ACTL4305/5305 Actuarial Data Analytic Application

Week 5: Proposed Solutions

GLM

Learning Objectives

- Perform data analytics modelling using GLM for response variables of aggregate claims, and claims frequency and severity.
- Understand the specifications of the GLM and the model assumptions. Know how to set offsets and adjust weights when building GLMs.
- Select and validate a GLM appropriately. Try different distributions and compare their fitting and predictive performance.

1 Frequency-Severity Analysis

Many insurance data sets feature information about how often claims arise, the frequency, in addition to the claim size, the severity. Observable responses can include:

- N, the the number of claims (Claim Count),
- $y_k, k = 1, ..., N$, the amount of each claim (Loss), and
- $S = y_1 + ... + y_N$, the aggregate claim amount (Aggregate Loss).
- E, for exposure.

By convention, the set $\{y_j\}$ is empty when $N = 0$.

For modeling the joint outcome (N, S) (or equivalently,$(N, \bar{S})$), it is customary to first condition on the frequency and then modeling the severity. Suppressing the $\{i\}$ subscript, we decompose the distribution (per unit Exposure) of the dependent variables as:

$$\begin{aligned} f(N,S) &= f(N) \times f(S|N) \\ \text{joint} &= \text{frequency} \times \text{conditional severity}, \end{aligned} \tag{1}$$

where $f(N, S)$ denotes the joint distribution of (N, S).

Here is a summary:

- Frequency = Claim Count / Exposure
- Severity = Aggregate Loss / Claim Count
- Pure Premium = Aggregate Loss / Exposure = Frequency $\times$ Severity

2 Data Import and Preparation

In this week's lab, we use the same French insurance datasets, `freMTPL2freq` and `freMTPL2sev`, as used in the earlier weeks. Following Wüthrich and Merz (2023), the code below performs some data cleaning and merges `freMTPL2freq` with the aggregated severities for each insurance policy in `freMTPL2sev`. It is interesting to note that this process does not retain the claim count information from `freMTPL2freq`; instead, it extracts this information from `freMTPL2sev`.

```
#Load the required packages
library(CASdatasets)
library(tidyverse)
library(MASS) #stepAIC and glm.nb()
library(statmod)
library(MLmetrics)
```

```
# Load datasets
data(freMTPL2freq)
data(freMTPL2sev)

#Data pre-processing
freq_data <- freMTPL2freq[, -2] # Remove the column of ClaimNb from the frequency data
freq_data$VehGas <- factor(freq_data$VehGas)
sev_data <- freMTPL2sev
sev_data$ClaimNb <- 1
agg_sev <- aggregate(sev_data, by=list(IDpol=sev_data$IDpol), FUN = sum)[c(1,3:4)]
names(agg_sev)[2] <- "ClaimTotal"
merged_data <- merge(x=freq_data, y=agg_sev, by="IDpol", all.x=TRUE)
merged_data[is.na(merged_data)] <- 0
merged_data <- merged_data[merged_data$ClaimNb <= 5, ]
merged_data$Exposure <- pmin(merged_data$Exposure, 1)

freMTPL2full <- merged_data
```

2.1 Task Solution: As pointed out in Wüthrich and Merz (2023), "the claim counts on the insurance policies with policy IDs $\leq$ 24500 in `freMTPL2freq` do not seem to be correct because these claims do not have claim severity counterparts in `freMTPL2sev`. For this reason, we work with the claim counts extracted from the latter file." Could you verify if this claim is possibly true? Also, what data cleaning steps are done in the code above?

To verify whether the claim about policy IDs $\leq$ 24500 is possibly true, you can check if there are policies with claim counts in `freMTPL2freq` that do not have corresponding entries in `freMTPL2sev`. If a significant number of such policies exist, this would support the assertion made by Wüthrich and Merz (2023). This can be done by comparing the policy IDs in both datasets and checking for discrepancies, as shown in the code below.

```
# Count duplicates in freMTPL2sev
num_duplicate_ids_sev <- freMTPL2sev %>%
  count(IDpol) %>%
  filter(n > 1) %>%
  nrow()

cat("Number of unique duplicated IDpol in freMTPL2sev:", num_duplicate_ids_sev, "\n")
```

```
## Number of unique duplicated IDpol in freMTPL2sev: 1379
```

```
# Count duplicates in freMTPL2freq
num_duplicate_ids_freq <- freMTPL2freq %>%
  count(IDpol) %>%
  filter(n > 1) %>%
  nrow()

cat("Number of unique duplicated IDpol in freMTPL2freq:", num_duplicate_ids_freq, "\n")
```

```
## Number of unique duplicated IDpol in freMTPL2freq: 0
```

```
# IDs with ClaimNb > 0 in freMTPL2freq
ids_claim_nb_gt_zero <- freMTPL2freq %>%
  filter(ClaimNb > 0) %>%
  pull(IDpol) %>%
  unique()

# Unique IDs in freMTPL2sev
unique_ids_sev <- freMTPL2sev %>%
  pull(IDpol) %>%
  unique()

# IDs not in freMTPL2sev
ids_not_in_sev <- setdiff(ids_claim_nb_gt_zero, unique_ids_sev)
num_ids_not_in_sev <- length(ids_not_in_sev)

cat("Number of unique IDs with ClaimNb > 0 in freMTPL2freq but not in freMTPL2sev:",
    num_ids_not_in_sev, "\n")
```

```
## Number of unique IDs with ClaimNb > 0 in freMTPL2freq but not in freMTPL2sev: 9116
```

From the above results, we can infer that each row in `freMTPL2freq` represents a unique policy, while each row in `freMTPL2sev` represents an individual claim. Therefore, we need to aggregate the claim amount information from `freMTPL2sev`. There are 9116 unique IDs with `ClaimNb > 0` in `freMTPL2freq` but not in `freMTPL2sev`, which would strongly support the assertion made by Wüthrich and Merz (2023).

Recalling from Week 2's lab, most of the following data cleaning steps were discussed in our EDA:

- `freq_data$VehGas <- factor(freq_data$VehGas)`: This line converts the `VehGas` column in the `freq_data` data frame from a character type to a factor.
- `merged_data[is.na(merged_data)] <- 0`: This replaces any `NA` values in the `merged_data` data frame with 0. This is done to handle missing data, treating them as zero claims or zero claim amounts during data aggregation.
- `merged_data <- merged_data[merged_data$ClaimNb <= 5, ]`: This line filters the `merged_data` data frame to include only policies with five or fewer claims. Policies with more than five claims are excluded from the data frame.
- `merged_data$Exposure <- pmin(merged_data$Exposure, 1)`: The `Exposure` column in the `merged_data` data frame is modified such that any value greater than 1 is capped at 1.

2.2 Data Splitting

We split the dataset `freMTPL2full` into training and test sets, with 70% allocated to the training set. The selected indices are stored in `train_indices`, which are used to subset the dataset into training and test sets, as shown in the commented-out lines where `train_data` and `test_data` are created. Note that in this week's lab, we primarily use `train_indices` to select the training or test sets as needed.

```
#Split the Data
set.seed(903)
train_indices <- sample(1:nrow(freMTPL2full), size = 0.7 * nrow(freMTPL2full))

#train_data <- freMTPL2full[train_indices, ]
#test_data <- freMTPL2full[-train_indices, ]
```

3 Modeling Claim Frequency Using GLMs

It has become routine for actuarial analysts to model the frequency based on covariates x_i using generalized linear models, GLMs. For count outcomes, one begins with a **Poisson** or **negative binomial** distribution. Moreover, to handle the excessive number of zeros relative to that implied by these distributions, analysts routinely examine zero-inflated.

A strength of GLMs relative to other non-linear models is that one can express the mean as a simple function of linear combinations of the covariates. In insurance, it is common to use a "logarithmic link" for this function and so express the mean as $\mu_i = \mathrm{E}Ni = \exp(x_i'\beta)$, where β is a vector of parameters associated with the covariates. This function is used because it yields desirable parameter interpretations, seems to fit data reasonably well, and ties well with other approaches traditionally used in actuarial ratemaking applications.

Our objective is to build GLMs to predict the frequency using related risk factors, select the most promising GLM, interpret the results and quantify its predictive accuracy.

Remark (Interaction terms): Two predictors are said to interact if the effect that one of them has on the mean response depends on the value of the other. The product term $x_1 \times x_2$ models an interaction between x_1 and x_2. Except in special circumstances, a model including a product term for interaction between two predictors should also include terms with each of the predictors.

Remark (Basis Expansions): Similar to linear models, we could also extend the features using basis functions, such as polynomial, splines, etc. The basis function model becomes

$$g(\mu) = \beta_0 + h(x_1)\beta_1 + \cdots + h(x_p)\beta_p.$$

Additionally, recall that **Generalized Additive Models (GAMs)** were introduced in the "Moving Beyond Linearity" section of the prerequisite course. Feel free to explore them further on your own!

Remark (Shrinkage Techniques): Recall the shrinkage techniques we discussed in Week 3. Using the `glmnet` package, we can do GLM with shrinkage techniques.

3.1 Poisson GLM for Claim Frequencies

3.1.1 Task Solution: Fit a Poisson GLM (Full Model) Using All Available Training Data Except `IDpol`. Model the Number of Claims per Policy as the Target Variable and Include an Offset for Exposure Volume.

```r
# Poisson GLM (full model) using training data
freqPoisson_full <- glm(ClaimNb ~ VehPower + VehAge + DrivAge + BonusMalus +
                          VehBrand + VehGas + Density + Region + Area,
                        data = freMTPL2full, subset = train_indices,
                        offset = log(Exposure), family = poisson(link = "log"))

summary(freqPoisson_full)
```

```
##
## Call:
## glm(formula = ClaimNb ~ VehPower + VehAge + DrivAge + BonusMalus +
##     VehBrand + VehGas + Density + Region + Area, family = poisson(link = "log"),
##     data = freMTPL2full, subset = train_indices, offset = log(Exposure))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4749  -0.3267  -0.2474  -0.1399   6.3287
##
## Coefficients:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -4.668e+00  1.344e-01 -34.734  < 2e-16 ***
## VehPower                         3.761e-02  3.918e-03   9.599  < 2e-16 ***
## VehAge                          -1.889e-02  1.538e-03 -12.280  < 2e-16 ***
## DrivAge                          5.322e-03  5.497e-04   9.681  < 2e-16 ***
## BonusMalus                       2.607e-02  3.964e-04  65.775  < 2e-16 ***
## VehBrandB10                     -5.049e-02  4.779e-02  -1.057 0.290705
## VehBrandB11                      1.435e-01  4.983e-02   2.880 0.003971 **
## VehBrandB12                     -3.153e-01  2.630e-02 -11.989  < 2e-16 ***
## VehBrandB13                      7.807e-02  5.262e-02   1.484 0.137865
## VehBrandB14                     -1.444e-01  1.032e-01  -1.399 0.161775
## VehBrandB2                      -2.953e-04  2.045e-02  -0.014 0.988477
## VehBrandB3                       3.868e-02  2.858e-02   1.353 0.175999
## VehBrandB4                       2.531e-02  3.890e-02   0.651 0.515259
## VehBrandB5                       8.241e-02  3.274e-02   2.517 0.011844 *
## VehBrandB6                      -3.121e-03  3.736e-02  -0.084 0.933415
## VehGasRegular                   -1.618e-01  1.519e-02 -10.649  < 2e-16 ***
## Density                         -3.890e-07  5.096e-06  -0.076 0.939165
## RegionAquitaine                  7.626e-02  1.267e-01   0.602 0.547156
## RegionAuvergne                  -2.471e-02  1.565e-01  -0.158 0.874580
## RegionBasse-Normandie           -9.192e-03  1.336e-01  -0.069 0.945132
## RegionBourgogne                  4.055e-03  1.376e-01   0.029 0.976486
## RegionBretagne                  -6.685e-04  1.247e-01  -0.005 0.995721
## RegionCentre                    -1.520e-02  1.227e-01  -0.124 0.901459
## RegionChampagne-Ardenne         -4.511e-02  1.836e-01  -0.246 0.805912
## RegionCorse                      2.104e-03  1.646e-01   0.013 0.989804
## RegionFranche-Comte             -1.295e-01  2.341e-01  -0.553 0.580229
## RegionHaute-Normandie           -8.565e-02  1.469e-01  -0.583 0.559896
## RegionIle-de-France             -1.724e-02  1.247e-01  -0.138 0.890056
## RegionLanguedoc-Roussillon      -9.995e-03  1.272e-01  -0.079 0.937389
## RegionLimousin                   2.517e-01  1.511e-01   1.666 0.095783 .
## RegionMidi-Pyrenees             -2.524e-01  1.364e-01  -1.851 0.064236 .
## RegionNord-Pas-de-Calais        -8.369e-02  1.258e-01  -0.665 0.505778
## RegionPays-de-la-Loire           8.354e-03  1.252e-01   0.067 0.946790
## RegionPicardie                   2.230e-02  1.402e-01   0.159 0.873600
```

```
## RegionPoitou-Charentes             6.059e-02  1.287e-01   0.471 0.637819
## RegionProvence-Alpes-Cotes-D'Azur  1.144e-01  1.233e-01   0.928 0.353478
## RegionRhone-Alpes                  2.028e-01  1.228e-01   1.652 0.098632 .
## AreaB                              1.044e-01  3.135e-02   3.330 0.000868 ***
## AreaC                              1.516e-01  2.606e-02   5.818 5.97e-09 ***
## AreaD                              2.982e-01  2.760e-02  10.806  < 2e-16 ***
## AreaE                              4.020e-01  3.582e-02  11.221  < 2e-16 ***
## AreaF                              4.616e-01  1.239e-01   3.724 0.000196 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 120058  on 474603  degrees of freedom
## Residual deviance: 115076  on 474562  degrees of freedom
## AIC: 150765
##
## Number of Fisher Scoring iterations: 6
```

3.1.2 What are Weights and Offsets?

To understand the need for weights and offsets in GLMs, consider the following excerpts from Sections 2.5 and 2.6 of Generalized Linear Models for Insurance Rating, targeted at actuaries in the property/casualty insurance industry:

For Weights:

> Frequently, the dataset going into a GLM will include rows that represent the averages of the outcomes of groups of similar risks rather than the outcomes of individual risks. For example, in a claim severity dataset, one row might represent the average loss amount for several claims, all with the same values for all the predictor variables. Or, perhaps, a row in a pure premium dataset might represent the average pure premium for several exposures with the same characteristics (perhaps belonging to the same insured).
>
> In such instances, it is intuitive that rows that represent a greater number of risks should carry more weight in the estimation of the model coefficients, as their outcome values are based on more data. GLMs accommodate that by allowing the user to include a weight variable, which specifies the weight given to each record in the estimation process.

For (Exposure) Offsets:

> Offsets are also used when modeling a target variable that is expected to vary directly with time on risk or some other measure of exposure. An example would be where the target variable is the number of claims per policy for an auto book of business where the term lengths of the policies vary; all else equal, a policy covering two car years is expected to have twice the claims as a one-year policy. This expectation can be reflected in a log-link GLM by including the (logged) number of exposures - car years in this example - as an offset.
>
> Note that this approach is distinct from modeling claims frequency, i.e., where the target variable is the number of claims divided by the number of exposures, which is the more common practice. In a frequency model, the number of exposures should be included as a weight, but not as an offset. In fact: a claim count model that includes exposure as an offset is exactly equivalent to a frequency model that includes exposure as a weight (but not as an offset) - that is, they will yield the same predictions, relativity factors and standard errors.

In general, weights can be identically one or can be inputs into the GLM. In insurance applications, the number of claims or exposures are often used as measures for these weights. Suppose your data contains the number of exposures, you might be curious whether the number of exposures should be introduced into the GLM as weights or offsets. Interestingly, a claim count model that includes exposure as an offset (i.e., where the response is the claim number) is exactly equivalent to a frequency model that includes exposure as a weight (but not as an offset; i.e., where the response is the claim frequency) when the claim distribution is a Poisson or overdispersed Poisson distribution. Please read the following paragraph extracted from Generalized Linear Models for Insurance Rating to understand the difference between weight and offset, and you are also encouraged to read the relevant sections of the book for a deeper understanding:

> To gain an intuition for this relationship, recall that an offset is an adjustment to the mean, while the weight is an adjustment to the variance. For a claim count model, additional exposures on a record carry the expectation of a greater number of claims, and so an offset is required. While the variance of the claim count is also expected to increase with increasing exposure—due to the exponential family's inherent expectation of greater mean implying greater variance-this is naturally handled by the GLM's assumed mean/variance relationship, and so no adjustment to variance (i.e., no weight variable) is necessary. For a claim frequency model, on the other hand, additional exposure carries the expectation of reduced variance (due to the larger volume of exposures yielding greater stability in the average frequency), but no change to the expected mean, and therefore a weight - but no offset - is needed.

The equivalence of the two models can be verified by the following code:

```
#freqPoisson_full is equivalent to:
freqPoisson_fullalt <- glm(ClaimNb/Exposure ~ VehPower + VehAge + DrivAge + BonusMalus +
                           VehBrand + VehGas + Density + Region + Area,
                         data = freMTPL2full, subset = train_indices,
                         weights = Exposure, family = poisson(link = "log"))
summary(freqPoisson_fullalt)
```

Note that the AIC is reported as `NA` here because AIC is based on the log-likelihood, and Poisson GLMs expect the response variable to be integer counts. When you divide by `Exposure`, the response becomes fractional, resulting in invalid likelihood values.

3.2 Model Interpretation

3.2.1 Task Solution: Interpret the GLM Coefficients. Refit the Full Model by Moving `Density` as the Last Predictor, and Compute an Analysis of Deviance Table for the New Generalized Linear Model Fit.

Let's focus on the summary from the Poisson model (full model) as example:

- The coefficient of `DrivAge` is 0.005322. When the driver's age increases by 1, the log of expected value of frequency μ will increase by 0.005322. So μ will increase by $exp(0.005322) - 1 = 0.534\%$.
- We have one categorical variable called `Area`, which can take `A`, `B`, `C`, `D`, `E` or `F`. We deal with this categorical variable by dummy coding. Notice we treat `A` as the 0 level, that's why we only have the cofficients for the other five areas (`AreaB`, `AreaC`, `AreaD`, `AreaE`, `AreaF`). For example, the coefficient of `AreaB` is 0.1044. It means if the `Area` is `B`, the $log(\mu)$ will be 0.1044 greater than that of `AreaA`. Or in another words, μ will increase by $exp(0.1044) - 1 = 11.004\%$.
- `z value` is the test-statistic for the Wald-test (see Section 5.8 on page 75, Generalized Linear model for Insurance Data) that the parameter is 0. The coefficients of most of the variables are not significantly different from 0 at 99.9% according `Pr(>|z|)`.

Why do we want an ANOVA table when I can just get Wald-test of my variables with standard output (summary(glm))? If you have a categorical/factor variable with more than two levels, the summary output is hard to interpret. It will give you tests of individual levels against the reference level, but won't give you a test of the factor as a whole.

Following the task instructions, let's refit the full model by moving Density as the last predictor:

```
freqPoisson_anova <- glm(ClaimNb ~ VehPower + VehAge + DrivAge + BonusMalus +
                           VehBrand + VehGas + Region + Area +  Density,
                         data = freMTPL2full, subset = train_indices,
                         offset = log(Exposure), family = poisson(link = "log"))
```

Double check the summary of summary(freqPoisson_anova) and summary(freqPoisson_full), you can find they are the same.

The ANOVA table will optionally contain test statistics (and P values) comparing the reduction in deviance for the row to the residuals. For models with known dispersion (e.g., binomial and Poisson fits) the chi-squared test is most appropriate, and for those with dispersion estimated by moments (e.g., gaussian, quasibinomial and quasipoisson fits) the F test is most appropriate. (https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/anova.glm)

```
anova(freqPoisson_anova,test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: poisson, link: log
##
## Response: ClaimNb
##
## Terms added sequentially (first to last)
##
##
##             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                        474603     120058
## VehPower     1     31.4    474602     120027 2.063e-08 ***
## VehAge       1     56.6    474601     119970 5.251e-14 ***
## DrivAge      1    398.8    474600     119571 < 2.2e-16 ***
## BonusMalus   1   3597.8    474599     115974 < 2.2e-16 ***
## VehBrand    10    265.3    474589     115708 < 2.2e-16 ***
## VehGas       1     63.7    474588     115645 1.422e-15 ***
## Region      20    311.2    474568     115333 < 2.2e-16 ***
## Area         5    257.7    474563     115076 < 2.2e-16 ***
## Density      1      0.0    474562     115076    0.9392
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#anova(freqPoisson_full,test="Chisq")
```

- From the ANOVA table, we can see the test of the factor Area as a whole. It suggests this factor is important to the model in explaining the response variable.
- In the ANOVA table, we can also observe the Deviance column. It measures reductions in the residual deviance if we include additional predictors into the model compared to the previous model. For example, if we do not include any predictor, the residual deviance is 120058. Then we include VehPower, so the residual deviance drops from 120058 to 120027, indicating the feature importance of VehPower.

- **Question:** How to compare two nested models and test which one is better? Assume Model 1 has q parameters with deviance D_1 and Model 2 has p parameters with deviance D_2. $p > q$. (refer to lecture notes in week 5)

You could use the ANOVA table to compare two nested models by computing the Chi-squared test statistics $(D_1 - D_2)$. Following the rule of thumb mentioned in the GLM video lecture, Model 2 is preferred if $D_1 - D_2 > 2(p - q)$.

You could also use `glm1` and `glm2` to fit two models, and then use `anova(glm1,glm2,test="Chisq")` to compare the two nested models. If more than one object is specified, the table has a row for the residual degrees of freedom and deviance for each model. For all but the first model, the change in degrees of freedom and deviance is also given. (This only makes statistical sense if the models are nested.) It is conventional to list the models from smallest to largest, but this is up to the user. Please try it yourself.

3.3 Choosing the Distribution

3.3.1 Task: Fit a Quasi-Poisson GLM (Full Model) Using All Available Training Data Except `IDpol`. Compare the Summary Output with the Corresponding Poisson GLM. Perform Diagnostic Checks on Both Models.

```
freqQuasipoisson_full <- glm(ClaimNb ~ VehPower + VehAge + DrivAge + BonusMalus +
                          VehBrand + VehGas + Density + Region + Area,
                        data = freMTPL2full,
                        subset = train_indices,
                        offset = log(Exposure),
                        family = quasipoisson(link = "log"))

summary(freqQuasipoisson_full)
```

```
##
## Call:
## glm(formula = ClaimNb ~ VehPower + VehAge + DrivAge + BonusMalus +
##     VehBrand + VehGas + Density + Region + Area, family = quasipoisson(link = "log"),
##     data = freMTPL2full, subset = train_indices, offset = log(Exposure))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4749  -0.3267  -0.2474  -0.1399   6.3287
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -4.668e+00  1.760e-01 -26.521  < 2e-16 ***
## VehPower                        3.761e-02  5.132e-03   7.329 2.32e-13 ***
## VehAge                         -1.889e-02  2.014e-03  -9.377  < 2e-16 ***
## DrivAge                         5.322e-03  7.199e-04   7.392 1.45e-13 ***
## BonusMalus                      2.607e-02  5.192e-04  50.222  < 2e-16 ***
## VehBrandB10                    -5.049e-02  6.259e-02  -0.807  0.41982
## VehBrandB11                     1.435e-01  6.526e-02   2.199  0.02786 *
## VehBrandB12                    -3.153e-01  3.444e-02  -9.154  < 2e-16 ***
## VehBrandB13                     7.807e-02  6.891e-02   1.133  0.25724
## VehBrandB14                    -1.444e-01  1.352e-01  -1.068  0.28539
## VehBrandB2                     -2.953e-04  2.678e-02  -0.011  0.99120
## VehBrandB3                      3.868e-02  3.744e-02   1.033  0.30151
```

```
## VehBrandB4                           2.531e-02  5.095e-02   0.497  0.61932
## VehBrandB5                           8.241e-02  4.288e-02   1.922  0.05465 .
## VehBrandB6                          -3.121e-03  4.893e-02  -0.064  0.94913
## VehGasRegular                       -1.618e-01  1.990e-02  -8.131 4.27e-16 ***
## Density                             -3.890e-07  6.675e-06  -0.058  0.95353
## RegionAquitaine                      7.626e-02  1.659e-01   0.460  0.64575
## RegionAuvergne                      -2.471e-02  2.050e-01  -0.121  0.90407
## RegionBasse-Normandie               -9.192e-03  1.749e-01  -0.053  0.95809
## RegionBourgogne                      4.055e-03  1.802e-01   0.023  0.98205
## RegionBretagne                      -6.685e-04  1.633e-01  -0.004  0.99673
## RegionCentre                        -1.520e-02  1.607e-01  -0.095  0.92468
## RegionChampagne-Ardenne             -4.511e-02  2.405e-01  -0.188  0.85119
## RegionCorse                          2.104e-03  2.156e-01   0.010  0.99221
## RegionFranche-Comte                 -1.295e-01  3.066e-01  -0.422  0.67282
## RegionHaute-Normandie               -8.565e-02  1.924e-01  -0.445  0.65622
## RegionIle-de-France                 -1.724e-02  1.634e-01  -0.106  0.91594
## RegionLanguedoc-Roussillon          -9.995e-03  1.666e-01  -0.060  0.95217
## RegionLimousin                       2.517e-01  1.979e-01   1.272  0.20345
## RegionMidi-Pyrenees                 -2.524e-01  1.787e-01  -1.413  0.15767
## RegionNord-Pas-de-Calais            -8.369e-02  1.647e-01  -0.508  0.61140
## RegionPays-de-la-Loire               8.354e-03  1.639e-01   0.051  0.95936
## RegionPicardie                       2.230e-02  1.836e-01   0.121  0.90332
## RegionPoitou-Charentes               6.059e-02  1.686e-01   0.359  0.71927
## RegionProvence-Alpes-Cotes-D'Azur    1.144e-01  1.614e-01   0.708  0.47866
## RegionRhone-Alpes                    2.028e-01  1.608e-01   1.261  0.20731
## AreaB                                1.044e-01  4.106e-02   2.543  0.01100 *
## AreaC                                1.516e-01  3.413e-02   4.442 8.91e-06 ***
## AreaD                                2.982e-01  3.614e-02   8.251  < 2e-16 ***
## AreaE                                4.020e-01  4.692e-02   8.568  < 2e-16 ***
## AreaF                                4.616e-01  1.623e-01   2.844  0.00446 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 1.715289)
##
##     Null deviance: 120058  on 474603  degrees of freedom
## Residual deviance: 115076  on 474562  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

You can see the coefficients from two models are totally the same but the standard errors of Quasi-Poisson is larger than those of Poisson. For Poisson model, the assumed relationship is that the variance equals the expected value. For Quasi-Poisson model, the variance is assumed to be a linear function of the mean. The dispersion parameter, which was forced to be 1 in Poisson model, is allowed to be estimated in Quasi-Poisson model. In fact, it is estimated at 1.715289. This parameter tells us how many times the variance is larger than the mean. Since our dispersion was larger than one, it turns out the conditional variance is actually larger than the conditional mean. We have over-dispersion in this case (under-dispersion if it is less than 1).

Randomized Quantile Residuals (RQR) v.s. Fitted Values (FV)

```r
# Sample 50,000 indices to use in the plots for faster generation
set.seed(809)
sample_indices <- sample(1:length(train_indices), 50000)
```

```r
par(mfrow=c(1,2))

set.seed(309)
plot(freqPoisson_full$fitted.values[sample_indices],qresiduals(freqPoisson_full)[sample_indices],
     xlab="Fitted values",ylab="Randomized Quantile Residuals",
     main="RQR vs FV, Poisson",
     cex.main=0.8, cex.lab=0.8, cex.axis=0.7)
abline(h=0,col="red",lty=2)

plot(freqQuasipoisson_full$fitted.values[sample_indices],
qres.pois(freqQuasipoisson_full)[sample_indices],
# use qres.pois directly as qresiduals can't automatically select distribution for quasipoisson
     xlab="Fitted values",ylab="Randomized Quantile Residuals",
     main="RQR vs FV, Quasipoisson",
     cex.main=0.8, cex.lab=0.8, cex.axis=0.7)
abline(h=0,col="red",lty=2)
```

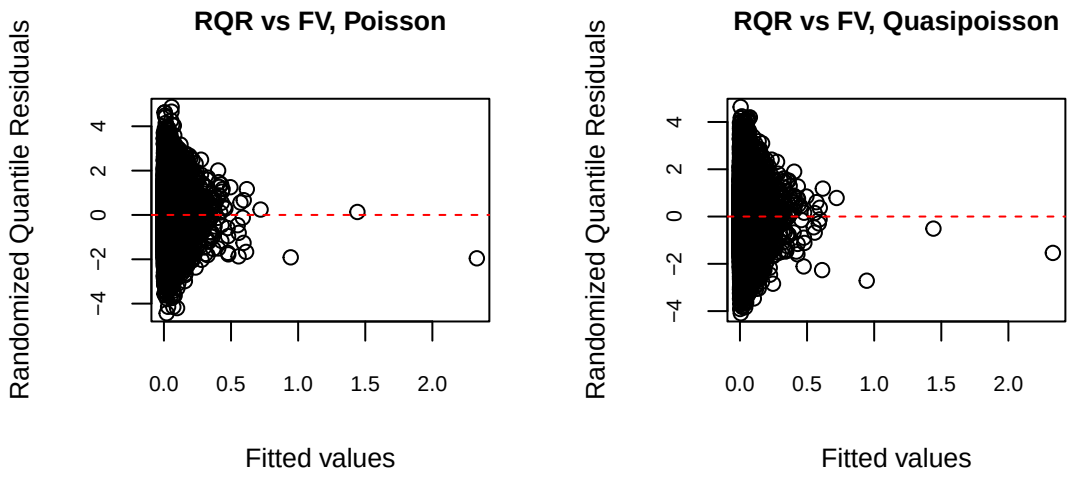


Figure 1: RQR vs FV, Poisson and Quasipoisson

For discrete distributions like Poisson or negative binomial, or distributions with a point mass such as Tweedie, deviance residuals often do not follow a normal distribution. This is because they account for the distribution's shape but not its discreteness, leading to clusters of residuals around common target values. As a result, deviance residuals are less effective for assessing the fit of these models. A potential solution is using **randomized quantile residuals**, which add random jitter to spread the discrete points more smoothly across the distribution (Goldburd et al. 2016).

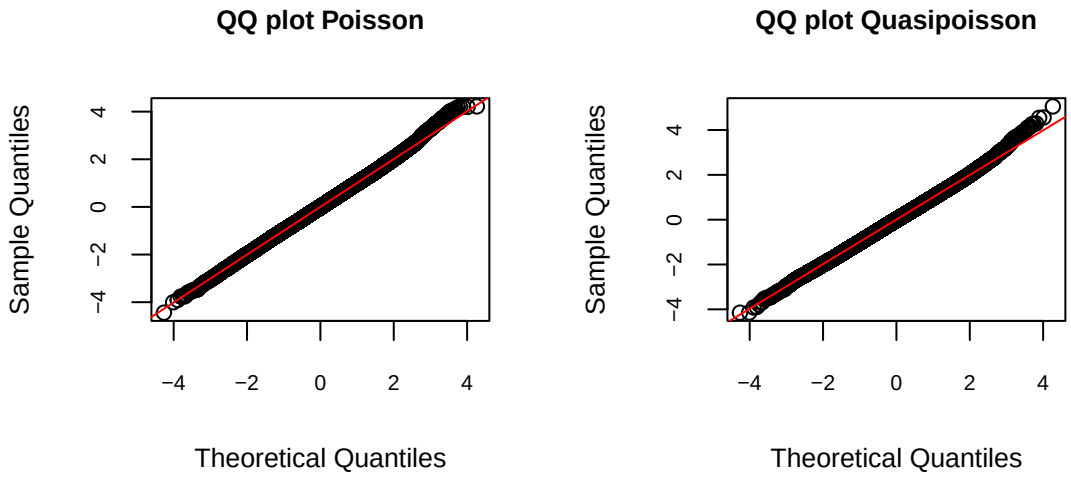
Ideally, in Figure 1, we want to see the randomized quantile residuals follow no predictable pattern and be normally distributed. However, the spread of residuals decreases as the fitted values increase.

QQ plot of randomized quantile residuals

The QQ plot, or quantile-quantile plot, is a graphical tool to help us assess if a set of data plausibly came from some theoretical distribution such as a Normal or exponential. If both sets of quantiles came from the same distribution, we should see the points forming a line that's roughly straight.

Deviance residuals are approximately normally distributed for most response distributions. We can use QQ plot of the randomized quantile residuals as a diagnostic tool.

```r
par(mfrow=c(1,2))

set.seed(409)

qqnorm(qresiduals(freqPoisson_full)[sample_indices],main="QQ plot Poisson",
       cex.main=0.8, cex.lab=0.8, cex.axis=0.7)
qqline(qresiduals(freqPoisson_full)[sample_indices],col="red")

qqnorm(qres.pois(freqQuasipoisson_full)[sample_indices],main="QQ plot Quasipoisson",
       cex.main=0.8, cex.lab=0.8, cex.axis=0.7)
qqline(qres.pois(freqQuasipoisson_full)[sample_indices],col="red")
```



- It looks generally good. The sample quantiles only deviate from the theoretical quantiles in the upper tail.

3.3.2 Task: Fit a Negative Binomial GLM (Full Model) Using All Available Training Data Except `IDpol`. Perform Diagnostic Checks and Comment on the Results.

```r
freqNb_full<-glm.nb(ClaimNb ~ offset(Exposure) + VehPower + VehAge + DrivAge + BonusMalus +
                      VehBrand + VehGas + Density + Region + Area,
                    data= freMTPL2full,
                    subset = train_indices)

summary(freqNb_full)
```

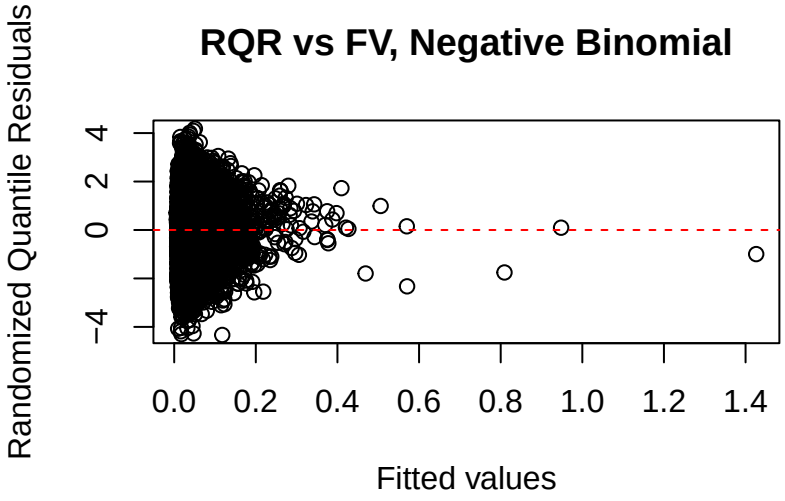
```
##
## Call:
## glm.nb(formula = ClaimNb ~ offset(Exposure) + VehPower + VehAge +
##     DrivAge + BonusMalus + VehBrand + VehGas + Density + Region +
##     Area, data = freMTPL2full, subset = train_indices, init.theta = 1.093825159,
##     link = log)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7752  -0.3049  -0.2532  -0.2044   5.0239
##
## Coefficients:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -5.857e+00  1.391e-01 -42.120  < 2e-16 ***
## VehPower                         3.306e-02  4.004e-03   8.257  < 2e-16 ***
## VehAge                          -1.559e-02  1.556e-03 -10.019  < 2e-16 ***
## DrivAge                          8.427e-03  5.744e-04  14.670  < 2e-16 ***
## BonusMalus                       2.391e-02  4.357e-04  54.892  < 2e-16 ***
## VehBrandB10                     -6.803e-02  4.900e-02  -1.388 0.165066
## VehBrandB11                      1.022e-01  5.128e-02   1.992 0.046356 *
## VehBrandB12                     -4.226e-01  2.688e-02 -15.724  < 2e-16 ***
## VehBrandB13                      7.194e-02  5.418e-02   1.328 0.184257
## VehBrandB14                     -1.534e-01  1.056e-01  -1.452 0.146493
## VehBrandB2                       3.959e-03  2.099e-02   0.189 0.850418
## VehBrandB3                       3.141e-02  2.939e-02   1.069 0.285059
## VehBrandB4                       2.311e-02  3.993e-02   0.579 0.562682
## VehBrandB5                       9.314e-02  3.371e-02   2.763 0.005721 **
## VehBrandB6                      -1.006e-02  3.834e-02  -0.262 0.792978
## VehGasRegular                   -1.379e-01  1.559e-02  -8.846  < 2e-16 ***
## Density                          7.058e-07  5.203e-06   0.136 0.892100
## RegionAquitaine                 -1.942e-02  1.304e-01  -0.149 0.881612
## RegionAuvergne                  -9.230e-02  1.605e-01  -0.575 0.565268
## RegionBasse-Normandie            2.522e-02  1.375e-01   0.183 0.854480
## RegionBourgogne                 -6.732e-02  1.415e-01  -0.476 0.634234
## RegionBretagne                   4.065e-02  1.284e-01   0.317 0.751554
## RegionCentre                    -7.320e-03  1.264e-01  -0.058 0.953820
## RegionChampagne-Ardenne         -1.421e-01  1.876e-01  -0.757 0.448850
## RegionCorse                     -1.011e-01  1.685e-01  -0.600 0.548687
## RegionFranche-Comte             -2.109e-01  2.385e-01  -0.884 0.376636
## RegionHaute-Normandie           -3.442e-01  1.506e-01  -2.286 0.022280 *
## RegionIle-de-France             -8.741e-02  1.284e-01  -0.681 0.496173
## RegionLanguedoc-Roussillon      -1.496e-01  1.309e-01  -1.142 0.253268
## RegionLimousin                   2.246e-01  1.554e-01   1.445 0.148422
## RegionMidi-Pyrenees             -3.623e-01  1.400e-01  -2.587 0.009685 **
## RegionNord-Pas-de-Calais        -1.484e-01  1.294e-01  -1.146 0.251678
## RegionPays-de-la-Loire          -4.351e-03  1.289e-01  -0.034 0.973072
## RegionPicardie                  -2.620e-02  1.442e-01  -0.182 0.855854
## RegionPoitou-Charentes           5.531e-02  1.325e-01   0.417 0.676426
## RegionProvence-Alpes-Cotes-D'Azur 3.205e-02  1.269e-01   0.252 0.800713
## RegionRhone-Alpes                1.833e-01  1.265e-01   1.449 0.147390
## AreaB                            9.852e-02  3.201e-02   3.078 0.002086 **
## AreaC                            1.353e-01  2.660e-02   5.088 3.62e-07 ***
## AreaD                            2.686e-01  2.819e-02   9.525  < 2e-16 ***
## AreaE                            3.543e-01  3.656e-02   9.690  < 2e-16 ***
## AreaF                            4.320e-01  1.261e-01   3.426 0.000614 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(1.0938) family taken to be 1)
##
##     Null deviance: 105759  on 474603  degrees of freedom
## Residual deviance: 101856  on 474562  degrees of freedom
```

```
## AIC: 150542
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  1.0938
##           Std. Err.:  0.0749
##
##  2 x log-likelihood:  -150456.4300
```

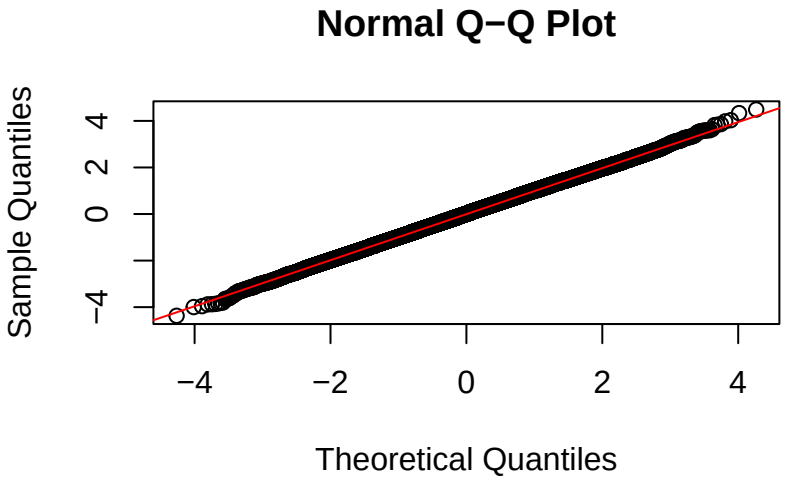
Randomized quantile residuals v.s. Fitted Values

```
set.seed(742)
plot(freqNb_full$fitted.values[sample_indices],qresiduals(freqNb_full)[sample_indices],
     xlab="Fitted values",ylab="Randomized Quantile Residuals",
     main="RQR vs FV, Negative Binomial")
abline(h=0,col="red",lty=2)
```



QQ plot of randomized quantile residuals

```
qqnorm(qresiduals(freqNb_full)[sample_indices])
qqline(qresiduals(freqNb_full)[sample_indices],col="red")
```



3.3.3 Other Distributions (Not Examinable): Zero-inflated models

Insurance claim data often have an excessive number of zeros. The advantage of zero-inflated models is that they are good at handling a mass at zero, which means no claims. A binary model (usually logit or probit) is used to model the zero-inflation part (non claim / claims) of the data. You are encouraged to explore this yourself.

- Hint: You can use the `zeroinfl` function from the `pscl` package to implement zero-inflated regression. It allows the use of an `offset`, similar to how we used it previously in the `glm` function.

3.4 Feature Selection

3.4.1 Forward and Backward AIC/BIC Selections

Regression requires you to add variables into your model and you test each one to see whether it is significantly different than zero. Stepwise regression allows you to do it by following ways.

- Forward: Start with a simple model and automatically add variables to it.
- Backward: Start with a complex model and automatically remove variables from it.
- Both: Iteratively add or remove variables each step.

To learn this part of knowledge, you can read Chapter 6 Section 6.1.2 of *An Introduction to Statistical Learning* and Week 3 Lecture Slides (filter, *wrapper*, embedded). The `MASS` package offers the `stepAIC` function and it is a wrapper method.

```
glm.full<-glm(ClaimNb ~ offset(log(Exposure)) + VehPower + VehAge + DrivAge + BonusMalus +
                VehBrand + VehGas + Density + Region + Area,
              data = freMTPL2full, subset = train_indices, family = poisson(link = "log"))

glm.none <- glm(ClaimNb ~ offset(log(Exposure)) + 1,
                data = freMTPL2full, subset = train_indices, family = poisson(link = "log"))
```

```r
# AIC backward
aic_back<-stepAIC(glm.full,
                  direction = "backward",
                  k = 2, # set k = 2 for AIC; k = log(n) for BIC
                  scope = list(upper = glm.full, lower = glm.none)
)

freqPoisson_step <- glm(ClaimNb ~ VehPower + VehAge + DrivAge + BonusMalus +
                          VehBrand + VehGas + Region + Area,
                        data = freMTPL2full, subset = train_indices,
                        offset = log(Exposure), family = poisson(link = "log"))
summary(freqPoisson_step)
```

We performed backward stepwise selection using AIC, which only removed `Density` compared to the full model.

Exercise:

1. Run the code above in the R Markdown file. Make sure you understand the intermediate outputs of the stepwise selection process.

2. So far, we've only used backward stepwise selection with AIC. Now, explore other stepwise selection methods (e.g., forward stepwise selection) or use a different criterion (e.g., BIC). Do these approaches result in a different set of predictors being selected?

3.4.2 Regularization Methods (from Week 3)

Alternatively, we can fit Poisson lasso or ridge regression. We leave these methods as an exercise for you to explore further.

3.5 Transformation of Variables

3.5.1 An Example of Transformation of `Age`

GLMs assume a linear relationship between the transformed predictor variables (after applying a link function) and the expected outcome. However, the relationship between predictors and the response variable may not always be linear. Applying transformations (e.g., logarithms, polynomials, or splines) can help capture more complex non-linear relationships. In many cases, the variable needs to be transformed to ensure that the resulting model provides a better fit to the data. In this subsection, we will explore some possible transformations of `age`, as previewed in our EDA during Week 2's lab.

Here, we consider two transformations as discussed in Schelldorfer and Wuthrich (2019). First, we add polynomial terms to the predictor `DrivAge` by including the log transformation and powers of `DrivAge` up to the fourth degree. Second, we apply a binning technique to the continuous `DrivAge` variable. We divide `DrivAge` into age brackets (e.g., "18-20", "21-25", etc.), treating these age ranges as categorical levels.

```r
#Adding Polynomial Terms
freqPoisson_age1 <- glm(ClaimNb ~ VehPower + VehAge + BonusMalus +
                          VehBrand + VehGas + Density + Region + Area +
                          + DrivAge + log(DrivAge) +  I(DrivAge^2) + I(DrivAge^3) + I(DrivAge^4),
                        data = freMTPL2full, subset = train_indices,
                        offset = log(Exposure), family = poisson(link = "log"))
summary(freqPoisson_age1)
```

```r
#Binning Continuous Predictors
freMTPL2full$DrivAgeBins <- as.factor(cut(freMTPL2full$DrivAge, c(18, 20, 25, 30, 40, 50, 70, 101),
                        labels = c("18-20", "21-25", "26-30", "31-40", "41-50", "51-70", "71+"),
                        include.lowest = TRUE))

freqPoisson_age2 <- glm(ClaimNb ~ VehPower + VehAge + BonusMalus +
                          VehBrand + VehGas + Density + Region + Area +
                          + DrivAgeBins,
                        data = freMTPL2full, subset = train_indices,
                        offset = log(Exposure), family = poisson(link = "log"))
summary(freqPoisson_age2)
```

Exercise: What are the limitations of these two transformations?

Hint: Please refer to Sections 5.4.2 and 5.4.3 of Generalized Linear Models for Insurance Rating for further insights on the limitations of polynomial terms and binning continuous predictors.

3.5.2 Poisson GLM with Pre-processed Data Following Wüthrich and Merz (2023)

Lastly, we present an example of a Poisson frequency model following the feature engineering steps in Wüthrich and Merz (2023), as shown in the code below:

```r
freMTPL2prep <- freMTPL2full
freMTPL2prep$AreaGLM <- as.integer(freMTPL2prep$Area)
freMTPL2prep$VehPowerGLM <- as.factor(pmin(freMTPL2prep$VehPower, 9))
freMTPL2prep$VehAgeGLM <- as.factor(cut(freMTPL2prep$VehAge, c(0, 5, 12, 101),
                                        labels = c("0-5", "6-12", "12+"),
                                        include.lowest = TRUE))
freMTPL2prep$DrivAgeGLM <- as.factor(cut(freMTPL2prep$DrivAge, c(18, 20, 25, 30, 40, 50, 70, 101),
                                         labels = c("18-20", "21-25", "26-30", "31-40", "41-50",
                                                    "51-70", "71+"), include.lowest = TRUE))
freMTPL2prep$BonusMalusGLM <- pmin(freMTPL2prep$BonusMalus, 150)
freMTPL2prep$DensityGLM <- log(freMTPL2prep$Density)
```

```r
freqPoisson_full2 <- glm(ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM + BonusMalusGLM +
                           VehBrand + VehGas + DensityGLM + Region + AreaGLM,
                         data = freMTPL2prep, subset = train_indices,
                         offset = log(Exposure), family = poisson(link = "log"))
summary(freqPoisson_full2)
```

```
##
## Call:
## glm(formula = ClaimNb ~ VehPowerGLM + VehAgeGLM + DrivAgeGLM +
##     BonusMalusGLM + VehBrand + VehGas + DensityGLM + Region +
##     AreaGLM, family = poisson(link = "log"), data = freMTPL2prep,
##     subset = train_indices, offset = log(Exposure))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4965  -0.3263  -0.2457  -0.1384   6.3663
##
## Coefficients:
##                                   Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)                          -4.3902492  0.1442319 -30.439  < 2e-16 ***
## VehPowerGLM5                          0.0730051  0.0261232   2.795 0.005196 **
## VehPowerGLM6                          0.1096382  0.0255980   4.283 1.84e-05 ***
## VehPowerGLM7                          0.0929838  0.0254440   3.654 0.000258 ***
## VehPowerGLM8                          0.1393255  0.0359724   3.873 0.000107 ***
## VehPowerGLM9                          0.2578063  0.0285490   9.030  < 2e-16 ***
## VehAgeGLM6-12                        -0.0354264  0.0174740  -2.027 0.042623 *
## VehAgeGLM12+                         -0.2932947  0.0226510 -12.948  < 2e-16 ***
## DrivAgeGLM21-25                      -0.4704740  0.0560478  -8.394  < 2e-16 ***
## DrivAgeGLM26-30                      -0.5668809  0.0547323 -10.357  < 2e-16 ***
## DrivAgeGLM31-40                      -0.3952612  0.0527897  -7.487 7.02e-14 ***
## DrivAgeGLM41-50                      -0.1223895  0.0535093  -2.287 0.022181 *
## DrivAgeGLM51-70                      -0.1851709  0.0535873  -3.455 0.000549 ***
## DrivAgeGLM71+                        -0.2040892  0.0599282  -3.406 0.000660 ***
## BonusMalusGLM                         0.0273194  0.0004379  62.386  < 2e-16 ***
## VehBrandB10                          -0.0363004  0.0480897  -0.755 0.450341
## VehBrandB11                           0.1816102  0.0497203   3.653 0.000260 ***
## VehBrandB12                          -0.2701251  0.0266707 -10.128  < 2e-16 ***
## VehBrandB13                           0.0708641  0.0527654   1.343 0.179271
## VehBrandB14                          -0.1379026  0.1034080  -1.334 0.182342
## VehBrandB2                            0.0069530  0.0205377   0.339 0.734949
## VehBrandB3                            0.0434720  0.0286898   1.515 0.129711
## VehBrandB4                            0.0251924  0.0390694   0.645 0.519049
## VehBrandB5                            0.0737139  0.0330941   2.227 0.025920 *
## VehBrandB6                           -0.0064650  0.0375096  -0.172 0.863158
## VehGasRegular                        -0.1529144  0.0159270  -9.601  < 2e-16 ***
## DensityGLM                            0.0530373  0.0169028   3.138 0.001702 **
## RegionAquitaine                       0.0783005  0.1266686   0.618 0.536475
## RegionAuvergne                       -0.0207106  0.1565428  -0.132 0.894747
## RegionBasse-Normandie                -0.0080679  0.1335590  -0.060 0.951831
## RegionBourgogne                       0.0115694  0.1375741   0.084 0.932980
## RegionBretagne                       -0.0108365  0.1245820  -0.087 0.930685
## RegionCentre                         -0.0172560  0.1226777  -0.141 0.888138
## RegionChampagne-Ardenne              -0.0280496  0.1835825  -0.153 0.878564
## RegionCorse                           0.0038359  0.1644847   0.023 0.981394
## RegionFranche-Comte                  -0.1430258  0.2340928  -0.611 0.541213
## RegionHaute-Normandie                -0.0867885  0.1469157  -0.591 0.554697
## RegionIle-de-France                  -0.0476711  0.1238943  -0.385 0.700406
## RegionLanguedoc-Roussillon           -0.0235679  0.1271251  -0.185 0.852922
## RegionLimousin                        0.2511484  0.1510084   1.663 0.096284 .
## RegionMidi-Pyrenees                  -0.2428027  0.1364165  -1.780 0.075098 .
## RegionNord-Pas-de-Calais             -0.0812544  0.1256010  -0.647 0.517680
## RegionPays-de-la-Loire               -0.0027133  0.1251385  -0.022 0.982701
## RegionPicardie                        0.0433285  0.1401706   0.309 0.757236
## RegionPoitou-Charentes                0.0578487  0.1286153   0.450 0.652869
## RegionProvence-Alpes-Cotes-D'Azur     0.1083850  0.1232660   0.879 0.379251
## RegionRhone-Alpes                     0.1912720  0.1227259   1.559 0.119108
## AreaGLM                               0.0336461  0.0227506   1.479 0.139164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 120058  on 474603  degrees of freedom
## Residual deviance: 114732  on 474556  degrees of freedom
```

```
## AIC: 150433
##
## Number of Fisher Scoring iterations: 6
```

3.6 Model Comparison

```
#Function for Poisson Deviance
compute_poisson_deviance <- function(model, test_data, exposure) {
  # Predict on the test set using the model
  predicted_values <- predict(model, newdata = test_data, type = "response")

  # Extract the observed values
  observed_values <- test_data$ClaimNb

  # Compute the Poisson deviance
  deviance <- 2 * sum(observed_values * log(observed_values / predicted_values) -
                        (observed_values - predicted_values), na.rm = TRUE)

  return(deviance)
}
```

```
#Find AIC and Poisson Deviance
dev_poisson_full <- compute_poisson_deviance(freqPoisson_full,
                                             test_data = freMTPL2full[-train_indices, ],
                                             exposure = freMTPL2full$Exposure[-train_indices])
print(dev_poisson_full)
```

```
## [1] 34015.79
```

```
aic_poisson_full <- AIC(freqPoisson_full)
print(aic_poisson_full)
```

```
## [1] 150765
```

```
dev_poisson_full2 <- compute_poisson_deviance(freqPoisson_full2,
                                              test_data = freMTPL2prep[-train_indices, ],
                                              exposure = freMTPL2prep$Exposure[-train_indices])
print(dev_poisson_full2)
```

```
## [1] 33857.75
```

```
aic_poisson_full2 <- AIC(freqPoisson_full2)
print(aic_poisson_full2)
```

```
## [1] 150433.5
```

In the code above, two Poisson regression models - `freqPoisson_full`, using the original data, and `freqPoisson_full2`, using pre-processed data following Wüthrich and Merz (2023) - are compared. AIC is used as an in-sample measure on the training data to evaluate model fit. Poisson deviance is computed on the test data as an out-of-sample measure to assess predictive performance. As expected, `freqPoisson_full2` outperforms `freqPoisson_full` on both measures.

4 Modeling Claim Severity Using GLMs

For insurance analysts, one strength of the GLM approach is that the same set of routines can be used for continuous as well as discrete outcomes. For the severity, it is common to use a **gamma** or **inverse Gaussian** distribution, often with a **logarithmic link** (primarily for parameter interpretability).

4.1 Gamma GLM for Claim Severities

4.1.1 Task Solution: Fit a Gamma GLM (Full Model) Using All Available Training Data Except `IDpol`. Model the Total Claim per Policy as the Target Variable, Include an Offset for Claim Count, and Set Claim Count as the GLM Weight.

```
#Total claim size
sevGamma_full <- glm(ClaimTotal ~ offset(log(ClaimNb)) + VehPower + VehAge + DrivAge + BonusMalus +
                       VehBrand + VehGas + Density + Region + Area,
                     family = Gamma(link = "log"),
                     weights = ClaimNb,
                     data = filter(freMTPL2full[train_indices, ], ClaimNb > 0))

summary(sevGamma_full)
```

```
##
## Call:
## glm(formula = ClaimTotal ~ offset(log(ClaimNb)) + VehPower +
##     VehAge + DrivAge + BonusMalus + VehBrand + VehGas + Density +
##     Region + Area, family = Gamma(link = "log"), data = filter(freMTPL2full[train_indices,
##     ], ClaimNb > 0), weights = ClaimNb)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.579  -1.035  -0.612  -0.262  34.796
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     6.354e+00  9.058e-01   7.015 2.39e-12 ***
## VehPower                        3.088e-02  2.630e-02   1.174  0.24043
## VehAge                          2.027e-03  1.055e-02   0.192  0.84758
## DrivAge                        -3.481e-03  3.861e-03  -0.902  0.36727
## BonusMalus                      9.157e-03  2.836e-03   3.229  0.00124 **
## VehBrandB10                     2.505e-01  3.208e-01   0.781  0.43497
## VehBrandB11                     6.265e-01  3.325e-01   1.884  0.05957 .
## VehBrandB12                     1.607e-01  1.754e-01   0.916  0.35951
## VehBrandB13                    -8.543e-02  3.551e-01  -0.241  0.80988
## VehBrandB14                     4.943e-03  6.936e-01   0.007  0.99431
## VehBrandB2                      1.625e-01  1.375e-01   1.182  0.23714
## VehBrandB3                      1.304e-01  1.921e-01   0.679  0.49721
## VehBrandB4                      1.746e-01  2.615e-01   0.668  0.50421
## VehBrandB5                     -1.192e-01  2.202e-01  -0.541  0.58820
## VehBrandB6                     -2.537e-01  2.510e-01  -1.011  0.31208
## VehGasRegular                   1.796e-01  1.022e-01   1.758  0.07875 .
## Density                         1.305e-05  3.436e-05   0.380  0.70414
## RegionAquitaine                 3.712e-01  8.514e-01   0.436  0.66283
```

```
## RegionAuvergne                    -2.926e-02  1.052e+00  -0.028  0.97782
## RegionBasse-Normandie              4.109e-01  8.980e-01   0.458  0.64725
## RegionBourgogne                    1.702e-01  9.252e-01   0.184  0.85402
## RegionBretagne                     4.083e-01  8.381e-01   0.487  0.62613
## RegionCentre                       6.854e-01  8.247e-01   0.831  0.40591
## RegionChampagne-Ardenne            1.946e+00  1.235e+00   1.576  0.11504
## RegionCorse                        9.935e-01  1.106e+00   0.898  0.36901
## RegionFranche-Comte                2.298e-01  1.574e+00   0.146  0.88394
## RegionHaute-Normandie              2.107e-01  9.881e-01   0.213  0.83116
## RegionIle-de-France                3.585e-01  8.381e-01   0.428  0.66885
## RegionLanguedoc-Roussillon         2.849e-01  8.555e-01   0.333  0.73909
## RegionLimousin                    -8.483e-02  1.016e+00  -0.083  0.93347
## RegionMidi-Pyrenees                1.717e-01  9.170e-01   0.187  0.85150
## RegionNord-Pas-de-Calais           2.953e-01  8.454e-01   0.349  0.72690
## RegionPays-de-la-Loire             1.875e-01  8.414e-01   0.223  0.82366
## RegionPicardie                     1.394e-03  9.421e-01   0.001  0.99882
## RegionPoitou-Charentes             1.328e-01  8.653e-01   0.153  0.87802
## RegionProvence-Alpes-Cotes-D'Azur  6.087e-01  8.286e-01   0.735  0.46258
## RegionRhone-Alpes                  5.520e-01  8.255e-01   0.669  0.50375
## AreaB                              2.660e-01  2.105e-01   1.263  0.20647
## AreaC                              1.949e-02  1.748e-01   0.112  0.91121
## AreaD                              1.609e-02  1.852e-01   0.087  0.93074
## AreaE                             -9.510e-02  2.401e-01  -0.396  0.69207
## AreaF                             -4.909e-01  8.403e-01  -0.584  0.55911
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 45.14967)
##
##     Null deviance: 33593  on 17500  degrees of freedom
## Residual deviance: 30042  on 17459  degrees of freedom
## AIC: 323412
##
## Number of Fisher Scoring iterations: 16
```

Here, we model the total claim size as the response. Since records with more claims are expected to have more stable experience, the claim count is set as the GLM weight, and we only use records with a nonzero claim count (Frees et al. 2014). Alternatively, we can model the average claim size as the response (the two models should produce the same results), as shown in the code below:

```r
#Average claim size
sevGamma_fullalt <- glm(ClaimTotal/ClaimNb ~ VehPower + VehAge + DrivAge + BonusMalus +
                          VehBrand + VehGas + Density + Region + Area,
                        family = Gamma(link = "log"),
                        weights = ClaimNb,
                        data = filter(freMTPL2full[train_indices, ], ClaimNb > 0))

summary(sevGamma_fullalt)
```

Please note that the main (or possibly the only) difference between the two equivalent models is that the first model will output the total claim amount when making predictions, while the second model will output the claim severity.

Modelling Individual Claim Size

In fact, if we don't merge the frequency and severity data, we can model the claim size using individual claim sizes, which is a more ideal approach compared to using average or total claim sizes (you might want to think about why this is the case).

```
freq_data2 <- freMTPL2freq[, -2]
freq_data2$VehGas <- factor(freq_data2$VehGas)
sev_data2 <- freMTPL2sev
sev_data2$ClaimNb <- 1
merged_data2 <- merge(x=sev_data2, y=freq_data2, by="IDpol", all.x=TRUE)
merged_data2 <- merged_data2[merged_data2$ClaimNb <= 5, ]
merged_data2$Exposure <- pmin(merged_data2$Exposure, 1)
freMTPL2full2 <- merged_data2
```

Note that `freMTPL2full2` only contains severity data for policies with a claim number greater than 0, and a policy may appear in multiple rows if it has multiple claims.

```
# Extract IDpols from freMTPL2full based on train_indices
train_IDpols <- freMTPL2full[train_indices, "IDpol"]
# Find rows in freMTPL2full2 where IDpol matches those from train_IDpols
train_indices2 <- which(freMTPL2full2$IDpol %in% train_IDpols)
```

If you have separate severity and frequency data, note that the severity data has a different format from the merged data `freMTPL2full` we used earlier (where a row represents either a policy or a claim). To ensure a fair comparison across different models on the testing set, we can filter the training data in `freMTPL2full2` by only keeping rows where the `IDpol` appears in `freMTPL2full[train_indices, ]`.

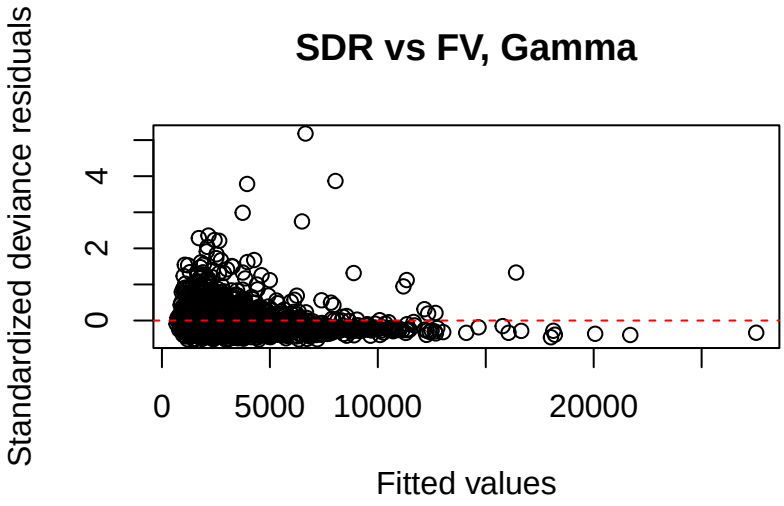
```
#Individual claim size
sevGamma_fullalt2 <- glm(ClaimAmount ~ VehPower + VehAge + DrivAge + BonusMalus +
                           VehBrand + VehGas + Density + Region + Area,
                         family = Gamma(link = "log"),
                         data = freMTPL2full2[train_indices2, ])

summary(sevGamma_fullalt2)
```

```
##
## Call:
## glm(formula = ClaimAmount ~ VehPower + VehAge + DrivAge + BonusMalus +
##     VehBrand + VehGas + Density + Region + Area, family = Gamma(link = "log"),
##     data = freMTPL2full2[train_indices2, ])
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.579  -1.020  -0.606  -0.261  34.795
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     6.354e+00  9.309e-01   6.825 9.05e-12 ***
## VehPower                        3.088e-02  2.703e-02   1.142  0.25336
## VehAge                          2.027e-03  1.084e-02   0.187  0.85164
## DrivAge                        -3.481e-03  3.968e-03  -0.877  0.38034
## BonusMalus                      9.158e-03  2.915e-03   3.142  0.00168 **
## VehBrandB10                     2.505e-01  3.297e-01   0.760  0.44748
## VehBrandB11                     6.265e-01  3.418e-01   1.833  0.06679 .
## VehBrandB12                     1.607e-01  1.803e-01   0.892  0.37263
```

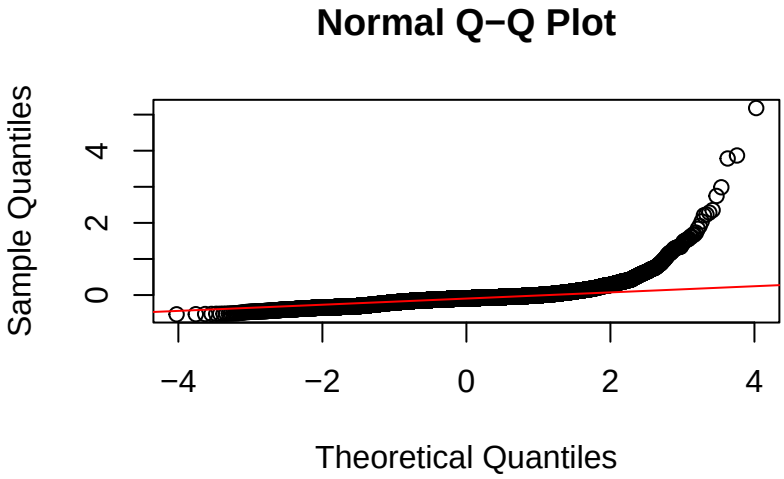
```
## VehBrandB13                          -8.543e-02  3.649e-01  -0.234  0.81492
## VehBrandB14                           4.939e-03  7.128e-01   0.007  0.99447
## VehBrandB2                            1.625e-01  1.413e-01   1.150  0.25004
## VehBrandB3                            1.304e-01  1.974e-01   0.661  0.50891
## VehBrandB4                            1.746e-01  2.687e-01   0.650  0.51581
## VehBrandB5                           -1.192e-01  2.263e-01  -0.527  0.59832
## VehBrandB6                           -2.537e-01  2.579e-01  -0.984  0.32533
## VehGasRegular                         1.796e-01  1.050e-01   1.711  0.08717 .
## Density                               1.305e-05  3.531e-05   0.369  0.71179
## RegionAquitaine                       3.712e-01  8.751e-01   0.424  0.67140
## RegionAuvergne                       -2.926e-02  1.081e+00  -0.027  0.97842
## RegionBasse-Normandie                 4.109e-01  9.230e-01   0.445  0.65616
## RegionBourgogne                       1.702e-01  9.509e-01   0.179  0.85793
## RegionBretagne                        4.083e-01  8.614e-01   0.474  0.63549
## RegionCentre                          6.854e-01  8.476e-01   0.809  0.41871
## RegionChampagne-Ardenne               1.946e+00  1.269e+00   1.533  0.12519
## RegionCorse                           9.935e-01  1.137e+00   0.874  0.38209
## RegionFranche-Comte                   2.298e-01  1.618e+00   0.142  0.88706
## RegionHaute-Normandie                 2.107e-01  1.016e+00   0.207  0.83566
## RegionIle-de-France                   3.585e-01  8.614e-01   0.416  0.67728
## RegionLanguedoc-Roussillon            2.849e-01  8.792e-01   0.324  0.74590
## RegionLimousin                       -8.483e-02  1.044e+00  -0.081  0.93527
## RegionMidi-Pyrenees                   1.717e-01  9.424e-01   0.182  0.85547
## RegionNord-Pas-de-Calais              2.953e-01  8.689e-01   0.340  0.73399
## RegionPays-de-la-Loire                1.875e-01  8.648e-01   0.217  0.82835
## RegionPicardie                        1.391e-03  9.682e-01   0.001  0.99885
## RegionPoitou-Charentes                1.328e-01  8.893e-01   0.149  0.88129
## RegionProvence-Alpes-Cotes-D'Azur     6.087e-01  8.516e-01   0.715  0.47476
## RegionRhone-Alpes                     5.520e-01  8.485e-01   0.651  0.51535
## AreaB                                 2.660e-01  2.164e-01   1.229  0.21898
## AreaC                                 1.949e-02  1.796e-01   0.109  0.91359
## AreaD                                 1.609e-02  1.903e-01   0.085  0.93261
## AreaE                                -9.510e-02  2.468e-01  -0.385  0.69999
## AreaF                                -4.909e-01  8.636e-01  -0.568  0.56979
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 47.69241)
##
##     Null deviance: 34585  on 18504  degrees of freedom
## Residual deviance: 31034  on 18463  degrees of freedom
## AIC: 320320
##
## Number of Fisher Scoring iterations: 16
```

The model fit for individual claim size is shown above.

```
plot(sevGamma_full$fitted.values,rstandard(sevGamma_full),
     xlab="Fitted values",ylab="Standardized deviance residuals",
     main="SDR vs FV, Gamma")
abline(h=0,col="red",lty=2)
```



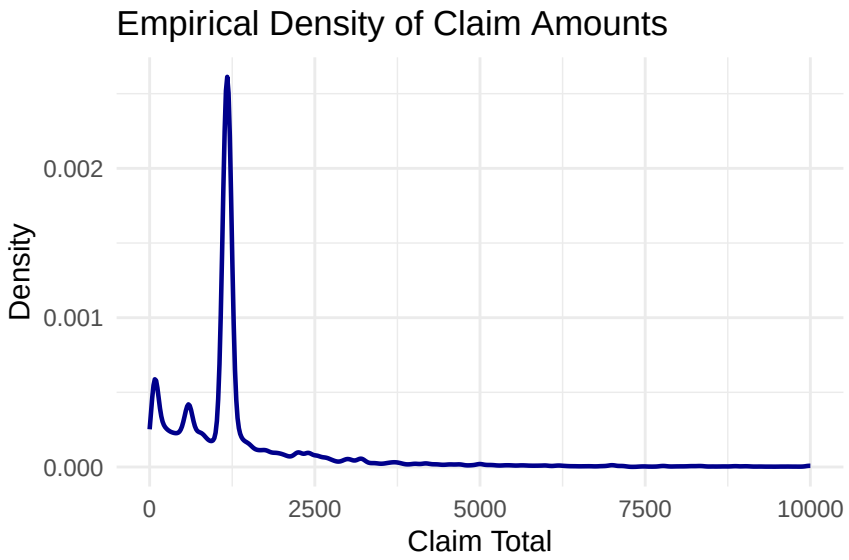

```
qqnorm(rstandard(sevGamma_full))
qqline(rstandard(sevGamma_full),col="red")
```



The residual analysis suggests that the gamma distribution does not fit the data well. (Note that we use standardized deviance residuals here, as the response is continuous.) In fact, if we plot the empirical distribution of claim amounts (as shown below), the French MTPL data appears to have three distinct modes with heavy tails, along with many fixed-size payments, as pointed out in Wüthrich and Merz (2023). This further confirms the unsuitability of the gamma distribution

```
# Create the density plot using ggplot
ggplot(filter(freMTPL2full[train_indices, ], ClaimNb > 0), aes(x = ClaimTotal)) +
  geom_density(color = "darkblue", linewidth = 0.8) +
  labs(title = "Empirical Density of Claim Amounts",
       x = "Claim Total",
       y = "Density") +
  xlim(0, 10000) +
  theme_minimal()
```

```
## Warning: Removed 364 rows containing non-finite values (‘stat_density()‘).
```



Note: We haven't shown the fit of the Inverse Gaussian GLM here, as the model encountered convergence issues with this dataset. You are welcome to explore this further on your own by setting `family = inverse.gaussian(link = "log")` in the `glm` function.

4.1.2 Gamma GLM with Pre-processed Data

Here, we fit a Gamma GLM corresponding to Section 3.5.2 using the same pre-processed data, following the steps in Wüthrich and Merz (2023). In this model, no predictors are dropped compared to the full model. You are encouraged to try it yourself. It is also important to note that you can conduct feature engineering separately or jointly with the frequency model, and the set of predictors used in the severity model can differ from those in the frequency model—this is a major advantage of the frequency-severity approach over pure premium modeling. However, for the sake of time, we use the same predictors as in our frequency modeling.

```
sevGamma_full2 <- glm(ClaimTotal ~ offset(log(ClaimNb)) + VehPowerGLM + VehAgeGLM + DrivAgeGLM +
                        BonusMalusGLM + VehBrand + VehGas + DensityGLM + Region + AreaGLM,
                      family = Gamma(link = "log"),
                      weights = ClaimNb,
                      data = filter(freMTPL2prep[train_indices, ], ClaimNb > 0))

summary(sevGamma_full2)
```

```
##
## Call:
## glm(formula = ClaimTotal ~ offset(log(ClaimNb)) + VehPowerGLM +
##     VehAgeGLM + DrivAgeGLM + BonusMalusGLM + VehBrand + VehGas +
##     DensityGLM + Region + AreaGLM, family = Gamma(link = "log"),
##     data = filter(freMTPL2prep[train_indices, ], ClaimNb > 0),
##     weights = ClaimNb)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5765  -0.9680  -0.5169  -0.2253  19.9152
```

```
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      8.941565   0.575180  15.546  < 2e-16 ***
## VehPowerGLM5                    -0.260472   0.104156  -2.501 0.012401 *
## VehPowerGLM6                    -0.033803   0.101583  -0.333 0.739316
## VehPowerGLM7                    -0.026562   0.100980  -0.263 0.792525
## VehPowerGLM8                    -0.173287   0.143499  -1.208 0.227222
## VehPowerGLM9                     0.024711   0.113047   0.219 0.826969
## VehAgeGLM6-12                   -0.105864   0.068883  -1.537 0.124342
## VehAgeGLM12+                    -0.006002   0.090110  -0.067 0.946898
## DrivAgeGLM21-25                 -1.802384   0.223301  -8.072 7.39e-16 ***
## DrivAgeGLM26-30                 -2.042690   0.217990  -9.371  < 2e-16 ***
## DrivAgeGLM31-40                 -1.988743   0.211609  -9.398  < 2e-16 ***
## DrivAgeGLM41-50                 -2.082229   0.214648  -9.701  < 2e-16 ***
## DrivAgeGLM51-70                 -2.035312   0.214668  -9.481  < 2e-16 ***
## DrivAgeGLM71+                   -1.611195   0.239446  -6.729 1.76e-11 ***
## BonusMalusGLM                    0.002640   0.001857   1.421 0.155281
## VehBrandB10                      0.148839   0.191361   0.778 0.436701
## VehBrandB11                      0.695214   0.197379   3.522 0.000429 ***
## VehBrandB12                      0.113642   0.105204   1.080 0.280064
## VehBrandB13                      0.003662   0.211016   0.017 0.986156
## VehBrandB14                     -0.005588   0.411589  -0.014 0.989168
## VehBrandB2                       0.042247   0.081839   0.516 0.605703
## VehBrandB3                       0.156159   0.114159   1.368 0.171361
## VehBrandB4                       0.226975   0.155523   1.459 0.144464
## VehBrandB5                      -0.022168   0.131782  -0.168 0.866415
## VehBrandB6                      -0.200535   0.149267  -1.343 0.179140
## VehGasRegular                   -0.006849   0.063666  -0.108 0.914332
## DensityGLM                       0.142411   0.067335   2.115 0.034448 *
## RegionAquitaine                  0.272577   0.504563   0.540 0.589050
## RegionAuvergne                  -0.133827   0.623600  -0.215 0.830078
## RegionBasse-Normandie            0.297633   0.532082   0.559 0.575914
## RegionBourgogne                  0.164838   0.548242   0.301 0.763672
## RegionBretagne                   0.343233   0.496370   0.691 0.489269
## RegionCentre                     0.388542   0.488639   0.795 0.426536
## RegionChampagne-Ardenne          1.572786   0.731790   2.149 0.031630 *
## RegionCorse                      1.077408   0.655033   1.645 0.100026
## RegionFranche-Comte              0.021170   0.932892   0.023 0.981895
## RegionHaute-Normandie            0.094126   0.585420   0.161 0.872266
## RegionIle-de-France              0.225874   0.493445   0.458 0.647138
## RegionLanguedoc-Roussillon       0.208207   0.506359   0.411 0.680942
## RegionLimousin                  -0.121840   0.601754  -0.202 0.839549
## RegionMidi-Pyrenees              0.049492   0.543356   0.091 0.927425
## RegionNord-Pas-de-Calais         0.196336   0.500484   0.392 0.694846
## RegionPays-de-la-Loire           0.092302   0.498559   0.185 0.853124
## RegionPicardie                  -0.028648   0.558349  -0.051 0.959080
## RegionPoitou-Charentes           0.064831   0.512276   0.127 0.899294
## RegionProvence-Alpes-Cotes-D'Azur 0.472696   0.491071   0.963 0.335771
## RegionRhone-Alpes                0.314501   0.488983   0.643 0.520120
## AreaGLM                         -0.197204   0.090341  -2.183 0.029058 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 15.8451)
```

```
##
##     Null deviance: 33593  on 17500  degrees of freedom
## Residual deviance: 27479  on 17453  degrees of freedom
## AIC: 321406
##
## Number of Fisher Scoring iterations: 22
```

5 Modeling Aggregate Loss Using GLMs

The Tweedie distribution is defined as a Poisson sum of gamma random variables. Specifically, suppose that N has a Poisson distribution with mean *lambda*, representing the number of claims. Let y_j be an i.i.d. sequence, independent of N, with each y_j having a gamma distribution with parameters α and β, representing the amount of a claim.

Then, $S = y_1 + ... + y_N$ is a Poisson sum of gammas.

5.1 Tweedie GLM for Pure Premiums

5.1.1 Task Solution: Fit a Tweedie GLM (Full Model) Using All Available Training Data Except `IDpol`. Model the Total Claim per Policy as the Target Variable and Include an Offset for Exposure Volume.

```
premTweedie_full <- glm(ClaimTotal ~ offset(log(Exposure)) + VehPower + VehAge + DrivAge + BonusMalus +
                        VehBrand + VehGas + Density + Region + Area,
                        family = tweedie(var.power=1.5, link.power=0),
                       data = freMTPL2full[train_indices,])
summary(premTweedie_full)
```

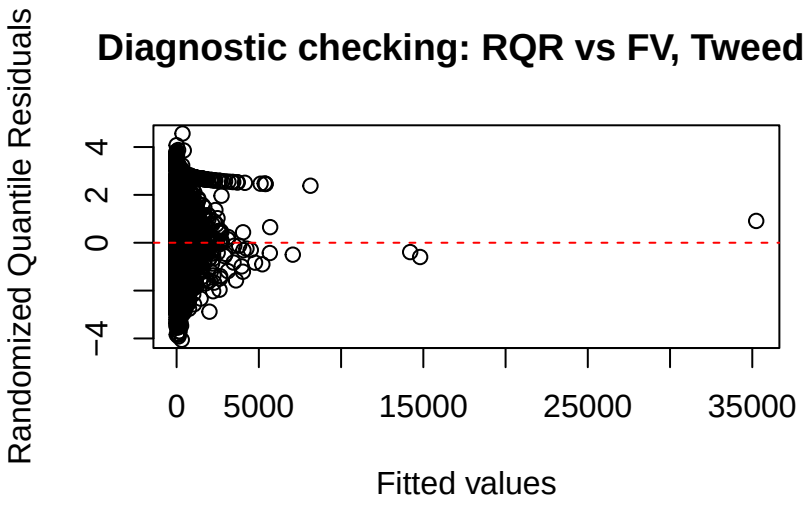
```
##
## Call:
## glm(formula = ClaimTotal ~ offset(log(Exposure)) + VehPower +
##     VehAge + DrivAge + BonusMalus + VehBrand + VehGas + Density +
##     Region + Area, family = tweedie(var.power = 1.5, link.power = 0),
##     data = freMTPL2full[train_indices, ])
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -32.55   -6.82   -5.82   -4.41  891.37
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     1.727e+00  1.434e+00   1.204   0.2285
## VehPower                        7.638e-02  3.558e-02   2.147   0.0318 *
## VehAge                         -1.992e-02  1.414e-02  -1.408   0.1590
## DrivAge                        -6.344e-04  5.392e-03  -0.118   0.9063
## BonusMalus                      3.863e-02  4.082e-03   9.464   <2e-16 ***
## VehBrandB10                     8.795e-02  4.482e-01   0.196   0.8444
## VehBrandB11                     4.905e-01  4.516e-01   1.086   0.2774
## VehBrandB12                    -2.305e-01  2.390e-01  -0.964   0.3348
## VehBrandB13                    -1.010e-01  5.383e-01  -0.188   0.8512
## VehBrandB14                    -3.272e-01  9.849e-01  -0.332   0.7397
## VehBrandB2                      2.234e-01  1.920e-01   1.163   0.2446
```

```
## VehBrandB3                          6.907e-02  2.760e-01   0.250   0.8024
## VehBrandB4                          2.146e-01  3.628e-01   0.592   0.5542
## VehBrandB5                         -1.644e-01  3.411e-01  -0.482   0.6297
## VehBrandB6                         -3.904e-01  3.911e-01  -0.998   0.3181
## VehGasRegular                       1.039e-01  1.425e-01   0.729   0.4660
## Density                            -1.082e-05  5.079e-05  -0.213   0.8312
## RegionAquitaine                     4.681e-01  1.366e+00   0.343   0.7318
## RegionAuvergne                      4.918e-02  1.631e+00   0.030   0.9759
## RegionBasse-Normandie               3.009e-01  1.435e+00   0.210   0.8339
## RegionBourgogne                     2.023e-01  1.464e+00   0.138   0.8900
## RegionBretagne                      3.773e-01  1.352e+00   0.279   0.7802
## RegionCentre                        7.523e-01  1.331e+00   0.565   0.5720
## RegionChampagne-Ardenne             2.722e+00  1.479e+00   1.840   0.0657 .
## RegionCorse                         7.980e-01  1.584e+00   0.504   0.6143
## RegionFranche-Comte                 2.222e-01  2.203e+00   0.101   0.9196
## RegionHaute-Normandie              -1.518e-02  1.533e+00  -0.010   0.9921
## RegionIle-de-France                 5.001e-01  1.347e+00   0.371   0.7104
## RegionLanguedoc-Roussillon          3.600e-01  1.366e+00   0.264   0.7922
## RegionLimousin                      1.144e-01  1.663e+00   0.069   0.9451
## RegionMidi-Pyrenees                -1.552e-01  1.437e+00  -0.108   0.9140
## RegionNord-Pas-de-Calais            2.924e-01  1.357e+00   0.215   0.8294
## RegionPays-de-la-Loire              3.979e-01  1.354e+00   0.294   0.7689
## RegionPicardie                     -4.926e-03  1.509e+00  -0.003   0.9974
## RegionPoitou-Charentes              1.875e-01  1.395e+00   0.134   0.8931
## RegionProvence-Alpes-Cotes-D'Azur   7.054e-01  1.336e+00   0.528   0.5976
## RegionRhone-Alpes                   8.220e-01  1.334e+00   0.616   0.5377
## AreaB                               4.317e-01  2.705e-01   1.596   0.1105
## AreaC                               1.049e-01  2.351e-01   0.446   0.6555
## AreaD                               2.747e-01  2.526e-01   1.087   0.2768
## AreaE                               4.306e-01  3.388e-01   1.271   0.2038
## AreaF                               3.616e-01  1.219e+00   0.297   0.7667
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Tweedie family taken to be 20894.17)
##
##     Null deviance: 33902147  on 474603  degrees of freedom
## Residual deviance: 30162455  on 474562  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 9
```

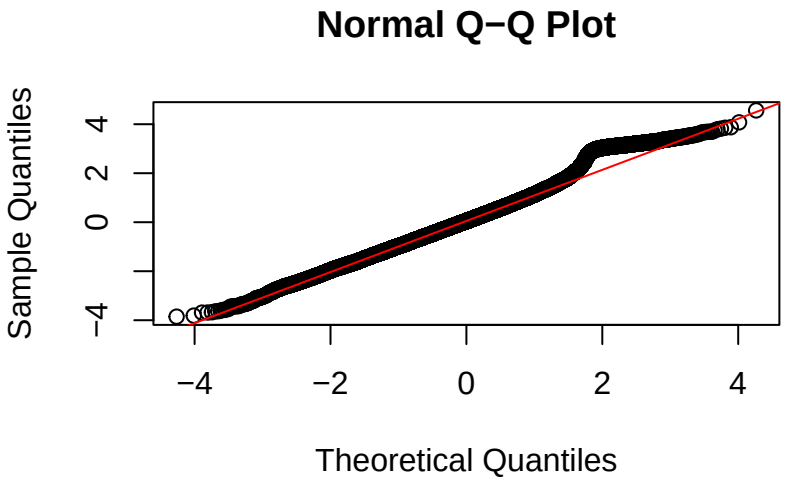
```
set.seed(823)
plot(premTweedie_full$fitted.values[sample_indices],qresiduals(premTweedie_full)[sample_indices],
     xlab="Fitted values",
     ylab="Randomized Quantile Residuals",
     main="Diagnostic checking: RQR vs FV, Tweedie")
```

```
## Loading required namespace: tweedie
```

```
abline(h=0,col="red",lty=2)
```




```
qqnorm(qresiduals(premTweedie_full)[sample_indices])
qqline(qresiduals(premTweedie_full)[sample_indices],col="red")
```



6 Model Validation and Evaluation

In this section, we summarize both quantitative measures and visual plotting techniques that can be used for model evaluation and comparison.

6.1 Measures of Model Performance

In Section 3.6, we use AIC and Poisson deviance to compare claim frequency models (there are other deviance loss functions, such as Gamma or Tweedie deviance loss, that you are welcome to explore on your own!).

However, you need to make sure to use the correct deviance loss function to evaluate the corresponding models. Note, models are comparable if their deviance loss function is the same (see a similar discussion in Section 6.1.3 of Generalized Linear Models for Insurance Rating).

In addition to these metrics, we use RMSE (Selvakumar et al. (2021)) and Spearman correlation (Frees, Lee, and Yang (2016), Lee and Shi (2019)) in this section to evaluate prediction performance. Other measures, such as the Gini index, will be explored in the following section.

```
freMTPL2fulltest <- freMTPL2full[-train_indices, ]
freMTPL2fulltest2 <- freMTPL2prep[-train_indices, ]
freMTPL2fulltest$AClaimNb <- freMTPL2full[-train_indices, ]$ClaimNb #save actual ClaimNb
freMTPL2fulltest2$AClaimNb <- freMTPL2prep[-train_indices, ]$ClaimNb

freMTPL2fulltest$ClaimNb <- predict(freqPoisson_full, newdata = freMTPL2fulltest, type = "response")
freMTPL2fulltest$predcost <- predict(sevGamma_full, newdata = freMTPL2fulltest, type = "response")

freMTPL2fulltest2$ClaimNb <- predict(freqPoisson_full2, newdata = freMTPL2fulltest2, type = "response")
freMTPL2fulltest2$predcost2 <- predict(sevGamma_full2, newdata = freMTPL2fulltest2, type = "response")

freMTPL2fulltest$ClaimNb_cost <- freMTPL2fulltest$ClaimNb
freMTPL2fulltest2$ClaimNb_cost <- freMTPL2fulltest2$ClaimNb

#RMSE
RMSE(freMTPL2fulltest$ClaimNb, freMTPL2full[-train_indices, ]$ClaimNb)
```

```
## [1] 0.2022986
```

```
RMSE(freMTPL2fulltest2$ClaimNb, freMTPL2prep[-train_indices, ]$ClaimNb)
```

```
## [1] 0.2020836
```

```
RMSE(freMTPL2fulltest$predcost[freMTPL2fulltest$AClaimNb > 0] /
       freMTPL2fulltest$ClaimNb[freMTPL2fulltest$AClaimNb > 0],
     freMTPL2full[-train_indices, ]$ClaimTotal[freMTPL2full[-train_indices, ]$ClaimNb > 0] /
       freMTPL2full[-train_indices, ]$ClaimNb[freMTPL2full[-train_indices, ]$ClaimNb > 0])
```

```
## [1] 8579.95
```

```
RMSE(freMTPL2fulltest2$predcost2[freMTPL2fulltest2$AClaimNb > 0] /
       freMTPL2fulltest2$ClaimNb[freMTPL2fulltest2$AClaimNb > 0],
     freMTPL2prep[-train_indices, ]$ClaimTotal[freMTPL2prep[-train_indices, ]$ClaimNb > 0] /
       freMTPL2prep[-train_indices, ]$ClaimNb[freMTPL2prep[-train_indices, ]$ClaimNb > 0])
```

```
## [1] 8796.3
```

```
RMSE(freMTPL2fulltest$predcost, freMTPL2full[-train_indices, ]$ClaimTotal)
```

```
## [1] 1697.361
```

```
RMSE(freMTPL2fulltest2$predcost2, freMTPL2prep[-train_indices, ]$ClaimTotal)
```

```
## [1] 1706.69
```

```r
# Spearman correlation
# frequency
cor(freMTPL2fulltest$ClaimNb,freMTPL2full[-train_indices, ]$ClaimNb, method = "spearman")
```

```
## [1] 0.1247832
```

```r
cor(freMTPL2fulltest2$ClaimNb,freMTPL2prep[-train_indices, ]$ClaimNb, method = "spearman")
```

```
## [1] 0.1271853
```

```r
# severity
cor(freMTPL2fulltest$predcost[freMTPL2fulltest$AClaimNb > 0] /
       freMTPL2fulltest$ClaimNb[freMTPL2fulltest$AClaimNb > 0],
     freMTPL2full[-train_indices, ]$ClaimTotal[freMTPL2full[-train_indices, ]$ClaimNb > 0] /
       freMTPL2full[-train_indices, ]$ClaimNb[freMTPL2full[-train_indices, ]$ClaimNb > 0]
    , method = "spearman")
```

```
## [1] 0.01733511
```

```r
cor(freMTPL2fulltest2$predcost2[freMTPL2fulltest2$AClaimNb > 0] /
       freMTPL2fulltest2$ClaimNb[freMTPL2fulltest2$AClaimNb > 0],
     freMTPL2prep[-train_indices, ]$ClaimTotal[freMTPL2prep[-train_indices, ]$ClaimNb > 0] /
       freMTPL2prep[-train_indices, ]$ClaimNb[freMTPL2prep[-train_indices, ]$ClaimNb > 0]
    , method = "spearman")
```

```
## [1] 0.04168927
```

```r
# pure premium
cor(freMTPL2fulltest$predcost,freMTPL2full[-train_indices, ]$ClaimTotal, method = "spearman")
```

```
## [1] 0.1177166
```

```r
cor(freMTPL2fulltest2$predcost2,freMTPL2prep[-train_indices, ]$ClaimTotal, method = "spearman")
```

```
## [1] 0.1213769
```

Please note that when predicting claim counts, the predicted values are already adjusted for the actual `Exposure` information, which is included as an offset in the Poisson frequency model. Similarly, when modeling the total claim size in `sevGamma_full` and `sevGamma_full2`, we need to adjust the predicted claim counts from `freqPoisson_full` and `freqPoisson_full2` to obtain the predicted claim costs. Specifically, the predicted cost is derived from model 2, after adjusting or offsetting the frequency model's prediction (i.e., by replacing the `ClaimNb` column in the test data with the predicted claim count from `freqPoisson_full` and `freqPoisson_full2`). If we model the average claim cost in `sevGamma_full` and `sevGamma_full2`, the predicted cost is obtained by multiplying the frequency model's prediction by the severity model's prediction.

From the metrics above, `freqPoisson_full2` outperforms `freqPoisson_full` across all metrics, and the combination of frequency-severity modeling performs better on the pre-processed data than on the original data. However, it is unclear whether `sevGamma_full2` outperforms `sevGamma_full1`.

6.2 Gini Curve (Measure the Lift)

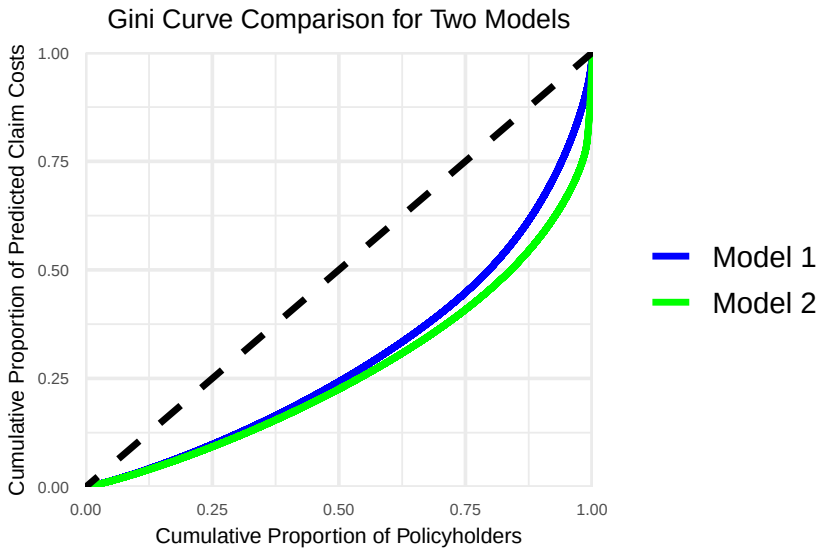
The Gini index can be used to measure the lift of an insurance rating plan by quantifying its ability to segment the population into the best and worst risks. Here, lift refers to a model's ability to charge each insured an actuarially fair rate, thereby minimizing the potential for adverse selection. This is a relative concept and requires two or more competing models. Following Section 7.2 of Generalized Linear Models for Insurance Rating, the Gini index for a model that creates a rating plan is calculated as follows:

1. Sort the dataset based on the model predicted loss cost. The records at the top of the dataset are then the risks which the model believes are best, and the records at the bottom of the dataset are the risks which the model believes are worst.
2. On the x-axis, plot the cumulative percentage of exposures.
3. On the y-axis, plot the cumulative percentage of losses.

In the code below, we modify this procedure by using the number of policies instead of the cumulative percentage of exposure. This is achieved by assigning an exposure of 1 to all policies, due to the presence of a significant number of very small exposures in the dataset. Please note that, typically, exposure (rather than policy count) is preferred.

```
freMTPL2fulltest$ClaimNb <- freMTPL2full[-train_indices, ]$ClaimNb #restore actual ClaimNb
freMTPL2fulltest2$ClaimNb <- freMTPL2prep[-train_indices, ]$ClaimNb
freMTPL2fulltest$AExposure <- freMTPL2fulltest$Exposure #save actual exposure
freMTPL2fulltest2$AExposure <- freMTPL2fulltest2$Exposure
freMTPL2fulltest$Exposure <- 1
freMTPL2fulltest2$Exposure <- 1

freMTPL2fulltest$ClaimNb <- predict(freqPoisson_full, newdata = freMTPL2fulltest, type = "response")
freMTPL2fulltest$predprem <- predict(sevGamma_full, newdata = freMTPL2fulltest, type = "response")

freMTPL2fulltest2$ClaimNb <- predict(freqPoisson_full2, newdata = freMTPL2fulltest2, type = "response")
freMTPL2fulltest2$predprem2 <- predict(sevGamma_full2, newdata = freMTPL2fulltest2, type = "response")

# Sort the data by predicted claim cost for both models
freMTPL2fulltest <- freMTPL2fulltest[order(freMTPL2fulltest$predprem), ]
freMTPL2fulltest2 <- freMTPL2fulltest2[order(freMTPL2fulltest2$predprem2), ]

# Calculate the cumulative sum of predicted claim costs and the cumulative proportion of
# observations for both models
freMTPL2fulltest$cumPredictedprem <- cumsum(freMTPL2fulltest$predprem) / sum(freMTPL2fulltest$predprem)
freMTPL2fulltest$cumObservations <- seq_along(freMTPL2fulltest$predprem) / nrow(freMTPL2fulltest)

freMTPL2fulltest2$cumPredictedprem <- cumsum(freMTPL2fulltest2$predprem2) / sum(freMTPL2fulltest2$predpre
freMTPL2fulltest2$cumObservations <- seq_along(freMTPL2fulltest2$predprem2) / nrow(freMTPL2fulltest2)

# Combine the data for both models into a single data frame for plotting
gini_data_full1 <- data.frame(
  cumObservations = freMTPL2fulltest$cumObservations,
  cumPredictedprem = freMTPL2fulltest$cumPredictedprem,
  Model = "Model 1"
)

gini_data_full2 <- data.frame(
  cumObservations = freMTPL2fulltest2$cumObservations,
```

```
  cumPredictedprem = freMTPL2fulltest2$cumPredictedprem,
  Model = "Model 2"
)

# Combine both datasets into one for plotting
gini_data_combined <- rbind(gini_data_full1, gini_data_full2)

# Create the Gini curve plot for both models with custom colors
ggplot(gini_data_combined, aes(x = cumObservations, y = cumPredictedprem, color = Model)) +
  geom_line(size = 1.2) +  # Set same size for both Gini curves
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "black", size = 1.2) +
  # Line of equality
  scale_x_continuous(limits = c(0, 1), expand = c(0, 0)) +
  # Ensure red line starts at (0, 0) and ends at (1, 1)
  scale_y_continuous(limits = c(0, 1), expand = c(0, 0)) +
  labs(title = "Gini Curve Comparison for Two Models",
       x = "Cumulative Proportion of Policyholders",
       y = "Cumulative Proportion of Predicted Claim Costs") +
  scale_color_manual(values = c("blue", "green")) +  # Custom colors for the models
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 10),
    axis.title = element_text(size = 8),
    axis.text = element_text(size = 6),
    legend.title = element_blank()  # Remove legend title
  )
```




```
# Function to calculate Gini Index correctly
gini_index <- function(cumObs, cumPred) {
  # Use trapezoidal rule to approximate area under the Lorenz curve
  auc <- sum((cumObs[-1] - cumObs[-length(cumObs)]) * (cumPred[-1] + cumPred[-length(cumPred)]) / 2)

  # The Gini coefficient is 1 minus twice the area under the Lorenz curve
  gini <- 1 - 2 * auc
  return(gini)
```

```
}

# Apply the function to your data
gini_full <- gini_index(freMTPL2fulltest$cumObservations, freMTPL2fulltest$cumPredictedprem)
print(gini_full)
```

```
## [1] 0.4053724
```

```
gini_full2 <- gini_index(freMTPL2fulltest2$cumObservations, freMTPL2fulltest2$cumPredictedprem)
print(gini_full2)
```

```
## [1] 0.4624761
```

```
#Alternatively, this can be computed using ineq package
library(ineq)
```

```
##
## Attaching package: 'ineq'

## The following object is masked from 'package:MLmetrics':
##
##     Gini
```

```
# Compute the Gini index
gini2_full <- ineq(freMTPL2fulltest$predprem, type = "Gini")
print(gini2_full)
```

```
## [1] 0.4053724
```

```
gini2_full2 <- ineq(freMTPL2fulltest2$predprem2, type = "Gini")
print(gini2_full2)
```

```
## [1] 0.4624761
```

Also, note that the Gini index on validation data can be used to compare a model's ranking performance, but not its calibration. The combination of frequency-severity modeling performs better on the pre-processed data than on the original data, as a larger Gini index suggests that the model segments risks more effectively by showing a greater difference between the worst and best risks.

6.3 Quantile Plot

Quantile plots can be used to visualize a model's ability to accurately differentiate between the best and worst risks. A common choice is the decile plot, which divides predictions into ten bins with equal exposure and contrasts actual versus modeled outcomes. Here, we focus specifically on claim frequency predictions (note that it can also be applied to pure premium predictions). The decile plot can help evaluate the following (Goldburd et al. 2016):

1) **Predictive Accuracy (Model Calibration)**: This refers to how well a model predicts the insurance outcome in each quantile. A well-calibrated model will show small differences between actual and predicted values in each bin, with no significant over- or underestimation in the top or bottom bins.

2) **Model Fit**: A model achieves a good fit if the ratio between the top and bottom bins is high, representing a large difference (or lift) between the groups we believe to have either the best or worst experience.

3) **Monotonicity**: By definition, the predicted insurance outcome should monotonically increase as the quantile increases. Ideally, the actual insurance outcome should also increase, although small reversals are acceptable.

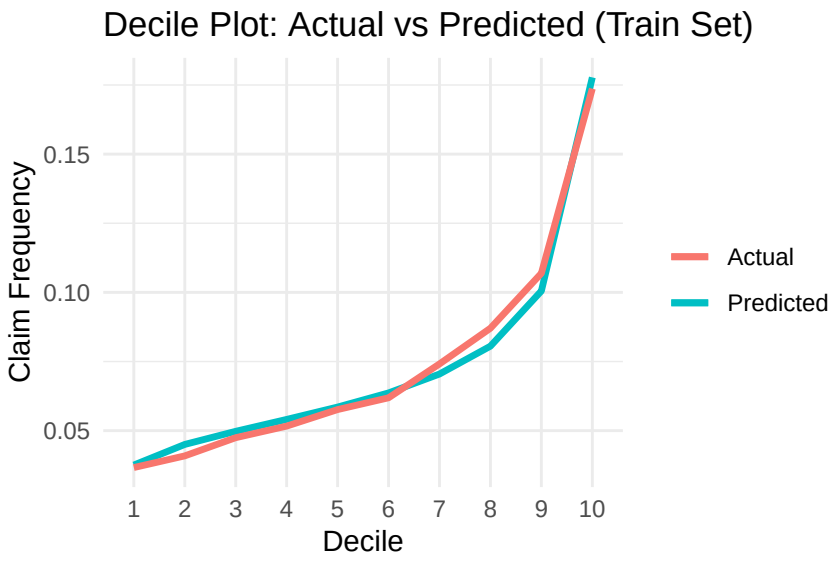
4) **Overfitting**: The plot can check for overfitting by assessing whether the model's performance deteriorates significantly between the training and testing data.

```
#Compute In-sample Predictions
freMTPL2fulltrain <- freMTPL2full[train_indices, ]
freMTPL2fulltrain2 <- freMTPL2prep[train_indices, ]
freMTPL2fulltrain$AClaimNb <- freMTPL2full[train_indices, ]$ClaimNb #save actual ClaimNb
freMTPL2fulltrain2$AClaimNb <- freMTPL2prep[train_indices, ]$ClaimNb

freMTPL2fulltrain$ClaimNb <- predict(freqPoisson_full, newdata = freMTPL2fulltrain, type = "response")
freMTPL2fulltrain$predcost <- predict(sevGamma_full, newdata = freMTPL2fulltrain, type = "response")
freMTPL2fulltrain2$ClaimNb <- predict(freqPoisson_full2, newdata = freMTPL2fulltrain2, type = "response")
freMTPL2fulltrain2$predcost2 <- predict(sevGamma_full2, newdata = freMTPL2fulltrain2, type = "response")

freMTPL2fulltrain$ClaimNb_cost <- freMTPL2fulltrain$ClaimNb
freMTPL2fulltrain$ClaimNb  <- freMTPL2fulltrain$AClaimNb
freMTPL2fulltrain2$ClaimNb_cost <- freMTPL2fulltrain2$ClaimNb
freMTPL2fulltrain2$ClaimNb  <- freMTPL2fulltrain2$AClaimNb
```

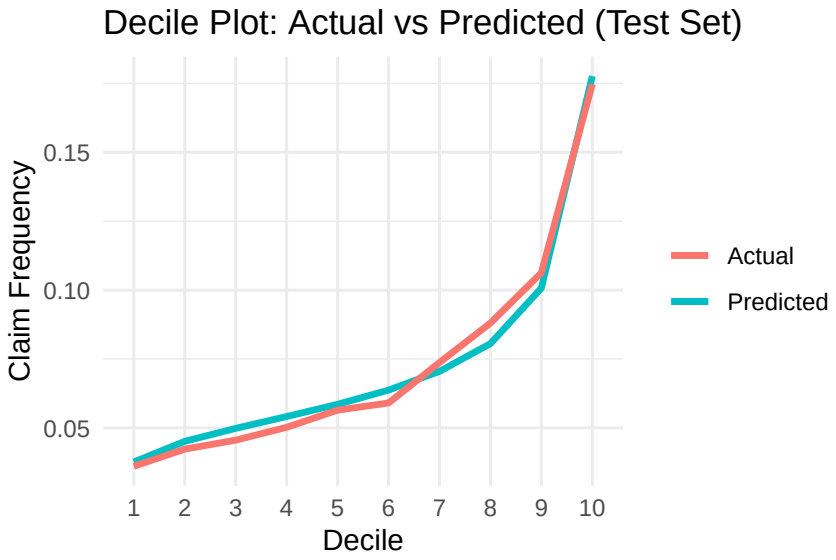
Model 1 (`freqPoisson_full`)

```
decile_fulltrain <- freMTPL2fulltrain
decile_fulltest <- freMTPL2fulltest
decile_fulltrain2 <- freMTPL2fulltrain2
decile_fulltest2 <- freMTPL2fulltest2

decile_fulltest$Exposure <- decile_fulltest$AExposure
decile_fulltest2$Exposure <- decile_fulltest2$AExposure

# Step 1: Sort the dataset by predicted claim frequency
freMTPL2fulltrain_sorted <- decile_fulltrain %>%
  arrange(ClaimNb_cost/Exposure)

freMTPL2fulltest_sorted <- decile_fulltest %>%
  arrange(ClaimNb_cost/Exposure)

# Step 2: Calculate cumulative exposure and split it into deciles manually to ensure
# equal exposure in each decile
total_exposure_train <- sum(freMTPL2fulltrain_sorted$Exposure)
total_exposure_test <- sum(freMTPL2fulltest_sorted$Exposure)

# Step 3: Use cut() to divide the cumulative exposure into deciles where exposure is roughly equal
freMTPL2fulltrain_sorted <- freMTPL2fulltrain_sorted %>%
  mutate(cum_exposure = cumsum(Exposure),
         decile = cut(cum_exposure, breaks = seq(0, total_exposure_train, length.out = 11),
                      include.lowest = TRUE, labels = FALSE))

freMTPL2fulltest_sorted <- freMTPL2fulltest_sorted %>%
```

```r
  mutate(cum_exposure = cumsum(Exposure),
         decile = cut(cum_exposure, breaks = seq(0, total_exposure_test, length.out = 11),
                      include.lowest = TRUE, labels = FALSE))

# Step 4: Calculate average predicted and actual claim frequencies within each decile
decile_summarytrain <- freMTPL2fulltrain_sorted %>%
  group_by(decile) %>%
  summarise(
    avg_predicted_claimnb = sum(ClaimNb_cost)/sum(Exposure),
    avg_actual_claimnb = sum(AClaimNb)/sum(Exposure)
  )

decile_summarytest <- freMTPL2fulltest_sorted %>%
  group_by(decile) %>%
  summarise(
    avg_predicted_claimnb = sum(ClaimNb_cost)/sum(Exposure),
    avg_actual_claimnb = sum(AClaimNb)/sum(Exposure)
  )

# Step 5: Plot actual vs predicted claim frequencies for each decile
# Plot for Train Set
ggplot(decile_summarytrain, aes(x = factor(decile))) +
  geom_line(aes(y = avg_predicted_claimnb, group = 1, color = "Predicted"), size = 1.2) +
  geom_line(aes(y = avg_actual_claimnb, group = 1, color = "Actual"), size = 1.2) +
  labs(x = "Decile", y = "Claim Frequency", title = "Decile Plot: Actual vs Predicted (Train Set)") +
  theme_minimal() +
  guides(color = guide_legend(title = NULL))
```



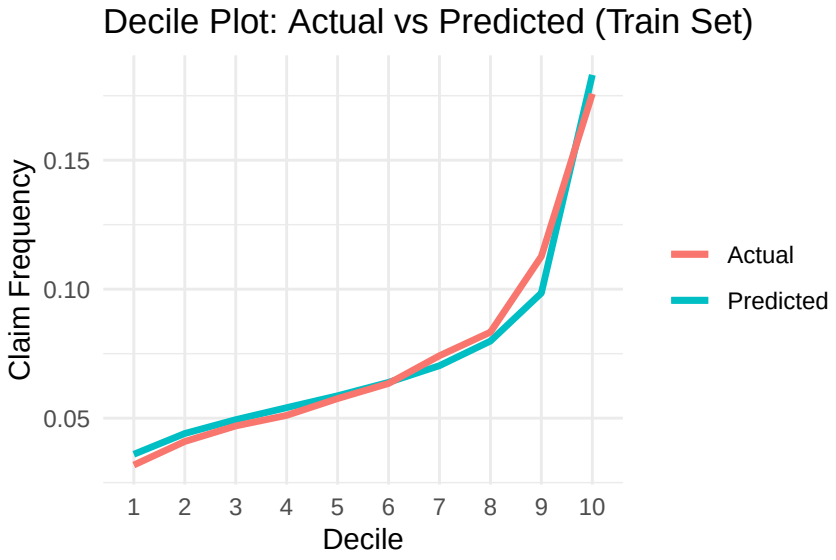

```r
# Plot for Test Set
ggplot(decile_summarytest, aes(x = factor(decile))) +
  geom_line(aes(y = avg_predicted_claimnb, group = 1, color = "Predicted"), size = 1.2) +
  geom_line(aes(y = avg_actual_claimnb, group = 1, color = "Actual"), size = 1.2) +
  labs(x = "Decile", y = "Claim Frequency", title = "Decile Plot: Actual vs Predicted (Test Set)") +
  theme_minimal() +
  guides(color = guide_legend(title = NULL))
```



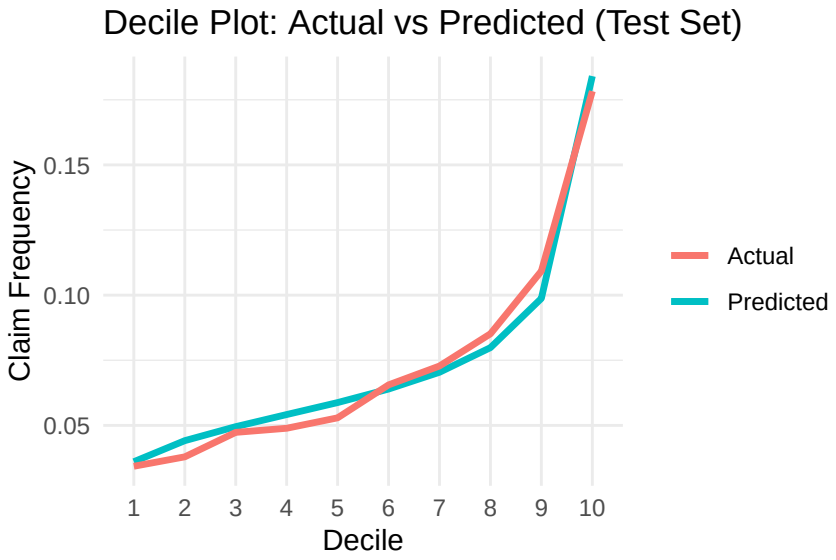
Model 2 (`freqPoisson_full2`)

```
# Step 1: Sort the dataset by predicted claim frequency
freMTPL2fulltrain2_sorted <- decile_fulltrain2 %>%
  arrange(ClaimNb_cost/Exposure)

freMTPL2fulltest2_sorted <- decile_fulltest2 %>%
  arrange(ClaimNb_cost/Exposure)

# Step 2: Calculate cumulative exposure and split it into deciles manually to ensure
# equal exposure in each decile
total_exposure_train2 <- sum(freMTPL2fulltrain2_sorted$Exposure)
total_exposure_test2 <- sum(freMTPL2fulltest2_sorted$Exposure)

# Step 3: Use cut() to divide the cumulative exposure into deciles where exposure is roughly equal
freMTPL2fulltrain2_sorted <- freMTPL2fulltrain2_sorted %>%
  mutate(cum_exposure = cumsum(Exposure),
         decile = cut(cum_exposure, breaks = seq(0, total_exposure_train, length.out = 11),
                      include.lowest = TRUE, labels = FALSE))

freMTPL2fulltest2_sorted <- freMTPL2fulltest2_sorted %>%
  mutate(cum_exposure = cumsum(Exposure),
         decile = cut(cum_exposure, breaks = seq(0, total_exposure_test, length.out = 11),
                      include.lowest = TRUE, labels = FALSE))

# Step 4: Calculate average predicted and actual claim frequencies within each decile
decile_summarytrain2 <- freMTPL2fulltrain2_sorted %>%
  group_by(decile) %>%
  summarise(
    avg_predicted_claimnb = sum(ClaimNb_cost)/sum(Exposure),
    avg_actual_claimnb = sum(AClaimNb)/sum(Exposure)
  )

decile_summarytest2 <- freMTPL2fulltest2_sorted %>%
  group_by(decile) %>%
  summarise(
```

```
    avg_predicted_claimnb = sum(ClaimNb_cost)/sum(Exposure),
    avg_actual_claimnb = sum(AClaimNb)/sum(Exposure)
  )

# Step 5: Plot actual vs predicted claim frequencies for each decile
# Plot for Train Set
ggplot(decile_summarytrain2, aes(x = factor(decile))) +
  geom_line(aes(y = avg_predicted_claimnb, group = 1, color = "Predicted"), size = 1.2) +
  geom_line(aes(y = avg_actual_claimnb, group = 1, color = "Actual"), size = 1.2) +
  labs(x = "Decile", y = "Claim Frequency", title = "Decile Plot: Actual vs Predicted (Train Set)") +
  theme_minimal() +
  guides(color = guide_legend(title = NULL))
```




```
# Plot for Test Set
ggplot(decile_summarytest2, aes(x = factor(decile))) +
  geom_line(aes(y = avg_predicted_claimnb, group = 1, color = "Predicted"), size = 1.2) +
  geom_line(aes(y = avg_actual_claimnb, group = 1, color = "Actual"), size = 1.2) +
  labs(x = "Decile", y = "Claim Frequency", title = "Decile Plot: Actual vs Predicted (Test Set)") +
  theme_minimal() +
  guides(color = guide_legend(title = NULL))
```



Both models fit well from the three perspectives we discussed.

Exercise: In addition to the Gini index introduced above, there are other techniques that can be used to directly compare two candidate models. Please read Section 7 of Generalized Linear Models for Insurance Rating, with a particular focus on actual vs. predicted plots [a], simple quantile plots and double lift charts.

[a] You can also plot this across all categories of the variable of interest by comparing the average actual values with the model predictions for each category.

7 Appendix

7.1 Chi-square goodness of fit test on frequency data

When an analyst attempts to fit a statistical model to observed data, he or she may wonder how well the model actually reflects the data. How "close" are the observed values to those which would be expected under the fitted model? One statistical test that addresses this issue is the chi-square goodness of fit test. Chi-Square Goodness of Fit Test

$$\chi^2 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$$

Chi-square goodness of fit statistics can be used to compare different models. If the computed test statistic is large, then the observed and expected values are not close and the model is a poor fit to the data.

Reference

Frees, Edward W, Gee Lee, and Lu Yang. 2016. "Multivariate Frequency-Severity Regression Models in Insurance." *Risks* 4 (1): 4.

Frees, Edward W, Glenn Meyers, Richard A Derrig, et al. 2014. "Predictive Modeling Applications in Actuarial Science. Volume 2, Case Studies in Insurance." Cambridge University Press.

Goldburd, Mark, Anand Khare, Dan Tevet, and Dmitriy Guller. 2016. "Generalized Linear Models for Insurance Rating." *Casualty Actuarial Society, CAS Monographs Series* 5: 77.

Lee, Gee Y, and Peng Shi. 2019. "A Dependent Frequency–Severity Approach to Modeling Longitudinal Insurance Claims." *Insurance: Mathematics and Economics* 87: 115–29.

Schelldorfer, Jürg, and Mario V Wuthrich. 2019. "Nesting Classical Actuarial Models into Neural Networks." *Available at SSRN 3320525*.

Selvakumar, V, Dipak Kumar Satpathi, PP Kumar, and VV Haragopal. 2021. "Predictive Modeling of Insurance Claims Using Machine Learning Approach for Different Types of Motor Vehicles." *Accounting and Finance* 9 (1): 1–14.

Wüthrich, Mario V, and Michael Merz. 2023. *Statistical Foundations of Actuarial Learning and Its Applications*. Springer Nature.