Micro-Level Reserving for Claim Count Data

Inaugural-Dissertation

zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Alexander Rosenstock
aus Düsseldorf

Düsseldorf, April 2024

aus dem Mathematischen Institut
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Berichterstatter:

1. Prof. Dr. Axel Bücher

2. Prof. Dr. Matthias Scherer

Tag der mündlichen Prüfung: 11. April 2024

Contents

1 Introduction **11**

2 Included Articles **19**

2.1 Micro-level prediction of outstanding claim counts based on novel mixture models and neural networks 20

2.2 Combined modelling of micro-level outstanding claim counts and individual claim frequencies in general insurance 70

2.3 Fitting Distributions and Neural Networks to Censored and Truncated Data: The R Package reservr 97

3 Outlook **131**

Author contribution statement **135**

Abstract

Determining the ultimate loss of all claims occurred in an accident year is of primal importance in actuarial practice. Widely used methods to determine these losses work on so-called loss triangles, contracting the available information to a tally of all payments for a particular accident year and development year combination. These triangles, while easier to work with than individual claims data, contain only a fraction of the information available to an insurer at a specific point in time, resulting in subpar predictions. With the progress of computational power and the advent of advanced analytical methods, research interest in more granular claim reserving methods has picked up in recent years, going back to Norberg [12]. For the subtask of determining the ultimate claim counts, a micro-level model with individual claims and individual contracts as the smallest unit of observation is developed. The model is used to derive a set of micro-level predictors for claim counts which are analyzed in a large scale simulation study and on a real-world general insurance dataset, showing increased out-of-sample accuracy on real-world data and increased robustness to violations of basic model assumptions on synthetic data.

When analyzing observable insurance data the random truncation inherent in the observations must be taken into account. Methods to perform parameter estimation and distribution learning for arbitrary distributions in this setting are developed. Accompanying software in the form of an R package available on CRAN enables swift application of the methods.

Acknowledgements

First, I would like to thank Prof. Dr. Axel Bücher for his invaluable support and guidance during the course of my studies. Thank you for numerous fruitful discussions, for your enthusiasm, and for your spontaneity in finding time for them.

I am grateful to Zouhair Haddou-Temsamani for providing me with the opportunity to pursue my research interests.

Thanks to ARAG SE for the computational infrastructure and real-world data entrusted to me, enabling me to field-test my research, and to the Centre for Information and Media Technology at Heinrich Heine University Düsseldorf for computational infrastructure and support for my large-scale simulation studies.

Further, thanks to my current and former colleagues both from the Chair of Mathematical Statistics and Probability Theory and at ARAG SE for the friendly working atmosphere and interesting discussions.

Finally, I wish to thank my wife Anna. Without your continuous support I would not be where I stand today.

1 Introduction

Modelling the number of incurred but not reported (IBNR) claims is a central problem in actuarial loss reserving. When combined with models for losses incurred during settlement of reported but not settled (RBNS) claims and losses for IBNR claims, one obtains a model of the ultimate loss, a driving factor in the financial reserves required by an insurance company under general accounting principles, such as the Solvency II framework. Actuarial pricing using a frequency-severity approach also requires estimation of ultimate claim numbers as part of the claims frequency modelling. Accurate prediction of claim counts is thus of primal importance in securing financial stability and profitability of insurance companies. An overview of established actuarial loss reserving methods can be found in Radtke, Schmidt, and Schnaus [15].

Insurance operations generate a large amount of structured data during claims handling. Each claim is linked to a policy with contract features $x^{(i)} \in \mathfrak{X}$, such as the coverage period and choice of deductible. The policies at risk, and their features $x^{(i)}$, are known beforehand by the insurer. In addition to that, upon its reporting to the insurer at time $T^{(i)}_{j,\text{report}}$, additional information about the j-th claim of the i-th policy becomes available to the insurer. Among this is the accident time $T^{(i)}_j$ and additional information about the claim, $Y^{(i)}_j \in \mathfrak{Y}$, such as a claims code. Classical reserving methods aggregate this data into so-called loss triangles, typically split by manually selected, sufficiently homogeneous subsets of $\mathfrak{X} \times \mathfrak{Y}$ and discretized by accident period and development period. Development triangles for a subset $A \subset \mathfrak{X} \times \mathfrak{Y}$ can be defined for claim counts as well as for payments. We introduce loss triangles with notation similar to Radtke, Schmidt, and Schnaus [15]. For claim counts, the incremental triangle $N^{(A)}$ is defined by

$$N^{(A)}_{k,l} \coloneqq \#\big\{i, j \;\big|\; (x^{(i)}, Y^{(i)}_j) \in A, \lfloor T^{(i)}_j \rfloor = k, \lfloor T^{(i)}_{j,\text{report}} \rfloor - \lfloor T^{(i)}_j \rfloor = l\big\}.$$

$0 \le k < \lfloor \tau \rfloor$ is the index for the accident period and $0 \le l < \lfloor \tau \rfloor$ denotes the development period, the difference between the accident period and the calendar period in which the claim is reported. It is called a triangle, because at calendar time τ, the insurer can observe all data where $0 \le k + l = \lfloor T^{(i)}_{j,\text{report}} \rfloor < \lfloor \tau \rfloor$, i.e., the upper left triangle of $N^{(A)}$. See Figure 1.1 for an illustration of $N^{(A)}$.

Aggregate reserving methods aim to complete this development triangle to a square by predicting entries below the observable diagonal $N^{(A)}_{k,l}$ for $k + l \ge \lfloor \tau \rfloor$. Note that in contrast to $N_{k,l}$ from Radtke, Schmidt, and Schnaus [15] (indexed by i, j therein), we allow restriction to a subset A of the available micro-level feature data. In practice, these subsets could be, for example, restrictions to certain lines of business or claims codes. When modelled independently across different sub-portfolios, these restrictions are equivalent to treating each sub-portfolio of policies and claims as a separate reserving problem. The number of IBNR

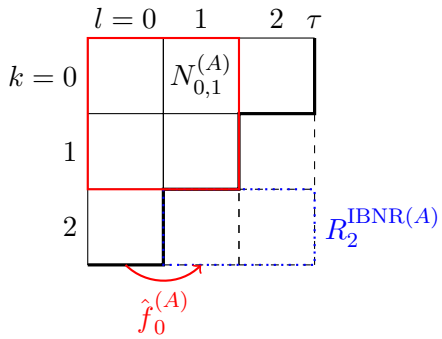


Figure 1.1: Illustration of $N_{i,j}^{(A)}$ for $\tau = 3$, the solid grid cells are observable, while the dashed grid cells $N_{1,2}^{(A)}, N_{2,1}^{(A)}$ and $N_{2,2}^{(A)}$ are not. $R_2^{\text{IBNR}(A)}$ is illustrated as a blue rectangle. $\hat{f}_0^{(A)}$ is illustrated a red rectangle, whose values are used in the Chain Ladder estimate of the first link ratio $\hat{f}_0^{(A)} = \frac{N_{0,0}^{(A)}+N_{0,1}^{(A)}+N_{1,0}^{(A)}+N_{1,1}^{(A)}}{N_{0,0}^{(A)}+N_{1,0}^{(A)}}$.

claims by accident period, $R_k^{\text{IBNR}(A)}$, is then defined by

$$R_k^{\text{IBNR}(A)} := \sum_{l=\lfloor\tau\rfloor-k}^{\lfloor\tau\rfloor-1} N_{k,l}^{(A)},$$

where it is assumed that development is complete after at most τ development periods, i.e. claim development for accident period $k = 0$ is complete. This assumption can be relaxed in practice using so-called tail estimation methods that aim to extrapolate further link ratios (link ratios are introduced below) $\hat{f}_l^{(A)}$ for $l \geq \lfloor\tau\rfloor$. For expository purposes, this tail estimation will not be considered here. We instead refer to the relevant section in [15, Tail Estimation].

A common reserving method, the Chain Ladder (CL) method, is based on estimating link ratios $f_l^{(A)}$ that describe multiplicative development of the cumulative claim triangle

$$C_{k,l}^{(A)} := \sum_{m=0}^{l} N_{k,m}^{(A)} = \#\left\{i, j \mid (x^{(i)}, Y_j^{(i)}) \in A, \lfloor T_j^{(i)}\rfloor = k, \lfloor T_{j,\text{report}}^{(i)}\rfloor - \lfloor T_j^{(i)}\rfloor \leq l\right\}, \text{ s.t.}$$
$$R_k^{\text{IBNR}(A)} = C_{k,\lfloor\tau\rfloor-1}^{(A)} - C_{k,\lfloor\tau\rfloor-k-1}^{(A)}.$$

More precisely, the standard CL method obtains predictions as follows. First, link ratios are estimated using observable data,

$$\hat{f}_l^{(A)} := \frac{\sum_{k=0}^{\lfloor\tau\rfloor-l-1} C_{k,l+1}^{(A)}}{\sum_{k=0}^{\lfloor\tau\rfloor-l-1} C_{k,l}^{(A)}}. \qquad 0 \leq l < \lfloor\tau\rfloor - 1$$

Then, the estimated link ratios are used to estimate future cumulative claim counts by multiplying the most recent available information for each accident period with the appropriate link ratios

$$\hat{C}_{k,l}^{(A)} := \begin{cases} C_{k,l}^{(A)} & \text{if } k + l < \lfloor\tau\rfloor \\ \hat{C}_{k,l-1}^{(A)} \hat{f}_{l-1}^{(A)} & \text{if } k + l \geq \lfloor\tau\rfloor \end{cases}. \qquad 0 \leq k < \lfloor\tau\rfloor, 0 \leq l < \lfloor\tau\rfloor$$

This leads to a reserve prediction for accident period k that only depends on $C^{(A)}_{k,\lfloor\tau\rfloor-k-1}$ and k:

$$\hat{R}_k^{\mathrm{IBNR}(A)} = C^{(A)}_{k,\lfloor\tau\rfloor-k-1} \cdot (\mathrm{FtU}_k^{(A)} - 1),$$

where

$$\mathrm{FtU}_k^{(A)} := \prod_{l=\lfloor\tau\rfloor-k-1}^{\lfloor\tau\rfloor-2} \hat{f}_l^{(A)} \text{ for } 0 \le k < \lfloor\tau\rfloor$$

is called the factor to ultimate and the empty product is defined as 1.

The CL method can be equipped with model assumptions to allow for analysis of prediction errors. Under some of these assumptions it can be shown that the CL method results in a minimum variannce unbiased estimator, motivating its popularity. Mack [11] introduced the model for losses, but it can be applied to claim numbers as well. This model makes three assumptions, under which the CL prediction is an unbiased estimator of the number of claims given the available data.

(CL1) There exist link ratios $f_l^{(A)}, 0 \le l < \lfloor\tau\rfloor - 1$, such that

$$\mathbb{E}(C^{(A)}_{k,l+1} | C^{(A)}_{k,0}, \dots, C^{(A)}_{k,l}) = C^{(A)}_{k,l} \cdot f_l^{(A)} \quad \text{for } 0 \le k < \lfloor\tau\rfloor - 1.$$

(CL2) There exist parameters $\sigma_k^{2(A)}, 0 \le l < \lfloor\tau\rfloor - 1$, such that

$$\mathrm{Var}(C^{(A)}_{k,l+1} | C^{(A)}_{k,0}, \dots, C^{(A)}_{k,l}) = C^{(A)}_{k,l} \cdot \sigma_k^{2;(A)} \quad \text{for } 0 \le k < \lfloor\tau\rfloor - 1.$$

(CL3) Accident periods $\{C^{(A)}_{k,0}, \dots, C^{(A)}_{k,\lfloor\tau\rfloor-1}\}$, $\{C^{(A)}_{j,0}, \dots, C^{(A)}_{j,\lfloor\tau\rfloor-1}\}$ are independent for $k \neq j$ and $0 \le k, j < \lfloor\tau\rfloor$.

It should be noted that Assumption (CL3) in particular is hardly satisfied under real-world conditions because of calendar effects such as inflation or media attention (e.g. legal expenses insurance claims resulting from the media attention to the Dieselgate scandal [16]). Mack [11] has shown that under assumptions (CL1), (CL2) and (CL3), and with the estimators

$$\hat{\sigma}_l^{2(A)} := \frac{1}{\lfloor\tau\rfloor - l - 2} \sum_{k=0}^{\lfloor\tau\rfloor-l-2} C^{(A)}_{k,l} \left(\frac{C^{(A)}_{k,l+1}}{C^{(A)}_{k,l}} - \hat{f}_l^{(A)} \right)^2 \qquad 0 \le l < \lfloor\tau\rfloor - 2, \text{ and}$$

$$\hat{\sigma}^{2(A)}_{\lfloor\tau\rfloor-2} := \min\left(\hat{\sigma}^{4(A)}_{\lfloor\tau\rfloor-3} / \hat{\sigma}^{2(A)}_{\lfloor\tau\rfloor-4}, \hat{\sigma}^{2(A)}_{\lfloor\tau\rfloor-3}, \hat{\sigma}^{2(A)}_{\lfloor\tau\rfloor-4} \right),$$

$\hat{f}_l^{(A)}$ and $\hat{\sigma}_l^{2(A)}$ (for $0 \le l < \lfloor\tau\rfloor - 2$) are unbiased estimators for $f_l^{(A)}$ and $\sigma_l^{2(A)}$. This suggests to estimate the mean squared error of prediction for the reserve,

$$\begin{aligned} \mathrm{mse}(\hat{R}_k^{\mathrm{IBNR}(A)}) &:= \mathbb{E}\left((\hat{R}_k^{\mathrm{IBNR}(A)} - R_k^{\mathrm{IBNR}(A)})^2 \;\middle|\; \{C^{(A)}_{k,l} | k + l < \lfloor\tau\rfloor\} \right) \\ &= C^{2(A)}_{k,\lfloor\tau\rfloor-1} \cdot \sum_{l=\lfloor\tau\rfloor-k}^{\lfloor\tau\rfloor-2} \frac{\sigma_l^{2(A)}}{f_l^{2(A)}} \left(\frac{1}{C^{(A)}_{k,l}} + \frac{1}{\sum_{j=0}^{\lfloor\tau\rfloor-l-1} C^{(A)}_{j,l}} \right). \end{aligned}$$

by the plug-in approach for $f_l^{(A)}, \sigma_l^{2(A)}$ (for $0 \leq l < \lfloor \tau \rfloor - 2$) and unknown parts of the cumulative claim triangle $C_{k,l}^{(A)}$ yielding

$$\widehat{\text{mse}}(\hat{R}_k^{\text{IBNR}(A)}) = \hat{C}_{k,\lfloor\tau\rfloor-1}^{2(A)} \cdot \sum_{l=\lfloor\tau\rfloor-k}^{\lfloor\tau\rfloor-2} \frac{\hat{\sigma}_l^{2(A)}}{\hat{f}_l^{2(A)}} \left(\frac{1}{\hat{C}_{k,l}^{(A)}} + \frac{1}{\sum_{j=0}^{\lfloor\tau\rfloor-l-1} C_{j,l}^{(A)}} \right).$$

In addition to the distribution-free CL model by Mack [11], several parametric CL models were also studied, e.g. by Verrall [19] and Taylor [18]. Verrall and Wüthrich [20, Section 6.1] showed that assuming the micro-level claim occurrence process to be a homogeneous marked poisson process implies $N_{k,l}^{(A)}$ to be poisson distributed, i.e. the (over-dispersed) Poisson CL model holds and thus the CL method obtains a minimum variance unbiased estimator of the total IBNR claim counts [18, Theorem 6.2].

Tackling the claim reserving problem using triangles requires choice of a partition $\mathcal{A} = \{A \subset \mathfrak{X} \times \mathfrak{Y}\}$ of disjoint subsets such that

$$\bigcup_{A \in \mathcal{A}} A = \mathfrak{X} \times \mathfrak{Y}.$$

Compared to the set of individual data points, $\{(x^{(i)}, Y_j^{(i)})\}_{i,j}$, a large amount of information remains unused when aggregating claims data into triangles $\{N^{(A)}\}_{A \in \mathcal{A}}$. This loss of information incurred when using established reserving methods, and the increase in available computational resources has lead to research in the area of micro-level reserving, investigating methods for predicting claims reserves using more of the available claim-level information. Approaches to micro-level reserving presently examined can be broadly categorized into (1) discrete-time process models, jointly modelling a set of highly granular triangle data $\{N^{(A)}\}_{A \in \mathcal{A}}$ and (2) continuous-time process models, directly modelling the claim occurrence process of individual policies or of sub-portfolios as a point process.

Triangle-based micro-level methods can be applied to different parts of the reserving problem and combined with macro-level triangle-based methods in other parts. For example, De Felice and Moriconi [6] suggest combining a micro-level RBNS model using individual-level triangle data $\mathcal{A} = \big\{\{(x,y)\} \;\big|\; x \in \mathfrak{X}, y \in \mathfrak{Y}\big\}$ with an aggregate prediction for IBNR losses. Similarly, Wüthrich [21] examines a discrete-time model for the number of RBNS payments and suggests application of the CL method to IBNR claim counts. Baudry and Robert [3] propose working with a discretized non-parametric position-dependent marked point process, using individual-level triangle data $\mathcal{A} = \big\{\{(x,y)\} \;\big|\; x \in \mathfrak{X}, y \in \mathfrak{Y}\big\}$ for IBNR claim counts, RBNS and IBNR loss. Antonio, Godecharle, and Van Oirbeek [1] model claim development for the RBNS reserve using a discrete-time multi-state approach, not considering IBNR claims.

One continuous-time occurrence process approach to the reserving problem, which is central to this thesis, goes back to Norberg [12], suggesting a position-dependent marked Poisson process (PDMPP) to model individual policies that can incur claims (points of the process) with features (markings of the process), among which we find the reporting delay $D := T_{\text{report}} - T$. This framework is also used by Antonio and Plat [2] on a portfolio level, implicitly assuming all policies in the portfolio to be independent - an assumption also made in the work of this thesis. A gentle introduction to poisson processes can be found in Karr [9] and Last and Penrose [10]. PDMPPs are defined by an intensity measure μ on $\mathbb{R}_+ \times \mathfrak{M}$ that decomposes into an intensity (called the claim frequency in the insurance context) $\lambda(t)$ and a

marking distribution $P_{M|T=t}$ on $\mathfrak{M}$ where M denotes the mark such that the process ξ with intensity measure μ satisfies

$$\mu(S) = \int_0^\infty \lambda(t) \int_{\mathfrak{M}} \mathbf{1}((t,m) \in S) \, \mathrm{d}P_{M|T=t}(m) \, \mathrm{d}t,$$

and $\xi(S) \sim \mathrm{Poi}(\mu(S))$, for $S \subset \mathbb{R}_+ \times \mathfrak{M}$ measurable. Furthermore, $\xi(S)$ and $\xi(S')$ are independent for all disjoint sets $S, S' \subset \mathbb{R}_+ \times \mathfrak{M}$. In the context of insurance claims, the set of markings is $\mathfrak{M} = \mathfrak{Y} \times \mathbb{R}_+$, the claim features and reporting delay respectively, and the processes of interest are $\xi^{(i)}$ associated to policy $i = 1, \ldots, N_{\mathrm{pol}}$, where N_{pol} is the number of policies under consideration. Position-dependence refers to the marking distribution $P_{M|T=t}$ being dependent on the position T (accident time in the insurance context). Individual claim frequencies $\lambda^{(i)}$ are usually assumed to obey some global structure, such as $\lambda^{(i)}(t) = \tilde{\lambda}(x^{(i)}, t)$ for some global frequency function $\tilde{\lambda} : \mathfrak{X} \times \mathbb{R}_+ \to \mathbb{R}_+$, i.e. the claim frequency is completely determined by the policy features $x^{(i)}$. The process $\xi^{(i)}$ is not fully observable due to reporting delays. Instead, we observe the restriction $\xi_r^{(i)} \coloneqq \xi^{(i)} \cap S_\tau$ where

$$S_\tau \coloneqq \{(T^{\mathrm{acc}}, Y, D) \in [0, \tau) \times \mathfrak{Y} \times \mathbb{R}_+ | D < \tau - T^{\mathrm{acc}}\}.$$

We do not observe the complement $\xi_{nr}^{(i)} \coloneqq \xi^{(i)} \cap S_\tau^{\complement}$ where $D \geq \tau - T^{\mathrm{acc}}$. The quantities of interest are usually the number of IBNR claims for some subset $\mathcal{X} \subset \mathfrak{X}$ of risk features and some occurrence period $[t_0, t_1)$, possibly restricted to claim features $\mathcal{Y} \subset \mathfrak{Y}$:

$$R^{\mu\mathrm{IBNR}}(\mathcal{X} \times [t_0, t_1) \times \mathcal{Y}) \coloneqq \sum_{i=1}^{N_{\mathrm{pol}}} \mathbf{1}(x^{(i)} \in \mathcal{X}) \xi_{nr}^{(i)}([t_0, t_1) \times \mathcal{Y} \times \mathbb{R}_+),$$

where the index μ refers to "micro-level". Under the assumptions $D < \tau - 1$, so that all claims are reported after at most τ development periods, and $\tau \in \mathbb{N}$, so that $\tau = \lfloor \tau \rfloor$, we have

$$\begin{aligned}
R^{\mu\mathrm{IBNR}}(\mathcal{X} \times [k, k+1) \times \mathcal{Y}) &= \sum_{i=1}^{N_{\mathrm{pol}}} \mathbf{1}(x^{(i)} \in \mathcal{X}) \xi_{nr}^{(i)}([k, k+1) \times \mathcal{Y} \times \mathbb{R}_+) \\
&= \sum_{i=1}^{N_{\mathrm{pol}}} \mathbf{1}(x^{(i)} \in \mathcal{X}) \xi^{(i)}\Big(\big\{(T^{\mathrm{acc}}, Y, D) \;\big|\; \lfloor T^{\mathrm{acc}} \rfloor = k, T^{\mathrm{acc}} + D \geq \tau, Y \in \mathcal{Y}\big\}\Big) \\
&= \sum_{l=\tau-k}^{\tau-1} \sum_{i=1}^{N_{\mathrm{pol}}} \mathbf{1}(x^{(i)} \in \mathcal{X}) \xi^{(i)}\Big(\big\{(T^{\mathrm{acc}}, Y, D) \;\big|\; \lfloor T^{\mathrm{acc}} \rfloor = k, \lfloor T^{\mathrm{acc}} + D \rfloor = k + l, Y \in \mathcal{Y}\big\}\Big) \\
&= \sum_{l=\tau-k}^{\tau-1} N_{k,l}^{(\mathcal{X} \times \mathcal{Y})} \\
&= R_k^{\mathrm{IBNR}(\mathcal{X} \times \mathcal{Y})}
\end{aligned}$$

for $0 \leq k < \lfloor \tau \rfloor$, linking the micro-level reserve to the aggregate reserve.

Micro-level reserving aims to predict $R^{\mu\mathrm{IBNR}}$ given $\{\xi_r^{(i)}\}_{i=1}^{N_{\mathrm{pol}}}$. The first approach, introduced in detail in Article 1, is to estimate the conditional distribution of $D = D$ given $X = x^{(i)}, T = T^{\mathrm{acc}} = t$ and the claim features $Y = y$ to obtain an individual factor to ultimate for each reported claim, via

$$\mathrm{FtU}^{\mathrm{ind}}(x, t, y) \coloneqq \left(\int_{[\lfloor t \rfloor, \lfloor t \rfloor + 1)} P(D < \tau - s | X = x, T = s, Y = y) \, \mathrm{d}s \right)^{-1}.$$

This requires estimating the conditional distribution $P_{D|X=x,T=t,Y=y}$ given randomly truncated observations $(x,t,y,d) \in \mathfrak{X} \times [0,\tau) \times \mathfrak{Y} \times \mathbb{R}_+$ which we only happen to see if $t+d<\tau$. Given a parametric family $\mathcal{F} = \{F_\theta | \theta \in \Theta\}$ of distributions on $\mathbb{R}_+$ with densities f_θ, and a parametric function family, such as a multi-layer perceptron (MLP), $\mathcal{G} = \{g : \mathfrak{X} \times [0,\tau) \times \mathfrak{Y} \to \Theta\}$, distributional regression aims to find $\hat{g}$ such that, given the claim dataset $\mathfrak{D}_\tau = \{(x,t,y,d) | x = x^{(i)}, (t,y,d) \in \xi_r^{(i)}, i = 1,\dots,N_{\text{pol}}\}$

$$\hat{g} \in \arg\max_{g \in \mathcal{G}} \sum_{(x,t,y,d) \in \mathfrak{D}_\tau} \log f_{g(x,t,y)}(d) - \log F_{g(x,t,y)}([0,\tau-d)),$$

the right-hand term reflecting random truncation. This results in an estimate

$$\widehat{R^{\mu\text{IBNR}}}(\mathcal{X} \times [k,k+1) \times \mathcal{Y}) \coloneqq \sum_{(x,t,y,d) \in \mathfrak{D}_\tau, x \in \mathcal{X}, t \in [k,k+1), y \in \mathcal{Y}} \widehat{\text{FtU}^{\text{ind}}}(x,t,y) - 1,$$

where $\widehat{\text{FtU}^{\text{ind}}}(x,t,y)$ is defined by replacing the term $P(D < \tau - s | X = x, T = t, Y = y)$ in FtU^{ind} by $F_{\hat{g}(x,t,y)}([0,\tau-s))$. The second approach to obtaining micro-level reserves, developed in Aritcle 2, is to estimate the full micro-level model, such that it becomes feasible to estimate the expected value of $\xi_{nr}^{(i)}$ directly, i.e.,

$$\begin{aligned}
&\widetilde{R^{\mu\text{IBNR}}}(\mathcal{X} \times [t_0,t_1) \times \mathcal{Y}) \\
&\coloneqq \sum_{i=1}^{N_{\text{pol}}} \mathbf{1}(x^{(i)} \in \mathcal{X}) \mathbb{E}(\xi_{nr}^{(i)}([t_0,t_1) \times \mathcal{Y} \times \mathbb{R}_+)) \\
&= \sum_{i=1}^{N_{\text{pol}}} \mathbf{1}(x^{(i)} \in \mathcal{X}) \\
&\qquad \int_{[t_0,t_1)} \tilde{\lambda}(x^{(i)},t) \int_{\mathcal{Y}} P(D \geq \tau - s | X = x^{(i)}, T = s, Y = y) \, \mathrm{d}P_{Y|X=x^{(i)},T=s}(y) \, \mathrm{d}s.
\end{aligned}$$

In comparison to factor-to-ultimate based prediction methods, this method predicts a positive reserve even for sub-portfolios without any reported claims as long as the estimated frequency $\tilde{\lambda}(x^{(i)},t)$ is positive and the probability of a claim occurrence being reported, $P(D+s<\tau | X = x^{(i)}, T = s)$ is less than 1.

The estimation problems mentioned in the previous paragraphs can be regarded as special cases of distributional regression subject to random right-truncation. Article 3 revolves around distributional regression, also allowing for non-informative interval censoring in addition to random truncation. Given a sample $\mathfrak{I}_{\text{reg}} = \{(y,x)_i\}_{i=1}^N$ of $N \in \mathbb{N}$ paired observations from $\mathfrak{X} \times \mathfrak{Y}$, a parametric family of distributions $\mathcal{F} = \{F_\theta | \theta \in \Theta\}$ on $\mathfrak{Y}$, and a family of functions $\mathcal{G} = \{g : \mathfrak{X} \to \Theta\}$, distributional regression aims to find an optimal approximation $\hat{g}$ such that $Y|X = x \sim F_{\hat{g}(x)}$ approximately. This means

$$\hat{g} \in \arg\max_{g \in \mathcal{G}} \ell(g | \mathfrak{I}_{\text{reg}}),$$

where

$$\ell(g | \mathfrak{I}_{\text{reg}}) = \sum_{i=1}^N \log f_{g(x_i)}(y_i)$$

is the log-likelihood. In comparison to classical regression, such as generalized linear models (GLMs), where the regression function links predictors x to the expected value $\mathbb{E}(Y|X = x)$, distributional regression allows for more flexible influence of the predictors on the outcome distribution by regressing the entire distribution. The loss function given here can be extended by adding weights to the observation, and by accounting for non-informative interval censoring and for random truncation, resulting in a more complex sample $\mathfrak{J}_{\text{reg}} = \{(m, v, l, u, w, x)_i\}_{i=1}^N$ where an observation is only made if the unobserved outcome y is within the truncation interval $(l, u]$ and we can observe an interval $(m, v]$ which contains the observation y (or the observation y directly, encoded as $y = m = v$). In this case, the loss for distributional regression of a non-informatively interval censored and randomly truncated sample, motivated in Article 3, is defined by

$$\ell(g|\mathfrak{J}_{\text{reg}}) = \sum_{(m,v,l,u,w,x)\in\mathfrak{J}_{\text{reg}}} w \cdot \begin{cases} \log f_{g(x)}(m) - \log F_{g(x)}((l, u]) & m = v \\ \log F_{g(x)}((m, v]) - \log F_{g(x)}((l, u]) & m < v \end{cases}.$$

Note that openness or closedness of the interval bounds considered for randomly truncated, non-informatively interval censored samples, can be chosen flexibly as required by the application at hand. Distributional regression for a sample $\mathfrak{J}_{\text{reg}}$ is defined analogously to the simpler case with a fully observed sample $\mathfrak{I}_{\text{reg}}$.

This thesis has a cumulative structure consisting of three separate articles in which the author was involved. These articles are listed in Chapter 2. Article 1 contains the first paper and its supplements, which is concerned with estimating the reporting delay distribution D under a PDMPP model of the claim occurrence process, conditional on the policy features X, the accident time T^{acc} and additional claim features Y. A predictor of the number of IBNR claims is constructed from the estimated conditional distribution and examined in two case studies, one on simulated datasets with various perturbations and the other on a real dataset from a german legal expenses insurer. The method is extended to a method for estimating the full claim occurrence process model in Article 2. Novel predictors for the number of IBNR claims which are suitable for application to smaller sub-portfolios are proposed and studied on the same datasets as the first article. Among the newly developed predictors is also a micro-level adaptation of the CL method that does not require estimation of the PDMPP model introduced in Article 1. Finally, Article 3 presents the R [14] package "reservr" which was developed to facilitate the calculations performed in the first two articles. The package features a more general framework that allows for distributional regression using neural networks with randomly truncated, non-informatively censored outcome variables. A brief outlook regarding future research opportunities is given in Chapter 3.

2 Included Articles

The following list contains the articles that are included in this thesis

2.1 Axel Bücher and Alexander Rosenstock. "Micro-level prediction of outstanding claim counts based on novel mixture models and neural networks." In: *Eur. Actuar. J.* 13.1 (2023), pp. 55–90. DOI: 10.1007/s13385-022-00314-4

2.2 Axel Bücher and Alexander Rosenstock. "Combined modelling of micro-level outstanding claim counts and individual claim frequencies in general insurance". In: *SSRN* (2023). DOI: 10.2139/ssrn.4564502 Submitted for publication in *Eur. Actuar. J.* and publicly available on SSRN.

2.3 Alexander Rosenstock. *Fitting Distributions and Neural Networks to Censored and Truncated Data: The R Package reservr.* JSS Preprint. 2023. URL: https://ashesitr.github.io/reservr/articles/jss_paper.html Submitted for publication in *Journal of Statistical Software* and publicly available on GitHub Pages.

European Actuarial Journal (2023) 13:55–90
https://doi.org/10.1007/s13385-022-00314-4

ORIGINAL RESEARCH PAPER

Micro-level prediction of outstanding claim counts based on novel mixture models and neural networks

Axel Bücher[1] · **Alexander Rosenstock**[1,2]

Received: 12 October 2021 / Revised: 31 Jan 2022 / Accepted: 14 April 2022 /
Published online: 12 May 2022

Abstract

Predicting the number of outstanding claims (IBNR) is a central problem in actuarial loss reserving. Classical approaches like the Chain Ladder method rely on aggregating the available data in form of loss triangles, thereby wasting potentially useful additional claims information. A new approach based on a micro-level model for reporting delays involving neural networks is proposed. It is shown by extensive simulation experiments and an application to a large-scale real data set involving motor legal insurance claims that the new approach provides more accurate predictions in case of non-homogeneous portfolios.

Keywords Loss reserving · Individual claim features · General insurance · Randomly truncated data · Expectation maximization algorithm · Mixture distribution

1 Introduction

One of the classical challenges in non-life insurance consists of predicting parameters associated with outstanding claims, commonly referred to as IBNR claims for *incurred but not reported* [30]. Conventional approaches like the Chain Ladder method or the Bornhuetter-Ferguson method (see [33] for an introduction), which were proposed decades ago in view of the historic need for moderate computational costs, are based on aggregate claims data collected in so-called development triangles. Such an aggregation of claims data, however, is known to result in a huge loss

✉ Axel Bücher
axel.buecher@hhu.de

Alexander Rosenstock
alexander.rosenstock@hhu.de

[1] Mathematisches Institut, Heinrich-Heine-Universität Düsseldorf, Universitätsstraße 1, 40225 Düsseldorf, Germany

[2] ARAG SE, ARAG-Platz 1, 40472 Düsseldorf, Germany

of information, and likewise, possible computational restrictions became more and more superfluous due to the significant progress in technology. Therefore, many researchers have recently promoted the development of claims reserving methods that operate on individual data.

Many proposals regarding individual loss reserving rely on applications of celebrated Machine Learning (ML) techniques (see [18, 20] for general overviews), see, e.g., [8, 10–13, 28, 37, 38], among others. Most of the proposed methods have in common that they aim at modeling the development of each individual claim (in particular, each RBNS claim, for *reported but not settled*) and, if at all, use a Frequency-Severity or Chain Ladder based approach to estimate IBNR reserves over discrete time steps, usually one year. More precisely, [37] uses neural networks to obtain individualized Chain Ladder factors. Reference [12] uses neural networks to predict sets of aggregated IBNR run-off triangles. References [10, 38] model RBNS reserves using ML models and feature a Chain Ladder based approach to IBNR reserves. References [11, 28] focus completely on predicting RBNS reserves using ML models. Reference [8] applies tree based methods to both parts of the reserve.

The current paper contributes to this branch of the literature by proposing a new method to predict IBNR claim numbers. Our approach is based on a new flexible parametric model for the reporting delay distribution of an incurred claim, whose parameters are explained in terms of observed claims features by a classical multilayer perceptron neural network with multiple outputs.

The new parametric model, which might be of independent interest for general time-to-event modeling, builds upon a mixture construction proposed in [21] and involves a generalized Pareto tail, an Erlang mixture body and certain point measures. Statistical challenges to fit the model arise from the fact that observed reporting delays are subject to (random) truncation, which hampers a direct application of the classical EM algorithm [14] for mixture fitting based on (conditional) maximum likelihood (see [35] for fitting Erlang mixtures with non-random truncation). As a circumvent, we propose a suitable adaptation that relies on the ECME algorithm [27]; note that the ECME algorithm may exhibit faster convergence properties than the EM algorithm.

Estimation of the neural network parameters is done using TensorFlow, an industry-standard implementation framework for neural networks [1]. Optimization is carried out using the Adam and Nesterov-Accelerated Adam optimizers (see [15, 23], respectively) and a custom loss function is developed to adapt to the problem of fitting a parametric distribution to (randomly) truncated data. Starting values are provided by the global model fit based on the ECME-algorithm. Most implementation code is written using the R language and involves the `keras` and `tensorflow` R packages from [2, 9], respecively, as a binding to TensorFlow. The implementations are freely available as an R package called `reservr` on GitHub ([34]).

Finally, once the joint model for reporting delays has been fitted, we construct predictors for IBNR claim numbers based on a classical model for claims development involving a position-dependent marked Poisson processes, see [6, 31, 32]. Successful applications of this general idea can be found in [4, 5, 22], among others.

The new predictors are evaluated in a simulation study as well as in an application to a large-scale real-life dataset (about 250,000 contracts) concerning motor

legal insurance claims. It is found that the new predictors outperform classical Chain Ladder approaches in simulation scenarios involving non-homogeneous portfolios and in the real-life example, with quite some advantage in the latter case.

The papers which are closest in spirit to the present approach are [4, 13]. The authors of the first paper concentrate on claim severities rather than claim numbers, and also use a neural network based approach for fitting semi-parametric distribution models of mixture type. A key difference to our approach is that the authors rely on three neural networks for modeling the distribution parameters, while our approach relies on only one neural network with multiple outputs. Additionally, we also face the challenge of (random) truncation, which is not present in the problem studied by [13]. On the other hand, [4] explicitly model reporting delays subject to (random) truncation using a parametric distribution. In contrast to our approach, they model small numbers of subgroups to allow more claim-level features to influence the distribution, which is close to our global approach used for finding suitable starting values for the neural network model.

The remaining parts of this paper are organized as follows. In Sect. 2, we start by summarizing the notation and then make some preliminary remarks on the integration of reporting delays into the classical position-dependent marked Poisson process model from [31]. We then construct both a new global model for reporting delays, with constant parameters not depending on individual claims features, and then a micro-level that incorporates the latter in terms of neural networks. Approaches to fit the models to (randomly) truncated data are presented in Sect. 3. The estimators may be transferred into predictors for IBNR claim counts, which is treated in Sect. 4. Results on a large-scale simulation study are presented in Sect. 5, and an application to a real dataset involving motor legal insurance claims is presented in Sect. 6.

2 Modelling reporting delays

2.1 Preliminaries on insurance portfolio data

Consider an insurance portfolio containing N_{pol} independent risks. Each risk $\mathcal{P}$ is described by a coverage period $C = [t_{\text{start}}, t_{\text{end}}]$, and by risk features $\bar{x} \in \bar{\mathfrak{X}}$, where $\bar{\mathfrak{X}}$ is a feature space containing both discrete and continuous features; for example, information on the insured product and chosen options such as deductibles. Subsequently, we write $x = (C, \bar{x}) \in \mathfrak{X} = \{\text{intervals on } [0, \infty)\} \times \bar{\mathfrak{X}}$, and assume that x is constant over the course of the contract. In practice, risk features do change over time, but not very often, whence such a contract could be modelled as two separate risks.

Each risk can potentially incur claims during its coverage period, formally modelled by a claim arrival process. If a claim occurs at a (calendar) accident time $t_{\text{acc}} \in [t_{\text{start}}, t_{\text{end}}]$, it will not be immediately known to the insurer. The delay between accident time and time of reporting (t_{report}) results in incomplete information on the insurers side and thus necessitates the assessment of incurred but not yet reported (IBNR) claims. Of primal importance for any subsequent analysis (e.g., on

cumulated claim sizes) is an accurate prediction of the number of IBNR claims, see below for details.

We view the reporting delay ($d_{\text{report}} := t_{\text{report}} - t_{\text{acc}}$) as a mark on the claim arrival process. In addition to the reporting delay, there are several other claim features that are known to the insurer as soon as the claim is reported. We denote this feature space by $\mathfrak{Y}$. It will typically include information on the type of claim and maybe on its severity. The individual claim arrival processes, associated with the individual risks in the portfolio, are assumed to be (position-dependent) marked Poisson processes as in [31]. More precisely, following the notation in [24], we make the following assumption.

Model 1 (Claim Arrivals) Associated with each risk $\mathcal{P}^{(i)}$ in the portfolio, with risk features $x^{(i)} \in \mathfrak{X}$ among which we find the coverage period $C^{(i)}$, there is a position-dependent marked Poisson process with $N^{(i)} \sim \text{Poi}\left(\int_{C^{(i)}} \lambda(x^{(i)}, t)\,\mathrm{d}t\right)$ points

$$\xi^{(i)} = \sum_{j=1}^{N^{(i)}} \delta_{\left(T^{(i)}_{\text{acc},j}, Y^{(i)}_j, D^{(i)}_{\text{report},j}\right)}$$

on $[0,\infty) \times \mathfrak{Y} \times [0,\infty)$ with:

(i) Intensity $\lambda(x^{(i)}, t) \cdot \mathbf{1}(t \in C^{(i)})$, i.e., for all intervals $[t_0, t_1] \subseteq [0,\infty)$, we have

$$\sum_{j=1}^{N^{(i)}} \mathbf{1}(T^{(i)}_{\text{acc},j} \in [t_0, t_1]) = \int_{t_0}^{t_1} \xi^{(i)}(\mathrm{d}t, \mathfrak{Y}, [0,\infty)) \sim \text{Poi}\left(\int_{t_0}^{t_1} \mathbf{1}(t \in C^{(i)})\lambda(x^{(i)}, t)\,\mathrm{d}t\right).$$

(ii) Conditional claim feature distribution $P_Y(x^{(i)}, t) = P_{Y|X=x^{(i)},T_{\text{acc}}=t}$. Here, Y denotes a generic claim feature variable containing all claim features except for the reporting delay, while X and T_{acc} are generic risk feature and accident time variables, respectively.

(iii) Conditional reporting delay distribution $P_D(x^{(i)}, t, y) = P_{D|X=x^{(i)},T_{\text{acc}}=t,Y=y}$. Here, $D = D_{\text{report}}$ denotes a generic reporting delay variable, whose distribution is modelled conditional on the risk-claim variable (X, T_{acc}, Y).

Moreover, $\xi^{(1)}, \ldots, \xi^{(N_{\text{pol}})}$ are mutually independent.

Note that the overall intensity measure of $\xi^{(i)}$ can be written as

$$\mu^{(i)}(A) = \int_{C^{(i)}} \int_{\mathfrak{Y}} \int_{[0,\infty)} \mathbf{1}((t,y,d) \in A)\lambda\left(x^{(i)}, t\right) P_D\left(x^{(i)}, t, y\right)(\mathrm{d}d) P_Y\left(x^{(i)}, t\right)(\mathrm{d}y)\,\mathrm{d}t.$$

This paper is mainly concerned with the reporting delay D_{report}. More precisely, in the subsequent sections, we will propose (1) a parametric model for P_D that is both flexible and analytically tractable (Sects. 2.2 and 2.3), and (2) an estimation approach for the model that adequately takes care of the major nuisance that available observations are typically randomly right-truncated (Sect. 3). We impose the following assumption on the data-generating process.

Observation Scheme 1 At given calendar time τ, the available dataset $\mathfrak{D} = \mathfrak{D}_\tau$ consists of all risk features $x^{(i)}, i \in \{1, \dots, N_{\text{pol}}\}$, and all reported claim data up to calendar time τ, i.e.

$$\left\{(x^{(i)}, t^{(i)}_{\text{acc},j}, y^{(i)}_j, d^{(i)}_{\text{report},j}) \,\middle|\, t^{(i)}_{\text{report},j} := d^{(i)}_{\text{report},j} + t^{(i)}_{\text{acc},j} \le \tau\right\}. \tag{1}$$

Equivalently, we observe, for each $i \in \{1, \dots, N_{\text{pol}}\}$, the risk feature $x^{(i)}$ and the restriction $\xi^{(i)}_r(\cdot) = \xi^{(i)}(\cdot \cap R_\tau)$, where $R_\tau = \{(t, y, d) : d + t \le \tau\}$ and where the lower index r stands for '*reported*'. Note that the observations in (1) are randomly right-truncated, which requires additional care when estimating the model.

Note that the ultimate objective of claims reserving is to obtain good (aggregate) predictions for characteristics that depend on the partly unobserved paths of $\xi^{(i)}$ across different time and feature sections, based on reported observations $\xi^{(i)}_r$ as in Observation Scheme 1. Details are provided in Sect. 4, where explicit predictors are derived that depend on the (fitted) reporting delay models described in the next two sections.

2.2 A Global Parametric Model based on Blended Distributions

Reporting delays exhibit some stylized facts that appear to be present in many empirical data sets:

- First, in the *lower tail*, they are non-negative with very short reporting delays (such as $0, 1, 2, \dots$ days) being quite common. Short reporting delays may further be influenced by certain calendar effects (e.g., across weekends), whence rather flexible models are needed for the lower tail.
- On an intermediate timescale (the *body of the distribution*), reporting delays can be considered quasi-continuous and only exhibit small specific patterns. However, the general shape of the distribution differs significantly between clusters of similar claims, suggesting the use of mixture type models for large heterogeneous portfolios.
- Finally, in the *upper tail*, very long reporting delays may exist depending on the line of business, suggesting some heavy tailed behaviour.

Models for each of the three parts of the distribution are described below, to be merged later into an appropriate mixture model.

First, in the interest of maximizing flexibility, we propose to model the discrete lower tail by a mixture of Dirac-components (see also [4]), i.e., by $\sum_{i=1}^{n} p_i^{(\delta)} \delta_{i-1}$, where δ_i denotes the Dirac measure at i, where $p_i^{(\delta)}$ are mixture weights, and where the choice of n is driven by a case-specific analysis of the data, a reasonable starting value being $n = 8$ corresponding to one week.

Next, consider modeling the body of the reporting delay distribution. A good choice for a flexible continuous model is provided by a (translated) Erlang Mixture, because the latter family is dense in the space of positive distributions with respect to weak convergence [26], and hence provides sufficient flexibility for adapting

to real-life distributions. For combining (mixing) the Erlang Mixture component with the discrete lower tail, we propose to translate the Erlang Mixture component by $n - \frac{1}{2}$ such that its support does not intersect with the discrete components but additionally the smallest possible observation that does not belong to the discrete components, namely $d_{\text{report}} = n$, is in the interior of the support of the continuous component. If we translated by n or $n - 1$ instead, observations from the data would touch the boundary of the support, leading to numerical instability.

Next, consider the tail model, whose need is motivated by the fact that the tail behaviour of Erlang Mixtures is, as they are mixtures of Gamma Distributions, fixed to exponential decay (i.e., the extreme value index is 0, see [16]). In order to better capture possible heavy tail behaviour, we chose to attach to the Erlang Mixture body a Generalized Pareto Distribution with non-negative shape parameter. The latter family satisfies our need for flexibility in the heaviness of the tail and for a parsimonious parametrization, and may further be motivated by the Pickands-Balkema-de Haan theorem ([16]). Recall that the Generalized Pareto Distribution $\text{GPD}(\mu, \sigma, \xi)$ has cumulative distribution function (cdf)

$$G_{\mu,\sigma,\xi}(x) = 1 - \left(1 + \xi \frac{x - \mu}{\sigma}\right)^{-1/\xi}, \quad x \geq \mu, \tag{2}$$

with parameters $\mu \in \mathbb{R}$ (location), $\sigma > 0$ (scale), and $\xi \geq 0$ (shape). Practically the reporting delay should have finite expectation, so we constrain the GPD component to have shape parameter $0 \leq \xi < 1$, where $\xi = 0$ degenerates to an Exponential distribution which is also a member of the Erlang Mixtures.

Classical approaches of attaching a heavy-tailed distribution to a body distribution use hard cut-off thresholds that result in jump discontinuities in the resulting density. This jump can be avoided at little extra computational cost by using what we call *blended distributions* below. The construction goes back to [21, Section 2], and relies on gradually mixing two cumulative distribution functions in a blending interval A centered at some (high) threshold κ, eventually yielding a smooth density. We follow up on their ideas but, in the interest of increased flexibility, allow κ to be a parameter of the family instead of being determined by the blended component distributions.

Definition 1 (*Blended Distribution family*) Given two distributions P, Q on $\mathbb{R}$ with cdfs $F(\cdot) = P((-\infty, \cdot])$ and $G(\cdot) = Q((-\infty, \cdot])$, respectively, and parameters $\kappa \in \mathbb{R}, \varepsilon \in \mathbb{R}_+, p \in [0,1]^2, p_1 + p_2 = 1$ such that $F(\kappa) > 0$ and $G(\kappa) < 1$, we define the *Blended Distribution* $B = \text{Blended}(P, Q; p, \kappa, \varepsilon)$ of P and Q with blending interval $[\kappa - \varepsilon, \kappa + \varepsilon]$ and mixture probabilities p via its cdf F_B:

$$
\begin{aligned}
p_{\kappa,\varepsilon}(x) &= \begin{cases} x & , \quad x \in (-\infty, \kappa-\varepsilon], \\ \frac{1}{2}(x+\kappa-\varepsilon) + \frac{\varepsilon}{\pi}\cos\left(\frac{\pi(x-\kappa)}{2\varepsilon}\right) & , \quad x \in (\kappa-\varepsilon, \kappa+\varepsilon], \\ \kappa & , \quad x \in (\kappa+\varepsilon, \infty), \end{cases} \\
q_{\kappa,\varepsilon}(x) &= \begin{cases} \kappa & , \quad x \in (-\infty, \kappa-\varepsilon], \\ \frac{1}{2}(x+\kappa+\varepsilon) - \frac{\varepsilon}{\pi}\cos\left(\frac{\pi(x-\kappa)}{2\varepsilon}\right) & , \quad x \in (\kappa-\varepsilon, \kappa+\varepsilon], \\ x & , \quad x \in (\kappa+\varepsilon, \infty), \end{cases} \\
F_B(x) &= p_1 \frac{F(p_{\kappa,\varepsilon}(x))}{F(\kappa)} + p_2 \frac{G(q_{\kappa,\varepsilon}(x)) - G(\kappa)}{1-G(\kappa)}.
\end{aligned} \tag{3}
$$

See the right panel in Fig. 1 for the graph of $p_{\kappa,\varepsilon}$ and $q_{\kappa,\varepsilon}$.

Given two families $\mathcal{F}, \mathcal{G}$ of distributions on $\mathbb{R}$, and parameters $\kappa \in \mathbb{R}, \varepsilon \in \mathbb{R}_+$ (where $\mathcal{F}$ or $\mathcal{G}$ are allowed to depend on κ and ε), we define the *Blended Distribution family* as the family of Distributions

$$
\text{Blended}\,(\mathcal{F}, \mathcal{G}; \kappa, \varepsilon) := \{\, \text{Blended}\,(P, Q; p, \kappa, \varepsilon) \mid P \in \mathcal{F}, Q \in \mathcal{G}, p \in [0,1]^2, \|p\|_1 = 1 \}. \tag{4}
$$

Note that F_B defined in (3) is a mixture of two distributions, say P' and Q', that are obtained from a certain truncation-like transformation applied to input distributions P and Q, respectively, in such a way that P' is supported on a subset of $(-\infty, \kappa+\varepsilon]$, while Q' is supported on a subset of $[\kappa-\varepsilon, \infty)$. The transformed cdfs and densities are illustrated for $P = \mathcal{N}(-1,1)$ and $Q = \text{Exp}\,(1)$ in Fig. 1, alongside with respective curves for the distributions obtained by plain upper or lower truncation at κ. Note that, in practice, the choice of a suitable blending region defined by κ and ε is similar to the choice of the cut-off threshold in conventional tail modelling problems. Throughout the applications in this paper, we experimented

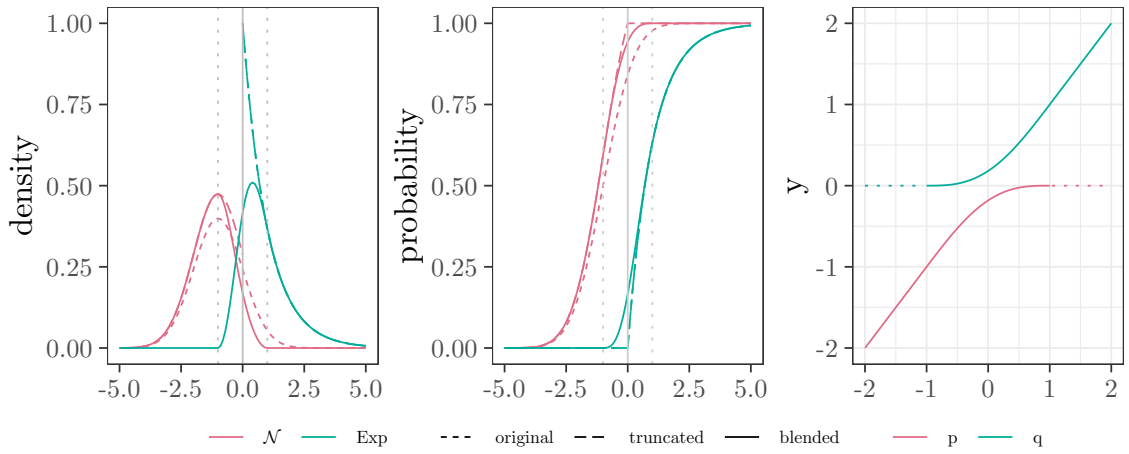


Fig. 1 Illustration of the mixture components in relation to the original families for Blended $(\mathcal{N}(-1,1), \text{Exp}\,(1); 0, 1)$. Depicted are the density and the cdf from the original, plain truncated and blended component distributions. Plain truncation refers to truncation from above at $\kappa = 0$ for $\mathcal{N}(-1,1)$ and truncation from below at $\kappa = 0$ for $\text{Exp}\,(1)$; note that the latter coincides with the original $\text{Exp}\,(1)$ distribution. The right panel shows the corresponding blending functions, $p_{\kappa,\varepsilon}$ and $q_{\kappa,\varepsilon}$. Irrelevant regions, where the corresponding components have no mass, are dotted. Compare [21, Figure 1]

with blending regions that are defined by different empirical quantiles close to 1. By doing so, we eventually control the number of observations used for fitting the tail.

If the families $\mathcal{F}$ and $\mathcal{G}$ in (4) are parameterized by sets Θ_F and Θ_G, then the mixture component families making up $\text{Blended}(\mathcal{F}, \mathcal{G}; \kappa, \varepsilon)$ are defined by their cdfs

$$\mathcal{F}' = \left\{ F' \;\middle|\; F' = \frac{F \circ p_{\kappa,\varepsilon}}{F(\kappa)} \text{ for some } F \in \mathcal{F} \right\}, \tag{5}$$

$$\mathcal{G}' = \left\{ G' \;\middle|\; G' = \frac{G \circ q_{\kappa,\varepsilon} - G(\kappa)}{1 - G(\kappa)} \text{ for some } G \in \mathcal{G} \right\}, \tag{6}$$

which are naturally parameterized by the same parameter space. Care must be taken to preserve identifiability of the parametrization as '$\theta_1 \neq \theta_2 \Rightarrow F_{\theta_1} \neq F_{\theta_2}$' does not necessarily imply the same property for $\mathcal{F}'$. For an example where this in not the case, consider the family of uniform distributions $\mathcal{U} = \{U_b = \text{Unif}(0, b) : b \in \Theta_{\mathcal{U}} = (0, \infty)\}$. If taken as the left side ($\mathcal{F}$) of a blended distribution, the blended components U_b' will be the same distribution for all $b \geq \kappa$. Note that a simple sufficient condition for identifiability is $\bigcup_{\theta \in \Theta_F} \text{supp}\, \mathcal{F}_\theta \subseteq (-\infty, \kappa]$ and $\bigcup_{\theta \in \Theta_G} \text{supp}\, \mathcal{G}_\theta \subseteq [\kappa, \infty)$.

The final distribution model that we employ for modelling reporting delays is as follows.

Definition 2 (*Blended Dirac-Erlang-Generalized Pareto family*) Given parameters $n, m \in \mathbb{N}_0$, and $\kappa, \varepsilon \in (0, \infty)$, we define the *Blended Dirac-Erlang-Generalized Pareto family* as the family of Distributions

$$\begin{aligned}
&\text{BDEGP}(n, m, \kappa, \varepsilon) \\
&:= \left\{ \sum_{i=1}^{n} p_i^{(\delta)} \delta_{i-1} + p_{n+1}^{(\delta)} \,\text{Blended}\left(\sum_{i=1}^{m} p_i^{(e)} (\Gamma_{\alpha_i,\theta} + n - \tfrac{1}{2}),\, \text{GPD}_{\mu=\kappa,\sigma,\xi}; p^{(b)}, \kappa, \varepsilon \right) \right. \\
&\left| \; p^{(\delta)} \in [0,1]^{n+1}, p^{(e)} \in [0,1]^m, p^{(b)} \in [0,1]^2, \right. \\
&\sum p_i^{(\delta)} = \sum p_i^{(e)} = p_1^{(b)} + p_2^{(b)} = 1, \\
&\left. \alpha \in \mathbb{N}^m, \alpha_1 < \cdots < \alpha_m, \theta \in \mathbb{R}_+, \sigma \in \mathbb{R}_+, \xi \in [0, 1) \right\}.
\end{aligned}$$

A specific distribution from $\text{BDEGP}(2, 3, 10, 3)$ is illustrated in Fig. 2.

This distribution family has $2m + n + 3$ degrees of freedom due to the constraints placed on the mixture parameters $p^{(\delta)}, p^{(e)}$, and $p^{(b)}$. Note that due to the restriction of $\xi < 1$, all members of this family are guaranteed to have finite expectation, though higher moments may not exist.

Returning to the context of Sect. 2, in a simplified parametric global model we assume that, for some fixed hyperparameters n, m, κ, and ϵ, the reporting delay

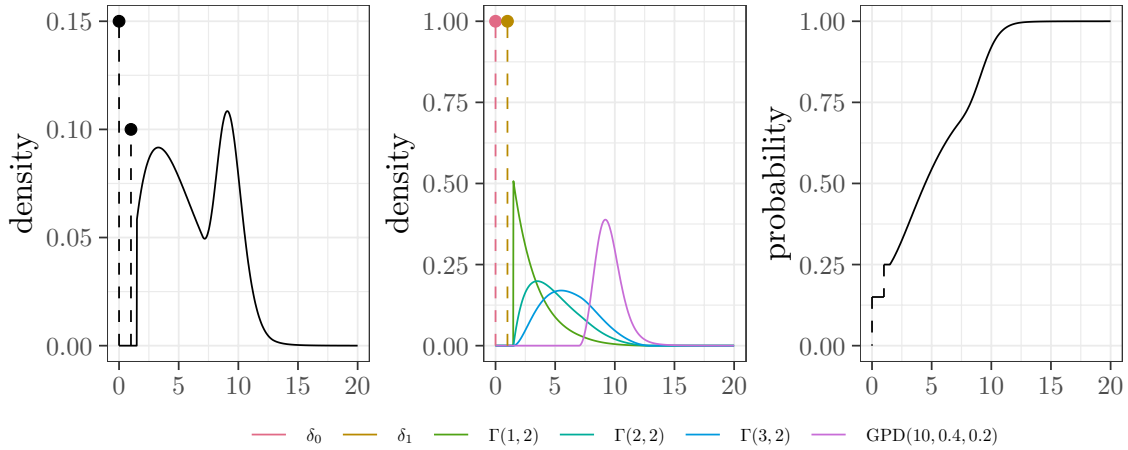


Fig. 2 BDEGP $(2, 3, 10, 3)$ distribution. Parameters: $p^{(\delta)} = (0.15, 0.1, 0.75)$, $p^{(b)} = (0.7, 0.3)$, $p^{(e)} = (0.2, 0.5, 0.3)$, $\theta = 2$, $\alpha = (1, 2, 3)$, $\sigma = 0.4$, $\xi = 0.2$. left: density, middle: component densities, right: cdf. Note how the component densities are smoothed over (7, 13) in comparison to truncated Erlang distributions or GPD $_{\mu=10,\sigma=0.4,\xi=0.2}$

distribution for each claim lies in BDEGP (n, m, κ, ϵ). Here, 'global' refers to the fact the reporting delay distribution does not depend on accident time or risk and claim features.

Model 2 (Parametric Global Model) Next to the assumptions made in Model 1 assume that, for some given (known) parameters n, m, κ, and ε, we have

$$P_D(x, t, y) \equiv B_{\tilde{\theta}} \qquad \text{for all } x, t, y \tag{7}$$

for some $B_{\tilde{\theta}} \in$ BDEGP $(n, m, \kappa, \varepsilon)$. Here, all free parameters of the BDEGP (n, m, κ, ϵ) -model are collected in a vector $\tilde{\theta} = (p^{(\delta)}, p^{(e)}, p^{(b)}, \alpha, \theta, \sigma, \xi)$ with respective parameter space $\Theta \subset [0, 1]^{n+1} \times [0, 1]^m \times [0, 1]^2 \times \mathbb{N}^m \times \mathbb{R}_+ \times \mathbb{R}_+ \times [0, 1) \subset \mathbb{R}^{2m+n+6}$ with effective dimension $2m + n + 3$.

Despite its simplicity, the global model will prove useful for finding good starting values for a fitting algorithm for the micro-level model introduced next.

2.3 A micro-level model based on neural networks

Quite naturally, the micro-level model is based on an extension of the global model by allowing $\tilde{\theta} = g(x, t, y)$ to depend on claim and risk features. More precisely, we assume that g is a neural network of some predefined architecture.

Model 3 (Micro-Level Model) Next to the assumptions made in Model 1 assume that, for some given (known) parameters n, m, κ, and ε, we have

$$P_D(x, t, y) \equiv B_{g(x,t,y)} \qquad \text{for all } x, t, y$$

for some $g \in \mathcal{G}$, where $\mathcal{G}$ denotes a set of neural networks $g : \mathfrak{X} \times \mathbb{R} \times \mathfrak{Y} \to \Theta$ such that $B_{g(x,t,y)} \in$ BDEGP $(n, m, \kappa, \varepsilon)$ for all $(x, t, y) \in$ dom (g).

Remark 1 Instead of postulating $\mathcal{G}$ to be a family of neural networks, it is also possible to consider alternative functional relationships $g : \mathfrak{X} \times \mathbb{R} \times \mathfrak{Y} \to \Theta$. For the sake of brevity, we limit ourselves to neural networks in this paper, which have proven useful in numerous applications due to their great flexibility and the efficient fitting algorithms. Likewise, neural networks may be combined with other parametric global models such as the Dirac-Weibull-mixture model from [4]. The latter was found to provide less efficient predictors in preliminary experiments, whence we restrict attention to the BDEGP family.

It remains to explain the class of neural networks $\mathcal{G}$; see [18] for a good introduction to neural networks. We chose a classical multilayer perceptron (MLP) neural network with N_{dense} hidden layers of dimension $n_1, \dots, n_{N_{\text{dense}}}$. Discrete data were incorporated using embedding layers, and the final dense layer was mapped to the parameter space Θ via canonical transformations (softmax for probability weights, softplus for positive parameters, sigmoid for interval-bounded parameters, and identity for unbounded parameters). We call this canonical mapping $f_{\text{adaptor}} : \mathbb{R}^{n_{\text{tail}}} \to \Theta$ where n_{tail} is the output dimension of the final dense layer. A more detailed description of the neural network architecture can be found in Appendix A in the supplementary material.

The neural network construction is not valid for integer components in Θ. For this reason, we must fix the shape parameters of the erlang components in the micro-level BDEGP-model. In the interest of maximizing flexibility, one could argue to fix the shapes to $1, \dots, M$ for some large integer M, such that $\mathcal{F}$ contains all erlang mixtures with shapes at most M. However, this heuristically results in overparametrization, whence we propose to fix the shapes to the values obtained from estimating the global model instead, say $\alpha = (\alpha_1, \dots, \alpha_m)$. In addition to that, we have found the parameter ξ, although real-valued, to pose numerical challenges. Individual-level parameter estimates of ξ quickly converged to 1 leading to poor performance and instability. Therefore, ξ was replaced by the (fixed) initial value obtained from fitting Model 2. Formally, this means that $\text{BDEGP}\,(n, m, \kappa, \varepsilon)$ in Model 3 will be replaced by

$$
\begin{aligned}
&\text{BDEGP}_{\,\text{fix}}(n, m, \kappa, \varepsilon, \alpha, \xi) \\
&:= \Big\{ \sum_{i=1}^{n} p_i^{(\delta)} \delta_{i-1} + p_{n+1}^{(\delta)} \,\text{Blended}\Big(\sum_{i=1}^{m} p_i^{(e)} (\Gamma_{\alpha_i,\theta} + n - \tfrac{1}{2}),\ \text{GPD}_{\,\mu=\kappa,\sigma,\xi=\xi}; p^{(b)}, \kappa, \varepsilon \Big) \\
&\Big| \ p^{(\delta)} \in [0,1]^{n+1}, p^{(e)} \in [0,1]^m, p^{(b)} \in [0,1]^2, \\
&\sum p_i^{(\delta)} = \sum p_i^{(e)} = p_1^{(b)} + p_2^{(b)} = 1, \theta \in \mathbb{R}_+, \sigma \in \mathbb{R}_+ \Big\}.
\end{aligned}
$$

This family leads to $n_{\text{tail}} = n + m + 5$ and the concrete definition $f_{\text{adaptor}}(x) = (\theta, \sigma, p^{(\delta)}, p^{(e)}, p^{(b)}) = (\text{sp}(x_1), \text{sp}(x_2), \text{sm}_{\mathbb{R}^{n+1}}(x_{3:n+3}), \text{sm}_{\mathbb{R}^m}(x_{n+4:n+m+3}), \text{sm}_{\mathbb{R}^2}(x_{n+m+4:n+m+5}))'$ where $x_{i:j} = (x_i, x_{i+1}, \dots, x_j)'$ denotes vector slices and where $\text{sp} = \text{softplus}$ and $\text{sm} = \text{softmax}$.

3 Fitting the reporting delay model to truncated data

In this section, we describe a conditional maximum-likelihood-based approach for fitting Model 3 in detail. We will start by deriving the conditional likelihood function for observed reporting delays from Observation Scheme 1 under the general setting of Model 1, see Sect. 3.1. We then proceed by considering the global model from Model 2, and describe an estimation approach based on a modified EM-Algorithm, see Sect. 3.2. Once we have an estimate for the global parameters, we can use them as starting values for an estimation procedure for the micro-level model from Model 3, see Sect. 3.3. Not using good starting values for the micro-level model proved detrimental to convergence of the estimation routine to the point of becoming unusable.

3.1 The conditional likelihood for truncated reporting delays

In this section we derive a conditional likelihood function for observed reporting delays from Observation Scheme 1 under the general setting of Model 1. It is worthwhile to mention that the resulting conditional likelihood is not bound to the case of reporting delays, but applies in any setting involving a parametric model for randomly truncated data, provided the model is dominated by a σ-finite measure and some (conditional) independence assumptions are met.

It follows from Model 1 that the reporting delays are conditionally independent given the claim features as well as the accident time, i.e.,

$$(D_j^{(i)} | X^{(i)} = x^{(i)}, T_{\text{acc},j}^{(i)} = t, Y_j^{(i)} = y) \text{ are independent with distribution } P_D(x^{(i)}, t, y),$$

for some distribution $P_D(x^{(i)}, t, y)$ depending only on $x^{(i)}, t, y$. While Models 2 and 3 are based on specific parametric assumptions, it is instructive to keep things universal, and only make the assumption that $P_D(x^{(i)}, t, y)$ has cumulative distribution function $F_{g(x^{(i)},t,y)} \in \{F_{g(x^{(i)},t,y)} : g \in \mathcal{G}\}$ for some suitable family $\mathcal{F} = \{F_\theta : \theta \in \Theta\}$ of distributions that is dominated by some σ-finite measure μ (the μ-densities are denoted by f_θ), and for some family $\mathcal{G}$ of functions $g : \mathfrak{X} \times (0, \infty) \times \mathfrak{Y} \to \Theta$ (in a global model, $\mathcal{G}$ would be the class of all constant functions $g \equiv \theta$ with $\theta \in \Theta$). Note that a natural dominating measure for the BDEGP family is $\mu = \text{Leb} + \sum_{i=0}^{n-1} \delta_i$.

To see how the data $\mathfrak{D}_\tau$ observed by an insurer at calendar time τ can be described as a truncated sample, consider points from $\xi = \xi^{(i)}$ (for the sake of readability, we omit the upper index i for the moment). They contain $(t_{\text{acc},j}, d_{\text{report},j})$, and are observed by the insurer if $t_{\text{acc},j} + d_{\text{report},j} \leq \tau$. Hence, every observed reporting delay is truncated to the interval $d_{\text{report},j} \in [0, \tau - t_{\text{acc},j}]$. As a consequence, the likelihood of every observed reporting delay must be calculated conditional on the event $D_j \in [0, \tau - T_{\text{acc},j}]$, i.e.,

$$f_{D|D\in[0,\tau-T_{\mathrm{acc}}],X=x,T_{\mathrm{acc}}=t_{\mathrm{acc},j},Y=y_j}(d_j) = \frac{f_{g(x,t_{\mathrm{acc},j},y_j)}(d_j)}{F_{g(x,t_{\mathrm{acc},j},y_j)}(\tau - t_{\mathrm{acc},j})},$$

where $(x, t_{\mathrm{acc},j}, y_j, d_j) = (x^{(i)}, t^{(i)}_{\mathrm{acc},j}, y^{(i)}_j, d^{(i)}_j)$. This leads to the following conditional log-likelihoods for Models 2 and 3, respectively:

$$\ell^G(\theta|\mathfrak{D}_\tau) = \sum_{(x,t,y,d)\in\mathfrak{D}_\tau} \log f_\theta(d) - \log F_\theta(\tau - t), \tag{8}$$

$$\ell^M(g|\mathfrak{D}_\tau) = \sum_{(x,t,y,d)\in\mathfrak{D}_\tau} \log f_{g(x,t,y)}(d) - \log F_{g(x,t,y)}(\tau - t). \tag{9}$$

Strategies to efficiently calculate the maximum of these functions are presented in the next two sections.

3.2 Estimating the global model

In this section, we describe how to maximize $\theta \mapsto \ell^G(\theta|\mathfrak{D}_\tau)$ from (8). In view of the fact that the underlying BDEGP family is essentially a mixture family, a natural approach consists of using a suitable version of the EM algorithm [14]. In fact, the procedure for fitting a BDEGP family to data is divided into subproblems which maximize conditional likelihoods on subsets of the parameter space. These building blocks need slight adaptations for blended distributions and Erlang mixture distributions, but are largely similar.

Before describing the algorithms, it is instructive to consider the underlying basics of a generic version of the EM algorithm that may be applied to samples of (both upper and lower) randomly truncated observations from a mixture model. Here, the generic mixture model shall be defined in terms of given parametric families $\mathcal{F}_1, \ldots, \mathcal{F}_k$, where the jth component family $\mathcal{F}_j$ has μ-density f_{j,θ_j} with parameter $\theta_j \in \Theta_j$, for some common dominating sigma-finite measure μ (often the sum of the Lebesgue measure on $\mathbb{R}$ and the counting measure on some subset of $\mathbb{Z}$). The mixture model, denoted $\mathcal{F}$, is then given by the family of μ-densities that are of the form

$$f_{(p,\theta)}(x) = \sum_{j=1}^{k} p_j f_{j;\theta_j}(x) \tag{10}$$

for some mixture weights $p \in (0,1)^k$ (with $\sum_{j=1}^k p_j = 1$) and some $\theta = (\theta_1, \ldots, \theta_k) \in \Theta = \bigotimes_{j=1}^k \Theta_j$.

The fact that observations are truncated can be modelled as follows: let (X, L, U) denote a random vector, where X is the variable of interest that is supposed to have a mixture density $f_{(p,\theta)}$ as in (10). The pair (L, U) is assumed to be independent of X and

shall satisfy $L \leq U$, with L possibly equal to $-\infty$ and U possibly equal to $+\infty$. Further, (L, U) shall have a density $f_{(L,U)}$ with respect to some dominating sigma-finite measure ν. A sample of interval truncated observations from (X, L, U) consists of independent observations (x_i, ℓ_i, u_i) that we only happen to see if $\ell_i \leq x_i \leq u_i$. As a consequence, any observed value can be regarded as being drawn from the $(\mu \otimes \nu)$-density

$$f_{(X,L,U)|L\leq X\leq U}(x, \ell, u) = \frac{f_{(L,U)}(\ell, u) f_{(p,\theta)}(x)}{\Pr(L \leq X \leq U)} \mathbf{1}(\ell \leq x \leq u). \tag{11}$$

Subsequently, we write (X^t, L^t, U^t) for a random vector following the above density, i.e,

$$f_{(X^t,L^t,U^t)}(x, \ell, u) = f_{(X,L,U)|L\leq X\leq U}(x, \ell, u).$$

Estimating (p, θ) based on plain maximum likelihood requires specifying a distribution for (L, U) (which can be regarded as a nuisance parameter) and calculating the denominator in (11). This (major) nuisance can be avoided by instead considering conditional maximum likelihood [3], which is known to produce consistent estimators as well. In our case, we rely on considering the density of X^t conditional on the value of $(L^t, U^t) = (\ell, u)$, which is given by

$$\begin{aligned} f_{X^t|L^t=\ell,U^t=u}(x) &= \frac{f_{(X^t,L^t,U^t)}(x, \ell, u)}{f_{(L^t,U^t)}(\ell, u)} \\ &= \frac{f_{(X,L,U)|L\leq X\leq U}(x, \ell, u)}{\int_{[\ell,u]} f_{(X,L,U)|L\leq X\leq U}(z, \ell, u)\, \mathrm{d}z} = \frac{f_{(p,\theta)}(x)}{F_{(p,\theta)}([\ell, u])} \end{aligned}$$

for $\ell \leq x \leq u$, where $F_{(p,\theta)}([\ell, u]) = \int_{[\ell,u]} f_{(p,\theta)}(z)\, \mathrm{d}\mu(z)$. As can be seen, the conditional density/likelihood is independent of the distribution of (L, U), and hence easily accessible.

For later purposes, it is helpful to attach a weight w_i to each observation (ℓ_i, x_i, u_i) (one might think of $w_i = 1$ for the moment). Denote the resulting sample by $\mathfrak{J} = \mathfrak{J}_w = \{(x_i, \ell_i, u_i, w_i) | \ell_i \leq x_i \leq u_i\}$, with sample size $N = |\mathfrak{J}|$. Based on the motivation in the previous paragraph, we aim at maximizing

$$\ell(p, \theta | \mathfrak{J}) = \sum_{(x,\ell,u,w)\in\mathfrak{J}} w \cdot \Big[\log f_{(p,\theta)}(x) - \log F_{(p,\theta)}([\ell, u]) \Big], \tag{12}$$

which is akin to maximizing $\ell^G(\theta | \mathfrak{D}_\tau)$ from (8), after identifying $\ell = 0$, $x = d_{\text{report}}$, $u = \tau - t_{\text{acc}}$ and $w = 1$. An approximate maximizer of (12), say $(\hat{p}, \hat{\theta})$, may be obtained by Algorithm 1.

Algorithm 1 General modified ECME-Algorithm

1: **function** ECME($\mathcal{F}, \mathfrak{I}, p_0, \theta_0, \varepsilon$)
2: $p \leftarrow p_0$
3: $\theta \leftarrow \theta_0$
4: $l \leftarrow -\infty$
5: **repeat**
6: $l_0 \leftarrow l$
7: $P \leftarrow \big(p_j \cdot f_{j;\theta_j}(x_i)\big)_{i=1,\dots,N, j=1,\dots,k}$
8: $P \leftarrow P \div (P \cdot e_k)$ ▷ $e_k = (1,\dots,1)^T \in \mathbb{R}^k$, division row-wise.
9: **for** $j = 1$ to k **do**
10: $\mathfrak{I}_j \leftarrow \{(x_i, \ell_i, u_i, w_i P_{i,j}) | (x, \ell, u, w)_i \in \mathfrak{I}\}$.
11: $\theta_j \leftarrow \mathrm{CML}(\mathcal{F}_j, \mathfrak{I}_j, \theta_j)$. ▷ ECM-Step j
12: **end for**
13: $p \leftarrow \arg\max_p \ell(p, \theta | \mathfrak{I})$ ▷ CM-Step $k+1$
14: $l \leftarrow \ell(p, \theta | \mathfrak{I})$
15: **until** $l - l_0 < \varepsilon$ ▷ Likelihood converged
16: **return** (p, θ).
17: **end function**

The algorithm can be motivated by the ECME principle (see [14, 27, 29]), and is derived in great detail in the supplementary material. The function CML used in line 11 of Algorithm 1 is defined as follows: for some given family $\mathcal{H}$ consisting of densities h_θ parametrized by $\theta \in \Theta$ and given a truncated and weighted sample $\mathfrak{X}$, possibly utilizing a starting value $\theta_0 \in \Theta$ for assessing the following maximum numerically, let

$$\mathrm{CML}(\mathcal{H}, \mathfrak{X}, \theta_0) := \arg\max_{\theta \in \Theta} \sum_{(x,\ell,u,w) \in \mathfrak{X}} w\big[\log h_\theta(x) - \log H_\theta([\ell, u])\big], \quad (13)$$

where H_θ is the corresponding distribution. Note that calculating the arg max can itself be based on applying an instance of an ECME algorithm if $\mathcal{H}$ is a mixture family (which is the case when applying Algorithm 1 to the BDEGP family from Definition 2). Furthermore, the densities $f_{j;\theta_j}$ used for computing the posterior probability matrix P in line 7 need to be with respect to the dominating σ-finite measure of $\mathcal{F}$, which may differ from the natural dominating measure of $\mathcal{F}_j$. This essentially leads to a separate treatment of discrete and continuous components since for each x_i for which there exists a component j such that $\{x_i\}$ has positive probability over $\mathcal{F}_j$, $P_{i,j}$ will be zero for all components with zero probability of $\{x_i\}$ even if x_i is in their support and has positive (Lebesgue) density.

Adaptations for blended distributions. In view of the fact that a blended distribution family (Definition 1) is of mixture type with $k = 2$, we could in principle directly use the general ECME algorithm to calculate a maximizer of the associated weighted conditional log-likelihood. However, this would require working with transformed versions of the original blended families, see (5) and (6). Alternatively, in each ECM-step, one may optimise the weighted conditional log-likelihood with respect to the original families by transforming the data $\mathfrak{X}$ to the scale of the original families.

More precisely, consider the first ECM step: if $\mathcal{F}_1 = \{f_{1;\theta_1} : \theta_1 \in \Theta_1\}$ denotes the first component of the blended family Blended$(\mathcal{F}_1, \mathcal{F}_2; \kappa, \varepsilon)$, then, in view of (5), the contribution of an observation $(x, \ell, u, w) \in \mathfrak{I}_1 \cap \{(x, \ell, u, w) : x < \kappa + \varepsilon\}$ to the objective function is

$$\begin{aligned}\log f'_{1;\theta_1}(x) - \log F'_{1;\theta_1}([\ell, u]) &= \log \frac{f_{1;\theta_1}(p(x)) \cdot \frac{\mathrm{d}}{\mathrm{d}x} p(x)}{F_{1;\theta_1}(\kappa)} - \log \frac{F_{1;\theta_1}([p(\ell), p(u)])}{F_\theta(\kappa)} \\ &= \log f_{1;\theta_1}(p(x)) + \log \tfrac{\mathrm{d}}{\mathrm{d}x} p(x) - \log F_{1;\theta_1}([p(\ell), p(u)]),\end{aligned}$$

where $p = p_{\kappa,\varepsilon}$. Hence, instead of calculating CML$(\mathcal{F}'_1, \mathfrak{I}_1, \theta_1)$ (line 11 of Algorithm 1) we may equivalently calculate CML$(\mathcal{F}_1, \tilde{\mathfrak{I}}_1, \theta_1)$, where $\tilde{\mathfrak{I}}_1 := \{(p(x), p(\ell), p(u), w) \mid (x, \ell, u, w) \in \mathfrak{I}_1\}$ is the transformed dataset. An analogous result can be obtained for the second component $\mathcal{F}_2$, where the transformation uses $q = q_{\kappa,\varepsilon}$. Note that the associated transformed sample $\tilde{\mathfrak{I}}_2$ is a left-truncated sample, truncated at $\ell = \kappa - \varepsilon$. Overall, we obtain Algorithm 2, where we define $b_1(x) := p_{\kappa,\varepsilon}(x)$ and $b_2(x) := q_{\kappa,\varepsilon}(x)$ for notational convenience.

Algorithm 2 Blended ECME-Algorithm

1: **function** BLENDEDECME($\mathcal{F}_1, \mathcal{F}_2, \mathfrak{I}, p_0, \theta_0, \kappa, \varepsilon,$ `tol`)
2: $p \leftarrow p_0$
3: $\theta \leftarrow \theta_0$
4: $l \leftarrow -\infty$
5: **repeat**
6: $l_0 \leftarrow l$
7: $P \leftarrow \left(p_j \cdot f_{j;\theta_j}(b_j(x_i)) \cdot b'_j(x_i)\right)_{i=1,\ldots,N, j=1,\ldots,k}$
8: $P \leftarrow P \div (P \cdot e_k)$ ▷ $e_k = (1, \ldots, 1)^T \in \mathbb{R}^k$, division row-wise.
9: **for** $j = 1$ to 2 **do**
10: $\tilde{\mathfrak{I}}_j \leftarrow \{(b_j(x_i), b_j(l_i), b_j(u_i), w_i \cdot P_{i,j}) \mid P_{i,j} > 0\}$ ▷ Transform data
11: $\theta_j \leftarrow \text{CML}(\mathcal{F}_j, \tilde{\mathfrak{I}}_j, \theta_j)$ ▷ ECM-Step j
12: **end for**
13: $p \leftarrow \arg\max_p \ell(p, \theta | \mathfrak{I})$ ▷ CM-Step 3
14: $l \leftarrow \ell(p, \theta | \mathfrak{I})$
15: **until** $l - l_0 <$ `tol` ▷ Likelihood converged
16: **return** (p, θ).
17: **end function**

Adaptations for Erlang Mixtures. Erlang Mixture families $\mathcal{F} = \{\sum_{i=1}^k p_i \cdot \Gamma_{\alpha_i,\theta} : p \in (0,1)^k, \|p\|_1 = 1, \theta \in (0,\infty), \alpha \in \mathbb{N}^k, \alpha_1 < \ldots < \alpha_k\}$ do not satisfy the definition of a mixture-type family because they possess the additional constraint that each of the Erlang components has the same scale parameter. This prevents fitting Erlang mixtures based on direct applications of the EM or the ECME algorithm, see also [19, 25, 35] for related problems with no or constant truncation bounds. For our current setting of truncation bounds that may vary with each observation, we need to adapt ideas from these papers. In particular, we rely on a version of the ECME algorithm when treating the (integer) shape parameters as fixed, and then propose a shape search algorithm to solve the remaining integer optimization problem. Details are

provided in Section C in the supplementary material, where we also explain the choice of starting values.

3.3 Estimating neural networks

In this section we describe our approach to fitting a neural network model $\mathcal{G}$ as described in Sect. 2.3 based on maximization of (9) under Model 3 with the BDEGP $_{\text{fix}}$ adaptation, i.e., holding α and ξ fixed as obtained from fitting the global model. As a consequence, the loss function is $-\ell^M(g \mid \mathfrak{D})$, which is a rather complex loss in comparison to standard losses used in ML. Training of the neural network was performed with the Adam algorithm [23] ($\text{lr} = 0.05, \beta_1 = \beta_2 = 0$) and stepsize reduction on plateau ($\text{factor} = 0.5, \text{patience} = 2, \text{min\_lr} = 10^{-4}, \text{min\_delta} = 10^{-6}$). TensorFlow [1] was used as the runtime for performing all necessary computations.

Once a specific network architecture $\mathcal{G}$ and a specific optimization routine have been chosen, fitting the neural network requires network initialization, i.e., choosing starting values for all free parameters of the model. The most common approach, introduced by Ref. [17], consists of global random initialization, where all free matrix parameters are i.i.d. uniform with mean zero and range dependent on the layer dimensions, and all free bias parameters are initialised to 0, which is then possibly applied repeatedly to potentially find better local optima. Such an approach, however, implies that the starting value of the distribution parameters feeding into the log-likelihood (9) is essentially random, and in view of the fact that the network is trained by direct optimization of the loss function, one may expect poor convergence (the latter has been confirmed in extensive preliminary experiments with simulated and real data). To alleviate this problem, we propose to only randomly initialise the free parameters of the embedding and hidden layers according to [17], while the output layer weights are chosen in such a way that the network output $g(x, t, y)$, for each observation (x, t, y), is close to some prespecified value, for instance the global estimate $\hat{\theta}$ from Sect. 3.2. This may be achieved by choosing the weights A and the bias b in the output layer in such a way that $A \approx 0$ has sufficiently small entries (see below) and $b = f_{\text{adaptor}}^{-1}(\hat{\theta})$, where $f_{\text{adaptor}}^{-1} : \Theta \to \mathbb{R}^{n_{\text{tail}}}$ is an inverse of the output layer link function f_{adaptor} defined in Sect. 2.3.

We have experimented with three different approaches to initialise A: either $A \equiv 0$ (then $g(x, t, y) \equiv \hat{\theta}$ deterministically), or A initialised by the default initialization [17], or A initialised with small random parameters on the same scale as b, i.e. $A_{i,j} \sim U[-0.1, 0.1] \cdot b_i$ for all rows $i = 1, \ldots, n_{\text{tail}}$ and columns $j = 1, \ldots, n_{N_{\text{dense}}}$ where $n_{N_{\text{dense}}}$ is the dimension of the final hidden layer in the MLP architecture. The latter method, called scaled uniform initialization, proved most efficient in practice, see Sect. 5.2 for more details. The method is summarized in Algorithm S.1 in the supplementary material.

4 Claims count prediction

The objective of claims reserving is to obtain good (aggregate) predictions for characteristics that depend on the partly unobserved paths of $\xi^{(i)}$ across different time and feature sections, based on reported observations $\xi_r^{(i)}$ as in Observation Scheme 1.

Among such characteristics is for instance the afore-mentioned total number of IBNR claims at calendar time τ, i.e.,

$$\sum_{i=1}^{N_{\text{pol}}} \sum_{j=1}^{N^{(i)}} \mathbf{1}(T^{(i)}_{\text{acc},j} \le \tau, T^{(i)}_{\text{acc},j} + D^{(i)}_{\text{report},j} > \tau) = \sum_{i=1}^{N_{\text{pol}}} \xi^{(i)}(S_\tau),$$

where $S_\tau := \{(t, y, d) : t \le \tau, t + d > \tau\}$. A further object, of primal interest for insurance pricing, is given by the total number of claims in a given time period $[t_0, t_1]$, for instance an occurence year that is not necessarily related in any specific way to τ, and for a certain class of risk feature $\mathfrak{X}' \subset \mathfrak{X}$ and claim features $\mathfrak{Y}' \subseteq \mathfrak{Y}$, i.e.,

$$\sum_{i: x^{(i)} \in \mathfrak{X}'} \sum_{j=1}^{N^{(i)}} \mathbf{1}(T^{(i)}_{\text{acc},j} \in [t_0, t_1], Y^{(i)}_j \in \mathfrak{Y}') = \sum_{i: x^{(i)} \in \mathfrak{X}'} \xi^{(i)}([t_0, t_1] \times \mathfrak{Y}' \times [0, \infty)).$$

The specific problems in the previous paragraph are special cases of the following task: for sets of interest $\mathfrak{X}' \subset \mathfrak{X}$ and $S = \{(t, y, d) : t \in [t_0, t_1], y \in \mathfrak{Y}', d \in I_t\}$ with $[t_0, t_1] \subset [0, \infty)$, some $\mathfrak{Y}' \subset \mathfrak{Y}$ and some $I_t \subset [0, \infty)$, predict the unobserved individual (and/or aggregated) claim counts in S, i.e.,

$$N^{(i)}(S) = \xi^{(i)}(S) \quad \text{and/or} \quad N(\mathfrak{X}' \times S) = \sum_{i: x^{(i)} \in \mathfrak{X}'} \xi^{(i)}(S)$$

based on a sample $\mathfrak{D}$ as in Observation Scheme 1. We will next discuss how such predictors may be derived based on knowledge of P_D only, given some suitable homogeneity constraints are met. In practice, P_D may be replaced by some estimate $\hat{P}_D$, for instance the neural network estimator from Sect. 3.3.

4.1 Predictions under local homogeneity assumptions

We start by simplifying the prediction problem by restricting attention to predictors that depend on $\mathfrak{D} = \mathfrak{D}_\tau$ through the reported numbers $N^{(i)}_r(S) = \xi^{(i)}_r(S)$ only. Let $\xi^{(i)}_{nr} = \xi^{(i)} - \xi^{(i)}_r = \xi^{(i)}(\cdot \cap R^c_\tau)$ denote the claim arrivals that are *not reported* by calendar time τ. By the restriction theorem (Theorem 5.2 in [24]), $\xi^{(i)}_{nr}$ and $\xi^{(i)}_r$ are independent Poisson processes with intensity measures $\mu^{(i)}(\cdot \cap R^c_\tau)$ and $\mu^{(i)}(\cdot \cap R_\tau)$, respectively. As a consequence, the best L^2-predictor for $N^{(i)}(S)$ in terms of $N^{(i)}_r(S)$ is given by

$$\hat{N}^{(i)}(S) = \mathbb{E}[N^{(i)}(S) \mid N^{(i)}_r(S)] = N^{(i)}_r(S) + \mathbb{E}[N^{(i)}_{nr}(S)], \tag{14}$$

and it remains to calculate the unconditional expectation on the right-hand side. Since

$$\mathbb{E}[N^{(i)}_{nr}(S)] = \mu^{(i)}(S \cap R^c_\tau) = \int_{S \cap C^{(i)} \cap R^c_\tau} \lambda(x^{(i)}, t) P_D(x^{(i)}, t, y)(\mathrm{d}d) P_Y(x^{(i)}, t)(\mathrm{d}y)\, \mathrm{d}t \tag{15}$$

by Campbell's theorem, the latter boils down to calculating a complicated high-dimensional integral. The fact that calculation of this integral must be feasible in practice is a major demand when designing models for the distributions involved in Model 1, i.e., for λ, P_Y and P_D. Under the following local homogeneity assumption, the calculation simplifies significantly.

Assumption 1 (Local homogeneity of claims developement) For a given interval $T \subset [0, \infty)$ and $\mathfrak{Y}' \subset \mathfrak{Y}$:

- $t \mapsto \lambda(x,t) =: \lambda(x) > 0$ is constant on T for any x.
- $t \mapsto P_Y(x,t)(\mathfrak{Y}') =: P_Y(x)(\mathfrak{Y}') > 0$ is constant on T for any x.
- $(t,y) \mapsto P_D(x,t,y) =: P_D(x)$ is constant on $T \times \mathfrak{Y}'$ for any x.

Even if the global claims process is highly inhomogeneous, these assumptions are approximately met for sufficiently small intervals T and sufficiently similar sets of claims $\mathfrak{Y}'$ (for continuity reasons, this particularly applies for the distributions P_D from Model 3). For instance, [4] implicitly imposes Assumption 1 for all subsequent intervals of length corresponding to one month and for $\mathfrak{Y}'$ representing either material or injury claims. Under Assumption 1, we can simplify

$$\begin{aligned}\mu^{(i)}(S) &= \int_{T\cap C^{(i)}} \int_{\mathfrak{Y}'} \int_{I_t} \lambda(x^{(i)},t) P_D(x^{(i)},t,y)(\mathrm{d}d) P_Y(x^{(i)},t)(\mathrm{d}y)\,\mathrm{d}t \\ &= \lambda(x^{(i)}) P_Y(x^{(i)})(\mathfrak{Y}') \int_{T\cap C^{(i)}} P_D(x^{(i)})(I_t)\,\mathrm{d}t,\end{aligned}$$

and likewise

$$\mu^{(i)}(S \cap R_\tau) = \lambda(x^{(i)}) P_Y(x^{(i)})(\mathfrak{Y}') \int_{T\cap C^{(i)}} P_D(x^{(i)})(I_t \cap [0, (\tau - t)_+])\,\mathrm{d}t, \tag{16}$$

$$\mu^{(i)}(S \cap R_\tau^c) = \lambda(x^{(i)}) P_Y(x^{(i)})(\mathfrak{Y}') \int_{T\cap C^{(i)}} P_D(x^{(i)})(I_t \cap ((\tau - t)_+, \infty))\,\mathrm{d}t. \tag{17}$$

As a consequence of (17), the predictor in (14) greatly simplifies, with only univariate integrals to be calculated for each i. Moreover, the previous equations may be manipulated in such a way that one obtains a natural estimator for the expectation on the right-hand side of (14) that does not depend on $\lambda(x^{(i)})$ or $P_Y(x^{(i)})(\mathfrak{Y}')$. Indeed, by Eqs. (16) and (17),

$$\mu^{(i)}(S \cap R_\tau^c) = \mu^{(i)}(S \cap R_\tau) \cdot \frac{\int_{T\cap C^{(i)}} P_D(x^{(i)})(I_t \cap ((\tau - t)_+, \infty))\,\mathrm{d}t}{\int_{T\cap C^{(i)}} P_D(x^{(i)})(I_t \cap [0, (\tau - t)_+])\,\mathrm{d}t},$$

provided the denominator is positive. Replacing $\mu^{(i)}(S \cap R_\tau)$ by its (unbiased) empirical analogue, $N_r^{(i)}(S)$, and then replacing the expectation on the right-hand side of (14) by the obtained expression, we finally obtain the predictor

$$\tilde{N}^{(i)}(S) = N_r^{(i)}(S) \cdot \frac{\int_{T\cap C^{(i)}} \hat{P}_D(x^{(i)})(I_t)\,\mathrm{d}t}{\int_{T\cap C^{(i)}} \hat{P}_D(x^{(i)})(I_t \cap [0,(\tau-t)_+,])\,\mathrm{d}t}, \tag{18}$$

where $\hat{P}_D$ is a suitable estimate of P_D. Note that, for the important special case of $S = T \times \mathfrak{Y}' \times [0,\infty)$, i.e., $I_t = [0,\infty)$, the numerator further reduces to $\mathrm{Leb}(T \cap C^{(i)})$.

Throughout the remaining parts of this paper, we will impose the homogeneity assumption from Assumption 1 for all intervals $((\ell-1)\cdot p, \ell\cdot p]$ with $\ell = 1,\dots,\lceil \tau/p \rceil$, and for all sufficiently small neighborhoods of points $y \in \mathfrak{Y}$. Note that the parameter p allows to control the restrictiveness of the local homogeneity assumption, which is less restrictive for smaller values of p. Given a set of features and accident times to be evaluated, say $A = \mathfrak{X}' \times [t_0,t_1] \times \mathfrak{Y}' \subset \mathfrak{X} \times [0,\tau] \times \mathfrak{Y}$, we aim at predicting the number of claims in A that are reported within a given (calendar) time interval $(\tau_0,\tau_1]$ with $0 \le \tau_0 < \tau_1 \le \infty$, i.e.,

$$N_{\tau_0:\tau_1}(A) = \sum_{i:x^{(i)}\in\mathfrak{X}'} \xi^{(i)}(S_{\tau_0:\tau_1})$$

with $S_{\tau_0:\tau_1} = \{(t,y,d) : t \in [t_0,t_1], y \in \mathfrak{Y}', \tau_0 < t+d \le \tau_1\}$. Note that $N_{0:\infty}(A)$ corresponds to the *ultimate number of claims in A*, while $N_{\tau:\tau+q}(A)$, is the *number of claims in A that are reported within a period of length* $q > 0$ *after calendar time* τ. The argumentation that lead to (18) suggests the following predictor for $N_{\tau_0:\tau_1}(A)$ based on observed values $\mathfrak{D}_\tau$:

$$\hat{N}^p_{\tau_0:\tau_1}(A;\mathfrak{D}_\tau) := \sum_{(x,t,y,d)\in(A\times\mathbb{R}_+)\cap\mathfrak{D}_\tau} \frac{\int_{I_p(x,t)\cap[t_0,t_1]} \hat{P}_D(x,t,y)((\tau_0-s,\tau_1-s])\,\mathrm{d}s}{\int_{I_p(x,t)\cap[t_0,t_1]} \hat{P}_D(x,t,y)([0,\tau-s])\,\mathrm{d}s}, \tag{19}$$

where $\hat{P}_D(x,t,y) \approx P_D(x,t,y)$ is the estimated reporting delay distribution and $I_p(x,t) = C(x) \cap (p\cdot\lfloor t/p\rfloor, p\cdot(\lfloor t/p\rfloor+1)]$ with $C(x)$ the coverage period of policy x and $(p\cdot\lfloor t/p\rfloor, p\cdot(\lfloor t/p\rfloor+1)]$ the interval containing accident time t on which $\xi^{(i)}$ is assumed homogeneous.

Note that the predictor in (19) can be updated continuously with the passing of time, either by reestimating $\hat{P}_D$ and then recalculating the predictor (which is computationally expensive), or by just updating the predictor with the estimated model $\hat{P}_D$ held fixed/updated only once in a while (which is less expensive). The main computational cost for predictor updates with fixed $\hat{P}_D$ lies in evaluation of the univariate integral in (19).

4.2 Evaluating claim count predictors

For comparing different methods we define evaluation metrics that measure prediction errors in a standardized way. These evaluation metrics will be used in case studies to compare model performance, as well as to perform model selection in a backtesting context. All methods will be supplied with a sample $\mathfrak{D}_\tau$ as in Observation Scheme 1.

A generic predictor for $N_{\tau_0:\tau_1}(A)$ based on observations $\mathfrak{D}_\tau$ is denoted by $\hat{N}_{\tau_0:\tau_1}(A;\mathfrak{D}_\tau)$. For simplicity, we only consider $(\tau_0,\tau_1)=(0,\infty)$ and $(\tau_0,\tau_1)=(\tau,\tau+q)$, which correspond to the ultimate number of claims and to the number of claims reported within the next period of length $q \in \{365, 365/4, 365/12\}$ (measured in days), respectively. For evaluating the predictor, we restrict attention to sets

$$A_{q,\ell,\mathfrak{Y}'} = \mathfrak{X} \times [(\ell-1)\cdot q, \ell \cdot q) \times \mathfrak{Y}',$$

where $\ell \in \{1,\dots,\tau/q\}$ denotes the ℓth period and $\mathfrak{Y}' \subset \mathfrak{Y}$. Root-mean-squared-error performance measures are then used to evaluate the performance, i.e.,

$$\mathrm{RMSE}_{\tau_0:\tau_1}(\mathfrak{Y}',q) := \left(\frac{q}{\tau}\cdot\sum_{\ell=1}^{\tau/q}\left(\hat{N}_{\tau_0:\tau_1}(A_{q,\ell,\mathfrak{Y}'};\mathfrak{D}_\tau) - N_{\tau_0:\tau_1}(A_{q,\ell,\mathfrak{Y}'})\right)^2\right)^{\frac{1}{2}}, \quad (20)$$

considered for $(\tau_0,\tau_1)=(0,\infty)$ and for $(\tau_0,\tau_1)=(\tau,\tau+q)$. Note that other error measures were examined as well, but the subsequent presentation is restricted to the above choices. In a real world scenario, $\mathrm{RMSE}_{\tau_0:\tau_1}$ is computable from $\mathfrak{D}_{\tau_1}$ at calendar time τ_1, enabling use of error measures with $\tau_1 < \infty$ outside of laboratory settings where the ground truth is known.

5 Simulation study

To demonstrate the effectiveness of the micro-level approach compared to a classical Chain Ladder based approach, we compare predictors arising from the two methods on simulated data from Model 1. Apart from a homogeneous portfolio with constant exposure, we also examine how the methods perform in the presence of smooth or abrupt changes in the claim arrival process.

5.1 Simulating car insurance portfolios

The portfolios considered throughout the simulation study build upon the car insurance data set described in Appendix A in [36]. The latter data set provides claim counts for 500,000 insurance policies, where each policy is associated with the risk features

(`age`, `ac`, `power`, `gas`, `brand`, `area`, `dens`, `ct`),

which correspond to age of driver, age of car, power of car, fuel type of car, brand of car, and area code, respectively; see also (A.1) in [36] for further details. Next to that, the data set also provides the variable `truefreq`, which corresponds to the claim intensity $\lambda(x)$ in our model. Note that the precise functional relationship $x \mapsto \lambda(x)$ has not been published by the authors.

In the following, we describe how the above data set was used to define nine different portfolios meeting the model assumptions described in Model 1 (in particular, we need to introduce a dynamic component, claim features as well as reporting

delays). Each portfolio is considered over ten periods of 365 days, that is, the portfolio coverage period is the interval [0, 3650]. We start with a baseline setting that corresponds to the classical *homogeneous portfolio*.

5.1.1 Scenario 1: a homogeneous portfolio

The homogeneous portfolio is characterized by a homogeneous *exposure* as well as position-independent *claim intensity*, *occurrence process*, and *reporting process*. It may be considered the vanilla portfolio that practitioners often aim at by careful selection of considered risks and suitable transformations, e.g., adjustment for inflation.

Exposure. New risks arrive according to a homogeneous Poisson process with intensity $50,000/365 \approx 137$ and contracts are assumed to run for exactly one year (the latter could be extended to some non-trivial annual churn rate; however, the fact that some of the considered claim features depend on calendar time and we do not know the true functional from of $x \mapsto \lambda(x)$ prevent us from doing this). Moreover, the portfolio starts with exactly 50,000 policies with $t_{\text{start}} = 0$ and with remaining contract duration that is uniform on [0, 365]. As a consequence, the total exposure is constant in expectation and we have $N_{\text{pol}} \sim 50,000 + \text{Poi}(500,000)$. Finally, for each risk in the portfolio we randomly draw (with replacement) risk features from the aforementioned data set from [36].

Claim Intensity. The claim frequency $\lambda(t, x) = \lambda(x)$ is independent of t and t_{start} and given by the variable `truefreq` that belongs to the risk selected in the previous paragraph.

Occurrence Process. The occurrence process is position-independent, i.e., $P_Y(x, t) = P_Y(x)$. In view of the fact that the original data set from [36] does not contain any individual claim variables, we employed a simple but realistic process that fits into the setting of motor liability claims. More precisely, we choose to work with two claim variables, $y = (\texttt{cc}, \texttt{severity})$, with claims code $\texttt{cc} \in \{\text{injury}, \text{material}\}$, and claim size $\texttt{severity} \in \mathbb{R}_+$. The claim feature distribution of `cc` is chosen to be a function of the policy features `ac`, `power`, and `dens` in such a way that material damages are more likely to occur in densely populated areas and with low-powered and newer cars (see Appendix D in the supplement for details on the precise relationship). The claim severity distribution of `severity` is log-normal with σ constant and with μ depending on `cc`, `brand`, `ac` and `power` in such a way that injury claims, especially with older high-powered cars, are more severe. Moreover, material damages for certain premium brands are also more severe. Again, details are provided in Appendix D in the supplement.

Reporting Process. The reporting process is position-independent, i.e, $P_D(x, t, y) = P_D(x, y)$. We choose to work with $P_D(x, y) \in \text{BDEGP}(n = 1, m = 3, \kappa = 3 \cdot 365, \varepsilon = 365/2)$ as a basic family, with fixed erlang shapes $\alpha = (1, 3, 6)$ that do not depend on x and y. The remaining 7 parameters (i.e., the four mixture weights of $\delta_0, \Gamma(1, \theta), \Gamma(3, \theta), \Gamma(6, \theta)$, and $\text{GPD}(\kappa, \sigma, \xi)$, as well as θ, σ, and ξ) are chosen to depend on `age`, `dens`, `ac` (only if `cc` is material), `cc`, and `severity` in such a way that more severe claims, material claims with new cars, and claims with younger drivers in populated areas will be reported sooner,

while low-severity injuries will be reported later; see Appendix D in the supplement for details.

A simulated portfolio from the baseline setting is illustrated in Fig. 3.

5.1.2 Scenarios 2a and 2b: changes in the exposure

The baseline setting from Scenario 1 is modified in such a way that the exposure is not constant, but either changes smoothly or abruptly in time.

In practice, smooth changes may result from a shift in the risk class distribution within the portfolio, for instance due to the fact that a competitor introduces a new product which is more attractive than the insurers own product for some risk class. In such a case, adverse selection would cause a shift in the newly written risks as the competitor product gains more visibility in the market. On the other hand, abrupt changes in the exposure may be caused by the introduction of a new risk class within the portfolio, for instance as a consequence of the introduction of a completely new product to the market, or the addition of a new sales channel reaching a new target group. Likewise, abrupt removal of an existing risk class may occur if underwriting policies change such that a product is no longer sold to a certain group, or if external factors such as OEM-provided insurance make the product obsolete for some risks.

For the simulation study, a smooth shift in exposure is realized by gradually reducing the proportion of new cars insured ($\texttt{ac} \leq 5$); see Appendix D in the supplement for details. Over the course of the simulation, the expected proportion of contracts with new cars reduces gradually from the starting value of 51.15–9.48%.

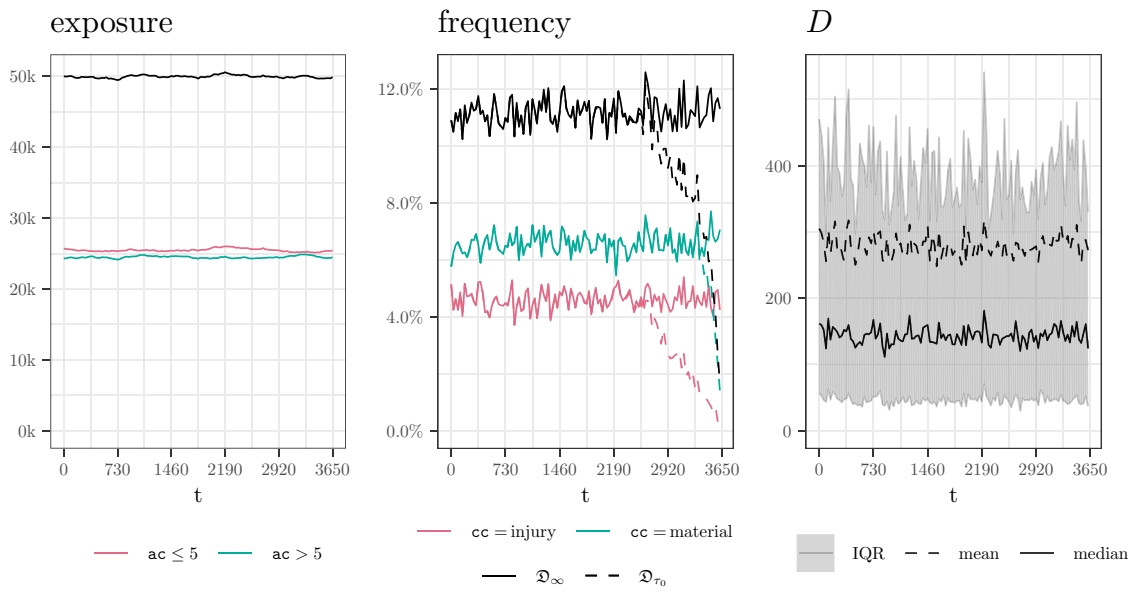


Fig. 3 A path simulated from the baseline scenario. Left: exposure by time t (i.e. active policies; $\#\{t \in C^{(i)} | i \in \{1, \dots, N_{\text{pol}}\}\}$) Center: claims frequency by accident time t (i.e. mean number of claims per policy and year). Reported claims (dashed) and occurred claims (solid). Right: monthly summary statistics of reporting delay D by accident time t

An abrupt change is introduced in the same way, by abruptly lowering the expected proportion of new cars insured to 9.48% halfway through the simulation.

5.1.3 Scenarios 3a and 3b: changes in the claim intensity

The baseline setting from Scenario 1 is modified in such a way that the claim intensity is not constant, but either changes smoothly or abruptly in time.

In practice, smooth shifts in the claim intensity may result from improved security devices reducing the risk of accidents by prevention. Preventive measures could also be implemented by the insurer, e.g. by rewarding safer driving styles in insurance telematics products [7]. On the other hand, abrupt changes may be caused by the introduction of a product with extended coverage or by external factors such as reduced traffic volume and thus decreased risk of traffic accidents, for instance due to COVID-19 related lockdown measures.

For the simulation study, a smooth shift of the claim intensity is realized by reducing the individual claim frequencies by 20% over the course of the simulation. Note that this also implies non-uniform occurrences. A shock is introduced by abruptly lowering the individual claim frequencies by 20% halfway through the simulation.

5.1.4 Scenarios 4a and 4b: changes in the occurrence process

The baseline setting from Scenario 1 is modified in such a way that the occurrence process is not constant, but either changes smoothly or abruptly in time.

In practice, smooth shifts in the claim feature distribution can be caused by a gradual macroeconomic or social change such as developments on the labor market. Abrupt changes in the claim feature distribution can be caused by external factors such as highly publicized events covered by the insurance in question. A practical example for legal insurance would be the Volkswagen emissions scandal 2015.

For the simulation study, a shift in the occurrence process is realized by making the probability $P(\texttt{cc} = \text{material})$ depend on the accident time. More precisely, the probability is chosen to increase from 58.73 to 77.51% (+0.9 on a logit scale for each risk). In addition, the severity distribution for material claims gets an increase by 1 in log-μ whereas injuries have a decrease of 0.5 in log-μ and an increase of 0.5 in log-σ. A shock is introduced in the same way, by abruptly increasing the probability of material claims and the severity distributions halfway through the simulation.

5.1.5 Scenarios 5a and 5b: changes in the reporting process

The baseline setting from Scenario 1 is modified in such a way that the reporting process is not constant, but either changes smoothly or abruptly in time.

In practice, smooth shifts in the reporting delay distribution could be caused by adaption of a new optional method for reporting claims, such as a customer portal.

An abrupt change in the reporting delay distribution could be caused by introducing a new product with specific requirements for the claims reporting process, or by a legislative change in the definition of accident occurrence.

For the simulation study, a shift in the reporting process is realized by gradually increasing the probabilities p_0 and p_1 of the δ_0 and $\Gamma(1, \theta)$ components by 2 on the logit scale, linearly with the accident time. The $\Gamma(3, \theta)$ component is also shifted such that the equation $p_2 = (1 - p_1) \cdot p_1$ still holds, see Appendix D in the supplement for details. A shock is introduced in the same way, by abruptly changing these probabilities halfway through the simulation.

Simulated portfolios from the eight non-homogeneous scenarios are illustrated in Fig. 4. Note that Scenarios 2a–3b do not yield large changes in the monthly summary statistics of the reporting delays, which suggests that IBNR prediction in these scenarios is simpler than in Scenarios 4a–5b.

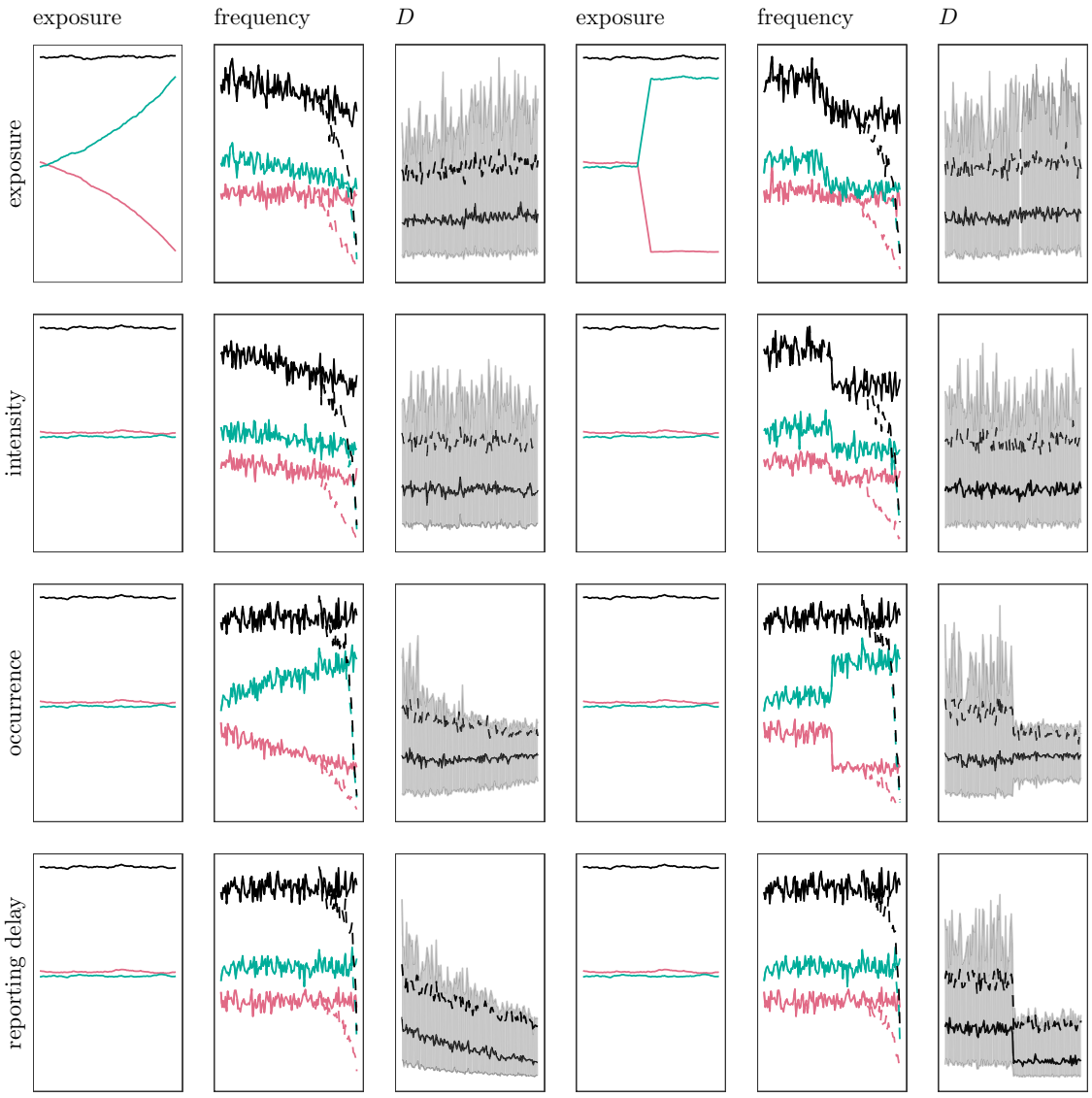


Fig. 4 Overview of all drift (left column) and shock (right column) scenarios applied to the various components (rows). The plots are explained in Fig. 3

5.2 Details on neural network predictors

Recall the generic predictor from (19), which depends on an estimated reporting delay distribution $\hat{P}_D(x,t,y)$ and a homogeneity parameter p. When employing the neural network estimator from Sect. 3.3, we denote the resulting predictor by $\hat{N}^{\mathrm{NNet},p}_{\tau_0:\tau_1}$.

For computational reasons, we choose the correct BDEGP model specification (i.e., the BDEGP $(n=1, m=3, \kappa = 3\cdot 365, \varepsilon = 365/2)$ family with unknown parameters) throughout the simulation study, as additional model selection is not feasible within a large scale simulation experiment. Note however that model selection was successfully applied in the real data application in Sect. 6.

A number of further choices have to be made for modelling and estimating P_D, the most crucial ones concerning the neural network architecture, the activation function and the weight initialization. In view of the fact that a case-by-case choice based on training and validation sets is computationally too demanding for a large-scale simulation study, we chose to fix one particular choice based on the results of a preliminary experiment in the baseline setting. The results are presented in Table 1; they concern the $\mathrm{RMSE}_{0:\infty}(\mathfrak{Y}, 365)$-performance measure each evaluated in 5000 simulation runs (50 portfolios with 100 initial weights), and suggest to use the softplus activation function, the scaled uniform initialization strategy and the neural network architecture with $N_{\mathrm{dense}} = 1$ hidden layer of size $n_1 = 5$.

Next, in view of the well-known nuisance that neural network training crucially depends on the initial network weights, a procedure is needed to choose among fits calculated from various initial weights. A natural approach consists of choosing the fit with the smallest loss. However, extensive experiments not shown in detail for the sake of brevity suggest that the following approach, partly tailored to the prediction problem at hand, yields substantially better results: among all candidate predictors (we use 100 initial weights for each data set), keep the one which minimizes the yearly backtesting error

$$\widehat{\mathrm{RMSE}}_{0:\infty}(\hat{N}_{0:\infty};\mathfrak{Y},365) := \left(\frac{1}{9}\cdot\sum_{\ell=1}^{9}\left(\hat{N}_{0:\infty}(A_{365,\ell,\mathfrak{Y}};\mathfrak{D}_{\tau-365}) - \hat{N}_{0:\infty}(A_{365,\ell,\mathfrak{Y}};\mathfrak{D}_{\tau})\right)^2\right)^{\frac{1}{2}},$$

Table 1 Median from 5000 runs of $\mathrm{RMSE}_{0:\infty}(\mathfrak{Y},365)$ for different hyperparameter choices, after 2000 epochs in the baseline setting. The Adam optimizer was used with 2000 epochs and parameters hand tuned to $\mathrm{lr} = 0.05$, $\beta_1 = \beta_2 = 0$. Hyperparameters are tuned one-by-one, the other parameters being held at softplus, $N_{\mathrm{dense}} = 1$, $n_1 = 5$, and scaled uniform initialization

Activation function		Architecture		Weight initialisation	
	RMSE_u		RMSE_u		RMSE_u
Relu	69.638	5	68.398	$A \leftarrow 0$	68.414
Softplus	68.398	10	69.382	$A_i \leftarrow U[-0.1, 0.1]^{\otimes n_{N_{\mathrm{dense}}}} \cdot b_i$	68.398
		10,5	71.600	$A \leftarrow$ [17]	69.377
		15,10,5	73.687		

which is obtained from $\mathrm{RMSE}_{0:\infty}(\mathfrak{Y}, 365)$ in Eq. (20) by plugging in $\hat{N}_{0:\infty}(\dots;\mathfrak{D}_\tau)$ for $N_{0:\infty}(\dots)$ and evaluating on $\hat{N}_{0:\infty}(\dots;\mathfrak{D}_{\tau-365})$. Note that the selection does not involve any data unseen by time τ.

5.3 Details on Chain Ladder predictors

Similar to the NNet predictor, the Chain Ladder method was used with different discretization periods $p \in \{365, 365/4, 365/12\}$. Based on cumulative link ratios $f_j = f_j(p)$ for development period $j \in \{1, \dots, \tau/p - 1\}$,

$$f_j = \frac{\#\{(x,t,y,d) \in \mathfrak{D}_\tau \,:\, \lfloor \frac{t+d}{p} \rfloor \le \lfloor t/p \rfloor + j \le \tau/p\}}{\#\{(x,t,y,d) \in \mathfrak{D}_\tau \,:\, \lfloor \frac{t+d}{p} \rfloor + 1 \le \lfloor t/p \rfloor + j \le \tau/p\}}, \tag{21}$$

the Chain Ladder predictors, for $A \subset \mathfrak{X} \times \mathbb{R}_+ \times \mathfrak{Y}$, are given by

$$\begin{aligned}
\hat{N}^{\mathrm{CL},p}_{0:\infty}(A) &= \sum_{i=1}^{3650/p} \hat{N}^{\mathrm{CL},p}_{0:\infty}(A \cap A_{p,i,\mathfrak{Y}}) \\
&= \sum_{i=1}^{3650/p} N_r(A \cap A_{p,i,\mathfrak{Y}}) \cdot \prod_{j=\tau/p-i+1}^{\tau/p-1} f_j \\
\hat{N}^{\mathrm{CL},p}_{\tau:\tau+365}(A) &= \sum_{i=1}^{3650/p} \hat{N}^{\mathrm{CL},p}_{\tau:\tau+365}(A \cap A_{p,i,\mathfrak{Y}}) \\
&= \sum_{i=1}^{3650/p} N_r(A \cap A_{p,i,\mathfrak{Y}}) \cdot \left(\prod_{j=\tau/p-i+1}^{(\tau+365)/p-i\wedge\tau/p-1} f_j - 1 \right).
\end{aligned}$$

Note that, unlike the neural network predictors, the Chain Ladder method can only be updated in discrete time steps which are multiples of the discretization period.

In view of the fact that cc is the main feature causing the perturbations in the non-homogeneous scenarios, we also applied Chain Ladder separately for $\mathrm{cc} \in \{\text{material}, \text{injury}\}$, resulting in the predictors

$$\begin{aligned}
\hat{N}^{\mathrm{CL}_{cc},p}_{0:\infty}(A) &= \sum_{i=1}^{3650/p} \sum_{c \in \{\text{material},\text{injury}\}} N_r(A \cap A_{p,i,\{c\}\times\mathbb{R}_+}) \cdot \prod_{j=\tau/p-i+1}^{\tau/p-1} f^c_j, \\
\hat{N}^{\mathrm{CL}_{cc},p}_{\tau:\tau+365}(A) &= \sum_{i=1}^{3650/p} \sum_{c \in \{\text{material},\text{injury}\}} N_r(A \cap A_{p,i,\{c\}\times\mathbb{R}_+}) \cdot \left(\prod_{j=\tau/p-i+1}^{(\tau+365)/p-i\wedge\tau/p-1} f^c_j - 1 \right),
\end{aligned}$$

where the Chain Ladder factors $\{f^c_j\}_j$ are calculated as in (21), but with $\mathfrak{D}_\tau$ replaced by $\mathfrak{D}_\tau \cap \{\mathrm{cc} = c\}$.

5.4 Results

Throughout this section we highlight important findings from the simulation study.

We start by providing a general overview of the performance across scenarios. For the sake of illustration, we restrict attention to three predictors only, namely $\hat{N}^{\mathrm{CL},365}_{0:\infty}$, $\hat{N}^{\mathrm{CL}_{cc},365}_{0:\infty}$ and $\hat{N}^{\mathrm{NNet},365}_{0:\infty}$, and to the evaluation metric $\mathrm{RMSE}_{0:\infty}(\mathfrak{Y}, q)$ with $q = 365$ (other predictors and evaluation periods lengths q will be considered below). The results are summarized in Fig. 5, where we depict, for each scenario described in Sect. 5.1, boxplots of the evaluation metric obtained from 50 simulation runs each. We observe that, for the baseline setting as well as for Scenarios 3a and 3b (Intensity), both Chain Ladder methods exhibit a slightly smaller overall error than the neural network predictor. This behavior may have been expected, since the global reporting delay distribution and thus the development pattern which Chain Ladder relies on is essentially constant over time in the two scenarios, as can be seen from Figs. 3 and 4. Within Scenarios 2a and 2b (Exposure), the global Chain Ladder predictor performs slightly worse that the neural network predictor, while CL_{cc} performs best. The latter may be explained by the fact that the introduced inhomogeneities have rather little influence on the frequency of the injury claims (see Fig. 4), whence restricted Chain Ladder performs well on that subset. The neural network predictor shines in Scenarios 4 and 5 (Occurrence and Reporting Delay, respectively), which both exhibit rather large inhomogeneities in the reporting process (see Fig. 4). These two scenarios greatly deteriorate the performance of Chain Ladder, while the neural network is able to adapt to the changes. Summarizing the findings, we find that the neural network predictor works reasonably well in all situations under consideration, with rather minor disadvantages in some scenarios, and substantial advantages in others.

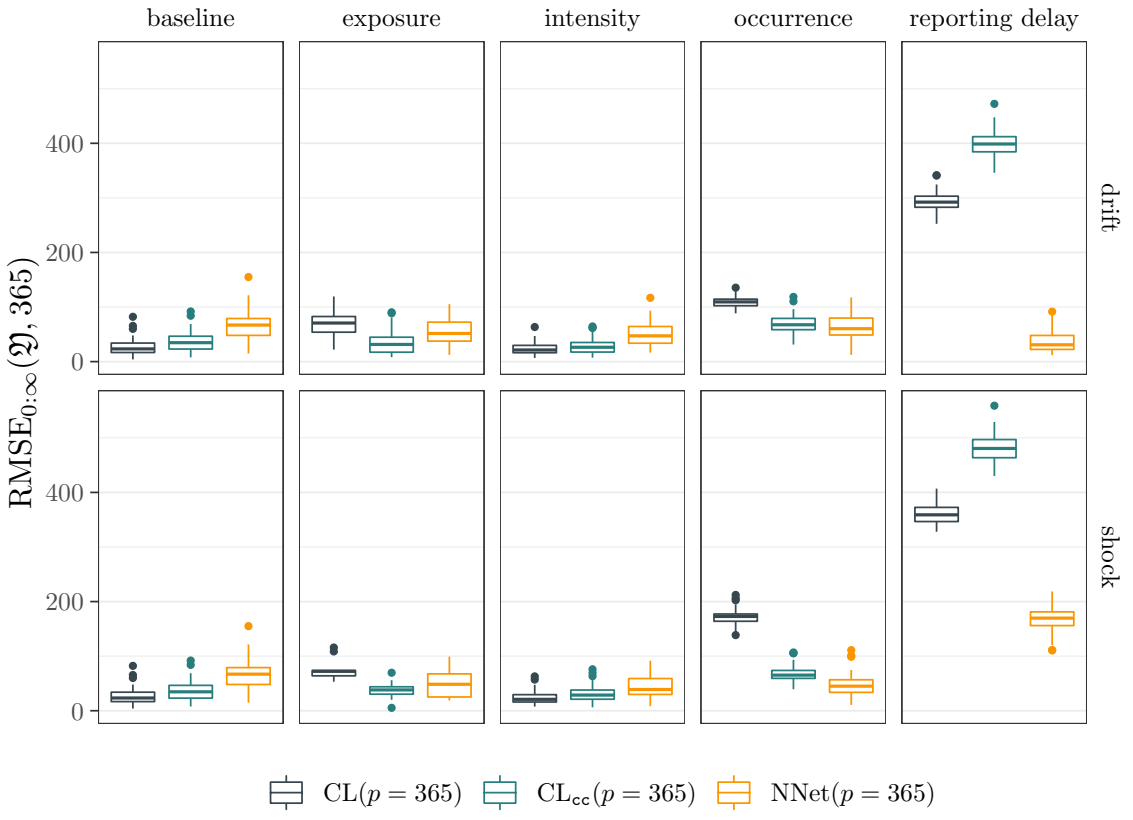


Fig. 5 Boxplots of the overall error measure $\mathrm{RMSE}_{0:\infty}(\mathfrak{Y}, 365)$, each based on $n = 50$ simulated paths

Next, in Fig. 6, we report analogous results separately by claims code for the performance measures $\mathrm{RMSE}_{0:\infty}(\mathfrak{Y}_m, 365)$ and $\mathrm{RMSE}_{0:\infty}(\mathfrak{Y}_i, 365)$, where $\mathfrak{Y}_m := \{\text{material}\} \times \mathbb{R}_+$ and $\mathfrak{Y}_i := \{\text{injury}\} \times \mathbb{R}_+$ are the subsets of $\mathfrak{Y}$ restricted to a single claims code. The message is simple: for all scenarios under consideration, the plain Chain Ladder predictor is unable to provide accurate, competitive predictions on the subsets defined by $\mathtt{cc} = \text{material}$ and $\mathtt{cc} = \text{injury}$. When comparing the neural network predictor with CL_{cc}, we observe the same qualitative behavior as in Fig. 5. It is, however, important to mention that the latter method requires prior identification of the relevant features (which might not be possible), while the neural network approach is universal, and can be applied with ease to any evaluation set of interest.

Finally, the results presented in Fig. 7 allow to compare the predictors with development period $p \in \{365, 365/4, 365/12\}$ with respect to performance measures with evaluation period $q \in \{365, 365/4, 365/12\}$. For the sake of brevity, we restrict attention to the baseline setting; qualitatively similar results were obtained for the other scenarios. We observe that, if the development period is larger than the evaluation period, the errors tend to increase drastically, in particular for the Chain Ladder method. On the other hand, if the development period is smaller than the evaluation period, the error increases slightly showing reduced stability. Overall it seems preferable to choose the smallest development period that still yields stable results as the period of choice. Another observation that can be made is that the difference between Chain Ladder and

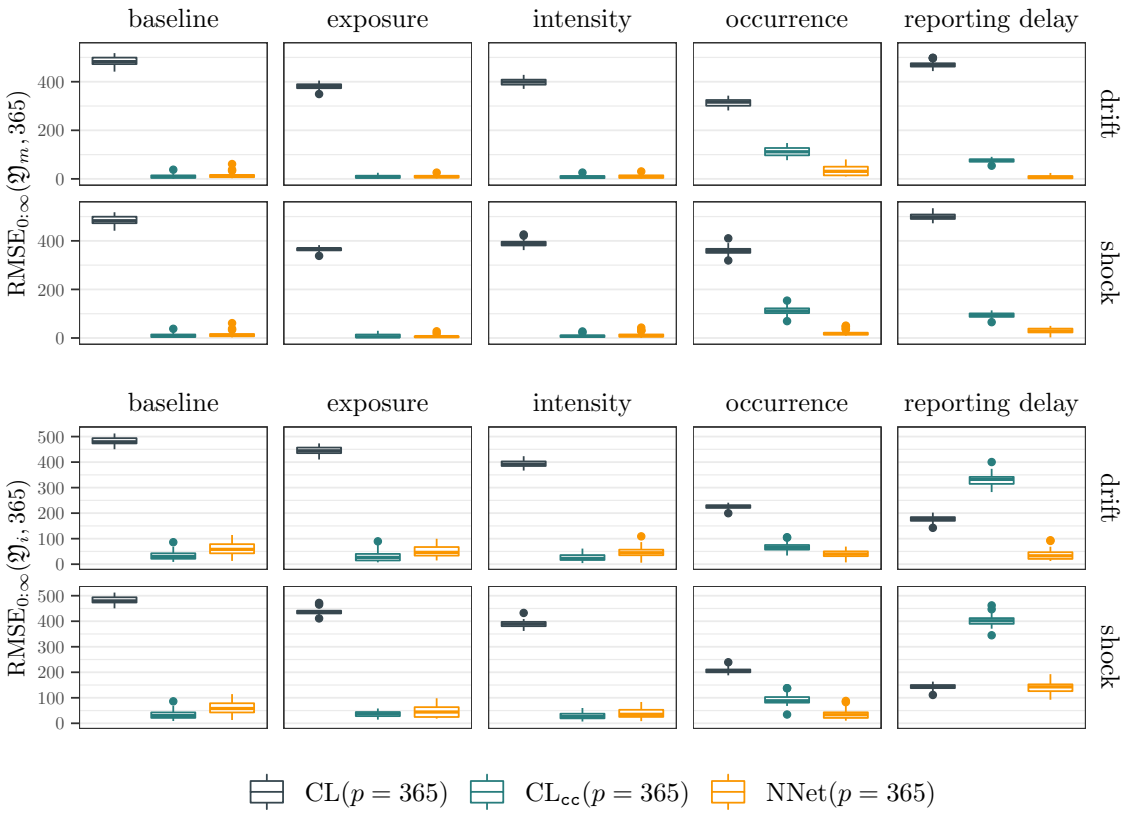


Fig. 6 Boxplots of error measures split by claims code $\mathrm{RMSE}_{0:\infty}(\mathfrak{Y}_m, 365)$ and $\mathrm{RMSE}_{0:\infty}(\mathfrak{Y}_i, 365)$, each based on $n = 50$ simulated paths

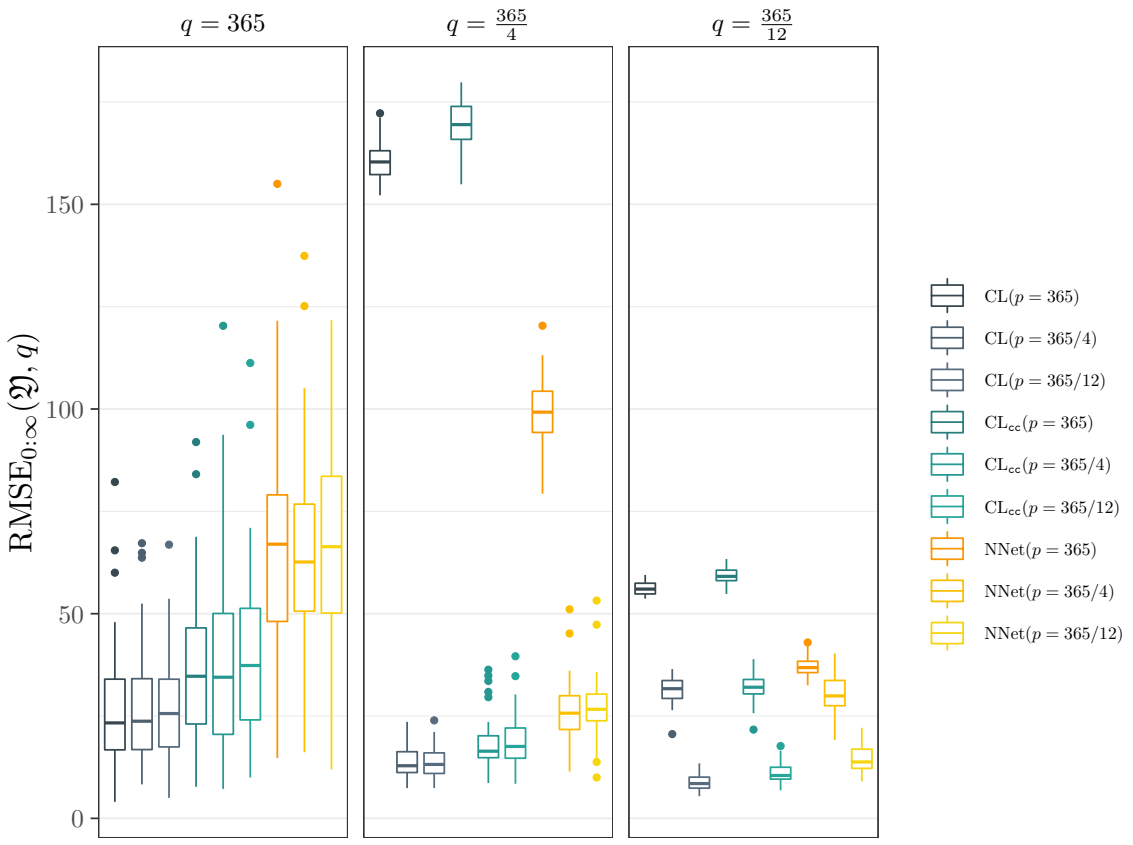


Fig. 7 Boxplots of error measure $\mathrm{RMSE}_{0:\infty}(\mathfrak{D}, q)$ for different evaluation periods $q \in \{365, 365/4, 365/12\}$ in the baseline setting

neural network based approaches gets smaller for shorter evaluation periods. In other words, the stability advantage of Chain Ladder with its comparatively few parameters diminishes as the number of Chain Ladder parameters (link ratios to be estimated) increases—even in the optimal setting for chain ladder where the portfolio and the occurrence process is homogeneous.

6 Application to real data

Throughout this section, we compare our new approach with Chain Ladder in an application to a large real dataset containing motor legal insurance claims provided by a German insurance company. Details on the dataset are provided in Sect. 6.1. The methods and results, including strategies applied for model selection (which have been omitted in Sect. 5), are discussed in Sect. 6.2. In a nutshell, the results show that the neural network predictors robustly provide more efficient predictions than Chain Ladder.

6.1 The dataset

The dataset contains a portfolio of about 250,000 motor legal insurance contracts with exposure information available monthly from 31st January 2014 to 31st December 2020. Information on the claims of this portfolio is available up to 31st December 2020. as well. In total, there are about 65,000 reported claims.

The policy features considered for modelling reporting delays are
(`tstart`, `cstart`, `tariff`, `dob`),
where

- `tstart` is the start date of the contract.
- `cstart` is the start date of the customer relationship.
- `tariff` gives information on the tariff (regular, public service, self-employed).
- `dob` contains the date of birth of the customer (accurate to months, contains missing values). Missing values were imputed using the median observed age at contract start (`tstart`) as a reference. An indicator variable to show missingness was also added.

In addition, several low-cardinality claim features available at time of reporting, as well as the accident time were included:
(`tacc`, `cc`, `covered`, `channel`, `reporter`),
where

- `tacc` is the accident date. The dataset contains inaccurate data, where the true accident time is unknown. These are encoded as January 1st and flagged with an indicator variable. Moreover, some rare claims have `tacc` = `tstart` (which, for instance, is due to legal consulting regarding claims that have happened before the contract has started); these claims are identified with an additional indicator variable.
- `cc` is the claims code, a rough categorization of the type of claim. It has five different categories numbered from 1 to 5.
- `covered` is the coverage status of the claim. It has four different categories, but is almost binary (covered, not covered, partially covered, coverage status unknown), with 'covered' and 'not covered' making up the majority of cases.
- `channel` is the channel by which the claim was reported. It has six different categories (mail, e-mail, fax, online, in person, telephone).
- `reporter` denotes the reporter of the claim. It has six different categories, but is almost binary (policyholder, additional insured, lawyer, intermediary, other, unknown). Most claims are reported by the policyholder or filed directly by a lawyer.

The rationale for including `tacc` as a feature in the neural network predictors is to help identify drifts in the reporting process.

Due to the extreme shock the COVID-19 pandemic had on the dataset, we chose to only consider data available up to 31st December 2019 for model evaluation, since none of the prediction methods provided remotely acceptable results when

validating the predicted number of claims given data up to 31st December 2019 compared to the actual numbers observed in 2020.

6.2 Results

Predictions were calculated based on a conventional Chain Ladder approach as well as on various neural network predictors (among which a final, data-adaptive choice was made as described below). To reduce the effect of a single calendar year on our studies, we examined two artificial truncation points, $\tau =$ 31st December 2017 and $\tau =$ 31st December 2018, and evaluated the methods using the one-year-ahead validation error $\mathrm{RMSE}_{\tau:\tau+365}(\mathfrak{Y}, q = 365)$ for both truncation points.

Regarding Chain Ladder, we chose to separately apply it to the five datasets defined by the different claims code (which corresponds to CL_{cc} from the previous section, and could be regarded as common actuarial practice). A visualization of the different reporting delay distributions by `cc` can be found in Fig. 8. Note that further subdivision may severely impact the stability of Chain Ladder methods, due to the combinatorial explosion of the number of different link ratio sequences that would have to be estimated.

Regarding the neural network predictors, the underlying BDEGP family was specified as follows: first, we held $\kappa = 160$ and $\varepsilon = 90$ fixed; mainly for computational reasons. Next, we chose to consider all combinations of $n \in \{7, 14, 21, 30\}$ and $m \in \{3, 5, 10, 15\}$. After computing global fits for all these families, we proceeded to repeatedly train a neural network model with fixed architecture $n_1 = 10, n_2 = 5$ for 2000 epochs, each ten times with different random starting values. During training, the available data were split into a training and a validation set in a ratio of 75% : 25%. The loss, i.e., the mean negative log-likelihood, was monitored for both datasets and logged for each epoch. This process was repeated for both data truncation points τ.

In an effort to reduce computational cost, only the six best families according to the mean backtesting error $\mathrm{RMSE}_{\tau-365:\tau}(\mathfrak{Y}, q = 365)$ over both years were

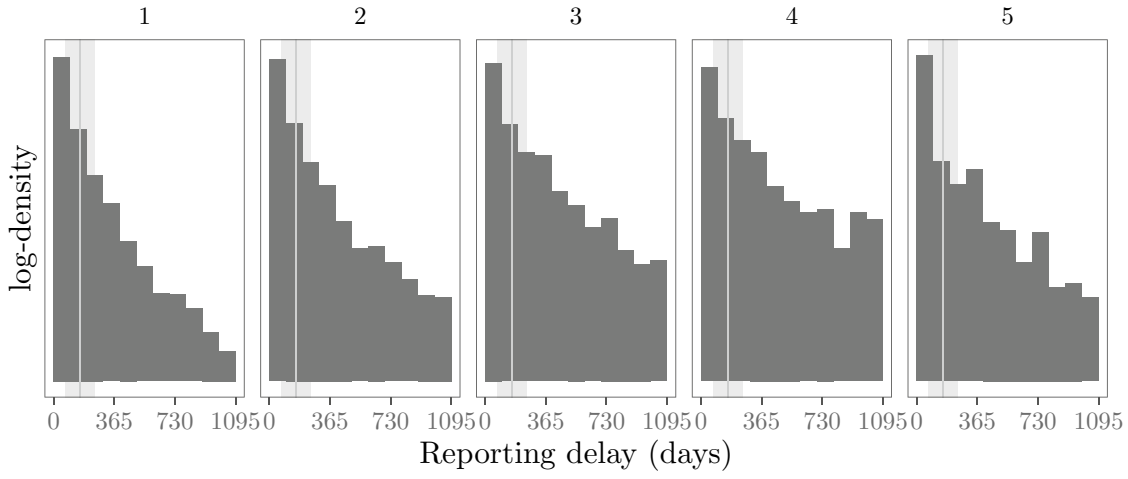


Fig. 8 Empirical reporting delay distributions by `cc`. Logarithmic axis. Vertical lines mark the chosen blending region κ and $\kappa \pm \varepsilon$

trained for additional 3000 epochs with the same method. The selected families correspond to $(n, m) \in \{(30, 10), (30, 3), (7, 10), (7, 15), (7, 3), (7, 5)\}$. The training and validation loss as a function of the epochs is exemplarily illustrated in Fig. 9 for the final selected models $(n, m) = (30, 3)$ with $\tau = 31$st December 2017 and $(n, m) = (30, 10)$ with $\tau = 31$st December 2018, which shows big loss improvements after 2000 epochs and a final loss close to convergence after 5000 epochs. Remarkably, the training loss is larger than the validation loss, which is due to a fortunate selection of the validation set.

Next, in order to select a final unique model, we performed model selection as follows: first, the best model per family (i.e, the best of ten random seeds for parameter initialization) was selected based on validation loss. Next, the overall model among the six remaining candidates was selected based on the backtesting error $\mathrm{RMSE}_{\tau-365:\tau}(\mathfrak{Y}, q = 365)$. Note that the latter evaluates the trained model on data it has already partially seen during training, which is readily available, and that it is a selection criterion which, unlike the final log-likelihood, allows for comparison across different architectures.

The results for the six models that were trained 5000 epochs, as well as the final selection and the Chain Ladder benchmark, are summarized in Fig. 10, where the 10 runs per model and year (τ) are illustrated by a boxplot. Horizontal lines show the corresponding values for Chain Ladder and for the final selected model.

It can be seen that, irrespective of the model or the random seed used for parameter initialization, the neural network predictors outperform Chain Ladder, with very few exceptions for 2017 and huge improvements for most cases. In view of the fact that the final selected models show a substantially different performance for 2017 and 2018, the model selection procedure should be taken with some care. Nevertheless, even a random choice would provide a viable selection criterion, which shows that the neural network approach is quite robust with respect to model selection.

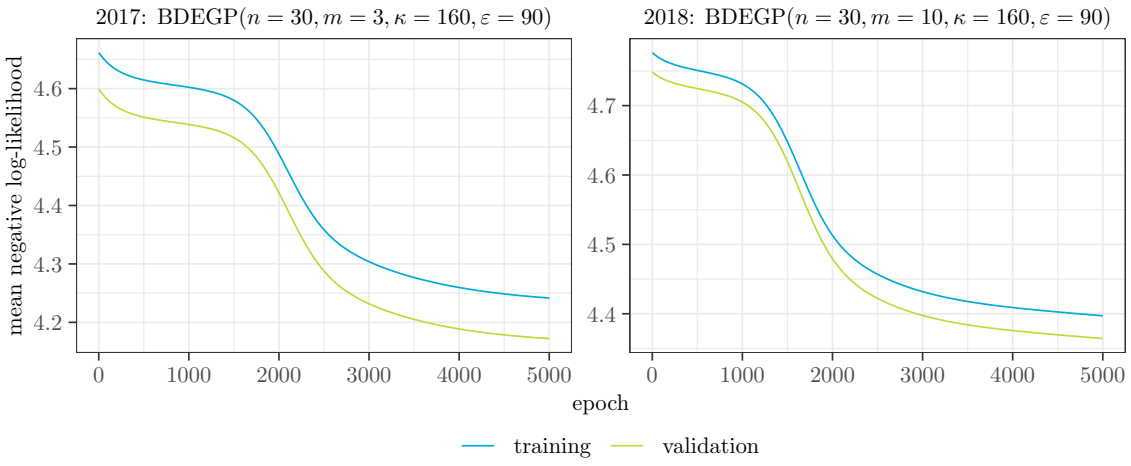


Fig. 9 Training and validation losses by epoch for the final selected models per τ

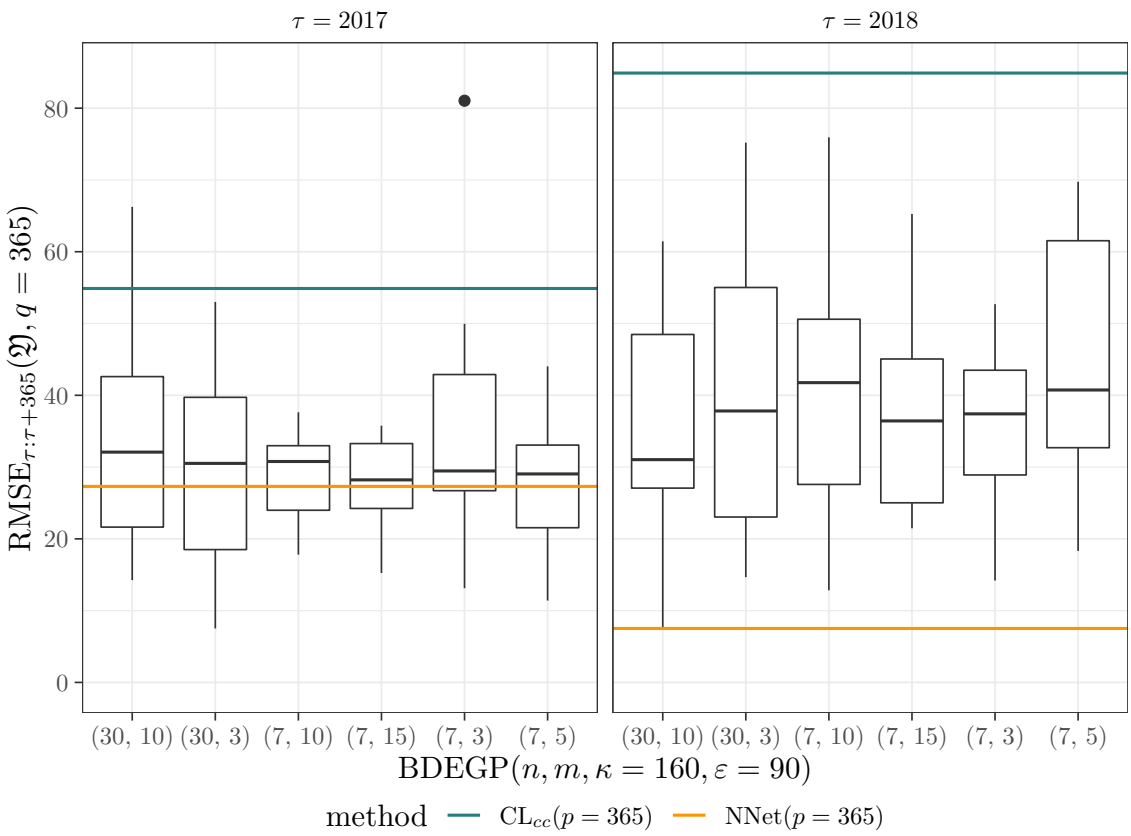


Fig. 10 Comparison of $\mathrm{RMSE}_{\tau:\tau+365}(\mathfrak{Y}, q = 365)$ for all runs with 5000 epochs. Horizontal lines show the corresponding values for CL_{cc} and the final selected model

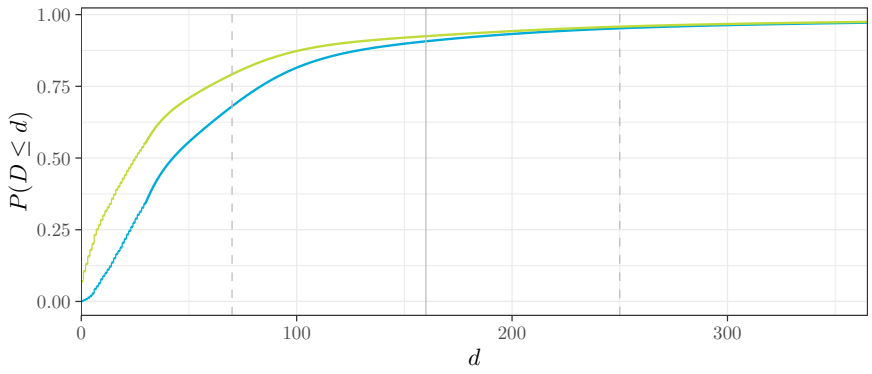


Fig. 11 Predicted reporting delay distributions for two claims from the final fit with $\tau =$ 31st December 2017. Note the discreteness of the distributions on $0 \leq d < 30$

Finally, predicted distributions for two exemplary claims in the training set are shown in Fig. 11 for $\tau =$ 31st December 2017. The claims have claim codes 1 and 2 respectively and show the flexibility of the BDEGP $(30, 3, 160, 90)$ family underlying the neural network model.

7 Conclusion

A new, flexible micro-level model for reporting delays has been developed and applied to predict IBNR claim counts. It was demonstrated that the approach performs well in comparison to classical actuarial methods in both simulation studies and on real world data. Strengths of the micro-level approach particularly emerge in the presence of heterogeneity in the data generating process, as is often the case in real world examples, and in the presence of complex relationships involving many features. Incorporating many features into classical methods becomes prohibitively difficult with an ever-increasing amount of available information. Another advantage of the newly developed method is the ability to continuously update predictions as new data becomes available, reducing critical information delay for stakeholders.

There are ample opportunities for further development on the approach:

1. The BDEGP family, while very flexible, might not be suitable for all applications. Future work could examine the choice of different families.
2. While Model 3 assumes a neural network functional relationship between reporting delay distribution and predictors, different functional relationships could be examined. The non-trivial nature of the conditional likelihood under study makes finding alternative functional forms with corresponding estimation techniques an interesting task.
3. The chosen MLP architecture is rather simple. Other architectures could be examined for their performance for the problem at hand.
4. The proposed claim count predictors are based on the the number of reported claims. This leads to the prediction becoming identical to zero if, in a particular portfolio, no claims were yet observed. By developing methods for estimating P_Y and λ, access to a predictor based on (15) would allow to overcome this disadvantage.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s13385-022-00314-4.

Acknowledgements Computational infrastructure and support were provided by the Centre for Information and Media Technology at Heinrich Heine University Düsseldorf. The dataset studied in Sect. 6 and computational infrastructure was provided by ARAG SE, Düsseldorf. The authors are grateful to two unknown referees, the co-editor and the editor for their useful comments on a previous version of this article.

Funding Open Access funding enabled and organized by Projekt DEAL. See Acknowledgements.

Availability of data, materials and code The script used for simulation studies is provided as supplementary material. It requires an R package hosted on GitHub [34]. The real dataset used in Sect. 6 is proprietary.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, and Zheng X (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/. (**Software available from tensorflow.org**)
2. Allaire J, Eddelbuettel D, Golding N and Tang Y (2016) tensorflow: R interface to tensorflow. https://github.com/rstudio/tensorflow
3. Andersen EB (1970) Asymptotic properties of conditional maximum-likelihood estimators. J Roy Stat Soc 32(2):283–301
4. Antonio K, Plat R (2014) Micro-level stochastic loss reserving for general insurance. Scand Actuar J 2014(7):649–669
5. Antonio K, Godecharle E and Van Oirbeek R (2016) A multi-state approach and flexible payment distributions for micro-level reserving in general insurance. https://doi.org/10.2139/ssrn.2777467
6. Arjas E (1989) The claims reserving problem in non-life insurance: some structural ideas. ASTIN Bull 19(2):139–152
7. Arvidsson S (2010) Reducing asymmetric information with usage-based automobile insurance. Swedish Natl Road Transp Res Inst (VTI) 2:2011
8. Baudry M, Robert CY (2019) A machine learning approach for individual claims reserving in insurance. Appl Stoch Model Bus Ind 2019(35):1127–1155
9. Chollet F, Allaire J, et al (2017) R interface to keras. https://github.com/rstudio/keras
10. De Felice M, Moriconi F (2019) Claim watching and individual claims reserving using classification and regression trees. Risks 7(4):102
11. Delong Ł, Wüthrich MV (2020) Neural networks for the joint development of individual payments and claim incurred. Risks 8(2):33
12. Delong Ł, Lindholm M and Wüthrich MV (2021) Collective reserving using individual claims data. Scandinavian Actuarial J 1–28
13. Delong Ł, Lindholm M, Wüthrich MV (2021) Gamma mixture density networks and their application to modelling insurance claim amounts. Insur Math Econ 101:240–261
14. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J Roy Stat Soc 39(1):1–38
15. Dozat T (2016) Incorporating Nesterov momentum into Adam. https://openreview.net/pdf/OM0jvwB8jIp57ZJjtNEZ.pdf
16. Embrechts P, Mikosch T, Klüppelberg C (1997) Modelling extremal events: for insurance and finance. Springer-Verlag, Berlin
17. Glorot X and Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Teh YW and Titterington M (eds) Proceedings of the thirteenth international conference on artificial intelligence and statistics, volume 9 of proceedings of machine learning research, pp 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR. http://proceedings.mlr.press/v9/glorot10a.html
18. Goodfellow IJ, Bengio Y and Courville A (2016) Deep learning. MIT Press, Cambridge. http://www.deeplearningbook.org

19. Gui W, Rongtan H, Lin X (2018) Fitting the erlang mixture model to data via a Gem-CMM algorithm. J Comput Appl Math 343:05
20. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning. Springer series in statistics. Springer LLC, New York
21. Holden L and Haug O (2009) A mixture model for unsupervised tail estimation. http://arxiv.org/abs/0902.4137
22. Jin X (2013) Micro-level loss reserving models with applications in workers compensation insurance. https://sites.google.com/site/xiaolijin2013/research/working-paper2
23. Kingma DP and Ba J (2015) Adam: a method for stochastic optimization. In: Proceedings of the third international conference on learning representations. ICLR'15. http://arxiv.org/abs/1412.6980
24. Last G, Penrose M (2018) Lectures on the poisson process. Cambridge University Press, Cambridge
25. Lee SCK, Lin XS (2010) Modeling and evaluating insurance losses via mixtures of erlang distributions. North Am Actuarial J 14(1):107–130
26. Lee SC, Lin XS (2012) Modeling dependent risks with multivariate erlang mixtures. ASTIN Bull 42(1):153–180
27. Liu C, Rubin DB (1994) The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence. Biometrika 81(4):633–648
28. Lopez O, Milhaud X (2021) Individual reserving and nonparametric estimation of claim amounts subject to large reporting delays. Scand Actuar J 1:34–53
29. Meng X-L, Rubin DB (1993) Maximum likelihood estimation via the ECM algorithm: a general framework. Biometrika 80(2):267–278
30. Mikosch T (2009) Non-life insurance mathematics, 2nd edn. Universitext. Springer-Verlag, Berlin
31. Norberg R (1993) Prediction of outstanding liabilities in non-life insurance. ASTIN Bull 23(1):95–115
32. Norberg R (1999) Prediction of outstanding liabilities II. Model variations and extensions. ASTIN Bull 29(1):5–25
33. Radtke M, Schmidt KD, Schnaus A (eds) (2016) Handbook on loss reserving. European Actuarial Academy (EAA) series. Springer, Cham
34. Rosenstock A (2021) Reservr. https://github.com/AshesITR/reservr
35. Verbelen R, Gong L, Antonio K, Badescu A, Lin S (2015) Fitting mixtures of erlangs to censored and truncated data using the EM algorithm. ASTIN Bull 45(3):729–758
36. Wüthrich MV and Buser C (2019) Data analytics for non-life insurance pricing. https://doi.org/10.2139/ssrn.2870308
37. Wüthrich MV (2018) Neural networks applied to chain-ladder reserving. Eur Actuar J 8(2):407–436
38. Wüthrich M (2018) Machine learning in individual claims reserving. Scand Actuar J 1–16:2018

Supplementary Material: Micro-level Prediction of Outstanding Claim Counts using Neural Networks

Axel Bücher [1*] and Alexander Rosenstock[1,2]

[1*] Mathematisches Institut, Heinrich-Heine-Universität Düsseldorf, Universitätsstraße 1, 40225 Düsseldorf, Germany.
[2] ARAG SE, ARAG-Platz 1, 40472 Düsseldorf, Germany.

*Corresponding author(s). E-mail(s): axel.buecher@hhu.de;
Contributing authors: alexander.rosenstock@hhu.de;

Abstract

This supplement contains more details on the neural network model and its estimation (Appendix A), on the general ECME-algorithm (Appendix B) and its adaptations for fitting Erlang mixtures (Appendix C), and on the model specifications used in the simulation study (Appendix D). Throughout, arabic section numbers always refer to the main paper.

Appendix A Details on the neural network architecture

In this section we provide a detailed description of the neural network architecture used for modelling the reporting delay distribution in Section 2.3.

Basics on neural networks can be found in [1]. Each function $g \in \mathcal{G}$ shall be a composition of different layer functions, each with their own set of independent parameters. For a general overview of the structure of g, we refer to Figure A1. It aids understanding to give an informal summary of the structure of $\mathcal{G}$ before diving into the formal definition:

- The *head* of the network consists of embedding layers for categorical features and scaling transformations for continuous features, which serves

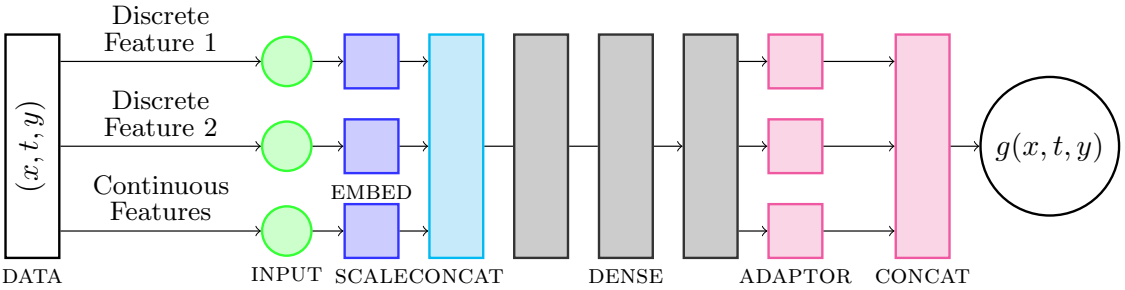


Figure A1: Schematic flow of input features to output for a neural network architecture with three hidden layers. The network will be trained with (9) as its loss.

to create a uniform-scale, continuous representation of all input features. The uniformity of scale is particularly important to conserve numerical stability in the presence of random initialization, see Section 3.3 for details.

- The *body* of the network aims at transforming the preprocessed features into an internal representation of the available information, based on one or more densely connected layers. This 'narrow' encoding of the data serves to extract the most important features with respect to the output variable (reporting delay in our case).
- The *tail* of the network aims at transforming the rich internal representation into concrete distributional parameters $\theta \in \Theta$. Intuitively, the tail doesn't need more complexity because all relevant structure is captured already in the internal representation.

We start by describing the *input layer*, which does not contain any free model parameters. Given an input $(x, t, y) \in \mathfrak{X} \times \mathbb{R} \times \mathfrak{Y}$, we first reorder the features, putting N_d discrete features to the front and N_c continuous features (e.g. t) to the back of a new intermediate vector. Furthermore, we shall code the ith discrete feature as an integer in $\{1, \ldots, c_i\}$ where $c_i \in \mathbb{N}$ is the number of different values that feature i can attain. Overall, we obtain the input layer function

$$g_{\text{in}} : \mathfrak{X} \times \mathbb{R} \times \mathfrak{Y} \to \prod_{i=1}^{N_d} \{1, \ldots, c_i\} \times \mathbb{R}^{N_c}$$

Next, for each of the discrete features, a so-called *embedding layer* is needed to transform the feature into a continuous variable for further use. For the ith discrete feature, the embedding function $g_{\text{embed},i} : \{1, \ldots, c_i\} \to \mathbb{R}^{d_i}$ has a fixed hyperparameter d_i, the *embedding dimension*, and $d_i \cdot c_i$ free parameters, denoted by $\alpha_1, \ldots, \alpha_{c_i}$ with $\alpha_j \in \mathbb{R}^{d_i}$, and is defined as

$$g_{\text{embed},i} : \{1, \ldots, c_i\} \to \mathbb{R}^{d_i}, \quad v \mapsto \alpha_v.$$

$g_{\text{embed},i}(v) = \alpha_v \in \mathbb{R}^{d_i}$ is called the *embedding of value* v. As illustrated in [2], continuous features should be brought to a common scale to improve

the numerical condition of the problem. This is realized with an affine transformation with parameters $\mu \in \mathbb{R}^{N_c}, \sigma \in \mathbb{R}_+^{N_c}$, namely $g_{\text{scale}} \coloneqq x \mapsto \frac{x-\mu}{\sigma}$ with component-wise division to center and scale continuous features. When fitting the neural network (see Section 3.3), we employ empirical means and standard deviations, respectively. The overall *embedding layer* of the neural network is defined as

$$g_{\text{embed}} : \prod_{i=1}^{N_d} \{1, \dots, c_i\} \times \mathbb{R}^{N_c} \to \mathbb{R}^{\sum_{i=1}^{N_d} d_i + N_c}, \quad g_{\text{embed}} \coloneqq \bigotimes_{i=1}^{N_d} g_{\text{embed},i} \otimes g_{\text{scale}}$$

which is a function with a total of $\sum_{i=1}^{N_d} c_i \cdot d_i$ free parameters. We write

$$g_{\text{head}} \coloneqq g_{\text{embed}} \circ g_{\text{in}} : \mathfrak{X} \times \mathbb{R} \times \mathfrak{Y} \to \mathbb{R}^{\sum_{i=1}^{N_d} d_i + N_c}.$$

After the head function that transforms all inputs into continuous variables, a series of *densely connected* layers (dense layers for short) are applied. The total number $N_{\text{dense}} \in \mathbb{N}$ of dense layers is to be considered as a hyperparameter of the model, and the same holds for the output dimension n_j, the *number of nodes*, of the jth layer. For brevity, we write $n_0 \coloneqq N_c + \sum_{i=1}^{N_d} d_i$. The jth dense layer is then defined as

$$g_{\text{dense},j} : \mathbb{R}^{n_{j-1}} \to \mathbb{R}^{n_j}, \quad v \mapsto g_{\text{dense},j}(v) \coloneqq \text{softplus}(Av + b),$$

where $A \in \mathbb{R}^{n_j \times n_{j-1}}$ and $b \in \mathbb{R}^{n_j}$ are the free parameters and where the nonlinear function $\text{softplus} : x \mapsto \log(1 + e^x)$, called the activation function, is applied coordinate-wise. The composition of the N_{dense} dense layers defines a function

$$g_{\text{body}} \coloneqq g_{\text{dense},N_{\text{dense}}} \circ \cdots \circ g_{\text{dense},1} : \mathbb{R}^{\sum_{i=1}^{N_d} d_i + N_c} \to \mathbb{R}^{n_{N_{\text{dense}}}}$$

with $\sum_{j=1}^{N_{\text{dense}}} (n_{j-1} + 1) \cdot n_j$ free parameters. When fitting the model (see Section 3.3), we also experimented with $g_{\text{dense},j}(v) \coloneqq \text{ReLU}(Av + b)$ but found results to be worse than with the smooth softplus activation function.

Composing g_{head} and g_{body} gives a function $g_{\text{body}} \circ g_{\text{head}} : \mathfrak{X} \times \mathbb{R} \times \mathfrak{Y} \to \mathbb{R}^{n_{N_{\text{dense}}}}$ which now needs to be mapped to the parameter set Θ of the chosen reporting delay distribution family $\mathcal{F}$ in order to obtain a usable family $\mathcal{G}$. A common feature of the families $\mathcal{F}$ that we invoke is that the parameter space Θ factors into (possibly bounded) intervals and probability parameters, that is parameters p constrained to $[0, 1]^d$ with $\|p\|_1 = 1$ (a method to cope with integer parameters present in Erlang mixtures and derived distributions is presented below). We can thus construct a natural *adapter family* of functions

$g_{\text{tail}} : \mathbb{R}^{n_{N_{\text{dense}}}} \to \Theta$ as follows: assume

$$\Theta = \mathbb{R}^{N_1} \times (0,\infty)^{N_2} \times (0,1)^{N_3} \times \prod_{i=1}^{N_P} [0,1]^{p_i} \cap S^1_{p_i},$$

where S^1_n denotes the unit sphere in $\mathbb{R}^n$ with respect to the 1-norm. Note that other interval constraints can be trivially reproduced by adding affine transformations, e.g. if $\theta_1 \in (a,b)$ is a constraint in the original parametrization, substitute $\tilde{\theta_1} = \frac{\theta_1 - a}{b-a} \in (0,1)$ as an equivalent parameter. The adaptor function for the $\text{BDEGP}_{\text{fix}}(n,m,\kappa,\varepsilon,\alpha,\xi)$ family used in the paper is obtained by setting $N_1 = 0, N_2 = 2, N_3 = 0, N_P = 3, p_1 = n+1, p_2 = m, p_3 = 2$ in this general parametrization and ordering the free parameters $(\theta, \sigma, p^{(\delta)}, p^{(e)}, p^{(b)})$.

With $n_{\text{tail}} = n_{N_{\text{dense}}+1} \coloneqq N_1 + N_2 + N_3 + \sum_{i=1}^{N_P} p_i$ the dimension of Θ, we define

$$\begin{aligned}
\text{sigmoid} &: \mathbb{R} \to (0,1), \quad x \mapsto \frac{\exp(x)}{\exp(x)+1} \\
\text{softmax}_{\mathbb{R}^n} &: \mathbb{R}^n \to (0,1)^n \cap S^1_n, \quad x \mapsto \Big(\exp(x_i) / \sum_{j=1}^{n} \exp(x_j) \Big)_{i=1}^{n} \\
f_{\text{adaptor}} &: \mathbb{R}^{n_{\text{tail}}} \to \Theta, \quad f_{\text{adaptor}} \coloneqq \text{Id}_{\mathbb{R}^{N_1}} \otimes \bigotimes_{i=1}^{N_2} \text{softplus} \otimes \bigotimes_{i=1}^{N_3} \text{sigmoid} \otimes \bigotimes_{i=1}^{N_P} \text{softmax}_{\mathbb{R}^{p_i}} \\
g_{\text{tail}} &: \mathbb{R}^{n_{N_{\text{dense}}}} \to \Theta, \quad v \mapsto f_{\text{adaptor}}(Av + b)
\end{aligned} \tag{S1}$$

where $A \in \mathbb{R}^{n_{\text{tail}} \times n_{N_{\text{dense}}}}$ and $b \in \mathbb{R}^{n_{\text{tail}}}$ are free parameters. Finally, $\mathcal{G}$ is defined as the collection of all $g = g_{\text{tail}} \circ g_{\text{body}} \circ g_{\text{head}}$. $\mathcal{G}$ is a family with

$$n_\Psi = \sum_{i=1}^{N_d} c_i \cdot d_i + \sum_{j=1}^{N_{\text{dense}}} (n_{j-1}+1) n_j + (n_{\text{dense}}+1) n_{\text{tail}}$$

free real parameters. We denote the natural parametrization of $\mathcal{G}$ by $\Psi = \mathbb{R}^{n_\Psi}$ and identify a network $g \in \mathcal{G}$ by its n_Ψ weights $\psi \in \Psi$.

The output layer link function f_{adaptor} has a preimage given by

$$f^{-1}_{\text{adaptor}} = \text{Id}_{\mathbb{R}^{N_1}} \otimes \bigotimes_{i=1}^{N_2} \text{softplus}^{-1} \otimes \bigotimes_{i=1}^{N_3} \text{sigmoid}^{-1} \otimes \bigotimes_{i=1}^{N_P} \text{softmax}^{-1}_{\mathbb{R}^{p_i}}$$

with $\text{softmax}^{-1}_{\mathbb{R}^{p_i}}(p) = \big(\log(p_j) - \log(\|p\|_\infty) \big)_{j=1}^{p_i}$ and $\|p\|_\infty$ the maximum norm of p (note that softplus and sigmoid are invertible). This is useful for choice of initial values based on a global estimate $\hat{\theta}$ during neural network initialization.

Algorithm S.1 Network Initialization

function InitialiseNNet($\mathcal{G}, \hat{\theta}$)
  $\psi \leftarrow ()$
  for $i = 1, \ldots, N_d$ **do** ▷ Initialise embedding layers
    Draw $\alpha \sim U[-0.05, 0.05]^{d_i \otimes c_i}$
    Append α to ψ
  end for
  for $j = 1, \ldots, N_{\text{dense}}$ **do** ▷ Initialise dense layers
    $l \leftarrow \sqrt{\frac{6}{n_j + n_{j-1}}}$
    Draw $A \sim U[-l, l]^{n_j \otimes n_{j-1}}$ ▷ [3]
    $b \leftarrow \vec{0} \in \mathbb{R}^{n_j}$
    Append (A, b) to ψ
  end for
  $b \leftarrow f^{-1}_{\text{adaptor}}(\hat{\theta})$ ▷ Initialize tail
  $A \leftarrow \text{Diag}(b) \cdot U[-0.1, 0.1]^{n_{\text{tail}} \otimes n_{N_{\text{dense}}}}$
  Append (A, b) to ψ
  Flatten $\psi \in \mathbb{R}^{n_\Psi}$
end function

Appendix B Derivation of the ECME Algorithm

In this section we provide a detailed motivation of Algorithm 1 given in Section 3.2. We follow the notation of the section in the paper.

First note that the conditional density of truncated observations is again of mixture type:

$$f_{X^t|L^t=\ell,U^t=u}(x) = \sum_{j=1}^{k} p_j \frac{f_{j;\theta_j}(x)}{F_{(p,\theta)}([\ell,u])} = \sum_{j=1}^{k} \tilde{p}_{j;p,\theta}(\ell,u) \tilde{f}_{j;\theta_j}(x;\ell,u), \qquad \ell \le x \le u, \tag{S2}$$

where, using the notation $F_{j;\theta_j}([\ell,u]) = \int_{[\ell,u]} f_{j;\theta_j}(z)\,\mathrm{d}\mu(z)$,

$$\tilde{p}_{j;p,\theta}(\ell,u) = p_j \cdot \frac{F_{j;\theta_j}([\ell,u])}{F_{(p,\theta)}([\ell,u])}, \qquad \tilde{f}_{j;\theta_j}(x;\ell,u) = \frac{f_{j;\theta_j}(x)}{F_{j;\theta_j}([\ell,u])}, \qquad \ell \le x \le u. \tag{S3}$$

The EM-algorithm for calculating a maximizer of $\ell(p,\theta|\mathfrak{I})$ is based on regarding the observations (ℓ_i, x_i, u_i) as being incomplete, in view of the fact we do not know from which conditional truncated component density $\tilde{f}_{j;\theta_j}(\,\cdot\,;\ell_i,u_i)$, see (S3), the observation $x_i \sim (X^t \mid (L^t,U^t) = (\ell_i,u_i))$ was simulated. More formally let $Z = Z(\ell,u) = (Z_1,\ldots,Z_k)$, conditioned on $(L^t,U^t) = (\ell,u)$, follow a multinomial distribution $Z|(L^t = \ell, U^t = u) \sim \text{Mult}(1, \tilde{p}_{1;p,\theta}(\ell,u),\ldots,\tilde{p}_{k;p,\theta}(\ell,u))$, i.e., $\Pr(Z = z|L^t = \ell, U^t = u) =$

$\tilde{p}^{z_1}_{1;p,\theta}(\ell,u)\cdot\ldots\cdot\tilde{p}^{z_k}_{k;p,\theta}(\ell,u)$ for $z=(z_1,\ldots,z_k)\in\{0,1\}^k$ with $\sum_{j=1}^k z_j=1$. If, conditional on $(L^t,U^t)=(\ell,u)$ and $Z_j=1$, the random variable X^t has density $\tilde{f}_{j;\theta_j}$ from (S3), then the conditional distribution of X^t given $(L^t,U^t)=(\ell,u)$ is precisely given by (S2). In view of this representation, we now regard each (x_i,ℓ_i,u_i) as an incomplete observation of $(x_i,\ell_i,u_i;z_{i,1},\ldots,z_{i,k})$ where $z_{i,j}=z_{i,j}(\ell_i,u_i)$ encodes the truncated component density $\tilde{f}_{j;\theta_j}$ from which x_i has been drawn. The respective conditional density of a generic complete observations of (X^t,Z) given $(L^t,U^t)=(\ell,u)$ becomes

$$\begin{aligned}f_{(X^t,Z)|(L^t=\ell,U^t=u)}(x,z)&=f_{X^t|(Z,L^t,U^t)=(z,\ell,u)}(x)\cdot\Pr(Z=z\mid L^t=\ell,U^t=u)\\&=\Big(\prod_{j=1}^k\tilde{f}_{j;\theta_j}(x;\ell,u)^{z_j}\Big)\cdot\Big(\prod_{j=1}^k\tilde{p}_{j;p,\theta}(\ell,u)^{z_j}\Big).\end{aligned}$$

As a consequence, the complete sample weighted conditional likelihood is given by

$$\begin{aligned}&\ell(p,\theta|\mathfrak{C})\\&=\sum_{i=1}^N\sum_{j=1}^k w_i z_{i,j}\Big[\log\tilde{p}_{j;p,\theta}(\ell_i,u_i)+\log f_{j,\theta_j}(x_i)-\log F_{j;\theta_j}([\ell_i,u_i])\Big]\\&=\sum_{j=1}^k\sum_{i=1}^N w_i z_{i,j}\Big[\log p_j+\log f_{j;\theta_j}(x_i)-\log\Big(p_jF_{j;\theta_j}([\ell_i,u_i])+\sum_{m\neq j}p_mF_{m;\theta_m}([\ell_i,u_i])\Big)\Big],\end{aligned}$$

where $\mathfrak{C}=(x_i,\ell_i,u_i,w_i;z_{i,1},\ldots,z_{i,k})_{i=1}^N$.

The E-step of the EM-algorithm now involves calculating the conditional expectation of the complete sample weighted conditional likelihood given the incomplete sample $\mathfrak{I}=(x_i,\ell_i,u_i,w_i)_{i=1}^N$. For that purpose, note that

$$\begin{aligned}P_j(x;p,\theta)&\equiv\Pr(Z_j=1\mid X^t=x,L^t=\ell,U^t=u)\\&=\frac{\tilde{p}_{j;p,\theta}(\ell,u)\cdot\tilde{f}_{j;\theta_j}(x;\ell,u)}{\sum_{m=1}^k\tilde{p}_{m;p;\theta}(\ell,u)\cdot\tilde{f}_{m;\theta_m}(x;\ell,u)}=\frac{p_j\cdot f_{j;\theta_j}(x)}{\sum_{m=1}^k p_m\cdot f_{m;\theta_m}(x)}\end{aligned}\tag{S4}$$

where the last equation follows from (S3). Next suppose that, at the tth iteration of the algorithm, we are given an estimate $(p^{(t)},\theta^{(t)})$. Then, under the assumption that $(p^{(t)},\theta^{(t)})$ is the true parameter that generated $\mathfrak{C}$, the conditional expectation of $\ell(p,\theta|\mathfrak{C})$ given the incomplete sample $\mathfrak{I}=\mathfrak{I}_1=(x_i,\ell_i,u_i)_{i=1}^N$ is given by

$$\begin{aligned}Q_{(p^{(t)},\theta^{(t)})}(p,\theta|\mathfrak{I})&\equiv\mathbb{E}_{(p^{(t)},\theta^{(t)})}[\ell(p,\theta|\mathfrak{C})\mid\mathfrak{I}]\\&=\sum_{i=1}^N\sum_{j=1}^k w_iP_j(x_i;p^{(t)},\theta^{(t)})\Big[\log p_j+\log f_{j;\theta_j}(x_i)\end{aligned}$$

$$- \log \Big(p_j F_{j;\theta_j}([\ell_i, u_i]) + \sum_{m \neq j} p_m F_{m;\theta_m}([\ell_i, u_i]) \Big) \Big], \tag{S5}$$

where we have used (S4). The M-step of the plain EM-algorithm [4] would now involve updating the parameter $(p^{(t)}, \theta^{(t)})$ by $(p^{(t+1)}, \theta^{(t+1)}) \in \arg\max_{(p,\theta)} Q_{(p^{(t)},\theta^{(t)})}(p, \theta|\mathfrak{I})$ and iterating until convergence. However, the M-step maximization problem is not feasible (which is essentially due to the random truncation), whence we propose to instead rely on a version of the EM-algorithm known as the ECME-algorithm [5]. The latter consists of dividing the M-step maximization problem into a series of $k+1$ lower-dimensional and feasible maximization problems (either 'ECM' or 'CM' steps), that are essentially based on successively maximizing $\theta_j \mapsto Q_{(p^{(t)},\theta^{(t)})}(p, \theta|\mathfrak{I})$ (ECM) and then $p \mapsto \ell(p, \theta|\mathfrak{I})$ (CM), holding all other parameters fixed:

Step 1. Maximize $Q_{(p^{(t)},\theta^{(t)})}(p, \theta|\mathfrak{I})$ subject to $g_1(p, \theta) = g_1(p^{(t)}, \theta^{(t)})$ where

$$g_1(p, \theta) = (p, (\theta_j)_{j \neq 1})$$

with solution $(p^{(t+1/(k+1))}, \theta^{(t+1/(k+1))})$.

Step 2. Maximize $Q_{(p^{(t)},\theta^{(t)})}(p, \theta|\mathfrak{I})$ subject to $g_2(p, \theta) = g_2(p^{(t+1/(k+1))}, \theta^{(t+1/(k+1))})$ where

$$g_2(p, \theta) = (p, (\theta_j)_{j \neq 2})$$

with solution $(p^{(t+2/(k+1))}, \theta^{(t+2/(k+1))})$.

$\vdots$

Step k. Maximize $Q_{(p^{(t)},\theta^{(t)})}(p, \theta|\mathfrak{I})$ subject to $g_k(p, \theta) = g_k(p^{(t+(k-1)/(k+1))}, \theta^{(t+(k-1)/(k+1))})$ where

$$g_k(p, \theta) = (p, (\theta_j)_{j \neq k})$$

with solution $(p^{(t+k/(k+1))}, \theta^{(t+k/(k+1))})$.

Step $k+1$. Maximize $\ell(p, \theta|\mathfrak{I})$ from (12) subject to $g_{k+1}(p, \theta) = g_{k+1}(p^{(t+k/(k+1))}, \theta^{(t+k/(k+1))})$ where

$$g_{k+1}(p, \theta) = \theta$$

with solution $(p^{(t+1)}, \theta^{(t+1)})$.

It can be shown that the space filling condition from Definition 2 in [6] is met (in that paper's notation we have $J_1(p, \theta) = \mathbb{R}^k \times \{0\} \times \mathbb{R}^{d_{\Theta_2}} \cdots \times \mathbb{R}^{d_{\Theta_k}}, \ldots, J_k(p, \theta) = \mathbb{R}^k \times \mathbb{R}^{d_{\Theta_1}} \times \cdots \times \mathbb{R}^{d_{\Theta_{k-1}}} \times \{0\}, J_{k+1}(p, \theta) = \{0\}^k \times \mathbb{R}^{d_\Theta}$, whence $J(p, \theta) = \bigcap_{j=1}^{k+1} J_j(p, \theta) = \{0\}$), and the algorithm is therefore known to converge to local maxima of $\ell(p, \theta|\mathfrak{I})$, given suitable regularity conditions.

Experimenting with the ECME-algorithm, we did in fact found a slight variant of the above algorithm that proved to converge more quickly in extensive computing experiments. The variant is based on modifying the jth step ($j = 1, \ldots, k$) described above as follows: instead of maximizing $(p, \theta) \mapsto Q_{(p^{(t)},\theta^{(t)})}(p, \theta|\mathfrak{I})$ subject to the given constraint, maximize

$$\theta_j \mapsto Q_{j;(p^{(t)},\theta^{(t)})}(\theta_j) = \sum_{i=1}^{N} w_i P_j(x_i; p^{(t)}, \theta^{(t)}) \Big[\log f_{j;\theta_j}(x_i) - \log F_{j;\theta_j}([\ell_i, u_i]) \Big]. \tag{S6}$$

Here, the change of the objective function may be motivated by rewriting

$$\begin{aligned} & Q_{(p^{(t)},\theta^{(t)})}(p, \theta|\mathfrak{I}) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{k} w_i P_j(x_i; p^{(t)}, \theta^{(t)}) \Big[\log \tilde{p}_{j;p,\theta}(\ell_i, u_i) + \log f_{j,\theta_j}(x_i) - \log F_{j;\theta_j}([\ell_i, u_i]) \Big] \\ &\approx \sum_{i=1}^{N} \sum_{j=1}^{k} w_i P_j(x_i; p^{(t)}, \theta^{(t)}) \Big[\log \tilde{p}_{j;p^{(t)},\theta^{(t)}}(\ell_i, u_i) + \log f_{j,\theta_j}(x_i) - \log F_{j;\theta_j}([\ell_i, u_i]) \Big], \end{aligned}$$

and maximizing the latter subject to the given constraints is equivalent to maximizing (S6). Note that the approximation in the last display is typically quite accurate for large t, when the algorithm is close to convergence.

Calculating a maximum of the function in (S6) may involve a further algorithm depending on some starting value, say $\theta_{j,0}$. Following the notation in (13) from the main paper, we rewrite any solution of such an algorithm as $\mathrm{CML}(\mathcal{F}_j, \mathfrak{I}_j, \theta_{j,0})$, where $\mathcal{F}_j = \{f_{j;\theta_j} : \theta_j \in \Theta_j\}$ denotes the jth (untruncated) component family, where $\mathfrak{I}_j = \mathfrak{I}_j^{(t)} := \{(x_i, \ell_i, u_i, w_i P_j(x_i; p^{(t)}, \theta^{(t)}))\}$ denotes re-weighted truncated data, and where $\theta_{j,0}$ denotes a starting value. In view of this notation, the overall algorithm may be summarized as in Algorithm 1 in the paper.

Appendix C Derivation of the Erlang Mixture adapted ECME Algorithm

In this section we will derive the adaptations for fitting Erlang Mixtures in the setting of Section 3.2. Recall that Erlang Mixture families are given by $\mathcal{F} = \{\sum_{i=1}^{k} p_i \cdot \Gamma_{\alpha_i,\theta} : p \in (0,1)^k, \|p\|_1 = 1, \theta \in (0,\infty), \alpha \in \mathbb{N}^k, \alpha_1 < \ldots < \alpha_k\}$.

We will start by providing details on a suitable version of the ECME when treating the shape parameters $\alpha \in \mathbb{N}^k$ as fixed and known. For some given interval truncated sample $\mathfrak{I} = \{(x_i, \ell_i, u_i, w_i) | \ell_i \le x_i \le u_i\}$ the goal is to find $(\hat{p}, \hat{\theta}) = \arg\max_{p,\theta} \ell(p, \theta|\mathfrak{I}, \alpha)$, where ℓ is the weighted conditional log likelihood function from (12) with $f_{(p,\theta)} = \sum_{j=1}^{k} p_j d\Gamma_{\alpha_j,\theta}$ and $F_{(p,\theta)}$ the respective

cdf. For solving the maximization problem, we propose to use an ECME algorithm that involves two maximization steps within each iteration. For that purpose note that, following the ideas at the beginning of Appendix B, the adaption from (S6) becomes

$$Q_{(p^{(t)},\theta^{(t)})}(\theta) \coloneqq \sum_{i=1}^{N}\sum_{j=1}^{k} w_i \cdot P_j(x_i; p^{(t)}, \theta^{(t)}) \Big[\log \mathrm{d}\Gamma_{\alpha_j,\theta}(x_i) - \log \Gamma_{\alpha_j,\theta}([\ell_i, u_i]) \Big].$$

Each iteration in the adapted ECME-Algorithm for Erlang Mixtures now consists of two steps:

Step 1. Maximize $Q_{(p^{(t)},\theta^{(t)})}(\theta)$ with solution $\theta^{(t+1/2)}$.

Step 2. Maximize $\ell(p, \theta|\mathfrak{I}, \alpha)$ subject to $g_2(p, \theta) = g_2(p^{(t)}, \theta^{(t+1/2)})$ where $g_2(p, \theta) = \theta$ with solution $(p^{(t+1)}, \theta^{(t+1)})$.

Note that the first step was decomposed into k separate steps when treating general mixtures that do not involve common parameters. The procedure is summarized in Algorithm S.2.

Algorithm S.2 Erlang Mixture ECME-Algorithm

function ErlangECME($\mathcal{F}, \mathfrak{I}, p_0, \theta_0, \alpha, \varepsilon$)
  $p \leftarrow p_0$
  $\theta \leftarrow \theta_0$
  $l \leftarrow -\infty$
  repeat
    $l_0 \leftarrow l$
    $\theta' \leftarrow \arg\max_{\theta'} Q_{(p,\theta)}(\theta')$ ▷ Erlang ECM-Step
    $\theta \leftarrow \theta'$
    $p \leftarrow \arg\max_{p} \ell(p, \theta|\mathfrak{I}, \alpha)$ ▷ Erlang CM-Step
    $l \leftarrow \ell(p, \theta, \alpha|\mathfrak{I})$
  until $l - l_0 < \varepsilon$ ▷ Likelihood converged
end function

Algorithm S.2 requires starting values p_0 and θ_0 and a fixed shape parameter α_0, for which we found a K-Means approach that is partly similar to [7] to work well:

1. Run K-Means on the observed data $\{x : (x, \ell, u) \in \mathfrak{I}_1\}$ – ignoring truncation – with prescribed number of components k. Denote the obtained cluster centers by $c_1 < c_2 < \cdots < c_k$.
2. Estimate the preliminary scale parameter $\delta = \min_j\{c_j - c_{j-1}\}$ with $c_0 = 0$. This choice ensures that all clusters are separated enough to obtain pairwise different shape parameters in the next step.
3. Use $\alpha_{0,j} = \mathrm{round}(c_i/\delta), j = 1, \ldots, k$ as starting values for the shape parameters.
4. Set $m = \max_{(x,\ell,u,w)\in\mathfrak{I}} x/\alpha_{0,k}$ such that $(0, m\alpha_{0,k}]$ covers all observations.

5. Initialise the mixture weights $p_{0,j} = \#(\{(x,\ell,u) \in \mathfrak{I}_1 : \alpha_{0,j-1}m < x \leq \alpha_{0,j}m\})/N$ (where $\alpha_{0,0} \coloneqq 0$) by tabulating, compare [8, Sec. 3.2].
6. Initialise θ_0 via moment matching – ignoring truncation, that is, let

$$\theta_0 = \frac{\bar{x}}{\sum_{j=1}^{k} p_{0;j}\alpha_{0,j}},$$

where $\bar{x} = \frac{1}{N}\sum_{(x,\ell,u,w)\in\mathfrak{I}} x$ is the observed mean value.

Finally, optimising the shape parameter essentially requires to solve an integer optimization problem. We propose to use a local shape search as in [8, Sec. 3.3]:

1. Compute a fit with starting values α_0, p_0, θ_0 as described above, and set $\alpha = \alpha_0$, with components $(\alpha_1, \ldots, \alpha_k)$.
2. Try fitting $(\alpha_1, \ldots, \alpha_{k-1}, \alpha_k + 1)$, and keep the new parameters if the log-likelihood improves, repeating until it no longer increases. Proceed increasing $\alpha_{k-1}, \ldots, \alpha_1$ one by one until no more improvements are found.
3. Try fitting $(\alpha_1 - 1, \alpha_2, \ldots, \alpha_k)$ and keep the new parameters if the log-likelihood improves, repeating until it no longer decreases. Proceed decreasing $\alpha_2, \ldots, \alpha_k$ one by one until no more improvements are found.
4. Go back to 2. if any improvement was found.

For steps 2. and 3. good starting values for p and θ are those from the currently selected parameters, since the potential changes in α are small for each step.

Appendix D True relationships in simulation

Throughout this section we provide details on the model specifications used within the simulation experiment in Section 5.

Baseline.

The following paragraph describes the relationships chosen for the baseline scenario. Other scenarios inherit their relationships from the baseline, changing only one relationship at a time.

The conditional claim feature distribution $P_Y(x,t) = P_Y(x)$ is defined in terms of the binary conditional distribution of `cc` and the conditional distribution of `severity`, conditioned on the variables shown:

$$\begin{aligned} P(\texttt{cc} = \text{material}|\texttt{ac},\texttt{power},\texttt{dens}) &= \text{logit}^{-1}\big(0.5 + 0.05\log(\texttt{dens}) - 0.1\cdot\min(10,\texttt{ac}) \\ &\qquad - 0.05\cdot\texttt{power} + 0.01\cdot\min(10,\texttt{ac})\cdot\texttt{power}\big), \\ &=: P^{\text{Baseline}}(\texttt{ac},\texttt{power},\texttt{dens}) \\ P(\texttt{severity} \in \cdot|\texttt{cc},\texttt{brand},\texttt{ac},\texttt{power}) &= \log\mathcal{N}(\mu,\sigma), \end{aligned}$$

where

$$\mu^{\text{Baseline}} \coloneqq \mu = 5 + 0.35\cdot\texttt{power} + 0.35\cdot(2-\texttt{ac})_+ - 0.05\cdot\mathbf{1}(\texttt{brand}\in\{\text{B1},\text{B2},\text{B12}\})$$

$$+ 1.0 \cdot \mathbf{1}(\texttt{brand} \in \{\text{B10}, \text{B11}\}),$$

$$\sigma^{\text{Baseline}} := \sigma = 9 - 0.01 \cdot (5 - \texttt{ac})_+ + 0.08 \cdot \texttt{power}.$$

The parametrization for the log-normal distribution is such that $X \sim \log\mathcal{N}(\mu, \sigma)$ means $\log X \sim \mathcal{N}(\mu, \sigma)$; in particular, $\mathbb{E}(X) = \exp(\mu + \sigma^2/2)$.

The reporting delay distribution $P_D(x, t, y) = P_D(x, y)$ is chosen as a $\text{BDEGP}(n = 1, m = 3, \kappa = 3 \cdot 365, \varepsilon = 365/2)$-distribution with parameters depending on $\texttt{dens}, \texttt{ac}, \texttt{cc}$ and $\texttt{severity}$ as follows: denoting

- p_0, the weight for δ_0
- p_1, the weight for $\Gamma(1, \theta)$
- p_2, the weight for $\Gamma(3, \theta)$
- p_3, the weight for $\Gamma(6, \theta)$
- p_4, the weight for $\text{GPD}(\kappa, \sigma, \xi)$
- θ, the common Erlang scale
- σ, the GPD scale
- ξ, the GPD shape

we set

$$\begin{aligned}
p_0 &= (1 - p_4)\,\text{logit}^{-1}\Big(-4 - 0.5 \cdot \mathbf{1}(\texttt{cc} = \text{material}) + 0.5 \cdot \mathbf{1}(\texttt{ac} \le 1 \wedge \texttt{cc} = \text{material}) \\
&\qquad - 0.25 \cdot \texttt{severity} \cdot \begin{cases} 10^{-3} & \texttt{cc} = \text{material} \\ 10^{-4} & \texttt{cc} = \text{injury} \end{cases} + 0.2 \cdot \min(1, |\texttt{age} - 45|/15)^2 \\
&\qquad + 0.01 \cdot \log(\texttt{dens})\Big) \\
&=: (1 - p_4) q_0^{\text{Baseline}} \\
p_1 &= (1 - p_4 - p_0) \cdot \text{logit}^{-1}\Big(1 - 0.5 \cdot \mathbf{1}(\texttt{cc} = \text{material}) + \mathbf{1}(\texttt{ac} \le 1 \wedge \texttt{cc} = \text{material}) \\
&\qquad - 2 \cdot \texttt{severity} \cdot \begin{cases} 10^{-3} & \texttt{cc} = \text{material} \\ 10^{-4} & \texttt{cc} = \text{injury} \end{cases} + 0.2 \cdot \min(1, 2 - \texttt{age}/25)_+\Big) \\
&=: (1 - p_4 - p_0) q_1^{\text{Baseline}} \\
p_2 &= (1 - p_4 - p_0 - p_1) \cdot \frac{p_1}{1 - p_4 - p_0} \\
p_3 &= 1 - p_4 - p_0 - p_1 - p_2 \\
p_4 &= \begin{cases} 0.0005 & \texttt{cc} = \text{material} \\ 0.02 & \texttt{cc} = \text{injury} \end{cases} \\
\theta &= \begin{cases} 30 & \texttt{cc} = \text{material} \\ 180 & \texttt{cc} = \text{injury} \end{cases} \\
\sigma &= \begin{cases} 180 & \texttt{cc} = \text{material} \\ 365 & \texttt{cc} = \text{injury} \end{cases} \\
\xi &= 0.2
\end{aligned}$$

Exposure Scenarios

There is a pool of $500,000$ risks from the original dataset, risk features of policies created at time t are drawn uniformly from the pool. In the exposure scenario, this pool for new risks is downsampled for $\texttt{ac} \leq 5$ by a factor of $f(t) = 0.1 + 0.9 \cdot (1 - t/3650)$ (2a) and $f(t) = 0.1 + 0.9 \cdot \mathbf{1}(t \geq 3650/2)$ (2b). That means, policies created at time t are drawn uniformly from a modified pool consisting of all $244,230$ original risks with $\texttt{ac} > 5$, and $f(t) \cdot 255,770$ risks with $\texttt{ac} \leq 5$ where the original dataset consists of $255,770$ risks with $\texttt{ac} \leq 5$.

Claim Intensity Scenarios

The baseline claim intensity of $\lambda(x,t) = \texttt{truefreq}$ is reduced by 20% for all risks:

$$\lambda(x,t) = \texttt{truefreq} \cdot (1 - 0.2 \cdot t/3650), \tag{3a}$$

$$\lambda(x,t) = \texttt{truefreq} \cdot (1 - 0.2 \cdot \mathbf{1}(t \geq 3650/2)). \tag{3b}$$

Occurence Process Scenarios

The parameters for the claim feature distributions (`cc` and `severity`) are changed depending on t: writing $p = P(\texttt{cc} = \text{material} | \texttt{ac}, \texttt{power}, \texttt{dens}, t)$, we have

$$\begin{cases} p = \text{logit}^{-1}\big(\text{logit}(P^{\text{Baseline}}(\texttt{ac}, \texttt{power}, \texttt{dens})) + 0.9t/3650\big), \\ \mu = \mu^{\text{Baseline}} + \begin{cases} 1.0 & \texttt{cc} = \text{material} \\ -0.5 & \texttt{cc} = \text{injury} \end{cases} \cdot t/3650, \\ \sigma = \sigma^{\text{Baseline}} + 0.5 \cdot \mathbf{1}(\texttt{cc} = \text{injury}) \cdot t/3650, \end{cases} \tag{4a}$$

$$\begin{cases} p = \text{logit}^{-1}\big(\text{logit}(P^{\text{Baseline}}(\texttt{ac}, \texttt{power}, \texttt{dens})) + 0.9 \cdot \mathbf{1}(t \geq 3650/2)\big), \\ \mu = \mu^{\text{Baseline}} + \begin{cases} 1.0 & \texttt{cc} = \text{material} \\ -0.5 & \texttt{cc} = \text{injury} \end{cases} \cdot \mathbf{1}(t \geq 3650/2), \\ \sigma = \sigma^{\text{Baseline}} + 0.5 \cdot \mathbf{1}(\texttt{cc} = \text{injury}, t \geq 3650/2). \end{cases} \tag{4b}$$

Reporting Process Scenarios

The baseline probabilities p_0 and p_1 are modified on the logit scale, keeping all other relationships in tact (i.e. p_4 does not change, p_2 and p_3 are defined via the modified p_0 and p_1 in the same way as in the baseline scenario):

$$\begin{cases} p_0 = (1 - p_4) \cdot \text{logit}^{-1}(\text{logit}(q_0^{\text{Baseline}}) + 2t/3650), \\ p_1 = (1 - p_4 - p_0) \cdot \text{logit}^{-1}(\text{logit}(q_1^{\text{Baseline}} + 2t/3650)), \end{cases} \tag{5a}$$

$$\begin{cases} p_0 = (1 - p_4) \cdot \text{logit}^{-1}(\text{logit}(q_0^{\text{Baseline}}) + 2 \cdot \mathbf{1}(t \geq 3650/2)), \\ p_1 = (1 - p_4 - p_0) \cdot \text{logit}^{-1}(\text{logit}(q_1^{\text{Baseline}} + 2 \cdot \mathbf{1}(t \geq 3650/2))). \end{cases} \tag{5b}$$

References

[1] Goodfellow, I.J., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, MA, USA (2016). http://www.deeplearningbook.org

[2] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15, pp. 448–456. JMLR.org, Lille, France (2015)

[3] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy (2010). http://proceedings.mlr.press/v9/glorot10a.html

[4] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society. Series B (Methodological) **39**(1), 1–38 (1977)

[5] Liu, C., Rubin, D.B.: The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence. Biometrika **81**(4), 633–648 (1994)

[6] Meng, X.-L., Rubin, D.B.: Maximum likelihood estimation via the ECM algorithm: A general framework. Biometrika **80**(2), 267–278 (1993)

[7] Gui, W., Rongtan, H., Lin, X.: Fitting the erlang mixture model to data via a gem-cmm algorithm. Journal of Computational and Applied Mathematics **343** (2018)

[8] Lee, S.C.K., Lin, X.S.: Modeling and evaluating insurance losses via mixtures of erlang distributions. North American Actuarial Journal **14**(1), 107–130 (2010)

Supplementary Material: Mirco-level Prediction of Outstanding Claim Counts using Neural Networks

Axel Bücher and Alexander Rosenstock

An R-File containing the functions used for the simulation study can be found on GitHub under the following link:

https://gist.github.com/AshesITR/d3023021e3e86ceb7ad661e91d5bc458

Combined modelling of micro-level outstanding claim counts and individual claim frequencies in general insurance

Axel Bücher, Alexander Rosenstock

7th September 2023

Abstract

Usually, the actuarial problems of predicting the number of claims incurred but not reported (IBNR) and of modelling claim frequencies are treated successively by insurance companies. New micro-level methods are proposed that address the two problems simultaneously. The methods are based on an elaborated occurence process model that includes both a claims frequency model and a claim development model. The influence of claim feature variables is modelled by suitable neural networks. Extensive simulation experiments and a case study on a large real data set from a motor legal insurance portfolio show accurate predictions at both the aggregate and individual policy level, as well as appropriate fitted models for claim frequencies. Moreover, a novel alternative approach combining data from classic triangle-based methods with a micro-level frequency model is introduced and compared to the full micro-level approach.

Contents

1 Introduction **2**

2 Preliminaries on insurance portfolio data **3**

3 Modelling and Fitting Claim Arrival Processes **4**
3.1 Modelling and Fitting the Reporting Delay Distribution 4
3.2 Modelling and Fitting the Claim Feature Distribution 5
3.3 Modelling and Fitting the Claim Intensity 7

4 Individual Claims Count Prediction based on Estimated Claim Arrival Processes **8**

5 Individual Claims Count Prediction based on Chain Ladder Networks for the Claim Frequency **9**

6 Evaluating Individual Claim Count Predictors **11**

7 Simulation study **12**
7.1 Training Procedure 13
7.2 Predictors 16
7.3 Results 18

8 Application to real data **22**
8.1 The Dataset 22
8.2 Predictors 23
8.3 Results 24

Acknowledgements

Computational infrastructure and support were provided by the Centre for Information and Media Technology at Heinrich Heine University Düsseldorf. The dataset studied in Section 8 and computational infrastructure was provided by ARAG SE, Düsseldorf.

1 Introduction

Actuarial departments in insurance companies are responsible for many different tasks related to risk modelling of insurance operations. Two of these tasks are closely interlinked: Reserving actuaries need to compute accurate predictions of incurred liabilities for insurance portfolios. Pricing actuaries use these results to develop risk models, which allow them to estimate expected losses for a range of policy parameters and thus to determine the prices at which these policies are sold. While developing models on per-policy data (subsequently referred to as the *micro-level*) is common practice for risk modelling, micro-level methods for reserving are still not widely adopted in the industry.

In the research literature on reserving in general insurance, there are two main approaches to micro-level reserving. On the one hand, there are methods that operate in discrete time (such as development years), which have similarities with macro-level reserving methods like Bornhuetter-Ferguson. Micro-level methods of this kind typically deal with claim-level information which is only available for reported claims, and therefore work on the reserve of claims that have been reported but not settled (RBNS). Examples of this kind are De Felice and Moriconi (2019), Baudry and Robert (2019), Gabrielli and Wüthrich (2018), Wüthrich and Merz (2015) and Chaoubi et al. (2023), among others. On the other hand, there are continuous time claim development models which are based on the assumption that claim development is regarded as a point process (e.g., a position-dependent marked poisson process); respective model parameters are then estimated from observed data. This strain of research goes back to Norberg (1993) and Norberg (1999). More recent papers studying a continuous time claim development model are Badescu et al. (2016), Antonio, Godecharle et al. (2016), Okine et al. (2022) and Wang et al. (2021), among others.

The current paper aims to combine the task of predicting IBNR claim counts on a policy level with the development of a matching micro-level claim frequency model, thus addressing the two tasks of reserving and risk modelling simultaneously in the hope of improving accuracy in both disciplines. Two main approaches are proposed to achieve this goal. Our first approach is based on an explicit micro-level claim occurrence process model inspired by Norberg (1993). Suitable methods are proposed for estimating all model parameters, which results in a fully fitted continuous-time process model that includes a claim frequency model. We thereby extend a related approach in Bücher and Rosenstock (2022a), where a submodel for reporting delays was studied. Secondly, we propose a new method that uses classical triangle-level reserving methods as input to the estimation of a micro-level claim frequency model. This approach allows for micro-level allocation of IBNR claim counts that is consistent with the triangle. Due to the nature of the underlying triangle-based reserving methods, the approach only allows discrete time steps.

For each of the two approaches, we discuss the design and estimation of suitable (parametric) sub-models involving classical multilayer perceptron neural networks. For the estimation, it is taken into account that the observed data is subject to random truncation. Predictors for the number of IBNR claims on a per-policy level are derived from the estimated models. They are compared with each other as well as with classical Chain Ladder methods in an extensive simulation study as well as on a real dataset. It is found that the new predictors provide accurate predictions as well as appropriate fitted models for claim

frequencies even in simulation scenarios involving non-homogeneous portfolios and in the real-life example.

For learning the neural networks, we utilize the TensorFlow framework (Abadi et al. 2015) with custom loss functions that take random truncation into account. The implementation is written using the R language and its interface binding packages `keras` and `tensorflow` (Allaire et al. 2016; Chollet, Allaire et al. 2017) to utilize TensorFlow. Core functionality used for estimation and prediction is freely available as an R package `reservr` (Rosenstock 2023).

The remaining parts of the paper are organized as follows. In Section 2, we introduce the micro-level claim process used throughout as well as the observation setting. Modelling and fitting of the individual components of the claim process are discussed in Section 3. Based on a completely estimated micro-level model, we derive a corresponding micro-level predictor for the IBNR claim count in Section 4. Section 5 introduces an alternative micro-level predictor based on similar ideas, but using a triangle-based global reserving method and discrete time steps. After defining performance metrics in Section 6, we present results on a large-scale simulation study in Section 7. An application to a real dataset from a motor legal insurance portfolio is presented in Section 8. Finally, Section 9 concludes.

2 Preliminaries on insurance portfolio data

The general model for the claim arrivals in a given insurance portfolio is the same as in Bücher and Rosenstock (2022a), and builds on the notion of *(position-dependent) marked Poisson processes* (Norberg 1993). More precisely, we consider an insurance portfolio containing N_{pol} independent risks. Each risk is described by a coverage period $C = [t_{\text{start}}, t_{\text{end}}]$, and by risk features $\bar{x} \in \bar{\mathfrak{X}}$, where $\bar{\mathfrak{X}}$ is a feature space containing both discrete and continuous features; for example, information on the insured product and chosen options such as deductibles. We write $x = (C, \bar{x}) \in \mathfrak{X} = \{\text{intervals on } [0,\infty)\} \times \bar{\mathfrak{X}}$, and assume that x is constant over the course of the contract. In practice, risk features do change over time, but not very often, whence such a contract could be modelled as two separate risks.

Each risk can potentially incur claims during its coverage period, which will formally be modelled by a claim arrival process. Each claim in that process occurs at some (calendar) accident time $t_{\text{acc}} \in [t_{\text{start}}, t_{\text{end}}]$, and will be associated with several claim features $y \in \mathfrak{Y}$ like the type of the claim or its severity; here, $\mathfrak{Y}$ denotes some suitable feature space. Moreover, the claim will not be immediately known to the insurer, but it will rather be reported at (calendar) reporting time $t_{\text{report}} \in [t_{\text{acc}}, \infty)$. Formally, both the claim features y and the reporting delay $d_{\text{report}} := t_{\text{report}} - t_{\text{acc}}$ will be assumed to be a mark on the claim arrival process. Following the notation in Last and Penrose (2018), we arrive at the following definition of a claim arrival process.

Definition 2.1 (Claim Arrivals and Portfolio)**.** *Consider a risk in the portfolio with risk features $x^{(i)} \in \mathfrak{X}$ among which we find the coverage period $C(x^{(i)})$. The claim arrival process associated with that risk is a position-dependent marked Poisson process with $N^{(i)} \sim \text{Poi}\big(\int_{C(x^{(i)})} \lambda(x^{(i)}, t)\,\mathrm{d}t\big)$ points*

$$\xi^{(i)} = \sum_{j=1}^{N^{(i)}} \delta_{(T^{(i)}_{\text{acc},j}, Y^{(i)}_j, D^{(i)}_{\text{report},j})}$$

on $[0,\infty) \times \mathfrak{Y} \times [0,\infty)$ with:

(i) Intensity $\lambda(x^{(i)}, t) \cdot \mathbf{1}(t \in C(x^{(i)}))$, i.e., for all intervals $[t_0, t_1] \subseteq [0,\infty)$, we have

$$\sum_{j=1}^{N^{(i)}} \mathbf{1}(T^{(i)}_{\text{acc},j} \in [t_0, t_1]) = \int_{t_0}^{t_1} \xi^{(i)}(\mathrm{d}t, \mathfrak{Y}, [0,\infty)) \sim \text{Poi}\left(\int_{t_0}^{t_1} \mathbf{1}(t \in C(x^{(i)}))\lambda(x^{(i)}, t)\,\mathrm{d}t\right).$$

(ii) Conditional claim feature distribution $P_{Y|x^{(i)},t} = P_{Y|X=x^{(i)},T_{\text{acc}}=t}$. *Here,* Y *denotes a generic* $\mathfrak{Y}$*-valued claim feature variable containing all claim features except for the reporting delay, while* X *and* T_{acc} *are generic risk feature and accident time variables, respectively.*

(iii) Conditional reporting delay distribution $P_{D|x^{(i)},t,y} = P_{D|X=x^{(i)},T_{\text{acc}}=t,Y=y}$. *Here,* $D = D_{\text{report}}$ *denotes a generic reporting delay variable in* $[0,\infty)$.

A portfolio consists of N_{pol} *risks* $\xi^{(1)},\dots,\xi^{(N_{\text{pol}})}$ *that are mutually independent.*

In practice, the three building blocks of Definition 2.1, i.e., $\lambda(x,t), P_{Y|x,t}$ and $P_{D|x,t,y}$, are unknown and must be estimated based on observational data that is available to the insurer at some given calendar time $\tau \geq 0$ of observing the portfolio. More precisely, the insurer observes data from Observation Scheme 1:

Observation Scheme 1. *At given calendar time* τ*, the available dataset* $\mathfrak{D} = \mathfrak{D}_\tau$ *consists of all risk features* $x^{(i)}$*,* $i \in \{1,\dots,N_{pol}\}$*, and all reported claim data up to calendar time* τ*, i.e.*

$$\left\{(x^{(i)}, t^{(i)}_{\text{acc},j}, y^{(i)}_j, d^{(i)}_{\text{report},j}) \,\middle|\, t^{(i)}_{\text{report},j} := d^{(i)}_{\text{report},j} + t^{(i)}_{\text{acc},j} \leq \tau\right\}. \tag{2.1}$$

Equivalently, we observe, for each $i \in \{1,\dots,N_{pol}\}$*, the risk feature* $x^{(i)}$ *and the restriction* $\xi^{(i)}_r(\cdot) = \xi^{(i)}(\cdot \cap R_\tau)$*, where* $R_\tau = \{(t,y,d) : d+t \leq \tau\}$ *and where the lower index* r *stands for 'reported'.*

Clearly, estimating the building blocks of Definition 2.1 can only be feasible if additional model assumptions are made. Those assumptions, as well as respective fitting procedures, will be described in the next section.

3 Modelling and Fitting Claim Arrival Processes

Modelling and estimating the intensity $\lambda(x,t)$, the claim feature distribution $P_{Y|x,t}$ and the reporting delay distribution $P_{D|x,t,y}$ from Definition 2.1 will be done iteratively, starting with the latter. We discuss each aspect in a separate subsection.

3.1 Modelling and Fitting the Reporting Delay Distribution

Modelling and fitting the reporting delay distribution has recently been considered in the accompanying paper Bücher and Rosenstock (2022a), subsequently abbreviated as BR22. A detailed description of all aspects is rather lengthy and not very informative for this paper, which is why we will be rather brief in our discussion. It is worthwhile to mention that any other method to obtain an estimate for $P_{D|x,t,y}$ may be employed as well.

BR22 start by discussing stylized facts of reporting delays. The discussion eventually gave rise to a micro-level model where the conditioning variables x,t,y enter via a suitable multilayer perceptron (MLP), i.e., a certain fully connected feedforward artificial neural network, into a flexible parametric mixture model.

Model 1 (Micro-Level Model for Reporting Delays)**.** *For some given parameters* $n,m \in \mathbb{N}_0$ *and* $\kappa,\varepsilon > 0$*, let* B_θ *denote the Blended Dirac-Erlang-Generalized Pareto distribution with parameter* $\theta \in \Theta = \Theta_{n,m,\kappa,\varepsilon}$ *(see Definition 2 in BR22 for a precise definition of the distribution and the parameter space). Further, let* $\mathcal{G}$ *denote a set of MLPs* $g : \mathfrak{X} \times [0,\infty) \times \mathfrak{Y} \to \Theta$ *(see Section 3.2 in BR22). We assume that the conditional reporting delay distribution satisfies, for some* $g \in \mathcal{G}$*,*

$$P_{D|x,t,y} = B_{g(x,t,y)} \qquad \forall\, x,t,y.$$

Details on fitting Model 1 to right-truncated observations as in Observation Scheme 1 can be found in Section 3 in Bücher and Rosenstock (2022a). A publicly available implementation is provided in Rosenstock (2023).

3.2 Modelling and Fitting the Claim Feature Distribution

Throughout this section, we assume that $P_{D|x,t,y}$ is available, for instance since it has been estimated as described in the previous section. For modelling and estimating the claim feature distribution $P_{Y|x,t}$, we assume that $\mathfrak{Y}$ can be written as a Q-fold cartesian product $\mathfrak{Y} = \mathfrak{Y}_1 \times \cdots \times \mathfrak{Y}_Q$ with $\mathfrak{Y}_q \subset \mathbb{R}$ for each $q \in \{1, \dots, Q\}$. Note that this assumption is of no practical concern, since claims data typically consists of a combination of real-valued and categorical features (if the qth feature is categorical, its categories may be identified with $1, \dots, n_q$). As a consequence, we may write a generic claim feature variable Y as $Y = (Y_1, \dots, Y_Q)$, and by the chain rule for conditional distributions we can decompose the conditional claim feature distribution as

$$P_{Y|X,T} = P_{Y_Q|X,T,Y_1,\dots,Y_{Q-1}} \cdot \ldots \cdot P_{Y_2|X,T,Y_1} \cdot P_{Y_1|X,T}, \tag{3.1}$$

where $T = T_{\text{acc}}$. This decomposition will be crucial for defining a flexible model for which iterative estimation (from left to right) is feasible.

Model 2 (Micro-Level Model for Claim Features)**.** *Assume that $P_{Y|X,T}$ allows for a decomposition as in (3.1) with $Q \in \mathbb{N}$. For each $q \in \{1, \dots, Q\}$, let $\mathcal{P}^{(q)} = \{P_\theta^{(q)} : \theta \in \Theta^{(q)}\}$ denote a parametric distribution family that is dominated by a sigma-finite measure $\mu^{(q)}$ and that has a finite-dimensional parameter space $\Theta^{(q)}$; the $\mu^{(q)}$-densities of $P_\theta^{(q)}$ will be written as $f_\theta^{(q)}$. Further, let $\mathcal{G}^{(q)}$ denote a set of MLPs $g^{(q)} : \mathfrak{X} \times [0, \infty) \times \mathfrak{Y}_1 \cdots \times \mathfrak{Y}_{q-1} \to \Theta^{(q)}$. We assume that there exists $g^{(q)} \in \mathcal{G}^{(q)}$ such that*

$$P_{Y_q|x,t,y_1,\dots,y_{q-1}} = P^{(q)}_{g^{(q)}(x,t,y_1,\dots,y_{q-1})} \qquad \forall\, x, t, y_1, \dots, y_{q-1}.$$

A natural choice for $\mathcal{P}^{(q)}$ in case of a categorical component is the multinomial distribution, which gives full flexibility. Distributions for continuous components must be decided case-by-case, bearing in mind that integration with respect to the chosen distribution will need to be performed. Therefor, choosing an overly flexible distribution could lead to numerical problems in application.

We will now describe how to iteratively estimate the unknown components of Model 2, taking into account the fact that observations are subject to random truncation. Suppose we have already fitted $P_{D|x,t,y}, P_{Y_Q|x,t,y_1,\dots,y_{Q-1}}, \dots, P_{Y_{q+1}|x,t,y_1,\dots,y_q}$, and we are to estimate $P_{Y_q|x,t,y_1,\dots,y_{q-1}}$ next.

For that purpose, we propose to maximize the following weighted conditional likelihood function over all functions $g \in \mathcal{G}^{(q)}$:

$$\tilde{\mathcal{L}}(g|\mathfrak{D}_\tau) = \sum_{(x,t,y,d) \in \mathfrak{D}_\tau} \tilde{\ell}_{(x,t,y_1,\dots,y_{q-1})}(g|y_q), \tag{3.2}$$

where

$$\tilde{\ell}_{(x,t,y_1,\dots,y_{q-1})}(g|y_q) = \frac{\log f^{(q)}_{g(x,t,y_1,\dots,y_{q-1})}(y_q)}{P(T + D \le \tau | X = x, T = t, Y_1 = y_1, \dots, Y_q = y_q)}. \tag{3.3}$$

Note that the denominator in the definition of $\tilde{\ell}$ does not depend on g, but only on objects that have already been fitted. Indeed, writing $y^{(q)} = (y_1, \dots, y_q)$, we have

$$\begin{aligned} &P(T + D \le \tau | X = x, T = t, Y^{(q)} = y^{(q)}) \\ &= \int_{\mathfrak{Y}_{q+1}} \cdots \int_{\mathfrak{Y}_Q} P_{D|x,t,y}([0, \tau - t]) \mathrm{d}P_{Y_Q|x,t,y^{(Q-1)}}(y_Q) \dots \mathrm{d}P_{Y_{q+1}|x,t,y^{(q)}}(y_{q+1}), \end{aligned} \tag{3.4}$$

which can readily be computed for each observation $(x, t, y, d) \in \mathfrak{D}_\tau$, subject to computational constraints.

The estimator for g may be motivated as follows: first of all, standard heuristics underlying the M-estimation principle suggest that a maximizer of $\tilde{\mathcal{L}}$ may be considered as an estimator for the maximizer of the (conditional) expected value $g \mapsto L(g) := \mathbb{E}_{Y_q|T+D\le\tau,x,t,y^{(q-1)}}[\tilde{\mathcal{L}}(g|\mathfrak{D}_\tau)]$, where $\mathbb{E}_{Y_q|T+D\le\tau,x,t,y^{(q-1)}}$ refers to integration with respect to the true conditional distribution of Y_q given $T+D\le\tau, X=x, T=t, Y^{(q-1)}=y^{(q-1)}$, for each observation. More precisely, we have

$$L(g) = \sum_{(x,t,y,d)\in\mathfrak{D}_\tau} \mathbb{E}[\tilde{\ell}_{(x,t,y_1,\dots,y_{q-1})}(g|Y_q) \mid T+D\le\tau, X=x, T=t, Y^{(q-1)}=y^{(q-1)}]. \tag{3.5}$$

The following lemma characterizes the maximizers of $g \mapsto L(g)$.

Lemma 3.1. *Assume that there is a true function $g_0 = g_0^{(q)} \in \mathcal{G}^{(q)}$ such that*

$$P_{Y_q|x,t,y_1,\dots,y_{q-1}} = P^{(q)}_{g_0(x,t,y_1,\dots,y_{q-1})} \qquad \forall\, x,t,y_1,\dots,y_{q-1}.$$

Then, for each fixed value of $x,t,y^{(q-1)}$, the summands of the objective function from (3.5)

$$g \mapsto \mathbb{E}[\tilde{\ell}_{(x,t,y_1,\dots,y_{q-1})}(g|Y_q) \mid T+D\le\tau, X=x, T=t, Y^{(q-1)}=y^{(q-1)}]$$

attain their maximal value at $g=g_0$.

Proof. For $x\in\mathfrak{X}, t\ge 0$ and $y=(y_1,\dots,y_Q)\in\mathfrak{Y}$, write $z^{(q)}=(x,t,y_1,\dots,y_q)$. Then, in view of the fact that the conditional density of Y_q given $D+T\le\tau$ and $Z^{(q-1)}=z^{(q-1)}$ may be written as

$$f_{Y_q|D+T\le\tau,Z^{(q-1)}=z^{(q-1)}}(y_q) = \frac{P(D+T\le\tau|Z^{(q)}=z^{(q)})}{P(D+T\le\tau|Z^{(q-1)}=z^{(q-1)})} f_{Y_q|Z^{(q-1)}=z^{(q-1)}}(y_q),$$

we may rewrite each summand in (3.5) as

$$\begin{aligned}
&\mathbb{E}[\tilde{\ell}_{(X,T,Y_1,\dots,Y_{q-1})}(g|Y_q) \mid T+D\le\tau, Z^{(q-1)}=z^{(q-1)}]\\
&= \int \frac{\log f_{g(z^{(q-1)})}(y_q)}{P(T+D\le\tau|Z^{(q)}=z^{(q)})} f_{Y_q|D+T\le\tau,Z^{(q-1)}=z^{(q-1)}}(y_q)\,\mathrm{d}\mu^{(q)}(y_q)\\
&= \frac{1}{P(D+T\le\tau|Z^{(q-1)}=z^{(q-1)})}\int \log f_{g(z^{(q-1)})}(y_q) f_{Y_q|Z^{(q-1)}=z^{(q-1)}}(y_q)\,\mathrm{d}\mu^{(q)}(y_q)\\
&= \frac{1}{P(D+T\le\tau|Z^{(q-1)}=z^{(q-1)})}\mathbb{E}[\log f_{g(Z^{(q-1)})}(Y_q) \mid Z^{(q-1)}=z^{(q-1)}].
\end{aligned}$$

Note that the factor in front of the expectation does not depend on g.

Write

$$M(g) = \mathbb{E}\big[\log f_{g(Z^{(q-1)})}(Y_q) - \log f_{g_0(Z^{(q-1)})}(Y_q) \mid Z^{(q-1)}=z^{(q-1)}\big],$$

and note that $M(g_0)=0$. Moreover, since $\log(x)\le 2(\sqrt{x}-1)$ for $x\ge 0$, we have, for all $g\in\mathcal{G}^{(q)}$,

$$\begin{aligned}
M(g) &\le 2\cdot\mathbb{E}\Big[\sqrt{f_{g(Z^{(q-1)})}(Y_q)/f_{g_0(Z^{(q-1)})}(Y_q)} - 1 \mid Z^{(q-1)}=z^{(q-1)}\Big]\\
&= 2\int_{\mathfrak{Y}_q}\Big(\sqrt{f_{g(z^{(q-1)})}(y_q)/f_{g_0(z^{(q-1)})}(y_q)} - 1\Big) f_{g_0(z^{(q-1)})}(y_q)\,\mathrm{d}\mu^{(q)}(y_q)\\
&= 2\int_{\mathfrak{Y}_q}\sqrt{f_{g(z^{(q-1)})}(y_q) f_{g_0(z^{(q-1)})}(y_q)}\,\mathrm{d}\mu^{(q)}(y_q) - 2\\
&= -\int_{\mathfrak{Y}_q}\Big(\sqrt{f_{g(z^{(q-1)})}(y_q)} - \sqrt{f_{g_0(z^{(q-1)})}(y_q)}\Big)^2\,\mathrm{d}\mu^{(q)} \le 0.
\end{aligned}$$

Hence, $M(g)\le 0 = M(g_0)$ for all $g\in\mathcal{G}^{(q)}$, which implies the assertion. □

During preliminary simulation experiments we found that more reliable estimates with a smaller variance may be obtained by smoothing the denominator in (3.3). This requires additional assumptions on top of Definition 2.1, the *local homogeneity* assumptions.

Assumption 1 (Local homogeneity of claims developement)**.** *Let $p > 0$ be a given period length measured in days; e.g., $p = 365$ days. For all intervals $I_p(k) = [k \cdot p, (k+1) \cdot p)$ with midpoints $t_k = k \cdot p + \frac{1}{2}, k \in \mathbb{N}_0$, we have:*
(i) $t \mapsto \lambda(x,t) = \lambda(x,t_k) > 0$ *is constant on* $I_p(k)$ *for any* x.
(ii) $t \mapsto P_{Y|x,t} = P_{Y|x,t_k}$ *is constant on* $I_p(k)$ *for any* x.
(iii) $t \mapsto P_{D|x,t,y} = P_{D|x,y,t_k}$ *is constant on* $I_p(k)$ *for any* x, y.

Even if the global claims process is highly inhomogeneous, these assumptions are approximately met for sufficiently small p. For our final predictors, p can be chosen appropriately to balance model bias and variance: a smaller choice for p increases estimation variance while allowing for a more flexible model and hence less bias. Implicitly assuming Assumption 1 for some given period length $p > 0$, we propose to replace the denominator $P(T + D \leq \tau | X = x, T = t, Y_1 = y_1, \ldots, Y_q = y_q)$ in (3.3), see also the alternative expression in (3.4), by

$$\frac{1}{\mathrm{Leb}(I_p(k_t) \cap C)} \int_{I_p(k_t) \cap C} \int_{\mathfrak{Y}_{q+1}} \cdots \int_{\mathfrak{Y}_Q} P_{D|x,t_{k_t},y}([0, \tau - s]) \\ \mathrm{d}P_{Y_Q|x,t_{k_t},y^{(Q-1)}}(y_Q) \ldots \mathrm{d}P_{Y_{q+1}|x,t_{k_t},y^{(q)}}(y_{q+1}) \, \mathrm{d}s, \quad (3.6)$$

where $k_t = \lfloor \frac{t}{p} \rfloor$ denotes the number of the period of length p containing t, which in turn, using the notation from Assumption 1, is $I_p(k_t) = [k_t \cdot p, (k_t + 1) \cdot p)$ with midpoint $t_{k_t} = k_t p + \frac{p}{2}$. Moreover, $C = C(x)$ is the coverage period associated with x. Note that both the denominator and the integral are non-zero for observed values $(x, t, y, d) \in \mathfrak{D}_\tau$. Overall, we aim at maximizing

$$\mathcal{L}^{(p)}(g | \mathfrak{D}_\tau) = \sum_{(x,t,y,d) \in \mathfrak{D}_\tau} \ell^{(p)}_{(x,t,y_1,\ldots,y_{q-1})}(g | y_q)$$

instead of (3.2), where

$$\ell^{(p)}_{(x,t,y_1,\ldots,y_{q-1})}(g | y_q) = \frac{\log f^{(q)}_{g(x,t,y_1,\ldots,y_{q-1})}(y_q)}{\mathrm{denom}_{q-1}(x, t_{k_t}, y_1, \ldots, y_{q-1})}$$

with $\mathrm{denom}_{q-1}(x, t_{k_t}, y_1, \ldots, y_{q-1})$ denoting the expression in (3.6).

3.3 Modelling and Fitting the Claim Intensity

Once a distribution for $P_{Y|X,T}$ has been fitted, the only unknown object in the model from Definition 2.1 is the claim intensity $\lambda = \lambda(x,t)$.

Recall that the reported claims process $\xi_r^{(i)}$ has intensity measure

$$\mu_r^{(i)}(S) = \int_{C(x^{(i)})} \int_{\mathfrak{Y}} \int_{[0, \tau - t]} \mathbf{1}_S(t, y, d) \lambda(x^{(i)}, t) \, \mathrm{d}P_{D|x^{(i)},t,y}(d) \, \mathrm{d}P_{Y|x^{(i)},t}(y) \, \mathrm{d}t,$$

where $S \subset [0, \infty) \times \mathfrak{Y} \times [0, \infty)$.

Assuming local homogeneity as in Assumption 1 for period length $p > 0$, and setting $S(k) = I_p(k) \times \mathfrak{Y} \times [0, \infty)$ with $I_p(k) = [kp, (k+1)p)$ with midpoint $t_k = kp + \frac{p}{2}$, we obtain that, for each $k \in \mathbb{N}_0$,

$$\xi_r^{(i)}(S(k)) \sim \mathrm{Poi}\Big(\lambda(x^{(i)}, t_k) \int_{\mathfrak{Y}} \int_{I_p(x^{(i)},k)} P_{D|x^{(i)},t_k,y}([0, \tau - s]) \, \mathrm{d}s \, \mathrm{d}P_{Y|x^{(i)},t_k}(y) \Big)$$

where we define

$$I_p(x,k) \coloneqq C(x) \cap I_p(k) = C(x) \cap [kp, (k+1)p) \tag{3.7}$$

with $C(x)$ the coverage period of policy x. Given a set of policies, this allows fitting a poisson model (e.g., a poisson GLM with log link) to the number of reported claims per period in the usual way by specifying a fixed offset o for each observation and estimating the common intensity factor $\lambda(x,t)$. Compared to a classical claim frequency model without truncation, where $\xi^{(i)}(S(k)) \sim \mathrm{Poi}\big(\lambda(x^{(i)},t_k)\mathrm{Leb}(I_p(x^{(i)},k))\big)$ with $\mathrm{Leb}(I_p(x^{(i)},k))$ usually called the *exposure*, the offset term $o = \log(\mathrm{Leb}(I_p(x^{(i)},k)))$ must be adjusted by the reporting probability from (3.9). See (Goldburd et al. 2016; Wüthrich and Buser 2019) for a more detailed introduction to the classical frequency modelling approach and offsets.

Model 3 (Micro-Level Model for Claim Frequency)**.** *Let $\mathcal{G}$ denote a set of MLPs $g : \mathfrak{X} \times [0,\infty) \to \mathbb{R}_+$. We assume that there exists $g \in \mathcal{G}$ such that $\lambda(x,t) = g\big(x,t_{k_t}\big)$ for all $x \in \mathfrak{X}$ and all $t > 0$, i.e., the Claim Frequency λ is given by a piecewise constant extension of $g(x,t_k)$ to the intervals $I_p(k)$ for $k = 0,1,\ldots$, which is consistent with using Assumption 1 for period length p.*

The Claim Frequency $\lambda(x,t)$ can hence be estimated by maximizing the Poisson likelihood

$$\mathcal{L}(g|\mathfrak{D}_\tau) = \sum_{i=1}^{N_{\mathrm{pol}}} \sum_{k=0}^{\infty} \xi_r^{(i)}(S(k)) \cdot \log\Big(g(x^{(i)},t_k) \cdot \mathrm{ex}(x^{(i)},k)\Big) - g(x^i,t_k) \cdot \mathrm{ex}(x^{(i)},k), \tag{3.8}$$

where

$$\mathrm{ex}(x^{(i)},k) \coloneqq \int_{\mathfrak{Y}} \int_{I_p(x^{(i)},k)} P_{D|x^{(i)},t_k,y}([0,\tau-s))\,\mathrm{d}s\,\mathrm{d}P_{Y|x^{(i)},t_k}(y). \tag{3.9}$$

If $\hat{g} \in \mathcal{G}$ maximizes $\mathcal{L}(g|\mathfrak{D}_\tau)$, we write

$$\hat{\lambda}^{\mathrm{NNet}}(x,t) = \hat{g}(x,t_{k_t}).$$

Note that maximization of $\mathcal{L}(g|\mathfrak{D}_\tau)$ is straight-forward once the exposures $\mathrm{ex}(x^{(i)},k)$ have been computed. The latter requires numerical integration over $\mathfrak{Y}$, after replacing $P_{D|x,t,y}$ and $P_{Y|x,t}$ by estimated versions thereof. Care must be taken in the choice of $\mathfrak{Y}$ during modelling, so this integral remains feasible: Choosing continuous covariates necessitates computation of possibly challenging (and maybe indefinite) integrals with respect to $P_{Y_q|x,t,y_1,\ldots,y_{q-1}}$, choosing too many discrete covariates results in combinatorial explosion of the number of summands to be computed when performing integration with respect to the counting measure.

4 Individual Claims Count Prediction based on Estimated Claim Arrival Processes

The models and estimators from the previous section can be used in various ways to define predictors für IBNR claim numbers; see Bücher and Rosenstock (2022a) for an example that only involves the reporting delay model. Throughout this section, we describe a predictor that is based on the full (estimated) claim arrival model. Alternative intermediate predictors will be defined in the simulation study.

More precisely, for each given period $I_p(k) = [k \cdot p, (k+1) \cdot p)$ of length $p > 0$ and each claim feature set $\mathfrak{Y}' \subset \mathfrak{Y}$ and each reporting interval $(\tau_0,\tau_1] \subset [0,\infty]$, we derive a predictor for the number of claims policy i has incurred within period $I_p(k)$ with claim features in $\mathfrak{Y}'$ and with a reporting time in $(\tau_0,\tau_1]$. For that purpose, let $S'(k) \coloneqq I_p(k) \times \mathfrak{Y}' \times [0,\infty)$ and $R_{\tau_0:\tau_1} \coloneqq \{(t,y,d) : \tau_0 < t + d \leq \tau_1\}$. For completeness, let $R_{\tau_0:\tau_1} = \varnothing$ if $\tau_0 \geq \tau_1$.

Note that the target number of claims can then be written as

$$N^{(i)}_{\tau_0:\tau_1}(S'(k)) \coloneqq \xi^{(i)}(S'(k) \cap R_{\tau_0:\tau_1}),$$

and that we observe, under Observation Scheme 1, the respective number of reported claims $\xi^{(i)}_r(S'(k) \cap R_{\tau_0:\tau_1}) = \xi^{(i)}\big(S'(k) \cap R_{\tau_0:\min(\tau_1,\tau)}\big)$, which is zero if $\tau_0 > \tau$.

Now, if Assumption 1 is met for the given period length $p > 0$, we obtain that, by the restriction theorem (Theorem 5.2 in Last and Penrose 2018),

$$\begin{aligned}
&\mathbb{E}[\xi^{(i)}(S'(k) \cap R_{\tau_0:\tau_1}) \mid \xi^{(i)}_r(S'(k) \cap R_{\tau_0:\tau_1})] \\
&= \xi^{(i)}_r(S'(k) \cap R_{\tau_0:\tau_1}) + \mathbb{E}[\xi^{(i)}_{nr}(S'(k) \cap R_{\tau_0:\tau_1})] \\
&= \xi^{(i)}_r(S'(k) \cap R_{\tau_0:\tau_1}) + \mathbb{E}[\xi^{(i)}(S'(k) \cap R_{\max(\tau_0,\tau):\tau_1})] \\
&= \xi^{(i)}_r(S'(k) \cap R_{\tau_0:\tau_1}) + \lambda(x^{(i)}, t_k) \int_{\mathfrak{Y}'} \int_{I_p(x^{(i)},k)} P_{D|x^{(i)},t_k,y}(I_{\tau_0:\tau_1}(\tau, s))\, \mathrm{d}s P_{Y|x^{(i)},t_k}(\mathrm{d}y),
\end{aligned}$$

where $\xi^{(i)}_{nr} = \xi^{(i)} - \xi^{(i)}_r$ denotes the unknown number of unreported claims, where $I_p(x,k)$ is defined in (3.7) and where $I_{\tau_0:\tau_1}(\tau, s) \coloneqq (\max(\tau, \tau_0) - s, \tau_1 - s]$, with the convention that the interval is the empty set if $\max(\tau, \tau_0) > \tau_1$. As is well-known, if $\lambda(x,t)$, $P_{Y|x,t}$ and $P_{D|x,t,y}$ were known, this would be the best L^2-predictor for $\xi^{(i)}(S'(k) \cap R_{\tau_0:\tau_1})$ in terms of $\xi^{(i)}_r(S'(k) \cap R_{\tau_0:\tau_1})$. Replacing the unknown objects on the right-hand side by suitable estimators as in the previous sections, we arrive at the predictor

$$\begin{aligned}
\hat{N}^{(i)}_{\tau_0:\tau_1}(S'(k)) &\coloneqq \hat{\xi}^{(i)}(S'(k) \cap R_{\tau_0:\tau_1}) \\
&\coloneqq \xi^{(i)}_r(S'(k) \cap R_{\tau_0:\tau_1}) \\
&\quad + \hat{\lambda}(x^{(i)}, t_k) \int_{\mathfrak{Y}'} \int_{I_p(x^{(i)},k)} \hat{P}_{D|x^{(i)},t_k,y}(I_{\tau_0:\tau_1}(\tau, s))\, \mathrm{d}s\, \mathrm{d}\hat{P}_{Y|x^{(i)},t_k}(y).
\end{aligned} \tag{4.1}$$

In contrast to classical factor-based reserving methods, this predictor may yield a non-zero expected number of claims even for policies without already reported claims. This allows for the individual-level count predictions to have a meaningful interpretation as the expected number of unreported claims for that particular policy.

Remark 4.1. *The predictor in (4.1) can be adapted to a general $S = I \times \mathfrak{Y}' \times [0, \infty)$ by summing over the intervals covering I as follows:*

$$\begin{aligned}
\hat{N}^{(i)}_{\tau_0:\tau_1}(S) &= \xi_r(S \cap R_{\tau_0:\tau_1}) \\
&\quad + \sum_{k=0}^{\tau/p-1} \hat{\lambda}(x^{(i)}, t_k) \int_{\mathfrak{Y}'} \int_{I_p(x^{(i)},k) \cap I} \hat{P}_{D|x^{(i)},t_k,y}(I_{\tau_0:\tau_1}(\tau, s))\, \mathrm{d}s\, \mathrm{d}\hat{P}_{Y|x^{(i)},t_k}(y).
\end{aligned}$$

5 Individual Claims Count Prediction based on Chain Ladder Networks for the Claim Frequency

We propose an alternative estimator for the claim frequency λ, which is similar to the estimator from Section 3.3. However, instead of being based on preliminary estimators of the claim feature and reporting delay distributions, the new estimator is based on classical chain ladder factors. In a second step, the estimator is used to define a new predictor for IBNR claims, similar to Section 4.

Given a partition $\mathfrak{Y} = \mathfrak{Y}_1 \cup \mathfrak{Y}_2 \cup \cdots \cup \mathfrak{Y}_M$ into groups of claim features and a development period length p where $\tau = P \cdot p$, we make the following classical chain ladder assumption: for any group index $m \in \{1, \ldots, M\}$ and any development period $j \in \{1, \ldots, P-1\}$, there exists a factor $f_j^{\mathrm{CL},\mathfrak{Y}_m}$ called *chain ladder factor* such that, for any policy i and any accident period $k \in \{0, \ldots, P-1\}$,

$$\mathbb{E}[\xi^{(i)}(S_m(k) \cap R_{0:(k+j+1)\cdot p})] = f_j^{\mathrm{CL},\mathfrak{Y}_m} \mathbb{E}[\xi^{(i)}(S_m(k) \cap R_{0:(k+j)\cdot p})],$$

where $S_m(k) \coloneqq I_p(k) \times \mathfrak{Y}_m \times [0,\infty)$ and $R_{\tau_0:\tau_1} \coloneqq \{(t,y,d) : \tau_0 < t+d \le \tau_1\}$. Iterating the equation, we obtain that

$$\mathbb{E}[\xi^{(i)}(S_m(k) \cap R_{0:(k+P)\cdot p})] = \mathrm{FtU}_k^{\mathrm{CL},\mathfrak{Y}_m} \cdot \mathbb{E}[\xi^{(i)}(S_m(k) \cap R_{0:P\cdot p})], \tag{5.1}$$

where

$$\mathrm{FtU}_k^{\mathrm{CL},\mathfrak{Y}_m} \coloneqq \prod_{j=P-k}^{P-1} f_j^{\mathrm{CL},\mathfrak{Y}_m}$$

is the *chain ladder factor-to-ultimate*. Under the additional assumption that every claim is developed within at most P periods, we have that $\xi^{(i)}(S_m(k) \cap R_{0:(k+P)\cdot p}) = \xi^{(i)}(S_m(k))$. Hence, if Assumption 1 is met for $p > 0$ specified above, the left-hand side of (5.1) can be written as

$$\begin{aligned}\mathbb{E}[\xi^{(i)}(S_m(k) \cap R_{0:(k+P)\cdot p})] &= \mathbb{E}[\xi^{(i)}(S_m(k))] \\ &= P_{Y|x^{(i)},t_k}(\mathfrak{Y}_m) \cdot \mathrm{Leb}(I_p(x^{(i)},k)) \cdot \lambda(x^{(i)},t_k).\end{aligned}$$

On the other hand, for the expression on the right-hand side of (5.1), we observe that $\xi^{(i)}(S_m(k) \cap R_{0:P\cdot p}) = \xi_r^{(i)}(S_m(k))$ is the reported number of claims with accident year k and claims from $\mathfrak{Y}_m$. Hence, combining the previous equations with (5.1), we obtain that

$$\mathbb{E}[\xi_r^{(i)}(S_m(k))] = \frac{1}{\mathrm{FtU}_k^{\mathrm{CL},\mathfrak{Y}_m}} P_{Y|x^{(i)},t_k}(\mathfrak{Y}_m) \cdot \mathrm{Leb}(I_p(x^{(i)},k)) \cdot \lambda(x^{(i)},t_k).$$

In view of the basic Poisson assumption on $\xi^{(i)}$ (and hence on $\xi_r^{(i)}$ and $\xi_{nr}^{(i)}$) from Definition 2.1, we obtain that

$$\begin{aligned}\xi_r^{(i)}(S_m(k)) &\sim \mathrm{Poi}\left(P_{Y|x^{(i)},t_k}(\mathfrak{Y}_m) \cdot \mathrm{Leb}(I_p(x^{(i)},k)) \cdot \lambda(x^{(i)},t_k) \cdot \frac{1}{\mathrm{FtU}_k^{\mathrm{CL},\mathfrak{Y}_m}}\right), \\ \xi_{nr}^{(i)}(S_m(k)) &\sim \mathrm{Poi}\left(P_{Y|x^{(i)},t_k}(\mathfrak{Y}_m) \cdot \mathrm{Leb}(I_p(x^{(i)},k)) \cdot \lambda(x^{(i)},t_k) \cdot \left(1 - \frac{1}{\mathrm{FtU}_k^{\mathrm{CL},\mathfrak{Y}_m}}\right)\right).\end{aligned}$$

This can be used to estimate the unknown claim frequencies on $\mathfrak{Y}_m$, i.e.,

$$\lambda^{\mathfrak{Y}_m}(x,t) \coloneqq P_{Y|x,t}(\mathfrak{Y}_m)\lambda(x,t).$$

Indeed, let $\mathcal{G}$ denote a set of MLPs $g : \mathfrak{X} \times [0,\infty) \to [0,\infty)$ as in Model 3. We assume that the claim frequency on $\mathfrak{Y}_m$ satisfies, for some $g^{\mathfrak{Y}_m} \in \mathcal{G}$,

$$\lambda^{\mathfrak{Y}_m}(x,t) = g^{\mathfrak{Y}_m}(x,t_{k_t}) \qquad \forall\, x,t.$$

Recalling the notation $t_k = kp + \frac{1}{2}$, we arrive at the per-triangle loss

$$\begin{aligned}\mathcal{L}^{\mathrm{CL},\mathfrak{Y}_m}(g^{\mathfrak{Y}_m}|\mathfrak{D}_\tau) = \sum_{i=1}^{N_{\mathrm{pol}}} \sum_{k=0}^{P-1} \xi_r^{(i)}(S_m(k)) \cdot \log\Big(g^{\mathfrak{Y}_m}(x^{(i)},t_k) \cdot \mathrm{ex}^{\mathrm{CL},\mathfrak{Y}_m}(x^{(i)},k)\Big) \\ - g^{\mathfrak{Y}_m}(x^{(i)},t_k) \cdot \mathrm{ex}^{\mathrm{CL},\mathfrak{Y}_m}(x^{(i)},k),\end{aligned}$$

where

$$\mathrm{ex}^{\mathrm{CL},\mathfrak{Y}_m}(x^{(i)},k) \coloneqq \mathrm{Leb}(I_p(x^{(i)},k)) \cdot \frac{1}{\mathrm{FtU}_k^{\mathrm{CL},\mathfrak{Y}_m}}.$$

In practice, the chain ladder factors within the loss must be estimated, for which we apply the well-known estimators

$$\begin{aligned}\hat{f}_j^{\mathrm{CL},\mathfrak{Y}_m} &\coloneqq \frac{\#\{(x,t,y,d) \in \mathfrak{D}_\tau \mid y \in \mathfrak{Y}_m, \lfloor t/p \rfloor \le (P-j-1), \lfloor (t+d)/p \rfloor - \lfloor t/p \rfloor \le j\}}{\#\{(x,t,y,d) \in \mathfrak{D}_\tau \mid y \in \mathfrak{Y}_m, \lfloor t/p \rfloor \le (P-j-1), \lfloor (t+d)/p \rfloor - \lfloor t/p \rfloor \le j-1\}}, \\ \widehat{\mathrm{FtU}}_k^{\mathrm{CL},\mathfrak{Y}_m} &\coloneqq \prod_{j=P-k}^{P-1} f_j^{\mathrm{CL},\mathfrak{Y}_m}.\end{aligned}$$

Note that in contrast to the micro-level approach from the previous sections, there is no explicit model for the distribution of claim features on $\mathfrak{Y}$. If $g^{\mathfrak{Y}_m} = \hat{\lambda}^{\mathrm{CL},\mathfrak{Y}_m}(x,t)$ are maxima of the per-triangle losses $\mathcal{L}^{\mathrm{CL},\mathfrak{Y}_m}$, the triangle-level intensity estimates can be aggregated to a common intensity estimator

$$\hat{\lambda}^{\mathrm{CL}}(x,t) := \sum_{m=1}^{M} \hat{\lambda}^{\mathrm{CL},\mathfrak{Y}_m}(x,t).$$

Finally, exploiting $\mathbb{E}[\xi^{(i)}(S) \mid \xi_r^{(i)}(S)] = \xi_r^{(i)}(S) + \mathbb{E}[\xi_{nr}^{(i)}(S)]$ similar as in Section 4, we may define an IBNR-predictor as follows: recalling $S_m(k) = I_p(k) \times \mathfrak{Y}_m \times [0,\infty)$, let

$$\hat{\xi}^{(i),\mathrm{CL}}(S_m(k)) := \xi_r^{(i)}(S_m(k)) + \hat{\lambda}^{\mathrm{CL},\mathfrak{Y}_m}(x^{(i)},t_k)\mathrm{Leb}(I_p(x^{(i)},k)) \cdot \Big(1 - \frac{1}{\widehat{\mathrm{FtU}}_k^{\mathrm{CL},\mathfrak{Y}_m}}\Big),$$

$$\hat{\xi}^{(i),\mathrm{CL}}(S(k)) := \sum_{m=1}^{M} \hat{\xi}^{(i),\mathrm{CL}}(S_m(k)).$$

Recalling the notation $R_{\tau_0:\tau_1} := \{(t,y,d) : \tau_0 < t+d \le \tau_1\}$, similar derivations show that this predictor can also be extended to a predictor for reporting times $\tau_0 = P_0 \cdot p$ to $\tau_1 = P_1 \cdot p$ with $P_0 < P_1 \in \mathbb{N}_0 \cup \{+\infty\}$:

$$\begin{aligned}\hat{N}_{\tau_0:\tau_1}^{(i),\mathrm{CL}}(S_m(k)) &:= \xi_r^{(i)}(S_m(k) \cap R_{\tau_0:\tau_1}) \\ &\quad + \hat{\lambda}^{\mathrm{CL},\mathfrak{Y}_m}(x^{(i)},t_k)\mathrm{Leb}(I_p(x^{(i)},k)) \cdot \Big(\frac{1}{\widehat{\mathrm{FtU}}_{k+P-P_1}^{\mathrm{CL},\mathfrak{Y}_m}} - \frac{1}{\widehat{\mathrm{FtU}}_{k+P-P_0}^{\mathrm{CL},\mathfrak{Y}_m}}\Big) \quad (5.2)\end{aligned}$$

Here, we define the empty product as 1, i.e., $\widehat{\mathrm{FtU}}_j^{\mathrm{CL},\mathfrak{Y}_m} := 1$ for all $j \le 0$. Finally, it is worthwhile to mention that in contrast to the predictor described in Section 4, it is not possible to use the Chain Ladder Networks to obtain predictions for arbitrary τ_0, τ_1 that are not whole multiples of p.

6 Evaluating Individual Claim Count Predictors

The quality of competing predictors may be assessed by suitable error measures. In this section, we define two such measures: an individual mean squared prediction error, and an aggregated global mean squared prediction error.

We start by considering the individual error measure. For $S = [0,\tau) \times \mathfrak{Y}' \times [0,\infty) \subset [0,\infty) \times \mathfrak{Y} \times [0,\infty)$ and $0 \le \tau_1 < \tau_1 \le \infty$, let $\hat{N}_{\tau_0:\tau_1}^{(i)}(S)$ denote individual claim count predictions for $N_{\tau_0:\tau_1}^{(i)}(S) = \xi^{(i)}(S \cap R_{\tau_0:\tau_1})$, the number of claims in S incurred by policy $x^{(i)}$ that are reported between τ_0 and τ_1; recall $R_{\tau_0:\tau_1} = \{(t,y,d) : \tau_0 < t+d \le \tau_1\}$. Let $q > 0$ denote an evaluation period length (for instance, $q = 365$ corresponding to a year; note that there should be no confusion with the running index q used in Section 3.2), which is assumed to be a divisor of the total observation length τ from Observation Scheme 1, and let $\mathfrak{Y}' \subset \mathfrak{Y}$ denote an evaluation set of claim features. We then define

$$\mathrm{RMSE}_{\tau_0:\tau_1}^{\mathrm{expo}}(\mathfrak{Y}',q) := \left(\frac{1}{\sum_{j=0}^{\tau/q-1} \#\mathcal{P}(q,j)} \sum_{\ell=0}^{\tau/q-1} \sum_{i \in \mathcal{P}(q,\ell)} \{\hat{N}_{\tau_0:\tau_1}^{(i)}(S'_q(\ell)) - N_{\tau_0:\tau_1}^{(i)}(S'_q(\ell))\}^2\right)^{\frac{1}{2}}, \quad (6.1)$$

where, for $\ell \in \mathbb{N}_0$, recalling the notation $I_q(x,\ell) = C(x) \cap [\ell \cdot q, (\ell+1) \cdot q)$,

$$S'_q(\ell) := [\ell \cdot q, (\ell+1) \cdot q) \times \mathfrak{Y}' \times [0,\infty), \quad \mathcal{P}(q,\ell) := \{i \in \{1,\dots,N_{\mathrm{pol}}\} : I_q(x^{(i)},\ell) \neq \emptyset\}.$$

Note that in practice the measure can only be calculated for $\tau_1 \leq \tau$ (with τ the most recent date for which data is available) and on selected tests sets (for instance, in a back-testing approach). In controlled simulation experiments, see Section 7, we may and will use $\tau_1 = \infty$, thereby aiming at predicting the total number of unreported claims for each policy. Moreover, for using the error measures with the predictors from Section 4 and 5, the evaluation period q must be a multiple of the homogeneity period length p (unless one is willing to use the extension discussed in Remark 4.1).

The quality of individual claim count predictors may alternatively be assessed by first aggregating the individual predictions and then using standard global error measures; the predictors may then even be compared with classical methods for aggregated data like the standard Chain Ladder approach. Aggregated predictions are obtained from individual predictions straightforwardly: for $A = \mathfrak{X}' \times S$ with $\mathfrak{X}' \subset \mathfrak{X}$ and S as above, let

$$\hat{N}_{\tau_0:\tau_1}(A) := \sum_{\substack{i \in \{1,\dots,N_{\text{pol}}\}: \\ x^{(i)} \in \mathfrak{X}'}} \hat{N}^{(i)}_{\tau_0:\tau_1}(S),$$

which is to be considered a predictor for the aggregated claim number

$$N_{\tau_0:\tau_1}(A) := \sum_{\substack{i \in \{1,\dots,N_{\text{pol}}\}: \\ x^{(i)} \in \mathfrak{X}'}} \xi^{(i)}(S \cap R_{\tau_0:\tau_1}).$$

For q and $\mathfrak{Y}'$ as in (6.1), we then define

$$\text{RMSE}_{\tau_0:\tau_1}(\mathfrak{Y}', q) := \left(\frac{q}{\tau} \sum_{\ell=0}^{\tau/q-1} \left\{ \hat{N}_{\tau_0:\tau_1}(A_{q,\ell,\mathfrak{Y}'}) - N_{\tau_0:\tau_1}(A_{q,\ell,\mathfrak{Y}'}) \right\}^2 \right)^{\frac{1}{2}}, \tag{6.2}$$

where $A_{q,\ell,\mathfrak{Y}'} := \mathfrak{X} \times [\ell \cdot q, (\ell+1) \cdot q) \times \mathfrak{Y}' \times [0, \infty)$. Note that this measure has also been used in Bücher and Rosenstock (2022a), Formula (20). Its application is limited to the constraints mentioned above for τ_1 and q.

7 Simulation study

In this section, we will study the performance of the new estimators and predictors within nine different simulation scenarios taken from Bücher and Rosenstock (2022a). We start by restating a brief, partially verbatim summary of the simulation models taken from the last-named paper:

The underlying portfolios build upon the car insurance data set described in Appendix A in Wüthrich and Buser (2019). The latter data set provides claim counts for 500,000 insurance policies, where each policy is associated with the risk features

$$(\texttt{age}, \texttt{ac}, \texttt{power}, \texttt{gas}, \texttt{brand}, \texttt{area}, \texttt{dens}, \texttt{ct}),$$

which correspond to age of driver, age of car, power of car, fuel type of car, brand of car, and area code, respectively; see also (A.1) in Wüthrich and Buser (2019) for further details. Next to that, the data set also provides the variable `truefreq`, which corresponds to the claim intensity $\lambda(x)$ in our model.

Each portfolio is considered over ten periods of 365 days, that is, the portfolio coverage period is the interval $[0, 3650]$. The different scenarios are as follows:

The baseline scenario. The baseline scenario/portfolio is characterized by a homogeneous *exposure* as well as position-independent *claim intensity*, *occurrence process*, and *reporting process*. It may be considered the vanilla portfolio that practitioners often aim at by careful selection of considered risks and suitable transformations, e.g., adjustment for inflation. More precisely:

• *Exposure.* New risks arrive according to a homogeneous Poisson process with intensity $50,000/365 \approx 137$ and contracts run for exactly one year. Moreover, the portfolio starts with exactly $50,000$ policies with $t_{\text{start}} = 0$ and with remaining contract duration that is uniform on $[0, 365]$. As a consequence, the total exposure is constant in expectation and we have $N_{\text{pol}} \sim 50,000 + \text{Poi}(500,000)$. Finally, for each risk in the portfolio we randomly draw (with replacement) risk features from the aforementioned data set from Wüthrich and Buser (2019).

• *Claim Intensity.* The claim frequency $\lambda(t, x) = \lambda(x)$ is independent of t and t_{start} and given by the variable `truefreq` that belongs to the risk selected in the previous paragraph.

• *Occurrence Process.* The occurrence process is position-independent, i.e., $P_{Y|X=x,T=t} = P_{Y|X=x}$ for all t. We choose to work with two claim variables, $y = (\texttt{cc}, \texttt{severity})$, with claims code $\texttt{cc} \in \{\text{injury}, \text{material}\}$, and claim size $\texttt{severity} \in \mathbb{R}_+$. The claim feature distribution of `cc` is chosen to be a function of the policy features `ac`, `power`, and `dens` in such a way that material damages are more likely to occur in densely populated areas and with low-powered and newer cars (see Appendix D in Bücher and Rosenstock 2022b for details on the precise relationship). The claim severity distribution of `severity` is log-normal with σ constant and with μ depending on `cc`, `brand`, `ac` and `power` in such a way that injury claims, especially with older high-powered cars, are more severe. Moreover, material damages for certain premium brands are also more severe. Again, details are provided in Appendix D in Bücher and Rosenstock (2022b).

• *Reporting Process.* The reporting process is position-independent, i.e, $P_{D|X=x,T=t,Y=y} = P_{D|X=x,Y=y}$. We choose to work with $P_{D|X=x,Y=y} \in \text{BDEGP}(n = 1, m = 3, \kappa = 3 \cdot 365, \varepsilon = 365/2)$ as a basic family, with fixed erlang shapes $\alpha = (1, 3, 6)$ that do not depend on x and y. The remaining 7 parameters (i.e., the four mixture weights of $\delta_0, \Gamma(1, \theta), \Gamma(3, \theta), \Gamma(6, \theta)$, and $\text{GPD}(\kappa, \sigma, \xi)$, as well as θ, σ, and ξ) are chosen to depend on `age`, `dens`, `ac` (only if `cc` is material), `cc`, and `severity` in such a way that more severe claims, material claims with new cars, and claims with younger drivers in populated areas will be reported sooner, while low-severity injuries will be reported later; see Appendix D in Bücher and Rosenstock (2022b) for details.

Eight non-homogeneous scenarios. Eight non-homogeneous scenarios are obtained by altering a single element of the baseline scenario:

1. *Exposure*: The distribution of `ac` changes continuously (drift) or abruptly (shock).
2. *Intensity*: $\lambda(x, t)$ decreases continuously (drift) or abruptly (shock).
3. *Occurrence*: The distribution of `cc` changes continuously (drift) or abruptly (shock).
4. *Reporting delay*: The distribution of D is altered by moving probability mass to shorter reporting delays, continuously (drift) or abruptly (shock).

Figure 1 illustrates the effect of the different scenarios on exposure, claim counts and reporting delays. The precise functional relationships are documented in Appendix D in Bücher and Rosenstock (2022b).

The simulation study was conducted with 50 data seeds for each of the 9 scenarios, i.e., with 450 simulated portfolio datasets in total.

7.1 Training Procedure

In this section, we describe details on the training and model selection procedure for the various neural networks used in the predictors. All networks were trained for $5,000$ epochs using the adam optimizer with fixed parameters $\alpha = 0.05, \beta_0 = \beta_1 = 0$ and an adaptive learning rate, halving the learning rate on plateaus (patience $= 2$) down to a minimum of $\underline{\alpha} = 10^{-4}$. In addition to this, the available (truncated) data was randomly split into 75% training data and 25% validation data. The validation data was not used for model calibration and instead kept aside to assess the generalization error.

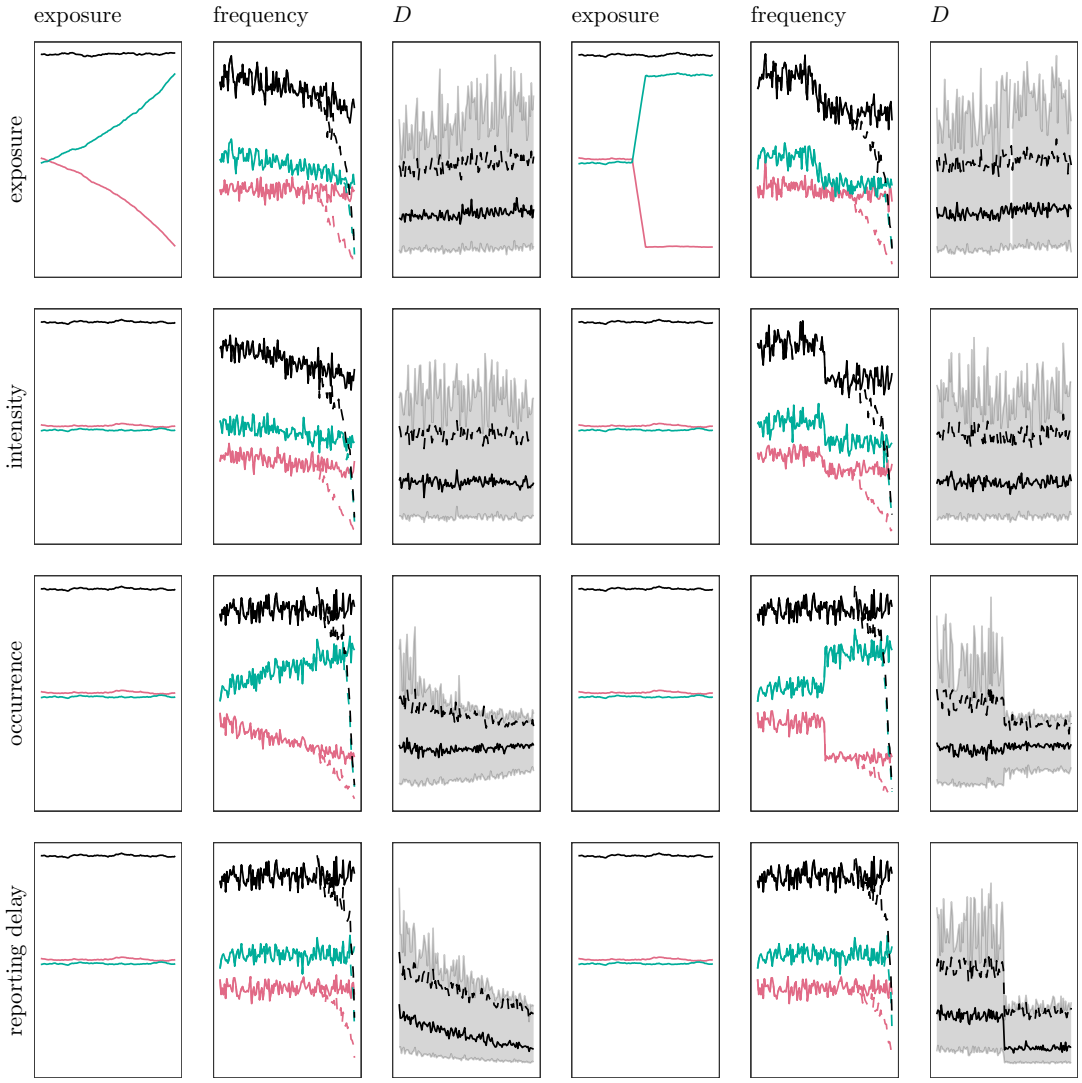


Figure 1: Overview of all scenarios, taken from Bücher and Rosenstock (2022a, Figure 4). Rows show different scenarios, the left three columns showing the drift variation and the right three columns showing the shock variation. Within the scenarios, the panels show, from left to right, the exposure at risk (aggregated and split by `ac` ≤ 5 shown in red and `ac` > 5 shown in blue), the number of claims (aggregated and split by `cc` with injury shown in red and material shown in blue; dashed line: reported, solid line: occurred), the reporting delay distribution (dashed line: mean, solid line: median, ribbon: first and third quartiles).

Estimating the Claim Arrival Process. Fitting of the claim arrival process was done in four steps, each requiring a slightly different neural network architecture. First, $\hat{P}_{D|x,t,y}$ was estimated as described in Bücher and Rosenstock (2022a), compare Section 3.1. Reporting delay networks were trained for 100 starting seeds (for parameter initialization) using the correct BDEGP specification (with unknown parameters). The top 10 performing reporting delay networks were chosen by computing $\text{RMSE}_{\tau-365:\tau}(\mathfrak{Y}, 365; \mathfrak{D}_{\tau-365})$ for the predictor based on $\hat{P}_{D|x,t,y_1,y_2}$ (denoted NNet in Section 7.2), i.e., by the back-testing error for one year in the past.

Next, as described in Section 3.2, for each of the 10 networks from the previous step, coordinate distributions for $\hat{P}_{Y|x,t}$ were estimated from the decomposition $\mathfrak{Y} = \mathfrak{Y}_1 \times \mathfrak{Y}_2$ with $\mathfrak{Y}_1 = \{\text{injury}, \text{material}\}$ describing the claims code and $\mathfrak{Y}_2 = \mathbb{R}_+$ describing the severity. First, $\hat{P}_{Y_2|x,y,y_1}$ was estimated using an MLP for the two parameters of a log-normal distribution, i.e., $\hat{P}_{Y_2|x,t,y_1} = \log\mathcal{N}(g^{(2)}(x,t,y_1))$. The network architecture $\mathcal{G}^{(2)}$ for $g^{(2)}(x,t,y_1)$ consisted of a $(10,5)$ MLP with a softplus activation function adapted to an output in $\mathbb{R} \times (0,\infty)$, matching the two parameters (μ,σ) defining a log-normal distribution. For each of the 10 reporting delay networks, 10 starting seeds were used for training the severity feature network $g^{(2)}(x,t,y_1)$, resulting in a total of 100 estimates for the pair $(\hat{P}_{D|x,t,y_1,x_2}, \hat{P}_{Y_2|x,t,y_1})$. Similar as in the previous step, the ten best estimates were chosen based on back-testing the error one year in the past, using the predictor based on $\hat{P}_{D|x,t,y_1,x_2}$ and $\hat{P}_{Y_2|x,t,y_1}$ (denoted $\text{NNet}_{\texttt{severity}}$ in Section 7.2). It should be noted that this predictor performed worse than the underlying NNet predictor based solely on $\hat{P}_{D|x,t,y_1,y_2}$. Nonetheless, using $\hat{P}_{Y_2|x,t,y_1}$ is necessary for the subsequent steps.

For each of the ten estimates for $(\hat{P}_{D|x,t,y_1,x_2}, \hat{P}_{Y_2|x,t,y_1})$ from the previous step, we next estimated $\hat{P}_{Y_1|x,t}$. The associated network architecture $\mathcal{G}^{(1)}$ consisted of a $(10,5)$ MLP with a softplus activation function outputting probability masses for a discrete distribution on $\{\text{injury}, \text{material}\}$. It was trained for 10 different starting seeds, resulting in a total of 100 estimates for $(\hat{P}_{D|x,t,y_1,y_2}, \hat{P}_{Y_2|x,t,y_1}, \hat{P}_{Y_1|x,t})$ and hence for $(\hat{P}_{D|x,t,y}, \hat{P}_{Y|x,t})$ using the definition $\hat{P}_{Y|x,t}(\mathrm{d}y_1, \mathrm{d}y_2) = \hat{P}_{Y_1|x,t}(\mathrm{d}y_1)\hat{P}_{Y_2|x,t,y_1}(\mathrm{d}y_2)$. Again, the 10 best estimates were chosen by evaluating the associated predictor (denotes $\text{NNet}_{\texttt{cc}}$ in Section 7.2) using the backtesting error $\text{RMSE}_{\tau-365:\tau}(\mathfrak{Y}, 365; \mathfrak{D}_{\tau-365})$.

Finally, for each of the ten estimates for $(\hat{P}_{D|x,t,y}, \hat{P}_{Y|x,t})$, ten frequency estimates $\hat{\lambda}^{\text{NNet}}(x,t)$ were obtained as described in Section 3.3, with ten different starting seeds. The underlying network architecture consisted of a $(10,5)$ MLP with a softplus activation function and a parameter-free skip connection for the offset term as described in Wüthrich (2019), leading to a single poisson parameter in $\mathbb{R}_+$. After training, the bias regularization method described in Wüthrich (2019) was applied. From the resulting 100 estimates for $(\hat{P}_{D|x,t,y}, \hat{P}_{Y|x,t}, \hat{\lambda}^{\text{NNet}}(x,t))$, the final estimate was chosen according to the backtesting error for its associated predictor, denoted $\text{NNet}_{\texttt{FreqNet}}$ in Section 7.2.

Overall, the number of trained networks for each data set is 400, resulting in a total of $450 \times 400 = 180,000$ trained networks for the simulation study.

Fitting Chain Ladder Networks. Training a Chain Ladder Network requires fitting M neural networks, $g^{\mathfrak{Y}_m}$ for $m = 1, \ldots, M$. As described in next section, we use both $M = 1$ (predictor CLFreqNet in Section 7.2) and $M = 2$ ($\text{CLFreqNet}_{\texttt{cc}}$ in Section 7.2), resulting in three networks to be trained for each data set. The MLP architecture was fixed as $(10,5)$ with a softplus activation function and a parameter-free skip connection for the offset term. After training, the bias regularization method described in Wüthrich (2019) was applied using the training dataset. For each data set, ten starting seeds were used, resulting in 30 networks for each data set, from which a best predictor was chosen based on the backtesting error $\text{RMSE}_{\tau-365:\tau}(\mathfrak{Y}, 365; \mathfrak{D}_{\tau-365})$. For the entire simulation study, $450 \times 30 = 13,500$ networks were trained.

7.2 Predictors

We provide a detailed overview of the predictors, tailored to the specific portfolios described at the beginning of Section 7. For the macro-level error measure from (6.2), we will compare a total of eight different predictors, three of which provide reasonable micro-level predictions as measured by (6.1). All predictors target the number of claims in $A = \mathfrak{X}' \times I_p(k) \times \mathfrak{Y}' \times [0, \infty)$ with some $\mathfrak{X}' \subset \mathfrak{X}$ and $\mathfrak{Y}' \subset \mathfrak{Y}$.

For index m encoding one of the methods specified below, let

$$\hat{N}_{\mathrm{m}}^{\mathrm{cw}}(A; \mathfrak{D}_\tau) := \sum_{(x,t,y,d) \in A \cap \mathfrak{D}_\tau} \hat{c}_{\mathrm{m}}(x,t,y,d), \qquad \hat{N}_{\mathrm{m}}^{\mathrm{pw}}(A; \mathfrak{D}_\tau) := \sum_{\substack{i \in \{1,\dots,N_{\mathrm{pol}}\}: \\ x^{(i)} \in \mathfrak{X}'}} \hat{c}_{\mathrm{m}}(i, A),$$

where the upper index cw and pw stand for *claim-wise* and *policy-wise*, respectively, and where $\hat{c}_{\mathrm{m}}(x,t,y,d)$ and $\hat{c}_{\mathrm{m}}(i,A)$ are suitable numbers, additionally depending on $\mathfrak{D}_\tau$, τ_0 and τ_1, as specified below. For $k \in \mathbb{N}_0, t \geq 0$ and $x \in \mathfrak{X}$, recall the notations $t_k = kp + \frac{p}{2}, k_t = \lfloor \frac{t}{p} \rfloor$ and $I_p(x,k) = C(x) \cap [kp, (k+1)p)$ with $C(x)$ the coverage period of policy x, see (3.7).

- **Predictor NNet.** We consider the original method from Bücher and Rosenstock (2022a) that only relies on modeling and estimating reporting delays, see formula (19) in that paper. More precisely, we define $\hat{N}_{\mathrm{NNet}} := \hat{N}_{\mathrm{NNet}}^{\mathrm{cw}}$ with constants

$$\hat{c}_{\mathrm{NNet}}(x,t,y,d) := \frac{\int_{I_p(x,k_t)} \hat{P}_{D|X=x,T=t_{k_t},Y=y}((\tau_0 - s, \tau_1 - s])\,\mathrm{d}s}{\int_{I_p(x,k_t)} \hat{P}_{D|X=x,T=t_{k_t},Y=y}([0, \tau - s])\,\mathrm{d}s}.$$

- **Predictor $\mathrm{NNet}_{\mathtt{severity}}$.** We additionally incorporate the estimated first stage claim feature model from Section 3.2, based upon the decomposition $\mathfrak{Y} = \{\mathrm{injury}, \mathrm{material}\} \times \mathbb{R}_+ =: \mathfrak{Y}_1 \times \mathfrak{Y}_2$ into claims code and claim severity. More precisely, we define $\hat{N}_{\mathrm{NNet}_{\mathtt{severity}}} := \hat{N}_{\mathrm{NNet}_{\mathtt{severity}}}^{\mathrm{cw}}$ with

$$\hat{c}_{\mathrm{NNet}_{\mathtt{severity}}}(x,t,y,d) := \frac{\hat{p}_{\mathrm{rep}}(\tau_0, \tau_1, x, k_t, y_1)}{\hat{p}_{\mathrm{rep}}(0, \tau, x, k_t, y_1)}.$$

  with $\hat{p}_{\mathrm{rep}} = \hat{p}_{\mathrm{rep}}(\tau_0, \tau_1, x, k_t, y_1)$ defined as

$$\hat{p}_{\mathrm{rep}} = \int_{\mathfrak{Y}'_{y_1}} \int_{I_p(x,k_t)} \hat{P}_{D|X=x,T=t_{k_t},Y_1=y_1,Y_2=w}((\tau_0 - s, \tau_1 - s])\,\mathrm{d}s\,\mathrm{d}\hat{P}_{Y_2|X=x,T=t_{k_t},Y_1=y_1}(w),$$

  where $\mathfrak{Y}'_{y_1} = \{w : (y_1, w) \in \mathfrak{Y}'\}$.
- **Predictor $\mathrm{NNet}_{\mathtt{cc}}$.** This predictor is built on the full estimated claim feature model, see Section 3.2. More precisely, we define $\hat{N}_{\mathrm{NNet}_{\mathtt{cc}}} := \hat{N}_{\mathrm{NNet}_{\mathtt{cc}}}^{\mathrm{cw}}$

$$\hat{c}_{\mathrm{NNet}_{\mathtt{cc}}}(x,t,y,d) := \frac{\hat{p}_{\mathrm{rep}}(\tau_0, \tau_1, x, k_t)}{\hat{p}_{\mathrm{rep}}(0, \tau, x, k_t)}$$

  with $\hat{p}_{\mathrm{rep}} = \hat{p}_{\mathrm{prep}}(\tau_0, \tau_1, x, k_t)$ defined as

$$\hat{p}_{\mathrm{rep}} := \int_{\mathfrak{Y}'} \int_{I_p(x,k_t)} \hat{P}_{D|X=x,T=t_{k_t},Y=y}((\tau_0 - s, \tau_1 - s])\,\mathrm{d}s\,\mathrm{d}\hat{P}_{Y|X=x,T=t_{k_t}}(y).$$

  For $\mathfrak{Y}' = \mathfrak{Y}$, this can be written as

$$\hat{p}_{\mathrm{rep}} = \sum_{u \in \{\mathrm{injury}, \mathrm{material}\}} \int_0^\infty \int_{I_p(x,k_t)} \hat{P}_{D|X=x,T=t_{k_t},Y_1=u,Y_2=w}((\tau_0 - s, \tau_1 - s])\,\mathrm{d}s \\ \mathrm{d}\hat{P}_{Y_2|X=x,T=t_{k_t},Y_1=u}(w) \cdot \hat{P}_{Y_1|X=x,T=t_{k_t}}(\{u\}).$$

- **Predictor FreqNet.** This predictor is the one from Section 4 that builds upon the full estimated model for the claim arrival process. More precisely, $\hat{N}_{\text{FreqNet}} = \hat{N}^{\text{pw}}_{\text{FreqNet}}$ with

$$\hat{c}_{\text{FreqNet}}(i, A) \coloneqq \xi_r^{(i)}(I_p(k) \times \mathfrak{Y}' \times [0,\infty)) + \hat{\lambda}^{\text{NNet}}(x^{(i)}, t_k) \cdot \int_{\mathfrak{Y}'} \int_{I_p(x^{(i)},k)} \hat{P}_{D|X=x^{(i)},T=t_k,Y=y}((\max(\tau,\tau_0)-s, \tau_1 - s])\,\mathrm{d}s\,\mathrm{d}\hat{P}_{Y|X=x^{(i)},T=t_k}(y),$$

  which corresponds to (4.1).
- **Predictor CL.** This predictor is the basic chain ladder predictor. More precisely, $\hat{N}_{\text{CL}} = \hat{N}^{\text{cw}}_{\text{CL}}$ with

$$\hat{c}_{\text{CL}}(x,t,y,d) = \mathbf{1}(P_0 \le \lfloor (T+D)/p \rfloor \le P_1) + \text{FtU}^{\mathfrak{Y}}_{k_t+P-P_0} - \text{FtU}^{\mathfrak{Y}}_{k_t+P-P_1},$$

  where $\tau = P \cdot p, \tau_0 = P_0 \cdot p$ and $\tau_1 = P_1 \cdot p$ must be whole multiples of the development period and $\text{FtU}^{\mathfrak{Y}}_k \coloneqq 1$ for $k < 0$.
- **Predictor CLFreqNet.** This is the basic Chain Ladder Network based predictor, and is only defined for $\mathfrak{Y}' = \mathfrak{Y}$. More precisely, $\hat{N}_{\text{CLFreqNet}} = \hat{N}^{\text{pw}}_{\text{CLFreqNet}}$ with

$$\hat{c}_{\text{CLFreqNet}}(i, A) \coloneqq \xi_r^{(i)}(I_p(k) \times \mathfrak{Y} \times [0,\infty) \cap R_{\tau_0:\tau_1}) + \text{Leb}(I_p(x^{(i)},k)) \cdot \hat{\lambda}^{\text{CL},\mathfrak{Y}}(x^{(i)}, t_k) \cdot \Big(\frac{1}{\text{FtU}^{\mathfrak{Y}}_{k+P-P_1}} - \frac{1}{\text{FtU}^{\text{CL},\mathfrak{Y}}_{k_t+P-P_0}} \Big).$$

  This formula comes from (5.2) with the trivial partition using $M = 1$ component.
- **Predictor CL_{cc}.** This predictor is the chain ladder predictor based on splitting by claims code $\mathfrak{Y} = \mathfrak{Y}^{\text{cc}}_1 \cup \mathfrak{Y}^{\text{cc}}_2 \coloneqq \{\text{injury}\} \times \mathbb{R}_+ \cup \{\text{material}\} \times \mathbb{R}_+$. More precisely, $\hat{N}_{\text{CL}_{\text{cc}}} = \hat{N}^{\text{cw}}_{\text{CL}_{\text{cc}}}$ with

$$\begin{aligned}\hat{c}_{\text{CL}_{\text{cc}}}(x,t,y,d) = \mathbf{1}(P_0 \le \lfloor (T+D)/p \rfloor \le P_1) \\ + (\text{FtU}^{\mathfrak{Y}^{\text{cc}}_1}_{k_t+P-P_0} - \text{FtU}^{\mathfrak{Y}^{\text{cc}}_1}_{k_t+P-P_1}) \cdot \mathbf{1}(y_1 = \text{injury}) \\ + (\text{FtU}^{\mathfrak{Y}^{\text{cc}}_2}_{k_t+P-P_0} - \text{FtU}^{\mathfrak{Y}^{\text{cc}}_2}_{k_t+P-P_1}) \cdot \mathbf{1}(y_1 = \text{material}).\end{aligned}$$

- **Predictor $\text{CLFreqNet}_{\text{cc}}$.** This is the Chain Ladder Network based predictor for the partition $\mathfrak{Y} = \mathfrak{Y}^{\text{cc}}_1 \cup \mathfrak{Y}^{\text{cc}}_2$ defined in the description of CL_{cc}. It is only defined for $\mathfrak{Y}' \in \{\mathfrak{Y}^{\text{cc}}_1, \mathfrak{Y}^{\text{cc}}_2, \mathfrak{Y}\}$. More precisely, $\hat{N}_{\text{CLFreqNet}_{\text{cc}}} = \hat{N}^{\text{pw}}_{\text{CLFreqNet}_{\text{cc}}}$ with

$$\hat{c}_{\text{CLFreqNet}_{\text{cc}}}(i, A) \coloneqq \xi_r^{(i)}(I_p(k) \times \mathfrak{Y}' \times [0,\infty) \cap R_{\tau_0:\tau_1}) + \text{Leb}(I_p(x^{(i)},k)) \cdot \sum_{\substack{\mathbf{Y} \in \{\mathfrak{Y}^{\text{cc}}_1, \mathfrak{Y}^{\text{cc}}_2\} \\ \mathbf{Y} \subset \mathfrak{Y}'}} \hat{\lambda}^{\text{CL},\mathbf{Y}}(x^{(i)}, t_k) \cdot \Big(\frac{1}{\text{FtU}^{\mathbf{Y}}_{k+P-P_1}} - \frac{1}{\text{FtU}^{\text{CL},\mathbf{Y}}_{k_t+P-P_0}} \Big).$$

  The formula again stems from applying (5.2), this time with $M = 2$ and the partition by claim code.
- **Predictor cheating.** This is the predictor using the true parameters of the simulated model for prediction; it is not available in practice and only serves as a benchmark for evaluating the other predictors. More precisely, $\hat{N}_{\text{cheating}} = \hat{N}^{\text{pw}}_{\text{cheating}}$ with

$$\hat{c}_{\text{cheating}}(i, A) = \xi_r^{(i)}(I_p(k) \times \mathfrak{Y}' \times [0,\infty) \cap R_{\tau_0:\tau_1}) + \lambda(x^{(i)}, t_k) \cdot \int_{\mathfrak{Y}'} \int_{I_p(x^{(i)},k)} P_{D|X=x^{(i)},T=t_k,Y=y}(I_{\tau_0:\tau_1}(\tau,s))\,\mathrm{d}s\,\mathrm{d}P_{Y|X=x^{(i)},T=t_k}(y),$$

  corresponding to (4.1) with estimated distributions replaced by true distributions.

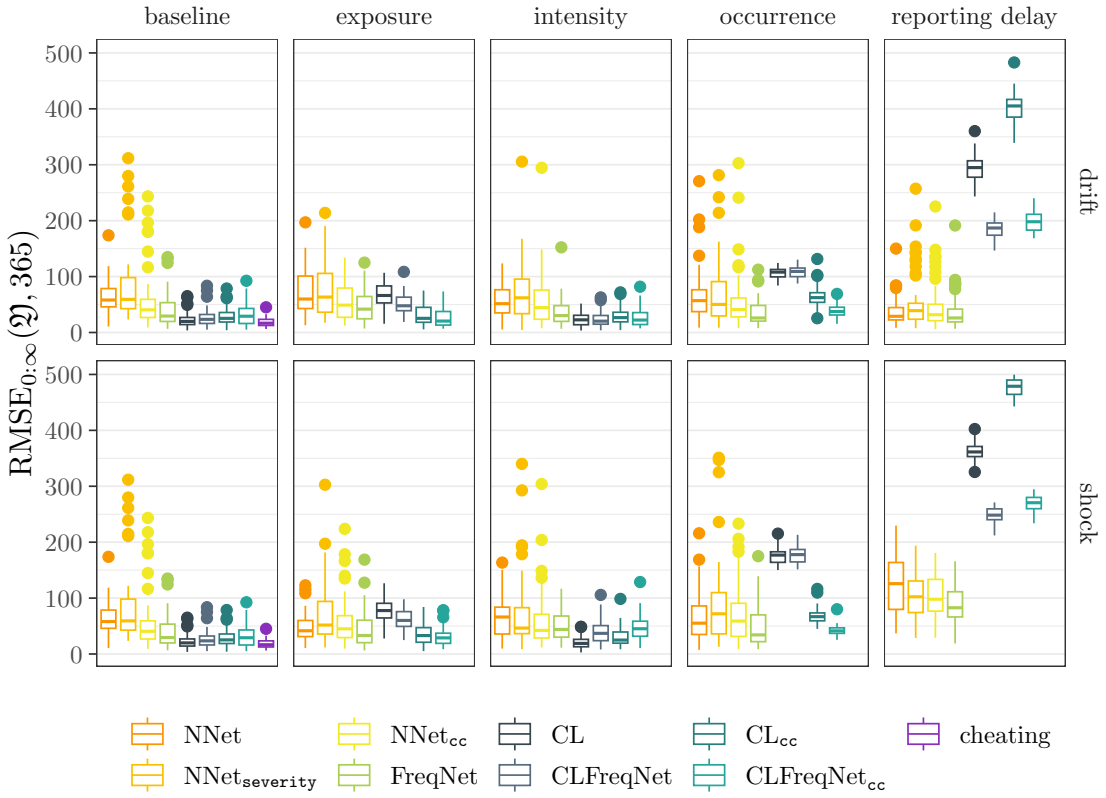


Figure 2: Boxplots of the overall error measure $\text{RMSE}_{0:\infty}(\mathfrak{Y}, 365)$, each based on $n = 50$ simulated paths. Legend is shown in column major order

7.3 Results

Throughout the simulation study, we use an evaluation period of $q = 365$ days. Figure 2 shows partly the same results as Bücher and Rosenstock (2022a, Figure 5), extended by the methods described in this paper and using the same color keys and predictor names, if applicable. The underlying error measure is the one from (6.2). Regarding the baseline scenario, we can see that modelling more and more parts of the claim arrival process, i.e., going from NNet to $\text{NNet}_{\texttt{cc}}$ and then finally to FreqNet, reduces the overall error with $\text{NNet}_{\texttt{cc}}$ seemingly exhibiting slightly larger variance. Only applying a partial model for the distribution of Y as in $\text{NNet}_{\texttt{severity}}$ increases the prediction error and its variance. We can also see that the Chain Ladder predictor provides close-to-optimal predictions on par with those obtained from the true model in this setting where the underlying Chain Ladder assumptions are exactly met.

For the Chain Ladder based methods, the error of the neural network predictors in the baseline scenario increases when compared to the pure factor based prediction - at the advantage of providing individual reserve predictions for each policy in the portfolio. This behavior can be explained by the training method used: All neural network fitting procedures with a poisson loss use the GLM skip connection described by Wüthrich (2019), but only on the 75% of the available data chosen for training. The 25% of the data used for hold-out validation therefor did not take part in the bias regularization, whereas the factor based methods had no hold-out data. If bias regularization was done on 100% of the data, the difference in errors would be smaller but not zero, because the bias regularization only ensures the total number of claims to remain constant, but not their allocation to accident years.

The results for the exposure scenario show that predictions can be improved when using portfolio information (in particular, exposure data); in fact, we see this improvement across all three approaches, i.e., NNet $\rightsquigarrow$ FreqNet, CL $\rightsquigarrow$ CLFreqNet and $\text{CL}_{\texttt{cc}}$ $\rightsquigarrow$ $\text{CLFreqNet}_{\texttt{cc}}$.

Heuristically, this can be explained by the fact that the drift in exposure is directly reflected by a drift in expected frequencies from the claim frequency models (see Figure 1), thereby influencing IBNR claim counts. The observed improvement is most prominent for the unpartitioned Chain Ladder approach, because the other basic methods can at least partially detect the changes via changes in the distribution of cc, which is also influenced by the exposure shift.

Changes in the claim intensity pose a challenge to frequency based approaches because it makes the underlying intensity, $\lambda(x,t)$, harder to train. Due to this disadvantage, one might expect to see a deterioration in prediction error for the frequency based approaches. Surprisingly, this is only found to be the case for the intensity shock scenario, and only so for the Chain Ladder based CLFreqNet and CLFreqNNet$_{\texttt{cc}}$. The intensity drift exhibits no noticeable deterioration in error and FreqNet shows a smaller improvement in error when confronted with an intensity shock compared to NNet, but an improvement nonetheless.

Regarding drifts and shocks in the occurrence process (i.e. in the distribution of cc), we do not observe a substantial effect on the prediction errors of the NNet based approach (i.e., they are similar as in the baseline scenario). Unpartitioned Chain Ladder does not deal well with these changes to the claims process and the frequency based extension doesn't manage to reduce the problem. Substantial improvements are found when moving from NNet to FreqNet and from CL$_{\texttt{cc}}$ to CLFreqNet$_{\texttt{cc}}$.

When the reporting delay distribution changes, Chain Ladder based methods start to perform very badly, even with partitioning. Since the introduced change effectively reduced the time-to-report, plain Chain Ladder approaches are confronted with higher claim counts upfront, which amplifies the error due to the multiplicative structure. This effect is dampened by frequency based extensions, because here the expected number of IBNR claims is based on the expected (long-term) frequency and not on short-term observations. Again, FreqNet shows a similar improvement compared to NNet as seen in other scenarios.

In summary, we can see that FreqNet performs well across all scenarios, improving on the method developed in Bücher and Rosenstock (2022a). The robustness to changes in exposure and reporting delays of Chain Ladder based estimates can be improved at little cost to overall accuracy by employing the CLFreqNet method.

We will now move our attention to the individual level results as measured by $\text{RMSE}^{\text{expo}}_{0:\infty}$ defined in (6.1), which are summarized in Figure 3. Within that figure, we do not display results for straightforward individual factor based predictions (i.e., multiplying the number of reported claims on an individual level by a factor-to-ultimate) because of their generally poor performance: for instance, in the baseline scenario, the mean $\text{RMSE}^{\text{expo}}_{0:\infty}(365)$ for CL, CL$_{\texttt{cc}}$ and noIBNR is $0.0843, 0.108$ and 0.0665, respectively, where noIBNR refers to simply predicting no IBNR claims at all. However, the results in Figure 3 show that the Chain Ladder based neural network predictors CLFreqNet and CLFreqNet$_{\texttt{cc}}$ provide viable solutions for allocating the IBNR claims from a Chain Ladder triangle to individual policies, albeit without yielding a full distributional model. In general, CLFreqNet and CLFreqNet$_{\texttt{cc}}$ exhibit very similar errors whereas FreqNet shows slightly smaller errors than the other two methods. Comparing the mean $\text{RMSE}^{\text{expo}}_{0:\infty}(365)$ for the methods noIBNR(0.0665), CLFreqNet(0.0656), CLFreqNet$_{\texttt{cc}}(0.0656)$, FreqNet(0.0655) and cheating(0.0653), we see that the Chain Ladder based methods score 72% of the performance of cheating when compared to noIBNR and FreqNet even achieves 78% of the improvement from noIBNR to cheating. It is interesting to note that the results are quite similar for all five scenarios, with the reporting delay scenario exhibiting the smallest difference between the noIBNR error and the error of the other three methods.

Of primal importance in insurance pricing is an accurately estimated risk model (which includes the frequency λ), for instance for reducing or avoiding cross-subsidisation within a portfolio. In Figure 4 we study the quality of the estimated frequency models obtained for the methods FreqNet, CLFreqNet and CLFreqNet$_{\texttt{cc}}$. As a measure for the quality of

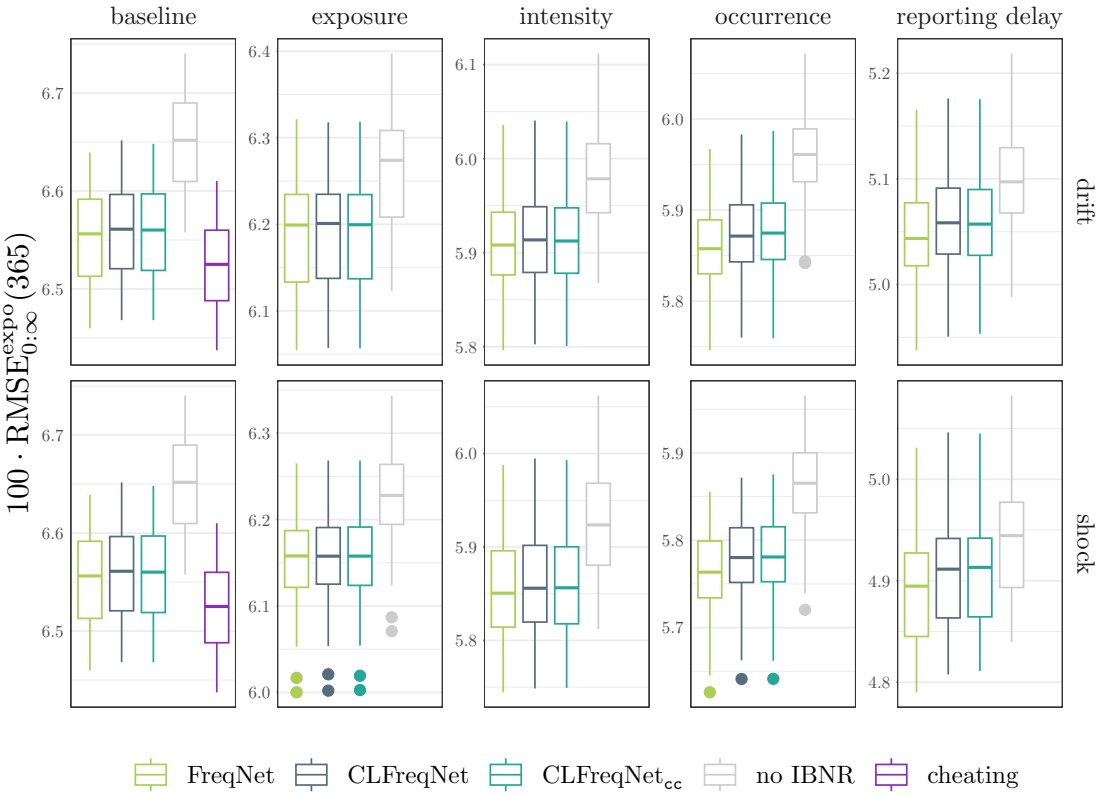


Figure 3: Boxplots of the individual-level error measure $\text{RMSE}^{\text{expo}}_{0:\infty}(365)$, each based on $n = 50$ simulated paths. The trivial "predictor" $\hat{N}^{\text{no IBNR}}(A) := N_r(A)$, predicting no IBNR claims, is also shown. Note that the predictors $\hat{N}^{\text{CL}}$ and $\hat{N}^{\text{CL}_{\text{cc}}}$ are not suitable for individual level claim count predictions as their $\text{RMSE}^{\text{expo}}_{0:\infty}$ is worse than $\hat{N}^{\text{no IBNR}}$. In the baseline scenario, $\text{RMSE}^{\text{expo}}_{0:\infty}(365)$ has a median of $8.45 \cdot 10^{-2}$ for $\hat{N}^{\text{CL}}$ and $10.8 \cdot 10^{-2}$ for $\hat{N}^{\text{CL}_{\text{cc}}}$, compared to $6.65 \cdot 10^{-2}$ for $\hat{N}^{\text{no IBNR}}$.

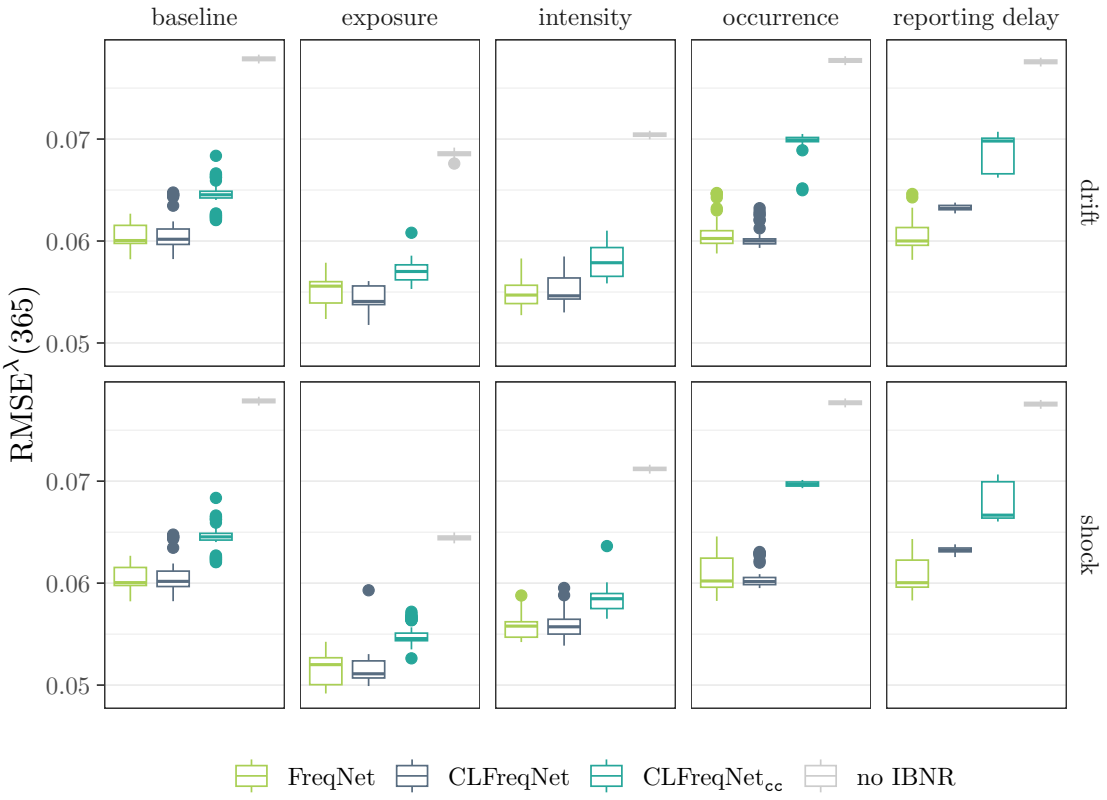


Figure 4: Boxplots of the frequency error measure $\mathrm{RMSE}^{\lambda}(365)$, each based on $n = 50$ simulated paths. For no IBNR, the estimate $\hat{\lambda}(x,t) \equiv \text{const} = \hat{N}_{0:\infty}(\mathfrak{X} \times [0,\tau) \times \mathfrak{Y} \times [0,\infty))/\sum_{i=1}^{N_{\text{pol}}} \mathrm{Leb}(C(x^{(i)}) \cap [0,\tau])$ was used.

the estimates, we use, for some evaluation period length q which is divisor of τ (as before, we fix $q = 365$ throughout),

$$\mathrm{RMSE}^{\lambda}(q) := \left(\frac{1}{\sum_{j=0}^{\tau/q-1} \#\mathcal{P}(q,j)} \sum_{\ell=0}^{\tau/q-1} \sum_{i \in \mathcal{P}(q,\ell)} \left(\hat{\lambda}(x^{(i)}, (\ell + \tfrac{1}{2}) \cdot q) - \lambda(x^{(i)}, (\ell + \tfrac{1}{2}) \cdot q) \right)^2 \right)^{\frac{1}{2}}, \tag{7.1}$$

where we have used the notation from Section 6. Note that, when using $\hat{\lambda}(x,t) \equiv \text{const} = \hat{N}_{0:\infty}(\mathfrak{X} \times [0,\tau) \times \mathfrak{Y} \times [0,\infty)) / \sum_{i=1}^{N_{\text{pol}}} \mathrm{Leb}(C(x^{(i)}) \cap [0,\tau])$ as an estimate for λ using the predictors CL and $\mathrm{CL}_{\mathtt{cc}}$ for $\hat{N}_{0:\infty}$ yields very similar $\mathrm{RMSE}^{\lambda}(365)$ as noIBNR, so they were left out of the plots in Figure 4 for readability. As an example, the baseline scenario has a mean $\mathrm{RMSE}^{\lambda}(365)$ of 0.0737 for noIBNR and of 0.0734 for both CL and $\mathrm{CL}_{\mathtt{cc}}$ whereas the frequency networks yield values from 0.0578 to 0.0632.

A priori, one would expect that RMSE^{λ} correlates with $\mathrm{RMSE}^{\text{expo}}_{0:\infty}$, since both are error measures at the individual policy level. Surprisingly, the results in Figure 4 show that this correlation breaks down for the Chain Ladder-based frequency models: while $\mathrm{RMSE}^{\text{expo}}_{0:\infty}$ is very similar for CLFreqNet and $\mathrm{CLFreqNet}_{\mathtt{cc}}$, RMSE^{λ} is smaller for CLFreqNet than it is for $\mathrm{CLFreqNet}_{\mathtt{cc}}$ in all scenarios. Heuristically, a larger variance of $\mathrm{CLFreqNet}_{\mathtt{cc}}$ may be explained by the fact that $\mathrm{CLFreqNet}_{\mathtt{cc}}$ is the only method of the three that uses two independent networks for the frequency of each claim code, and hence is based on twice the number of parameters. It is not fully clear however why this would deteriorate model quality. Another interesting observation is that despite FreqNet having a worse accident year level error (Figure 2), its underlying frequency model is comparable in quality to that of CLFreqNet in the baseline scenario.

8 Application to real data

In this section we will apply the different methods to a large real dataset containing motor legal insurance claims provided by a German insurance company. The dataset is described in Section 8.1. More detail on the prediction methods and estimation procedure can be found in Section 8.2. Due to the nature of real world data, observations are only available for a limited time frame. Therefor, model performance metrics cannot use ∞ as the time of evaluation, but must instead use a finite cutoff date. We examined two artificial truncation points, $\tau =$ 31st December 2017 and $\tau =$ 31st December 2018 and evaluate predictions for one year into the future, i.e. $\mathrm{RMSE}_{\tau:\tau+365}(\mathfrak{Y}, 365)$ and $\mathrm{RMSE}^{\text{expo}}_{\tau:\tau+365}(\mathfrak{Y}, 365)$. Results of this examination are presented and discussed in Section 8.3.

8.1 The Dataset

The dataset is the same as Bücher and Rosenstock (2022a). It contains a portfolio of about 250,000 motor legal insurance contracts and 65,000 corresponding claims with exposure and claims information available monthly from 31st December 2014 to 31st December 2020. Due to the extreme shock the COVID-19 pandemic had on the dataset, we chose to only consider data available up to 31st December 2019 for model evaluation.

Available policy-level data consists of

$$\mathfrak{X} = \{(\mathtt{tstart}, \mathtt{cstart}, \mathtt{product}, \mathtt{product\_version}, \mathtt{tariff}, \mathtt{installments}, \mathtt{dob}, \mathtt{dob\_missing}, \mathtt{sex}, \mathtt{coverage})\}.$$

Each feature is described in the following table.

Feature	Domain	Description
`tstart`	$\mathbb{R}$	start date of the contract
`cstart`	$\mathbb{R}$	start date of the customer relationship
`product`	$\{0,\dots,7\}$	insurance product
`product_version`	$\{0,\dots,13\}$	version of the insurance product
`tariff`	$\{0,1,2\}$	regular, public service, self-employed
`installments`	$\{0,\dots,3\}$	payment installments (annual, semi-annual, quarterly, monthly)
`dob`	$\mathbb{R}$	date of birth (missing values are imputed using median age at contract start)
`dob_missing`	$\{0,1\}$	indicates missing date of birth
`sex`	$\{0,1,2\}$	sex of the customer (male, female, missing)
`coverage`	$\{0,1\}$	group of covered people (family, single)

Claim-level data consists of accident time, `tacc`, reporting time `treport`, and the features

$$\mathcal{Y} = \{(\texttt{cc}, \texttt{covered}, \texttt{channel}, \texttt{reporter}, \texttt{tacc\_info})\}.$$

Feature	Domain	Description
`tacc`	$\mathbb{R}$	accident time
`treport`	$\mathbb{R}$	reporting time
`cc`	$\{0,\dots,4\}$	claim code
`covered`	$\{0,1\}$	indicates whether the claim is covered
`channel`	$\{0,\dots,3\}$	means of reporting (letter, fax, telephone, other)
`reporter`	$\{0,1,2\}$	who reported the claim (lawyer, policy holder, other)
`tacc_info`	$\{0,1,2\}$	regular, accident before policy start, accident date imprecise

Note that additional claim information (apart from accident time and reporting time) is categorical with a total of $5 \cdot 2 \cdot 4 \cdot 3 \cdot 3 = 360$ possible combinations.

8.2 Predictors

We provide a detailed overview of the predictors evaluated on the dataset described in Section 8.1. For the macro-level error measure, $\mathrm{RMSE}_{\tau:\tau+365}(\mathfrak{Y}, 365)$, we will compare a total of six predictors, three of which can provide viable micro-level predictors. The micro-level predictors are compared using $\mathrm{RMSE}^{\mathrm{expo}}_{\tau:\tau+365}(\mathfrak{Y}, 365)$. Most of the predictors are defined analogously to those in Section 7.2 and we will reuse the notation defined there.

- **Predictor NNet.** The original method from (Bücher and Rosenstock 2022a), formula (19).
- **Predictor $\mathrm{NNet}_{\mathfrak{Y}}$.** This predictor is based on the estimated claim feature model. Since all claim features are discrete, this modelling step was done using a single discrete distribution with 360 different possible outcomes. More precisely, recalling the notation $\hat{N}^{\mathrm{cw}}$ from Section 7.2, we define $\hat{N}_{\mathrm{NNet}_{\mathfrak{Y}}} := \hat{N}^{\mathrm{cw}}_{\mathrm{NNet}_{\mathfrak{Y}}}$

$$\hat{c}_{\mathrm{NNet}_{\mathrm{cc}}}(x,t,y,d) := \frac{\hat{p}_{\mathrm{rep}}(\tau_0,\tau_1,x,k_t)}{\hat{p}_{\mathrm{rep}}(0,\tau,x,k_t)}$$

  with $\hat{p}_{\mathrm{rep}} = \hat{p}_{\mathrm{prep}}(\tau_0,\tau_1,x,k_t)$ defined as

$$\hat{p}_{\mathrm{rep}} := \sum_{y\in\mathfrak{Y}'} \int_{I_p(x,k_t)} \hat{P}_{D|X=x,T=t_{k_t},Y=y}((\tau_0-s,\tau_1-s])\,\mathrm{d}s\hat{P}_{Y|X=x,T=t_{k_t}}(y).$$

- **Predictor FreqNNet.** The predictor from (4.1).

- **Predictor CL.** This predictor is the basic chain ladder predictor.
- **Predictor $\mathbf{CL}_{\mathtt{cc}}$.** This predictor is the chain ladder predictor based on splitting by claims code $\mathfrak{Y} = \mathfrak{Y}_0^{\mathtt{cc}} \cup \mathfrak{Y}_1^{\mathtt{cc}} \cup \mathfrak{Y}_2^{\mathtt{cc}} \cup \mathfrak{Y}_3^{\mathtt{cc}} \cup \mathfrak{Y}_4^{\mathtt{cc}}$. More precisely, recalling the notation $\hat{N}^{\mathrm{cw}}$ from Section 7.2, $\hat{N}_{\mathrm{CL}_{\mathtt{cc}}} = \hat{N}^{\mathrm{cw}}_{\mathrm{CL}_{\mathtt{cc}}}$ with

$$\begin{aligned}\hat{c}_{\mathrm{CL}_{\mathtt{cc}}}(x,t,y,d) = \mathbf{1}(P_0 \le \lfloor (T+D)/p \rfloor \le P_1) \\ + (\mathrm{FtU}^{\mathfrak{Y}_0^{\mathtt{cc}}}_{k_t+P-P_0} - \mathrm{FtU}^{\mathfrak{Y}_0^{\mathtt{cc}}}_{k_t+P-P_1}) \cdot \mathbf{1}(y_1 = 0) \\ + (\mathrm{FtU}^{\mathfrak{Y}_1^{\mathtt{cc}}}_{k_t+P-P_0} - \mathrm{FtU}^{\mathfrak{Y}_1^{\mathtt{cc}}}_{k_t+P-P_1}) \cdot \mathbf{1}(y_1 = 1) \\ + (\mathrm{FtU}^{\mathfrak{Y}_2^{\mathtt{cc}}}_{k_t+P-P_0} - \mathrm{FtU}^{\mathfrak{Y}_2^{\mathtt{cc}}}_{k_t+P-P_1}) \cdot \mathbf{1}(y_1 = 2) \\ + (\mathrm{FtU}^{\mathfrak{Y}_3^{\mathtt{cc}}}_{k_t+P-P_0} - \mathrm{FtU}^{\mathfrak{Y}_3^{\mathtt{cc}}}_{k_t+P-P_1}) \cdot \mathbf{1}(y_1 = 3) \\ + (\mathrm{FtU}^{\mathfrak{Y}_4^{\mathtt{cc}}}_{k_t+P-P_0} - \mathrm{FtU}^{\mathfrak{Y}_4^{\mathtt{cc}}}_{k_t+P-P_1}) \cdot \mathbf{1}(y_1 = 4),\end{aligned}$$

  where $y_1 = \mathtt{cc}$.
- **Predictor $\mathbf{CLFreqNNet}_{\mathtt{cc}}$.** This is the Chain Ladder Network based predictor for the partition $\mathfrak{Y} = \mathfrak{Y}_0^{\mathtt{cc}} \cup \mathfrak{Y}_1^{\mathtt{cc}} \cup \mathfrak{Y}_2^{\mathtt{cc}} \cup \mathfrak{Y}_3^{\mathtt{cc}} \cup \mathfrak{Y}_4^{\mathtt{cc}}$ defined in the description of $\mathrm{CL}_{\mathtt{cc}}$. The formula stems from applying (5.2) with $M = 5$ and the partition by claim code.

8.3 Results

The seven available predictors are evaluated for one year ahead and on an evaluation period of $q = 365$. In comparison to the simulation study, the **cheating** predictor is missing and the two plain Chain Ladder predictors have no uncertainty due to their deterministic algorithm. Also, because stepwise estimation of the claim feature distribution was not necessary, there is only one predictor $\mathbf{NNet}_{\mathfrak{Y}}$ based on the full claim feature distribution instead of the two predictors $\mathbf{NNet}_{\mathtt{severity}}$ and $\mathbf{NNet}_{\mathtt{cc}}$.

Summarily, despite the fact that the model selection strategy has not been fine-tuned to the problem at hand and showed a rather unreliable performance overall, **FreqNet** shows promising results on a micro-level at an acceptable cost on the macro-level.

Figure 5 shows the accident year level prediction error $\mathrm{RMSE}_{\tau:\tau+365}(\mathfrak{Y}, 365)$ for one year ahead across the seven methods for the two artificial truncation points. In contrast to most simulation results on the ultimate accident year level prediction error, we see an increase in error of $\mathbf{NNet}_{\mathfrak{Y}}$ compared to **NNet**. This loss could possibly be overcome by optimizing the claim feature model architecture, which was fixed as a $(10, 5)$ feed-forward network for simplicity. For $\tau =$ 31st December 2017 the distribution of errors for **FreqNet** also seems to deteriorate, whereas $\tau =$ 31st December 2018 exhibits behavior more consistent with the simulation study, decreasing the error while maintaining a similar variance. While for $\tau =$ 31st December 2017, the selected model has a very low error compared to all candidates, the model selected for $\tau =$ 31st December 2018 exhibits a worse accident year level error than the Chain Ladder methods, even though the median error among all candidate models was lower than that of Chain Ladder. Regarding **CLFreqNet** and $\mathbf{CLFreqNet}_{\mathtt{cc}}$, one can see the impact of the training procedure (holding out 25% of the data for validation) increasing the overall variance in error compared to their Chain Ladder counterparts.

In summary, the new methods seem to provide similar accuracy on an accident year level when compared to the underlying methods **NNet**, **CL** or $\mathbf{CL}_{\mathtt{cc}}$.

The new methods **FreqNet**, **CLFreqNet** and $\mathbf{CLFreqNet}_{\mathtt{cc}}$ provide exposure-level IBNR predictions, which can be compared in Figure 6. As with the simulation study, plain triangle based methods can not be used to obtain viable predictors for micro-level claim counts, so the trivial **noIBNR** is used as a basic reference. Unfortunately it is not possible to also provide a theoretical best prediction on real data, so there is no **cheating** benchmark with which the results could be compared. We refer to Figure 3 for

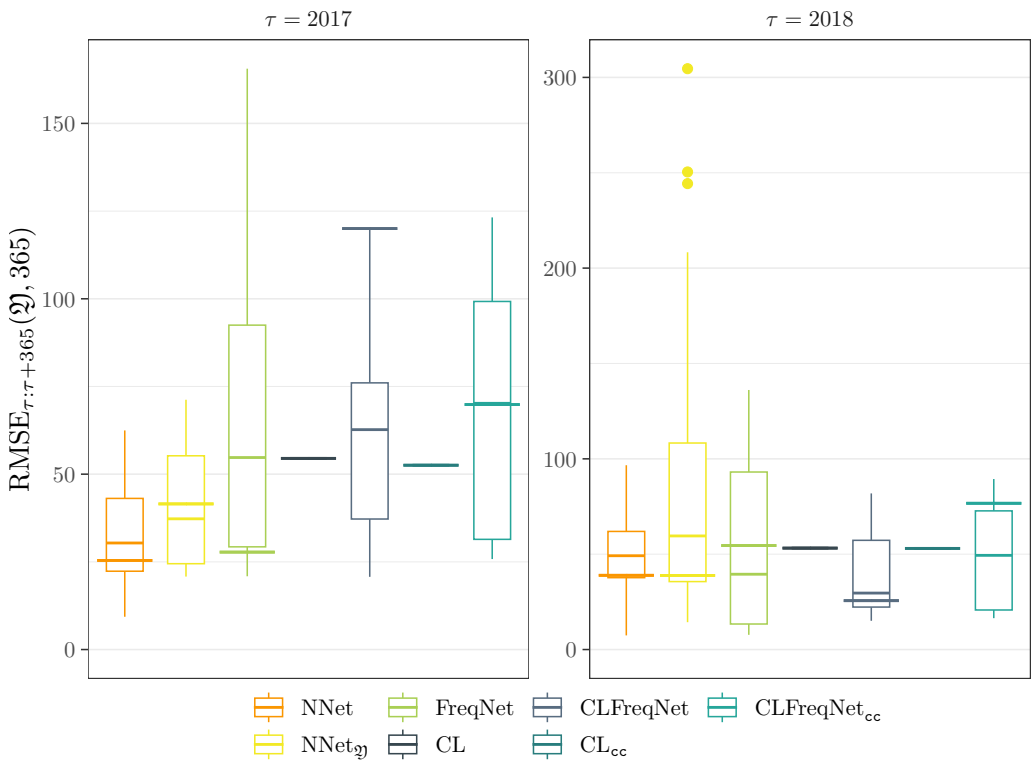


Figure 5: Boxplot comparison of $\text{RMSE}_{\tau:\tau+365}(\mathfrak{Y}, 365)$ for the different methods. Final selected models shown as wide horizontal line.

the corresponding simulation study results which do have this benchmark. The micro-level results are more comparable across the different truncation times, showing a similar pattern to that of the simulation study with one exception: There is a larger separation between the errors of **FreqNet** and those of **CLFreqNet** and **CLFreqNet$_{cc}$**.

9 Conclusion

Two new methods for joint prediction of micro-level IBNR claim counts and claim frequencies have been developed and applied to real and simulated data. Results show promising accuracy on an exposure level compared to the theoretical optimum under laboratory conditions. The new methods also permit assigning IBNR claim count predictions on a policy level such that policies without claims can receive a non-zero IBNR prediction, which is an advantage for analysis of small portfolios where Chain Ladder estimates - even with external parameters estimated on a larger dataset - fail to produce accurate estimates. The presented case studies uncover several opportunities for further research:

1. The distributional assumption of a BDEGP family for reporting delays in Model 1 might not be suitable for all applications. Future work could examine results with other reporting delay distribution families.
2. The functional relationships in Model 1, Model 2 and Model 3 have all been chosen as MLPs. All of these relationships could be chosen from a different function family, e.g., other families used in machine learning such as regression trees.
3. The architecture of all MLPs was simply chosen and no hyperparameter-optimization was performed. Strategies for architecture selection or different architectures could be examined.
4. Different strategies for model selection of a final model among candidate models could be explored.
5. Definition 2.1 can be extended by a claim settlement process for each claim, such as the

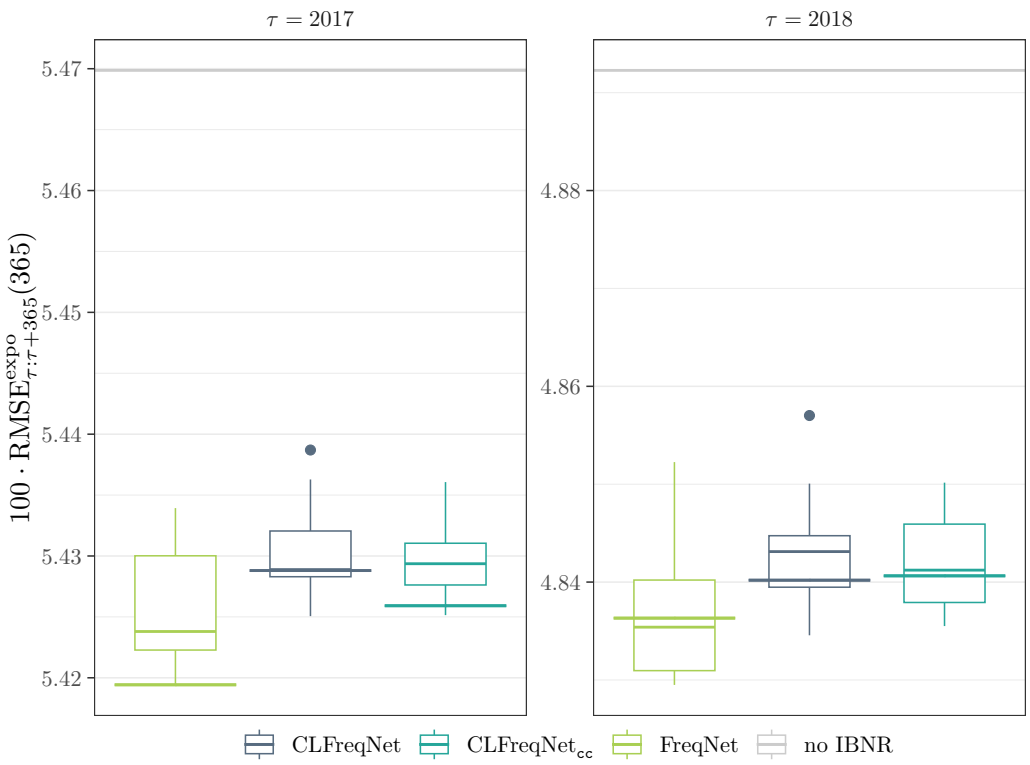


Figure 6: Comparison of $\text{RMSE}^{\text{expo}}_{\tau:\tau+365}(\mathfrak{Y}, 365)$ for the different methods.

one presented in Antonio and Plat (2014) but on a policy level, to allow joint modelling of IBNR and RBNS payments.

References

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Allaire, JJ et al. (2016). *tensorflow: R Interface to TensorFlow*. URL: https://github.com/rstudio/tensorflow.

Antonio, Katrien, Els Godecharle and Robin Van Oirbeek (2016). 'A Multi-State Approach and Flexible Payment Distributions for Micro-Level Reserving in General Insurance'. In: *SSRN*.

Antonio, Katrien and Richard Plat (2014). 'Micro–level stochastic loss reserving for general insurance'. In: *Scandinavian Actuarial Journal* 2014.7, pp. 649–669.

Badescu, Andrei L., X. Sheldon Lin and Dameng Tang (2016). 'A marked Cox model for the number of IBNR claims: theory.' In: *Insurance: Mathematics and Economics* 69, pp. 29–37.

Baudry, Maximilien and Christian Y. Robert (2019). 'A machine learning approach for individual claims reserving in insurance'. In: *Applied Stochastic Models in Business and Industry* 2019.35, pp. 1127–1155.

Bücher, Axel and Alexander Rosenstock (2022a). 'Micro-level prediction of outstanding claim counts based on novel mixture models and neural networks.' In: *Eur. Actuar. J.*

— (2022b). *Supplementary Material: Micro-level prediction of outstanding claim counts based on novel mixture models and neural networks*. URL: https://link.springer.com/article/10.1007/s13385-022-00314-4#Sec27.

Chaoubi, Ihsan et al. (2023). 'Micro-level reserving for general insurance claims using a long short-term memory network'. In: *Applied Stochastic Models in Business and Industry* 1, pp. 1–26. DOI: 10.1002/asmb.2750.

Chollet, François, JJ Allaire et al. (2017). *R Interface to Keras*. URL: https://github.com/rstudio/keras.

De Felice, Massimo and Franco Moriconi (2019). 'Claim Watching and Individual Claims Reserving Using Classification and Regression Trees'. In: *Risks* 7.4.

Gabrielli, Andrea and Mario V. Wüthrich (Mar. 2018). 'An Individual Claims History Simulation Machine'. In: *Risks* 6.2, p. 29.

Goldburd, Mark et al. (2016). 'Generalized linear models for insurance rating'. In: *Casualty Actuarial Society, CAS Monographs Series* 5.

Last, Günter and Mathew Penrose (2018). *Lectures on the Poisson Process*. Cambridge: Cambridge University Press.

Norberg, Ragnar (1993). 'Prediction of Outstanding Liabilities in Non-Life Insurance'. In: *ASTIN Bulletin* 23.1, pp. 95–115.

— (1999). 'Prediction of Outstanding Liabilities II. Model Variations and Extensions'. In: *ASTIN Bulletin* 29.1, pp. 5–25.

Okine, A. Nii-Armah, Edward W. Frees and Peng Shi (2022). 'Joint model prediction and application to individual-level loss reserving'. In: *ASTIN Bulletin: The Journal of the IAA* 52.1, pp. 91–116. DOI: 10.1017/asb.2021.28.

Rosenstock, Alexander (2023). *reservr: Fit Distributions and Neural Networks to Censored and Truncated Data*. R package version 0.0.1. URL: https://ashesitr.github.io/reservr/.

Wang, Zhigao, Xianyi Wu and Chunjuan Qiu (2021). 'THE IMPACTS OF INDIVIDUAL INFORMATION ON LOSS RESERVING'. In: *ASTIN Bulletin: The Journal of the IAA* 51.1, pp. 303–347. DOI: 10.1017/asb.2020.42.

Wüthrich, Mario V. (2019). 'Bias Regularization in Neural Network Models for General Insurance Pricing'. In: *SSRN*.

Wüthrich, Mario V. and Christoph Buser (2019). 'Data Analytics for Non-Life Insurance Pricing'. In: *SSRN*.

Wüthrich, Mario V. and Michael Merz (2015). 'Stochastic Claims Reserving Manual: Advances in Dynamic Modeling'. In: *SSRN*.

Fitting Distributions and Neural Networks to Censored and Truncated Data: The R Package reservr

Alexander Rosenstock
Heinrich-Heine-Universität Düsseldorf
ARAG SE

Abstract

Random truncation, i.e. truncated observations where the point of truncation varies by observation, and interval censoring arise naturally in various fields, such as insurance, operations research or medicine. This article presents the R package **reservr**, which implements distribution parameter estimation and distributional regression for randomly truncated and interval censored data based on (conditional) maximum likelihood. The package provides a flexible interface to specify (weighted) randomly truncated and interval censored observations, to specify distribution families to be estimated, and to compute (conditional) maximum-likelihood based parameter estimates. Distributional regression is supported via an interface to the R package **tensorflow** to build neural network models for distributional regression of censored and randomly truncated outcomes with arbitrary distribution families. The interface allows for arbitrary network architectures, including multi-modality and pre-initialization of network weights from a global parameter estimate to improve stability. Additional utilities for application in a general insurance context, as well as the usual random sampling, density, probability and quantile functions for distributions are provided.

Keywords: distribution fitting, random truncation, interval censoring, distributional regression, deep learning, R.

1. Introduction

Statistical analyses are typically concerned with modelling and estimating the distribution of some measured variable of interest Y, called the outcome, possibly conditional on the value of one or several endogenous variables X, called predictors. In the absence of endogenous variables, this process is usually called distribution fitting, and in the presence of endogenous variables it is called regression. Classical regression, such as via generalized linear models (GLMs), is concerned with the influence of endogenous variables on the mean of the outcome, i.e., $\mathsf{E}(Y|X) = f(X)$, and often links other parameters of the conditional outcome distribution to its mean. A gentle introduction to generalized linear models can be found in Dobson and Barnett (2018). An implementation of GLMs is available in the **stats** R package, which is part of R itself (R Core Team 2023). Some models also allow specification of additional parameters of the conditional outcome distribution, such as Generalized Additive Models for Location, Scale and Shape (Stasinopoulos and Rigby 2007). More recently, deep distributional

regression has been proposed, which allows for flexible specification of individual outcome distribution parameters (Rügamer *et al.* 2023).

Statistical methods (such as those described and implemented in the previously mentioned papers) often require complete data, that is full information on all observations (X, Y) of interest. In this paper, we describe an R-package that allows for distributional regression in three common observation schemes that do not provide complete data. First of all, data with *interval censoring* applied to the outcome Y refers to the case where only lower and upper bounds for Y are observed, instead of the actual value. Next, truncated data misses observations for which the outcome Y falls out of a certain lower and upper truncation bound. We consider the case of *random truncation*, where these truncation bounds are also random variables that may vary for each observation. Finally, we consider a combination of the two, *randomly truncated interval censoring*.

The three scenarios can be combined into a single general scheme: instead of observing the real-valued target variable Y (with μ-density f_θ and c.d.f. F_θ, where μ is a sigma-finite measure on $\mathbb{R}$ and θ is a parameter vector in some parameter space Θ), we observe the vector (M, V, L, U), which satisfies $L \leq M \leq V \leq U$ and $L < U$. Its coordinates have the following interpretation: the last two coordinates, which satisfy $-\infty \leq L < U \leq \infty$, encode truncation: we only happen to observe (M, V, L, U) if $L < Y \leq U$; in particular, a non-truncated observations means that $L = -\infty$ and $U = \infty$. The first two coordinates, which satisfy $L \leq M \leq V \leq U$ and which may be $\mp\infty$, encode censoring: the observation $(M, V) = (m, v)$ means that the target variable Y satisfies $Y \in (m, v]$ if $m < v$ and $Y = m$ if $m = v$; the latter corresponds to an uncensored observation of Y.

It is instructive to focus on the simpler problem of distribution parameter estimation before proceeding with distributional regression. Suppose we observe an independent sample $\mathcal{J} = \{(m_i, v_i, l_i, u_i) : i = 1, \ldots, n\}$ of (M, V, L, U). Suggested by standard maximum (conditional) likelihood approaches for truncated (Dörre and Emura 2019) and censored observations (Sun 2006), we suggest to estimate θ by maximizing the objective function

$$\ell(\theta) = \sum_{(m,v,l,u)\in\mathcal{J}} \Big\{ \log f_\theta(m)\mathbf{1}(m = v) + \log F_\theta((m, v])\mathbf{1}(m < v) \Big\} - \log F_\theta((l, u]), \tag{1}$$

where we use the notation $F_\theta((l, u]) = F_\theta(u) - F_\theta(l)$. A detailed motivation for this approach under suitable conditions ensuring that the censoring is non-informative is given in Section 1.1 below. For later purposes, it is helpful to attach a weight w_i to each observation (m_i, v_i, l_i, u_i). Denoting the resulting sample by $\mathfrak{I} = \{(m_i, v_i, l_i, u_i, w_i)\}$, we aim at maximizing the weighted sum of the (conditional) log-likelihoods

$$\ell(\theta) = \sum_{(m,v,l,u,w)\in\mathfrak{I}} w \cdot \Big[\Big\{ \log f_\theta(m)\mathbf{1}(m = v) + \log F_\theta((m, v])\mathbf{1}(m < v) \Big\} - \log F_\theta((l, u]) \Big]. \tag{2}$$

A practical example of random truncation arises when modelling the reporting delay of claims in general insurance. The target variable Y is the reporting delay of an accident happening at accident time T_0, which is hence reported to the insurer at calendar time $Y + T_0$. The truncation bounds (L, U) for Y will be equal to $(0, \tau - T_0)$ with τ the current calendar time.

Combined random truncation with interval censoring can occur when modelling failure times when only survival data at two (or more) maintenance appointments some time after purchase

is captured, and only for items that are sold. The target variable Y is the failure time of an item. Item condition (failed / functional) can be observed at maintenance times M_0 and M_1, which may vary for each item. For each maintained item, the production time P_0 and the purchase time P_1 is also known. Only items that are functional at purchase time P_1 are observed at the maintenance times. This gives rise to truncation bounds $(L, U) = (P_1 - P_0, \infty)$ and censoring interval bounds $(M, V) \in \{(P_1 - P_0, M_0 - P_0), (M_0 - P_0, M_1 - P_0), (M_1 - P_0, \infty)\}$, depending on the item condition at times M_0 and M_1.

In the setting of distributional regression, weighted samples $\mathfrak{I}$ (M, V, L, U, W) have associated predictors $X \in \mathfrak{X}$, resulting in observations of the shape $\mathfrak{I}_{\text{reg}} = \{(m_i, v_i, l_i, u_i, w_i, x_i) : i = 1, \ldots, n\}$. We are interested in estimating a regression function $g : \mathfrak{X} \to \Theta$ given a sample $\mathfrak{I}_{\text{reg}}$, a parameterized family $\mathcal{F} = \{F_\theta \mid \theta \in \Theta\}$ and a family $\mathcal{G}$ of functions from $\mathfrak{X}$ to Θ. It is assumed that there exists a fuction $g \in \mathcal{G}$ such that the conditional distribution of $Y|X = x$ is $F_{g(x)}$. Distributional regression can be formulated as the maximization problem

$$\begin{aligned} \hat{g} &\in \arg\max_{g \in \mathcal{G}} \ell(g|\mathfrak{I}_{\text{reg}}), \text{ where} \\ \ell(g|\mathfrak{I}_{\text{reg}}) &:= \sum_{(m,v,l,u,w,x) \in \mathfrak{I}_{\text{reg}}} w \cdot \begin{cases} \log f_{g(x)}(m) - \log F_{g(x)}((l,u]) & m = v \\ \log F_{g(x)}((m,v]) - \log F_{g(x)}((l,u]) & m < v \end{cases}. \end{aligned} \tag{3}$$

Compared to Equation (2), the global parameter θ is replaced by the regression function g evaluated at the associated predictors x.

1.1. Motivation of Equation (1).

It is instructive to start by considering an untruncated, censored observation where $l = -\infty, u = \infty$ and $m < v$. The only information we obtain from the observation (m, v, l, u) is then that $Y \in (m, v]$. For deriving the relevant likelihood contribution, we may follow the stochastic approach to interval censored observations described in Groeneboom and Wellner (1992, Case 2): let (C_1, C_2) denote a random vector in $\mathbb{R}^2$ that is independent of the target variable Y and which satisfies $\mathsf{P}(C_1 < C_2) = 1$. Let

$$D := \mathbf{1}(Y > C_1) + \mathbf{1}(Y > C_2),$$

and define new random variables $(M, V) = f(Y, C_1, C_2)$ by

$$(M, V) := \begin{cases} (-\infty, C_1), & D = 0, \\ (C_1, C_2), & D = 1, \\ (C_2, \infty), & D = 2. \end{cases}$$

Note that D can be reconstructed from (M, V): we have $D = 0$ if $M = -\infty$, $D = 1$ if $-\infty < M < V < \infty$ and $D = 2$ if $V = \infty$.

It is instructive to proceed with the case where (C_1, C_2) and hence (M, V) is discrete. Then, for $(m, v) \in \operatorname{supp}(M, V) \cap \mathbb{R}^2$, we have

$$\begin{aligned} \mathsf{P}(M = m, V = v) &= \mathsf{P}(M = m, V = v, D = 1) \\ &= \mathsf{P}(C_1 = m, C_2 = v, Y \in (m, v]) \\ &= F_\theta((m, v]) \cdot \mathsf{P}(C_1 = m, C_2 = v). \end{aligned}$$

Likewise, for $(m, v) \in \operatorname{supp}(M, V) \cap (\{-\infty\} \times \mathbb{R})$, we obtain

$$\begin{aligned}\mathsf{P}(M = -\infty, V = v) &= \mathsf{P}(M = -\infty, V = v, D = 0) \\ &= \mathsf{P}(C_1 = v, Y \leq v) \\ &= F_\theta((-\infty, v]) \cdot \mathsf{P}(C_1 = v)\end{aligned}$$

and finally, for $(m, v) \in \operatorname{supp}(M, V) \cap (\mathbb{R} \times \{\infty\})$,

$$\begin{aligned}\mathsf{P}(M = m, V = \infty) &= \mathsf{P}(M = m, V = \infty, D = 2) \\ &= \mathsf{P}(C_2 = m, Y > m) \\ &= F_\theta((m, \infty]) \cdot \mathsf{P}(C_2 = m).\end{aligned}$$

If we assume that the distribution of the censoring variable (C_1, C_2) is non-informative, i.e., its distribution does not depend on θ, the likelihood of observing $(M, V) = (m, v)$ is equal to $F_\theta((m, v])$, up to a factor that does not depend on θ. A similar argumentation can be used in the non-discrete case. Overall, noting that $F_\infty((-\infty, \infty]) = 1$, we have motivated the likelihood contribution $F_\theta((m, v]) \cdot \mathbf{1}(m < v)$ for a censored, untruncated observation in (1).

Next, consider an uncensored, truncated observation (m, v, l, u) where $y = m = v$; we may hence identify such an observation with (y, l, u). We may then proceed as in Bücher and Rosenstock (2022a) and assume that (L, U) is independent of Y and satisfies $L \leq U$, with L possibly equal to $-\infty$ and U possibly equal to ∞. Further, (L, U) shall have a density $f_{(L,U)}$ with respect to some dominating sigma-finite measure ν. Truncation means that we only happen to observe (Y, L, U) if $L < Y \leq U$. As a consequence, any observed value with $M = V$ can be regarded as being drawn from the $(\mu \otimes \nu)$-density

$$f_{(Y,L,U)|L<Y\leq U}(y, l, u) = \frac{f_{(L,U)}(l, u) f_\theta(y)}{\mathsf{P}(L < Y \leq U)} \mathbf{1}(l < y \leq u). \tag{4}$$

Subsequently, we write $(Y^{(t)}, L^{(t)}, U^{(t)})$ for a random vector following the above density, i.e.,

$$f_{(Y^{(t)},L^{(t)},U^{(t)})}(y, l, u) = f_{(Y,L,U)|L<Y\leq U}(y, l, u).$$

Conditioning this density on $(L^{(t)}, U^{(t)}) = (l, u)$, we arrive at an expression that does not involve the nuisance density $f_{(L,U)}$:

$$\begin{aligned}f_{Y^{(t)}|L^{(t)}=l,U^{(t)}=u}(y) &= \frac{f_{(Y^{(t)},L^{(t)},U^{(t)})}(y, l, u)}{f_{(L^{(t)},U^{(t)})}(l, u)} \\ &= \frac{f_{(Y,L,U)|L<Y\leq U}(y, l, u)}{\int_{(l,u]} f_{(Y,L,U)|L<Y\leq U}(z, l, u) \, \mathrm{d}\mu(z)} = \frac{f_\theta(y)}{\int_{(l,u]} f_\theta(z) \, \mathrm{d}\mu(z)}.\end{aligned}$$

Overall, we arrive at the (conditional) log-likelihood contribution $\log f_\theta(y) - \log F_\theta((l, u])$ for an uncensored, truncated observation in (1).

Finally, truncation and censoring can occur at the same time, i.e., we have $l \leq m < v \leq u$ with either $l \neq -\infty$ or $u \neq \infty$. In accordance with the previous two cases, we make the

assumption that $Y, (C_1, C_2)$ and (L, U) are mutually independent and satisfy $C_1 < C_2$ and $L < U$. Define

$$D = \mathbf{1}(Y > C_1) + \mathbf{1}(Y > C_2)$$

and

$$(M, V) := \begin{cases} (L, \min(U, C_1)), & D = 0, \\ (\max(L, C_1), \min(C_2, U)), & D = 1, \\ (\max(L, C_2), U), & D = 2. \end{cases}$$

For simplicity, assume that all random variables are discrete. For any observation (m, v, l, u), one of the following four cases is met

$$l < m < v < u, \quad l = m < v < u, \quad l < m < v = u, \quad l = m < v = u.$$

In case $l < m < v < u$, we have

$$\begin{aligned} \mathsf{P}(M = m, V = v | L = l, U = u, L < Y \leq U) &= \frac{\mathsf{P}(C_1 = m, C_2 = v, Y \in (m, v], L = l, U = u)}{\mathsf{P}(L = l, U = u, l < Y \leq u)} \\ &= \frac{\mathsf{P}(C_1 = m, C_2 = v) F_\theta((m, v])}{F_\theta((l, u])} \end{aligned}$$

by the independence assumption. The factor in front does not depend on θ and is irrelevant for the (conditional) likelihood contribution. Likewise, in case $l = m < v < u$, we have

$$\mathsf{P}(M = l, V = v | L = l, U = u, L < Y \leq U) = \frac{\mathsf{P}(M = l, V = v, L = l, U = u, l < Y \leq u)}{\mathsf{P}(L = l, U = u, l < Y \leq u)}.$$

By definition of (M, V), the event in the numerator is the disjoint union of the following two sets:

$$\begin{aligned} &\{D = 0, C_1 = v, L = l, U = u, l < Y \leq u\} = \{C_1 = v, L = l, U = u, Y \in (l, v]\} \\ &\{D = 1, C_1 \leq l, C_2 = v, L = l, U = u, l < Y \leq u\} = \{C_1 \leq l, C_2 = v, L = l, U = u, Y \in (l, v]\}. \end{aligned}$$

By independence, we obtain that

$$\mathsf{P}(M = l, V = v | L = l, U = u, L < Y \leq U) = \{\mathsf{P}(C_1 = v) + \mathsf{P}(C_1 \leq l, C_2 = v)\} \frac{F_\theta((l, v])}{F_\theta((l, u])}.$$

Again, the factor in front of the fraction is independent of θ and is irrelevant for the likelihood. The two cases $l < m < v = u$ and $l = m < v = u$ can be treated similarly; in all cases, the likelihood contribution is equal to $F_\theta((m, v])/F_\theta((l, u])$ times a factor that does not depend on θ.

1.2. Related packages

For the less general cases of non-informative censoring without random truncation and fixed truncation, i.e., (L, U) constant for all observations, as well as for estimation of distribution

parameters in the absence of censoring or random truncation, there are a number of R packages that can fit distributions, some of them also supporting weights. Among these are **MASS** (Venables and Ripley 2002), **fitdistrplus** (Delignette-Muller and Dutang 2015), **survival** (Therneau 2023), **flexsurv** (Jackson 2016). Note that fixed truncation is an operation that can be baked into the distribution family whose parameters are to be estimated, allowing for classical maximum likelihood estimation. Many of the packages also support classic regression of expected values given predictors. Distributional regression packages, such as **gamlss** (Stasinopoulos and Rigby 2007) and **deepregression** (Rügamer *et al.* 2023) currently do not support interval censoring or random truncation. See the following table for an overview of available features for each package.

Package	sample weights	censoring	random truncation	regression
MASS	no	no	no	classic
fitdistrplus	only integer	supported	no	no
survival	supported	supported	no	classic
flexsurv	supported	supported	no	classic
gamlss	supported	no	no	distributional
deepregression	supported	no	no	distributional
reservr	supported	supported	supported	distributional

Another R6-based interface is provided by **ROOPSD** (Robin 2022).

reservr builds upon the R packages **tensorflow** (Allaire and Tang 2022) and **keras** (Chollet, Allaire *et al.* 2017) as an interface to the machine learning library **TensorFlow** (Abadi *et al.* 2015) to perform distributional regression. This underlying infrastructure is shared with the distributional regression package **deepregression** (Rügamer *et al.* 2023). The latter also supports distributional regression, but at the time of writing requires complete samples and does not support truncation or censoring.

The remaining parts of this paper are structured as follows: Section 2 details the core functionality of the corresponding R package **reservr**. It is split into definition of samples $\mathfrak{I}$ (Section 2.1), definition of distribution families (Section 2.2), mathematical definitions of some available distribution families (Section 2.3), estimation of distribution parameters (Section 2.4) and distributional regression using **tensorflow** (Section 2.5). A conclusion is given in Section 3.

2. Usage of reservr

The package serves two main goals: fitting distributions to randomly truncated non-informatively interval censored data and performing (deep) distributional regression with randomly truncated non-informatively interval censored data. Four main components are integrated with each other to facilitate the analysis goals

1. Methods for representing a randomly truncated non-informatively interval censored sample $\mathfrak{I}$.
2. Methods for specifying a parametrized distribution family $\mathcal{F} = \{F_\theta | \theta \in \Theta\}$ to be fitted.

3. Methods for estimating distribution parameters θ given a sample $\mathfrak{I}$.
4. Methods for regression of distribution parameters given a regression sample $\mathfrak{I}_{\text{reg}}$, a parametrized family $\mathcal{F}$ and a general **tensorflow** network $\mathcal{G} : \mathfrak{X} \to \Theta$ that processes X to estimate the conditional distribution of $Y|X = x$ by $F_{g(x)}$ with $g \in \mathcal{G}$.

Each of these components is described one by one in the following sections.

2.1. Working with samples

A sample $\mathfrak{I} = \{(m, v, l, u, w)_i\}$ is represented as a tibble (from package **tibble**). The core function to create this tibble is `trunc_obs()`. A tibble created by `trunc_obs()` consists of five columns:

- `x`: If observed, the exact value of the random variable, referred to as Y in Section 1. Otherwise `NA`.
- `xmin`: Lower interval censoring bound (M in Section 1) for the observation. If the observation is not censored, `xmin` is equal to `x`.
- `xmax`: Upper interval censoring bound (V in Section 1) for the observation. If the observation is not censored, `xmax` is equal to `x`.
- `tmin`: Lower truncation bound (L in Section 1). Only observations with `x` $\geq$ `tmin` are observed. Can be $-\infty$ to indicate no lower truncation.
- `tmax`: Upper truncation bound (U in Section 1). Only observations with `x` $\leq$ `tmax` are observed. Can be ∞ to indicate no upper truncation.
- `w`: The weight associated with the observation. Defaults to 1.

Note that, unlike in Section 1, the lower bounds of intervals in `trunc_obs` are included, that is, we allow for `x` $\geq$ `tmin` rather than `x` $>$ `tmin`, and that the unknown variable of interest is called `x` instead of Y. For continuous random variables, the formulas are equivalent to the half-open formulation. For discrete random variables, `xmin` and `tmin` may have to be appropriately shifted, e.g., by replacing `xmin` by `xmin` $- 0.5$ for integer valued variables. The following code defines a sample of size 1 without truncation and censoring, with the realized value of 1.3.

```
R> trunc_obs(1.3)

    x xmin xmax tmin tmax w
1 1.3  1.3  1.3 -Inf  Inf 1
```

Simulating randomly truncated and interval censored data from a standard normal distribution with 80% of the observations randomly interval censored and random uniform truncation $L \sim \text{Unif}[-2, 0]$ and $U \sim \text{Unif}[0, 2]$ can be simulated as follows

```
R> set.seed(123)
R> N <- 1000L
R> x <- rnorm(N)
R> is_censored <- rbinom(N, size = 1L, prob = 0.8) == 1L
```

```
R>
R> c_lower <- runif(sum(is_censored), min = -2.0, max = 0.0)
R> c_upper <- c_lower + runif(sum(is_censored), min = 0, max = 1.0)
R>
R> x_lower <- x
R> x_upper <- x
R>
R> x_lower[is_censored] <- dplyr::case_when(
+   x[is_censored] <= c_lower ~ -Inf,
+   x[is_censored] <= c_upper ~ c_lower,
+   TRUE ~ c_upper
+ )
R> x_upper[is_censored] <- dplyr::case_when(
+   x[is_censored] <= c_lower ~ c_lower,
+   x[is_censored] <= c_upper ~ c_upper,
+   TRUE ~ Inf
+ )
R>
R> t_lower <- runif(N, min = -2.0, max = 0.0)
R> t_upper <- runif(N, min = 0.0, max = 2.0)
R>
R> is_observed <- t_lower <= x & x <= t_upper
R>
R> obs <- trunc_obs(
+   xmin = pmax(x_lower, t_lower)[is_observed],
+   xmax = pmin(x_upper, t_upper)[is_observed],
+   tmin = t_lower[is_observed],
+   tmax = t_upper[is_observed]
+ )
```

Observations look like:

```
R> obs[8L:12L, ]
```

```
# A tibble: 5 x 6
       x    xmin   xmax   tmin  tmax     w
   <dbl>   <dbl>  <dbl>  <dbl> <dbl> <dbl>
1 NA     -0.479   1.15  -1.93  1.15      1
2 NA     -0.177   1.79  -0.210 1.79      1
3 -0.556 -0.556  -0.556 -0.957 0.791     1
4 NA     -0.379   0.616 -0.379 0.616     1
5 NA      0.0575  1.45  -0.437 1.45      1
```

The total number of observations is smaller than the base population of 1000 due to truncation:

```
R> nrow(obs)
```

```
[1] 623
```

The total number of censored observations is roughly $0.8 \cdot$ `nrow(obs)`.

```
R> sum(is.na(obs$x))
```

```
[1] 496
```

In addition to the `trunc_obs()` constructor function, there are functions `as_trunc_obs()` for coercion, `truncate_obs()` for artificially changing truncation bounds, and `repdel_obs()` for computing randomly truncated reporting delay observations from general insurance claims data containing accident date, reporting delay and evaluation date information. The latter takes inputs of the form $(T_{\text{acc}}, D, \tau)$ where $T_{\text{acc}} < \tau$ are accident dates with corresponding reporting delays $D \geq 0$ and τ is the calendar date of observation. It returns the sample ($\texttt{xmin} = \texttt{xmax} = D, \texttt{tmin} = 0, \texttt{tmax} = \tau - T_{\text{acc}}, \texttt{w} = 1$) suitable for estimating the reporting delay distribution where a claim is only observed if it has been reported by the evaluation date, i.e., $T_{\text{acc}} + D \leq \tau$. Such an analysis was performed using **reservr** in Bücher and Rosenstock (2022a, 2023).

2.2. Definition of distribution families

Distribution families are implemented using the **R6** class system (Chang 2021). They inherit from the class `Distribution` and feature a common interface to

- manage fixed and free parameters of the underlying familiy,
- use basic distribution functions for random number generation and computation of the density, cumulative distribution, hazard and quantile function,
- use additional functions supporting parameter estimation procedures such as computing support or presence of a point mass,
- compile performance enhanced functions to speed up basic functions for repeated evaluation,
- provide **tensorflow**-specific implementations to support (deep) distributional regression.

A `Distribution` object represents a distribution family $\mathcal{F}$ supported on a subset of the real line and parameterized by a fixed finite-dimensional parameter space Θ. The family may be a singleton, in which case it is rather a distribution than a distribution family.

reservr provides a set of basic distribution families, optionally with some fixed parameters, as well as transformations of distribution families that take one or more underlying distribution families. At the time of writing, these are:

Generator function	Description
`dist_bdegp(n, m, u, epsilon)`	A Blended Dirac Erlang Generalized Pareto distribution family, see Section 2.3.4
`dist_beta(shape1, shape2, ncp)`	A (non-central) Beta distribution family

Generator function	Description
`dist_binomial(size, prob)`	A Binomial distribution family
`dist_dirac(point)`	A Dirac distribution family with full mass at `point`
`dist_discrete(size, probs)`	A discrete distribution family with fixed support $\{1, \dots, \text{size}\}$ and $P(X = k) = \text{probs}_k$
`dist_erlangmix(shapes, scale, probs)`	An Erlang mixture distribution family, see Section 2.3.2
`dist_exponential(rate)`	An Exponential distribution family
`dist_gamma(shape, rate)`	A Gamma distribution family
`dist_genpareto(u, sigmau, xi)`	A Generalized Pareto Distribution family
`dist_genpareto1(u, sigmau, xi)`	A Generalized Pareto Distribution family with the tail index ξ constrained to $(0, 1)$
`dist_lognormal(meanlog, sdlog)`	A Log-Normaldistribution family
`dist_negbinomial(size, mu)`	A negative Binomial distribution family
`dist_normal(mean, sd)`	A Normal distribution family
`dist_pareto(shape, scale)`	A Pareto Type I distribution family, i.e., a Generalized Pareto distribution family with $\mathtt{u} = 0$
`dist_poisson(lambda)`	A Poisson distribution family
`dist_uniform(min, max)`	A uniform distribution family
`dist_weibull(shape, scale)`	A Weibull distribution family

Transformation function	Description
`dist_blended(dists, probs, breaks, bandwidths)`	A Blended mixture distribution family, see Section 2.3.3
`dist_mixture(dists, probs)`	A general Mixture distribution family, see Section 2.3.1
`dist_translate(dist, offset, multiplier)`	The affine transformation family consisting of all distributions of $\mathtt{multiplier} \cdot X + \mathtt{offset}$ with $X \sim F \in \mathtt{dist}$
`dist_trunc(dist, min, max)`	The truncated distribution family consisting of all distributions of $X \vert (\mathtt{min} \leq X \leq \mathtt{max})$ with $X \sim F \in \mathtt{dist}$

Parameters

Parameters of distribution families can either be fixed to a constant value, or free. Free parameters (*placeholders*) are those that should be estimated from data whereas fixed parameters are held constant. Most `Distribution` methods have an argument `with_params` to provide values for the free parameters and need fully specified parameters to work. For example, generating samples from a distribution is only possible if it is fully parameterized using fixed parameters and the `with_params` argument of `Distribution$sample()`.

```
R> dist <- dist_normal(sd = 1.0)
```

We have now defined `dist` to be a normal distribution family with standard deviation 1 and free mean. Since not all parameters required for a normal distribution are fixed, `dist$sample()` will error if not provided with a `mean` parameter.

```
R> dist$sample(1L)

Error in (function (n, mean = 0, sd = 1) : invalid arguments
```

The `with_params` argument can be used both to provide free parameters and to override fixed parameters, if necessary.

```
R> set.seed(10L)
R> dist$sample(1L, with_params = list(mean = 0.0))

[1] 0.01874617

R> set.seed(10L)
R> dist$sample(1L, with_params = list(mean = 0.0, sd = 2.0))

[1] 0.03749234
```

The two observations were drawn from a standard normal and a normal distribution with mean zero and standard deviation 2, respectively. Since the chosen seed was identical, the second sample is exactly double the first sample. Whenever the output length is greater than one, such as when taking more than one sample, `with_params` can optionally contain individual parameters for each entry.

```
R> set.seed(10L)
R> dist$sample(3L, with_params = list(mean = 0.0:2.0, sd = 0.5))

[1] 0.009373085 0.907873729 1.314334725
```

The three observations were drawn from $\mathcal{N}(\mu = 0, \sigma = 0.5)$, $\mathcal{N}(\mu = 1, \sigma = 0.5)$ and $\mathcal{N}(\mu = 2, \sigma = 0.5)$, respectively.

`Distribution`s have a set of fields and methods related to managing parameters:

- The active binding `default_params` gets or sets the list of all parameters and their fixed values, `NULL` represents a free parameter. Component families are included as `Distribution` objects.
- `get_params()` gets the list of all parameters and their fixed values, traversing component distribution families.
- `get_placeholders()` gets the list of free parameters with `NULL` as values.
- The active binding `param_bounds` gets or sets the domain of all regular family parameters as an `Interval` object. Setting a bound via the `param_bounds` active binding allows restricting the natural parameter space of a family.

- `get_param_bounds()` returns the bounds of all free parameters as a list of `Intervals`, traversing component distribution families.
- `get_param_constraint()` returns `NULL` or a function that evaluates constraints on the parameter set. The function must return a vector of constraint values (that need to be equal to 0 for valid parameters) or a list with elements `constraints` and `jacobian`. When returning a list, the `jacobian` element should contain the jacobian of the constraint function. Used in `nloptr::slsqp(heq=)` for estimation. An example is that mixture families require the `probs` parameters to sum to 1 in addition to the box constraint that each parameter is in $[0, 1]$. Note that box constraints are handled by `param_bounds` and need not be specified as a constraint function.
- `get_components()` returns a list of component families for transformations or mixtures. The list is empty for basic families.

Here is an example for a normal family with fixed standard deviation $\sigma = 1$ and a mixture distribution family with two components, one of which is specified as a normal distribution family:

```
R> dist <- dist_normal(sd = 1.0)
R> mix <- dist_mixture(dists = list(dist_normal(), NULL))
R>
R> dist$default_params

$mean
NULL

$sd
[1] 1

R> mix$default_params

$dists
$dists[[1]]
A NormalDistribution with 2 dof

$dists[[2]]
NULL


$probs
$probs[[1]]
NULL

$probs[[2]]
NULL

R> str(dist$get_placeholders())
```

```
List of 1
 $ mean: NULL
```

```
R> str(mix$get_placeholders())
```

```
List of 2
 $ dists:List of 2
  ..$ :List of 2
  .. ..$ mean: NULL
  .. ..$ sd  : NULL
  ..$ : NULL
 $ probs:List of 2
  ..$ : NULL
  ..$ : NULL
```

```
R> str(dist$param_bounds)
```

```
List of 2
 $ mean:Classes 'Interval', 'R6' (-Inf, Inf)
 $ sd  :Classes 'Interval', 'R6' (0, Inf)
```

```
R> str(mix$param_bounds)
```

```
List of 2
 $ dists:List of 1
  ..$ : NULL
 $ probs:List of 1
  ..$ :Classes 'Interval', 'R6' [0, 1]
```

```
R> str(dist$get_param_bounds())
```

```
List of 1
 $ mean:Classes 'Interval', 'R6' (-Inf, Inf)
```

```
R> str(mix$get_param_bounds())
```

```
List of 2
 $ dists:List of 1
  ..$ :List of 2
  .. ..$ mean:Classes 'Interval', 'R6' (-Inf, Inf)
  .. ..$ sd  :Classes 'Interval', 'R6' (0, Inf)
 $ probs:List of 2
  ..$ :Classes 'Interval', 'R6' [0, 1]
  ..$ :Classes 'Interval', 'R6' [0, 1]
```

```
R> str(dist$get_param_constraints())
```

```
 NULL

R> str(mix$get_param_constraints())

function (params)

R> dist$get_components()

list()

R> mix$get_components()

[[1]]
A NormalDistribution with 2 dof

[[2]]
NULL
```

Basic distribution functions

The basic distribution functions (density, probability, hazard and quantile function, as well as random number generation) are provided by each distribution family. In general, the argument `with_params` can be used to both specify missing parameters (placeholders) and to override fixed distribution parameters. If the provided parameters are vectors of length greater than 1, they must conform to the input dimension (e.g. `length(x)` for `density`). In this case, the parameters are "vectorized" in the sense that the ith output element will be computed using the ith entry from the parameter list.

- `density(x, log = FALSE, with_params = list())` computes the (log-)density.
- `probability(q, lower.tail = TRUE, log.p = FALSE, with_params = list()` computes the (log-)cumulative distribution function or (log-)survival function.
- `hazard(x, log = FALSE. with_params = list())` computes the (log-)hazard function.
- `quantile(p, lower.tail = TRUE, log.p = FALSE, with_params = list())` computes upper or lower quantiles.
- `sample(n, with_params = list())` generates a random sample of size `n`. (`with_params` can contain length `n` vectors in this case).

Additional functions

In addition to the basic functions, there are several supporting functions useful for, e.g., estimation of parameters.

- `export_functions(name, with_params = list())` exports `{d,p,q,r}<name>` functions adhering to the common R convention for distribution functions.

- `get_type()` returns one of `"continuous"`, `"discrete"`, or `"mixed"` depending on whether the distribution family has a density with respect to the Lebesgue measure, the counting measure, or the sum of the Lebesgue measure with one or many point measures.
- `is_continuous()` and `is_discrete()` testing for the particular type.
- `has_capability(caps)` gives information on whether a specific implementation provides some or all of the features described. Possible capabilities are `"sample"`, `"density"`, `"probability"`, `"quantile"`, `"diff_density"`, `"diff_probability"`, `"tf_logdensity"`, `"tf_logprobability"`.
- `require_capability(caps)` errors if the specified capabilities are not implemented for the family at hand.
- `is_discrete_at(x, with_params = list())` returns a logical vector indicating whether the distribution has a point mass at `x`.
- `is_in_support(x, with_params = list())` returns a logical vector indicating whether the distribution has any mass at `x`.

Performance enhancements

When working with larger data or many calls to distribution functions, such as when performing a fit, it can be beneficial to just-in-time compile specialized functions that avoid overhead for dealing with the generic structure of distributions and their parametrization. `Distribution`s offer a set of "compiler" functions that return simplified, faster, versions of the basic distribution functions, or that analytically compute gradients. Those functions are not necessarily implemented for all `Distribution` classes, but will be automatically used by, e.g., `fit_dist()` if useful. The input structure for `param_matrix` can be obtained by `flatten_params_matrix(dist$get_placeholders())` where `dist` is the `Distribution` object in question.

- `compile_density()` compiles a fast function with signature `(x, param_matrix, log = FALSE)` that will compute the density with fixed parameters hard-coded and taking the free parameters as a matrix with defined layout instead of a nested list.
- `compile_probability()` compiles a fast replacement for `probability` with signature `(q, param_matrix, lower.tail = TRUE, log.p = FALSE)`.
- `compile_probability_interval()` compiles a fast function with signature `(qmin, qmax, param_matrix, log.p = FALSE)` computing $P(X \in [\texttt{qmin}, \texttt{qmax}])$ or its logarithm efficiently. This expression is necessary for computing truncation probabilities.
- `compile_sample()` compiles a fast replacement for `sample` with signature `(n, param_matrix)`.
- `diff_density(x, log = FALSE, with_params = list())` computes the (log-)gradients of the density function with respect to free distribution family parameters, useful for maximum likelihood estimation.
- `diff_probability(q, lower.tail = TRUE, log.p = FALSE, with_params = list())` computes the (log-)gradients of the cumulative density function with respect to free distribution family parameters. This is useful for conditional maximum likelihood estimation in the presence of random truncation or non-informative interval censoring.

```
R> dist <- dist_normal()
R> flatten_params_matrix(dist$get_placeholders())
```

```
     mean sd
[1,]   NA NA

R> denscmp <- dist$compile_density()
R>
R> if (requireNamespace("bench", quietly = TRUE)) {
+   bench::mark(
+     dist$density(-2:2, with_params = list(mean = 0.0, sd = 1.0)),
+     denscmp(-2:2, matrix(c(0.0, 1.0), nrow = 5L, ncol = 2L, byrow = TRUE)),
+     dnorm(-2:2, mean = rep(0.0, 5L), sd = rep(1.0, 5L))
+   )
+ }

# A tibble: 3 x 6
  expression                              min  median `itr/sec` mem_alloc `gc/sec`
  <bch:expr>                            <bch:t> <bch:t>     <dbl> <bch:byt>    <dbl>
1 dist$density(-2:2, with_params =~ 18.94us 21.41us    45213.        0B     49.8
2 denscmp(-2:2, matrix(c(0, 1), nr~  3.16us  3.89us   250359.        0B     50.1
3 dnorm(-2:2, mean = rep(0, 5L), s~  1.08us  1.33us   698335.    2.58KB    140.
```

tensorflow *interface*

Use of distribution families from within **tensorflow** networks requires specialized implementations using the **tensorflow** APIs instead of regular R functions. These are tailored to the needs of maximizing (conditional) likelihoods of weighted, censored and randomly truncated data. Details on working with **tensorflow** can be found in Section 2.5.

- `tf_compile_params(input, name_prefix = "")` creates **keras** layers that take an `input` layer and transform it into a valid parametrization of the distribution family.
- `tf_is_discrete_at()` returns a **tensorflow**-ready version of `is_discrete_at()`.
- `tf_logdensity()` returns a **tensorflow**-ready version of `compile_density()` with implied `log = TRUE`.
- `tf_logprobability()` returns a **tensorflow**-ready version pf `compile_probability_interval()` with implied `log.p = TRUE`.
- `tf_make_constants()` creates a list of constant tensors for all fixed distribution family parameters.

2.3. Special families

Some of the distribution families available in **reservr** have tailored algorithms for parameter estimation, or are not commonly known. This section contains mathematical definitions of those function families.

Mixture distribution families

A mixture distribution family is defined by a fixed number k of component families $\{\mathcal{F}_i\}_{i=1}^k$ via the set of distributions

$$\text{Mixture}(\mathcal{F}_1, \ldots, \mathcal{F}_k) := \left\{ F = \sum_{i=1}^{k} p_i F_i \;\middle|\; F_i \in \mathcal{F}_i, p_i \in [0,1], \sum_{i=1}^{k} p_i = 1 \right\}.$$

Erlang mixture distribution families

An Erlang mixture distribution family is defined by its number of components k as a mixture of Erlang distributions (Gamma distributions with integer shape parameter) with common scale parameter. If $\Gamma_{\alpha,\theta}$ denotes a Gamma distribution with shape α and scale θ, the erlang mixture family with k components can be defined as follows:

$$\text{ErlangMixture}(k) := \left\{ F = \sum_{i=1}^{k} p_i \Gamma_{\alpha_i,\theta} \;\middle|\; \alpha_i \in \mathbb{N}, \theta \in (0,\infty), p_i \in [0,1], \sum_{i=1}^{k} p_i = 1 \right\}.$$

Note that for $k \to \infty$, Erlang mixtures are dense in the space of distributions on $(0,\infty)$ with respect to weak convergence (Lee and Lin 2012), making them a useful modeling choice for general positive continuous distributions. However, the tail index of all Erlang mixture distributions is always zero due to the exponential decay of Gamma densities.

Blended distribution families

A Blended distribution is defined in Bücher and Rosenstock (2022a) as follows: Given two underlying distributions P, Q on $\mathbb{R}$ with cdfs $F(\cdot) = P((-\infty, \cdot])$ and $G(\cdot) = Q((-\infty, \cdot])$, respectively, and parameters $\kappa \in \mathbb{R}, \varepsilon \in (0,\infty), p_1, p_2 \in [0,1], p_1 + p_2 = 1$ such that $F(\kappa) > 0$ and $G(\kappa) < 1$, we define the *Blended Distribution* $B = \text{Blended}(P, Q; p, \kappa, \varepsilon)$ of P and Q with blending interval $[\kappa - \varepsilon, \kappa + \varepsilon]$ and mixture probabilities p via its cdf F_B:

$$p_{\kappa,\varepsilon}(x) = \begin{cases} x & , x \in (-\infty, \kappa - \varepsilon], \\ \frac{1}{2}(x + \kappa - \varepsilon) + \frac{\varepsilon}{\pi} \cos\left(\frac{\pi(x-\kappa)}{2\varepsilon}\right) & , x \in (\kappa - \varepsilon, \kappa + \varepsilon], \\ \kappa & , x \in (\kappa + \varepsilon, \infty), \end{cases}$$

$$q_{\kappa,\varepsilon}(x) = \begin{cases} \kappa & , x \in (-\infty, \kappa - \varepsilon], \\ \frac{1}{2}(x + \kappa + \varepsilon) - \frac{\varepsilon}{\pi} \cos\left(\frac{\pi(x-\kappa)}{2\varepsilon}\right) & , x \in (\kappa - \varepsilon, \kappa + \varepsilon], \\ x & , x \in (\kappa + \varepsilon, \infty), \end{cases}$$

$$F_B(x) = p_1 \frac{F(p_{\kappa,\varepsilon}(x))}{F(\kappa)} + p_2 \frac{G(q_{\kappa,\varepsilon}(x)) - G(\kappa)}{1 - G(\kappa)}.$$

The following illustration shows the components of a Blended($\mathcal{N}(\mu = -1, \sigma = 1)$, Exp($\lambda = 1$); $p = (0.5, 0.5), \kappa = 0, \varepsilon = 1$) distribution.

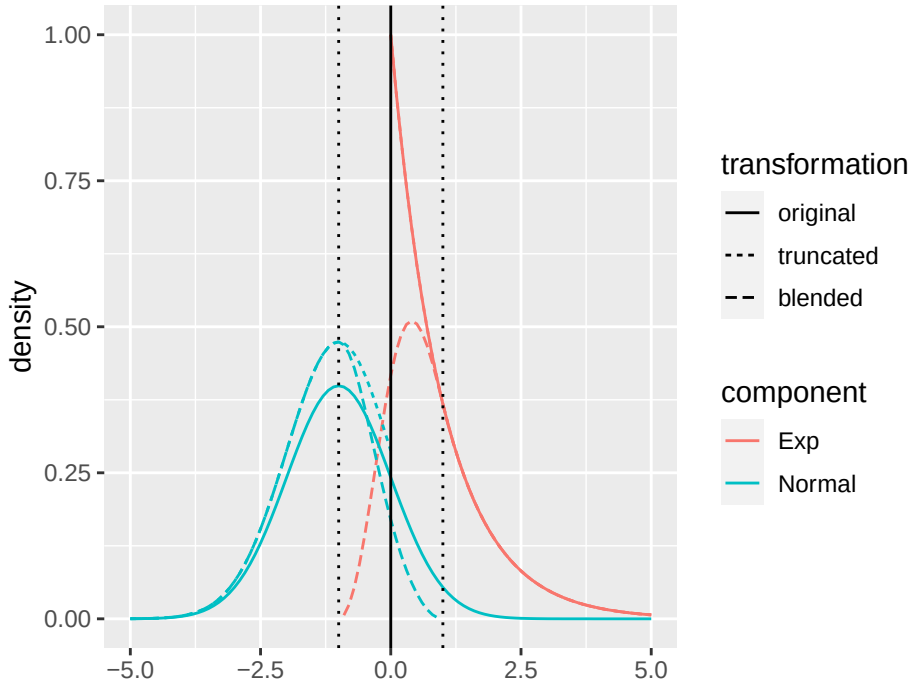
```
R> dist1 <- dist_normal(mean = -1.0, sd = 1.0)
R> dist2 <- dist_exponential(rate = 1.0)
R> distb <- dist_blended(
```

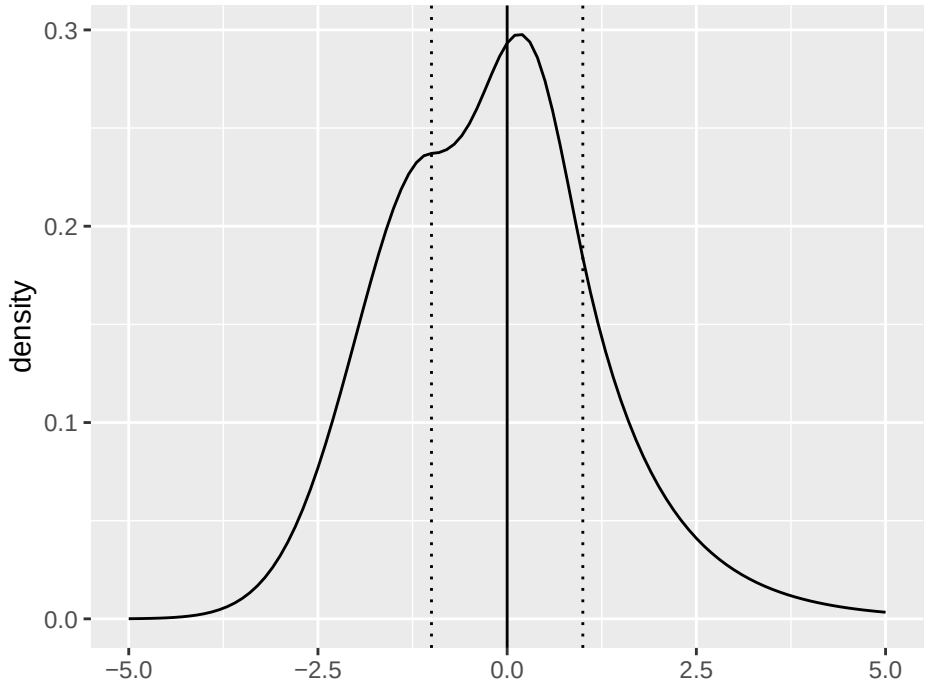
```
+   dists = list(dist1, dist2),
+   breaks = list(0.0),
+   bandwidths = list(1.0),
+   probs = list(0.5, 0.5)
+ )
```

The transformation of the original component distributions ($\mathcal{N}$ and Exp) can be illustrated by first right- and left-truncating at $\kappa = 0$ respectively, and then applying the blending transformations $p_{\kappa,\varepsilon}$ and $q_{\kappa,\varepsilon}$. The latter distributions can be obtained in **reservr** by setting the probability weights of the blended distribution to $p = (1, 0)$ and $p = (0, 1)$ respectively. Intermediate truncated distributions are obtained via `trunc_dist()`, with κ as upper or lower bound respectively.

```
R> distt1 <- dist_trunc(dist1, min = -Inf, max = 0.0)
R> distt2 <- dist_trunc(dist2, min = 0.0, max = Inf)
R>
R> distb1 <- distb$clone()
R> distb1$default_params$probs <- list(1.0, 0.0)
R> distb2 <- distb$clone()
R> distb2$default_params$probs <- list(0.0, 1.0)
```

We show the resulting density at each of the steps, and the final blended density obtained by weighting the blended component densities.





The definition of a blended distribution leads to the definition of a blended distribution family by allowing P, Q, κ and ε to vary:

Given two families $\mathcal{F}, \mathcal{G}$ of distributions on $\mathbb{R}$, and parameters $\kappa \in \mathbb{R}, \varepsilon \in (0, \infty)$, we define the *Blended Distribution family* as the family of Distributions

$$\text{Blended}(\mathcal{F}, \mathcal{G}; \kappa, \varepsilon) := \{\text{Blended}(P, Q; p, \kappa, \varepsilon) \mid P \in \mathcal{F}, Q \in \mathcal{G}, p_1, p_2 \in [0, 1], p_1 + p_2 = 1\}.$$

Blended distribution families can be generalized to a number of components k by letting κ and ε become vectors of dimension $k-1$ such that $\kappa_i + \varepsilon_i \leq \kappa_{i+1} - \varepsilon_{i+1}$ for $i = 1, \ldots, k-2$. Compared to piecewise distribution families obtained by mixture of truncated distribution families with supports $(-\infty, \kappa]$ and $[\kappa, \infty)$ such as those commonly used for extreme value modelling, blended distribution families exhibit a continuous density within the blending region $(\kappa - \varepsilon, \kappa + \varepsilon)$.

reservr provides an implementation via `dist_blended()`, with limited support for more than two component families.

The Blended Dirac Erlang Generalized Pareto distribution family

Using the construction of a Blended distribution family, we can define the Blended Dirac Erlang Generalized Pareto (BDEGP) family as follows, see Bücher and Rosenstock (2022a, Definition 2).

Given parameters $n \in \mathbb{N}, m \in \mathbb{N}, \kappa \in \mathbb{R}$ and $\varepsilon \in (0, \infty)$, we define the *Blended Dirac Erlang Generalized Pareto family* as the family of distributions

$$
\begin{aligned}
\mathrm{BDEGP}(n, m, \kappa, \varepsilon) := \mathrm{Mixture}(&\\
&\{\delta_0\}, \{\delta_1\}, \ldots, \{\delta_{n-1}\},\\
&\mathrm{Blended}(\\
&\quad \mathrm{ErlangMixture}(m),\\
&\quad \{\mathrm{GPD}(\kappa, \sigma, \xi) \mid \sigma \in (0, \infty), \xi \in [0, 1)\};\\
&\quad \kappa, \varepsilon\\
&)\\
),&
\end{aligned}
$$

where δ_k is the dirac distribution at k and GPD is the generalized Pareto distribution. Note the constraint on the tail index $\xi \in [0, 1)$, guaranteeing finite expectation.

This distribution family has three features making it useful in modelling very general heavy-tailed distributions on $(0, \infty)$:

1. A maximally flexible lower tail
2. A flexible family of distributions for its body
3. A flexible tail index due to the generalized Pareto component

2.4. Methods of estimating distribution parameters

This section describes the functions for the problem of estimating a parameter $\theta \in \Theta$ given a sample $\mathfrak{I}$ and a parameterized family $\mathcal{F} = \{F_\theta \mid \theta \in \Theta\}$. Sometimes, the conditional log-likelihood in (2) can be directly maximized, yielding an estimate for θ. This is the default behavior in **reservr** if no specialized estimation routine for the provided family $\mathcal{F}_\theta$ is defined. Depending on whether there are box constraints, nonlinear constraints or no constraints on the parameter space Θ, different implementations of nonlinear optimization algorithms from **nloptr** (Johnson 2007), in particular truncated Newton (Dembo and Steihaug 1983) for unconstrained families, L-BFGS (Liu and Nocedal 1989) for box-constrained families and SLSQP (Kraft 1994) for general constrained families are employed.

In addition to the naive direct optimization approach, some families lend themselves to specialized estimation algorithms which usually show faster convergence due to making use of special structures in the parameter space Θ.

Estimating distribution parameters from truncated observations is handled using the generic `fit()` method. It delegates to `fit_dist()`, which is also generic with signature:

- `dist`: The distribution family to be fit
- `obs`: The `trunc_obs` object, or a vector of observed values
- `start`: Starting parameters, as a list compatible with `dist$get_placeholders()`.

At the time of writing there are specialized algorithms for six types of families:

1. Blended distribution families (Bücher and Rosenstock 2022a, Algorithm 2)

2. Erlang mixture distribution families (Bücher and Rosenstock 2022b, Algorithm S.2)
3. Generalized pareto distribution families with free lower bound `u` (estimated by the minimum of `xmin` over the sample)
4. Mixture distribution families (Bücher and Rosenstock 2022a, Algorithm 1)
5. Translated distribution families with fixed `offset` and `multiplier` (transform the sample via $\frac{\cdot - \text{offset}}{\text{multiplier}}$ and fit the component distribution family to the transformed sample)
6. Uniform distribution families with free lower bound `min` or upper bound `max` (estimated by the minimum of `xmin`, for `min`, and the maximum of `xmax`, for `max`, over the sample)

If not present, the `start` parameter is obtained via the `fit_dist_start()` generic. This generic implements a family specific method of generating valid starting values for all placeholder parameters. A notable implementation is `fit_dist_start.ErlangMixtureDistribution()` for Erlang mixture distribution families. If the shape parameters are free, there are different initialization strategies that can be chosen using additional arguments to `fit_dist_start()`:

- `init = "shapes"` paired with `shapes = c(...)` manually specifies starting shape parameters α
- `init = "fan"` paired with `spread = d` uses $\alpha = (1, 1 + d, \ldots, 1 + (k-1) \cdot d)$ with a default of $d = 1$ resulting in $\alpha = (1, \ldots, k)$
- `init = "kmeans"` uses 1-dimensional K-means based clustering of the sample observations such that each cluster corresponds to a unique shape
- `init = "cmm"` uses the centralized method of moments procedure described in Gui, Huang, and Lin (2018)

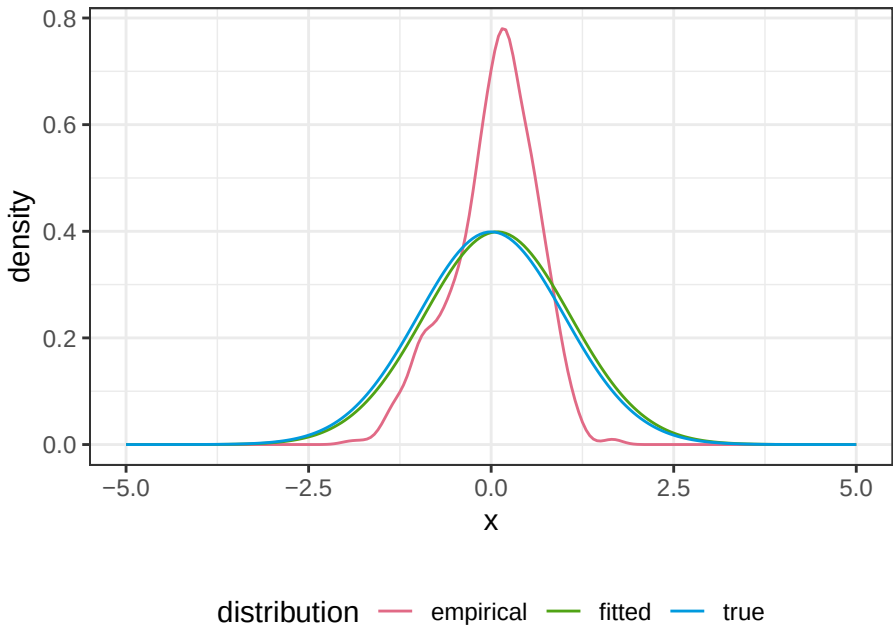
Re-using `dist <- dist_normal(sd = 1.0)` from above and the generated sample `obs`, we can fit the free parameter `mean`:

```
R> dist <- dist_normal(sd = 1.0)
R> the_fit <- fit(dist, obs)
R> str(the_fit)

List of 3
 $ params:List of 1
  ..$ mean: num 0.0822
 $ opt   :List of 5
  ..$ par        : Named num 0.0822
  .. ..- attr(*, "names")= chr "mean"
  ..$ value      : num 341
  ..$ iter       : int 7
  ..$ convergence: int 1
  ..$ message    : chr "NLOPT_SUCCESS: Generic success return value."
 $ logLik:Class 'logLik' : -341 (df=1)
```

Using the function `plot_distributions()` we can also assess the quality of the fit.

```
R> plot_distributions(
+   true = dist,
```

```
+   fitted = dist,
+   empirical = dist_empirical(0.5 * (obs$xmin + obs$xmax)),
+   .x = seq(-5, 5, length.out = 201),
+   plots = "density",
+   with_params = list(
+     true = list(mean = 0.0, sd = 1.0),
+     fitted = the_fit$params
+   )
+ )
```



Here, the density labelled `empirical` corresponds to a kernel density estimate with automatic bandwidth selection.

We follow with an example of fitting an ErlangMixture(3) distribution family using various initialization strategies. Note that both, `"kmeans"` and `"cmm"` use the random number generator for internal K-means clustering. This necessitates setting a constant seed before running `fit_dist_start()` and `fit()` to ensure the chosen starting parameters are the same for both calls.

```
R> dist <- dist_erlangmix(list(NULL, NULL, NULL))
R> params <- list(
+   shapes = list(1L, 4L, 12L),
+   scale = 2.0,
+   probs = list(0.5, 0.3, 0.2)
+ )
R>
R> set.seed(1234)
```

```
R> x <- dist$sample(100L, with_params = params)
R>
R> set.seed(32)
R> init_true <- fit_dist_start(dist, x, init = "shapes",
+                              shapes = as.numeric(params$shapes))
R> init_fan <- fit_dist_start(dist, x, init = "fan", spread = 3L)
R> init_kmeans <- fit_dist_start(dist, x, init = "kmeans")
R> init_cmm <- fit_dist_start(dist, x, init = "cmm")
R> rbind(
+   flatten_params(init_true),
+   flatten_params(init_fan),
+   flatten_params(init_kmeans),
+   flatten_params(init_cmm)
+ )

     shapes[1] shapes[2] shapes[3]    scale probs[1] probs[2] probs[3]
[1,]         1         4        12 1.590800     0.43     0.33     0.24
[2,]         1         4         7 2.688103     0.55     0.32     0.13
[3,]         1         5        13 1.484960     0.43     0.36     0.21
[4,]         2        10        24 1.010531     0.56     0.27     0.17

R> set.seed(32)
R> str(fit(dist, x, init = "shapes", shapes = as.numeric(params$shapes)))

List of 4
 $ params     :List of 3
  ..$ probs :List of 3
  .. ..$ : num 0.43
  .. ..$ : num 0.33
  .. ..$ : num 0.24
  ..$ shapes:List of 3
  .. ..$ : num 1
  .. ..$ : num 4
  .. ..$ : num 13
  ..$ scale : num 1.59
 $ params_hist: list()
 $ iter       : int 1
 $ logLik     :Class 'logLik' : -290 (df=6)

R> fit(dist, x, init = "fan", spread = 3L)$logLik

'log Lik.' -292.0026 (df=6)

R> fit(dist, x, init = "kmeans")$logLik

'log Lik.' -289.2834 (df=6)
```

```
R> fit(dist, x, init = "cmm")$logLik

'log Lik.' -293.1273 (df=6)
```

It should be noted that the different initialization methods had a considerable impact on the outcome in the example due to the discrete nature of Erlang mixture distribution shape parameters and thus the combinatorial difficulty of picking optimal shapes α. The `fit()` result for Erlang mixture distribution families contains an element named `"params_hist"`. This can be populated by passing `trace = TRUE` to `fit()` and will record parameters after all ECME steps in the ECME-based estimation algorithms from Bücher and Rosenstock (2022a, Algorithms 1 and 2) and Bücher and Rosenstock (2022b, Algorithm S.2). The element `"iter"` contains the number of full ECME-Iterations that were performed.

2.5. Distributional regression using tensorflow integration

The maximization problem (3) is delegated to **tensorflow**, which supplies ample stochastic optimization algorithms. Functions in **reservr** are necessary to create a suitable output layer for **tensorflow** that maps onto Θ and to provide an implementation of the (negative) log-likelihood in (3) as a loss function. These two tasks are combined in `tf_compile_model()`. The function returns an object of class `reservr_keras_model`, which can be used for the estimation procedure.

Given input layers `inputs` and an intermediate output layer `intermediate_output` as well as a family of distributions `dist`, the function

- Compiles the loss for `dist` defined by (3) as $l(g) = -\frac{1}{\#(\mathfrak{I}_{\text{reg}})}\ell(g|\mathfrak{I}_{\text{reg}})$, optionally disabling `censoring` or `truncation` for efficiency.
- Creates a list of final output layers mapping `intermediate_output` onto the parameter space Θ of `dist` using `Distribution$tf_compile_params()`. This step adds additional degrees of freedom to the overall model, and the approach is described in Bücher and Rosenstock (2022b, Section A)
- Runs `keras::compile()` on the underlying `keras.engine.training.Model`.

The following example defines a linear model with homoskedasticity assumption and fits it using 100 iterations of the Adam optimization algorithm (Kingma and Ba 2015). First, we simulate data (Y, X) from the model defined by $X \sim \text{Unif}(10, 20)$ and $Y|X = x \sim \mathcal{N}(\mu = 2x, \sigma = 1)$.

```
R> set.seed(1431L)
R> tensorflow::set_random_seed(1432L)
R>
R> dataset <- tibble::tibble(
+   x = runif(100, min = 10, max = 20),
+   y = 2 * x + rnorm(100)
+ )
```

Next, we specify the distribution family $\mathcal{F} = \{\mathcal{N}(\mu, \sigma = 1)|\mu \in \mathbb{R}\}$, incorporating the homoskedasticity assumption.

```
R> dist <- dist_normal(sd = 1.0)
```

Using **keras**, we define an empty neural network, just taking x as an input and performing no transformation.

```
R> nnet_input <- keras::layer_input(shape = 1L, name = "x_input")
R> nnet_output <- nnet_input
```

Then, `tf_compile_model()` adapts the input layer to the free parameter space $\Theta = \mathbb{R}$. This introduces two parameters to the function family $\mathcal{G}$ and implies the functional relationship $\mu = g(x) := \theta_1 \cdot x + \theta_0$. Since our sample is fully observed, we disable censoring and truncation, leading to the simplified loss

$$l(g) = -\tfrac{1}{100} \sum_{x,y} \log f_{g(x)}(y),$$

where $f_\mu(y)$ is the density of $\mathcal{N}(\mu = \mu, \sigma = 1)$ evaluated at y.

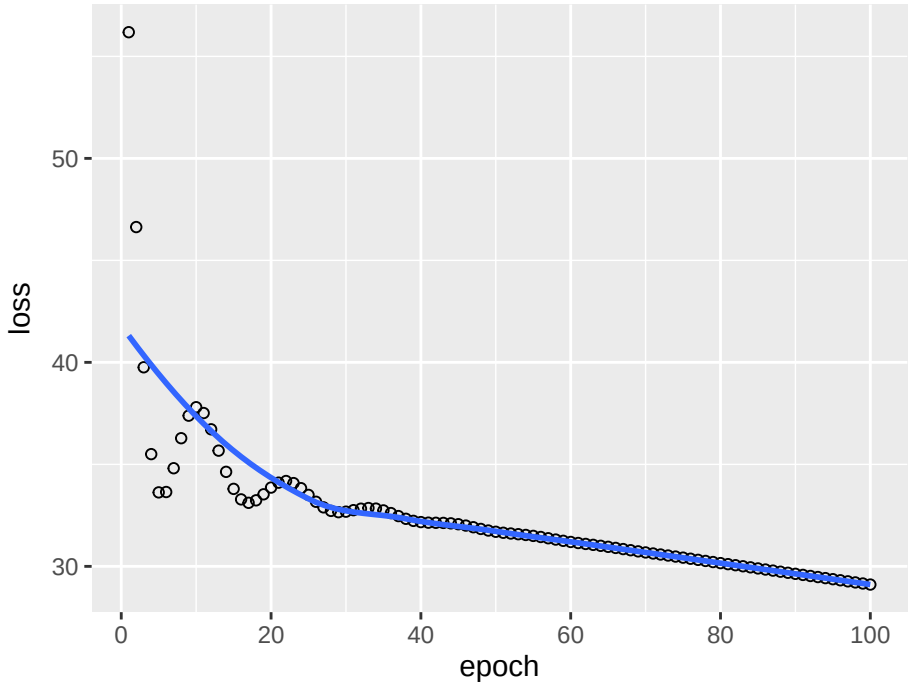
```
R> nnet <- tf_compile_model(
+   inputs = list(nnet_input),
+   intermediate_output = nnet_output,
+   dist = dist,
+   optimizer = keras::optimizer_adam(learning_rate = 0.1),
+   censoring = FALSE,
+   truncation = FALSE
+ )
R> nnet$dist

A NormalDistribution with 1 dof

R> nnet$model

Model: "model"
_________________________________________________________________________
 Layer (type)                Output Shape              Param #
=========================================================================
 x_input (InputLayer)        [(None, 1)]               0
 mean (Dense)                (None, 1)                 2
=========================================================================
Total params: 2 (8.00 Byte)
Trainable params: 2 (8.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________________
```

The fit can now be performed, modifying the parameters (weights) of `nnet` in-place. Note that the argument `y` of fit accepts a `trunc_obs` object. In our example, the vector `y` is silently converted to an untruncated, uncensored `trunc_obs` object. `fit()` returns the `keras_training_history` of the underlying call to `fit()` on the `keras.engine.training.Model`.

```
R> nnet_fit <- fit(
+   nnet,
+   x = dataset$x,
+   y = dataset$y,
+   epochs = 100L,
+   batch_size = 100L,
+   shuffle = FALSE,
+   verbose = FALSE
+ )
```

The training history can be plotted, displaying the loss by epoch (black circles), and a blue smoothing line.

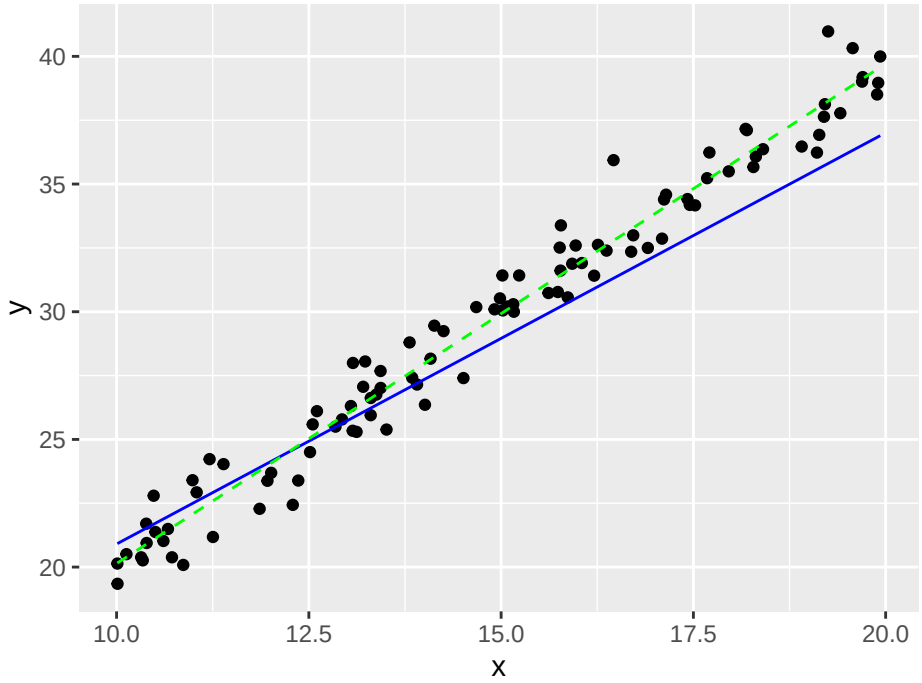
```
R> plot(nnet_fit)
```



The `predict()` method of `reservr_keras_model` takes input tensors and returns the predicted distribution parameters as a list compatible with `dist$get_placeholders()`. We can thus extract the only parameter `mean` and compare it to an OLS fit for the same dataset:

```
R> pred_params <- predict(nnet, data = list(keras::k_constant(dataset$x)))
R>
R> lm_fit <- lm(y ~ x, data = dataset)
R>
R> dataset$y_pred <- pred_params$mean
R> dataset$y_lm <- predict(lm_fit, newdata = dataset, type = "response")
R>
```

```
R> library(ggplot2)
R> ggplot(dataset, aes(x = x, y = y)) +
+   geom_point() +
+   geom_line(aes(y = y_pred), color = "blue") +
+   geom_line(aes(y = y_lm), linetype = 2L, color = "green")
```



Since a `reservr_keras_model` includes the underlying `keras.engine.training.Model`, its parameters can also be extracted and compared to the OLS coefficients

```
R> coef_nnet <- rev(as.numeric(nnet$model$get_weights()))
R> coef_lm <- unname(coef(lm_fit))
R>
R> str(coef_nnet)

 num [1:2] 4.8 1.61

R> str(coef_lm)

 num [1:2] 0.565 1.957
```

We now discuss a more complex example involving censoring, using the right-censored `ovarian` dataset bundled with the **survival** package (R Core Team 2023). Our goal is to predict the rate parameter of an exponential survival time distribution in cancer patients given four features $X = (\texttt{age}, \texttt{resid.ds}, \texttt{rx}, \texttt{ecog.ps})$ collected in the study. The variables `resid.ds`, `rx` and `ecog.ps` are indicator variables coded in $\{1, 2\}$. `age` is a continuous variable with

values in $(38, 75)$. Due to the different scale of the `age` variable, it is useful to separate it from the other variables in order to perform normalization. Normalization using `keras::layer_normalization()` transforms its input variables to zero mean and unit variance. This step is not necessary for the categorical features.

```
R> set.seed(1219L)
R> tensorflow::set_random_seed(1219L)
R> keras::k_set_floatx("float32")
R>
R> dist <- dist_exponential()
R> ovarian <- survival::ovarian
R> dat <- list(
+   y = trunc_obs(
+     xmin = ovarian$futime,
+     xmax = ifelse(ovarian$fustat == 1, ovarian$futime, Inf)
+   ),
+   x = list(
+     age = keras::k_constant(ovarian$age, shape = nrow(ovarian)),
+     flags = k_matrix(ovarian[, c("resid.ds", "rx", "ecog.ps")] - 1.0)
+   )
+ )
```

Next, we define the input layers and shapes, conforming to our input predictor list `dat$x`.

```
R> nnet_inputs <- list(
+   keras::layer_input(shape = 1L, name = "age"),
+   keras::layer_input(shape = 3L, name = "flags")
+ )
```

`age` will be normalized and then concatenated to the other features, stored in `flags`, resulting in a 4-dimensional representation. We then add two hidden ReLU-layers each with 5 neurons to the network and compile the result, adapting the 5-dimensional hidden output to the parameter space $\Theta = (0, \infty)$ for the rate parameter of an exponential distribution. This is accomplished using a dense layer with 1 neuron and the softplus activation function.

```
R> hidden1 <- keras::layer_concatenate(
+   keras::layer_normalization(nnet_inputs[[1L]]),
+   nnet_inputs[[2L]]
+ )
R> hidden2 <- keras::layer_dense(
+   hidden1,
+   units = 5L,
+   activation = keras::activation_relu
+ )
R> nnet_output <- keras::layer_dense(
+   hidden2,
```

```
+   units = 5L,
+   activation = keras::activation_relu
+ )
R>
R> nnet <- tf_compile_model(
+   inputs = nnet_inputs,
+   intermediate_output = nnet_output,
+   dist = dist,
+   optimizer = keras::optimizer_adam(learning_rate = 0.01),
+   censoring = TRUE,
+   truncation = FALSE
+ )
R> nnet$model


Model: "model_1"
__________________________________________________________________________________
 Layer (type)       Output Shape         Para   Connected to         Trainable
                                         m #
==================================================================================
 age (InputLayer)   [(None, 1)]          0      []                   Y
 normalization (No  (None, 1)            3      ['age[0][0]']        Y
 rmalization)
 flags (InputLayer  [(None, 3)]          0      []                   Y
 )
 concatenate (Conc  (None, 4)            0      ['normalization[0]   Y
 atenate)                                       [0]',
                                                 'flags[0][0]']
 dense (Dense)      (None, 5)            25     ['concatenate[0][0   Y
                                                ]']
 dense_1 (Dense)    (None, 5)            30     ['dense[0][0]']      Y
 rate (Dense)       (None, 1)            6      ['dense_1[0][0]']    Y
==================================================================================
Total params: 64 (260.00 Byte)
Trainable params: 61 (244.00 Byte)
Non-trainable params: 3 (16.00 Byte)
__________________________________________________________________________________
```

For stability reasons, the default weight initialization is not optimal. To circumvent this, we estimate a global exponential distribution fit on the observations and initialize the final layer weights such that the global fit is the initial prediction of the network.

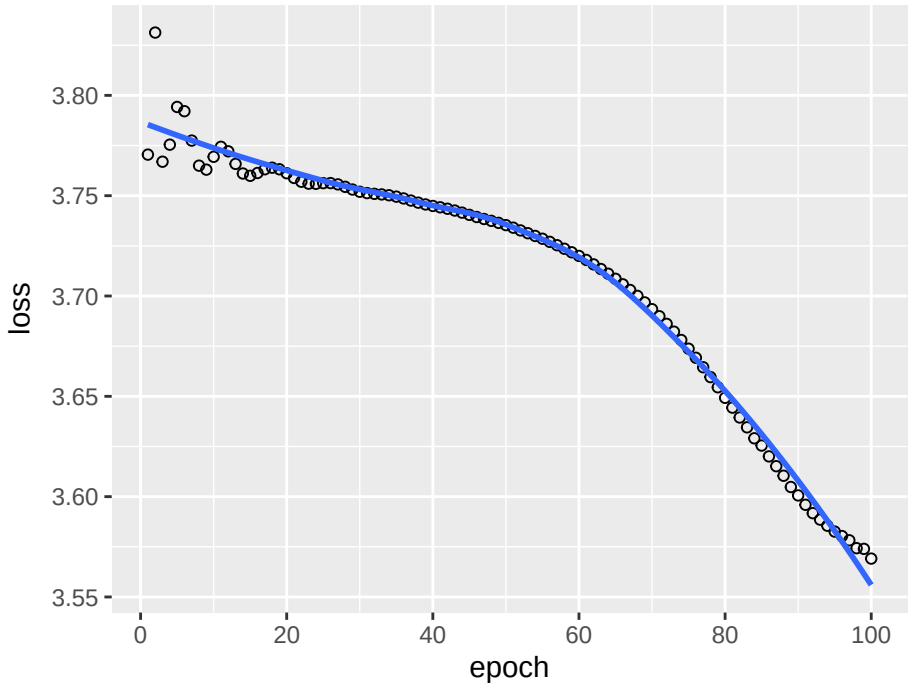
```
R> str(predict(nnet, dat$x))


List of 1
 $ rate: num [1:26] 2.15e-15 7.91e-16 6.10e-14 1.52e-11 5.70e-11 ...
```

```
R> global_fit <- fit(dist, dat$y)
R> tf_initialise_model(nnet, params = global_fit$params, mode = "zero")
R> str(predict(nnet, dat$x))
```

```
List of 1
 $ rate: num [1:26] 0.00077 0.00077 0.00077 0.00077 0.00077 ...
```

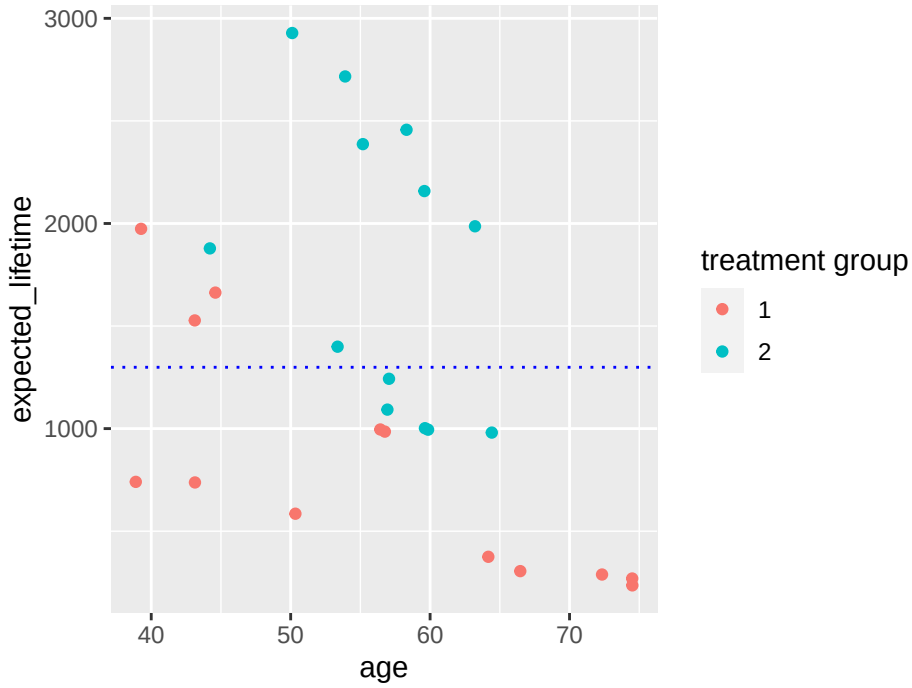
Finally, we can train the network and visualize the predictions.

```
R> nnet_fit <- fit(
+   nnet,
+   x = dat$x,
+   y = dat$y,
+   epochs = 100L,
+   batch_size = nrow(dat$y),
+   shuffle = FALSE,
+   verbose = FALSE
+ )
R> plot(nnet_fit)
```

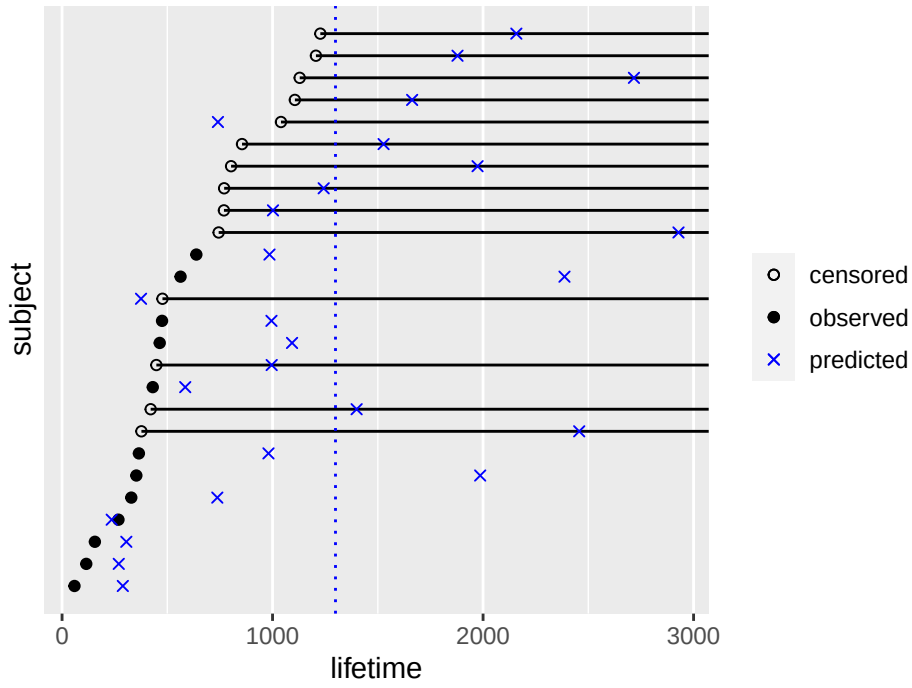



```
R> ovarian$expected_lifetime <- 1.0 / predict(nnet, dat$x)$rate
```

A plot of expected lifetime by (`age`, `rx`) shows that the network learned longer expected lifetimes for lower `age` and for treatment group (`rx`) 2. The global fit is included as a dashed blue line.



Individual predictions and observations can also be plotted on a subject level.



3. Conclusion

We presented **reservr**, a package that supports distribution parameter estimation and distributional regression using R. Both tasks are supported for samples with or without interval censoring and with or without random truncation, a more general form of truncation than what typical packages support. The package includes facilities for (1) description of randomly truncated non-informatively interval censored samples, (2) definition of distribution families to consider, (3) global distribution parameter estimation under an i.i.d. assumption on the sample and (4) distributional regression - employing the **tensorflow** package for flexibility and speed.

Acknowledgements

The Author would like to thank Axel Bücher for proofreading and valuable comments on an earlier version of this article.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015). "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." Software available from tensorflow.org, URL https://www.tensorflow.org/.

Allaire J, Tang Y (2022). *tensorflow: R Interface to 'TensorFlow'*. R package version 2.11.0, URL https://CRAN.R-project.org/package=tensorflow.

Bücher A, Rosenstock A (2022a). "Micro-level prediction of outstanding claim counts based on novel mixture models and neural networks." *Eur. Actuar. J.* doi:10.1007/s13385-022-00314-4.

Bücher A, Rosenstock A (2022b). "Supplementary Material: Micro-level prediction of outstanding claim counts based on novel mixture models and neural networks." URL https://link.springer.com/article/10.1007/s13385-022-00314-4#Sec27.

Bücher A, Rosenstock A (2023). "Combined modelling of micro-level outstanding claim counts and individual claim frequencies in general insurance." *SSRN*. doi:10.2139/ssrn.4564502.

Chang W (2021). *R6: Encapsulated Classes with Reference Semantics*. R package version 2.5.1, URL https://CRAN.R-project.org/package=R6.

Chollet F, Allaire J, *et al.* (2017). *keras: R Interface to 'Keras'*. URL https://github.com/rstudio/keras.

Delignette-Muller ML, Dutang C (2015). "fitdistrplus: An R Package for Fitting Distributions." *Journal of Statistical Software*, **64**(4), 1–34. doi:10.18637/jss.v064.i04.

Dembo RS, Steihaug T (1983). "Truncated-Newton algorithms for large-scale unconstrained optimization." *Mathematical Programming*, **26**, 190–212. doi:10.1007/bf02592055.

Dobson AJ, Barnett AG (2018). *An introduction to generalized linear models*. 4 edition. CRC Press, Taylor & Francis Group. ISBN 978-1-315-18278-0. doi:10.1201/9781315182780.

Dörre A, Emura T (2019). *Analysis of Doubly Truncated Data*. SpringerBriefs in Statistics. Springer Singapore. ISBN 978-981-13-6241-5. doi:10.1007/978-981-13-6241-5.

Groeneboom P, Wellner JA (1992). *Information Bounds and Nonparameteric Maximum Likelihood Estimation*. Oberwolfach Seminars. Birkhäuser Basel. ISBN 978-3-0348-8621-5. doi:10.1007/978-3-0348-8621-5.

Gui W, Huang R, Lin XS (2018). "Fitting the Erlang mixture model to data via a GEM-CMM algorithm." *Journal of Computational and Applied Mathematics*, **343**, 189–205. ISSN 0377-0427. doi:https://doi.org/10.1016/j.cam.2018.04.032.

Jackson C (2016). "flexsurv: A Platform for Parametric Survival Modeling in R." *Journal of Statistical Software*, **70**(8), 1–33. doi:10.18637/jss.v070.i08.

Johnson SG (2007). "The NLopt nonlinear-optimization package." https://github.com/stevengj/nlopt.

Kingma DP, Ba J (2015). "Adam: A Method for Stochastic Optimization." In *Proceedings of the Third International Conference on Learning Representations*, ICLR'15.

Kraft D (1994). "Algorithm 733: TOMP–Fortran modules for optimal control calculations." *ACM Transactions on Mathematical Software*, **20**, 262–281. doi:10.1145/192115.192124.

Lee SC, Lin XS (2012). "Modeling Dependent Risks with Multivariate Erlang Mixtures." *ASTIN Bulletin*, **42**(1), 153–180. doi:10.2143/AST.42.1.2160739.

Liu DC, Nocedal J (1989). "On the limited memory BFGS method for large scale optimization." *Mathematical Programming*, **45**, 503–528. doi:10.1007/bf01589116.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Robin Y (2022). *ROOPSD: R Object Oriented Programming for Statistical Distribution*. R package version 0.3.8, URL https://CRAN.R-project.org/package=ROOPSD.

Rügamer D, Kolb C, Fritz C, Pfisterer F, Kopper P, Bischl B, Shen R, Bukas C, Barros de Andrade e Sousa L, Thalmeier D, Baumann PFM, Kook L, Klein N, Müller CL (2023). "deepregression: A Flexible Neural Network Framework for Semi-Structured Deep Distributional Regression." *Journal of Statistical Software*, **105**(2), 1–31. doi:10.18637/jss.v105.i02.

Stasinopoulos DM, Rigby RA (2007). "Generalized Additive Models for Location Scale and Shape (GAMLSS) in R." *Journal of Statistical Software*, **23**(7), 1–46. doi:10.18637/jss.v023.i07.

Sun J (2006). *The statistical analysis of interval-censored failure time data*. Statistics for Biology and Health. Springer, New York. ISBN 978-0-387-32905-5. doi:10.1007/0-387-37119-2.

Therneau TM (2023). *A Package for Survival Analysis in R*. R package version 3.5-7, URL https://CRAN.R-project.org/package=survival.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Fourth edition. Springer, New York. ISBN 0-387-95457-0, URL https://www.stats.ox.ac.uk/pub/MASS4/.

Affiliation:

Alexander Rosenstock
Heinrich-Heine-Universität Düsseldorf
ARAG SE
Mathematisches Institut
Heinrich-Heine-Universität Düsseldorf
Universitätsstraße 1, 40225 Düsseldorf, Germany
E-mail: alexander.rosenstock@hhu.de

3 Outlook

This chapter provides a brief overview of possible future research opportunities.

The present material only considers IBNR claim counts, providing no explicit model for claim settlement or IBNR loss. An extension in this regard would be useful because it provides a full micro-level reserving model, which could be compared to macro-level loss reserving methods. A natural way to tackle both problems at the same time would be to extend the marking space with a development process for each claim, which would allow the development of each claim to depend on the policy and claim features as well as the reporting delay. A prediction of RBNS reserve could then be obtained by computing the conditional expectation of a claims loss given its already observed features and the state of its development process. IBNR losses would result from simulation of the development process of IBNR claims with number, features and reporting delay sampled from the estimated IBNR claim count model, conditional on the claim being IBNR at the time of evaluation.

Another aspect open for extension would be the introduction of time-dependent features of the policies, e.g. the age of the policyholder. In the present work, change of features was viewed as an independent policy with new features. Note that modelling time-dependent features within a single policy provides no additional generality under the PDMPP assumption because disjoint restrictions of the claim process, such as before and after change of a policy feature, are independent. A potentially useful generalization could be obtained by moving away from the Poisson process assumption, for example to a Cox process with hidden individual risk parameters associated to each policy, introducing a dependence in the claim occurrence process across time intervals. In this case, time-dependent features would require modelling the evolution of these features.

Without extension of the model under study, several aspects of the modelling process could be researched further. More research into the parametric distributions assumed for distributional regression in the central PDMPP model is also possible. For example, the distributional assumption of a BDEGP family for the reporting delay could be substituted by other survival-type distributions, such as the Weibull distribution used by Antonio and Plat [2]. All functional relationships assumed in the distributional regression tasks were assumed to be MLPs. These could be replaced by other function families, such as trees or single-layer perceptrons with a similar structure to generalized linear models. Even withing the class of MLPs, different hyperparameters could be employed and approaches for hyperparameter tuning could be explored. Further algorithms from automatic machine learning (AutoML) could be tested. See [8] for a review of currently available AutoML frameworks. Model selection using backtesting error was performed in the articles 1 and 2. Different approaches to model selection may lead to improved generalization error of the selected model.

Insurance rating [13] is another important aspect of the actuarial profession, requiring a

model for the expected loss for each policy, called the technical premium. The PDMPP framework, when extended with a claim development process, yields such a model. This implies that estimation of reserves and technical premiums can be done simultaneously and the uncertainty of individual reserves can be accounted for when computing risk capital allocations for individual polices. See [7] for an introduction into the subject of risk capital allocation.

Bibliography

[1] Katrien Antonio, Els Godecharle, and Robin Van Oirbeek. "A Multi-State Approach and Flexible Payment Distributions for Micro-Level Reserving in General Insurance". In: *SSRN* (2016). DOI: 10.2139/ssrn.2777467.

[2] Katrien Antonio and Richard Plat. "Micro–level stochastic loss reserving for general insurance". In: *Scandinavian Actuarial Journal* 2014.7 (2014), pp. 649–669. DOI: 10.1080/03461238.2012.755938.

[3] Maximilien Baudry and Christian Y. Robert. "A machine learning approach for individual claims reserving in insurance". In: *Applied Stochastic Models in Business and Industry* 2019.35 (2019), pp. 1127–1155. DOI: 10.1002/asmb.2455.

[4] Axel Bücher and Alexander Rosenstock. "Combined modelling of micro-level outstanding claim counts and individual claim frequencies in general insurance". In: *SSRN* (2023). DOI: 10.2139/ssrn.4564502.

[5] Axel Bücher and Alexander Rosenstock. "Micro-level prediction of outstanding claim counts based on novel mixture models and neural networks." In: *Eur. Actuar. J.* 13.1 (2023), pp. 55–90. DOI: 10.1007/s13385-022-00314-4.

[6] Massimo De Felice and Franco Moriconi. "Claim Watching and Individual Claims Reserving Using Classification and Regression Trees". In: *Risks* 7.4 (2019). DOI: 10.3390/risks7040102.

[7] Jan Dhaene et al. "Optimal capital allocation principles". In: *Journal of Risk and Insurance* 79.1 (2012), pp. 1–28. DOI: 10.1111/j.1539-6975.2011.01408.x.

[8] Pieter Gijsbers et al. "An Open Source AutoML Benchmark". In: (2019). arXiv: 1907.00909 [cs.LG].

[9] Alan F. Karr. *Point Processes and their Statistical Inference*. New York, Basel: marcel dekker inc., 1990. ISBN: 0-8247-7513-9.

[10] Günter Last and Mathew Penrose. *Lectures on the Poisson Process*. Cambridge: Cambridge University Press, 2018. ISBN: 1-107-45843-9.

[11] Thomas Mack. "Distribution-free Calculation of the Standard Error of Chain Ladder Reserve Estimates". In: *ASTIN Bulletin: The Journal of the IAA* 23.2 (1993), pp. 213–225. DOI: 10.2143/AST.23.2.2005092.

[12] Ragnar Norberg. "Prediction of Outstanding Liabilities in Non-Life Insurance". In: *ASTIN Bulletin* 23.1 (1993), pp. 95–115. DOI: 10.2143/AST.23.1.2005103.

[13] Esbjörn Ohlsson and Björn Johansson. *Non-life insurance pricing with generalized linear models*. Vol. 174. Springer, 2010. ISBN: 978-3-642-10790-0.

[14] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2023. URL: https://www.R-project.org/.

[15] Michael Radtke, Klaus D. Schmidt, and Anja Schnaus, eds. *Handbook on loss reserving*. European Actuarial Academy (EAA) Series. Cham: Springer, 2016. ISBN: 978-3-319-30054-2.

[16] M.F. di Rattalma. *The Dieselgate: A Legal Perspective*. Springer International Publishing, 2017. ISBN: 978-3-319-48323-8.

[17] Alexander Rosenstock. *Fitting Distributions and Neural Networks to Censored and Truncated Data: The R Package reservr*. JSS Preprint. 2023. URL: https://ashesitr.github.io/reservr/articles/jss_paper.html.

[18] Greg Taylor. "Maximum Likelihood and Estimation Efficiency of the Chain Ladder". In: *ASTIN Bulletin: The Journal of the IAA* 41.1 (2011), pp. 131–155. DOI: 10.2143/AST.41.1.2084389.

[19] Richard J Verrall. "An investigation into stochastic claims reserving models and the chain-ladder technique". In: *Insurance: Mathematics and Economics* 26.1 (2000), pp. 91–99. DOI: 10.1016/S0167-6687(99)00038-4.

[20] Richard J Verrall and Mario V Wüthrich. "Understanding reporting delay in general insurance". In: *Risks* 4.3 (2016), p. 25. DOI: 10.3390/risks4030025.

[21] Mario Wüthrich. "Machine learning in individual claims reserving". In: *Scandinavian Actuarial Journal* 2018 (2018), pp. 1–16. DOI: 10.1080/03461238.2018.1428681.

Author contribution statement

In this section, each authors individual contributions to the articles included in this thesis are lined out.

1. Axel Bücher and Alexander Rosenstock. "Micro-level prediction of outstanding claim counts based on novel mixture models and neural networks." In: *Eur. Actuar. J.* 13.1 (2023), pp. 55–90. DOI: 10.1007/s13385-022-00314-4
   The initial idea to investigate micro-level methods for claim count prediction was developed by the second author, who also developed the BDEGP family of distributions for modelling and proposed respective estimation algorithms and predictors. Theoretical motivations for the proposed distributions, estimation algorithms and predictors were developed mostly by the first author. Both the simulation study and the case study on real data were developed, performed and interpreted by the second author. Preliminary results were discussed with the first author which lead to improvements in the setup and new ideas for additional simulation experiments. The manuscript was mostly drafted by the second author. Throughout the project, the first author supported the development of the article, and made several improvements, clarifications and corrections that resulted in the final version of the article. Correspondence with the editors and reviewers was handled by the first author. Both authors contributed equally to a minor revision of the article, which the authors were asked to make after the peer review process for the original submission.

2. Axel Bücher and Alexander Rosenstock. "Combined modelling of micro-level outstanding claim counts and individual claim frequencies in general insurance". In: *SSRN* (2023). DOI: 10.2139/ssrn.4564502
   The second author proposed the estimation method for the micro-level model, whose core ideas go back to Norberg [12]. Theoretical motivation based on the M-estimation principle was added by the first author. Lemma 2 and its proof were developed jointly. Both the simulation study and the case study on real data were developed, performed and interpreted by the second author. Preliminary results were discussed with the first author which lead to improvements in the setup and new ideas for additional simulation experiments. The manuscript was mostly drafted by the second author. Throughout the project, the first author supported the development of the article, and made several improvements, clarifications and corrections that resulted in the final version of the article. Correspondence with the editors was handled by the first author.

3. Alexander Rosenstock. *Fitting Distributions and Neural Networks to Censored and Truncated Data: The R Package reservr*. JSS Preprint. 2023. URL: https://ashesitr.github.io/reservr/articles/jss_paper.html
   The article and code was written in its entirety by the first author. Some proofreading was done by the thesis advisor, Prof. Dr. Axel Bücher.

Eidesstattliche Versicherung

Ich versichere an Eides Statt, dass die Dissertation von mir selbständig und ohne unzulässige fremde Hilfe unter Beachtung der „Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Heinrich-Heine-Universität Düsseldorf“ erstellt worden ist. Die Dissertation wurde in der vorgelegten oder ähnlicher Form noch bei keiner anderen Institution eingereicht. Ich habe bisher keine erfolglosen Promotionsversuche unternommen.

Düsseldorf, den 15. April 2024 Alexander Rosenstock