Annals of Actuarial Science (2021), **15**, pp. 441–457
doi:10.1017/S1748499521000099

PAPER

mvClaim: an R package for multivariate general insurance claims severity modelling

Sen Hu[1,2*], T. Brendan Murphy[1,2] and Adrian O'Hagan[1,2]

[1]School of Mathematics and Statistics, University College Dublin, Dublin 4, Ireland and [2]Insight Centre for Data Analytics, University College Dublin, Dublin 4, Ireland
*Corresponding author. E-mail: sen.hu.1@ucdconnect.ie

(Received 28 February 2020; revised 14 December 2020; accepted 18 February 2021; first published online 05 April 2021)

Abstract

The mvClaim package in R provides flexible modelling frameworks for multivariate insurance claim severity modelling. The current version of the package implements a parsimonious mixture of experts (MoE) model family with bivariate gamma distributions, as introduced in Hu *et al.*, and a finite mixture of copula regressions within the MoE framework as in Hu & O'Hagan. This paper presents the modelling approach theory briefly and the usage of the models in the package in detail. This package is hosted on GitHub at https://github.com/senhu/.

Keywords: A priori ratemaking; Bivariate gamma distribution; Copula; General insurance; Generalised linear model; Model-based clustering; Mixture of experts model; Mixture of copulas

1. Introduction

mvClaim is a package in the R statistical computing language (R Core Team, 2019) that is developed to provide useful modelling frameworks for joint insurance claims. The frameworks provided in this package are based on the concept of finite mixture models and mixture of experts (MoE) models in statistics and data science, and are incorporated in two common approaches targeting joint or multivariate claims modelling, namely bivariate or multivariate gamma distributions and copulas. The package was written and developed with general insurance claim modelling in mind, but the authors note that the modelling techniques provided in this package can be applied to any positively continuous data, beyond that found in actuarial science.

On the Comprehensive R Archive Network (CRAN, https://cran.r-project.org/), there are various packages providing MoE modelling frameworks, such as MoEClust (Murphy & Murphy 2019, 2020), flexmix (Grün & Leisch, 2007, 2008) and mixtools (Benaglia *et al.*, 2009). There are also various packages considering finite mixture models without covariates, especially model-based clustering (which is essentially a special case of MoE), such as mclust (Scrucca *et al.*, 2016), EMMIXskew (Wang *et al.*, 2018). But none of the currently available packages in R implement the MoE framework using bivariate or multivariate gamma distributions, which are suitable for insurance claim modelling, especially for the severity aspect of modelling.

In general insurance, it is a common phenomenon that claims from multiple risk categories are correlated; however, many current insurance pricing approaches assume independence amongst multiple categories, focusing on independent modelling, and the sum of the risks is then designated as the total risk the insurer takes on. For GI claims modelling, generalised linear models (GLMs) (Nelder & Wedderburn, 1972) have become the industry's standard approach for

© The Author(s), 2021. Published by Cambridge University Press on behalf of Institute and Faculty of Actuaries

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

claim modelling, commonly using a univariate Poisson distribution for the frequency aspect of modelling and a univariate gamma distribution for severity. Therefore, modelling multiple risk perils simultaneously using multivariate Poisson distributions (with Poisson univariate margins) and multivariate gamma distributions (with gamma univariate margins) provides a natural extension beyond their univariate counterparts. For joint claim frequency modelling, much work has used (finite mixtures of) multivariate Poisson regressions. Notably, Bermúdez & Karlis (2012) investigated finite mixtures of multivariate Poisson regressions in GI a pariori ratemaking. For joint severity, Hu *et al.* (2019) used a form of bivariate gamma distributions and MoE models to show that when the dependence is taken into consideration, claim predictions and understanding of risk structures in the policy portfolio can be improved, which is part of what the `mvClaim` package focuses on. For a more detailed literature review on joint claim severity modelling, please refer to Hu *et al.* (2019).

If not using completely specified statistical multivariate distributions, an alternative approach for joint insurance claim modelling is via copulas. The `mvClaim` package also implements copula methods, especially finite mixtures of copula regressions following the framework of MoE. Furthermore, following the ideology of univariate GI claim modelling, only univariate gamma distributions are considered in the copula regression approach in `mvClaim`. On CRAN, `gcmr` (Masarotto & Varin, 2017) and `CopulaRegression` (Kraemer *et al.*, 2013) provide modelling frameworks for copula regressions, but `gcmr` only uses the Gaussian copula, while `CopulaRegression` is only targeted to a specific situation where the claim frequency and severity are dependent (i.e. one margin is modelling count frequency data while the other margin is modelling continuous severity data). Furthermore, neither focus on finite mixtures of copula regressions. The popular `copula` package (Hofert *et al.*, 2018) provides functions to create mixtures of copulas but does not include copula regressions or mixtures of copula regressions.

This paper presents various features of version 0.1.0 of `mvClaim` package. Because the package can be split into two main functionalities, the structure of the paper proceeds as follows: section 2 reviews the bivariate gamma distribution and the related MoE model family, and how to fit such models to data using `mvClaim`; section 3 reviews copula modelling briefly and illustrates how to fit the copula-based models using `mvClaims`. The article concludes in section 4.

Throughout this article, all `R` code demonstration requires that the package has been installed and loaded. The package is under active development, and the latest version can be installed via:

```
R> install.packages("devtools")
R> devtools:install_github("senhu/mvClaim")
```

Then the package can be loaded with:

```
R> library(mvClaim)
```

The `mvClaim` package also contains package vignette in addition to the function documentations, which can be accessed via:

```
R> vignette("mvClaim")
```

2. Bivariate Gamma Mixture of Experts Model Family for Joint Claim Severity

This section reviews a form of bivariate gamma distributions and their associated bivariate gamma MoE model family, as introduced in Hu *et al.* (2019). The particular type of the bivariate gamma distributions focused on in the `mvClaim` package is the one provided by Cheriyan (1941) and Ramabhadran (1951), and detailed in Mathai & Moschopoulos (1991), which is largely overlooked in the literature so far; to the authors' knowledge, there is no current implementation of such distributions in `R`. The bivariate case of the multivariate gamma distribution (BG) is primarily used in the MoE frameworks in `mvClaim`.

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

Let X_1, X_2, X_3 be independent gamma random variables, where $X_i \sim \text{Gamma}(\alpha_i, \beta)$ for $i = 1, 2, 3$, with different shape parameters $\alpha_i > 0$ and a common rate parameter $\beta > 0$. Then vector $\boldsymbol{Y}$ is defined as

$$\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} X_1 + X_3 \\ X_2 + X_3 \end{bmatrix} \sim \text{BG}(\alpha_1, \alpha_2, \alpha_3, \beta) \tag{1}$$

It follows that it has density

$$f_{Y_1,Y_2}(y_1, y_2) = \frac{\beta^{\alpha_1+\alpha_2+\alpha_3} e^{-\beta(y_1+y_2)}}{\Gamma(\alpha_1)\Gamma(\alpha_2)\Gamma(\alpha_3)} \int_{x_3=0}^{\min(y_1,y_2)} e^{\beta x_3} x_3^{\alpha_3-1} (y_1 - x_3)^{\alpha_1-1} (y_2 - x_3)^{\alpha_2-1} dx_3 \tag{2}$$

Since the density is not of analytical form, numerical integration is needed to calculate the density.

Within `mvClaim`, following the standard notation of functions to compute the probability density function (pdf) and to generate variates from the specified distribution in `R`, the utility functions are named `dbivgamma` and `rbivgamma`, respectively. For example,

```
R> a <- dbivgamma(c(1,1), alpha=c(0.3, 0.4, 0.5), beta=1, log = TRUE)
R> a
$value
[1] 2.163084

$message
[1] "OK"

R> b <- rbivgamma(1000, alpha=c(1,1,1), beta=1)
```

The MoE model facilitates modelling, extending the standard mixture models to allow the parameters of the model to depend on concomitant covariates $\boldsymbol{w}_i$, and can be interpreted as model-based clustering with covariates or mixtures of regressions for predictions. Let $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ be an identical and independently distributed bivariate sample of outcomes from a population. Suppose the population consists of G components. Each component can be modelled by a bivariate gamma distribution $f(\boldsymbol{y}_i|\theta_g)$ with component-specific parameters $\theta_g = \{\alpha_{1g}, \alpha_{2g}, \alpha_{3g}, \beta_g\}$, for $g = 1, \ldots, G$ and $i = 1, \ldots, n$. There are also vectors of concomitant covariates $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$ available on which the distribution parameters depend, and which are used to predict future outcome variables. The observed density (conditional on covariates $\boldsymbol{w}_i$) is

$$\begin{aligned} p(\boldsymbol{y}_i|\boldsymbol{w}_i) &= \sum_{g=1}^{G} \tau_g(\boldsymbol{w}_i) p(y_{1i}, y_{2i}|\theta_g(\boldsymbol{w}_i)) \\ &= \sum_{g=1}^{G} \tau_g(\boldsymbol{w}_{0i}) p(y_{1i}, y_{2i}|\alpha_{1ig}(\boldsymbol{w}_{1i}), \alpha_{2ig}(\boldsymbol{w}_{2i}), \alpha_{3ig}(\boldsymbol{w}_{3i}), \beta_i(\boldsymbol{w}_{4i})) \end{aligned} \tag{3}$$

where

$$\begin{aligned} \log(\alpha_{kig}) &= \boldsymbol{\gamma}_{kg}^{\top} \boldsymbol{w}_{ki}, \quad \text{for} \quad k = 1, 2, 3 \\ \log(\beta_{ig}) &= \boldsymbol{\gamma}_{4g}^{\top} \boldsymbol{w}_{4i} \end{aligned} \tag{4}$$

and τ_g is the mixing proportion, i.e. $\sum_{g=1}^{G} \tau_g = 1$. Different (subsets of) concomitant covariates $\boldsymbol{w}_i$ can go to different parts of the regression models for different parameters since the parameters are independent of each other and hence there exists $\boldsymbol{w}_{0i}, \boldsymbol{w}_{1i}, \boldsymbol{w}_{2i}, \boldsymbol{w}_{3i}, \boldsymbol{w}_{4i}$. In the machine learning literature, $\tau_g(\boldsymbol{w}_{0i})$ is called the gating network and $p(y_{1i}, y_{2i}|\theta_g(\boldsymbol{w}_i))$ is called the expert

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

Table 1. Parsimonious parameterisations of the bivariate gamma MoE models. Each $\boldsymbol{\tau}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ can be modelled via different choices, depending on whether covariates are used in modelling that parameter

Gating τ	Expert $\alpha_k, k = 1, 2, 3$	Expert β
C (τ_g; $\nexists\gamma_{0g}$)	C (α_{kg}; $\nexists\gamma_{kg}$)	C (β_g; $\nexists\gamma_{4g}$)
V (τ_{ig}; $\exists\gamma_{0g}$)	V (α_{kig}; $\exists\gamma_{kg}$)	V (β_{ig}; $\exists\gamma_{4g}$)
E ($\tau_g = 1/G$)	E ($\alpha_{ki.}$; $\exists\gamma_k$)	E (β_i; $\exists\gamma_4$)
	I (α_k; $\nexists\gamma_k$)	I (β; $\nexists\gamma_4$)

network. When the mixing proportion is regressed on covariates, it is typically modelled using a multinomial logistic regression, with $\hat{\tau}_g(\boldsymbol{w}_{0i}) = \frac{\exp(\hat{\boldsymbol{\gamma}}_{0g}^{\top}\boldsymbol{w}_{0i})}{\sum_{g'=1}^{G}\exp(\hat{\boldsymbol{\gamma}}_{0g'}^{\top}\boldsymbol{w}_{0i})}$ (when $G = 2$ it becomes a logistic regression). The expert network $p(y_{1i}, y_{2i}|\theta_g(\boldsymbol{w}_i))$ is typically modelled via a GLM framework with a log link function. $\boldsymbol{\gamma}_{0g}$, $\boldsymbol{\gamma}_{1g}$, $\boldsymbol{\gamma}_{2g}$, $\boldsymbol{\gamma}_{3g}$, $\boldsymbol{\gamma}_{4g}$ are the regression coefficients for each parameter and component group.

For MoE it is possible that none, some or all model parameters could depend on the covariates; and since all parameters α_{1g}, α_{2g}, α_{3g}, β_g are independent of one another, interest lies in allowing more parsimonious parameterisations within the bivariate gamma MoE. Depending on the parameterisation of the framework, its interpretations, and whether one allows covariates to be incorporated in the mixing proportions and/or the bivariate gamma densities of the finite mixture models, a family of models is proposed, including:

- Bivariate gamma distribution estimation: implemented via the function `BGE`.
- Bivariate gamma regression: implemented via the function `BGR`, for `EE`, `EI` and `IE` model types.
- Mixture of bivariate gamma clusterings: implemented via the function `MBGC`, for `*CC`, `*CI` and `*IC` model types.
- Mixture of bivariate gamma regressions (i.e. mixture of bivariate gamma clusterings with covariates): implemented via the function `MBGR`, for `*VC`, `*VI`, `*VV`, `*VE`, `*CV`, `*IV`, `*EV`, `*EC` and `*CE` model types.

The notation "*" represents the gating network, which can be either `E` (equal), `C` (constant) or `V` (variable); within the expert networks, `C` represents constant density parameters without covariates, `I` represents identical density parameters without covariates, `V` represents variable parameter values depending on covariates and `E` represents equal parameters depending on covariates. The summaries of the model notation and model types are presented in Tables 1 and 2.

2.1 `BGE`: bivariate gamma distribution estimation

Given there are observations from a bivariate gamma distribution, the distribution parameters can be estimated via:

```
R> a <- rbivgamma(1000, alpha=c(1,1,1), beta=1)
R> est <- BGE(a)
R> summary(est)

-------------------------------------------------------
Bivariate gamma distribution estimated by EM algorithm
-------------------------------------------------------
```

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

Table 2. Bivariate gamma MoE model family based on parsimonious parameterisation: under each model name "$*$" represents the gating network, which can be either "C" (with covariates), "V" (no covariates) or "E" (predefined mixing proportion $1/G$). While all models are possible when $G = 1$, they are essentially equivalent to one of the four models indicated by "•". It also indicates which models have covariates in the expert networks. In the table, $k = 1, 2, 3$ for different α

Name	Model	Parameters	G=1	Covariates in expert
II	Distribution estimation	α_k, β	•	
$*$CC	Standard model-based clustering	α_{kg}, β_g		
$*$CI	Model-based clustering	α_{kg}, β		
$*$IC	Model-based clustering	α_k, β_g		
EE	Bivariate gamma regression	α_{ki}, β_i	•	•
EI	Bivariate gamma regression over α_{ki}	α_{ki}, β	•	•
IE	Bivariate gamma regression over β_i	α_k, β_i	•	•
$*$VC	α_{kig} regressed on covariates	α_{kig}, β_g		•
$*$VI	α_{kig} regressed on covariates	α_{kig}, β		•
$*$VV	α_{kig}, β_{ig} both regressed on covariates	α_{kig}, β_{ig}		•
$*$VE	α_{kig}, β_i both regressed on covariates	α_{kig}, β_i		•
$*$CV	β_{ig} regressed on covariates	α_g, β_{ig}		•
$*$IV	β_{ig} regressed on covariates	α, β_{ig}		•
$*$EV	α_{ki}, β_{ig} both regressed on covariates	α_{ki}, β_{ig}		•
$*$EC	α_{ki} regressed on covariates	α_{ki}, β_g		•
$*$CE	β_i regressed on covariates	α_g, β_i		•

```
Estimated parameters:
   alpha1     alpha2     alpha3       beta
1.0481348  1.0316507  0.9567118  0.9775101

log-likelihood     n  df       AIC       BIC
     -3061.447  1000   4  6130.893  6150.524
```

The `summary` function shows final estimated parameter values, which are very close to the true values in data simulation.

2.2 MBGC: mixture of bivariate gamma clustering

The function `MBGC` implements model-based clustering with bivariate gamma distributions, allowing covariates to enter gating networks if needed. There are three model types within this bivariate gamma MoE family: `*CC, *CI and *IC`. The `mvClaim` package contains several simulated data sets. The detailed simulation process can be found in each data set's documentation in `mvClaim`. One such data set is called `gatingsim`, simulated based on a gating network MoE, i.e. covariates were used only in the gating network to assist with identifying which component the observation was from. The detailed data simulation description can be found in the data set documentation in the package. The pairwise plot of the data are shown in Figure 1, via:

```
R> data("gatingsim")
R> pairs(gatingsim[,1:5], col=c("#00AFBB", "#FC4E07")[gatingsim$label+1],
         pch=c(0,4)[gatingsim$label+1], cex=.5,
         labels = expression("y"[1],"y"[2],"w"[1],"w"[2],"w"[3]))
```

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

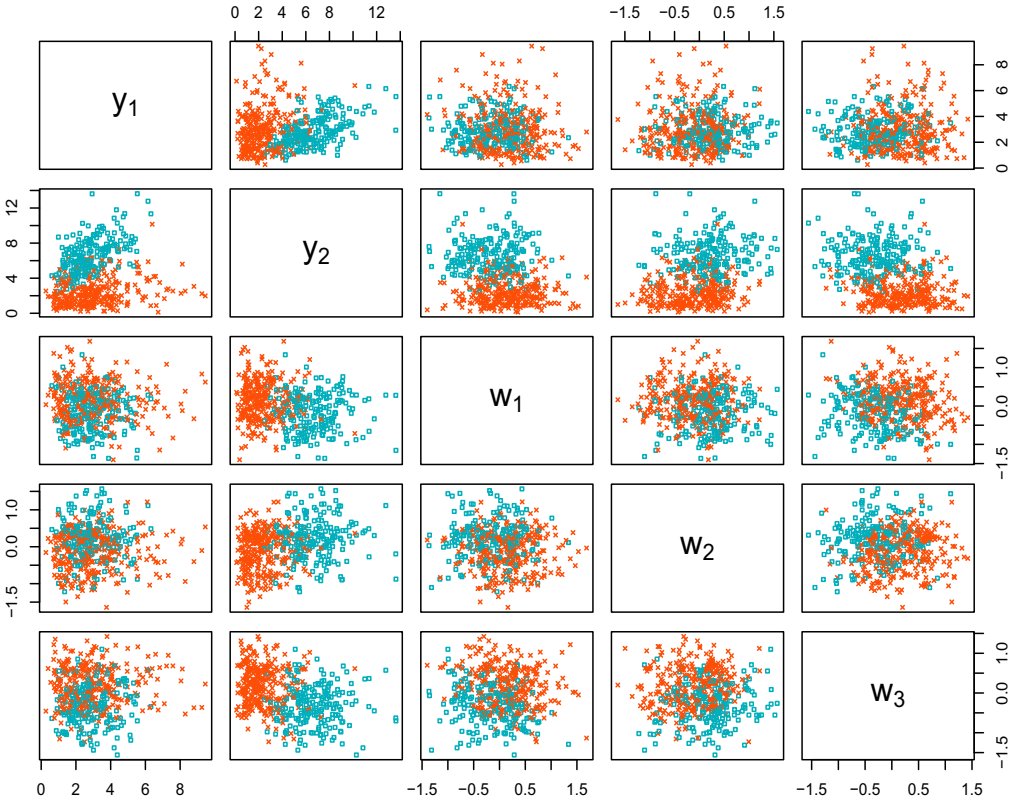


Figure 1. Scatter plot matrix of the simulated data set, coloured by component membership: the teal "□" is component 1 and the red "×" is component 2.

Assuming that the data are a mixture of bivariate gamma distributions and there are no covariates in the experts networks, and the expert network is of type CC and the gating network is of type C or E, the mixture model can be fitted via:

```
R> c0 <- MBGC(modelName = "CC", y = c("y1","y2"), G=1, gating = "C",
              data = gatingsim)
R> c1 <- MBGC(modelName = "CC", y = c("y1","y2"), G=2, gating = "C",
              data = gatingsim)
R> c2 <- MBGC(modelName = "CC", y = c("y1","y2"), G=2, gating = "E",
              data = gatingsim)
R> c3 <- MBGC(modelName = "CC", y = c("y1","y2"), G=3, gating = "C",
              data = gatingsim)
R> c4 <- MBGC(modelName = "CC", y = c("y1","y2"), G=3, gating = "E",
              data = gatingsim)
R> c5 <- MBGC(modelName = "CC", y = c("y1","y2"), G=4, gating = "C",
              data = gatingsim)
R> c6 <- MBGC(modelName = "CC", y = c("y1","y2"), G=4, gating = "E",
              data = gatingsim)
```

Note that fitting G=1 in this case will be equivalent to using the BGE function, and when G=1 in MBGC it is equivalent to setting gating = "C'' or gating = "E''. Model types can also be set

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

using `modelName = "CI"` or `modelName = "IC"`. For the fitted models `c0-c6`, the best model according to `AIC` is `c1` with the model summary:

```
R> summary(c1)

-------------------------------------------------------------------
Mixture of bivariate gamma clusterings (MBGC) fitted by EM algorithm
-------------------------------------------------------------------

MBGC CCC model with two components:

Gating Network Covariates: None

Estimated expert networks parameters:

                alpha1   alpha2    alpha3     beta
component 1 : 2.702119 9.155857 2.6032738 1.781464
component 2 : 2.549917 1.759405 0.6862386 1.091977

log-likelihood    n  df       AIC       BIC
     -1974.077  500   9  3966.155  4004.086

Clustering table:

  1   2
229 271
```

Furthermore, covariates can enter the gating network with, for example, `gating = w1+w2+w3`. For the artificially simulated data set gatingsim, we know the true data generating process of this data set which can be fitted via:

```
R> c9 <- MBGC(modelName = "CC", y=c("y1","y2"), G=2,
              gating = ~w1+w2+w3, data = gatingsim)
R> summary(c9)

-------------------------------------------------------------------
Mixture of bivariate gamma clusterings (MBGC) fitted by EM algorithm
-------------------------------------------------------------------

MBGC VCC model with two components:

Gating Network Covariates: w1 + w2 + w3

Estimated expert networks parameters:

                alpha1   alpha2    alpha3     beta
component 1 : 2.465062 9.158279 2.8686772 1.841325
component 2 : 2.468450 1.647335 0.7596319 1.067756

log-likelihood     n df        AIC    BIC
     -1879.712   500 12   3783.425   3834

Clustering table:

  1   2
227 273
```

The estimated parameter values are very close to the true data generating process. See the package documentation for gatingsim for the true values. The classification plot is shown in Figure 2, via:

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

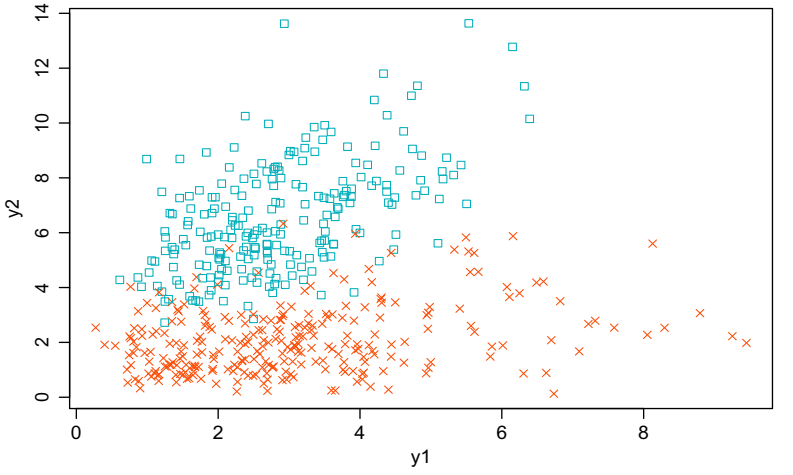


Figure 2. The classification plot of the two components of the c9 model: the teal "□" is component 1 and the red "×" is component 2.

```
R> plot(c9, what = "classification", col=c("#00AFBB", "#FC4E07"),
pch=c(0,4))
```

More details about the data set gatingsim and the analysis can be found in Hu *et al.* (2019).

2.3 BGR: bivariate gamma regression

Bivariate gamma regression (with no mixtures) can be implemented via the function BGR. The function includes three model types, namely EE, EI and IE, see Table 2. Within the mvClaim package, there is a sample data set that is artificially simulated based on one bivariate gamma regression (with no mixtures) called bgr.sim. The details of how this data set is simulated can be found in the data documentation, via ?bgr.sim, and this data set will be used for illustration of the BGR procedure:

```
R> data("bgr.sim")
R> d1 <- BGR(modelName = "EE", y = c("y1","y2"), data = bgr.sim,
             f1 = ~ w1 + w2,
             f2 = ~ w2 + w3,
             f3 = ~ w1 + w2 + w3,
             f4 = ~ w1 + w2 + w3)
R> d2 <- BGR(modelName = "EI", y = c("y1","y2"), data = bgr.sim,
             f1 = ~ w1 + w2,
             f2 = ~ w2 + w3,
             f3 = ~ w1 + w2 + w3)
R> d3 <- BGR(modelName = "IE", y = c("y1","y2"), data = bgr.sim,
             f4 = ~ w1 + w2 + w3)
```

The summary of, for example, model d1 can be extracted as

```
R> summary(d1)

-----------------------------------------------------
Bivariate gamma regression (BGR) fitted by EM algorithm
-----------------------------------------------------
```

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

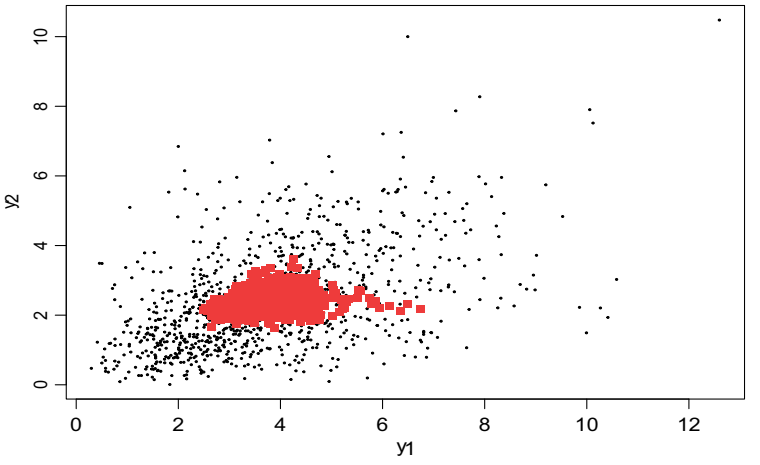


Figure 3. The fitted values of the model `d1`, where the true values are plotted in small black "▪" and the predictions are plotted in red "■". Clearly the fitted values are able to capture the dependence structure within the data.

```
BGR EE model :

    (Intercept)          w1         w2
f1    0.9839882  0.09132803  0.364462
    (Intercept)          w2         w3
f2    0.1135414   0.2271994  0.136847
    (Intercept)          w1         w2         w3
f3    0.5567046    0.114965   0.15995  0.3351603
    (Intercept)          w1         w2         w3
f4    0.1952972  0.09827049  0.1439378  0.1603818

log-likelihood     n df       AIC       BIC
     -3362.788  1000 14  6753.575  6822.284
```

Predictions can be achieved via the predict function, with the plot shown in Figure 3. Clearly the fitted values capture the dependence structure and the overall trend within the data.

```
R> pred <- predict(d1, bgr.sim)
"R> plot(bgr.sim$y1, bgr.sim$y2, xlab=expression("y"[1]),
         ylab=expression("y"[2]), pch=20, cex=0.4)
R> points(pred$fit, col="brown2", pch=15)
```

2.4 MBGR: mixture of bivariate gamma regressions

The function MBGR is illustrated using another simulated data set called fullsim as illustrated in Hu *et al.* (2019), available in the mvClaim package. The data were simulated based on a full MoE model, i.e. with covariates entering both the gating and experts networks. The detailed simulation process can be found in data documentation within the package using `?finsim`. The data set is loaded and plotted using the following code, and the pairwise plot of the data are shown in Figure 4:

```
R> data("fullsim")
R> pairs(fullsim[,1:5], col=c("#00AFBB", "#FC4E07")[fullsim$label+1],
         pch=c(0,4)[fullsim$label+1],
         labels = expression("Y"[1],"Y"[2],"w"[1],"w"[2],"w"[3]),
         cex=0.5,)
```

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

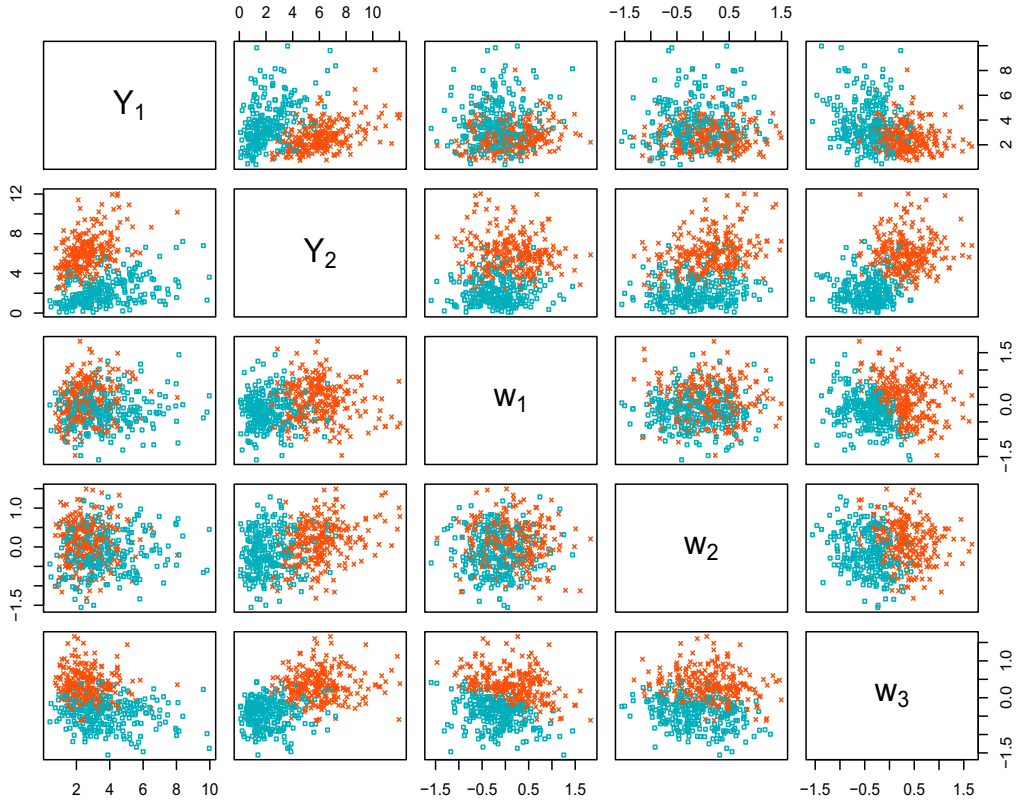


Figure 4. Scatter plot matrix of the simulated data set, coloured by component membership: the teal "□" is component 1 and the red "×" is component 2.

We consider only the cases when covariates are entering the expert networks. In such cases, there are nine different model types: `*VC`, `*VI`, `*VV`, `*VE`, `*CV`, `*IV`, `*EV`, `*EC`, `*CE`. Typically the model selection issue needs to be addressed if the true model is unknown. We recommend stepwise forward selection starting from a null model, i.e. fitting one bivariate gamma distribution without covariates. Then either add one extra component to the current optimal model, or change the model type, or add one covariate to any (combination) of the expert networks of the current model. If $G \geq 2$, covariates should also be added to the gating network. More details can be found in Hu *et al.* (2019). This will lead to many models to be fitted. The optimal model can be identified using AIC or BIC. Here, since the true model is known, it can be fitted via:

```
R> e1 <- MBGR(modelName = "VV", y=c("y1","y2"), G=2, data = fullsim,
              f1 = ~ w1 + w2,
              f2 = ~ w2 + w3,
              f3 = ~ w1 + w2 + w3,
              f4 = ~ w1 + w2 + w3,
              gating = ~ w1 + w2 + w3)
R> summary(e1)

-------------------------------------------------------------
Mixture of bivariate gamma regressions fitted by EM algorithm
-------------------------------------------------------------
```

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

```
MBGR VVV model with two components:

Gating Network Covariates: w1 + w2 + w3
Alpha1 Expert Network Covariates: w1 + w2
Alpha2 Expert Network Covariates: w2 + w3
Alpha3 Expert Network Covariates: w1 + w2 + w3
Beta Expert Network Covariates: w1 + w2 + w3

log-likelihood    n df       AIC       BIC
     -1635.083  500 32  3334.167  3469.034


Clustering table:

  1   2
249 251
```

The estimated coefficients can be extracted via:

```
R> e1$coefficients

$expert
$expert$alpha1

                g=1       g=2
intercept  0.3705786 1.0208859
w1         0.5543963 0.1389520
w2        -0.3160657 0.1290235

$expert$alpha2

               g=1        g=2
intercept 2.1922028 0.20928608
w2        0.1800699 0.05974238
w3        0.1315515 0.58844458

$expert$alpha3

               g=1       g=2
intercept 1.46690414 0.4397885
w1        0.12588675 0.1517797
w2        0.14188440 0.3239069
w3        0.04820575 -0.0470299

$expert$beta

                g=1       g=2
intercept 0.827563190.17362981
w1        0.089311250.07127085
w2       -0.018876540.09544351
w3        0.067158790.07443204

$gating

  (Intercept)        w1        w2        w3
2   -12.72619 -60.15125 -44.48708 -143.2479
```

which are very close to the true parameter values in the data simulation process.

If focusing only on the clustering result, the classification based on the mixture of bivariate gamma distributions with covariates can be plotted via:

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

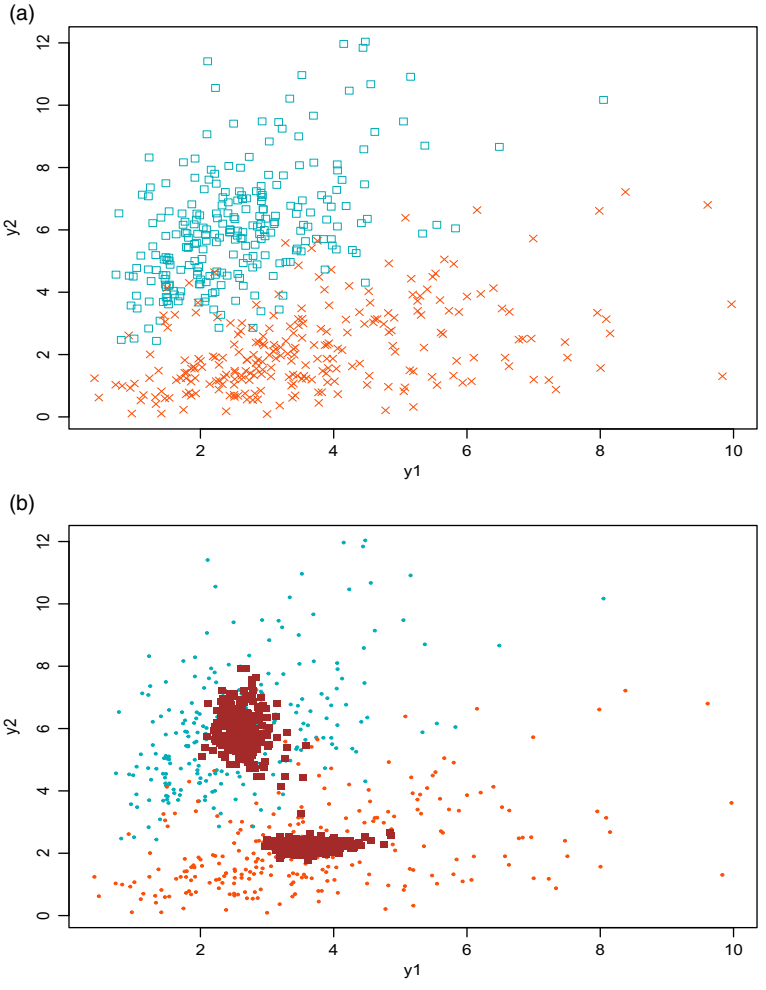


Figure 5. The fitted e1 model: (a) shows the classification plot of $\boldsymbol{y}$, where the teal "□" is component 1, the red "×" is component 2; (b) shows the fitted values, where the true values are plotted in small teal and red "." and the predictions are plotted in maroon "■".

```
R> plot(e1, what = "classification", col=c("#00AFBB", "#FC4E07"),
        pch=c(0,4))
```

The plot is shown in Figure 5. Furthermore, the plot function also allows the fitted values of the model to be plotted, as shown in Figure 5, via:

```
R> plot(e1, what = "fitted", col=c("#00AFBB", "#FC4E07"))
```

This plot shows that the model is able to capture the two components with roughly two regression lines, following the overall structure of the data.

There is also an accompanying fullsim.test data set available in mvClaim that has been simulated similarly for test data prediction purposes. Based on the fitted model, predictions on the new data can be done via:

```
R> pred <- predict(e1, newdata=fullsim.test)
```

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

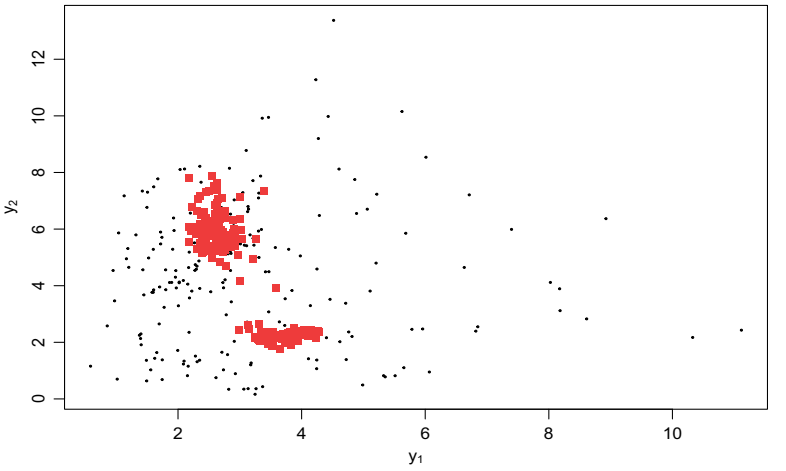


Figure 6. Predicted values of ***y*** for the test data using model `e1`. The true values are plotted in small black "." and the predictions are plotted in red "■". Clearly the predicted values identified the dependence structures within different components and can be shown to perform better than those of univariate modelling.

The predictions are plotted in Figure 6. Clearly the out-of-sample predictions identified the dependence structures within two different components. In fact, the predicted values are the means of the predictive distributions and each prediction has a distribution. It can also be shown that these predictions are better than separate univariate GLM modelling. See Hu *et al.* (2019) for more details.

```
R> data("fullsim.test")
R> plot(fullsim.test$y1, fullsim.test\$y2, pch=20, cex=0.4,
        xlab=expression("y"[1]), ylab=expression("y"[2]))
R> points(pred$fit, col="brown2", pch=15)
```

3. Finite Mixture of Copula Regressions

Although it is often preferred to model joint risks using bivariate distributions directly, this can be conceptually challenging due to the complex specification and implementation of such distributions compared to univariate distributions. There is also a limited number of options for such bivariate distributions. When modelling joint insurance claim risks, a copula is another popular option in the literature. The `mvClaim` package includes implementation of copulas in a MoE framework, namely finite mixtures of copula clusterings and regressions, built on top of the popular `copula` package on CRAN.

Suppose there are two random variables, Y_1 and Y_2, with joint distribution function $F(y_1, y_2)$ and two marginal distributions $F_1(y_1)$ and $F_2(y_2)$, respectively. By Sklar's theorem (Sklar (1959)) any bivariate CDF can be written in the form

$$F(y_1, y_2) = C\left[F_1(y_1), F_2(y_2)\right] \tag{5}$$

where the function $C(u, v)$ is a copula (Joe, 1997; Nelsen, 2007; Joe, 2014). There are various approaches to estimating the parameter θ of a given parametric copula $C(\theta)$, which also depends on estimation of the marginal distributions. For a review, please refer to Hu & O'Hagan (2021). When the marginal distributions are estimated parametrically, covariates can be incorporated in the marginals following a GLM framework as extra information, assisting in marginal and further parametric copula estimations. This is sometimes called copula regression (Yan, 2007; Masarotto

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

& Varin, 2017; Kraemer *et al.*, 2013). Furthermore, copulas can also be mixed to form finite mixtures of copulas for complex and heterogeneous dependence structures, see Kosmidis & Karlis (2016) and Arakelian & Karlis (2014). Therefore, copulas can be estimated via such finite mixtures of copula regressions, following the MoE models construction and inference.

The setup of the copula MoE model is now briefly reviewed, following Hu & O'Hagan (2021). Suppose there are bivariate identical and independently distributed data $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$ of size N, which consist of G components. There also exist vectors of covariates $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, based on which let $\boldsymbol{X}_j$ be the covariate design matrix for margin j, for $j = 1, 2$; i.e. covariates are not necessarily the same for both marginals. Each component g can be specified by a copula function C_g and two marginal distributions of CDF F_{jg} and PDF f_{jg}, for $g = 1, \ldots, G$. Now let $\boldsymbol{\alpha}_g$ be the copula parameter, $\boldsymbol{\beta}_{jg}$ be the regression coefficients of margin j, γ_{jg} be any distribution specific parameters of margin j (e.g. shape parameters of gamma distributions), so that each margin follows a specified distribution within a GLM framework such as $\mu_{ji} = \delta^{-1}(\boldsymbol{x}_{ijg}^{\top}\boldsymbol{\beta}_{jg})$ where δ is the link function. Then each component can be regarded as a bivariate distribution (or component-specific copula regression), denoted as $h_g(\boldsymbol{y}_i; \boldsymbol{\theta}_g)$, where $\boldsymbol{\theta}_g = \left\{\boldsymbol{\alpha}_g, \boldsymbol{\beta}_{jg}, \boldsymbol{\gamma}_{jg}\right\}$. The observed data likelihood of the mixture copula model is

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \prod_{i=1}^{N}\sum_{g=1}^{G}\tau_g h_g(\boldsymbol{y}_i; \boldsymbol{\theta}_g) \\ &= \prod_{i=1}^{N}\sum_{g=1}^{G}\tau_g c_g\left(F_1(y_{1i}; \boldsymbol{\beta}_{1g}, \boldsymbol{\gamma}_{1g}), F_2(y_{2i}; \boldsymbol{\beta}_{2g}, \boldsymbol{\gamma}_{2g}); \boldsymbol{\alpha}_g\right) f_1(y_{1i}; \boldsymbol{\beta}_{1g}, \boldsymbol{\gamma}_{1g}) f_2(y_{2i}; \boldsymbol{\beta}_{2g}, \boldsymbol{\gamma}_{2g}) \quad (6)\end{aligned}$$

where τ_g is the mixing proportion, i.e. $\sum_{g=1}^{G}\tau_g = 1$ and c_g is the density of a specified copula. This model is estimated via the EM algorithm (Dempster *et al.*, 1977).

The mixtures of copula regressions functions are demonstrated here using a simulated data set called `simdat.mcgr`. The data were simulated based on a mixture of two copula regressions, i.e. covariates were used to estimate the marginal distributions to assist in the mixture of copulas estimation. Details of the sample data simulation process can be found in the package data documentation and Hu & O'Hagan (2021). The true data generating process for the data set `simdat.mcgr` is known, based on which the model can be fitted via:

```
R> data("simdat.mcgr")
R> f1 <- MCGR(copula = list(gumbelCopula(dim=2), frankCopula(dim=2)),
              f1 = y1 ~ x1+x2,
              f2 = y2 ~ x1+x2,
              G=2, gating = "C", data=simdat.mcgr)
R> summary(f1)

-----------------------------------------------------------------------
Mixture of copula regressions with gamma margins fitted by EM algorithm
-----------------------------------------------------------------------

Mixture of gumbelCopula and frankCopula model, two components:

Gating Network Covariates: None
Margin 1 Covariates: x1 + x2
Margin 2 Covariates: x1 + x2

log-likelihood     n df       AIC       BIC
     -3962.677  1000 19  7963.353  8056.601
```

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

```
Clustering table:
  1   2
480 520
```

The estimated coefficients can be extracted via

R> f1$coefficients

```
$beta1
                     g=1          g=2
(Intercept)  0.999389588   0.78177218
x1           0.017843961   0.02627032
x2           0.007871754  -0.05035551

$shape1
    g=1      g=2
6.233493 1.625311

$beta2
                    g=1          g=2
(Intercept)  2.02593159   0.79408397
x1           0.01804302  -0.02776712
x2           0.03273169   0.02989932

$shape2
    g=1      g=2
2.503474 1.646948

$gating
NULL
```

which are close to the true parameter values in the data simulation process.

If G=1, as an alternative, copula regression with gamma GLMs as marginals (no mixtures) can be fitted via, for example,

```
R> f2 <- copreg.gamma(f1 = y1 ~ x1+x2,
                      f2 = y2 ~ x1+x2, copula = gumbel
                      Copula(dim=2),
                      data = simdat.mcgr)
R> summary(f2)

------------------------------------------------------------
Bivariate Copula Regression with Gamma Margins
------------------------------------------------------------

gumbelCopula(1.6177) regression:

    (Intercept)          x1           x2     shape
f1    0.8859739  0.02454751  -0.03093132  2.474138

   (Intercept)           x1           x2     shape
f2    1.654651 -0.009443249 -0.005921523  1.153718


log-likelihood       n df        AIC         BIC
       -4101.2     1000 9     8220.4     8264.57
```

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

4. Conclusion and Remarks

This paper illustrates various functionalities of the R package `mvClaim` targeting joint insurance claims severity modelling, using two approaches: bivariate gamma distributions and bivariate copulas with gamma marginals. By using both methods, not only is the dependence structure taken into account in regression for simultaneously modelling two risks, but different claim patterns can be identified with different dependence structures, targeting the heterogeneity issue.

The package is still undergoing active development and in future further functionality will be added. For example, the current version of the package only focuses on bivariate joint severity modelling, whereas in real-world applications, it may be necessary to consider more risk perils simultaneously. Therefore, higher dimensional cases of the MoE model family using multivariate gamma distributions will form part of the future work. This package does not cover the frequency aspect of joint insurance claims modelling, using bivariate or multivariate Poisson distributions, or copulas targeted for count data. In the literature, there is some existing work in this area and the authors hope to include this in the package over time. Furthermore, this package focuses on joint insurance claims predictions and clustering heterogeneous risks, but does not cover further collective risk analysis using joint cumulative distribution functions, particularly using convolution of bivariate gamma distributions. This will be part of our future work in this area. Finally, to cite the package in publications, please use the citation information given in

```
R> citation("mvClaim")
```

Acknowledgements. This work was supported by the Science Foundation Ireland funded Insight Centre for Data Analytics (SFI/12/RC/2289_P2).

References

Arakelian, V. & Karlis, D. (2014). Clustering dependencies via mixtures of copulas. *Communications in Statistics - Simulation and Computation*, **43**(7), 1644–1661.

Benaglia, T., Chauveau, D., Hunter, D.R. & Young, D. (2009). mixtools: An R package for analyzing finite mixture models. *Journal of Statistical Software*, **32**(6), 1–29.

Bermúdez, L. & Karlis, D. (2012). A finite mixture of bivariate Poisson regression models with an application to insurance ratemaking. *Computational Statistics & Data Analysis*, **56**(12), 3988–3999.

Cheriyan, K. (1941). A bivariate correlated gamma-type distribution function. *Journal of the Indian Mathematical Society*, **5**, 133–144.

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**, 1–38.

Grün, B. & Leisch, F. (2007). Fitting finite mixtures of generalized linear regressions in R. *Computational Statistics & Data Analysis*, **51**(11), 5247–5252.

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.

Hofert, M., Kojadinovic, I., Maechler, M. & Yan, J. (2018). *copula: Multivariate Dependence with Copulas*. R package version 0.999-19.1.

Hu, S., Murphy, T.B. & O'Hagan, A. (2019). Bivariate gamma mixture of experts models for joint insurance claims modelling. To appear; arXiv: arxiv.org/abs/1904.04699.

Hu, S. & O'Hagan, A. (2021). Copula averaging for tail dependence in insurance claims data. To appear; arxiv.org/abs/2103.10912

Joe, H. (1997). *Multivariate Models and Dependence Concepts*. Chapman & Hall/CRC, Boca Raton.

Joe, H. (2014). *Dependence Modeling with Copulas*. Chapman & Hall/CRC, Boca Raton.

Kosmidis, I. & Karlis, D. (2016). Model-based clustering using copulas with applications. *Statistics and Computing*, **26**(5), 1079–1099.

Kraemer, N., Brechmann, E., Silvestrini, D. & Czado, C. (2013). Total loss estimation using copula-based regression models. *Insurance: Mathematics and Economics*, **53**, 829–839.

Masarotto, G. & Varin, C. (2017). Gaussian copula regression in R. *Journal of Statistical Software*, **77**(8), 1–26.

Mathai, A.M. & Moschopoulos, P.G. (1991). On a multivariate gamma. *Journal of Multivariate Analysis*, **39**(1), 135–153.

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press

Murphy, **K. & Murphy**, **T.B.** (2019). `MoEClust`: *Gaussian Parsimonious Clustering Models with Covariates and a Noise Component*. R package version 1.2.2.

Murphy, **K. & Murphy**, **T.B.** (2020). Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*. **14**(2), 293–325

Nelder, **J.A. & Wedderburn**, **R.W.M.** (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, **135**(3), 370–384.

Nelsen, **R.B.** (2007). *An Introduction to Copulas*. Springer, New York.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ramabhadran, **V.** (1951). A multivariate Gamma-type distribution. *Sankhya*, **11**, 45–46.

Scrucca, **L.**, **Fop**, **M.**, **Murphy**, **T.B. & Raftery**, **A.E.** (2016). mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*. **8**(1), 205–233.

Sklar, **A.** (1959). Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut Statistique de l'Université de Paris*, **8**, 229–231.

Wang, **K.**, **Ng**, **A. & McLachlan**, **G.** (2018). *EMMIXskew: The EM Algorithm and Skew Mixture Distribution*. R package version 1.0.3.

Yan, **J.** (2007). Enjoy the joy of copulas: with a package copula. *Journal of Statistical Software*, **21**(4), 1–21.

Cite this article: Hu S, Murphy TB and O'Hagan A (2021). `mvClaim`: an `R` package for multivariate general insurance claims severity modelling, *Annals of Actuarial Science* **15**, 441–457. https://doi.org/10.1017/S1748499521000099

https://doi.org/10.1017/S1748499521000099 Published online by Cambridge University Press