Online Supplements. To support the paper: Dependence Modeling of Multivariate Longitudinal Data with Dropout

Contents

Date: 09 September 2019

1 Online Supplement 1: Copula Regression Insurance Applications

1.1 Dependence and Foundations of Insurance

At a basic level, modeling dependence is critical for insurance. Insurance systems are predicated on the pooling of risks. Insurers pool risks in order to enjoy the benefits of diversification; but, those benefits depend upon relationships among risks.

Standard introductory models assume independence among risks. However, there are also examples where risks are negatively related to one another and so provide a natural hedge. An example of this is mortality risk, where longer than anticipated mortality means additional costs for annuity policies (that pay while someone is alive) yet less costly (on a present value basis) for life insurance that pay when someone dies. More common is positive association among risks, such as the risk of flood to homes that are located close to one another. In these cases, there may be few diversification benefits; for example, flood insurance is not readily available on the private markets in the US because insurers are not able to diversify these risks.

There are many sources of dependencies among risks in insurance. For example, analysts have begun to look at the *joint* effects of several types of claims outcomes. To illustrate, auto insurers cover claims that cover (a) injury to a party other than the insured; (b) damages to the insured (including injury and property damage), and (c) property damage to a party other than the insured. As another example, in homeowners insurance, analysis look at separate distributions for claims due to fire, theft, hail damage, and so forth. In healthcare management, there are several different types of medical expenditures (e.g., office based, inpatient hospital, emergency room, and so forth). In a similar vein, we can think about the dependence between how often claims occur (the frequency) and the claim amount (the severity) as different types of claim outcomes.

Insurers are recognizing the many sources of dependencies among insurance outcomes in today's world where data information is becoming increasingly available,. As noted above, there may be several coverages under a single contract whose outcomes are naturally related. In a similar yet different way, there may be several people listed under a single contract (e.g., in auto or health) with different yet related coverages. Insurers are becoming more customer-focused and wish to understand relationships among the several risks associated with each contract. For example, at the personal level, it is common for a customer to have an auto, home, and umbrella policy with a company. Further, temporal relationships represent a component of dependence modeling that has been long recognized by insurers. Prior claim history can reveal important aspects of an insured's distribution that are not captured in rating variables.

Claims are not the only type of outcome of interest to insurers where dependencies may be important. For example, in this paper, we examine whether or not a policy is renewed with an insurer (or the converse, whether a policyholder lapses).

Insurers not only model risks at a contract level but also over a portfolio, or collection, of contracts. At the portfolio level, insurers understand intuitively the effects of spatial dependence. For example, for homeowners insurance, effects of hurricanes, hail, and earthquakes, can be important sources of dependence. Large commercial risks may be disaggregated geography but also by industry. Also at the portfolio level, risks share a common economic and political environment and so may depend on one another. For example, a portfolio of life insurance contracts share a common interest rate environment.

1.2 Copula Regression Modeling in Insurance

Copula regression modeling is ideally suited for applications where there are many variables available to explain outcomes (the regression portion) and where structural dependence among outcomes is critical (the copula portion). Compared to other multivariate techniques, copulas are particularly suitable in insurance applications because there is a lack of theory to support specification of a dependence structure and data-driven methods, such as copula modeling, fare well.

The literature on copula insurance regression modeling is developing. See Kolev and Paiva (2009) for a survey of the literature up until 2009. This section provides readers with additional background on selected topics.

Multivariate Severity Claim Outcomes

The classic insurance application of copula modeling involves several claim types, all of which are modeled as continuous outcomes. This is now a relatively mature area. To illustrate, X. Yang, Frees, and Zhang (2011) investigate the three outcomes of interest, bodily injury, liability payments and the time-to-settlement, using auto injury data from the Insurance Research Council's Closed Claim Survey. From this survey, many policy, accident, drive, claimant and legal characteristics are available. In this paper, marginal distributions were fit using an exponential GB2 with additive variables and a new copula, a multivariate GB2, was introduced.

For a different type of application involving multivariate continuous claims outcomes, Shi and Frees (2011) examined paid personal and commercial auto claims. However, the unit of observation was not at the claim level but rather the sum of an insurer's claims by year of incurral and development year. The purpose of this modeling is to forecast claims and so explanatory variables used different functions of time. The paper investigates lognormal and gamma marginal distributions as well as a Frank and Gaussian copula to represent contemporaneous associations.

Longitudinal Claim Outcomes

Although (general) insurance contracts tend to be short term, policyholders routinely renew coverage and so it is natural to follow subjects over time, resulting in longitudinal data. To illustrate, Frees and Wang (2005) study automobile bodily injury liability claims from a sample of n=29 Massachusetts towns of six years. They incorporated town characteristics such as per capita income and population per square mile as predictor variables. This paper uses Weibull, lognormal, gamma, for marginal fits and a t-copula for temporal associations. The focus of this paper is on predicting claims known as "credibility" in the insurance literature.

Another example of longitudinal data is that of Sun, Frees, and Rosenberg (2008) who study the occupancy rate of Wisconsin nursing home facilities. A novelty of this paper is that standard marginal distributions did not well fit this rate and so a generalized beta of a second kind, GB2, was introduced into the copula literature. There were many characteristics of nursing home facilities, such as number of beds and organizational structure, available for modeling. The paper studied 377 facilities observed over six years and used a t-copula with temporal associations.

Multivariate Frequency Severity Claim Outcomes

For models of frequency and severity, there have been fewer copula applications because of limitations imposed by the discreteness in frequency. To illustrate, Frees, Meyers, and Cummings (2010) studied nine types of homeowners claims (e.g., wind, water damage, theft, and so forth). Each claim type was decomposed into a zero-one frequency and severity component. Many policyholder characteristics were available. The marginal models consisted of logistic regressions for the frequency and gamma regressions for the severity. Gaussian copulas used for severity but dependencies among frequency were modeled using classic dependency ratio methods for multivariate binary outcomes, not copulas. As another example of this approach, Frees, Jin, and Lin (2013) studied five types of medical expenditures (e.g, inpatient hospital, emergency room and so forth).

Papers utilizing copulas for the frequency portion are only starting to emerge. Czado et al. (2012) fit a Gaussian copula to the number and average claim size for 12,850 claims from auto policies. For marginal distributions, they used a Poisson model for frequencies and gamma for severities. In this paper, only claims are analyzed and so they restricted the number to be greater than zero.

A longitudinal, multivariate, frequency severity model was fit in a recent paper by Frees, Lee, and Yang (2016). This paper studied approximately 1,000 local government entities are observed over five years. The outcomes of interest are property damage, auto collision and comprehensive, and contractor's equipment and characteristics of the entities were used as predictor variables. Zeroes in the frequency are retained and were model with zero/one inflated Poisson and negative binomial distributions. A GB2 was used for severity and a Tweedie distribution for the combined frequency-severity. Gaussian and *t*-copulas are used for the dependence.

Multivariate Frequency Outcomes

The review of Nikoloulopoulos (2013) well summarizes this application area.

2 Online Supplement 2: Multivariate Gaussian Copula Details

Although properties of the multivariate Gaussian copula has been developed extensively, it is worthwhile to collect facts needed for this paper in this supplement. Section 2.1 is largely drawn from Joe (2014). Section 2.2 contains new results on derivatives with respect to association parameters.

Consider a d dimensional multivariate normal distribution with variance-covariance matrix $\boldsymbol{\Sigma}$. As we will use this as a basis for defining copulas, consider the mean to be zero and variance to be 1 so that the diagonal elements of $\boldsymbol{\Sigma}$ equal 1. Let $\Phi_d(\cdot;\boldsymbol{\Sigma})$ be the corresponding distribution function. With this, the Gaussian copula can be expressed as

$$C\left(u_1,\ldots,u_d\right)=\Phi_d\left(z_1,\ldots,z_d;\boldsymbol{\Sigma}\right).$$

In this expression, we use the normal scores defined as $z_j=\Phi^{-1}(u_j), j=1,\ldots,d$.

2.1 Copula Distribution Function Derivatives

We wish to calculate partial derivatives of the copula distribution function. For elliptical copulas, it is natural to relate them to conditional copulas using

$\partial_{u_{d-k+1}\cdots u_d}C(u_1,\ldots u_d)=\frac{\partial^k}{\partial u_{d-k+1}\cdots\partial u_d}C(u_1,\ldots u_d)$

$$\begin{aligned}
&= \int^{u_1}\cdots\int^{u_{d-k}} c\left(z_1,\ldots,z_{d-k},u_{d-k+1},\ldots,u_d\right)dz_1\cdots dz_{d-k}\\
&= \int^{u_1}\cdots\int^{u_{d-k}} c\left(z_1,\ldots,z_{d-k}|u_{d-k+1},\ldots,u_d\right)c\left(u_{d-k+1},\ldots,u_d\right)dz_1\cdots dz_{d-k}\\
&= C\left(u_1,\ldots,u_{d-k}|u_{d-k+1},\ldots,u_d\right)c\left(u_{d-k+1},\ldots,u_d\right). \qquad (1)
\end{aligned}$$

It is convenient to partition the association matrix as

$$\boldsymbol{\Sigma}=\left(\begin{array}{cc}\boldsymbol{\Sigma}_{1:d-k,1:d-k} & \boldsymbol{\Sigma}_{1:d-k,d-k+1:d}\\ \boldsymbol{\Sigma}'_{1:d-k,d-k+1:d} & \boldsymbol{\Sigma}_{d-k+1:d,d-k+1:d}\end{array}\right),$$

so that $\boldsymbol{\Sigma}_{1:d-k,1:d-k}$ is the submatrix for the first $d-k$ elements and similarly for the other entries.

With this, for the Gaussian copula, we have

$C\left(u_1,\ldots,u_{d-k}|u_{d-k+1},\ldots,u_d\right)$

$$\begin{aligned}
&= \Pr\left(\Phi(N_1)\le u_1,\ldots,\Phi(N_{d-k})\le u_{d-k}|\Phi(N_{d-k+1})=u_{d-k+1},\ldots,\Phi(N_d)=u_d\right)\\
&= \Pr\left(N_1\le\Phi^{-1}(u_1),\ldots,N_{d-k}\le\Phi^{-1}(u_{d-k})|N_{d-k+1}=\Phi^{-1}(u_{d-k+1}),\ldots,N_d=\Phi^{-1}(u_d)\right)\\
&= \Phi_{d-k}\left(z_1-\mu_{1\cdot2,1},\ldots,z_{d-k}-\mu_{1\cdot2,d-k};\boldsymbol{\Sigma}_{11\cdot2}\right), \qquad (2)
\end{aligned}$$

Here, $\mu_{1\cdot2,j}$ is the jth component of

$$\boldsymbol{\mu}_{1\cdot2}=\boldsymbol{\Sigma}_{1:d-k,d-k+1:d}\boldsymbol{\Sigma}^{-1}_{d-k+1:d,d-k+1:d}\left(\begin{array}{c}z_{d-k+1}\\ \vdots\\ z_d\end{array}\right).$$

Further, $\Phi_{d-k}(\cdot;\boldsymbol{\Sigma}_{11\cdot2})$ is a $d-k$ dimensional multivariate normal distribution function with mean zero and variance-covariance matrix

$$\boldsymbol{\Sigma}_{11\cdot2}=\boldsymbol{\Sigma}_{1:d-k,1:d-k}-\boldsymbol{\Sigma}_{1:d-k,d-k+1:d}\boldsymbol{\Sigma}^{-1}_{d-k+1:d,d-k+1:d}\boldsymbol{\Sigma}'_{1:d-k,d-k+1:d}.$$

2.2 Derivatives for the Conditional Density/Mass Function

For the conditional hybrid probability density/mass function in equation (**??**), we need the case where $k = 1$

$$\begin{aligned} C_d\left(u_1,\ldots,u_d\right) &= \frac{\partial}{\partial u_d} C(u_1,\ldots,u_d) = C\left(u_1,\ldots,u_{d-1}|u_d\right) \\ &= \Phi_{d-1}\left(z_1 - \mu_{\{1,\ldots,d-1\}\cdot d,1},\ldots,z_{d-1}-\mu_{\{1,\ldots,d-1\}\cdot d,d-1};\boldsymbol{\Sigma}_{\{1,\ldots,d-1\}\cdot d}\right), \end{aligned} \tag{3}$$

where

$$\boldsymbol{\mu}_{\{1,\ldots,d-1\}\cdot d} = \boldsymbol{\Sigma}_{1:d-1,d:d}\ z_d$$

and

$$\boldsymbol{\Sigma}_{\{1,\ldots,d-1\}\cdot d} = \boldsymbol{\Sigma}_{1:d-1,1:d-1} - \boldsymbol{\Sigma}_{1:d-1,d:d}\boldsymbol{\Sigma}'_{1:d-1,d:d}.$$

In the same way, for $k = 2$, we have

$$\begin{aligned} &C_{d-1,d}\left(u_1,\ldots,u_d\right) = \frac{\partial^2}{\partial u_{d-1}\partial u_d} C(u_1,\ldots,u_d) \\ &= C\left(u_1,\ldots,u_{d-2}|u_{d-1},u_d\right)c(u_{d-1},u_d) \\ &= \Phi_{d-2}\left(z_1-\mu_{\{1,\ldots,d-2\}\cdot\{d-1,d\},1},\ldots,z_{d-2}-\mu_{\{1,\ldots,d-2\}\cdot\{d-1,d\},d-2};\boldsymbol{\Sigma}_{\{1,\ldots,d-2\}\cdot\{d-1,d\}}\right)c(u_{d-1},u_d), \end{aligned} \tag{4}$$

where

$$\boldsymbol{\mu}_{\{1,\ldots,d-2\}\cdot\{d-1,d\}} = \boldsymbol{\Sigma}_{1:d-2,d-1:d}\boldsymbol{\Sigma}^{-1}_{d-1:d,d-1:d}\begin{pmatrix} z_{d-1} \\ z_d \end{pmatrix}$$

and

$$\boldsymbol{\Sigma}_{\{1,\ldots,d-2\}\cdot\{d-1,d\}} = \boldsymbol{\Sigma}_{1:d-2,1:d-2} - \boldsymbol{\Sigma}_{1:d-2,d-1:d}\boldsymbol{\Sigma}^{-1}_{d-1:d,d-1:d}\boldsymbol{\Sigma}'_{1:d-2,d-1:d}.$$

Trivariate Gaussian Copula Distribution Functions

To illustrate, consider the trivariate case with $d = 3$. For simplicity, we use the following generic expression for the association matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{pmatrix}.$$

For a derivative with respect to one argument, we have

$$C_3\left(u_1,u_2,u_3\right) = C\left(u_1,u_2|u_3\right) = \Phi_2\left(z_1-\mu_{12\cdot 3,1}, z_2-\mu_{12\cdot 3,2};\boldsymbol{\Sigma}_{12\cdot 3}\right)$$

where

$$\boldsymbol{\mu}_{12\cdot 3} = \begin{pmatrix} \mu_{12\cdot 3,1} \\ \mu_{12\cdot 3,2} \end{pmatrix} = z_3\begin{pmatrix} \rho_{13} \\ \rho_{23} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_{12\cdot 3} = \begin{pmatrix} 1 & \rho_{12} \\ \rho_{12} & 1 \end{pmatrix} - \begin{pmatrix} \rho_{13} \\ \rho_{23} \end{pmatrix}\begin{pmatrix} \rho_{13} & \rho_{23} \end{pmatrix}.$$

For derivatives with respect to two arguments, we have

$$C_{23}\left(u_1,u_2,u_3\right) = C\left(u_1|u_2,u_3\right)c\left(u_2,u_3\right) \quad \text{with} \quad C\left(u_1|u_2,u_3\right) = \Phi\left(z_1-\mu_{1\cdot 23};\sigma_{1\cdot 23}\right),$$

where

$$\mu_{1\cdot 23} = \begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix} \quad \text{and} \quad \sigma_{1\cdot 23} = 1 - \begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix}.$$

3 Online Supplement 3. Simulation of Multivariate Tweedie

3.1 Background

This file documents the simulation study of the GMM estimation section of the paper *Joint Models of Insurance Lapsation and Claims*. A **.pdf** version provides a hard copy, the **.html** version allows one to hide `R` code and the **.Rmd** version allows one to run the simulation, changing input parameters as desired. (To run the **.Rmd**, search on **eval=FALSE** and change to **eval=TRUE**.)

Simulation Input Parameters

```r
MeanClaim <- 1000
nSim <- 2
# Association (over years) parameter, Autoregressive of order one
rhofVec <- c(0.6,0.3)
#rhofVec <- c(.6)
# Tweedie dispersion parameter # With a mean=1000, phi=500 for 94% zeros
# phi <- 2 for near continuous data, phi=42 for data for about half zeros
#ExternalphiVec <- c(2,42,500)
ExternalphiVec <- c(500)
# Number of policyholders
#NsampVec <- c(100,250)
NsampVec <- c(100)
p <- 5                     # Number of years # DO NOT CHANGE
```

R Packages Needed to Run this Simulation

```r
# Here are the packages that you need to install to run this simulation
library(tweedie)
library(reshape)
library(statmod)
library(knitr)
library(BB)
library(MASS)
library(copula)
library(numDeriv)
library(VineCopula)
library(mvtnorm)
time0 <- Sys.time()  -> time1 # define terms to check the run time
```

3.2 Model Specification

Marginal Outcome Model

We represent the marginal distributions of the claims random variables using a Tweedie distribution so that the distributions have a mass at zero and are otherwise positive. For each claim variable, we use a logarithmic link to form the mean claims of the form

$$\mu_{it} = \exp\left(\mathbf{x}_{it}^{\prime}\boldsymbol{\beta}\right).$$

Each claim is simulated using the Tweedie distribution, a mean, and two other parameters, ϕ (for dispersion) and P (the *power* parameter).

Recall, for a Tweedie distribution, that the variance is $\phi_j \mu^P$. For this simulation study, we use $P = 1.67$ based on our experiences analyzing real insurance data sets. In the Tweedie model, the probability of a zero claim is $e^{-\lambda}$, where $\lambda = \mu^{2-P}/(\phi * (2-P))$.

The regression coefficients in the vector $\boldsymbol{\beta}$ were set so that the average mean was approximately $\mu = 1000$. With this, the probability of a zero claim is $\exp\left[-1000^{0.33}/(\phi * 0.33)\right]$. For example, by selecting $\phi = 42$, the probability of a zero claim is 49.4%, or about half zeros. In the same way, the choice of $\phi = 2$ represents almost no zeros (continuous data) and $\phi = 500$ represents 94% zeros (common in personal insurance lines of business). So, if we use $\mu = 1000$, then the probability of a zero claim is $\exp\left[-1000^{0.33}/(\phi * 0.33)\right]$. For example, by selecting $\phi = 42$, the probability of a zero claim is 49.4%.

In addition to the claims, we have two rating (explanatory) variables:

- x_1 a binary variable that takes on values 1 or 2 depending on whether or not an attribute holds, and
- x_2 a generic continuous explanatory variable.

R Code for Generating Covariates

```
# Generate covariates and means
# Time constant Bernoulli variable
Generate_Covariates <- function(Nsamp) {
  x11   <- 1+rbinom(Nsamp, size=1, prob=0.4)
  x1    <- cbind(x11,x11,x11,x11,x11)
  x2    <- matrix(1+(rnorm(p*Nsamp)^2/10),nrow=Nsamp,ncol=p)
  beta1 <- 2
  beta2 <- 0.3
  mu1   <- exp((beta1*x11+beta2*x2)/2)
  mu    <- MeanClaim*mu1/mean(mu1) # Rescale so that the mean claim is MeanClaim
  Z <- cbind(x1,x2,mu)
  return(Z)
  }
```

Dependence Model

Claims are associated using a Gaussian copula with an autoregressive of order 1 ($AR1$) pattern determined by the autocorrelation parameter ρ. With $p = 5$ time series replications, the association matrix is

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho & \rho^2 & \rho^3 & \rho^4 \\ \rho & 1 & \rho & \rho^2 & \rho^3 \\ \rho^2 & \rho & 1 & \rho & \rho^2 \\ \rho^3 & \rho^2 & \rho & 1 & \rho \\ \rho^4 & \rho^3 & \rho^2 & \rho & 1 \end{pmatrix}.$$

R Code for Simulating Dependent Outcomes

```
Generate_SampleData <- function(rhof,Nsamp,Externalphi,Z) {
  # AR(1) Dependence Structure
  BigSigma <- matrix(c(1,rhof^1,rhof^2,rhof^3,rhof^4,
                    rhof^1,1,rhof^1,rhof^2,rhof^3,
                    rhof^2,rhof^1,1,rhof^1,rhof^2,
                    rhof^3,rhof^2,rhof^1,1,rhof^1,
                    rhof^4,rhof^3,rhof^2,rhof^1,1),nrow=5,ncol=5)
  BigSigma <- chol(BigSigma)
  # Start with dependent multivariate Gaussian
  Z1      <- matrix(rnorm(p*Nsamp),nrow=Nsamp,ncol=p)%*%BigSigma
  UCop    <- pnorm(Z1)
  muVec   <- as.vector(matrix(Z[,11:15],nrow=Nsamp*p,ncol=1))
  UCopVec <- as.vector(matrix(UCop,nrow=Nsamp*p,ncol=1))
  # Simulate Tweedie claims
  Claims  <- qtweedie(UCopVec, power=Externalxi, mu=muVec, phi=Externalphi)
  yearMat <- t(matrix(rep(1:p,Nsamp),nrow=p,ncol=Nsamp))
  PolIDMat<- t(matrix(rep(1:Nsamp,each=p),nrow=p,ncol=Nsamp))
  year    <- rep(1:p,Nsamp)
  PolID   <- rep(1:Nsamp, each=p)
  SampleData <- as.data.frame(cbind(
    Claims,
    matrix(yearMat, nrow=Nsamp*p,ncol=1),
    matrix(PolIDMat,nrow=Nsamp*p,ncol=1),
    matrix(Z[,1:1], nrow=Nsamp*p,ncol=1),
    matrix(Z[,6:10],nrow=Nsamp*p,ncol=1),
    matrix(muVec,   nrow=Nsamp*p,ncol=1) ))
  colnames(SampleData) <- c("Claims","year","PolID","x1","x2","mu")
  SampleData <- SampleData[order(SampleData$PolID,SampleData$year),]
  return(SampleData)
  }
```

3.3 Tweedie (Claims) Regression Estimation

The Tweedie is commonly used in insurance applications for claims. In part, this is because it can be expressed as a generalized linear model. In the following illustrative code, we have skipped the determination of the *power* parameter ($P = 1.67$ for us).

R Code for Tweedie Regression Estimation

```
fit_MargRegress <- function(SampleData) {
    phiAssumed <- Externalphi  # Use this if the tweedie regression does not converge
    SampleData$fitted <- SampleData$mu  # Use this if the tweedie regression does not converge
    Z2 <- data.frame(y = SampleData$Claims, x1 = SampleData$x1, x2 = SampleData$x2)
    tryCatch({
        tweedie.fit <- glm(y ~ ., data = Z2, control = glm.control(maxit = 500),
            family = tweedie(var.power = Externalxi, link.power = 0))
        phiAssumed <- summary(tweedie.fit)$dis
        SampleData$fitted <- tweedie.fit$fitted.values
    }, error = function(err) {
        print(paste("Tweedie did not converge:", err))
        NonConvergeTweedie <- NonConvergeTweedie + 1
        return(NonConvergeTweedie)
    })
    # Probability Integral Transform
    dfTweedieA <- ptweedie(SampleData$Claims, xi = Externalxi, mu = SampleData$fitted,
        phi = phiAssumed)
    SampleData$dfTweedie <- pmin(pmax(1e-05, dfTweedieA), 0.99999)
    return(SampleData)
}
```

3.4 Joint Model Specification

For more background, see the Appendix Section on **Hybrid Distributions**.

Tweedie Likelihoods

In insurance, it is common to refer to a random variable with a mass at zero and a continuous density over the positive reals as a Tweedie random variable (corresponding to a Tweedie distribution). Suppose that both Y_j and Y_k are Tweedie random variables. The joint distribution function has a hybrid probability density/mass function of the form:

$$f_{jk}(y_j, y_k) = \begin{cases} \Pr(Y_j = 0, Y_k = 0) = F_{jk}(0,0) & y_j = 0, y_k = 0 \\ C_1\left(F_j(y_j), F_k(0)\right) f_j(y_j) & y_j > 0, y_k = 0 \\ C_2\left(F_j(0), F_k(y_k)\right) f_k(y_k) & y_j = 0, y_k > 0 \\ c\left(F_j(y_j), F_k(y_k)\right) f_j(y_j) f_k(y_k) & y_j > 0, y_k > 0. \end{cases}$$

To illustrate, when both observations are 0, then the likelihood $F_{jk}(0,0) = C\left(F_j(0), F_k(0)\right)$ requires evaluation of the distribution function C. With a Gaussian copula, this is a two-dimensional integral.

R Code for Pairwise Likelihood Functions

```
# These are the four cases from the hybrid joint mass/density function
Create_Data_Subsets <- function(SampleData) {
    # Reshape the data
    TweedieLike1 <- SampleData[order(-SampleData$PolID, SampleData$year), ]
    VarsLike <- c("PolID", "year", "Claims", "dfTweedie")
    TweedieLike <- TweedieLike1[VarsLike]
    TweedieLike2 <- melt(TweedieLike, id = c("PolID", "year"), measured = c("Claims",
        "dfTweedie"))
    TweedieLike3 <- (cast(TweedieLike2, PolID ~ variable ~ year))
    calcOrder <- 1:length(TweedieLike3[, 1, 1])
    amd1 <- NA
    amd2 <- NA
    amd3 <- NA
    amd4 <- NA
    tcount <- 0
    for (t1 in 1:4) {
        for (t2 in (t1 + 1):5) {
            tcount <- tcount + 1
            caset1t2 <- 1 * (TweedieLike3[, 1, t1] == 0) * (TweedieLike3[, 1,
                t2] == 0) + 2 * (TweedieLike3[, 1, t1] > 0) * (TweedieLike3[,
                1, t2] == 0) + 3 * (TweedieLike3[, 1, t1] == 0) * (TweedieLike3[,
                1, t2] > 0) + 4 * (TweedieLike3[, 1, t1] > 0) * (TweedieLike3[,
                1, t2] > 0)
            u <- cbind(TweedieLike3[, 2, t1], TweedieLike3[, 2, t2])
            zu <- qnorm(u)
            mydata <- data.frame(caset1t2, u, zu, calcOrder)
            names(mydata) <- c("caset1t2", "u1", "u2", "zu1", "zu2", "calcOrder")
            mydata1 <- mydata[which(caset1t2 == 1), ]
            mydata2 <- mydata[which(caset1t2 == 2), ]
            mydata3 <- mydata[which(caset1t2 == 3), ]
            mydata4 <- mydata[which(caset1t2 == 4), ]
            amd1 <- c(amd1, list(mydata1))
            amd2 <- c(amd2, list(mydata2))
            amd3 <- c(amd3, list(mydata3))
            amd4 <- c(amd4, list(mydata4))
        }
    }
    totalallmydata <- list(amd1[-1], amd2[-1], amd3[-1], amd4[-1])
    return(totalallmydata)
}
```

```
PairLikeTime <- function(t1,t2,rhos) {
  rhos   <- pmin(pmax(-.99,rhos),.99)
  sigma  <- matrix(c(1,rhos,rhos,1),nrow=2,ncol=2)
  likehd <- 0*calcOrder
  tcount <- (t1==1)*(t2-1) + (t1==2)*(t2+2) + (t1==3)*(t2+4) + (t1==4)*10
  mydata1 <- totalallmydata[[1]][[tcount]]
  mydata2 <- totalallmydata[[2]][[tcount]]
  mydata3 <- totalallmydata[[3]][[tcount]]
  mydata4 <- totalallmydata[[4]][[tcount]]
# See the VineCopula package for the functions 'BiCopCDF', 'BiCopHfunc', and 'BiCopPDF'
  if (nrow(mydata1)>0) {likehd[mydata1$calcOrder] <-
    BiCopCDF(mydata1$u1,mydata1$u2, family=1, par=rhos)
  }
  if (nrow(mydata2)>0) {likehd[mydata2$calcOrder] <-
    BiCopHfunc1(mydata2$u1,mydata2$u2, family=1, par=rhos)
  }
  if (nrow(mydata3)>0) {likehd[mydata3$calcOrder] <-
    BiCopHfunc2(mydata3$u1,mydata3$u2, family=1, par=rhos)
  }
  if (nrow(mydata4)>0) {likehd[mydata4$calcOrder] <-
    BiCopPDF(mydata4$u1,mydata4$u2, family=1, par=rhos)
  }
  return(log(likehd))
}
```

More Pairwise Functions

```
PairLikeSum <- function(rhos) {
  LikelihoodSum <- 0
  for (t1 in 1:4) {
    for (t2 in (t1+1):5) {rhoAR1 <- rhos^(abs(t2-t1))
      LikelihoodSum <- LikelihoodSum + sum(PairLikeTime(t1,t2,rhoAR1))
      } }
  return(-LikelihoodSum)
}

PLogLikelihood <- function(rhos) {
  VecLogLike <- NA
  for (t1 in 1:4)
    {for (t2 in (t1+1):5) {rhoAR1 <- rhos^(abs(t2-t1))
      VecLogLike <- cbind(VecLogLike,PairLikeTime(t1,t2,rhoAR1))
      } }
  return(VecLogLike[,-1])
}
```

Evaluation of GMM Scores

For a recursive method such as the *GMM*, one needs starting values for the recursion. We will use the likelihood estimators developed in Section 3.1. With the initial estimator, we can now calculate the *GMM* score function. This allows us to minimize this function in order to get our *GMM* estimator, with an asymptotic variance.

We now evaluate the scores. For two zero outcomes, the score can be expressed as

$$g_{\theta,jk}(0,0) = \partial_\theta \ln f_{jk}(0,0) = \frac{\partial_\theta\{C(F_j(0),F_k(0))\}}{C(F_j(0),F_k(0))}.$$

For a zero and a positive $y_k > 0$ outcome, we have

$$g_{\theta,jk}(0,y_k) = \partial_\theta \ln\left[\{C_2\left(F_j(0),F_k(y_k)\right)\} f_k(y_k)\right] = \frac{\partial_\theta\{C_2(F_j(0),F_k(y_k))\}}{C_2(F_j(0),F_k(y_k))}$$

For two positive outcomes, y_j and y_k, we have

$$g_{\theta,jk}(y_j,y_k) \quad = \partial_\theta \ln\left[c(F_{Y_j}(y_j),F_{Y_k}(y_k))f_j(y_j)f_k(y_k)\right] = \frac{\partial_\theta \;\, c(F_{Y_j}(y_j),F_{Y_k}(y_k))}{c(F_{Y_j}(y_j),F_{Y_k}(y_k))}.$$

R Code for GMM Functions

```r
# GMM Functions
scoreTime <- function(t1,t2,rhos) {
  rhos  <- pmin(pmax(-.99,rhos),.99)
  sigma <- matrix(c(1,rhos,rhos,1),nrow=2,ncol=2)
  score <- 0*calcOrder
  tcount  <- (t1==1)*(t2-1) + (t1==2)*(t2+2) + (t1==3)*(t2+4) + (t1==4)*10
  mydata1 <- totalallmydata[[1]][[tcount]]
  mydata2 <- totalallmydata[[2]][[tcount]]
  mydata3 <- totalallmydata[[3]][[tcount]]
  mydata4 <- totalallmydata[[4]][[tcount]]
  if (nrow(mydata1)>0) {score[mydata1$calcOrder] =
    dmvnorm(cbind(mydata1$zu1,mydata1$zu2), mean=rep(0, 2), sigma=sigma, log=FALSE) /
    BiCopCDF(mydata1$u1,mydata1$u2, family=1, par=rhos)
  }
  if (nrow(mydata2)>0) {score[mydata2$calcOrder] <-
    BiCopHfuncDeriv(mydata2$u2,mydata2$u1, family=1, par=rhos, deriv="par")  /
    BiCopHfunc1(mydata2$u1,mydata2$u2, family=1, par=rhos)
  }
  if (nrow(mydata3)>0) {score[mydata3$calcOrder] <-
    BiCopHfuncDeriv(mydata3$u1,mydata3$u2, family=1, par=rhos, deriv="par")  /
    BiCopHfunc2(mydata3$u1,mydata3$u2, family=1, par=rhos)
  }
  if (nrow(mydata4)>0) {score[mydata4$calcOrder] <-
    BiCopDeriv(mydata4$u1,mydata4$u2, family=1, par=rhos, deriv="par", log=FALSE) /
    BiCopPDF(mydata4$u1,mydata4$u2, family=1, par=rhos)
  }
  return(score)
}
```

More GMM Functions

Although g_θ is a mean zero vector containing information about θ, the number of elements in g_θ exceeds the number of parameters and so we use GMM to estimate the parameters. Specifically, the GMM estimator of θ, say θ_{GMM}, is the minimizer of the expression $g_\theta \left(\mathrm{Var}\ g_{\hat{\theta}}\right)^{-1} g'_\theta$. To implement this, we use the plug-in estimator of the variance

$$\widehat{\mathrm{Var}}\ g_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^{n} g_{\hat{\theta},i}(Y_{i1}, \ldots, Y_{ip})\ g_{\hat{\theta},i}(Y_{i1}, \ldots, Y_{ip})'.$$

For this example, there are $\binom{5}{2} = 10$ different scores, so that the dimensions of $\widehat{\mathrm{Var}}\ g_{\hat{\theta}}$ is 10×10. Because this can be unstable for small samples, we also combine the scores in some fashion.

```
scoreTimeVec <- function(rhos) {
  tcount <- 0
  ScoreMat <- NA
  for (t1 in 1:4) {
    for (t2 in (t1+1):5) {tcount <- tcount+1
      rhoAR1   <- rhos^(abs(t2-t1))
      temp     <- scoreTime(t1,t2,rhoAR1)*abs(t2-t1)*rhos^(abs(t2-t1-1))
      ScoreMat <- cbind(ScoreMat,temp)
    } }
  return(ScoreMat[,-1])
}
GMMFunc <- function(rhos) {temp <- colSums(scoreTimeVec(rhos))
  t(temp) %*% VarhatInv %*% temp
}

scoreTimeVecA <- function(rhos) {
  # Treats Lag One as important, groups others
  tcount <- 0
  ScoreMat <- NA
  for (t1 in 1:4) {
    for (t2 in (t1+1):5) {tcount <- tcount+1
      rhoAR1   <- rhos^(abs(t2-t1))
      temp     <- scoreTime(t1,t2,rhoAR1)*abs(t2-t1)*rhos^(abs(t2-t1-1))
      ScoreMat <- cbind(ScoreMat,temp)
    } }
    ScoreMatA <- ScoreMat[,-1]
    ScoreMatB <- ScoreMatA[,c(1,5,8,10)]
    ScoreMatC <- ScoreMatA[,2]+ScoreMatA[,3]+ScoreMatA[,4]+
                 ScoreMatA[,6]+ScoreMatA[,7]+ScoreMatA[,9]
  return(cbind(ScoreMatB,ScoreMatC))
}

GMMFuncA <- function(rhos) {temp <- colSums(scoreTimeVecA(rhos))
  t(temp) %*% VarhatInvA %*% temp
}
```

R Code for the Simulation Loop

This code produces the simulation results. As noted above, to run the **.Rmd**, change **eval=FALSE** to **eval=TRUE**.

```
time1 <- Sys.time()
set.seed(123457)
NumRuns <- length(rhofVec) * length(ExternalphiVec) * length(NsampVec)
OverResults <- matrix(0, nrow = NumRuns, ncol = 11)
colnames(OverResults) <- c("NumSim", "NumSamp", "phi", "rho", "PairBias", "PairSqRootMSE",
    "PairAvgSE", "GMMBias", "GMMSqRootMSE", "GMMAvgSE", "TimeTaken")
ResultsSim <- matrix(0, nrow = nSim, ncol = 4)
iResultCount <- 0
NonConvergeTweedie <- 0

for (iNsamp in 1:length(NsampVec)) {
    for (iExternalphi in 1:length(ExternalphiVec)) {
        for (irhof in 1:length(rhofVec)) {
            iResultCount <- iResultCount + 1
            OverResults[iResultCount, 1] <- nSim
            OverResults[iResultCount, 2] <- Nsamp <- NsampVec[iNsamp]
            OverResults[iResultCount, 3] <- Externalphi <- ExternalphiVec[iExternalphi]
            OverResults[iResultCount, 4] <- rhof <- rhofVec[irhof]

            # Start Simulation Loop
            for (iSim in 1:nSim) {
                # Generate Covariates
                Z <- Generate_Covariates(Nsamp)
                Externalxi <- 1.67  # Tweedie power parameter
                # Generate Data
                SampleData <- Generate_SampleData(rhof = rhof, Nsamp = Nsamp,
                  Externalphi = Externalphi, Z)
                # Fit Regression
                SampleData <- fit_MargRegress(SampleData)
                # Reshape the data
                SampleData <- SampleData[order(-SampleData$PolID, SampleData$year),
                  ]
                calcOrder <- 1:(length(SampleData$PolID)/p)
                totalallmydata <- Create_Data_Subsets(SampleData)
                # Pairwise parameter estimate
                PLikeResult <- optim(par = 0, fn = PairLikeSum, method = c("L-BFGS-B"),
                  control = list(factr = 10^10))
                PLikeEstimate <- ResultsSim[iSim, 1] <- PLikeResult$par
                gradient <- jacobian(func = PLogLikelihood, PLikeResult$par,
                  method = "simple", method.args = list(eps = 0.005))
                PLstderror <- ResultsSim[iSim, 2] <- 1/sqrt(sum(gradient^2))

                # GMM parameter estimate
                GHatA <- scoreTimeVecA(PLikeEstimate)
                VarhatInvA <- ginv(t(GHatA) %*% GHatA)
                GMMResult <- optim(par = PLikeEstimate, fn = GMMFuncA, method = c("L-BFGS-B"),
                  control = list(factr = 10^10))
                FinalEstimate <- ResultsSim[iSim, 3] <- GMMResult$par
                gradient <- jacobian(func = scoreTimeVecA, FinalEstimate, method = "simple",
                  method.args = list(eps = 0.005))
                GMMstderror <- ResultsSim[iSim, 4] <- 1/sqrt(sum(gradient^2))


            }  # This finishes the simulation loop

            # round(ResultsSim,digits=3)
            AverResults <- colMeans(ResultsSim)
            VarResults <- colMeans(ResultsSim * ResultsSim) - (colMeans(ResultsSim))^2
            OverResults[iResultCount, 5] <- AverResults[1] - rhof
            OverResults[iResultCount, 6] <- sqrt(VarResults[1] + OverResults[iResultCount,
                5]^2)/sqrt(nSim)
            OverResults[iResultCount, 7] <- AverResults[2]
            OverResults[iResultCount, 8] <- AverResults[3] - rhof
            OverResults[iResultCount, 9] <- sqrt(VarResults[3] + OverResults[iResultCount,
                8]^2)/sqrt(nSim)
            OverResults[iResultCount, 10] <- AverResults[4]
            OverResults[iResultCount, 11] <- difftime(Sys.time(), time1, units = "mins")
            time1 <- Sys.time()
            write.csv(OverResults, "GMMSimResultsNewScoresApril2018c.csv", row.names = F)
        }
    }
}
```

```r
round(OverResults, digits = 4)
OverResultsTemp <- OverResults

# Number of Non convergence for Tweedie Fits
NonConvergeTweedie
```

3.5 Simulation Results

We ran this program several times and collect results in the following.

Basic Output

The following table compares pairwise likelihood and GMM estimators by different choices of the sample size n, the proportion of zeros through the parameter ϕ, and the autocorrelation parameter ρ.The study is based on 500 simulations. The *Bias* gives the average estimator centered about the true parameter ($\theta = \rho$). The $\sqrt{MSE}$ is the average squared deviation of the simulated estimate from the true parameter divided by the square root of the number of simulations. These statistics indicate that both procedures do well, on average, and that the simulation size is sufficient for demonstration purposes. It is not surprising that the quality of the estimators decreases as the proportion of zeros increase, e.g., the largest biases (in absolute value) occur for $\phi = 500$, corresponding to approximately 94% zeros. It is also not surprising that the quality of the estimators increases as the sample size increases from $n = 100$ to $n = 250$.

The table also provides $AvgSE$, the average asymptotic standard error of the estimators. The squared ratio of $AvgSE$ for the pairwise likelihood and GMM estimators gives the *Ratio Var* column, a measure of relative estimator efficiency. All values of this column exceed one indicating that, as anticipated, the GMM estimator has a lower standard error than the pairwise estimator. In this sense it is more efficient. We also calculated the corresponsding mean square erro, given in the *Ratio MSE* column, as the ratio of the bias squared plus the standard error squared of each estimator. In these small samples, we see that the *GMM* does not always outperform the pairwise estimator, particularly for data with more discreteness (as ϕ becomes larger) and as the association parameter ρ increases. We address these instances in the next two subsections.

Num	n	ϕ	ρ	**Pair**	**Pair**	**Pair**	**GMM**	**GMM**	**GMM**	**Time**	**Ratio**	**Ratio**
Sim				**Bias**	$\sqrt{MSE}$	**AvgSE**	**Bias**	$\sqrt{MSE}$	**AvgSE**	**Taken**	**Var**	**MSE**
500	100	2	-0.3	-0.002	0.002	0.040	0.033	0.003	0.014	23.87	8.32	1.21
500	100	2	0	-0.008	0.002	0.050	-0.004	0.001	0.018	25.39	7.93	7.70
500	100	2	0.3	0.002	0.002	0.039	-0.031	0.003	0.013	24.01	8.50	1.37
500	100	2	0.6	-0.009	0.002	0.021	-0.014	0.002	0.004	24.93	24.06	2.49
500	100	42	-0.3	0.009	0.003	0.050	0.060	0.004	0.021	26.95	5.93	0.65
500	100	42	0	-0.004	0.003	0.060	-0.002	0.001	0.025	32.70	5.68	5.66
500	100	42	0.3	-0.008	0.003	0.049	-0.058	0.004	0.021	28.24	5.58	0.64
500	100	42	0.6	-0.014	0.003	0.028	-0.032	0.003	0.007	36.13	13.81	0.89
500	100	500	-0.3	-0.008	0.008	0.357	-0.002	0.008	0.088	62.02	16.61	16.60
500	100	500	0	-0.074	0.010	0.303	-0.080	0.010	0.089	67.99	11.70	6.82
500	100	500	0.3	-0.063	0.010	0.174	-0.114	0.010	0.069	63.92	6.39	1.93
500	100	500	0.6	-0.054	0.006	0.088	-0.108	0.009	0.034	62.07	6.56	0.83
500	250	2	-0.3	0.002	0.001	0.025	0.020	0.002	0.008	53.96	9.10	1.34
500	250	2	0	-0.003	0.001	0.032	-0.001	0.001	0.011	55.48	7.95	7.93
500	250	2	0.3	-0.004	0.001	0.025	-0.020	0.002	0.008	54.20	9.29	1.36
500	250	2	0.6	-0.003	0.001	0.013	-0.004	0.001	0.003	55.65	26.50	7.46
500	250	42	-0.3	0.002	0.002	0.032	0.030	0.002	0.012	55.61	6.70	0.95
500	250	42	0	-0.001	0.002	0.039	-0.001	0.001	0.016	68.68	5.80	5.79
500	250	42	0.3	-0.001	0.002	0.031	-0.028	0.002	0.012	59.50	6.22	0.99
500	250	42	0.6	-0.007	0.002	0.017	-0.014	0.002	0.004	75.17	16.66	1.65
500	250	500	-0.3	-0.006	0.006	0.167	-0.003	0.006	0.047	132.74	12.43	12.38
500	250	500	0	-0.022	0.006	0.139	-0.040	0.006	0.050	154.28	7.82	4.83
500	250	500	0.3	-0.018	0.005	0.093	-0.111	0.007	0.042	131.65	4.92	0.64
500	250	500	0.6	-0.030	0.004	0.052	-0.099	0.006	0.022	140.49	5.40	0.35

Larger Sample Sizes

One thing to verify is whether the lack of efficiency is simply a small sample characteristic - we can do this check by increasing the sample size. The following table summarizes results in the same fashion as the earlier subsection. Note that the number of simulations is smaller so that the run *Time Taken* (given in minutes) is kept to a manageable level while the $\sqrt{MSE}$ column suggests that the estimates do not suffer overly from simulation error. By increasing the sample size to $n = 2000$, we see that the GMM is now more efficient in all scenarios for the *Ratio Var* criterion and all but one for the *Ratio MSE*. For the latter, it was only the case of substantial discreteness $\phi = 500$ and strong correlation $\rho = 0.3$ where the GMM was outperformed by the pairwise estimator.

Num	n	ϕ	ρ	**Pair**	**Pair**	**Pair**	**GMM**	**GMM**	**GMM**	**Time**	**Ratio**	**Ratio**
Sim				**Bias**	$\sqrt{MSE}$	**AvgSE**	**Bias**	$\sqrt{MSE}$	**AvgSE**	**Taken**	**Var**	**MSE**
100	2000	2	-0.3	0.000	0.001	0.009	0.003	0.001	0.003	83.28	10.06	4.30
100	2000	2	0	-0.001	0.001	0.011	0.000	0.001	0.004	82.96	8.00	7.98
100	2000	2	0.3	0.001	0.001	0.009	-0.002	0.001	0.003	82.28	10.23	7.11
100	2000	2	0.6	0.000	0.001	0.005	0.000	0.001	0.001	84.20	27.46	25.61
100	2000	42	-0.3	0.000	0.001	0.011	0.005	0.001	0.004	91.07	7.75	3.43
100	2000	42	0	-0.001	0.001	0.014	-0.001	0.001	0.006	104.59	5.84	5.80
100	2000	42	0.3	-0.001	0.001	0.011	-0.005	0.001	0.004	99.56	6.99	2.63
100	2000	42	0.6	0.000	0.001	0.006	-0.001	0.001	0.001	123.75	18.78	13.36
100	2000	500	-0.3	0.010	0.005	0.049	-0.018	0.009	0.014	189.38	13.10	5.14
100	2000	500	0	0.005	0.005	0.045	-0.003	0.002	0.017	265.28	7.00	6.90
100	2000	500	0.3	-0.003	0.004	0.032	-0.042	0.007	0.015	185.25	4.86	0.52
100	2000	500	0.6	0.002	0.003	0.017	-0.010	0.003	0.005	213.20	9.88	2.19
100	5000	2	-0.3	-0.001	0.001	0.006	0.000	0.001	0.002	208.61	10.24	10.37
100	5000	2	0	-0.001	0.001	0.007	-0.001	0.000	0.002	206.97	7.99	7.93
100	5000	2	0.3	0.000	0.001	0.006	-0.002	0.001	0.002	208.73	10.29	4.02

Alternative Score Methods

Although the *GMM* estimator is better asymptotically, one would like variations of it where it also does well in smaller sample sizes. Our prior analyses used the basic estimator that is based on the vector of scores

$$g_{\theta,i}(Y_{i1},\ldots,Y_{ip}) = \left(\begin{array}{c} g_{\theta,i12}(Y_{i1},Y_{i2}) \\ \vdots \\ g_{\theta,i1p}(Y_{i1},Y_{ip}) \\ \vdots \\ g_{\theta,i,p-1,p}(Y_{i,p-1},Y_{ip}) \end{array}\right)$$

a column vector $\binom{5}{2} = 10$ different scores. Thus, the dimensions of $\widehat{\text{Var}}\ g_{\hat{\theta}}$ is 10×10. This can be unstable for small samples - as an alternative, we propose combining the scores.

For the table below, we retained the scores corresponding to one time period separation and grouped the others. The argument is that the information in a score at times s and t is $\rho^{|t-s|}$, so the more time separation the less information there is about the parameter ρ. With the resulting five scores, the dimension of $\widehat{\text{Var}}\ g_{\hat{\theta}}$ is 5×5; the estimation of this is presumably more stable.

The following table shows that this conjecture is substantiated. All efficiency measures *Ratio MSE* are greater than 1 for $n = 250$ and all but one for $n = 100$.

Num	n	ϕ	ρ	**Pair**	**Pair**	**Pair**	**GMM**	**GMM**	**GMM**	**Time**	**Ratio**	**Ratio**
Sim				**Bias**	$\sqrt{MSE}$	**AvgSE**	**Bias**	$\sqrt{MSE}$	**AvgSE**	**Taken**	**Var**	**MSE**
500	100	2	−0.3	−0.002	0.002	0.040	0.008	0.002	0.011	27.19	11.97	8.32
500	100	2	0	−0.008	0.002	0.050	−0.007	0.002	0.018	27.11	8.07	7.22
500	100	2	0.3	0.002	0.002	0.039	−0.004	0.002	0.011	27.29	12.42	11.12
500	100	2	0.6	−0.009	0.002	0.021	−0.009	0.002	0.003	28.36	43.19	5.55
500	100	42	−0.3	0.003	0.003	0.050	0.017	0.003	0.017	30.71	8.89	4.44
500	100	42	0	−0.007	0.003	0.061	−0.006	0.002	0.025	32.28	5.86	5.63
500	100	42	0.3	0.002	0.003	0.048	−0.012	0.003	0.016	36.03	8.58	5.64
500	100	42	0.6	−0.018	0.003	0.028	−0.025	0.003	0.005	46.06	26.97	1.69
500	100	500	−0.3	0.001	0.008	0.377	−0.059	0.009	0.075	216.15	25.39	15.74
500	100	500	0	−0.062	0.010	0.277	−0.122	0.012	0.078	207.32	12.73	3.85
500	100	500	0.3	−0.078	0.010	0.178	−0.147	0.012	0.065	64.85	7.59	1.46
500	100	500	0.6	−0.066	0.007	0.090	−0.125	0.010	0.031	87.19	8.58	0.75
500	250	2	−0.3	0.000	0.001	0.025	0.003	0.001	0.007	210.36	12.48	10.08
500	250	2	0	−0.002	0.001	0.032	−0.002	0.001	0.011	210.77	8.01	7.88
500	250	2	0.3	−0.004	0.001	0.025	−0.006	0.001	0.007	206.98	12.55	7.13
500	250	2	0.6	−0.005	0.001	0.013	−0.004	0.001	0.002	206.13	45.14	8.68
500	250	42	−0.3	0.004	0.002	0.032	0.009	0.002	0.010	193.78	9.43	5.32
500	250	42	0	0.002	0.002	0.039	0.002	0.001	0.016	200.59	5.86	5.80
500	250	42	0.3	−0.007	0.002	0.031	−0.012	0.002	0.010	225.64	8.81	4.05
500	250	42	0.6	−0.005	0.002	0.017	−0.008	0.002	0.003	327.96	30.39	4.04
500	250	500	−0.3	−0.004	0.006	0.169	−0.097	0.007	0.033	506.60	26.74	2.73
500	250	500	0	−0.023	0.006	0.137	−0.093	0.008	0.045	471.67	9.39	1.82
500	250	500	0.3	−0.020	0.005	0.095	−0.085	0.007	0.039	497.02	5.85	1.06
500	250	500	0.6	−0.024	0.003	0.052	−0.055	0.004	0.016	584.21	11.02	1.02

3.6 Appendix - Hybrid Distributions

Assume that the random variables Y may have both discrete and continuous components. In insurance and many other fields, the term *mixture* is used for distributions with different sub-populations that are combined using latent variables. So, we prefer to refer to this as a *hybrid* combination of discrete and continuous components to avoid confusion with mixture distributions.

For a random variable Y, let y^d represent a mass point (d for discrete) and let y^c represent a point of continuity (where the density is positive). Let us now write the likelihood for Y_j and Y_k in terms of the discrete and continuous components.

Probability Density/Mass Function

Suppose that we wish to evaluate the likelihood at points of continuity, y_j^c and y_k^c. This corresponds to the classic case of two random variables with probability density function

$$\begin{aligned} f_{jk}(y_j^c, y_k^c) &= \frac{\partial^2}{\partial y_j^c \partial y_k^c} F_{jk}(y_j^c, y_k^c) = \partial_{12} C\left(F_j(y_j^c), F_k(y_k^c)\right) \\ &= c\left(F_j(y_j^c), F_k(y_k^c)\right) f_j(y_j^c) f_k(y_k^c). \end{aligned}$$

Here, $c(\cdot)$ represents the copula density and f_j, f_k are the marginal probability density functions corresponding to F_j, F_k. The notation ∂_{12} means taking the partial derivative with respect to the first and second arguments, y_j and y_k, respectively.

Suppose that we wish to evaluate the likelihood at points of discreteness y_j^d and y_k^d. Then, the joint probability of Y_j and Y_k can be expressed in terms of a copula using the inclusion-exclusion rule

$$\begin{aligned} f_{jk}(y_j^d, y_k^d) &= \Pr(Y_j = y_j^d, Y_k = y_k^d) \\ &= \Pr(Y_j \le y_j^d, Y_k \le y_k^d) - \Pr(Y_j \le y_j^d-, Y_k \le y_k^d) \\ &\quad - \Pr(Y_j \le y_j^d, Y_k \le y_k^d-) + \Pr(Y_j \le y_j^d-, Y_k \le y_k^d-) \\ &= C\left(F_j(y_j^d), F_k(y_k^d)\right) - C\left(F_j(y_j^d-), F_k(y_k^d)\right) \\ &\quad -C\left(F_j(y_j^d), F_k(y_k^d-)\right) + C\left(F_j(y_j^d-), F_k(y_k^d-)\right). \end{aligned}$$

The notation y_k^d- means evaluate y_k^d as a left-hand limit. The last expression uses a notation convention assuming that the mass points are on the integers.

Next, suppose that we wish to evaluate the likelihood of Y_j at mass point y_j^d and of Y_k at point of continuity y_k^c. Then, the joint distribution function has a hybrid probability density/mass function of the form:

$$\begin{aligned} f_{jk}(y_j^d, y_k^c) &= \partial_2 \Pr(Y_j = y_j^d, Y_k \le y_k^c) \\ &= \partial_2 \left\{\Pr(Y_j \le y_j^d, Y_k \le y_k^c) - \Pr(Y_j \le y_j^d-, Y_k \le y_k^c)\right\} \\ &= \partial_2 \left\{C\left(F_j(y_j^d), F_k(y_k^c)\right) - C\left(F_j(y_j-), F_k(y_k^c)\right)\right\} \\ &= \left\{C_2\left(F_j(y_j^d), F_k(y_k^c)\right) - C_2\left(F_j(y_j^d-), F_k(y_k^c)\right)\right\} f_k(y_k^c). \end{aligned}$$

Here, C_2 represents the partial derivative of the copula C with respect to the second argument.

Scores - Derivatives of Probability Density/Mass Functions

We define the score function

$$g_{\theta,ijk}(Y_{ij}, Y_{ik}) = \partial_\theta \ln f_{ijk}(Y_{ij}, Y_{ik}). \tag{5}$$

We now evaluate the scores. For two discrete outcomes, the score can be expressed as

$$\begin{aligned} g_{\theta,ijk}(y_j^d, y_k^d) &= \partial_\theta \ln f_{ijk}(y_j^d, y_k^d) \\ &= \frac{\partial_\theta\left\{C\left(F_j(y_j^d),F_k(y_k^d)\right)-C\left(F_j(y_j^d-),F_k(y_k^d)\right)-C\left(F_j(y_j^d),F_k(y_k^d-)\right)+C\left(F_j(y_j^d-),F_k(y_k^d-)\right)\right\}}{C\left(F_j(y_j^d),F_k(y_k^d)\right)-C\left(F_j(y_j^d-),F_k(y_k^d)\right)-C\left(F_j(y_j^d),F_k(y_k^d-)\right)+C\left(F_j(y_j^d-),F_k(y_k^d-)\right)}. \end{aligned}$$

For a discrete (y_j^d) and a continuous y_k^c outcomes, we have

$$
\begin{aligned}
g_{\theta,ijk}(y_j^d, y_k^c) \quad &= \partial_\theta \ln \left[\left\{ C_2\left(F_j(y_j^d), F_k(y_k^c)\right) - C_2\left(F_j(y_j^d -), F_k(y_k^c)\right) \right\} f_k(y_k^c) \right] \\
&= \frac{\partial_\theta \left\{ C_2\left(F_j(y_j^d), F_k(y_k^c)\right) - C_2\left(F_j(y_j^d -), F_k(y_k^c)\right) \right\}}{C_2\left(F_j(y_j^d), F_k(y_k^c)\right) - C_2\left(F_j(y_j^d -), F_k(y_k^c)\right)}
\end{aligned}
$$

For two continuous outcomes, y_j^c and y_k^c, we have

$$
g_{\theta,ijk}(y_j^c, y_k^c) = \partial_\theta \ln \left[c(F_{Y_{ij}}(y_j^c), F_{Y_{ik}}(y_k^c)) f_{ij}(y_j^c) f_{ik}(y_k^c) \right] = \frac{\partial_\theta \ c(F_{Y_{ij}}(y_j^c), F_{Y_{ik}}(y_k^c))}{c(F_{Y_{ij}}(y_j^c), F_{Y_{ik}}(y_k^c))}.
$$

An advantage of restricting ourselves to pairwise distributions is that most of the functions are available from the R package `VineCopula`. We use an additional relationship, from Plackett (1954),

$$
\frac{\partial}{\partial \rho} C(u_1, u_2) = \phi_2(z_1, z_2).
$$

Here, ϕ_2 is a bivariate normal probability density function and $z_j = \Phi^{-1}(u_j)$, $j = 1, 2$ are the normal scores corresponding to residuals u_j.

4 Online Supplement 4. Lapse Simulation

4.1 Background

This file documents the simulation study of the lapse estimation section of the paper *Joint Models of Insurance Lapsation and Claims*. A **.pdf** version provides a hard copy, the **.html** version allows one to hide `R` code and the **.Rmd** version allows one to run the simulation, changing input parameters as desired. (To run the **.Rmd**, search on **eval=FALSE** and change to **eval=TRUE**.)

Simulation Input Parameters

```
time0 <- Sys.time()  -> time1 # define terms so that we can use to check the run time
# Input Parameters - These are the parameters coded
# Nsamp <-  100       # Number of policyholders
# rhoLA <- -0.2       # Association between Retention and Type 1 Claims
# rhoLH <-  0.2       # Association between Retention and Type 2 Claims
# rhoAH <- 0.1
# Externalphi1 <- 2   # Tweedie dispersion parameter # With a mean=1000, phi=500 for 94% zeros
# Externalphi2 <- 2   # phi=2 for near continuous data, phi=42 for data for about half zeros
# isim <- 1
##################################
#  You can change these vectorized versions of the main input parameters
nSim <- 100
rhoLAVec <- 0.0#c(-0.2)
rhoLHVec <- c(-0.3,0,0.3) #c(0.2)
rhoAHVec <- c(-0.3,0, 0.3) #c(0.1)
Externalphi1Vec <- 42#c(2,42,500)
Externalphi2Vec <- 42#c(2,42,500)
NsampVec <- c(2000)
#rhoLAVec <- c(-0.4, 0, 0.4) ->  rhoLHVec
# rhoAHVec <- c(0,0.2,0.4)
# rhoLHVec <- c(-0.4, 0, 0.4)
# Externalphi1Vec <- c(2,42,500)
# Externalphi2Vec <- c(2,42,500)
```

R Packages Needed to Run this Simulation

```
# Here are the packages that you need to install to run this simulation
library(tweedie)
library(reshape)
library(statmod)
library(knitr)
library(BB)
library(MASS)
library(copula)
library(numDeriv)
library(VineCopula)
library(mvtnorm)
```

4.2 Model Specification

Marginal Outcome Model

We represent the marginal distributions of the claims random variables using a Tweedie distribution so that the distributions have a mass at zero and are otherwise positive. For each claim variable, we use a logarithmic link to form the mean claims of the form

$$\mu_{j,it} = \exp\left(\mathbf{x}_{it}'\boldsymbol{\beta}_j\right), j = 1, 2.$$

Thus, the parameters are allowed to differ between auto and homeowners claims. As a consequence of this, you need not use the same variables for each claim type (a zero beta means that the variable is not part of the mean). Each claim is simulated using the Tweedie distribution, a mean, and two other parameters, ϕ (for dispersion) and P (the *power* parameter).

For the lapse variable, the expected value is of the form

$$\pi_{it} = \frac{\exp\left(\mathbf{x}_{it}'\boldsymbol{\beta}_L\right)}{1 + \exp\left(\mathbf{x}_{it}'\boldsymbol{\beta}_L\right)},$$

a common form for the logit model.

Marginal Input Parameters

```
# You can change these input parameters (except for the number of time replications 'p')
p <- 5                 # Number of years # Do not change
MeanClaim1   <-  5000  # Mean for Type 1 Claims
MeanClaim2   <- 10000  # Mean for Type 2 Claims
MeanLapse    <- 0.05   # Mean for Lapse
Externalxi   <- 1.67   # Tweedie power parameter
beta11 <- 0.2;  beta12 <- 2        # Coefficients for Auto
beta21 <- 0.3;  beta22 <- 3        # Coefficients for Home
betaL1 <- 2;   betaL2  <- -0.25    # Coefficients for Lapse
```

Rating Variables

For this simulation study, we have five rating (explanatory) variables:

- x_1, a binary variable that takes on values 1 or 2 depending on whether or not an attribute holds
- x_2, x_3, x_4, generic continuous explanatory variables
- x_5, time trend ($x_{5it} = t$)

With these values of covariate parameters, the systematic components are

- auto: $\mathbf{x}_{it}'\boldsymbol{\beta}_1 = \beta_{0,1}$+ 0.2 x_1 + 2 x_2
- home: $\mathbf{x}_{it}'\boldsymbol{\beta}_2 = \beta_{0,2}$+ 0.3 x_3 + 3 x_4
- lapse: $\mathbf{x}_{it}'\boldsymbol{\beta}_L = \beta_{0,L}$+ 2 x_2 + -0.25 x_5 .

Intercept parameters are determined using the overall mean terms specified above.

R Code for Generating Covariates

```
Generate_Covariates <- function(Nsamp) {
    x11 <- 1 + rbinom(Nsamp, size = 1, prob = 0.4)  # Time constant Bernoulli variable
    x1 <- matrix(x11, nrow = Nsamp, ncol = p)
    x2 <- matrix(1 + (runif(p * Nsamp) * 2), nrow = Nsamp, ncol = p)
    x3 <- matrix(1 + (runif(p * Nsamp)^2), nrow = Nsamp, ncol = p)
    x4 <- matrix(1 + (runif(p * Nsamp)^2), nrow = Nsamp, ncol = p)
    mu1a <- exp(beta11 * x1 + beta12 * x2)
    mu2a <- exp(beta21 * x3 + beta22 * x4)
    mu1 <- MeanClaim1 * mu1a/mean(mu1a)  # Rescale so that the average claim is MeanClaim1
    mu2 <- MeanClaim2 * mu2a/mean(mu2a)
    yearMat <- t(matrix(rep(1:p, Nsamp), nrow = p, ncol = Nsamp))
    LapseInter <- log(MeanLapse/(1 - MeanLapse)) - mean(betaL1 * x2 + betaL2 *
        yearMat)  # Set the intercept so that the average lapse is approximately MeanLapse
    PI_it <- exp(LapseInter + betaL1 * x2 + betaL2 * yearMat)/(1 + exp(LapseInter +
        betaL1 * x2 + betaL2 * yearMat))
    PolIDMat <- t(matrix(rep(1:Nsamp, each = p), nrow = p, ncol = Nsamp))
    Z <- list(x1, x2, x3, x4, mu1, mu2, yearMat, PI_it, PolIDMat)
    return(Z)
}
```

Dependence Model

Dependence among outcome variables is taken to be a Gaussian copula with the following structure

$$\mathbf{\Sigma} = \begin{pmatrix} 1 & \rho_{LA} & \rho_{LH} \\ \rho_{LA} & 1 & \rho_{AH} \\ \rho_{LH} & \rho_{AH} & 1 \end{pmatrix}.$$

R Code for Simulating Dependent Outcomes

```
Generate_SampleData <- function(param, Nsamp, Externalphi1, Externalphi2, Z) {
    rhoLA <- param[1]
    rhoLH <- param[2]
    rhoAH_L <- param[3]
    rhoAH <- rhoLA * rhoLH + rhoAH_L * sqrt((1 - rhoLA^2) * (1 - rhoLH^2))
    BigSigma <- matrix(c(1, rhoLA, rhoLH, rhoLA, 1, rhoAH, rhoLH, rhoAH, 1),
        nrow = 3, ncol = 3)
    BigSigmaChol <- chol(BigSigma)
    Z1 <- matrix(rnorm(p * Nsamp * 3), nrow = p * Nsamp, ncol = 3) %*% BigSigmaChol
    # Generating Dependent Lapses and Claims
    UCop <- pnorm(Z1)
    PI_it <- Z[[8]]
    PIVec <- as.vector(matrix(Z[[8]], nrow = Nsamp * p, ncol = 1))
    mu1Vec <- as.vector(matrix(Z[[5]], nrow = Nsamp * p, ncol = 1))
    mu2Vec <- as.vector(matrix(Z[[6]], nrow = Nsamp * p, ncol = 1))
    # Simulate Tweedie claims
    Lapse <- 1 * (UCop[, 1] > 1 - PIVec)  # high values of U ==> lapse
    TAvailable <- matrix(1, nrow = Nsamp, ncol = p)
    TAvailable1 <- matrix(1 - Lapse, nrow = Nsamp, ncol = p)
    for (icol in 2:p) {
        TAvailable[, icol] = TAvailable[, icol - 1] * TAvailable1[, icol - 1]
    }
    # TAvailable dictates the number of observed variables
    Claims1 <- qtweedie(UCop[, 2], power = Externalxi, mu = mu1Vec, phi = Externalphi1)
    Claims2 <- qtweedie(UCop[, 3], power = Externalxi, mu = mu2Vec, phi = Externalphi2)
    SampleDataC <- as.data.frame(cbind(Lapse, Claims1, Claims2, matrix(TAvailable,
        nrow = Nsamp * p, ncol = 1), matrix(Z[[7]], nrow = Nsamp * p, ncol = 1),
        matrix(Z[[9]], nrow = Nsamp * p, ncol = 1), matrix(Z[[1]], nrow = Nsamp *
            p, ncol = 1), matrix(Z[[2]], nrow = Nsamp * p, ncol = 1), matrix(Z[[3]],
            nrow = Nsamp * p, ncol = 1), matrix(Z[[4]], nrow = Nsamp * p, ncol = 1),
        PIVec, mu1Vec, mu2Vec))
    colnames(SampleDataC) <- c("Lapse", "Claims1", "Claims2", "TAvail", "year",
        "PolID", "x1", "x2", "x3", "x4", "PI", "mu1", "mu2")
    SampleDatC <- SampleDataC[order(SampleDataC$PolID, SampleDataC$year), ]
    SampleData <- subset(SampleDataC, TAvail == 1)
    return(SampleData)
}
```

4.3 Regression Modeling

The Tweedie is commonly used in insurance applications for claims. In part, this is because it can be expressed as a generalized linear model. In the following illustrative code, we have skipped the determination of the *power* parameter ($P = 1.67$ for us).

R Code for Regression Estimation

This code estimates parameters of each marginal distribution. It also calculates the observations transformed to a uniform scale via the probability integral transform.

```
MargRegress <- function(SampleData) {
    SampleData$dfLapse <- 1 - SampleData$PI
    # Use this if the logistic regression does not converge Use this if tweedie
    # 1 regression does not converge
    dfTweedie1A <- ptweedie(SampleData$Claims1, xi = Externalxi, mu = SampleData$mu1,
        phi = Externalphi1)
    SampleData$dfTweedie1 <- pmin(pmax(1e-05, dfTweedie1A), 0.99999)
    # Use this if tweedie regression 2 does not converge
    dfTweedie2A <- ptweedie(SampleData$Claims2, xi = Externalxi, mu = SampleData$mu2,
        phi = Externalphi2)
    SampleData$dfTweedie2 <- pmin(pmax(1e-05, dfTweedie2A), 0.99999)
    tryCatch({
        logistic.fit <- glm(Lapse ~ x2 + year, data = SampleData, control = glm.control(maxit = 100),
            family = binomial(link = logit))
        SampleData$dfLapse <- 1 - logistic.fit$fitted.values
        # *(SampleData$Lapse==0) - not correct but makes the code easier later...
    }, error = function(err) {
        print(paste("Logistic did not converge:", err))
        NonConvergeLogistic <- NonConvergeLogistic + 1
        return(NonConvergeLogistic)
    })
    tryCatch({
        tweedie.fit1 <- glm(Claims1 ~ x1 + x2, data = SampleData, control = glm.control(maxit = 100),
            family = tweedie(var.power = Externalxi, link.power = 0))
        dfTweedie1A <- ptweedie(SampleData$Claims1, xi = Externalxi, mu = tweedie.fit1$fitted.values,
            phi = summary(tweedie.fit1)$dis)
        SampleData$dfTweedie1 <- pmin(pmax(1e-05, dfTweedie1A), 0.99999)
    }, error = function(err) {
        print(paste("Tweedie1 did not converge:", err))
        NonConvergeTweedie1 <- NonConvergeTweedie1 + 1
        return(NonConvergeTweedie1)
    })
    tryCatch({
        tweedie.fit2 <- glm(Claims2 ~ x3 + x4, data = SampleData, control = glm.control(maxit = 100),
            family = tweedie(var.power = Externalxi, link.power = 0))
        dfTweedie2A <- ptweedie(SampleData$Claims2, xi = Externalxi, mu = tweedie.fit2$fitted.values,
            phi = summary(tweedie.fit2)$dis)
        SampleData$dfTweedie2 <- pmin(pmax(1e-05, dfTweedie2A), 0.99999)
    }, error = function(err) {
        print(paste("Tweedie2 did not converge:", err))
        NonConvergeTweedie2 <- NonConvergeTweedie2 + 1
        return(NonConvergeTweedie2)
    })
    return(SampleData)
}
```

R Code for Data Preparation

This code separates the data into different subsets needed for likelihood calculations. The subsets include lapse and no lapse, as well as four different claim outcomes: (i) no auto/home claim, (ii) an but no home claim, (iii) no auto but a home claim, and (iv) an auto and a home claim.

```
Create_Data_Subsets <- function(SampleData) {
    # Reshape the data
    TweedieLike1 <- SampleData[order(-SampleData$PolID, SampleData$year), ]
    VarsLike <- c("PolID", "year", "Lapse", "dfLapse", "Claims1", "dfTweedie1",
        "Claims2", "dfTweedie2")
    TweedieLike <- TweedieLike1[VarsLike]
    calcOrder <- 1:length(TweedieLike$PolID)
    # These are the four cases from the hybrid joint mass/density function
    caset1t2 <- 1 * (TweedieLike$Claims1 == 0) * (TweedieLike$Claims2 == 0) +
        2 * (TweedieLike$Claims1 > 0) * (TweedieLike$Claims2 == 0) + 3 * (TweedieLike$Claims1 ==
        0) * (TweedieLike$Claims2 > 0) + 4 * (TweedieLike$Claims1 > 0) * (TweedieLike$Claims2 >
        0)
    u <- as.matrix(cbind(TweedieLike$dfLapse, TweedieLike$dfTweedie1, TweedieLike$dfTweedie2))
    zu <- qnorm(u)
    mydata <- data.frame(caset1t2, u, zu, calcOrder, TweedieLike$Lapse)
    names(mydata) <- c("caset1t2", "u1", "u2", "u3", "zu1", "zu2", "zu3", "calcOrder",
        "Lapse")
    mydataLapse <- mydata[which(mydata$Lapse == 1), ]
    mydataNoLapse <- mydata[which(mydata$Lapse == 0), ]
    mydata1 <- mydata[which(caset1t2 == 1), ]
    mydata2 <- mydata[which(caset1t2 == 2), ]
    mydata3 <- mydata[which(caset1t2 == 3), ]
    mydata4 <- mydata[which(caset1t2 == 4), ]
    totalallmydata <- list(mydataLapse, mydataNoLapse, mydata1, mydata2, mydata3,
        mydata4, mydata)
    return(totalallmydata)
}
```

4.4 Joint Model Specification

Pairwise Likelihoods

For initial estimation, we first consider the bivariate likelihood between lapse and auto and between lapse and home claims. Consider two random variables where the first is binary (for lapse) and the second may have a Tweedie distribution. The probability density/mass function of the following form

$$f(y_1,y_2) = \begin{cases} \Pr(Y_1=0,Y_2=0) & = C\left(F_1(0),F_2(0)\right) & \text{for } y_1=0, y_2=0 \\ \frac{\partial}{\partial y}\Pr(Y_1=0,Y_2\le y)|_{y=y_1} & = C_2\left(F_1(0),F_2(y)\right)f_2(y) & \text{for } y_1=0, y_2>0 \\ \Pr(Y_1=1,Y_2=0) & = F_2(0) - C\left(F_1(0),F_2(0)\right) & \text{for } y_1=1, y_2=0 \\ \frac{\partial}{\partial y}\Pr(Y_1=1,Y_2\le y)|_{y=y_1} & = \{1-C_2\left(F_1(0),F_2(y)\right)\}f_2(y) & \text{for } y_1=1, y_2>0 \end{cases}$$

Pairwise likelihood for auto and home was described in our earlier study of *GMM* estimators without lapse.

R Code for Pairwise Likelihood Calculation

In the following pairwise likelihood calculations, we drop the marginal densities $f_2(y)$ as they contain no information about the dependence parameters. See the documentation for the R package VineCopula for explanations of the functions to evaluate bivariate copulas and their derivatives.

```
NegBivariateLikelihood <- function(rho, type) {
    bilikehd <- 0 * calcOrder
    if (nrow(mydataNoLapse) > 0) {
        dat <- mydataNoLapse
        u <- (type == 1) * dat$u2 + (type == 2) * dat$u3
        ClaimInd <- 1 * (dat$caset1t2 == 4) + 1 * (dat$caset1t2 == 2) * (type ==
            1) + 1 * (dat$caset1t2 == 3) * (type == 2)
        ZeroLike <- as.matrix(BiCopCDF(dat$u1, u, family = 1, par = rho), ncol = 1)
        PosLike <- as.matrix(BiCopHfunc2(dat$u1, u, family = 1, par = rho),
            ncol = 1)
        bilikehd[dat$calcOrder] <- (ClaimInd == 0) * ZeroLike + (ClaimInd >
            0) * PosLike
    }
    if (nrow(mydataLapse) > 0) {
        dat <- mydataLapse
        u <- (type == 1) * dat$u2 + (type == 2) * dat$u3
        ClaimInd <- 1 * (dat$caset1t2 == 4) + 1 * (dat$caset1t2 == 2) * (type ==
            1) + 1 * (dat$caset1t2 == 3) * (type == 2)
        ZeroLike <- u - as.matrix(BiCopCDF(dat$u1, u, family = 1, par = rho),
            ncol = 1)
        PosLike <- 1 - as.matrix(BiCopHfunc2(dat$u1, u, family = 1, par = rho),
            ncol = 1)
        bilikehd[dat$calcOrder] <- (ClaimInd == 0) * ZeroLike + (ClaimInd >
            0) * PosLike
    }
    bilikehd[is.na(bilikehd)] <- 0
    bilikehd <- pmin(pmax(1e-12, bilikehd), 1e+12)
    return(-sum(log(bilikehd)))
}

BiLikeLA <- function(rho) {
    return(NegBivariateLikelihood(rho, 1))
}
BiLikeLH <- function(rho) {
    return(NegBivariateLikelihood(rho, 2))
}
```

```
BiLikeAH <- function(rho) {
    # See the VineCopula package for the functions 'BiCopCDF', 'BiCopHfunc', and
    # 'BiCopPDF'
    likehd <- 0 * calcOrder
    if (nrow(mydata1) > 0) {
        likehd[mydata1$calcOrder] <- BiCopCDF(mydata1$u2, mydata1$u3, family = 1,
            par = rho)
    }
    if (nrow(mydata2) > 0) {
        likehd[mydata2$calcOrder] <- BiCopHfunc1(mydata2$u2, mydata2$u3, family = 1,
            par = rho)
    }
    if (nrow(mydata3) > 0) {
        likehd[mydata3$calcOrder] <- BiCopHfunc2(mydata3$u2, mydata3$u3, family = 1,
            par = rho)
    }
    if (nrow(mydata4) > 0) {
        likehd[mydata4$calcOrder] <- BiCopPDF(mydata4$u2, mydata4$u3, family = 1,
            par = rho)
    }
    likehd <- pmin(pmax(1e-12, likehd), 1e+12)
    return(-sum(log(likehd)))
}
```

GMM Estimation

Let θ be a three-dimensional vector that represents the parameters that quantify the association among $\{L_{it}, Y_{1,it}, Y_{2,it}\}$. Given $T_i = t$, the hybrid probability density/mass function of $Y_{1,it}$ and $Y_{2,it}$ is $f_{i12,s|t}(\cdot,\cdot)$, as specified in Appendix A.

To estimate θ, for $s \leq t$, define the scores

$$g_{\theta,i}(y_1, y_2, T, t) = \mathrm{I}(T = t)\ \partial_\theta \ln f_{i12,s|t}(y_1, y_2).$$

This is a mean zero random variable that contains information about θ.

In this simulation, the two random variables Y_1 and Y_2 both follow a Tweedie distribution. The scores have different expressions when (i) no lapse is involved and when (ii) there is lapse.

No Lapse

Consider the first case where outcomes are prior to lapse so that $s < t \leq m+1$. For two zero outcomes, this can be expressed as

$$\partial_\theta \ln f_{i12,s|t}(0,0) = \frac{\partial_\theta \; C\left(F_{Lis}(0), F_{1,is}(0), F_{2,is}(0)\right)}{C\left(F_{Lis}(0), F_{1,is}(0), F_{2,is}(0)\right)}.$$

For $s < t \leq m+1$ and a single positive outcome, $y > 0$, we have

$$\begin{aligned} \partial_\theta \ln f_{i12,s|t}(y,0) \quad &= \partial_\theta \ln\left[C_2\left(F_{Lis}(0), F_{1,is}(y), F_{2,is}(0)\right) f_{1,is}(y)\right] \\ &= \tfrac{\partial_\theta C_2(F_{Lis}(0),F_{1,is}(y),F_{2,is}(0))}{C_2(F_{Lis}(0),F_{1,is}(y),F_{2,is}(0))} \end{aligned} \quad .$$

For $s < t \leq m+1$ and two positive outcomes, $y_1 > 0$ and $y_2 > 0$, we have

$$\begin{aligned} \partial_\theta \ln f_{i12,s|t}(y_1,y_2) \quad &= \partial_\theta \ln\left[C_{23}\left(F_{Lis}(0), F_{1,is}(y_1), F_{2,it}(y_2)\right)\right] \\ &= \tfrac{\partial_\theta \; C_{23}(F_{Lis}(0),F_{1,is}(y_1),F_{2,is}(y_2))}{C_{23}(F_{Lit}(0),F_{1,is}(y_1),F_{2,is}(y_2))}. \end{aligned}$$

Lapse

Consider the second case where outcomes occur during the lapse year so that $s = t \leq m$. For two zero outcomes, the score can be expressed as

$$\partial_\theta \ln f_{i12,s|t}(0,0) = \frac{\sum_{i=0}^{1} (-1)^i \, \partial_\theta C\left(F_{Lis}(0)^i, F_{1,is}(0), F_{2,is}(0)\right)}{\sum_{i=0}^{1} (-1)^i \, C\left(F_{Lis}(0)^i, F_{1,is}(0), F_{2,is}(0)\right)}.$$

For $s = t \leq m$ and a single positive outcome, $y > 0$, we have

$$\partial_\theta \ln f_{i12,s|t}(y,0) = \frac{\sum_{i=0}^{1} (-1)^i \, \partial_\theta C_2\left(F_{Lis}(0)^i, F_{1,is}(y), F_{2,is}(0)\right)}{\sum_{i=0}^{1} (-1)^i \, C_2\left(F_{Lis}(0)^i, F_{1,is}(y), F_{2,is}(0)\right)}.$$

In the same way, for $s = t \leq m$ and two positive outcomes, $y_1 > 0$ and $y_2 > 0$, we have

$$\partial_\theta \ln f_{i12,s|t}(y_1,y_2) = \frac{\sum_{i=0}^{1} (-1)^i \, \partial_\theta C_{23}\left(F_{Lis}(0)^i, F_{1,is}(y_1), F_{2,is}(y_2)\right)}{\sum_{i=0}^{1} (-1)^i \, C_{23}\left(F_{Lis}(0)^i, F_{1,is}(y_1), F_{2,is}(y_2)\right)}.$$

Trivariate Gaussian Copula Derivatives with Respect to Association Parameters

To compute the copula and its derivatives we assume a parametric Gaussian copula; this section summarizes results to evaluate the scores using Gaussian copulas. The derivations in Appendix B. Specifically, We evaluate $\frac{\partial}{\partial\rho}C(u_1,u_2,u_3)$ in Appendix B.1, $\frac{\partial}{\partial\rho}C_2(u_1,u_2,u_3)$ in Appendix B.2, and $\frac{\partial}{\partial\rho}C_{23}(u_1,u_2,u_3)$ in Appendix B.3. For simplicity, we use the following generic expression for the association matrix

$$\mathbf{\Sigma} = \left(\begin{array}{ccc} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{array}\right).$$

No Derivatives

First, with uniform random variables u_j, we define the normal scores $z_j = \Phi^{-1}(u_j)$. Thus,

$$\frac{\partial}{\partial\rho_{12}} C(u_1,u_2,u_3) = \phi_2\left(\left(\begin{array}{c} z_1 \\ z_2 \end{array}\right); \mathbf{\Sigma}_{11}\right) \; \Phi(z_3^*; \Sigma_{22\cdot1}),$$

where

$$z_3^* = z_3 - \mathbf{\Sigma}_{12}'\mathbf{\Sigma}_{11}^{-1}\begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \quad \mathbf{\Sigma}_{11} = \begin{pmatrix} 1 & \rho_{12} \\ \rho_{12} & 1 \end{pmatrix}, \quad \mathbf{\Sigma}_{12} = \begin{pmatrix} \rho_{13} \\ \rho_{23} \end{pmatrix}, \quad \Sigma_{22\cdot 1} = 1 - \mathbf{\Sigma}_{12}'\mathbf{\Sigma}_{11}^{-1}\mathbf{\Sigma}_{12}.$$

One Derivative

Second,

$$\tfrac{\partial}{\partial\rho}C_3\left(u_1,u_2,u_3\right) = h_1^*\left(z_1^*,z_2^*;\rho_x\right)\tfrac{\partial}{\partial\rho}z_1^* + h_1^*\left(z_2^*,z_1^*;\rho_x\right)\tfrac{\partial}{\partial\rho}z_2^* + \phi_2\left(z_1^*,z_2^*;\rho_x\right)\tfrac{\partial}{\partial\rho}\rho_x,$$

where $z_1^* = (z_1 - z_3\rho_{13})/\sigma_1$, $z_2^* = (z_2 - z_3\rho_{23})/\sigma_2$, $\sigma_1^2 = 1-\rho_{13}^2$, $\sigma_2^2 = 1-\rho_{23}^2$, and $\rho_X\sigma_1\sigma_2 = \rho_{12} - \rho_{13}\rho_{23}$, and

$$h_1^*(x_1,x_2,\rho_X) = \phi\left(x_1\right)\Phi\left(\frac{x_2-\rho_X x_1}{\sqrt{1-\rho_X^2}}\right).$$

We also need

$$\frac{\partial}{\partial\rho}z_1^* = \frac{\partial}{\partial\rho}\left(\frac{z_1-\mu_{12\cdot 3,1}}{\sigma_1}\right) = \frac{\partial}{\partial\rho}\left(\frac{z_1-z_3\rho_{13}}{\sqrt{1-\rho_{13}^2}}\right) = \begin{cases} 0 & \rho=\rho_{12} \\ \frac{z_1\rho_{13}-z_3}{(1-\rho_{13}^2)^{3/2}} & \rho=\rho_{13} \\ 0 & \rho=\rho_{23} \end{cases},$$

$$\frac{\partial}{\partial\rho}z_2^* = \frac{\partial}{\partial\rho}\left(\frac{z_2-\mu_{12\cdot 3,2}}{\sigma_2}\right) = \frac{\partial}{\partial\rho}\left(\frac{z_2-z_3\rho_{23}}{\sqrt{1-\rho_{23}^2}}\right) = \begin{cases} 0 & \rho=\rho_{12} \\ 0 & \rho=\rho_{13} \\ \frac{z_2\rho_{23}-z_3}{(1-\rho_{23}^2)^{3/2}} & \rho=\rho_{23} \end{cases},$$

and

$$\frac{\partial}{\partial\rho}\rho_x = \frac{\partial}{\partial\rho}\left(\frac{\rho_{12}-\rho_{13}\rho_{23}}{\sigma_1\sigma_2}\right) = \frac{\partial}{\partial\rho}\left(\frac{\rho_{12}-\rho_{13}\rho_{23}}{(1-\rho_{13}^2)^{1/2}(1-\rho_{23}^2)^{1/2}}\right) = \begin{cases} \frac{1}{(1-\rho_{13}^2)^{1/2}(1-\rho_{23}^2)^{1/2}} & \rho=\rho_{12} \\ \frac{\rho_{12}\rho_{13}-\rho_{23}}{(1-\rho_{13}^2)^{3/2}(1-\rho_{23}^2)^{1/2}} & \rho=\rho_{13} \\ \frac{\rho_{12}\rho_{23}-\rho_{13}}{(1-\rho_{13}^2)^{1/2}(1-\rho_{23}^2)^{3/2}} & \rho=\rho_{23} \end{cases}.$$

Two Derivatives

Third,

$$\frac{\partial}{\partial\rho}C_{23}\left(u_1,u_2,u_3\right) = \Phi\left(z_1-\mu_{1\cdot 23};\sigma_{1\cdot 23}\right)\frac{\partial}{\partial\rho}c\left(u_2,u_3\right) + \left(\frac{\partial}{\partial\rho}C\left(u_1|u_2,u_3\right)\right)c\left(u_2,u_3\right),$$

where

$$\tfrac{\partial}{\partial\rho}C\left(u_1|u_2,u_3\right) = -\phi\left(\tfrac{z_1-\mu_{1\cdot 23}}{\sqrt{\sigma_{1\cdot 23}}}\right)\tfrac{1}{\sigma_{1\cdot 23}^{3/2}}\left(\sigma_{1\cdot 23}\tfrac{\partial}{\partial\rho}\mu_{1\cdot 23} + \tfrac{1}{2}(z_1-\mu_{1\cdot 23})\tfrac{\partial}{\partial\rho}\sigma_{1\cdot 23}\right).$$

We also need $\mu_{1\cdot 23} = \begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix}$. With this, we have

$$\frac{\partial}{\partial\rho}\mu_{1\cdot 23} = \begin{cases} \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix} & \rho=\rho_{12} \\ \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix} & \rho=\rho_{13} \\ \begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix} & \rho=\rho_{23} \end{cases}$$

Further, with $\sigma_{1\cdot 23} = 1 - \begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix}$, we have

$$\frac{\partial}{\partial \rho}\sigma_{1\cdot 23} = \begin{cases} -2\begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix} & \rho = \rho_{12} \\ -2\begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix} & \rho = \rho_{13} \\ -\begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix} & \rho = \rho_{23} \end{cases}$$

Derivatives of the bivariate density $c(u_2, u_3)$ follow directly from results of Schepsmeier and Stober (2012, page 2).

R Code for GMM Functions

Top level functions are here.

```
GMMgthetaFunct<- function(param) {
   Scores <- data.frame(GMMScore(param)[,c(2:4)])
    names(Scores) <- c("Score12", "Score13", "Score23")
  return(as.matrix(Scores))
}

GMMFunc<- function(param) {
  GMMgtheta <- GMMgthetaFunct(param)
  GMMScorex <- t(colSums(GMMgtheta)) %*% ginv(Vargtheta) %*% colSums(GMMgtheta) /
                 length(GMMgtheta[,1])
  return(GMMScorex)
}
```

R Code for GMM Scores

The likelihood and scores corresponding to all outcomes are calculated here. It uses as input the following subsection that provides calculations only for the trivariate piece. The function returns the likelihood and three scores (corresponding to the three association parameters).

```
GMMScore <- function(param) {
    rhoLA <- param[1]
    rhoLH <- param[2]
    rhoAH_L <- param[3]
    # Transformed parameter
    rhoAH <- rhoLA * rhoLH + rhoAH_L * sqrt((1 - rhoLA^2) * (1 - rhoLH^2))
    rhoLA <- pmin(pmax(-0.99, rhoLA), 0.99)
    rhoLH <- pmin(pmax(-0.99, rhoLH), 0.99)
    rhoAH <- pmin(pmax(-0.99, rhoAH), 0.99)
    SigmaList <- SigmaFct(rhoLA, rhoLH, rhoAH)
    Sigma <- SigmaList[[1]]
    Sigma12 <- SigmaList[[2]]
    Sigma13 <- SigmaList[[3]]
    Sigma23 <- SigmaList[[4]]
    Sigma13.2 <- SigmaList[[5]]
    Sigma12.3 <- SigmaList[[6]]
    Sigma23.1 <- SigmaList[[7]]
    Sigma3.12 <- SigmaList[[8]]
    Sigma1.23 <- SigmaList[[9]]
    Sigma2.13 <- SigmaList[[10]]

    vec111 <- as.vector(cbind(1, 1, 1))
    vec001 <- as.vector(cbind(0, 0, 1))
    ScoreLike <- matrix(0, length(mydata[, 1]), 4)
    # (Auto=0,Home=0) Case
    if (nrow(mydata1) > 0) {
        dat <- mydata1
        Reten0 <- 1 - dat$Lapse
        score <- GMMScore00(rhoLA, rhoLH, rhoAH, dat$u1, dat$u2, dat$u3)
        fbb <- score[, 2:4]
        Fbb <- as.matrix(score[, 1], ncol = 1) %*% vec111
        Fbb[, 1] <- pmax(1e-05, Fbb[, 1])
        Fbb[, 2] <- pmax(1e-05, Fbb[, 2])
        Fbb[, 3] <- pmax(1e-05, Fbb[, 3])
        faa <- as.matrix(dmvnorm(cbind(dat$zu2, dat$zu3), mean = rep(0, 2),
            sigma = Sigma23, log = FALSE), ncol = 1) %*% vec001
        Faa <- as.matrix(BiCopCDF(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec111
        Faa_Fbb <- pmin(pmax(1e-05, (Faa - Fbb)[, 1]), 0.99999)
        ScoreLike[dat$calcOrder, 1] <- Reten0 * log(Fbb)[, 1] + (1 - Reten0) *
            log(Faa_Fbb)
        ScoreLike[dat$calcOrder, 2:4] <- Reten0 * fbb/Fbb + (1 - Reten0) * (faa -
            fbb)/Faa_Fbb
    }
    # (Auto=1,Home=0) Case
    if (nrow(mydata2) > 0) {
        dat <- mydata2
        Reten0 <- 1 - dat$Lapse
        score <- GMMScore10(rhoLA, rhoLH, rhoAH, dat$u1, dat$u2, dat$u3)
        fbb <- score[, 2:4]
        Fbb <- as.matrix(score[, 1], ncol = 1) %*% vec111
        Fbb[, 1] <- pmax(1e-05, Fbb[, 1])
        Fbb[, 2] <- pmax(1e-05, Fbb[, 2])
        Fbb[, 3] <- pmax(1e-05, Fbb[, 3])
        faa <- as.matrix(BiCopHfuncDeriv(dat$u3, dat$u2, family = 1, par = rhoAH),
            ncol = 1) %*% vec001
        Faa <- as.matrix(BiCopHfunc1(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec111
        Faa_Fbb <- pmin(pmax(1e-05, (Faa - Fbb)[, 1]), 0.99999)
        ScoreLike[dat$calcOrder, 1] <- Reten0 * log(Fbb)[, 1] + (1 - Reten0) *
            log(Faa_Fbb)
        ScoreLike[dat$calcOrder, 2:4] <- Reten0 * fbb/Fbb + (1 - Reten0) * (faa -
            fbb)/Faa_Fbb
    }
    # (Auto=0,Home=1) Case
    if (nrow(mydata3) > 0) {
        dat <- mydata3
        Reten0 <- 1 - dat$Lapse
        score <- GMMScore01(rhoLA, rhoLH, rhoAH, dat$u1, dat$u2, dat$u3)
        fbb <- score[, 2:4]
        Fbb <- as.matrix(score[, 1], ncol = 1) %*% vec111
        Fbb[, 1] <- pmax(1e-05, Fbb[, 1])
        Fbb[, 2] <- pmax(1e-05, Fbb[, 2])
        Fbb[, 3] <- pmax(1e-05, Fbb[, 3])
        faa <- as.matrix(BiCopHfuncDeriv(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec001
        Faa <- as.matrix(BiCopHfunc2(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec111
        Faa_Fbb <- pmin(pmax(1e-05, (Faa - Fbb)[, 1]), 0.99999)
        ScoreLike[dat$calcOrder, 1] <- Reten0 * log(Fbb)[, 1] + (1 - Reten0) *
            log(Faa_Fbb)
        ScoreLike[dat$calcOrder, 2:4] <- Reten0 * fbb/Fbb + (1 - Reten0) * (faa -
            fbb)/Faa_Fbb
    }
    # (Auto=1,Home=1) Case
    if (nrow(mydata4) > 0) {
        dat <- mydata4
        Reten0 <- 1 - dat$Lapse
        score <- GMMScore11(rhoLA, rhoLH, rhoAH, dat$u1, dat$u2, dat$u3)
        fbb <- score[, 2:4]
        Fbb <- as.matrix(score[, 1], ncol = 1) %*% vec111
        Fbb[, 1] <- pmax(1e-05, Fbb[, 1])
        Fbb[, 2] <- pmax(1e-05, Fbb[, 2])
        Fbb[, 3] <- pmax(1e-05, Fbb[, 3])
        faa <- as.matrix(BiCopDeriv(pnorm(dat$zu2), pnorm(dat$zu3), family = 1,
            par = rhoAH), ncol = 1) %*% vec001
        Faa <- as.matrix(BiCopPDF(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec111
        Faa_Fbb <- pmin(pmax(1e-05, (Faa - Fbb)[, 1]), 0.99999)
        ScoreLike[dat$calcOrder, 1] <- Reten0 * log(Fbb)[, 1] + (1 - Reten0) *
            log(Faa_Fbb)
        ScoreLike[dat$calcOrder, 2:4] <- Reten0 * fbb/Fbb + (1 - Reten0) * (faa -
            fbb)/Faa_Fbb
    }
    return(ScoreLike)
}
```

R Code for Basic GMM Score Functions

The likelihood and scores corresponding to trivariate outcomes are calculated here. This has four functions corresponding to our four data cases: (i) (Auto=0,Home=0), (ii) (Auto=1,Home=0), (iii) (Auto=0,Home=1), and (iv) (Auto=1,Home=1). Each function returns the likelihood and three scores (corresponding to the three association parameters).

```
# (Auto=0,Home=0) Case
GMMScore00 <- function(rho12, rho13, rho23, u1, u2, u3) {
    norm.cops <- normalCopula(param = c(rho12, rho13, rho23), dispstr = "un",
        dim = 3)
    like <- pCopula(cbind(u1, u2, u3), copula = norm.cops, algorithm = TVPACK(abseps = 1e-08))
    like <- pmin(pmax(1e-05, like), 0.99999)

    zu1 <- as.vector(qnorm(u1))
    zu2 <- as.vector(qnorm(u2))
    zu3 <- as.vector(qnorm(u3))
    SigmaList <- SigmaFct(rho12, rho13, rho23)
    Sigma <- SigmaList[[1]]
    Sigma12 <- SigmaList[[2]]
    Sigma13 <- SigmaList[[3]]
    Sigma23 <- SigmaList[[4]]
    Sigma3.12 <- SigmaList[[8]]
    Sigma1.23 <- SigmaList[[9]]
    Sigma2.13 <- SigmaList[[10]]

    zstar12 <- zu3 - as.matrix(cbind(zu1, zu2)) %*% ginv(Sigma12) %*% Sigma[c(1,
        2), 3]
    score12 <- dmvnorm(cbind(zu1, zu2), mean = rep(0, 2), sigma = Sigma12, log = FALSE) *
        pnorm(zstar12, sd = sqrt(Sigma3.12))
    zstar13 <- zu2 - as.matrix(cbind(zu1, zu3)) %*% ginv(Sigma13) %*% Sigma[c(1,
        3), 2]
    score13 <- dmvnorm(cbind(zu1, zu3), mean = rep(0, 2), sigma = Sigma13, log = FALSE) *
        pnorm(zstar13, sd = sqrt(Sigma2.13))
    zstar23 <- zu1 - as.matrix(cbind(zu2, zu3)) %*% ginv(Sigma23) %*% Sigma[c(2,
        3), 1]
    score23 <- dmvnorm(cbind(zu2, zu3), mean = rep(0, 2), sigma = Sigma23, log = FALSE) *
        pnorm(zstar23, sd = sqrt(Sigma1.23))
    return(cbind(like, score12, score13, score23))
}

# (Auto=1,Home=0) Case
GMMScore10 <- function(rho12, rho13, rho23, u1, u2, u3) {
    zu1 <- as.vector(qnorm(u1))
    zu2 <- as.vector(qnorm(u2))
    zu3 <- as.vector(qnorm(u3))
    sig1 <- sqrt(1 - rho12^2)
    sig2 <- sqrt(1 - rho23^2)
    rhox <- (rho13 - rho12 * rho23)/(sig1 * sig2)
    rhox <- pmin(pmax(-0.999, rhox), 0.999)
```

```r
    z1s <- (zu1 - zu2 * rho12)/sig1
    z3s <- (zu3 - zu2 * rho23)/sig2
    like <- BiCopCDF(pnorm(z1s), pnorm(z3s), family = 1, par = rhox)
    like <- pmin(pmax(1e-05, like), 0.99999)

    rhoxMat <- matrix(c(1, rhox, rhox, 1), nrow = 2, ncol = 2)
    scoreAiii <- dmvnorm(cbind(z1s, z3s), mean = rep(0, 2), sigma = rhoxMat,
        log = FALSE)
    score13 <- 0 + 0 + scoreAiii/(sig1 * sig2)
    score12 <- bideriv1(z1s, z3s, rhox) * (zu1 * rho12 - zu2)/sig1^3 + 0 + scoreAiii *
        (rho12 * rho13 - rho23)/(sig1^3 * sig2)
    score23 <- 0 + bideriv1(z3s, z1s, rhox) * (zu3 * rho23 - zu2)/sig2^3 + scoreAiii *
        (rho13 * rho23 - rho12)/(sig1 * sig2^3)
    return(cbind(like, score12, score13, score23))
}
# Helpful Function
bideriv1 <- function(x1, x2, rhox) {
    return(dnorm(x1) * pnorm((x2 - rhox * x1)/sqrt(1 - rhox^2)))
}
# (Auto=0,Home=1) Case
GMMScore01 <- function(rho12, rho13, rho23, u1, u2, u3) {
    zu1 <- as.vector(qnorm(u1))
    zu2 <- as.vector(qnorm(u2))
    zu3 <- as.vector(qnorm(u3))
    sig1 <- sqrt(1 - rho13^2)
    sig2 <- sqrt(1 - rho23^2)
    rhox <- (rho12 - rho13 * rho23)/(sig1 * sig2)
    rhox <- pmin(pmax(-0.999, rhox), 0.999)
    z1s <- (zu1 - zu3 * rho13)/sig1
    z2s <- (zu2 - zu3 * rho23)/sig2
    like <- BiCopCDF(pnorm(z1s), pnorm(z2s), family = 1, par = rhox)
    like <- pmin(pmax(1e-05, like), 0.99999)

    rhoxMat <- matrix(c(1, rhox, rhox, 1), nrow = 2, ncol = 2)
    scoreAiii <- dmvnorm(cbind(z1s, z2s), mean = rep(0, 2), sigma = rhoxMat,
        log = FALSE)
    score12 <- 0 + 0 + scoreAiii/(sig1 * sig2)
    score13 <- bideriv1(z1s, z2s, rhox) * (zu1 * rho13 - zu3)/sig1^3 + 0 + scoreAiii *
        (rho12 * rho13 - rho23)/(sig1^3 * sig2)
    score23 <- 0 + bideriv1(z2s, z1s, rhox) * (zu2 * rho23 - zu3)/sig2^3 + scoreAiii *
        (rho12 * rho23 - rho13)/(sig1 * sig2^3)
    return(cbind(like, score12, score13, score23))
}
# (Auto=1,Home=1) Case
GMMScore11 <- function(rho12, rho13, rho23, u1, u2, u3) {
    zu1 <- as.vector(qnorm(u1))
    zu2 <- as.vector(qnorm(u2))
    zu3 <- as.vector(qnorm(u3))
    SigmaList <- SigmaFct(rho12, rho13, rho23)
    Sigma <- SigmaList[[1]]
    Sigma12 <- SigmaList[[2]]
    Sigma13 <- SigmaList[[3]]
    Sigma23 <- SigmaList[[4]]
    Sigma13.2 <- SigmaList[[5]]
    Sigma12.3 <- SigmaList[[6]]
    Sigma23.1 <- SigmaList[[7]]
    Sigma3.12 <- SigmaList[[8]]
    Sigma1.23 <- as.numeric(SigmaList[[9]])
    Sigma2.13 <- SigmaList[[10]]
    tempSig <- ginv(Sigma[2:3, 2:3])
    mu1.23 <- cbind(zu2, zu3) %*% tempSig %*% Sigma[1, 2:3]
    z1s <- (zu1 - as.vector(mu1.23))/as.numeric(sqrt(Sigma1.23))
    like <- pnorm(z1s) * BiCopPDF(u2, u3, family = 1, par = rho23)
    like <- pmax(1e-05, like)

    partialCop1 <- -dnorm(z1s)/Sigma1.23^(3/2)
    mu1.231 <- as.vector(cbind(zu2, zu3) %*% tempSig %*% as.vector(c(1, 0)))
    mu1.232 <- as.vector(cbind(zu2, zu3) %*% tempSig %*% as.vector(c(0, 1)))
    mu1.233 <- -as.vector(cbind(zu2, zu3) %*% tempSig %*% matrix(c(0, 1, 1,
        0), nrow = 2, ncol = 2) %*% tempSig %*% Sigma[1, 2:3])

    sig1.231 <- as.numeric(-2 * Sigma[2:3, 1] %*% tempSig %*% as.vector(c(1,
        0)))
    sig1.232 <- as.numeric(-2 * Sigma[2:3, 1] %*% tempSig %*% as.vector(c(0,
        1)))
    sig1.233 <- as.numeric(Sigma[2:3, 1] %*% tempSig %*% matrix(c(0, 1, 1, 0),
        nrow = 2, ncol = 2) %*% tempSig %*% Sigma[1, 2:3])

    partialCop12 <- partialCop1 * (Sigma1.23 * mu1.231 + 0.5 * (zu1 - mu1.23) *
        sig1.231)
    partialCop13 <- partialCop1 * (Sigma1.23 * mu1.232 + 0.5 * (zu1 - mu1.23) *
        sig1.232)
    partialCop23 <- partialCop1 * (Sigma1.23 * mu1.233 + 0.5 * (zu1 - mu1.23) *
        sig1.233)

    score12 <- 0 + partialCop12 * BiCopPDF(u2, u3, family = 1, par = rho23)
    score13 <- 0 + partialCop13 * BiCopPDF(u2, u3, family = 1, par = rho23)
    score23 <- pnorm(z1s) * BiCopDeriv(u2, u3, family = 1, par = rho23, deriv = "par",
        log = FALSE) + partialCop23 * BiCopPDF(u2, u3, family = 1, par = rho23)
    return(cbind(like, score12, score13, score23))
}
```

R Code for Matrices of Association Parameters

This function helps to organize the association parameters used througout.

```
SigmaFct <- function(rho12, rho13, rho23) {
    Sigma <- matrix(c(1, rho12, rho13, rho12, 1, rho23, rho13, rho23, 1), nrow = 3,
        ncol = 3)
    Sigma12 <- Sigma[c(1, 2), c(1, 2)]
    Sigma13 <- Sigma[c(1, 3), c(1, 3)]
    Sigma23 <- Sigma[c(2, 3), c(2, 3)]
    Sigma13.2 <- Sigma[c(1, 3), c(1, 3)] - Sigma[c(1, 3), 2] %*% t(Sigma[c(1,
        3), 2])
    Sigma12.3 <- Sigma[c(1, 2), c(1, 2)] - Sigma[c(1, 2), 3] %*% t(Sigma[c(1,
        2), 3])
    Sigma23.1 <- Sigma[c(2, 3), c(2, 3)] - Sigma[c(2, 3), 1] %*% t(Sigma[c(2,
        3), 1])
    Sigma3.12 <- 1 - Sigma[3, 1:2] %*% ginv(Sigma[1:2, 1:2]) %*% Sigma[3, 1:2]
    Sigma1.23 <- 1 - Sigma[1, 2:3] %*% ginv(Sigma[2:3, 2:3]) %*% Sigma[1, 2:3]
    Sigma2.13 <- 1 - Sigma[2, c(1, 3)] %*% ginv(Sigma[c(1, 3), c(1, 3)]) %*%
        Sigma[2, c(1, 3)]
    Sigma3.12 <- Sigma3.12 * (Sigma3.12 > 0)
    Sigma1.23 <- Sigma1.23 * (Sigma1.23 > 0)
    Sigma2.13 <- Sigma2.13 * (Sigma2.13 > 0)
    list(Sigma, Sigma12, Sigma13, Sigma23, Sigma13.2, Sigma12.3, Sigma23.1,
        Sigma3.12, Sigma1.23, Sigma2.13)
}
```

R Code for Simulation Loop

```
time1 <- Sys.time()
set.seed(123457)
NonConvergeTweedie1 <- NonConvergeTweedie2 <- 0
NumRuns <- length(rhoLAVec) * length(rhoLHVec) * length(rhoAHVec) * length(Externalphi1Vec) *
```

```
    length(NsampVec)  # same phis
OverResults <- matrix(0, nrow = NumRuns, ncol = 26)
colnames(OverResults) <- c("NumSim", "NumSamp", "phi1", "phi2", "rhoLA", "rhoLH",
    "rhoAH", "PairBiasLA", "PairBiasLH", "PairBiasAH", "PairLASqRootMSE", "PairLHSqRootMSE",
    "PairLHSqRootMSE", "PairLAAvgSD", "PairLHAvgSD", "PairAHAvgSD", "GMMBiasLA",
    "GMMBiasLH", "GMMBiasAH", "GMMLASqRootMSE", "GMMLHSqRootMSE", "GMMAHSqRootMSE",
    "GMMLAAvgSE", "GMMLHAvgSE", "GMMAHAvgSE", "RunTime")
ResultsSim <- matrix(0, nrow = nSim, ncol = 12)
set.seed(123457)
iResultCount <- 0

for (iNsamp in 1:length(NsampVec)) {
    for (iExternalphi1 in 1:length(Externalphi1Vec)) {
        # for (iExternalphi2 in 1:length(Externalphi2Vec)) {
        for (irhoLA in 1:length(rhoLAVec)) {
            for (irhoLH in 1:length(rhoLHVec)) {
                for (irhoAH in 1:length(rhoAHVec)) {
                  iResultCount <- iResultCount + 1
                  OverResults[iResultCount, 1] <- nSim
                  OverResults[iResultCount, 2] <- Nsamp <- NsampVec[iNsamp]
                  OverResults[iResultCount, 3] <- Externalphi1 <- Externalphi1Vec[iExternalphi1]
                  # OverResults[iResultCount,4] <- Externalphi2Vec[iExternalphi2] ->
                  # Externalphi2
                  OverResults[iResultCount, 4] <- Externalphi2  #same phisss
 <- Externalphi1Vec[iExternalphi1]
                  OverResults[iResultCount, 5] <- rhoLA <- rhoLAVec[irhoLA]
                  OverResults[iResultCount, 6] <- rhoLH <- rhoLHVec[irhoLH]
                  OverResults[iResultCount, 7] <- rhoAH <- rhoAHVec[irhoAH]

                  # Use Conditional/partial correlation for optimization
                  rhoAH_L <- (rhoAH - rhoLA * rhoLH)/sqrt((1 - rhoLA^2) * (1 -
                    rhoLH^2))
                  trueparam <- c(rhoLA, rhoLH, rhoAH_L)
                  trueparamOriginal <- c(rhoLA, rhoLH, rhoAH)
                  ################################################# Start Simulation Loop
                  for (iSim in 1:nSim) {
                    # Generate Covariates
                    Z <- Generate_Covariates(Nsamp)
                    # Generate Data
                    SampleData <- Generate_SampleData(param = trueparam, Nsamp = Nsamp,
                      Externalphi1, Externalphi2, Z)
                    # Fit Regression
                    SampleData <- MargRegress(SampleData)
                    # Reshape the data
                    SampleData <- SampleData[order(-SampleData$PolID, SampleData$year),
                      ]
                    calcOrder <- 1:(length(SampleData$PolID)/p)
                    totalallmydata <- Create_Data_Subsets(SampleData)
                    mydataLapse <- totalallmydata[[1]]
                    mydataNoLapse <- totalallmydata[[2]]
                    mydata1 <- totalallmydata[[3]]
                    mydata2 <- totalallmydata[[4]]
                    mydata3 <- totalallmydata[[5]]
                    mydata4 <- totalallmydata[[6]]
                    mydata <- totalallmydata[[7]]
                    ######################### Pairwise First - For Starting Values
                    toler <- 0.25  #0.4
                    trueparam <- c(rhoLA, rhoLH, rhoAH_L)
                    lbound <- trueparam - toler * c(1, 1, 0.4)
                    lbound <- pmax(-0.8, lbound)
                    ubound <- trueparam + toler * c(1, 1, 0.4)
                    ubound <- pmin(0.8, ubound)
                    opLA <- optim(0, BiLikeLA, method = c("L-BFGS-B"), lower = lbound[1],
                      upper = ubound[1], hessian = TRUE)
                    tryCatch(PairSELA <- sqrt(diag(ginv(opLA$hessian))), error = function(e) {
                      PairSELA <- 0
                    })
                    opLH <- optim(0, BiLikeLH, method = c("L-BFGS-B"), lower = lbound[2],
                      upper = ubound[2], hessian = TRUE)
                    tryCatch(PairSELH <- sqrt(diag(ginv(opLH$hessian))), error = function(e) {
                      PairSELH <- 0
                    })
                    opAH <- optim(0, BiLikeAH, method = c("L-BFGS-B"), lower = lbound[3],
                      upper = ubound[3], hessian = TRUE)
                    tryCatch(PairSEAH <- sqrt(diag(ginv(opAH$hessian))), error = function(e) {
                      PairSEAH <- 0
                    })
                    ResultsSim[iSim, 1:3] <- rbind(opLA$par, opLH$par, opAH$par)
                    ResultsSim[iSim, 4:6] <- rbind(PairSELA, PairSELH, PairSEAH)
                    ######################### GMM parameter estimate
                    GMMInit <- c(opLA$par, opLH$par, opAH$par)
                    GMMgthetaInit <- GMMgthetaFunct(GMMInit)
                    GMMInitdev <- GMMgthetaInit - matrix(1, nrow = length(GMMgthetaInit[,
                      1]), ncol = 1) %*% colMeans(GMMgthetaInit)
                    Vargtheta <- t(GMMInitdev) %*% GMMInitdev/length(GMMgthetaInit[,
                      1])
                    GMMResult2 <- optim(par = GMMInit, GMMFunc, method = c("L-BFGS-B"),
                      lower = lbound, upper = ubound, control = list(factr = 10^12))
                    GMMEst <- GMMResult2$par
                    # Standard Error Adjustments for Reparameterization
                    GTransform <- matrix(c(1, 0, 0, 0, 1, 0, GMMEst[2] - GMMEst[1] *
                      GMMEst[3] * sqrt((1 - GMMEst[2]^2)/(1 - GMMEst[1]^2)),
                      GMMEst[1] - GMMEst[2] * GMMEst[3] * sqrt((1 - GMMEst[1]^2)/(1 -
                        GMMEst[2]^2)), sqrt((1 - GMMEst[1]^2) * (1 - GMMEst[2]^2))),
                      nrow = 3, ncol = 3)
                    GMMgthetaSumFunct <- function(param) {
                      colSums(GMMgthetaFunct(param))
                    }
                    gradient <- jacobian(func = GMMgthetaSumFunct, GMMEst, method = "simple",
                      method.args = list(eps = 0.005))
                    GMMgtheta <- GMMgthetaFunct(GMMEst)
                    Vargtheta <- t(GMMgtheta) %*% GMMgtheta/length(GMMgtheta[,
                      1])
                    GMMVar <- t(gradient) %*% ginv(Vargtheta) %*% gradient/length(GMMgtheta[,
                      1])
                    TransformGMMVar <- t(GTransform) %*% ginv(GMMVar) %*% GTransform
                    tryCatch(GMMstderror <- sqrt(diag(TransformGMMVar)), error = function(e) {
                      GMMstderror <- 0 * TransformGMMVar
                    })
                    ResultsSim[iSim, 10:12] <- GMMstderror
                    GMMEst[3] <- GMMEst[1] * GMMEst[2] + GMMEst[3] * sqrt((1 -
                      GMMEst[1]^2) * (1 - GMMEst[2]^2))
                    ResultsSim[iSim, 7:9] <- GMMEst
                  }
                  # This finishes the simulation loop

                  ######################### round(ResultsSim,digits=3)
                  ResultsSim[is.na(ResultsSim)] <- 0
                  AverResults <- colMeans(ResultsSim)
                  VarResults <- colMeans(ResultsSim * ResultsSim) - (colMeans(ResultsSim))^2

                  OverResults[iResultCount, 8:10] <- AverResults[1:3] - trueparamOriginal[c(1,
                    2, 3)]
                  OverResults[iResultCount, 11:13] <- sqrt(VarResults[1:3] +
                    OverResults[iResultCount, 8:10]^2)
                  OverResults[iResultCount, 14:16] <- AverResults[4:6]
                  OverResults[iResultCount, 17:19] <- AverResults[7:9] - trueparamOriginal
                  OverResults[iResultCount, 20:22] <- sqrt(VarResults[7:9] +
                    OverResults[iResultCount, 17:19]^2)
                  OverResults[iResultCount, 23:25] <- AverResults[10:12]
                  OverResults[iResultCount, 26] <- Sys.time() - time1
                  time1 <- Sys.time()
                }
                write.csv(OverResults, "LapseGMMSim01May2018_2000B.csv", row.names = F)
```

```
            }
        }
    }
}  #}
round(OverResults, digits = 4)

NonConvergeTweedie1
NonConvergeTweedie2
```

4.5 Simulation Results

Effects of Sample Size and Dispersion Parameters

The following table shows the performance of the *GMM* lapse estimator by varying the sample size and dispersion parameters, ϕ_1 (for auto), and ϕ_2 (for home). For this table, we used $\rho_{LA} = -0.2$, $\rho_{LH} = 0.2$, and $\rho_{AH} = 0.1$ for the association parameters, these being comparable to the results of our empirical work. Further, we use $\phi_1 = \phi_2$ for simplicity. For smaller samples, $n = 100, 250$, we used 500 simulations to make sure that the bias was being determined appropriately. This was less of a concern with larger sample sizes, $n = 500, 1000, 2000$, and so for convenience we used 100 simulations in this study.

Some aspects of the results are consistent with our *GMM* study (without lapse). As the dispersion parameters ϕ increase, there are more discrete observations resulting in larger biases and standard errors for all sample sizes. The magnitude of biases suggest that our general procedure may not be suitable for sample sizes as small as $n = 100$. However, even for $n = 250$ (and above), we deem their performance acceptable on the bias criterion.

In contrast, for the standard error criterion, we view the smaller sample sizes $n = 100, 250$ as unacceptable. For example, if $n = 100$, $\phi_1 = \phi_2 = 500$, and $\rho_{LA} = -0.2$, it is hard to imagine recommending a procedure where the average standard error is 0.158. Only for nearly continuous data, when $\phi_1 = \phi_2 = 2$, do the standard errors seem desirable with $n = 500$. In general, for more discrete data where $\phi_1 = \phi_2 = 500$, we recommend samples sizes of $n = 2,000$ and more. Most users that we work with are primarily interested in point estimates but also want to say something about statistical significance; reliable standard errors are important.

We use the pairwise estimators as starting values in the more complete *GMM* estimators. In this application, we had few scores available and so it is not surprising that the *GMM* estimators are close to thie pairwise starting values, meaning that the biases are largely equivalent. Interestingly, in almost every case, the standard errors of the pairs are slightly larger than the corresponding *GMM* estimators. The exception occurs for $n = 1000$, $\phi_1 = \phi_2 = 500$ for the **AH** pair. Given the consistency of results for other cases, this may be simply due to simulation error.

Num	Num	ϕ_1	Pair	Bias		Pair	Std	Err	GMM	Bias		GMM	Std	Err	Time
Sim	Samp	ϕ_2	LA	LH	AH	LA	LH	AH	LA	LH	AH	LA	LH	AH	Taken
500	100	2	−0.004	0.008	0.003	0.095	0.094	0.049	−0.004	0.007	−0.001	0.092	0.091	0.049	1.28
500	100	42	−0.006	−0.011	0.004	0.098	0.096	0.053	−0.006	−0.011	−0.001	0.096	0.094	0.053	1.04
500	100	500	−0.035	−0.013	0.014	0.187	0.144	0.136	−0.037	−0.010	−0.015	0.158	0.141	0.132	1.34
500	250	2	−0.002	0.001	0.000	0.059	0.059	0.031	−0.002	0.000	0.000	0.059	0.058	0.031	3.20
500	250	42	0.005	0.002	0.000	0.063	0.061	0.034	0.005	0.002	0.000	0.062	0.060	0.034	2.58
500	250	500	−0.018	−0.016	0.012	0.111	0.091	0.086	−0.019	−0.016	−0.002	0.108	0.090	0.086	3.54
100	500	2	−0.002	0.001	−0.002	0.042	0.042	0.022	−0.001	0.000	−0.002	0.042	0.041	0.022	1.61
100	500	42	−0.002	−0.005	−0.003	0.044	0.043	0.024	−0.002	−0.005	−0.003	0.044	0.043	0.024	1.27
100	500	500	−0.012	−0.009	0.003	0.078	0.064	0.061	−0.011	−0.008	−0.006	0.076	0.063	0.061	1.94
100	1000	2	−0.003	0.002	0.000	0.030	0.030	0.016	−0.002	0.002	0.000	0.030	0.029	0.015	3.22
100	1000	42	0.009	0.001	−0.002	0.031	0.030	0.017	0.009	0.001	−0.002	0.031	0.030	0.017	2.62
100	1000	500	−0.001	−0.003	0.001	0.055	0.045	0.043	−0.002	−0.003	−0.002	0.054	0.045	0.044	4.79
100	2000	2	−0.001	0.002	0.001	0.021	0.021	0.011	0.000	0.001	0.001	0.021	0.021	0.011	5.06
100	2000	42	0.004	−0.005	0.001	0.022	0.022	0.012	0.004	−0.005	0.001	0.022	0.021	0.012	4.13
100	2000	500	0.004	0.000	−0.002	0.039	0.032	0.031	0.004	0.000	−0.003	0.038	0.032	0.031	6.06

Effects of Association Parameters

This section investigates the performance of the *GMM* lapse estimator by varying the size of the association parameters. Consistent with earlier results on the performance of the estimators, We used $n = 2000$ for the sample size and 100 for the number of simulations. Further, this section only reports results for $\phi_1 = \phi_2 = 42$, the case where approximately half the outcomes are zero and the other half a continuous amount. We are interested in the performance of these *GMM* copula based estimators in the presence of a significant amount of discrete data.

The following table shows little difference in the performance of the *GMM* due to changes in the association parameters. This suggests that the estimator is robust to the specification of values of ρ.

ρ_{LA}	ρ_{LH}	ρ_{AH}	**GMM**	**Bias**		**GMM**	**Std**	**Error**
			LA	**LH**	**AH**	**LA**	**LH**	**AH**
-0.3	-0.3	-0.3	0.001	0.001	0.000	0.021	0.021	0.011
-0.3	-0.3	0	0.003	-0.002	0.000	0.021	0.021	0.012
-0.3	-0.3	0.3	0.001	0.000	-0.001	0.021	0.021	0.011
-0.3	0	-0.3	0.002	-0.002	0.001	0.021	0.022	0.011
-0.3	0	0	0.001	0.001	0.001	0.022	0.022	0.012
-0.3	0	0.3	-0.002	-0.002	0.000	0.021	0.022	0.011
0.3	-0.3	-0.3	0.001	0.002	0.001	0.020	0.021	0.011
0.3	-0.3	0	0.002	-0.001	0.001	0.020	0.021	0.012
0.3	-0.3	0.3	0.003	0.001	0.000	0.020	0.021	0.011
0.3	0	-0.3	0.001	0.002	0.001	0.020	0.022	0.011
0.3	0	0	0.002	-0.004	0.001	0.020	0.022	0.012
0.3	0	0.3	0.005	0.004	0.001	0.021	0.023	0.011

4.6 Appendix A. Lapse Likelihood

With the independence over time, the joint distribution function is

$$\begin{aligned}
&\Pr\left(L_{i1} \le r_1, \ldots, L_{im} \le r_m, Y_{1,i1} \le y_{11}, \ldots, Y_{1,im} \le y_{1m}, Y_{2,i1} \le y_{21}, \ldots, Y_{2,im} \le y_{2m}\right) \\
&\quad = \textstyle\prod_{t=1}^{m} \Pr\left(L_{it} \le r_t, Y_{1,it} \le y_{1t}, Y_{2,it} \le y_{2t}\right) \\
&\quad = \textstyle\prod_{t=1}^{m} C\left(F_{Lit}(r_t), F_{1,it}(y_{1t}), F_{2,it}(y_{2t})\right).
\end{aligned}$$

Here, F_{Lit}, $F_{1,it}$, and $F_{2,it}$ represent the marginal distributions of L_{it}, $Y_{1,it}$, and $Y_{2,it}$, respectively.

Dependence among these three random variables is modeled using a Gaussian copula C with dependence parameters

$$\mathbf{\Sigma} = \left(\begin{array}{ccc} 1 & \rho_{LA} & \rho_{LH} \\ \rho_{LA} & 1 & \rho_{AH} \\ \rho_{LH} & \rho_{AH} & 1 \end{array} \right).$$

It is convenient to introduce the time to lapse variable T_i that represents the time that the ith policyholder lapses. Specifically,

$$T_i = \left\{ \begin{array}{cl} 1 & \text{if } L_{i1} = 1 \\ 2 & \text{if } L_{i1} = 0, L_{i2} = 1 \\ \vdots & \quad\vdots \\ t & \text{if } L_{i1} = 0, \ldots, L_{i,t-1} = 0, L_{i,t} = 1 \\ \vdots & \quad\vdots \\ m & \text{if } L_{i1} = 0, \ldots, L_{i,m-1} = 0, L_{im} = 1 \\ m+1 & \text{if } L_{i1} = 0, \ldots, L_{i,m} = 0 \end{array} \right.$$

We can organize the observed data based on the time to lapse variable. Specifically, suppose that we observe $T_i = t$ outcome periods for $t = 1, \ldots, m+1$. Then, the observed likelihood is based on the distribution function

$$\begin{aligned}&\Pr\left(T_i = t, Y_{1,i1} \leq y_{11}, \ldots, Y_{1,it} \leq y_{1t}, Y_{2,i1} \leq y_{21}, \ldots, Y_{2,it} \leq y_{2t}\right)\\ &\quad = \Pr\left(L_{i1} = 0, \ldots, L_{i,t-1} = 0, L_{i,t} = 1, L_{i,t+1} \leq \infty, \ldots, L_{im} \leq \infty,\right.\\ &\qquad\qquad Y_{1,i1} \leq y_{11}, \ldots, Y_{1,it} \leq y_{1t}, Y_{1,i,t+1} \leq \infty, \ldots, Y_{1,im} \leq \infty,\\ &\qquad\qquad \left.Y_{2,i1} \leq y_{21}, \ldots, Y_{2,it} \leq y_{2t}, Y_{2,i,t+1} \leq \infty, \ldots, Y_{2,im} \leq \infty\right)\end{aligned}$$

If a policy is renewed for all m periods, then $T_i = m + 1$ and the observed likelihood is based on

$$\begin{aligned}&\Pr\left(T_i = m + 1, Y_{1,i1} \leq y_{11}, \ldots, Y_{1,im} \leq y_{1m}, Y_{2,i1} \leq y_{21}, \ldots, Y_{2,im} \leq y_{2m}\right)\\ &\quad = \prod_{s=1}^{m} C\left(F_{Lis}(0), F_{1,is}(y_{1s}), F_{2,is}(y_{2s})\right).\end{aligned}$$

If lapse occurs, then $T_i \leq m$ and the observed likelihood is based on the distribution function

$$\begin{aligned}&\Pr\left(T_i = t, Y_{1,i1} \leq y_{11}, \ldots, Y_{1,it} \leq y_{1t}, Y_{2,i1} \leq y_{21}, \ldots, Y_{2,it} \leq y_{2t}\right)\\ &\quad = \left\{C\left(1, F_{1,it}(y_{1t}), F_{2,it}(y_{2t})\right) - C\left(F_{Lit}(0), F_{1,it}(y_{1t}), F_{2,it}(y_{2t})\right)\right\} \prod_{s=1}^{t-1} C\left(F_{Lis}(0), F_{1,is}(y_{1s}), F_{2,is}(y_{2s})\right).\end{aligned}$$

Note that the evaluation of this function involves a trivariate copula.

For the time to lapse variable, the marginal distribution is

$$\begin{aligned}\Pr\left(T_i = t\right) &= \Pr\left(T_i = t, Y_{1,i1} \leq \infty, \ldots, Y_{1,it} \leq \infty, Y_{2,i1}\infty, \ldots, Y_{2,it} \leq \infty\right)\\ &= \begin{cases} \prod_{s=1}^{m} F_{Lis}(0) & t = m + 1\\ \left(1 - F_{Li,t}(0)\right) \prod_{s=1}^{t-1} F_{Lis}(0) & 1 \leq t \leq m.\end{cases}\end{aligned}$$

Thus, the conditional distribution function is

$$\begin{aligned}&\Pr\left(Y_{1,i1} \leq y_{11}, \ldots, Y_{1,it} \leq y_{1t}, Y_{2,i1} \leq y_{21}, \ldots, Y_{2,it} \leq y_{2t} | T_i = t\right)\\ &\quad = \frac{\Pr(T_i=t, Y_{1,i1} \leq y_{11}, \ldots, Y_{1,it} \leq y_{1t}, Y_{2,i1} \leq y_{21}, \ldots, Y_{2,it} \leq y_{2t})}{\Pr(T_i = t)}\\ &\quad = \left(\frac{C(1, F_{1,it}(y_{1t}), F_{2,it}(y_{2t})) - C(F_{Lit}(0), F_{1,it}(y_{1t}), F_{2,it}(y_{2t}))}{1 - F_{Lit}(0)}\right)^{I(t \leq m)} \prod_{s=1}^{t-1} \frac{C(F_{Lis}(0), F_{1,is}(y_{1s}), F_{2,is}(y_{2s}))}{F_{Lis}(0)}\end{aligned}$$

for $t = 1, \ldots, m + 1$. From this, the corresponding conditional distribution function is

$$\Pr\left(Y_{1,is} \leq y_1, Y_{2,is} \leq y_2 | T_i = t\right) = \begin{cases} \frac{C(F_{Lis}(0), F_{1,is}(y_1), F_{2,is}(y_2))}{F_{Lis}(0)} & \text{for } s < t \leq m + 1\\ \frac{C(1, F_{1,is}(y_1), F_{2,is}(y_2)) - C(F_{Lis}(0), F_{1,is}(y_1), F_{2,is}(y_2))}{1 - F_{Lis}(0)} & \text{for } s = t \leq m \end{cases}$$

Thus, the hybrid mass function/density has different expressions when (i) no lapse is involved and when (ii) there is lapse involved.

No Lapse

For the first case where lapse has not yet occurred, $s < t \leq m + 1$, this has hybrid probability density/mass function

$$f_{i12,s|t}(y_1, y_2) = \frac{1}{F_{Lis}(0)} \begin{cases} C\left(F_{Lis}(0), F_{1,is}(0), F_{2,is}(0)\right) & y_1 = 0, y_2 = 0\\ C_2\left(F_{Lis}(0), F_{1,is}(y_1), F_{2,is}(0)\right) f_{1,is}(y_1) & y_1 > 0, y_2 = 0\\ C_3\left(F_{Lis}(0), F_{1,is}(0), F_{2,is}(y_2)\right) f_{2,is}(y_2) & y_1 = 0, y_2 > 0\\ C_{23}\left(F_{Lis}(0), F_{1,is}(y_1), F_{2,is}(y_2)\right) f_{1,is}(y_1) f_{2,is}(y_2) & y_1 > 0, y_2 > 0 \end{cases}$$

Here, $C_2(u_1, u_2, u_3) = \frac{\partial}{\partial u_2} C(u_1, u_2, u_3)$ represents the partial derivative of the copula with respect to the second argument and similarly for C_3. The term C_{23} is a second derivative with respect to the second and third arguments. Further $f_{j,is}$ is the density function corresponding to the distribution function $F_{j,is}$.

Lapse

For the second case where lapse has occurred, $s = t \leq m$, this has hybrid probability density/mass function

$$f_{i12,s|t}(y_1, y_2) = \tfrac{1}{1-F_{Lit}(0)} \times$$

$$\begin{cases} \sum_{i=0}^{1} (-1)^i \, C\left(F_{Lis}(0)^i, F_{1,is}(0), F_{2,is}(0)\right) & y_1 = 0, y_2 = 0 \\ \left\{\sum_{i=0}^{1} (-1)^i \, C_2\left(F_{Lis}(0)^i, F_{1,is}(y_1), F_{2,is}(0)\right)\right\} f_{1,is}(y_1) & y_1 > 0, y_2 = 0 \\ \left\{\sum_{i=0}^{1} (-1)^i \, C_3\left(F_{Lis}(0)^i, F_{1,is}(0), F_{2,is}(y_2)\right)\right\} f_{2,is}(y_2) & y_1 = 0, y_2 > 0 \\ \left\{\sum_{i=0}^{1} (-1)^i \, C_{23}\left(F_{Lis}(0)^i, F_{1,is}(y_1), F_{2,is}(y_2)\right)\right\} f_{1,is}(y_1) f_{2,is}(y_2) & y_1 > 0, y_2 > 0. \end{cases}$$

Using this notation, the logarithmic likelihood is

$$L = \sum_{i=1}^{n} \sum_{s=1}^{t_i \wedge m} \ln f_{i12s|t_i}(y_{1is}, y_{2is}).$$

4.7 Appendix B. Trivariate Gaussian Copula Derivatives with Respect to Association Parameters

From the score function, we see we need to evaluate derivatives of a trivariate copula. We evaluate $\frac{\partial}{\partial \rho} C(u_1, u_2, u_3)$ in Appendix B.1, $\frac{\partial}{\partial \rho} C_2(u_1, u_2, u_3)$ in Appendix B.2, and $\frac{\partial}{\partial \rho} C_{23}(u_1, u_2, u_3)$ in Appendix B.3. For simplicity, we use the following generic expression for the association matrix

$$\boldsymbol{\Sigma} = \left(\begin{array}{ccc} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{array} \right).$$

Appendix B.1. No Derivatives

We first cite a general result due to Plackett (1954). Consider a d dimensional multivariate normal distribution with variance-covariance matrix $\boldsymbol{\Sigma}$. As we will use this as a basis for defining copulas, consider the mean to be zero and variance to be 1 so that the diagonal elements of $\boldsymbol{\Sigma}$ equal 1. Let $\Phi_d(\cdot; \boldsymbol{\Sigma})$ be the corresponding distribution function. Partition the matrix as

$$\boldsymbol{\Sigma} = \left(\begin{array}{cc} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}' & \boldsymbol{\Sigma}_{22} \end{array} \right) \qquad \boldsymbol{\Sigma}_{11} = \left(\begin{array}{cc} 1 & \rho_{12} \\ \rho_{12} & 1 \end{array} \right),$$

so that $\boldsymbol{\Sigma}_{11}$ is the submatrix for the first two elements and ρ_{12} is the corresponding correlation coefficient. Then, from Plackett (1954), we have

$$\frac{\partial}{\partial \rho_{12}} \Phi_d(\mathbf{z}; \boldsymbol{\Sigma}) = \phi_2(\mathbf{z}_1; \boldsymbol{\Sigma}_{11}) \; \Phi_{d-2}(\mathbf{z}_2^*; \boldsymbol{\Sigma}_{22 \cdot 1}),$$

where $\phi_2(\cdot)$ is a bivariate normal density, $\mathbf{z}_1 = (z_1, z_2)'$, and

$$\mathbf{z}_2^* = \left(\begin{array}{c} z_3 \\ \vdots \\ z_d \end{array} \right) - \boldsymbol{\Sigma}_{12}' \boldsymbol{\Sigma}_{11}^{-1} \mathbf{z}_1 \quad \text{and} \quad \boldsymbol{\Sigma}_{22 \cdot 1} = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{12}' \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{12}.$$

Starting with uniform random variables u_j, we define the normal scores $z_j = \Phi^{-1}(u_j)$. For $d = 3$, we have

$$\frac{\partial}{\partial \rho_{12}} C(u_1, u_2, u_3) = \phi_2 \left(\left(\begin{array}{c} z_1 \\ z_2 \end{array} \right); \boldsymbol{\Sigma}_{11} \right) \; \Phi(z_3^*; \Sigma_{22 \cdot 1}).$$

where

$$z_3^* = z_3 - \boldsymbol{\Sigma}_{12}'\boldsymbol{\Sigma}_{11}^{-1}\begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \quad \boldsymbol{\Sigma}_{11} = \begin{pmatrix} 1 & \rho_{12} \\ \rho_{12} & 1 \end{pmatrix}, \quad \boldsymbol{\Sigma}_{12} = \begin{pmatrix} \rho_{13} \\ \rho_{23} \end{pmatrix}, \quad \Sigma_{22\cdot1} = 1 - \boldsymbol{\Sigma}_{12}'\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}.$$

Appendix B.2. One Derivative

We next derive the partial derivative with respect to the association parameter of the partial derivative of the copula function $C_3(u_1, u_2, u_3)$. For a derivative with respect to one argument, we have

$$C_3(u_1, u_2, u_3) = C(u_1, u_2|u_3) = \Phi_2(z_1 - \mu_{12\cdot3,1}, z_2 - \mu_{12\cdot3,2}; \boldsymbol{\Sigma}_{12\cdot3})$$

where

$$\boldsymbol{\mu}_{12\cdot3} = \begin{pmatrix} \mu_{12\cdot3,1} \\ \mu_{12\cdot3,2} \end{pmatrix} = z_3\begin{pmatrix} \rho_{13} \\ \rho_{23} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_{12\cdot3} = \begin{pmatrix} 1 & \rho_{12} \\ \rho_{12} & 1 \end{pmatrix} - \begin{pmatrix} \rho_{13} \\ \rho_{23} \end{pmatrix}\begin{pmatrix} \rho_{13} & \rho_{23} \end{pmatrix}.$$

Recall, for two mean zero, variance one, normally distributed random variables X_1 and X_2, that the conditional distribution of X_2 given $X_1 = x_1$ is normally distributed with mean $\rho_X x_1$ and variance $(1-\rho_X^2)$. Because $\frac{\partial}{\partial x_1}\Pr(X_1 \le x_1, X_2 \le x_2) = f_{X_1}(x_1)\Pr(X_2 \le x_2|X_1 = x_1)$, we may define

$$h_1^*(x_1, x_2, \rho_X) = \frac{\partial}{\partial x_1}\Phi_2(x_1, x_2, \rho_X) = \phi(x_1)\,\Phi\left(\frac{x_2 - \rho_X x_1}{\sqrt{1-\rho_X^2}}\right).$$

Consider a matrix $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$ and

$$\Phi_2(x_1, x_2, \boldsymbol{\Sigma}) = \Phi_2\left(\frac{x_1}{\sigma_1}, \frac{x_2}{\sigma_2}, \rho_x\right) = C\left(\Phi\left(\frac{x_1}{\sigma_1}\right), \Phi\left(\frac{x_2}{\sigma_2}\right), \rho_x\right)$$

where $\rho_x = \sigma_{12}/(\sigma_1\sigma_2)$.

We now evaluate a derivative of this using the matrix $\boldsymbol{\Sigma}_{12\cdot3}$, so that $\sigma_1^2 = 1 - \rho_{13}^2$, $\sigma_2^2 = 1 - \rho_{23}^2$, and $\rho_X\sigma_1\sigma_2 = \rho_{12} - \rho_{13}\rho_{23}$. With this notation, we have

$$\begin{aligned} \tfrac{\partial}{\partial\rho}\Phi_2(z_1 - \mu_{12\cdot3,1}, z_2 - \mu_{12\cdot3,2}; \boldsymbol{\Sigma}_{12\cdot3}) &= \tfrac{\partial}{\partial\rho}\Phi_2\left(\tfrac{z_1-\mu_{12\cdot3,1}}{\sigma_1}, \tfrac{z_2-\mu_{12\cdot3,2}}{\sigma_2}; \rho_x\right) \\ &= \tfrac{\partial}{\partial\rho}\Phi_2(z_1^*, z_2^*; \rho_x) \\ &= h_1^*(z_1^*, z_2^*; \rho_x)\tfrac{\partial}{\partial\rho}z_1^* + h_1^*(z_2^*, z_1^*; \rho_x)\tfrac{\partial}{\partial\rho}z_2^* + \phi_2(z_1^*, z_2^*; \rho_x)\tfrac{\partial}{\partial\rho}\rho_x, \end{aligned}$$

where $z_1^* = (z_1 - \mu_{12\cdot3,1})/\sigma_1$ and similarly for z_2^*. The last equality uses a special case of the Plackett (1954) result

$$\frac{\partial}{\partial\rho_x}\Phi_2(y_1, y_2; \boldsymbol{\Sigma}) = \phi_2(y_1, y_2; \boldsymbol{\Sigma})$$

where $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho_x \\ \rho_x & 1 \end{pmatrix}$. We also need

$$\frac{\partial}{\partial\rho}z_1^* = \frac{\partial}{\partial\rho}\left(\frac{z_1 - \mu_{12\cdot3,1}}{\sigma_1}\right) = \frac{\partial}{\partial\rho}\left(\frac{z_1 - z_3\rho_{13}}{\sqrt{1-\rho_{13}^2}}\right) = \begin{cases} 0 & \rho = \rho_{12} \\ \frac{z_1\rho_{13} - z_3}{(1-\rho_{13}^2)^{3/2}} & \rho = \rho_{13} \\ 0 & \rho = \rho_{23} \end{cases}$$

In the same way

$$\frac{\partial}{\partial\rho}z_2^* = \frac{\partial}{\partial\rho}\left(\frac{z_2 - \mu_{12\cdot3,2}}{\sigma_2}\right) = \frac{\partial}{\partial\rho}\left(\frac{z_2 - z_3\rho_{23}}{\sqrt{1-\rho_{23}^2}}\right) = \begin{cases} 0 & \rho = \rho_{12} \\ 0 & \rho = \rho_{13} \\ \frac{z_2\rho_{23} - z_3}{(1-\rho_{23}^2)^{3/2}} & \rho = \rho_{23} \end{cases}$$

Similarly,

$$\frac{\partial}{\partial \rho}\rho_x = \frac{\partial}{\partial \rho}\left(\frac{\rho_{12}-\rho_{13}\rho_{23}}{\sigma_1\sigma_2}\right) = \frac{\partial}{\partial \rho}\left(\frac{\rho_{12}-\rho_{13}\rho_{23}}{(1-\rho_{13}^2)^{1/2}(1-\rho_{23}^2)^{1/2}}\right) = \begin{cases} \frac{1}{(1-\rho_{13}^2)^{1/2}(1-\rho_{23}^2)^{1/2}} & \rho=\rho_{12} \\ \frac{\rho_{12}\rho_{13}-\rho_{23}}{(1-\rho_{13}^2)^{3/2}(1-\rho_{23}^2)^{1/2}} & \rho=\rho_{13} \\ \frac{\rho_{12}\rho_{23}-\rho_{13}}{(1-\rho_{13}^2)^{1/2}(1-\rho_{23}^2)^{3/2}} & \rho=\rho_{23} \end{cases}$$

Summarizing

$$\tfrac{\partial}{\partial \rho}C_3\left(u_1,u_2,u_3\right) = h_1^*\left(z_1^*,z_2^*;\rho_x\right)\tfrac{\partial}{\partial \rho}z_1^* + h_1^*\left(z_2^*,z_1^*;\rho_x\right)\tfrac{\partial}{\partial \rho}z_2^* + \phi_2\left(z_1^*,z_2^*;\rho_x\right)\tfrac{\partial}{\partial \rho}\rho_x,$$

where $z_1^*=(z_1-z_3\rho_{13})/\sigma_1$, $z_2^*=(z_2-z_3\rho_{23})/\sigma_2$, $\sigma_1^2=1-\rho_{13}^2$, $\sigma_2^2=1-\rho_{23}^2$, and $\rho_X\sigma_1\sigma_2=\rho_{12}-\rho_{13}\rho_{23}$, and

$$h_1^*(x_1,x_2,\rho_X) = \phi\left(x_1\right)\Phi\left(\frac{x_2-\rho_X x_1}{\sqrt{1-\rho_X^2}}\right).$$

Appendix B.3. Two Derivatives

We next derive the partial derivative with respect association parameter of the copula function $C_{23}\left(u_1,u_2,u_3\right)$. For derivatives with respect to two arguments, we have

$$C_{23}\left(u_1,u_2,u_3\right) = C\left(u_1|u_2,u_3\right)c\left(u_2,u_3\right) \quad \text{with} \quad C\left(u_1|u_2,u_3\right) = \Phi\left(z_1-\mu_{1\cdot 23};\sigma_{1\cdot 23}\right),$$

where

$$\mu_{1\cdot 23} = \begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix} \quad \text{and} \quad \sigma_{1\cdot 23} = 1-\begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix}.$$

Now

$$\frac{\partial}{\partial \rho}C_{23}\left(u_1,u_2,u_3\right) = C\left(u_1|u_2,u_3\right)\frac{\partial}{\partial \rho}c\left(u_2,u_3\right) + \left(\frac{\partial}{\partial \rho}C\left(u_1|u_2,u_3\right)\right)c\left(u_2,u_3\right).$$

Further,

$$\begin{aligned} \tfrac{\partial}{\partial \rho}C\left(u_1|u_2,u_3\right) &= \tfrac{\partial}{\partial \rho}\Phi\left(z_1-\mu_{1\cdot 23};\sigma_{1\cdot 23}\right) \\ &= \phi\left(\tfrac{z_1-\mu_{1\cdot 23}}{\sqrt{\sigma_{1\cdot 23}}}\right)\tfrac{\partial}{\partial \rho}\left(\tfrac{z_1-\mu_{1\cdot 23}}{\sqrt{\sigma_{1\cdot 23}}}\right) \\ &= -\phi\left(\tfrac{z_1-\mu_{1\cdot 23}}{\sqrt{\sigma_{1\cdot 23}}}\right)\tfrac{1}{\sigma_{1\cdot 23}^{3/2}}\left(\sigma_{1\cdot 23}\tfrac{\partial}{\partial \rho}\mu_{1\cdot 23} + \tfrac{1}{2}(z_1-\mu_{1\cdot 23})\tfrac{\partial}{\partial \rho}\sigma_{1\cdot 23}\right). \end{aligned}$$

First recall $\frac{\partial}{\partial \rho}\mathbf{\Sigma}^{-1} = -\mathbf{\Sigma}^{-1}\left(\frac{\partial}{\partial \rho}\mathbf{\Sigma}\right)\mathbf{\Sigma}^{-1}$. Now, with $\mu_{1\cdot 23} = \begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix}$, we have

$$\frac{\partial}{\partial \rho}\mu_{1\cdot 23} = \begin{cases} \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix} & \rho=\rho_{12} \\ \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix} & \rho=\rho_{13} \\ \begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} z_2 \\ z_3 \end{pmatrix} & \rho=\rho_{23} \end{cases}$$

Further, with $\sigma_{1\cdot 23} = 1-\begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix}$, we have

$$\frac{\partial}{\partial\rho}\sigma_{1\cdot 23}=\begin{cases} -2\begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix} & \rho=\rho_{12} \\ -2\begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix} & \rho=\rho_{13} \\ -\begin{pmatrix} \rho_{12} & \rho_{13} \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & \rho_{23} \\ \rho_{23} & 1 \end{pmatrix}^{-1}\begin{pmatrix} \rho_{12} \\ \rho_{13} \end{pmatrix} & \rho=\rho_{23} \end{cases}$$

Derivatives of the bivariate density $c\left(u_2,u_3\right)$ follow directly from results of Schepsmeier and Stober (2012, page 2).

Summarizing,

$$\frac{\partial}{\partial\rho}C_{23}\left(u_1,u_2,u_3\right)=\Phi\left(z_1-\mu_{1\cdot 23};\sigma_{1\cdot 23}\right)\frac{\partial}{\partial\rho}c\left(u_2,u_3\right)+\left(\frac{\partial}{\partial\rho}C\left(u_1|u_2,u_3\right)\right)c\left(u_2,u_3\right),$$

where

$$\tfrac{\partial}{\partial\rho}C\left(u_1|u_2,u_3\right)=-\phi\left(\tfrac{z_1-\mu_{1\cdot 23}}{\sqrt{\sigma_{1\cdot 23}}}\right)\tfrac{1}{\sigma_{1\cdot 23}^{3/2}}\left(\sigma_{1\cdot 23}\tfrac{\partial}{\partial\rho}\mu_{1\cdot 23}+\tfrac{1}{2}(z_1-\mu_{1\cdot 23})\tfrac{\partial}{\partial\rho}\sigma_{1\cdot 23}\right).$$

5 Online Supplement 5. Case Study: Joint Modeling of Insurance Claims and Lapsation

5.1 Introduction

In insurance analytics, generalized linear models (GLM) are nowadays a standard procedure for analyzing claims and customers' decision to renew the policies. For instance, automobile and homeowners pricing is regularly calculated based on models for the number and the size of claims. Some analysts have explored time to event approaches inspired by basic survival analysis to understand the duration that a customer stays in a company and some insurers have also tried to predict profitability in the long-run. In doing so, it has become evident that there is a trade-off between price and the decision to renew a contract. There is a natural struggle between price and renewal. The higher the price, lower the probabilities to renew the policy. On the other side, if the price is too low than, the policy holder may stay in the company but could not be profitable at the end. Most companies have separated the process of calculating price and renewal prospects. To us, this is a huge mistake and the aim of this case study is to show precisely, why this is so, and why there is a correlation between pricing a renewal, but not a causal relationship. The situation gets specially complicated when there is more than one single policy, for instance when the policy holder has at least one home Insurance and one motor Insurance. This is the reason why we use this particular example to illustrate the joint modelling methodology.

We start with a panel of insureds which at the beginning of the observation period has one homeowners policy and one motor policy. They are all observed yearly and some abandon the pool at some point because they cancel at least one of those two policies. The study period is five years.

This case study, and the companion paper, considers a joint model of insurance claims and lapsation. For example, if a policyholder is aggressive or a risk seeker (or just careless), then we would expect that customer to have both large auto as well as homeowner claims. As another example, if a policyholder has an auto claim during the year, then we might think that this outcome is related to the decision to renew (or its converse, lapse) an insurance contract because the price of his contract will increase. Using a sample of real data, this case study shows how to predict the size of two types of claims (auto and homeowners) using the Tweedie model. Logistic regression is used for lapse (or renewal) model. The novel aspect is that we specify their joint behavior through a copula. Estimation is done using both a traditional composite likelihood approach as well as a new (in this context) generalized method of moments technique. The models and estimation techniques are built into this demonstration and are not required knowledge in order to review and interactively assess this case study.

Background

Consider the case where we follow policyholders over time. During the year, there are **three** outcome variables of interest. The claims outcomes are

- Y_1 which represents claims from an auto coverage and
- Y_2 which represents claims from a homeowners coverage.

As claims outcomes, these variables may take on value of zero (representing no claim) and are otherwise positive continuous outcomes (representing claim amount). We use subscripts i to distinguish among policyholders and t to distinguish observations over time. Thus, $Y_{1,it}$ and $Y_{2,it}$ represent auto and homeowner claims for the ith policyholder at time t.

The third random variable, L, is a binary variable that represents a policyholder's decision to lapse one of the policies. Specifically,

- $L_{it} = 1$ indicates that the ith policyholder in the tth year decides to lapse one of the policies and

- $L_{it} = 0$ indicates that the ith policyholder in the tth year decides to not lapse the two policies, i.e, to renew.

Note that if $L_{it} = 1$ then we do not observe the policy at time $t + 1$. In the same way, if $L_{it} = 0$, then we observe the policy at time $t + 1$, subject to limitations on the number of time periods available. We use m to represent the maximal number of observations over time.

Associated with each policyholder is a set of (possibly time varying) rating variables $\mathbf{x}_{it}$ for the ith policyholder at time t that is described in Section 2. We represent the marginal distribution of each outcome variables in terms of a generalized linear model. Specifically, following standard insurance industry practice, we represent the marginal distributions of the claims random variables using a Tweedie distribution so that the distributions have a mass at zero and are otherwise positive. The marginal distribution of the renewal variable is modeled using a logit function. Marginal distributions may use common rating variables and so are naturally related in this sense.

The **dependence** among lapse and claims outcomes is captured using a copula function. That is, we allow outcomes from the same time period (and the same policyholder) to depend on one another. This specification permits, for example, large claims to influence the tendency to lapse a policy or a latent variable to simultaneously influence both lapse and claims outcome. Lapsation dictates the availability of data which may be related to the outcomes, a violation of the statistical principle known as *missing at random*. This means that analyzing claims while ignoring lapse can lead to biased estimation. Thus, joint modeling of lapse and claims are critical because the claims model depends on the data observed through the lapsation/renewal process.

This case study is **interactive** in two ways. First, if you are viewing the .html file in a web browser, you will be able to reveal `R` code and output by clicking on the text. For example, you can click on the `Code` button to get a list of the `R` packages needed to run this case study.

```r
# Loading packages
library(tweedie)
library(reshape)
library(statmod)
library(knitr)
library(BB)
library(MASS)
library(copula)
library(numDeriv)
library(VineCopula)
library(mvtnorm)
time1 <- Sys.time()  # define a variable that we can use to check the run time
```

Context and Motivation

We consider insurance policy holders together with their claims experience data from a major general insurance company in Spain. The genuine characteristic of these data is that we have information on motor insurance, homeowners insurance and lapse behavior. The observations cover a period of 5 complete years starting in 2010 and ending in mid-2015, but since not all policies start the January 1st we observed one full year from that date. The only requirement to be part of this sample is that all policyholders are covered by one vehicle policy and one homeowner policy. If one of these contracts is cancelled, then the policy holder is not considered in our group and is called a non-renewal or a "lapse".

We are interested in the dependence of risks, because we assume that there is a relationship between the two coverages, motor and home insurance, but in addition we also believe that having had claims has some relationship with the decision to renew the policy in the next year.

Insurance companies in practice calculate the price to be charged to a customer per line of business and then, if a particular customer has more than one policy, the aggregate price is just the sum of the price of every policy. It is unclear whether there is any reward from having more than one policy, and if there is such bonus, it is generally argued that the rebate is the consequence of a marketing strategy to promote loyalty. Generally, if one client only has one policy, then the price of the insurance contract (i.e. the premium) is calculated based on historical information on this particular line of business in the insurance company.

In our sample, there may be clients that start their contract later than January 2010, so they are observed less than five years. Some others do not renew one of the policies (either home or motor, or both) and leave the study group.

As in many countries, in Spain owners of automobiles are obliged to have some minimum form of insurance coverage for personal injury to third parties. Home insurance is optional. The reasons why citizens decide to have these two types of insurance may be quite different. So, we believe that our sample of customers having the two is not representative of the whole population, because not everyone selects to buy homeowner coverage or can afford to buy it. Even if ownership is vastly extended in Spain, and one may think that motor and home insurance should go together, home insurance coverage is often linked to a mortgage and, so it is not necessarily sold by the same insurance company that is covering motor insurance. In the recent years, many insurers are trying to cross-sell in their existing portfolios, so they have made an enormous effort to identify and offer a home insurance to those having only motor insurance and the other way around.

We have combined information on three different sources for the sampled individuals. First, the customer characteristics including age, gender or driving experience, among others and dates of renewal for the two types of policies considered here. Note that we have only considered clients corresponding to persons and not commercial firms that can also underwrite home and motor insurance policies. Our segment is personal/private customers. Second, the policy data file for motor vehicle insurance consists of all vehicle insurance coverage including power, driving area or whether there is a second driver that drives the car occasionally. Third, the policy data file for homeowners insurance has information on the property such as value of the building (essentially the value of the home without any furniture, apparel and personal items), location and type of dwelling. Besides these three sources, we have access to data containing information on the number of claims and total cost of those claims per year and per policy type. The claims file provides a record of each accident claim filed with the insurer during the observation period and is linked to the corresponding policy file. The payment file consists of information on payments made during the observation period and is linked to the claims file. However, even if it is frequent that claims have multiple payments made (to health providers, to repair shops, to all parties affected by the claim), we only consider here for each claimed accident the total cost that was compensated by the insurer. A few clients have more than one claim per year per policy. In this case, we consider the sum of all claims in the given policy in a two-step procedure: first we consider the number of claims and then the severity of each claim is added to the aggregate yearly claim compensation of that policy. So, for all policies that are in force, we finally have up to a five year record of the yearly cost of claims in the motor insurance and in the home coverage. If the customer does not renew one of those two policies or both, we do not have more information after this lapse occurs.

Insurers consider non-renewals a critical disruption for their business, because they consider that losing a customer implies that the expected benefits of that client in the short-term and long-term future disappear. Since the basis of insurance is the law of large numbers, insurance companies always seek to increase the number of customers in their portfolios. A high lapsing rate is a threat for their activity, so insurers not only constantly look for new customer, but they also make a great afford to retain those customers that they already have.

Our aim is to price insurance clients with more than one policy type according to the risk they have and to take into account the dependence between the two types of coverage together with the possibility that the client decides to cancel one of them. This is not exactly what insurers do in practice. They evaluate risks in a separate analysis, by traditional claim analysis called ratemaking models, than can even be designed by different people in the insurance company, on the one hand those that are responsible for motor insurance, and on the other those that work on home insurance. The sum of the two prices, which have usually been calculated independently, is the premium to be paid by the client. In some companies, the marketing department, evaluates the risk of non-renewal and may establish rebates that are aimed at keeping the customer in the company. (see, Guelman et al. 2015, for price optimization in only one line of business). Frees and Valdez (2008) analyze more than a single type of claim incurred in motor insurance; for example, an automobile accident can result in damages to a driver's own property, as well as damages to a third party involved in the accident.

We model the simultaneous occurrence of motor and home claims and the renewal of the two policies together. This is the unique feature in this case study. From a multivariate analysis perspective, this problem is new, because we rarely observed all three phenomena. One of the difficulties is that claims occur during one particular policy-year, while the renewal decision is taken at the end of that policy-year. Here we show the differences in pricing the two lines separately or jointly. This has many implications in the way the insurance industry can treat the clients with several policies, which are in fact their core clients. We are not shocked to find that claim amounts among motor and home insurance are positively related. It is not surprising that there is a positive association due to wealthy customers that own more expensive cars and more expensive houses have repairs than would be more costly than the rest, but this dependence should already be mostly accounted for in the inclusion of covariates on the power of the car and the value of the house. What we find in our results is that there is a residual dependence that is not captured in the observed risk factors. This positive correlation can be the result of unobserved factors such as carefulness or a general attitude of the customer on risk mitigation. Those that have more accidents and more costly claims in motor insurance tend to coincide with those that have more incidents at home, or those that have more expensive homeowner claim events than the rest. That is an indication of an attitude known as moral hazard, which means that once the policyholder knows that the insurance company compensates for the claims, then the policyholder reduces protection of their goods. What is not yet well understood is the relationship with policy renewal. In the Spanish market we know that claiming induces renewal, due to the fact that the customer cannot find another company. The market is such that companies deny new contracts to new policyholders that recently had a claim, so it is difficult to find a good price elsewhere. What we do not know is whether or not the fact that claims increase implying that renewals also increase makes the companies find an opportunity for a natural diversification. In fact, the larger the claims, the larger are the costs per policy, but in turn, as the odds of renewal increase with claims, the best is the situation of the company in terms of expected benefit.

In the following, this report is split into additional sections.

- Section 2 on summarizing the data. This section provides basic summary statistics to understand data features.
- Section 3 on fitting the marginal distributions. This section provides the usual regression model fitting using Tweedie distributions for claims and a logistic model for lapse. Residuals from the model fits are calculated and used to display patterns among outcomes that are not accounted for in the marginal models.
- Section 4 sets the stage for joint estimation.
  - Section 4.1 on estimation using a pairwise likelihood. A copula (Gaussian) model is specified to accommodate dependencies. This model is fit using pairwise maximum likelihood, as is common in the literature.
  - Section 4.2 using generalized method of moments. This procedure is novel in the copula context.
- Section 5 provides interpretations and implications.

5.2 Summarize the Data

```
# setwd("D:\\RiskCenter\\DataAnalysisiFrees\\Final_Data_5")
SampleData <- read.table("data_ex.csv", header =T, sep = ",")
SampleData$Lapse <-  1*(SampleData$Retention==0)
SampleData$PosClaim1 <- 1*(SampleData$Claims1>0)
SampleData$PosClaim2 <- 1*(SampleData$Claims2>0)
```

This section demonstrates some basic techniques to look at the data.

Basic Summary Statistics

From the unbalanced panel of policyholders over 5 years, there are $N = 122935$ observations in the data set. Of these, there were 29296 lapses, for a lapse rate of 23.83%.

For type 1 (auto) claims, we have 1967 or about 1.6% (positive) claims. For type 2 (home) claims, we have 2189 or about 1.78% (positive) claims.

As is common, we begin by examining basic measures that summarize the distribution of each variable, initially focus on continuous outcomes.

```r
VarsDep <- c("Lapse", "NClaims1", "NClaims2", "Claims1", "Claims2")
SumDatDep <- summary(SampleData[VarsDep])
knitr::kable(SumDatDep, digits = 2, caption = "Dependent Variable Summary Measures")
```

Table 1: Dependent Variable Summary Measures

Lapse	NClaims1	NClaims2	Claims1	Claims2
Min. :0.0000	Min. :0.00000	Min. :0.00000	Min. : 0.00	Min. : 0.000
1st Qu.:0.0000	1st Qu.:0.00000	1st Qu.:0.00000	1st Qu.: 0.00	1st Qu.: 0.000
Median :0.0000	Median :0.00000	Median :0.00000	Median : 0.00	Median : 0.000
Mean :0.2383	Mean :0.02977	Mean :0.03415	Mean : 26.41	Mean : 7.939
3rd Qu.:0.0000	3rd Qu.:0.00000	3rd Qu.:0.00000	3rd Qu.: 0.00	3rd Qu.: 0.000
Max. :1.0000	Max. :6.00000	Max. :4.00000	Max. :109720.71	Max. :27279.820

```r
VarsExplan <- c("Age_client","Client_Seniority","Car_power_M",
                "Insuredcapital_continent_re")
SumDatExplan <- summary(SampleData[VarsExplan])
knitr::kable(SumDatExplan,digits=3, caption="Continuous Variable Summary Measures")
```

Table 2: Continuous Variable Summary Measures

Age_client	Client_Seniority	Car_power_M	Insuredcapital_continent_re
Min. :18.00	Min. : 5.002	Min. : 4.0	Min. : 5.737
1st Qu.:50.00	1st Qu.: 6.579	1st Qu.: 82.0	1st Qu.:11.180
Median :59.00	Median : 8.049	Median :105.0	Median :11.557
Mean :59.67	Mean :10.265	Mean :111.5	Mean :11.480
3rd Qu.:70.00	3rd Qu.:12.860	3rd Qu.:130.0	3rd Qu.:12.007
Max. :95.00	Max. :44.706	Max. :560.0	Max. :15.365

The structure of our problem set-up is complicated. We have three outcome variables, several rating variables, and observe a cross-section of policyholders over time. Analysts encountering data with this structure typically do a far more complete analysis of the basic summary statistics than presented here. The purpose of this section is just to provide a taste of the type of analyses needed at this step. We assume that readers are familiar with these tasks and so proceed to more interesting steps.

Several characteristics are available to explain and predict lapse, as well as the number of claims and their cost. We considered only year, gender (male=1) and age of the customer in this study, but gender is not used in the motor analysis as a covariate due to the fact that gender cannot be used as a rating factor. We also consider client seniority, i.e. the number of years he has been a customer in the company and a binary indicator (metro_code) that indicates the place of residence of the customer (urban or metropolitan versus rural). Some additional covariates included in automobile accident frequency and severity model, namely the power of the car and the presence of a second driver, besides we also consider the payment method which is indicated by a yearly payment versus monthly, which is the baseline. For homeowners' insurance frequency and severity model, we consider the value of the property and the type of property, which is indicated by a binary variable that distinguishes apartment for other types of dwellings like houses or semi-attached houses.

Variable	Description
year	
gender	1 for male, 0 for female
Age client	age of the customer
Client Seniority	the number of years with the company
metro code	1 for urban or metropolitan, 0 for rural
Car power M	power of the car
Car 2ndDriver M	presence of a second driver
Policy PaymentMethodA	1 for annual payment, 0 for monthly payment
Insuredcapital continent re	value of the property
appartment	1 for apartment, 0 for houses or semi-attached houses
Policy PaymentMethodH	1 for annual payment, 0 for monthly payment

Tables can be used to summarize discrete variables.

```
# Discrete Variables x1, year and Retention
XVar = matrix(0,7,2)
colnames(XVar) <- c("0","1")
rownames(XVar) <-  c("Retention", "Gender", "Metro Code", "Car Second Driver",
                     "Apartment", "Policy Payment Method Auto",
                     "Policy Payment Method Home")
XVar[1,] <- 100*table(SampleData$Retention)/nrow(SampleData)
XVar[2,] <- 100*table(SampleData$gender)/nrow(SampleData)
XVar[3,] <- 100*table(SampleData$metro_code)/nrow(SampleData)
XVar[4,] <- 100*table(SampleData$Car_2ndDriver_M)/nrow(SampleData)
XVar[5,] <- 100*table(SampleData$appartment)/nrow(SampleData)
XVar[6,] <- 100*table(SampleData$Policy_PaymentMethodA)/nrow(SampleData)
XVar[7,] <- 100*table(SampleData$Policy_PaymentMethodH)/nrow(SampleData)
knitr::kable(XVar,digits=2,
             caption="Distributions of Binary Explanatory Variables, in Percent")
```

Table 3: Distributions of Binary Explanatory Variables, in Percent

	0	1
Retention	23.83	76.17
Gender	22.18	77.82
Metro Code	83.64	16.36
Car Second Driver	86.15	13.85
Apartment	37.06	62.94
Policy Payment Method Auto	22.93	77.07
Policy Payment Method Home	7.43	92.57

```
# Discrete Variables x1, year and Retention
XVar = matrix(0,6,6)
colnames(XVar) <- c("x=0 Lapse","x=1 Lapse", "x=0 PosClaim1","x=1 PosClaim1",
                    "x=0 PosClaim2","x=1 PosClaim2")
rownames(XVar) <-  c("Gender", "Metro Code", "Car Second Driver",
                     "Apartment", "Policy Payment Method Auto", "Policy Payment Method Home")
XVar[1,1:2] <- 100*aggregate(Lapse ~ gender, data=SampleData, mean)$Lapse
XVar[1,3:4] <- 100*aggregate(PosClaim1 ~ gender, data=SampleData, mean)$PosClaim1
XVar[1,5:6] <- 100*aggregate(PosClaim2 ~ gender, data=SampleData, mean)$PosClaim2
XVar[2,1:2] <- 100*aggregate(Lapse ~ metro_code, data=SampleData, mean)$Lapse
XVar[2,3:4] <- 100*aggregate(PosClaim1 ~ metro_code, data=SampleData, mean)$PosClaim1
XVar[2,5:6] <- 100*aggregate(PosClaim2 ~ metro_code, data=SampleData, mean)$PosClaim2
XVar[3,1:2] <- 100*aggregate(Lapse ~ Car_2ndDriver_M, data=SampleData, mean)$Lapse
XVar[3,3:4] <- 100*aggregate(PosClaim1 ~ Car_2ndDriver_M, data=SampleData, mean)$PosClaim1
XVar[3,5:6] <- 100*aggregate(PosClaim2 ~ Car_2ndDriver_M, data=SampleData, mean)$PosClaim2
XVar[4,1:2] <- 100*aggregate(Lapse ~ appartment, data=SampleData, mean)$Lapse
XVar[4,3:4] <- 100*aggregate(PosClaim1 ~ appartment, data=SampleData, mean)$PosClaim1
XVar[4,5:6] <- 100*aggregate(PosClaim2 ~ appartment, data=SampleData, mean)$PosClaim2
XVar[5,1:2] <- 100*aggregate(Lapse ~ Policy_PaymentMethodA, data=SampleData, mean)$Lapse
XVar[5,3:4] <- 100*aggregate(PosClaim1 ~ Policy_PaymentMethodA, data=SampleData, mean)$PosClaim1
XVar[5,5:6] <- 100*aggregate(PosClaim2 ~ Policy_PaymentMethodA, data=SampleData, mean)$PosClaim2
XVar[6,1:2] <- 100*aggregate(Lapse ~ Policy_PaymentMethodH, data=SampleData, mean)$Lapse
XVar[6,3:4] <- 100*aggregate(PosClaim1 ~ Policy_PaymentMethodH, data=SampleData, mean)$PosClaim1
XVar[6,5:6] <- 100*aggregate(PosClaim2 ~ Policy_PaymentMethodH, data=SampleData, mean)$PosClaim2
knitr::kable(XVar,digits=2,
             caption="Distributions of Outcomes by Binary Explanatory Variables, in Percent")
```

Table 4: Distributions of Outcomes by Binary Explanatory Variables, in Percent

	x=0 Lapse	x=1 Lapse	x=0 PosClaim1	x=1 PosClaim1	x=0 PosClaim2	x=1 PosClaim2
Gender	23.95	23.80	1.48	1.63	1.81	1.77
Metro Code	23.39	26.11	1.63	1.44	1.77	1.85
Car Second Driver	23.20	27.76	1.45	2.55	1.77	1.85
Apartment	23.26	24.17	1.79	1.49	0.47	2.55
Policy Payment Method Auto	28.61	22.41	2.06	1.46	2.23	1.65
Policy Payment Method Home	27.02	23.57	2.03	1.57	2.01	1.76

Outcomes by Year

You should examine the distribution of outcomes, auto and home claims, as well as lapse, *over time*. After all, the whole point is think about how the availability of observations, as dictated by lapse/renewal, impacts the claims distribution.

For this case study, by design the distribution of the claims frequency and severity is fairly stable over time. The largest type 1 (auto) claim is 109.72 and the largest type 2 (home) claim is 27.28, both in thousands.

```
SumClaims = matrix(0,9,5)
colnames(SumClaims) <-  c("F2010","F2011","F2012","F2013","F2014")
rownames(SumClaims) <-  c("Number of Obs","Number of Lapse","% of Lapses",
                          "Number of Clients with positive Claims 1",
                          "Average Number of Claim 1","Average cost of Claim 1",
                          "Clients with positive Claims 2",
                          "Average Number of Claim 2","Average cost of Claim 2")
SumClaims[1,] <- aggregate(Lapse ~ year, data=SampleData, length)$Lapse
SumClaims[2,] <- aggregate(Lapse ~ year, data=SampleData, sum)$Lapse
SumClaims[3,] <- SumClaims[2,]/SumClaims[1,]
SumClaims[4,] <- aggregate(PosClaim1 ~ year, data=SampleData, sum)$PosClaim1
SumClaims[5,] <- aggregate(NClaims1 ~ year, data=SampleData, mean)$NClaims1
SumClaims[6,] <- aggregate(Claims1 ~ year, data=SampleData, sum)$Claims1/SumClaims[4,]
SumClaims[7,] <- aggregate(PosClaim2 ~ year, data=SampleData, sum)$PosClaim2
SumClaims[8,] <- aggregate(NClaims2 ~ year, data=SampleData, mean)$NClaims2
SumClaims[9,] <- aggregate(Claims2 ~ year, data=SampleData, sum)$Claims2/SumClaims[7,]
knitr::kable(SumClaims,digits=2, caption="Lapse and Claims Summary by Year")
```

Table 5: Lapse and Claims Summary by Year

	F2010	F2011	F2012	F2013	F2014
Number of Obs	40284.00	29818.00	22505.00	17044.00	13284.00
Number of Lapse	10466.00	7313.00	5461.00	3760.00	2296.00
% of Lapses	0.26	0.25	0.24	0.22	0.17
Number of Clients with positive Claims 1	769.00	547.00	318.00	209.00	124.00
Average Number of Claim 1	0.04	0.03	0.03	0.02	0.02
Average cost of Claim 1	1539.99	1689.84	2031.20	1629.18	1222.13
Clients with positive Claims 2	660.00	531.00	448.00	310.00	240.00
Average Number of Claim 2	0.03	0.03	0.04	0.03	0.03
Average cost of Claim 2	447.85	501.59	410.73	348.10	508.86

Dependence Summary Statistics

Of particular interest is the relationship among outcome variables. First, we take a look at the number of observations where a policy has:

- Type 1: neither an auto nor a home claim
- Type 2: an auto but not a home claim
- Type 3: not an auto but a home claim
- Type 4: both an auto and a home claim

This frequency distribution is given for our simulated data in the table below, followed by the R code that generated the table.

```
Types = 1*(SampleData$Claims1==0)*(SampleData$Claims2==0)+
          2*(SampleData$Claims1>0) *(SampleData$Claims2==0)+
          3*(SampleData$Claims1==0)*(SampleData$Claims2>0)+
          4*(SampleData$Claims1>0) *(SampleData$Claims2>0)
table(Types)
```

```
Types
     1      2      3      4
118820   1926   2148     41
```

We can also summarize relationships among outcome variables using association measures such as *correlations*. However, our claims variables are a hybrid combination of zeros (for no claims) and long-tailed continuous variables (for positive claim amounts). Although this feature is captured by the Tweedie distribution, it can sometimes be difficult to establish dependence with basic summary statistics. Depending on parameter values, there can be many zeros (98.4% for this data set) and when positive, claims distributions tend to be right skewed. Here is some code that provides **Spearman** correlations, a nonparametric correlation coefficient.

As you experiment with different parameter values, you will find that the more zeros in the data, the more difficult it is to establish dependence with basic techniques. This is interesting because we know that, when generating the data, that important dependencies exist.

Table 6: Spearman Outcome Correlations

	Lapse	Claims1	Claims2
Lapse	1.000	0.051	0.018

	Lapse	Claims1	Claims2
Claims1	0.051	1.000	0.003
Claims2	0.018	0.003	1.000

When summarizing the data, it is sometimes convenient to work in terms of lapse, as this is the event that is of interest to insurers. However, going forward, we work with its complement, renewal (= one minus lapse). This is slightly more convenient mathematically in that we condition on a policy renewing to examine the claims distribution in subsequent periods.

5.3 Fit the Marginal Distributions

After careful work to summarize data (only a small portion shown here), the next step is fit marginal models. In this context, the descriptor *marginal* means analyzing each outcome without reference to the others. In subsequent sections, we join the marginal models via the copula.

Marginal model estimation is typically done assuming that each year has the same set of parameters and that observations from different years are independent. This is not necessary but provides a convenient starting point.

Logistic (Lapse) Regression Results

To model lapse, we employ a simple logistic regression (marginal) model.

```
logistic.fit <- glm(Lapse ~ year+gender+Age_client+
                    Client_Seniority+metro_code,data=SampleData,
                    control = glm.control(maxit = 50),family=binomial(link=logit))
sum.logistic.fit <- summary(logistic.fit)
knitr::kable(coefficients(sum.logistic.fit),digits=3,
             caption="Logistic Lapse Model Summary")
```

Table 7: Logistic Lapse Model Summary

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	0.324	0.034	9.393	0
year	-0.078	0.005	-15.191	0
gender	0.097	0.016	5.886	0
Age_client	-0.023	0.001	-41.184	0
Client_Seniority	-0.006	0.001	-4.503	0
metro_code	0.163	0.018	9.132	0

```
SampleData$dfLapse <- 1 - logistic.fit$fitted.values
                #*(SampleData$Lapse==0) - not correct but makes the code easier later...
```

Interpreting the Lapse Marginal Model

To model lapse, we have employed a simple logistic regression (marginal) model. The interpretation is straightforward. Since the time trend is negative it means that there is a general trend of decreasing lapses from 2010 to 2014, once the other characteristics such as age and gender of the policyholders are controlled. A negative and significant coefficient for Client seniority means that those that stay longer in the company also tend to lapse the policy less than customer that have been in the company less time. Gender (=1 for men) have higher probability to lapse than women. Coefficient for age-client indicates older clients tend to lapse less than younger and the same for length in the company. Urban customers tend to lapse more less than the rest.

Tweedie (Claims) Regression Results

The Tweedie is commonly used in insurance applications for claims. In part, this is because it can be expressed as a generalized linear model.

Marginal Auto Model

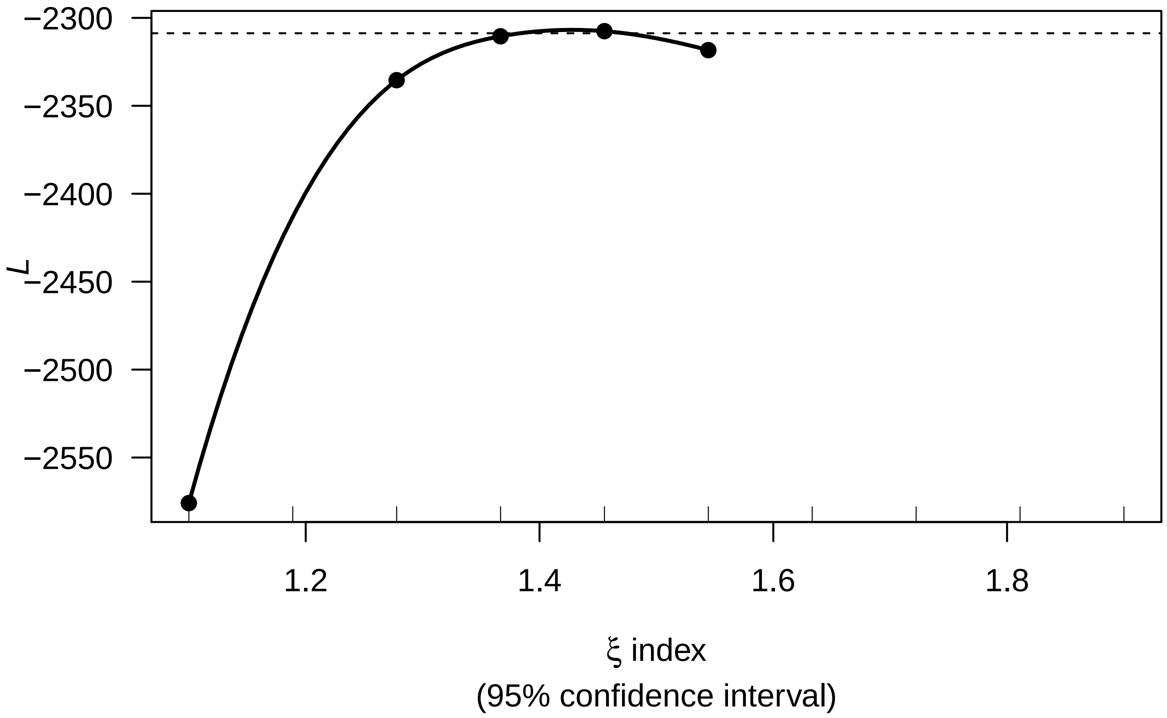
For type 1 (auto) claims model, we first need to find an initial parameter *p*. In order to make this procedure faster, we work with a random sample and then we proceed to estimate the optimal *p* with the whole.

– We select a random sample of size 10000 and estimate *p*:

```r
# Randomly re-order data - "shuffle it"
n <- nrow(SampleData)
set.seed(12347)
shuffled_SampleData <- SampleData[sample(n), ]
subset_SampleData <- shuffled_SampleData[1:10000,]

out1 <- tweedie.profile(Claims1 ~year+Age_client+
                    Client_Seniority+metro_code+Car_power_M+Car_2ndDriver_M+
                    Policy_PaymentMethodA,data=subset_SampleData,
                    xi.vec=seq(1.1, 1.9, length=10), do.plot=TRUE)
```

```
1.1 1.188889 1.277778 1.366667 1.455556 1.544444 1.633333 1.722222 1.811111 1.9
..........Done.
```



```
out1$xi.max
```

```
[1] 1.426531
```

– We estimate p with the whole sample and find the optimal p which has minimum deviance. We also find the other parameters by maximum likelihood:

```
funtweedie<-function(p){glm(Claims1 ~year+Age_client+
                    Client_Seniority+metro_code+Car_power_M+Car_2ndDriver_M+
                    Policy_PaymentMethodA,data=SampleData,
                    control = glm.control(maxit = 200),
                    family=tweedie(var.power=p, link.power=0))$deviance}
deviance_l_p_sample<-funtweedie(out1$xi.max-0.01)
deviance_p_sample<-funtweedie(out1$xi.max)
deviance_u_p_sample<-funtweedie(out1$xi.max+0.01)
p<-out1$xi.max-0.01
if(deviance_l_p_sample<deviance_p_sample)
  {while(deviance_l_p_sample<deviance_p_sample){
         deviance_p_sample<-deviance_l_p_sample
         deviance_l_p_sample<-funtweedie(p-0.01)
         p<-p-0.01}
} else{
  deviance_p_sample<-funtweedie(out1$xi.max)
  p<-out1$xi.max+0.01
   while(deviance_u_p_sample<=deviance_p_sample){
         deviance_p_sample<-deviance_u_p_sample
         deviance_u_p_sample<-funtweedie(p+0.01)
         p<-p+0.01}}
p
```

```
[1] 1.766531
```

```
tweedie.fit1 <- glm(Claims1 ~year+Age_client+Client_Seniority+
                    metro_code+Car_power_M+Car_2ndDriver_M+
                    Policy_PaymentMethodA,data=SampleData,
                    control = glm.control(maxit = 200),
                    family=tweedie(var.power=p, link.power=0))
sum.tweedie.fit1 <- summary(tweedie.fit1)
knitr::kable(coefficients(sum.tweedie.fit1),digits=3,
             caption="Tweedie Claims 1 (Auto) Model Summary")
```

Table 8: Tweedie Claims 1 (Auto) Model Summary

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	21.132	3.788	5.579	0.000
year	-1.179	0.441	-2.677	0.007
Age_client	-0.421	0.055	-7.586	0.000
Client_Seniority	0.170	0.272	0.626	0.532
metro_code	-4.356	1.932	-2.254	0.024
Car_power_M	0.122	0.005	23.538	0.000
Car_2ndDriver_M	-2.354	4.790	-0.491	0.623
Policy_PaymentMethodA	3.542	1.322	2.680	0.007

```
dfTweedie1A <- ptweedie(SampleData$Claims1,xi=p,
    mu=tweedie.fit1$fitted.values,phi=summary(tweedie.fit1)$dis)
SampleData$dfTweedie1  <- pmin(pmax( 1e-05,dfTweedie1A),.99999)
```

Interpreting the Frequency-Severity Marginal Model for Vehicle Insurance Claims

In the automobile claims marginal model results presented, we find a number of factors that influence significantly the expected claim size per year. We first note a decreasing trend in losses in car insurance. This was expected due to a severe economic crisis in Spain that covers the observed period of time, with much less claims than in previous periods. We find a negative effect of age, which is expected because younger drivers usually have more accidents and more severe accidents than older drivers. The same result holds for power of the car, which means probably safer vehicles, but this factor influence is not significant at the 0.05 level. Vehicles with more than one driver are identified by presence of an occasional second driver, which usually coincides with a younger driver in the household using the car and so, with the existence of more claims. The metro code indicates no differences by area, with lower severity in big cities but this is not significant. Payment method (annual) means less expected claims cost.

Marginal Homeowners Model

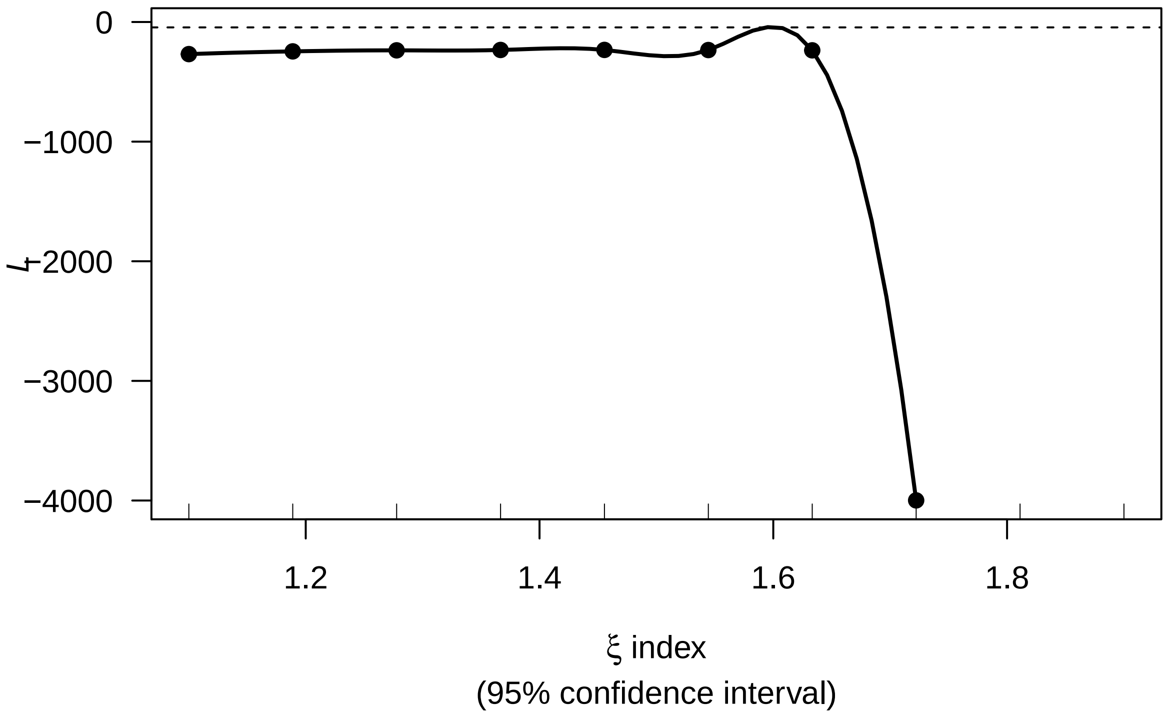
The type 2 (home) claims model estimation procedure is analogous to the case of motor claims.

– We select a random sample of size 1000 and estimate p:

```
# Randomly re-order data - "shuffle it"
n <- nrow(SampleData)
set.seed(12347)
shuffled_SampleData <- SampleData[sample(n), ]
subset_SampleData <- shuffled_SampleData[1:1000,]
```

```
out2 <- tweedie.profile(Claims2 ~year+Age_client+Client_Seniority+metro_code+
                      Insuredcapital_continent_re+appartment+
                      Policy_PaymentMethodH,data=subset_SampleData,
                      xi.vec=seq(1.1, 1.9, length=10), do.plot=TRUE)
```

```
1.1 1.188889 1.277778 1.366667 1.455556 1.544444 1.633333 1.722222 1.811111 1.9
..........Done.
```



```
out2$xi.max
```

```
[1] 1.595238
```

– As in the case of motor insurance, we find the optimal p with the whole sample and then estimate the other parameters.

```
funtweedie<-function(p){glm(Claims2 ~year+Age_client+
                              Client_Seniority+metro_code+
                      Insuredcapital_continent_re+appartment+
                        Policy_PaymentMethodH,data=SampleData,
                    control = glm.control(maxit = 200),
                    family=tweedie(var.power=p, link.power=0))$deviance}
deviance_l_p_sample<-funtweedie(out2$xi.max-0.01)
deviance_p_sample<-funtweedie(out2$xi.max)
deviance_u_p_sample<-funtweedie(out2$xi.max+0.01)
p<-out2$xi.max-0.01
if(deviance_l_p_sample<deviance_p_sample)
  {while(deviance_l_p_sample<deviance_p_sample){
         deviance_p_sample<-deviance_l_p_sample
         deviance_l_p_sample<-funtweedie(p-0.01)
         p<-p-0.01}
} else{
  deviance_p_sample<-funtweedie(out2$xi.max)
  p<-out2$xi.max+0.01
   while(deviance_u_p_sample<=deviance_p_sample){
         deviance_p_sample<-deviance_u_p_sample
         deviance_u_p_sample<-funtweedie(p+0.01)
         p<-p+0.01}}
p
```

```
[1] 1.705238
```

```
tweedie.fit2 <- glm(Claims2 ~year+Age_client+Client_Seniority+metro_code+
                      Insuredcapital_continent_re+appartment+
                      Policy_PaymentMethodH,data=SampleData,
                    control = glm.control(maxit = 200),
                    family=tweedie(var.power=p, link.power=0))
sum.tweedie.fit2 <- summary(tweedie.fit2)
knitr::kable(coefficients(sum.tweedie.fit2),digits=3,
             caption="Tweedie Claims 2 (Home) Model Summary")
```

Table 9: Tweedie Claims 2 (Home) Model Summary

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-2.794	1.084	-2.577	0.010
year	-0.015	0.047	-0.316	0.752
Age_client	0.013	0.005	2.600	0.009
Client_Seniority	-0.004	0.013	-0.329	0.742
metro_code	0.261	0.167	1.565	0.118
Insuredcapital_continent_re	0.348	0.082	4.250	0.000
appartment	1.097	0.153	7.164	0.000
Policy_PaymentMethodH	-0.765	0.225	-3.397	0.001

```
dfTweedie2A <- ptweedie(SampleData$Claims2,xi=p,
            mu=tweedie.fit2$fitted.values,phi=summary(tweedie.fit2)$dis)
SampleData$dfTweedie2  <- pmin(pmax( 1e-05,dfTweedie2A),.99999)
```

Interpreting the Frequency-Severity Marginal Model for Home Insurance Claims

The results of the marginal model for the homeowners insurance claims model also show a decreasing year trend but non-significant. Only the effect of Insuredcapital_continent re is positive meaning that the cost of claims is larger for more expensive homes. Apartments compared to other types of homes have also a tendency to have larger claims. In this model compared to motor insurance there are several additional comments. The age of the client has a positive effect, which means that the older the client the larger the expected size of future claims. Payment method (annual) means also implies less expected claims cost.

Residual Checking

As with all model estimation procedures, it is good standard practice to check model assumptions via an examination of the residuals. For generalized linear models, one typically examines **deviance residuals**. The following provides an example of a standard set of diagnostic plots based on the deviance residuals.

Click Here to Show Residual Plots

```
par(mfrow=c(2, 2))
#plot(tweedie.fit1) #not plotted to reduce filesize
```

```
par(mfrow=c(2, 2))
#plot(tweedie.fit2) #not plotted  to reduce filesize
```

Table 10: Spearman Correlations

	Lapse	Claims 1 Resids	Claims 2 Resids
Lapse	1.000	0.395	-0.178
Claims 1 Resids	0.395	1.000	-0.105
Claims 2 Resids	-0.178	-0.105	1.000

Standard residual plots from the Tweedie model can be difficult to assess due to mass at zero. Another type of residual can be calculated via the **probability integral transform**. That is, if Y is a random variable with distribution function F, then $F(Y)$ has a uniform distribution. We can use this relationship to assess quality of our identification of the distribution. Like the deviance residuals, this relationship breaks down in the presence of mass points, e.g. zeros, but can still be used to supplement the usual diagnostic modeling checking. We refer to these as **Cox Snell** residuals.

5.4 Joint Model Specification

R Code for Data Preparation

This code separates the data into different subsets needed for likelihood calculations. The subsets include lapse and no lapse, as well as four different claim outcomes: (i) no auto/home claim, (ii) an but no home claim, (iii) no auto but a home claim, and (iv) an auto and a home claim.

```
# Reshape the data
TweedieLike1 <- SampleData[order(-SampleData$PolID, SampleData$year),]
VarsLike    <- c("PolID", "year", "Lapse", "dfLapse",
                   "Claims1","dfTweedie1","Claims2","dfTweedie2")
TweedieLike <- TweedieLike1[VarsLike]
calcOrder   <- 1:nrow(TweedieLike)
# These are the four cases from the hybrid joint mass/density function
caset1t2 <- 1*(TweedieLike$Claims1==0)*(TweedieLike$Claims2==0)+
              2*(TweedieLike$Claims1>0) *(TweedieLike$Claims2==0)+
              3*(TweedieLike$Claims1==0)*(TweedieLike$Claims2>0)+
              4*(TweedieLike$Claims1>0) *(TweedieLike$Claims2>0)
u  <- as.matrix(cbind(TweedieLike$dfLapse,TweedieLike$dfTweedie1,TweedieLike$dfTweedie2))
zu <- qnorm(u)
mydata <- data.frame(caset1t2,u,zu,calcOrder,TweedieLike$Lapse)
names(mydata) <- c("caset1t2","u1","u2","u3","zu1","zu2","zu3","calcOrder","Lapse")
mydataLapse   <- mydata[which(mydata$Lapse==1),]
mydataNoLapse <- mydata[which(mydata$Lapse==0),]
mydata1 <- mydata[which(caset1t2==1),]
mydata2 <- mydata[which(caset1t2==2),]
mydata3 <- mydata[which(caset1t2==3),]
mydata4 <- mydata[which(caset1t2==4),]
```

Pairwise Likelihood Estimation

R Code for Pairwise Likelihood Calculation

In the following pairwise likelihood calculations, we drop the marginal densities $f_2(y)$ as they contain no information about the dependence parameters. See the documentation for the R package VineCopula for explanations of the functions to evaluate bivariate copulas and their derivatives.

```
NegBivariateLikelihood <- function(rho, type) {
    bilikehd <- 0 * calcOrder
    if (nrow(mydataNoLapse) > 0) {
        dat <- mydataNoLapse
        u <- (type == 1) * dat$u2 + (type == 2) * dat$u3
        ClaimInd <- 1 * (dat$caset1t2 == 4) + 1 * (dat$caset1t2 == 2) * (type ==
            1) + 1 * (dat$caset1t2 == 3) * (type == 2)
        ZeroLike <- as.matrix(BiCopCDF(dat$u1, u, family = 1, par = rho), ncol = 1)
        PosLike <- as.matrix(BiCopHfunc2(dat$u1, u, family = 1, par = rho),
            ncol = 1)
        bilikehd[dat$calcOrder] <- (ClaimInd == 0) * ZeroLike + (ClaimInd >
            0) * PosLike
    }
    if (nrow(mydataLapse) > 0) {
        dat <- mydataLapse
        u <- (type == 1) * dat$u2 + (type == 2) * dat$u3
        ClaimInd <- 1 * (dat$caset1t2 == 4) + 1 * (dat$caset1t2 == 2) * (type ==
            1) + 1 * (dat$caset1t2 == 3) * (type == 2)
        ZeroLike <- u - as.matrix(BiCopCDF(dat$u1, u, family = 1, par = rho),
            ncol = 1)
        PosLike <- 1 - as.matrix(BiCopHfunc2(dat$u1, u, family = 1, par = rho),
            ncol = 1)
        bilikehd[dat$calcOrder] <- (ClaimInd == 0) * ZeroLike + (ClaimInd >
            0) * PosLike
    }
    bilikehd[is.na(bilikehd)] <- 0
    bilikehd <- pmin(pmax(1e-12, bilikehd), 1e+12)
    return(-sum(log(bilikehd)))
}

BiLikeLA <- function(rho) {
    return(NegBivariateLikelihood(rho, 1))
}
BiLikeLH <- function(rho) {
    return(NegBivariateLikelihood(rho, 2))
}
```

```
BiLikeAH <- function(rho) {
  # See the VineCopula package
  #  for the functions 'BiCopCDF', 'BiCopHfunc', and 'BiCopPDF'
     likehd <- 0*calcOrder
  if (nrow(mydata1)>0) {likehd[mydata1$calcOrder] <-
    BiCopCDF(mydata1$u2,mydata1$u3, family=1, par=rho)
  }
  if (nrow(mydata2)>0) {likehd[mydata2$calcOrder] <-
    BiCopHfunc1(mydata2$u2,mydata2$u3, family=1, par=rho)
  }
  if (nrow(mydata3)>0) {likehd[mydata3$calcOrder] <-
    BiCopHfunc2(mydata3$u2,mydata3$u3, family=1, par=rho)
  }
  if (nrow(mydata4)>0) {likehd[mydata4$calcOrder] <-
    BiCopPDF(mydata4$u2,mydata4$u3, family=1, par=rho)
  }
    likehd <-   pmin(pmax(1e-12,likehd),1e12)
    return(-sum(log(likehd)))
}
```

Pairwise Likelihood Estimation Results

```
toler     <- 0.4
opLA <- optim(0,BiLikeLA,method=c("L-BFGS-B"),
              lower=-toler,upper=toler,hessian=TRUE)
PairSELA <- sqrt(diag(ginv(opLA$hessian)))
opLH <- optim(0,BiLikeLH,method=c("L-BFGS-B"),
              lower=-toler,upper=toler,hessian=TRUE)
PairSELH <- sqrt(diag(ginv(opLH$hessian)))
opAH <- optim(0,BiLikeAH,method=c("L-BFGS-B"),
              lower=-toler,upper=toler,hessian=TRUE)
PairSEAH <- sqrt(diag(ginv(opAH$hessian)))

PairEst<-rbind(opLA$par,opLH$par,opAH$par)
PairSE<-rbind(PairSELA,PairSELH,PairSEAH)

EstResults <- cbind(PairEst,PairSE)
rownames(EstResults) <- c("Lapse-Auto","Lapse-Home","Auto-Home")
colnames(EstResults) <- c("Estimate","Std Error")
knitr::kable(EstResults,digits=4,
             caption="Pairwise Likelihood Estimation Results")
```

Table 11: Pairwise Likelihood Estimation Results

	Estimate	Std Error
Lapse-Auto	0.0995	0.0068
Lapse-Home	0.0683	0.0109
Auto-Home	0.0985	0.0162

Generalized Method of Moments Estimation

R Code for GMM Functions

Top level functions are here.

```
GMMgthetaFunct<- function(param) {
  Scores <- data.frame(GMMScore(param)[,c(2:4)])
  names(Scores) <- c("Score12", "Score13", "Score23")
  return(as.matrix(Scores))
}

GMMFunc<- function(param) {
  GMMgtheta <- GMMgthetaFunct(param)
  GMMScorex <- t(colSums(GMMgtheta)) %*%
                 ginv(Vargtheta) %*% colSums(GMMgtheta) / length(GMMgtheta[,1])
  return(GMMScorex)
}
```

R Code for GMM Scores

The likelihood and scores corresponding to all outcomes are calculated here. It uses as input the following subsection that provides calculations only for the trivariate piece. The function returns the likelihood and three scores (corresponding to the three association parameters).

```
GMMScore <- function(param) {
    rhoLA <- param[1]
    rhoLH <- param[2]
    rhoAH_L <- param[3]
    # Transformed parameter
    rhoAH <- rhoLA * rhoLH + rhoAH_L * sqrt((1 - rhoLA^2) * (1 - rhoLH^2))
    rhoLA <- pmin(pmax(-0.99, rhoLA), 0.99)
    rhoLH <- pmin(pmax(-0.99, rhoLH), 0.99)
    rhoAH <- pmin(pmax(-0.99, rhoAH), 0.99)
    SigmaList <- SigmaFct(rhoLA, rhoLH, rhoAH)
    Sigma <- SigmaList[[1]]
    Sigma12 <- SigmaList[[2]]
    Sigma13 <- SigmaList[[3]]
    Sigma23 <- SigmaList[[4]]
    Sigma13.2 <- SigmaList[[5]]
    Sigma12.3 <- SigmaList[[6]]
    Sigma23.1 <- SigmaList[[7]]
    Sigma3.12 <- SigmaList[[8]]
    Sigma1.23 <- SigmaList[[9]]
    Sigma2.13 <- SigmaList[[10]]

    vec111 <- as.vector(cbind(1, 1, 1))
    vec001 <- as.vector(cbind(0, 0, 1))
    ScoreLike <- matrix(0, length(mydata[, 1]), 4)
    # (Auto=0,Home=0) Case
    if (nrow(mydata1) > 0) {
        dat <- mydata1
        Reten0 <- 1 - dat$Lapse
        score <- GMMScore00(rhoLA, rhoLH, rhoAH, dat$u1, dat$u2, dat$u3)
        fbb <- score[, 2:4]
        Fbb <- as.matrix(score[, 1], ncol = 1) %*% vec111
        Fbb[, 1] <- pmax(1e-05, Fbb[, 1])
        Fbb[, 2] <- pmax(1e-05, Fbb[, 2])
        Fbb[, 3] <- pmax(1e-05, Fbb[, 3])
        faa <- as.matrix(dmvnorm(cbind(dat$zu2, dat$zu3), mean = rep(0, 2),
            sigma = Sigma23, log = FALSE), ncol = 1) %*% vec001
        Faa <- as.matrix(BiCopCDF(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec111
        Faa_Fbb <- pmin(pmax(1e-05, (Faa - Fbb)[, 1]), 0.99999)
        ScoreLike[dat$calcOrder, 1] <- Reten0 * log(Fbb)[, 1] + (1 - Reten0) *
            log(Faa_Fbb)
        ScoreLike[dat$calcOrder, 2:4] <- Reten0 * fbb/Fbb + (1 - Reten0) * (faa -
            fbb)/Faa_Fbb
    }
    # (Auto=1,Home=0) Case
    if (nrow(mydata2) > 0) {
        dat <- mydata2
        Reten0 <- 1 - dat$Lapse
        score <- GMMScore10(rhoLA, rhoLH, rhoAH, dat$u1, dat$u2, dat$u3)
        fbb <- score[, 2:4]
        Fbb <- as.matrix(score[, 1], ncol = 1) %*% vec111
        Fbb[, 1] <- pmax(1e-05, Fbb[, 1])
        Fbb[, 2] <- pmax(1e-05, Fbb[, 2])
        Fbb[, 3] <- pmax(1e-05, Fbb[, 3])
        faa <- as.matrix(BiCopHfuncDeriv(dat$u3, dat$u2, family = 1, par = rhoAH),
            ncol = 1) %*% vec001
        Faa <- as.matrix(BiCopHfunc1(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec111
        Faa_Fbb <- pmin(pmax(1e-05, (Faa - Fbb)[, 1]), 0.99999)
        ScoreLike[dat$calcOrder, 1] <- Reten0 * log(Fbb)[, 1] + (1 - Reten0) *
            log(Faa_Fbb)
        ScoreLike[dat$calcOrder, 2:4] <- Reten0 * fbb/Fbb + (1 - Reten0) * (faa -
            fbb)/Faa_Fbb
    }
    # (Auto=0,Home=1) Case
    if (nrow(mydata3) > 0) {
        dat <- mydata3
        Reten0 <- 1 - dat$Lapse
        score <- GMMScore01(rhoLA, rhoLH, rhoAH, dat$u1, dat$u2, dat$u3)
        fbb <- score[, 2:4]
        Fbb <- as.matrix(score[, 1], ncol = 1) %*% vec111
        Fbb[, 1] <- pmax(1e-05, Fbb[, 1])
        Fbb[, 2] <- pmax(1e-05, Fbb[, 2])
        Fbb[, 3] <- pmax(1e-05, Fbb[, 3])
        faa <- as.matrix(BiCopHfuncDeriv(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec001
        Faa <- as.matrix(BiCopHfunc2(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec111
        Faa_Fbb <- pmin(pmax(1e-05, (Faa - Fbb)[, 1]), 0.99999)
        ScoreLike[dat$calcOrder, 1] <- Reten0 * log(Fbb)[, 1] + (1 - Reten0) *
            log(Faa_Fbb)
        ScoreLike[dat$calcOrder, 2:4] <- Reten0 * fbb/Fbb + (1 - Reten0) * (faa -
            fbb)/Faa_Fbb
    }
    # (Auto=1,Home=1) Case
    if (nrow(mydata4) > 0) {
        dat <- mydata4
        Reten0 <- 1 - dat$Lapse
        score <- GMMScore11(rhoLA, rhoLH, rhoAH, dat$u1, dat$u2, dat$u3)
        fbb <- score[, 2:4]
        Fbb <- as.matrix(score[, 1], ncol = 1) %*% vec111
        Fbb[, 1] <- pmax(1e-05, Fbb[, 1])
        Fbb[, 2] <- pmax(1e-05, Fbb[, 2])
        Fbb[, 3] <- pmax(1e-05, Fbb[, 3])
        faa <- as.matrix(BiCopDeriv(pnorm(dat$zu2), pnorm(dat$zu3), family = 1,
            par = rhoAH), ncol = 1) %*% vec001
        Faa <- as.matrix(BiCopPDF(dat$u2, dat$u3, family = 1, par = rhoAH),
            ncol = 1) %*% vec111
        Faa_Fbb <- pmin(pmax(1e-05, (Faa - Fbb)[, 1]), 0.99999)
        ScoreLike[dat$calcOrder, 1] <- Reten0 * log(Fbb)[, 1] + (1 - Reten0) *
            log(Faa_Fbb)
        ScoreLike[dat$calcOrder, 2:4] <- Reten0 * fbb/Fbb + (1 - Reten0) * (faa -
            fbb)/Faa_Fbb
    }
    return(ScoreLike)
}
```

R Code for Basic GMM Score Functions

The likelihood and scores corresponding to trivariate outcomes are calculated here. This has four functions corresponding to our four data cases: (i) (Auto=0,Home=0), (ii) (Auto=1,Home=0), (iii) (Auto=0,Home=1), and (iv) (Auto=1,Home=1). Each function returns the likelihood and three scores (corresponding to the three association parameters).

```
# (Auto=0,Home=0) Case
GMMScore00 <- function(rho12,rho13,rho23,u1,u2,u3){
  norm.cops  <- normalCopula(param=c(rho12,rho13,rho23), dispstr="un", dim=3)
  like <- pCopula(cbind(u1,u2,u3), copula=norm.cops, algorithm=TVPACK(abseps=1e-8))
  like <- pmin(pmax(1e-05,like),.99999)

  zu1 <- as.vector(qnorm(u1));zu2 <- as.vector(qnorm(u2));zu3 <- as.vector(qnorm(u3))
  SigmaList <- SigmaFct(rho12,rho13,rho23);
  Sigma <- SigmaList[[1]]
  Sigma12   <- SigmaList[[2]];
  Sigma13 <- SigmaList[[3]];
  Sigma23 <- SigmaList[[4]];
  Sigma3.12 <- SigmaList[[8]];
  Sigma1.23 <- SigmaList[[9]];
  Sigma2.13 <- SigmaList[[10]]

  zstar12 <- zu3 - as.matrix(cbind(zu1,zu2))%*%ginv(Sigma12)%*%Sigma[c(1,2),3]
  score12 <- dmvnorm(cbind(zu1,zu2),mean=rep(0,2),sigma=Sigma12,log=FALSE) *
                  pnorm(zstar12, sd=sqrt(Sigma3.12))
  zstar13 <- zu2 - as.matrix(cbind(zu1,zu3))%*%ginv(Sigma13)%*%Sigma[c(1,3),2]
  score13 <- dmvnorm(cbind(zu1,zu3),mean=rep(0,2),sigma=Sigma13,log=FALSE) *
                  pnorm(zstar13,sd=sqrt(Sigma2.13))
  zstar23 <- zu1 - as.matrix(cbind(zu2,zu3))%*%ginv(Sigma23)%*%Sigma[c(2,3),1]
  score23 <- dmvnorm(cbind(zu2,zu3),mean=rep(0,2),sigma=Sigma23,log=FALSE) *
                  pnorm(zstar23,sd=sqrt(Sigma1.23))
  return(cbind(like,score12,score13,score23))
}

# (Auto=1,Home=0) Case
GMMScore10 <- function(rho12,rho13,rho23,u1,u2,u3){
  zu1 <- as.vector(qnorm(u1));zu2 <- as.vector(qnorm(u2));zu3 <- as.vector(qnorm(u3))
  sig1 <- sqrt(1-rho12^2)
  sig2 <- sqrt(1-rho23^2)
  rhox <- (rho13-rho12*rho23)/(sig1*sig2)
  rhox <- pmin(pmax(-.999,rhox),.999)
  z1s  <- (zu1 -zu2*rho12)/sig1
  z3s  <- (zu3 -zu2*rho23)/sig2
  like <- BiCopCDF(pnorm(z1s),pnorm(z3s), family=1, par=rhox)
  like <- pmin(pmax(1e-05,like),.99999)

  rhoxMat <- matrix(c(1,rhox,rhox,1),nrow=2,ncol=2)
  scoreAiii <- dmvnorm(cbind(z1s,z3s),mean=rep(0,2),sigma=rhoxMat,log=FALSE)
  score13 <- 0  + 0 + scoreAiii/(sig1*sig2)
  score12 <- bideriv1(z1s,z3s,rhox)*(zu1*rho12-zu2)/sig1^3  + 0 +
                   scoreAiii*(rho12*rho13-rho23)/(sig1^3*sig2)
  score23 <- 0  + bideriv1(z3s,z1s,rhox)*(zu3*rho23-zu2)/sig2^3 +
                  scoreAiii*(rho13*rho23-rho12)/(sig1*sig2^3)
  return(cbind(like,score12,score13,score23))
}

# Helpful Function
bideriv1 <- function(x1,x2,rhox)
  {return(dnorm(x1)*pnorm((x2-rhox*x1)/sqrt(1-rhox^2))) }

# (Auto=0,Home=1) Case
GMMScore01 <- function(rho12,rho13,rho23,u1,u2,u3){
  zu1 <- as.vector(qnorm(u1));
  zu2 <- as.vector(qnorm(u2));
  zu3 <- as.vector(qnorm(u3))
  sig1 <- sqrt(1-rho13^2)
  sig2 <- sqrt(1-rho23^2)
  rhox <- (rho12-rho13*rho23)/(sig1*sig2)
  rhox <- pmin(pmax(-.999,rhox),.999)
  z1s  <- (zu1 -zu3*rho13)/sig1
  z2s  <- (zu2 -zu3*rho23)/sig2
  like <- BiCopCDF(pnorm(z1s),pnorm(z2s),family=1,par=rhox)
  like <- pmin(pmax(1e-05,like),.99999)

  rhoxMat   <-  matrix(c(1,rhox,rhox,1),nrow=2,ncol=2)
  scoreAiii <- dmvnorm(cbind(z1s,z2s), mean=rep(0, 2), sigma=rhoxMat, log=FALSE)
  score12   <- 0  + 0 + scoreAiii/(sig1*sig2)
  score13   <- bideriv1(z1s,z2s,rhox)*(zu1*rho13-zu3)/sig1^3  + 0 +
                 scoreAiii*(rho12*rho13-rho23)/(sig1^3*sig2)
  score23   <- 0  + bideriv1(z2s,z1s,rhox)*(zu2*rho23-zu3)/sig2^3 +
                 scoreAiii*(rho12*rho23-rho13)/(sig1*sig2^3)
  return(cbind(like,score12,score13,score23))
}
# (Auto=1,Home=1) Case
GMMScore11 <- function(rho12,rho13,rho23,u1,u2,u3){
  zu1 <- as.vector(qnorm(u1));
  zu2 <- as.vector(qnorm(u2));
  zu3 <- as.vector(qnorm(u3))
  SigmaList <- SigmaFct(rho12,rho13,rho23);
  Sigma <- SigmaList[[1]]
  Sigma12   <- SigmaList[[2]];
  Sigma13 <- SigmaList[[3]];
  Sigma23 <- SigmaList[[4]];
  Sigma13.2 <- SigmaList[[5]];
  Sigma12.3 <- SigmaList[[6]];
  Sigma23.1 <- SigmaList[[7]];
  Sigma3.12 <- SigmaList[[8]];
  Sigma1.23 <- as.numeric(SigmaList[[9]]);
  Sigma2.13 <- SigmaList[[10]]
  tempSig   <- ginv(Sigma[2:3,2:3])
  mu1.23    <- cbind(zu2,zu3) %*% tempSig %*% Sigma[1,2:3]
  z1s  <- (zu1 - as.vector(mu1.23))/as.numeric(sqrt(Sigma1.23))
  like <- pnorm(z1s)*BiCopPDF(u2,u3, family=1, par=rho23)
  like <- pmax(1e-05,like)

  partialCop1 <- -dnorm(z1s)/Sigma1.23^(3/2)
  mu1.231 <- as.vector(cbind(zu2,zu3) %*% tempSig %*% as.vector(c(1,0)))
  mu1.232 <- as.vector(cbind(zu2,zu3) %*% tempSig %*% as.vector(c(0,1)))
  mu1.233 <- -as.vector(cbind(zu2,zu3) %*% tempSig %*%
                  matrix(c(0,1,1,0),nrow=2,ncol=2) %*% tempSig %*% Sigma[1,2:3])

  sig1.231 <- as.numeric(-2*Sigma[2:3,1] %*% tempSig %*% as.vector(c(1,0)))
  sig1.232 <- as.numeric(-2*Sigma[2:3,1] %*% tempSig %*% as.vector(c(0,1)))
  sig1.233 <- as.numeric(Sigma[2:3,1] %*% tempSig %*%
                  matrix(c(0,1,1,0),nrow=2,ncol=2) %*% tempSig %*% Sigma[1,2:3])

  partialCop12 <- partialCop1*(Sigma1.23*mu1.231+0.5*(zu1-mu1.23)*sig1.231)
  partialCop13 <- partialCop1*(Sigma1.23*mu1.232+0.5*(zu1-mu1.23)*sig1.232)
  partialCop23 <- partialCop1*(Sigma1.23*mu1.233+0.5*(zu1-mu1.23)*sig1.233)

  score12 <- 0  + partialCop12*BiCopPDF(u2,u3, family=1, par=rho23)
  score13 <- 0  + partialCop13*BiCopPDF(u2,u3, family=1, par=rho23)
  score23 <- pnorm(z1s)*BiCopDeriv(u2,u3,family=1,par=rho23,deriv="par",log=FALSE) +
                    partialCop23*BiCopPDF(u2,u3, family=1, par=rho23)
  return(cbind(like,score12,score13,score23))
}
```

R Code for Matrices of Association Parameters

```
SigmaFct <- function(rho12,rho13,rho23){
  Sigma <- matrix(c(1,rho12,rho13,  rho12,1,rho23, rho13,rho23,1),
                  nrow=3,ncol=3)
  Sigma12   <- Sigma[c(1,2),c(1,2)]
  Sigma13   <- Sigma[c(1,3),c(1,3)]
  Sigma23   <- Sigma[c(2,3),c(2,3)]
  Sigma13.2 <- Sigma[c(1,3),c(1,3)]-Sigma[c(1,3),2] %*% t(Sigma[c(1,3),2])
  Sigma12.3 <- Sigma[c(1,2),c(1,2)]-Sigma[c(1,2),3] %*% t(Sigma[c(1,2),3])
  Sigma23.1 <- Sigma[c(2,3),c(2,3)]-Sigma[c(2,3),1] %*% t(Sigma[c(2,3),1])
  Sigma3.12 <- 1-Sigma[3,1:2] %*% ginv(Sigma[1:2,1:2]) %*% Sigma[3,1:2]
  Sigma1.23 <- 1-Sigma[1,2:3] %*% ginv(Sigma[2:3,2:3]) %*% Sigma[1,2:3]
  Sigma2.13 <- 1-Sigma[2,c(1,3)] %*% ginv(Sigma[c(1,3),c(1,3)]) %*% Sigma[2,c(1,3)]
  Sigma3.12 <- Sigma3.12*(Sigma3.12>0)
  Sigma1.23 <- Sigma1.23*(Sigma1.23>0)
  Sigma2.13 <- Sigma2.13*(Sigma2.13>0)
  list(Sigma,Sigma12,Sigma13,Sigma23,Sigma13.2,
       Sigma12.3,Sigma23.1,Sigma3.12,Sigma1.23,Sigma2.13)
}
```

GMM Estimation Results

```r
# GMMInit <- c(0,0,0)
GMMInit  <- c(opLA$par,opLH$par,opAH$par)
toler = .1
lowerbd <- GMMInit - toler
upperbd <- GMMInit + toler
GMMgthetaInit <- GMMgthetaFunct(GMMInit)
GMMInitdev <- GMMgthetaInit -
              matrix(1, nrow=length(GMMgthetaInit[,1]),ncol=1) %*%
              colMeans(GMMgthetaInit)
Vargtheta  <- t(GMMInitdev) %*% GMMInitdev / length(GMMgthetaInit[,1])
GMMResult2 <- optim(par=GMMInit,GMMFunc,method=c("L-BFGS-B"),
                    lower=lowerbd,upper=upperbd,control=list(factr=10^12))
GMMEst     <- GMMResult2$par
# Standard Error
# Adjustments for Reparameterization
GTransform <- matrix(c(1,0,0,0,1,0,
               GMMEst[2]-GMMEst[1]*GMMEst[3]*sqrt( (1-GMMEst[2]^2)/(1-GMMEst[1]^2) )   ,
               GMMEst[1]-GMMEst[2]*GMMEst[3]*sqrt( (1-GMMEst[1]^2)/(1-GMMEst[2]^2) )   ,
               sqrt( (1-GMMEst[1]^2)*(1-GMMEst[2]^2) ) ) ,
               nrow=3,ncol=3)
GMMgthetaSumFunct<- function(param) { colSums(GMMgthetaFunct(param))  }
gradient   <- jacobian(func=GMMgthetaSumFunct,GMMEst,
                       method="simple",method.args=list(eps=5e-3))
GMMgtheta  <- GMMgthetaFunct(GMMEst)
Vargtheta  <- t(GMMgtheta) %*% GMMgtheta / length(GMMgtheta[,1])
GMMVar     <- t(gradient) %*% ginv(Vargtheta) %*% gradient / length(GMMgtheta[,1])
TransformGMMVar <- t(GTransform) %*% ginv(GMMVar) %*% GTransform
tryCatch(GMMstderror <- sqrt(diag(TransformGMMVar)) ,
             error=function(e) {GMMstderror <- 0*TransformGMMVar})
GMMEst[3] <- GMMEst[1]*GMMEst[2]  + GMMEst[3]*sqrt( (1-GMMEst[1]^2)*(1-GMMEst[2]^2) )
EstResults1 <- cbind(PairEst,PairSE,GMMEst,GMMstderror)

rownames(EstResults1) <- c("Lapse-Auto","Lapse-Home","Auto-Home")
colnames(EstResults1) <- c("Like Est","Like Std Error", "GMM Est", "GMM Std Error")
knitr::kable(EstResults1,digits=6, caption="GMM Estimation Results")
```

Table 12: GMM Estimation Results

	Like Est	Like Std Error	GMM Est	GMM Std Error
Lapse-Auto	0.099538	0.006810	0.101060	0.007143
Lapse-Home	0.068258	0.010855	0.068640	0.011222
Auto-Home	0.098482	0.016213	0.117902	0.028527

5.5 Interpretations and Implications

Correlations

The new important result is that the correlations between the predictions is significant for all three pairs. All pair-wise correlations significantly are different from zero with the GMM method and with maximum likelihood. The ML estimation and the GMM provide very similar results for the parameter and for their corresponding standard errors. These correlations must be interpreted as a dependence structure under the joint model estimation. The correlation between lapse and auto claims is positive and large. The correlation between lapse and home is also positive and about half its size as the correlation between lapse and automobile insurance. There is also a positive and significant correlation between the size of claims in motor and in home insurance, meaning that home and auto are indeed positively associated dependent risks.

What does this mean? It means that even if we consider the characteristics of the client, and the vehicles and homes that are being covered, there is still some relationship between the claimed losses for vehicle insurance, the claimed losses for home insurance and the expected renewal behavior. If a customer claims motor insurance losses this implies a higher tendency to lapse. The reason we argue is that having a claim induces a decision to change to another company. A higher expected claims' size for homeowner insurance implies more tendency to lapse too.

The main takeaway from this market is that the existence of claims in motor-insurance is an alert for non-renewal, and to a lesser extend the same happens with homeowners insurance. When an insured has large claims, the odds of abandoning the company increase. In addition, we have found evidence of a positive correlation between the size of claims in motor and homeowners insurances. This latter fact has implications reserving practice because it shows that the two risks are not independent.

Implications for insurance

A simple calculation to find the value of an insurance customer is based on estimating the cost if the customer renews the two policies. Therefore, this is equivalent to calculating the expectation of the product of two random variables: claim size times lapse. If only the marginal model for claim size is used then it is implicitly assumed that the policy is already renewed and as such this would be a biased procedure before the individual decision is taken. In practice, companies just assume that the policy will be renewed and that the claim size is predicted with the marginal model. Instead, we consider the lapsing as part of the evaluation mechanism. Following the notation in Section 1, our estimate (without taking into account that there are general expenses such as managerial and advertising, plus solvency requirements that impact the costs of the insurance activities) is calculated as:

$$\mathrm{E}\left(Y_{j,it}\left(1-L_{j,it}\right)\right) \text{for } j=1,2.$$

This expression is equal to

$$\mathrm{E}\left(Y_{j,it}\right)-\rho_{jL}\mathrm{SD}_{Yj,it}\sqrt{\mathrm{E}\left(L_{j,it}\right)\left(1-\mathrm{E}\left(L_{j,it}\right)\right)}-\mathrm{E}\left(Y_{j,it}\right)\cdot\mathrm{E}\left(L_{j,it}\right),$$

where $\mathrm{SD}_{Yj,it}$ is the standard deviation of the claim cost random variable for the jth line of business. As a result from this expression we can conclude that $\mathrm{E}\left(Y_{j,it}\left(1-L_{j,it}\right)\right)$ should be lower than the marginal expected cost $\mathrm{E}\left(Y_{j,it}\right)$ if ρ_{jL} is positive. Moreover, in the Tweedie model $\mathrm{SD}_{Yj,it}$ can be expressed as a function of parameters ϕ and p and the $\mathrm{E}\left(Y_{j,it}\right)$.

The procedure gives the insurance company a direct way to compute a value of the customer that automatically introduces the expectations about the renewal behavior.

–>

Run Time

Time taken for this program to compile:

```
Time difference of 1.707557 hours
```

Time taken for this report: 102.458762633801 minutes.

References

Czado, Claudia, Rainer Kastenmeier, Eike Christian Brechmann, and Aleksey Min. 2012. "A Mixed Copula Model for Insurance Claims and Claim Sizes." *Scandinavian Actuarial Journal* 2012 (4). Taylor & Francis: 278–305.

Frees, Edward W., and Ping Wang. 2005. "Credibility Using Copulas." *North American Actuarial Journal* 9 (2). Taylor; Francis: 31–48.

Frees, Edward W., Xiaoli Jin, and Xiao Lin. 2013. "Actuarial Applications of Multivariate Two-Part Regression Models." *Annals of Actuarial Science* 7 (02). Cambridge Univ Press: 258–87.

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

Frees, Edward W., Glenn Meyers, and A. David Cummings. 2010. "Dependent Multi-Peril Ratemaking Models." *Astin Bulletin* 40 (02). Cambridge Univ Press: 699–726.

Joe, Harry. 2014. *Dependence Modeling with Copulas.* CRC Press.

Kolev, Nikolai, and Delhi Paiva. 2009. "Copula-Based Regression Models: A Survey." *Journal of Statistical Planning and Inference* 139 (11). Elsevier: 3847–56.

Nikoloulopoulos, Aristidis K. 2013. "Copula-Based Models for Multivariate Discrete Response Data." In *Copulae in Mathematical and Quantitative Finance*, 231–49. Springer.

Shi, Peng, and Edward W. Frees. 2011. "Dependent Loss Reserving Using Copulas." *Astin Bulletin* 41 (02). Cambridge Univ Press: 449–86.

Sun, Jiafeng, Edward W. Frees, and Marjorie A. Rosenberg. 2008. "Heavy-Tailed Longitudinal Data Modeling Using Copulas." *Insurance: Mathematics and Economics* 42 (2). Elsevier: 817–30.

Yang, Xipei, Edward W. Frees, and Zhengjun Zhang. 2011. "A Generalized Beta Copula with Applications in Modeling Multivariate Long-Tailed Data." *Insurance: Mathematics and Economics* 49 (2). Elsevier: 265–84.