Journal of Statistical Software

October 2023, Volume 107, Issue 10. doi: 10.18637/jss.v107.i10

hdpGLM: An R Package to Estimate Heterogeneous Effects in Generalized Linear Models Using Hierarchical Dirichlet Process

Diogo Ferrari
University of California, Riverside

Abstract

The existence of latent clusters with different responses to a treatment is a major concern in scientific research, as latent effect heterogeneity often emerges due to latent or unobserved features – e.g., genetic characteristics, personality traits, or hidden motivations – of the subjects. Conventional random- and fixed-effects methods cannot be applied to that heterogeneity if the group markers associated with that heterogeneity are latent or unobserved. Alternative methods that combine regression models and clustering procedures using Dirichlet process are available, but these methods are complex to implement, especially for non-linear regression models with discrete or binary outcomes. This article discusses the R package **hdpGLM** as a means of implementing a novel hierarchical Dirichlet process approach to estimate mixtures of generalized linear models outlined in Ferrari (2020). The methods implemented make it easy for researchers to investigate heterogeneity in the effect of treatment or background variables and identify clusters of subjects with differential effects. This package provides several features for out-of-the-box estimation and to generate numerical summaries and visualizations of the results. A comparison with other similar R packages is provided.

Keywords: latent effect heterogeneity, regression, semi-parametric Bayesian regression, hierarchical Dirichlet process, Dirichlet process, clustering methods, unsupervised learning, R, mixture models.

1. Introduction

Latent heterogeneity in the effect of explanatory variables is a major concern in science. Uncovering that latent heterogeneity is crucial when the result of the analysis will guide practices and interventions. For instance, increasing the dosage of a new medication may improve symptoms of a disease, but it is a major concern of public health authorities to ensure that a dosage increase does not have the opposite effect (worsening symptoms) for

some latent sub-populations whose members are unknown in advance due to, for example, unobserved genetic traits. The COVID pandemic provides a recent example on a global scale of such concerns about adversarial effect of vaccines for latent sub-populations. Here is another example from labor economics: The intensity of a job market training program can be beneficial on average, but can increase marginalization of some groups because of unobserved personality traits. In these examples, the identification of the latent groups with adversarial effects is crucial to prevent undesirable consequences, *even though the factor causing heterogeneous effects remains latent and unknown by the researcher.*

The presence of latent or unobserved features that create latent effect heterogeneity cannot be avoided in practice. When there is *unobserved* or *unknown* features of sub-populations causing latent effect heterogeneity and the membership in those sub-populations is unknown, conventional estimation methods, such as fixed- and random-effects models, are no longer useful to account for group-specific effects because group identifiers are not observed.

Recent developments, combining machine-learning clustering procedures with traditional regression estimation, provide an attractive alternative. An important class of such methods include the finite mixture of regression models (Ng, McLachlan, Wang, Ben-Tovim Jones, and Ng 2006; Villarroel, Marshall, and Barón 2009), and Dirichlet process regression (DPR) (Hannah, Blei, and Powell 2011; Heinzl and Tutz 2013). However, these methods are complex, require expertise in programming languages to implement, and are often difficult to estimate due to the non-convexity of the objective function and the computational cost involved. Moreover, while there are resourceful, reliable, and user-friendly packages available for R (R Core Team 2023) users to estimate conventional regression and fixed- and random-effects models, the same is not true for DPR models, which makes the latter approaches inaccessible for many practitioners, including those who possess intermediate-level programming skills.

This article discusses the main methodological, implementation, and numerical features of the software package **hdpGLM**, available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=hdpGLM`. In a nutshell, the software provides an interface for practitioners to investigate latent effect heterogeneity in their regression analysis. The **hdpGLM** package can be used to identify latent heterogeneity across sub-populations in the association between the explanatory variables and the outcome. The package provides an easy-to-use implementation of a hierarchical Dirichlet process prior (DPP) regression, as proposed in Ferrari (2020), for out-of-the-box estimation of linear and generalized linear models (GLMs), while simultaneously searching for latent clusters in the data that differ because of the value that the linear coefficients have for one or more of the observed covariates. When the estimation doesn not find a cluster, it returns linear coefficients whose posterior average are indistinguishable from the results of conventional regression methods in R – e.g., those obtained from the function `lm`.

The estimation using the **hdpGLM** package is intuitive and easy to conduct. It adopts standard R symbolic formula notation, as used in the built-in **stats** package and well-known R functions for regression estimation, such as `lm` and `glm`, providing a seamless transition for practitioners from classical models to a more complex regression estimation that accounts for latent effect heterogeneity. The package provides the function `hdpGLM` to estimate hierarchical and non-hierarchical DPR.

This function implements the hierarchical Dirichlet process generalized linear model (hdpGLM) proposed in Ferrari (2020) to estimate a Bayesian Dirichlet process mixture of regression mod-

els, and to estimate latent heterogeneity in the effect of the linear coefficients. The function takes R standard regression formulas as an argument, e.g., `y` $\sim$ `var_1 + var_2 + ⋯ + var_d` and conducts a data-driven search for clusters. It returns $K \in \mathbb{N}$ sets of $d+1$ linear coefficients (intercept included) that best fit, in a Bayesian sense, the correlation between the covariates (`var_1`, ..., `var_d`) and the outcome (`y`). The estimation uses MCMC blocked Gibbs sampler for Gaussian outcomes and Hamiltonian Monte Carlo (HMC) within Gibbs for non-Gaussian outputs – e.g., binary dependent variables.

In addition, the default methods `coef`, `print`, `plot`, `summary`, `family`, `nobs`, and `predict` are supported for objects returned by `hdpGLM`, making it easy for R users at all levels to manipulate the estimation results. The package also provides a number of post-processing functions for generating tidy summaries, visualizations, and classify data points into clusters based on how the treatment or observed covariates affect the response variable in different latent subgroups found during the estimation. In particular, the function `plot` builds on the **ggplot2** package (Wickham 2016), and users can easily customize its aesthetic elements to fit their needs. The estimation algorithms are fully implemented in C++ (Stroustrup 2013) for efficient computation using the **Rcpp** API (Eddelbuettel and François 2011; Eddelbuettel and Balamuta 2018). The C++ code is seamlessly integrated in the package and the user can access everything, from estimation to visualization, within R using standard R syntax.

A search on CRAN reveals that the package **PReMiuM** (Liverani, Hastie, Azizi, Papathomas, and Richardson 2015) is the only other software available that implements DPP regression models in R, and whose goal is similar to the **hdpGLM** software package. Other packages that implement Dirichlet process models on CRAN are not directly comparable due to differences in the scope and goal of the application. Even the **PReMiuM** and **hdpGLM** packages differ in various ways, and are designed to accomplish different goals, making them complementary rather than alternative options to one another. The package **PReMiuM** provides a rich set of features, including clustering the data, variable selection methods, procedures to handle missing values in the explanatory variables, and built-in Markov Chain Monte Carlo (MCMC) diagnostic checks. However, the Dirichlet process is used to model only the intercept term of the regression models in that package.

This means that the **hdpGLM** is unique in two ways. First, it is the only package that estimates and returns the linear coefficients of each cluster found in the analysis, and investigates heterogeneous effects for all covariates across clusters. In that sense, the goal is very different than in other packages, such as **PReMiuM** and **DPpackage** (Jara, Hanson, Quintana, Müller, and Rosner 2011), that use Dirichlet process models. Second, the **hdpGLM** is the only package that provides the user an option to estimate a hierarchical DPR. The hierarchical DPP regression in the **hdpGLM** package investigates if the latent effect heterogeneity across clusters is explained by upper-level covariates, as originally proposed in Ferrari (2020). Table 1 gives a summary of the main similarities and differences between the two main packages on CRAN, namely **hdpGLM** and **PReMiuM**, for DPR estimation in R. Section 6 compares the performance of these two packages.

In the remainder of this article, I focus on the R implementation of the software package **hdpGLM**. For completeness, the next section provides a brief self-contained introduction of the main ideas behind the underlying hdpGLM implemented in R with the software **hdpGLM**. Then, details of the MCMC samplers and some core implementation and design features are presented. Finally, a general analysis workflow with a real application is illustrated.

	Software	
Feature	**PReMiuM**	**hdpGLM**
Main function	`profRegr`	`hdpGLM`
Similarities		
Types of outcome supported[(a)]	Discrete or continuous	Discrete or continuous
MCMC algorithm[(b)]	Gibbs	Gibbs
DPR clustering	✓	✓
Prediction	✓	✓
Differences		
Hierarchical DPR	×	✓
Clusters are based on all coefficients	×	✓
Use standard R formula syntax	×	✓
Include default methods[(c)]	×	✓
Other features		
Variable selection	✓	×
Handle missing values	✓	×
MCMC diagnostics[(d)]	Built-in	External

Table 1: Comparison between the **hdpGLM** package and other alternatives. The symbol ✓ indicates the feature is available, and × indicates it is not available. Note: (a) Current version of **PReMiuM** supports binary, categorical, count and continuous outcome variables. **hdpGLM**'s algorithm is built to support all GLM models; current version includes binary and continuous outcomes only; (b) For the **hdpGLM**, Gibbs is used for Gaussian outcomes, and HMC within Gibbs is required for non-Gaussian outcomes to cluster the data based on all linear coefficients; (c) Current version of **hdpGLM** includes `coef, family, nobs, plot, predict, print`, and `summary`; (d) Diagnostics for the **hdpGLM** can be computed using specialized packages designed for that purpose, such as **coda** (Plummer *et al.* 2006).

2. Background

In classical linear regression models, we estimate a single vector of linear coefficients $\beta = (\beta_0, \ldots, \beta_d)$, that is, for $\beta \in \mathbb{R}^{D_x+1}$ the model is:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d + \epsilon \qquad \beta \in \mathbb{R}^{D_x+1}, d = D_x$$

The coefficient β_l, for $l \in \{1, \ldots, d\}$ represents the marginal association between the outcome y and x_l. Under some circumstances, it can be interpreted as the causal effect of x_l on y (Angrist and Pischke 2008; Pearl 2009; Imbens and Rubin 2015). One of the assumptions of those models is that every observation comes from the same underlying model. That is, a feature x_l affects the outcome in the same way for every observation. It means that, if $\mathsf{E}[\epsilon \mid X] = 0$, for every observation whose value of x_l changes, for example, from -10 to 10, *ceteris paribus*, the average effect on the outcome will be, with probability 1 (certainty), $20\beta_l$. The same applies for GLMs, except that β_l affects the log odds ratio or another transformation of the average parameter. The core point here is that, by modeling assumptions in linear regressions, the average effect (e.g., β_l) of the covariates (e.g., x_l) is constant; that is, there is no heterogeneity in that effect.

Dirichlet and Dirichlet process regressions, on the other hand, do not assume such a homogeneity on the average effect across subjects (Hannah *et al.* 2011). Instead, it assumes that some or all features X can have different effects, on average, on the outcome for different groups of subjects. When that is the case, if there are K groups, each one with different average effects, we can describe the model for each group $k \in \{1, \dots, K\}$ as:

$$y_k = \beta_{0k} + \beta_{1k}x_1 + \cdots + \beta_{dk}x_d + \epsilon_k \qquad \beta_k \in \mathbb{R}^{D_x+1}, \beta \in \mathbb{R}^{(D_x+1)\times K}, d = D_x$$

If we knew the number of subgroups (K) and the group membership for each observation, we could estimate K separate regressions, one for each group, or include a group indicator as an interactive term. The big advantage of the DPR is that the groups with differential effects don't need to be known or observed in advance. In other words, we can use a DPR model to investigate if there is effect heterogeneity caused by unobserved or omitted variables. The model estimates the group membership, the average effects of each covariate in each group, and the proportion of subjects in each cluster.

Denote $Z_i = k$, or Z_{ik} for short, the group membership of subject i on group k. Let π_k denote the proportion of subjects on group k. The regression model becomes:

$$\begin{aligned}
\pi \mid \alpha &\sim \text{Dir}(\alpha) && \alpha = (\alpha_1, \dots, \alpha_K), \alpha_l > 0, l = 1, \dots, K, \\
& && \pi = (\pi_1, \dots, \pi_k, \dots, \pi_K), \textstyle\sum_{k=1}^{K} \pi_k = 1 \\
Z_i \mid \pi &\sim \text{Cat}(\pi) && Z_i \in \{1, \dots, k, \dots, K\} \\
\beta_k \mid Z_{ik} &\sim f_\beta(b \mid Z) && \beta_k \in \mathbb{R}^{D_x+1}, \beta \in \mathbb{R}^{(D_x+1)\times K}, Z_{ik} \triangleq Z_i = k \\
\epsilon_i \mid Z_{ik} &\sim f_\epsilon(e \mid Z) && \\
y_{ik} &= \beta_{k0} + \cdots + \beta_{kd}x_{id} + \epsilon_i && d = D_x
\end{aligned} \tag{1}$$

The same formulation works for a Dirichlet mixture of GLMs, with a little modification in the last two lines of Model 1 above. The Dirichlet *process* prior (DPP) regression generalizes this model for an unknown, possible infinite number of groups K, so the total number of groups K don't need to be set in advance. In DPP, we substitute the Dirichlet distribution as prior for π in the Model 1 and use a Dirichlet *process* prior instead. A constructive way to describe the model with a DPP was presented in Sethuraman (1994). It is called the stick-breaking construction (SBC) and for more details, see Teh, Jordan, Beal, and Blei (2006) and Ferrari (2020). If we denote Δ^∞ the infinity simplex and g^{-1} the inverse link function (e.g., identity function for Gaussian outcomes, exponential function for Poisson outcomes, inverse-logit for Bernoulli outcomes, and so on, following standard notation for generalized linear models; see McCullagh and Nelder (1989)), then the Model 1 can be written as follows in terms of the SBC and for GLMs:

$$\begin{aligned}
V_l \mid \alpha_o &\sim \text{Beta}(1, \alpha_o) && \\
\pi_k &= \begin{cases} V_1, & k = 1 \\ V_k \prod_{l=1}^{k-1}(1 - V_l), & k > 1 \end{cases} && \\
Z_i \mid \pi &\sim \text{Cat}(\pi) && \pi \in \Delta^\infty, \quad Z_i \in \mathbb{N} \\
\beta_k \mid Z_{ik} &\sim f_\beta(b) && \beta_k \in \mathbb{R}^{D_x+1}, \beta \in \mathbb{R}^{(D_x+1)\times K}, Z_{ik} \triangleq Z_i = k \\
y_i \mid Z_{ik}, \beta_k &\sim f(y \mid Z_{ik}, \beta_k) && \mathsf{E}[y_i \mid Z_{ik}, \beta_k] = g^{-1}(X_i^\top \beta_k) \\
& && f(y \mid Z_{ik}, \beta_k) \text{ from exponential family}
\end{aligned} \tag{2}$$

Researchers have applied Model 2 or some variation of it in many different problems, including to relax distributional assumptions on random effects models (Verbeke and Lesaffre 1997; Heinzl and Tutz 2013); estimate population latent heterogeneity when the group membership and number of heterogeneous groups are unknown (Mukhopadhyay and Gelfand 1997; Kleinman and Ibrahim 1998; Hannah *et al.* 2011; Heinzl and Tutz 2013); model latent instrumental variables to deal with endogeneity of covariates (Ebbes, Wedel, Böckenholt, and Steerneman 2005; Ebbes, Wedel, and Böckenholt 2009); study effect heterogeneity in job market training programs (Aakvik, Heckman, and Vytlacil 2005; Heckman and Vytlacil 2007; Matzkin 2007; Ichimura and Todd 2007; Chen 2007); investigate consumer demand in discrete choice models (Rossi, Allenby, and McCulloch 2012; Rossi 2014); study the length of time political appointees stay in their appointed position (Gill and Casella 2009); discern political priorities of senators (Grimmer 2009); track intraparty voting (Spirling and Quinn 2010); study immigrant turnout in elections (Traunmuller, Murr, and Gill 2015); and investigate dynamic aspects of public support for welfare policies (Stegmueller 2013).

Ferrari (2020) proposes a model called hdpGLM which generalizes Model 2 for hierarchical data and estimates context-dependent DPP regressions. The generalization makes it possible to use the model on data sets with and without hierarchical structures. The reader can find detailed information about hierarchical Dirichlet process more generally in Teh *et al.* (2006). Hierarchical data is very common in science. Examples include students nested within schools, voters nested within cities or states, patients nested within hospitals, people nested within countries, legislators nested within state legislatures, and so on.

Essentially, the main idea of the hdpGLM model is the following: Let's call the upper-level information (e.g., school, cities, states, hospitals, or countries) the *context* of the lower-level data point (e.g., students, voters, patients, people). The hdpGLM models heterogeneity in the effect of lower-level features (e.g., family income) of subjects (e.g., patients) across latent clusters in each context (e.g., hospitals or cities), and estimates how the within-context latent heterogeneity depends on features of the context (e.g., hospitals' investments in post-treatment care). In that sense, it differs substantially from the goals of conventional hierarchical models – random and fixed effects–or their Dirichlet extension (see Kyung, Gill, and Casella 2009, 2010).

To model the dependence of the clusters on context-level features, the hdpGLM adds a parameter $\tau \in \mathbb{R}^{(D_w+1)\times(D_x+1)}$ to the model (2), where D_w is the number of context-level covariates (W), and D_x is the dimension of within-context, lower-level covariates (X). The parameter τ captures the dependence of the linear coefficients β on features W of the context. The hdpGLM leaves that dependence undefined to express its generality. The **hdpGLM** package lets the user express that dependence as an upper-level regression on W, as explained in Section 4. The result of the estimation is a set of vectors of linear coefficients, one for each latent group within each context, and a set of context-level coefficients indicating how context features affect within-context effects and latent heterogeneity.

If we use J to denote the number of contexts (e.g., schools, hospitals, states) in the data, W the context-level features (e.g., school budget, number of teachers per student, etc.), X the lower level features (e.g., student performance, gender, race, etc.), and $C_i = j$, or C_{ij} for short, a variable indicating that the observation i comes from context j, then the hdpGLM

model can be formally described as follows:

$$
\begin{aligned}
& V_l \mid \alpha_o \sim \text{Beta}(1, \alpha_o) && \\
& \pi_k = \begin{cases} V_1, & k = 1 \\ V_k \prod_{l=1}^{k-1}(1 - V_l), & k > 1 \end{cases} && \\
& Z_i \mid \pi \sim \text{Cat}(\pi) && \pi \in \Delta^{\infty}, Z_i \in \mathbb{N} \\
& \tau_d \sim f_\tau(t) && d = 1, \ldots, D_x + 1 \\
& \beta_{kj} \mid Z_{ik}, \tau, C_{ij}, W \sim f_\beta(b \mid W, \tau) && X \in \mathbb{R}^{D_x+1}; W \in \mathbb{R}^{D_w+1}, \\
& && \beta_{kj} \in \mathbb{R}^{D_x+1}, \beta \in \mathbb{R}^{(D_x+1)\times K\times(D_w+1)}, \\
& && Z_{ik} \triangleq Z_i = k, C_{ij} \triangleq C_i = j \in \{1, \ldots, J\} \\
& y_i \mid Z_{ik}, \beta_{kj}, X_i, C_{ij} \sim f(y_i \mid Z_{ik}, C_{ij}, X_i, \beta_{kj}) && \mathsf{E}[y_i \mid Z_{ik}, \beta_{kj}, X_i, C_{ij}] = g^{-1}(X_i^\top \beta_{kj}), \\
& && f(y_i \mid Z_{ik}, C_{ij}, X_i, \beta_{kj}) \text{ from} \\
& && \text{exponential family}
\end{aligned}
\tag{3}
$$

Basically, the Model 3 generalizes DPP regression models. More details can be found in Ferrari (2020). Note that in this model, the coefficients β are three-dimensional real-valued matrices, that is, $\beta \in \mathbb{R}^{(D_x+1)\times K\times(D_w+1)}$. Henceforth, the indices lkj in β_{lkj} indicate the coefficient of x_l for group k in context j, and β_{kj} represents the entire $D_x + 1$ vector of coefficients of X for cluster k and context j.

3. MCMC sampler

The package **hdpGLM** implements Models 2 and 3 for R users using the MCMC algorithm proposed in Ferrari (2020). There are a few options to estimate DPP regression models using MCMC (Ishwaran and Zarepour 2000; Neal 2000; Blei and Jordan 2006; Walker 2007). Ferrari (2020) follows Ishwaran and James (2001) and proposes a blocked Gibbs sampler for the hdpGLM. One of the advantages of the MCMC implementation in the **hdpGLM** package is that the number of clusters grow "as needed" during the estimation. The MCMC starts with all points classified into a single cluster. Then, it follows the rules in Algorithm 1 and more clusters are generated based on the posterior density. The algorithm uses a normal distribution for f_τ and f_β, and makes β a linear function of context-level features for each latent cluster, that is $\mathsf{E}[\beta_{kj} \mid \cdot] = W_j^\top \tau$ in the prior distribution of β. When the outcome variable is continuous, these specifications lead to a conjugate prior, and Gibbs update is available for all parameters of the model (see proof in Ferrari 2020). In that case, we substitute the last line of Model 3 with a linear model with Gaussian outcomes, and model the prior parameters as follows:

$$
\begin{aligned}
\sigma_k^2 \mid Z_{ik} &\sim \text{Scaled-inv-}\chi^2(v, s^2) \\
\epsilon_i \mid \sigma_k, Z_{ik} &\sim \mathcal{N}(0, \sigma_k) \\
y_i &= \beta_{0kj} + \beta_{1kj} x_1 + \cdots + \beta_{dkj} x_d + \epsilon_i \qquad Z_i = k, C_i = j
\end{aligned}
\tag{4}
$$

It leads to a Gibbs sampler described in Algorithm 1 (for proof, see Ferrari 2020, Appendix A). Following Ferrari (2020), in the algorithm, Z^* represents the cluster labels at the current

Algorithm 1 Gibbs Sampler for the hdpGLM with Gaussian mixtures

Require: $Z^{(t)} = (Z_1^{(t)}, \ldots, Z_n^{(t)}), \theta_{Z_i}^{(t)}, \tau^{(t)}, \pi^{(t)}$

1: For all $d \in \{1, \ldots, D_x + 1\}$ sample $\tau_d^{(t+1)} \mid \beta^{(t)}, \mathbf{W} \sim \mathcal{N}(\overline{\mu}_{\tau_{dj}}, \overline{\Sigma}_{\tau_d})$ such that $\overline{\mu}_{\tau_{dj}} = \frac{1}{K}\sum_{k=1}^{K} \mu_A^{(k)}$; $\overline{\Sigma}_{\tau_d} = \frac{1}{K}\Sigma_A$; $S_A = \left(\Sigma_\tau^{-1}\sigma_\beta^2 + \mathbf{W}^\top\mathbf{W}\right)^{-1}$; $\mu_A^{(k)} = S_A\mathbf{W}^\top\beta_{dk}^{(t)}$; $\Sigma_A = S_A\sigma_\beta^2$

2: For $j = 1, \ldots, J$

For all $k \in Z_j^*$ sample $\beta_{kj}^{(t+1)} \mid Z^{(t)}, \sigma^{2(t)}, \tau^{(t+1)}, \mathbf{X}, \mathbf{W}, C, y \sim \mathcal{N}_{D+1}(\overline{\mu}_\beta, \overline{\Sigma}_\beta)$ such that $S_\beta = \left(\Sigma_\beta^{-1}\sigma_k^2 + X_{kj}^\top X_{kj}\right)^{-1}$; $\overline{\mu}_\beta = S_\beta\left[\Sigma_\beta^{-1}(\mathbf{W}^\top\tau^{(t+1)})^\top + \frac{X_{kj}^\top y_{kj}}{\sigma_k^{2(t)}}\right]\sigma_k^2$; $\overline{\Sigma}_\beta = S_\beta\sigma_k^{2(t)}$

For all $k \in Z_j^{*C}$ sample $\beta_{kj}^{(t+1)} \mid \tau^{(t+1)}, \mathbf{W} \sim \mathcal{N}_{D+1}((\mathbf{W}^\top\tau^{(t+1)})^\top, \Sigma_\beta)$

3: For all $k \in Z^*$ sample $\sigma_k^{2(t+1)} \mid Z^{(t)}, \beta^{(t+1)}, \tau^{(t+1)}, \mathbf{X}, \mathbf{W}, C, y \sim$ Scale-inv-$\chi_2(\overline{\nu}, \overline{s}^2)$ such that $\overline{\nu} = \nu + N_k^{(t)}$; $\overline{s}^2 = \frac{\nu s^2 + N_k^{(t)}\hat{s}^2}{\nu + N_k^{(t)}}$; $\hat{s}^2 = \frac{1}{N_k^{(t)}}(y_k - X_k\beta_k^{(t+1)})^\top(y_k - X_k\beta_k^{(t+1)})$

For all $k \in Z^{*C}$ sample $\sigma_k^{2(t+1)} \mid Z_i = k \sim$ Scale-inv-$\chi^2(\nu, s^2)$

4: For $i = 1, \ldots, n$, sample $Z_i^{(t+1)} \mid \theta^{(t+1)}, \pi^{(t)}, X_i, y \sim \sum_{k=1}^{K} p_{ik}\delta(Z_i = k)$ such that $p_{ik} \propto \pi_k^{(t)} p(y_i \mid X_i, Z_{ik}^{(t)}, C_{ij}, \theta_{kj}^{(t+1)})$

5: For $k = 1, \ldots, K-1$ sample $v_k^{(t+1)} \overset{iid}{\sim} \text{Beta}\left(1 + N_k^{(t+1)}, \alpha_o + \sum_{l=k+1}^{K} N_l^{(t+1)}\right)$ such that $N_k^{(t+1)} = \sum_{i=1}^{n} I(Z_{ik}^{(t+1)})$

Set $v_K^{(t+1)} = 1$ and compute $\pi_k^{(t+1)} = \begin{cases} v_1^{(t+1)}, & k = 1 \\ v_k^{(t+1)}\prod_{l=1}^{k-1}(1 - v_l^{(t+1)}), & k = 2, \ldots, K \end{cases}$

iteration of the MCMC, and Z^{*C} the values between 1 and K that are not in Z^*. The number K represents the maximum number of clusters selected by the user (see discussion below in Section 4.2). The subscript j indicates the context. The variable N_k is the total number of data points in cluster k, and X_{jk} (or y_{jk}) the covariates (outcome variable) of the units i in context j classified to k in the current iteration.

When the outcome is not continuous, Gibbs is not available for coefficients β. In that case, Ferrari (2020) proposes a Riemann Manifold Hamiltonian Monte Carlo (RMHMC) update (Girolami and Calderhead 2011) within Gibbs to sample from the posterior distribution of those parameters. HMC is more efficient than traditional Metropolis-Hastings updates to explore the posterior distribution, because it takes advantage of the gradient of the posterior distribution (Neal 2011). The HMC uses an ancillary variable v (called momentum) with the same dimension of β (called position variable). For β_{kj} denoting the linear parameter for cluster k in context j, the Hamiltonian equation is defined as:

$$H(\beta_{kj}, v) = U(\beta_{kj}, v) + K(\beta_{kj}, v)$$

The HMC (Neal 2011) defines $U(\beta_{kj}, v) = -\ln f_{\beta_{kj}}(b \mid \cdot)$ and $K(\beta_{kj}, v) = \ln f_v(v)$, where $f_{\beta_{kj}}(b \mid \cdot)$ is the posterior distribution of β_{kj}, and f_v is the distribution of v. Both $U()$ and $K()$ are convex. Based on the definition of $U()$ and $K()$, the gradient is:

$$\nabla_v H(\beta_{kj}, v) = \nabla_v \ln f_v(v)$$
$$\nabla_{\beta_{kj}} H(\beta_{kj}, v) = -\nabla_{\beta_{kj}} \ln f_{\beta_{kj}}(b \mid \cdot)$$

Algorithm 2 Riemann Manifold Hamiltonian Monte Carlo

Require: $Z^{(t)}, \beta^{(t)}, \pi^{(t)}, \tau^{(t)}$

for $j = 1, \ldots, J$ and for $k \in Z_j^*$ **do**
  sample $v^{\text{current}} \sim f_v(v)$
  Let $v \leftarrow v^{\text{current}}, \beta_{kj} \leftarrow \beta_{kj}^{(t)}$
  Set $v \leftarrow v - \frac{e}{2}\nabla_{\beta_{kj}} U(\beta_{kj})$ {*Start the leapfrog steps*}
  for $l = L - 1$ **do**
    $\beta_{kj} \leftarrow \beta_{kj} + e\nabla_v K(\beta_{kj}, v)$
    $v \leftarrow v - \frac{e}{2}\nabla_{\beta_{kj}} U(\beta_{kj})$
  end for
  Set $v \leftarrow v - \frac{e}{2}\nabla_{\beta_{kj}} U(\beta_{kj})$
  Set $v = -v$
  sample $u \sim U(0, 1)$ {*HMC update*}
  if $u < \min\left\{1, \exp\left\{-H(\beta_{kj}, v) + H(\beta_{kj}^{(t)}, v^{\text{current}})\right\}\right\}$ **then**
    $\beta_{kj}^{(t+1)} \leftarrow \beta_{kj}$
  else
    $\beta_{kj}^{(t+1)} \leftarrow \beta_{kj}^{(t)}$
  end if
end for

The leapfrog integration solves this system of simultaneous equations (Girolami and Calderhead 2011; Calin and Chang 2006) using L leapfrog steps of size e. Each step $l = 1, \ldots, L$ is defined by:

$$
\begin{aligned}
v^{l+\frac{\epsilon}{2}} &= v^{l} - \frac{\epsilon}{2}\nabla_{\beta_{kj}} H\left(\beta_{kj}^{l}, v^{l+\frac{\epsilon}{2}}\right) \\
\beta_{kj}^{l+\epsilon} &= \beta_{kj}^{l} + \frac{\epsilon}{2}\left[\nabla_v H\left(\beta_{kj}^{l}, v^{l+\frac{\epsilon}{2}}\right) + \nabla_v H\left(\beta_{kj}^{l+\epsilon}, v^{l+\frac{\epsilon}{2}}\right)\right] \\
v^{l+\epsilon} &= v^{l+\frac{\epsilon}{2}} - \frac{\epsilon}{2}\nabla_{\beta_{kj}} H\left(\beta_{kj}^{l+\epsilon}, v^{l+\frac{\epsilon}{2}}\right)
\end{aligned}
$$

This solution leads to the RMHMC update for the parameter β when the outcome variable is not continuous. Algorithm 2 describes the RMHMC update (see also Neal 2011).

4. hdpGLM package

4.1. Design philosophy, MCMC implementation, and general syntax

The design choices behind the **hdpGLM** software package, and the `hdpGLM` function in particular, seek to: (1) maximize usability for R users at all levels of experience; (2) maximize the integration of the **hdpGLM** package into commonly-used R package ecosystems; (3) provide out-of-the-box DPP and hierarchical DPP models to practitioners to investigate latent heterogeneity in all explanatory variables of regression models, and; (4) offer a computationally-efficient estimation of DPP and hierarchical DPP regression models.

On the latter goal, the MCMC algorithms that estimate the models are fully implemented in C++ for efficient computation, and all of the package functionalities are available through the

R language. The C++ code is integrated in R via the C++ API available in the **Rcpp** package (Eddelbuettel and François 2011). The MCMC in the **hdpGLM** package exactly matches Algorithms 1 and 2 presented in Section 3. For the RMHMC algorithm, the implementation followed the convention and adopted a normal distribution for the momentum variable $v \sim \mathcal{N}(0, I_{(D_x+1)\times(D_x+1)})$ (Neal 2011), which simplifies a term of the leapfrog integrator to:

$$\nabla_v H(\beta_{kj}, v) = v$$

The package is fully implemented for continuous and binary outcome variables, and future versions will include discrete and ordinal outcomes as well. For binary outcome $y_i \sim \text{Ber}(p_{kj})$, the package uses a logit transformation for the average parameter:

$$p_{kj} = \frac{1}{1 + e^{-X_i^\top \beta_{kj}}}$$

For that case, the elements of the RMHMC becomes, for each active cluster k:

$$\begin{aligned} U(\beta_{kj}) = -\ln f_{\beta_{kj}}(b \mid \cdot) \propto -\Bigg[& -\frac{D_x+1}{2}\ln 2\pi - \frac{1}{2}\ln(\det(\Sigma_\beta)) \\ & -\frac{1}{2}(\beta_{kj} - (W_j^\top \tau)^\top)^\top \Sigma_\beta^{-1}(\beta_{kj} - (W_j^\top \tau)^\top) \\ & -\sum_{i \in I_k} y_i \ln\left(1 + e^{-X_i^\top \beta_{kj}}\right) - \sum_{i \in I_k}(1-y_i)\ln\left(1 + e^{X_i^\top \beta_{kj}}\right)\Bigg] \end{aligned}$$

$$\nabla_{\beta_{kj}} U(\beta_{kj}) = -\left[-(\beta_{kj} - (W_j^\top \tau)^\top)^\top \Sigma_\beta^{-1} + \sum_{i \in I_k} X_i y_i p(y_i = 0 \mid \cdot) - \sum_{i \in i_k} X_i (1-y_i) p(y_i = 1 \mid \cdot)\right]$$

The DPP regression is a special case of hdpGLM when there is just one *context* (J=1) and there is no context-level variable W. The algorithm used for DPP regression is the same as algorithm 1, except that it sets $\mathsf{E}[\beta] = 0$ for the prior distribution of β and, obviously, does not sample τ because that parameter is not used. Section 6 contrasts the performance of the MCMC algorithm implemented in the **hdpGLM** against the existing alternative in R.

The other goals – integration, usability, and out-of-the-box DPP regression analysis – are achieved in various ways in the package design. Section 5.1 demonstrates the usability and integration features in detail, with a workflow example for practical application.

First, the **hdpGLM** package returns the posterior samples in a '`mcmc`' object of the R package **coda** (Plummer *et al.* 2006). Various other R packages use **coda** as the main engine for analysis of MCMC outputs, such as the **MCMCpack** (Martin, Quinn, and Park 2011), **R2WinBUGS** (Sturtz, Ligges, and Gelman 2005), **MCMCglmm** (Hadfield 2010), **boa** (Smith 2007), and many others. The user can use packages specialized in MCMC diagnostics directly on the output object of the estimation function `hdpGLM`, and take advantage of all the **coda** facilities to evaluate convergence of the MCMC and visualize sample outputs of DPP regression.

Second, R users don't need to learn new, intricate syntax or a series of different functions to estimate the model and create summaries with the results. The design mimics the syntax of standard, widely-used built-in regression estimation methods from the **stats** package, such as the `lm` and `glm` functions. The `hdpGLM` function receives the same basic arguments as those other functions, including R default symbolic `formulas` for regression. Moreover, the underlying method to create design matrices from input `data.frames` to run the regression,

and the default methods to print, summarize, and visualize the results, follow standard R design. For instance, it becomes immediately obvious for R users who have used the built-in `lm` function the meaning of the summary output from `hdpGLM`, provided the differences between the underlying models are understood – i.e., clustering and Bayesian methods for `hdpGLM` regression.

Third, when non-hierarchical DPP regression models are estimated using the **hdpGLM** package, only one R-standard symbolic formula is required and the argument is mandatory (more details are provided in the next section). When hierarchical DPP regression are estimated, the main function `hdpGLM` requires an additional formula specifying the upper-level covariates. This design differs from other available options in R to estimate conventional hierarchical models – i.e., fixed and random effects – such as those provided in the **lme4** package (Bates, Mächler, Bolker, and Walker 2015).

The option for two separate formulas when hierarchical DPP regression is required is due to several reasons. First, the first formula is mandatory and shares some goals with other widely-used R functions, such as the `lm` and `glm`. I opted to follow standard R syntax to facilitate usage and remind users that the model shares similar goals of those other regression estimation functions. In all cases (`lm`, `glm`, and `hdpGLM`), users depart from observed covariates that they want to include in the regression, based on domain knowledge. Users then run the model and get the estimated linear coefficients. So, the first formula reflects that common goal. The main difference is that the `hdpGLM` function relaxes the modeling assumption of no latent heterogeneity in the association between the covariates and the outcome, as assumed by the `lm` and `glm` functions.

The second formula is optional and used only if the user wants to investigate the dependency of the latent heterogeneity on higher-level covariates. Using two separate formula arguments makes the syntax and the documentation cleaner and more straightforward, and the manipulation of those formulas by internal ancillary functions easier to handle. So, it provides an end-to-end benefit, from maintenance and development to the usage of the package. Finally, different from the first formula, the second formula has a different goal that is not matched by any other R package currently on CRAN. When the second formula is used, the hierarchical model estimates the effect of higher-order observed covariates on the latent heterogeneity of the linear effects of lower-level covariates. Different from the first formula, the goal of the second formula is very different from other hierarchical models implemented in R, e.g., for fixed and random effects estimation, such as those provided by the **lme4** package. Hence, the package separates formulas instead of adopting **lme4**-like syntax to remind the user of the important distinction between the estimation goals of these packages.

4.2. Estimation

A single function called `hdpGLM` handles the estimation of DPPs (Model 2) and hierarchical DPP (Model 3) regression models. It can receive one or two different regression formulas that follow standard R symbolic formula notation. Which model is estimated (DPP or hierarchical DPP) depends on the numbers of formulas provided, and the function automatically selects the appropriate sampler to use. A discussion of the design decisions is presented on Section 4.1. The signature of the function with its default arguments is:

```
hdpGLM(formula1, formula2 = NULL, data, mcmc, family = "gaussian", K = 100,
    context.id = NULL, constants = NULL, n.display = 1000)
```

Both the arguments `formula1` and `formula2` receive standard regression formulas, but only the former is mandatory. For instance, if we set `formula2 = NULL` (default) and `formula1 = y ~ x1 + x2`, the function `hdpGLM` will run a DPP regression (Model 2), which will search for latent subgroups that differ on how the outcome y responds to the covariates x_1 and x_2. When `formula2` is also provided, the function estimates a hierarchical DPP (Model 3) instead. Following the same example, suppose the observations (e.g., persons) come from J contexts (e.g., states), and we measured a variable w with context-level information (e.g., state GDP). If we provided `formula2 = y ~ w` alongside `formula1` as above, we can estimate the effect τ of w on the latent clusters across contexts (states).

The argument `data` receives an R `data.frame` or a more efficient data structure compatible with it, such as **tibble** (Müller and Wickham 2023) or **data.table** (Dowle and Srinivasan 2023). The data must contain the column names specified in the formulas. It is advisable to normalize all the variables in the data before the estimation for algorithm numerical stability.

The user can provide a string to the argument `context.id` with the name of the column in the data frame containing the context labels. This not required, but can facilitate post-estimation reports because the package provides general as well as context-specific summaries and plots of the clusters and linear coefficients. The argument can be a meaningful label of the contexts. For example, if the context is states, we could have a variable named `state` in the data with the name of the states. Otherwise, arbitrary labels are assigned to the contexts to identify them.

The argument `mcmc` receives a named list with the parameters for the MCMC. The list must contain two mandatory named elements, `burn.in` and `n.iter`. Additionally, it can receive three optional elements, `epsilon`, `leapFrog`, and `hmc_iter`. The element named `burn.in` must be an integer, and it indicates the number of iterations to discard before recording the samples (MCMC burn-in period). The element `n.iter` is the number of iterations to keep after the burn-in. The other elements of the list are only needed if the user wants to tune in the HMC updates, which is used for non-Gaussian outcomes (see Section 3 and Algorithm 2). The default values follow recommendations from the literature (Neal 2011) and perform well for in a variety of cases (Ferrari 2020). If users want to modify those parameters, `epsilon` must be a positive number. It is used in the Stormer-Verlet Integrator (a.k.a. leapfrog integrator) to solve the Hamiltonian Monte Carlo in the estimation of the model. The default value is 0.01. `leapFrog` must be an integer. It indicates the number of steps taken at each iteration of the HMC for the Stormer-Verlet Integrator. The default is 40. Finally, `hmc_iter` must receive an integer, and it indicates the number of HMC iteration(s) for each Gibbs iteration. The default is 1. An example of the argument `mcmc` is:

```
list(n.iter = 10000, burn.in = 2000, epsilon = 0.01,
    leapFrog = 40, hmc_iter = 1)
```

The initial state of the Gibbs sampler comes from the prior distribution of the parameters, and the optional argument `constants` of the function `hdpGLM` receives a named list with the constants (prior distributions parameters) of the model specifying those priors. The user can take advantage of the argument `constants` to conduct sensitivity analyses. The list received by that argument can contain a vector named `mu_theta`, which is the average parameter of the multivariate normal prior for β, and whose size must match the number of covariates specified in the argument `formula1` plus one for the intercept. The first element of the vector will be the prior average for the intercept, the second the prior for the coefficient of the first covariate

that appears in `formula1`, and so on (see the complete example below). The list can also contain a square matrix named `Sigma_beta`, which is the covariance matrix of the multivariate normal prior of β, and the dimensions must match the size of `mu_theta`. Users can also set the prior parameter α_0, which is the parameter of the Beta distribution for the SBC (Model 4), by including in the list an element named `alpha` with a positive number. Other elements of the list depend on the family of the GLMs used to capture the outcome variable, which is specified in the argument `family`. If the outcome is continuous and modeled with a Gaussian distribution (`family = "gaussian"`), then the list can contain an element named `s2_sigma`, which is the parameter of the scaled inverse-χ^2 distribution of σ_k^2 as described in Equation 4. The user can also set `df_sigma`, which is the parameter v (degrees of freedom of the scaled inverse-χ^2 distribution) in Model 4. Both `s2_sigma` and `df_sigma` must be positive numbers. If `constants = NULL` (default), the package sets `mu_beta = (0, ..., 0)` and `Sigma_beta = 10I`, where `I` is an identity matrix. The dimensions are automatically adjusted to match the model formulas. The prior for the Beta distribution of the SBC is set to `alpha = 1`. For Gaussian outcome, it sets the priors for σ_k as `s2_sigma = 10` and `df_sigma = 10`. These values produce good estimation performance under various case scenarios (Ferrari 2020). Here is an example of how to use the argument `constants` to select the prior parameters in a regression with two covariates (see Models 3 and 4 for reference):

```
list(mu_beta = c(0, 0, 0), Sigma_beta = 10*diag(3), df_sigma = 10,
    s2_sigma = 10, alpha = 1)
```

The user can select the argument K to set the upper bound of the truncated Gibbs sampler. As discussed in Section 3, one of the advantages of the MCMC implementation in the **hdpGLM** package for efficient computation is that the number of clusters grow "as needed" during the estimation. The MCMC starts with all points classified into a single cluster. Then, it follows the rules in Algorithm 1 and more clusters are generated based on the posterior density. However, although the actual active clusters (i.e., clusters with data points) in each iteration can be small, the prior π on the cluster indicator Z must be large enough to let the active clusters grow as needed. The argument K sets the upper bound for the number of clusters, which is required by the algorithm to set the size of π (see Ishwaran and James (2001) and Ferrari (2020)). For instance, if $K = 100$ (default), the number of active clusters can grow during the estimation up to 100. If the estimation achieves that upper bound, no additional cluster is created to allocate the data points. Too large of an upper bound K will require more memory. If it is too small, it can artificially truncate the number of clusters and classify the data in less clusters than it should. Users are advised to monitor the number of *active clusters* - that is, the number of clusters actually occupied with data points - either while the MCMC is running or after the estimation is completed. If the upper bound is achieved at any point, the user can increase it and rerun the analysis (this process will be automated in future versions).

The package provides two ways to monitor the number of occupied clusters to check if K needs to increase. After the estimation is completed, the maximum number of active clusters used at any point during the MCMC is saved in the output of the function `hdpGLM` (more details about the complete output are provided below). The user can check if that maximum is equal or close to K, in which case the user can increase K. For instance:

```
R> library("hdpGLM")
R> set.seed(777)
```

```
R> sim <- hdpGLM_simulateData(n = 200, nCov = 2, K = 2, nCovj = 1, J = 5,
+    family = "gaussian")
R> df <- sim$data
R> mod <- hdpGLM(y ~ X1 + X2, y ~ W1, data = df, K = 50,
+    mcmc = list(burn.in = 0, n.iter = 100), n.display = 30)
R> mod$max_active

[1] 8
```

In the example above, the maximum number of clusters activated at any point during the MCMC was 8, which is much lower than the value of 50 used for K. It indicates that the algorithm needed much less than 50 clusters to sample from the regions of high density, so the truncation of π to 50 is not affecting any result.

The user doesn not need to wait until the estimation finishes to check the maximum number of active clusters. A partial report will be display in R during the MCMC iterations every time it reaches as many iterations as indicated in the argument `n.display` of the function `hdpGLM`. For instance, if `n.display = 30`, the function `hdpGLM` will display a report at every other 30 iterations. For example, the report after 60 iterations for the previous model will look like this:

```
Family of the distribution of the outcome variable of the
 mixture components: gaussian

Burn-in: 0
Number of MCMC Iterations : 100

Iteration: 60

Acceptance Rate for beta         : 1
Average Acceptance Rate for beta : 1

Maximum Number of Clusters Allowed (K)                      : 50
Maximum Number of Clusters Activated Among All Contexts     : 8
Maximum Number of Clusters Active in the Current Iteration : 8

(displaying only clusters with more than 5% of the data)
Clusters in Context 1
    1.0000    3.0000
   30.0000   69.0000

Clusters in Context 2
    1.0000    2.0000    3.0000    4.0000    5.0000
   14.0000   13.0000   18.0000   39.0000   13.5000

Clusters in Context 3
    1.0000    2.0000    5.0000
```

```
   31.0000   39.5000   26.0000

Clusters in Context 4
    1.0000    4.0000    5.0000
   19.0000   50.0000   28.0000

Clusters in Context 5
    1.0000    2.0000    7.0000
    6.5000   32.0000   60.5000
```

The report contains several important pieces of information. It shows the current iteration when the report was displayed (`Iteration: 60`); the acceptance rate (current and on average) for the parameter β, which is equal to 1 for Gaussian models because they use a Gibbs sampler, but it is usually smaller than 1 when HMC within Gibbs is used to sample β, which is the case for regression with non-Gaussian outcomes. It also shows the maximum number of clusters allowed, which is set by the user using the argument K of the function `hdpGLM`; the maximum number of activated clusters up to the iteration the report was generated (8 in the example); the number of active clusters in the current iteration (8 in the example); and the percentage of data points in each active cluster in the current iteration (clusters with less than 5% of the data are omitted in the report).

The output of the function `hdpGLM` is either an object of class '`dpGLM`' or '`hdpGLM`', depending on which model is estimated (DPP or hierarchical DPP). The object is a list with various elements, and here is an example of the main ones:

```
R> class(mod)

[1] "hdpGLM"

R> str(mod)

 $ samples      : 'mcmc' num [1:2344, 1:6] 1 2 1 2 3 4 5 6 1 2 ...
 $ tau          : 'mcmc' num [1:100, 1:6] 1.0425 -0.0564 -1.152 0.3185 ...
 $ pik          : num [1:1000, 1:50] 0.446 0.564 0.416 0.347 0.554 ...
 $ max_active   : int 8
 $ n.iter       : int 100
 $ burn.in      : int 0
```

The object `hdpGLM` returned by the function `hdpGLM` contains an element named `samples`, which has the samples of the linear coefficients β for all clusters. That element belongs to the class '`mcmc`', which is a class defined and used in various specialized R packages to produce diagnostics for MCMC samples, such as the package **coda**. The element named `tau` is the posterior distribution of the parameter τ. It is only returned when users estimate a hierarchical DPP regression.

The `pik` element of the object `hdpGLM` is a $n \times K$ matrix with the estimated probabilities that the observation $i = 1, \ldots, n$ belongs to the cluster $k = 1, \ldots, K$, where K is the maximum number of clusters allowed (argument K of the function `hdpGLM`). The classification of the data points into clusters use those probabilities.

4.3. Estimation numerical summaries

The software package `hdpGLM` uses the default method `summary` to create a summary of the posterior samples. To illustrate the `summary` function provided by the **hdpGLM** package, consider the model fit previously, `mod`. Suppose we estimate the following model, which examines the effect of a new assignment policy (x_1) and teacher personality (x_2) on student performance (y). We collected data on many different schools, and measured school investment in extracurricular activities (W_1).

The dataset `df` contains columns named `y`, `X1`, `X2`, and `W1` measuring the relevant variables. We can estimate the coefficients β and search for latent subgroups (k) of students who respond differently to x_1 and x_2 by using `formula1 = y ~ X1 + X2` in the function `hdpGLM`, and we can estimate how school investment in extracurricular activities affects the clustering of student response to covariates by using `formula2 = y ~ W1`. After the estimation, if we use the function `summary` to summarize the object returned by the function `hdpGLM` it will print the following (suppose `df` is the name of the `data.frame` with the school-student data):

```
R> summary(mod)

--------------------------------
hdpGLM Object

Maximum number of clusters activated during the estimation: 8
Numer of MCMC iterations: 100
Burn-in: 0

Number of contexts : 5

Number of clusters (summary across contexts):

  Average Std.Dev Median Min. Max.
1       3  1.2247      3    2    5
--------------------------------


Summary statistics of clusters with data points in each context

--------------------------------
Coefficients and clusters for context 1

              Post.Mean Post.Median HPD.lower HPD.upper Cluster
1 (Intercept)  -0.97611    -0.98064   -1.3143  -0.60395       1
2          X1  -5.57226    -5.58286   -5.8340  -5.34086       1
3          X2   2.16103     2.12966    1.7468   2.69694       1
4 (Intercept)  -1.49072    -1.46188   -1.8145  -1.15886       3
5          X1  -5.75276    -5.73522   -6.1323  -5.43723       3
6          X2   3.14486     3.19251    2.5276   3.52279       3
```

```
--------------------------------
Coefficients and clusters for context 2

                Post.Mean Post.Median  HPD.lower HPD.upper Cluster
1  (Intercept)  -3.554739    -3.42616  -4.608871  -2.93400       1
2           X1   1.265157     1.18973   0.184679   2.47285       1
3           X2 -17.004367   -17.15814 -17.998191 -15.94061       1
4  (Intercept)  -1.039218    -1.14437  -1.808516  -0.50807       2
5           X1   0.415246     0.39460  -0.038041   0.76823       2
6           X2 -17.141898   -17.07015 -17.882653 -16.34140       2
7  (Intercept)  -3.206114    -3.58882  -4.696327   3.32715       3
8           X1   0.019245    -0.20007  -1.334364   1.42690       3
9           X2 -16.419047   -17.04453 -20.739489 -12.20822       3
10 (Intercept)  -2.903687    -3.12625  -3.711885  -2.58690       4
11          X1   0.470094     0.49289  -0.177659   0.98629       4
12          X2 -16.326264   -16.49657 -16.890567 -15.94055       4
13 (Intercept)  -3.609961    -3.77480  -6.672626  -2.12280       5
14          X1  -0.148885    -0.29762  -4.082790   2.07480       5
15          X2 -16.979781   -17.37436 -17.868185 -14.08068       5

--------------------------------
... (truncated)
--------------------------------
Context-level coefficients:
                          Description Post.Mean HPD.lower HPD.upper
1                Intercept of beta[0]  -0.67849   -3.5305   2.10751
2                Intercept of beta[1]  -2.52145   -5.4717   0.40664
3                Intercept of beta[2]  -0.46091   -3.4801   2.08948
4 Effect of context term 1 on beta[0]   0.74241   -1.2963   2.97354
5 Effect of context term 1 on beta[1]  -1.45167   -3.4145   0.69349
6 Effect of context term 1 on beta[2]   4.64269    2.8063   6.44313

--------------------------------
```

It gives a comprehensive assessment of the main results. The first block of the summary (named `hdpGLM Object`) gives the number of iterations used in the MCMC and the number of clusters activated during the estimation. For a hierarchical DPP regression, it also provides the number of contexts, and a summary of the clustering results (average, median, minimum, maximum, and standard deviation of the number of clusters found across contexts). Then, it prints a full list of the posterior summaries for the linear coefficients of each cluster and each context. Lastly, for hierarchical DPP regression, the function `summary` also prints the posterior summary for the parameter τ, grouped under "Context-level coefficients." The applied examples below illustrate the interpretation of these coefficients.

The package **hdpGLM** also provides a function `summary_tidy` to create tidy summaries of the output (Wickham 2014). The tidy summary returns the summary output in a `tibble` format. Because that function returns a tidy summary, the user can easily subset and slice the summary output to retrieve estimates for particular contexts (e.g., schools) or clusters.

All standard R functions available for `data.frame`-like structures are readily available to manipulate the summary output to plot, using packages such as the **ggplot2** (Valero-Mora 2010; Ginestet 2011; Wilkinson 2012; Wickham 2016; Gómez-Rubio 2017)), or to create tables and reports using package ecosystems such as the **tidyverse** tools (Wickham *et al.* 2019; Lee, Sriutaisuk, and Kim 2020) or the R package **kableExtra** (Zhu 2021) to export the output table to LaTeX.

For a non-hierarchical DPP regression, the function `summary_tidy` returns a `tibble` (Müller and Wickham 2023) with the posterior summary for β. For a hierarchical DPP regression, it returns a list of two `tibble` objects, one for β and one for τ:

```
R> summary_tidy(mod)

$beta
# A tibble: 60 × 9
       k     j Parameter term           Mean  Median     SD
   <dbl> <dbl> <chr>     <chr>         <dbl>   <dbl>  <dbl>
 1     1     1 beta[0]   (Intercept)  -0.976  -0.981 0.167
 2     1     1 beta[1]   X1           -5.57   -5.58  0.126
 3     1     1 beta[2]   X2            2.16    2.13  0.256
 4     1     1 sigma     sigma         0.616   0.553 0.203
 5     1     2 beta[0]   (Intercept)  -3.55   -3.43  0.564
 6     1     2 beta[1]   X1            1.27    1.19  0.635
 7     1     2 beta[2]   X2          -17.0   -17.2   0.705
 8     1     2 sigma     sigma         0.490   0.481 0.0743
 9     1     3 beta[0]   (Intercept)   1.07    1.08  0.217
10     1     3 beta[1]   X1          -11.5   -11.5   0.269
   HPD.lower HPD.upper
       <dbl>     <dbl>
 1    -1.31     -0.604
 2    -5.83     -5.34
 3     1.75      2.70
 4     0.377     1.10
 5    -4.61     -2.93
 6     0.185     2.47
 7   -18.0     -15.9
 8     0.387     0.620
 9     0.761     1.48
10   -11.8     -11.0
# 50 more rows
# Use `print(n = ...)` to see more rows

$tau
# A tibble: 6 × 8
      w  beta Parameter Description               Mean    SD
  <int> <int> <chr>     <chr>                    <dbl> <dbl>
1     0     0 tau[0][0] Intercept of beta[0]    -0.678 1.52
2     0     1 tau[0][1] Intercept of beta[1]    -2.52  1.61
```

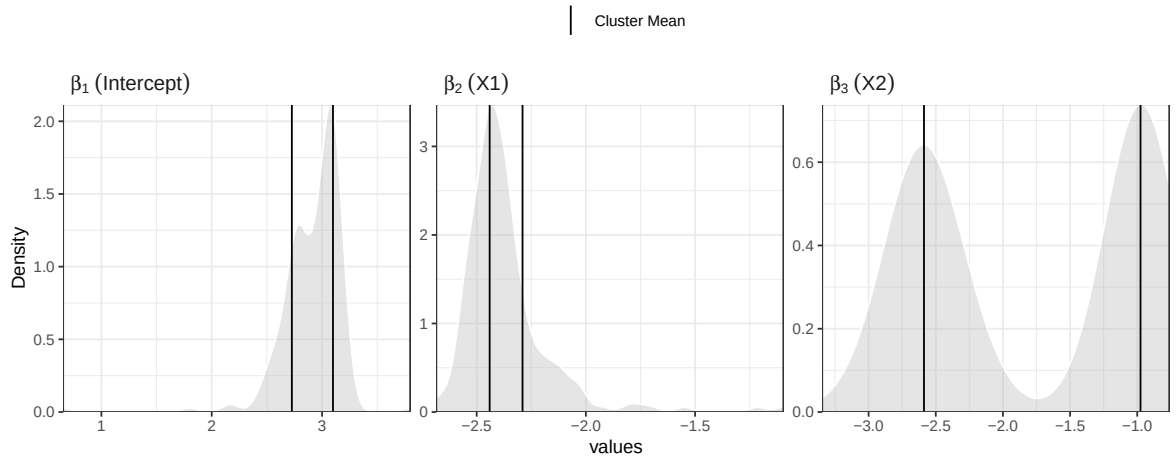

Figure 1: Posterior distribution of β produced by function `plot` of the **hdpGLM** package.

```
3     0     2 tau[0][2] Intercept of beta[2]    -0.461 1.53
4     1     0 tau[1][0] Effect of W1 on beta[0]  0.742 1.05
5     1     1 tau[1][1] Effect of W1 on beta[1] -1.45  1.06
6     1     2 tau[1][2] Effect of W1 on beta[2]  4.64  0.953
  HPD.lower HPD.upper
      <dbl>     <dbl>
1     -3.53     2.11
2     -5.47     0.407
3     -3.48     2.09
4     -1.30     2.97
5     -3.41     0.693
6      2.81     6.44
```

4.4. Visualization

The package **hdpGLM** provides a variety of options to visualize the output of the estimation. It provides a generic method `plot` that takes the output of the `hdpGLM` function with the posterior samples and generates plots that include cluster posterior averages. The features of the plot depend on (1) the model estimated (DPP or hierarchical DPP), and (2) the user's choice to visualize aggregate or cluster-specific estimates of the coefficients.

Figure 1 shows an example of the output of the `plot` function for a non-hierarchical DPP regression estimated using the `hdpGLM` function with two covariates named x_1 and x_2 in the data. The plot shows the posterior distribution of all linear coefficients β for all clusters. The vertical lines represent the cluster posterior averages.

Users have many built-in options to control plot aesthetics. They can set the argument `separate = TRUE` in the `plot` function to plot the clusters separately. In that case, the plots also include the 95% highest posterior density intervals for each cluster and coefficient. The user can also select which coefficients to plot by including an argument `term`, which receives a vector of strings with the names of the covariates. For instance, if we use `plot(mod,`

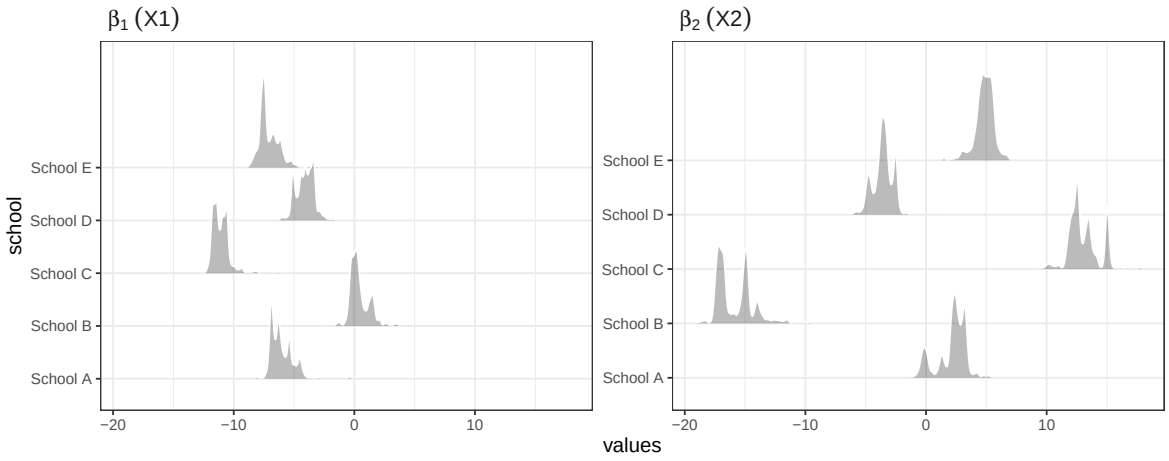

Figure 2: Posterior distribution of β in each context (e.g., schools) produced by function `plot` of the **hdpGLM** package.

`separate = TRUE, term = c("X1"))`, the plot created will display the posterior distribution of all clusters separately with their respective 95% HPD intervals, and it will plot only the distribution of the coefficients of `X1`.

If one estimated a hierarchical DPP, then the set of linear coefficients are distributed by cluster and contexts. Moreover, it estimates an additional set of coefficients τ representing the context-level effect. The package provides three built-in functions to visualize the results of the model estimation in that case.

Figure 2 is created by estimating a hierarchical DPP regression estimated using the `hdpGLM` function with two covariates named x_1 and x_2 in the data, and five contexts (e.g., schools) with a context-level feature named W_1:

```
R> mod <- hdpGLM(y ~ X1 + X2, y ~ W1, data = sim$data, context.id = "school",
+    mcmc = list(burn.in = 0, n.iter = 100))
R> plot(mod, context.id = "school", term = c("X1", "X2"))
```

In this example, a variable `school` in the data contains the labels of the contexts: from `School A` up to `School E`. The panels in Figure 2 show the posterior distribution of all linear coefficients β for all contexts.

The package provides two additional plot functions, `plot_tau` and `plot_pexp_beta`, to display the effect of context-level features on the posterior expectation of the cluster coefficients β. The function `plot_tau` displays the posterior distribution of the parameter τ (see Model 3). Figure 3 shows the plot produced by the function `plot_pexp_beta`. The bottom row shows the posterior average of β_{kj} for each cluster k and context j. The top row shows the association between the context-level feature (W_1) and the posterior averages of the β_{kj}.

All the plot functions in the **hdpGLM** package return a `ggplot2` object. Therefore, users of **ggplot2** (Wickham 2016) can easily extend the plots and adjust its elements to fit their preferences.

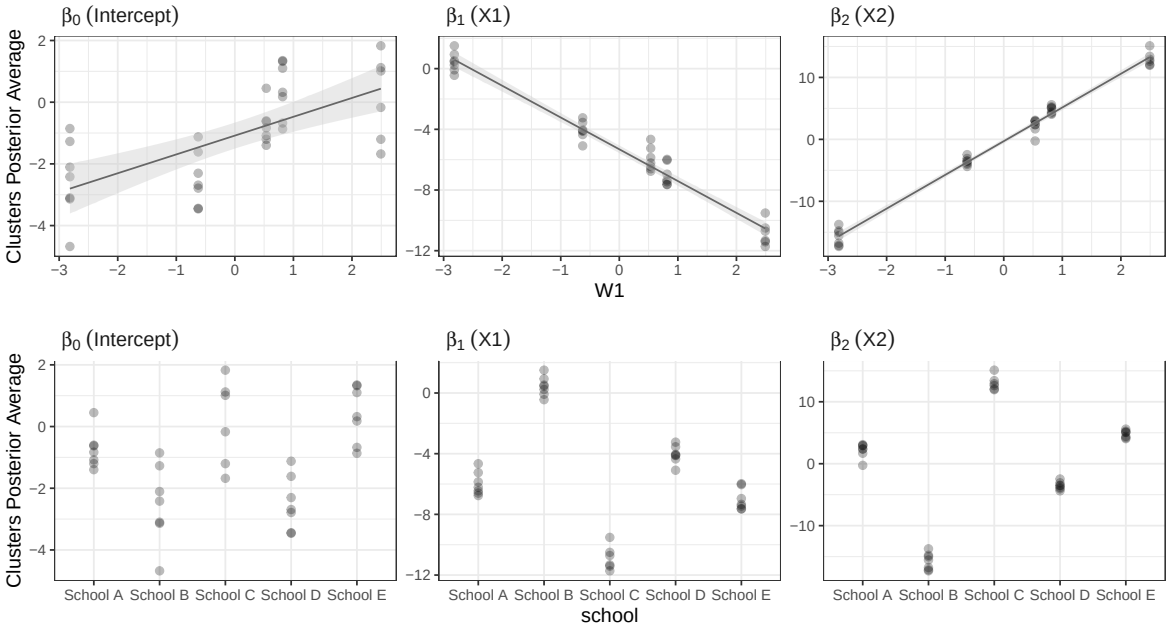

Figure 3: Posterior average of β for each context and cluster and the effect τ of cluster level features (W_1).

4.5. Classification/clustering

After the estimation, we can classify the data points into clusters. It is achieved using the function `classify`. The classification uses the posterior probability of the cluster membership for each data point. Each point is classified into the cluster it has the highest probability to belong. The function receives two arguments, `data` and `samples`. The former receives the `data.frame` used in the estimation. The latter uses the output of the estimation. It will return the same data frame with an additional column called `Cluster` indicating the cluster membership of each observation.

5. Analysis workflow

5.1. Applied example

Dirichlet Process Regression

This section shows a typical analysis workflow for applied research using the **hdpGLM** package. It assumes the goal of the analysis is to find latent heterogeneity in the association between the regressors and the outcome, and then cluster the sample based on that heterogeneity. From that standpoint, the typical analysis using **hdpGLM** follows the same pattern of other regression analyses in R, e.g., using the built-in `lm` R function. The only additional step for the **hdpGLM** is the MCMC convergence checks. The analysis starts with a set of observed

covariates and an outcome variable, selected by the researchers based on domain knowledge. Then, the researchers estimate the model and check the linear coefficients and their summary statistics. If one is using the `lm` function, that means looking at the p-values and 95% confidence intervals of the linear coefficients. With the `hdpGLM`, one checks the posterior mean, 95% highest posterior density (HPD) intervals for each cluster, and the number of clusters found. The example below illustrates this approach.

A classic problem in political science is understanding the effect of people's income, the level of inequality in their neighborhood, and their ideology on their attitudes toward welfare policies (Alt and Iversen 2017; Armingeon and Weisstanner 2021). Consider the toy dataset `welfare` provided with the **hdpGLM** package. The dataset contains four variables. The outcome is people's support for welfare policies (`support`). It is a function of levels of inequality in people's neighborhoods (`inequality`), their personal income (`income`), and political ideology (`ideology`).

```
R> data("welfare", package = "hdpGLM")
R> head(welfare)
```

```
    support inequality   income ideology
1 -18.56496    0.33927  0.14251  1.92260
2  -9.39058   -0.99066 -0.51171  0.24833
3   0.92762   -2.23185 -0.38563 -1.36192
4 -12.35945   -3.00795 -0.94406 -0.20887
5  -2.48344    0.10005  0.83222  0.13214
6 -11.41879   -0.95439 -0.88105  0.29164
```

We can use this data to estimate the effect of `inequality`, `income`, and `ideology` on people's `support` for welfare. Using a classical linear model and the R built-in function `lm`, one would have the following results:

```
R> mod <- lm(support ~ inequality + income + ideology, data = welfare)
R> summary(mod)
```

```
Call:
lm(formula = support ~ inequality + income + ideology, data = welfare)

Residuals:
   Min     1Q Median     3Q    Max
-7.098 -1.351  0.014  1.367  7.026

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.8445     0.0452  -85.10  < 2e-16 ***
inequality    0.2891     0.0449    6.43  1.6e-10 ***
income        3.8446     0.0442   87.01  < 2e-16 ***
ideology     -8.3094     0.0456 -182.17  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 2.02 on 1996 degrees of freedom
Multiple R-squared:  0.954,        Adjusted R-squared:  0.954
F-statistic: 1.39e+04 on 3 and 1996 DF,  p-value: <2e-16
```

Looking at the results, one would conclude that all covariates are significantly associated with the outcome. The variables `income` and `inequality` have a positive effect, while `ideology` has a negative one. However, latent heterogeneity is likely in these associations due to omitted factors such as education and gender (Bullock 2021). We can use the **hdpGLM** to search for latent subpopulations whose effect of the covariates on the response differs. Starting from the same set of covariates and the linear specification used above, one can simply run:

```
R> set.seed(777)
R> mcmc <- list(burn.in = 1000, n.iter = 5000)
R> hdpGLM(support ~ inequality + income + ideology, data = welfare,
+    mcmc = mcmc, K = 30)
R> summary(mod)

--------------------------------
dpGLM model object

Maximum number of clusters activated during the estimation: 15
Numer of MCMC iterations: 5000
burn-in: 1000
--------------------------------

Summary statistics of clusters with data points

--------------------------------
Coefficients for cluster 1 (cluster label 1)

             Post.Mean Post.Median HPD.lower HPD.upper
1 (Intercept)   -3.8714    -3.87189  -3.94199   -3.7978
2  inequality    1.9988     1.99843   1.93266    2.0719
3      income    3.8512     3.85160   3.77841    3.9232
4    ideology   -8.3070    -8.30627  -8.37800   -8.2364
5       sigma    1.0018     0.99971   0.91401    1.0872

--------------------------------
Coefficients for cluster 2 (cluster label 6)

             Post.Mean Post.Median HPD.lower HPD.upper
1 (Intercept)  -3.81892    -3.81934  -3.89457   -3.7484
2  inequality  -1.52476    -1.52559  -1.59384   -1.4548
3      income   3.88159     3.88182   3.82049    3.9541
4    ideology  -8.25600    -8.25560  -8.32457   -8.1826
5       sigma   0.97402     0.97472   0.86668    1.0796
```

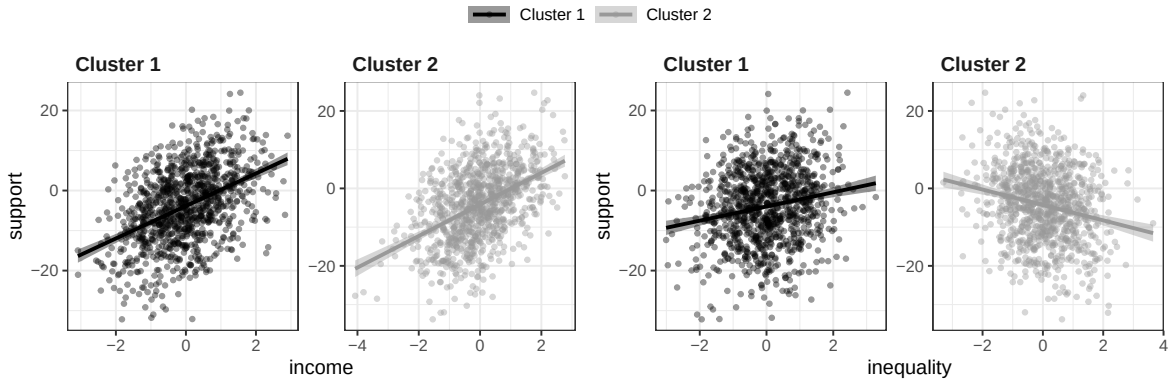

Figure 4: Association between income and support for welfare (left panels) and inequality and support for welfare (right panels) in each cluster

---

The `summary` function shows the posterior summaries. The interpretation is straightforward. The summary shows the posterior averages and 95% HPD intervals. There is a clear indication that the effect of inequality is heterogeneous in the sample, but not the effect of `income`. Users can go further and check the number of clusters with `nclusters(mod)`, classify the data using `classify(welfare2, mod)`, or plot the fitted values for a selected variable for each cluster, as shown in Figure 4 (code in reproducibility material; it uses **ggplot** and the output of `summary_tidy(mod)`).

As in any Bayesian analysis using MCMC, an important step is to check the MCMC convergence. The user can run MCMC diagnostics directly on the output of the `hdpGLM` function on the element `samples` of the list it returns. For instance, using the **coda** package (Plummer *et al.* 2006), users can run `geweke.diag(mod$samples)` and get the Geweke convergence check (Geweke 1992), or check the MCMC standard error (Flegal, Haran, and Jones 2008) with the package `mcmcse` and the function `mcse(mod$samples)` (Flegal, Hughes, Vats, Dai, Gupta, and Maji 2021). It is not recommended to use the Gelman-Rubin convergence diagnostic (Brooks and Gelman 1998) due to the multimodal nature of the posterior distribution of the linear coefficients. A good summary of various MCMC convergence diagnostics can be found in (Cowles and Carlin 1996), and the package **coda** implements the various convergence checks discussed in that article.

Discussions about which model to choose, e.g., between `lm` and `hdpGLM`, when the results differ is beyond the scope of this article. Readers are referred to Ferrari (2020), which demonstrates that in a variety of circumstances, if there is no latent subgroups in the sample, the posterior average of the linear coefficients estimated using the `hdpGLM` function are indistinguishable from those produced by the `lm` function. The example above shows this feature for the effect of `income`, as the clusters differ only on the effect of inequality (see also the simulated example on Section 5.2). When clusters are found, one option is to compare root mean squared error (RMSE) of the predictive values, which can be achieved using the function `predict`, available for the objects returned by both functions.

Hierarchical Dirichlet process regression

Following the same example, consider the data set `welfare2`, also included in the **hdpGLM** package. This new dataset includes the same variables than the `welfare` dataset, but from five different countries. It has index to indicate the country (`country`), and the country-level gender gap in country's provision of public good (`gap`). Here are the first rows of that dataset:

```
R> data("welfare2", package = "hdpGLM")
R> head(welfare2)
```

```
  support inequality income ideology country   gap
    <dbl>      <dbl>  <dbl>    <dbl>   <dbl> <dbl>
1 -18.6        0.339  0.143    1.92        0   0.1
2  -9.39      -0.991 -0.512    0.248       0   0.1
3   0.928     -2.23  -0.386   -1.36        0   0.1
4 -12.4       -3.01  -0.944   -0.209       0   0.1
5  -2.48       0.100  0.832    0.132       0   0.1
6 -11.4       -0.954 -0.881    0.292       0   0.1
```

We can estimate the effect of the country-level gender gap on the latent heterogeneity we found in the previous example, but now using the hierarchical DPP regression. It is done by running the following code:[1]

```
R> mcmc = list(burn.in = 1, n.iter = 50)
R> mod <- hdpGLM(support ~ inequality + income + ideology,
+    context.id = "country", support ~ gap, data = welfare2, mcmc = mcmc)
```

Either `summary` or `summary_tidy` will generate a numerical summary of the posterior samples. Using the `summary_tidy` function, we get a list of tidy summaries for β and τ. As explained in Section 4.3, the `summary_tidy` gives the summaries of the posterior distribution for the parameter β, but now for each context (country) as well. Additionally, the same summaries (posterior Mean and 95% HPD intervals) are provided for the parameter τ (see Model 3).

```
R> summary_tidy(mod)
```

```
$beta
# A tibble: 125 × 9
       k     j Parameter term           Mean  Median     SD
   <dbl> <dbl> <chr>     <chr>         <dbl>   <dbl>  <dbl>
 1     1     1 beta[0]   (Intercept) -3.82   -3.82   0.0702
 2     1     1 beta[1]   inequality  -1.40   -1.53   0.413
 3     1     1 beta[2]   income       3.87    3.87   0.0405
 4     1     1 beta[3]   ideology    -8.27   -8.28   0.0325
 5     1     1 sigma     sigma        0.947   0.951  0.0630
```

[1]In this toy example, I kept the MCMC small and didn't prioritize convergence diagnostics to facilitate replication. The goal is to focused on illustrating the package's functionalities and interpretation of the results. The outputs presented and those obtained with different MCMC sizes can be different unless it reaches convergence, as is usual in this type of estimation.

```
 6     1     2 beta[0]   (Intercept)  0.208   0.165  0.140
 7     1     2 beta[1]   inequality  -0.613  -0.609  0.121
 8     1     2 beta[2]   income      -0.319  -0.335  0.0688
 9     1     2 beta[3]   ideology    -1.76   -1.77   0.0739
10     1     2 sigma     sigma        0.0982  0.0954 0.0314
   HPD.lower HPD.upper
       <dbl>     <dbl>
 1   -3.93      -3.60
 2   -1.61       0.133
 3    3.80       3.95
 4   -8.34      -8.22
 5    0.805      1.05
 6    0.0150     0.463
 7   -0.810     -0.444
 8   -0.437     -0.209
 9   -1.90      -1.62
10    0.0634     0.110
# 115 more rows
# Use `print(n = ...)` to see more rows

$tau
# A tibble: 8 × 8
      w  beta Parameter Description                Mean    SD
  <int> <int> <chr>     <chr>                     <dbl> <dbl>
1     0     0 tau[0][0] Intercept of beta[0]     -0.213   1.26
2     0     1 tau[0][1] Intercept of beta[1]     -0.0778  1.40
3     0     2 tau[0][2] Intercept of beta[2]     -0.171   1.55
4     0     3 tau[0][3] Intercept of beta[3]     -0.622   1.56
5     1     0 tau[1][0] Effect of gap on beta[0] -0.0490  1.25
6     1     1 tau[1][1] Effect of gap on beta[1]  0.191   1.01
7     1     2 tau[1][2] Effect of gap on beta[2] -0.742   1.20
8     1     3 tau[1][3] Effect of gap on beta[3]  0.676   1.08
  HPD.lower HPD.upper
      <dbl>     <dbl>
1     -2.38      2.14
2     -2.64      2.26
3     -2.47      3.19
4     -3.75      2.42
5     -2.19      1.83
6     -1.88      2.09
7     -3.01      1.58
8     -1.25      2.51
```

Figure 5 shows the posterior distribution of the linear coefficients β for each context. The following command generated the plot:

```
R> plot(mod, ncol = 4, context.id = "country")
```

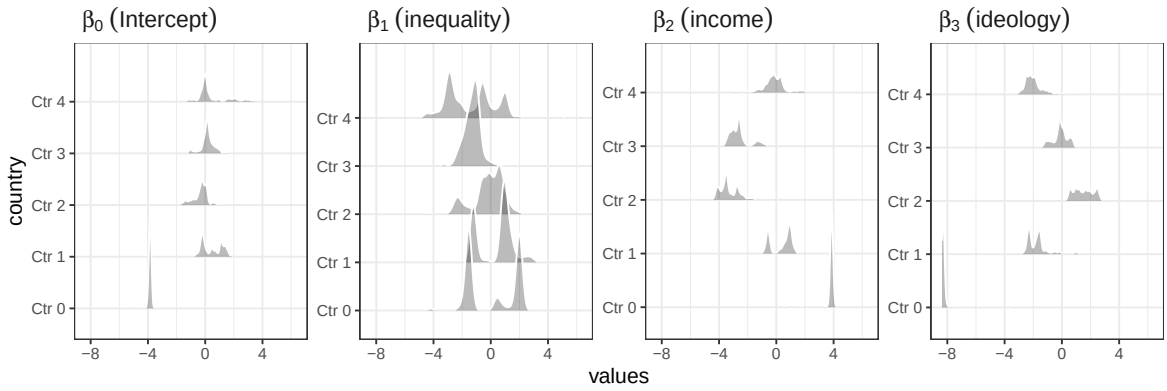

Figure 5: Posterior distribution of β for each context.

To illustrate the interpretation, in this toy example the effect of inequality is negative and homogeneous in country 3, as indicated by a unimodal distribution of β_1 on Figure 5, but there are latent subpopulations in countries 0, 1, 2, and 4. For some subpopulations in those countries, the effect of inequality is negative, but it is positive for other subgroups (as indicated by a multimodal distribution of β_1 in those contexts).

The parameter τ captures how the latent heterogeneity in each context is associated with the context-level features, `gap` in the example. We can plot the posterior distribution of τ using the function `plot_tau` or visualize its effect directly using the function `plot_exp_beta`. Figure 6 uses the function `plot_pexp_beta` to display the posterior averages of β in each context and cluster as function of `gap`.

To illustrate the interpretation, when the country gender gap in the provision of public goods (`gap`) increases, the posterior expectation of the effect of income decreases for all clusters (upper row, third column of Figure 6), but `gap` does not affect how inequality is associated with the outcome (support for welfare policies) across various clusters in different countries.

As discussed in the previous section, MCMC diagnostics can be conducted using external packages to check convergence of the chain. For instance, users can run `geweke.diag(mod$tau)` and `geweke.diag(mod$samples)` to get the Geweke convergence check (Geweke 1992), or use other diagnostic checks for MCMC objects.

5.2. Simulated example

Dirichlet process regression

This example estimates the DPP regression on Model 2 with a simulated data of 2,000 subjects divided into two clusters of equal size. The goal is to show how the `hdpGLM` function estimates cluster-specific effects even if the effect heterogeneity occurs only on one coefficient. Ferrari (2020) shows a comprehensive results for different numbers of clusters and heterogeneity in the effect of more than one covariate. In the simulation, I use three covariates and a Gaussian outcome. The three clusters differ only in the effect (β_2) of the first covariate (x_1). More precisely, the β coefficients are set to values according to Table 2:

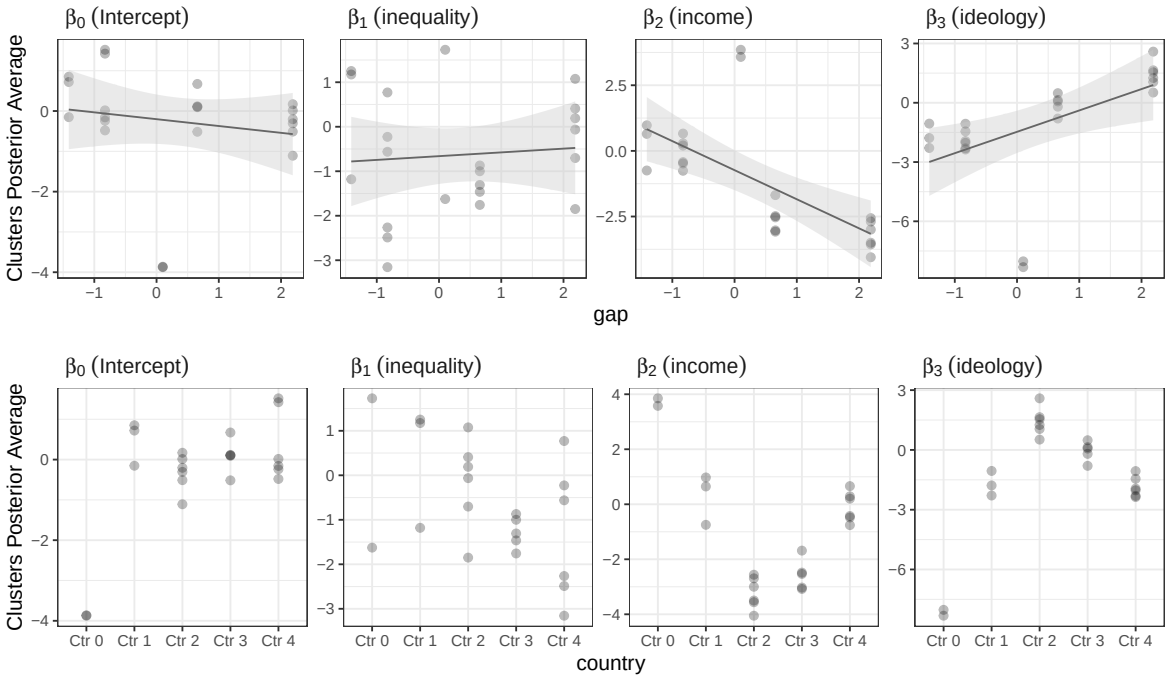

Figure 6: Posterior averages β for each cluster and context as a function of the context-level feature.

Coefficient	Value cluster $k = 1$	Value cluster $k = 2$
β_0	-0.15	-0.15
β_1	2.00	-1.5
β_2	9.90	9.90
β_3	3.90	3.90

Table 2: Values of the β coefficients in the simulated data.

We can estimate the DPP model using the following code:

```
R> mod <- hdpGLM(y ~ X1 + X2 + X3, data = df,
+    mcmc = list(burn.in = 500, n.iter = 10000))
R> summary(mod)

---------------------------------
dpGLM model object

Maximum number of clusters activated during the estimation: 17
Numer of MCMC iterations: 10000
burn-in: 500
---------------------------------
```

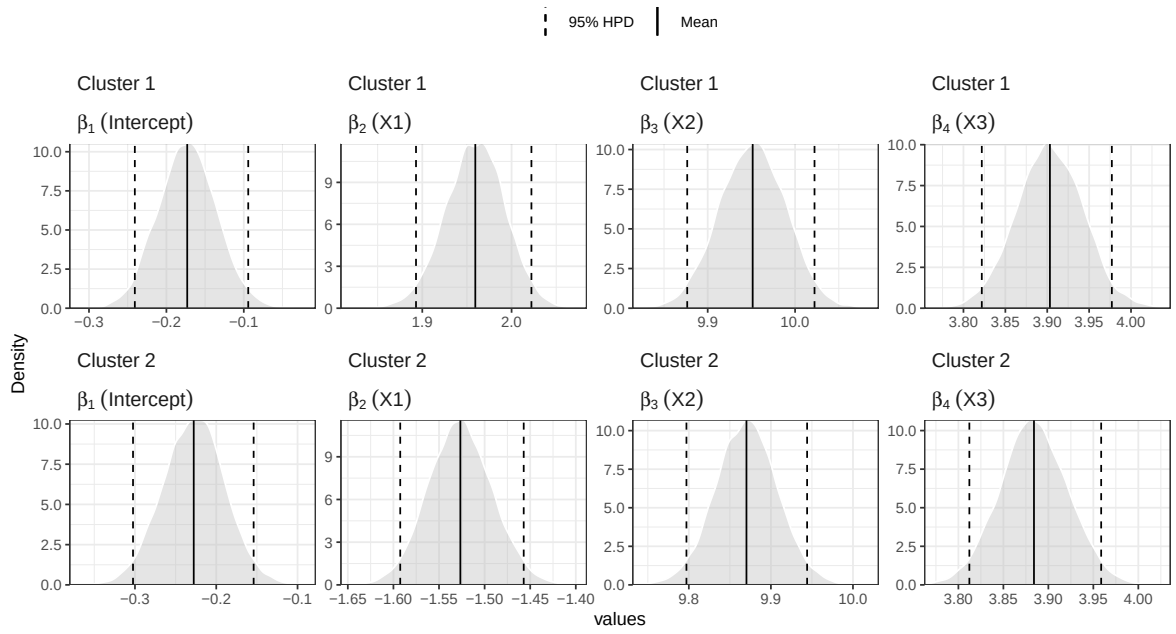

Figure 7: Posterior distribution of the β coefficients generated by the function `plot` of the package **hdpGLM** after the estimation using function `hdpGLM` and the simulated data whose true value of the coefficients are shown on Table 2.

```
Summary statistics of clusters with data points

--------------------------------
Coefficients for cluster 1 (cluster label 1)

             Post.Mean Post.Median HPD.lower HPD.upper
1 (Intercept)  -0.17301    -0.17297  -0.24077 -0.094259
2           X1   1.95931     1.95999   1.89786  2.027607
3           X2   9.95138     9.95170   9.87663 10.021926
4           X3   3.90324     3.90282   3.82230  3.977121
5        sigma   1.00128     0.99995   0.91356  1.085871

--------------------------------
Coefficients for cluster 2 (cluster label 2)

             Post.Mean Post.Median HPD.lower HPD.upper
1 (Intercept)  -0.22755    -0.22699  -0.30188  -0.15415
2           X1  -1.52658    -1.52689  -1.59253  -1.45713
3           X2   9.87029     9.87032   9.79561   9.94214
4           X3   3.88416     3.88378   3.81242   3.95890
5        sigma   1.00082     0.99989   0.91319   1.08996

--------------------------------
```

We can see that the results of the estimation recover the number of clusters used to generate the data, that is, $K = 2$. Moreover, the HPD intervals show that the clusters differ only in the coefficient β_1. The function `plot` returns a `ggplot2` object, and we can use it by running `plot(mod, separate = TRUE, ncol = 4)` to visualize the results of estimation, as shown on Figure 7.

To classify the data points into clusters, we run the function `classify`. In this example, it is `classify(data = df, samples = mod)`. It returns the dataset with a new column named `Cluster` that indicates the cluster membership of each observation.

Ferrari (2020) shows other examples. Note that this is the only package currently on CRAN capable of estimating the latent heterogeneity in the linear coefficients, as illustrated in this example.

6. Performance comparison between hdpGLM and PReMiuM

Table 3 compares the two packages currently available in R – **hdpGLM** and **PReMiuM** – that have similar purposes of estimating DPP regression models. The tests were run serially on an AMD EPYC 7702 CPU at 2.18 GHz on a 64-bit Debian Linux system with 514GB RAM.

The comparison was based on the time the MCMC algorithm takes to run 500 iterations using the packages' default MCMC parameters. A similar performance test was conducted by **PReMiuM** developers (Liverani *et al.* 2015), and I mimic their test here. The number of predictors and the size of the data was set in advance, but the number of clusters in the data was randomly selected in each test.

Table 3 shows how the packages scale with the number of subjects. The **hdpGLM** outperforms the **PReMiuM** software package in all runs when the outcome variable was Gaussian. The gap in performance increases substantially with the size of the data set. The **hdpGLM** is around five times faster when the data set had 1,500 sample points and around 17 times faster when there were 5,000 observations. The opposite happens when the outcome variable is binary. The **PReMiuM** software is faster than the **hdpGLM** in this case, but the gap does not increase as much with the size of the data.

It is important to note that the two packages have a few overlapping functionalities, but differ in many aspects (see Table 1). One important difference is that the **hdpGLM** investigates latent heterogeneity on all predictors, not only the intercept, as the **PReMiuM** does. This extra feature of the **hdpGLM** package requires a HMC update within Gibbs for non-Gaussian outcomes, as discussed, which accounts for differences in performance for binary outcomes. Nevertheless, the performance of the two packages for binary outcomes are not drastically different as they are for Gaussian outcomes.

There are two limitations of the **hdpGLM** package, which are more generally related to computational aspects of Gibbs sampling or MCMC. One is scalability; the other is that large moves on the optimization curve are limited due to the restricted changes in each sampling step. In terms of scalability, the estimation may require large memory and processing time for large data sets. Table 3 shows that the package performs well with moderate-size data, and it is much faster than other existing alternatives to estimate similar models, but performance may be an issue depending on the size of the data. There are alternative approaches for DP and hierarchical DP mixtures using sub-cluster splits (Chang and Fisher III 2013, 2014) and variational inference (Hughes and Sudderth 2013; Hughes, Kim, and Sudderth 2015),

Sample size	Number of predictors	Number of clusters	Software: **hdpGLM** (seconds)	Software: **PReMiuM** (seconds)	Ratio (**PReMiuM**/ **hdpGLM**)
Gaussian outcome variable					
1500	5	3	2.26	11.28	4.98
1500	10	4	2.43	11.29	4.63
1500	15	1	2.41	11.87	4.92
2500	5	1	3.13	34.41	10.98
2500	10	2	3.48	34.32	9.87
2500	15	4	4.17	35.19	8.44
5000	5	4	6.89	132.88	19.28
5000	10	4	7.58	133.81	17.65
5000	15	4	8.14	135.52	16.66
Binary outcome variable					
1500	5	1	99.60	11.10	0.11
1500	10	5	151.33	11.65	0.08
1500	15	5	100.39	12.32	0.12
2500	5	1	119.58	34.88	0.29
2500	10	4	167.58	35.54	0.21
2500	15	5	151.47	36.49	0.24
5000	5	3	201.62	132.82	0.66
5000	10	4	243.48	133.26	0.55
5000	15	3	395.95	140.12	0.35

Table 3: Comparing the time required to run 500 iterations in the two packages **hdpGLM** and **PReMiuM** R currently on CRAN to estimate DPR models. Note: The **hdpGLM** clusters data after investigating latent-heterogeneity on all linear coefficients, while the **PReMiuM** package models heterogeneity in the intercept term only.

but these models need to be adapted to be used with hierarchical DP mixtures of GLMs. In future versions, the package **hdpGLM** should explore these alternative sampling methods and adapt them to estimate the hdpGLM.

7. Discussion

DPP regression is an alternative to classical regression models. It can be used in a variety of situations when we suspect that there is heterogeneity in the effect of one or more covariates on the response variable. DPP is one of the core approaches in semi-parametric Bayesian regression models and unsupervised learning regression estimation (Abbring and Heckman 2007; Hastie, Tibshirani, and Friedman 2009; Hastie *et al.* 2009; Hannah *et al.* 2011; Traunmuller *et al.* 2015). This paper describes the R package **hdpGLM**, which implements a generalization of DPP regression models, developed in Ferrari (2020), for easy-to-use estimation, summary, and visualization of DPP regression by R users. Users can find further documentation details and examples at `http://www.diogoferrari.com/hdpGLM/`. The package is designed to facil-

itate its integration into other widely used external R packages, and it uses common R formula syntax for regression estimation. In addition to the functionalities built into the **hdpGLM** package to estimate, predict, summarize, and visualize the results, users can take advantage of other existing R packages to manipulate, analyze, report, or visualize the estimation results of DPP regression models implemented in the **hdpGLM** package, such as the package **coda** for MCMC diagnostics (Plummer *et al.* 2006). The outputs of summary and plot functions can be easily manipulated and extended for users of **ggplot2** (Valero-Mora 2010; Ginestet 2011; Wilkinson 2012; Wickham 2016; Gómez-Rubio 2017), or used to create tables and reports using package ecosystems such as **tidyverse** Wickham *et al.* (2019); Lee *et al.* (2020) or the R package **kableExtra** (Zhu 2021).

References

Aakvik A, Heckman JJ, Vytlacil EJ (2005). "Estimating Treatment Effects for Discrete Outcomes When Responses to Treatment Vary: An Application to Norwegian Vocational Rehabilitation Programs." *Journal of Econometrics*, **125**(1-2), 15–51. doi:10.1016/j.jeconom.2004.04.002.

Abbring JH, Heckman JJ (2007). "Econometric Evaluation of Social Programs, Part III: Distributional Treatment Effects, Dynamic Treatment Effects, Dynamic Discrete Choice, and General Equilibrium Policy Evaluation." *Handbook of Econometrics*, **6**, 5145–5303. doi:10.1016/s1573-4412(07)06072-2.

Alt J, Iversen T (2017). "Inequality, Labor Market Segmentation, and Preferences for Redistribution." *American Journal of Political Science*, **61**(1), 21–36. doi:10.1111/ajps.12264.

Angrist JD, Pischke JS (2008). *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press.

Armingeon K, Weisstanner D (2021). "Objective Conditions Count, Political Beliefs Decide: The Conditional Effects of Self-Interest and Ideology on Redistribution Preferences." *Political Studies*, **70**(4), 887–900. ISSN 0032-3217. doi:10.1177/0032321721993652.

Bates D, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Using **lme4**." *Journal of Statistical Software*, **67**(1), 1–48. doi:10.18637/jss.v067.i01.

Blei DM, Jordan MI (2006). "Variational Inference for Dirichlet Process Mixtures." *Bayesian Analysis*, **1**(1), 121–143. doi:10.1214/06-ba104.

Brooks SP, Gelman A (1998). "General Methods for Monitoring Convergence of Iterative Simulations." *Journal of Computational and Graphical Statistics*, **7**(4), 434–455. doi:10.1080/10618600.1998.10474787.

Bullock JG (2021). "Education and Attitudes toward Redistribution in the United States." *British Journal of Political Science*, **51**, 1230–1250. doi:10.1017/s0007123419000504.

Calin O, Chang DC (2006). *Geometric Mechanics on Riemannian Manifolds: Applications to Partial Differential Equations*. Springer-Verlag, Birkhauser, Boston.

Chang J, Fisher III JW (2013). "Parallel Sampling of DP Mixture Models Using Sub-Cluster Splits." In CJ Burges, L Bottou, M Welling, Z Ghahramani, KQ Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. URL https://proceedings.neurips.cc/paper/2013/hash/bca82e41ee7b0833588399b1fcd177c7-Abstract.html.

Chang J, Fisher III JW (2014). "Parallel Sampling of HDPs Using Sub-Cluster Splits." In Z Ghahramani, M Welling, C Cortes, N Lawrence, KQ Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. URL https://proceedings.neurips.cc/paper/2014/hash/149e9677a5989fd342ae44213df68868-Abstract.html.

Chen X (2007). "Large Sample Sieve Estimation of Semi-Nonparametric Models." *Handbook of Econometrics*, **6**, 5549–5632. doi:10.1016/s1573-4412(07)06076-x.

Cowles MK, Carlin BP (1996). "Markov Chain Monte Carlo Convergence Diagnostics: a Comparative Review." *Journal of the American Statistical Association*, **91**(434), 883–904. doi:10.1080/01621459.1996.10476956.

Dowle M, Srinivasan A (2023). ***data.table****: Extension of* `data.frame`. R package version 1.14.8, URL https://CRAN.R-project.org/package=data.table.

Ebbes P, Wedel M, Böckenholt U (2009). "Frugal IV Alternatives to Identify the Parameter for an Endogenous Regressor." *Journal of Applied Econometrics*, **24**(3), 446–468. doi:10.1002/jae.1058.

Ebbes P, Wedel M, Böckenholt U, Steerneman T (2005). "Solving and Testing for Regressor-Error (in) Dependence When No Instrumental Variables Are Available: With New Evidence for the Effect of Education on Income." *Quantitative Marketing and Economics*, **3**(4), 365–392. doi:10.1007/s11129-005-1177-6.

Eddelbuettel D, Balamuta JJ (2018). "Extending R with C++: A Brief Introduction to **Rcpp**." *The American Statistician*, **72**(1), 28–36. doi:10.1080/00031305.2017.1375990.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Ferrari D (2020). "Modeling Context-Dependent Latent Effect Heterogeneity." *Political Analysis*, **28**(1), 20–46. doi:10.1017/pan.2019.13.

Flegal JM, Haran M, Jones GL (2008). "Markov Chain Monte Carlo: Can We Trust the Third Significant Figure?" *Statistical Science*, **23**(2), 250–260. doi:10.1214/08-sts257.

Flegal JM, Hughes J, Vats D, Dai N, Gupta K, Maji U (2021). ***mcmcse****: Monte Carlo Standard Errors for MCMC*. R package version 1.5-0, URL https://CRAN.R-project.org/package=mcmcse.

Geweke J (1992). "Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments." In JM Bernardo, J Berger, AP Dawid, AFM Smith (eds.), *Bayesian Statistics*, 4 edition, pp. 169–193. Oxford University Press, Oxford.

Gill J, Casella G (2009). "Nonparametric Priors for Ordinal Bayesian Social Science Models: Specification and Estimation." *Journal of the American Statistical Association*, **104**(486), 453–454. doi:10.1198/jasa.2009.0039.

Ginestet C (2011). "**ggplot2**: Elegant Graphics for Data Analysis." *Journal of the Royal Statistical Society A*, **174**(1), 245–246. doi:10.1111/j.1467-985x.2010.00676_9.x.

Girolami M, Calderhead B (2011). "Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods." *Journal of the Royal Statistical Society B*, **73**(2), 123–214. doi:10.1111/j.1467-9868.2010.00765.x.

Gómez-Rubio V (2017). "**ggplot2** – Elegant Graphics for Data Analysis." *Journal of Statistical Software*, **77**(1), 1–3. doi:10.18637/jss.v077.b02.

Grimmer J (2009). "A Bayesian Hierarchical Topic Model for Political Texts: Measuring Expressed Agendas in Senate Press Releases." *Political Analysis*, **18**(1), 1–35. doi:10.1093/pan/mpp034.

Hadfield JD (2010). "MCMC Methods for Multi-Response Generalized Linear Mixed Models: The **MCMCglmm** R Package." *Journal of Statistical Software*, **33**(2), 1–22. doi:10.18637/jss.v033.i02.

Hannah LA, Blei DM, Powell WB (2011). "Dirichlet Process Mixtures of Generalized Linear Models." *Journal of Machine Learning Research*, **12**(Jun), 1923–1953. doi:10.1145/1015330.1015439.

Hastie T, Tibshirani R, Friedman J (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics, 2nd edition. Springer-Verlag.

Heckman JJ, Vytlacil EJ (2007). "Econometric Evaluation of Social Programs, Part II: Using the Marginal Treatment Effect to Organize Alternative Econometric Estimators to Evaluate Social Programs, and to Forecast Their Effects in New Environments." *Handbook of Econometrics*, **6**, 4875–5143. doi:10.1016/s1573-4412(07)06071-0.

Heinzl F, Tutz G (2013). "Clustering in Linear Mixed Models with Approximate Dirichlet Process Mixtures Using EM Algorithm." *Statistical Modelling*, **13**(1), 41–67. doi:10.1177/1471082x12471372.

Hughes M, Kim DI, Sudderth E (2015). "Reliable and Scalable Variational Inference for the Hierarchical Dirichlet Process." In *Artificial Intelligence and Statistics*, pp. 370–378. PMLR. URL https://proceedings.mlr.press/v38/hughes15.html.

Hughes MC, Sudderth E (2013). "Memoized Online Variational Inference for Dirichlet Process Mixture Models." In CJ Burges, L Bottou, M Welling, Z Ghahramani, KQ Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. URL https://proceedings.neurips.cc/paper/2013/hash/d490d7b4576290fa60eb31b5fc917ad1-Abstract.html.

Ichimura H, Todd PE (2007). "Implementing Nonparametric and Semiparametric Estimators." *Handbook of Econometrics*, **6**, 5369–5468. doi:10.1016/s1573-4412(07)06074-6.

Imbens GW, Rubin DB (2015). *Causal Inference in Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press.

Ishwaran H, James LF (2001). "Gibbs Sampling Methods for Stick-Breaking Priors." *Journal of the American Statistical Association*, **96**(453), 161–173. doi:10.1198/016214501750332758.

Ishwaran H, Zarepour M (2000). "Markov Chain Monte Carlo in Approximate Dirichlet and Beta Two-Parameter Process Hierarchical Models." *Biometrika*, **87**(2), 371–390. doi:10.1093/biomet/87.2.371.

Jara A, Hanson TE, Quintana FA, Müller P, Rosner GL (2011). "**Dppackage**: Bayesian Semi-and Nonparametric Modeling in R." *Journal of Statistical Software*, **40**(5), 1–30. doi:10.18637/jss.v040.i05.

Kleinman KP, Ibrahim JG (1998). "A Semiparametric Bayesian Approach to the Random Effects Model." *Biometrics*, **54**(3), 921–938. doi:10.2307/2533846.

Kyung M, Gill J, Casella G (2009). "Characterizing the Variance Improvement in Linear Dirichlet Random Effects Models." *Statistics & Probability Letters*, **79**(22), 2343–2350. doi:10.1016/j.spl.2009.08.024.

Kyung M, Gill J, Casella G (2010). "Estimation in Dirichlet Random Effects Models." *The Annals of Statistics*, **38**(2), 979–1009. doi:10.1214/09-aos731.

Lee S, Sriutaisuk S, Kim H (2020). "Using the **tidyverse** Package in R for Simulation Studies in SEM." *Structural Equation Modeling: A Multidisciplinary Journal*, **27**(3), 468–482. doi:10.1080/10705511.2019.1644515.

Liverani S, Hastie D, Azizi L, Papathomas M, Richardson S (2015). "**PReMiuM**: An R Package for Profile Regression Mixture Models Using Dirichlet Processes." *Journal of Statistical Software, Articles*, **64**(7), 1–30. ISSN 1548-7660. doi:10.18637/jss.v064.i07.

Martin AD, Quinn KM, Park JH (2011). "**MCMCpack**: Markov Chain Monte Carlo in R." *Journal of Statistical Software*, **42**(9), 22. doi:10.18637/jss.v042.i09.

Matzkin RL (2007). "Nonparametric Identification." *Handbook of Econometrics*, **6**, 5307–5368. doi:10.1016/s1573-4412(07)06073-4.

McCullagh P, Nelder JA (1989). *Generalized Linear Models*. 2nd edition. Chapman and Hall.

Mukhopadhyay S, Gelfand AE (1997). "Dirichlet Process Mixed Generalized Linear Models." *Journal of the American Statistical Association*, **92**(438), 633–639. doi:10.1080/01621459.1997.10474014.

Müller K, Wickham H (2023). ***tibble**: Simple Data Frames*. R package version 3.2.1, URL https://CRAN.R-project.org/package=tibble.

Neal RM (2000). "Markov Chain Sampling Methods for Dirichlet Process Mixture Models." *Journal of Computational and Graphical Statistics*, **9**(2), 249–265. doi:10.1080/10618600.2000.10474879.

Neal RM (2011). "MCMC Using Hamiltonian Dynamics." In *Handbook of Markov Chain Monte Carlo*, volume 2, chapter 11. Chapman & Hall.

Ng SK, McLachlan GJ, Wang K, Ben-Tovim Jones L, Ng SW (2006). "A Mixture Model with Random-Effects Components for Clustering Correlated Gene-Expression Profiles." *Bioinformatics*, **22**(14), 1745–1752. doi:10.1093/bioinformatics/btl165.

Pearl J (2009). *Causality: Models, Reasoning and Inference*. 2nd edition. Cambridge University Press.

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://CRAN.R-project.org/doc/Rnews/.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rossi P (2014). *Bayesian Non- And Semi-Parametric Methods and Applications*. Princeton University Press.

Rossi PE, Allenby GM, McCulloch R (2012). *Bayesian Statistics and Marketing*. John Wiley & Sons, Chichester, England.

Sethuraman J (1994). "A Constructive Definition of Dirichlet Priors." *Statistica Sinica*, **4**, 639–650. doi:10.21236/ada238689.

Smith BJ (2007). "**boa**: An R Package for MCMC Output Convergence Assessment and Posterior Inference." *Journal of Statistical Software*, **21**(11), 1–37. doi:10.18637/jss.v021.i11.

Spirling A, Quinn K (2010). "Identifying Intraparty Voting Blocs in the UK House of Commons." *Journal of the American Statistical Association*, **105**(490), 447–457. doi:10.1198/jasa.2009.ap07115.

Stegmueller D (2013). "Modeling Dynamic Preferences: A Bayesian Robust Dynamic Latent Ordered Probit Model." *Political Analysis*, **21**(3), 314–333. doi:10.1093/pan/mpt001.

Stroustrup B (2013). *The C++ Programming Language*. 4th edition. Addison-Wesley.

Sturtz S, Ligges U, Gelman A (2005). "**R2WinBUGS**: A Package for Running WinBUGS from R." *Journal of Statistical Software*, **12**(3), 1–16. doi:10.18637/jss.v012.i03.

Teh YW, Jordan MI, Beal MJ, Blei DM (2006). "Hierarchical Dirichlet Processes." *Journal of the American Statistical Association*, **101**, 1566–1581. doi:10.1198/016214506000000302.

Traunmuller R, Murr A, Gill J (2015). "Modeling Latent Information in Voting Data with Dirichlet Process Priors." *Political Analysis*, **23**(1), 1. doi:10.1093/pan/mpu018.

Valero-Mora PM (2010). "**ggplot2**: Elegant Graphics for Data Analysis." *Journal of Statistical Software*, **35**(b01). doi:10.18637/jss.v035.b01.

Verbeke G, Lesaffre E (1997). "The Effect of Misspecifying the Random-Effects Distribution in Linear Mixed Models for Longitudinal Data." *Computational Statistics & Data Analysis*, **23**(4), 541–556. doi:10.1016/s0167-9473(96)00047-3.

Villarroel L, Marshall G, Barón AE (2009). "Cluster Analysis Using Multivariate Mixed Effects Models." *Statistics in Medicine*, **28**(20), 2552–2565. doi:10.1002/sim.3632.

Walker SG (2007). "Sampling the Dirichlet Mixture Model with Slices." *Communications in Statistics-Simulation and Computation®*, **36**(1), 45–54. doi:10.1080/03610910601096262.

Wickham H (2014). "Tidy Data." *Journal of Statistical Software*, **59**(10), 1–23. doi:10.18637/jss.v059.i10.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. 3rd edition. Springer-Verlag. ISBN 978-3-319-24277-4. doi:10.1007/978-0-387-98141-3.

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the **tidyverse**." *Journal of Open Source Software*, **4**(43), 1686. doi:10.21105/joss.01686.

Wilkinson L (2012). "The Grammar of Graphics." In *Handbook of Computational Statistics*, pp. 375–414. Springer-Verlag.

Zhu H (2021). ***kableExtra**: Construct Complex Table with **kable** and Pipe Syntax*. R package version 1.3.4, URL https://CRAN.R-project.org/package=kableExtra.

Affiliation:

Diogo Ferrari
Department of Political Science
University of California, Riverside
2234 Watkins Hall
Riverside, California 92521, United States of America
E-mail: diogo.ferrari@ucr.edu
URL: https://diogoferrari.com/.

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
October 2023, Volume 107, Issue 10 *Submitted:* 2022-05-17
doi:10.18637/jss.v107.i10 *Accepted:* 2023-01-28