Estimating Causal Effect using the Bayesian Method with the **R** Package BayesCACE

Jincheng Zhou
Gilead

Jinhui Yang
University of Minnesota

James S. Hodges
University of Minnesota

Lifeng Lin
Florida State University

Haitao Chu
University of Minnesota

Abstract

Noncompliance, a common problem in randomized clinical trials (RCTs), complicates the analysis of the causal treatment effect especially in meta-analysis of RCTs. The complier average causal effect (CACE) measures the effect of an intervention in the latent subgroup of the population that complies with its assigned treatment (the compliers). Recently, Bayesian hierarchical approaches have been proposed to estimate the CACE in a single RCT and a meta-analysis of RCTs. We develop an R package, **BayesCACE**, to provide user-friendly functions for implementing CACE analysis for binary outcomes based on the flexible Bayesian hierarchical framework. This package includes functions for analyzing data from a single study and for performing a meta-analysis with either complete or incomplete compliance data. The package also provides various functions for generating forest, trace, posterior density, and auto-correlation plots, to review noncompliance rates, visually assess the model, and obtain study-specific and overall CACEs.

Keywords: Bayesian inference, CACE, causal effect, meta-analysis, randomized clinical trial.

1. Introduction

1.1. Noncompliance in randomized clinical trials and causal effect

Randomized clinical trials (RCTs) are often used to test healthcare-related interventions. An RCT typically compares an experimental treatment to a standard treatment or to a placebo. A common problem in RCTs is that not all patients fully comply with the allocated treatments. Although RCT investigators control the randomization process, the actual treatments received by study participants may not follow the randomization allocation; this is called noncompliance. For example, in trials of a therapist-led intervention, noncompliance occurs when individuals randomized to the intervention fail to take the intervention (e.g., due to severe adverse events), or when some patients assigned to the control, under some circumstances, figure out a way to take the intervention. In some cases, investigators can collect outcome data on all of these patients, regardless of whether they followed interventions. When compliance status is incompletely observed, it causes even more complication in analyzing the treatment effect.

Conventionally, researchers use the intention-to-treat (ITT) analysis, in which data are analyzed based on treatments originally allocated rather than treatments actually received. The

ITT method estimates the effect of being offered the intervention, namely, the overall effect in the real world in which the intervention is made available. However, our interest may lie in a different question, namely the causal effect of actually receiving the treatment. When using ITT, the treatment effect tends to be diluted by including people who do not receive the treatment to which they were randomly allocated (Freedman 1990).

To identify a treatment's causal effect, the principal stratification framework (Frangakis and Rubin 2002) is proposed, which stratifies subjects on the joint potential post-randomization variables. This causal inference method is widely used in handling various intercurrent events (also called as intermediate variable) in the areas like vaccine effect (Hudgens and Halloran 2006; Zhou, Chu, Hudgens, and Halloran 2016), pain relief use (Baccini, Mattei, and Mealli 2017), surrogate endpoint evaluation (Gilbert, Gabriel, Huang, and Chan 2015), noncompliance (Zhou, Hodges, Suri, and Chu 2019), etc. An estimator called the "complier average causal effect" (CACE) has been proposed under the , in which patients are classified into different principal strata (compliers, never-takers, always-takers, and defiers) based on their *potential* behavior after assignment to both the treatment and control arms. *Compliers* are patients who receive the treatment as assigned in either arm; *never-takers* are those who do not receive treatment, regardless of treatment assignment; *always-takers* are those who receive treatment regardless of treatment assignment; and patients who always do the opposite of their treatment assignment are called *defiers*. The CACE is then the effect of the intervention estimated from compliers' outcomes. Because patients are assumed to be compliers (or not) before the randomization, the CACE retains the benefit of the randomization so it is an unbiased estimate of the difference in outcomes for compliers in the intervention group compared to those in the control group who would have engaged with treatment had they been randomized to the intervention group.

The biggest challenge in estimating the CACE is that we cannot actually identify which participants are compliers. Some of those receiving the treatment in the intervention group are compliers, but the rest are always-takers. Similarly, some of those not receiving the treatment in the control arm are compliers, but others are never-takers. Several R packages are available to perform CACE analysis in a single study. For example, the **noncomplyR** package (Coggeshall 2017) provides convenient functions for using Bayesian methods to perform inferences on the CACE. The package **eefAnalytics** (Kasim, Xiao, Higgings, and De Troyer 2017) provides tools for exploratory CACE analysis of simple randomized trials, cluster randomized trials, and multi-site trials with a focus on education trials. Besides the CACE analysis, another method quite commonly used to account for noncompliance is the instrumental variable (IV) method estimating the treatment effect with two-staged least squares (2SLS) regression (White 1982); the R package **ivpack** (Jiang and Small 2014) performs this type of analysis.

1.2. CACE in meta-analysis

All of the above methods are framed in a single study setting. However, for analyzing multiple trials in the presence of noncompliance, little software is available for causal effect analysis and specifically for meta-analysis. When noncompliance data are reported in each trial, one could intuitively implement a two-step approach by first estimating CACE for each study and then combining the study-specific estimates using a fixed-effect or random-effects model to estimate the population-averaged CACE. Recently, Zhou *et al.* (2019) proposed a Bayesian hierarchical model to estimate the CACE in a meta-analysis of randomized trials where compliance may be

heterogeneous between studies. It is also common that noncompliance data are not available for some trials. Simply excluding trials with incomplete noncompliance data from a meta-analysis can be inefficient and potentially biased. Zhou, Hodges, and Chu (2021) proposed an improved flexible Bayesian hierarchical CACE framework to account simultaneously for heterogeneous noncompliance and incomplete noncompliance data. The package **BayesCACE** focuses on providing user-friendly functions to estimate CACE in either a single study or meta-analysis using models based on Zhou *et al.* (2019), Baker (2020), Zhou, Hodges, and Chu (2020) and Zhou *et al.* (2021).

This article introduces the R package **BayesCACE**, which performs CACE analysis for binary outcomes in a single study, and meta-analysis with either complete or incomplete noncompliance information. This package is available from GitHub at `https://github.com/JinchengZ/BayesCACE`. It uses Markov chain Monte Carlo (MCMC) methods on the R platform through JAGS. JAGS is a program for analyzing Bayesian hierarchical models using MCMC simulation, which is available for diverse computer platforms including Windows and Mac OS X. Convergence of the MCMC routine can be assessed by the function outputs. The package also provides functions to make posterior trace plots, density plots, and auto-correlation plots. For meta-analysis, the package provides a forest plot of study-specific CACE estimates with 95% credible intervals as well as the overall CACE estimate, to visually display the causal treatment effect comparisons.

This article is organized as follows. The next subsection defines CACE in mathematical notation that will be used throughout the paper. We also describe the assumptions needed to make the CACE a valid causal effect estimator. Section 2 presents an overview of the Bayesian hierarchical models for CACE implemented in the **BayesCACE** package. Section 3 illustrates use of the package with a case study example and discusses the output structures. Finally, Section 4 gives a brief discussion with potential future improvements.

1.3. Assumptions and definition of CACE

The CACE is a measure of the causal effect of a treatment or intervention on patients who received it as intended by the original group allocation. It is an unbiased causal effect estimate based on five standard assumptions commonly used in causal inference research. First, it assumes that potential outcomes for each participant are independent of the potential outcomes for other participants, known as the *Stable Unit Treatment Value Assumption (SUTVA)*. Second, it assumes that assignment to treatment is random, so that the proportion of compliers should be the same in the intervention and control groups, thus allowing us to estimate one of the core unobserved parameters needed to derive a CACE estimate. Third, it assumes that treatment assignment has an effect on the outcome only if it changes the actual treatment taken, an assumption known as *exclusion restriction*. For never-takers, for instance, it assumes that simply being assigned to treatment does not affect their outcomes, as they do not actually receive the treatment assignd to them. Fourth, it assumes that assigning the study treatment to participants in the intervention condition induces at least some participants to receive the treatment, so the compliance rate is not zero. Finally, it assumes that there is a monotonic relationship between treatment assignment and treatment receipt, which implies that there are no individuals for whom assignment to treatment actually reduces the likelihood of receiving treatment (i.e., no defiers). This assumption reduces the number of compliance types for whom estimates are derived, permitting a properly identified model.

We follow Zhou *et al.* (2019) and introduce notation both on the individual level and on the study level. Suppose a meta-analysis reviews I two-armed RCTs, and N_i is the number of subjects in the i-th trial for $i \in \{1, \ldots, I\}$. If the data include a single study only, then $I = 1$ and we can remove the subscript i from all notation.

On the individual level, define notation as follows for subject j in trial i.

(1) Let $R_{ij} = r$ index the randomization assignment with $r = 0$ for those randomized to control and $r = 1$ for those randomized to the intervention.

(2) Let $T_{ij}^r = t \in \{0, 1\}$ be the indicator of whether the individual received the intervention. This is a *potential* outcome under the randomization assignment $r \in \{0, 1\}$, i.e., what the value of t would be for individual (i, j) if $r = 0$ or $r = 1$, respectively.

(3) Let $Y_{ij}^{r,t} = o \in \{0, 1\}$ be the *potential* binary outcome under randomization assignment r and treatment received t. Note that the *exclusion restriction* assumption allows us to define $Y_{ij}^t \equiv Y_{ij}^{r,t}$.

(4) The sets of $\{Y_{ij}^{r,t}\}$ and $\{T_{ij}^r\}$ are the *potential* outcome and treatment-received status respectively under possible r and t, but for each subject in a trial, only one of the possible values of each set can be observed. Therefore, we denote the observed response and received treatment variables as Y_{ij} and T_{ij}.

(5) We allow $T_{ij} = *$ if the actual received treatment is not recorded. Then let $M_{ij} = m$ be the missing indicator corresponding to whether subject j has actual treatment received status on record ($m = 0$) or missing ($m = 1$).

(6) Using these potential outcomes, we can define the compliers and the CACE. Let C_{ij} be the latent compliance class of individual j in trial i, defined as follows:

$$C_{ij} = \begin{cases} 0, & \text{for never-taker with } (T_{ij}^0, T_{ij}^1) = (0, 0) \\ 1, & \text{for complier with } (T_{ij}^0, T_{ij}^1) = (0, 1) \\ 2, & \text{for always-taker with } (T_{ij}^0, T_{ij}^1) = (1, 1) \\ 3, & \text{for defier with } (T_{ij}^0, T_{ij}^1) = (1, 0) \end{cases}.$$

A subject's compliance status C_{ij} is not observable because in a two-arm trial, only one of T_{ij}^1 and T_{ij}^0 can be observed. Based on the observed randomization group and actual treatment received, the compliance classes can be only partially identified.

Now, the complier average causal effect of the i-th trial is the average difference between potential outcomes for compliers. In this case, the CACE in study i is $\theta_i^{\text{CACE}} = E(Y_{ij}^1 - Y_{ij}^0 | C_{ij} = 1)$, where the patients for whom $C_{ij} = 1$ are the compliers.

On the study level, N_{irto} denotes the observed number of individuals in study i, randomization group r, actual received treatment group t, and outcome o. If the compliance status of individual j in trial i is not on record, $T_{ij} = t = *$ so the corresponding count is N_{ir*o}, which is the sum of the two unobserved counts N_{ir0o} and N_{ir1o}.

2. Estimating CACE

This section briefly describes the Bayesian hierarchical models used to estimate CACE. These models form the basis of the framework proposed by Zhou *et al.* (2019) and underlie the **BayesCACE** package. Besides the notation defined in Section 1.3, we define the following parameters for study i.

(1) Let π_{ia} and π_{in} be the probabilities of being an always-taker and a never-taker, respectively. Because defiers are ruled out by the monotonicity assumption introduced in Section 1.3, each trial has at most only three compliance classes. Thus the probability of being a complier in study i is $\pi_{ic} = 1 - \pi_{ia} - \pi_{in}$.

(2) Define these response probabilities: u_{i1} for a complier randomized to the treatment group; v_{i1} for a complier randomized to the control/placebo group; s_{i1} for a never-taker; and b_{i1} for an always-taker. Thus for study i, the parameters included in the model are $\boldsymbol{\beta}_i = (\pi_{ia}, \pi_{in}, u_{i1}, v_{i1}, s_{i1}, b_{i1})$.

As the outcome is binary, the expected difference between outcomes from the two treatment groups among compliers is just the risk difference between u_{i1} and v_{i1}. Therefore, the CACE defined in Section 1.3 can be written as $\theta_i^{\text{CACE}} = E(Y_{ij}^1 - Y_{ij}^0 | C_{ij} = 1) = u_{i1} - v_{i1}$.

2.1. CACE for a single trial with noncompliance

Consider first a single trial with noncompliance, i.e., $I = 1$, so all notation and parameters defined earlier are reduced to the version without subscript i. According to Zhou *et al.* (2019), each observed N_{rto} has a corresponding probability that can be written in terms of parameters defined earlier (see Table 1), where $\lambda = \Pr(R_j = 1)$ is the proportion of assigning the active treatment, which is usually known in randomized trials. Thus the vector $(N_{000}, N_{001}, N_{010}, N_{011}, N_{100}, N_{101}, N_{110}, N_{111})$ follows a multinomial distribution with parameters N and $\mathbf{p}$, where $N = \sum N_{rto}$ and the elements of $\mathbf{p}$ are listed in Table 1.

Table 1: Observed data and probabilities

Observed	Probabilities
N_{000}	$(1 - \lambda)\{\pi_c(1 - v_1) + \pi_n(1 - s_1)\}$
N_{001}	$(1 - \lambda)(\pi_c v_1 + \pi_n s_1)$
N_{010}	$(1 - \lambda)\pi_a(1 - b_1)$
N_{011}	$(1 - \lambda)\pi_a b_1$
N_{100}	$\lambda \pi_n(1 - s_1)$
N_{101}	$\lambda \pi_n s_1$
N_{110}	$\lambda\{\pi_c(1 - u_1) + \pi_a(1 - b_1)\}$
N_{111}	$\lambda(\pi_c u_1 + \pi_a b_1)$

Therefore, the log likelihood is

$$
\begin{aligned}
\log L(\boldsymbol{\beta}) =& N_{000} \log\{\pi_c(1 - v_1) + \pi_n(1 - s_1)\} + N_{001} \log(\pi_c v_1 + \pi_n s_1) + N_{010} \log\{\pi_a(1 - b_1)\} \\
&+ N_{011} \log\{\pi_a b_1\} + N_{100} \log\{\pi_n(1 - s_1)\} + N_{101} \log(\pi_n s_1) \\
&+ N_{110} \log\{(\pi_c(1 - u_1) + \pi_a(1 - b_1)\} + N_{111} \log(\pi_c u_1 + \pi_a b_1) + \text{constant}.
\end{aligned}
$$

(1)

Assigning a vague prior distribution $f(\boldsymbol{\beta})$ to the parameters $\boldsymbol{\beta} = (\pi_a, \pi_n, u_1, v_1, s_1, b_1)$, by Bayes' theorem the joint posterior distribution is proportional to $L(\boldsymbol{\beta})f(\boldsymbol{\beta})$. Functionals of the posterior distribution can be estimated by Gibbs and Metropolis–Hastings sampling algorithms using the software JAGS via the **rjags** package in R. The CACE for a single study is $u_1 - v_1$, so the posterior of θ^{CACE} is the posterior of $u_1 - v_1$.

2.2. CACE for a meta-analysis with complete compliance information

This section introduces two methods for performing a meta-analysis of the CACE when noncompliance data are reported in each trial.

The two-step approach

As described in Section 2.1, using the observed data N_{irto}, θ_i^{CACE} is identified for study i. Therefore, to estimate the population-average CACE in a meta-analysis, intuitively we can combine the study-specific estimates and standard errors using a standard meta-analysis method such as the fixed-effect (Laird and Mosteller 1990) or random-effects model (Hedges and Vevea 1998; Hedges and Olkin 1985). We call this a "two-step" approach. As the CACE measure is a risk difference, a transformation may be necessary to ensure that the normal distribution assumption is approximately true. Building upon the well-developed R package **metafor**, various estimators suggested in the literature can be estimated to account for potential between-study heterogeneity in the CACE, e.g., the Hunter–Schmidt estimator, the Hedges estimator, the DerSimonian–Laird estimator, the maximum-likelihood or restricted maximum-likelihood estimator, or the empirical Bayes estimator (Viechtbauer 2010).

The Bayesian hierarchical model

In a meta-analysis, the CACE can also be estimated using the joint likelihood from the Bayesian hierarchical model. This method is systematically introduced in Zhou *et al.* (2019). The log likelihood contribution of trial i is given by Equation (1) by adding a subscript i to each parameter. Then the log likelihood for all trials in the meta-analysis is $\log \mathcal{L}(\boldsymbol{\beta}) = \sum_i \log L_i(\boldsymbol{\beta}_i)$. Because the studies are probably not exactly identical in their eligibility criteria, measurement techniques, study quality, etc., differences in methods and sample characteristics may introduce heterogeneity to the meta-analysis. One way to model the heterogeneity is to use a random-effects model.

To guarantee the desired properties of study i's latent compliance classes and to account for possible between-study heterogeneity in the compliance class and response probabilities, we use these transformations:

(1) $\pi_{in} = \frac{\exp(n_i)}{1+\exp(n_i)+\exp(a_i)}, \pi_{ia} = \frac{\exp(a_i)}{1+\exp(n_i)+\exp(a_i)}$, where $n_i = \alpha_n + \delta_{in}, a_i = \alpha_a + \delta_{ia}$, and $(\delta_{in}, \delta_{ia})^\top \sim N(0, \boldsymbol{\Sigma}_{ps})$, $\boldsymbol{\Sigma}_{ps} = \begin{pmatrix} \sigma_n^2 & \rho\sigma_n\sigma_a \\ \rho\sigma_n\sigma_a & \sigma_a^2 \end{pmatrix}$.

(2) $g(s_{i1}) = \alpha_s + \delta_{is}$, $g(b_{i1}) = \alpha_b + \delta_{ib}$, $g(u_{i1}) = \alpha_u + \delta_{iu}$, $g(v_{i1}) = \alpha_v + \delta_{iv}$, where $g(\cdot)$ is a link function such as the logit or probit, $\delta_{is} \sim N(0, \sigma_s^2)$, $\delta_{ib} \sim N(0, \sigma_b^2)$, $\delta_{iu} \sim N(0, \sigma_u^2)$, $\delta_{iv} \sim N(0, \sigma_v^2)$.

Here we allow correlation between n_i and a_i, and assign random effect variables to all parameters. However, if a parameter does not vary between trials, it can be modeled as a fixed effect.

Let $f(\boldsymbol{\beta}_i|\boldsymbol{\beta}_0, \boldsymbol{\Sigma}_0)$ be the distributions described above of all parameters $\boldsymbol{\beta}_i = (\pi_{ia}, \pi_{in}, s_{i1}, b_{i1}, u_{i1}, v_{i1})$, where $\boldsymbol{\beta}_0$ is the vector of mean hyper-parameters $(\alpha_n, \alpha_a, \alpha_s, \alpha_b, \alpha_u, \alpha_v)$, and $\boldsymbol{\Sigma}_0$ is the diagonal covariance matrix containing $\boldsymbol{\Sigma}_{ps}$, σ_s^2, σ_b^2, σ_u^2 and σ_v^2. If we specify $f(\boldsymbol{\beta}_0)$ and $f(\boldsymbol{\Sigma}_0)$ as the prior distributions for the hyper-parameters, then the joint posterior distribution is proportional to the likelihood times the priors, i.e., $\prod_i L_i(\boldsymbol{\beta}_i) f(\boldsymbol{\beta}_i|\boldsymbol{\beta}_0, \boldsymbol{\Sigma}_0) f(\boldsymbol{\beta}_0) f(\boldsymbol{\Sigma}_0)$.

As stated at the beginning of Section 2, $\theta_i^{\mathrm{CACE}} = u_{i1} - v_{i1}$ for study i, so for the meta-analysis, the overall CACE is $\theta^{\mathrm{CACE}} = E(\theta_i^{\mathrm{CACE}}) = E(u_{i1}) - E(v_{i1})$. When a random effect δ_{iu} or δ_{iv} is not assigned in the model, $E(u_{i1}) = g^{-1}(\alpha_u)$ and $E(v_{i1}) = g^{-1}(\alpha_v)$. Otherwise, $E(u_{i1})$ and $E(v_{i1})$ can be estimated by integrating out the random effects, e.g., $E(u_{i1}) = \int_{-\infty}^{+\infty} g^{-1}(\alpha_u + t)\sigma_u^{-1}\phi(\frac{t}{\sigma_u})dt$, where $\phi(\cdot)$ is the standard Gaussian density. If the function $g(\cdot)$ is the probit link, this expectation has a closed form: $E(u_{i1}) = \Phi(\frac{\alpha_u}{\sqrt{1+\sigma_u^2}})$. If the link function $g(\cdot)$ is logit, a well-established approximation $E(u_{i1}) \approx \mathrm{logit}^{-1}(\frac{\alpha_u}{\sqrt{1+C^2\sigma_u^2}})$ can be used, where $C = \frac{16\sqrt{3}}{15\pi}$ (Zeger, Liang, and Albert 1988). The above formulas also apply to $E(v_{i1})$, the expected response rate of a complier in the control group.

The two-step approach, stated by Lin and Zeng (2010), can be viewed as asymptotically equivalent to the model using the joint likelihood. However, as the two-step approach requires the whole set of parameters to be estimated independently for each study, the total number of effective parameters tends to be larger than this method, so estimates using the Bayesian hierarchical model are likely to be more efficient.

2.3. CACE for meta-analysis with incomplete compliance information

Another advantage of the Bayesian hierarchical model is that it can include trials with incomplete compliance data. Commonly, some trials do not report noncompliance data because study investigators do not collect actual received treatment status for some subjects or simply do not report compliance. The two-step approach needs counts for all of the groups defined by randomized assignment, treatment received, and outcome in order to estimate the study specific θ_i^{CACE}. Thus, by using this method, trials with incomplete compliance data are simply excluded, making estimation less efficient and potentially biased.

Zhou *et al.* (2021) proposed a comprehensive framework to incorporate both heterogeneous and incomplete noncompliance data for estimating the CACE in a meta-analysis of RCTs. Here we present the data structure needed for binary outcomes. Table 2 shows the probabilities corresponding to the observed counts data. For study i, randomization group $r \in \{0, 1\}$, if the compliance information is reported, then values of N_{ir0o} and N_{ir1o} are reported, where $o \in \{0, 1\}$, so we assign the marginal count $N_{ir*o} = 0$. Otherwise, we do not have data on outcomes for groups defined by actually received treatment, so only the marginal N_{ir*o} is observed, where N_{ir*o} is the number of patients randomized to treatment arm r who had outcome o, for $r, o \in \{0, 1\}$. In this situation, the two unobserved counts N_{ir0o} and N_{ir1o} are assigned as 0.

After organizing the observed data as above, Table 2 shows the relation between each observed count and the corresponding probability, which is a function of the parameters defined in Section 2.2.2. As before, λ_i is the known allocation ratio for study i, i.e., $\lambda_i = \mathrm{Pr}(R_{ij} = 1)$.

Table 2: Observed data and probabilities in study i

Observed	Probabilities
N_{i000}	$(1 - \lambda_i)\{\pi_{ic}(1 - v_{i1}) + \pi_{in}(1 - s_{i1})\}$
N_{i001}	$(1 - \lambda_i)(\pi_{ic}v_{i1} + \pi_{in}s_{i1})$
N_{i010}	$(1 - \lambda_i)\pi_{ia}(1 - b_{i1})$
N_{i011}	$(1 - \lambda_i)\pi_{ia}b_{i1}$
N_{i100}	$\lambda_i\pi_{in}(1 - s_{i1})$
N_{i101}	$\lambda_i\pi_{in}s_{i1}$
N_{i110}	$\lambda_i\{(\pi_{ic}(1 - u_{i1}) + \pi_{ia}(1 - b_{i1})\}$
N_{i111}	$\lambda_i(\pi_{ic}u_{i1} + \pi_{ia}b_{i1})$
N_{i0*0}	$(1 - \lambda_i)\{\pi_{ic}(1 - v_{i1}) + \pi_{in}(1 - s_{i1}) + \pi_{ia}(1 - b_{i1})\}$
N_{i0*1}	$(1 - \lambda_i)(\pi_{ic}v_{i1} + \pi_{in}s_{i1} + \pi_{ia}b_{i1})$
N_{i1*0}	$\lambda_i\{(\pi_{ic}(1 - u_{i1}) + \pi_{ia}(1 - b_{i1}) + \pi_{in}(1 - s_{i1})\}$
N_{i1*1}	$\lambda_i(\pi_{ic}u_{i1} + \pi_{ia}b_{i1} + \pi_{in}s_{i1})$

The log likelihood contribution for trial i is obtained from the multinomial distribution:

$$
\begin{aligned}
&\log L_i(\boldsymbol{\beta}_i) \\
=&N_{i000}\log\{\pi_{ic}(1 - v_{i1}) + \pi_{in}(1 - s_{i1})\} + N_{i001}\log(\pi_{ic}v_{i1} + \pi_{in}s_{i1}) \\
&+N_{i010}\log\{\pi_{iia}(1 - b_{i1})\} + N_{i011}\log(\pi_{ia}b_{i1}) + N_{i100}\log\{\pi_{in}(1 - s_{i1})\} \\
&+N_{i101}\log(\pi_{in}s_{i1}) + N_{i110}\log\{(\pi_{ic}(1 - u_{i1}) + \pi_{ia}(1 - b_{i1})\} + N_{i111}\log(\pi_{ic}u_{i1} + \pi_{ia}b_{i1}) \\
&+N_{i0*0}\log\{\pi_{ic}(1 - v_{i1}) + \pi_{in}(1 - s_{i1}) + \pi_{ia}(1 - b_{i1})\} + N_{i0*1}\log(\pi_{ic}v_{i1} + \pi_{in}s_{i1} + \pi_{ia}b_{i1}) \\
&+N_{i1*0}\log\{(\pi_{ic}(1 - u_{i1}) + \pi_{ia}(1 - b_{i1}) + \pi_{in}(1 - s_{i1})\} + N_{i1*1}\log(\pi_{ic}u_{i1} + \pi_{ia}b_{i1} + \pi_{in}s_{i1})
\end{aligned}
\tag{2}
$$

Because the parameters $\boldsymbol{\beta}_i = (\pi_{ia}, \pi_{in}, s_{i1}, b_{i1}, u_{i1}, v_{i1})$ are the same as in Section 2.2.2, the estimation process is also the same: assign distributions $f(\boldsymbol{\beta}_i|\boldsymbol{\beta}_0, \boldsymbol{\Sigma}_0)$, where $\boldsymbol{\beta}_0$ is the vector of mean hyper-parameters, and $\boldsymbol{\Sigma}_0$ is the covariance matrix; then specify prior distributions for $f(\boldsymbol{\beta}_0)$ and $f(\boldsymbol{\Sigma}_0)$, so the joint posterior is proportional to $\prod_i L_i(\boldsymbol{\beta}_i)f(\boldsymbol{\beta}_i|\boldsymbol{\beta}_0, \boldsymbol{\Sigma}_0)f(\boldsymbol{\beta}_0)f(\boldsymbol{\Sigma}_0)$. Similarly, the CACE for this meta-analysis incorporating incomplete compliance data is $\theta^{\text{CACE}} = E(\theta_i^{\text{CACE}}) = E(u_{i1}) - E(v_{i1}) = \Phi(\frac{\alpha_u}{\sqrt{1+\sigma_u^2}}) - \Phi(\frac{\alpha_v}{\sqrt{1+\sigma_v^2}})$ if the probit link function is used for u_{i1} and v_{i1}.

3. Using the R package BayesCACE

The primary objective of the **BayesCACE** package is to provide a user-friendly implementation of the Bayesian method for estimating the CACE, described in Section 2. The package is now available to download and install via the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=BayesCACE, or from GitHub at https://github.com/JinchengZ/BayesCACE. It can be installed within R using the command `install.packages("BayesCACE")`. The latest version of the package is 1.1.

The **BayesCACE** package depends on the R packages **rjags** (Plummer 2018), **coda** (Plummer, Best, Cowles, and Vines 2006), and **forestplot** (Gordon and Lumley 2017). Users need to install JAGS separately from its homepage http://mcmc-jags.sourceforge.net as the

BayesCACE package does not include a copy of the JAGS library. The current version of JAGS is 4.3.0, which is the version of the package **BayesCACE** requires; earlier versions of JAGS may not guarantee exactly reproducible results. Once the package has been correctly installed, it replicates the analyses described in this section to within MCMC error.

3.1. Data structure for estimating the CACE

We introduce the data structures through the illustrative example included in the package **BayesCACE**: epidural_c and epidural_ic. These two data sets were obtained from Bannister-Tyrrell, Miladinovic, Roberts, and Ford (2015), who conducted an exploratory meta-analysis of the association between using epidural analgesia in labor and the risk of cesarean section. The dataset epidural_c contains 10 trials with full compliance information; each trial has 8 observed counts, denoted by N_{irto} and presented in columns Nirto for $i = 1, \ldots, 10$ and $r, t, o \in \{0, 1\}$. These data were re-analyzed by Zhou *et al.* (2019) in a meta-analysis using their proposed Bayesian hierarchical model to estimate the CACE. The function cace.meta.c() introduced in Section 3.3 performs this analysis. The column study.id contains IDs for the 10 studies, and study.name labels each study by its first author's surname and its publication year.

The data can be loaded and printed using these commands:

```
R> library("BayesCACE")
R> data("epidural_c", package = "BayesCACE")
R> epidural_c
```

	study.id	study.name	N000	N001	N010	N011	N100	N101	N110	N111
1	1	Bofill, 1997	37	2	11	1	2	0	42	5
2	2	Clark, 1998	72	6	68	16	7	2	134	13
3	3	Halpern, 2004	62	5	44	7	0	0	112	12
4	4	Head, 2002	51	7	2	0	3	0	43	10
5	5	Jain, 2003	72	11	0	0	0	2	36	7
6	6	Nafisi, 2006	179	19	0	0	0	0	173	24
7	7	Nikkola, 1997	6	0	4	0	0	0	10	0
8	8	Ramin, 1995	546	17	95	8	230	2	393	39
9	9	Sharma, 1997	336	16	5	0	114	1	231	12
10	10	Volmanen, 2008	23	1	3	0	1	0	23	1

The other dataset epidural_ic represents the situation in which not all trials report complete compliance data. It contains 27 studies, only 10 out of which have full compliance information and were included in epidural_c. This dataset is also drawn from Bannister-Tyrrell *et al.* (2015) but only the method introduced in Section 2.3 can include the studies with incomplete compliance information when estimating the CACE. The function cace.meta.ic() performs this analysis; see Section 3.3.3 for details.

Each study is represented by one row in the dataset; the columns study.id and study.name have the same meanings as in the dataset epidural_c. Each study's data are summarized in 12 numbers (columns) denoted by N_{irto} and N_{ir*o} as described in Section 2.3. For a particular randomization group $r \in \{0, 1\}$, the observed counts are presented either as N_{irto} or N_{ir*o} depending on whether the compliance information is available; values for other columns

are denoted by 0. The corresponding column names in the dataset are `Nirto` and `Nirso`, respectively.

The first 6 rows of the dataset `epidural_ic` are printed below.

```
R> data("epidural_ic", package = "BayesCACE")
R> head(epidural_ic)
```

```
  study.id         study.name n000 n001 n010 n011 n0s0 n0s1 n100 n101 n110
1        1       Bofill, 1997   37    2   11    1    0    0    2    0   42
2        2        Clark, 1998   72    6   68   16    0    0    7    2  134
3        3   Dickinson, 2002    0    0    0    0  428   71    0    0    0
4        4        Evron, 2008   40    4    0    0    0    0    0    0    0
5        5 El Kerdawy, 2010    0    0    0    0   12    3    0    0    0
6        6    Gambling, 1998    0    0    0    0  573   34  206   10  371
  n111 n1s0 n1s1
1    5    0    0
2   13    0    0
3    0  408   85
4    0  129   19
5    0   11    4
6   29    0    0
```
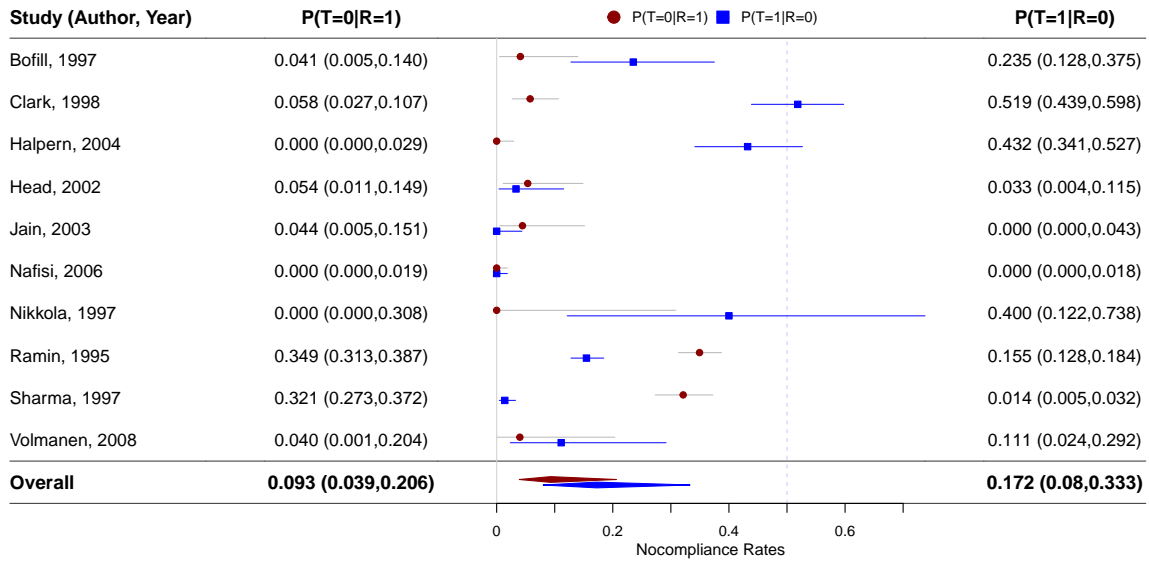
Note that `NA` is not allowed in a dataset for the package **BayesCACE**, but some trials may have 0 events or 0 noncompliance rates.

3.2. Plotting noncompliance rates

Before performing the CACE analysis, one might want a visual overview of study-specific noncompliance rates in both randomization arms. The function `plt.noncomp` provides a forest plot of noncompliance rates in an R plot window. The function can be simply called as

```
plt.noncomp(data, overall = TRUE)
```

where `data` is a dataset with structure like `epidural_c` or `epidural_ic`. Only studies with full compliance information are included in this plot because noncompliance rates cannot be calculated without compliance data. Figure 1 shows the resulting plot, where the red dot with its horizontal line shows the study-specific noncompliance rate with its 95% exact confidence interval for the patients randomized to the treatment arm, and the blue square with its horizontal line represents that rate and interval for those in the control arm. The confidence intervals are calculated by the Clopper–Pearson exact method (Clopper and Pearson 1934), which is based on the cumulative distribution function of the binomial distribution. Using the default `overall = TRUE`, the figure also gives a summary estimate of the compliance rates per randomization group. This overall rate is estimated using a logit generalized linear mixed model. Otherwise, if the argument `overall` is `FALSE`, the plot shows only study-specific noncompliance rates. Any additional parameters passed to the function will be automatically used in the `forestplot` function in the **forestplot** package.

Study (Author, Year)	P(T=0|R=1)	● P(T=0|R=1) ■ P(T=1|R=0)	P(T=1|R=0)
Bofill, 1997	0.041 (0.005,0.140)		0.235 (0.128,0.375)
Clark, 1998	0.058 (0.027,0.107)		0.519 (0.439,0.598)
Halpern, 2004	0.000 (0.000,0.029)		0.432 (0.341,0.527)
Head, 2002	0.054 (0.011,0.149)		0.033 (0.004,0.115)
Jain, 2003	0.044 (0.005,0.151)		0.000 (0.000,0.043)
Nafisi, 2006	0.000 (0.000,0.019)		0.000 (0.000,0.018)
Nikkola, 1997	0.000 (0.000,0.308)		0.400 (0.122,0.738)
Ramin, 1995	0.349 (0.313,0.387)		0.155 (0.128,0.184)
Sharma, 1997	0.321 (0.273,0.372)		0.014 (0.005,0.032)
Volmanen, 2008	0.040 (0.001,0.204)		0.111 (0.024,0.292)
Overall	**0.093 (0.039,0.206)**		**0.172 (0.08,0.333)**

Nocompliance Rates

Figure 1: Noncompliance rates plot generated by the function `plt.noncomp()`.

3.3. CACE analysis for a single study or in a meta-analysis

The major functions in the **BayesCACE** package are `cace.study()`, `cace.meta.c()`, and `cace.meta.ic()`, which implement the models introduced in Section 2 to perform Bayesian CACE analysis for different data structures. In particular, `cace.study()` performs CACE analysis for a single study using the likelihood and model specified in Section 2.1. The function `cace.meta.c()` performs CACE analysis for a meta-analysis when each trial reports noncompliance information. Users can choose to perform the analysis either by the two-step approach or using the Bayesian hierarchical model, as introduced in Section 2.2. When some trials do not report noncompliance data, the function `cace.meta.ic()` can be applied to perform a CACE meta-analysis using the likelihood in Equation 2. The commands in each function may take 1–15 minutes to run. Generally the two-step approach using the function `cace.meta.c()` takes longer because MCMC chains are run on the studies one by one. The actual run time depends on the amount of data and the user's processor.

Function `cace.study()` for a study-specific analysis or a two-step meta-analysis

For the default interface, the arguments of the function `cace.study()` are

```
cace.study(data, param = c("CACE", "u1", "v1", "s1", "b1", "pi.c", "pi.n",
  "pi.a"), re.values = list(), model.code = '', digits = 3, n.adapt = 1000,
  n.iter = 100000, n.burnin = floor(n.iter/2), n.chains = 3, n.thin =
  max(1,floor((n.iter-n.burnin)/1e+05)), conv.diag = FALSE, mcmc.samples =
  FALSE, two.step = FALSE, method = "REML")
```

where users need to input `data` with the same structure as `epidural_c`, containing either one row of observations for a single study, or multiple rows referring to multiple studies in a

meta-analysis. This function fits a model for a single study as described in Section 2.1. If the data includes more than one study, the study-specific CACEs will be estimated by retrieving data row by row.

The argument `param` is a character string vector indicating the parameters to be tracked and estimated. By default all parameters shown in Section 2.1 are included: θ^{CACE} (`CACE`), u_1 (`u1`), v_1 (`v1`), s_1 (`s1`), b_1 (`b1`), π_a (`pi.a`), π_n (`pi.n`), and $\pi_c = 1 - \pi_a - \pi_n$ (`pi.c`). Users can modify the string vector to only include parameters of interest besides θ^{CACE}. Users can specify the prior distributions (mean and standard deviation) of $n, a, \alpha_s, \alpha_b, \alpha_u, \alpha_v$ with the `re.values` parameter. By default, the `re.values` list is empty, and they are assigned to the transformed scale of the following parameters: $\pi_n = \frac{\exp(n)}{1+\exp(n)+\exp(a)}$, $\pi_a = \frac{\exp(a)}{1+\exp(n)+\exp(a)}$, $\mathrm{logit}(s_1) = \alpha_s$, $\mathrm{logit}(b_1) = \alpha_b$, $\mathrm{probit}(u_1) = \alpha_u$, and $\mathrm{probit}(v_1) = \alpha_v$, where $n, a \sim N(0, 2.5^2)$ and $\alpha_s, \alpha_b, \alpha_u, \alpha_v \sim N(0, 2^2)$. With these settings, a 95% prior probability interval for any of the probabilities π_{in}, π_{ia}, and π_{ic} ranges from about 0.001 to 0.91, and a 95% prior interval for the probabilities s_1, b_1, u_1, and v_1 ranges approximately from 0.01 to 0.98. The prior parameters are passed into the `model.study` function to get the model code, which first calls the `prior.study` to get the custom prior distribution. Here we give an example output of `prior.study` when assigning $N(0, 10^{-2})$ to every parameter:

```
out.string <-
  "# priors
  n ~ dnorm(0, 0.01)
  a ~ dnorm(0, 0.01)
  alpha.s ~ dnorm(0, 0.01)
  alpha.b ~ dnorm(0, 0.01)
  alpha.u ~ dnorm(0, 0.01)
  alpha.v ~ dnorm(0, 0.01)"
"
```

To customize the model fully, user can pass their complete model string to the `cace.study()` function with the parameter `model.code`. The arguments `n.adapt`, `n.iter`, `n.burnin`, `n.chains`, and `n.thin` control the MCMC algorithm run by the R package **rjags** (Plummer 2018). The argument `n.adapt` is the number of iterations for adaptation; it is used to maximize the sampling efficiency, and the default is set as 1,000. The argument `n.chains` determines the number of MCMC chains (the default is 3); `n.iter` is the number of iterations of each MCMC chain; `n.burnin` is the number of burn-in iterations at the beginning of each chain to be discarded; `n.thin` is the thinning rate for MCMC chains, which is used to avoid potential high auto-correlation and to save computer memory when `n.iter` is large. The default of `n.thin` is set as `1` or the largest integer not greater than (`(n.iter - n.burnin)/1e+05)`), whichever is larger. The argument `conv.diag` specifies whether to compute the Gelman and Rubin convergence statistic ($\hat{R}$) of each parameter as a convergence diagnostic (Brooks and Gelman 1998; Gelman and Rubin 1992). It is considered the chains are well mixed and have converged to the target distribution if $\hat{R} \le 1.1$. If the argument `mcmc.samples = TRUE`, the function saves each chain's MCMC samples for all parameters, which can be used to produce trace, posterior density, and auto-correlation plots by calling the functions `plt.trace`, `plt.density`, and `plt.acf`.

By default, the function `cace.study()` returns a list including posterior estimates (posterior mean, standard deviation, median, and a 95% credible interval (CI) with 2.5% and 97.5%

quantiles as the lower and upper bounds), and the deviance information criterion (DIC) statistic (Spiegelhalter, Best, Carlin, and Van Der Linde 2002) for each study. The argument `two.step` is a logical value indicating whether to conduct a two-step meta-analysis. If `two.step = TRUE`, the posterior mean and standard deviation of study-specific θ_i^{CACE} are used to perform a standard meta-analysis, using the R package **metafor**. The default estimation method is the REML (restricted maximum-likelihood estimator) method for the random-effects model (Harville 1977). Users can change the argument `method` to obtain different meta-analysis estimators from either a random-effects model or a fixed-effect model, e.g., `method = "DL"` refers to the DerSimonian–Laird estimator, `method = "HE"` returns the Hedges estimator, and `method = "HS"` gives the Hunter–Schmidt estimator. More details are available from the documentation of the function `metafor::rma` (Viechtbauer 2010). If the input data include only one study, the meta-analysis result is just the same as the result from the single study.

Here is an example to demonstrate the function's usage. We call the function `cace.study()` on the dataset `epidural_c` as follows:

```
R> data("epidural_c", package = "BayesCACE")
R> set.seed(123)
R> out.study <- cace.study(data = epidural_c, conv.diag = TRUE,
+   mcmc.samples = TRUE, two.step = TRUE)
```

The following messages are output as the code runs:

```
NA is not allowed in the input data set;
the rows containing NA are removed.
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 2
   Unobserved stochastic nodes: 6
   Total graph size: 44

Initializing model

  |+++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
  |**************************************************| 100%
  |**************************************************| 100%
MCMC convergence diagnostic statistics are calculated and saved in conv.out
```

If the dataset contains more than one study, e.g., the `epidural_c` dataset has 10 trials, then once the JAGS model compiles for the first study, it automatically continues to run on the next study's data. The results are saved in the object `out.study`, a list containing the model name, posterior information for each monitored parameter, and DIC of each study. We can use parameter names to display the corresponding estimates. The argument `digits` in the function `cace.study()` can be used to change the number of significant digits to the right of the decimal point. Here, we used the default setting `digits = 3`. For example, the estimates

of θ^{CACE} for each single study (posterior mean and standard deviation, posterior median, 95% credible interval, and time-series standard error) can be displayed as

```
R> out.study$CACE
```

```
          Mean     SD     2.5%        50%   97.5% Time-series SE
 [1,]   0.04960 0.0796 -0.0944   4.41e-02 0.2180 2.52e-04
 [2,]  -0.02460 0.0488 -0.1220  -2.19e-02 0.0789 1.48e-04
 [3,]  -0.02180 0.0609 -0.1270  -2.88e-02 0.1130 1.93e-04
 [4,]   0.07180 0.0762 -0.0769   7.12e-02 0.2240 2.05e-04
 [5,]   0.08260 0.0765 -0.0620   8.13e-02 0.2370 2.52e-04
 [6,]   0.02600 0.0318 -0.0362   2.58e-02 0.0887 7.42e-05
 [7,]   0.01420 0.1560 -0.2770   2.11e-04 0.4000 4.07e-04
 [8,]   0.05020 0.0247  0.0024   5.00e-02 0.0992 7.26e-05
 [9,]  -0.01090 0.0234 -0.0571  -1.08e-02 0.0349 6.29e-05
[10,]   0.00127 0.0649 -0.1340  -3.87e-06 0.1430 1.53e-04
```

If the argument `conv.diag` is specified as `TRUE`, the output list contains a sub-list `conv.out`, which outputs the point estimates of the 'potential scale reduction factor' (the Gelman and Rubin convergence statistic, labelled `Point est.`) calculated for each parameter from each single study, and their upper confidence limits (labelled `Upper C.I.`). Approximate convergence is diagnosed when the upper limit is close to 1 (Brooks and Gelman 1998; Gelman and Rubin 1992). For example, the first sub-list from `conv.out` is

```
R> out.study$conv.out[[1]]
```

```
      Point est. Upper C.I.
CACE  1.0000007   1.000003
b1    1.0000224   1.000060
pi.a  1.0000338   1.000127
pi.c  1.0000380   1.000135
pi.n  1.0000148   1.000063
s1    1.0000135   1.000042
u1    1.0000121   1.000028
v1    0.9999995   1.000012
```

Also, in this example, we included `mcmc.samples = TRUE` in the argument, so the output object list `out.study` includes each chain's MCMC samples for all parameters. They can be used with our plotting functions to generate the trace, posterior density, and auto-correlation plots for further model diagnostics.

If the dataset used by the function `cace.study()` has more than one study, specifying the argument `two.step = TRUE` causes the two-step meta-analysis for θ^{CACE} to be done. The outcomes are saved as a sub-list object `meta`. Note that users can obtain different meta-analysis estimators by changing the `method` argument as described earlier.

```
R> out.study$meta
```

```
Random-Effects Model (k = 10; tau^2 estimator: REML)

tau^2 (estimated amount of total heterogeneity): 0.0002 (SE = 0.0008)
tau (square root of estimated tau^2 value):      0.0129
I^2 (total heterogeneity / total variability):   8.00%
H^2 (total variability / sampling variability):  1.09

Test for Heterogeneity:
Q(df = 9) = 5.9134, p-val = 0.7486

Model Results:

estimate      se    zval    pval    ci.lb   ci.ub
  0.0183  0.0142  1.2854  0.1986  -0.0096  0.0462

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

Function `cace.meta.c()` *for meta-analysis with complete compliance data*

The function `cace.meta.c()` performs the Bayesian hierarchical model method for meta-analysis when the dataset has complete compliance information for all studies, as described in Section 2.2.2. The function's default arguments are given by

```
cace.meta.c(data, param = c("CACE", "u1out", "v1out", "s1out", "b1out",
  "pic", "pin", "pia"), random.effects = list(), re.values = list(),
  model.code = '', digits = 3, n.adapt = 1000, n.iter = 100000,
  n.burnin = floor(n.iter/2), n.chains = 3, n.thin =
  max(1,floor((n.iter-n.burnin)/100000)), conv.diag = FALSE,
  mcmc.samples = FALSE, study.specific = FALSE)
```

The arguments controlling the MCMC algorithm are mostly similar to those of `cace.study()`. One major difference is that users need to specify parameters that are modeled as random effects. In Section 2.2.2, we showed how to specify random effects for each parameter on the transformed scales, namely δ_{in}, δ_{ia}, δ_{iu}, δ_{iv}, δ_{is}, and δ_{ib}, and allowed a non-zero correlation ρ between δ_{in} and δ_{ia}. The model with all of these random effects as well as the correlation ρ is considered the full model. However, this function is flexible, allowing users to choose which random effects to include by specifying the `random.effects` argument. By default, the list is empty and all of the list values are set to TRUE. Users can customize that by setting `delta.n`, `delta.a`, `delta.u`, `delta.v`, `delta.s`, `delta.b`, and/or `cor` to FALSE. Note that ρ (`cor`) can only be included when both δ_{in} (`delta.n`) and δ_{ia} (`delta.a`) are set to TRUE. Otherwise, a warning occurs and the model continues running by forcing `delta.n = TRUE` and `delta.a = TRUE`. The default parameters to be monitored depend on which parameters are modeled as random effects. For example, `u1out` refers to $E(u_{i1})$ as described in Section 2.2.2, where for the probit link, $E(u_{i1}) = \Phi(\alpha_u)$ if δ_u is not specified in the model, and $E(u_{i1}) = \Phi(\frac{\alpha_u}{\sqrt{1+\sigma_u^2}})$ when the random effect δ_u is included.

Users can use the `re.values` parameter to customize the prior distribution. Like the function `cace.study()`, by default, weakly informative priors $\alpha_n, \alpha_a \sim N(0, 2.5^2)$ and $\alpha_s, \alpha_b, \alpha_u, \alpha_v \sim N(0, 2^2)$ are assigned to the means of these transformed parameters: $\pi_{in} = \frac{\exp(n_i)}{1+\exp(n_i)+\exp(a_i)}$, $\pi_{ia} = \frac{\exp(a_i)}{1+\exp(n_i)+\exp(a_i)}$, where $n_i = \alpha_n + \delta_{in}$, $a_i = \alpha_a + \delta_{ia}$, $\text{logit}(s_{i1}) = \alpha_s + \delta_{is}$, $\text{logit}(b_{i1}) = \alpha_b + \delta_{ib}$, $\text{probit}(u_{i1}) = \alpha_u + \delta_{iu}$, and $\text{probit}(v_{i1}) = \alpha_v + \delta_{iv}$. For the random effects, we have $\delta_{is} \sim N(0, \sigma_s^2)$, $\delta_{ib} \sim N(0, \sigma_b^2)$, $\delta_{iu} \sim N(0, \sigma_u^2)$, and $\delta_{iv} \sim N(0, \sigma_v^2)$, as response rates are assumed to be independent between latent classes. A $Gamma(2, 2)$ hyper-prior distribution is assigned to the precision parameters σ_s^{-2}, σ_b^{-2}, σ_u^{-2} and σ_v^{-2}, which corresponds to a 95% interval of $(0.6, 2.9)$ for the corresponding standard deviations, allowing moderate heterogeneity in the response rates. In a reduced model with one of δ_{in} or δ_{ia} set to 0, the prior of the other precision parameter is also assumed to be $Gamma(2, 2)$, which gives moderate heterogeneity for latent compliance classes probabilities, whereas for the full model, $(\delta_{in}, \delta_{ia})^\top \sim N(0, \mathbf{\Sigma}_{ps})$, the prior for the variance-covariance matrix $\mathbf{\Sigma}_{ps}$ is $InvWishart(\mathbf{I}, 3)$, where $\mathbf{I}$ is the identity matrix.

Similar to `cace.study()`, to customize the model fully, user can pass their complete model string with the parameter `model.code`. Because the function `cace.meta.c()` is more complicated depending on the choice of random effects, as an illustration we show an example of the customized prior distributions when assigning `delta.n = TRUE`, `delta.a = FALSE`, `delta.u = TRUE`, `delta.v = FALSE`, `delta.s = TRUE`, and `cor = FALSE` to function `cace.meta.c()`, while keeping default values for `re.values`.

```
string <-
"# priors
alpha.n ~  dnorm(0, 0.16)
alpha.a ~ dnorm(0, 0.16)
alpha.s ~  dnorm(0, 0.25)
alpha.b ~  dnorm(0, 0.25)
alpha.u ~  dnorm(0, 0.25)
alpha.v ~  dnorm(0, 0.25)

II[1,1] <- 1
II[2,2] <- 1
II[1,2] <- 0
II[2,1] <- 0

Omega.rho ~  dwish (II[,], 3)
Sigma.rho <- inverse(Omega.rho)
sigma.n <- Sigma.rho[1, 1]
sigma.a <- Sigma.rho[2, 2]
rho <- Sigma.rho[1, 2]
u1out <- phi(alpha.u/sqrt(1+sigma.u^2))
tau.u ~ dgamma(2, 2)
sigma.u <- 1/sqrt(tau.u)
v1out <- phi(alpha.v)
CACE <- u1out-v1out
s1out <- ilogit(alpha.s/sqrt(1 + (16^2*3/(15^2*pi^2))*sigma.s^2))
```

```
tau.s ~ dgamma(2, 2)
sigma.s <- 1/sqrt(tau.s)
b1out <- ilogit(alpha.b/sqrt(1 + (16^2*3/(15^2*pi^2))*sigma.b^2))
tau.b ~ dgamma(2, 2)
sigma.b <- 1/sqrt(tau.b)
"
```

The `epidural_c` dataset is used as a real-study example:

```
R> data("epidural_c", package = "BayesCACE")
R> set.seed(123)
R> out.meta.c <- cace.meta.c(data = epidural_c, conv.diag = TRUE,
+ mcmc.samples = TRUE, study.specific = TRUE)
```

The usage of arguments `conv.diag` and `mcmc.samples` are the same as for the function `cace.study`. When the argument `study.specific` is specified as `TRUE`, the model will first check the logical status of arguments `delta.u` and `delta.v`. If both are `FALSE`, meaning that neither response rate u_{i1} or v_{i1} is modeled with a random effect, then the study-specific θ_i^{CACE} is the same across studies. The function gives a warning and continues by making `study.specific = FALSE`. Otherwise, the study-specific θ_i^{CACE} are estimated and saved as the parameter `cacei`.

In this example, by calling the object `smry` from the output list `out.meta.c`, posterior estimates (posterior mean, standard deviation, posterior median, 95% credible interval, and time-series standard error) are displayed.

```
R> out.meta.c$smry
```

	Mean	SD	2.5%	50%	97.5%	Time-series SE
CACE	0.020900	0.0632	-0.10200	1.94e-02	0.1510	7.72e-04
b1out	0.127000	0.0451	0.05930	1.20e-01	0.2340	3.91e-04
cacei[1]	0.044000	0.0678	-0.08140	4.08e-02	0.1870	2.32e-04
cacei[2]	-0.023100	0.0491	-0.11500	-2.50e-02	0.0820	1.84e-04
cacei[3]	-0.007330	0.0566	-0.10900	-1.14e-02	0.1130	2.13e-04
cacei[4]	0.065400	0.0680	-0.06650	6.46e-02	0.2020	1.66e-04
cacei[5]	0.053800	0.0685	-0.07310	5.11e-02	0.1950	2.42e-04
cacei[6]	0.026300	0.0308	-0.03390	2.61e-02	0.0872	6.78e-05
cacei[7]	0.003040	0.0933	-0.18900	6.39e-05	0.2100	3.56e-04
cacei[8]	0.048400	0.0237	0.00215	4.83e-02	0.0953	6.25e-05
cacei[9]	-0.010700	0.0224	-0.05530	-1.06e-02	0.0331	5.58e-05
cacei[10]	0.000278	0.0604	-0.12100	-1.29e-03	0.1290	2.08e-04
pia	0.121000	0.0804	0.02550	1.02e-01	0.3450	4.69e-03
pic	0.815000	0.0948	0.55900	8.34e-01	0.9330	5.84e-03
pin	0.064500	0.0401	0.01540	5.59e-02	0.1590	2.32e-03
s1out	0.183000	0.1040	0.04540	1.60e-01	0.4400	8.93e-04
u1out	0.128000	0.0480	0.05540	1.20e-01	0.2430	6.14e-04
v1out	0.107000	0.0406	0.04740	1.00e-01	0.2040	4.61e-04

The posterior estimates of θ_i^{CACE} can be used to make a forest plot by calling the function `plt.forest`, which will be introduced in Section 3.5.

Users can manually do model selection procedures by including different random effects and comparing DIC from the outputs. DIC and its two components are saved as an object `DIC` in the output list.

```
R> out.meta.c$DIC
```

```
D.bar 204.34102
pD     44.74046
DIC   249.08148
```

DIC is the penalized deviance, calculated as the sum of `D.bar` and `pD`, where `D.bar` is the posterior expectation of the deviance, reflecting the model fit, and `pD` reflects the effective number of parameters in the model. `D.bar` is usually lower when more parameters are included in the model, but complex models may lead to overfitting. Thus `DIC` balances the model's fit against the effective number of parameters. Generally a model with smaller DIC is preferred. However, it is difficult to conclude what constitutes an important improvement in DIC. Following Lunn, Jackson, Best, Spiegelhalter, and Thomas (2012), we suggest that a reduction of less than 5 is not a substantial improvement. When fitting models to a particular dataset, it is usually uncertain which random effect variables should be included in the model. The function `cace.meta.c()` allows users to specify candidate models with different random effects, and thus to conduct a forward/backward/stepwise model selection procedure to choose the best fitting model.

Function `cace.meta.ic()` *for meta-analysis with incomplete compliance information*

Another major function in the package **BayesCACE** is `cace.meta.ic()`. It also estimates θ^{CACE} using the Bayesian hierarchcal model but can accommodate studies with incomplete compliance data. The necessary data structure and the likelihood function are presented in Section 2.3. The arguments of this function are

```
cace.meta.ic(data, param = c("CACE", "u1out", "v1out", "s1out", "b1out",
  "pic", "pin", "pia"), random.effects = list(), re.values = list(),
  model.code = '', digits = 3, n.adapt = 1000, n.iter = 100000,
  n.burnin = floor(n.iter/2), n.chains = 3, n.thin =
  max(1,floor((n.iter-n.burnin)/100000)), conv.diag = FALSE,
  mcmc.samples = FALSE, study.specific = FALSE)
```

The arguments of `cace.meta.ic()` are mostly similar to those of `cace.meta.c()`, though `cace.meta.ic()` calls a different built-in model file from the package **BayesCACE**. The major difference in using this function is that users need to create a dataset with the same structure as `epidural_ic`. Please check Section 3.1 for data preparation details. As for `cace.meta.c()`, users can set their customized prior distributions. Here we use the `epidural_ic` dataset as an example:

```
R> data("epidural_ic", package = "BayesCACE")
R> set.seed(123)
```

```
R> out.meta.ic <- cace.meta.ic(data = epidural_ic, conv.diag = TRUE,
+   mcmc.samples = TRUE, study.specific = TRUE)
```

The results are saved in the object `out.meta.ic`, a list containing posterior estimates for monitored parameters, DIC, convergence diagnostic statistics, and MCMC samples. In this example, the argument `study.specific` is TRUE, so the summary for each study-specific θ_i^{CACE} is displayed in the object `out.meta.ic$smry` together with other parameters.

Note that when compiling the JAGS model, the warning "adaptation incomplete" may occasionally occur, indicating that the number of iterations for the adaptation process is not sufficient. The default value of `n.adapt` (the number of iterations for adaptation) is 1,000. This is an initial sampling phase during which the samplers adapt their behavior to maximize their efficiency (e.g., a Metropolis–Hastings random walk algorithm may change its step size) (Plummer 2018). The "adaptation incomplete" warning indicates that the MCMC algorithm may not achieve maximum efficiency, but it generally has little impact on the posterior estimates of the treatment effects. To avoid this warning, users may increase `n.adapt`.

3.4. Plotting the trace plot, posterior density, and auto-correlation

When compiling the JAGS models, it is helpful to assess the performance of the MCMC algorithm. The functions `plt.trace`, `plt.density`, and `plt.acf` provide diagnostic plots for the MCMC, namely trace plots, kernel density estimation plots, and auto-correlation plots. Both trace plots and auto-correlation plots can be used to examine whether the MCMC chains appear to be drawn from stationary distributions. A posterior density plot for a parameter visually shows the posterior distribution. Users can simply call this function on objects produced by `cace.study()`, `cace.meta.c()`, or `cace.meta.ic()`.

The arguments of this plot function are

```
plt.trace(obj, param = c("CACE"), trialnumber = 1, ...)
plt.density(obj, param = c("CACE"), trialnumber = 1, ...)
plt.acf(obj, param = c("CACE"), trialnumber = 1, ...)
```

We use the objects list obtained from fitting the Bayesian hierarchical model `cace.meta.ic()` in Section 3.3.3 as an example to generate the three plots. To avoid lengthy output we just illustrate how these plots are produced for θ^{CACE}. The relevant code is:

```
R> plt.trace(obj = out.meta.ic)
R> plt.density(obj = out.meta.ic)
R> plt.acf(obj = out.meta.ic)
```

The produced plots are shown in Figures 2–4. The trace plots in Figure 2 show the parameter values sampled at each iteration versus the iteration number. Each chain is drawn as a separate trace plot to avoid overlay. Here we used the default `n.chains = 3`, so three trace plots are drawn. These plots show evidence that the posterior samples of θ^{CACE} are drawn from the stationary distribution.

The density plot in Figure 3 is smoothed using the R function `density()`. It shows that the kernel-smoothed posterior of θ^{CACE} is almost symmetric. The posterior mean is not far from 0, indicating that the complier average causal effect of using epidural analgesia in labor on cesarean section is likely not significant.
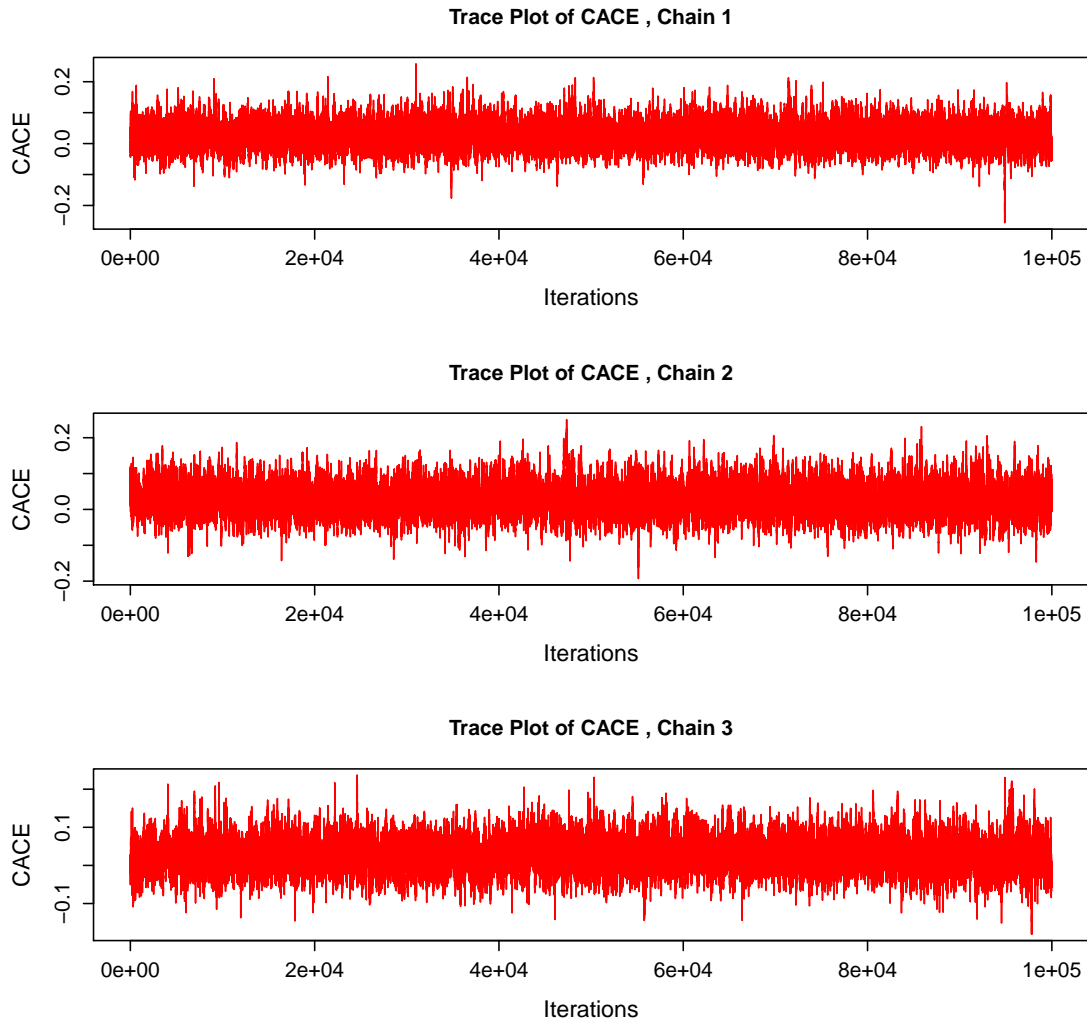
Trace Plot of CACE , Chain 1

Trace Plot of CACE , Chain 2

Trace Plot of CACE , Chain 3

Figure 2:   Trace plots for θ^{CACE} from the `epidural_ic` dataset fit using `cace.meta.ic()`.

The auto-correlation plot in Figure 4 is a bar plot displaying the auto-correlation for different lags. At lag 0, the value of the chain has perfect auto-correlation with itself. As the lag becomes greater, the values become less correlated. After a lag of about 50, the auto-correlation drops below 0.1. If the plot shows high auto-correlation, users can run the chain longer or can choose a larger `n.thin`, e.g., `n.thin = 10` would keep only 1 out of every 10 iterations, so that the thinned out chain is expected to have the auto-correlation dropping quickly. Any additional parameters passed to the 3 plotting function will be automatically used in the `plot` function for `plt.trace` and `plt.density`, and in the `acf` function for `plt.acf`.

3.5. Plotting the study-specific CACE in a forest plot

A graphical overview of the results can be obtained by creating a forest plot (Lewis and Clarke 2001). The function `plt.forest()` draws a forest plot for θ^{CACE} estimated from the meta-analysis. Users can call this function for the objects from `cace.meta.c()` or `cace.meta.ic()`.

Density of CACE
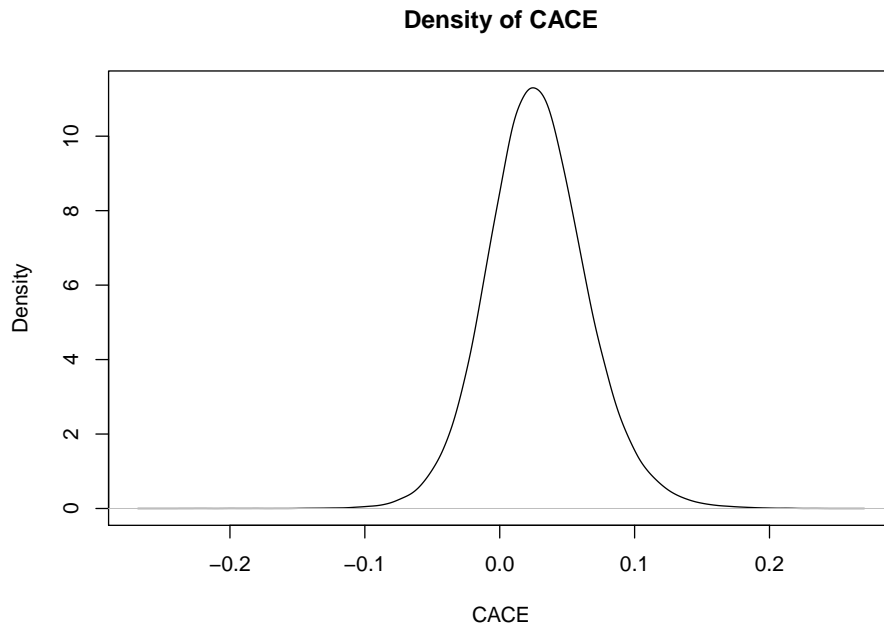

Figure 3: The kernel smoothed density for θ^{CACE} from the function `cace.meta.ic()` applied to the epidural analgesia in labor meta-analysis.

Here is an example using the object `out.meta.ic`:

```
R> plt.forest(data = epidural_ic, obj = out.meta.ic)
```

Note that in addition to the object `out.meta.ic`, users also need to specify the dataset used to compute that object, from which the `plt.forest()` function extracts the study names and publication years for the figure.

Figure 5 is a forest plot of θ_i^{CACE} for each study individually, using the Bayesian method with full random effects and default priors. The summary estimate based on the model `cace.meta.ic()` is automatically added to the figure, with the outer edges of the polygon indicating the confidence interval limits. The 95% credible interval of the summary θ^{CACE} covers zero, indicating a non-significant complier average causal effect estimate for using epidural analgesia in labor on the risk of cesarean section for the meta-analysis with 27 trials. For a study with incomplete data on compliance status, a dashed horizontal line in the forest plot is used to represent the posterior 95% credible interval of θ_i^{CACE} from the Bayesian hierarchical model fit. The study-specific θ_i^{CACE} vary from negative to positive in individual studies, while most of the 95% credible intervals cover zero. As the θ_i^{CACE} for a trial without complete compliance data is not estimable using only data from that single trial, dashed lines tend to have longer credible intervals than those with complete data (solid lines).

4. Discussion

This article provides an overview of the **BayesCACE** package for conducting CACE analysis
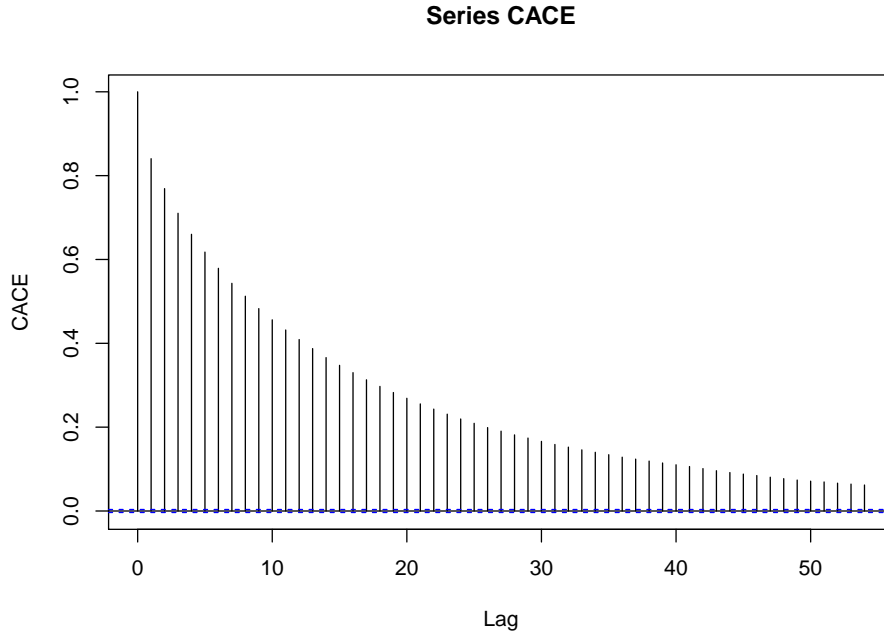
Figure 4:   Auto-correlation plot of θ^{CACE} from the model `cace.meta.ic()` fit to the `epidural_ic` dataset.

with R. Bayesian hierarchical models estimating the CACE in individual studies and in meta-analysis are introduced to demonstrate the underlying methods of the functions. Practical usage of various functions is illustrated using real meta-analyses datasets `epidural_c` and `epidural_ic`. The package provides several plots for interpretation of model outputs and model diagnosis.

It is important to note that the two-step approach for meta-analysis is included in the package **BayesCACE** because by using the full observed data from a single study i, θ_i^{CACE} is identifiable, making it possible to pool the estimated posterior means and standard deviations of the θ_i^{CACE} in a meta-analysis. However, the Bayesian hierarchical-model meta-analysis method for estimating the overall CACE, introduced in Sections 2.2.2 and 2.3, is preferred for two reasons: the conventional two-step approach requires the whole set of parameters to be estimated for each trial, giving a larger total number of parameters than the random effect model, so the estimate of the CACE can be less efficient. Also, when study i does not report complete compliance data, it must be excluded from the two-step approach because θ_i^{CACE} is no longer directly estimable by simply using the incomplete data from this individual study, while the function `cace.meta.ic()` can use the incomplete information and thus help improve the efficacy in estimation.

The Gelman and Rubin convergence statistics, time-series standard errors, trace plots, and auto-correlation plots are provided by the package **BayesCACE** to examine whether the MCMC chains are drawn from stationary distributions. However, in practice, any sample is finite, thus there is no guaranteed way to prove that the sampler has converged (Kass, Carlin, Gelman, and Neal 1998; Cowles and Carlin 1996). Additional techniques may be required to determine the effective sample size for adequate convergence (Robert and Casella 2004).
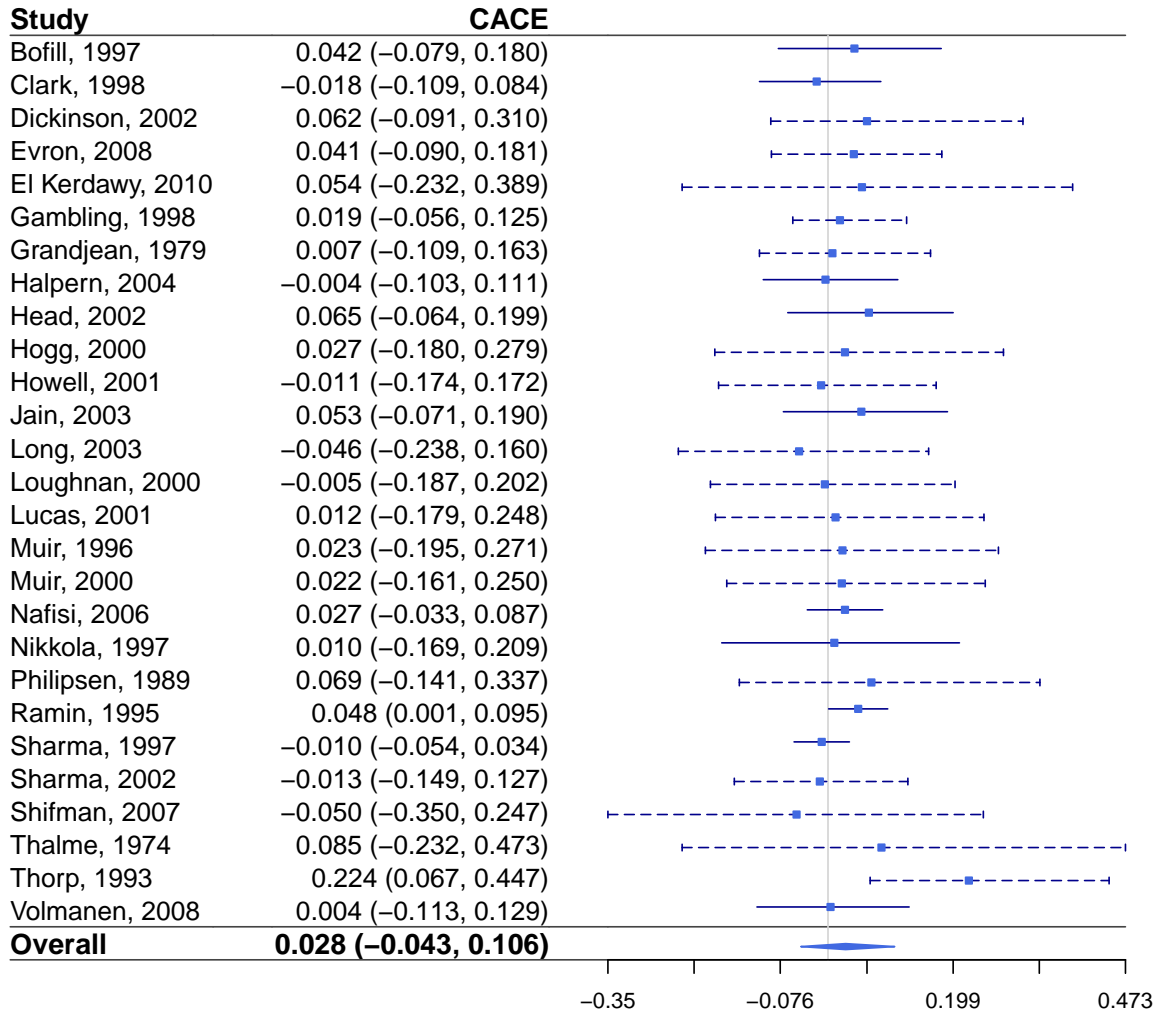
Study	CACE	
Bofill, 1997	0.042 (−0.079, 0.180)	
Clark, 1998	−0.018 (−0.109, 0.084)	
Dickinson, 2002	0.062 (−0.091, 0.310)	
Evron, 2008	0.041 (−0.090, 0.181)	
El Kerdawy, 2010	0.054 (−0.232, 0.389)	
Gambling, 1998	0.019 (−0.056, 0.125)	
Grandjean, 1979	0.007 (−0.109, 0.163)	
Halpern, 2004	−0.004 (−0.103, 0.111)	
Head, 2002	0.065 (−0.064, 0.199)	
Hogg, 2000	0.027 (−0.180, 0.279)	
Howell, 2001	−0.011 (−0.174, 0.172)	
Jain, 2003	0.053 (−0.071, 0.190)	
Long, 2003	−0.046 (−0.238, 0.160)	
Loughnan, 2000	−0.005 (−0.187, 0.202)	
Lucas, 2001	0.012 (−0.179, 0.248)	
Muir, 1996	0.023 (−0.195, 0.271)	
Muir, 2000	0.022 (−0.161, 0.250)	
Nafisi, 2006	0.027 (−0.033, 0.087)	
Nikkola, 1997	0.010 (−0.169, 0.209)	
Philipsen, 1989	0.069 (−0.141, 0.337)	
Ramin, 1995	0.048 (0.001, 0.095)	
Sharma, 1997	−0.010 (−0.054, 0.034)	
Sharma, 2002	−0.013 (−0.149, 0.127)	
Shifman, 2007	−0.050 (−0.350, 0.247)	
Thalme, 1974	0.085 (−0.232, 0.473)	
Thorp, 1993	0.224 (0.067, 0.447)	
Volmanen, 2008	0.004 (−0.113, 0.129)	
Overall	**0.028 (−0.043, 0.106)**	

Figure 5: Forest plot of study-specific θ^{CACE} from the model `cace.meta.ic()` with full random effects fit to the `epidural_ic` dataset.

For example, the well-developed R package **mcmcse** (Flegal, Hughes, Vats, and Dai 2017) can be used to assess whether MCMC has been run for enough iterations (sufficient chain lengths). To call the functions in **mcmcse**, users can specify the argument `mcmc.samples = TRUE` in `cace.study()`, `cace.meta.c()`, and `cace.meta.ic()`, so the MCMC posterior samples of monitored parameters are saved in the output objects.

The current version of **BayesCACE** only applies to binary outcomes. However, the Bayesian hierarchical model can be extended to handle ordinal outcomes $o \in \{1, \ldots, O\}$. By selecting weighting scores $\{W_1, W_2, \ldots, W_O\}$ to reflect distances between outcomes categories $\{1, \ldots, O\}$, θ_i^{CACE} is defined as $E(Y_{ij}^1 - Y_{ij}^0 | C_{ij} = 1) = \sum_o (W_o \times u_{io}) - \sum_o (W_o \times v_{io})$ (Zhou *et al.* 2021, 2019). Equally spaced scores $\{1, 2, ..., O\}$, their linear transforms, and midranks are reasonable weight choices (Agresti 2013). Future work will add CACE meta-analysis functions for ordinal outcomes, and allow users to choose their preferred weights $\{W_1, W_2, \ldots, W_O\}$. Note that ordinal outcomes lead to more complex correlation structures in the parameters related to response rates, so multivariate prior distributions are necessary to analyze such outcomes. Functions to handle ordinal outcomes and various random effects options are also currently under development and may be included in the package at a later point.

References

Agresti A (2013). *Categorical Data Analysis.* Third edition. John Wiley & Sons, Hoboken, NJ.

Baccini M, Mattei A, Mealli F (2017). "Bayesian inference for causal mechanisms with application to a randomized study for postoperative pain control." *Biostatistics*, **18**(4), 605–617.

Baker SG (2020). "CACE and meta-analysis (Letter to the Editor)." *Biometrics*, **76**(4), 1383–1384.

Bannister-Tyrrell M, Miladinovic B, Roberts CL, Ford JB (2015). "Adjustment for Compliance Behavior in Trials of Epidural Analgesia in Labor Using Instrumental Variable Meta-Analysis." *Journal of Clinical Epidemiology*, **68**(5), 525–533.

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

Clopper CJ, Pearson ES (1934). "The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial." *Biometrika*, **26**(4), 404–413.

Coggeshall S (2017). **noncomplyR**: *Bayesian Analysis of Randomized Experiments with Non-Compliance.* R package version 1.0, URL https://CRAN.R-project.org/package=noncomplyR.

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

Flegal JM, Hughes J, Vats D, Dai N (2017). **mcmcse**: *Monte Carlo Standard Errors for MCMC.* R package version 1.3-2, URL https://CRAN.R-project.org/package=mcmcse.

Frangakis CE, Rubin DB (2002). "Principal Stratification in Causal Inference." *Biometrics*, **58**(1), 21–29.

Freedman LS (1990). "The Effect of Partial Noncompliance on the Power of a Clinical Trial." *Controlled Clinical Trials*, **11**(3), 157–168.

Gelman A, Rubin DB (1992). "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**(4), 457–472.

Gilbert PB, Gabriel EE, Huang Y, Chan IS (2015). "Surrogate endpoint evaluation: Principal stratification criteria and the prentice definition." *Journal of causal inference*, **3**(2), 157–175.

Gordon M, Lumley T (2017). **forestplot***: Advanced Forest Plot Using 'grid' Graphics.* R package version 1.7.2, URL https://CRAN.R-project.org/package=forestplot.

Harville DA (1977). "Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems." *Journal of the American Statistical Association*, **72**(358), 320–338.

Hedges LV, Olkin I (1985). *Statistical Methods for Meta-Analysis.* Academic Press, Orlando, FL.

Hedges LV, Vevea JL (1998). "Fixed- and Random-Effects Models in Meta-Analysis." *Psychological Methods*, **3**(4), 486–504.

Hudgens MG, Halloran ME (2006). "Causal vaccine effects on binary postinfection outcomes." *Journal of the American Statistical Association*, **101**(473), 51–64.

Jiang Y, Small D (2014). **ivpack***: Instrumental Variable Estimation.* R package version 1.2, URL https://CRAN.R-project.org/package=ivpack.

Kasim A, Xiao Z, Higgings S, De Troyer E (2017). **eefAnalytics***: Analysing Education Trials.* R package version 1.0.6, URL https://CRAN.R-project.org/package=eefAnalytics.

Kass RE, Carlin BP, Gelman A, Neal RM (1998). "Markov Chain Monte Carlo in Practice: A Roundtable Discussion." *The American Statistician*, **52**(2), 93–100.

Laird NM, Mosteller F (1990). "Some Statistical Methods for Combining Experimental Results." *International Journal of Technology Assessment in Health Care*, **6**(1), 5–30.

Lewis S, Clarke M (2001). "Forest Plots: Trying to See the Wood and the Trees." *BMJ*, **322**(7300), 1479–1480.

Lin DY, Zeng D (2010). "On the Relative Efficiency of Using Summary Statistics Versus Individual-Level Data in Meta-Analysis." *Biometrika*, **97**(2), 321–332.

Lunn D, Jackson C, Best N, Spiegelhalter D, Thomas A (2012). *The BUGS Book: A Practical Introduction to Bayesian Analysis.* Chapman and Hall/CRC, New York, NY.

Plummer M (2018). **rjags***: Bayesian Graphical Models using MCMC.* R package version 4-8, URL https://CRAN.R-project.org/package=rjags.

Plummer M, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11.

Robert C, Casella G (2004). *Monte Carlo Statistical Methods.* Springer Science & Business Media, New York, NY.

Spiegelhalter DJ, Best NG, Carlin BP, Van Der Linde A (2002). "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **64**(4), 583–639.

Viechtbauer W (2010). "Conducting Meta-Analyses in R with the **metafor** Package." *Journal of Statistical Software*, **36**(3), 1–48.

White H (1982). "Instrumental Variables Regression with Independent Observations." *Econometrica*, **50**(2), 483–499.

Zeger SL, Liang KY, Albert PS (1988). "Models for Longitudinal Data: A Generalized Estimating Equation Approach." *Biometrics*, **44**(4), 1049–1060.

Zhou J, Chu H, Hudgens MG, Halloran ME (2016). "A Bayesian approach to estimating causal vaccine effects on binary post-infection outcomes." *Statistics in medicine*, **35**(1), 53–64.

Zhou J, Hodges JS, Chu H (2020). "Rejoinder to âĂIJCACE and meta-analysis (Letter to the editor)âĂİ by Stuart Baker." *Biometrics*, **76**(4), 1385.

Zhou J, Hodges JS, Chu H (2021). "A Bayesian Hierarchical CACE Model Accounting for Incomplete Noncompliance With Application to a Meta-analysis of Epidural Analgesia on Cesarean Section." *Journal of the American Statistical Association*, pp. 1–13.

Zhou J, Hodges JS, Suri MFK, Chu H (2019). "A Bayesian Hierarchical Model Estimating Cace in Meta-Analysis of Randomized Clinical Trials with Noncompliance." *Biometrics.* In press.

Affiliation:

Jincheng Zhou
Clinical Data Science
Gilead
Foster City, CA 94404, United States of America
E-mail: jeni.zhou9@gilead.com

Jinhui Yang
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455, United States of America
E-mail: yang7004@umn.edu

James S. Hodges, Haitao Chu
Division of Biostatistics
School of Public Health
University of Minnesota
Minneapolis, MN 55455, United States of America
E-mail: hodge003@umn.edu, chux0051@umn.edu

Lifeng Lin
Department of Statistics
Florida State University
Tallahassee, FL 32306, United States of America
E-mail: linl@stat.fsu.edu