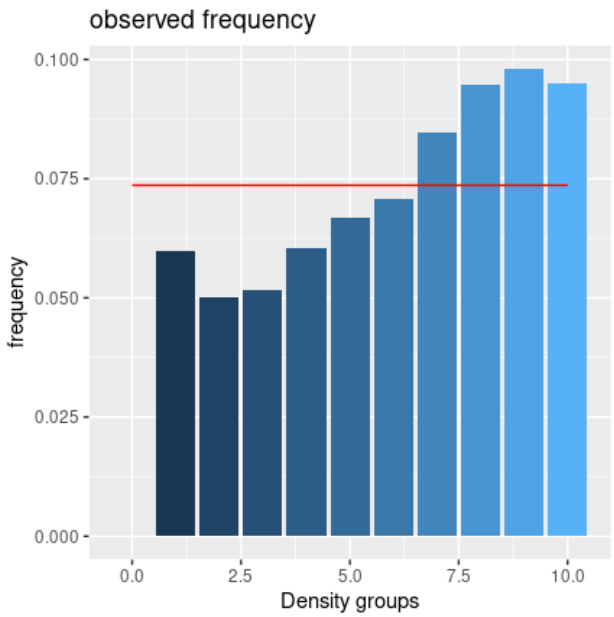
Density

```
runMultiPlot(dat, VarName = "Density")
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



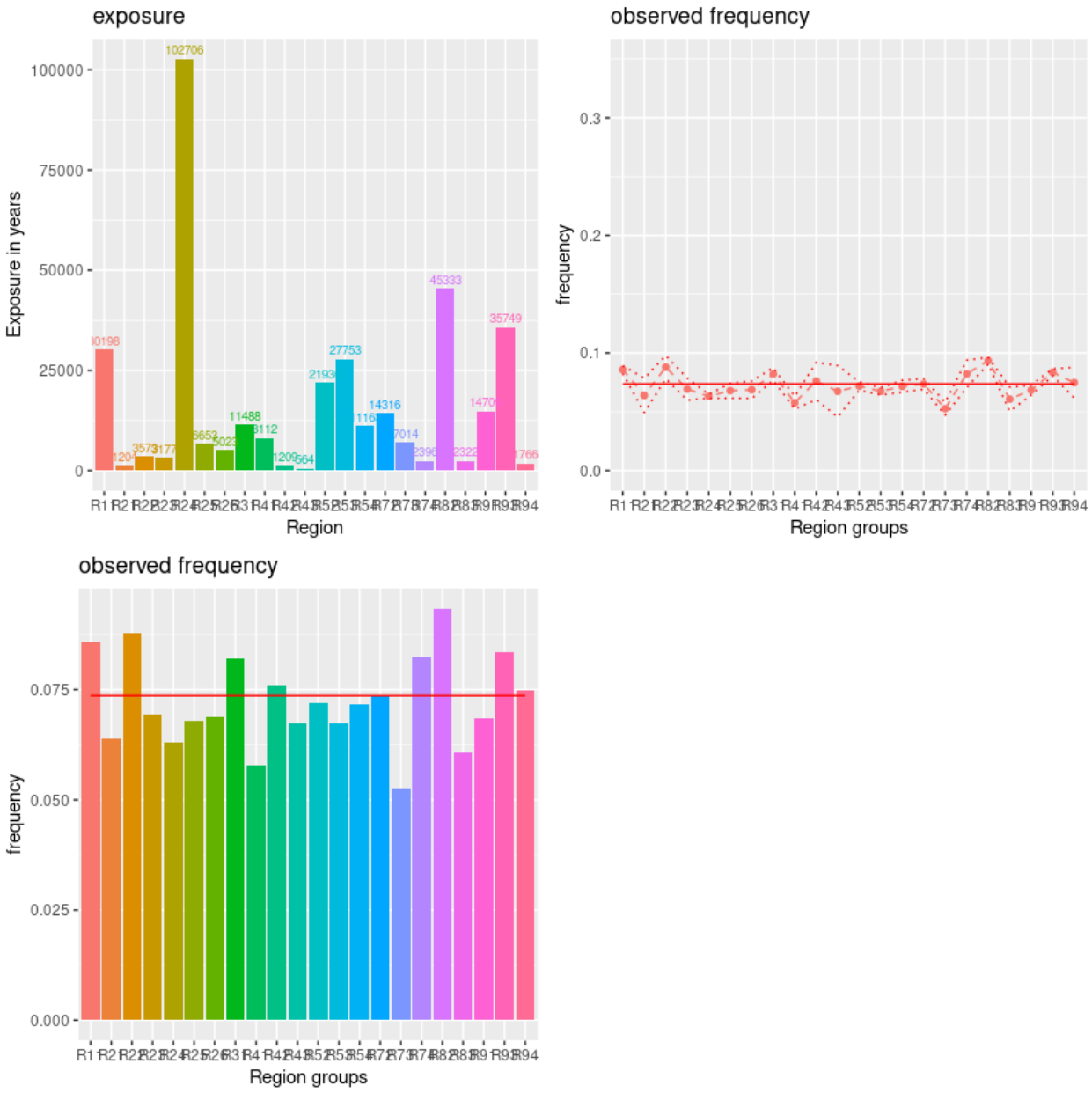




Region

```
runMultiPlot(dat, VarName = "Region")
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



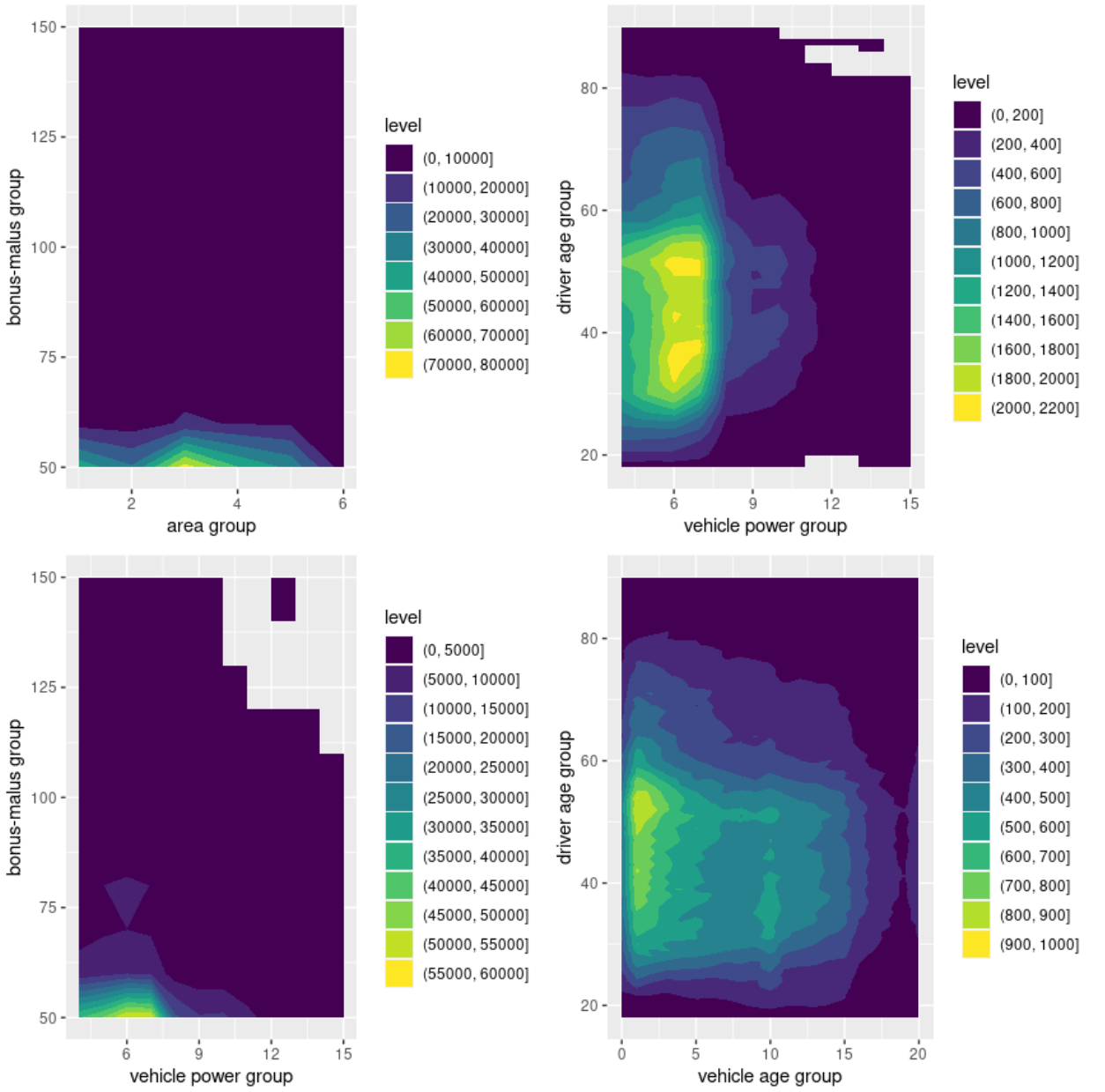
Two-dimensional contour plots: Exposure

In the previous sections, we have focused on the one-dimensional marginal exposure and frequencies. We provide the two-dimensional plots of the portfolio distribution (exposure) of some variables. These plots are useful to detect collinearity in the feature components.

The function below takes two variables as inputs and their names, and `plot_2dim_contour` shows the sum of the exposure by a unique combination of variable 1 and variable 2.

```
p1 <- plot_2dim_contour(dat, "Area", "BonusMalus", "area group",  "bonus-malus group")
p2 <- plot_2dim_contour(dat, "VehPower", "DrivAge", "vehicle power group",  "driver age group")
p3 <- plot_2dim_contour(dat, "VehPower", "BonusMalus", "vehicle power group",  "bonus-malus group")
p4 <- plot_2dim_contour(dat, "VehAge", "DrivAge", "vehicle age group",  "driver age group")

grid.arrange(p1, p2, p3, p4, ncol = 2)
```



Observations are: (a) the area code has a slight positive dependence with the bonus-malus level and a slight negative dependence with the vehicle age and driver's age, (b) the vehicle power has a slight positive dependence with driver's age, (c) the vehicle power has a slight negative dependence with the bonus-malus level, (d) younger people drive newer cars.

Exercise: Change the function `plot_2dim_contour` to use the claim frequency as the color in the charts above, and not the exposure.

Exercise: Change the function `plot_2dim_contour` to use the number of policies as the color in the charts above, and not the exposure.

Exercise: Look at other combinations of variables in the data, e.g. area code and the density of the population.

Correlations

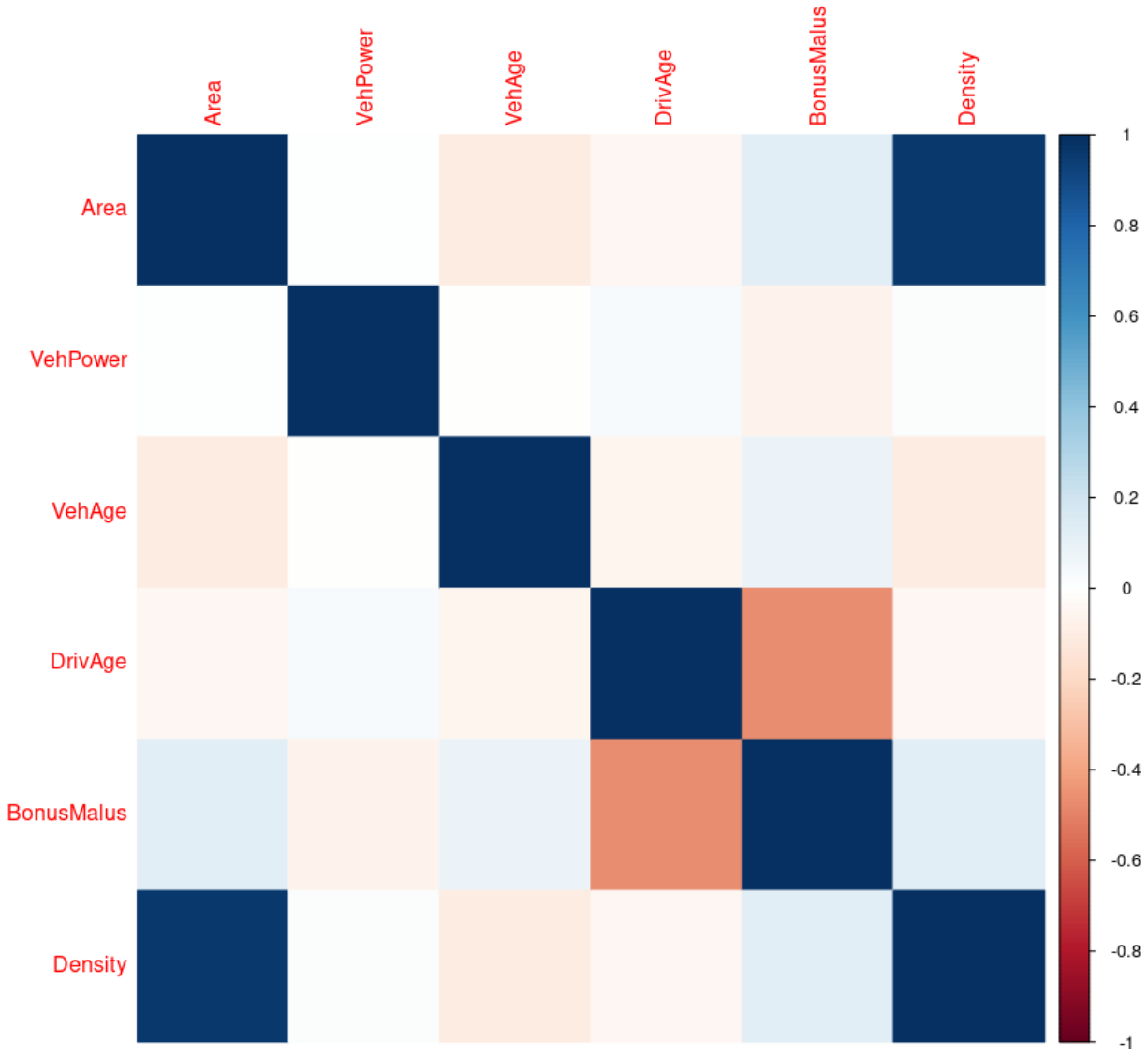
Let us calculate the Pearson and Spearman correlation between the variables.

```
df_cor <- dat %>%
  select(Area, VehPower, VehAge, DrivAge, BonusMalus, Density)
df_cor$Area <- as.numeric(df_cor$Area)
df_cor$VehPower <- as.numeric(df_cor$VehPower)
```

```
M <- round(cor(df_cor, method = "pearson"), 2)
knitr::kable(M)
```

	Area	VehPower	VehAge	DrivAge	BonusMalus	Density
Area	1.00	0.00	-0.11	-0.05	0.12	0.96
VehPower	0.00	1.00	-0.01	0.03	-0.07	0.01
VehAge	-0.11	-0.01	1.00	-0.06	0.08	-0.11
DrivAge	-0.05	0.03	-0.06	1.00	-0.47	-0.05
BonusMalus	0.12	-0.07	0.08	-0.47	1.00	0.12
Density	0.96	0.01	-0.11	-0.05	0.12	1.00

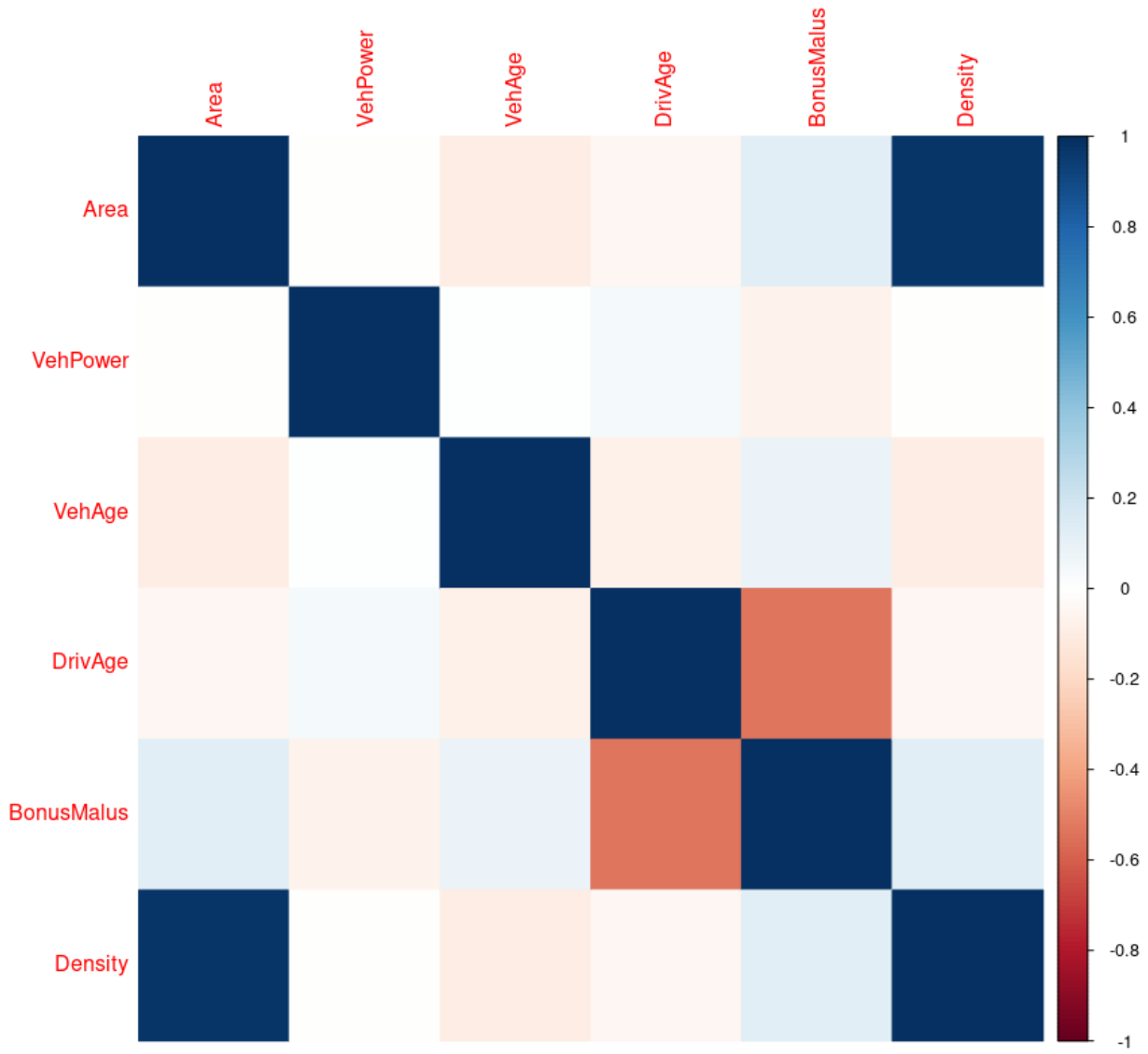
```
corrplot(M, method = "color")
```




```
M <- round(cor(df_cor, method = "spearman"), 2)
knitr::kable(M)
```

	Area	VehPower	VehAge	DrivAge	BonusMalus	Density
Area	1.00	-0.01	-0.10	-0.05	0.13	0.97
VehPower	-0.01	1.00	0.00	0.04	-0.07	-0.01
VehAge	-0.10	0.00	1.00	-0.08	0.08	-0.10
DrivAge	-0.05	0.04	-0.08	1.00	-0.54	-0.05
BonusMalus	0.13	-0.07	0.08	-0.54	1.00	0.13
Density	0.97	-0.01	-0.10	-0.05	0.13	1.00

```
corrplot(M, method = "color")
```



Geographical maps

As Region is available in the data, we are interested in plotting the claim frequencies by region in a geographical map.

First, let us calculate the metrics per region, transforming the categorical variables to numerical in order to be able to compare values. Mutating categorical variables to numerical is not ideal, but it allows to visualize data in a more convenient form and still derive some insights.

```
reg_sum <- dat %>%
  group_by(Region) %>%
  mutate(VehGas = factor(VehGas)) %>%
```

```
  mutate_at(c("Area", "VehPower", "VehGas"), as.numeric) %>%
  summarize(NrObs = length(Exposure),
            Exp = sum(Exposure),
            Freq = sum(ClaimNb) / sum(Exposure),
            Area = mean(Area),
            VehPower = mean(VehPower),
            VehAge = mean(VehAge),
            DrivAge = mean(DrivAge),
            BonusMalus = mean(BonusMalus),
            VehGas = mean(VehGas),
            Density = mean(Density))

knitr::kable(head(reg_sum, n = 10))
```

Region	NrObs	Exp	Freq	Area	VehPower	VehAge	DrivAge	BonusMalus	VehGas	Density
R11	69791	30198.132	0.0858000	4.745325	6.722758	4.771532	44.62333	62.51565	1.607987	8.113639
R21	3026	1204.342	0.0639354	2.674818	6.482155	4.017845	44.48579	59.86451	1.405155	5.127231
R22	7994	3572.840	0.0878853	3.193145	6.638229	5.565924	41.90418	63.21366	1.449712	5.779335
R23	8784	3176.882	0.0692503	3.497609	6.412910	6.591644	43.41541	60.00342	1.451730	6.183288
R24	160601	102706.443	0.0630438	2.418939	6.332819	8.836446	46.13285	58.81557	1.479630	4.798575
R25	10893	6652.651	0.0679428	2.971633	6.456899	7.104471	45.80621	59.01129	1.483980	5.586799
R26	10492	5022.989	0.0686842	2.570816	6.441098	5.992280	45.11580	60.56996	1.497331	4.890202
R31	27285	11488.074	0.0821722	3.845043	6.172219	6.205717	40.89141	63.39674	1.428770	6.671541
R41	12990	8112.488	0.0576888	3.452887	6.217937	6.740801	48.50092	57.03849	1.587067	6.132640
R42	2200	1208.636	0.0761189	3.901364	6.502727	6.198182	44.89591	61.44545	1.615909	6.642727

The visualizations can be done using data containing the boundaries of regions and the two R packages `rgdal` and `rgeos`. Please check their reference manual for further details about them.

First, we need to import the files which contain the map to be displayed. Geo data have some standard formats, which we are not going to discuss here. See the R packages `rgal` and `rgeos` for further information and links.

```
# Downloaded shapefiles from http://www.diva-gis.org/gData and extracted all the files from the zip fil
area <- rgdal::readOGR(file.path("../../data/shapefiles", "FRA_adm2.shp"))
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/work/actu-deep-learn-r/data/shapefiles/FRA_adm2.shp", layer: "FRA_adm2"
## with 96 features
## It has 11 fields
## Integer64 fields read as strings:  ID_0 ID_1 ID_2
```

Second, we need to merge the aggregated information from the insurance data to the geographical data.

```
reg_sum$id <- sapply(reg_sum$Region, substr, 2, 3)
area_points <- fortify(area, region = "ID_1")  # convert to data.frame
```

Unfortunately, the numerical regional labels in the geographical data (labeled as $1, ..., 22$) do not match the numerical labels in the insurance data, hence we need to convert them in order to be able to merge the insurance and geographical data by a unique key. We do it manually (see `mapvalues` for another way).

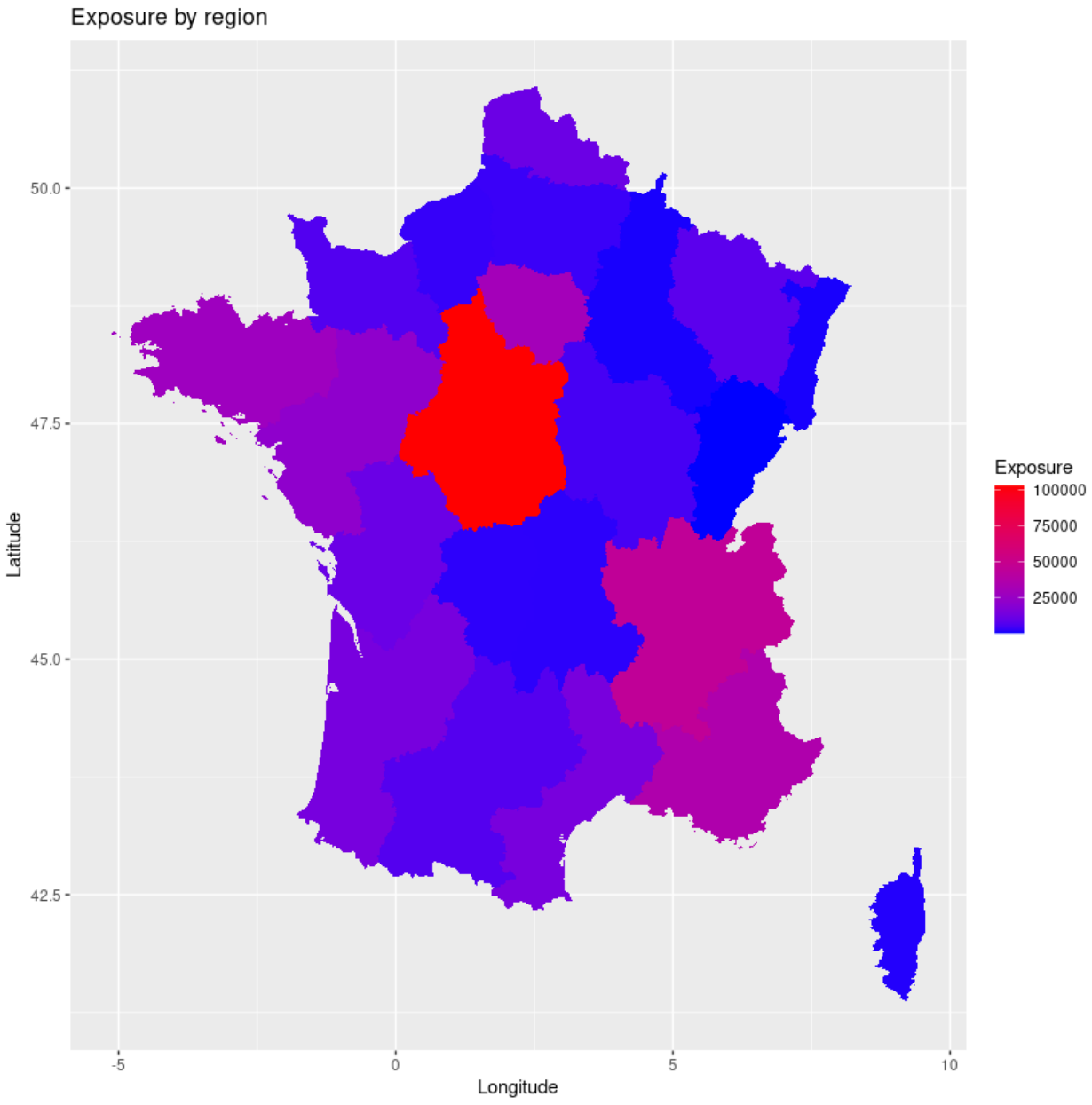
```
area_points$id <- recode(
  area_points$id,
  "1"="42","2"="72","3"="83","4"="11","5"="25","6"="26","7"="53","8"="24","9"="21",
  "10"="94","11"="43","12"="23","13"="91","14"="74","15"="41","16"="73","17"="31",
```

```
  "18"="52","19"="22","20"="54","21"="93","22"="82"
)
```

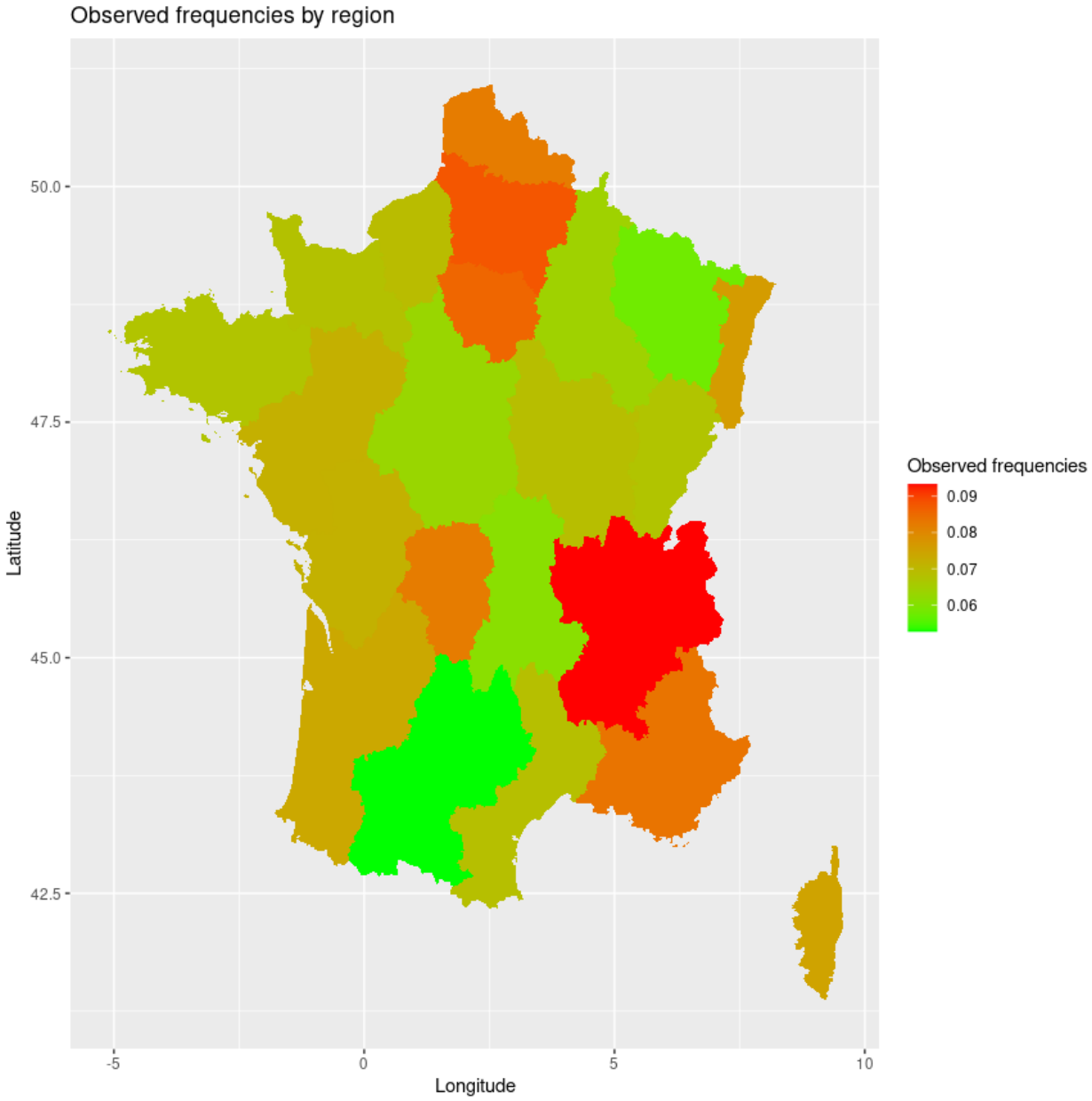
```
area_points <- merge(
  area_points,
  reg_sum[, c("id","Exp","Freq","Area","VehPower","VehAge","DrivAge","BonusMalus","VehGas","Density")],
  by.x = "id",
  by.y = "id",
  all.x = TRUE
)
area_points <- area_points[order(area_points$order), ]  # Has to be ordered correctly to plot.
```

Third, we can do the map plots.

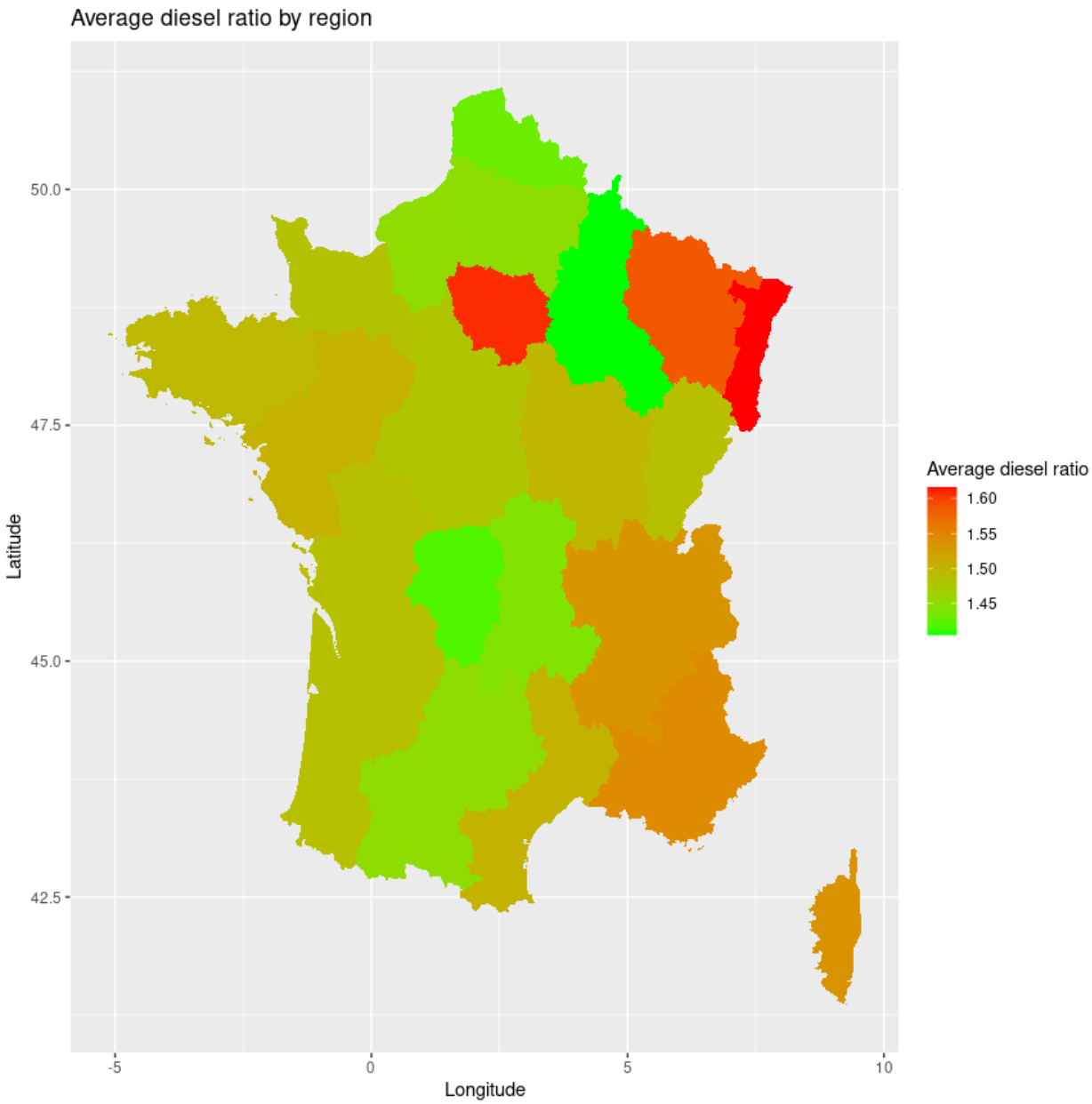
```
plotMap(area_points, "Exp", "Exposure", "blue", "red")
```



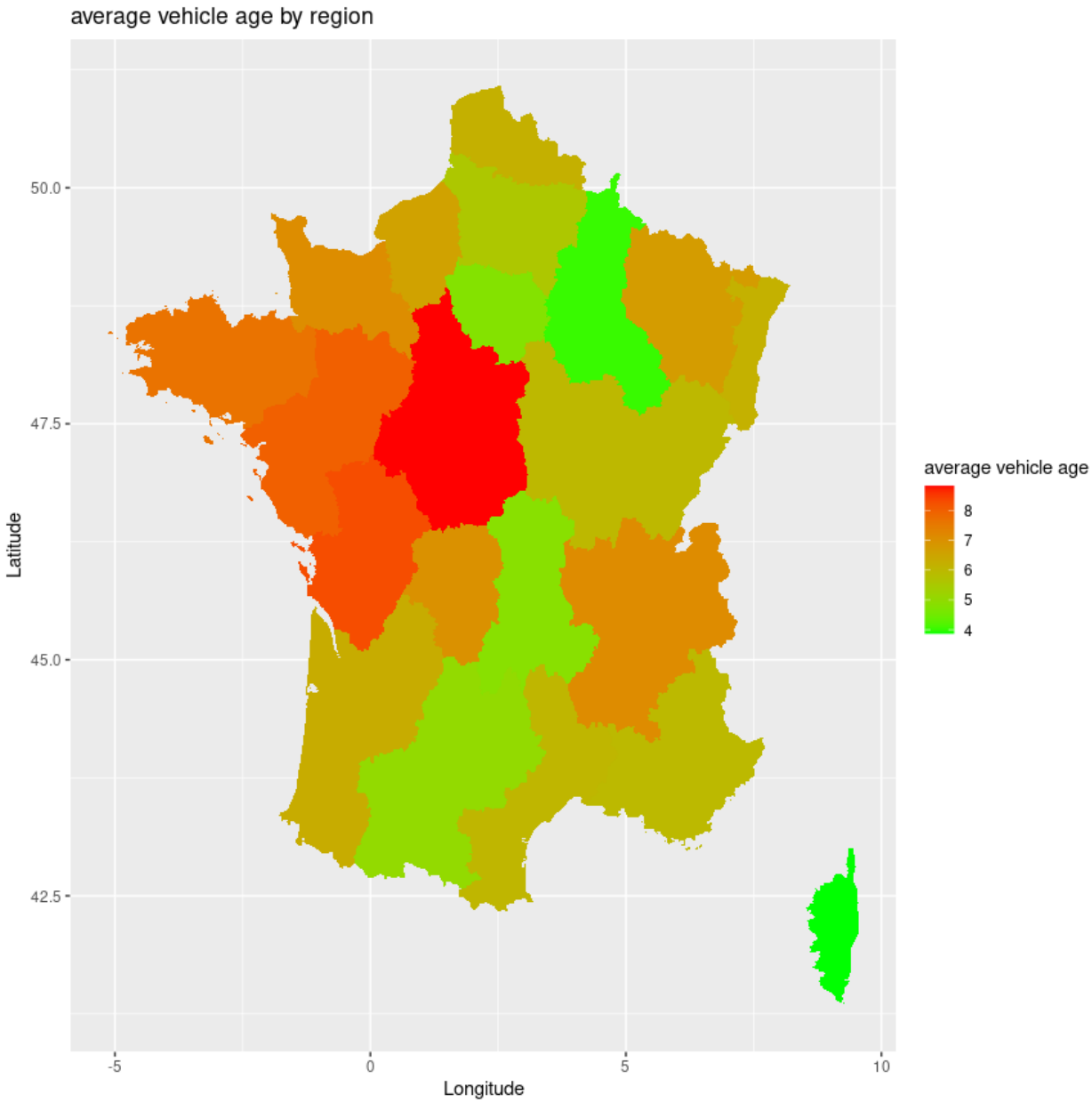
```
plotMap(area_points, "Freq", "Observed frequencies", "green", "red")
```




```
plotMap(area_points, "VehGas", "Average diesel ratio", "green", "red")
```




```
plotMap(area_points, "VehAge", "average vehicle age", "green", "red")
```

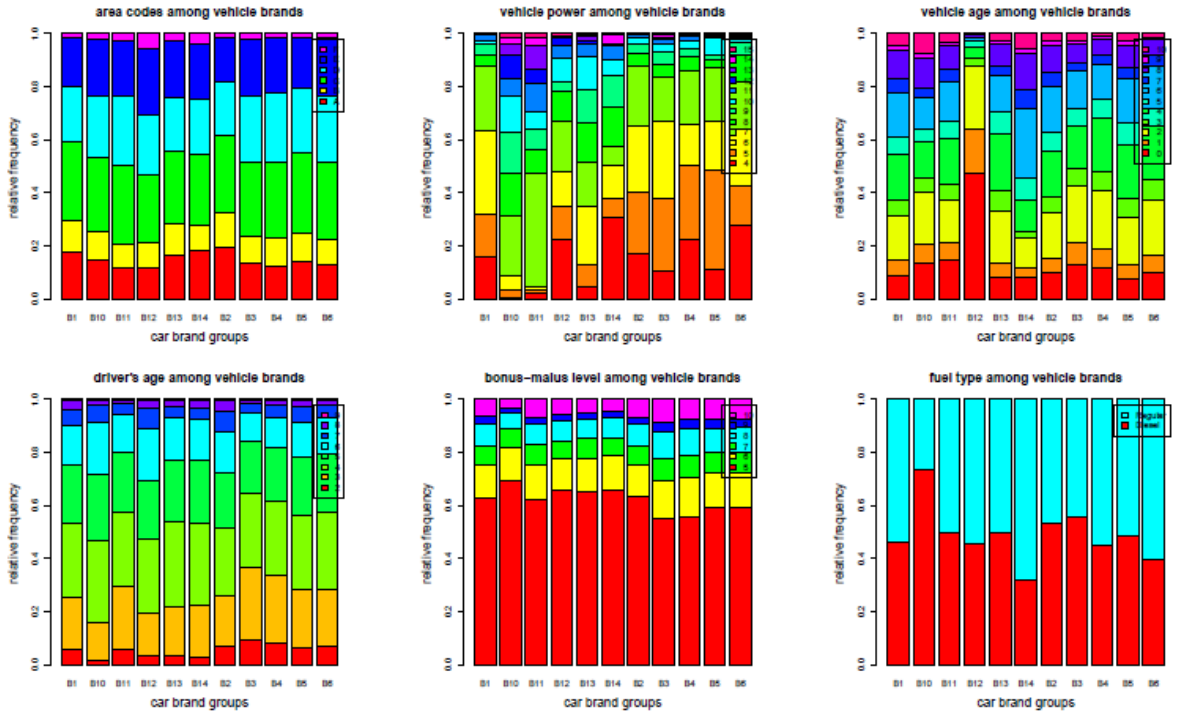


Please find some more charts on the raw data in the tutorial.

Exercise: Do additional plots for the following variables: `density`, `BonusMalus` and `VehGas`. You might need to convert the categorical features to some numerical values in order to get a meaningful plot.

Exercise: The coloring of the mas is not necessarily illustrative due to the skewed distribution of metrics by region. Amend the function plotMap to better capture the skewness.

Exercise: In the tutorial, you find the subsequent charts (which describe the distribution of the variables `Area`, `VehPower`, `VehAge`, `DrivAge`, `BonusMalus`, `VehGas` for each car brand `VehBrand` individually), write the code for them.



Session Info

The html is generated with the follow packages (which might be slightly newer than the ones used in the published tutorial).

```
sessionInfo()
```

```
## R version 4.0.5 (2021-03-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.2 LTS
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.8.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=C
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] repr_1.1.3    corrplot_0.90 gridExtra_2.3 ggplot2_3.3.5 dplyr_1.0.7
## [6] rgdal_1.5-27  sp_1.4-5
##
```

```
## loaded via a namespace (and not attached):
##  [1] highr_0.9         pillar_1.6.2      compiler_4.0.5    base64enc_0.1-3
##  [5] tools_4.0.5       digest_0.6.27     viridisLite_0.4.0 jsonlite_1.7.2
##  [9] evaluate_0.14     lifecycle_1.0.0   tibble_3.1.4      gtable_0.3.0
## [13] lattice_0.20-41   pkgconfig_2.0.3   rlang_0.4.11      rstudioapi_0.13
## [17] cli_2.5.0         DBI_1.1.1         yaml_2.2.1        xfun_0.23
## [21] withr_2.4.2       stringr_1.4.0     knitr_1.34        rgeos_0.5-7
## [25] generics_0.1.0    vctrs_0.3.8       isoband_0.2.4     grid_4.0.5
## [29] tidyselect_1.1.1  glue_1.4.2        R6_2.5.0          fansi_0.4.2
## [33] foreign_0.8-81    rmarkdown_2.11    farver_2.1.0      purrr_0.3.4
## [37] magrittr_2.0.1    maptools_1.1-2    ps_1.6.0          scales_1.1.1
## [41] ellipsis_0.3.2    htmltools_0.5.1.1 assertthat_0.2.1  colorspace_2.0-1
## [45] labeling_0.4.2    utf8_1.2.1        stringi_1.6.1     munsell_0.5.0
## [49] crayon_1.4.1
```