Efficient Bayesian Modeling of Binary and Categorical Data in R: The UPG Package

Gregor Zens
Vienna University of Economics and Business

Sylvia Frühwirth-Schnatter
Vienna University of Economics and Business

Helga Wagner
Johannes Kepler University Linz

Abstract

In this vignette, we introduce the **UPG** package for efficient Bayesian inference in probit, logit, multinomial logit and binomial logit models. **UPG** offers a convenient estimation framework for balanced and imbalanced data settings where sampling efficiency is ensured through marginal data augmentation. **UPG** provides several methods for fast production of output tables and summary plots that are easily accessible to a broad range of users.

Keywords: logit, multinomial, probit, binomial, imbalanced data, MCMC, data augmentation.

1. Introduction

Modeling binary and categorical data is one of the most commonly encountered tasks of applied statisticians and econometricians. Binary probit and logit models, as well as their extensions to multinomial and binomial outcomes, are widely used. In this vignette, we present **UPG**, an R package for Bayesian analysis of well-known binary and categorical data models. **UPG** is based on a number of highly efficient '**U**ltimate **P**ólya **G**amma' Markov chain Monte Carlo (MCMC) algorithms that have been developed in Frühwirth-Schnatter, Zens, and Wagner (2020). The package also features a number of 'plug&play' solutions to facilitate analysis and communication of results.

UPG is especially well suited for analysis of imbalanced data, as the implemented algorithms make efficient posterior simulation possible in these settings. Bayesian analysis of imbalanced data has so far not been the focus of any package released in R while being a highly relevant problem in applied statistics (Johndrow, Smith, Pillai, and Dunson 2019; Frühwirth-Schnatter *et al.* 2020). In general, the Bayesian paradigm has a number of pronounced benefits when it comes to estimation of binary and categorical data regression models. Besides the intuitive appeal of Bayesian uncertainty quantification, it is well known that Bayesian methods are useful in situations characterized by *perfect separation*. This phenomenon occurs when a given covariate (quasi-)perfectly separates the outcome variable of interest. To avoid that parameters drift off to $\pm\infty$ in such scenarios, frequentist statistics suggests for instance penalized likelihood methods (Heinze and Schemper 2002). In a Bayesian context, the combination of a potentially ill-defined likelihood function with a weakly informative prior with finite support usually suffices to resolve the issues related to perfect separation (Gelman, Jakulin, Pittau, Su *et al.* 2008; Rainey 2016). Tightly related to the occurrence of perfect separation are

scenarios where certain outcome categories are observed only very rarely or not at all. For similar reasons, Bayesian inference is often able to avoid implausible parameter estimates in these cases.

Apart from the practical and methodological benefits raised above, **UPG** aims to provide a range of functionality in order to be appealing to different groups of R users. First, researchers that are already familiar with Bayesian statistical analysis can easily introduce the underlying MCMC algorithms in **UPG** as an additional sampling block to pre-existing Gibbs sampling algorithms using a few lines of code. This may prove useful in several applications, including mixture-of-experts models (Gormley and Frühwirth-Schnatter 2019) or analysis of Markov switching models (Frühwirth-Schnatter 2006). Second, for a much broader group of users, the package implements methods for easy and fast production of tables and plots from the estimation output provided. This also facilitates analysis for users that are not commonly working within the Bayesian paradigm.

UPG is licensed under the GNU General Public License 3 and is openly available on the Comprehensive R Archive Network (CRAN, https://cran.r-project.org/package=UPG).

The remainder of this article is structured as follows. Section 2 provides a short overview of the methodology behind **UPG**. Section 3 gives a brief introduction to the package, intended as a quick-start guide. Section 4 presents an extended illustration of the functionality of **UPG**. Finally, Section 5 concludes.

2. Brief methodological overview

This section provides a very brief summary of the latent representations underlying the inner workings of the models implemented in **UPG**. Most of the contents and ideas are directly taken from Frühwirth-Schnatter *et al.* (2020), where the authors develop the methodology underlying **UPG**. This is also where the reader is referred to for full theoretical and computational details.

2.1. Binary regression

Binary regression models for a set of N binary data $\mathbf{y} = (y_1, \ldots, y_N)$ are defined by

$$\Pr(y_i = 1|\mathbf{x}_i, \boldsymbol{\beta}) = F_\varepsilon(\mathbf{x}_i\boldsymbol{\beta}). \tag{1}$$

Choosing the cdf $F_\varepsilon(\varepsilon) = \Phi(\varepsilon)$ of the standard normal distribution leads to the probit model $\Pr(y_i = 1|\mathbf{x}_i, \boldsymbol{\beta}) = \Phi(\mathbf{x}_i\boldsymbol{\beta})$, whereas the cdf $F_\varepsilon(\varepsilon) = e^\varepsilon/(1 + e^\varepsilon)$ of the logistic distribution leads to the logit model

$$\Pr(y_i = 1|\mathbf{x}_i, \boldsymbol{\beta}) = e^{\mathbf{x}_i\boldsymbol{\beta}}/(1 + e^{\mathbf{x}_i\boldsymbol{\beta}}).$$

A latent variable representation of model (1) involving the latent utility z_i is given by:

$$y_i = I\{z_i > 0\}, \qquad z_i = \mathbf{x}_i\boldsymbol{\beta} + \varepsilon_i, \quad \varepsilon_i \sim f_\varepsilon(\varepsilon_i), \tag{2}$$

where $f_\varepsilon(\varepsilon) = F'_\varepsilon(\varepsilon) = \phi(\varepsilon)$ is equal to the standard normal pdf for a probit model and equal to $f_\varepsilon(\varepsilon) = e^\varepsilon/(1 + e^\varepsilon)^2$ for a logit model.

While MCMC estimation based on (2) is straightforward for the probit model using one level of data augmentation involving the latent utilities $(z_1, \ldots, z_N)$ (Albert and Chib 1993),

for the logit model a second level of data augmentation is required in addition to z_i, based on a mixture representation of the logistic distribution. In **UPG**, we apply the mixture representation of the logistic distribution from Frühwirth-Schnatter *et al.* (2020),

$$f_\varepsilon(\varepsilon) = e^\varepsilon/(1+e^\varepsilon)^2 = \frac{1}{4}\int e^{-\omega\,\varepsilon^2/2} p(\omega) d\,\omega, \tag{3}$$

where $\omega \sim \mathcal{PG}(2,0)$ follows a Pólya-Gamma distribution (Polson, Scott, and Windle 2013) with parameters $b = 2$ and $\kappa = 0$. Conveniently, $\omega_i|\varepsilon_i$ again follows a Pólya-Gamma distribution which is easy to sample from. This allows to set up a Gibbs sampler for posterior simulation in a rather straightforward manner.

2.2. Multinomial logistic regression

Let $\{y_i\}$ be a sequence of categorical data, $i = 1, \dots, N$, where y_i is equal to one of at least three unordered categories. The categories are labeled by $L = \{0, \dots, m\}$, and for any k the set of all categories but k is denoted by $L_{-k} = L \setminus \{k\}$. We assume that the observations are mutually independent and that for each $k \in L$ the probability of y_i taking the value k depends on covariates $\mathbf{x}_i$ in the following way:

$$\mathsf{P}y_i = k|\boldsymbol{\beta}_0, \dots, \boldsymbol{\beta}_m = \pi_{ki}(\boldsymbol{\beta}_0, \dots, \boldsymbol{\beta}_m) = \frac{\exp(\mathbf{x}_i\boldsymbol{\beta}_k)}{\sum_{l=0}^{m}\exp(\mathbf{x}_i\boldsymbol{\beta}_l)}, \tag{4}$$

where $\boldsymbol{\beta}_0, \dots, \boldsymbol{\beta}_m$ are category specific unknown parameters of dimension d. To make the model identifiable, the parameter $\boldsymbol{\beta}_{k_0}$ of a baseline category k_0 is set equal to $\mathbf{0}$: $\boldsymbol{\beta}_{k_0} = \mathbf{0}$. Thus, the parameter $\boldsymbol{\beta}_k$ is relative to the baseline category k_0 in terms of the change in log-odds. In the following, we assume without loss of generality that $k_0 = 0$. This multinomial regression model has the following well-known representation:

$$z_{ki} = \mathbf{x}_i\boldsymbol{\beta}_k - \xi_{ki}(\boldsymbol{\beta}_{-k}) + \varepsilon_{ki}, \quad \varepsilon_{ki} \sim \mathcal{LO} \tag{5}$$

$$y_i = \begin{cases} k, & z_{ki} > 0, \\ \neq k, & z_{ki} \leq 0. \end{cases} \tag{6}$$

where the error term ε_{ki} follows a logistic distribution, $z_{ki} = u_{ki} - \max_{\ell L_{-k}} u_{\ell,i}$ is the utility gap between category k and all its alternatives. and the offset $\xi_{ki}(\boldsymbol{\beta}_{-k})$ is defined as:

$$\xi_{ki}(\boldsymbol{\beta}_{-k}) = \log\left(1 + \sum_{\ell \neq \{k,0\}} \exp(\mathbf{x}_i\boldsymbol{\beta}_\ell)\right).$$

UPG uses a Gibbs sampling scheme based on this representation. The details of the scheme and the underlying MCMC boosting algorithm are given in Frühwirth-Schnatter *et al.* (2020).

2.3. Binomial logistic regression

Finally, **UPG** can handle regression models with binomial outcomes, i.e. models of the form

$$y_i \sim \text{BiNom}(N_i, \pi_i), \qquad \text{logit}\,\pi_i = \mathbf{x}_i\boldsymbol{\beta}, \qquad i = 1, \dots, N, \tag{7}$$

where y_i can be interpreted as the number of successes out of N_i trials of individual i. As shown in Frühwirth-Schnatter *et al.* (2020), the binomial model has the following random utility representation for $0 < y_i < N_i$:

$$\begin{aligned} w_i &= \mathbf{x}_i\boldsymbol{\beta} + \varepsilon_{w,i}, \quad \varepsilon_{w,i} \sim \mathcal{GL}_{\text{II}}(k), \\ v_i &= \mathbf{x}_i\boldsymbol{\beta} + \varepsilon_{v,i}, \quad \varepsilon_{v,i} \sim \mathcal{GL}_{\text{I}}(N_i - k), \\ y_i &= k \Leftrightarrow w_i > 0, v_i < 0, \end{aligned} \tag{8}$$

where $\mathcal{GL}_{\text{I}}(\nu)$ and $\mathcal{GL}_{\text{II}}(\nu)$ are, respectively, the generalized logistic distributions of type I and type II. For $y_i = 0$, the model reduces to

$$v_i = \mathbf{x}_i\boldsymbol{\beta} + \varepsilon_{v,i}, \quad \varepsilon_{v,i} \sim \mathcal{GL}_{\text{I}}(N_i), \quad y_i = 0 \Leftrightarrow v_i < 0.$$

For $y_i = N_i$, the model reduces to

$$w_i = \mathbf{x}_i\boldsymbol{\beta} + \varepsilon_{w,i}, \quad \varepsilon_{w,i} \sim \mathcal{GL}_{\text{II}}(N_i), \quad y_i = N_i \Leftrightarrow w_i > 0.$$

For $N_i = 1$, the logistic model results, as both $\mathcal{GL}_{\text{I}}(\nu)$ and $\mathcal{GL}_{\text{II}}(\nu)$ reduce to a logistic distribution for $\nu = 1$. For $y_i = 0$, $z_i = v_i$, whereas for $y_i = 1$, $z_i = w_i$, and the choice equation reduces to $y_i = I\{z_i > 0\}$. To estimate $\boldsymbol{\beta}$ in this framework, it is possible to derive mixture representations similar to (3) for the $\mathcal{GL}_{\text{I}}(\nu)$ and $\mathcal{GL}_{\text{II}}(\nu)$ error distributions, see Frühwirth-Schnatter *et al.* (2020) for details.

2.4. Increasing sampling efficiency through marginal data augmentation

It is well known that Bayesian estimation of binary and categorical data models using data augmentation may result in inefficient sampling behavior, especially in settings with imbalanced data (Johndrow *et al.* 2019). The samplers that are outlined in the previous subsections are, in principle, no exemption from this rule. To tackle this issue, **UPG** implements *boosted* MCMC algorithms that have been developed in Frühwirth-Schnatter *et al.* (2020) to enable highly efficient posterior sampling in a broad range of settings. These MCMC boosting methods are similar in spirit to previous work on MCMC sampling efficiency, see for instance Kastner and Frühwirth-Schnatter (2014) or Kastner, Frühwirth-Schnatter, and Lopes (2017) for MCMC boosting in the context of (factor) stochastic volatility models. Specifically, the sampling algorithms available in **UPG** rely on marginal data augmentation (Liu and Wu 1999; van Dyk and Meng 2001) to increase sampling efficiency. This involves location-based and scale-based expansion of the latent variable representations introduced above. Theoretical and computational details, as well as a number of large-scale simulation studies demonstrating the potential gains in sampling efficiency, may be found in (Frühwirth-Schnatter *et al.* 2020).

3. UPG Basics

The **UPG** package provides efficient sampling algorithms for Bayesian analysis of the probit, logit, multinomial logit and binomial logit model. This section covers the basics of the package, including data requirements, estimation as well as the methods included in **UPG**.

In terms of inputs, the minimum requirement for probit, logit and multinomial logit models is a suitable $N \times 1$ dependent vector `y` and a $N \times d$ design matrix `X`. An additional $N \times 1$

Estimation Command	**Model**
`UPG(y, X, model = "probit")`	Probit
`UPG(y, X, model = "logit")`	Logit
`UPG(y, X, model = "mnl")`	Multinomial Logit
`UPG(y, X, Ni, model = "binomial")`	Binomial Logit

Table 1: Estimation commands for the models included in **UPG**

S3 Method	**Usage**
`print`	Object overview
`summary`	Summary of posterior estimates as well table output
`plot`	Plot coefficient point estimates and credible intervals
`predict`	Predict probabilities for new data or input data
`coef`	Extract posterior means and credible intervals of coefficients
`logLik`	Extract log-likelihood based on posterior mean

Table 2: `S3` methods included in **UPG**

vector of total number of trials `Ni` is necessary to estimate a binomial logit model. For probit and logit models, `y` is supposed to be binary. For multinomial logit models, `y` is a categorical vector containing one realized category out of the set $L = \{0, \dots, m\}$ for each observation. The baseline category k_0 can be freely chosen by the user through parameter `baseline`. If no baseline is provided, the most frequently observed category is used as baseline. For binomial logits, `y` contains the number of successes of each observation. Inputs of class `integer`, `numeric`, `matrix` and `data.frame` are accepted. In the multinomial logit case, `character` and `factor` are accepted as dependent vector types as well. Depending on the specified model type, **UPG** will use a variety of data checks to ensure proper estimation.

The necessary tools for efficient estimation of binary and categorical data models in a Gibbs sampling framework are wrapped into a single estimation function `UPG()` to provide a minimalistic user interface. The four different models included in **UPG** can be called using the `model` parameter as shown in Table 1. An illustration of the estimation process and the most important posterior analysis methods using **UPG** are discussed in the next section.

In terms of output, `UPG()` will return one out of four S3 object classes, depending on the specified `model`. The classes are `UPG.Probit`, `UPG.Logit`, `UPG.MNL` and `UPG.Binomial`. These objects hold the full posterior distribution for all parameters. In addition, all user inputs are copied into the output object for further analysis. Several `S3` methods can be applied to any of these objects. The main task of these methods is to conveniently summarize the generated posterior samples. The methods themselves are summarized in Table 2 and will be discussed in further detail in the subsequent section using extensive examples.

4. Analyzing binary and categorical data using UPG

This section provides information on the data sets that are included in **UPG** and uses these data sets as running examples to demonstrate the functionality of the package.

4.1. Bayesian binary regression: Probit and Logit

To demonstrate how to estimate and analyze Bayesian probit and logit models using **UPG**, a microeconomic data set on female labor force particpation from the US *Panel Study of Income Dynamics* is included. It features a binary variable indicating labor force status as well as a number of additional covariates for 753 women:

```
R>  data("lfp", package = "UPG")
R>  head(lfp, 5)
```

```
  lfp intercept k5 k618        age wc hc       lwg   inc
1   1         1  1    0 -1.3053889  0  0 1.2101647 10.91
2   1         1  0    2 -1.5531414  0  0 0.3285041 19.50
3   1         1  1    3 -0.9337602  0  0 1.5141279 12.04
4   1         1  0    3 -1.0576365  0  0 0.0921151  6.80
5   1         1  1    2 -1.4292651  1  0 1.5242802 20.10
```

The binary dependent variable `lfp` takes the value of 1 if the woman is participating in the labor force. `k5` gives the number of children under the age of 5, `k618` indicates the number of children between 6 and 18 years, `age` is a standardized age index and `wc` as well as `hc` are binary indicators capturing whether a college degree was obtained by the wife and the husband, respectively. In addition, two income related predictors are included, where `lwg` describes the expected log wage of the woman and `inc` gives the logarithm of family income exclusive of the income of the woman. This data set comes from the **carData** package and has been originally analyzed in Mroz (1987).

Model estimation

To construct a suitable design matrix `X` and a binary dependent vector `y` for probit and logit models, it suffices to split the data set as follows:

```
R>  y <- lfp[, 1]
R>  X <- lfp[, -1]
```

In order to estimate a Bayesian logit model, we can use

```
R>  results.logit <- UPG(y = y, X = X, model = "logit")
```

```
Checking data & inputs ...
Initializing Gibbs Sampler ...
Simulating from posterior distribution ...
  |============================================| 100%
Sampling succesful!
Saving output ...
Finished! Posterior simulation took 2.8 seconds.
```

In the remainder of this subsection, it is assumed that the goal is to estimate and analyze a logit model using `model = 'logit'`. Changing the `type` parameter to `model = 'probit'` allows to estimate a probit model. The syntax in the illustration below holds for both models.

Tabulating results

Applying `summary` to the output object results in a quick overview of the regression results in the form of tabulated parameter estimates. Continuing the running example, it is easy to generate a table with posterior means and standard deviations as well as credible intervals:

```
R>  summary(results.logit)

--- Bayesian Logit Results ---

N = 753
Analysis based on 1000 posterior draws after
an initial burn-in period of 1000 iterations.
MCMC sampling took a total of 2.8 seconds.


|          |  Mean|   SD|  Q2.5| Q97.5| 95% CI excl. 0 |
|:---------|-----:|----:|-----:|-----:|:--------------:|
|intercept |  0.50| 0.24|  0.05|  1.00|       *        |
|k5        | -1.47| 0.20| -1.85| -1.11|       *        |
|k618      | -0.06| 0.07| -0.20|  0.07|                |
|age       | -0.50| 0.11| -0.72| -0.31|       *        |
|wc        |  0.81| 0.24|  0.34|  1.26|       *        |
|hc        |  0.12| 0.21| -0.27|  0.52|                |
|lwg       |  0.61| 0.15|  0.31|  0.92|       *        |
|inc       | -0.04| 0.01| -0.05| -0.02|       *        |
```

In terms of interpretation, it is for instance visible that women with a college degree (`wc`) are more likely to participate in the labor force compared to women with no formal tertiary education, holding everything else constant. On the contrary, women who have small children under the age of 5 (`k5`) are *ceteris paribus* less likely to be active in the labor force compared to women without young children.

A number of possibilities for exporting summary tables to LaTeX, HTML or Microsoft Word using `summary(obj, type = c("html","latex","pandoc"))` are available.[1] The user can choose from a number of different options to customize the table output directly, including the upper and lower bounds for the credible intervals based on posterior quantiles specified using `q`, the names of the variables using `names`, the number of significant digits using `digits`, the subset of variables to be used in the table using `include` and the table caption using `cap`. Further customizations are easy to implement, as `summary` returns a `knitr_kable` object that can be further modified using the **knitr** package (Xie 2020). More details can be found in the package manual.

Visualizing results

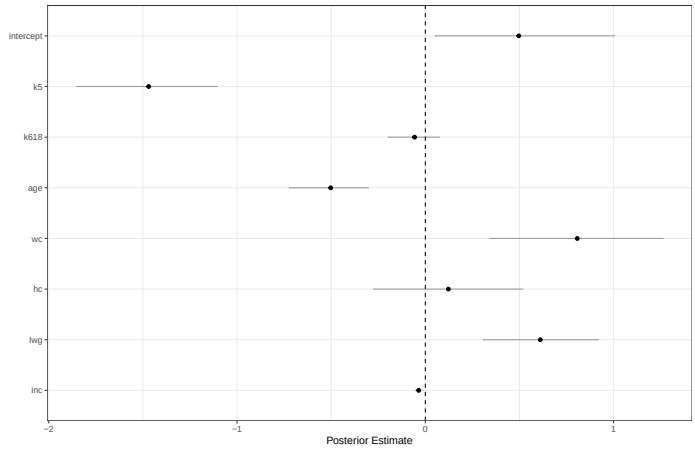
In case a more visual representation of the model output is desired, the `plot` function can be used to generate publication-ready coefficient plots for all four available models using **ggplot2**

[1] If a LaTeX table is desired, the 'booktabs' package has to be included in the preambel of the LaTeX document.

(Wickham 2016). Similar to the `summary` function, `plot` allows the user to customize a number of pre-specified parameters such as axis labels (`xlab`, `ylab`), coefficient names (`names`), the width of the credible intervals (`q`), and the set of included variables (`include`). `plot` will return a `ggplot2` object that can be further modified using the arsenal of tools from the `ggplot2` universe.

Continuing the logit example, we can generate a simple coefficient plot using

```
R>  plot(results.logit)
```



These plots provide point estimates as well as credible intervals for each covariate by default. The variables may be sorted by estimated effect size using `sort = TRUE`. Otherwise, they appear in the same order as in `X`.

Predicting probabilities

In several situations, applied researchers are not necessarily interested in examining the estimated coefficients, but in using these estimates to generate predictions. For these scenarios, `predict` may be used to produce point estimates and credible intervals of predicted probabilities based on the estimated model. These predictions can be generated using either the data provided for model estimation or new, external data provided by the user. Continuing the running example,

```
R>  predict(results.logit)
```

will return a list containing the posterior mean as well as the 97.5% and 2.5% posterior quantiles of the predicted probabilities given the input data. In case the user wants to predict probabilities using external data, a suitable explanatory matrix with the same number of columns and same variable ordering must be provided. The syntax in that case is

```
R>  predict(results.logit, newdata = X.new)
```

where `X.new` is the new design matrix used for prediction. Similar to the other available `S3` methods in **UPG**, the credible intervals can be specified using the parameter `q`.

Log-likelihood

In case the user is interested in the log-likelihood of the data given the parameters, a `logLik` method is available. Applying this method to the output will extract the log-likelihood evaluated at the posterior mean of the parameters:

```
R>  logLik(results.logit)

'log Lik.' -452.645 (df=8)
```

This log-likelihood object holds information on the number of observations as well as the number of estimated parameters.

4.2. Bayesian binomial logistic regression

To demonstrate how to estimate binomial logit model using **UPG**, aggregated individual passenger data of the *RMS Titanic* is included as an example data set:

```
R>  data("titanic", package = "UPG")
R>  head(titanic, 5)

  survived total intercept pclass female age.group
1        0     1         1      1      1         5
2        5     5         1      2      1         5
3       12    17         1      3      1         5
4        2     2         1      1      0         5
5        8     8         1      2      0         5
```

The passengers have been split into several groups that are based on passenger class (`pclass`), five year age groups (`age.group`) and gender (`female`). For each group, total passenger counts (`total`) and the number of passengers that survived the disaster (`survived`) are provided. The data set is an aggregate version of the well-known titanic data set (Hilbe 2007, Table 6.11) that has for instance been previously analyzed in Frühwirth-Schnatter, Frühwirth, Held, and Rue (2009).[2]

Model estimation

In this case, the dependent vector of successes is `survived` whereas the number of total trials corresponds to `total`. Both vectors have to be provided in addition to some explanatory variables to be able to estimate a binomial logit model using **UPG**. Hence, the data needs to be split into three parts parts prior to estimation:

```
R>  y  <- titanic[,1]
R>  Ni <- titanic[,2]
R>  X  <- titanic[,-c(1,2)]
R>  results.binomial <- UPG(y = y, X = X, Ni = Ni, model = "binomial")
```

[2]See https://www.kaggle.com/c/titanic/ for more details.

```
Checking data & inputs ...
Initializing Gibbs Sampler ...
Simulating from posterior distribution ...
  |=============================================| 100%
Sampling succesful!
Saving output ...
Finished! Posterior simulation took 1.84 seconds.
```

All further steps of analysis are similar to before. As an example, we tabulate the results using a credible interval based on the 10% and 90% posterior quantiles:

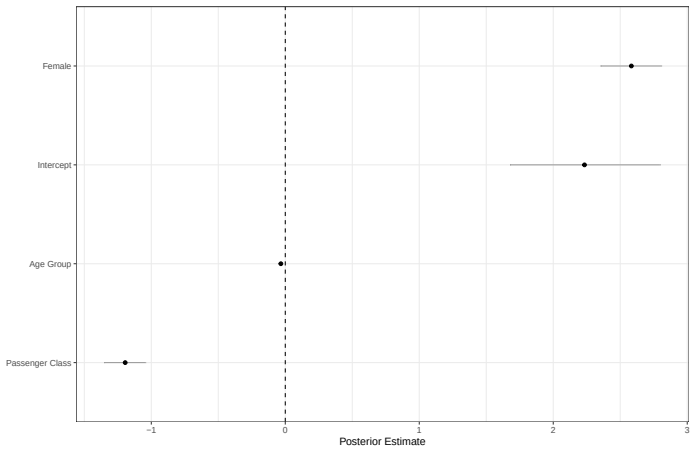
```
R>  summary(results.binomial, q = c(0.1, 0.9))
```

```
--- Bayesian Binomial Logit Results ---

N = 78
Analysis based on 1000 posterior draws after
an initial burn-in period of 1000 iterations.
MCMC sampling took a total of 1.84 seconds.
```

	Mean	SD	Q10	Q90	80% CI excl. 0
intercept	2.23	0.43	1.68	2.80	*
pclass	-1.19	0.12	-1.35	-1.05	*
female	2.58	0.18	2.35	2.81	*
age.group	-0.03	0.01	-0.04	-0.02	*

In terms of interpretation, we can for instance see that female passengers have had a much higher survival probability compared to their male counterparts. A higher passenger class (corresponding to cheaper tickets) is associated with higher mortality as well. Finally, the log-odds of survival decrease with increasing age. To demonstrate custom credible intervals when plotting results, the estimation output is visualized using `q = c(0.1, 0.9)`. This results in a 80% credible interval based on the 0.1 and 0.9 quantiles of the posterior distribution. In addition, custom variable names are provided and `sort = TRUE` ensures that the variables are ordered based on estimated (average) effect size:

```
R>  plot(results.binomial,
R>       sort  = TRUE,
R>       q     = c(0.1, 0.9),
R>       names = c("Intercept", "Passenger Class", "Female", "Age Group"))
```



4.3. Bayesian multinomial logistic regression

For the multinomial logit model, a data set on 200 high school students and their program choice (general, vocational or academic) is included together with a binary variable taking the value of 1 for female students (`female`), a categorical variable indicating socio-economic status (`ses`) and standardized results of a writing test (`write`):

```
R>  data("program",package="UPG")
R>  head(program,5)

   program intercept female ses     write
1 vocation         1      1   1 -1.875280
2  general         1      0   2 -2.086282
3 vocation         1      0   3 -1.453276
4 vocation         1      0   1 -1.664278
5 vocation         1      0   2 -2.297284
```

This data set is also known as the `hsbdemo` data set and is provided online by the University of California, Los Angeles Statistical Consulting Group. This data is used in several R packages and in other software tools as example data for multinomial logistic regression.[3]

Model estimation

As mentioned above, dependent variables for multinomial logit estimation have to be provided as a categorical vector. By default, the category that occurs most often is chosen as baseline category. An alternative baseline category may be specified using `baseline`. In the example data set, `academic` is chosen 105 times out of 200 observations and will thus serve as baseline category. The code to create `y` and `X` is quite similar to the probit and logit case:

```
R>  y <- program[,1]
R>  X <- program[,-1]
```

[3]See for instance https://stats.idre.ucla.edu/stata/dae/multinomiallogistic-regression/ for usage of the data in Stata.

To estimate a multinomial logit model, `model = 'mnl'` has to be specified when using the `UPG` command:

```
R>  results.mnl <- UPG(y = y, X = X, model = 'mnl', verbose = FALSE)
```

where we have set `verbose = FALSE` to suppress any output during estimation for illustration purposes. Handling the resulting `UPG.MNL` object is similar to the cases outlined above and is thus only discussed briefly. Tabulation of the results is based on a grouped representation of the model output:

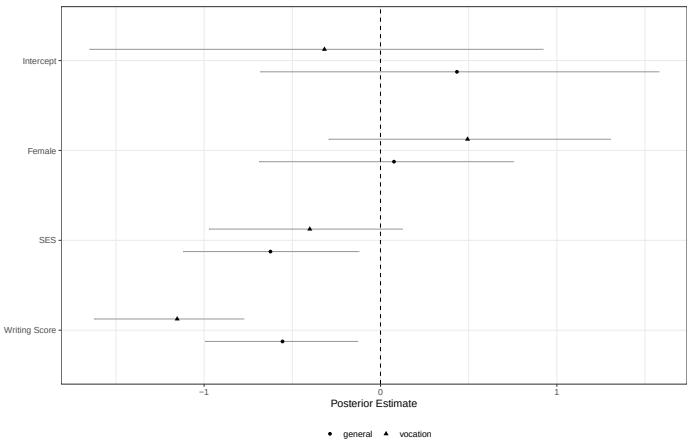
```
R>  summary(results.mnl,
R>          names  = c("Intercept", "Female", "SES", "Writing Score"))
```

```
--- Bayesian Multinomial Logit Results ---

N = 200
Analysis based on 1000 posterior draws after
an initial burn-in period of 1000 iterations.
MCMC sampling took a total of 2.35 seconds.

Category 'academic' is the baseline category.
```

	Mean	SD	Q2.5	Q97.5	95% CI excl. 0
Category 'general'					
Intercept	0.43	0.58	-0.68	1.58	
Female	0.08	0.37	-0.69	0.75	
SES	-0.62	0.26	-1.12	-0.12	*
Writing Score	-0.56	0.22	-0.99	-0.13	*
Category 'vocation'					
Intercept	-0.32	0.65	-1.65	0.92	
Female	0.49	0.40	-0.29	1.30	
SES	-0.40	0.28	-0.97	0.12	
Writing Score	-1.15	0.21	-1.62	-0.78	*

From the output, it becomes obvious that, in the observed sample, higher scores on a writing test decrease the probability of choosing a general or vocational program compared to the baseline of choosing an academic program. Similar conclusions can be drawn from a coefficient plot that is grouped by outcome category:

```
plot(results.mnl,
     names  = c("Intercept", "Female", "SES", "Writing Score"))
```



4.4. UPG-within-Gibbs

In certain applications, users might want to use **UPG** as a single sampling step within a pre-existing Gibbs sampler. Examples where this might be useful include mixture-of-experts models, where a multinomial logit prior can be implemented (see e.g. Gormley and Frühwirth-Schnatter 2019). Similarly, probits as well as binary and multinomial logits do often serve as prior models in Bayesian Markov switching frameworks (Frühwirth-Schnatter 2006).

To implement 'UPG-within-Gibbs' it is possible to access the underlying sampling algorithms directly in order to bypass all checks. Consider the example of a binary logit model. Assuming a starting value for `beta.draw` is given, it suffices to add

```
draw      = UPG:::upg.logit(y,
                            X,
                            nsave = 1,
                            nburn = 0,
                            verbose = F,
                            beta.start = beta.draw)
beta.draw = t(draw$beta)
```

as code block in an existing Gibbs sampler. In this example, `nsave` is set to 1 and `nburn` is set to 0 to generate exactly one posterior sample without burn-in period. `verbose = F` suppresses all console output and parameter `beta.start` is used to provide the current value of `beta.draw` as starting value. Iterating over this code M times and saving the resulting posterior draws of `beta` gives equivalent results to generating M posterior draws from `UPG` directly. However, note that this will, in general, be slightly slower than generating M samples from `UPG` directly. This is due to the overhead that results from repeated function calls in R.

4.5. Further details

The estimation of binary, multinomial and binomial logit models requires simulating from Pólya Gamma distributions. This is accomplished using an implementation in **pgdraw** (Makalic and Schmidt 2016). In terms of prior distributions, the elements of $\boldsymbol{\beta}$ are assumed to follow

independent Gaussian distributions a priori. In order to change the prior variances, the parameters `A0` and `B0` of the `UPG` function are available, refering to the variances of the intercept and the remaining coefficients, respectively. Both A0 and B0 have 4 as default value.

4.6. Sampling efficiency and sampling speed

In order to shed some light on the performance of the implemented models in specific applications, the user can compute several MCMC diagnostic measures using the command `UPG.Diag`. Specifically, a call to `UPG.Diag` will return the effective sample size (ESS) for each coefficient derived using `effectiveSize` from **coda** (Plummer, Best, Cowles, and Vines 2006).[4] In addition, inefficiency factors (IE; given by the number of saved draws divided by the effective sample size) and the effective sampling rate (ESR; given by the effective sample size divided by the running time of the sampler in seconds) are returned. To allow for a more convenient 'quick check' of the behavior of the Markov chain, `UPG.Diag` also returns the minimum, maximum and median across all coefficients as summary statistics of these three diagnostic measures.

To give a sense of magnitude with respect to computational performance, we summarize ESS, IE and ESR for probit as well as binary, binomial and multinomial logit models using the presented example data sets. For each model, 10,000 posterior draws are sampled after an initial burn-in period of 1,000 iterations. All simulations have been run on an AMD Ryzen 5 5500U. The results of this exercise are shown in Table 3. While the table shows that the MCMC algorithms in **UPG** exhibit rather efficient sampling behavior, a pronounced drop in sampling *speed* is visible when comparing the probit regression framework to the remaining models. This is due to the increased computational effort that results from sampling Pólya Gamma random variables. While these are not needed in the MCMC scheme of the probit model, they are necessary for all logit models in **UPG**, increasing computation time in each sweep of the sampler. Nevertheless, due to high levels of sampling *efficiency*, an effective posterior sample size that is sufficient for inference can be generated speedily in the logit frameworks as well.

5. Conclusion

In this vignette, the R package **UPG** is introduced as a software tool for Bayesian estimation of probit, logit, multinomial and binomial logit models. In addition to an implementation that enables efficient estimation through marginal data augmentation, the package is designed to provide easy access to Bayesian models for binary and categorical data for researchers that might not be familiar with the Bayesian paradigm. At the same time, users have the possibility to easily include the provided models as a new sampling step in existing Gibbs samplers. Moreover, the package includes a variety of functions that may be used to produce tables and plots that summarize the estimation output. These methods have been introduced and illustrated through applied examples using data sets that come with the package.

[4] Effective sample sizes in **coda** are derived from the spectral density at 0 which is estimated based on fitting an autoregressive process to the posterior draws.

		Probit	Binary Logit	Multinomial Logit	Binomial Logit
	N	753	753	200	78
	d	8	8	4	4
	$\sum_i N_i$				887
ESS	Min.	3557	2516	1507	3421
	Median	3834	3055	1944	3595
	Max.	3976	3193	2277	3923
IE	Min.	2.51	3.13	4.39	2.55
	Median	2.61	3.27	5.14	2.78
	Max.	2.81	3.97	6.64	2.92
ESR	Min.	843	200	114	386
	Median	909	243	147	406
	Max.	942	254	172	443
Time (in sec.)		4.22	12.56	13.23	8.86

Table 3: Sampling efficiency and sampling speed of the implemented models. Results are based on 10,000 saved draws after an initial burn-in period of 1,000 iterations using the example data sets as input.

Acknowledgments

The authors would like to thank Maximilian Böck, Nikolas Kuschnig, Darjus Hosszejni and Peter Knaus for helpful comments and for being valuable discussion partners during package development.

References

Albert JH, Chib S (1993). "Bayesian analysis of binary and polychotomous response data." *Journal of the American Statistical Association*, **88**, 669–679.

Frühwirth-Schnatter S (2006). *Finite mixture and Markov switching models*. Springer Science & Business Media.

Frühwirth-Schnatter S, Frühwirth R, Held L, Rue H (2009). "Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data." *Statistics and Computing*, **19**(4), 479.

Frühwirth-Schnatter S, Zens G, Wagner H (2020). "Ultimate Pólya Gamma Samplers – Efficient MCMC for possibly imbalanced binary and categorical data." *arXiv Preprint*. 2011.06898.

Gelman A, Jakulin A, Pittau MG, Su YS, *et al.* (2008). "A weakly informative default prior distribution for logistic and other regression models." *The Annals of Applied Statistics*, **2**(4), 1360–1383.

Gormley IC, Frühwirth-Schnatter S (2019). "Mixture of experts models." In S Frühwirth-Schnatter, G Celeux, CP Robert (eds.), *Handbook of Mixture Analysis*, chapter 12, pp. 271–307. CRC Press, Boca Raton, FL.

Heinze G, Schemper M (2002). "A solution to the problem of separation in logistic regression." *Statistics in Medicine*, **21**, 2409–2419.

Hilbe JM (2007). *Negative binomial regression*. Cambridge University Press.

Johndrow JE, Smith A, Pillai N, Dunson DB (2019). "MCMC for imbalanced categorical data." *Journal of the American Statistical Association*, **114**(527), 1394–1403.

Kastner G, Frühwirth-Schnatter S (2014). "Ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC estimation of stochastic volatility models." *Computational Statistics & Data Analysis*, **76**, 408–423.

Kastner G, Frühwirth-Schnatter S, Lopes HF (2017). "Efficient Bayesian inference for multivariate factor stochastic volatility models." *Journal of Computational and Graphical Statistics*, **26**(4), 905–917.

Liu JS, Wu YN (1999). "Parameter Expansion for Data Augmentation." *Journal of the American Statistical Association*, **94**(448), 1264–1274. ISSN 01621459.

Makalic E, Schmidt DF (2016). "High-dimensional Bayesian regularised regression with the BayesReg package." *arXiv preprint arXiv:1611.06649*.

Mroz TA (1987). "The sensitivity of an empirical model of married women's hours of work to economic and statistical assumptions." *Econometrica*, **55**, 765–799.

Plummer M, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL https://journal.r-project.org/archive/.

Polson NG, Scott JG, Windle J (2013). "Bayesian inference for logistic models using Pólya-Gamma latent variables." *Journal of the American Statistical Association*, **108**, 1339–49.

Rainey C (2016). "Dealing with Separation in Logistic Regression Models." *Political Analysis*, **24**, 339–355. doi:10.1093/pan/mpw014.

van Dyk D, Meng XL (2001). "The art of data augmentation." *Journal of Computational and Graphical Statistics*, **10**, 1–50.

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. URL http://ggplot2.org.

Xie Y (2020). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.30, URL https://yihui.org/knitr/.

Affiliation:

Gregor Zens
Vienna University of Economics and Business
1020 Vienna, Austria
E-mail: gzens@wu.ac.at
URL: gregorzens.github.io