CLAIM FREQUENCY ESTIMATION

Challenge 4

Neža Habjan, Eva Šraj

10. 6. 2021

Table of contents

1 Claim frequency modelling

In an insurance context, it is very important to be aware of all the factors that effect and change the estimated result, no matter which value you intend to predict. More is not always better, which is sometimes hard to implement. We have to take into consideration only the most probable features of the data, which would support our assumptions in the future. Therefore, when it comes to claim frequency modelling, which is the main topic of this challenge, it is crucial to investigate our data, to search for the most important causes of claims and maybe also correlations between them, some specific overall features and specialties of the dataset that should be included in our predictions. Additionally, if we want to build a good regression model, we have to discover the most important variables and relations between them.

We will firstly make an Ad hoc analysis of the factors included in the data and after that present you various models, which could be applied to it, their advantages and weaknesses. At the end, we will conclude with the selection of the best model and final prediction of the numbers of claims for each factor combination in the given data.

1.1 Descriptive statistics

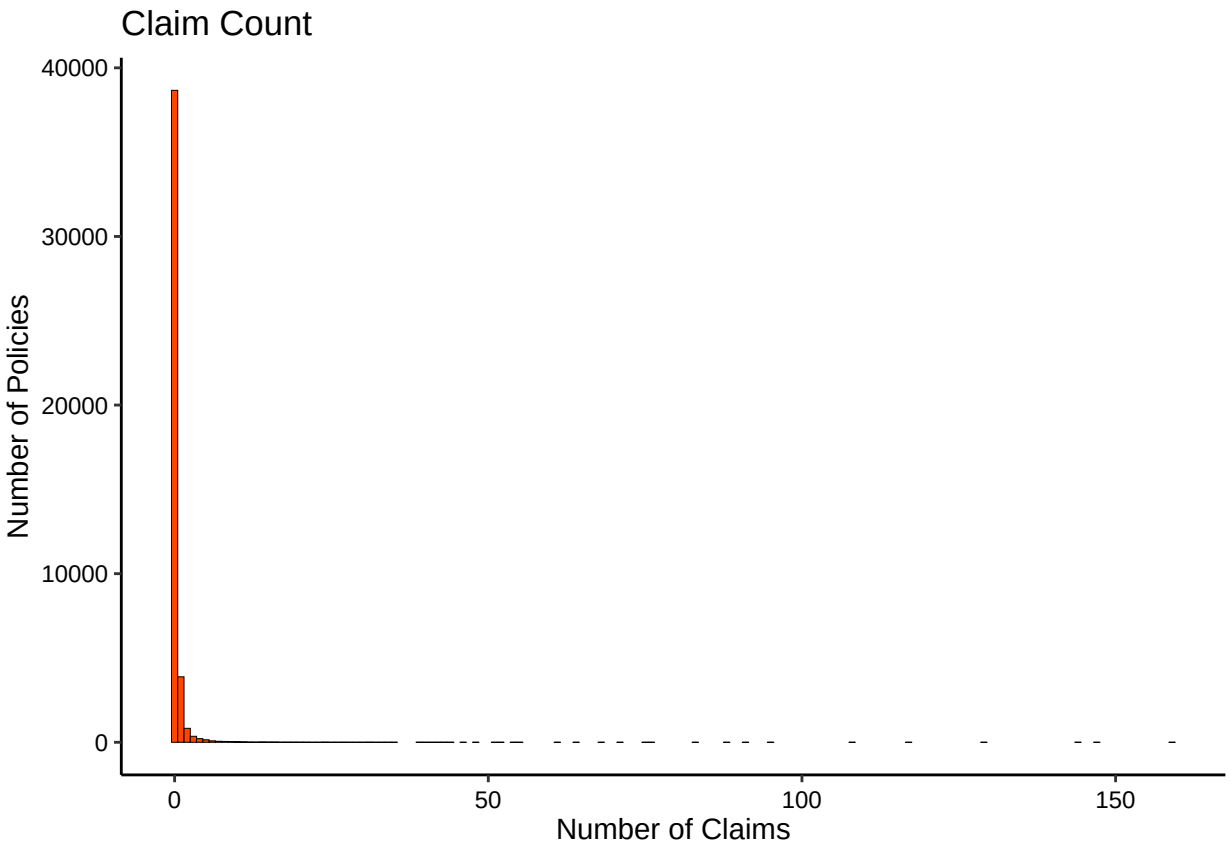
Our job is to investigate a sample of insurance policies (named `data1`), where for each policy we have numerous factors, which in combination yield a specific value of exposure and number of claims. The latter will serve us also as a benchmark when we fitting our model to the data. Below, we present you a short summary about number of policies and total exposure for each claim number included in the model.

```
kable(table_claims_exposure)
```

Number of Claims	Number of Policies	Total Exposure
0	38665	120480.56
1	3882	52934.04
2	829	33477.70
3	356	26063.91
4	216	22252.54
5	149	19110.28
6	86	15466.14
7	60	13681.31
8	51	12067.59
9	43	11146.69
10	31	9792.11
11	24	8995.43
12	14	4783.41
13	9	4222.32
14	18	6412.11
15	12	5004.90
16	12	6539.97
17	5	2861.43
18	6	4791.87
19	7	2967.78
20	5	3814.07
21	5	3348.22
22	2	1781.30
23	2	2056.87
24	8	9018.19
25	2	2132.05

Number of Claims	Number of Policies	Total Exposure
26	2	1552.66
27	3	2767.34
28	1	764.98
29	2	3614.65
30	2	2137.48
31	4	5369.51
32	1	1530.20
33	1	1672.52
34	1	1476.27
35	1	1816.31
39	3	5315.73
40	2	3659.01
41	2	3418.53
42	1	1913.76
43	3	6980.04
44	1	1621.46
46	2	4591.26
48	1	2183.85
51	1	1628.83
52	1	1147.35
54	1	1615.94
55	1	1978.73
61	1	2542.10
64	1	3040.76
68	1	2583.93
71	1	1968.18
75	1	4183.21
76	2	7681.36
83	1	2859.02
88	1	4164.37
91	2	7133.91
95	1	4047.68
108	1	3729.73
117	1	4866.28
129	1	4239.57
144	1	6585.43
147	1	5701.89
159	1	6820.80

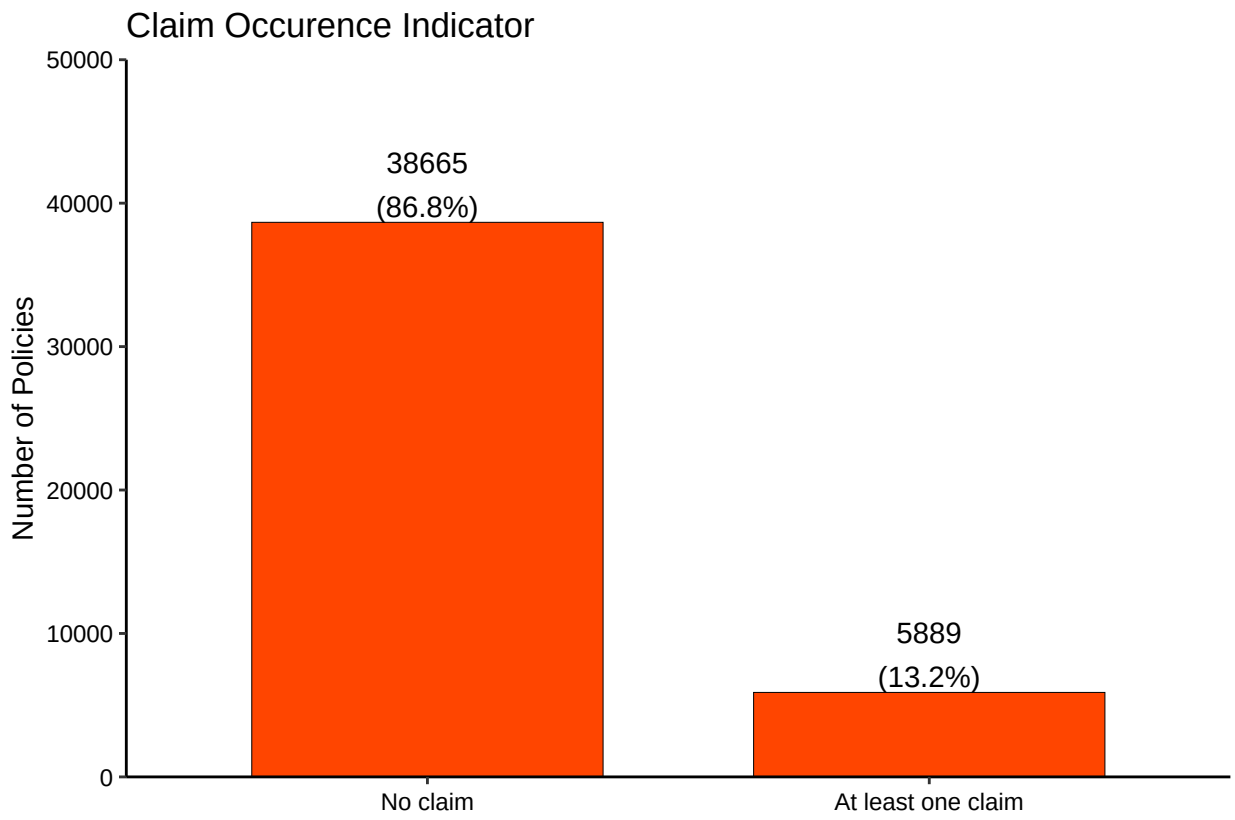
The graph represents the number of policies for every claim number. The difference between first and second column shows that there is significantly more policies without claims than with one or more claims.



1.1.1 Data drop

Soon we notice that there is significantly more policies with zero number of claims than of positive one and there are prevailing policies with small exposures. It is the main reason for our manipulation of the data described in the further analysis.

The following graph shows that the data consists of policies without claims with 86.8%, 13.2% of all policies have at least 1 claim, most of them have only 1 claim.



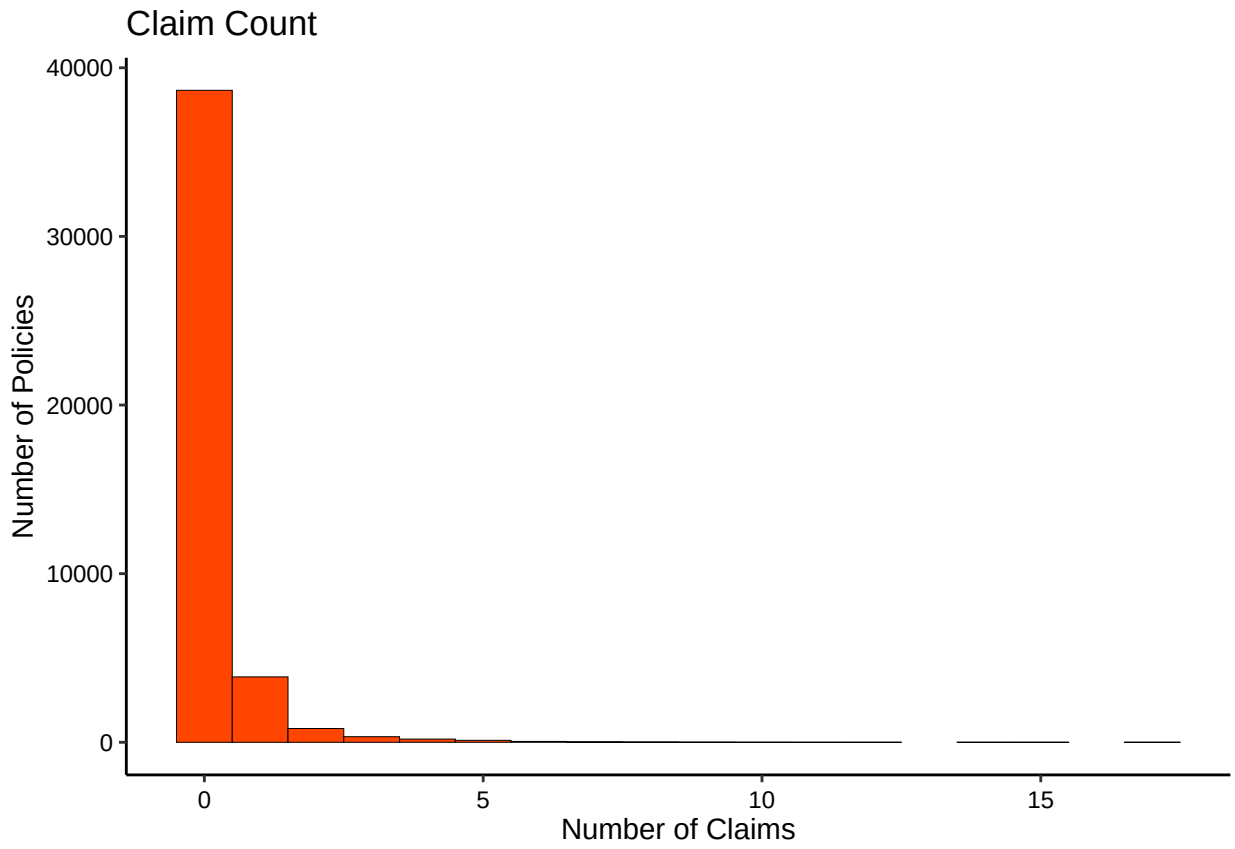
It is the main reason for our manipulation of the data described in the further analysis.

We have decided to drop the outliers, which are very high exposed policies, since we only have a few of them. The majority of the policies (seen on the table above) has a lower exposure and small number of claims. The inclusion of the most exposed policies could therefore only ruin our model and according to the overall small exposures, it is not really probable to expect in the future.

That is why we constructed `data_drop`, where we removed the top 1% of the data ranked from the lowest to the highest exposure. It is seen on the graph below, what the drop of the most exposed policies means for our data.

```
top_one_percent <- quantile(data1$EXPOSURE, .99)

# data_drop = data set without the most exposed policies
data_drop <- data1 %>% filter(EXPOSURE < top_one_percent)
```



1.1.2 Data recast

For each of 10 categorical variables in our data, we have prepared histograms of *annual mean claim frequency*. It is calculated as number of claims over exposure value. We decided to focus on this indicator instead of for example number of policyholders, because we want to inspect also the effect of exposure for each of the covariates. Two policies with similar number of policyholders and significantly different total exposures cannot be treated in the same way.

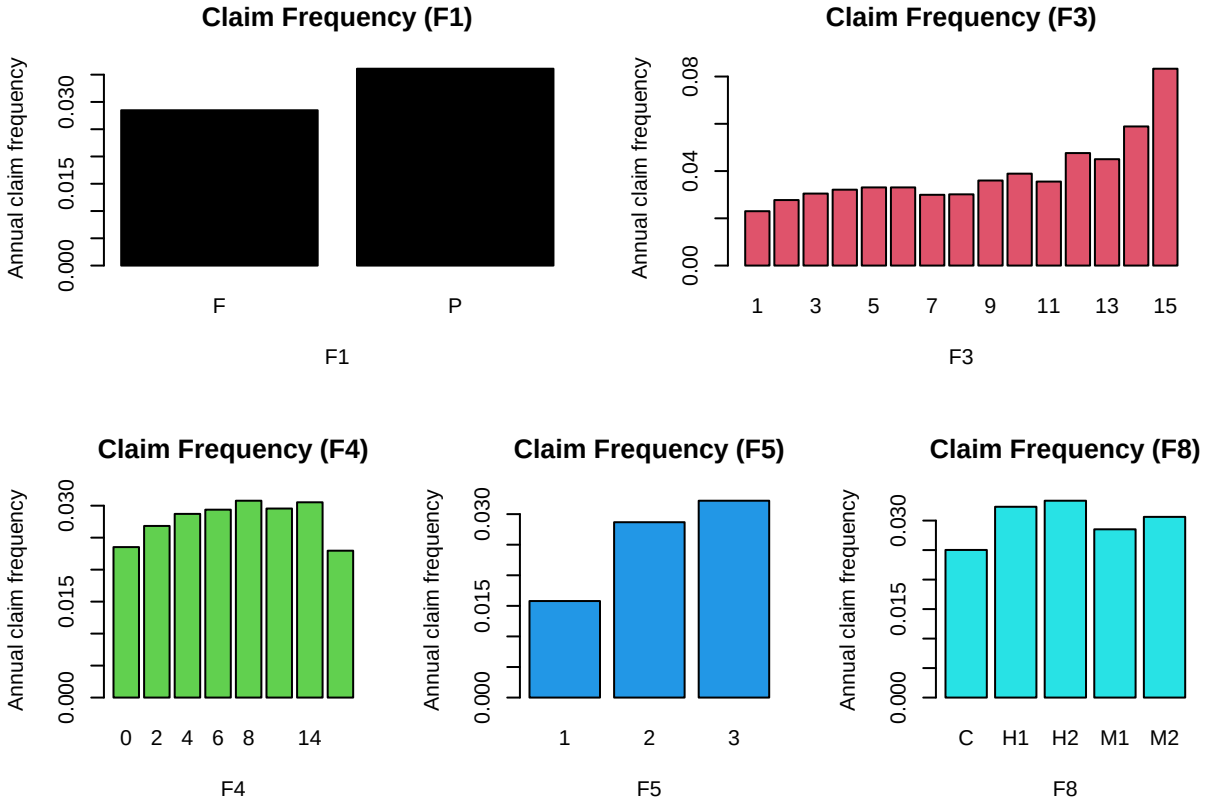
To get better accuracy of our data, we combined some variables' classes, as represented below each graph. We melt together classes with similar claim frequency indicator. Some of the factors were left unchanged, because we thought that each of their classes already include enough of data.

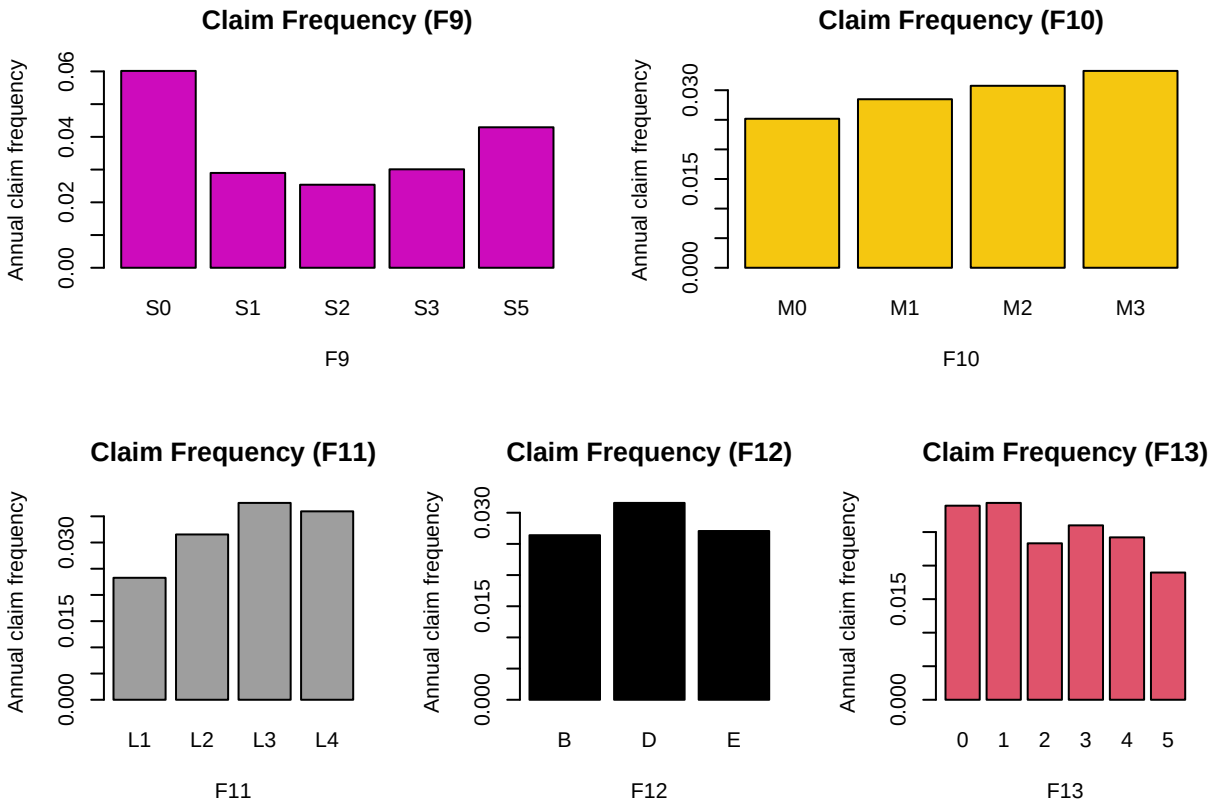
- **F3**:
  $G_0 = \{0\}$
  $G_1 = \{2, 3, 4, 5, 6, 7, 8\}$
  $G_2 = \{9, 10, 11\}$
  $G_3 = \{12, 13\}$
  $G_4 = \{14, 15\}$
- **F4**:
  $B_1 = \{0, 20\}$
  $B_2 = \{2, 4, 8, 10\}$
  $B_3 = \{8, 14\}$
- **F8**:
  $M = \{M_1, M_2\}$

$H = \{H_1, H_2\}$
$C = C$

- **F9**:
  $S_{1-3} = \{S_1, S_2, S_3\}$
  $S_0 = S_0$
  $S_5 = S_5$

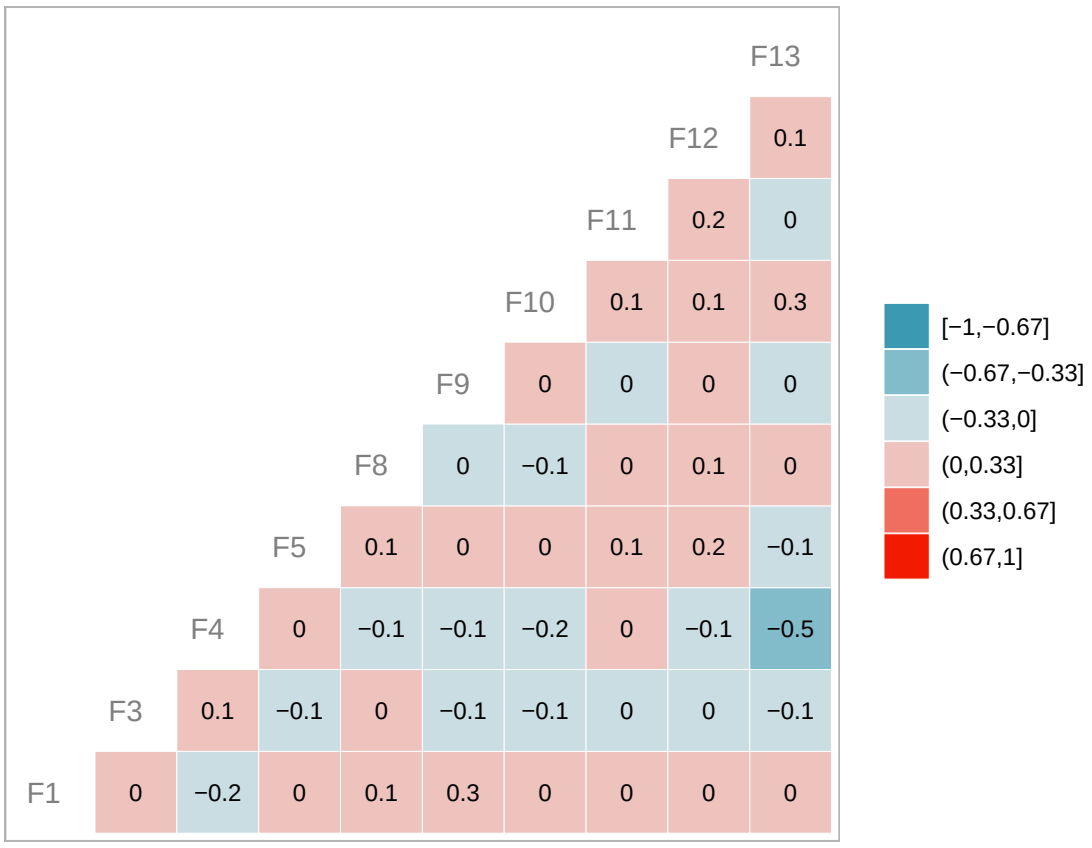
- **F13**:
  $F_{0-1} = \{0, 1\}$
  $F_{2-4} = \{2, 3, 4\}$
  $F_5 = \{5\}$





1.1.2.1 Independence of variables Along with the examination of variables we wanted to check whether they are independent or there might be some connections (correlations) between each two of them. In case of the latter, we should include the interaction in our model and therefore obtain dependent variables differently.

Below, there is a correlation matrix, that shows how strong each pair of factors is correlated. We see that besides the negative correlation between covariates F4 and F13 ($\text{corr}(F_4, F_{13}) = -0.5$), there is no other significant connections. This is something we will include and obtain more in details in further analysis.



1.1.3 Data relevel

Next step in our data filtering was to relevel our data. With that expression we mean that we extracted the most relevant classes of each variable and set it as the dominant one. We determined each dominant class by the claim count indicator, so the most frequent (already merged from recast step) class will be compared to all the others.

1.1.4 Creation of training and testing datasets

At the end, before we started to apply different models on our data, we created two datasets out of our starting data, via specific functions in R. One training dataset with 80% of our data that is the most relevant and fits our model the best is constructed via function `CreateDataPartition`. Other 20% is included in the testing dataset, and will serve as a test to our trained model.

```
data_partition1 <- createDataPartition(data_drop$CLAIMNO, times = 1,p = 0.8,list = FALSE)
training1 <- data_drop[data_partition1,] # 80% of data_drop
testing1  <- data_drop[-data_partition1,] # 20% of data_drop
```

2 Models applying

Generally speaking, when predicting values of a model that allows response variables with arbitrary distribution (other than simply normal), it is the best to use a Generalized Linear Model (GLM). Each of the model's outcome Y of the dependent variables is assumed to be generated from a particular distribution in an exponential family, a large class of probability distributions that includes the Normal, Binomial, Poisson and Gamma distribution. There are several types of GLM, but we will focus only on some of them.

2.1 Poisson GLM

$$
\begin{aligned}
CLAIMNO_i &\sim Poisson(\mu_i) \\
E(CLAIMNO_i) &= \mu_i \\
Var(CLAIMNO_i) &= \mu_i \\
\log(\mu_i) &= \beta_0 + \beta_1 \cdot F_1 + \beta_2 \cdot F_3 + \beta_3 \cdot F_4 + \beta_4 \cdot F_5 + \beta_5 \cdot F_8 + \beta_6 \cdot F_9 + \beta_7 \cdot F_{10} + \beta_8 \cdot F_{11} + \\
&+ \beta_9 \cdot F_{12} + \beta_{10} \cdot F_{13} \\
\mu_i &= e^{\beta_0+\beta_1 \cdot F_1+\beta_2 \cdot F_3+\beta_3 \cdot F_4+\beta_4 \cdot F_5+\beta_5 \cdot F_8+\beta_6 \cdot F_9+\beta_7 \cdot F_{10}+\beta_8 \cdot F_{11}+\beta_9 \cdot F_{12}+\beta_{10} \cdot F_{13}}
\end{aligned}
$$

First one is a Poisson regression model, since it is the best choice when modelling events, where the outcomes are counts. More specifically, count data i.e. discrete data with non-negative integer values that count something, like the number of times an event occurs during a given timeframe or the number of people in line at the grocery store. (*ref*: https://www.dataquest.io/blog/tutorial-poisson-regression-in-r/)

We are trying to predict numbers of claims, that is why we wanted first of all to check how do the assumptions of the Poisson model behave on our data.

Running R function `glm()` on the original data (`data1`) gives us the following outcome:

```
##
## Call:
## glm(formula = CLAIMNO ~ F1 + F3 + F4 + F5 + F8 + F9 + F10 + F11 +
##     F12 + F13, family = "poisson", data = data1, offset = log(EXPOSURE))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0809  -0.4159  -0.2865  -0.2122   4.7525
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.66963    0.20824 -22.425  < 2e-16 ***
## F1P          0.02911    0.05252   0.554 0.579436
## F32          0.20043    0.03248   6.171 6.78e-10 ***
## F33          0.29505    0.03517   8.390  < 2e-16 ***
## F34          0.34597    0.03723   9.293  < 2e-16 ***
## F35          0.38951    0.04239   9.189  < 2e-16 ***
## F36          0.40067    0.04593   8.724  < 2e-16 ***
## F37          0.31983    0.05030   6.358 2.04e-10 ***
## F38          0.32236    0.05097   6.324 2.55e-10 ***
## F39          0.48349    0.04750  10.180  < 2e-16 ***
## F310         0.54406    0.04476  12.154  < 2e-16 ***
## F311         0.44247    0.04029  10.981  < 2e-16 ***
## F312         0.67757    0.05259  12.885  < 2e-16 ***
```

```
## F313         0.66249    0.05411  12.242  < 2e-16 ***
## F314         0.92875    0.03400  27.313  < 2e-16 ***
## F315         1.22027    0.09191  13.277  < 2e-16 ***
## F42          0.03545    0.04579   0.774 0.438871
## F44          0.08079    0.04528   1.784 0.074390 .
## F46          0.13936    0.04434   3.143 0.001672 **
## F48          0.19932    0.04398   4.532 5.85e-06 ***
## F410         0.15389    0.04132   3.725 0.000196 ***
## F414         0.20490    0.04203   4.875 1.09e-06 ***
## F420        -0.02531    0.05456  -0.464 0.642766
## F52          0.65798    0.19701   3.340 0.000838 ***
## F53          0.69491    0.20129   3.452 0.000556 ***
## F8H1         0.04515    0.06143   0.735 0.462406
## F8H2         0.05783    0.08986   0.644 0.519838
## F8M1         0.05688    0.03358   1.694 0.090328 .
## F8M2         0.03790    0.04426   0.856 0.391885
## F9S1        -0.39682    0.04181  -9.492  < 2e-16 ***
## F9S2        -0.34078    0.04283  -7.956 1.78e-15 ***
## F9S3        -0.01758    0.04506  -0.390 0.696413
## F9S5        -0.13130    0.07167  -1.832 0.066946 .
## F10M1        0.08666    0.03359   2.580 0.009873 **
## F10M2        0.14196    0.04140   3.429 0.000605 ***
## F10M3        0.21339    0.06751   3.161 0.001574 **
## F11L2        0.26307    0.01936  13.588  < 2e-16 ***
## F11L3        0.41816    0.03732  11.206  < 2e-16 ***
## F11L4        0.35351    0.06648   5.318 1.05e-07 ***
## F12D         0.09411    0.01975   4.766 1.88e-06 ***
## F12E         0.10395    0.13992   0.743 0.457530
## F131        -0.01339    0.03701  -0.362 0.717519
## F132        -0.14658    0.06456  -2.271 0.023167 *
## F133        -0.04835    0.08666  -0.558 0.576903
## F134        -0.13830    0.12931  -1.070 0.284841
## F135        -0.43157    0.29151  -1.480 0.138740
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 21566  on 44553  degrees of freedom
## Residual deviance: 19529  on 44508  degrees of freedom
## AIC: 33801
##
## Number of Fisher Scoring iterations: 6
```

We get values of linear predictors, which show us the relation between the dominant class of the variable and all other classes. Additionally we see some p-values of the covariates (p-value $> 5\%$ means we should apply this covariate to the model, because it has a significant impact on the result), number of degrees of freedom, null and residual deviance for comparison to the nested model...

We wanted to see which of the steps in our data reconstruction turns out to be the most accurate in estimation of claim numbers. That is why we compared Poisson model applied to all different datasets. It is shown in the table below:

	POIS_base	POIS_drop	POIS_recast1	POIS_relevel	POIS_train1
RMSE	0.606	0.444	0.449	0.449	0.443
AIC	33801.158	31460.044	31610.708	31610.708	25095.426
p-value for Over-dispersion	0.005	0.006	0.002	0.002	0.013
alpha	0.024	0.024	0.032	0.032	0.023

When studying our results, we focused on the **Root-mean-square deviation (RMSE)** value, which is the »measure of how well a regression line fits the data points and is calculated as the square root of the mean of the square of all of the errors« (differences between original and fitted values). The smallest the RMSE, the better the model. (*ref*: https://www.geeksforgeeks.org/root-mean-square-error-in-r-programming/)

Its value is decreasing with almost every step of our data reconstruction. The most significant is the decrease in RMSE when removing the top 1 % of the most exposed policies. The recast of our data does not provide any improvements to the fit of the model, so we decided not to include this step in further analysis. It is the same with relevelling, which again has no impact on the RMSE indicator. But on the other hand, training set (named `training1`) constructed out of the `drop_data` table, fits the original data the best (the smallest RMSE value).

Along with the described indicator, also **Akaike information criterion (AIC)** coefficient leads us to the same conclusion. AIC indicator estimates the relative amount of information lost by a given model; the less information a model loses, the higher the quality of that model. (*ref*: https://en.wikipedia.org/wiki/Akaike_information_criterion)

One more thing we wanted to check is the previously mentioned interaction between variables F4 and F13.

The notation, which indicated the interaction included in the model is `F4 * F13`.

```
## glm(formula = CLAIMNO ~ F1 + F3 + F4 * F13 + F5 + F8 + F9 + F10 +
##     F11 + F12, family = poisson(link = "log"), data = training1,
##     offset = log(EXPOSURE))
```

	POIS_train1	INTER_POIS_train1
RMSE	0.443	0.442
AIC	25095.426	25108.916
p-value for Over-dispersion	0.013	0.019
alpha	0.023	0.022

In the last column of the table above, according to the RMSE indicator we see that the inclusion of interaction really improves our model. It will be also shown later via Likelihood-ratio test.

Another important thing when comparing Poisson models on different datasets is to *check the overdispersion of the data*. Since one of the main assumptions of the Poisson regression model is that the variance equals the expected value, we check it on our data.

```
# Test for dispersion
poisson_glm_train1DISPERSION <- dispersiontest(poisson_glm_train1,trafo=1)
poisson_glm_train1DISPERSION
```

```
##
##  Overdispersion test
##
## data:  poisson_glm_train1
```

```
## z = 2.236, p-value = 0.01268
## alternative hypothesis: true alpha is greater than 0
## sample estimates:
##      alpha
## 0.02313371
```

```
poisson_glm_train1ALPHA <- poisson_glm_train1DISPERSION$estimate
poisson_glm_train1ALPHA
```

```
##      alpha
## 0.02313371
```

It turns out that p-value $=0.0127 < 5\%$, which means that we reject the null hypothesis (variance = expected value, no presence of under-/over-dispersion or in other words $\alpha = 0$).

All of our Poisson regressions on different datasets give $\alpha > 0$, which means that an **over-dispersion is presented**. (https://medium.com/swlh/modeling-insurance-claim-frequency-a776f3bf41dc)

Therefore, we made an important conclusion that Poisson model is not the best choice for fitting our original data and forecasting the number of claims.

2.2 Negative Binomial with an offset

$$
\begin{aligned}
CLAIMNO_i &\sim NegBin(\mu_i, \theta) \\
E(CLAIMNO_i) &= \mu_i \\
Var(CLAIMNO_i) &= \frac{\mu_i + \mu_i^2}{\theta} \\
\log(\mu_i) &= \beta_0 + \beta_1 \cdot F_1 + \beta_2 \cdot F_3 + \beta_3 \cdot F_4 + \beta_4 \cdot F_5 + \beta_5 \cdot F_8 + \beta_6 \cdot F_9 + \beta_7 \cdot F_{10} + \beta_8 \cdot F_{11} + \\
&+ \beta_9 \cdot F_{12} + \beta_{10} \cdot F_{13} \\
\mu_i &= e^{\beta_0 + \beta_1 \cdot F_1 + \beta_2 \cdot F_3 + \beta_3 \cdot F_4 + \beta_4 \cdot F_5 + \beta_5 \cdot F_8 + \beta_6 \cdot F_9 + \beta_7 \cdot F_{10} + \beta_8 \cdot F_{11} + \beta_9 \cdot F_{12} + \beta_{10} \cdot F_{13}}
\end{aligned}
$$

Next model that we took into consideration is a Negative Binomial model with an offset. It is a two parametric model (mean and shape parameters) and is therefore very appropriate for modelling count variables, *especially for over-dispersed count outcome variables.* If the conditional distribution of the outcome variable is over-dispersed, the confidence intervals for the Negative Binomial regression are likely to be wider as compared to those from a Poisson regression model. (*ref*: https://stats.idre.ucla.edu/r/dae/negative-binomial-regression/)

Here is the summary of indicators for Negative Binomial model, when applying it to all of the steps in the data reconstruction.

```
kable(NEGBIN)
```

	NEGBIN_base	NEGBIN_drop	NEGBIN_recast1	NEGBIN_train1	ITER_NEGBIN_train1
RMSE	0.607	0.444	0.449	0.443	0.442
AIC	33801.272	31462.077	31610.719	25097.626	25111.142

As with the previous model, it is again seen that training dataset (`training1`) is the best choice when fitting data with Negative Binomial model. The RMSE coefficients in the last two columns are the smallest and again the interaction between F3 and F14 contributes to the better fit.

2.3 Zero-Inflated and Hurdle models

Many empirical count data sets exhibit more zero observations than would be allowed for by the Poisson model. We have already discussed this problem of excessive zero values in our dataset. It is an important feature of the data and we should check whether policies with zero number of claims should be treated and modelled differently than others. Useful solution for this are zero-inflated models, »which attempt to account for excess zeros. In other words, two kinds of zeros are thought to exist in the data, "true zeros" and "excess zeros".

Zero outcome is therefore by assumption due to two different processes.

For instance, in the insurance example presented here, the two processes are that a subject has suffered claim vs. not suffered any claim. If not, the only outcome possible is zero. But if policy holder has suffered a claim, it is then a count process. **Zero-Inflated models** represent a mixture between one GLM for the dichotomous outcome that a count Y is equal to zero and a conventional event-count GLM (Poisson or Negative Binomial regression).

On the other hand, when dealing with **Hurdle model**, we have to understand that there are two-component models included in it: A truncated count component, such as Poisson, Geometric or Negative Binomial, is employed for positive counts, and a hurdle component models zero counts. (*ref*: https://stats.idre.ucla.edu/r/dae/negative-binomial-regression/)

In the next section we will check how our data fits to both mentioned models. But here we have to be aware of the main implication stemming from these properties which is that it is necessary to compare the Zero-Inflated model with a simple count model using a test for non-nested models. The conventional Likelihood-ratio test, Wald test, or Lagrange multiplier test cannot be used.

2.3.1 Zero-Inflated Negative Binomial model

The results of Zero-Inflated Negative Binomial models by each dataset (`recast_data1, training1`) are represented below. It is seen that the interaction between F4 and F13 does not improve our model. The best fit of Zero-Inflated Negative Binomial model is achieved by dataset `training1`.

	ZI.negbin_recast1	ZI.negbin_train1	INTER_ZI.negbin_train1
RMSE	0.449	0.441	0.623
LOGLIK	-15759.719	-12410.970	-15343.742

2.3.2 Zero-Inflated Poisson model

Dataset `training1` is again the best for our model and also here the interaction between F4 and F13 does not improve it.

	ZI.pois_recast1	ZI.pois_train1	INTER_ZI.pois_train1
RMSE	0.449	0.441	0.634
LOGLIK	-15759.792	-12423.687	-15727.513

2.3.3 Hurdle Negative Binomial model

When dealing with Hurdle models we did not take into account the interaction. From the table below it is seen that the RMSE indicator is larger than in Zero-Inflated models. (*ref*: https://data.library.virginia.edu/getting-started-with-hurdle-models/)

	HU.negbin_recast1	HU.negbin_train1
RMSE	0.598	0.568
LOGLIK	-19042.230	-14852.311

2.3.4 Hurdle Poisson model

As for all models above we again concluded that `training1` dataset is the most appropriate, but still this model fits worser than Zero-Inflated POisson model.

	HU.pois_recast1	HU.pois_train1
RMSE	0.598	0.568
LOGLIK	-19042.230	-14852.311

3 Model selection

Above we have represented results of modelling our data by different models. According to results in each of the mentioned models, it is the best to use training dataset with included interaction between F4 and F13. But now, we would like to choose the most appropriate model that would fit the selected data the best.

3.1 Model comparison between Poisson and Negative Binomial model

With dispersion test for Poisson model we rejected a null hypothesis, which says there is no over-dispersion. Although it should be the most appropriate model for counting processes, we decided to choose Negative Binomial model with offset. AIC coefficient for both models are nevertheless really similar.

3.2 Model comparison between Zero-Inflated Negative Binomial and Hurdle Negative Binomial models

As seen before (RMSE values and AIC coefficients) Hurdle models give for our datasets worser results. That is why we focused on Zero-Inflated Negative Binomial model and compared it to Negative Binomial model.

3.2.1 Zero counts and Vuong test

Because of previuosly mentioned excessive number of zero counts, we wanted to check whether special treatment of the data without claims really do contribute significantly to the fit. Firstly, we collected numbers of policies with zero claims that are predicted via each model. It is represented in a short report below. Here column `delta` notes the difference between number of zero claims in dataset `training1` (row `observations`) and the same value estimated by each of the models.

	Number of Zero Claims	delta
observations	30941	0
Poisson_glm	30959	18
NegBin_glm	30960	19
ZeroInflatedPois	30980	39
ZeroInflatedNegBin	30977	36
HurdlePoisson	30918	-23
HurdleNegPoisson	30918	-23

Since none of the models predicts a significantly lower or higher number of policies without claims, we can not determine which is the best. After all, out of more than 44500 policies, there is not much difference between values 18 or 40. Therefore, next step is to do a more relevant comparison with **Vuong test**.

3.2.1.1 Vuong test The Vuong test is designed to compare two models fit to the same data by maximum likelihood. Specifically, it tests the null hypothesis that the two models fit the data equally well. The models need not be nested, nor does one of the models need to represent the correct specification. (*ref:* https://journals.sagepub.com/doi/pdf/10.1177/1536867X1301300408)

Below, there is a short summary of comparison all previously mentioned combinations of Zero-Inflated and Hurdle models with the original models that do not account for excessive zero counts.

```
vuong(negbin_glm_train1, ZInegbin_train1)
```

```
## Vuong Non-Nested Hypothesis Test-Statistic:
## (test-statistic is asymptotically distributed N(0,1) under the
##  null that the models are indistinguishible)
## -------------------------------------------------------------
##                Vuong z-statistic             H_A    p-value
## Raw                  -12.802170 model2 > model1 < 2.22e-16
## AIC-corrected         -6.319571 model2 > model1 1.3114e-10
## BIC-corrected         21.138348 model1 > model2 < 2.22e-16
```

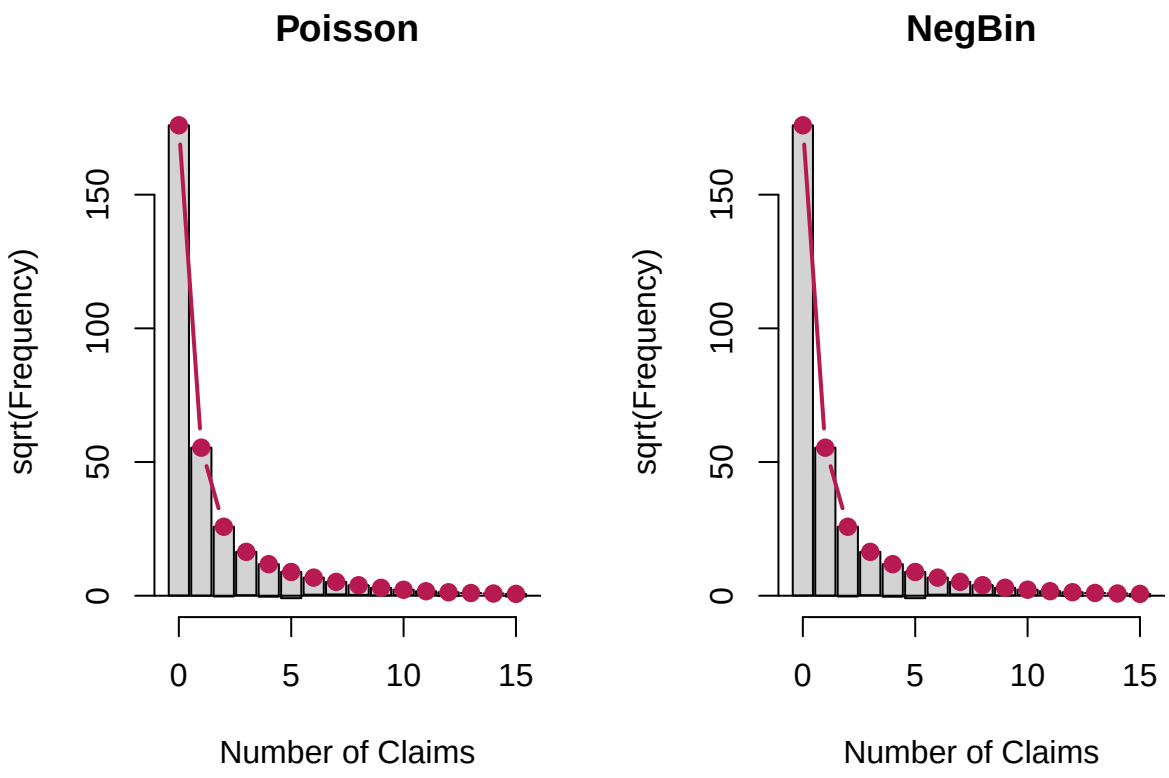
The p-value$< 2.22 \cdot 10^{-16}$, which is much smaller than 5%. This confirms that Zero-Inflated Negative Binomial model predicts better fit for our data.

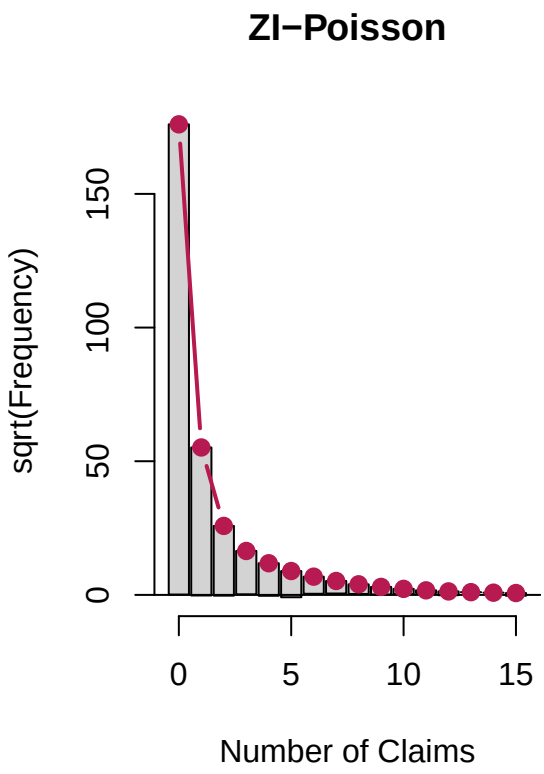
3.3 Graphical representation of comparison between models

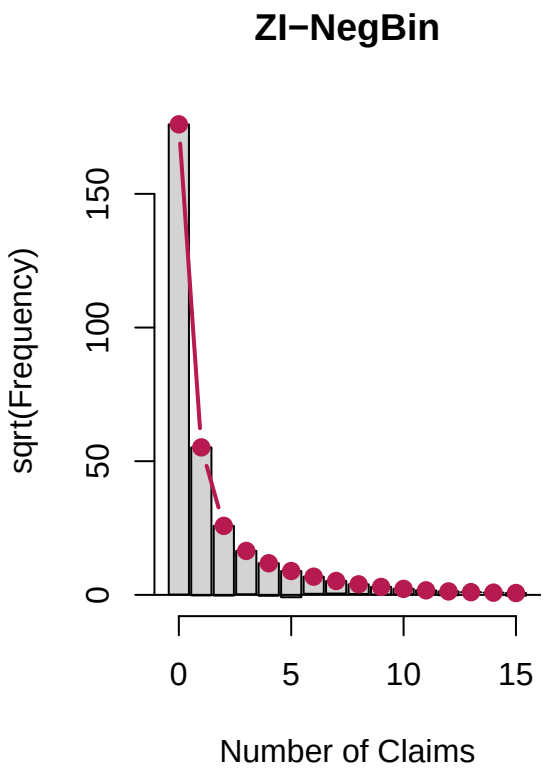
To represent the adequacy of our chosen model, we decided to draw some rootograms. It is a graphical data analysis technique for summarizing the distributional information of a variable. It consists of:

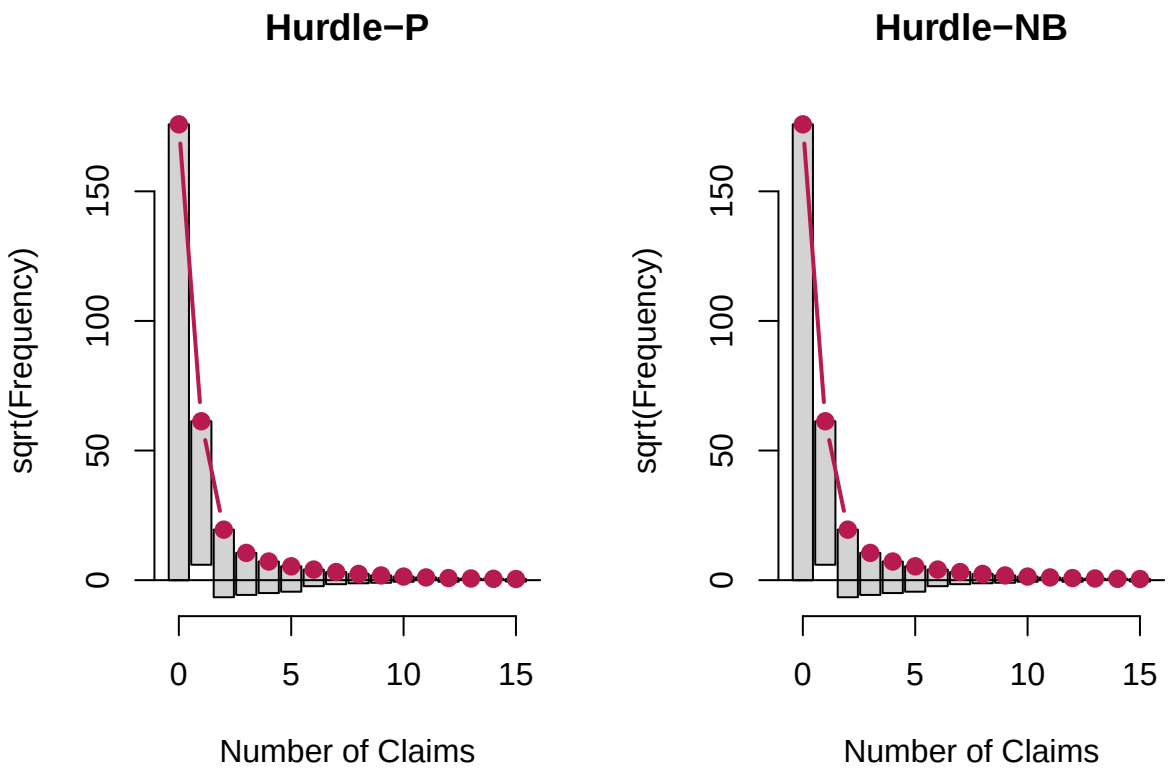
- vertical axis = square root of frequencies or relative frequencies;
- horizontal axis = response variable.

Rootograms below are restricted up to 15 claims on the horizontal axis, because after choosing the most appropriate dataset `training1` (claim number in `training1`=17), our highest frequency of claims was 17.




ZI−Poisson
sqrt(Frequency)
0
50
100
150
0
5
10
15
Number of Claims


ZI−NegBin
sqrt(Frequency)
0
50
100
150
0
5
10
15
Number of Claims



It is seen clearly that the rootogram of Zero-Inflated Negative Binomial model fits our data the best. There is a small deviation on the number of claims 5, but in comparison to others, where the differences appear already at 3 claims (Zero-Inflated Poisson) it is much better. At higher values (at 6 and 7 claims), there is a little difference at Poisson and Negative Binomial model, but when we check Hurdle models, it is seen from the pictures that they both fit very bad.

4 Conclusion

After checking all of the above described models and making different comparisons between them, we decided to choose Zero-Inflated Negative Binomial model. First step was to reject Poisson model due to overdispersion, Negative Binomial model fits our data better.

Because of excess number of zeros in our original data, we decided to select a model that could handle that feature. After applying Hurdle models on our data, we rejected them due to bad fitting parameters, but at the end Zero-Inflated Negative Binomial model was the most appropriate one. Additionally also better than normal Negative Binomial model. That confirms our decision to model excess zeros separately. We applied fitted parameters from original data on the new dataset `dataT` and wrote the results to csv.

Thus for prediction of numbers of claims in given dataset `dataT`, we used function `predict` on Zero-Inflated Negative Binomial fit with dataset `training1`.

```
predict_ZInegbin_train1 <- predict(ZInegbin_train1, dataT, type = "response")
RESULT <- cbind(dataT, "Predicted_number_of_claims" = round(predict_ZInegbin_train1,0))
```

The output of our challenge is written in `RESULT.csv` via R function `write.csv`.