mvord: An R Package for Fitting Multivariate Ordinal Regression Models

Rainer Hirk
WU Wirtschafts-
universität Wien

Kurt Hornik
WU Wirtschafts-
universität Wien

Laura Vana
WU Wirtschafts-
universität Wien

Abstract

The R package **mvord** implements composite likelihood estimation in the class of multivariate ordinal regression models with a multivariate probit and a multivariate logit link. A flexible modeling framework for multiple ordinal measurements on the same subject is set up, which takes into consideration the dependence among the multiple observations by employing different error structures. Heterogeneity in the error structure across the subjects can be accounted for by the package, which allows for covariate dependent error structures. In addition, different regression coefficients and threshold parameters for each response are supported. If a reduction of the parameter space is desired, constraints on the threshold as well as on the regression coefficients can be specified by the user. The proposed multivariate framework is illustrated by means of a credit risk application.

Keywords: composite likelihood estimation, correlated ordinal data, multivariate ordinal logit regression model, multivariate ordinal probit regression model, R.

Preface

This vignette corresponds to the article "**mvord**: An R Package for Fitting Multivariate Ordinal Regression Models" which is published in the *Journal of Statistical Software*. When citing the paper and package please use Hirk, Hornik, and Vana (2020a) by calling `citation("mvord")`.

1. Introduction

The analysis of ordinal data is an important task in various areas of research. One of the most common settings is the modeling of preferences or opinions (on a scale from, say, poor to very good or strongly disagree to strongly agree). The scenarios involved range from psychology (e.g., aptitude and personality testing), marketing (e.g., consumer preferences research) and economics and finance (e.g., credit risk assessment for sovereigns or firms) to information retrieval (where documents are ranked by the user according to their relevance) and medical sciences (e.g., modeling of pain severity or cancer stages).

Most of these applications deal with correlated ordinal data, as typically multiple ordinal measurements or outcomes are available for a collection of subjects or objects (e.g., interviewees answering different questions, different raters assigning credit ratings to a firm, pain levels being recorded for patients repeatedly over a period of time, etc.). In such a multi-

variate setting, models which are able to deal with the correlation in the ordinal outcomes are desired. One possibility is to employ a multivariate ordinal regression model where the marginal distribution of the subject errors is assumed to be multivariate. Other options are the inclusion of random effects in the ordinal regression model and conditional models (see, e.g., Fahrmeir and Tutz 2001).

Several ordinal regression models can be employed for the analysis of ordinal data, with cumulative link models being the most popular ones (e.g., Tutz 2012; Christensen 2019a). Other approaches include continuation-ratio or adjacent-category models (e.g., Agresti 2002, 2010). Different packages to analyze and model ordinal data are available in R (R Core Team 2020). For univariate ordinal regression models with fixed effects the function `polr()` of the **MASS** package (Venables and Ripley 2002), the function `clm()` of the **ordinal** package (Christensen 2019b), which supports scale effects as well as nominal effects, and the function `vglm()` of the **VGAM** package (Yee 2010) are available. Another package which accounts for heteroskedasticity is **oglmx** (Carroll 2018). Package **ordinalNet** (Wurm, Rathouz, and Hanlon 2017) offers tools for model selection by using an elastic net penalty, whereas package **ordinalgmifs** (Archer, Hou, Zhou, Ferber, Layne, and Gentry 2014) performs variable selection by using the generalized monotone incremental forward stagewise (GMIFS) method. Moreover, ordinal logistic models can be fitted by the functions `lms()` and `orm()` in package **rms** (Harrell Jr 2019), while ordinal probit models can be fitted by the `MCMCoprobit()` function in package **MCMCpack** (Martin, Quinn, and Park 2011) which uses Markov chain Monte Carlo methods to fit ordinal probit regression models.

An overview on ordinal regression models in other statistical software packages like Stata (StataCorp. 2018), SAS (SAS Institute Inc. 2018b) or SPSS (IBM Corporation 2017) is provided by Liu (2009). These software packages include the Stata procedure `OLOGIT`, the SAS procedure `PROC LOGISTIC` and the SPSS procedure `PLUM` which perform ordinal logistic regression models. The software procedure `PLUM` additionally includes other link functions like probit, complementary log-log, cauchit and negative log-log. Ordinal models for multinomial data are available in the SAS package `PROC GENMOD`, while another implementation of ordinal logistic regression is available in JMP (SAS Institute Inc. 2018a). In Python (Python Software Foundation 2018), package **mord** (Pedregosa-Izquierdo 2015) implements ordinal regression methods.

While there are sufficient software tools in R which deal with the univariate case, the ready-to-use packages for dealing with the multivariate case fall behind, mainly due to computational problems or lack of flexibility in the model specification. However, there are some R packages which support correlated ordinal data. One-dimensional normally distributed random effects in ordinal regression can be handled by the `clmm()` function of package **ordinal** (Christensen 2019b). Multiple possibly correlated random effects are implemented in package **mixor** (Hedeker, Archer, Nordgren, and Gibbons 2018). Note that this package uses multidimensional quadrature methods and estimation becomes infeasible for increasing dimension of the random effects. Bayesian multilevel models for ordinal data are implemented in package **brms** (Bürkner 2017). Multivariate ordinal probit models, where the subject errors are assumed to follow a multivariate normal distribution with a general correlation matrix, can be estimated with package **PLordprob** (Kenne Pagui and Canale 2018), which uses maximum composite likelihood methods estimation. This package works well for standard applications but lacks flexibility. For example, the number of levels of the ordinal responses needs to be equal across all dimensions, threshold and regression coefficients are the same for all multiple

measurements and the package does not account for missing observations in the outcome variable. Polychoric correlations, which are used to measure association among two ordinal outcomes, can be estimated by the `polychor()` function of package **polycor** (Fox 2019), where a simple bivariate probit model without covariates is estimated using maximum likelihood estimation. None of these packages support at the time of writing covariate dependent error structures. A package which allows for different error structures in non-linear mixed effects models is package **nlme** (Pinheiro, Bates, and R Core Team 2020), even though models dealing with ordinal data are not supported.

The original motivation for this package lies in a credit risk application, where multiple credit ratings are assigned by various credit rating agencies (CRAs) to firms over several years. CRAs have an important role in financial markets, as they deliver subjective assessments or opinions of an entity's creditworthiness, which are then used by the other players on the market, such as investors and regulators, in their decision making process. Entities are assigned to rating classes by CRAs on an ordinal scale by using both quantitative and qualitative criteria. Ordinal credit ratings can be seen as a coarser version of an underlying continuous latent process, which is related to the ability of the firm to meet its financial obligations. In the literature, this latent variable motivation has been used in various credit rating models (e.g., Blume, Lim, and Mackinlay 1998; Afonso, Gomes, and Rother 2009; Alp 2013; Reusens and Croux 2017).

This setting is an example of an application where correlated ordinal data arise naturally. On the one hand, multiple ratings assigned by different raters to one firm at the same point in time can be assumed to be correlated. On the other hand, given the longitudinal dimension of the data, for each rater, there is serial dependence in the ratings assigned over several periods. Moreover, aside from the need of a model class that can handle correlated ordinal data, additional flexibility is desired due to the following characteristics of the problem at hand: Firstly, there is heterogeneity in the rating methodology. Raters use different labeling as well as a different number of rating classes. Secondly, the credit risk measure employed in assessing creditworthiness can differ among raters (e.g., probability of default versus recovery in case of default), which leads to heterogeneity in the covariates, as raters might use different variables in their rating process and assign different importance to the variables employed. Thirdly, the data have missing values and are unbalanced, as firms can leave the data set before the end of the observation period due to various reasons such as default but also because of mergers and acquisitions, privatizations, etc., or ratings can be withdrawn. Moreover, there are missings in the multiple ratings, as not all firms are rated by all raters at each time point.

The scope of the application of multivariate ordinal regression models reaches far beyond credit risk applications. For example, pain severity studies are a popular setting where repeated ordinal measurements occur. A migraine severity study was employed by Varin and Czado (2010), where patients recorded their pain severity over some time period. In addition to a questionnaire with personal and clinical information, covariates describing the weather conditions were collected. Another application area constitutes the field of customer satisfaction surveys, where questionnaires with ordinal items are often divided into two separate blocks (e.g., Kenne Pagui and Canale 2016). A first block contains questions regarding the general importance of some characteristics of a given service, and a second block relates more to the actual satisfaction on the same characteristics. An analysis of the dependence structure between and within the two blocks is of particular interest. Furthermore, in the presence of multi-rater agreement data, where several raters assign ordinal rankings to different indi-

viduals, the influence of covariates on the ratings can be investigated and an analysis and a comparison of the rater behavior can be conducted (e.g., DeYoreo and Kottas 2018). In addition to these few examples mentioned above, the class of multivariate ordinal regression models implemented in **mvord** (Hirk, Hornik, and Vana 2020b) can be applied to other settings where multiple or repeated ordinal observations occur.

This paper discusses package **mvord** for R which aims at providing a flexible framework for analyzing correlated ordinal data by means of the class of multivariate ordinal regression models and which is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=mvord`. In this model class, each of the ordinal responses is modeled as a categorized version of an underlying continuous latent variable which is slotted according to some threshold parameters. On the latent scale we assume a linear model for each of the underlying continuous variables and the existence of a joint distribution for the corresponding error terms. A common choice for this joint distribution is the multivariate normal distribution, which corresponds to the multivariate probit link. We extend the available software in several directions. The flexible modeling framework allows imposing constraints on threshold as well as regression coefficients. In addition, various assumptions about the variance-covariance structure of the errors are supported, by specifying different types of error structures. These include a general correlation, a general covariance, an equicorrelation and an $AR(1)$ error structure. The general error structures can depend on a categorical covariate, while in the equicorrelation and $AR(1)$ structures both numerical and categorical covariates can be employed. Moreover, in addition to the multivariate probit link, we implement a multivariate logit link for the class of multivariate ordinal regression models.

This paper is organized as follows: Section 2 provides an overview of the model class and the estimation procedure, including model specification and identifiability issues. Section 3 presents the main functions of the package. A couple of worked examples are given in Section 4. Section 5 concludes.

2. Model class and estimation

Multivariate ordinal regression models are an appropriate modeling choice when a vector of correlated ordinal response variables, together with covariates, is observed for each unit or subject in the sample. The response vector can be composed of different variables, i.e., multiple measurements on the same subject (e.g., different credit ratings assigned to a firm by different CRAs, different survey questions answered by an interviewee, etc.) or repeated measurements on the same variable at different time points.

In order to introduce the class of multivariate ordinal regression models considered in this paper, we start with a brief overview on univariate cumulative link models.

2.1. Univariate cumulative link models

Cumulative link models are often motivated by the assumption that the observed categories Y_i are a categorized version of an underlying latent variable $\widetilde{Y}_i$ with

$$\widetilde{Y}_i = \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} + \epsilon_i,$$

where β_0 is an intercept term, $\boldsymbol{x}_i$ is a $p \times 1$ vector of covariates, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top$ is a vector of regression coefficients and ϵ_i is a mean zero error term with distribution function F. The

link between the observed variable Y_i with K categories and the latent variable $\widetilde{Y}_i$ is given by:

$$Y_i = r_i \quad \Leftrightarrow \quad \theta_{r_i-1} < \widetilde{Y}_i \leq \theta_{r_i}, \qquad r_i \in \{1, \ldots, K\},$$

where $-\infty \equiv \theta_0 < \theta_1 < \cdots < \theta_{K-1} < \theta_K \equiv \infty$ are threshold parameters on the latent scale (see, e.g., Agresti 2010; Tutz 2012). In such a setting the ordinal response variable Y_i follows a multinomial distribution with parameter $\boldsymbol{\pi}_i$. Let denote by π_{ir_i} the probability that observation i falls in category r_i. Then the cumulative link model (McCullagh 1980) is specified by:

$$\mathsf{P}(Y_i \leq r_i) = \mathsf{P}(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} + \epsilon_i \leq \theta_{r_i}) = F(\theta_{r_i} - \beta_0 - \boldsymbol{x}_i^\top \boldsymbol{\beta}) = \pi_{i1} + \cdots + \pi_{ir_i}.$$

Typical choices for the distribution function F are the normal and the logistic distributions.

2.2. Multivariate ordinal regression

Univariate cumulative link models can be extended to a multivariate setting by assuming the existence of several latent variables with a joint error distribution (see, e.g., Varin and Czado 2010; Bhat, Varin, and Ferdous 2010; Kenne Pagui and Canale 2016). Let Y_{ij} denote an ordinal observation and $\boldsymbol{x}_{ij}$ be a p-dimensional vector of covariates for subject i and outcome j, where $i = 1, \ldots, n$ and $j \in J_i$, for J_i a subset of all available outcomes J in the data set. Moreover, we denote by $q = |J|$ and $q_i = |J_i|$ the number of elements in the sets J and J_i, respectively. Following the cumulative link modeling approach, the ordinal response Y_{ij} is assumed to be a coarser version of a latent continuous variable $\widetilde{Y}_{ij}$. The observable categorical outcome Y_{ij} and the unobservable latent variable $\widetilde{Y}_{ij}$ are connected by:

$$Y_{ij} = r_{ij} \quad \Leftrightarrow \quad \theta_{j,r_{ij}-1} < \widetilde{Y}_{ij} \leq \theta_{j,r_{ij}}, \qquad r_{ij} \in \{1, \ldots, K_j\},$$

where r_{ij} is a category out of K_j ordered categories and $\boldsymbol{\theta}_j$ is a vector of suitable threshold parameters for outcome j with the following restriction: $-\infty \equiv \theta_{j,0} < \theta_{j,1} < \cdots < \theta_{j,K_j-1} < \theta_{j,K_j} \equiv \infty$. Note that in this setting binary observations can be treated as ordinal observations with two categories ($K_j = 2$).

The following linear model is assumed for the relationship between the latent variable $\widetilde{Y}_{ij}$ and the vector of covariates $\boldsymbol{x}_{ij}$:

$$\widetilde{Y}_{ij} = \beta_{j0} + \boldsymbol{x}_{ij}^\top \boldsymbol{\beta}_j + \epsilon_{ij}, \tag{1}$$

where β_{j0} is an intercept term, $\boldsymbol{\beta}_j = (\beta_{j1}, \ldots, \beta_{jp})^\top$ is a vector of regression coefficients, both corresponding to outcome j. We further assume the n subjects to be independent. Note that the number of ordered categories K_j as well as the threshold parameters $\boldsymbol{\theta}_j$ and the regression coefficients $\boldsymbol{\beta}_j$ are allowed to vary across outcome dimensions $j \in J$ to account for possible heterogeneity across the response variables.

Category-specific regression coefficients. By employing one set of regression coefficients $\boldsymbol{\beta}_j$ for all categories of the jth outcome it is implied that the relationship between the covariates and the responses does not depend on the category. This assumption is called parallel regression or proportional odds assumption (McCullagh 1980) and can be relaxed for one or more covariates by allowing the corresponding regression coefficients to be category-specific (see, e.g., Peterson and Harrell 1990).

Link functions. The dependence among the different responses is accounted for by assuming that, for each subject i, the vector of error terms $\boldsymbol{\epsilon}_i = [\epsilon_{ij}]_{j \in J_i}$ follows a suitable multivariate distribution. We consider two multivariate distributions which correspond to the multivariate probit and logit link functions. For the multivariate probit link, we assume that the errors follow a multivariate normal distribution: $\boldsymbol{\epsilon}_i \sim \mathcal{N}_{q_i}(\mathbf{0}, \boldsymbol{\Sigma}_i)$. A multivariate logit link is constructed by employing a multivariate logistic distribution family with univariate logistic margins and a t copula with certain degrees of freedom as proposed by O'Brien and Dunson (2004). For a vector $\boldsymbol{z} = (z_1, \ldots, z_q)^\top$, the multivariate logistic distribution function with $\nu > 2$ degrees of freedom, location vector $\boldsymbol{\mu}$ and positive-definite dispersion matrix $\boldsymbol{\Sigma}$ is defined as:

$$F_{\nu,\boldsymbol{\mu},\boldsymbol{\Sigma}}(\boldsymbol{z}) = t_{\nu,\boldsymbol{R}}(\{g_\nu((z_1 - \mu_1)/\sigma_1), \ldots, g_\nu((z_q - \mu_q)/\sigma_q)\}^\top), \tag{2}$$

where $t_{\nu,\boldsymbol{R}}$ is the q-dimensional multivariate t distribution with ν degrees of freedom and correlation matrix $\boldsymbol{R}$ implied by $\boldsymbol{\Sigma}$, $g_\nu(x) = t_\nu^{-1}(\exp(x)/(\exp(x)+1))$, with t_ν^{-1} the quantile function of the univariate t distribution with ν degrees of freedom and $\sigma_1^2, \ldots, \sigma_q^2$ the diagonal elements of $\boldsymbol{\Sigma}$.

Hirk, Hornik, and Vana (2019) employed this t copula based multivariate logistic family, while Nooraee, Abegaz, Ormel, Wit, and Van den Heuvel (2016) used a multivariate t distribution with $\nu = 8$ degrees of freedom as an approximation for this multivariate logistic distribution. The employed distribution family differs from the conventional multivariate logistic distributions of Gumbel (1961) or Malik and Abraham (1973) in that it offers a more flexible dependence structure through the correlation matrix of the t copula, while still keeping the log odds interpretation of the regression coefficients through the univariate logistic margins.

2.3. Identifiability issues

As the absolute scale and the absolute location are not identifiable in ordinal models, further restrictions on the parameter set need to be imposed. Assuming $\boldsymbol{\Sigma}_i$ to be a covariance matrix with diagonal elements $[\sigma_{ij}^2]_{j \in J_i}$, only the quantities $\boldsymbol{\beta}_j/\sigma_{ij}$ and $(\theta_{j,r} - \beta_{j0})/\sigma_{ij}$ are identifiable in the model in Equation 1. Hence, in order to obtain an identifiable model the parameter set is typically constrained in one of the following ways:

- Fixing the intercept β_{j0} (e.g., to zero), using flexible thresholds $\boldsymbol{\theta}_j$ and fixing σ_{ij} (e.g., to unity) $\forall j \in J_i$, $\forall i \in \{1, \ldots, n\}$.
- Leaving the intercept β_{j0} unrestricted, fixing one threshold parameter (e.g., $\theta_{j,1} = 0$) and fixing σ_{ij} (e.g., to unity) $\forall j \in J_i$, $\forall i \in \{1, \ldots, n\}$.
- Fixing the intercept β_{j0} (e.g., to zero), fixing one threshold parameter (e.g., $\theta_{j,1} = 0$) and leaving σ_{ij} unrestricted $\forall j \in J_i$, $\forall i \in \{1, \ldots, n\}$.
- Leaving the intercept β_{j0} unrestricted, fixing two threshold parameters (e.g., $\theta_{j,1} = 0$ and $\theta_{j,2} = 1$) and leaving σ_{ij} unrestricted $\forall j \in J_i$, $\forall i \in \{1, \ldots, n\}$[1].

Note that the first two options are the most commonly used in the literature. All of these alternative model parameterizations are supported by the **mvord** package, allowing the user

[1]Note that this parameterization cannot be applied to the binary case.

to choose the most convenient one for each specific application. Table 2 in Section 3.5 gives an overview on the identifiable parameterizations implemented in the package.

2.4. Error structures

Different structures on the covariance matrix $\boldsymbol{\Sigma}_i$ can be imposed.

Basic model

The basic multivariate ordinal regression model assumes that the correlation (and possibly variance, depending on the parameterization) parameters in the distribution function of the $\boldsymbol{\epsilon}_i$ are constant for all subjects i.

Correlation. The dependence between the multiple measurements or outcomes can be captured by different correlation structures. Among them, we concentrate on the following three:

- The general correlation structure assumes different correlation parameters between pairs of outcomes $\mathsf{COR}(\epsilon_{ik}, \epsilon_{il}) = \rho_{kl}$. This error structure is among the most common in the literature (e.g., Scott and Kanaroglou 2002; Bhat *et al.* 2010; Kenne Pagui and Canale 2016).
- The equicorrelation structure $\mathsf{COR}(\epsilon_{ik}, \epsilon_{il}) = \rho$ implies that the correlation between all pairs of outcomes is constant.
- When faced with longitudinal data, especially when moderate to long subject-specific time series are available, an $AR(1)$ autoregressive correlation model of order one can be employed. Given equally spaced time points this $AR(1)$ error structure implies an exponential decay in the correlation with the lag. If k and l are the time points when Y_{ik} and Y_{il} are observed, then $\mathsf{COR}(\epsilon_{ik}, \epsilon_{il}) = \rho^{|k-l|}$.

Variance. If a parameterization with identifiable variance is used (see Section 2.3), in the basic model we assume that for each multiple measurement the variance is constant across all subjects ($\mathsf{VAR}(\epsilon_{ij}) = \sigma_j^2$).

Extending the basic model

In some applications, the constant correlation (and variance) structure across subjects may be too restrictive. We hence extend the basic model by allowing the use of covariates in the correlation (and variance) specifications.

Correlation. For each subject i and each pair (k, l) from the set J_i, the correlation parameter ρ_{ikl} is assumed to depend on a vector $\boldsymbol{s}_i$ of m subject-specific covariates. In this paper we use the hyperbolic tangent transformation to reparameterize the linear term $\alpha_{0kl} + \boldsymbol{s}_i^\top \boldsymbol{\alpha}_{kl}$ in terms of a correlation parameter:

$$\frac{1}{2} \log \left(\frac{1 + \rho_{ikl}}{1 - \rho_{ikl}} \right) = \alpha_{0kl} + \boldsymbol{s}_i^\top \boldsymbol{\alpha}_{kl}, \qquad \rho_{ikl} = \frac{e^{2(\alpha_{0kl} + \boldsymbol{s}_i^\top \boldsymbol{\alpha}_{kl})} - 1}{e^{2(\alpha_{0kl} + \boldsymbol{s}_i^\top \boldsymbol{\alpha}_{kl})} + 1}.$$

If $\boldsymbol{\alpha}_{kl} = 0$ for all $k, l \in J_i$, this model would correspond to the general correlation structure in the basic model. Moreover, if $\alpha_{0kl} = 0$ and $\boldsymbol{\alpha}_{kl} = 0$ for all $k, l \in J_i$, the correlation matrix is the identity matrix and the responses are uncorrelated.

For the more parsimonious error structures of equicorrelation and $AR(1)$, in the extended model the correlation parameters are modeled as:

$$\frac{1}{2}\log\left(\frac{1+\rho_i}{1-\rho_i}\right) = \alpha_0 + \boldsymbol{s}_i^\top \boldsymbol{\alpha}, \qquad \rho_i = \frac{e^{2(\alpha_0 + \boldsymbol{s}_i^\top \boldsymbol{\alpha})} - 1}{e^{2(\alpha_0 + \boldsymbol{s}_i^\top \boldsymbol{\alpha})} + 1}.$$

Variance. Similarly, one can model the heterogeneity among the subjects through the variance parameters $\mathsf{VAR}(\epsilon_{ij}) = \sigma_{ij}^2$ by employing the following linear model on the log-variance:

$$\log(\sigma_{ij}^2) = \gamma_{0j} + \boldsymbol{s}_i^\top \boldsymbol{\gamma}_j.$$

Note that other suitable link functions for the correlation and variance parameterizations could also be applied. The positive-semi-definiteness of the correlation (or covariance) matrix $\boldsymbol{\Sigma}_i$ can be ensured by the use of special algorithms such as the one proposed by Higham (1988).

2.5. Composite likelihood estimation

In order to estimate the model parameters we use a composite likelihood approach, where the full likelihood is approximated by a pseudo-likelihood which is constructed from lower dimensional marginal distributions, more specifically by "aggregating" the likelihoods corresponding to pairs of observations (Varin, Reid, and Firth 2011).

For a given parameter vector $\boldsymbol{\delta}$, which contains the threshold parameters, the regression coefficients and the parameters of the error structure, the likelihood is given by:

$$\mathscr{L}(\boldsymbol{\delta}) = \prod_{i=1}^{n} \mathsf{P}\Big(\bigcap_{j \in J_i} \{Y_{ij} = r_{ij}\}\Big)^{w_i} = \prod_{i=1}^{n} \Big(\int_{D_i} f_{i,q_i}(\widetilde{\boldsymbol{Y}}_i; \boldsymbol{\delta}) d^{q_i}\widetilde{\boldsymbol{Y}}_i\Big)^{w_i},$$

where $D_i = \prod_{j \in J_i}(\theta_{j,r_{ij}-1}, \theta_{j,r_{ij}})$ is a Cartesian product, w_i are subject-specific non-negative weights (which are set to one in the default case) and f_{i,q_i} is the q_i-dimensional density of the error terms $\boldsymbol{\epsilon}_i$. We approximate this full likelihood by a pairwise likelihood which is constructed from bivariate marginal distributions. If the number of observed outcomes for subject i is less than two ($q_i < 2$), the univariate marginal distribution enters the likelihood. The pairwise log-likelihood function is obtained by:

$$p\ell(\boldsymbol{\delta}) = \sum_{i=1}^{n} w_i \Bigg[\mathbb{1}_{\{q_i \geq 2\}} \sum_{k=1}^{q-1} \sum_{l=k+1}^{q} \mathbb{1}_{\{k,l \in J_i\}} \log\left(\mathsf{P}(Y_{ik} = r_{ik}, Y_{il} = r_{il})\right) + \mathbb{1}_{\{q_i = 1\}} \sum_{k=1}^{q} \mathbb{1}_{\{k \in J_i\}} \log\left(\mathsf{P}(Y_{ik} = r_{ik})\right) \Bigg]. \tag{3}$$

Denoting by $f_{i,1}$ and $f_{i,2}$ the uni- and bivariate density functions corresponding to the error

distribution, the uni- and bivariate probabilities are given by:

$$\mathsf{P}(Y_{ik} = r_{ik}, Y_{il} = r_{il}) = \int_{\theta_{k,r_{ik}-1}}^{\theta_{k,r_{ik}}} \int_{\theta_{l,r_{il}-1}}^{\theta_{l,r_{il}}} f_{i,2}(\widetilde{Y}_{ik}, \widetilde{Y}_{il}; \boldsymbol{\delta}) d\widetilde{Y}_{ik} d\widetilde{Y}_{il},$$
$$\mathsf{P}(Y_{ik} = r_{ik}) = \int_{\theta_{k,r_{ik}-1}}^{\theta_{k,r_{ik}}} f_{i,1}(\widetilde{Y}_{ik}; \boldsymbol{\delta}) d\widetilde{Y}_{ik}.$$

The maximum pairwise likelihood estimates $\hat{\boldsymbol{\delta}}_{p\ell}$ are obtained by direct maximization of the composite likelihood given in Equation 3. The threshold and error structure parameters to be estimated are reparameterized such that unconstrained optimization can be performed. Firstly, we reparameterize the threshold parameters in order to achieve monotonicity. Secondly, for all unrestricted correlation (and covariance) matrices we use the spherical parameterization of Pinheiro and Bates (1996). This parameterization has the advantage that it can be easily applied to correlation matrices. Thirdly, for equicorrelated or $AR(1)$ errors, we use the hyperbolic tangent transformation.

Computation of the standard errors is needed in order to quantify the uncertainty of the maximum pairwise likelihood estimates. Under certain regularity conditions, the maximum pairwise likelihood estimates are consistent as the number of responses is fixed and $n \to \infty$. In addition, the maximum pairwise likelihood estimator is asymptotically normal with asymptotic mean $\boldsymbol{\delta}$ and a covariance matrix which equals the inverse of the Godambe information matrix:

$$G(\boldsymbol{\delta})^{-1} = H(\boldsymbol{\delta})^{-1} V(\boldsymbol{\delta}) H(\boldsymbol{\delta})^{-1},$$

where $H(\boldsymbol{\delta})$ is the Hessian (sensitivity matrix) and $V(\boldsymbol{\delta})$ the variability matrix. The variability matrix $V(\boldsymbol{\delta})$ and the Hessian $H(\boldsymbol{\delta})$ can be estimated as:

$$\widehat{V}(\boldsymbol{\delta}) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\partial p\ell_i(\boldsymbol{\delta})}{\partial \boldsymbol{\delta}} \right) \left(\frac{\partial p\ell_i(\boldsymbol{\delta})}{\partial \boldsymbol{\delta}} \right)^{\top} \Bigg|_{\boldsymbol{\delta} = \hat{\boldsymbol{\delta}}_{p\ell}},$$

and

$$\widehat{H}(\boldsymbol{\delta}) = -\frac{1}{n} \sum_{i=1}^{n} \frac{\partial^2 p\ell_i(\boldsymbol{\delta})}{\partial \boldsymbol{\delta} \partial \boldsymbol{\delta}^{\top}} \Bigg|_{\boldsymbol{\delta} = \hat{\boldsymbol{\delta}}_{p\ell}} = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{q} \sum_{l=k+1}^{q-1} \mathbb{1}_{\{k,l \in J_i\}} \left(\frac{\partial p\ell_{ikl}(\boldsymbol{\delta})}{\partial \boldsymbol{\delta}} \right) \left(\frac{\partial p\ell_{ikl}(\boldsymbol{\delta})}{\partial \boldsymbol{\delta}} \right)^{\top} \Bigg|_{\boldsymbol{\delta} = \hat{\boldsymbol{\delta}}_{p\ell}},$$

where $p\ell_i(\boldsymbol{\delta})$ is the component of the pairwise log-likelihood corresponding to subject i and $p\ell_{ikl}(\boldsymbol{\delta})$ corresponds to subject i and pair (k, l).

In order to compare different models, the composite likelihood information criterion by Varin and Vidoni (2005) can be used: $\text{CLIC}(\boldsymbol{\delta}) = -2\ p\ell(\hat{\boldsymbol{\delta}}_{p\ell}) + k\ \text{tr}(\widehat{V}(\boldsymbol{\delta})\widehat{H}(\boldsymbol{\delta})^{-1})$ (where $k = 2$ corresponds to CLAIC and $k = \log(n)$ corresponds to CLBIC). A comprehensive overview and further details on the properties of the maximum composite likelihood estimates are provided in Varin (2008).

2.6. Interpretation of the coefficients

Unlike in linear regression models, the interpretation of the regression coefficients and of the threshold parameters in ordinal models is not straightforward. Estimated thresholds

and coefficients represent only signal to noise ratios and cannot be interpreted directly (see Section 2.3). For one particular outcome j, the coefficients can be interpreted in the same way as in univariate cumulative link models. Let us assume without loss of generality that a higher latent score leads to better ratings on the ordinal scale. This implies that the first category is the worst and category K_j is the best category. In this section we assume for sake of notational simplicity that $\mathbf{\Sigma}_i$ is a correlation matrix implying that marginally the errors of subject i have variance one and univariate marginal distribution function F_1 for each outcome j. In the more general case with non-constant variances σ_{ij}^2, $F_{i,1}^j$ should be used instead of F_1. The marginal cumulative probabilities implied by the model in Equation 1 are then given by the following relationship:

$$\mathsf{P}(Y_{ij} \leq r_{ij}|\boldsymbol{x}_{ij}) = \mathsf{P}(\boldsymbol{x}_{ij}^\top\boldsymbol{\beta}_j + \epsilon_{ij} \leq \theta_{j,r_{ij}}) = \mathsf{P}(\epsilon_{ij} \leq \theta_{j,r_{ij}} - \boldsymbol{x}_{ij}^\top\boldsymbol{\beta}_j) = F_1(\theta_{j,r_{ij}} - \boldsymbol{x}_{ij}^\top\boldsymbol{\beta}_j).$$

One natural way to interpret ordinal regression models is to analyze partial effects, where one is interested in how a marginal change in one variable x_{ijv} changes the outcome distribution. The partial probability effects in the cumulative model are given by:

$$\delta^j_{r_{ij},v}(\boldsymbol{x}_{ij}) = \frac{\partial \mathsf{P}(Y_{ij} = r_{ij}|\boldsymbol{x}_{ij})}{\partial x_{ijv}} = -\left(f_1(\theta_{j,r_{ij}} - \boldsymbol{x}_{ij}^\top\boldsymbol{\beta}_j) - f_1(\theta_{j,r_{ij}-1} - \boldsymbol{x}_{ij}^\top\boldsymbol{\beta}_j)\right)\beta_{jv},$$

where f_1 is the density corresponding to F_1, x_{ijv} is the vth element in $\boldsymbol{x}_{ij}$ and β_{jv} is the vth element in $\boldsymbol{\beta}_j$. In case of discrete variables it is more appropriate to consider the changes in probability before and after the change in the variable instead of the partial effects using:

$$\Delta\mathsf{P}(Y_{ij} = r_{ij}|\boldsymbol{x}_{ij}, \tilde{\boldsymbol{x}}_{ij}) = \mathsf{P}(Y_{ij} = r_{ij}|\tilde{\boldsymbol{x}}_{ij}) - \mathsf{P}(Y_{ij} = r_{ij}|\boldsymbol{x}_{ij}),$$

where all elements of $\tilde{\boldsymbol{x}}_{ij}$ are equal to $\boldsymbol{x}_{ij}$ except for the vth element, which is equal to $\tilde{x}_{ijv} = x_{ijv} + \Delta x_{ijv}$ for the change Δx_{ijv} in the variable x_v. We refer to Greene and Hensher (2010) and Boes and Winkelmann (2006) for further discussion of the interpretation of partial effects in ordered response models.

In the presence of the probit link function, we have the following relationship between the cumulative probabilities and the latent process:

$$\Phi^{-1}\left(\mathsf{P}(Y_{ij} \leq r_{ij}|\boldsymbol{x}_{ij})\right) = \theta_{j,r_{ij}} - \boldsymbol{x}_{ij}^\top\boldsymbol{\beta}_j.$$

An increase of one unit in variable v of outcome j (given that all other variables are held constant) changes the probit of the probability that category r_{ij} or lower is observed by the value of the coefficient β_{jv} of this variable. In other words $\mathsf{P}(Y_{ij} \leq r_{ij}|\boldsymbol{x}_{ij})$, the probability that category r_{ij} or lower is observed, changes by the increase/decrease in the distribution function. Moreover, predicted probabilities for all ordered response categories can be calculated and compared for given sets of explanatory variables.

In the presence of the logit link function, the regression coefficients of the underlying latent process are scaled in terms of marginal log odds (McCullagh 1980):

$$\log\left(\frac{\mathsf{P}(Y_{ij} \leq r_{ij}|\boldsymbol{x}_{ij})}{\mathsf{P}(Y_{ij} > r_{ij}|\boldsymbol{x}_{ij})}\right) = \theta_{j,r_{ij}} - \boldsymbol{x}_{ij}^\top\boldsymbol{\beta}_j.$$

For a one unit increase in variable v of outcome j holding all the others constant, we expect a change of size of the coefficient β_{jv} of this variable in the expected value on the log odds scale.

Due to the fact that the marginal effects of the odds ratios do not depend on the category, one often exponentiates the coefficients in order to obtain the following convenient interpretation in terms of odds ratios:

$$\frac{\mathsf{P}(Y_{ij} \leq r_{ij}|\boldsymbol{x}_{ij})/\mathsf{P}(Y_{ij} > r_{ij}|\boldsymbol{x}_{ij})}{\mathsf{P}(Y_{ij} \leq r_{ij}|\tilde{\boldsymbol{x}}_{ij})/\mathsf{P}(Y_{ij} > r_{ij}|\tilde{\boldsymbol{x}}_{ij})} = \exp((\tilde{\boldsymbol{x}}_{ij} - \boldsymbol{x}_{ij})^\top \boldsymbol{\beta}_j).$$

This means for a one unit increase in variable v of outcome j, holding all the other variables constant, changes the odds ratio by $\exp(\beta_{jv})$. In other words, the odds after a one unit change in variable v of outcome j are the odds before the change multiplied by $\exp(-\beta_{jv})$:

$$\frac{\mathsf{P}(Y_{ij} \leq r_{ij}|\boldsymbol{x}_{ij})}{\mathsf{P}(Y_{ij} > r_{ij}|\boldsymbol{x}_{ij})} \exp(-\beta_{jv}) = \frac{\mathsf{P}(Y_{ij} \leq r_{ij}|\tilde{\boldsymbol{x}}_{ij})}{\mathsf{P}(Y_{ij} > r_{ij}|\tilde{\boldsymbol{x}}_{ij})}.$$

If the regression coefficients vary across the multiple responses, they cannot be compared directly due to the fact that the measurement units of the underlying latent processes differ. Nevertheless, one possibility to compare coefficients is through the concept of importance. Reusens and Croux (2017) extend an approach for comparing coefficients of probit and logit models by Hoetker (2007) in order to compare the coefficients across repeated measurements. They analyze the importance ratio

$$R_{jv} = \frac{\beta_{jv}}{\beta_{j,base}},$$

where $\beta_{j,base}$ is the coefficient of a base variable and v is one of the remaining $p-1$ variables. This ratio can be interpreted as follows: A one unit increase in the variable v has in expectation the same effect in the *base* variable multiplied by the ratio R_{jv}. Another interpretation is the so called compensation variation: The ratio is the required increase in the *base* variable that is necessary to compensate a one unit decrease in the variable v in a way that the score of the outcome remains the same. It is to be noted that the importance ratio R_{jv} depends on the scale of the *base* variable and variable v of outcome j. This implies that the comparison among the measurements j should be done only if the scales of these variables are equal across the multiple measurements. For this purpose, standardization of the covariates for each measurement should be employed.

3. Implementation

The **mvord** package contains six data sets and the built-in functions presented in Table 1. Multivariate ordinal regression models in the R package **mvord** can be fitted using the main function `mvord()`. Two different data structures can be passed on to the `mvord()` function through the use of two different multiple measurement objects `MMO` and `MMO2` in the left-hand side of the model formula. `MMO` uses a long data format, which has the advantage that it allows for varying covariates across multiple measurements. This flexibility requires to specify a subject index as well as a multiple measurement index. In contrast to `MMO`, the multiple measurement object `MMO2` has a simplified data structure but is only applicable in settings where the covariates do not vary between the multiple measurements. In this case, the multiple ordinal observations as well as the covariates are stored in different columns of a '`data.frame`'. We refer to this data structure as wide data format.

Function	Description
Fitting function	
`mvord(formula, data, ...)`	Estimates the multivariate ordinal regression model.
Prediction functions	
`predict(object, type, ...)`	Obtains different types of predicted or fitted values from the joint distribution of the responses for objects of class '`mvord`'.
`marginal_predict(object, type, ...)`	Obtains different types of predictions or fitted values from the marginal distributions of the responses for objects of class '`mvord`'.
`joint_probabilities(object, response.cat, ...)`	For each subject, the joint probability of observing a predefined configuration of responses `response.cat` is computed for objects of class '`mvord`'.
Utility functions	
`coef(object, ...)`	Extracts the estimated regression coefficients.
`thresholds(object, ...)`	Extracts the estimated threshold coefficients.
`error_structure(object, type, ...)`	Extracts for each subject the estimated parameters of the error structure.
`constraints(object)`	Extracts the constraint matrices corresponding to each regression coefficient.
`names_constraints(formula, data, ...)`	Extracts the names of the regression coefficients in the model matrix.
`pseudo_R_squared(object, ...)`	Computes McFadden's Pseudo R^2.
Other methods for objects of class 'mvord'.	
`summary()`, `print()`, `vcov()`, `fitted()`, `model.matrix()`, `terms()`, `nobs()`, `logLik()`	

Table 1: This table summarizes fitting, prediction, utility functions and other methods implemented in **mvord**.

For illustration purposes we use a worked example based on a simulated data set consisting of 100 subjects for which two multiple ordinal responses (`Y1` and `Y2`), two continuous covariates (`X1` and `X2`) and two factor covariates (`f1` and `f2`) are available. The ordinal responses each have three categories labeled with 1, 2 and 3.

```
R> data("data_mvord_toy", package = "mvord")
R> str(data_mvord_toy)

'data.frame':        100 obs. of  6 variables:
 $ Y1: Ord.factor w/ 3 levels "1"<"2"<"3": 1 3 3 1 2 1 2 2 2 3 ...
 $ Y2: Ord.factor w/ 3 levels "1"<"2"<"3": 1 3 3 1 2 1 2 2 1 3 ...
 $ X1: num  -0.789 0.93 2.804 1.445 -0.191 ...
 $ X2: num  1.3653 -0.00982 -0.25878 3.90187 0.04958 ...
```

```
 $ f1: Factor w/ 3 levels "A","B","C": 3 2 2 3 3 3 2 2 3 1 ...
 $ f2: Factor w/ 2 levels "c1","c2": 2 2 2 1 2 2 1 2 2 1 ...
```

The data set `data_mvord_toy` has a wide format. We convert the data set into the long format, where the first column contains the subject index i and the second column the multiple measurement index j.

```
R> str(data_toy_long)
```

```
'data.frame':        200 obs. of  7 variables:
 $ i : int  1 2 3 4 5 6 7 8 9 10 ...
 $ j : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Y : Ord.factor w/ 3 levels "1"<"2"<"3": 1 3 3 1 2 1 2 2 2 3 ...
 $ X1: num  -0.789 0.93 2.804 1.445 -0.191 ...
 $ X2: num  1.3653 -0.00982 -0.25878 3.90187 0.04958 ...
 $ f1: Factor w/ 3 levels "A","B","C": 3 2 2 3 3 3 2 2 3 1 ...
 $ f2: Factor w/ 2 levels "c1","c2": 2 2 2 1 2 2 1 2 2 1 ...
```

3.1. Implementation `MMO`

The fitting function `mvord()` requires two compulsory input arguments, a `formula` argument and a `data` argument:

```
R> res <- mvord(formula = MMO(Y, i, j) ~ 0 + X1 + X2, data = data_toy_long)
```

(runtime 1.72 seconds).[2]

Data structure

In `MMO` we use a long format for the input of `data`, where each row contains a subject index `i`, a multiple measurement index `j`, an ordinal observation `Y` and all the covariates (`X1` to `Xp`). This long format data structure is internally transformed to an $n \times q$ matrix of responses which contains `NA` in the case of missing entries and a list of covariate matrices $\boldsymbol{X}_j$ for all $j \in J$. This is performed by the multiple measurement object `MMO(Y, i, j)` which specifies the column names of the subject index and the multiple measurement index in `data`. The column containing the ordinal observations can contain integer or character values or inherit from class (ordered) '`factor`'. When using the long data structure, this column is basically a concatenated vector of each of the multiple ordinal responses. Internally, this vector is then split according to the measurement index. Then the ordinal variable corresponding to each measurement index is transformed into an ordered '`factor`'. For an integer or a character vector the natural ordering is used (ascending, or alphabetical). If for character vectors the alphabetical order does not correspond to the ordering of the categories, the optional argument `response.levels` allows to specify the levels for each response explicitly. This is performed by a list of length q, where each element contains the names of the levels of the ordered categories in ascending (or if desired descending) order. If all the multiple measurements use

[2]Computations have been performed with R version 3.4.4 on a machine with an Intel Core i5-4200U CPU 1.60GHz processor and 8GB RAM.

the same number of classes and same labeling of the classes, the column `Y` can be stored as an ordered '`factor`' (as it is often the case in longitudinal studies).

The order of the multiple measurements is needed when specifying constraints on the threshold or regression parameters (see Sections 3.5 and 3.6). This order is based on the type of the multiple measurement index column in `data`. For '`integer`', '`character`' or '`factor`' the natural ordering is used (ascending, or alphabetical). If a different order of the multiple responses is desired, the multiple measurement index column should be an ordered factor with a corresponding ordering of the levels.

Formula

The multiple measurement object `MMO` including the ordinal responses `Y`, the subject index `i` and the multiple measurement index `j` is passed on the left-hand side of a '`formula`' object. The covariates `X1`, ..., `Xp` are passed on the right-hand side. In order to ensure identifiability intercepts can be included or excluded in the model depending on the chosen model parameterization.

Model without intercept. If the intercept should be removed, the `formula` can be specified in the following ways:

```
formula = MMO(Y, i, j) ~ 0 + X1 + ... + Xp
```

or

```
formula = MMO(Y, i, j) ~ -1 + X1 + ... + Xp
```

Model with intercept. If one wants to include an intercept in the model, there are two equivalent possibilities to specify the model formula. Either the intercept is included explicitly by:

```
formula = MMO(Y, i, j) ~ 1 + X1 + ... + Xp
```

or by

```
formula = MMO(Y, i, j) ~ X1 + ... + Xp
```

Note on the intercept in the formula. We differ in our approach of specifying the model formula from the model formula specification in, e.g., `MASS::polr()` or `ordinal::clm()`, in that we allow the user to specify models without an intercept. This option is not supported in the **MASS** and **ordinal** packages, where an intercept is always specified in the model formula as the threshold parameters are treated as intercepts. We choose to allow for this option, in order to have a correspondence to the identifiability constraints presented in Section 2.3.

Even so, the user should be aware that the threshold parameters are basically category- and outcome-specific intercepts. This implies that, even if the intercept is explicitly removed from the model through the '`formula`' object and hence set to zero, the rest of the covariates should be specified in such a way that multicollinearity does not arise. This is of primary importance when including categorical covariates, where one category will be taken as baseline by default.

3.2. Implementation MMO2

We use the same worked example as above to show the usage of `mvord()` with the multiple measurement object `MMO2`. The data set `data_mvord_toy` has already the required data structure with each response and all the covariates in separate columns. The multiple measurement object `MMO2` combines the different response columns on the left-hand side of the `formula` object:

```
R> res <- mvord(formula = MMO2(Y1, Y2) ~ 0 + X1 + X2, data = data_mvord_toy)
```

(runtime 1.68 seconds).

The multiple measurement object `MMO2` is only applicable for settings where the covariates do not vary between the multiple measurements.

Data structure

The data structure applied by `MMO2` is slightly simplified, where the multiple ordinal observations as well as the covariates are stored as columns in a '`data.frame`'. Each subject i corresponds to one row of the data frame, where all outcomes $Y_{i1}, \ldots, Y_{iq}$ (with missing observations set to `NA`) and all the covariates $x_{i1}, \ldots, x_{ip}$ are stored in different columns. Ideally each outcome column is of type ordered '`factor`'. If columns of the responses have types like '`integer`', '`character`' or '`factor`' a warning is displayed and the natural ordering is used (ascending, or alphabetical).

Formula

In order to specify the model we use a multivariate '`formula`' object of the form:

```
formula = MMO2(Y1, ..., Yq) ~ 0 + X1 + ... + Xp
```

The ordering of the responses is given by the ordering in the left-hand side of the model formula. `MMO2` performs like `cbind()` in fitting multivariate models in, e.g., `lm()` or `glm()`.

3.3. Link functions

The multivariate link functions are specified as objects of class '`mvlink`', which is a list with elements specifying the distribution function of the errors, functions for computing the corresponding univariate and bivariate probabilities, as well as additional arguments specific to each link. If gradient functions are passed on, these will be used in the computation of the standard errors. This design was inspired by the design of the '`family`' class in package **stats** and facilitates the addition of new link functions to the package.

We offer two different multivariate link functions, the multivariate probit link and a multivariate logit link. For the multivariate probit link a multivariate normal distribution for the errors is applied. The bivariate normal probabilities which enter the pairwise log-likelihood are computed with package **pbivnorm** (Genz and Kenkel 2015). The multivariate probit link is the default link function and can be specified by:

```
link = mvprobit()
```

For the multivariate logit link a t copula based multivariate distribution with logistic margins is used (as explained in Section 2.2) and can be specified by:

```
link = mvlogit(df = 8L)
```

The `mvlogit()` function has an optional integer valued argument `df` which specifies the degrees of freedom to be used for the t copula. The default value of the degrees of freedom parameter is 8. When choosing $\nu \approx 8$, the multivariate logistic distribution in Equation 2 is well approximated by a multivariate t distribution (O'Brien and Dunson 2004). This is also the value chosen by Nooraee *et al.* (2016) in their analysis. We restrict the degrees of freedom to be integer valued because the most efficient routines for computing bivariate t probabilities do not support non-integer degrees of freedom. We use the Fortran code from Alan Genz (Genz and Bretz 2009) to compute the bivariate t probabilities. As the degrees of freedom parameter is integer valued, we do not estimate it in the optimization procedure. If the optimal degrees of freedom are of interest, we leave the task of choosing an appropriate grid of values of `df` to the user, who could then estimate a separate model for each value in the grid. The best model can be chosen by CLAIC or CLBIC.

3.4. Error structures

Different error structures are implemented in **mvord** and can be specified through the argument `error.structure`. The error structure objects are of class '`error_struct`'. This approach slightly differs from the approach in package **nlme**, where the error structure is defined by two classes: '`varFunc`' for the variance function and '`corStruct`' for the correlation structure. We also define the following subclasses for the error structures: '`cor_general`' (similar to **nlme**'s '`corSymm`'), '`cor_equi`' (similar to '`corCompSymm`'), '`cor_ar1`' (similar to '`corAR1`') and '`cov_general`' (similar to '`corSymm`' with variance function '`varIdent`'). The different error structures are chosen through the argument `error.structure`.

Basic model

In the basic model we support three different correlation structures and one covariance structure.

Correlation. For the basic model specification the following correlation structures are implemented in **mvord**:

- `cor_general(formula = ~ 1)` – A general error structure, where the correlation matrix of the error terms is unrestricted and constant across all subjects: $\mathsf{COR}(\epsilon_{ik}, \epsilon_{il}) = \rho_{kl}$.
- `cor_equi(formula = ~ 1)` – An equicorrelation structure is used with $\mathsf{COR}(\epsilon_{ik}, \epsilon_{il}) = \rho$.
- `cor_ar1(formula = ~ 1)` – An autoregressive error structure of order one is used with $\mathsf{COR}(\epsilon_{ik}, \epsilon_{il}) = \rho^{|k-l|}$.

Variance. A model with variance parameters $\mathsf{VAR}(\epsilon_{ij}) = \sigma_j^2$ corresponding to each outcome, when the identifiability requirements are fulfilled, can be specified in the following way:

- `cov_general(formula = ~ 1)` – The estimation of σ_j^2 is only implemented in combination with the general correlation structure.

Extending the basic model

The basic model can be extended by allowing covariate dependent error structures.

Correlation. The following correlation structures are implemented in **mvord**:

- `cor_general(formula = ~ f)` – For the heterogeneous general correlation structure, the current implementation only allows the use of one '`factor`' variable `f` as covariate. As previously mentioned, this factor variable should be subject-specific and hence should not vary across the multiple responses. This implies that a correlation matrix will be estimated for each factor level.
- `cor_equi(formula = ~ S1 + ... + Sm)` – Estimating an equicorrelation structure depending on m subject-specific covariates `S1`, ..., `Sm`.
- `cor_ar1(formula = ~ S1 + ... + Sm)` – Estimating an $AR(1)$ correlation structure depending on m subject-specific covariates `S1`, ..., `Sm`.

Variance. The following variance structure is implemented in **mvord**:

- `cov_general(formula = ~ f)` – As in the basic model, the estimation of the heterogeneous variance parameters can be performed for the general covariance structure. A subject-specific '`factor`' `f` can be used as a covariate in the log-variance equation. In addition to the correlation matrices, which are estimated for each factor level of `f`, a vector of dimension q of variance parameters will be estimated for each factor level.

3.5. Constraints on thresholds

The package supports constraints on the threshold parameters. Firstly, the user can specify whether the threshold parameters should be equal across some or all response dimensions. Secondly, the values of some of the threshold parameters can be fixed. This feature is important for users who wish to further restrict the parameter space of the thresholds or who wish to specify values for the threshold parameters other than the default values used in the package. Note that some of the thresholds have to be fixed for some of the parameterizations presented in Table 2 in order to ensure identifiability of the model.

Threshold constraints across responses

Such constraints can be imposed by a vector of positive integers `threshold.constraints`, where dimensions with equal threshold parameters get the same integer. When restricting two outcome dimensions to be equal, one has to be careful that the number of categories

in the two outcome dimensions must be the same. In the worked example, if one wishes to restrict the threshold parameters of the two outcomes `Y1` and `Y2` to be equal ($\boldsymbol{\theta}_1 = \boldsymbol{\theta}_2$), this can be specified as:

```
threshold.constraints = c(1, 1)
```

where the first value corresponds to the first response `Y1` and the second to the second response `Y2`. This order of the responses is defined as explained in Sections 3.1 and 3.2

Fixing threshold values

Values for the threshold parameters can be specified by the argument `threshold.values`. For this purpose the user can pass a '`list`' with q elements, where each element is a '`vector`' of length $K_j - 1$ (where K_j is the number of ordered categories for ordinal outcome j). A numeric value in this vector fixes the corresponding threshold parameter to a specified value while `NA` leaves the parameter flexible and indicates it should be estimated.

After specifying the error structure (through the `error.structure` argument) and the inclusion/exclusion of an intercept in the `formula` argument, the user can choose among five possible options for fixing the thresholds:

- Leaving all thresholds flexible.
- Fixing the first threshold $\theta_{j,1}$ to a constant a_j for all $j \in J$.
- Fixing the first and second thresholds $\theta_{j,1} = a_j$, $\theta_{j,2} = b_j$ for all outcomes with $K_j > 2$.
- Fixing the first and last thresholds $\theta_{j,1} = a_j$, $\theta_{j,K_j-1} = b_j$ for all outcomes with $K_j > 2$.
- An extra option is fixing all of the threshold parameters, for all $j \in J$.

Note that the option chosen needs to be consistent across the different outcomes (e.g., it is not allowed to fix the first and the last threshold for one outcome and the first and the second threshold for a different outcome). Table 2 provides information about the options available for each combination of error structure and intercept, as well as about the default values in case the user does not specify any threshold values. In the presence of binary observations ($K_j = 2$), if a `cov_general` error structure is used, the intercept has always to be fixed to some value due to identifiability constraints. In a correlation structure setting no further restrictions are required.

For example, if the following restrictions should apply to the worked example:

- $\theta_{11} = -1 \leq \theta_{12}$,
- $\theta_{21} = -1 \leq \theta_{22}$,

this can be specified as:

```
threshold.values = list(c(-1, NA), c(-1, NA))
```

Error structure	Intercept	Threshold parameters				
		All flexible	One fixed $\theta_{j,1} = a_j$	Two fixed $\theta_{j,1} = a_j$ $\theta_{j,2} = b_j$	Two fixed $\theta_{j,1} = a_j$ $\theta_{j,K_j-1} = b_j$	All fixed
`cor`	no	✓	✓	✓	✓	✓
	yes		✓	✓	✓	✓
`cov`	no		✓	✓	✓	✓
	yes			✓	✓	✓

Table 2: This table displays different model parameterizations in the presence of ordinal observations ($K_j > 2\ \forall j \in J$). The row `cor` includes error structures `cor_general`, `cor_equi` and `cor_ar1`, while row `cov` includes the error structure `cov_general`. The minimal restrictions (default) to ensure identifiability are given in green. The default threshold values (in case `threshold.values = NULL`) are always $a_j = 0$ and $b_j = 1$.

3.6. Constraints on coefficients

The package supports constraints on the regression coefficients. Firstly, the user can specify whether the regression coefficients should be equal across some or all response dimensions. Secondly, values of some of the regression coefficients can be fixed.

As there is no unanimous way to specify such constraints, we offer two options. The first option is similar to the specification of constraints on the thresholds. The constraints can be specified in this case as a vector or matrix of integers, where coefficients getting the same integer value are set equal. Values of the regression coefficients can be fixed through a matrix. Alternatively, constraints on the regression coefficients can be specified by using the design employed by the **VGAM** package. The constraints in this setting are set through a named list, where each element of the list contains a matrix of full-column rank. If the values of some regression coefficients should be fixed, offsets can be used. This design has the advantage that it supports constraints on outcome-specific as well as category-specific regression coefficients. While the first option has the advantage of requiring a more concise input, it does not support category-specific coefficients. The second option offers a more flexible design in this respect.

Coefficient constraints across responses

Such constraints can be specified by the argument `coef.constraints`, which can be either a vector or a matrix of integer values. If vector constraints of the type $\boldsymbol{\beta}_k = \boldsymbol{\beta}_l$ are desired, which should hold for all regression coefficients corresponding to outcome k and l, the easiest way to specify this is by means of a vector of integers of dimension q, where outcomes with equal vectors of regression coefficients get the same integer.

Consider the following specification of the latent processes in the worked example:

$$\widetilde{Y}_{i1} = \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_{i1}, \quad \widetilde{Y}_{i2} = \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_{i2},$$

where the regression coefficients for variables `X1` and `X2` are set to be equal across the two outcomes ($\boldsymbol{\beta}_1 = \boldsymbol{\beta}_2$) by:

```
coef.constraints = c(1, 1)
```

A more flexible framework allows the user to specify constraints for each of the regression coefficients of the p covariates[3] and not only for the whole vector. Such constraints will be specified by means of a matrix of dimension $q \times p$, where each column specifies constraints for one of the p covariates in the same way as presented above. Moreover, a value of `NA` indicates that the corresponding coefficient is fixed (as we will show below) and should not be estimated.

Consider the following specification of the latent processes in the worked example:

$$\widetilde{Y}_{i1} = \beta_{11} x_{i1} + \beta_3 \mathbb{1}_{\{f_{i2}=c2\}} + \epsilon_{i1}, \quad \widetilde{Y}_{i2} = \beta_{21} x_{i1} + \beta_{22} x_{i2} + \beta_3 \mathbb{1}_{\{f_{i2}=c2\}} + \epsilon_{i2}, \tag{4}$$

where $\mathbb{1}_{\{f_{i2}=c2\}}$ is the indicator function which equals one in case the categorical covariate `f2` is equal to class `c2`. Class `c1` is taken as the baseline category. These restrictions on the regression coefficients are imposed by:

```
coef.constraints = cbind(c(1, 2), c(NA, 1), c(1, 1))
```

Specific values of coefficients can be fixed through the `coef.values` argument, as we will show in the following.

Fixing coefficient values

In addition, specific values on the regression coefficients can be set in the $q \times p$ matrix `coef.values`. Again each column corresponds to the regression coefficients of one covariate. This feature is to be used if some of the covariates have known slopes, but also for excluding covariates from the mean model of some of the outcomes (by fixing the regression coefficient to zero). Fixed coefficients are treated internally as offsets and are not displayed in the model output.

By default, if no `coef.values` are passed by the user, all the regression coefficients which receive an `NA` in `coef.constraints` will be set to zero. `NA` in the `coef.values` matrix indicates the regression coefficient ought to be estimated. Setting `coef.values` in accordance with the `coef.constraints` from above (not needed as this is the default case):

```
coef.values = cbind(c(NA, NA), c(0, NA), c(NA, NA))
```

Constraints on category-specific coefficients

If the parallel regression or proportional odds assumption ought to be relaxed, the constraint design of package **VGAM** can be employed. Let us consider the model specification in Equation 4. For illustration purposes we now relax the parallel regression assumption partially for covariates `X1` and `X2` in the following way:

- $\beta_{11,1} \neq \beta_{11,2}$,
- $\beta_{22,1} \neq \beta_{22,2}$,

[3]Note that if categorical covariates or interaction terms are included in the `formula`, p denotes the number of columns of the design matrix.

where $\beta_{jk,r}$ denotes the regression coefficient of covariate k in the linear predictor of the rth cumulative probit or logit for measurement j. By the first restriction, for the first outcome two regression coefficients are employed for covariate `X1`: $\beta_{11,1}$ for the first linear predictor and $\beta_{11,2}$ for the second linear predictor. Covariate `X2` only appears in the model for the second outcome. For each of the two linear predictors a different regression coefficient is estimated: $\beta_{22,1}$ and $\beta_{22,2}$.

The constraints are set up as a named list where the names correspond to the names of all covariates in the model matrix. To check the name of the covariates in the model matrix one can use the auxiliary function `names_constraints()` available in **mvord** (see also next subsection):

```
R> names_constraints(formula = Y ~ 0 + X1 + X2 + f2, data = data_mvord_toy)

[1] "X1"   "X2"   "f2c2"
```

The number of rows is equal to the total number of linear predictors $\sum_j (K_j - 1)$ of the ordered responses; in the example above $2 + 2 = 4$ rows. The number of columns represents the number of parameters to be estimated for each covariate:

```
coef.constraints = list(
  X1 = cbind(c(1, 0, 0, 0), c(0, 1, 0, 0),  c(0, 0, 1, 1)),
  X2 = cbind(c(0, 0, 1, 0), c(0, 0, 0, 1)), f2c2 = cbind(rep(1, 4)))
```

For more details we refer the reader to the documentation of the **VGAM** package.

Interaction terms and categorical covariates

When constraints on the regression coefficients should be specified in models with interaction terms or categorical covariates, the `coef.constraints` matrix has to be constructed appropriately. If the order of the terms in the covariate matrix is not clear to the user, it is helpful to call the function `names_constraints()` before constructing the `coef.constraints` and `coef.values` matrices. The names of each column in the covariate matrix can be obtained by:

```
R> formula <- MMO2(Y1, Y2) ~ 1 + X1 : X2 + f1 + f2 * X1
R> names_constraints(formula, data = data_mvord_toy)

[1] "(Intercept)" "f1B"         "f1C"         "f2c2"
[5] "X1"          "X1:X2"       "f2c2:X1"
```

This should be used when setting up the coefficient constraints. Please note that by default category `A` for factor `f1` and category `c1` for factor `f2` are taken as baseline categories. This can be changed by using the optional argument `contrasts`. In models without intercept, the estimated threshold parameters relate to the baseline category and the coefficients of the remaining factor levels can be interpreted as a shift of the thresholds.

3.7. Additional arguments

`weights.name`

Weights on each subject i are chosen in a way that they are constant across multiple measurements. Weights should be stored in a column of `data`. The column name of the weights in `data` should be passed as a character string to this argument by `weights.name = "weights"`. If `weights.name = NULL` all weights are set to one by default. Negative weights are not allowed.

`offset`

If offsets are not specified in the model `formula`, a list with a vector of offsets for each multiple measurement can be passed.

`contrasts`

`contrasts` can be specified by a named list as in the argument `contrasts.arg` of the default method of `model.matrix()`.

`PL.lag`

In longitudinal studies, where q_i is possibly large, the pairwise likelihood estimation can be time consuming as it is built from all two-dimensional combinations of $j, k \in J_i$. To overcome this difficulty, one can construct the likelihood using only the bivariate probabilities for pairs of observations less than *lag* in "time units" apart. A similar approach was proposed by Varin and Czado (2010). Assuming that, for each subject i, we have a time series of consecutive ordinal observations, the ith component of the pairwise likelihood has the following form:

$$p\ell_i^{lag}(\boldsymbol{\delta}) = w_i \left[\sum_{k=1}^{q_i-1} \sum_{l=k+1}^{q_i} \mathbb{1}_{\{|k-l| \leq lag\}} \log \mathsf{P}(Y_{ik} = r_{ik}, Y_{il} = r_{il}) \right].$$

The *lag* can be fixed by a positive integer argument `PL.lag` and it can only be used along with `error.structure = cor_ar1()`. The use of this argument is, however, not recommended if there are missing observations in the time series, i.e., if the ordinal variables are not observed in consecutive years. Moreover, one should also proceed with care if the observations are not missing at random.

3.8. Control function `mvord.control()`

Control arguments can be passed by the argument `control` and are hidden in the sub-function `mvord.control()` with the following arguments.

`solver`

This argument can either be a character string or a function. All general purpose optimizers of the R package **optimx** (Nash and Varadhan 2011; Nash 2014) can be used for maximization of the composite log-likelihood by passing the name of the solver as a character string to the `solver` argument. The available solvers in **optimx** are, at the time of writing, `"Nelder-Mead"`, `"BFGS"`, `"CG"`, `"L-BFGS-B"`, `"nlm"`, `"nlminb"`, `"spg"`, `"ucminf"`,

"newuoa", "bobyqa", "nmkb", "hjkb", "Rcgmin" and "Rvmmin". The default in **mvord** is solver "newuoa". The "BFGS" solver performs well in terms of computation time, but it suffers from convergence problems, especially for the mvlogit() link.

Alternatively, the user has the possibility of applying other solvers by using a wrapper function with arguments starting.values and objFun of the following form:

```
solver = function(starting.values, objFun) {
  optRes <- solver.function(...)
  list(optpar = optRes$optpar, objvalue = optRes$objvalue,
    convcode = optRes$convcode, message = optRes$message)
}
```

The solver.function() should return a list with the following three elements: optpar, objvalue and convcode. The element optpar should be a vector of length equal to the number of parameters to optimize containing the estimated parameters, while the element objvalue should contain the value of the objective function after the optimization procedure. The convergence status of the optimization procedure should be returned in element convcode with 0 indicating successful convergence. Moreover, an optional solver message can be returned in element message.

solver.optimx.control

A list of control arguments that are to be passed to the function optimx(). For further details see Nash and Varadhan (2011).

se

If se = TRUE standard errors are computed analytically using the Godambe information matrix (see Section 2.5).

start.values

A list of starting values for threshold as well as regression coefficients can be passed by the argument start.values. This list contains a list (with a vector of starting values for each dimension) theta of all flexible threshold parameters and a list beta of all flexible regression parameters.

3.9. Output and methods for class 'mvord'

The function mvord() returns an object of class 'mvord', which is a list containing the following components:

- beta: A named 'matrix' of regression coefficients.
- theta: A named 'list' of threshold parameters.
- error.struct: An object of class 'error_struct'.
- sebeta: A named 'matrix' of standard errors of the regression coefficients.

- `setheta`: A named '`list`' of standard errors of the threshold parameters.
- `seerror.struct`: A '`vector`' of standard errors for the parameters of the error structure.
- `rho`: A '`list`' of objects that are used in `mvord()`.

Several methods are implemented for the class '`mvord`'. These methods include a `summary()` and a `print()` function to display the estimation results, a `coef()` function to extract the regression coefficients, a `thresholds()` function to extract the threshold coefficients and a function `error_structure()` to extract the estimated parameters of the correlation/covariance structure of the errors. The pairwise log-likelihood can be extracted by the function `logLik()`, function `vcov()` extracts the variance-covariance matrix of the parameters and `nobs()` provides the number of subjects. Other standard methods such as `terms()` and `model.matrix()` are also available. Functions `AIC()` and `BIC()` can be used to extract the composite likelihood information criteria CLAIC and CLBIC.

In addition, joint probabilities can be extracted by the `predict()` or `fitted()` functions:

```
R> predict(res, subjectID =  1:6)
```

```
        1         2         3         4         5         6
0.9982776 0.2830394 0.9985192 1.0000000 0.8782797 0.9963333
```

as well as joint cumulative probabilities:

```
R> predict(res, type = "cum.prob", subjectID =  1:6)
```

```
        1         2         3         4         5         6
0.9982776 1.0000000 1.0000000 1.0000000 0.9745760 0.9963333
```

and classes:

```
R> predict(res, type = "class", subjectID =  1:6)
```

```
  Y1 Y2
1  1  1
2  2  2
3  3  3
4  1  1
5  2  2
6  1  1
```

The function `marginal_predict()` provides marginal predictions for the types probability, cumulative probability and class, while `joint_probabilities()` extracts fitted joint probabilities (or cumulative probabilities) for given response categories from a fitted model.

4. Examples

In credit risk applications, multiple credit ratings from different credit rating agencies are available for a panel of firms. Such a data set has been analyzed in Hirk *et al.* (2019), where a multivariate model of corporate credit ratings has been proposed. Unfortunately, this original data set is not freely re-distributable. Therefore, we resort to the simulation of illustrative data sets by taking into consideration key features of the original data.

We simulate relevant covariates corresponding to firm-level and market financial ratios in the original data set. The following covariates are chosen in line with literature on determinants of credit ratings (e.g., Campbell, Hilscher, and Szilagyi 2008; Puccia, Collett, Kernan, Palmer, Mettrick, and Deslondes 2013): `LR` (liquidity ratio relating the current assets to current liabilities), `LEV` (leverage ratio relating debt to earnings before interest and taxes), `PR` (profitability ratio of retained earnings to assets), `RSIZE` (logarithm of the relative size of the company in the market), `BETA` (a measure of systematic risk). We fit a distribution to each covariate using the function `fitdistr()` of the **MASS** package. The best fitting distribution among all available distributions in `fitdistr()` has been chosen by AIC.

We generate two data sets for illustration purposes. The first data set `data_cr` consists of multiple ordinal rating observations at the same point in time for a collection of 690 firms. We generate ratings from four rating sources `rater1`, `rater2`, `rater3` and `rater4`. Raters `rater1` and `rater2` assign ordinal ratings on a five-point scale (from best to worst `A`, `B`, `C`, `D` and `E`), `rater3` on a six-point scale (from best to worst, `F`, `G`, `H`, `I`, `J` and `K`) and `rater4` distinguishes between investment and speculative grade firms (from best to worst, `L` and `M`). The panel of ratings in the original data set is unbalanced, as not all firms receive ratings from all four sources. We therefore keep the missingness pattern and remove the simulated ratings that correspond to missing observations in the original data set. For `rater1` we remove 5%, for `rater2` 30%, and for `rater3` 50% of the observations. This data set has a wide data format.

The second data set `data_cr_panel` contains ordinal rating observations assigned by one rater to a panel of 1415 firms over a period of eight years on a yearly basis. In addition to the covariates described above, a business sector variable (`BSEC`) with eight levels is included for each firm. This data set has a long format, with 11320 firm-year observations.

4.1. Example 1: A simple model of firm ratings assigned by multiple raters

The first example presents a multivariate ordinal regression model with probit link and a general correlation error structure `cor_general(~ 1)`. The simulated data set contains the ratings assigned by raters `rater1`, `rater2`, `rater3` and `rater4` and the five covariates `LR`, `LEV`, `PR`, `RSIZE` and `BETA` for a cross-section of 690 firms. A value of `NA` indicates a missing observation in the corresponding outcome variable.

```
R> data("data_cr", package = "mvord")
R> head(data_cr, n = 3)
```

```
  rater1 rater2 rater3 rater4 firm_id       LR       LEV          PR
1      B      B      H      L       1 1.720041 2.1144513  0.37792213
2      C      D   <NA>      M       2 1.836574 0.8826725 -0.15032402
3      C      D   <NA>      M       3 2.638177 2.2997237 -0.05205389
```

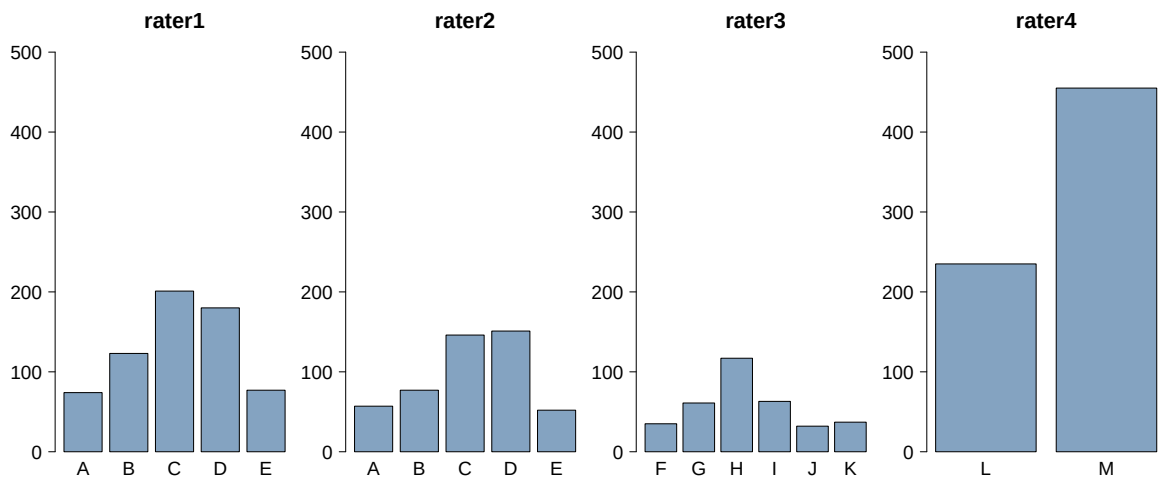

Figure 1: This figure displays the rating distribution of all the raters.

```
      RSIZE      BETA
1 -6.365053 0.8358773
2 -7.839813 0.4895358
3 -7.976650 0.8022900
```

```
R> str(data_cr, vec.len = 2.9)
```

```
'data.frame':        690 obs. of  10 variables:
 $ rater1 : Ord.factor w/ 5 levels "A"<"B"<"C"<"D"<..: 2 3 3 2 5 4 3 ...
 $ rater2 : Ord.factor w/ 5 levels "A"<"B"<"C"<"D"<..: 2 4 4 2 5 NA 3 ...
 $ rater3 : Ord.factor w/ 6 levels "F"<"G"<"H"<"I"<..: 3 NA NA NA 6 NA 2 ...
 $ rater4 : Ord.factor w/ 2 levels "L"<"M": 1 2 2 1 2 2 2 ...
 $ firm_id: int  1 2 3 4 5 6 7 ...
 $ LR     : num  1.72 1.84 2.64 1.31 ...
 $ LEV    : num  2.114 0.883 2.3 2.638 ...
 $ PR     : num  0.3779 -0.1503 -0.0521 0.3289 ...
 $ RSIZE  : num  -6.37 -7.84 -7.98 -5.86 ...
 $ BETA   : num  0.836 0.49 0.802 1.137 ...
```

We include five financial ratios as covariates in the model without intercept through the following `formula`:

```
formula = MMO2(rater1, rater2, rater3, rater4) ~ 0 + LR + LEV + PR + RSIZE +
  BETA
```

We are dealing with a wide data format, as the covariates do not vary among raters. Hence, the estimation can be performed by applying multiple measurement object `MMO2` in the fitting function `mvord()`. A model with multivariate probit link (default) is fitted by:

```
R> res_cor_probit_simple <- mvord(formula = MMO2(rater1, rater2, rater3,
+    rater4) ~ 0 + LR + LEV + PR + RSIZE + BETA, data = data_cr)
```

(runtime 4 minutes).

The results are displayed by the function `summary()`:

```
R> summary(res_cor_probit_simple, call = FALSE)

Formula: MMO2(rater1, rater2, rater3, rater4) ~ 0 + LR + LEV + PR + RSIZE +
    BETA

    link threshold nsubjects ndim    logPL   CLAIC   CLBIC fevals
mvprobit  flexible       690    4 -2925.83 6037.38 6458.64  15701

Thresholds:
           Estimate Std. Error z value  Pr(>|z|)
rater1 A|B  8.04920    0.44306  18.167 < 2.2e-16 ***
rater1 B|C  9.56783    0.47381  20.194 < 2.2e-16 ***
rater1 C|D 11.34991    0.51747  21.933 < 2.2e-16 ***
rater1 D|E 13.51638    0.60129  22.479 < 2.2e-16 ***
rater2 A|B  8.59513    0.49834  17.247 < 2.2e-16 ***
rater2 B|C 10.05454    0.53941  18.640 < 2.2e-16 ***
rater2 C|D 11.85857    0.59749  19.847 < 2.2e-16 ***
rater2 D|E 14.33336    0.70092  20.450 < 2.2e-16 ***
rater3 F|G  8.24303    0.51775  15.921 < 2.2e-16 ***
rater3 G|H  9.77659    0.55593  17.586 < 2.2e-16 ***
rater3 H|I 11.70601    0.62343  18.777 < 2.2e-16 ***
rater3 I|J 13.09380    0.68810  19.029 < 2.2e-16 ***
rater3 J|K 14.17279    0.72153  19.643 < 2.2e-16 ***
rater4 L|M 13.52691    1.00085  13.515 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Coefficients:
         Estimate Std. Error  z value  Pr(>|z|)
LR 1     0.208652   0.068020   3.0675  0.002159 **
LR 2     0.153212   0.073420   2.0868  0.036906 *
LR 3     0.180531   0.078643   2.2956  0.021700 *
LR 4     0.150887   0.111750   1.3502  0.176944
LEV 1    0.430213   0.043774   9.8281 < 2.2e-16 ***
LEV 2    0.432842   0.050129   8.6346 < 2.2e-16 ***
LEV 3    0.399459   0.050826   7.8594 3.861e-15 ***
LEV 4    0.624967   0.074146   8.4289 < 2.2e-16 ***
PR 1    -2.573616   0.194046 -13.2629 < 2.2e-16 ***
PR 2    -2.827056   0.217061 -13.0242 < 2.2e-16 ***
PR 3    -2.678170   0.222705 -12.0257 < 2.2e-16 ***
PR 4    -2.793063   0.279732  -9.9848 < 2.2e-16 ***
RSIZE 1 -1.129967   0.056379 -20.0424 < 2.2e-16 ***
RSIZE 2 -1.196432   0.061776 -19.3671 < 2.2e-16 ***
RSIZE 3 -1.196481   0.066503 -17.9914 < 2.2e-16 ***
```

```
RSIZE 4 -1.565903   0.115749 -13.5285 < 2.2e-16 ***
BETA 1   1.602187   0.110868  14.4513 < 2.2e-16 ***
BETA 2   1.801526   0.140123  12.8567 < 2.2e-16 ***
BETA 3   1.517633   0.139319  10.8932 < 2.2e-16 ***
BETA 4   1.988644   0.203338   9.7800 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error Structure:
                   Estimate Std. Error z value  Pr(>|z|)
corr rater1 rater2 0.875996   0.024554  35.677 < 2.2e-16 ***
corr rater1 rater3 0.913588   0.023462  38.940 < 2.2e-16 ***
corr rater1 rater4 0.896170   0.033495  26.755 < 2.2e-16 ***
corr rater2 rater3 0.831679   0.042740  19.459 < 2.2e-16 ***
corr rater2 rater4 0.926139   0.031808  29.117 < 2.2e-16 ***
corr rater3 rater4 0.866521   0.051285  16.896 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The threshold parameters are labeled with the name of the corresponding outcome and the two adjacent categories which are separated by a vertical bar `|`. For each covariate the estimated coefficients are labeled with the covariate name and a number. This number is from the sequence along the number of columns in the list element of `constraints()` which corresponds to the covariate. Note that if no constraints are set on the regression coefficients, this number of the coefficient corresponds to the outcome dimension. If constraints are set on the parameter space, we refer the reader to Section 4.2. The last part of the summary contains the estimated error structure parameters. For error structures `cor_general` and `cov_general` the correlations (and variances) are displayed. The coefficients corresponding to the error structure are displayed for `cor_ar1` and `cor_equi`. Correlations and Fisher-z scores for each subject are obtained by function `error_structure()`.

Another option to display the results is the function `print()`. The threshold coefficients can be extracted by the function `thresholds()`:

```
R> thresholds(res_cor_probit_simple)

$rater1
      A|B       B|C       C|D       D|E
 8.049203  9.567831 11.349908 13.516375

$rater2
      A|B       B|C       C|D       D|E
 8.595126 10.054536 11.858567 14.333362

$rater3
      F|G       G|H       H|I       I|J       J|K
 8.243027  9.776591 11.706012 13.093799 14.172787
```

```
$rater4
     L|M
13.52691
```

The regression coefficients are obtained by the function `coef()`:

```
R> coef(res_cor_probit_simple)

      LR 1       LR 2       LR 3       LR 4      LEV 1      LEV 2
 0.2086517  0.1532119  0.1805309  0.1508872  0.4302126  0.4328417
     LEV 3      LEV 4       PR 1       PR 2       PR 3       PR 4
 0.3994591  0.6249669 -2.5736158 -2.8270564 -2.6781697 -2.7930627
   RSIZE 1    RSIZE 2    RSIZE 3    RSIZE 4     BETA 1     BETA 2
-1.1299673 -1.1964316 -1.1964814 -1.5659034  1.6021871  1.8015260
    BETA 3     BETA 4
 1.5176327  1.9886441
```

The error structure for firm with `firm_id = 11` is displayed by `error_structure()`:

```
R> error_structure(res_cor_probit_simple)[[11]]

          rater1    rater2    rater3    rater4
rater1 1.0000000 0.8759965 0.9135876 0.8961703
rater2 0.8759965 1.0000000 0.8316790 0.9261393
rater3 0.9135876 0.8316790 1.0000000 0.8665213
rater4 0.8961703 0.9261393 0.8665213 1.0000000
```

4.2. Example 2: A more elaborate model of ratings by multiple raters

In the second example, we extend the setting of Example 1 by imposing constraints on the regression as well as on the threshold parameters and changing the link function to the multivariate logit link. We include the following features in the model:

- We assume that `rater1` and `rater2` use the same rating methodology. This means that they use the same rating classes with the same labeling and the same thresholds on the latent scale. Hence, we set the following constraints on the threshold parameters:

  ```
  threshold.constraints = c(1, 1, 2, 3)
  ```

- We assume that some covariates are equal for some of the raters. We assume that the coefficients of `LR` and `PR` are equal for all four raters, that the coefficients of `RSIZE` are equal for the raters `rater1`, `rater2` and `rater3` and the coefficients of `BETA` are the same for the raters `rater1` and `rater2`. The coefficients of `LEV` are assumed to vary for all four raters. These restrictions are imposed by:

  ```
  coef.constraints = cbind(LR = c(1, 1, 1, 1), LEV = c(1, 2, 3, 4),
    PR = c(1, 1, 1, 1), RSIZE = c(1, 1, 1, 2), BETA = c(1, 1, 2, 3))
  ```

The estimation can now be performed by the function `mvord()`:

```
R> res_cor_logit <- mvord(formula = MMO2(rater1, rater2, rater3, rater4) ~
+    0 + LR + LEV + PR + RSIZE + BETA, data = data_cr, link = mvlogit(),
+    coef.constraints = cbind(LR = c(1, 1, 1, 1), LEV = c(1, 2, 3, 4),
+      PR = c(1, 1, 1, 1), RSIZE = c(1, 1, 1, 2), BETA = c(1, 1, 2, 3)),
+    threshold.constraints = c(1, 1, 2, 3))
```

(runtime 6 minutes).
The results are displayed by the function `summary()`:

```
R> summary(res_cor_logit, call = FALSE)

Formula: MMO2(rater1, rater2, rater3, rater4) ~ 0 + LR + LEV + PR + RSIZE +
    BETA

   link threshold nsubjects ndim    logPL   CLAIC   CLBIC fevals
mvlogit  flexible       690    4 -2926.42 5987.81 6293.98   9338

Thresholds:
           Estimate Std. Error z value  Pr(>|z|)
rater1 A|B 15.04918    0.82409  18.262 < 2.2e-16 ***
rater1 B|C 17.75218    0.89728  19.785 < 2.2e-16 ***
rater1 C|D 20.97821    1.00773  20.817 < 2.2e-16 ***
rater1 D|E 25.13047    1.17487  21.390 < 2.2e-16 ***
rater3 F|G 14.47061    0.83922  17.243 < 2.2e-16 ***
rater3 G|H 17.17327    0.89515  19.185 < 2.2e-16 ***
rater3 H|I 20.56634    1.01119  20.339 < 2.2e-16 ***
rater3 I|J 23.00523    1.11045  20.717 < 2.2e-16 ***
rater3 J|K 24.97258    1.18725  21.034 < 2.2e-16 ***
rater4 L|M 23.92770    1.63196  14.662 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Coefficients:
         Estimate Std. Error  z value  Pr(>|z|)
LR 1     0.340210   0.110547   3.0775  0.002087 **
LEV 1    0.784294   0.075977  10.3228 < 2.2e-16 ***
LEV 2    0.779694   0.078364   9.9496 < 2.2e-16 ***
LEV 3    0.718329   0.093425   7.6888 1.485e-14 ***
LEV 4    1.107836   0.123681   8.9572 < 2.2e-16 ***
PR 1    -4.917963   0.343464 -14.3187 < 2.2e-16 ***
RSIZE 1 -2.093378   0.103690 -20.1889 < 2.2e-16 ***
RSIZE 2 -2.746163   0.188731 -14.5507 < 2.2e-16 ***
BETA 1   3.135692   0.221944  14.1283 < 2.2e-16 ***
BETA 2   2.733088   0.252960  10.8044 < 2.2e-16 ***
BETA 3   3.572691   0.349493  10.2225 < 2.2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error Structure:
                   Estimate Std. Error z value  Pr(>|z|)
corr rater1 rater2 0.859776   0.027907  30.809 < 2.2e-16 ***
corr rater1 rater3 0.908834   0.024636  36.891 < 2.2e-16 ***
corr rater1 rater4 0.903956   0.031858  28.374 < 2.2e-16 ***
corr rater2 rater3 0.834904   0.044259  18.864 < 2.2e-16 ***
corr rater2 rater4 0.932242   0.032173  28.976 < 2.2e-16 ***
corr rater3 rater4 0.856234   0.058392  14.664 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If constraints on the threshold or regression coefficients are imposed, duplicated estimates are not displayed. If thresholds are set equal for two outcome dimensions only the thresholds for the former dimension are shown. In the example above only the thresholds for `rater1` are displayed. For each covariate the estimated coefficients are labeled with the covariate name and a number. This number is from a sequence along the number of columns in the list element of the corresponding covariate in `constraints()` (see Section 3.6). The auxiliary function `constraints()` can be used to extract the constraints on the coefficients. The column names of the constraint matrices for each outcome correspond to the coefficient names displayed in the `summary`. For each covariate the coefficients to be estimated are numbered consecutively. In the above example this means that for covariates `LR` and `PR` only one covariate is estimated, a coefficient for each outcome is estimated for `LEV`, while for covariate `RSIZE` two and for covariate `BETA` three coefficients are estimated. For example, the coefficient `BETA 1` is used by `rater1` and `rater2`, the coefficient `BETA 2` is used by `rater3` while `BETA 3` is the coefficient for `rater4`. The constraints for covariate `BETA` can be extracted by:

```
R> constraints(res_cor_logit)$BETA
```

```
    BETA 1 BETA 2 BETA 3
A|B      1      0      0
B|C      1      0      0
C|D      1      0      0
D|E      1      0      0
A|B      1      0      0
B|C      1      0      0
C|D      1      0      0
D|E      1      0      0
F|G      0      1      0
G|H      0      1      0
H|I      0      1      0
I|J      0      1      0
J|K      0      1      0
L|M      0      0      1
```

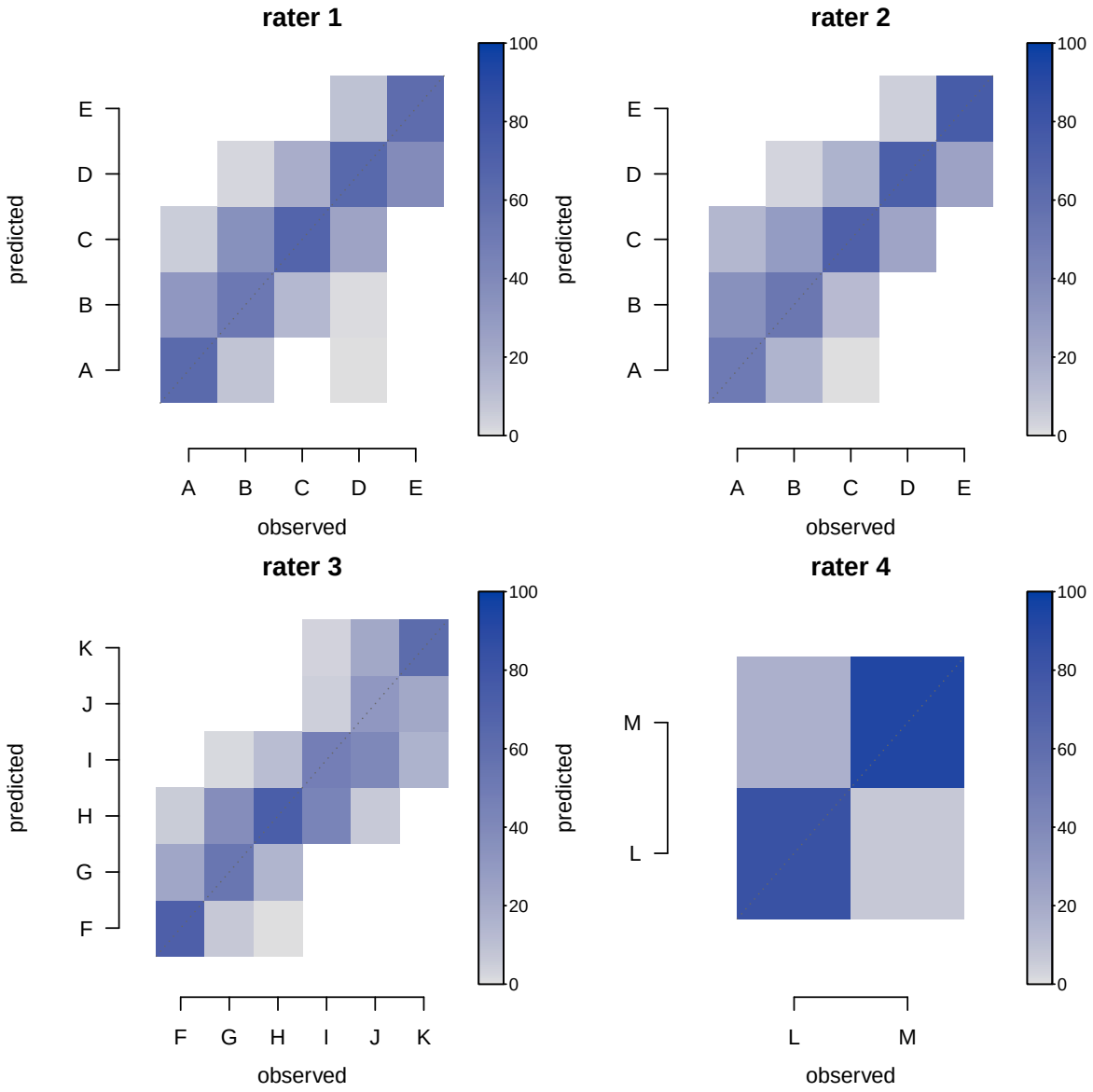

Figure 2: This figure displays agreement plots of the predicted categories of the model `res_cor_logit` against the observed rating categories for all raters. For each observed rating class the distribution of the predicted ratings is displayed.

Comparing the model fits of examples one and two

Note that the composite likelihood information criteria can be used for model comparison. For objects of class '`mvord`' CLAIC and CLBIC are computed by `AIC()` and `BIC()`, respectively. The value of the pairwise log-likelihood of the two models can be extracted by `logLik()`. The model fit of examples one and two are compared by means of BIC and AIC. From Table 3 we observe that the model of Example 2 has a lower BIC and AIC, which indicates a better model fit.

	logLik()	BIC()	AIC()
Example 1	−2925.83	6458.64	6037.38
Example 2	−2926.42	6293.98	5987.81

Table 3: This table displays measures of fit for the multivariate probit model in Example 1 (presented in Section 4.1) and the multivariate logit model in Example 2 (presented in Section 4.2).

4.3. Example 3: Ratings assigned by one rater to a panel of firms

In the third example, we present a longitudinal multivariate ordinal probit regression model with a covariate dependent $AR(1)$ error structure using the data set `data_cr_panel`:

```
R> data("data_cr_panel")
R> str(data_cr_panel, vec.len = 3)

'data.frame':        11320 obs. of  9 variables:
 $ rating : Ord.factor w/ 5 levels "A"<"B"<"C"<"D"<..: 5 3 3 3 3 1 3 3 ...
 $ firm_id: int  1 2 3 4 5 6 7 8 ...
 $ year   : Factor w/ 8 levels "year1","year2",..: 1 1 1 1 1 1 1 1 ...
 $ LR     : num  572.86 1.38 7.46 10.9 ...
 $ LEV    : num  1.2008 0.0302 0.1517 0.5485 ...
 $ PR     : num  0.1459 -0.0396 0.0508 0.1889 ...
 $ RSIZE  : num  1.423 -1.944 2.024 -0.433 ...
 $ BETA   : num  1.148 1.693 0.761 2.24 ...
 $ BSEC   : Factor w/ 8 levels "BSEC1","BSEC2",..: 3 6 3 7 6 7 7 7 ...

R> head(data_cr_panel, n = 3)

  rating firm_id  year         LR        LEV          PR     RSIZE
1      E       1 year1 572.864658 1.20084294  0.14585117  1.422948
2      C       2 year1   1.379547 0.03022761 -0.03962597 -1.944265
3      C       3 year1   7.462706 0.15170420  0.05083517  2.024098
       BETA  BSEC
1 1.1481020 BSEC3
2 1.6926956 BSEC6
3 0.7610057 BSEC3
```

The simulated data set has a long data format and contains the credit risk measure `rating` and six covariates for a panel of 1415 firms over eight years. The number of firm-year observations is 11320.

We include five financial ratios as covariates in the model with an intercept by a `formula` with multiple measurement object `MMO`:

```
formula = MMO(rating, firm_id, year) ~ LR + LEV + PR + RSIZE + BETA
```

Additionally, the model has the following features:

- The threshold parameters are constant over the years. This can be specified through the argument `threshold.constraints`:

  ```
  threshold.constraints = rep(1, nlevels(data_cr_panel$year))
  ```

- In order to ensure identifiability in a model with intercept, some thresholds need to be fixed. We fix the first thresholds for all outcome dimensions to zero by the argument `threshold.values`:

  ```
  threshold.values = rep(list(c(0, NA, NA, NA)), 8)
  ```

- We assume that there is a break-point in the regression coefficients after `year4` in the sample. This break-point could correspond to the beginning of a crisis in a real case application. Hence, we use one set of regression coefficients for years `year1`, `year2`, `year3` and `year4` and a different set for `year5`, `year6`, `year7` and `year8`. This can be specified through the argument `coef.constraints`:

  ```
  coef.constraints = c(rep(1, 4),  rep(2, 4))
  ```

- Given the longitudinal aspect of the data, an $AR(1)$ correlation structure is an appropriate choice. Moreover, we use the business sector as a covariate in the correlation structure. The dependence of the correlation structure on the business sector is motivated by the fact that in some sectors, such as manufacturing, ratings tend to be more "sticky", i.e., do not change often over the years, while in more volatile sectors like IT there is less "stickiness" in the ratings.

  ```
  error.structure = cor_ar1(~ BSEC)
  ```

The estimation is performed by calling the function `mvord()`:

```
R> res_AR1_probit <- mvord(formula = MMO(rating, firm_id, year) ~ LR + LEV +
+    PR + RSIZE + BETA, error.structure = cor_ar1(~ BSEC), link = mvprobit(),
+    data = data_cr_panel, coef.constraints = c(rep(1, 4), rep(2, 4)),
+    threshold.constraints = rep(1, 8), threshold.values = rep(list(c(0, NA,
+      NA, NA)),8), control = mvord.control(solver = "BFGS"))
```

(runtime 8 minutes).
The results of the model can be presented by the function `summary()`:

```
R> summary(res_AR1_probit, short = TRUE, call = FALSE)

Formula: MMO(rating, firm_id, year) ~ LR + LEV + PR + RSIZE + BETA

    link threshold nsubjects ndim     logPL     CLAIC     CLBIC fevals
mvprobit fix1first      1415    8 -74805.56 150098.19 151377.93    181

Thresholds:
          Estimate Std. Error z value  Pr(>|z|)
year1 A|B 0.000000   0.000000      NA        NA
```

```
year1 B|C 1.016137   0.026379  38.521 < 2.2e-16 ***
year1 C|D 2.444887   0.041170  59.385 < 2.2e-16 ***
year1 D|E 3.891073   0.058843  66.127 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Coefficients:
               Estimate Std. Error  z value  Pr(>|z|)
(Intercept) 1  1.4190682  0.0532243  26.6620 < 2.2e-16 ***
(Intercept) 2  1.4975500  0.0494038  30.3124 < 2.2e-16 ***
LR 1           0.0193546  0.0006938  27.8966 < 2.2e-16 ***
LR 2           0.0323230  0.0010102  31.9955 < 2.2e-16 ***
LEV 1          0.0268125  0.0019263  13.9191 < 2.2e-16 ***
LEV 2          0.0180050  0.0013143  13.6993 < 2.2e-16 ***
PR 1          -1.0930080  0.0427073 -25.5930 < 2.2e-16 ***
PR 2          -0.7411681  0.0372295 -19.9081 < 2.2e-16 ***
RSIZE 1       -0.3676665  0.0114038 -32.2407 < 2.2e-16 ***
RSIZE 2       -0.3608979  0.0115297 -31.3016 < 2.2e-16 ***
BETA 1         0.0541624  0.0296028   1.8296    0.0673 .
BETA 2         0.1099998  0.0236754   4.6462 3.382e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error Structure:
             Estimate Std. Error  z value  Pr(>|z|)
(Intercept)  1.490395   0.051540  28.9173 < 2.2e-16 ***
BSECBSEC2   -0.552317   0.069241  -7.9767 1.503e-15 ***
BSECBSEC3   -0.062028   0.066553  -0.9320    0.3513
BSECBSEC4   -0.085592   0.065126  -1.3142    0.1888
BSECBSEC5   -0.066598   0.085692  -0.7772    0.4371
BSECBSEC6   -0.683429   0.069184  -9.8785 < 2.2e-16 ***
BSECBSEC7   -0.863911   0.064855 -13.3206 < 2.2e-16 ***
BSECBSEC8   -0.757997   0.078506  -9.6553 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the fixed threshold coefficient `year1 A|B`, the z values and the corresponding p values are set to `NA`.

The default `error_structure()` method for a '`cor_ar1`' gives:

```
R> error_structure(res_AR1_probit)

        V16         V17         V18         V19         V20
 1.49039489 -0.55231725 -0.06202790 -0.08559155 -0.06659849
        V21         V22         V23
-0.68342920 -0.86391149 -0.75799729
```

In addition, the correlation parameters ρ_i for each firm are obtained by choosing `type = "corr"` in `error_structure()`:

```
R> head(error_structure(res_AR1_probit, type = "corr"), n = 3)

  Correlation
1   0.8913315
2   0.6679130
3   0.8913315
```

Moreover, the correlation matrices for each specific firm are obtained by choosing `type = "sigmas"` in `error_structure()`:

```
R> head(error_structure(res_AR1_probit, type = "sigmas"), n = 1)

$`1`
          year1     year2     year3     year4     year5     year6
year1 1.0000000 0.8913315 0.7944718 0.7081377 0.6311854 0.5625954
year2 0.8913315 1.0000000 0.8913315 0.7944718 0.7081377 0.6311854
year3 0.7944718 0.8913315 1.0000000 0.8913315 0.7944718 0.7081377
year4 0.7081377 0.7944718 0.8913315 1.0000000 0.8913315 0.7944718
year5 0.6311854 0.7081377 0.7944718 0.8913315 1.0000000 0.8913315
year6 0.5625954 0.6311854 0.7081377 0.7944718 0.8913315 1.0000000
year7 0.5014590 0.5625954 0.6311854 0.7081377 0.7944718 0.8913315
year8 0.4469662 0.5014590 0.5625954 0.6311854 0.7081377 0.7944718
          year7     year8
year1 0.5014590 0.4469662
year2 0.5625954 0.5014590
year3 0.6311854 0.5625954
year4 0.7081377 0.6311854
year5 0.7944718 0.7081377
year6 0.8913315 0.7944718
year7 1.0000000 0.8913315
year8 0.8913315 1.0000000
```

5. Conclusion

The present paper is meant to provide a general overview on the R package **mvord**, which implements the estimation of multivariate ordinal probit and logit regression models using the pairwise likelihood approach. We offer the following features which (to the best of our knowledge) enhance the currently available software for multivariate ordinal regression models in R:

- Different error structures like a general correlation and a covariance structure, an equicorrelation structure and an $AR(1)$ structure are available.

- We account for heterogeneity in the error structure among the subjects by allowing the use of subject-specific covariates in the specification of the error structure.
- We allow for outcome-specific threshold parameters.
- We allow for outcome-specific regression parameters.
- The user can impose further restrictions on the threshold and regression parameters in order to achieve a more parsimonious model (e.g., using one set of thresholds for all outcomes).
- We offer the possibility to choose different parameterizations, which are needed in ordinal models to ensure identifiability.

Additional flexibility is achieved by allowing the user to implement alternative multivariate link functions or error structures (e.g., alternative transformations for the variance or correlation parameters can be implemented). Furthermore, the long as well as the wide data format are supported by either applying `MMO` or `MMO2` as a multiple measurement object to estimate the model parameters. The functionality of the package is illustrated by a credit risk application. Further examples from different areas of application are presented in the package vignette.

Further research and possible extensions of **mvord** could consist of the implementation of variable selection procedures in multivariate ordinal regression models and the inclusion of multivariate semi- or non-parametric ordinal models.

References

Afonso A, Gomes P, Rother P (2009). "Ordered Response Models for Sovereign Debt Ratings." *Applied Economics Letters*, **16**(8), 769–773. doi:10.1080/13504850701221931.

Agresti A (2002). *Categorical Data Analysis*. 2nd edition. John Wiley & Sons.

Agresti A (2010). *Analysis of Ordinal Categorical Data*. 2nd edition. John Wiley & Sons. doi:10.1002/9780470594001.

Alp A (2013). "Structural Shifts in Credit Rating Standards." *The Journal of Finance*, **68**(6), 2435–2470. doi:10.1111/jofi.12070.

Archer KJ, Hou J, Zhou Q, Ferber K, Layne JG, Gentry AE (2014). "**ordinalgmifs**: An R Package for Ordinal Regression in High-Dimensional Data Settings." *Cancer Informatics*, **13**, 187–195. doi:10.4137/cin.s20806.

Bhat CR, Varin C, Ferdous N (2010). "A Comparison of the Maximum Simulated Likelihood and Composite Marginal Likelihood Estimation Approaches in the Context of the Multivariate Ordered-Response Model." In W Greene, R Carter Hill (eds.), *Maximum Simulated Likelihood Methods and Applications*, volume 26 of *Advances in Econometrics*, pp. 65–106. Emerald Group Publishing Limited. doi:10.1108/s0731-9053(2010)0000026007.

Blume ME, Lim F, Mackinlay AC (1998). "The Declining Credit Quality of U.S. Corporate Debt: Myth or Reality?" *The Journal of Finance*, **53**(4), 1389–1413. doi:10.1111/0022-1082.00057.

Boes S, Winkelmann R (2006). "Ordered Response Models." *AStA Advances in Statistical Analysis*, **90**(1), 167–181. doi:10.1007/s10182-006-0228-y.

Bürkner PC (2017). "**brms**: An R Package for Bayesian Multilevel Models Using Stan." *Journal of Statistical Software*, **80**(1), 1–28. doi:10.18637/jss.v080.i01.

Campbell JY, Hilscher J, Szilagyi J (2008). "In Search of Distress Risk." *The Journal of Finance*, **63**(6), 2899–2939. doi:10.1111/j.1540-6261.2008.01416.x.

Carroll N (2018). ***oglmx**: Estimation of Ordered Generalized Linear Models*. R package version 3.0.0.0, URL https://CRAN.R-project.org/package=oglmx.

Christensen RHB (2019a). *Cumulative Link Models for Ordinal Regression with the R Package **ordinal***. R package vignette version 2019.12-10, URL https://CRAN.R-project.org/package=ordinal.

Christensen RHB (2019b). "**ordinal**: Regression Models for Ordinal Data." R package version 2019.12-10, URL https://CRAN.R-project.org/package=ordinal.

DeYoreo M, Kottas A (2018). "Bayesian Nonparametric Modeling for Multivariate Ordinal Regression." *Journal of Computational and Graphical Statistics*, **27**(1), 71–84. doi:10.1080/10618600.2017.1316280.

Fahrmeir L, Tutz G (2001). *Multivariate Statistical Modelling Based on Generalized Linear Models*. 2nd edition. Springer-Verlag.

Fox J (2019). ***polycor**: Polychoric and Polyserial Correlations*. R package version 0.7-10, URL https://CRAN.R-project.org/package=polycor.

Genz A, Bretz F (2009). *Computation of Multivariate Normal and t Probabilities*. Springer-Verlag.

Genz A, Kenkel B (2015). ***pbivnorm**: Vectorized Bivariate Normal CDF*. R package version 0.6.0, URL https://CRAN.R-project.org/package=pbivnorm.

Greene WH, Hensher DA (2010). *Modeling Ordered Choices: A Primer*. Cambridge University Press.

Gumbel EJ (1961). "Bivariate Logistic Distributions." *Journal of the American Statistical Association*, **56**(294), 335–349. doi:10.1080/01621459.1961.10482117.

Harrell Jr FE (2019). ***rms**: Regression Modeling Strategies*. R package version 5.1-4, URL https://CRAN.R-project.org/package=rms.

Hedeker D, Archer KJ, Nordgren R, Gibbons RD (2018). ***mixor**: Mixed-Effects Ordinal Regression Analysis*. R package version 1.0.4, URL https://CRAN.R-project.org/package=mixor.

Higham NJ (1988). "Computing a Nearest Symmetric Positive Semidefinite Matrix." *Linear Algebra and Its Applications*, **103**, 103–118. doi:10.1016/0024-3795(88)90223-6.

Hirk R, Hornik K, Vana L (2019). "Multivariate Ordinal Regression Models: An Analysis of Corporate Credit Ratings." *Statistical Methods & Applications*, **28**(3), 507–539. doi:10.1007/s10260-018-00437-7.

Hirk R, Hornik K, Vana L (2020a). "mvord: An R Package for Fitting Multivariate Ordinal Regression Models." *Journal of Statistical Software*, **93**(4), 1–41. doi:10.18637/jss.v093.i04.

Hirk R, Hornik K, Vana L (2020b). ***mvord**: An R Package for Fitting Multivariate Ordinal Regression Models*. R package version 1.1.0, URL https://CRAN.R-project.org/package=mvord.

Hoetker G (2007). "The Use of Logit and Probit Models in Strategic Management Research: Critical Issues." *Strategic Management Journal*, **28**(4), 331–343. doi:10.1002/smj.582.

IBM Corporation (2017). *IBM SPSS Statistics 25*. IBM Corporation, Armonk. URL http://www.ibm.com/software/analytics/spss/.

Kenne Pagui EC, Canale A (2016). "Pairwise Likelihood Inference for Multivariate Ordinal Responses with Applications to Customer Satisfaction." *Applied Stochastic Models in Business and Industry*, **32**(2), 273–282. doi:10.1002/asmb.2147.

Kenne Pagui EC, Canale A (2018). ***PLordprob**: Multivariate Ordered Probit Model via Pairwise Likelihood*. R package version 1.1, URL https://CRAN.R-project.org/package=PLordprob.

Liu X (2009). "Ordinal Regression Analysis: Fitting the Proportional Odds Model Using Stata, SAS and SPSS." *Journal of Modern Applied Statistical Methods*, **8**(2), 30. doi:10.22237/jmasm/1257035340.

Malik HJ, Abraham B (1973). "Multivariate Logistic Distributions." *The Annals of Statistics*, **1**(3), 588–590. doi:10.1214/aos/1176342430.

Martin AD, Quinn KM, Park JH (2011). "**MCMCpack**: Markov Chain Monte Carlo in R." *Journal of Statistical Software*, **42**(9), 1–21. doi:10.18637/jss.v042.i09.

McCullagh P (1980). "Regression Models for Ordinal Data." *Journal of the Royal Statistical Society B*, **42**(2), 109–142. doi:10.1111/j.2517-6161.1980.tb01109.x.

Nash JC (2014). "On Best Practice Optimization Methods in R." *Journal of Statistical Software*, **60**(2), 1–14. doi:10.18637/jss.v060.i02.

Nash JC, Varadhan R (2011). "Unifying Optimization Algorithms to Aid Software System Users: **optimx** for R." *Journal of Statistical Software*, **43**(9), 1–14. doi:10.18637/jss.v043.i09.

Nooraee N, Abegaz F, Ormel J, Wit E, Van den Heuvel ER (2016). "An Approximate Marginal Logistic Distribution for the Analysis of Longitudinal Ordinal Data." *Biometrics*, **72**(1), 253–261. doi:10.1111/biom.12414.

O'Brien SM, Dunson DB (2004). "Bayesian Multivariate Logistic Regression." *Biometrics*, **60**(3), 739–746. doi:10.1111/j.0006-341x.2004.00224.x.

Pedregosa-Izquierdo F (2015). *Feature Extraction and Supervised Learning on fMRI: From Practice to Theory*. Ph.D. thesis, Université Pierre et Marie Curie – Paris VI. URL https://tel.archives-ouvertes.fr/tel-01100921.

Peterson B, Harrell FE (1990). "Partial Proportional Odds Models for Ordinal Response Variables." *Journal of the Royal Statistical Society C*, **39**(2), 205–217. doi:10.2307/2347760.

Pinheiro J, Bates D, R Core Team (2020). ***nlme**: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-144, URL https://CRAN.R-project.org/package=nlme.

Pinheiro JC, Bates DM (1996). "Unconstrained Parametrizations for Variance-Covariance Matrices." *Statistics and Computing*, **6**(3), 289–296. doi:10.1007/bf00140873.

Puccia M, Collett LA, Kernan P, Palmer AD, Mettrick MS, Deslondes G (2013). "Request for Comment: Corporate Criteria." *Technical report*, Standard and Poor's Rating Services.

Python Software Foundation (2018). *Python Language Reference, Version 2.7*. Wilmington. URL https://www.python.org/.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Reusens P, Croux C (2017). "Sovereign Credit Rating Determinants: A Comparison before and after the European Debt Crisis." *Journal of Banking & Finance*, **77**, 108–121. doi:10.1016/j.jbankfin.2017.01.006.

SAS Institute Inc (2018a). ***JMP**, Version 14*. Cary. URL https://www.JMP.com.

SAS Institute Inc (2018b). *SAS/STAT Software, Version 9.4*. SAS Institute Inc., Cary. URL http://www.sas.com/.

Scott DM, Kanaroglou PS (2002). "An Activity-Episode Generation Model That Captures Interactions between Household Heads: Development and Empirical Analysis." *Transportation Research Part B: Methodological*, **36**(10), 875–896. doi:10.1016/s0191-2615(01)00039-x.

StataCorp (2018). *Stata Statistical Software: Release 15*. StataCorp LLC, College Station. URL https://www.stata.com/.

Tutz G (2012). *Regression for Categorical Data*. Cambridge University Press.

Varin C (2008). "On Composite Marginal Likelihoods." *AStA Advances in Statistical Analysis*, **92**(1), 1–28. doi:10.1007/s10182-008-0060-7.

Varin C, Czado C (2010). "A Mixed Autoregressive Probit Model for Ordinal Longitudinal Data." *Biostatistics*, **11**(1), 127–138. doi:10.1093/biostatistics/kxp042.

Varin C, Reid N, Firth D (2011). "An Overview of Composite Likelihood Methods." *Statistica Sinica*, **21**(1), 5–42.

Varin C, Vidoni P (2005). "A Note on Composite Likelihood Inference and Model Selection." *Biometrika*, **92**(3), 519–528. doi:10.1093/biomet/92.3.519.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York. doi:10.1007/978-0-387-21706-2.

Wurm M, Rathouz P, Hanlon B (2017). ***ordinalNet**: Penalized Ordinal Regression*. R package version 2.1, URL https://CRAN.R-project.org/package=ordinalNet.

Yee TW (2010). "The **VGAM** Package for Categorical Data Analysis." *Journal of Statistical Software*, **32**(10), 1–34. doi:10.18637/jss.v032.i10.

Affiliation:

Rainer Hirk, Kurt Hornik, Laura Vana
Department of Finance, Accounting and Statistics
Institute for Statistics and Mathematics
WU Wirtschaftsuniversität Wien
1020 Vienna, Austria
E-mail: Rainer.Hirk@wu.ac.at, Kurt.Hornik@wu.ac.at, Laura.Vana@wu.ac.at