baker: An R package for Nested Partially-Latent Class Models

Irena Chen
University of Michigan, Ann Arbor

Qiyuan Shi
University of Michigan, Ann Arbor

Scott L. Zeger
The Johns Hopkins University, Baltimore

Zhenke Wu
University of Michigan, Ann Arbor

Abstract

This paper describes and illustrates the functionality of the **baker** R package. The package estimates a suite of nested partially-latent class models (NPLCM) for multivariate binary responses that are observed under a case-control design. The **baker** package allows researchers to flexibly estimate population-level class prevalences and posterior probabilities of class membership for individual cases. Estimation is accomplished by calling a cross-platform automatic Bayesian inference software JAGS through a wrapper R function that parses model specifications and data inputs. The **baker** package provides many useful features, including data ingestion, exploratory data analyses, model diagnostics, extensive plotting and visualization options, catalyzing communications between practitioners and domain scientists. Package features and workflows are illustrated using simulated and real data sets.

Keywords: Case-control studies, Latent class models, Measurement error, Markov chain Monte Carlo, R, JAGS.

1. Introduction

This paper introduces the **baker** R package that estimates a suite of nested partially-latent class models (NPLCM) for multivariate binary responses that are observed under a case-control design. There are five popular R packages that provide functionalities to perform latent class analysis and some extensions on Comprehensive R Archive Network (CRAN) Task View of "Cluster Analysis and Finite Mixture Models" (Leisch and Gruen 2022). Our software is unique in its contribution to provide models and associated diagnostic and plotting functions for conducting Bayesian latent class analyses using data collected under a case-control design, where the primary statistical goal is to estimate the population- and individual-level class mixing weights among the cases. In particular, functions in **baker** implement recent methodological developments in Wu, Deloria-Knoll, Hammitt, Zeger, and the PERCH Study Team (2016), Wu, Deloria-Knoll, and Zeger (2017), and Wu and Chen (2021). In practice, the package provides a simple interface that will allow researchers to reap the benefits of the NPLCMs via Markov chain Monte Carlo (MCMC) sampling without having to code the algorithms.

First formulated by Lazarsfeld (1950), latent class models (LCMs) have become an important tool for modeling multivariate discrete responses (e.g., Goodman 1974; Dunson and Xing 2009) and model-based clustering (e.g., Vermunt and Magidson 2002). LCMs and various extensions have been used as primary workhorses driving discoveries and improved predictions in numerous scientific fields including psychology (e.g., Xu 2017), sociology (e.g., McCormick, Li, Calvert, Crampin, Kahn, and Clark 2016), and public health (e.g., Stephenson, Herring, and Olshan 2019). There are currently several popular R packages that can perform general-purpose latent class analysis. The **poLCA** package developed by Linzer and Lewis (2011) provides a rich collection of functions to conduct latent class analysis for polytomous response variables and allows for the inclusion of regression variables to influence the class membership probabilities for each individual. Missing data is also handled under the assumption of missing at random. The **BayesLCA** package (White and Murphy 2014) provides functions for latent class analysis of multivariate binary responses in a Bayesian framework via expectation-maximization, MCMC, or variational Bayes. However, missing data is not handled in its current version (Version 1.9). In addition, we note that the models fitted by both **poLCA** and **BayesLCA** make a classical local independence (LI) assumption for the multiple responses given class membership, which may be violated in many real-world settings. The **randomLCA** package provides functions to fit LCMs with optional random effects that cause local dependence (LD), of which LI is a special case.

The **baker** package provides multiple novel advantages to existing software. First, **baker** enables case-control analyses with or without covariates in the NPLCM framework. The case-control design is vital to valid scientific inference in many large-scale clinical and biomedical applications. For example, in the largest pediatric pneumonia etiology study to date (PERCH Study Group 2019), biological samples are collected from subjects with clinically-defined disease ("cases") and subjects without disease ("controls"). Panel molecular diagnostic tests targeting multiple putative disease-causing agents may be performed on the collected samples, resulting in multivariate binary data under a case-control design. The control subjects have the observed class of not having the said disease. Their data serve as statistical control to estimate the measurement specificity when interpreting the error-prone test results in the cases.

Second, **baker** can fit models under deviations from the classical LI assumption in latent class analyses. Different from the continuous random effects approach taken in Qu, Tan, and Kutner (1996), the methodology implemented by **baker** uses a parallel factor decomposition with a stick-breaking prior to enable parsimonious approximation of potential LD between the multivariate binary responses given class membership. This enables faster computation and data-driven learning of empirical LD structures.

Third, **baker** is designed to handle multiple sets of case-control or case-only measurements of distinct degrees of measurement error. These measurements are classified into two broad types: 1) bronze-standard (BrS) data that are available for both cases and controls but with imperfect sensitivity or specificity; and 2) silver-standard (SS) data that are only available among cases, with perfect specificity but imperfect sensitivity. The **baker** can also work under missing data under the assumption of missing at random.

Finally, the **baker** package conducts full Bayesian inference via MCMC by calling a cross-platform and versatile automatic Bayesian software JAGS (Plummer *et al.* 2003; Plummer 2022) via a wrapper function `baker::nplcm()` that parses model specifications and data inputs (see Section 3 for the software design choice). The main quantities of interest are 1) the population-level class prevalences and 2) posterior probabilities of class membership for the individual cases. The **baker** package quantifies the uncertainty associated with these estimates and provides numerical and graphical summaries to assist in interpreting and communicating these results.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of the NPLCM framework as a case-control extension of the classical LCMs. The NPLCM likelihood and prior specifications are detailed in Section 2.3 without explanatory covariates; Section 2.4 covers the regression extension. Section 2.5 discusses model fitting and diagnostics. Section 3 gives a brief description of workflows and code underlying **baker**. This is followed in Section 4 by analyses of simulated and real data sets that demonstrate many of the package's features. Finally, Section 5 summarizes the main advantages of the **baker** package relative to existing software and future developments to expand its utility.

All figures in this paper can be reproduced by following the user vignette provided along with this article. The stable version of the package is available via CRAN (`https://CRAN.R-project.org/package=baker`); the development version can be accessed at `https://github.com/zhenkewu/baker`.

2. Model

2.1. Latent class models: A brief review

Latent class models (LCMs) for discrete latent and discrete manifest variables were developed and widely applied since the 1950s (e.g. Lazarsfeld 1950; Goodman 1974). LCMs constitute a family of distributions for correlated discrete measurements. The conventional LCM generally makes *local independence* (LI) assumption that manifest variables are independent of one another given the latent class. In the multivariate binary case, individual i's measurement vector, $\boldsymbol{M}_i = (M_{i1}, ..., M_{iJ})^\top$, is linked to her latent class (I_i) by the simple product likelihood $\mathbb{P}(\boldsymbol{M}_i \mid I_i = \ell, \boldsymbol{\theta}) = \prod_{j=1}^{J} \mathbb{P}(M_{ij} \mid I_i = \ell, \boldsymbol{\theta})$, where I_i takes value from $\{1, \ldots, L\}$ and $\boldsymbol{\theta}$ represents the collection of measurement parameters — sensitivities and specificities. We then obtain the observed likelihood by summing over all the possible values of I_i, i.e., $\mathbb{P}(\boldsymbol{M}_i \mid \boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{\ell=1}^{L} \pi_\ell \prod_{j=1}^{J} \mathbb{P}(M_{ij} \mid I_i = \ell, \boldsymbol{\theta})$, where $\boldsymbol{\pi}$ is a vector of mixing weights of length L. The LI assumption implies that the latent membership I_i completely explains the marginal dependence in $\boldsymbol{M}_i$. Under local identifiability conditions (Allman, Matias, and Rhodes 2009), we can estimate $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$ by the values that optimally reduce the observed dependence among measurements given latent class, e.g., through the expectation-maximization (EM) algorithms. Individual classification can then proceed by applying Bayes rule using the estimated parameters.

Below, we introduce the NPLCM family of models using case-control BrS measurements obtained from a single source (referred to as a "slice" in the **baker** package). In Section 3.1, we generalize the model to using multiple slices of BrS measurements, and to integrating case-only SS measurements which are special cases of BrS measurements having false positive rate of zero (perfect specificity).

2.2. Data structure and notation for case-control studies

Let $Y_i = 1$ indicate a case subject with the clinically-defined disease and $Y_i = 0$ indicate a control subject without disease. Let $\boldsymbol{M}_i = (M_{i1}, ..., M_{iJ})^\top \in \{0,1\}^J$ represent the multivariate binary case-control, non-gold-standard diagnostic test results from subject i. Let $\mathcal{D} = \{(\boldsymbol{M}_i, Y_i, \boldsymbol{X}_i Y_i, \boldsymbol{W}_i), i = 1, \ldots, N\}$ represent data, where $\boldsymbol{X}_i = (X_{i1}, \ldots, X_{ip})^\top$ are the p primary covariates in CSCF functions and hence must be available for cases, and $\boldsymbol{W}_i = (W_{i1}, \ldots, W_{iq})^\top$ are q covariates that are available in the cases and the controls. $\boldsymbol{X}_i$ and $\boldsymbol{W}_i$ may be identical, overlapping or completely different. $\boldsymbol{X}_i Y_i = \boldsymbol{X}_i$ for a case $Y_i = 1$; $\boldsymbol{X}_i Y_i$ is a vector of zeros for a control subject. For notational convenience, we have ordered the continuous variables, if any, in $\boldsymbol{X}_i$ and $\boldsymbol{W}_i$ as the first p_1 and q_1 elements, respectively. In this paper, we focus on pre-specified $\boldsymbol{X}_i$ and $\boldsymbol{W}_i$.

Classes defined by latent states

We first introduce notation for the true but unobserved latent classes (e.g, causes of disease) among the case subjects. Suppose a total of J "agents" or "items" are measured by the diagnostic tests. Let a binary variable ι_{ij} indicate whether ($\iota_{ij} = 1$) or not ($\iota_{ij} = 0$) the j-th agent caused case i's disease. We also allow more than one agent to cause the disease. We therefore have $\boldsymbol{\iota}_i = (\iota_{i1}, \ldots, \iota_{iJ})^\top \in \{0,1\}^J$ which is a vector of multiple binary indicators that represents the causes for subject i. We will also refer to $\boldsymbol{\iota}_i$ as "latent states" for case subject i. Note that we allow the all-zero latent states $\boldsymbol{\iota}_i = \mathbf{0}_{J\times 1}$ to represent a case with a "Not Specified" (NoS) cause. For example, in the Pneumonia Etiology Research for Child Health (PERCH) study, "NoS" can represent the subgroup of cases whose diseases are caused by agents not specified as molecular targets in the diagnostic tests (such as polymerase chain reaction, PCR). We will refer to cases having the same pattern of multivariate binary pattern $\boldsymbol{\iota}_i$ as belonging to the same "disease class" or "class" for short.

In this paper, we assume that there are L classes of *pre-specified* latent state patterns (possibly "NoS") among the cases. Let the set $\mathcal{A}$ comprise the *pre-specified* distinct multivariate binary patterns so that $|\mathcal{A}| = L$, where $|\mathcal{A}|$ is the cardinality of $\mathcal{A}$. We then introduce class indicators by arbitrarily labeling elements in $\mathcal{A}$ from 1 to L. We can now use I_i that takes value from $\{1, \ldots, L\}$ to indicate case subject i's class. We also let $\mathcal{C}_\ell = \{j : \iota_{ij} = 1, I_i = \ell, j = 1, \ldots, J\}$ represent the subset of causative agents for disease class ℓ; for the NoS class, we have $\mathcal{C}_{\mathsf{NoS}} = \emptyset$.

For a control i', we use $I_{i'} = 0$ to indicate $\boldsymbol{\iota}_{i'} = \mathbf{0}_{J\times 1}$, e.g., no lung infection in the PERCH study. For a case or a control, the value of I_i thus corresponds to a particular state pattern, so we can write $\boldsymbol{\iota}_i = \boldsymbol{\iota}_i(I_i)$.

To illustrate the scientific meaning of the notation, consider a hypothetical list of $J = 5$ species of pathogens ("items") in the context of PERCH study; they are targeted by the panel diagnostic tests. First, under the assumption that there are only single-pathogen causes and no NoS class, we have $L = J = 5$ disease classes with distinct patterns of $\boldsymbol{\iota}$:

$$\mathcal{A} = \{(1,0,0,0,0)^\top, (0,1,0,0,0)^\top, \cdots, (0,0,0,0,1)^\top\}.$$

We can label the five disease classes by $1, \ldots, L = 5$, so that, for example, $I_i = 2$ corresponds to $\boldsymbol{\iota}_i = (0, 1, 0, 0, 0)^\top$ and $\mathcal{C}_2 = \{2\}$, $I_i = 5$ corresponds to $\boldsymbol{\iota}_i = (0, 0, 0, 0, 1)^\top$ and $\mathcal{C}_5 = \{5\}$. Second, under a less restrictive assumption of single- or double-pathogen causes (still no NoS class), we have $L = \binom{J}{1} + \binom{J}{2} = 5 + 10 = 15$ disease classes. For example, cases with the first and the third pathogen infecting the lung are represented by $\boldsymbol{\iota}_i = (1, 0, 1, 0, 0)^\top$. It has the subset of causative agents $\mathcal{C}_\ell = \{1, 3\}$ where $I_i = \ell$ is an arbitrary integer label of the disease class.

2.3. Nested partially latent class models for case-control studies

Wu *et al.* (2016) and Wu *et al.* (2017) introduce a generalization to the latent class model in order to address two aspects of our particular setting. First, the latent classes are called "partially latent" since class membership is known for the subset of controls, but not cases. Second, the conditional independence assumption is relaxed by allowing for *nested subclasses* within each class. The inclusion of subclasses accounts for the possibility of correlation or dependence between measurements. The **baker** package implements this *nested partially latent class model* (NPLCM) framework. For ease of interepretation, we present the models by referring to terminologies in the PERCH study.

Likelihood

The NPLCM model likelihood can be specified via the following generative processes for the controls and the cases, respectively.

$$\begin{aligned} &\textsf{control subclass}: && Z_i \mid Y_i = 0 \sim \textsf{Categorical}_K\{\boldsymbol{\nu}\}, \boldsymbol{\nu} \in \mathcal{S}_{K-1}, && (1)\\ &\textsf{control data}: && M_{ij} \mid \iota_{ij} = 0, Z_i = k \sim \textsf{Bern}\left\{\psi_k^{(j)}\right\}, \text{ independently for } j = 1, ..., J, && (2)\end{aligned}$$

where $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_K)^\top$ is the vector of subclass probabilities and lies in a probability simplex. When $K = 1$, the model is referred to as PLCM (Wu *et al.* 2016). Let $\boldsymbol{\Psi} = \{\psi_k^{(j)} \in (0, 1)\}$ be a $J \times K$ matrix comprising false positive rates (FPRs), which are necessary for modeling the imperfect binary measurements among the controls. Let $\boldsymbol{\psi}^{(j)}$ and $\boldsymbol{\psi}_k$ represent the j-th row and k-th column. The data generating process for cases is as follows, with an additional Step (4) for drawing a subclass indicator Z_i for each case subject:

$$\begin{aligned} &\textsf{disease class}: && I_i \mid Y_i = 1 \sim \textsf{Categorical}_L\{\boldsymbol{\pi}\}, \boldsymbol{\pi} \in \mathcal{S}_{L-1}, && (3)\\ &\textsf{case subclass}: && Z_i \mid Y_i = 1 \sim \textsf{Categorical}_K\{\boldsymbol{\eta}\}, \boldsymbol{\eta} \in \mathcal{S}_{K-1}, && (4)\\ &\textsf{convert class to states}: && \boldsymbol{\iota}_i = \boldsymbol{\iota}_i(I_i) \in \mathcal{A}; &&\\ &\textsf{case data}: && M_{ij} \mid \iota_{ij}, Z_i = k, I_i = \ell, \sim \textsf{Bern}\left\{p_{k\ell}^{(j)}\right\}, \text{ independently for } j = 1, ..., J, && (5)\\ &\textsf{response probabilities}: && p_{k\ell}^{(j)} = \begin{cases} \theta_k^{(j)}, & \iota_{ij} = 1;\\ \psi_k^{(j)}, & \iota_{ij} = 0, \end{cases} \quad k = 1, \ldots, K, \text{ and } \ell = 1, \ldots, L. && (6)\end{aligned}$$

At Step (4), the NPLCM introduces K unobserved subclasses with weights $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_K)^\top$. The weights are shared across L disease classes. Let $\boldsymbol{\Theta} = \{\theta_k^{(j)} \in (0, 1)\}$ be a $J \times K$ matrix where $\theta_k^{(j)}$ represents the positive response probability in subclass k if item j is causative in a disease class. We also refer to $\theta_k^{(j)}$ as true positive rate (TPR) or sensitivity as in PLCM.

Let $\boldsymbol{\theta}^{(j)}$ and $\boldsymbol{\theta}_k$ represent the j-th row and k-th column. In Step (6), $p_{k\ell}^{(j)}$ represents the positive response probability of M_{ij} in subclass k of disease class ℓ, which equals the TPR $\theta_k^{(j)}$ for a causative pathogen and the FPR $\psi_k^{(j)}$ otherwise. We collect all the positive response probabilities for subclass k in disease class ℓ into $\boldsymbol{p}_{k\ell} = (p_{k\ell}^{(1)}, \dots, p_{k\ell}^{(J)})^\top$.

The population-level class prevalences in the cases are referred to as "cause-specific case fractions"(CSCF): $\boldsymbol{\pi} = (\pi_1, \dots, \pi_L)^\top$, which represent the population-level distribution of disease classes. $\boldsymbol{\pi}$ is the population-level class prevalences among the cases and is often of primary scientific interest. $\boldsymbol{\pi}$ is also referred to as cause-specific case fractions (CSCFs Wu and Chen 2021).

Prior

For NPLCM, we specify the prior distributions on unknown parameters as follows:

$$\begin{aligned}
\boldsymbol{\pi} &\sim \mathsf{Dirichlet}(a_1, \dots, a_L), & (7)\\
\psi_k^{(j)} &\sim \mathsf{Beta}(b_{1kj}, b_{2kj}), j = 1, ..., J; k \leq K, & (8)\\
\theta_k^{(j)} &\sim \mathsf{Beta}(c_{1kj}, c_{2kj}), j = 1, ..., J; k \leq K, & (9)\\
\eta_k &\sim U_k \prod_{s<k} [1 - U_s], \quad U_k \sim \mathsf{Beta}(1, \alpha_1), k < K; U_K = 1; & (10)\\
\nu_k &\sim V_k \prod_{s<k} [1 - V_s], \quad V_k \sim \mathsf{Beta}(1, \alpha_0), k < K; V_K = 1; & (11)\\
\alpha_0, \alpha_1 &\sim \mathsf{Gamma}(0.25, 0.25), & (12)
\end{aligned}$$

where prior independence is also assumed among these parameters. As discussed in more detail by Wu *et al.* (2016), the NPLCM likelihood similarly has the TPRs $\boldsymbol{\Theta}$ that are not fully identified by the model likelihood and hence is partially identified (Jones, Johnson, Hanson, and Christensen 2010). Therefore, we choose $(c_{1kj}, c_{2kj}), \forall k, j$, so that the 2.5% and 97.5% quantiles of the Beta distribution with parameters (c_{1kj}, c_{2kj}) match the prior minimum and maximum TPR values elicited from domain experts. Otherwise, we use the default value of 1s for the Beta hyperparameters. Hyperparameters for the etiology prior, $(a_1, ..., a_J)^\mathsf{T}$, are usually 1s to denote equal and flat prior weights for each disease class if expert prior knowledge is unavailable. Finally, in (10) and (11), we have specified truncated stick-breaking priors for both $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ that on average place decreasing weights on the kth subclass as k increases (Sethuraman 1994).

2.4. Regression extensions of NPLCM

Likelihood

An extension of the NPLCM allows for covariates to predict latent class membership by allowing the priors of the CSCFs and the subclass mixing weights to be a function of the observed explanatory variables. There may be biological or epidemiological support to include covariates in the likelihood function. For example, date of diagnosis may be informative if the disease of interest is known to have seasonal patterns.

We let the CSCFs depend on $\boldsymbol{X}_i$ by using a classical multinomial logistic regression:

$$\pi_{i\ell} = \pi_\ell(\boldsymbol{X}_i) = \exp\{\phi_\ell(\boldsymbol{X}_i)\} / \sum_{\ell'=1}^{L} \exp\{\phi_{\ell'}(\boldsymbol{X}_i)\}, \ell = 1, ..., L, \tag{13}$$

where $\phi_\ell(\boldsymbol{X}_i) - \phi_L(\boldsymbol{X}_i)$ is the log odds of case i in disease class ℓ relative to L: $\log \pi_{i\ell}/\pi_{iL}$. We treat all the disease classes symmetrically in this formulation, which simplifies the prior specification.

The regression extension assumes the control subclass weights are covariate-dependent:

$$\textsf{Extend (1) – control subclass}: \quad Z_i \mid \boldsymbol{W}_i, Y_i = 0 \sim \textsf{Categorical}_K\left\{\boldsymbol{\nu}_i\right\}, \boldsymbol{\nu}_i = \boldsymbol{\nu}(\boldsymbol{W}_i) \in \mathcal{S}_{K-1}, \tag{14}$$

where, as in NPLCM (Wu *et al.* 2017), the subclass indicators Z_i's are nuisance quantities for inducing dependence among the multivariate binary responses $\boldsymbol{M}_i$, but now given covariates. $\boldsymbol{\nu}_i = (\nu_{i1}, \ldots, \nu_{iK})^\top$ is the vector of control subclass probabilities that now may depend on $\boldsymbol{W}_i$. Scientifically, we are not interested in how the subclass probabilities are associated with covariates. We introduce $\boldsymbol{\nu}(\boldsymbol{W})$ here because, upon integrating over the distribution of Z_i in (14), it helps define a flexible conditional distribution of $\boldsymbol{M}_i$ given covariates $\boldsymbol{W}_i$.

For cases, we follow the case model for NPLCM, but extend in two aspects: let CSCFs depend on covariates $\boldsymbol{X}_i$ and let case subclass weight depend on covariates $\boldsymbol{W}_i$. That is,

$$\textsf{Extend (3) – disease class}: \quad I_i \mid \boldsymbol{X}_i, Y_i = 1 \sim \textsf{Categorical}_L\left\{\boldsymbol{\pi}_i\right\}, \boldsymbol{\pi}_i = \boldsymbol{\pi}(\boldsymbol{X}_i) \in \mathcal{S}_{L-1}, \tag{15}$$

$$\textsf{Extend (4) – case subclass}: \quad Z_i \mid \boldsymbol{W}_i, Y_i = 1 \sim \textsf{Categorical}_K\left\{\boldsymbol{\eta}_i\right\}, \boldsymbol{\eta}_i = \boldsymbol{\eta}(\boldsymbol{W}_i) \in \mathcal{S}_{K-1}, \tag{16}$$

where $\boldsymbol{\pi}(\boldsymbol{X}_i) = (\pi_1(\boldsymbol{X}_i), \ldots, \pi_L(\boldsymbol{X}_i))^\top$ are CSCF functions evaluated at $\boldsymbol{X}_i$, and $\boldsymbol{\eta}(\boldsymbol{W}_i) = (\eta_{i1}(\boldsymbol{W}_i), \ldots, \eta_{iK}(\boldsymbol{W}_i))^\top$ is the vector of case subclass probabilities evaluated at $\boldsymbol{W}_i$. Both $\boldsymbol{\pi}_i$ and $\boldsymbol{\eta}_i$ are quantities from probability simplexes.

Detailed regression specification

CSCF regression. We further assume additivity in a partially linear model:

$$\phi_\ell(\boldsymbol{x}; \Gamma_\ell^\pi) = \sum_{j=1}^{p_1} f_{\ell j}^\pi(x_j; \boldsymbol{\beta}_{\ell j}^\pi) + \widetilde{\boldsymbol{x}}^\top \boldsymbol{\gamma}_\ell^\pi, \tag{17}$$

where $\widetilde{\boldsymbol{x}}$ is the subvector of the predictors $\boldsymbol{x}$ that enters the model for all disease classes as linear predictors which may include an intercept, and $\Gamma_\ell^\pi = [(\boldsymbol{\beta}_{\ell 1}^\pi)^\top, \ldots, (\boldsymbol{\beta}_{\ell p_1}^\pi)^\top, (\boldsymbol{\gamma}_\ell^\pi)^\top]^\top$ is the vector of the regression coefficients for disease class ℓ. For covariates such as enrollment date that serve as proxy for factors driven by seasonality, non-linear functional dependence is expected. We approximate unknown functions of a standardized continuous variable such as $f_{\ell j}^\pi$ via basis expansions and along with a prior on the basis coefficients to encourage smoothness.

Control subclass weight regression. We specify ν_{ik} by logistic stick-breaking parameterization:

$$\nu_{ik} = g(\alpha_{ik}^\nu) \prod_{s<k} \{1 - g(\alpha_{is}^\nu)\}, \text{ if } k < K, \text{ and } \prod_{s<k} \{1 - g(\alpha_{is}^\nu)\} \text{ otherwise, where} \tag{18}$$

$$\alpha_{ik}^\nu = \alpha_k^\nu(\boldsymbol{W}_i = \boldsymbol{w}; \Gamma_k^\nu) = \mu_{k0} + \sum_{j=1}^{q_1} f_{kj}^\nu(w_j; \boldsymbol{\beta}_{kj}^\nu) + \widetilde{\boldsymbol{w}}^\top \boldsymbol{\gamma}_k^\nu, \text{ for } k = 1, \ldots, K-1. \tag{19}$$

Let $\Gamma_k^\nu = [(\boldsymbol{\beta}_{k1}^\nu)^\top, \ldots, (\boldsymbol{\beta}_{kq_1}^\nu)^\top, (\boldsymbol{\gamma}_k^\nu)^\top]^\top$ be the regression coefficients in the k-th subclass, and α_{ik}^ν is subject i's linear predictor at stick-breaking step $k = 1, \ldots, K-1$; $g(\cdot) : \mathbb{R} \mapsto [0,1]$ is a link function. In the **baker** package, we use the logistic function $g(\alpha) = 1/\{1 + \exp(-\alpha)\}$ which is consistent with (13) so that the priors of the coefficients Γ_k^ν and Γ_ℓ^π can be similar.

Case subclass weight regression. The case subclass weight curve $\boldsymbol{\eta}_k(\boldsymbol{W})$ is also specified via a logistic stick-breaking regression as in the controls but with different linear predictors α_{ik}^η: $\eta_{ik} = g(\alpha_{ik}^\eta)\prod_{s<k}\{1 - g(\alpha_{is}^\eta)\}$, $\forall k = 1, \ldots, K-1$; $\eta_{iK} = \prod_{s<K}\{1 - g(\alpha_{is}^\eta)\}$. Given $\boldsymbol{\Theta}$ and $\boldsymbol{\Psi}$, $\boldsymbol{\eta}_k(\boldsymbol{W})$ fully determines the joint distribution $[\boldsymbol{M} \mid \boldsymbol{W}, I = \ell \neq 0, \boldsymbol{\Theta}, \boldsymbol{\Psi}]$. We do not assume $\eta_k(\boldsymbol{w}) = \nu_k(\boldsymbol{w}), \forall \boldsymbol{w}$. Consequently, relative to the controls, the individuals in disease class ℓ may have different strength and direction of observed dependence between the causative $\{M_j : j \in \mathcal{C}_\ell\}$ and non-causative $\{M_j : j \notin \mathcal{C}_\ell\}$ pathogens, or between the non-causative pathogens. Let the k-th linear predictor

$$\alpha_{ik}^\eta = \alpha_k^\eta(\boldsymbol{W}_i = \boldsymbol{w}; \Gamma_k^\eta) = \mu_{k0} + \sum_{j=1}^{q_1} f_{kj}^\eta(w_j; \boldsymbol{\beta}_{kj}^\eta) + \widetilde{\boldsymbol{w}}^\top \boldsymbol{\gamma}_k^\eta, \tag{20}$$

where f_{kj}^η and f_{kj}^ν (from the control model) share the basis functions but the regression coefficients $\Gamma_k^\eta = [(\boldsymbol{\beta}_{k1}^\eta)^\top, \ldots, (\boldsymbol{\beta}_{kq_1}^\eta)^\top, (\boldsymbol{\gamma}_k^\eta)^\top]^\top$ differ from the control counterpart (Γ_k^ν). In addition, we have used the same intercepts $\{\mu_{k0}\}$ in (19) to ensure only important subclasses in the controls are used in the cases. For example, absent covariates $\boldsymbol{W}$, a large and positive μ_{k0} effectively halts the stick-breaking procedure at step k for the controls. This is because the k-th stick-breaking will take almost the entire remaining stick, resulting in ν_{k+1} that is approximately zero. Applying the same intercept μ_{k0} to the cases makes $\eta_{k+1} \approx 0$.

Section 4 provides examples of how to include discrete and continuous covariates in the model specification using the **baker** package.

Priors

The number of parameters in the model likelihood for the regression model $(\{\Gamma_\ell^\pi\}, \{\Gamma_k^\eta\}, \{\Gamma_k^\nu\}, \{\mu_{k0}\}, \boldsymbol{\Theta}, \boldsymbol{\Psi})$ is $\mathcal{O}(LC_{\max}p_1 + KC_{\max}q_1 + JK)$ where $C_{\max}$ is the maximum number of basis functions in $\{f_{\ell j}^\pi, f_{kj}^\nu, f_{kj}^\eta\}$. It easily exceeds the number of observed distinct binary measurement patterns. To overcome potential overfitting and increase model interpretability, we *a priori* encourage the following two features: (a) few non-trivial subclasses uniformly over $\boldsymbol{W}_i$ values, and (b) constant subclass weights over $\boldsymbol{W}_i$ values $\eta_k(\cdot) = \eta_k$ and $\nu_k(\cdot) = \nu_k$. See Wu and Chen (2021) for the exact technical specifications.

2.5. Posterior inference via MCMC

We perform posterior inference via Markov chain Monte Carlo (MCMC) algorithm that draws posterior samples of the unknowns to approximate their joint posterior distribution (Gelfand and Smith 1990). Flexible posterior inferences about any functions of the model parameters and individual latent variables are available by plugging in the posterior samples of the unknowns. All the models presented so far are available in the **baker** package. See Wu *et al.* (2016, 2017); Wu and Chen (2021) for details of the sampling algorithm.

3. Software: design features and main function

The Bayesian method for estimating population-level class prevalences and posterior probabilities of class memberships for individual cases is implemented by connecting R with another freely available cross-platform automatic Bayesian fitting program (JAGS 4.2.0 Plummer 2022). Figure 1 shows a schematic workflow that connects some **baker** functions and arguments to the steps in a data analysis or simulation pipeline. The **baker** package implements both exploratory and model-based analyses of case-control multivariate binary data. The package enables an analyst to organize multivariate binary diagnostic test results by their measurement standards (BrS or SS) and to calculate summaries such as the marginal positive rates for each item in the cases and controls; pairwise odds ratios can also be computed and visualized. The analyst can then specify which subsets of measurement data to use, the model likelihood, and the prior distributions for true positive rates and population-level CSCFs, among other model components. Based on these model specifications, R calls and instructs JAGS to fit the corresponding model to the data, performs model diagnostics, and stores the posterior results for ensuing inference of the key unknown quantities, such as the population-level class prevalences and class membership for individual cases. Finally, the package offers numerical and graphical summaries to display the evidence in the data and to facilitate model criticism.

The **baker** package uses the JAGS program to fit the specified NPLCMs; pre-installation of JAGS is required - the accompanying vignette contains detailed instructions about setting up JAGS for **baker** along with other required R package dependence.

`nplcm()` is the main function of the **baker** package and takes in three required arguments:

- `data_nplcm`: a named list containing data consisting of the following: 1) measurements `Mobs` - a named list containing `MBS` for BrS measurements and `MSS` for SS measurements, 2) case-control status `Y`, and 3) covariates `X`;
- `model_options`: a named list that specifies the data sources, model likelihood, and prior distributions for the model parameters;
- `mcmc_options`: a named list that specifies how to set up the MCMC sampling algorithm for posterior inference.

We will provide detailed examples for each of the three arguments in the next section. The output of `nplcm()` is an object of class `"nplcm"` which contains the path to where results were stored (accessible via `$DIR_NPLCM`) and the sampled values of model parameters which can be further manipulated via external posterior processing packages such as **coda** and **ggmcmc** (see Section 4.5). It is designed such that intermediate model results, model specifications, input data are retained for debugging and re-purposing for analyses not included in **baker**, such as post-stratification of CSCFs by discrete covarates (e.g., age group) using model results obtained from an NPLCM fitted without covariates. In addition, in high-performance computing, we may organize simulation settings by folder with proper names indicating the differences in the ground truth. Separate functions can be written and applied to these folders to obtain simulation results for various comparisons. Although the downside is the extra storage of results in the folder (and thus, the cost of additional disk memory), we believe that retaining this information is often more beneficial in complex substantive applications. Fortunately, generic functions in the **baker** package can read and organize these information if the fitted object is provided.

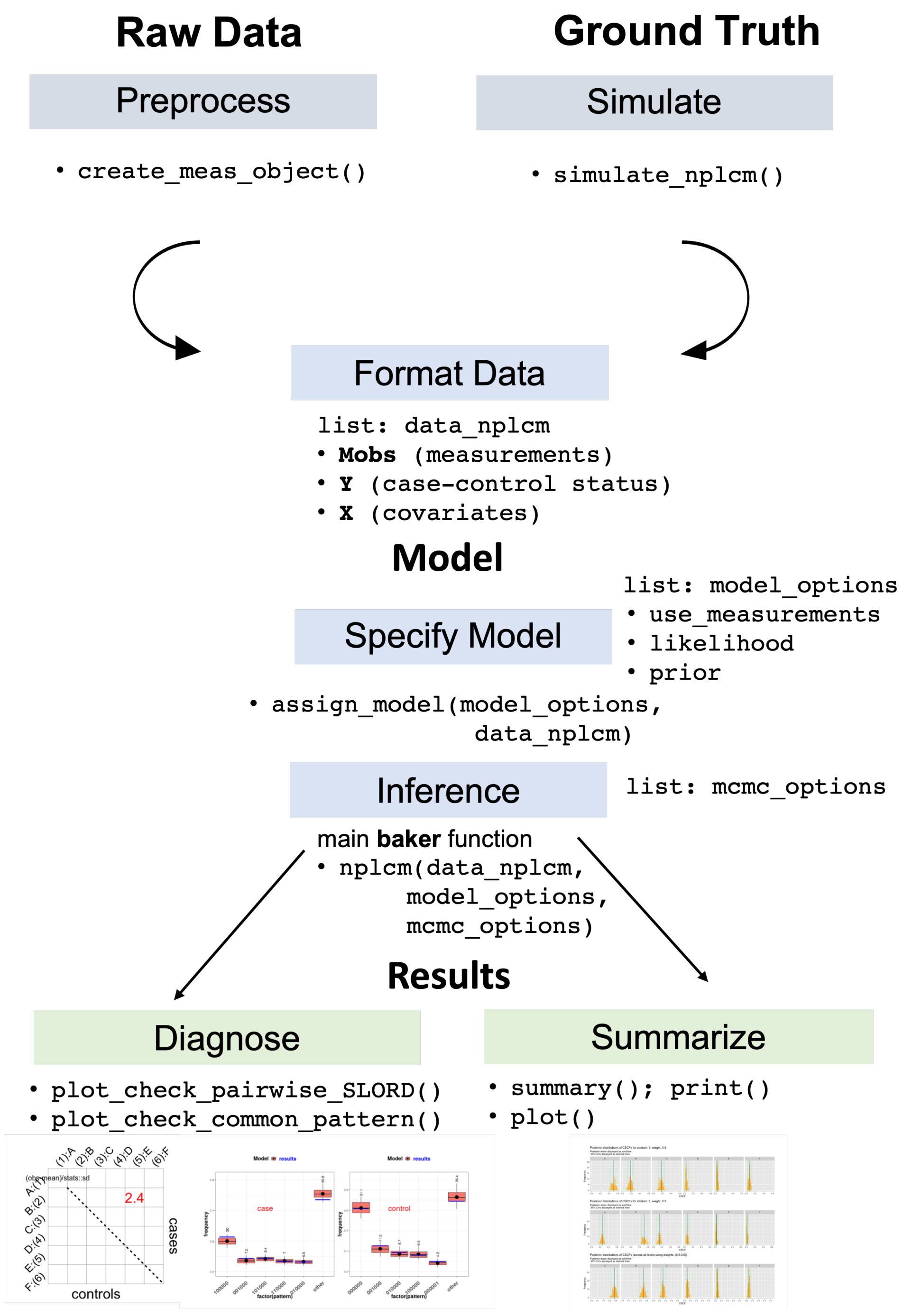

Figure 1: **baker** package workflow.

3.1. Multiple slices of BrS data and SS data

`nplcm()` can readily integrate more than one source of BrS measurements by supplying data to the argument `data_nplcm` and using data as specified in the argument `model_options`. For example, in the PERCH study, besides the NPPCR test for bacteria and viruses, pleural fluid PCR on the same set of pathogen targets may be performed; they are obtained from a different specimen with the same technology PCR and have different TPRs and FPRs. In the argument `data_nplcm` (a list), we can add these measurements to the list `data_nplcm$Mobs$MBS` which itself may contain multiple elements; we refer to each source of BrS measurements as a "slice". The **baker** package can integrate multiple slices of BrS measurements. In addition, case-only SS measurements may be available, e.g., blood culture results on the subset of bacteria. SS measurements are assumed to have perfect specificity, i.e., measurements on controls are assumed to never return positive results; SS data are by definition case-only. Similarly, one may add SS data into the list `data_nplcm$Mobs$MSS` which itself may contain multiple elements or "slices". The model likelihood with additional BrS and/or SS data will be modified automatically by the `nplcm()` function. In the argument `model_options` (a list), we simply set the element `use_measurements = "BrS"` (`"SS"`) to use all slices of the provided BrS (SS) data; setting `use_measurements = c("BrS","SS")` will use both BrS and SS data for model estimation. We illustrate these data source specifications in Section 4.3.

4. Illustrations

In this section, we give code snippets with explanations for data simulation, model specification, fitting, and numerical and graphical summaries of model results. By using simulated and a real data set, we illustrate the practical functionalities of the **baker** package. See the reproducible `RMarkdown` file provided along with this article for more illustrative examples of standard workflows.

Below, we first illustrate the three required arguments of the main function `nplcm()`: `data_nplcm` (Section 4.1), `model_options` (Section 4.3), `mcmc_options` (Section 4.4). Second, we fit specified models to illustrate the outputs of the main function `nplcm()`. Finally, we demonstrate how to use functions in the **baker** package to produce numerical and graphical summaries of the model results. Section 4.2 considers data simulation with covariates.

4.1. Setting up data inputs: Simulation and structure

In the following, we simulate and store BrS and SS measurement data to be used in the argument `data_nplcm`. `simulate_nplcm()` is a function that takes in the data generating parameters stored in a named list (e.g., `"set_parameter_noreg"` below) and outputs a data set containing 1) measurements (`Mobs`), which itself is a list with an element `MBS` that stores BrS case-control measurements and another element `MSS` that stores SS case-only measurements, and 2) case-control status (`Y`).

Here, we provide an example of relevant parameters that are useful in simulating data. We illustrate the values that these model parameters can take. We need to specify the number of items in a slice of bronze-standard measurements (`J.BrS = 7`), the number of items in a slice of silver-standard measurements (`J.SS`), and the number of subclasses (`K`). In the code snippet below, we simulate data for `Nd = 300` cases and `Nu = 300` controls. `cause_list` specifies the names of the true causes, with `etiology` specifying the population-level proportions of cases due to each cause (CSCFs). The BrS data used to infer the classes are case-control measurements on six items `c("A","B","C","D","E","F")` which in this example happens to be the exact set of agents that can cause the disease; in general, the measured items may include non-causative items or miss causative items. SS data are case-only and can measure fewer items targeted by BrS measurements. For the BrS data, true (`ThetaBS`) and false positive rates (`PsiBS`) for all the subclasses must be specified; they are of identical dimensions (`J.BrS` by `K`). In addition, one needs to specify two possibly different vectors of subclass weights for the case (`Eta`) and the control populations (`Lambda`), respectively; they determine the conditional dependence structure of the BrS measurements in the cases and the controls. For SS data, the false positive rates (`PsiSS`) must be all zeros indicating perfect specificity; the true positive rates (`ThetaSS`) can take positive values between 0 and 1. No subclass is assumed by the NPLCM models for SS data. We then use function `simulate_nplcm()` to produce a simulated data set:

```
R> J.BrS <- 6; J.SS <- 2; K <- 2
R> set_parameter_noreg <- list(Nd = 300,  Nu = 300,
+     cause_list      = c("A","B","C","D","E","F"),
+     etiology        = c(0.5,0.2,0.15,0.05,0.05,0.05),
+     meas_nm         = list(MBS = c("MBS1"),MSS=c("MSS1")),
+     pathogen_BrS    = c("A","B","C","D","E","F"),
+     PsiBS           = cbind(c(0.25,0.25,0.2,0.15,0.15,0.15),
+                             c(0.2, 0.2, 0.25,0.1,0.1,0.1)),
+     ThetaBS         = cbind(c(0.95,0.9,0.9,0.9,0.9,0.9),
+                             c(0.95,0.9,0.9,0.9,0.9,0.9)),
+     Eta             = t(replicate(J.BrS,c(0,1))),
+     Lambda          = c(0.5,0.5) ,
+     pathogen_SS     = c("A","B"),
+     PsiSS           = c(0,0,NA,NA,NA,NA),
+     ThetaSS         = c(0.15,0.1,NA,NA,NA,NA)
+ )
R> data_nplcm_noreg   <- simulate_nplcm(set_parameter_noreg)$data_nplcm
```

The data set can also be directly accessed by `data(data_nplcm_noreg)`. We can use the `summarize_BrS()` function to get summary statistics of the BrS measurements of the simulated data set. This function outputs the number of cases and controls, the observed marginal means for each measured item in the cases and the controls, along with the names of the measurements. The following line of code computes summaries for the single slice of the BrS data (`data_nplcm_noreg$Mobs$MBS[[1]]`) given the vector of case-control statuses (`data_nplcm_noreg$Y`); another function `summarize_SS()` can be used similarly for SS measurements:

```
R> summarize_BrS(data_nplcm_noreg$Mobs$MBS[[1]],data_nplcm_noreg$Y)
R> summarize_SS(data_nplcm_noreg$Mobs$MSS[[1]],data_nplcm_noreg$Y)
```

In addition to producing quick summary statistics, **baker** also provides functionalities to organize and store pertinent information about the BrS (or SS) measurements. For example, in the context of the PERCH study, the specimen name can be saved in the `specimen` argument, e.g., `"NP"` for a nasopharyngeal specimen. Another piece of information can be saved in the `test` argument, e.g. `"PCR"` (polymerase chain reaction) - here we use `"1"` for illustration; `quality` specifies the measurement quality (e.g. `"BrS"` or `"SS"`). The output of this function can be found in Appendix C.1.

```
R> BrS_object_1 <- make_meas_object(patho = set_parameter_noreg$pathogen_BrS,
+    specimen = "MBS", test = "1", quality = "BrS", cause_list =
+    set_parameter_noreg$cause_list)
R> SS_object_1 <- make_meas_object(patho=LETTERS[1:J.SS],
+   "MSS","1","SS",set_parameter_noreg$cause_list)
R> clean_options <- list(BrS_objects = make_list(BrS_object_1),
+   SS_objects  = make_list(SS_object_1))
```

4.2. Simulate data with covariates

In Appendix A, we provide 1) example code to simulate data `data_nplcm_reg_nest_strat` with two subclasses (“nested”) and two covariate strata; and 2) code to load a pre-simulated data set with a continuous covariate and a two-level discrete covariate: `data(data_nplcm_reg_nest)`. We refer the reader `simulate_nplcm()` in the help files of **baker** for more examples of how to simulate data with individual-level covariates.

4.3. Specifying models

We provide examples of `model_options` by specifying four models:

- `model_options_no_reg`;
- `model_options_no_reg_with_SS`;
- `model_options_reg_nest_strat`;
- `model_options_reg_nest`,

the first two of which does not perform regression and the last two perform regression. Various NPLCMs can be specified via three named lists:

- `use_measurements`: can be `"BrS"` or `"SS"` or `c("BrS","SS")` to represent the quality of the data sources used for model fitting;
- `likelihood`: a named list defining the model likelihood of the desired NPLCM;
- `prior`: a named list defining the prior distributions for the associated parameters.

Specify models without regression

In the code example below, `use_measurements = "BrS"` indicates that only the bronze-standard data are used in fitting an NPLCM. For specifying the likelihood, `k_subclass` indicates whether or not to use the conditional independence model (`k_subclass = 1` corresponds to the model with conditional independence given a cause, referred to as "non-nested model"). `Eti_formula` specifies the regression formula for relating the CSCFs to case covariates; `FPR_formula` specifies the regression formula for relating the subclass weights to covariates (must be common to case and control subjects). In terms of the prior distribution (`prior`), `Eti_prior` here specifies a numeric vector of length equal to the number of causes; this vector are Dirichlet hyperparameter for the population CSCFs. The `TPR_prior` specifies informative priors for the true positive rates; for example, we can specify a range `"0.55"` to `"0.99"` for `MBS1` measurement.

```
R> cause_list <- c("A","B","C","D","E","F")
R> model_options_no_reg <- list(use_measurements = c("BrS"),
+  likelihood   = list(cause_list = cause_list, k_subclass = 2,
+    Eti_formula = ~-1, FPR_formula = list(MBS1 = ~-1)),
+  prior= list(Eti_prior = overall_uniform(1,cause_list),
+    TPR_prior  = list(BrS = list(info  = "informative", input = "match_range",
+     val = list(MBS1 = list(up =  list(rep(0.99,J.BrS)),
+                           low = list(rep(0.55,J.BrS)))) ))))
```

The main function (`nplcm()`) will use another built-in function `assign_model()` to check the `model_options` argument against the `data_nplcm` argument and will return information about the desired NPLCM. Users can use this to check that they have set up their `model_options` correctly. In particular, the output from the following code snippet can be found in Appendix C.2.

```
R> assign_model(model_options_no_reg,data_nplcm_noreg)
```

In addition, we can specify a model that uses both BrS and SS data to fit an NPLCM without regression. To do this, we just need to modify the `data_nplcm` argument by adding SS data to the list `data_nplcm$Mobs$MSS` and specify a TPR prior for the SS data, e.g., `"0.01"` to `"0.5"` for `"MSS1"` measurements. This flexibility shows the package can work with multiple sources of data.

```
R> model_options_no_reg_with_SS <- model_options_no_reg
R> model_options_no_reg_with_SS$use_measurements <- c("BrS","SS")
R> model_options_no_reg_with_SS$prior$TPR_prior$SS <-
  list(info  = "informative", input = "match_range",
   val   = list(MSS1 = list(up = list(rep(0.5,length(SS_object_1$patho))),
                          low = list(rep(0.01,length(SS_object_1$patho))))))
```

Specify models for regression analyses

To set the `model_options` argument with regression covariates, we need to modify the `Eti_formula` and `FPR_formula` arguments. Here we use simulated data for illustration (see the final line of Appendix A). Recall that we do not let TPR vary by covariates in standard NPLCMs. In the following, because all the covariates are discrete, we specify symmetric Dirichlet priors with hyperparameters `1`s for the vector of CSCFs in each stratum of the covariate `SITE`. Other priors that have been set to defaults include the smoothness selection hyperparamters for the case and control FPRs (usually taken to be non-informative, given that FPRs can be estimated from the data).

```
R> model_options_reg_nest_strat <-  list(use_measurements = c("BrS"),
+  likelihood  = list(cause_list = cause_list,
+   k_subclass = 2, Eti_formula = ~ -1+as.factor(SITE),
+   FPR_formula = list(MBS1 =  ~ -1 + as.factor(SITE))),
+  prior= list(Eti_prior  = c(2,2),
+   TPR_prior  = list(BrS = list(info  = "informative", input = "match_range",
+     val = list(MBS1 = list(up =  list(rep(0.99,J.BrS)),
+                            low = list(rep(0.55,J.BrS)))) ))))
```

Again, we can check that we have set up the NPLCM correctly using `assign_model()`.

To fit the same model as above, but with an additional continuous covariate `"DATE"`, all we need to do is modify the regression formula:

```
R> model_options_reg_nest <- model_options_reg_nest_strat
R> model_options_reg_nest$likelihood$Eti_formula <-
+  ~ -1+s_date_Eti(DATE,Y,basis='ps',dof=7)+as.factor(SITE)
R> model_options_reg_nest$likelihood$FPR_formula <-
+   list(MBS1 =  ~ -1 +s_date_FPR(DATE,Y,basis = "ps",dof=5) + as.factor(SITE))
```

4.4. Setting up MCMC

Finally, we specify the `mcmc_options` argument of `nplcm()`, including the number of chains (`n.chains`), the number of total iterations (`n.itermcmc`), the number of burn-in iterations (`n.burnin`), the thinning interval (`n.thin`), whether or not individual level latent class predictions are desired (`individual.pred = TRUE` or `FALSE`) and whether or not to sample from the posterior predictive distributions (`ppd = TRUE` or `FALSE`). In addition, we must specify the path to the directory where the model should write the posterior samples (`result.folder`) and the path to the directory of the `.bug` model files (`bugsmodel.dir`).

```
R> thedir <- paste0(tempdir(),"_no_reg"); dir.create(thedir)
R> dput(data_nplcm_noreg,file.path(thedir,"data_nplcm.txt"))
R> dput(clean_options, file.path(thedir,"data_clean_options.txt"))
R> mcmc_options_no_reg <- list( n.chains = 3, n.itermcmc = 2000,
+   n.burnin = 1000, n.thin = 1, individual.pred = TRUE,
+   ppd = TRUE, result.folder = thedir, bugsmodel.dir = thedir)
```

Now that we have specified all the required arguments for `nplcm()`, we can fit our NPLCM using the following code:

```
R> nplcm_noreg <- nplcm(data_nplcm_noreg, model_options_no_reg,
+    mcmc_options_no_reg)
```

We can similarly obtain the other fitted models: `nplcm_noreg_with_SS`, `nplcm_reg_nest_strat`, and `nplcm_reg_nest`; see Appendix B for three separate uses of the main function `nplcm()`.

Because we use JAGS for automatic and versatile Bayesian inference of the models, we need to supply JAGS with `.bug` files with the requested forms of model likelihood and prior distribution. In the above code, `nplcm()` automatically interprets the specified model options and writes them in `.bug` files. In this case, `nplcm()` uses some internal function to generate a `.bug` model file for conditional independence models without regression, using BrS data. This model file will be stored in the path specified by `result.folder`. In addition, `fs::dir_tree(path = nplcm_noreg$DIR_NPLCM, recurse = TRUE)` lists all the resulting files in the folder storing model results. The output of `nplcm()` is an `S3` object of class `nplcm` which may be used by generic methods such as `summary()`, `print()`, and `plot()` (see Section 4.6).

4.5. Convergence and model diagnostics

After MCMC iterations are completed, we can assess convergence of the sampling chains for parameters using recorded information. The file `jagsdata.txt` contains all the data and pre-specified hyperparameters used when fitting the model. The files `CODAchain1.txt` and `CODAindex.txt` together record the posterior samples. Here we illustrate by using `pEti`, which stores the posterior samples of the CSCFs; we illustrate by focusing on model results obtained from an NPLCM without regression `nplcm_noreg`. Our package does not provide built-in functions to assess convergence and mixing of the sampling algorithm; in the following, we illustrate how to extract sampled values based on the outputs of `nplcm()`. However, **baker** provides built-in functions for performing posterior predictive checking. Posterior samples can be read into the `coda` format using the **coda** package. In particular, to retrieve the posterior samples, recall that the output from the `nplcm()` function contains the path to the directory where we stored the posterior samples and can be called using `$DIR_NPLCM`, as shown in the following example:

```
R> res_nplcm_noreg <- coda::read.coda(file.path(nplcm_noreg$DIR_NPLCM,
"CODAchain1.txt"), file.path(nplcm_noreg$DIR_NPLCM,"CODAindex.txt"),quiet=TRUE)
R> get_res   <- function(x,res) res[,grep(x,colnames(res))]
R> res <- get_res("pEti",res_nplcm_noreg)
```

Convergence diagnostics

The posterior samples obtained above can be further manipulated using external packages that provide algorithm convergence diagnostic functionalities. For example, `coda::raftery.diag(mcmc.list(res))` produces the Raftery diagonistic for convergence and the effective sample sizes for our model parameters. As another example, `ggmcmc::ggs_traceplot(ggmcmc::ggs(res))` plots the sampling trajectories for parameters of interest. See Brooks, Gelman, Jones, and Meng (2011) for a more complete review of approaches and considerations for convergence assessment.

Model diagnostics

The **baker** package provides two useful posterior predictive checking functions. For each slice of BrS measurement data:

- Standardized log odds ratio differences (SLORD); this is based on pairwise associations among the BrS measurements: near-zero SLORDs indicate the association is adequately characterized by the model;

- Probabilities of multivariate binary patterns with the highest frequencies; this is based on all the dimensions of measurements, not just pairwise, hence providing additional capability to assess the adequacy of the model in capturing the observed frequencies of multiple binary patterns.

Because the posterior predictive samples are stored in `$DIR_NPLCM` (when `ppd = TRUE` was set in the MCMC options), one can read in these samples and perform desired posterior predictive checking with other statistics if desired.

First, the following code snippet outputs a figure of posterior predicted pairwise LOR (log-odds ratios) compared with the observed LOR for the BrS data.

```
R> plot_check_pairwise_SLORD(nplcm_noreg$DIR_NPLCM,slice=1)
```

Second, the following code snippet produces a figure shown in Figure 6 of Appendix based on the `S3` object of class `nplcm` (`"nplcm_noreg"`), for the first slice of the bronze-standard data (`slice_vec = 1`), and `npat = 5` patterns with the highest frequency in the observed data:

```
R> plot_check_common_pattern(list(nplcm_noreg$DIR_NPLCM),slice_vec=1,n_pat=5)
```

4.6. Summary, plot, print

The **baker** package provides `summary()` methods to produce quick numerical summaries of the model specifications, posterior means and 95% credible intervals for population CSCFs. For example, `summary(nplcm_no_reg)` shows information about the result obtained from an NPLCM without regression; the actual outputs are shown in the Appendix C.3.

The generic `plot()` methods in **baker** produces graphical summaries of the fitted model according to the type of NPLCMs fitted. For an NPLCM without regression, this will produce a multi-paneled figure that summarizes the data, prior and posterior. In our experience, the figure facilitates answering questions like "where does the information come from?", e.g., how the BrS and SS data summaries are consistent with the obtained posterior inferential results for each cause. Figure 2 displays the output of the following code snippet. The detailed explanation of the details in the resulting figures can be found in the user manual of the **baker** package:

```
R> plot(nplcm_noreg_with_SS, bg_color = NULL)
```

In the case of an NPLCM with one or more discrete covariates for CSCF regression (without continuous covariates), `plot()` will visualize the posterior distributions of the CSCFs for each covariate stratum. Figure 3 displays the output of the following code snippet:

```
R> plot(nplcm_reg_nest_strat,show_levels = c(0,1,2))
```

The above code produces a figure for each stratum (the final row is for the overall CSCF estimates as a weighted average across strata). To only plot the marginalized posterior distributions, we can set `show_levels = 0`.

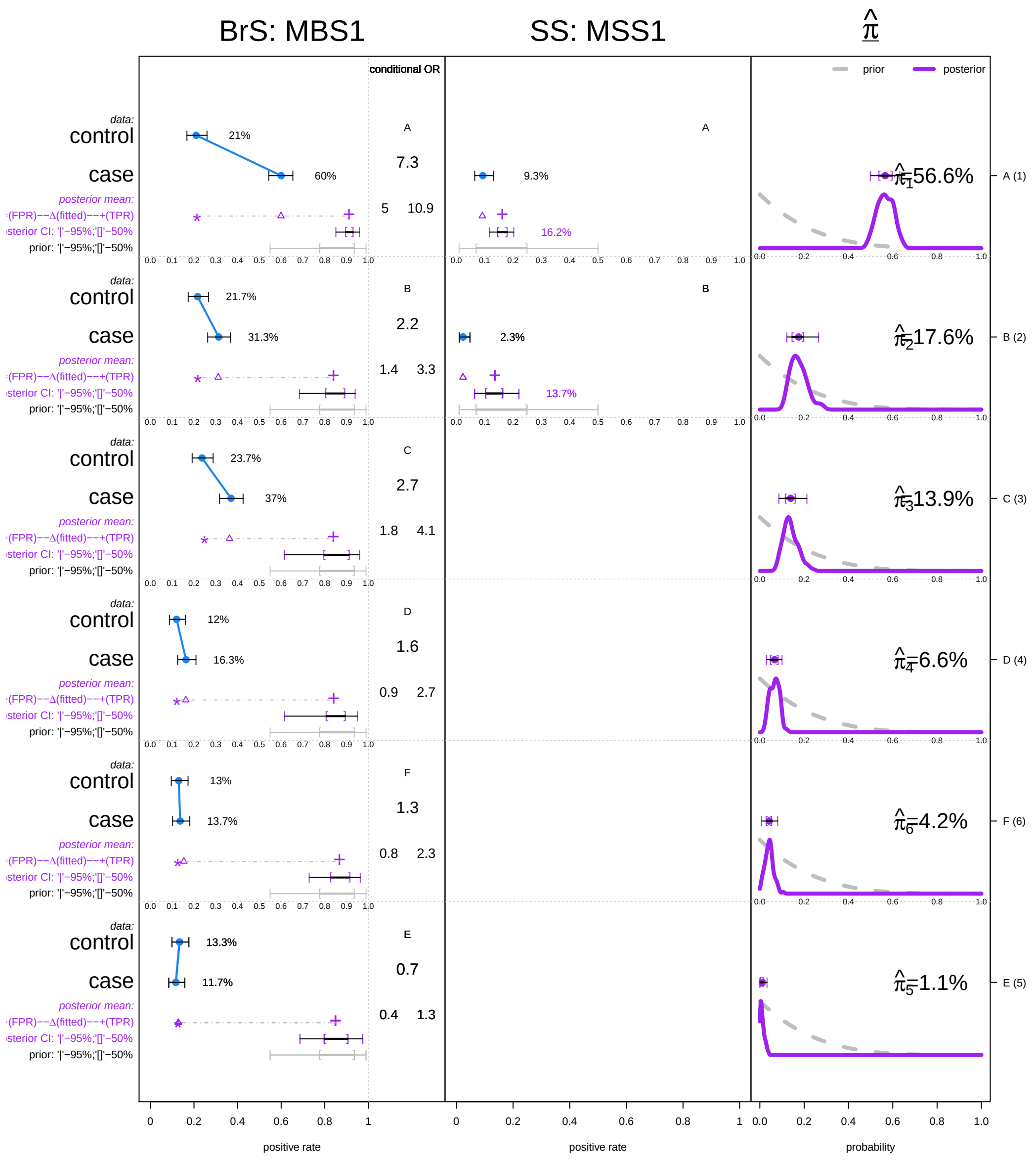

Figure 2: Output from `plot` for an `nplcm` object `"nplcm_no_reg_with_SS"`. The data, prior, and posterior summaries are displayed for each latent class in the rows; see Wu *et al.* (2016, Figure 3,) for additional figure descriptions.

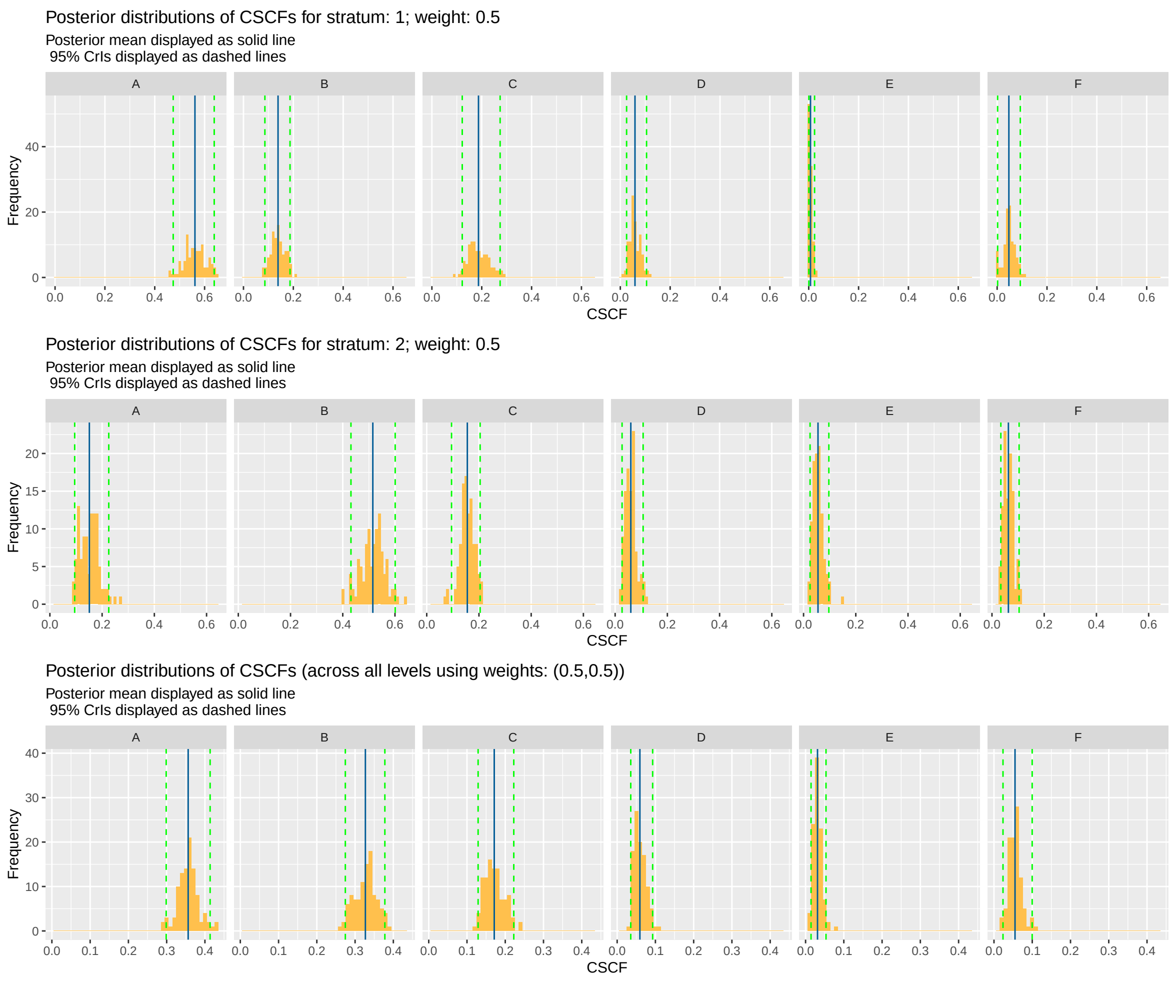

Figure 3: A graphical summary from `plot` of a fitted `nplcm` object `"nplcm_reg_nest_strat"` with a two-level discrete covariate. The final row shows the marginal posterior distributions for causes `"A"` to `"F"` where stratum weights are 0.5 and 0.5; the weights default to empirical weights and can be user-specified. The other rows show the marginal posterior distributions of the CSCFs for each stratum.

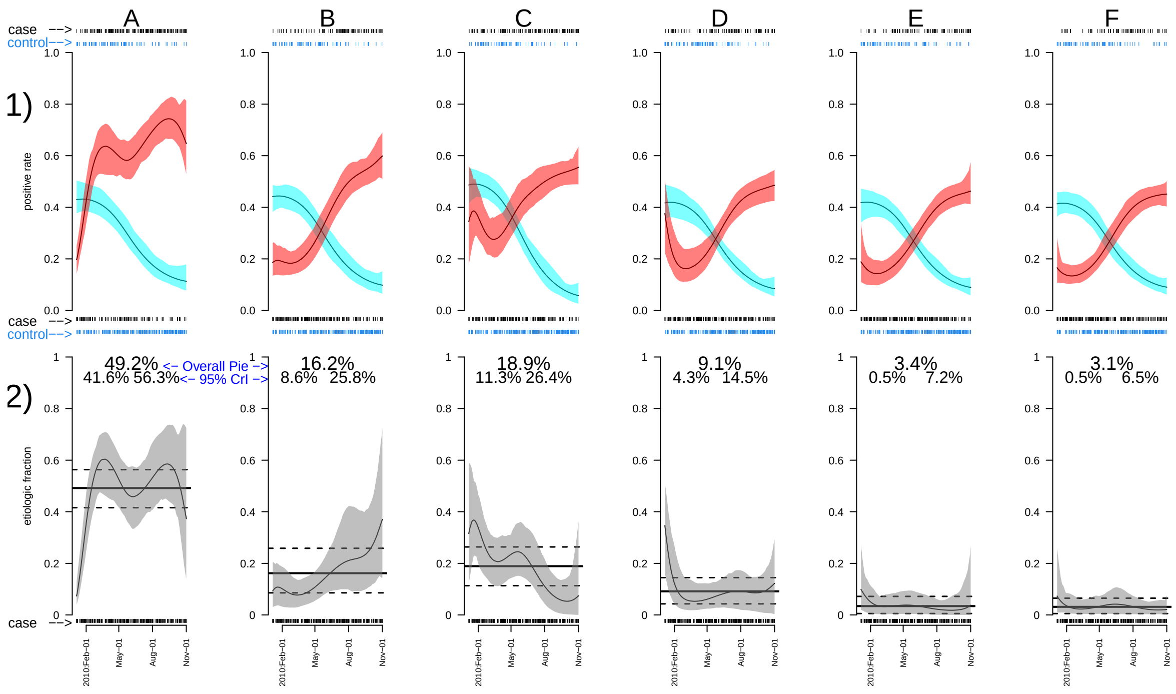

Figure 4: Output from `plot` for an `nplcm` object `"nplcm_reg_nest"` with a continuous regression covariate (seasonality). The top plot displays the estimated TPR for each latent class and the bottom plot displays the estimated CSCFs for each latent class. See for detailed description of the figure in Wu and Chen (2021).

In the case of an NPLCM with a continuous covariate, the `plot()` will produce a figure with two rows: one of the estimated marginal positive rates for cases and controls and one of the CSCFs for each disease class among the cases. We show an example of this plot in Figure 4 where CSCFs may vary by enrollment date (x-axis). If we have discrete covariates as well as a continuous covariate in the CSCF regression, we can make these plots for each stratum of the discrete covariates by including the stratum as an argument in the `plot` function:

```
R> DISCRETE_BOOL <- data_nplcm_reg_nest$X$SITE == 1
R> plot(nplcm_reg_nest, stratum_bool = DISCRETE_BOOL)
```

4.7. PERCH study example

Pneumonia is a clinical condition associated with infection of the lung tissue, which can be caused by multiple species of pathogens. In studies of pneumonia etiology, a cause is the subset of one or more pathogens infecting the lung. Knowledge about population-level cause-specific etiologic contributions can help prioritize prevention programs and design treatment algorithms. The Pneumonia Etiology Research for Child Health (PERCH) study is a seven-country case-control study of the etiology of severe and very severe pneumonia. (PERCH Study Group 2019) The primary aim of the study is to estimate the etiologic contributions quantified by *cause-specific case fractions* (CSCFs), which may vary by individual-level factors such as age, disease severity, nutrition status and human immunodeficiency virus (HIV) status.

In the PERCH study, tabulating case frequencies by cause is infeasible, because the lung-infecting pathogen(s) can rarely be directly observed due to potential clinical complications associated with invasive lung aspiration procedure (PERCH Study Group 2019). As an alternative, a non-invasive real-time polymerase chain reaction (PCR) test was made on each case's nasopharyngeal (NP) specimen, outputting presence or absence of a list of pathogens in the nasal cavity. The NP multivariate binary measurements are imprecise indicators for what pathogens infected the lung. In particular, detecting a pathogen in a case's nasal cavity does not indicate it caused lung infection. To provide statistical control for false positive detections, the PERCH study also performed NPPCR tests on pneumonia-free controls.

We illustrate with a regression analysis with 518 cases and 964 controls from one of the PERCH study sites in the Southern Hemisphere that collected more complete information on age (dichotomized to younger or older than one year), HIV status (positive or negative), disease severity for cases (severe or very severe), and presence or absence of seven species of pathogens (five viruses and two bacteria, representing a subset of pathogens evaluated) in NPPCR. The names of the pathogens and the abbreviations are (i) bacteria: *Haemophilus influenzae* (HINF) and *Streptococcus pneumoniae* (PNEU), (ii) viruses: adenovirus (ADENO), human metapneumovirus type A or B (HMPV_A_B), parainfluenza type 1 virus (PARA_1), rhinovirus (RHINO), and respiratory syncytial virus (RSV). We also include in the analysis the case-only, perfectly specific but imperfectly sensitive blood culture (BCX) diagnostic test results for two bacteria from cases only. For BCX data, we assume perfect specificity which is guided by the fact that if a pathogen did not infect the lung, it cannot be cultured from the blood (so we do not need control data to estimate the specificities). Detailed analyses of the entire data are reported elsewhere (The PERCH Team 2019).

Since the PERCH study data is not yet public and freely accessible, we provide example code of running `nplcm()` with model specifications (`perch_model`) and model fitting options (`perch_mcmc`). To more fully illustrate the functionality of **baker**, we use a pre-run posterior analysis of PERCH data and demonstrate numerical and graphical summaries of a simple regression analysis using **baker**. Sampling for this model in JAGS took approximately 5 minutes on a machine with an Quad-Core Intel i7 2.9 GHZ CPU and 16 GB of RAM running OSX Version 12.3 Beta.

First, we show the structure of the input data:

```
R> str(perch_data)
# List of 3
# $ Mobs:List of 2
# ..$ MBS:List of 1
# .. ..$ NPPCR:'data.frame': 1488 obs. of  7 variables:
#   .. .. ..$ HINF    : int [1:1488] 0 1 0 0 1 1 0 0 0 0 ...
# .. .. ..$ PNEU    : int [1:1488] 0 0 0 0 0 0 0 0 0 0 ...
# .. .. ..$ ADENO   : int [1:1488] 0 0 0 0 NA NA NA NA 0 1 ...
# .. .. ..$ HMPV_A_B: int [1:1488] 0 0 0 0 NA NA NA NA 0 0 ...
# .. .. ..$ PARA_1  : int [1:1488] 0 0 0 0 NA NA NA NA 0 0 ...
# .. .. ..$ RHINO   : int [1:1488] 0 0 0 0 NA NA NA NA 0 0 ...
# .. .. ..$ RSV     : int [1:1488] 0 0 0 0 NA NA NA NA 0 0 ...
# ..$ MSS:List of 1
# .. ..$ BCX:'data.frame': 1488 obs. of  2 variables:
```

```
#   .. .. ..$ HINF: int [1:1488] 0 0 0 0 0 0 0 0 0 0 ...
# .. .. ..$ PNEU: int [1:1488] 0 0 0 0 0 0 0 0 0 0 ...
# $ X   :'data.frame': 1488 obs. of  4 variables:
# ..$ patid         : chr [1:1488] "S00021" "S00023" "S00026" "S00027" ...
# ..$ AGE           : int [1:1488] 0 1 0 0 0 0 0 0 1 0 ...
# ..$ HIV2          : int [1:1488] 0 0 0 0 0 0 0 0 1 0 ...
# ..$ ALL_VS        : int [1:1488] 0 0 1 0 1 0 1 0 0 0 ...
# $ Y   : num [1:1488] 1 1 1 1 1 1 1 1 1 1 ...
```

We specify `L = 8` causes comprised of seven singleton-pathogen causes along with a cause named `"other"` that represents a generic non-specified ("NoS") cause. For BrS data, we use nasopharyngeal polymerase chain reaction (NPPCR), which results in case-control measurements upon `J.BrS` targeted pathogens (bacteria and viruses with abbreviated names `"HINF"`, `"PNEU"`, `"ADENO"`, `"HMPV_A_B"`, `"PARA_1"`, `"RHINO"`, `"RSV"`). For SS data, we use blood culture (BCX) results for two species of bacteria (`"HINF"`,`"PNEU"`). We then organize these measurement information into `BrS_object_1` and `SS_object_1`, respectively. Finally, we create a temporary folder for storing model results.

```
R> cause_list <- c("HINF","PNEU","ADENO","HMPV_A_B","PARA_1","RHINO","RSV","other")
R> patho_BrS_NPPCR <- c("HINF","PNEU","ADENO","HMPV_A_B","PARA_1","RHINO","RSV")
R> patho_SS_BCX <- c("HINF","PNEU")
R> BrS_object_1 <- make_meas_object(patho_BrS_NPPCR,"NP","PCR","BrS",cause_list)
R> SS_object_1  <- make_meas_object(patho_SS_BCX,"B","CX","SS",cause_list)
R> perch_clean <- list(BrS_objects = list(BrS_object_1),
+                      SS_objects = list(SS_object_1))
R> result_folder  <- tempdir()
R> dir.create(result_folder)
```

We the specify an NPLCM with five subclasses. In addition, we 1) let population etiology ("CSCFs") depend on age, severity status, and HIV status, and 2) let subclass weights depend on age and HIV status. We use both BrS and SS data for estimation. The TPR priors are specified via prior 2.5% and 97.5% prior Beta quantiles (0.5 to 0.9 for BrS measurement TPRs; 0.05 to 0.2 for SS measurement TPRs); these prior ranges were elicited from domain scientists. This can be achieved by the following code.

```
R> perch_model <- list(use_measurements = c("BrS","SS"),
+ likelihood  = list(cause_list = cause_list, k_subclass = c(5),
+   Eti_formula = ~ -1+as.factor(AGE)+as.factor(ALL_VS)+as.factor(HIV2),
+   FPR_formula = list(NPPCR =  ~ -1+as.factor(AGE)+as.factor(HIV2) )),
+ prior = list(Eti_prior  = c(2,2),
+   TPR_prior   = list(
+    BrS = list(info  = "informative", input = "match_range",
+      val = list(NPPCR = list(up = list(rep(0.9,length(BrS_object_1$patho))),
+                        low = list(rep(0.5,length(BrS_object_1$patho)))  ))),
+    SS = list(info = "informative", input = "match_range",
+      val = list(MSS1 = list(up = list(rep(0.2,length(SS_object_1$patho))),
+                  low = list(rep(0.05,length(SS_object_1$patho))) ))))))
```

We then specify the settings for the MCMC algorithm (`perch_mcmc`):

```
R> perch_mcmc <- list(n.chains   = 1, n.itermcmc = 200, n.burnin   = 100,
+    n.thin = 1, individual.pred = TRUE, ppd = TRUE,
+    result.folder = result_folder, bugsmodel.dir = result_folder)
```

Below, we fit the specified model. Here we assume the data set (`data_nplcm`) has been loaded from the real data (not yet publicly available). We store data (to `"data_nplcm.txt"`), store data cleaning information (to `"data_clean_options.txt"`) check model specifications via `assign_model()`, and fit the model by `nplcm()`:

```
R> dput(perch_data,file.path(perch_data$result.folder,"data_nplcm.txt"))
R> dput(perch_clean,file.path(perch_mcmc$result.folder,"data_clean_options.txt"))
R> assign_model(perch_model,perch_data)
R> rjags::load.module("glm")
R> perch_fit <- nplcm(data_nplcm,model_options,perch_mcmc)
```

We can summarize the fitted object using `summary(perch_fit)` (results not shown here). We can also visualize rich information about the posterior distribution of the population-level CSCFs in the cases using generic function `plot()` which shows the high population level etiologic importance of the virus RSV:

```
R> plot(perch_fit)
```

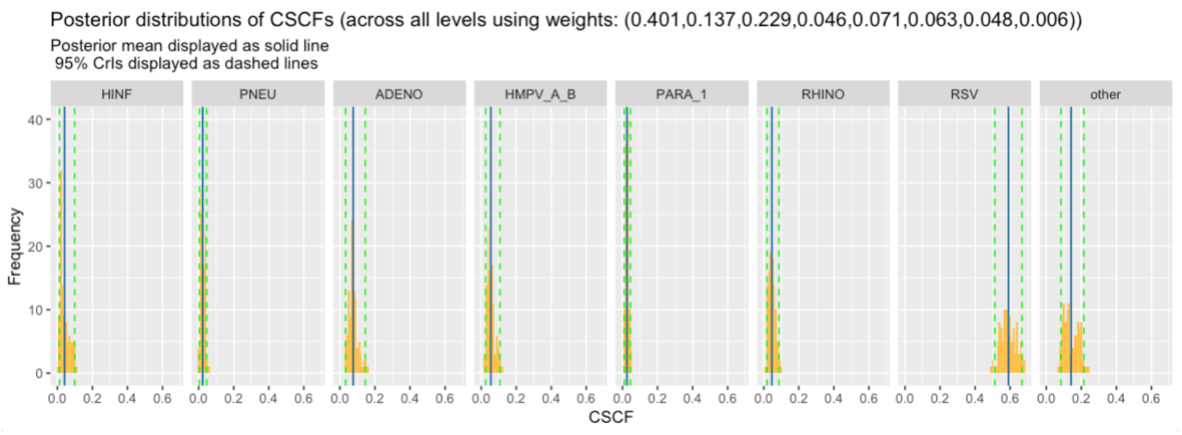

Figure 5: Marginal posterior distributions for each of `L = 8` pre-specified causes as visualized by the output from `plot` for an `nplcm` object `"perch_fit"`.

As mentioned before, **baker** can produce versatile posterior inferences about various unknown quantities based on the posterior samples. In particular, we can estimate the posterior probabilities of class membership probabilities for each individual case given each individual case's observed measurements:

```
R> get_individual_prediction(perch_fit)
#       HINF PNEU ADENO HMPV_A_B PARA_1 RHINO  RSV other
#  [1,] 0.03 0.00  0.02     0.03   0.04  0.03 0.49  0.36
#  [2,] 0.59 0.01  0.06     0.04   0.02  0.05 0.11  0.12
#  [3,] 0.05 0.01  0.07     0.06   0.02  0.05 0.71  0.03
#  [4,] 0.04 0.00  0.02     0.02   0.01  0.01 0.59  0.31
#  [5,] 0.10 0.00  0.02     0.03   0.02  0.07 0.76  0.00
```

Here we show the posterior probabilities of eight disease classes (columns) for five random cases (rows). For each individual, the probabilities differ across causes indicating varying posterior etiologic importance. For each cause, subjects differ in the BrS and SS measurements and covariates, leading to distinct posterior class membership probabilities.

5. Summary

The **baker** package provides functionalities to estimate a suite of NPLCM-based models (Wu *et al.* 2016, 2017; Wu and Chen 2021) for multivariate binary responses that are observed under a case-control design. **baker** has three major strengths: 1) it enables case-control analyses with or without covariates in the NPLCM framework, 2) it relaxes the "conditional independence" assumption often used in latent class analyses, and 3) it is designed to handle multiple sets of case-control or case-only measurements of distinct quality. Model results, posterior uncertainty assessment, and model diagnostics can also be readily summarized by **baker**.

Our future aim is to accommodate mixed categorical, ordinal, and continuous responses. The latent class model is naturally suited for multivariate discrete responses. We can extend our framework to handle multiple response levels by adding additional response probability parameters for each response level. Second, to accommodate ordinal and continuous responses, e.g., those produced by measurement technologies such as antibody titers and real-time PCR, class-specific mixture component likelihood functions may be specified via latent random Gaussian vectors, a subset of which are then linked to non-continuous responses via thresholding. Fast computational techniques for sampling Gaussian covariance matrices in multivariate probit models akin to Zhang, Nishimura, Bastide, Ji, Payne, Goulder, Lemey, and Suchard (2021) can be implemented. Finally, our current functions for NPLCM regression analyses will be expanded to accommodate higher dimensional covariates for CSCF and subclass weights using sparse Bayesian Additive Regression Trees (Linero 2018).

Acknowledgments

This work was supported by a Michigan Institute of Data Science (MIDAS) seed grant (to Z.W.); the Patient-Centered Outcomes Research Institute (PCORI) Award [ME-1408-20318 to Z.W. and S.L.Z.]; and the National Institutes of Health grants [P30CA046592 to Z.W.]. We thank the PERCH study team led by Katherine O'Brien for providing the data and scientific advice, Maria Deloria-Knoll and Christine Prosperi for valuable feedback about **baker** and Jing Chu for preliminary simulations. We also thank John Kubale for valuable feedback.

References

Allman ES, Matias C, Rhodes JA (2009). "Identifiability of parameters in latent structure models with many observed variables." *The Annals of Statistics*, **37**(6A), 3099–3132.

Brooks S, Gelman A, Jones G, Meng XL (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

Dunson D, Xing C (2009). "Nonparametric Bayes modeling of multivariate categorical data." *Journal of the American Statistical Association*, **104**(487), 1042–1051.

Gelfand A, Smith A (1990). "Sampling-based approaches to calculating marginal densities." *Journal of the American Statistical Association*, **85**(410), 398–409.

Goodman L (1974). "Exploratory latent structure analysis using both identifiable and unidentifiable models." *Biometrika*, **61**(2), 215–231.

Jones G, Johnson W, Hanson T, Christensen R (2010). "Identifiability of models for multiple diagnostic testing in the absence of a gold standard." *Biometrics*, **66**(3), 855–863.

Lazarsfeld PF (1950). *The logical and mathematical foundations of latent structure analysis*, volume IV, chapter The American Soldier: Studies in Social Psychology in World War II, pp. 362–412. Princeton, NJ: Princeton University Press.

Leisch F, Gruen B (2022). "CRAN Task View: Cluster Analysis and Finite Mixture Models." `https://cran.r-project.org/web/views/Cluster.html`. [Online; accessed 01-January-2022].

Linero AR (2018). "Bayesian regression trees for high-dimensional prediction and variable selection." *Journal of the American Statistical Association*, **113**(522), 626–636.

Linzer DA, Lewis JB (2011). "poLCA: An R Package for Polytomous Variable Latent Class Analysis." *Journal of Statistical Software*, **42**(10), 1–29. URL `http://www.jstatsoft.org/v42/i10/`.

McCormick TH, Li ZR, Calvert C, Crampin AC, Kahn K, Clark SJ (2016). "Probabilistic cause-of-death assignment using verbal autopsies." *Journal of the American Statistical Association*, **111**(515), 1036–1049.

PERCH Study Group (2019). "Causes of severe pneumonia requiring hospital admission in children without HIV infection from Africa and Asia: the PERCH multi-country case-control study." *The Lancet*, **392**(10200), 757–779.

Plummer M (2022). "JAGS: Just Another Gibbs Sampler." `https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/`. [Online; accessed 01-January-2022].

Plummer M, *et al.* (2003). "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling." In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, volume 124.

Qu Y, Tan M, Kutner MH (1996). "Random effects models in latent class analysis for evaluating accuracy of diagnostic tests." *Biometrics*, **52**(3), 797–810.

Sethuraman J (1994). "A constructive definition of Dirichlet priors." *Statistica Sinica*, pp. 639–650.

Stephenson BJ, Herring AH, Olshan A (2019). "Robust clustering with subpopulation-specific deviations." *Journal of the American Statistical Association*.

The PERCH Team (2019). ""Visualizing PERCH Results: Etiology of Pneumonia in Children Hospitalized in 7 Countries"." http://perchresults.org/. [Online; accessed 21-Feb-2022].

Vermunt JK, Magidson J (2002). "Latent class cluster analysis." *Applied Latent Class Analysis*, **11**, 89–106.

White A, Murphy TB (2014). "BayesLCA: An R Package for Bayesian Latent Class Analysis." *Journal of Statistical Software*, **61**(13), 1–28. URL http://www.jstatsoft.org/v61/i13/.

Wu Z, Chen I (2021). "Probabilistic cause-of-disease assignment using case-control diagnostic tests: A latent variable regression approach." *Statistics in Medicine*, **40**(4), 823–841.

Wu Z, Deloria-Knoll M, Hammitt LL, Zeger SL, the PERCH Study Team (2016). "Partially latent class models for case–control studies of childhood pneumonia aetiology." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **65**(1), 97–114.

Wu Z, Deloria-Knoll M, Zeger SL (2017). "Nested partially latent class models for dependent binary data; estimating disease etiology." *Biostatistics (Oxford, England)*, **18**, 200–213. ISSN 1468-4357. doi:10.1093/biostatistics/kxw037.

Xu G (2017). "Identifiability of restricted latent class models with binary responses." *The Annals of Statistics*, **45**(2), 675 – 707. doi:10.1214/16-AOS1464. URL https://doi.org/10.1214/16-AOS1464.

Zhang Z, Nishimura A, Bastide P, Ji X, Payne RP, Goulder P, Lemey P, Suchard MA (2021). "Large-scale inference of correlation among mixed-type biological traits with phylogenetic multivariate probit models." *The Annals of Applied Statistics*, **15**(1), 230–251.

A. Additional code to simulate data with covariates

```
R> N.SITE     <- 2
R> N          <- N.SITE*(300+300)
R>
R> CSCF_allsites <- list(c(0.5,0.2,0.15,0.05,0.05,0.05),
R>                           c(0.2,0.5,0.15,0.05,0.05,0.05))
R>
R> out_list <- lapply(1:N.SITE,function(siteID){
R>   set_parameter <- list(
R>     cause_list   = c("A","B","C","D","E","F"),
R>     etiology     = CSCF_allsites[[siteID]],
R>     pathogen_BrS = LETTERS[1:J.BrS],
R>     SS           = TRUE,
R>     pathogen_SS  = c("A","B"),
R>     meas_nm      = list(MBS = c("MBS1"),MSS=c("MSS1")),
R>     Lambda       = c(0.5,0.5),  # control subclass weight for BrS
R>     Eta          = t(replicate(J.BrS,c(0,1))),
R>     PsiBS        = cbind(c(0.25,0.25,0.2,0.15,0.15,0.15),
R>                          c(0.2, 0.2, 0.25,0.1,0.1,0.1)),
R>     PsiSS        = cbind(rep(0,J.BrS),rep(0,J.BrS)),
R>     ThetaBS      = cbind(c(0.95,0.9,0.9,0.9,0.9,0.9),
R>                          c(0.95,0.9,0.9,0.9,0.9,0.9)),
R>     ThetaSS      = cbind(c(0.25,0.10,0.15,0.05,0.15,0.15),
R>                          c(0.25,0.10,0.15,0.05,0.15,0.15)),
R>     Nd = 300,
R>     Nu = 300
R>   )
R>   out     <- simulate_nplcm(set_parameter)
R>   res   <- out$data_nplcm
R>   res$X <- data.frame(SITE=rep(siteID,(set_parameter$Nd+set_parameter$Nu)))
R>   return(res)
R> })
R> data_nplcm_unordered  <- combine_data_nplcm(out_list)
R> data_nplcm_reg_nest_strat <- subset_data_nplcm_by_index(data_nplcm_unordered,
R>                                    order(-data_nplcm_unordered$Y))
R> # load another data in `baker` with a continuous covariate
R> data(data_nplcm_reg_nest)
```

B. Additional code to fit models

```
R> nplcm_noreg_with_SS <- nplcm(data_nplcm_noreg,
+   model_options_no_reg_with_SS,mcmc_options_no_reg_with_SS)
R> nplcm_reg_nest_strat <- nplcm(data_nplcm_reg_nest_strat,
+   model_options_reg_nest_strat,mcmc_options_reg_nest_strat)
R> nplcm_reg_nest <- nplcm(data_nplcm_reg_nest,
+   model_options_reg_nest,mcmc_options_reg_nest)
```

C. Selected code outputs

C.1. Organized BrS data meta-information

```
R> BrS_object_1
# $quality
# [1] "BrS"
# $patho
# [1] "A" "B" "C" "D" "E" "F"
# $name_in_data
# [1] "A_MBS1" "B_MBS1" "C_MBS1" "D_MBS1" "E_MBS1" "F_MBS1"
# $template
#      [,1] [,2] [,3] [,4] [,5] [,6]
# [1,]    1    0    0    0    0    0
# [2,]    0    1    0    0    0    0
# [3,]    0    0    1    0    0    0
# [4,]    0    0    0    1    0    0
# [5,]    0    0    0    0    1    0
# [6,]    0    0    0    0    0    1
# [7,]    0    0    0    0    0    0
# $specimen
# [1] "MBS"
# $test
# [1] "1"
# $nm_spec_test
# [1] "MBS1"
```

C.2. Specifying model

```
R> assign_model(model_options_no_reg,data_nplcm_noreg)
# $num_slice
# MBS MSS MGS
#   1   0   0
# $nested
# [1] TRUE
# $regression
# $regression$do_reg_Eti
# [1] FALSE
# $regression$do_reg_FPR
#  MBS1
# FALSE
# $regression$is_discrete_predictor
# $regression$is_discrete_predictor$Eti
# [1] FALSE
# $regression$is_discrete_predictor$FPR
#  MBS1
# FALSE
```

C.3. Summary

```
R> summary(nplcm_noreg_with_SS)
# [baker] summary: model structure
#            fitted type:  no_reg
# ---
#      name measurements:  MBS MSS MGS
# slices of measurements:  1 1 0
#                 nested:  TRUE
# ---
#             regression:
#                   etiology:  FALSE
#                   name FPR:  MBS1
#                        FPR:  FALSE
# ---
# all discrete predictor:
#                   etiology:  FALSE
#                   name FPR:  MBS1
#                        FPR:  FALSE
#
# ------- posterior summary -----------
#    post.mean     post.sd      CrI_025    CrI_0975
# A 0.56573053 0.036129324 0.4992234000 0.63213780
# B 0.17558873 0.037365233 0.1216957500 0.26571990
# C 0.13901539 0.032986620 0.0864293800 0.21263615
# D 0.06632659 0.021147999 0.0300819800 0.10015262
# E 0.01137363 0.009374251 0.0003651512 0.03293634
# F 0.04196514 0.019310869 0.0084014205 0.08096354
```

C.4. Posterior predictive checking

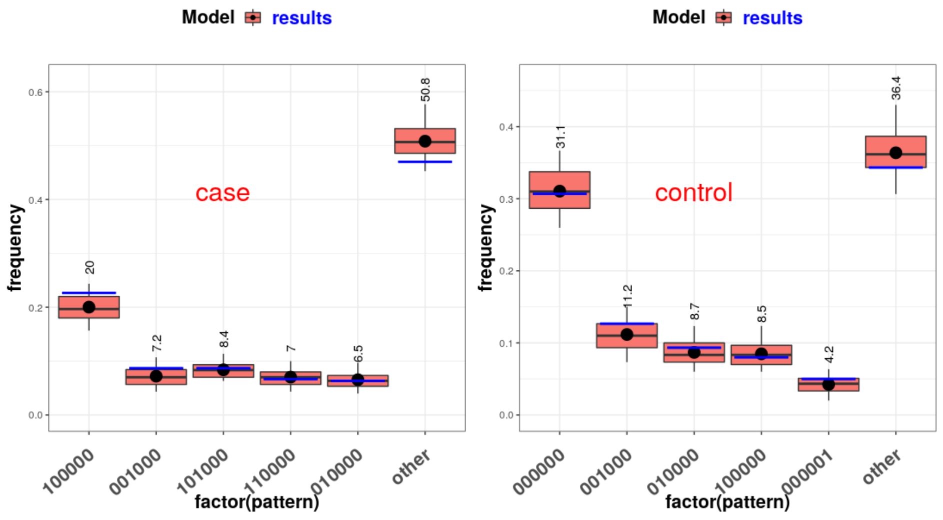

Figure 6: For the cases and the controls, posterior predictive checking based on the probability of multivariate binary patterns (five top patterns in the actual data and the rest aggregated). For each pattern, the posterior predictive distribution is shown by a boxplot; the horizontal blue bar indicates the actual observed frequency. A large deviation of a horizontal bar from the corresponding boxplot suggests potential model misfit.

Affiliation:

Irena Chen, Qiyuan Shi, Zhenke Wu*
Department of Biostatistics
University of Michigan
1415 Washington Heights
Ann Arbor, Michigan 48109, U.S.A.
*Corresponding author E-mail: zhenkewu@umich.edu
*Corresponding author URL: zhenkewu.com

Scott L. Zeger
Department of Biostatistics
The Johns Hopkins University
615 N. Wolfe Street
Baltimore, Maryland 21205, U.S.A.