Annals of Actuarial Science (2021), **15**, pp. 419–440
doi:10.1017/S1748499521000087

PAPER

LRMoE.jl: a software package for insurance loss modelling using mixture of experts regression model

Spark C. Tseung[1], Andrei L. Badescu[1], Tsz Chai Fung[2] and X. Sheldon Lin*[1]

[1]Department of Statistical Sciences, University of Toronto, 100 St George Street, Toronto, ON M5S 3G3, Canada and [2]RiskLab, Department of Mathematics, ETH Zurich, Zurich 8092, Switzerland, Department of Risk Management and Insurance, Georgia State University, Atlanta, GA 30303, USA
*Corresponding author. E-mail: sheldon@utstat.utoronto.ca

(Received 29 February 2020; revised 03 December 2020; accepted 18 February 2021; first published online 18 March 2021)

Abstract

This paper introduces a new `julia` package, LRMoE, a statistical software tailor-made for actuarial applications, which allows actuarial researchers and practitioners to model and analyse insurance loss frequencies and severities using the Logit-weighted Reduced Mixture-of-Experts (LRMoE) model. LRMoE offers several new distinctive features which are motivated by various actuarial applications and mostly cannot be achieved using existing packages for mixture models. Key features include a wider coverage on frequency and severity distributions and their zero inflation, the flexibility to vary classes of distributions across components, parameter estimation under data censoring and truncation and a collection of insurance ratemaking and reserving functions. The package also provides several model evaluation and visualisation functions to help users easily analyse the performance of the fitted model and interpret the model in insurance contexts.

Keywords: Multivariate regression analysis; Censoring and truncation; Expectation conditional maximisation algorithm; Insurance ratemaking and reserving; `julia`.

1. Introduction

The Logit-weighted Reduced Mixture-of-Experts (LRMoE) is a flexible regression model introduced by Fung *et al.* (2019b), which is regarded as the regression version of finite mixture model with the mixing weights (called the *gating function*) which depend on the covariates. We may interpret the LRMoE as clustering policyholders into different subgroup with varying probabilities. Conditioned on the subgroup component to which each policyholder is assigned, the distributional properties of loss frequency or severity are governed by mixture component functions (called the *expert function*). Model flexibility, parsimony and mathematical tractability are justified (see Fung *et al.* 2019b), demonstrating a sound theoretical foundation of LRMoE in a general insurance loss modelling perspective. Considering some specific choices of expert functions, Fung *et al.* (2019a) and Fung *et al.* (forthcoming) construct Expectation Conditional Maximisation (ECM) algorithms for efficient frequency and severity model calibrations and show potential usefulness of LRMoE in terms of insurance ratemaking and reserving.

While the existing R package flexmix (Leisch 2004 and Grün & Leisch 2008) may perform parameter estimation for some special cases of LRMoE, it offers only limited choices of component functions (Poisson, Gaussian, Gamma and Binomial) for model fitting. Miljkovic & Grün (2016) have used its extensibility feature to prototype new mixture models with alternative component functions (such as Lognormal, Weibull and Burr), but users are still constrained to choosing a single parametric distribution for all the components.

This paper introduces a new package in `julia` language, LRMoE, a statistical software tailor-made for actuarial applications, which allows actuarial researchers and practitioners to model and analyse insurance loss frequencies and severities using the LRMoE model. The package offers several new distinctive features, which are motivated by various actuarial applications and mostly cannot be achieved by existing packages, including:

- Fast fitting: With the increasing need to analyse large insurance datasets with hundreds of thousands of observations, it is crucial that a statistical package can fit models within a reasonable time frame. Compared with traditional languages such as R, our implementation in `julia` significantly shortens the runtime, allowing users to obtain and analyse results much faster (see section 4.1 for comparison).
- Wider coverage on frequency and severity distributions: Apart from the severity distributions covered by Miljkovic & Grün (2016), the package also covers more frequency expert functions important for actuarial loss modelling, including negative binomial distribution and gamma-count distribution.
- Zero-inflated distributions: Often actuaries are more interested in analysing the aggregate loss for each policyholder instead of considering frequency and severity separately. In this situation, it is common in practice to observe excessive zeroes, which motivates the use of zero-inflated expert functions in the LRMoE, which is offered in this package. Note that efficient computation of zero-inflated LRMoE requires defining an additional latent variable (see section 2.2 for details), further hindering the effectiveness of using the extensibility feature of flexmix.
- Package extensibility: In addition to providing a wide coverage of distributions, our package also allows users to define their customised expert functions with simple guidance from the package documentation. Hence, the package can be used not only within the actuarial community, but also in a wider range of research and practical problems.
- Varying classes of distributions across components: Insurance loss data may exhibit a mismatch between body and tail behaviours, which should be captured using different distributions. One approach is to choose two distributions and combine them using a peaks-over-threshold method (see, e.g. Lee *et al.* 2012 and Scollnik & Sun 2012). Another is to consider a finite mixture model based on different component distributions (see, e.g. Blostein & Miljkovic 2019). The LRMoE package is similar to the latter, and users can select different expert functions across different mixture components, which allows for more flexible and realistic modelling of data.
- Incomplete data: In many actuarial applications, including reinsurance, operational risk management, deductible ratemaking and loss reserving, censored and truncated data are often observed and need to be dealt with. Censoring and truncation of LRMoE is introduced by Fung *et al.* (2020a) with the expert functions restricted to univariate gamma distribution. The new package removes such restriction by offering users versatility to fit randomly censored and truncated multivariate data with many choices of expert functions.
- Model selection and visualisation: In addition to model fitting function, the new package also provides several model evaluation (AIC, BIC) and visualisation (e.g. latent class probabilities, covariate influence) functions to help users easily analyse the performance of the fitted model and interpret the fitted model in the insurance context.
- Insurance ratemaking and reserve calculation: The package further contains a number of pricing and reserving functions (e.g. mean, variance, value-at-risk (VaR), conditional tail expectation (CTE)), which enable actuaries to simultaneously perform ratemaking to multiple insurance contracts with different characteristics, based on abundant choices of premium principles.

The paper is organised as follows. Section 2 reviews the LRMoE model and parameter estimation using the ECM algorithm. In section 3, we use a simulated dataset to demonstrate the basic fitting procedure in the LRMoE package. Section 4 contains more package utilities such as parameter initialisation, model visualisation and pricing function, which are illustrated using a French auto insurance dataset. The paper is concluded with some remarks in section 5. For brevity, we only present code lines which are the most relevant to our new package. The source code, package documentation and complete replication code for all examples in this paper are available on https://github.com/sparktseung/LRMoE.jl and https://sparktseung.github.io/LRMoE.jl/dev/. Further, we have developed a corresponding R package (accelerated by Rcpp) for fitting LRMoE with similar functionalities for users interested in running the package in R instead. We refer such readers to Tseung *et al.* (2020) for the vignette, and https://github.com/sparktseung/LRMoE for the code and documentations.

2. LRMoE Model and Parameter Estimation

In this section, we provide a brief overview of the LRMoE model proposed in Fung *et al.* (2019b), and discuss the ECM algorithm for parameter estimation. For brevity of presentation, we will assume, in sections 2.1 and 2.2, that all response variables (claim frequency or severity) are observed exactly. In section 2.3, we will address data truncation and censoring for the LRMoE model.

2.1. Logit-weighted reduced mixture of experts

Let $\boldsymbol{x}_i = (x_{i0}, x_{i1}, \ldots, x_{iP})^T$ denote the $(P+1)$-dimensional covariate vector for policyholder i ($i = 1, 2, \ldots, n$) with intercept $x_{i0} = 1$, and $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iD})^T$ denote the D-dimensional vector of their response variables, which can be either claim frequency or severity. Let $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^T$ and $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n)^T$ denote all covariates and responses for a group of n policyholders.

Based on the covariates, policyholder i is classified into one of the g latent risk classes by a logit gating function

$$\pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j) = \frac{exp(\boldsymbol{\alpha}_j^T \boldsymbol{x}_i)}{\sum_{j'=1}^{g} exp(\boldsymbol{\alpha}_{j'}^T \boldsymbol{x}_i)}, \quad j = 1, 2, \ldots, g \tag{1}$$

where $\boldsymbol{\alpha}_j = (\alpha_{j0}, \alpha_{j1}, \ldots, \alpha_{jP})^T$ is a vector of regression coefficients for latent class j.

Given the assignment of latent class $j \in \{1, 2, \ldots, g\}$, the response variables $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \ldots, y_{iD})^T$ are conditionally independent. For $d = 1, \ldots, D$ and $i = 1, \ldots, n$, the dth dimension probability density function (or probability mass function) of y_{id} is given by

$$g_{jd}(y_{id}; \delta_{jd}, \boldsymbol{\psi}_{jd}) = \delta_{jd} \mathbf{1}\{y_{id} = 0\} + (1 - \delta_{jd}) f_{jd}(y_{id}; \boldsymbol{\psi}_{jd}) \tag{2}$$

where $\delta_{jd} \in [0, 1]$ is a parameter representing the probability of assigning an observation into a zero-inflated component (i.e. a probability mass at zero), $\mathbf{1}\{y_{id} = 0\}$ is the indicator function, and $f_{jd}(y_{id}; \boldsymbol{\psi}_{jd})$ is the density of some commonly used distribution with parameters $\boldsymbol{\psi}_{jd}$ for modelling insurance losses. Table 1 gives a list of parametric distributions supported by the LRMoE package. Note that a probability mass δ_{jd} at zero allows for more realistic modelling of insurance data which usually exhibit excess zeros. As a naming convention, we will refer to g_{jd} as a *component distribution/expert function*, and f_{jd} as its *positive part*, although claim frequency distributions (e.g. Poisson) also have some probability mass at zero.

Table 1. Distributions supported by LRMoE.

Root	Distribution	$f_{jd}(y)$	Parameters
`gamma`	Gamma	$\frac{1}{\theta^m \Gamma(m)} y^{m-1} e^{-y/\theta}$	$m > 0, \theta > 0$
`lnorm`	Lognormal	$\frac{1}{y\sigma\sqrt{2\pi}} \exp\left[-\frac{(\log y-\mu)^2}{2\sigma^2}\right]$	$\mu \in \mathbb{R}, \sigma > 0$
`invgauss`	Inverse Gaussian	$\sqrt{\frac{\lambda}{2\pi y^3}} \exp\left[-\frac{\lambda(y-\mu)^2}{2\mu^2 y}\right]$	$\mu > 0, \lambda > 0$
`weibull`	Weibull	$\frac{k}{\lambda}\left(\frac{y}{\lambda}\right)^{k-1} \exp\left[-\left(\frac{y}{\lambda}\right)^k\right]$	$k > 0, \lambda > 0$
`burr`	Burr	$\frac{ck}{\lambda}\left(\frac{y}{\lambda}\right)^{c-1}\left[1+\left(\frac{y}{\lambda}\right)^c\right]^{-k-1}$	$k > 0, c > 0, \lambda > 0$
`poisson`	Poisson	$e^{-\lambda}\frac{\lambda^y}{y!}$	$\lambda > 0$
`nbinom`	Negative Binomial	$\binom{y+n-1}{n-1} p^n (1-p)^y$	$n > 0, 0 < p < 1$
`gammacount`	Gamma-Count	$\int_0^{ms} \frac{1}{\Gamma(ys)} u^{ys-1} \exp\{-u\} du$ $- \int_0^{ms} \frac{1}{\Gamma((y+1)s)} u^{(y+1)s-1} \exp\{-u\} du$	$m > 0, s > 0$
`ZI-root`	All above	$g_{jd} = \delta_{jd}\mathbf{1}\{y=0\} + (1-\delta_{jd}) f_{jd}$	$0 < \delta_{jd} < 1$

Under LRMoE, the conditional probability density function (or probability mass fuction) of $\boldsymbol{y}_i$ given covariates $\boldsymbol{x}_i$ is

$$h(\boldsymbol{y}_i; \boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\Psi}) = \sum_{j=1}^{g}\left[\pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j) \times \prod_{d=1}^{D} g_{jd}(y_{id}; \delta_{jd}, \boldsymbol{\psi}_{jd})\right] \tag{3}$$

where $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_g)^T$ is a $g \times (P+1)$-matrix of mixing weights with $\boldsymbol{\alpha}_g = (0, 0, \ldots, 0)^T$ representing the default class, $\boldsymbol{\delta} = (\delta_{jd})_{1\leq j\leq g, 1\leq d\leq D}$ is a $(g \times D)$-matrix of probability masses at zero by component and by dimension, and $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_{jd}: 1 \leq j \leq g, 1 \leq d \leq D\}$ is a list of parameters for the positive part f_{jd} by component and by dimension. Note that $\boldsymbol{\alpha}_g = (0, 0, \ldots, 0)^T$ ensures that the model is identifiable, i.e. there is a one-to-one mapping between the regression distributions and the parameters (see Jiang & Tanner 1999 and Fung *et al.* 2019a). Note that the total number of parameters of the model is given by $(g-1) \times (P+1) + g \times D + \sum_{j=1}^{g}\sum_{d=1}^{D} \#\{\boldsymbol{\psi}_{jd}\}$, where $\#\{\boldsymbol{\psi}_{jd}\}$ is the length of $\boldsymbol{\psi}_{jd}$.

For a group of n policyholders, the likelihood function is given by

$$L(\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\Psi}; \boldsymbol{X}, \boldsymbol{Y}) = \prod_{i=1}^{n} h(\boldsymbol{y}_i; \boldsymbol{x}_i, \boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\Psi}) = \prod_{i=1}^{n}\left\{\sum_{j=1}^{g}\left[\pi_j(\boldsymbol{x}_i; \boldsymbol{\alpha}_j) \times \prod_{d=1}^{D} g_{jd}(y_{id}; \delta_{jd}, \boldsymbol{\psi}_{jd})\right]\right\} \tag{4}$$

2.2. Parameter estimation

For parameter estimation in finite mixture models, the Expectation–Maximisation (EM) algorithm is commonly used (see, e.g. Dempster *et al.* 1977 and McLachlan & Peel 2004). However, for the LRMoE model, the M-step requires maximisation of a non-concave function over all elements of $\boldsymbol{\alpha}$. Fung *et al.* (2019a) thus use the ECM algorithm (Meng & Rubin 1993), which breaks the M-step into several substeps. The ECM algorithm implemented in the LRMoE package is described as follows.

Denote $\boldsymbol{\Phi} = (\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\Psi})$ as the parameters to estimate. For $i = 1, 2, \ldots, n$, we introduce a latent random vector $\boldsymbol{Z}_i = (Z_{i1}, Z_{i2}, \ldots, Z_{ig})^T$, where $Z_{ij} = 1$ if $\boldsymbol{y}_i$ comes from the jth component distribution and $Z_{ij} = 0$ otherwise. For $d = 1, 2, \ldots, D$, we further write $Z_{ij} = Z_{ijd0} + Z_{ijd1}$, where

$Z_{ijd0}=0$ and $Z_{ijd1}=1$ if the dth dimension of $\boldsymbol{y}_i$ comes from the positive part f_{jd} of the jth component, and $Z_{ijd0}=1$ and $Z_{ijd1}=0$ if it comes from the zero inflation δ_{jd}.

The complete-data log-likelihood function is given by

$$\begin{aligned} l^{\text{com}}(\boldsymbol{\Phi};\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}) &= \sum_{i=1}^{n}\sum_{j=1}^{g} Z_{ij}\left\{\log \pi_j\left(\boldsymbol{x}_i;\boldsymbol{\alpha}_j\right)+\sum_{d=1}^{D}\log g_{jd}(y_{id};\delta_{jd},\boldsymbol{\psi}_{jd})\right\} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{g} Z_{ij}\log \pi_j(\boldsymbol{x}_i;\boldsymbol{\alpha}_j)+\sum_{i=1}^{n}\sum_{j=1}^{g}\sum_{d=1}^{D}\left\{Z_{ijd0}\log \delta_{jd}+Z_{ijd1}\log(1-\delta_{jd})\right\} \\ &+\sum_{i=1}^{n}\sum_{j=1}^{g}\sum_{d=1}^{D} Z_{ijd1}\log f_{jd}\left(y_{id};\boldsymbol{\psi}_{jd}\right) \end{aligned} \tag{5}$$

E-Step:

For each $i=1,2,\ldots,n$, the random vector $\boldsymbol{Z}_i$ follows a multinomial distribution with count 1 and probabilities $(\pi_1(\boldsymbol{x}_i;\boldsymbol{\alpha}_1),\pi_2(\boldsymbol{x}_i;\boldsymbol{\alpha}_2),\ldots,\pi_g(\boldsymbol{x}_i;\boldsymbol{\alpha}_g))$. Given $Z_{ij}=1$, the conditional distribution of Z_{ijd0} is Bernoulli with probability δ_{jd}. Hence, at the tth iteration, the posterior expectations of Z_{ij}, Z_{ijd0} and Z_{ijd1} are

$$\begin{aligned} z_{ij}^{(t)} &= \mathrm{E}\{Z_{ij}|\boldsymbol{\Phi}^{(t-1)},\boldsymbol{X},\boldsymbol{Y}\}=\mathrm{P}\{Z_{ij}=1|\boldsymbol{\Phi}^{(t-1)},\boldsymbol{X},\boldsymbol{Y}\} \\ &= \frac{\pi_j\left(\boldsymbol{x}_i;\boldsymbol{\alpha}_j^{(t-1)}\right)\times\prod_{d=1}^{D} g_{jd}\left(y_{id};\delta_{jd}^{(t-1)},\boldsymbol{\psi}_{jd}^{(t-1)}\right)}{\sum_{j'=1}^{g}\pi_{j'}\left(\boldsymbol{x}_i;\boldsymbol{\alpha}_{j'}^{(t-1)}\right)\times\prod_{d=1}^{D} g_{j'd}\left(y_{id};\delta_{j'd}^{(t-1)},\boldsymbol{\psi}_{j'd}^{(t-1)}\right)}, \end{aligned} \tag{6}$$

$$\begin{aligned} z_{ijd0}^{(t)} &= \mathrm{E}\{Z_{ijd0}|\boldsymbol{\Phi}^{(t-1)},\boldsymbol{X},\boldsymbol{Y}\} \\ &= \mathrm{P}\{Z_{ijd0}=1|\boldsymbol{\Phi}^{(t-1)},\boldsymbol{X},\boldsymbol{Y},Z_{ij}=1\}\times\mathrm{P}\{Z_{ij}=1|\boldsymbol{\Phi}^{(t-1)},\boldsymbol{X},\boldsymbol{Y}\} \\ &= \frac{\delta_{jd}^{(t-1)}\mathbf{1}\{y_{id}=0\}}{\delta_{jd}^{(t-1)}\mathbf{1}\{y_{id}=0\}+\left(1-\delta_{jd}^{(t-1)}\right)F_{jd}\left(0;\boldsymbol{\psi}_{jd}^{(t-1)}\right)}\times z_{ij}^{(t)} \end{aligned} \tag{7}$$

and

$$z_{ijd1}^{(t)}=z_{ij}^{(t)}-z_{ijd0}^{(t)} \tag{8}$$

where F_{jd} is the cumulative distribution function of the positive part f_{jd}.

CM-Step:

In the CM-step, we aim to maximise $Q(\boldsymbol{\Phi};\boldsymbol{\Phi}^{(t-1)},\boldsymbol{X},\boldsymbol{Y})$, which can be decomposed into three parts as

$$Q(\boldsymbol{\Phi};\boldsymbol{\Phi}^{(t-1)},\boldsymbol{X},\boldsymbol{Y})=Q_{\boldsymbol{\alpha}}^{(t)}+Q_{\boldsymbol{\delta}}^{(t)}+Q_{\boldsymbol{\Psi}}^{(t)} \tag{9}$$

where

$$Q_{\boldsymbol{\alpha}}^{(t)}=\sum_{i=1}^{n}\sum_{j=1}^{g} z_{ij}^{(t)}\log \pi_j(\boldsymbol{x}_i;\boldsymbol{\alpha}_j), \tag{10}$$

$$Q_{\boldsymbol{\delta}}^{(t)}=\sum_{i=1}^{n}\sum_{j=1}^{g}\sum_{d=1}^{D}\left\{z_{ijd0}^{(t)}\log \delta_{jd}+z_{ijd1}^{(t)}\log(1-\delta_{jd})\right\} \tag{11}$$

and

$$Q_{\boldsymbol{\Psi}}^{(t)} = \sum_{i=1}^{n}\sum_{j=1}^{g}\sum_{d=1}^{D} z_{ijd1}^{(t)} \log f_{jd}\left(y_{id};\boldsymbol{\psi}_{jd}\right) \tag{12}$$

To maximise $Q_{\boldsymbol{\alpha}}^{(t)}$, we use the same conditional maximisation as described in Fung *et al.* (2019a). We first maximise it with respect to $\boldsymbol{\alpha}_1$ with $\boldsymbol{\alpha}_j$ fixed at $\boldsymbol{\alpha}_j^{(t-1)}$ for $j = 2, 3, \ldots, g-1$. The next step is to maximise with respect to $\boldsymbol{\alpha}_2$ with updated $\boldsymbol{\alpha}_1^{(t)}$ and other $\boldsymbol{\alpha}_j$ fixed at $\boldsymbol{\alpha}_j^{(t-1)}$ for $j = 3, 4, \ldots, g-1$. The process continues until all $\boldsymbol{\alpha}$'s have been updated. For obtaining each $\boldsymbol{\alpha}_j^{(t)}$, the Iteratively Reweighted Least Square (IRLS) approach (Jordan & Jacobs 1994) is used until convergence.

For $Q_{\boldsymbol{\delta}}^{(t)}$, each δ_{jd} can be updated using the following closed-form solution

$$\delta_{jd}^{(t)} = \frac{\sum_{i=1}^{n} z_{ijd0}^{(t)}}{\sum_{i=1}^{n}\left(z_{ijd0}^{(t)} + z_{ijd1}^{(t)}\right)} \tag{13}$$

The maximisation of $Q_{\boldsymbol{\Psi}}^{(t)}$ can also be divided into smaller problems by component j and by dimension d. For each $j = 1, \ldots, g$ and $d = 1, \ldots, D$, the problem is reduced to maximise a weighted log-likelihood of an expert function where the weights of each observations are given by $z_{ijd1}^{(t)}$. For updating each $\boldsymbol{\psi}_{jd}^{(t)}$, therefore, closed-form solutions are only available for very special distributions (e.g. Poisson, Lognormal). Numerical optimisation is used in most cases, especially when the observation $\boldsymbol{y}_i$'s are not observed exactly (see section 2.3).

As discussed in McLachlan & Peel (2004), mixture of severity distributions may have unbounded likelihood, which leads to spurious models with extremely large or small parameter values. In the LRMoE package, we adopt the same maximum a posteriori (MAP) approach in Fung *et al.* (forthcoming), which uses appropriate prior distributions to penalise the magnitude of fitted parameters (see section 3). The rationale of including penalty functions is to avoid obtaining spurious model due to unbounded nature of log-likelihood function. The penalty itself should be small enough so that it results to negligible impacts on the fitted model. On the other hand, the penalty functions avoid parameters diverging to unreasonable values so that the fitted model would become more robust. For more details regarding to the rationales and executions, readers are recommended to refer to section 4 of Fung *et al.* (forthcoming).

2.3. LRMoE with censoring and truncation

Censoring and truncation are common in insurance data sets and need to be dealt with. For example, when a policy limit is applied, loss amounts above the limit will be recorded as the limit only, which creates right censoring of the complete loss data; when a policy deductible is applied, loss amounts below the deductible are not reported to the insurer, thus leading to left truncation.

Fung *et al.* (2020a) have discussed the LRMoE model with censoring and truncation, where all component distributions are Gamma. For parameter estimation with data censoring and truncation, the ECM algorithm in section 2.2 is slightly modified, with an additional E-step to remove the uncertainties arising from censoring and truncation. Since the main purpose of this paper is to demonstrate the application of LRMoE package, we will omit the details and refer interested readers to the cited paper.

For all distributions included in it, the LRMoE package can perform parameter estimation in the presence of data truncation and censoring. Consequently, the user's input is slightly different from existing packages for mixture models. A detailed example on model fitting in our package is given in section 3.

2.4. Ratemaking and reserving in LRMoE

The model structure of LRMoE allows for easy computation of quantities relevant to actuarial ratemaking and reserving (see Fung *et al.* 2019b). At the policyholder level, the moments and common measures of dependence (e.g. Kendall's tau and Spearman's rho) of $\boldsymbol{y}_i$ can be computed in simple forms. The value-at-risk (VaR) and conditional tail expectation (CTE) can also be numerically solved without much difficulty. Various premium principles can be applied to price insurance contracts, including pure premium, standard deviation (SD) premium, limited expected value (LEV) and stop-loss (SL) premium. Risk measures can also be calculated for each individual policyholder (e.g. 99% VaR). At the portfolio level, simulation can be conducted to obtain the distribution of the aggregate loss of all policyholders, which is useful for calculating the total loss reserve and premium calculation. The simulation process is facilitated by a data simulator included in our package (see section 4.4).

3. Example: Simulated Dataset

In this section, we will demonstrate how to fit an LRMoE model using a simulated dataset which accompanies the package. The variables of this simulated dataset are described in Table 2, and the dataset is generated by an LRMoE model given in Table 3. The package and dataset can be installed and loaded as follows.

```
# Install and load package
> using Pkg, JLD2
> Pkg.add(url="https://github.com/sparktseung/LRMoE.jl")
> using LRMoE
# Load demo data
> @load "X_obs.jld2" X_obs
> @load "Y_obs.jld2" Y_obs
```

To address data truncation and censoring, the user's input of response $\boldsymbol{Y}$ is different from existing packages. For each dimension d of observation $\boldsymbol{y}_i$, instead of a single numeric input, a quadruple $0 \le t_{id}^l \le y_{id}^l \le y_{id}^u \le t_{id}^u \le \infty$ is needed, where t_{id}^l and t_{id}^u are the lower and upper bounds of truncation, and y_{id}^l and y_{id}^u are the lower and upper bounds of censoring. The exact value of y_{id} lies between censoring bounds such that $y_{id}^l \le y_{id} \le y_{id}^u$. For a sample of size n, an $n \times (4D)$-matrix is needed, where each $n \times 4$-block describes one dimension of $\boldsymbol{Y}$.

Sample rows of `Y_obs` in the demo dataset are shown as follows. These rows of data illustrate three possible scenarios of data truncation or censoring. For the first row, both dimensions of $\boldsymbol{y}_i$ are observed exactly without truncation, so $t_{i1}^l = t_{i2}^l = 0$, $y_{i1}^l = y_{i1}^u = y_{i1}$, $y_{i2}^l = y_{i2}^u = y_{i2}$ and $t_{i1}^u = t_{i2}^u = \infty$. For the second row, the second dimension of $\boldsymbol{y}_i$ is truncated at 5 (e.g. by imposing a policy deductible) but not censored, so $t_{i2}^l = 5$. For the third row, the second dimension of $\boldsymbol{y}_i$ is right-censored at 100 (e.g. by applying a policy limit) and the exact value of y_{i2} is unknown, so $y_{i2}^l = 100$ and $y_{i2}^u = \infty$.

```
> Y_obs[[1,6003,7847],:]
3 x 8 DataFrame
Row
Row  tl_1  yl_1  yu_1  tu_1  tl_2  yl_2     yu_2     tu_2
1    0.0   6.0   6.0   Inf   0.0   89.0332  89.0332  Inf
2    0.0   8.0   8.0   Inf   5.0   37.4133  37.4133  Inf
3    0.0   7.0   7.0   Inf   0.0   100.0    Inf      Inf
```

Table 2. Description of demo dataset.

Covariate[1]	Name	Description
x_{i0}	`intercept`	Constant 1
x_{i1}	`sex`	1 for male, 0 for female
x_{i2}	`agedriver`	Driver's Age: 20–80
x_{i3}	`agecar`	Car's Age: 0–10
x_{i4}	`region`	1 for urban, 0 for rural
Response	Name	Description
y_{i1}	`Y[,1]`	Claim count from business line 1
y_{i2}	`Y[,2]`	Claim severity from business line 2
Row Index	`Y[,1]`	`Y[,2]`
1–6,000	No censoring or truncation	No censoring or truncation
6,001–8,000	No censoring or truncation	Left-truncated at 5[2]
8,001–10,000	No censoring or truncation	Right-censored at 100

Note 1: All covariates are generated independently and uniformly at random.
Note 2: The complete dataset `(X, Y)` contains 10,000 rows. As a result of left-truncating `Y[,2]`, 163 rows of data are discarded, and the observed dataset (`X_obs`, `Y_obs`) has 9837 rows only.

Table 3. True model of demo dataset.

Logit regression coefficients:

$$\boldsymbol{\alpha} = \begin{bmatrix} -0.50 & 1.00 & -0.05 & 0.10 & 1.25 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Component distributions:

	$j=1$	$j=2$
$d=1$	Poisson($\lambda=6$)	ZI-GammaCount($\delta=0.20, m=30, s=0.50$)
$d=2$	Lognormal($\mu=4.0, \sigma=0.30$)	Inverse Gaussian($\mu=20.0, \lambda=20.0$)

To fit an LRMoE model, the user only needs to minimally specify the following: initial guess of logit regression coefficients (`alpha_init`) and what component distributions to use for each dimension and each component, as well as initial guess of their parameters (`comp_init`).

For illustration purposes, we first assume that the user's choice of component distributions coincides with the true model, and the initial guesses of parameters are also close to the true ones. The following sample code provides an example: With two components and five covariates, the initial guess `alpha_init` is a 2×5 matrix, where entries of zero indicate an non-informative guess. As for the component distributions, `comp_init` is a 2×2 matrix, where the number of rows (or columns) corresponds to the number of components (or dimension of response). Each entry of `comp_init` indicates a choice of expert function. In this case, y_{i1} is a mixture of Poisson with mean $\lambda = 10.0$ and zero-inflated Gamma-count distribution with zero inflation $\delta = 0.50$, shape parameter $m = 40$ and dispersion parameter $s = 0.80$. Similarly, y_{i2} is a mixture of Lognormal($\mu = 3.0, \sigma = 1.0$) and Inverse Gaussian($\mu = 15.0, \lambda = 15.0$).

```
# Assume a non-informative guess
alpha_init = fill(0.0, 2, 5)

# Correctly specified component distributions
model_init = [PoissonExpert(10.0)     ZIGammaCountExpert(0.50, 40, 0.80);
              LognormalExpert(3.0, 1.0) Inverse GaussianExpert(15.0, 15.0)]
```

Table 4. Fitted model 1 of demo dataset.

Logit regression coefficients:		
$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} -0.4318 & 1.0688 & -0.0502 & 0.0951 & 1.1967 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$		
Component distributions:		
	$j=1$	$j=2$
$d=1$	Poisson($\lambda=6.016$)	ZI-GammaCount($\delta=0.206$, $m=29.956$, $s=0.489$)
$d=2$	Lognormal($\mu=4.001$, $\sigma=0.297$)	Inverse Gaussian($\mu=20.304$, $\lambda=21.756$)

The fitting function of LRMoE can be called as follows, which will return a fitted model as well as log-likelihood and information criteria AIC and BIC. The result can be inspected by a `summary()` function, or by standard `julia` methods.

```
# Call fitting function
> result_1 = fit_LRMoE(Y_obs, X_obs, alpha_init, model_init)
# Result summary
> summary(result_1)
Model: LRMoE
Fitting converged after seven iterations
Dimension of response: 2
Number of components: 2
Loglik: -73153.32112999677
Loglik (no penalty): -73147.35233984645
AIC: 146320.7046796929
BIC: 146414.22545854456

# Inspect fitted model
> result_1.model_fit.alpha
2 x 5 Array{Float64,2}:
-0.431755   1.06875   -0.0501667   0.0951105   1.19667
0.0         0.0       0.0          0.0         0.0
> result_1.model_fit.comp_dist
2 x 2 Array:
PoissonExpert(6.01676)   ZIGammaCountExpert(0.206196, 29.9564, 0.488854)
LogNormalExpert(4.00105, 0.296734) InverseGaussianExpert(20.3037,
21.7569)
```

The fitted model is summarised in Table 4. The parameter estimates are quite close to the true ones. Considering simulated random noises and loss of information due to censoring and truncation, the fitting function is able to identify the true model when it is known.

In practice, when the true underlying model is not known, the user needs to perform some preliminary analysis on the dataset to determine model specification and parameter initialisation.

Table 5. Fitted model 2 of demo dataset.

Logit regression coefficients:		
$$\tilde{\boldsymbol{\alpha}} = \begin{bmatrix} -0.4023 & 1.0643 & -0.0499 & 0.0955 & 1.1788 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$		
Component distributions:		
	$j=1$	$j=2$
$d=1$	ZI-Poisson($\delta=0.003, \lambda=6.103$)	ZI-Poisson($\delta=0.206, \lambda=30.585$)
$d=2$	Burr($k=1.309, c=5.232, \lambda=58.799$)	Gamma($k=1.624, \theta=12.398$)

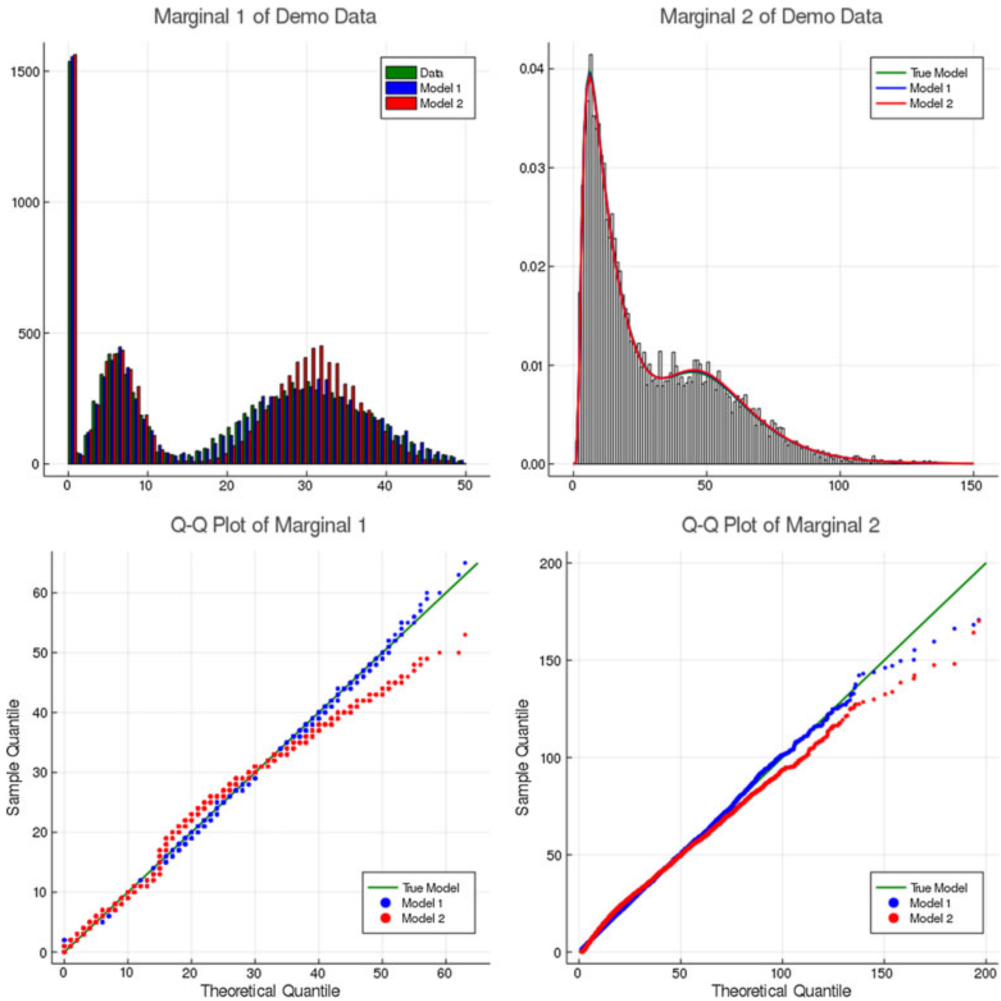

Figure 1. Fitting results of DemoData.

Table 5 contains the parameter estimates of another user-specified LRMoE model for the demo dataset, which has quite different component distributions compared with the true model.

The fitted log-likelihood values for Model 1 and Model 2 are -73,147.35 and -74,491.24, respectively. A graphical comparison of the two models are given in Figure 1. While both models have similar fitting performance for claim severity, Model 2 is noticeably worse in fitting small and extreme values of claim frequency.

Table 6. Description of French auto insurance data.

Covariate	Name	Description
x_{i0}	`Intercept`	Constant 1. Default class for categorical variables.
x_{i1}	`CarAge`	Vehicle age in years. Range: $0 \sim 100$
x_{i2}	`DriverAge`	Driver's age in years. Range: $18 \sim 99$
$x_{i3} \sim x_{i,13}$	`Power`	Power of the car as ordered categorical: d $\sim$ o. Default is 'd'.
$x_{i,14} \sim x_{i,19}$	`Brand`	Brand of the car: 7 categories. Default is "Fiat".
$x_{i,20}$	`Gas`	Car gas: Diesel or Regular. Default is "Diesel".
$x_{i,21} \sim x_{i,29}$	`Region`	Policy region in France: 10 categories. Default is "Aquitaine".
Response	Name	Description
y_{i1}	`ClaimAmount`	Claim amount of the policyholder

4. Example: Real Dataset

In this section, we illustrate how to fit an LRMoE model to a real insurance dataset. As the basic fitting procedure has been discussed in the previous section, we will focus on other utilities of our package, including parameter initialisation, model uncertainty, simulation, actuarial pricing functions and model visualisation.

4.1. Dataset and computation time

Throughout this section, we will use a French auto insurance claims dataset `freMTPLfreq` and `freMTPLsev` included in the R package `CASdatasets` (Dutang & Charpentier, 2019). Exploratory analysis and data cleaning procedures can be found on the accompanying website of our package. The cleaned dataset has 412,609 observations. The claim amount has high zero inflation, as less than 4% of policyholders have filed at least one claim. For positive claim amounts, the distribution is right-skewed, multimodal and heavy-tailed. The covariates used for modelling are described in Table 6.

Before proceeding, we remark on the computational time of our package. Compared with the demo dataset in section 3, analysing `freMTPLfreq` and `freMTPLsev` resembles a realistic actuarial modelling problem with many covariates and observations. In such case, the `julia` programming language is significantly more advantageous compared with traditional statistical languages such as R. Some standard benchmarks can be found on https://julialang.org/benchmarks/. Our experience confirms `julia`'s better performance: it takes around 20 hours in R (with optimisation in Rcpp) to fit a model in this section, while `julia` takes less than 5 hours to complete the same procedure. For the corresponding R packages which we have developed, we refer readers to Tseung *et al.* (2020).

4.2. Parameter initialisation

Since the fitting procedure of LRMoE involves multivariate optimisation, a good initialisation of parameters will often lead to faster convergence, compared with a non-informative guess. Gui *et al.* (2018) have proposed an initialisation procedure for fitting mixture of Erlang distributions, which involves k-means clustering and clusterised method of moments (CMM). This has been used in Fung *et al.* (forthcoming) and offers reasonably good starting values of parameters.

Our package contains an initialisation function which applies the CMM method to all component distributions. Some preliminary analysis is needed to determine the number of clusters (components) to use. Since the positive part of all distributions included in our package is unimodal, a heuristic starting point is to examine the empirical histogram of data and count the number of peaks (see Figure 3).

As an example, the procedure to initialise a three-component LRMoE is given as follows. The user needs to input response Y and covariate X, as used in the fitting function. In addition, the third argument 3 corresponds to the number of components, while the last argument ["continuous"] indicates that the response Y is one-dimensional and continuous. For the demo dataset used in section 3, the input would be ["discrete" "continuous"].

```
# Initialise a three-component model
init_3 = cmm_init(Y, X, 3, ["continuous"])
```

The *cmm_init* function will return a list of parameter initialisation. For the logit regression coefficients, we assume a non-informative guess on covariate influence, resulting in zero coefficients on all covariates. The user is free to incorporate prior knowledge by modification afterwards. The relative size of each cluster is reflected by the intercept terms. In the following initialisation, the size of three latent clusters is proportional to $(e^{-1.41667}, e^{-2.8687}, e^{0.0})$, i.e. the proportion of these clusters within the entire dataset is given by $(0.3842, 0.0330, 0.5827)$.

```
> init_3.alpha_init
3 x 30 Array{Float64,2}:
  -1.41667   0.0   0.0   ...   0.0
  -2.8687   0.0   0.0   ...   0.0
  0.0   0.0   0.0   ...   0.0
```

As for the parameters of component distributions, for each dimension of Y, we will apply the CMM method to obtain initialisation for all types of expert functions. A summary of all initialisation is given in Table 7. The initialisation function cmm_init also returns summary statistics (mean, coefficient of variation, skewness and kurtosis), which helps with choosing what combination of expert functions to use. Initialisation with extremely large or small parameter values could result in a spurious model or bad fit, thus should be avoided.

The `cmm_init` function also returns two suggested models based on the highest log-likelihood (`ll_best`) and the best Kolmogorov–Smirnov test (`ks_best`). For example, the model initialisation with the highest log-likelihood is given by the following.

```
> init_3.ll_best
1 x 3 Array{ZILogNormalExpert{Float64},2}:
 ZILogNormalExpert{Float64}(0.960674, 6.89627, 1.05169)
 ZILogNormalExpert{Float64}(0.961532, 6.819891, 1.097471)
 ZILogNormalExpert{Float64}(0.958489, 6.80334, 1.09848)
```

4.3. Fitting results and model selection

For illustration purposes, we fit only a selected number of LRMoEs to the entire French auto insurance dataset. The fitted log-likelihood, AIC and BIC are summarised in Table 8. In most cases, adding more components will increase the fitted log-likelihood (e.g. consider models `iwl`, `will` and `wllil`). The models selected by AIC and BIC have six and five components, respectively. In this particular setting, BIC heavily penalises models with more components, since the sample size is large, and adding one component roughly increases the number of parameters by 30 (the number of logit regression coefficients).

Table 7. Sample initialisation of parameters.

	Component	1	2	3
Proportion		0.3842	0.0330	0.5827
Zero inflation		0.9615	0.9584	0.9607
Positive data:				
- Mean		1, 686.01	1, 622.73	1, 680.57
- Coefficient of variation		2.51	2.50	2.07
- Skewness		12.15	13.57	11.61
- Kurtosis		185.67	230.49	183.04
- Parameter initialisation				
Gamma	k	0.16	0.16	0.23
	λ	10, 616.28	10, 146.28	7, 231.11
LogNormal	μ	6.82	6.80	6.90
	σ	1.10	1.10	1.05
Inverse Gaussian	μ	1, 868.01	1, 622.73	1, 680.57
	λ	267.76	259.53	390.58
Weibull[1]	k	1.00	1.00	1.00
	θ	1, 686.01	1, 622.73	1, 680.57
Burr[1]	k	351.85	126.52	40.00
	c	0.88	0.90	0.97
	λ	12, 18, 490	317, 504	69, 819

Note: Since the moments cannot be written in closed form, these parameters are solved by maximising log-likelihood based on the observation in clusters.

Table 8. Fitting results of French auto insurance data.

$g = 3$	Log-likelihood	AIC	BIC	$g = 4$	Log-likelihood	AIC	BIC
`lll` (69)	-190,010	380,157	380,991	`lllb` (103)	-189,676	379,558	380,684
`lib` (70)	-190,192	380,524	381,289	`lill` (102)	-189,884	379,973	381,087
`iwl` (69)	-190,110	380,358	381,112	`llll` (102)	-190,008	380,220	381,335
`llb` (70)	-190,202	380,545	381,310	`will` (102)	-190,057	380,319	381,445
`wll` (69)	-190,225	380,588	381,342	`bill` (103)	-190,109	380,422	381,537
$g = 5$	Log-likelihood	AIC	BIC	$g = 6$	Log-likelihood	AIC	BIC
`lllll` (135)	-189,323	378,916	**380,392**	`llllll` (168)	**-189,268**	**378,872**	380,708
`lllil` (135)	-189,662	379,594	381,070	`llblll` (169)	-189,441	379,220	381,067
`bllil` (136)	-189,696	379,663	381,150	`liwlll` (168)	-189,939	380,214	382,050
`bwliw` (136)	-189,767	379,805	381,292	`llwlll` (168)	-190,235	380,805	382,642
`wllil` (135)	-189,903	380,076	381,551	`liwlbl` (169)	-190,276	380,890	382,738

Note 1: Component distributions are represented by first letter, for example, l for Lognormal, b for Burr, etc. All components are zero-inflated. The number in brackets is the total number of parameters.

Note 2: Stopping criterion is < 0.05 improvement in log-likelihood, or 500 iterations.

Note 3: For each g, log-likelihood values are in decreasing order. The overall optimal values are bolded.

Apart from AIC and BIC, cross-validation (CV) is an alternative model selection criterion which avoids overfitting with too many latent components. For example, Gui *et al.* (2018) consider a 10-fold CV for fitting mixture of Erlangs, where the averaged log-likelihood on the test sets is used as a score function to select the optimal number of components. CV can be implemented with the help of parallel computing in `julia` (see also section 4.5).

Table 9. Selected policyholders in French auto insurance dataset.

	Claim	ID	Car age	Driver's age	Car power	Brand	Gas	Region
A	0	1	0	46	g	JK	Diesel	A
B	302	33	1	61	g	JK	Regular	IF
C	9,924	96	0	51	j	JK	Regular	IF

4.4. Pricing and reserving functions

Our package contains a collection of functions related to actuarial pricing, reserving and risk management, including calculation of mean, variance, VaR, CTE, LEV $E[(Y \wedge u)]$ and SL premium $E[(Y - d)_+]$ of the response variable. These functions start with root predict_, followed by appropriate quantities of interest (`mean, var, VaR, CTE, limit, excess`) and corresponding function arguments.

As an illustration, we consider three policyholders in Table 9, where A has no claim history, B has a medium-sized claim and C has a large claim. Below is the sample code to calculate different quantities of interest.

```
# Mean of claim amount of Policyholders A, B and C.
> predict_mean(X[[1, 33, 96],:], alpha_fit, comp_fit)
[52.859, 52.577, 56.368]
# 99% VaR of claim amount of Policyholders A, B and C.
> predict_VaR(X_obs[[1, 33, 96],:], alpha_fit, comp_fit, 0.99)
[1198.970, 1195.295, 1204.096]

# Mean excess of claim amount (d=1000) of Policyholders A, B and C.
> predict_excess(X_obs[[1, 33, 96],:], alpha_fit, comp_fit, 1000)
[29.825, 28.512, 30.435]
```

Actuarial pricing calculation can be done based on either individual loss distributions, or the aggregated loss distribution of the entire portfolio. Both can be achieved using built-in functions in our package.

On the individual level, the functions above can be called directly to calculate the pure premium $E[Y]$, as well as the premium value in the presence of a policy deductible ($E[(Y - d)_+]$) or policy limit ($E[(Y \wedge u)]$). The first part of Table 10 summarises the premium calculation for Policyholders A, B and C, based on individual-level premium principles.

On the portfolio level, the sim_dataset function (see also section 4.6) will simulate one response value (i.e. claim amount) for each individual policyholder, which can be summed up to the aggregated portfolio loss under one possible scenario. Repeated simulation of the entire portfolio will produce the empirical distribution for the aggregated loss. The VaR and CTE of the aggregated loss can be obtained from the simulated sample, which are useful for setting the insurer's total reserve. In addition, the VaR and CTE can be allocated back to policyholders as a loaded premium, according to some weighting scheme which reflects policyholders' relative riskiness (e.g. relative magnitude of their pure premium). The second part of Table 10 summarises the premium calculation for Policyholders A, B and C, based on portfolio-level premium principles.

4.5. Parameter uncertainty

In addition to obtaining point estimates of model parameters, it is crucial to also calculate their confidence intervals in order to identify significant parameters. Given that the log-likelihood in

Table 10. Pricing calculation for selected policyholders in French Auto Insurance Dataset. For portfolio-level premium principles, the allocation is based on the relative size of pure premium, where the weight for policyholder i is $w_i = E[Y_i]/\sum_j E[Y_j]$. For each policyholder and premium principle, the calculation is done based on both prior probability (left column) and posterior probability (right column).

	Policyholder	A		B		C	
	Premium principle	Prior	Posterior	Prior	Posterior	Prior	Posterior
Individual-level	$E[Y]$	52.86	51.99	52.58	215.42	56.37	200.28
	$E[(Y-1,000)_+]$	29.83	29.29	28.51	149.53	30.44	145.76
	$E[Y \wedge 100,000]$	39.22	38.46	39.67	162.63	43.15	140.00
Portfolio-level	VaR(90)	55.89	54.99	55.59	227.83	59.60	211.81
	VaR(95)	57.20	56.28	56.90	233.16	61.00	216.77
	CTE(70)	55.77	54.87	55.48	227.34	59.47	211.36
	CTE(80)	56.59	55.68	56.29	230.69	60.35	214.46
	CTE(90)	58.04	57.10	57.73	236.59	61.89	219.95

equation (4) becomes much more complicated when data are truncated and/or censored, analytically deriving the variance and confidence intervals of parameter estimates would not be feasible, and could be subject to numerical issues in implementation.

Instead, we can use bootstrapping methods in Grün & Leisch (2004). A straightforward nonparametric bootstrapping algorithm is described as follows:

1. Fit an LRMoE using the original dataset, and obtain an estimate $\hat{\mathbf{\Phi}}$.
2. For a fixed number of total iterations, say 200, sample with replacement from the original dataset, on which to fit an LRMoE again with the same component distributions.
3. The estimates $\hat{\mathbf{\Phi}}_1, \hat{\mathbf{\Phi}}_2, \ldots, \hat{\mathbf{\Phi}}_{200}$ from the bootstrapped samples will provide an estimate of the variance of parameters.

The algorithm above can be easily implemented in parallel in `julia` as follows. We have included the estimated confidence intervals in the Appendix, calculated based on 200 bootstrapped samples.

```
> using Distributed
> @Distributed for i in 1:200
  result = fit_LRMoE(Y, X, alpha_init, model_init)
  @save "result_"*"(i)"*".JLD2" result # save the result
  end
```

4.6. Model simulation

In the LRMoE setting, the loss distributions of policyholders are mixtures of the same expert functions, but with potentially different mixing weights. Consequently, the distribution of the aggregate loss, as a sum of individual losses, usually do not admit a simple form.

Our package contains a dataset simulator, which helps with analysing the distribution of aggregated loss. Given a portfolio of policyholders X and a model specification alpha and comp_dist, the simulator will return one set of random realisation of losses for each policyholder.

```
> sim_dataset(alpha, X, comp_dist)
```

With multiple calls to sim_dataset, an approximation of the aggregated loss can be obtained. This has applied in section 4.4 to calculate premium based on portfolio-level prinmium principles.

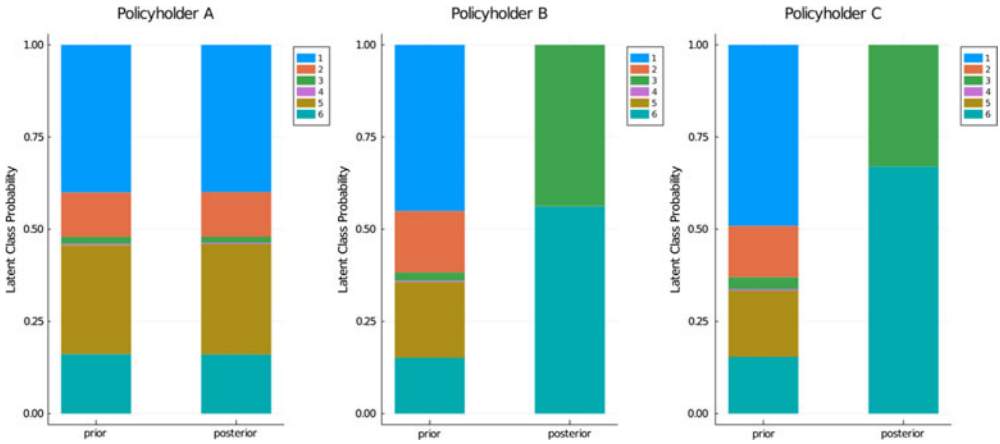

Figure 2. Prior and posterior latent class probabilities for selected policyholders. Component 6 (Lognormal(6.58, 1.95)) corresponds to the tail, while Component 3 (Lognormal(6.95, 1.05)) corresponds to the largest spike in the dataset.

4.7. Model visualisation

After fitting and choosing an appropriate LRMoE model, the user can visualise it with in-package plotting functions, or create more customised plots using generic simulation functions combined with plotting utilities or other dedicated packages such as Plots.jl and StatsPlots.jl. In this subsection, we will use the six-component llllll model for demonstration.

Latent Class Probabilities

The logit regression in LRMoE assigns each policyholder into latent risk classes based on covariates. Given a fitted mode and a vector of covariates, the probability of latent classes can be computed and visualised using the built-in `predict_class_prior` and `plot_class_prob` functions.

Consider the policyholders in Table 9. The predict_class_prior function will return both latent class probabilities (`prob`), as well as the most likely class (`max_prob_idx`). The following sample code calculates their latent class probabilties, where each row represents a policyholder and each column represents a latent class. The prior probabilities are plotted in Figure 2 based solely on covariates information, there is not a large difference between these policyholders, and all are most likely coming from the first latent class.

```
# Predict latent class probabilities, based on covariates and a model
> predict_class_prior(X[[1, 33, 96],:], alpha_fit).prob

0.400565 0.120057 0.0192732 0.00377685 0.295405 0.160923
0.450407 0.166925 0.0226136 0.00350641 0.204941 0.151607
0.490915 0.139407 0.0325873 0.00430055 0.179139 0.153652
```

```
# Predict latent class probabilities, based on covariates and a model
> predict_class_prior(X[[1, 33, 96],:], alpha_fit).max_prob_idx
[1, 1, 1]
```

For policyholders with known claim history, it may be more informative to consider the posterior latent class probabilities by calling corresponding functions for posterior probabilities.

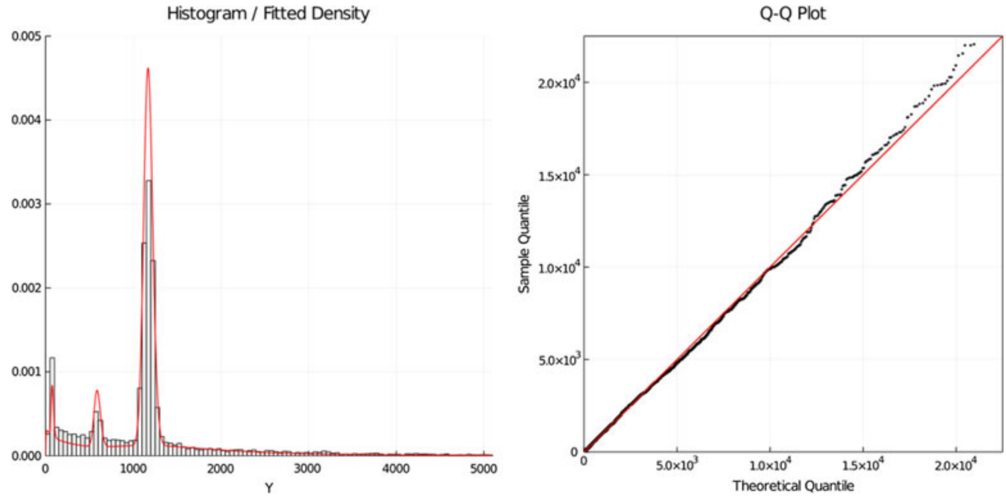

Figure 3. Overall goodness of fit of positive claims in the French auto insurance dataset.

The posterior probability of latent class j is given by $\mathrm{P}\{Z_{ij} = 1|\hat{\mathbf{\Phi}}, \boldsymbol{X}, \boldsymbol{Y}\}$ for fitted parameters $\hat{\mathbf{\Phi}}$ which can be computed analogous to equation (6). Readers may also refer to section 6.3.2 of Fung *et al.* (2019a) for more details.

Consider again the same policyholders in Table 9. The code to calculate posterior probabilities is similar as above. The posterior probabilities are also plotted in Figure 2. Given that Policyholders B and C have non-zero claims, a lot of latent class probability is shifted to Classes 3 and 6, which correspond to the middle and tail parts of the loss distribution (see also the largest spike in Figure 3). Meanwhile, Policyholder A has no claim, resulting in little change to the latent class probabilities since all components have very similar zero inflation.

The posterior probabilities are also helpful for adjusting premium rates. In Table 10, we contrast premium calculation based on both prior and posterior probabilities. It is clear that Policyholder A is rewarded by a decrease in premium for having no claim history, while B and C have significant increase in premium rates.

```
# Predict latent class probabilities, based on covariates and a model
> predict_class_posterior(Y[[1, 33, 96],:], X[[1, 33, 96],:],
                         alpha_fit, comp_fit).prob

0.399295     0.121097     0.0165664   0.00351454    0.299413      0.160114
9.7166e-207  1.36555e-25  0.437268    4.50508e-13   0.000754288   0.561978
0.0          0.0          0.329656    2.77853e-153  0.000165016   0.670179

# Predict latent class probabilities, based on covariates and a model
> predict_class_posterior(Y[[1, 33, 96],:], X[[1, 33, 96],:],
                         alpha_fit, comp_fit).max_prob_idx
[1, 6, 6]
```

Overall Goodness of Fit

The overall goodness of fit can be examined by contrasting the empirical histogram against fitted density curve and the Q–Q plot of empirical and fitted quantiles. These can be produced with the sim_dataset function included in our package, combined with basic plotting functions in `julia`. The corresponding plots are shown in Figure 3.

Covariate Influence

The partial dependence plot is commonly used in machine learning to investigate the influence of a particular covariate on the response, assuming independence amongst covariates (Friedman 2001). For example, the marginal effect of a covariate on the mean claim amount can be obtained using the following steps:

1. Fix the covariate at a particular value (say, car brand = "F") for all policyholders, while other covariates remain unchanged;
2. Use predict_mean_prior function to compute mean response values by policyholder (see section 4.4);
3. Compute the grand mean of all values in step (2), which averages out the effect of other covariates and yields the mean response value when the car brand is of type "F";
4. Repeat steps (1)–(3) for a range of values of the same covariate of interest, for example, varying car brand from "F" to "VAS".

The procedure above can be generalised to continuous covariates and other quantities of interest, such as the quantile of the response variable. The covariate influence plot graphically illustrates how the characteristics of a policyholder are associated with a particular measure of the response variable. For example, plotting the mean (or VaR/CTE) of the response variable against car brand reflects how a policyholder's overall riskiness (or tail risk) is related to the car brand.

Figure 4 shows the influence of covariates Brand, Region and Car Age, which may be interpreted as follows. For car brand, Mercedes, Chrysler or BMW (MCB), Opel, General Motors or Ford (OGF) and Volkswagen, Audi, Skoda or Seat (VAS) are generally riskier. Also, compared with other regions, policies issued in regions IF (Ile-de-France) and L (Limousin) may be considered riskier. In addition, older cars are generally associated with lower risks.

5. Summary and Outlook

This paper presented a new `julia` package LRMoE for actuarial loss modelling. In this first version, the package can cover the most basic need to fit an LRMoE model, as well as help the user visualise the fitted model and calculate insurance premium. There are several future developments in our plan, including:

- Model selection tools: As of the current version, model selection is done by AIC/BIC/CV, all of which require the user to choose and run a selection of model. Some automated model selection procedures may be potentially incorporated in the package (e.g. SCAD type penalty in Fan & Li, 2001 and Yin & Lin, 2016).
- Feature selection tools: Datasets usually contain a large number of covariates, but not all are important for predicting the response. While plots of covariate influence may offer some intuition of their relative importance, it may be more insightful to quantify their influence and provide a function to automatically choose the most influential covariates.

Furthermore, it is crucial to compare the fitting and predictive performances between our proposed LRMoE and classical regression models. Several existing papers have already shown that Erlang Count LRMoE (EC-LRMoE) performs much better than Negative Binomial GLM (see section 6.1 (Table 7) of Fung *et al.* 2019a), and Transformed-gamma LRMoE (TG-LRMoE) performs much better than various severity GLM (see section 5.3.1 (Figure 2 and Table 2) of Fung *et al.* forthcoming). Nonetheless, it is still desirable to conduct extensive comparisons between

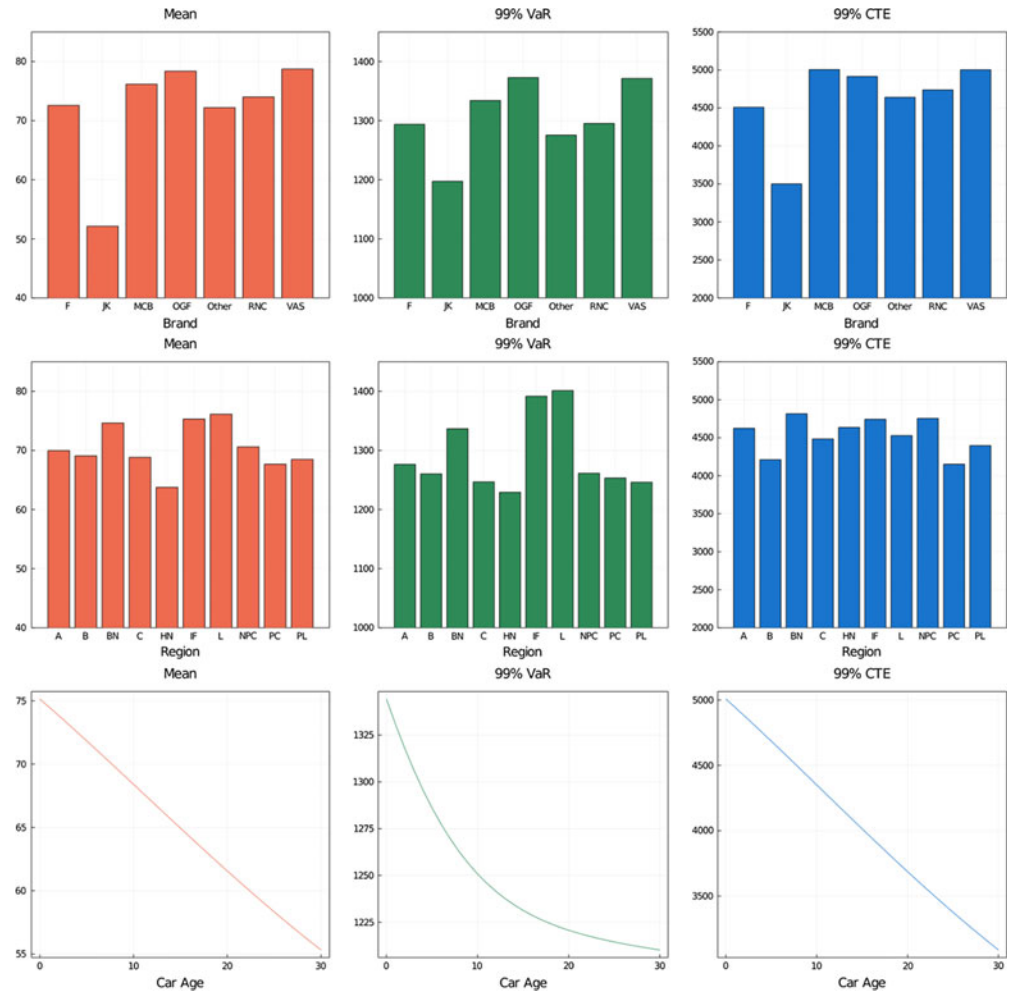

Figure 4. Covariate influence: brand, region and car age.

LRMoE under different expert functions and a wider range of classical models (including, e.g. GAM), and this will be addressed in a new paper that we are currently working with.

Acknowledgements. The authors would like to thank the two anonymous referees for their valuable comments and suggestions. The authors acknowledge an academic research grant provided by the Canadian Institute of Actuaries. This work is also partly supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Spark C. Tseung and Tsz Chai Fung further acknowledge the financial support by the Edwin S.H. Leong Scholarship program and the SOA Hickman Scholars Program, respectively.

References

Blostein, **M.** & **Miljkovic**, **T.** (2019). On modeling left-truncated loss data using mixtures of distributions. *Insurance: Mathematics and Economics*, **85**, 35–46. ISSN 0167-6687.

Dempster, **A.P.**, **Laird**, **N.M.** & **Rubin**, **D.B.** (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, **39**(1), 1–22.

Dutang, **C.** & **Charpentier**, **A.** (2019). CASdatasets: insurance datasets. R package version 1.0-10.

Fan, **J.** & **Li**, **R.** (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, **96**(456), 1348–1360.
Friedman, **J.H.** (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, **29**(5), 1189–1232.
Fung, **T.C.**, **Badescu**, **A.L.** & **Lin**, **X.S.** (2019a). A class of mixture of experts models for general insurance: application to correlated claim frequencies. *ASTIN Bulletin*, **49**(3), 647–688.
Fung, **T.C.**, **Badescu**, **A.L.** & **Lin**, **X.S.** (2019b). A class of mixture of experts models for general insurance: theoretical developments. *Insurance: Mathematics and Economics*, **89**, 111–127.
Fung, **T.C.**, **Badescu**, **A.L.** & **Lin**, **X.S.** (2020a). Fitting censored and truncated regression data using the mixture of experts models. Available in SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3740061
Fung, **T.C.**, **Badescu**, **A.L.** & **Lin**, **X.S.** (forthcoming). A new class of severity regression models with an application to IBNR prediction. *North American Actuarial Journal*. https://doi.org/10.1080/10920277.2020.1729813
Grün, **B.** & **Leisch**, **F.** (2004). Bootstrapping Finite Mixture Models. Mathematics.
Grün, **B.** & **Leisch**, **F.** (2008). FlexMix version 2: finite mixtures with concomitant variables and varying and constant parameters. *Journal of Statistical Software*, **28**(4), 1–35.
Gui, **W.**, **Huang**, **R.** & **Lin**, **X.S.** (2018). Fitting the Erlang mixture model to data via a GEM-CMM algorithm. *Journal of Computational and Applied Mathematics*, **343**, 189–205.
Jiang, **W.** & **Tanner**, **M.A.** (1999). On the identifiability of mixtures-of-experts. *Neural Networks*, **12**(9), 1253–1258.
Jordan, **M.I.** & **Jacobs**, **R.A.** (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, **6**(2), 181–214.
Lee, **D.**, **Li**, **W.K.** & **Wong**, **T.S.T.** (2012). Modeling insurance claims via a mixture exponential model combined with peaks-over-threshold approach. *Insurance: Mathematics and Economics*, **51**(3), 538–550.
Leisch, **F.** (2004). FlexMix: a general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, **11**(8), 1–18.
McLachlan, **G.** & **Peel**, **D.** (2004). *Finite Mixture Models*. New York: John Wiley & Sons.
Meng, **X.L.** & **Rubin**, **D.B.** (1993). Maximum likelihood estimation via the ECM algorithm: a general framework. *Biometrika*, **80**(2), 267–278. ISSN 0006-3444.
Miljkovic, **T.** & **Grün**, **B.** (2016). Modeling loss data using mixtures of distributions. *Insurance: Mathematics and Economics*, **70**, 387–396. ISSN 0167-6687.
Scollnik, **D.P.** & **Sun**, **C.** (2012). Modeling with Weibull-Pareto models. *North American Actuarial Journal*, **16**(2), 260–272.
Tseung, **S.C.**, **Badescu**, **A.L.**, **Fung**, **T.C.** & **Lin**, **X.S.** (2020). LRMoE: an R package for flexible actuarial loss modelling using mixture of experts regression model. Available in SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3740215
Yin, **C.** & **Lin**, **X.S.** (2016). Efficient estimation of Erlang mixtures using iSCAD penalty with insurance application. *ASTIN Bulletin: The Journal of the IAA*, **46**(3), 779–799.

5. Appendix

The appendix contains the parameter estimates of model llllll presented in section, as well as their 95% confidence intervals. Significant parameters are marked in bold.

$\hat{\alpha}$	Class 1		Class 2		Class 3	
Intercept	**0.733**	(0.538, 0.916)	**-1.038**	(-1.221, -0.675)	**-0.496**	(-0.74, -0.243)
Car age	**0.039**	(0.031, 0.043)	**-0.015**	(-0.039, -0.011)	-0.001	(-0.013, 0.003)
Driver age	**0.009**	(0.006, 0.01)	-0.001	(-0.006, 0.002)	-0.002	(-0.007, 0)
Power: e	0.047	(-0.012, 0.192)	-0.099	(-0.23, 0.129)	**0.201**	(0.134, 0.368)
Power: l	-0.124	(-0.402, 0.119)	**-0.506**	(-0.895, -0.077)	-0.016	(-0.467, 0.361)
Power: m	-0.071	(-0.503, 0.32)	**-0.546**	(-0.863, -0.088)	-0.062	(-0.621, 0.436)
Power: n	**-0.577**	(-1.083, -0.155)	0.078	(-0.657, 0.747)	-0.032	(-0.655, 0.501)
Power: o	-0.228	(-0.577, 0.355)	-0.466	(-0.973, 0.099)	0.192	(-0.603, 0.8)
Brand: JK	**-0.121**	(-0.403, -0.07)	**1.023**	(0.663, 1.16)	**-1.493**	(-1.813, -1.269)
Brand: MCB	**-0.381**	(-0.502, -0.173)	**-0.404**	(-0.786, -0.103)	-0.231	(-0.443, 0.039)
Brand: OGF	-0.138	(-0.26, 0.024)	**-0.299**	(-0.539, -0.04)	0.066	(-0.115, 0.248)
Brand: Other	-0.181	(-0.391, 0.03)	**-0.422**	(-0.778, -0.066)	-0.259	(-0.514, 0.006)

$\hat{\alpha}$	Class 1		Class 2		Class 3	
Brand: RNC	**-0.189**	(-0.306, -0.031)	-0.043	(-0.237, 0.201)	**-0.178**	(-0.335, -0.003)
Brand: VAS	**-0.220**	(-0.342, -0.036)	-0.219	(-0.45, 0.127)	-0.020	(-0.167, 0.176)
Gas: Regular	**-0.170**	(-0.251, -0.148)	**-0.097**	(-0.259, -0.018)	**-0.240**	(-0.338, -0.174)
Region: BN	0.066	(-0.075, 0.284)	0.067	(-0.216, 0.396)	**0.248**	(0.008, 0.531)
Region: B	**0.466**	(0.39, 0.683)	**-0.317**	(-0.583, -0.055)	**0.341**	(0.194, 0.587)
Region: C	**0.111**	(0.028, 0.262)	0.048	(-0.08, 0.257)	-0.040	(-0.159, 0.179)
Region: HN	**-0.464**	(-0.808, -0.332)	0.106	(-0.372, 0.307)	**-0.745**	(-1.233, -0.396)
Region: IF	**0.174**	(0.126, 0.357)	**0.517**	(0.426, 0.79)	**0.494**	(0.361, 0.747)
Region: L	**0.514**	(0.315, 0.855)	0.242	(-0.118, 0.78)	**0.763**	(0.363, 1.185)
Region: NPC	-0.102	(-0.25, 0.018)	0.120	(-0.123, 0.306)	-0.176	(-0.421, 0.05)
Region: PL	**0.194**	(0.078, 0.335)	0.086	(-0.082, 0.362)	0.036	(-0.188, 0.245)
Region: PC	**0.449**	(0.343, 0.712)	0.144	(-0.078, 0.583)	**0.332**	(0.18, 0.638)
$\hat{\delta}$	**0.970**	(0.969, 0.970)	**0.981**	(0.980, 0.984)	**0.836**	(0.822, 0.840)
$\hat{\mu}$	**7.065**	(7.064, 7.067)	**6.379**	(6.370, 6.387)	**6.950**	(6.892, 7.031)
$\hat{\sigma}$	**0.044**	(0.042, 0.045)	**0.061**	(0.053, 0.068)	**1.046**	(0.958, 1.109)

$\hat{\alpha}$	Class 4		Class 5		Class 6
Intercept	**-1.362**	(-1.845, -0.902)	-0.077	(-0.095, 0.327)	0
Car age	-0.010	(-0.029, 0.004)	**-0.012**	(-0.029, -0.011)	0
Driver age	**-0.015**	(-0.022, -0.011)	**-0.002**	(-0.005, -0.001)	0
Power: e	**0.275**	(0.035, 0.561)	-0.026	(-0.091, 0.091)	0
Power: f	**0.248**	(0.031, 0.529)	**-0.082**	(-0.217, -0.03)	0
Power: g	0.108	(-0.165, 0.343)	-0.029	(-0.117, 0.034)	0
Power: h	0.098	(-0.328, 0.463)	-0.014	(-0.21, 0.079)	0
Power: i	-0.009	(-0.413, 0.369)	**-0.313**	(-0.619, -0.257)	0
Power: j	0.135	(-0.335, 0.584)	**-0.207**	(-0.425, -0.109)	0
Power: k	0.483	(0, 1.003)	**-0.296**	(-0.523, -0.151)	0
Power: l	0.518	(-0.146, 1.14)	-0.154	(-0.478, 0.091)	0
Power: m	0.134	(-0.786, 0.796)	-0.133	(-0.504, 0.156)	0
Power: n	0.560	(-0.528, 1.085)	-0.311	(-0.769, 0.043)	0
Power: o	0.559	(-0.453, 1.502)	**-0.528**	(-0.966, -0.319)	0
Brand: JK	**-1.794**	(-2.387, -1.432)	**0.797**	(0.728, 1.025)	0
Brand: MCB	0.016	(-0.391, 0.357)	**0.254**	(0.134, 0.453)	0
Brand: OGF	0.093	(-0.234, 0.465)	**-0.128**	(-0.347, -0.01)	0
Brand: Other	0.100	(-0.401, 0.528)	0.087	(-0.085, 0.339)	0
Brand: RNC	-0.287	(-0.562, 0.034)	0.005	(-0.18, 0.091)	0
Brand: VAS	-0.304	(-0.636, 0.108)	-0.051	(-0.236, 0.066)	0

$\hat{\boldsymbol{\alpha}}$	Class 4		Class 5		Class 6
Gas: Regular	-0.035	(-0.207, 0.079)	**0.080**	(0.019, 0.148)	0
Region: BN	0.321	(-0.248, 0.756)	**-0.395**	(-0.641, -0.271)	0
Region: B	**0.434**	(0.121, 0.778)	**-0.609**	(-0.908, -0.625)	0
Region: C	**0.318**	(0.094, 0.595)	**-0.379**	(-0.587, -0.355)	0
Region: HN	-0.530	(-1.258, 0.012)	**0.468**	(0.408, 0.746)	0
Region: IF	0.260	(-0.048, 0.579)	**-0.347**	(-0.445, -0.23)	0
Region: L	-0.287	(-1.042, 0.43)	**-0.714**	(-0.99, -0.527)	0
Region: NPC	**-0.570**	(-1.064, -0.182)	**-0.185**	(-0.376, -0.092)	0
Region: PL	0.388	(-0.009, 0.682)	**-0.462**	(-0.66, -0.389)	0
Region: PC	**0.670**	(0.293, 1.02)	**-0.525**	(-0.755, -0.455)	0
$\hat{\delta}$	**0.905**	(0.896, 0.911)	**0.986**	(0.983, 0.992)	**0.968** (0.962, 0.974)
$\hat{\mu}$	**4.350**	(4.334, 4.366)	**7.341**	(7.172, 7.420)	**6.580** (6.443, 6.688)
$\hat{\sigma}$	**0.183**	(0.165, 0.200)	**0.415**	(0.076, 0.478)	**1.948** (1.835, 2.023)

Cite this article: Tseung SC, Badescu AL, Fung TC and Lin XS (2021). LRMoE.jl: a software package for insurance loss modelling using mixture of experts regression model, *Annals of Actuarial Science* **15**, 419–440. https://doi.org/10.1017/S1748499521000087