Generalized Linear Model (GLM)

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(mgcv)
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(tibble)
library(ggplot2)
library(splitTools)
```

```
# plotting parameters in R Markdown notebook
knitr::opts_chunk$set(fig.width = 9, fig.height = 9)
# plotting parameters in Jupyter notebook
```

```r
library(repr)  # only needed for Jupyter notebook
options(repr.plot.width = 9, repr.plot.height = 9)
```

Set global parameters

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

```r
# set seed to obtain best reproducibility. note that the underlying architecture may affect results non
seed <- 100
```

Helper functions

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

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

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

```r
# Poisson deviance
PoissonDeviance <- function(pred, obs) {
  200 * (sum(pred) - sum(obs) + sum(log((obs / pred)^(obs)))) / length(pred)
}
```

```r
plot_freq <- function(test, xvar, title, model, mdlvariant) {
  out <- test %>% group_by(!!sym(xvar)) %>% summarize(obs = sum(ClaimNb) / sum(Exposure),
                                                      pred = sum(!!sym(mdlvariant)) / sum(Exposure))

  ggplot(out, aes(x = !!sym(xvar), group = 1)) + geom_point(aes(y = pred, colour = model)) +
    geom_point(aes(y = obs, colour = "observed")) +
    geom_line(aes(y = pred, colour = model), linetype = "dashed") +
    geom_line(aes(y = obs, colour = "observed"), linetype = "dashed") +
    ylim(0, 0.35) + labs(x = xvar, y = "frequency", title = title) +
    theme(legend.position = "bottom")
}
```

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.

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

General data preprocessing

A priori, there is not sufficient information about this data to do a sensible decision about the best consideration of the exposure measure, either as feature or as offset. In the following we treat the exposure always as an offset.

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. We leave away the rounding used in the first notebook, which were mainly used for nicer visualizations of the data.

We are adding a group_id identifying rows possibly referring to the same policy. Respecting group_id in data splitting techniques (train/test, cross-validation) is essential. This is different to the tutorial where another splitting has been used. As a consequence, the figures in this notebook do not match the figures in the tutorial, but the conclusions drawn are the same.

In addition to the previous tutorial, we decide to truncate the `ClaimNb` and the $`Exposure` in order to correct for unreasonable data entries and simplifications for the modeling part.

```r
# Grouping id
distinct <- freMTPL2freq %>%
  distinct_at(vars(-c(IDpol, Exposure, ClaimNb))) %>%
  mutate(group_id = row_number())
```

```r
dat <- freMTPL2freq %>%
  left_join(distinct) %>%
  mutate(ClaimNb = pmin(as.integer(ClaimNb), 4),
         VehAge = pmin(VehAge, 20),
         DrivAge = pmin(DrivAge, 90),
         BonusMalus = pmin(BonusMalus, 150),
         Density = round(log(Density), 2),
         VehGas = factor(VehGas),
         Exposure = pmin(Exposure, 1))
```

```
## Joining, by = c("Area", "VehPower", "VehAge", "DrivAge", "BonusMalus", "VehBrand", "VehGas", "Densit
```

```r
# Group sizes of suspected clusters
table(table(dat[, "group_id"]))
```

```
##
##      1      2      3      4      5      6      7      8      9     10     11
## 429576  84201  13940   2437    966    754    720    475    400    269    142
##     12     13     14     15     18     22
##    191      3      1      2      1      1
```

Feature pre-processing for generalized linear models

As previously mentioned, typically features x_i need pre-processing before being used for a specific model. In our Poisson GLM the regression function is modeled by a log-linear shape in the continuous feature components. From the marginal empirical frequency plots in the previous file we see that such a log-linear form is not always appropriate. We make the following choices here:

- `Area`: we choose a continuous (log-linear) feature component for $\{A, ..., F\} \mapsto \{1, ..., 6\}$
- `VehPower`: we choose a categorical feature component where we merge vehicle power groups bigger and equal to 9 (totally 6 classes)
- `VehAge`: we build 3 categorical classes $[0, 1), [1, 10], (10, \infty)$
- `DrivAge`: we build 7 categorical classes $[18, 21), [21, 26), [26, 31), [31, 41), [41, 51), [51, 71), [71, \infty)$

- `BonusMalus`: continuous log-linear feature component (we cap at value 150)
- `VehBrand`: categorical feature component (totally 11 classes)
- `VehGas`: binary feature component;
- `Density`: log-density is chosen as continuous log-linear feature component (note that we have very small volumes for small log-densities)
- `Region`: categorical feature component (totally 22 classes)

Thus, we consider 3 continuous feature components (`Area`, `BonusMalus`, `log-Density`), 1 binary feature component (`VehGas`) and 5 categorical feature components (`VehPower`, `VehAge`, `DrivAge`, `VehBrand`, `Region`). The categorical classes for `VehPower`, `VehAge` and `DrivAge` have been done based on expert opinion, only. This expert opinion has tried to find homogeneity within class labels (levels) and every class label should receive a sufficient volume (of observations). We could also make a data-driven choice by using a (marginal) regression tree for different feature components, see references in the tutorial.

```
dat2 <- dat %>% mutate(
  AreaGLM = as.integer(Area),
  VehPowerGLM = as.factor(pmin(VehPower, 9)),
  VehAgeGLM = cut(VehAge, breaks = c(-Inf, 0, 10, Inf), labels = c("1","2","3")),
  DrivAgeGLM = cut(DrivAge, breaks = c(-Inf, 20, 25, 30, 40, 50, 70, Inf), labels = c("1","2","3","4","5
  BonusMalusGLM = as.integer(pmin(BonusMalus, 150)),
  DensityGLM = as.numeric(Density),
  VehAgeGLM = relevel(VehAgeGLM, ref = "2"),
  DrivAgeGLM = relevel(DrivAgeGLM, ref = "5"),
  Region = relevel(Region, ref = "R24")
)
```

We remark that for categorical variables we use the data type factor in R. This data type automatically considers dummy coding in the corresponding R procedures. Categorical variables are initialized to one class (reference level). We typically initialize to the class with the biggest volume. This initialization is achieved by the command relevel, see above. This initialization does not influence the fitted means but provides a unique parametrization. See `?relevel` for further details.

Inspect the prepared dataset

```
knitr::kable(head(dat2))
```

IDpol	Exposure	Area	VehPower	VehAge	DrivAge	BonusMalus	VehBrand	VehGas	Density	Region	ClaimNb	ClaimTotal	group_id	AreaGLM	VehPowerGLM	VehAgeGLM	DrivAgeGLM	BonusMalusGLM	DensityGLM
1	0.10	D	5	0	55	50	B12	Regular	7.10	R82	0	0	1	4	5	1	6	50	7.10
3	0.77	D	5	0	55	50	B12	Regular	7.10	R82	0	0	1	4	5	1	6	50	7.10
5	0.75	B	6	2	52	50	B12	Diesel	3.99	R22	0	0	2	2	6	2	6	50	3.99
10	0.09	B	7	0	46	50	B12	Diesel	4.33	R72	0	0	3	2	7	1	5	50	4.33
11	0.84	B	7	0	46	50	B12	Diesel	4.33	R72	0	0	3	2	7	1	5	50	4.33
13	0.52	E	6	2	38	50	B12	Regular	8.01	R31	0	0	4	5	6	2	4	50	8.01

```
str(dat2)
```

```
## 'data.frame':    678007 obs. of  20 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  7.1 7.1 3.99 4.33 4.33 8.01 8.01 4.92 4.92 4.09 ...
##  $ Region       : Factor w/ 22 levels "R24","R11","R21",..: 18 18 4 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 ...
##  $ group_id     : int  1 1 2 3 3 4 4 5 5 6 ...
##  $ AreaGLM      : int  4 4 2 2 2 5 5 3 3 2 ...
##  $ VehPowerGLM  : Factor w/ 6 levels "4","5","6","7",..: 2 2 3 4 4 3 3 4 4 4 ...
##  $ VehAgeGLM    : Factor w/ 3 levels "2","1","3": 2 2 1 2 2 1 1 2 2 2 ...
##  $ DrivAgeGLM   : Factor w/ 7 levels "5","1","2","3",..: 6 6 6 1 1 5 5 5 5 1 ...
##  $ BonusMalusGLM: int  50 50 50 50 50 50 50 68 68 50 ...
##  $ DensityGLM   : num  7.1 7.1 3.99 4.33 4.33 8.01 8.01 4.92 4.92 4.09 ...
```

```
summary(dat2)
```

```
##      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   : 59.76   B3     : 53395
##  3rd Qu.:11.000   3rd Qu.:55.0   3rd Qu.: 64.00   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.: 4.520   R82    : 84752   1st Qu.:      0
##                   Median : 5.970   R93    : 79315   Median :      0
##                   Mean   : 5.982   R11    : 69791   Mean   :     88
##                   3rd Qu.: 7.410   R53    : 42122   3rd Qu.:      0
##                   Max.   :10.200   R52    : 38751   Max.   :4075401
##                                    (Other):202675
##     ClaimNb           group_id         AreaGLM      VehPowerGLM VehAgeGLM
##  Min.   :0.00000   Min.   :     1   Min.   :1.00   4:115343    2:434492
##  1st Qu.:0.00000   1st Qu.:149318   1st Qu.:2.00   5:124821    1: 57739
##  Median :0.00000   Median :273211   Median :3.00   6:148976    3:185776
##  Mean   :0.03891   Mean   :275320   Mean   :3.29   7:145401
##  3rd Qu.:0.00000   3rd Qu.:404072   3rd Qu.:4.00   8: 46956
##  Max.   :4.00000   Max.   :534079   Max.   :6.00   9: 96510
##
##  DrivAgeGLM BonusMalusGLM      DensityGLM
##  5:165185   Min.   : 50.00   Min.   : 0.000
##  1:  6816   1st Qu.: 50.00   1st Qu.: 4.520
##  2: 32079   Median : 50.00   Median : 5.970
##  3: 65594   Mean   : 59.76   Mean   : 5.982
##  4:170097   3rd Qu.: 64.00   3rd Qu.: 7.410
##  6:198871   Max.   :150.00   Max.   :10.200
```

```
##  7: 39365
```

Modeling

With the prepared dataset ready, we are ready for the modeling part.

One of the frequent mistakes is to do the pre-processing after the split or inconsistently between various model to be compared. This results in not a fair comparison of the model performance.

In the following, we will fit various claim frequency models based on a Poisson assumption, to be more precise we make the following assumptions:

Model Assumptions 3.2 (Model GLM1) *Choose feature space $\mathcal{X}$ as in* (3.1) *and define the regression function $\lambda : \mathcal{X} \to \mathbb{R}_+$ by*

$$\boldsymbol{x} \;\mapsto\; \log \lambda(\boldsymbol{x}) = \beta_0 + \sum_{l=1}^{d} \beta_l x_l,$$

for parameter vector $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_d)' \in \mathbb{R}^{d+1}$. Assume that for $i \geq 1$

$$N_i \overset{\text{ind.}}{\sim} \operatorname{Poi}\left(\lambda(\boldsymbol{x}_i) v_i\right).$$

A priori, there is not sufficient information about this data to do a sensible decision about the best consideration of the exposure measure, either as feature or as offset. In the following we treat the exposure v_i as offset to be consistent.

Split train and test data

First, we split the dataset into train and test. Due to the potential grouping of rows in policies we can not just do a random split. For this purpose, we use the function `partition(...)` from the `splitTools` package.

```r
ind <- partition(dat2[["group_id"]], p = c(train = 0.8, test = 0.2),
                 seed = seed, type = "grouped")
train <- dat2[ind$train, ]
test <- dat2[ind$test, ]
```

It describes our choices of the learning data set $\mathcal{D}$ and the test data set $\mathcal{T}$ That is, we allocate at random 80% of the policies to $\mathcal{D}$ and the remaining 20% of the policies to $\mathcal{T}$.

Usually, an 90/10 or 80/20 is used for training and test data. This is a rule-of-thumb and best practice in modeling. A good explanation can be found here, citing as follows: "There are two competing concerns: with less training data, your parameter estimates have greater variance. With less testing data, your performance statistic will have greater variance. Broadly speaking you should be concerned with dividing data such that neither variance is too high, which is more to do with the absolute number of instances in each category rather than the percentage."

Exercise: Change the split from 90%/10% to 80%/20% to compare the results. If you use a split like 50%/50% the results are much worse on the test data set.

Exercise: Check how the final results differ if a different seed is used.

```r
# size of train/test
n_l <- nrow(train)
n_t <- nrow(test)
sprintf("Number of observations (train): %s", n_l)
```

```
## [1] "Number of observations (train): 542331"
```

```r
sprintf("Number of observations (test): %s", n_t)
```

```
## [1] "Number of observations (test): 135676"
```

```r
# Claims frequency of train/test
sprintf("Empirical frequency (train): %s", round(sum(train$ClaimNb) / sum(train$Exposure), 4))
```

```
## [1] "Empirical frequency (train): 0.0736"
```

```r
sprintf("Empirical frequency (test): %s", round(sum(test$ClaimNb) / sum(test$Exposure), 4))
```

```
## [1] "Empirical frequency (test): 0.0736"
```

```r
# exposure and number of claims of train/test
# see https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3164764, p. 11 (figures do not match)
train1 <- train %>% group_by(ClaimNb) %>% summarize(n = n(), exp = sum(Exposure))
print(train1)
```

```
## # A tibble: 5 × 3
##   ClaimNb      n       exp
##     <dbl>  <int>     <dbl>
## 1       0 522390 272711.
## 2       1  18845  13060.
## 3       2   1044    730.
## 4       3     47     31.0
## 5       4      5      1.51
```

```r
print(round(100 * train1$n / sum(train1$n), 3))
```

```
## [1] 96.323  3.475  0.193  0.009  0.001
```

```r
test1 <- test %>% group_by(ClaimNb) %>% summarize(n = n(), exp = sum(Exposure))
print(test1)
```

```
## # A tibble: 5 × 3
##   ClaimNb      n      exp
##     <dbl>  <int>    <dbl>
## 1       0 130679 68379.
## 2       1   4726  3255.
## 3       2    254   179.
## 4       3     15    11.4
## 5       4      2     1.56
```

```r
print(round(100 * test1$n / sum(test1$n), 3))
```

```
## [1] 96.317  3.483  0.187  0.011  0.001
```

Store model results

As we are going to compare various models, we create a table which stores the metrics we are going to use for the comparison and the selection of the best model.

```r
# table to store all model results for comparison
df_cmp <- tibble(
  model = character(),
  run_time = numeric(),
```

```
  parameters = numeric(),
  aic = numeric(),
  in_sample_loss = numeric(),
  out_sample_loss = numeric(),
  avg_freq = numeric()
)
```

Exercise: Think of other metrics to be included in the table for the model comparison, amend the table and the code below to store the new metrics in the table.

In the following, we fit and compare various claim frequency models. We compare them by using the metrics defined above.

GLM0 (Homogeneous Model)

Let us start with the trivial model where we estimate the global mean and no features are included.

Fitting

```
exec_time <- system.time(glm0 <- glm(ClaimNb ~ 1, data = train, offset = log(Exposure), family = poisson
exec_time[1:5]
```

```
##  user.self   sys.self    elapsed user.child  sys.child
##      7.862      5.499      4.368      0.000      0.000
```

```
summary(glm0)
```

```
##
## Call:
## glm(formula = ClaimNb ~ 1, family = poisson(), data = train,
##     offset = log(Exposure))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.3837  -0.3740  -0.2602  -0.1383   6.6467
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.608866   0.006885  -378.9   <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: 136692  on 542330  degrees of freedom
## Residual deviance: 136692  on 542330  degrees of freedom
## AIC: 177269
##
## Number of Fisher Scoring iterations: 6
```

Validation

```
# Predictions
train$fitGLM0 <- fitted(glm0)
```

```r
test$fitGLM0 <- predict(glm0, newdata = test, type = "response")
dat$fitGLM0 <- predict(glm0, newdata = dat2, type = "response")
```

```r
# in-sample and out-of-sample losses (in 10^(-2))
sprintf("100 x Poisson deviance GLM (train): %s", PoissonDeviance(train$fitGLM0, train$ClaimNb))
```

```
## [1] "100 x Poisson deviance GLM (train): 25.204447763748"
```

```r
sprintf("100 x Poisson deviance GLM (test): %s", PoissonDeviance(test$fitGLM0, test$ClaimNb))
```

```
## [1] "100 x Poisson deviance GLM (test): 25.3483497521802"
```

```r
# Overall estimated frequency
sprintf("average frequency (test): %s", round(sum(test$fitGLM0) / sum(test$Exposure), 6))
```

```
## [1] "average frequency (test): 0.073618"
```

```r
df_cmp[1, ] <- list("GLM0", round(exec_time[[3]], 0), length(coef(glm0)), round(AIC(glm0), 0),
                    round(PoissonDeviance(train$fitGLM0, as.vector(unlist(train$ClaimNb))), 4),
                    round(PoissonDeviance(test$fitGLM0, as.vector(unlist(test$ClaimNb))), 4),
                    round(sum(test$fitGLM0) / sum(test$Exposure), 4))
knitr::kable(df_cmp)
```

model	run_time	parameters	aic	in_sample_loss	out_sample_loss	avg_freq
GLM0	4	1	177269	25.2044	25.3483	0.0736

GLM1 (all feature components considered)

Fitting

```r
exec_time <- system.time(
  glm1 <- glm(ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM + BonusMalusGLM + VehBrand +
                        VehGas + DensityGLM + Region + AreaGLM,
              data = train, offset = log(Exposure), family = poisson()))
exec_time[1:5]
```

```
##  user.self   sys.self    elapsed user.child  sys.child
##    107.912    121.780     50.264      0.000      0.000
```

```r
summary(glm1)
```

```
##
## Call:
## glm(formula = ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM +
##     BonusMalusGLM + VehBrand + VehGas + DensityGLM + Region +
##     AreaGLM, family = poisson(), data = train, offset = log(Exposure))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5267  -0.3252  -0.2462  -0.1383   6.9267
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4.5071915  0.0474049 -95.079  < 2e-16 ***
## VehPowerGLM5    0.0560543  0.0243476   2.302 0.021321 *
## VehPowerGLM6    0.0881555  0.0238852   3.691 0.000224 ***
```

```
## VehPowerGLM7   0.0655053  0.0237692   2.756 0.005853 **
## VehPowerGLM8   0.0985604  0.0338024   2.916 0.003548 **
## VehPowerGLM9   0.2384515  0.0265444   8.983  < 2e-16 ***
## VehAgeGLM1    -0.0194227  0.0341911  -0.568 0.569993
## VehAgeGLM3    -0.1798662  0.0163248 -11.018  < 2e-16 ***
## DrivAgeGLM1    0.1147680  0.0498191   2.304 0.021240 *
## DrivAgeGLM2   -0.3541245  0.0332153 -10.661  < 2e-16 ***
## DrivAgeGLM3   -0.4673887  0.0281294 -16.616  < 2e-16 ***
## DrivAgeGLM4   -0.2752602  0.0203805 -13.506  < 2e-16 ***
## DrivAgeGLM6   -0.0595770  0.0188630  -3.158 0.001586 **
## DrivAgeGLM7   -0.0607337  0.0313548  -1.937 0.052748 .
## BonusMalusGLM  0.0273746  0.0004106  66.662  < 2e-16 ***
## VehBrandB2    -0.0090795  0.0193285  -0.470 0.638536
## VehBrandB3     0.0586285  0.0266943   2.196 0.028071 *
## VehBrandB4     0.0553248  0.0362296   1.527 0.126746
## VehBrandB5     0.0845338  0.0308544   2.740 0.006148 **
## VehBrandB6     0.0125473  0.0349836   0.359 0.719848
## VehBrandB10    0.0087216  0.0443095   0.197 0.843958
## VehBrandB11    0.1842800  0.0468777   3.931 8.46e-05 ***
## VehBrandB12   -0.2530951  0.0244159 -10.366  < 2e-16 ***
## VehBrandB13    0.0579934  0.0499195   1.162 0.245342
## VehBrandB14   -0.1610953  0.0979390  -1.645 0.100000
## VehGasRegular -0.1567649  0.0149231 -10.505  < 2e-16 ***
## DensityGLM     0.0395909  0.0158146   2.503 0.012299 *
## RegionR11     -0.0106148  0.0310973  -0.341 0.732846
## RegionR21     -0.0061355  0.1313714  -0.047 0.962749
## RegionR22      0.1713814  0.0641387   2.672 0.007539 **
## RegionR23     -0.0423486  0.0786584  -0.538 0.590311
## RegionR25     -0.0373635  0.0555638  -0.672 0.501301
## RegionR26      0.0455146  0.0612741   0.743 0.457601
## RegionR31      0.0169606  0.0405496   0.418 0.675751
## RegionR41     -0.1563557  0.0551307  -2.836 0.004567 **
## RegionR42      0.0238802  0.1167664   0.205 0.837953
## RegionR43     -0.1453884  0.1896170  -0.767 0.443232
## RegionR52      0.0263957  0.0320072   0.825 0.409554
## RegionR53      0.0213173  0.0295165   0.722 0.470161
## RegionR54      0.0370954  0.0426490   0.870 0.384418
## RegionR72      0.1078509  0.0372809   2.893 0.003817 **
## RegionR73     -0.1729571  0.0594376  -2.910 0.003616 **
## RegionR74      0.4129842  0.0795082   5.194 2.06e-07 ***
## RegionR82      0.2241953  0.0236667   9.473  < 2e-16 ***
## RegionR83      0.0146094  0.0938785   0.156 0.876332
## RegionR91     -0.0015807  0.0383337  -0.041 0.967109
## RegionR93      0.1443111  0.0266916   5.407 6.42e-08 ***
## RegionR94      0.1473866  0.0980039   1.504 0.132611
## AreaGLM        0.0465559  0.0212848   2.187 0.028722 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 136692  on 542330  degrees of freedom
## Residual deviance: 130707  on 542282  degrees of freedom
## AIC: 171380
```

```
##
## Number of Fisher Scoring iterations: 6
```

A detailed analysis of the output provides that all considered features are significant, except the area code AreaGLM. This can be seen from the p-value, which is above 5%, which corresponds to "not significant".

Exercise: Check what happens if the same conclusion on `AreaGLM` is reached, if you consider the area code as a categorical variable instead of a continuous one.

The `summary()` functions for a `glm` objects provides the statistical tests of significance for every single parameter. However, with categorical variables the primary interest is to know if a categorical variable is significant at all. This can be done using the R function `drop1`, see its help file for further details. It performs a Likelihood Ratio Test (LRT) which states that the p-value for `AreaGLM` is between 1% and 5%.

```r
# needs sufficient resources!
drop1(glm1, test = "LRT")
```

```
## Single term deletions
##
## Model:
## ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM + BonusMalusGLM +
##     VehBrand + VehGas + DensityGLM + Region + AreaGLM
##               Df Deviance    AIC    LRT Pr(>Chi)
## <none>             130707 171380
## VehPowerGLM    5   130794 171458   87.6  < 2e-16 ***
## VehAgeGLM      2   130830 171500  123.8  < 2e-16 ***
## DrivAgeGLM     6   131181 171843  474.6  < 2e-16 ***
## BonusMalusGLM  1   134219 174890 3512.1  < 2e-16 ***
## VehBrand      10   130916 171570  209.7  < 2e-16 ***
## VehGas         1   130817 171489  110.4  < 2e-16 ***
## DensityGLM     1   130713 171385    6.3  0.01225 *
## Region        21   130911 171543  204.8  < 2e-16 ***
## AreaGLM        1   130711 171383    4.8  0.02871 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Below we provide the sequential reduction in in-sample loss by adding one feature component after the other (ANOVA analysis). This also shows that the area code is not needed, after having already included all other feature components. From this we conclude that we may drop the area code which is not a surprise because of the strong collinearity with the feature component Density.

Note that the ANOVA analysis is sensitive in the order in which the feature components are considered. If we exchange the role of the area code and the density variable we obtain a similar result saying that the density variable may be dropped if the area code is already in the model.

```r
# needs sufficient resources!
anova(glm1)
```

```
## Analysis of Deviance Table
##
## Model: poisson, link: log
##
## Response: ClaimNb
##
## Terms added sequentially (first to last)
##
##
##               Df Deviance Resid. Df Resid. Dev
```

```
## NULL                              542330     136692
## VehPowerGLM    5      76.8     542325     136615
## VehAgeGLM      2      55.8     542323     136559
## DrivAgeGLM     6    1063.8     542317     135495
## BonusMalusGLM  1    3867.4     542316     131628
## VehBrand      10     214.8     542306     131413
## VehGas         1      61.7     542305     131351
## DensityGLM     1     434.0     542304     130917
## Region        21     205.9     542283     130711
## AreaGLM        1       4.8     542282     130707
```

Exercise: Extract the number of estimated coefficients from the glm object.

Validation

```
# Predictions
train$fitGLM1 <- fitted(glm1)
test$fitGLM1 <- predict(glm1, newdata = test, type = "response")
dat$fitGLM1 <- predict(glm1, newdata = dat2, type = "response")
```

```
# in-sample and out-of-sample losses (in 10^(-2))
sprintf("100 x Poisson deviance GLM (train): %s", PoissonDeviance(train$fitGLM1, train$ClaimNb))
```

```
## [1] "100 x Poisson deviance GLM (train): 24.1009071997707"
```

```
sprintf("100 x Poisson deviance GLM (test): %s", PoissonDeviance(test$fitGLM1, test$ClaimNb))
```

```
## [1] "100 x Poisson deviance GLM (test): 24.1774831766011"
```

```
# Overall estimated frequency
sprintf("average frequency (test): %s", round(sum(test$fitGLM1) / sum(test$Exposure), 4))
```

```
## [1] "average frequency (test): 0.0737"
```

```
df_cmp[2, ] <- list("GLM1", round(exec_time[[3]], 0), length(coef(glm1)), round(AIC(glm1), 0),
                    round(PoissonDeviance(train$fitGLM1, as.vector(unlist(train$ClaimNb))), 4),
                    round(PoissonDeviance(test$fitGLM1, as.vector(unlist(test$ClaimNb))), 4),
                    round(sum(test$fitGLM1) / sum(test$Exposure), 4))
knitr::kable(df_cmp)
```

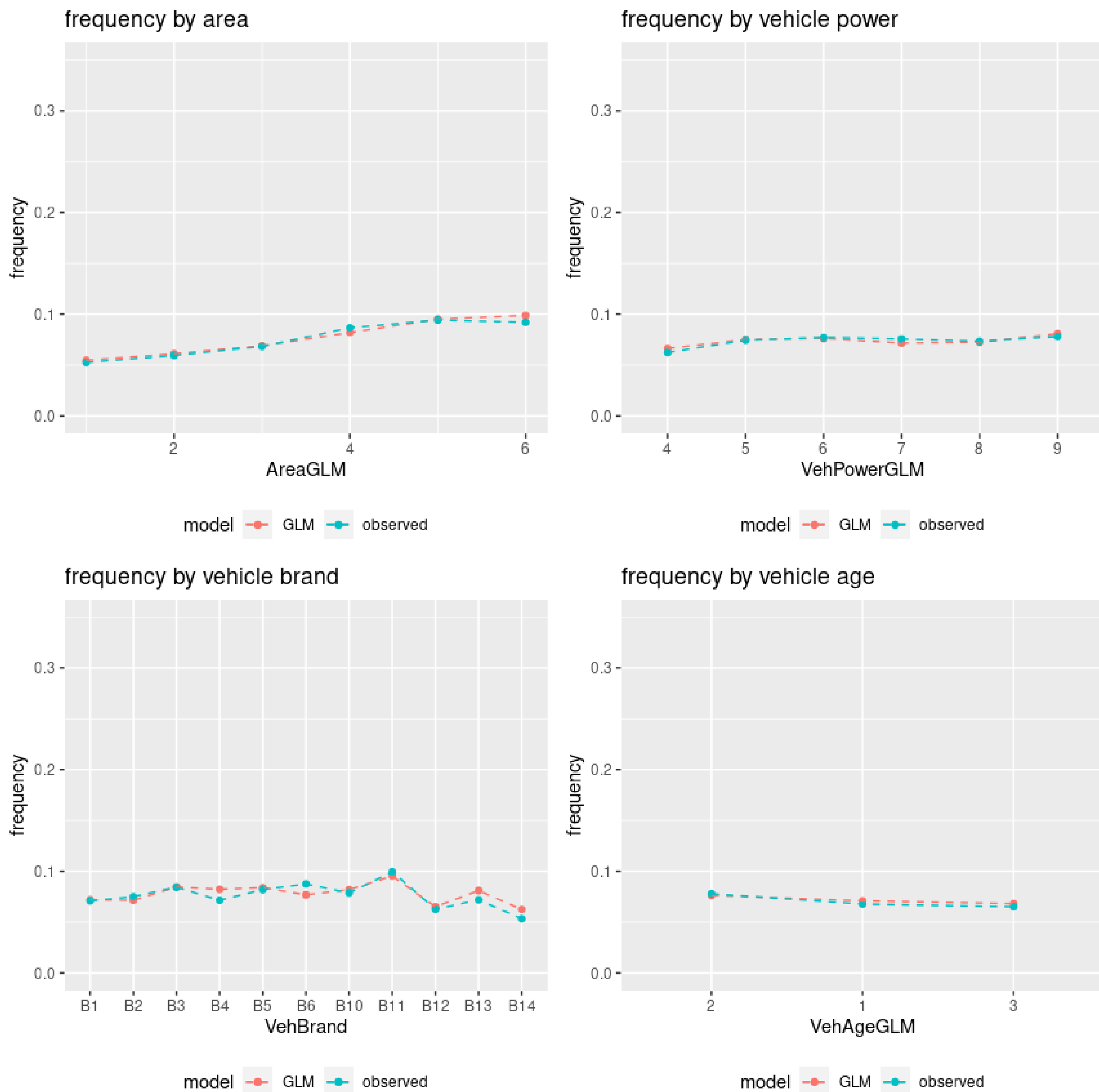
model	run_time	parameters	aic	in_sample_loss	out_sample_loss	avg_freq
GLM0	4	1	177269	25.2044	25.3483	0.0736
GLM1	50	49	171380	24.1009	24.1775	0.0737

Calibration

In addition to fitting and validating the model with a few metrics, it is important to check if the model is well calibrated across the feature space. E.g. it could be that the overall fit of a model is good, but that there are areas where the model under- and overestimates the claim frequencies. It is the goal of the subsequent calibration plots to ensure the proper fit along the whole feature space.

```
# Area
p1 <- plot_freq(test, "AreaGLM", "frequency by area", "GLM", "fitGLM1")

# VehPower
p2 <- plot_freq(test, "VehPowerGLM", "frequency by vehicle power", "GLM", "fitGLM1")
```

```r
# VehBrand
p3 <- plot_freq(test, "VehBrand", "frequency by vehicle brand", "GLM", "fitGLM1")

# VehAge
p4 <- plot_freq(test, "VehAgeGLM", "frequency by vehicle age", "GLM", "fitGLM1")

gridExtra::grid.arrange(p1, p2, p3, p4)
```



GLM2 (drop feature component Area compared to GLM1)

Fitting

```
exec_time <- system.time(
  glm2 <- glm(ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM + BonusMalusGLM +
                        VehBrand + VehGas + DensityGLM + Region,
              data = train, offset = log(Exposure), family = poisson()))
exec_time[1:5]
```

```
##  user.self   sys.self    elapsed user.child  sys.child
##     80.587     85.712     33.601      0.000      0.000
```

```
summary(glm2)
```

```
##
## Call:
## glm(formula = ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM +
##     BonusMalusGLM + VehBrand + VehGas + DensityGLM + Region,
##     family = poisson(), data = train, offset = log(Exposure))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5074  -0.3252  -0.2463  -0.1383   6.9300
##
## Coefficients:
##                 Estimate Std. Error  z value Pr(>|z|)
## (Intercept)   -4.5535601  0.0424451 -107.281  < 2e-16 ***
## VehPowerGLM5   0.0561277  0.0243478    2.305 0.021153 *
## VehPowerGLM6   0.0880346  0.0238853    3.686 0.000228 ***
## VehPowerGLM7   0.0653112  0.0237694    2.748 0.006002 **
## VehPowerGLM8   0.0984740  0.0338022    2.913 0.003577 **
## VehPowerGLM9   0.2381172  0.0265441    8.971  < 2e-16 ***
## VehAgeGLM1    -0.0197698  0.0341907   -0.578 0.563115
## VehAgeGLM3    -0.1800278  0.0163245  -11.028  < 2e-16 ***
## DrivAgeGLM1    0.1164603  0.0498130    2.338 0.019390 *
## DrivAgeGLM2   -0.3530009  0.0332110  -10.629  < 2e-16 ***
## DrivAgeGLM3   -0.4668602  0.0281276  -16.598  < 2e-16 ***
## DrivAgeGLM4   -0.2748028  0.0203789  -13.485  < 2e-16 ***
## DrivAgeGLM6   -0.0594679  0.0188631   -3.153 0.001618 **
## DrivAgeGLM7   -0.0610475  0.0313546   -1.947 0.051534 .
## BonusMalusGLM  0.0273619  0.0004106   66.639  < 2e-16 ***
## VehBrandB2    -0.0088691  0.0193282   -0.459 0.646328
## VehBrandB3     0.0591588  0.0266934    2.216 0.026676 *
## VehBrandB4     0.0553731  0.0362296    1.528 0.126415
## VehBrandB5     0.0849807  0.0308538    2.754 0.005882 **
## VehBrandB6     0.0126982  0.0349837    0.363 0.716623
## VehBrandB10    0.0083877  0.0443092    0.189 0.849857
## VehBrandB11    0.1847014  0.0468774    3.940 8.14e-05 ***
## VehBrandB12   -0.2528452  0.0244161  -10.356  < 2e-16 ***
## VehBrandB13    0.0577628  0.0499197    1.157 0.247225
## VehBrandB14   -0.1609823  0.0979392   -1.644 0.100239
## VehGasRegular -0.1564887  0.0149229  -10.486  < 2e-16 ***
## DensityGLM     0.0727947  0.0044694   16.287  < 2e-16 ***
## RegionR11     -0.0126572  0.0311012   -0.407 0.684031
```

```
## RegionR21      -0.0067609  0.1313725   -0.051 0.958956
## RegionR22       0.1771838  0.0640813    2.765 0.005693 **
## RegionR23      -0.0393615  0.0786468   -0.500 0.616734
## RegionR25      -0.0357374  0.0555567   -0.643 0.520055
## RegionR26       0.0483819  0.0612597    0.790 0.429655
## RegionR31       0.0204820  0.0405214    0.505 0.613236
## RegionR41      -0.1561137  0.0551337   -2.832 0.004632 **
## RegionR42       0.0301938  0.1167317    0.259 0.795898
## RegionR43      -0.1438132  0.1896156   -0.758 0.448184
## RegionR52       0.0284549  0.0319926    0.889 0.373776
## RegionR53       0.0243362  0.0294837    0.825 0.409137
## RegionR54       0.0372142  0.0426484    0.873 0.382892
## RegionR72       0.1097042  0.0372712    2.943 0.003246 **
## RegionR73      -0.1698906  0.0594203   -2.859 0.004248 **
## RegionR74       0.4144723  0.0795053    5.213 1.86e-07 ***
## RegionR82       0.2237981  0.0236836    9.449  < 2e-16 ***
## RegionR83       0.0162112  0.0938741    0.173 0.862895
## RegionR91       0.0027717  0.0382837    0.072 0.942285
## RegionR93       0.1495329  0.0265877    5.624 1.86e-08 ***
## RegionR94       0.1480863  0.0980039    1.511 0.130782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 136692  on 542330  degrees of freedom
## Residual deviance: 130711  on 542283  degrees of freedom
## AIC: 171383
##
## Number of Fisher Scoring iterations: 6
```

```r
# needs sufficient resources!
drop1(glm2, test = "LRT")
```

```
## Single term deletions
##
## Model:
## ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM + BonusMalusGLM +
##     VehBrand + VehGas + DensityGLM + Region
##               Df Deviance    AIC    LRT  Pr(>Chi)
## <none>              130711 171383
## VehPowerGLM    5   130799 171460   87.3 < 2.2e-16 ***
## VehAgeGLM      2   130835 171503  124.0 < 2.2e-16 ***
## DrivAgeGLM     6   131185 171845  473.7 < 2.2e-16 ***
## BonusMalusGLM  1   134221 174891 3509.7 < 2.2e-16 ***
## VehBrand      10   130921 171573  209.8 < 2.2e-16 ***
## VehGas         1   130821 171491  110.0 < 2.2e-16 ***
## DensityGLM     1   130979 171648  267.2 < 2.2e-16 ***
## Region        21   130917 171547  205.9 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# needs sufficient resources!
anova(glm2)
```

```
## Analysis of Deviance Table
```

```
##
## Model: poisson, link: log
##
## Response: ClaimNb
##
## Terms added sequentially (first to last)
##
##
##               Df Deviance Resid. Df Resid. Dev
## NULL                        542330     136692
## VehPowerGLM    5     76.8   542325     136615
## VehAgeGLM      2     55.8   542323     136559
## DrivAgeGLM     6   1063.8   542317     135495
## BonusMalusGLM  1   3867.4   542316     131628
## VehBrand      10    214.8   542306     131413
## VehGas         1     61.7   542305     131351
## DensityGLM     1    434.0   542304     130917
## Region        21    205.9   542283     130711
```

Validation

```r
# Predictions
train$fitGLM2 <- fitted(glm2)
test$fitGLM2 <- predict(glm2, newdata = test, type = "response")
dat$fitGLM2 <- predict(glm2, newdata = dat2, type = "response")
```

```r
# in-sample and out-of-sample losses (in 10^(-2))
sprintf("100 x Poisson deviance GLM (train): %s", PoissonDeviance(train$fitGLM2, train$ClaimNb))
```

```
## [1] "100 x Poisson deviance GLM (train): 24.1017895369265"
```

```r
sprintf("100 x Poisson deviance GLM (test): %s", PoissonDeviance(test$fitGLM2, test$ClaimNb))
```

```
## [1] "100 x Poisson deviance GLM (test): 24.1762018177717"
```

```r
# Overall estimated frequency
sprintf("average frequency (test): %s", round(sum(test$fitGLM2) / sum(test$Exposure), 4))
```

```
## [1] "average frequency (test): 0.0737"
```

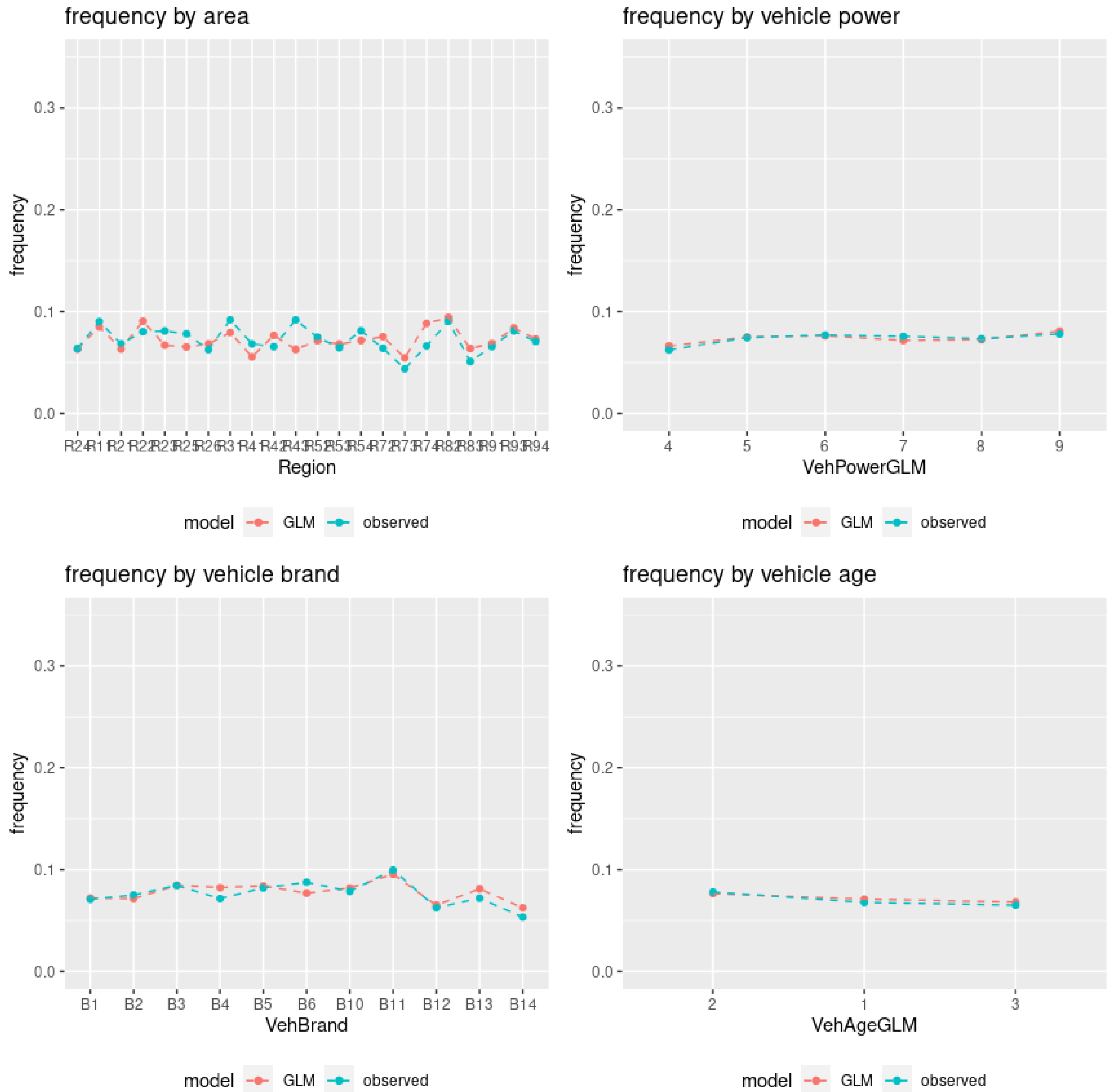
```r
df_cmp[3, ] <- list("GLM2", round(exec_time[[3]], 0), length(coef(glm2)), round(AIC(glm2), 0),
                   round(PoissonDeviance(train$fitGLM2, as.vector(unlist(train$ClaimNb))), 4),
                   round(PoissonDeviance(test$fitGLM2, as.vector(unlist(test$ClaimNb))), 4),
                   round(sum(test$fitGLM2) / sum(test$Exposure), 4))
knitr::kable(df_cmp)
```

model	run_time	parameters	aic	in_sample_loss	out_sample_loss	avg_freq
GLM0	4	1	177269	25.2044	25.3483	0.0736
GLM1	50	49	171380	24.1009	24.1775	0.0737
GLM2	34	48	171383	24.1018	24.1762	0.0737

Calibration

```r
# Area
p1 <- plot_freq(test, "Region", "frequency by area", "GLM", "fitGLM2")
```

```r
# VehPower
p2 <- plot_freq(test, "VehPowerGLM", "frequency by vehicle power", "GLM", "fitGLM2")

# VehBrand
p3 <- plot_freq(test, "VehBrand", "frequency by vehicle brand", "GLM", "fitGLM2")

# VehAge
p4 <- plot_freq(test, "VehAgeGLM", "frequency by vehicle age", "GLM", "fitGLM2")

gridExtra::grid.arrange(p1, p2, p3, p4)
```



Exercise: Perform the calibration with other variables not yet in the charts above.

GLM3 (drop feature components Area and VehBrand compared to GLM1)

Fitting

```
exec_time <- system.time(
  glm3 <- glm(ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM + BonusMalusGLM +
                        VehGas + DensityGLM + Region,
              data = train, offset = log(Exposure), family = poisson()))
exec_time[1:5]
```

```
##  user.self   sys.self    elapsed user.child  sys.child
##     48.556     49.161     20.996      0.000      0.000
```

```
summary(glm3)
```

```
##
## Call:
## glm(formula = ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM +
##     BonusMalusGLM + VehGas + DensityGLM + Region, family = poisson(),
##     data = train, offset = log(Exposure))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4935  -0.3251  -0.2500  -0.1399   6.9567
##
## Coefficients:
##                 Estimate Std. Error  z value Pr(>|z|)
## (Intercept)   -4.5857871  0.0406849 -112.715  < 2e-16 ***
## VehPowerGLM5   0.0967589  0.0240015    4.031 5.55e-05 ***
## VehPowerGLM6   0.1172939  0.0236246    4.965 6.87e-07 ***
## VehPowerGLM7   0.0899707  0.0235749    3.816 0.000135 ***
## VehPowerGLM8   0.0942006  0.0335874    2.805 0.005037 **
## VehPowerGLM9   0.2408108  0.0256134    9.402  < 2e-16 ***
## VehAgeGLM1    -0.0956365  0.0336968   -2.838 0.004538 **
## VehAgeGLM3    -0.1388828  0.0159342   -8.716  < 2e-16 ***
## DrivAgeGLM1    0.1528420  0.0497410    3.073 0.002121 **
## DrivAgeGLM2   -0.3326211  0.0331551  -10.032  < 2e-16 ***
## DrivAgeGLM3   -0.4528913  0.0281073  -16.113  < 2e-16 ***
## DrivAgeGLM4   -0.2685908  0.0203751  -13.182  < 2e-16 ***
## DrivAgeGLM6   -0.0637833  0.0188503   -3.384 0.000715 ***
## DrivAgeGLM7   -0.0663759  0.0312921   -2.121 0.033907 *
## BonusMalusGLM  0.0271880  0.0004108   66.179  < 2e-16 ***
## VehGasRegular -0.1641791  0.0147785  -11.109  < 2e-16 ***
## DensityGLM     0.0757161  0.0044557   16.993  < 2e-16 ***
## RegionR11     -0.1053719  0.0303973   -3.466 0.000527 ***
## RegionR21     -0.1227636  0.1310621   -0.937 0.348922
## RegionR22      0.1201210  0.0638956    1.880 0.060114 .
## RegionR23     -0.0770914  0.0785885   -0.981 0.326618
## RegionR25     -0.0505537  0.0555360   -0.910 0.362671
## RegionR26      0.0046770  0.0611666    0.076 0.939050
## RegionR31     -0.0284079  0.0403547   -0.704 0.481462
## RegionR41     -0.1822973  0.0550092   -3.314 0.000920 ***
## RegionR42     -0.0062145  0.1166647   -0.053 0.957519
## RegionR43     -0.2166781  0.1895342   -1.143 0.252950
## RegionR52      0.0118791  0.0319611    0.372 0.710135
```

```
## RegionR53      0.0268713  0.0294635    0.912 0.361758
## RegionR54      0.0259336  0.0426187    0.609 0.542854
## RegionR72      0.0677604  0.0370946    1.827 0.067747 .
## RegionR73     -0.2589025  0.0590628   -4.384 1.17e-05 ***
## RegionR74      0.3592016  0.0793837    4.525 6.04e-06 ***
## RegionR82      0.2008557  0.0236031    8.510  < 2e-16 ***
## RegionR83     -0.0773981  0.0936084   -0.827 0.408334
## RegionR91     -0.0514338  0.0380551   -1.352 0.176516
## RegionR93      0.0986185  0.0262685    3.754 0.000174 ***
## RegionR94      0.0245020  0.0975180    0.251 0.801616
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 136692  on 542330  degrees of freedom
## Residual deviance: 130921  on 542293  degrees of freedom
## AIC: 171573
##
## Number of Fisher Scoring iterations: 6
```

```r
# needs sufficient resources!
drop1(glm3, test = "LRT")
```

```
## Single term deletions
##
## Model:
## ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM + BonusMalusGLM +
##     VehGas + DensityGLM + Region
##               Df Deviance    AIC    LRT  Pr(>Chi)
## <none>              130921 171573
## VehPowerGLM    5   131012 171653   90.4 < 2.2e-16 ***
## VehAgeGLM      2   131002 171649   80.5 < 2.2e-16 ***
## DrivAgeGLM     6   131374 172013  452.5 < 2.2e-16 ***
## BonusMalusGLM  1   134389 175038 3467.6 < 2.2e-16 ***
## VehGas         1   131045 171694  123.4 < 2.2e-16 ***
## DensityGLM     1   131212 171862  290.9 < 2.2e-16 ***
## Region        21   131152 171762  231.0 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# needs sufficient resources!
anova(glm3)
```

```
## Analysis of Deviance Table
##
## Model: poisson, link: log
##
## Response: ClaimNb
##
## Terms added sequentially (first to last)
##
##
##               Df Deviance Resid. Df Resid. Dev
## NULL                         542330     136692
## VehPowerGLM    5     76.8    542325     136615
```

```
## VehAgeGLM      2     55.8    542323     136559
## DrivAgeGLM     6   1063.8    542317     135495
## BonusMalusGLM  1   3867.4    542316     131628
## VehGas         1     72.9    542315     131555
## DensityGLM     1    402.7    542314     131152
## Region        21    231.0    542293     130921
```

Validation

```r
# Predictions
train$fitGLM3 <- fitted(glm3)
test$fitGLM3 <- predict(glm3, newdata = test, type = "response")
dat$fitGLM3 <- predict(glm3, newdata = dat2, type = "response")
```

```r
# in-sample and out-of-sample losses (in 10^(-2))
sprintf("100 x Poisson deviance GLM (train): %s", PoissonDeviance(train$fitGLM3, train$ClaimNb))
```

```
## [1] "100 x Poisson deviance GLM (train): 24.140465191401"
```

```r
sprintf("100 x Poisson deviance GLM (test): %s", PoissonDeviance(test$fitGLM3, test$ClaimNb))
```

```
## [1] "100 x Poisson deviance GLM (test): 24.2213594809758"
```

```r
# Overall estimated frequency
sprintf("average frequency (test): %s", round(sum(test$fitGLM3) / sum(test$Exposure), 4))
```

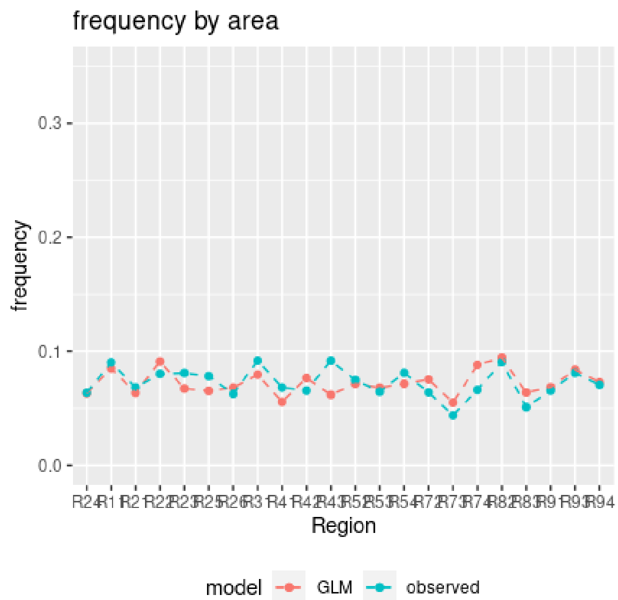
```
## [1] "average frequency (test): 0.0737"
```

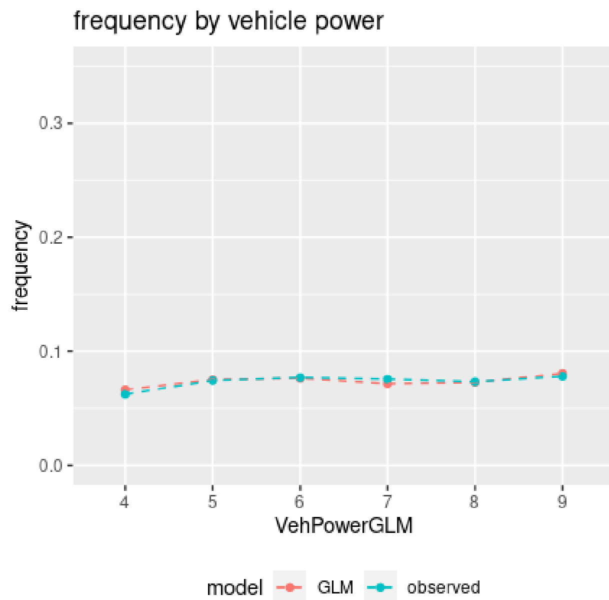
```r
df_cmp[4, ] <- list("GLM3", round(exec_time[[3]], 0), length(coef(glm3)), round(AIC(glm3), 0),
                    round(PoissonDeviance(train$fitGLM3, as.vector(unlist(train$ClaimNb))), 4),
                    round(PoissonDeviance(test$fitGLM3, as.vector(unlist(test$ClaimNb))), 4),
                    round(sum(test$fitGLM3) / sum(test$Exposure), 4))
knitr::kable(df_cmp)
```

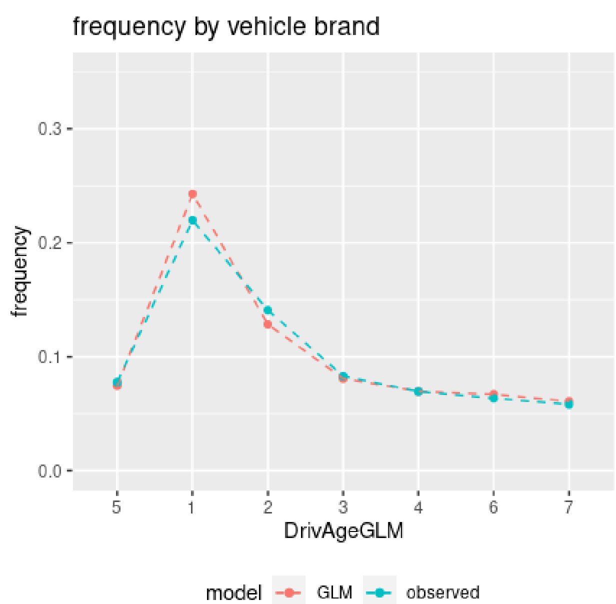
model	run_time	parameters	aic	in_sample_loss	out_sample_loss	avg_freq
GLM0	4	1	177269	25.2044	25.3483	0.0736
GLM1	50	49	171380	24.1009	24.1775	0.0737
GLM2	34	48	171383	24.1018	24.1762	0.0737
GLM3	21	38	171573	24.1405	24.2214	0.0737

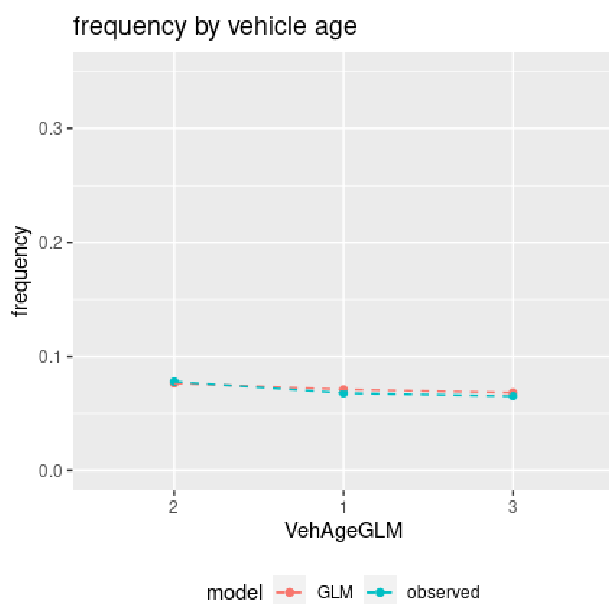
Calibration

```r
# Region
p1 <- plot_freq(test, "Region", "frequency by area", "GLM", "fitGLM3")

# VehPowerGLM
p2 <- plot_freq(test, "VehPowerGLM", "frequency by vehicle power", "GLM", "fitGLM3")

# DriveAgeGLM
p3 <- plot_freq(test, "DrivAgeGLM", "frequency by vehicle brand", "GLM", "fitGLM3")

# VehAgeGLM
p4 <- plot_freq(test, "VehAgeGLM", "frequency by vehicle age", "GLM", "fitGLM3")

gridExtra::grid.arrange(p1, p2, p3, p4)
```









Model Comparison

Comparing metrics

We have fitted three different models, as follows:

Model GLM1	all feature components considered as in Model Assumptions 3.2
Model GLM2	drop feature component `Area` compared to Model GLM1
Model GLM3	drop feature components `Area` and `VehBrand` compared to Model GLM1

We present the results of these three models below. These results are obtained by first fitting the three models to the learning data set $\mathcal{D}$, which provides the corresponding MLEs. These MLEs are then used to calculate the in-sample loss on $\mathcal{D}$. The fitted model is then applied to the testing data set $\mathcal{T}$, which provides the out-of-sample loss.

```
knitr::kable(df_cmp)
```

model	run_time	parameters	aic	in_sample_loss	out_sample_loss	avg_freq
GLM0	4	1	177269	25.2044	25.3483	0.0736
GLM1	50	49	171380	24.1009	24.1775	0.0737
GLM2	34	48	171383	24.1018	24.1762	0.0737
GLM3	21	38	171573	24.1405	24.2214	0.0737

We can draw the following conclusions:

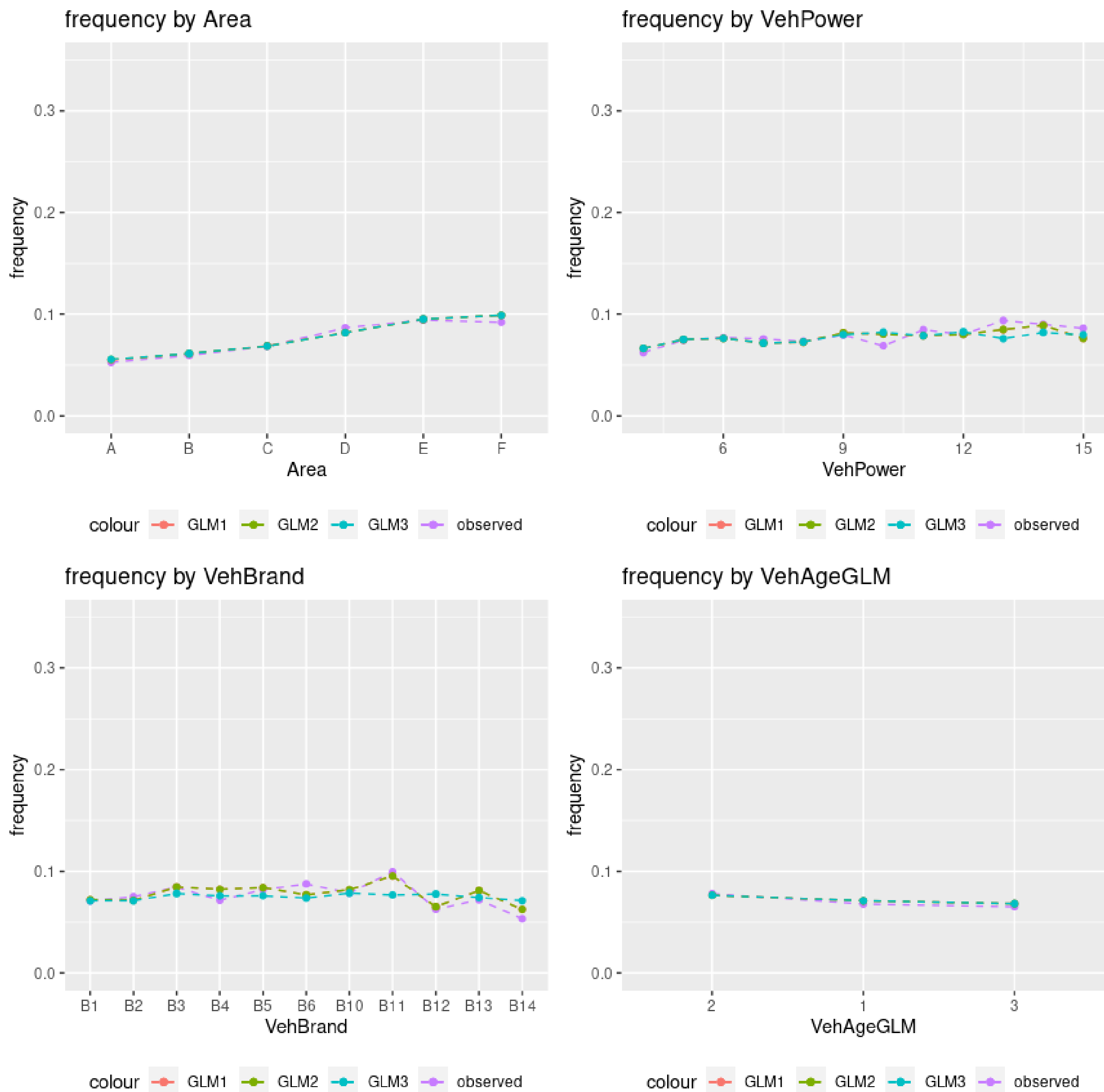
- The first observation from the table is that the in-sample loss is smaller than the out-of-sample loss. Of course, this is not surprising because we fit on the learning data, but if this difference is too big, this may either be a sign of over-fitting or a sign that learning and test data are rather different. If the in-sample loss is larger than the out-of-sample loss, it is an indication that there are some rows which belong together and are present in both datasets.
- As stated above, the split into train and test data is highly critical in practice and is an often encountered error.
- Considering Akaike's information criterion (AIC), which introduces a penalty term for over-fitting (to mimic an out-of-sample loss), the model with the smallest AIC value should be preferred. In our case, AIC (slightly) prefers Model GLM1. However, this model has a worse out-of-sample performance than Model GLM2. Thus, we do not get a clear (and good) advise from AIC and our out-of-sample analysis here, and for later purposes we will stick to Model GLM1 as benchmark model. Note that Model GLM3 is not competitive, and the component `VehBrand` is needed, in particular, for car brand `B12`.

Comparing predicted claim frequency by feature level

In this section, we are going to compare the predicted claim frequency split by features. This is similar to the calibration charts above and allows a visual comparison of the models.

```
plot_freq_2 <- function(xvar, title) {
  out <- test %>% group_by(!!sym(xvar)) %>% summarize(obs = sum(ClaimNb) / sum(Exposure),
                                                      glm1 = sum(fitGLM1) / sum(Exposure),
                                                      glm2 = sum(fitGLM2) / sum(Exposure),
                                                      glm3 = sum(fitGLM3) / sum(Exposure))

  ggplot(out, aes(x = !!sym(xvar), group = 1)) +
    geom_point(aes(y = obs, colour = "observed")) + geom_line(aes(y = obs, colour = "observed"), linetyp
    geom_point(aes(y = glm1, colour = "GLM1")) + geom_line(aes(y = glm1, colour = "GLM1"), linetype = "
    geom_point(aes(y = glm2, colour = "GLM2")) + geom_line(aes(y = glm2, colour = "GLM2"), linetype = "
    geom_point(aes(y = glm3, colour = "GLM3")) + geom_line(aes(y = glm3, colour = "GLM3"), linetype = "
    ylim(0, 0.35) + labs(x = xvar, y = "frequency", title = title) + theme(legend.position = "bottom")
}

# Area
p1 <- plot_freq_2("Area", "frequency by Area")

# VehPower
p2 <- plot_freq_2("VehPower", "frequency by VehPower")

# VehBrand
p3 <- plot_freq_2("VehBrand", "frequency by VehBrand")
```

```
# VehAgeGLM
p4 <- plot_freq_2("VehAgeGLM", "frequency by VehAgeGLM")

gridExtra::grid.arrange(p1, p2, p3, p4)
```



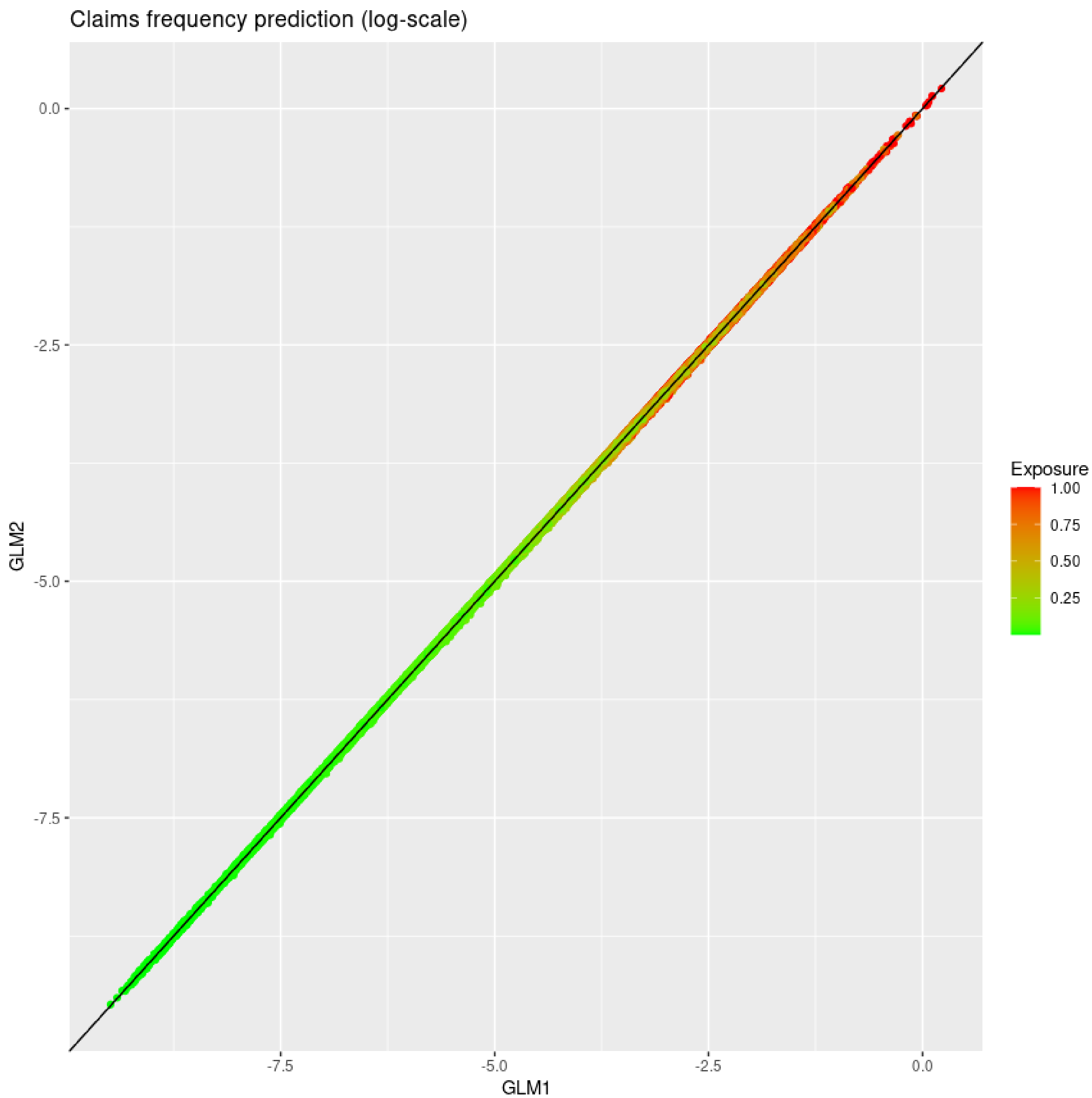
The charts show that the predictions for area are very close for all models. For vehicle power, the models are similar but they deviate from the observation.

We can conclude that all models provide very similar predictions, hence the best model selected above should be used.

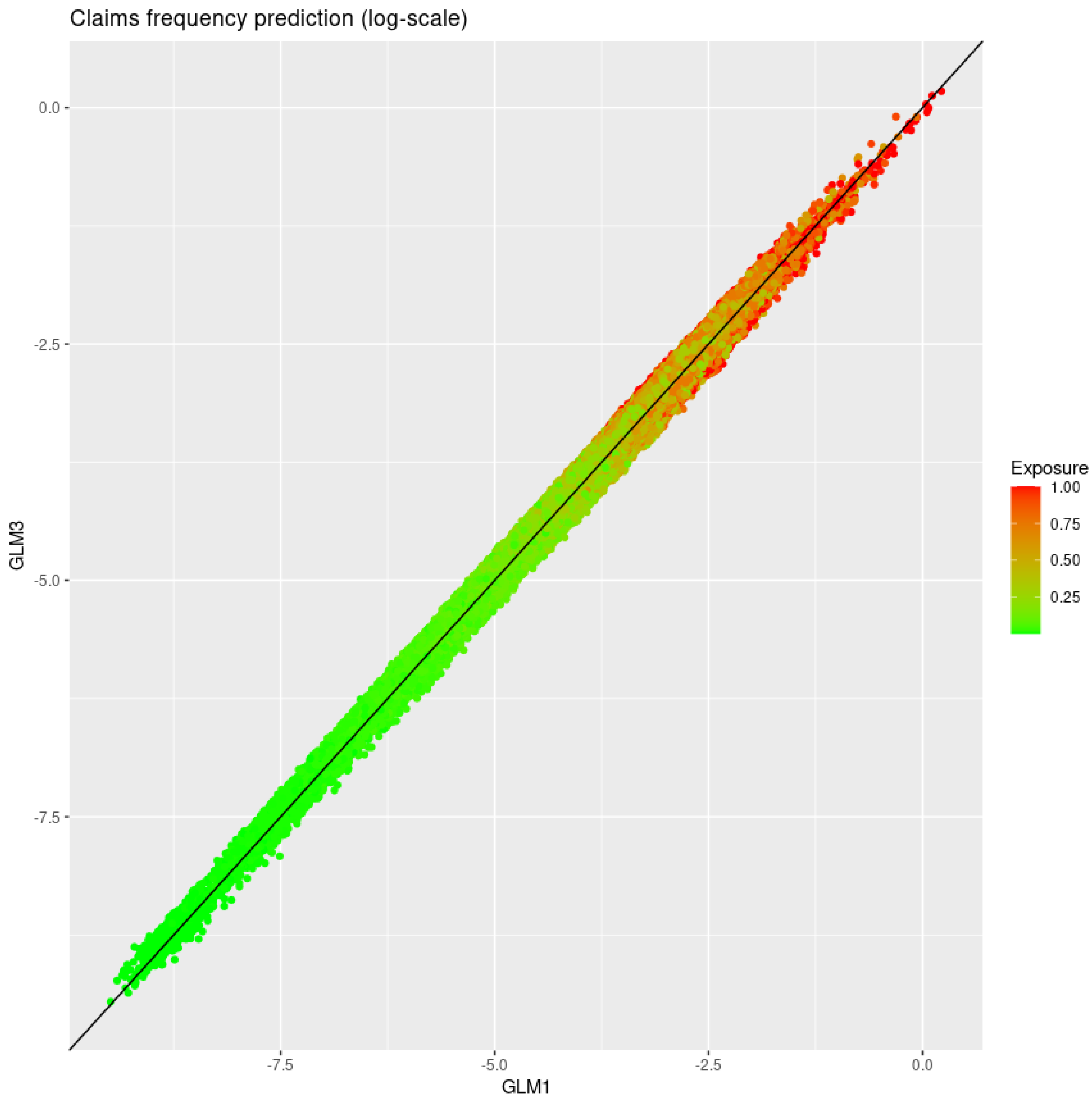
Comparing individual predicted claim frequency

Below we compare the out-of-sample claim frequency predictions (on log-scales) for two models. It allows to (maybe) identify if there are areas in the feature space where the predicted claim frequencies differ more/less than in other areas of the feature space.

```
axis_min <- log(max(test$fitGLM1, test$fitGLM2))
axis_max <- log(min(test$fitGLM1, test$fitGLM2))

ggplot(test, aes(x = log(fitGLM1), y = log(fitGLM2), colour = Exposure)) + geom_point() +
  geom_abline(colour = "#000000", slope = 1, intercept = 0) +
  xlim(axis_max, axis_min) + ylim(axis_max, axis_min) +
  labs(x = "GLM1", y = "GLM2", title = "Claims frequency prediction (log-scale)") +
  scale_colour_gradient(low = "green", high = "red")
```



```r
axis_min <- log(max(test$fitGLM1, test$fitGLM3))
axis_max <- log(min(test$fitGLM1, test$fitGLM3))

ggplot(test, aes(x = log(fitGLM1), y = log(fitGLM3), colour = Exposure)) + geom_point() +
  geom_abline(colour = "#000000", slope = 1, intercept = 0) +
  xlim(axis_max, axis_min) + ylim(axis_max, axis_min) +
  labs(x = "GLM1", y = "GLM3", title = "Claims frequency prediction (log-scale)") +
  scale_colour_gradient(low = "green", high = "red")
```



Session Info

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

```r
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       splitTools_0.3.1 ggplot2_3.3.5    tibble_3.1.4
## [5] dplyr_1.0.7
##
## 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     jsonlite_1.7.2    evaluate_0.14
##  [9] lifecycle_1.0.0   gtable_0.3.0      pkgconfig_2.0.3   rlang_0.4.11
## [13] rstudioapi_0.13   cli_2.5.0         DBI_1.1.1         yaml_2.2.1
## [17] xfun_0.23         gridExtra_2.3     withr_2.4.2       stringr_1.4.0
## [21] knitr_1.34        generics_0.1.0    vctrs_0.3.8       grid_4.0.5
## [25] tidyselect_1.1.1  glue_1.4.2        R6_2.5.0          fansi_0.4.2
## [29] rmarkdown_2.11    farver_2.1.0      purrr_0.3.4       magrittr_2.0.1
## [33] ps_1.6.0          scales_1.1.1      ellipsis_0.3.2    htmltools_0.5.1.1
## [37] assertthat_0.2.1  colorspace_2.0-1  labeling_0.4.2    utf8_1.2.1
## [41] stringi_1.6.1     munsell_0.5.0     crayon_1.4.1
```