rdborrow: An R package for Causal Inference Incorporating External Controls in Randomized Controlled Trials with Longitudinal Outcomes

Lei Shi[1,2], Herbert Pang[2,*], Chen Chen[2], Jiawen Zhu[2]

[*] *Corresponding author*

[1] *Division of Biostatistics, University of California, Berkeley, Berkeley, CA, USA*

[2] *PD Data Sciences, Genentech Inc., South San Francisco, CA, USA*

Summary.
Randomized controlled trials (RCTs) are considered the gold standard for treatment effect evaluation in clinical development. However, designing and analyzing RCTs poses many challenges such as how to ensure the validity and improve the power for hypothesis testing with a limited sample size or how to account for a crossover in treatment allocation. One promising approach to circumvent these problems is to incorporate external controls from additional data sources. This manuscript introduces a new R package called **rdborrow**, which implements several external control borrowing methods under a causal inference framework to facilitate the design and analysis of clinical trials with longitudinal outcomes. More concretely, our package provides an `Analysis` module, which implements the weighting methods proposed in Zhou et al. (2024b), as well as the difference-in-differences and synthetic control methods proposed in Zhou et al. (2024a) for external control borrowing. Meanwhile, our package features a `Simulation` module which can be used to simulate trial data for study design implementation, evaluate the performance of different estimators, and conduct power analysis. In reproducible code examples, we generate simulated data sets mimicking the real data and illustrate the process users can follow to conduct simulation and analysis based on the proposed causal inference methods for randomized controlled trial data incorporating external control data.

Keywords: causal inference, external controls, longitudinal outcome, clinical trial design, R

1. Introduction

Randomized controlled trials (RCTs) are considered the gold standard for evaluating the treatment effect of a therapeutic product on an outcome of interest for a particular disease. In recent years, many efforts have been devoted to exploring the possibility of incorporating additional data sources to facilitate the analysis of RCTs. One promising approach in this realm is to combine the RCT data with the so-called "external controls" (Pocock, 1976; Yap et al., 2022; Chen et al., 2023). The external controls can be a group of people from an earlier time (historical control) or during the same time period (concurrent control) but in another setting (U.S. Food and Drug Administration, 2023). External controls can bring many benefits to medical product development and approval as they can enrich the sample size for rare disease studies to improve power and provide insights for long-term treatment effect evaluation. Many works have explored methodologies for incorporating external data sources, within which Bayesian and causal inference frameworks play a dominating role.

From a Bayesian perspective, Wang et al. (2019) extended the Bayesian power prior approach (Ibrahim and Chen, 2000) for a single-arm study (the current study) to leverage external real-world data. Many follow-up works further extended the power prior approach to other scenarios, say Song et al. (2023) in the application of diagnostic clinical studies, Lu et al. (2023) for an extension to an adaptive design framework. Fu et al. (2023) proposed a dynamic borrowing framework under a Bayesian hierarchical model. Hobbs et al. (2011) proposed the commensurate prior model, which models the commensurability of the current and historical data and adjusts the parameter of the current data conditional on the data from external sources.

Another trend for incorporating external data is led by causal inference methods. For example, Ho et al. (2023) provides a landscape assessment of relevant causal inference frameworks for study design and analysis that generates the use of real-world evidence (RWE) to support regulatory decisions alternative to traditional clinical trials. Li and Yue (2023) provides a tutorial on the propensity score-based methods from the basic idea to their implementation in regulatory settings for causal inference and external data leveraging. Li et al. (2023) derived a doubly robust and locally efficient estimator by combining clinical trial data and external controls and showed its improvement for efficiency. Recently, Zhou et al. (2024b) proposed causal weighting estimators for augmenting trial data with external control data to boost statistical power. Zhou et al. (2024a) proposed several difference-in-differences type methods and a synthetic control method for long-term causal effect estimation when the control group switches to treatments in the open-label extension (OLE) phase.

This paper focuses on software development for analyzing clinical trials with longitudinal outcomes. Many public R packages have implemented methods for such purposes. For example, when external controls are not considered, **mmrm** (Sabanes Bove et al., 2024) implemented the mixed models for repeated measures (MMRM), which are a popular choice for analyzing longitudinal continuous outcomes in randomized controlled trials and beyond; see Cnaan et al. (1997); Mallinckrod et al. (2008). **mmrm** implements MMRM based on the marginal linear model without random effects using Template Model Builder (TMB) which enables fast and robust model fitting. **ltmle** by Lendle et al. (2017) implements the Targeted Maximum Likelihood Estimation (TMLE) of treatment/censoring specific mean outcome or marginal structural model for point-treatment and longitudinal data. These packages target more on the analysis but less on the design perspective.

When external control borrowing is considered, there are also several packages available. For example, **psborrow** provides a tool that aims to help evaluate the effect of external borrowing using an integrated approach described in Lewis et al. (2019) combining propensity score and Bayesian dynamic borrowing methods. As a successor, **psborrow2** (Gravestock, 2024) is an R package for conducting Bayesian dynamic borrowing analyses (Viele et al., 2014). It also provides an additional feature for trial simulation and evaluation, which compares different trial and borrowing characteristics in a unified way in simulation studies to inform trial design. **bayesDP** provides a package of functions implementing data augmentation using the Bayesian discount prior method for single-arm and two-arm clinical trials, as described in Haddad et al. (2017). **BACCT** implements the Bayesian Augmented Control (BAC) method under clinical trial setting. **hctrial** provides functions for designing phase II clinical trials adjusting for the heterogeneity of the population using known subgroups or historical controls. Eggleston et al. (2021) introduces a package, **BayesCTDesign**, for two-arm Bayesian designs that might include historical control data, which allows using simulation to estimate trial design characteristics under user-defined scenarios. Lee et al. (2024) showcased the use of the R package **genRCT** for treatment effect evaluation, which implements a set of statistical methods, termed "genRCT", for improving the generalizability of the trial using calibration weighting to enforce the covariates balance between the RCT and observational study. While the above packages serve as powerful tools for external data borrowing, the number of the packages is relatively sparse. Besides, most existing packages focus on a Bayesian approach instead of a causal inference perspective.

This manuscript introduces a new R package **rdborrow**, which implements a set of causal inference methods that incorporate external controls in clinical trials with longitudinal outcomes. The package is developed on GitHub at `https://github.com/pathwayrf/rdborrow`. The methods are based on two recent proposals for external control borrowing (Zhou et al., 2024a,b). The package highlights several crucial features: (1) It provides an `Analysis` module that implements the external control borrowing methods in randomized controlled trials with longitudinal outcomes to facilitate the estimation and inference, which includes weighting-based methods, difference-in-differences methods, and a synthetic control method. (2) It incorporates a `Simulation` module that can be utilized to simulate randomized controlled trials and external control data, in addition to performing Monte Carlo simulations to evaluate the performance of various estimators and inference methods. It also supports power calculation, which can guide the determination of sample size at the planning stage of a clinical trial. (3) `S4` classes are built for implementation, which gives an organized structure of the package through inheritance, function overloading, etc., which leaves space for future maintenance and development.

The paper is organized as follows. Section 2 introduces the causal inference framework for external control borrowing. Section 3 depicts the architecture of the **rdborrow** package, including the main classes and functions and the high-level usage of the features. Section 4 uses a simulated data example as well as simulated datasets to illustrate the use of the **rdborrow** package. Section 5 summarizes the results and discusses potential extensions of the package.

2. Methods

2.1. *Statistical background and a causal inference framework*

The presentation we adopted is based on the potential outcome framework (Imbens and Rubin, 2015). The dataset at hand is composed of two parts: randomized controlled trial data and external control data.

2.1.1. *Randomized controlled trial data*

The first part contains data for a randomized controlled trial enrolling n patients, denoted by $\mathcal{R}$ (for randomized), assessing the efficacy of a binary treatment on longitudinal outcomes $\mathbf{Y} = (Y_1, \ldots, Y_T)$ over T time points. The period I, for $t \in \mathcal{T}_1 = (0, T_1]$, is a placebo-controlled phase, after which, during period II (i.e., OLE phase or $t \in \mathcal{T}_2 = (T_1, T_2]$), all patients from the control group are switched to the treatment. Due to the special study design, for any patient i, a vector $\mathbf{A}_i = (A_{pi}, A_{oi})$ to indicate the sequence of treatment assignments:

$$\mathbf{A}_i = \begin{cases} (1,1), & \text{if Unit } i \text{ received treatment in both } \mathcal{T}_1 \text{ and } \mathcal{T}_2; \\ (0,1), & \text{if Unit } i \text{ received control in } \mathcal{T}_1 \text{ and treatment in } \mathcal{T}_2. \end{cases} \tag{1}$$

Due to the characteristics of the OLE phase, the $A_{oi} = 1$ for all patients in RCT. Therefore, A_{pi} determines the treatment arm of a given RCT patient. $\mathbf{Y}_i^{(\mathbf{a}_i)} = (Y_{i1}^{(\mathbf{a}_i)}, \ldots, Y_{iT}^{(\mathbf{a}_i)})$ are the potential outcomes had the patient i received the sequence of treatment $\mathbf{a}_i$. $\mathbf{Y}_i$ denotes the observed outcomes. Under the assumption of Stable Unit Treatment Values Assumption (Rubin, 1980, abbreviated as SUTVA), the observed sequence of outcomes equals the potential outcomes under the treatment sequence that is actually received, i.e., $\mathbf{Y}_i = \mathbf{Y}_i^{(\mathbf{A}_i)}$. At the initial randomization, the probability of treatment assignment is denoted as $\pi_A = P_{\mathcal{R}}(A_1 = 1)$ (could depend on a subset of $\mathbf{X}$ for stratified randomization), where $\mathbf{X}_i$ is a p-dimensional vector of measured baseline covariates for patient i. Let n_1 be the number of treated patients and n_0 be the number of (initially) control patients.

2.1.2. *External control data*

The second part is an external control data, denoted by $\mathcal{E}$ (for external controls), containing m patients, who are never treated by the experimental treatment from the trial of interest. Hence, we use $\mathbf{A}_i = (0, 0)$ to indicate the treatment status for all the subjects in the external control data. The observed outcomes, $\mathbf{Y}_i$'s, equal the potential outcomes under control $\mathbf{Y}_i = \mathbf{Y}_i^{(0,0)}$. We assume the same outcome measures were captured for the external control patients in the same manner as the trial of interest. Ideally, their outcomes were also repeatedly measured on T discrete time points over a time duration $t \in \mathcal{T}_1 \cup \mathcal{T}_2 = (0, T_2]$, though it is allowed that the exact time at which the external controls were observed comes earlier than the trial of interest.

2.1.3. Probability models and notations

We assume that the RCT data and EC data are from different populations: each trial patient $i \in \mathcal{R}$ is sampled from the current RCT population described by $P_{\mathcal{R}}(\mathbf{X}, \mathbf{A}, \mathbf{Y}^{(1,1)}, \mathbf{Y}^{(0,1)}, \mathbf{Y}^{(0,0)})$, which is the target population of interest, while each external control patient $i \in \mathcal{E}$ is sampled from $P_{\mathcal{E}}(\mathbf{X}, \mathbf{A} = (0,0), \mathbf{Y}^{(0,0)})$, labeled from $i = n+1$ to $n+m$. $P_{\mathcal{E}}$ could represent the distribution of a larger disease population in the real world or a population targeted by another trial. The sample size of the available external control data m could potentially be large. We use S_i to denote trial participation status, with $S_i = 1$ for $i \in \mathcal{R}$ and $S_i = 0$ for $i \in \mathcal{E}$. Let $\pi_S(\mathbf{X})$ and π_S denote the true conditional and marginal probability of trial participation, respectively. The parameter-indexed $\pi_S(\mathbf{X}; \beta)$ imposes a working model for estimating $\pi_S(\mathbf{X})$. We use $\mu_{\mathcal{R}}(\mathbf{a})$ and $\mu_{\mathcal{E}}(\mathbf{a})$ to denote the expected potential outcomes under treatment sequence a for the trial population and the external control population, respectively. $\mu(\mathbf{X}, S, \mathbf{A}, t)$ represents the true conditional expected observed outcomes given the covariates, population, initial treatment assignment, and time. The parameter-indexed version $\mu(\mathbf{X}, S, \mathbf{A}, t; \gamma)$ gives an assumed working model (longitudinal model for the time component) for estimating $\mu(\mathbf{X}, S, \mathbf{A}, t)$. Throughout the paper, we will use indices $\mathcal{R}$ and $\mathcal{E}$ to denote quantities (probability, expectation, variance, covariance) taken with respect to these populations, for example, $E_{\mathcal{R}}(\cdot)$ for an expectation over $P_{\mathcal{R}}(\cdot)$.

2.2. Primary analysis with weighting methods

Primary analysis targets the collected data from the first period of the study (i.e., the placebo-controlled phase). The causal estimands are given by the trial population (the target population) average treatment effect (ATE):

$$\tau_t = \mathbb{E}_{\mathcal{R}}[Y_t^{(1,1)} - Y_t^{(0,0)}] = \mu_{\mathcal{R},t}^{(1,1)} - \mu_{\mathcal{R},t}^{(0,0)}, \quad t \in \mathcal{T}_1. \tag{2}$$

(2) is a time-indexed ATE. For estimating (2), Zhou et al. (2024b) proposed the External Controls Enhanced (Augmented) Inverse Probability Weighting (EC-(A)IPW for short) estimator, which is given by

$$\widehat{\tau}^{\text{EC-IPW}} = \frac{\sum_{i\in\mathcal{R}} A_{pi}\mathbf{Y}_i\widehat{w}_{11}(\mathbf{X}_i)}{\sum_{i\in\mathcal{R}} A_{pi}\widehat{w}_{11}(\mathbf{X}_i)} - \left\{(1-w)\frac{\sum_{i\in\mathcal{R}}(1-A_{pi})\mathbf{Y}_i\widehat{w}_{10}(\mathbf{X}_i)}{\sum_{i\in\mathcal{R}}(1-A_{pi})\widehat{w}_{10}(\mathbf{X}_i)} + w\frac{\sum_{i\in\mathcal{E}}\mathbf{Y}_i\widehat{w}_0(\mathbf{X}_i)}{\sum_{i\in\mathcal{E}}\widehat{w}_0(\mathbf{X}_i)}\right\}. \tag{3}$$

Here $\widehat{w}_{11}(\mathbf{X})$ and $\widehat{w}_{10}(\mathbf{X})$ are the estimated propensity score for the (initially) treated arm and control arm, respectively (Zhou et al., 2024b). $\widehat{w}_0(\mathbf{X})$ is an estimator for the density ratio of the covariates within the randomized trial population and the external control population:

$$w_0(\mathbf{X}) = \frac{P_{\mathcal{R}}(\mathbf{X})}{P_{\mathcal{E}}(\mathbf{X})} = \frac{\pi_S(\mathbf{X})(1-\pi_S)}{(1-\pi_S(\mathbf{X}))\pi_S}. \tag{4}$$

w is a borrowing weight that determines the extent of external control borrowing to be applied for constructing the estimators. When $w = 0$, the estimator only utilizes the RCT data. When $w > 0$, the external controls are also incorporated. In practice, the borrowing weight can be specified by researchers based on domain knowledge or chosen in a data-adaptive fashion as proposed in Zhou et al. (2024b).

When $\pi_S(\mathbf{X})$ follows a logit model

$$\text{logit}(\pi_S(\mathbf{X})) = \mathbf{z}^\top\boldsymbol{\beta}, \quad \mathbf{z} = g(\mathbf{X}), \tag{5}$$

an estimator for $\boldsymbol{\beta}$ is given by solving the estimating equations:

$$\sum_{i\in\mathcal{R}\cup\mathcal{E}} \boldsymbol{\Psi}(\mathcal{O}_i; \widehat{\boldsymbol{\theta}}) = 0.$$

Then the IPW estimator $\widehat{\tau}^{\text{EC-IPW}}(w)$ with a given weight w as well as a variance estimator can be obtained by a transformation of the solutions to the estimation equation, as detailed in Zhou et al. (2024b). Alternatively, bootstrap methods can be used to perform inference. Moreover, the EC-IPW estimator can be further augmented by incorporating an estimated outcome model, which is called EC-AIPW. More technical discussions on the construction of the estimator and confidence intervals for EC-IPW and EC-AIPW can be found in Zhou et al. (2024b).

2.3. *Open-label extension phase analysis with difference-in-differences and synthetic control methods*

For the OLE phase analysis, the goal is to evaluate the long-term causal effect beyond the placebo-controlled phase. The causal estimands of interest are given by the average treatment effects in the OLE phase $\mathcal{T}_2$:

$$\tau_t = \mathbb{E}_{\mathcal{R}}\left[Y_t^{(1)} - Y_t^{(0)}\right] = \mu_{\mathcal{R},t}^{(1)} - \mu_{\mathcal{R},t}^{(0)}, \quad t \in \mathcal{T}_2. \tag{6}$$

2.3.1. *Difference-in-differences type methods*

Difference-in-differences (DID) methods are widely used in treatment effect estimation to study the differential effect of a treatment on a treatment arm versus a control group. Zhou et al. (2024a) adapted the classical DID methods to the current external control setting under a modified conditional parallel trends assumption.

Zhou et al. (2024a) proposed three DID-based estimators. The first one is based on outcome regression:

$$\begin{aligned}\widehat{\tau}_t^{\text{DID-EC-OR}} = \frac{1}{n}\sum_{i\in\mathcal{R}}[&(\mu(\mathbf{X}_i, S=1, A=1, t) - \overline{\mu}(\mathbf{X}_i, S=1, A=0, \mathcal{T}_1)) \\ &-(\mu(\mathbf{X}_i, S=0, A=0, t) - \overline{\mu}(\mathbf{X}_i, S=0, A=0, \mathcal{T}_1))].\end{aligned}$$

The second one is based on inverse probability weighting:

$$\begin{aligned}\widehat{\tau}_t^{\text{DID-EC-IPW}} =&\sum_{i\in\mathcal{R}}\left\{\frac{A_{pi}\widehat{w}_{11}(\mathbf{X}_i)\mathbf{Y}_{it}}{\sum_{i\in\mathcal{R}} A_{pi}\widehat{w}_{11}(\mathbf{X}_i)} - \frac{(1-A_{pi})\widehat{w}_{10}(\mathbf{X}_i)\overline{\mathbf{Y}}_i(\mathcal{T}_1)}{\sum_{i\in\mathcal{R}}(1-A_{pi})\widehat{w}_{10}(\mathbf{X}_i)}\right\} \\ &-\sum_{i\in\mathcal{E}}\left\{\frac{\widehat{w}_0(\mathbf{X}_i)}{\sum_{i\in\mathcal{E}}\widehat{w}_0(\mathbf{X}_i)}(\mathbf{Y}_{it} - \overline{\mathbf{Y}}_i(\mathcal{T}_1))\right\}.\end{aligned}$$

Moreover, combining the outcome modeling and inverse probability weighting leads to the augmented IPW estimator $\widehat{\tau}_t^{\text{DID-EC-AIPW}}$, which is omitted here. How to perform inference for these estimators is still an area with many ongoing research works. In this work and the R package implementation, we use bootstrap to construct confidence intervals.

2.3.2. *Synthetic control method*

The synthetic control method (Abadie et al., 2010, 2015) is a widely used approach for evaluating the effects of a certain intervention on a longitudinal outcome before/after the intervention.

In the RCT setting, the synthetic control idea is to find a few external control patients for each RCT control patient that share similar values of $\mathbf{X}$ and the outcomes in the period I (i.e. Y_t for $t \in \mathcal{T}_1$), such that the weighted average of the selected external control patients (i.e. the "synthetic control") are as similar as possible to the RCT control patient under consideration, in both covariates $\mathbf{X}$ and period I trajectories Y_t for $t \in \mathcal{T}_1$. The way to find those "synthetic control" patients based on the external control data is by a matching approach that solves an optimization problem minimizing an objective function measuring the discrepancies. More implementation details and interpretations can be obtained from Zhou et al. (2024a).

3. Software

The whole **rdborrow** package contains two main modules: an Analysis module, which is designed for implementing the proposed external control borrowing methods for estimating causal effects for longitudinal data; and a Simulation module, which provides a systematic way to perform simulation studies to evaluate different data-generating processes and different borrowing methods.

3.1. Analysis Module

The analysis module contains several components (see Figure 1). The major component is an S4 class named '`analysis_obj`' (Figure 2), which wraps up several analytical components into a single module, including the input dataset, indicators of variable names, the specification of methods, etc. Two subclasses, '`analysis_primary_obj`' and '`analysis_OLE_obj`', are inherited from '`analysis_obj`' to target more specific settings, i.e., primary analysis and OLE phase analysis, respectively. Within the '`analysis_obj`' class, the methods are set up as an instance from another class called '`method_obj`'. The '`method_obj`' contains several slots, including the method name, specifications for bootstrap, etc.

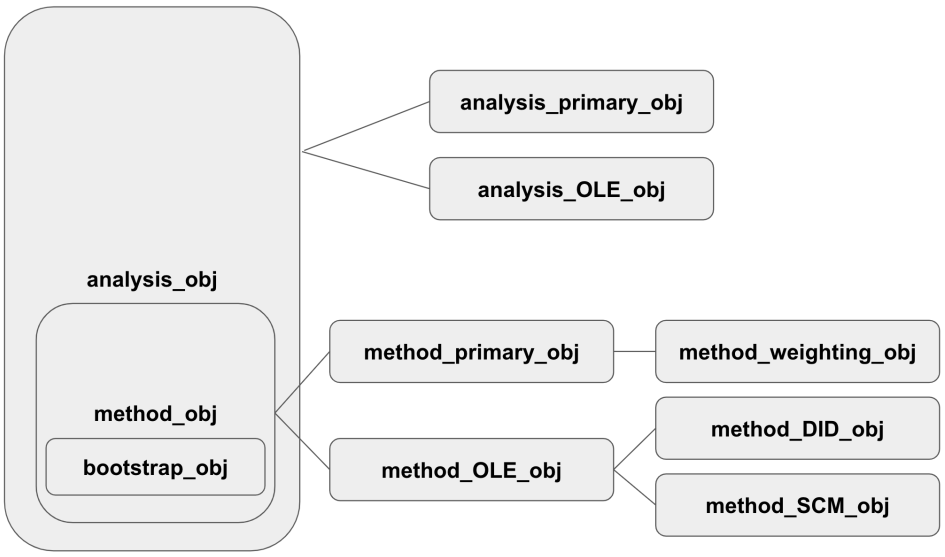

Fig. 1. Analysis Module

The method class, '`method_obj`', is used to set up the method specifications. Similar to the '`analysis_obj`', two more subclasses are inherited from the '`method_obj`' class: '`method_primary_obj`' and '`method_OLE_obj`', targeting primary analysis and OLE phase analysis, respectively. For the primary analysis, the methods introduced in Section 2.2 are weighting-based external control borrowing, which further extends to a subclass '`method_weighting_obj`'. For the OLE phase analysis, the DID methods and synthetic control methods introduced in Section 2.3 are implemented as '`method_DID_obj`' and '`method_SCM_obj`', respectively. The inheritance structure enables building specific modules for different analytical questions while leaving a flexible space for future methodology development and implementation.

Another important class in the package is the '`bootstrap_obj`' (Figure 4), which wraps up several bootstrap specifications for analysis, including the number of replications and the type of bootstrap confidence intervals to be used for inference. The '`bootstrap_obj`' class will set up the parameter specifications for utilizing the function `boot()` from the package **boot** (Angelo Canty and B. D. Ripley, 2024; A. C. Davison and D. V. Hinkley, 1997). The '`bootstrap_obj`'

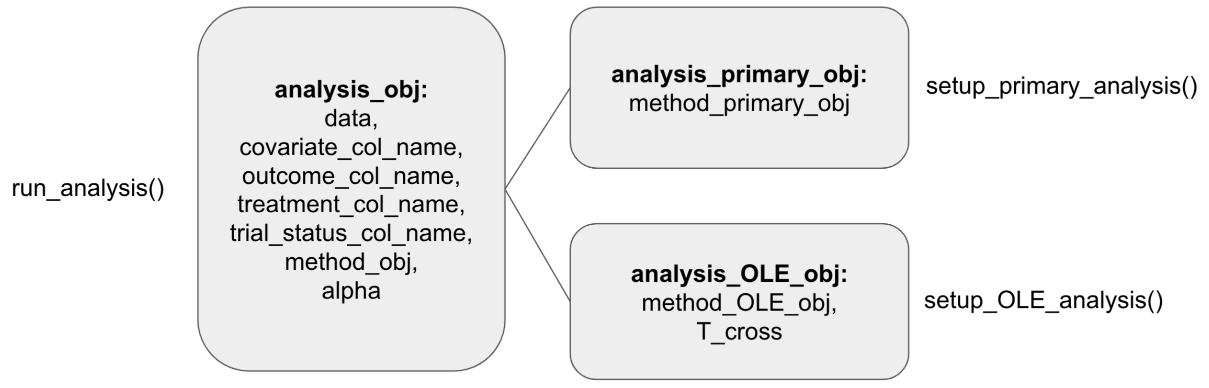

Fig. 2. Analysis Class

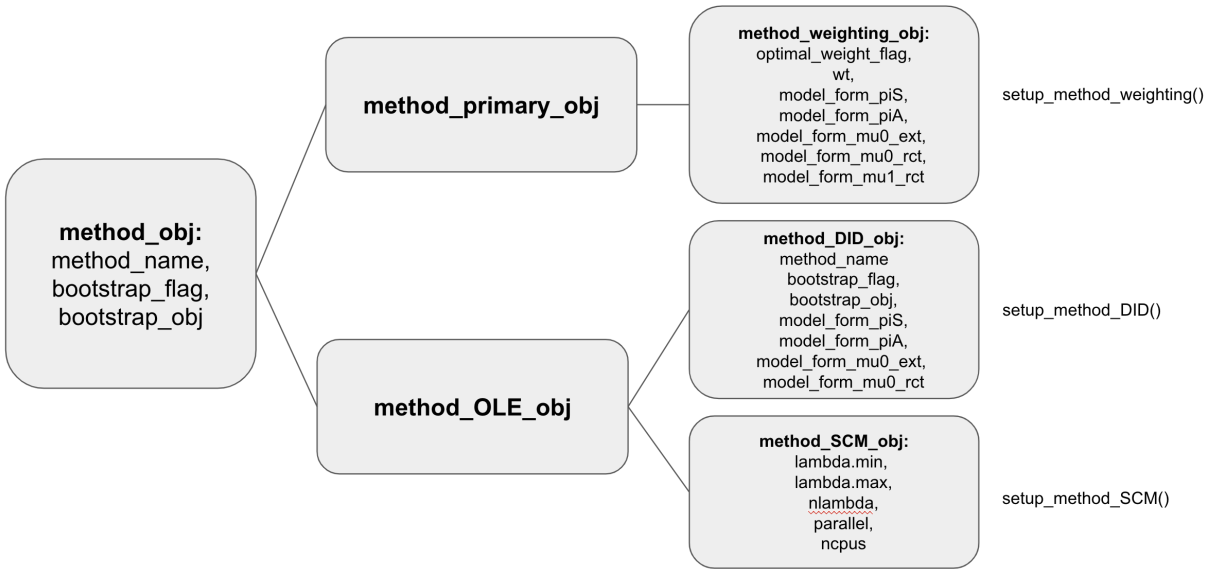

Fig. 3. Method Class

class appears as a slot in the '`method_obj`'. For primary analysis, bootstrap is an alternative inference method to the parametric methods. For the OLE phase analysis, as the parametric inference methods have not been fully developed, bootstrap is a crucial part of inference.

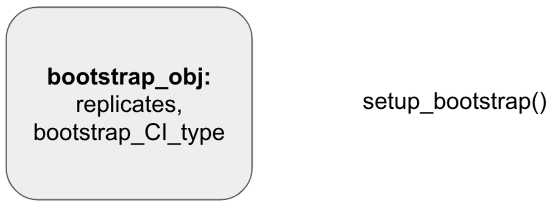

Fig. 4. Bootstrap Class

For more details about the classes and functions in the Analysis Module, we refer interested readers to Section 4.1 and Section A in the Appendix for more details.

3.2. Simulation Module

The simulation module is built for evaluating the performance of different model setups and implementation of methods through Monte Carlo simulation (Figure 5). It contains several components. The major component is an S4 class named '`simulation_obj`', which wraps up several parts, including two data lists, indicators of variable names, a method list, etc. Similar to the Analysis Module, two subclasses further branched from the '`simulation_obj`' class: '`simulation_primary_obj`' for the primary analysis, and '`simulation_OLE_obj`' for the OLE phase analysis. Figure 6 provides a detailed presentation of the structure of the simulation class.

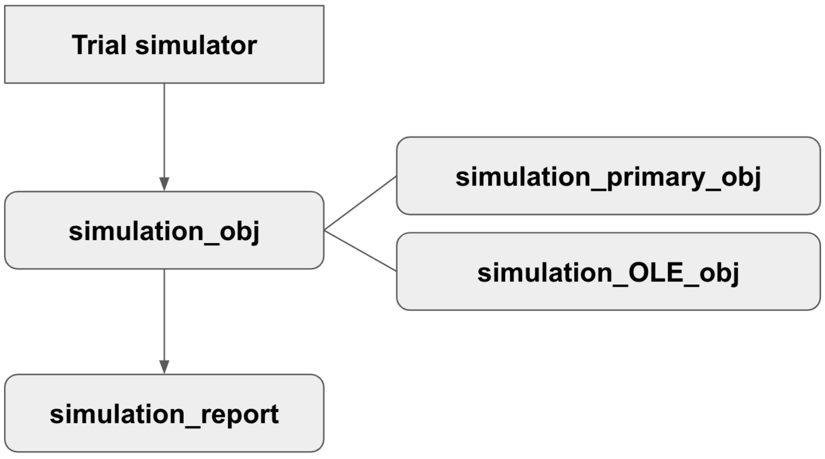

Fig. 5. Simulation Module

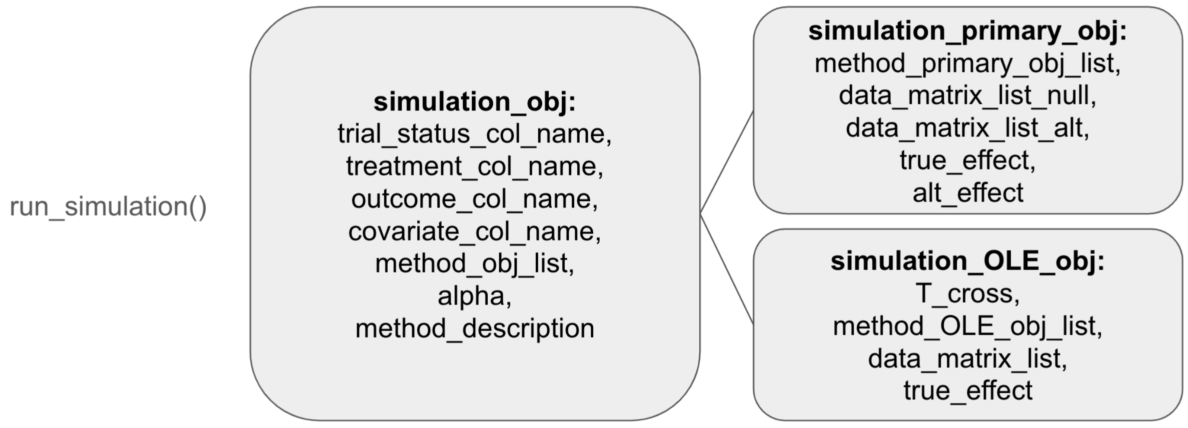

Fig. 6. Simulation Class

The simulation results are obtained by feeding an instance of '`simulation_obj`' into the function `run_simulation()`, which returns an object from another class, '`simulation_report_obj`' (Figure 7). '`simulation_report_obj`' embeds several evaluation metrics, including `bias`, `variance`, `mse`, `coverage`, `type_I_error`, as well as `power`, which are empirical summary statistics generated from Monte Carlo simulations. The function `show()` is overloaded to generate a report for the simulation class.

For more details of the Simulation module, see Section 4.2 and Section B in the Appendix for more discussions.

simulation_report_obj:
Bias,
MSE,
Variance,
Coverage
show()

Fig. 7. Simulation Report Class

4. Workflow for rdborrow

In this section, we provide a detailed illustration for the usage of **rdborrow**. Section 4.1 instructs on the use of **rdborrow** for performing analysis on a simulated dataset. Section 4.2 presents the codes and results for performing simulations using **rdborrow** to evaluate different methods and facilitate the study of new designs.

4.1. Analysis based on one simulated dataset

In the first part, we analyze the trial data from the SUNFISH study. The SUNFISH Trial (NCT02908685) is a randomized study assessing efficacy, safety, pharmacokinetics, pharmacodynamics, and tolerability of risdiplam in patients aged 2-25 years with confirmed 5q autosomal recessive type 2 or type 3 spinal muscular atrophy (SMA). This study consisted of two phases, starting with a 12-month placebo-controlled phase, where patients were randomly assigned to receive either daily oral risdiplam or a placebo. During this phase, their Motor Function Measure (MFM) and other important clinical indicators were regularly monitored. In the subsequent phase, control group participants were also given the treatment. Extended follow-up past 12 months was necessary to gauge long-term treatment benefits. During this phase, the control arm from another trial (the olesoxime trial, NCT01302600) (Berry et al., 2010) serves as external controls to augment the SUNFISH study (McIver et al., 2023).

We created a simulated dataset based on the characteristics of the real trial data. Specifically, the synthetic data has the same number and type of baseline covariates and outcomes as the true data. Moreover, the generating process utilized some important summary statistics from the true data, such as the distribution of the baseline covariates, the model coefficients, the noise levels of the outcomes, etc. For example, the baseline covariate x_5 in the simulated data represents the baseline MFM 32 (which is the outcome of interest); y_1 and y_2 represent change from baseline in MFM 32 during the placebo-controlled phase while y_3 and y_4 represent change from baseline in MFM 32 during the OLE phase. The true treatment effects for the four follow-ups are set up as $(0.0, 1.0, 2.0, 5.0)$, respectively. The simulated data is built in the data folder and can be loaded by simply calling the name `SyntheticData`:

```
> head(SyntheticData)
  x1 x2 x3 x4       x5 A S T_cross         y1         y2         y3        y4
1  1  1  1  9 54.59836 1 1       2  3.4512377 -0.7642287 -2.4713591  3.935466
2  0  1  0  8 33.08006 1 1       2  0.4518106  6.3516296  4.5231869 -0.198674
3  1  1  1  7 48.51653 0 1       2  3.0532714 -2.0453190  5.9064870 -1.374919
4  1  1  1 15 31.68766 1 1       2 -9.1183948  0.2304339  4.7858172  8.490757
5  1  1  1 12 29.98495 0 1       2 -1.4270057  1.5878794  3.7006101  9.449632
6  1  1  0  7 46.08991 0 1       2 -2.6967072 -0.6130288  0.7482786 -2.413717
```

4.1.1. Primary Analysis

We first analyze the synthetic data over the placebo-controlled phase. The discussion for this section corresponds to the vignette file `primary_analysis_workflow.Rmd` in the **rdborrow** package. We walk through IPW and AIPW methods, both with and without data-adaptive external control borrowing. For inference, we present both parametric and bootstrap inference results. At the end of this section, we add discussions to summarize the performance of the methods.

IPW. We can analyze the data with IPW methods. The first strategy is to use only the within-trial data and ignore the external control data, which corresponds to a weight of zero for external borrowing:

```
method_weighting_obj = setup_method_weighting(
  method_name = "IPW",
  optimal_weight_flag = F,
  wt = 0,
  model_form_piS = "S ~ x1 + x2 + x3 + x4 + x5")

analysis_primary_obj = setup_analysis_primary(
  data = SyntheticData,
  trial_status_col_name = "S",
  treatment_col_name = "A",
  outcome_col_name = c("y1", "y2"),
  covariates_col_name = c("x1", "x2", "x3", "x4", "x5"),
  method_weighting_obj = method_weighting_obj)

res = run_analysis(analysis_primary_obj)
```

We get the following output:

```
> res
$results
       point_estimates standard_deviation lower_CI_normal upper_CI_normal
tau1     -0.02808704          0.5428382      -1.0920304        1.035856
tau2      0.40959558          0.5627990      -0.6934702        1.512661

$borrow_weight
[1] 0
```

From the output, we can see that the confidence intervals achieved good coverage for the ground truth for both the first and second time points.

Alternatively, we can decide the weight in a data-driven fashion:

```
method_weighting_obj = setup_method_weighting(
  method_name = "IPW",
  optimal_weight_flag = T,
  model_form_piS = "S ~ x1 + x2 + x3 + x4 + x5")

analysis_primary_obj = setup_analysis_primary(
  data = SyntheticData,
  trial_status_col_name = "S",
  treatment_col_name = "A",
  outcome_col_name = c("y1", "y2"),
  covariates_col_name = c("x1", "x2", "x3", "x4", "x5"),
  method_weighting_obj = method_weighting_obj)

res = run_analysis(analysis_primary_obj)
```

The analysis generates the following results:

```
> res
$results
     point_estimates standard_deviation lower_CI_normal upper_CI_normal
tau1      -0.1971969          0.5193265      -1.2150581       0.8206643
tau2       0.4697209          0.5382500      -0.5852297       1.5246714

$borrow_weight
[1] 0.1475196
```

Similarly, the confidence intervals achieved good coverage for the ground truth for both the first and second time points, and we are getting narrower confidence bands by incorporating the external controls.

AIPW. We can further incorporate an estimated outcome model to facilitate the analysis. We can only use the within-trial data:

```
method_weighting_obj = setup_method_weighting(
  method_name = "AIPW",
  optimal_weight_flag = F,
  wt = 0,
  model_form_piS = "S ~ x1 + x2 + x3 + x4 + x5",
  model_form_mu0_ext = c("y1 ~ x1 + x2 + x3 + x4 + x5",
                         "y2 ~ x1 + x2 + x3 + x4 + x5"))

analysis_primary_obj = setup_analysis_primary(
  data = SyntheticData,
  trial_status_col_name = "S",
  treatment_col_name = "A",
  outcome_col_name = c("y1", "y2"),
  covariates_col_name = c("x1", "x2", "x3", "x4", "x5"),
  method_weighting_obj = method_weighting_obj)

res = run_analysis(analysis_primary_obj)
```

The following results are obtained:

```
> res
$results
     point_estimates standard_deviation lower_CI_normal upper_CI_normal
tau1      -0.4361151          0.5744615      -1.5620390       0.6898088
tau2       0.4422248          0.5756078      -0.6859457       1.5703954

$borrow_weight
[1] 0
```

Alternatively, we can incorporate the external controls with a data-adaptive strategy:

```
method_weighting_obj = setup_method_weighting(
  method_name = "AIPW",
  optimal_weight_flag = T,
  model_form_piS = "S ~ x1 + x2 + x3 + x4 + x5",
  model_form_mu0_ext = c("y1 ~ x1 + x2 + x3 + x4 + x5",
                         "y2 ~ x1 + x2 + x3 + x4 + x5"))
```

```
analysis_primary_obj = setup_analysis_primary(
  data = SyntheticData,
  trial_status_col_name = "S",
  treatment_col_name = "A",
  outcome_col_name = c("y1", "y2"),
  covariates_col_name = c("x1", "x2", "x3", "x4", "x5"),
  method_weighting_obj = method_weighting_obj)

res = run_analysis(analysis_primary_obj)
```

This generates the following results:

```
> res
$results
     point_estimates standard_deviation lower_CI_normal upper_CI_normal
tau1      -0.5463256          0.5490709      -1.6224848       0.5298336
tau2       0.5401750          0.5565028      -0.5505505       1.6309004

$borrow_weight
[1] 0.1475196
```

Compared with the ground truth, AIPW confidence intervals also achieved good coverage for both time points.

Bootstrap inference. We can also apply Bootstrap to construct confidence intervals. For IPW, we use the following setup:

```
bootstrap_obj = setup_bootstrap(
  replicates = 2e3,
  bootstrap_CI_type = "perc"
)

method_weighting_obj = setup_method_weighting(
  method_name = "IPW",
  optimal_weight_flag = T,
  bootstrap_flag = T,
  bootstrap_obj = bootstrap_obj,
  wt = 0,
  model_form_piS = "S ~ x1 + x2 + x3 + x4 + x5")

analysis_primary_obj = setup_analysis_primary(
  data = SyntheticData,
  trial_status_col_name = "S",
  treatment_col_name = "A",
  outcome_col_name = c("y1", "y2"),
  covariates_col_name = c("x1", "x2", "x3", "x4", "x5"),
  method_weighting_obj = method_weighting_obj)

res = run_analysis(analysis_primary_obj)
```

The following output is generated:

```
> res
$results
     point_estimates standard_deviation lower_CI_boot upper_CI_boot
```

```
tau1      -0.1971969          0.5193265    -1.2282308      0.9219488
tau2       0.4697209          0.5382500    -0.6203081      1.5471742

$borrow_weight
[1] 0.1475196
```

And for AIPW, we use the other setup:

```
bootstrap_obj = setup_bootstrap(
  replicates = 2e3,
  bootstrap_CI_type = "perc"
)

method_weighting_obj = setup_method_weighting(
  method_name = "AIPW",
  optimal_weight_flag = T,
  wt = 0,
  bootstrap_flag = T,
  bootstrap_obj = bootstrap_obj,
  model_form_piS = "S ~ x1 + x2 + x3 + x4 + x5",
  model_form_mu0_ext = c("y1 ~ x1 + x2 + x3 + x4 + x5",
                         "y2 ~ x1 + x2 + x3 + x4 + x5"))

analysis_primary_obj = setup_analysis_primary(
  data = SyntheticData,
  trial_status = "S",
  treatment = "A",
  outcome = c("y1", "y2"),
  covariates = c("x1", "x2", "x3", "x4", "x5"),
  method_weighting_obj = method_weighting_obj)


res = run_analysis(analysis_primary_obj)
```

We obtain the following results:

```
> res
$results
     point_estimates standard_deviation lower_CI_boot upper_CI_boot
tau1      -0.5463256          0.5490709    -1.1798523      0.8428615
tau2       0.5401750          0.5565028    -0.5700838      1.5651988

$borrow_weight
[1] 0.1475196
```

Comparing the results. Based on the outputs, the IPW and AIPW methods are quite consistent for evaluating the simulated data. Both achieved good coverage for the ground truth for the two time points during the placebo-controlled phase. By incorporating the external controls with a data-adaptive weight, the confidence band is further shortened.

4.1.2. Open-label extension phase analysis

In this subsection, we analyze the simulation data in the OLE phase. The discussion for this section corresponds to the vignette file `OLE_analysis_workflow.Rmd` in the **rdborrow** package.

Difference-in-differences with IPW. The first method is based on DID with IPW:

```
bootstrap_obj = setup_bootstrap(
  replicates = 2e3,
  bootstrap_CI_type = "perc"
)

method_DID_obj = setup_method_DID(
  method_name = "IPW",
  bootstrap_flag = T,
  bootstrap_obj = bootstrap_obj,
  model_form_piS = "S ~ x1 + x2 + x3 + x4 + x5",
  model_form_piA = "A ~ x1 + x2 + x3 + x4 + x5")

analysis_OLE_obj = setup_analysis_OLE(
  data = SyntheticData,
  trial_status_col_name = "S",
  treatment_col_name = "A",
  outcome_col_name = c("y1", "y2", "y3", "y4"),
  covariates_col_name = c("x1", "x2", "x3", "x4", "x5"),
  T_cross = 2,
  method_OLE_obj = method_DID_obj)

res = run_analysis(analysis_OLE_obj)
```

The following outputs are obtained by calling `res`:

```
> res
     point_estimates lower_CI_boot upper_CI_boot
tau3        2.075926     0.1003525      4.021258
tau4        4.389438     1.1176003      7.495404
```

For DID with AIPW or outcome regression, the estimation and inference can be implemented similarly, and we omit the details here.

SCM. The second method for analyzing the data in the OLE phase is SCM. The computation for SCM is more time-consuming because the Bootstrap inference for SCM requires repeating the synthetic control construction many times, each involving a relatively large number of iterations. To reduce the time complexity, we incorporated the parallelization feature in the **boot** package, which improves the computational performance by utilizing multiple cores simultaneously. The following code sets up an analysis of the simulated dataset based on SCM, with 200 rounds of Bootstrap resampling conducted over four cores.

```
bootstrap_obj = setup_bootstrap(
  replicates = 200,
  bootstrap_CI_type = "perc"
)

method_SCM_obj = setup_method_SCM(
  method_name = "SCM",
  bootstrap_flag = T,
  bootstrap_obj = bootstrap_obj,
  lambda.min = 0,
  lambda.max = 1e-3,
  nlambda = 10,
  parallel = "multicore",
  ncpus = 4)
```

```
analysis_OLE_obj = setup_analysis_OLE(
  data = SyntheticData,
  trial_status_col_name = "S",
  treatment_col_name = "A",
  outcome_col_name = c("y1", "y2", "y3", "y4"),
  covariates_col_name = c("x1", "x2", "x3", "x4", "x5"),
  T_cross = 2,
  method_OLE_obj = method_SCM_obj)

run_analysis(analysis_OLE_obj)
```

The above code generates the following output:

```
       point_estimates   lower_CI_boot     upper_CI_boot
tau3       2.134082        0.5927815          4.001694
tau4       3.950783        1.5104049          7.142396
```

From the results, we can see that the point estimates are close to the truth ($\tau_3^\star = 2.0$, $\tau_4^\star = 5.0$), and the Bootstrap confidence intervals achieve good coverage. In terms of computation time, when evaluated on a 2020 Macbook Pro with an Apple M1 chip, the parallelized Bootstrap with four cores took 9 minutes to run, compared to around 32 minutes when no parallelization is adopted, which suggests a sharp advantage of the multicore parallelization.

4.2. *Monte Carlo simulation for method evaluation and trial designing*

In this section, we use some examples to demonstrate the usage of the simulation module as well as the performance of the estimators.

4.2.1. *Simulating Trial Data*

We simulate a list of randomized controlled trials and external controls with $N = 300$ patients. The record for the i-th patient is encoded into a data point $(\mathbf{X}, S, \mathbf{A}, \mathbf{Y})$. Here $\mathbf{X}$ contains $p = 5$ baseline covariates that are generated mimicking the real study data and coupled with a Gaussian copula. The simulated study consists of four repeated post-baseline measurements ($T = 4$). The first two time points (period I or $\mathcal{T}_1$) are set up for primary analysis to evaluate the short-term causal effects. Within this phase, half of the patients in the randomized study ($S = 1$) receive treatment A_p while the rest half receive control. All the patients in the external study ($S = 0$) receive control with $A_p = 0$. After the first two time points, the study enters the OLE phase (period II or $\mathcal{T}_2$), and all the patients from the control group switched to the treatment arm, leading to $A_o = 1$, while the external patients remain untreated with $A_o = 0$.

Each simulated patient has a four-dimensional outcome vector $\mathbf{Y} = (Y_1, Y_2, Y_3, Y_4)^\top$. The outcomes are generated from a set of structural causal equations:

$$Y_t = Y_t(\mathbf{A}) = \begin{cases} A_p\tau_t + \mathbf{X}^\top\boldsymbol{\gamma} + \epsilon_t, & t \in \mathcal{T}_1; \\ A_o\tau_t + \mathbf{X}^\top\boldsymbol{\gamma} + \epsilon_t, & t \in \mathcal{T}_2. \end{cases}$$

In the numerical simulation, the noise ϵ_t is generated from a normal distribution $N(0, \sigma^2)$ with σ calibrated from real data. Moreover, we choose $\boldsymbol{\gamma}$ based on the real SMA dataset by fitting the corresponding models. In practice the user can choose the coefficients $\boldsymbol{\gamma}$ based on their own real dataset, by fitting similar regressions and obtaining the summary statistics such as regression coefficients and estimated variance for the noise to set up a more customized simulation.

The above setup can be translated into a complete dataset from the `Simulation` module. For example, for the simple case of zero effects across all phases, i.e., $\tau_t = 0$ for $t = 1, 2, 3, 4$, we can create a list of outcome model parameter specifications and use the function `simulate_trial()` to build a data frame. We used the following setup in our simulation:

```
normal <- copula::normalCopula(param = c(0.8), dim = 4, dispstr = "ar1")

#========== generate internal covariates ============
X_int <- simulate_X_copula(n = 200,
                           p = 4,
                           cp = normal,  # copula
                           margins = c("binom", "binom", "binom", "exp"),
                           paramMargins = list(list(size = 1, prob = 0.7),
                                               list(size = 1, prob = 0.9),
                                               list(size = 1, prob = 0.3),
                                               list(rate = 1/10))
)

X_int$x4 = round(X_int$x4) + 1
X_int$x5 = 30 + 10 * X_int$x1 + (7) * X_int$x2 + (-6) * X_int$x3 +
         (-0.5) * X_int$x4 + rnorm(200, mean = 0, sd = 10)

varnames = c("1", paste0("x", 1:5))

#============ generate external covariates ==============
X_ext <- simulate_X_copula(n = 100,
                           p = 4,
                           cp = normal,  # copula
                           margins = c("binom", "binom", "binom", "exp"),
                           paramMargins = list(list(size = 1, prob = 0.7),
                                               list(size = 1, prob = 0.9),
                                               list(size = 1, prob = 0.3),
                                               list(rate = 1/10))
)

X_ext$x4 = round(X_ext$x4) + 1
X_ext$x5 = 50 + 10 * X_ext$x1 + (2) * X_ext$x2 + (-1) * X_ext$x3 +
         (-0.3) * X_ext$x4 + rnorm(100, mean = 0, sd = 10)

varnames = c("1", paste0("x", 1:5))

#============ Specify outcome models ==============
model_form_x_t1 = setNames(c(10.0, 0.05, -1.5, -1.0, -0.2, -0.1), varnames)
model_form_x_t2 = setNames(c(6.0, 0.5, -0.5, -1.0, -0.3, -0.06), varnames)
model_form_x_t3 = setNames(c(5.0, 1.9, 1.4, -1.3, -0.4, -0.15), varnames)
model_form_x_t4 = setNames(c(1.2, 1.0, 2.0, -0.5, -0.4, -0.10), varnames)

outcome_model_specs = list(
  list(effect = 0, model_form_x = model_form_x_t1,
       noise_mean = 0, noise_sd = 4),
  list(effect = 0, model_form_x = model_form_x_t2,
       noise_mean = 0, noise_sd = 4),
  list(effect = 0, model_form_x = model_form_x_t3,
       noise_mean = 0, noise_sd = 4),
  list(effect = true_effect_long, model_form_x = model_form_x_t4,
       noise_mean = 0, noise_sd = 4)
)

#=========== generate trial data ============
Data = simulate_trial(X_int,
                      X_ext,
                      num_treated = 150,
                      OLE_flag = T,
                      T_cross = 2,
                      outcome_model_specs)
```

4.2.2. Primary Analysis and Hypothesis Testing

For primary analysis, we use one Monte Carlo experiment to evaluate the performance of the estimators for testing the following hypothesis:

$$H_0 : \tau_2 = 0; \quad H_1 : \tau_2 > 0.$$

We compare eight sets of methods for statistical inference, specified by combinations of the following three factors: (i) whether IPW or AIPW is used; (ii) whether zero or optimal weighting is used for external control borrowing; (iii) whether parametric method or bootstrap is used for inference. The goal is to report several evaluation metrics for each of these estimation and inference strategies, including bias, variance, MSE, coverage rate, and Type I error. The power of the estimators can also be calculated when a specific effect size is chosen under H_1, say $\tau_2 = 2$.

The discussion for this section corresponds to the vignette file `primary_simulation_workflow.Rmd` in the **rdborrow** package. To perform the simulation task with the `Simulation` module, we can first generate two lists of trial data, `data_matrix_list_null` and `data_matrix_list_alt`, by looping the R code from the previous section under the null and alternative hypothesis, respectively. Then we create a list of method objects `method_obj_list`. For example, one method we need to set up is IPW with optimal weighting and parametric inference, which can be achieved with the following code:

```
method_IPW_optimal_weight = setup_method_weighting(
  method_name = "IPW",
  optimal_weight_flag = T,
  model_form_piS = "S ~ x1 + x2 + x3 + x4 + x5")
```

Then for parametric inference evaluation, we can build a `simulation_primary_obj`, then run the simulation and generate a simulation report:

```
# Create a simulation object for primary analysis
simulation_primary_obj = setup_simulation_primary(
  data_matrix_list_null = data_matrix_list_null,
  data_matrix_list_alt = data_matrix_list_alt,
  trial_status_col_name = trial_status_col_name,
  treatment_col_name = treatment_col_name,
  outcome_col_name = outcome_col_name,
  covariates_col_name = covariates_col_name,
  method_obj_list = method_obj_list,
  true_effect = 0,
  alt_effect = 2,
  alpha = alpha,
  method_description = c("IPW, optimal weight",
                         "AIPW, optimal weight",
                         "IPW, zero weight",
                         "AIPW, zero weight"))
# Run simulation
simulation_report = run_simulation(simulation_primary_obj)
```

```
# Output the simulation report
simulation_report
```

We can directly output the results in the Console, which returns a data frame that summarizes several metrics from the simulation:

```
> simulation_report
  method_description     bias   variance    mse   coverage type_I_error power
 IPW, optimal weight   0.0049    0.3117  0.3117  0.948       0.052     0.940
AIPW, optimal weight   0.0041    0.3205  0.3205  0.940       0.060     0.936
    IPW, zero weight  -0.0071    0.3338  0.3338  0.946       0.054     0.930
   AIPW, zero weight  -0.0084    0.3414  0.3415  0.948       0.052     0.934
```

Similarly, we have evaluation results based on bootstrap inference:

```
> simulation_report_bootstrap
        method_description        bias  variance   mse  coverage type_I_error  power
 IPW, optimal weight, bootstrap   0.0049  0.3117   0.3117  0.936     0.064      0.936
AIPW, optimal weight, bootstrap   0.0047  0.3244   0.3244  0.940     0.060      0.936
    IPW, zero weight, bootstrap  -0.0071  0.3338   0.3338  0.946     0.054      0.934
   AIPW, zero weight, bootstrap  -0.0079  0.3443   0.3443  0.940     0.060      0.932
```

4.2.3. Open-label extension phase

For the OLE phase analysis, we can compare the performance of different estimation & inference strategies based on the DID methods. The discussion corresponds to the vignette file `OLE_simulation_workflow.Rmd` in the **rdborrow** package.

```
> simulation_report
  method_description     bias variance      mse coverage type_I_error
           IPW, DID 0.3159233 2.616525 2.716332     0.93         0.07
          AIPW, DID 0.3721942 2.756155 2.894683     0.95         0.05
            OR, DID 0.1626763 1.274840 1.301303     0.97         0.03
```

We can see that all three methods have achieved the desired coverage.

5. Discussion

In this paper, we have presented the package **rdborrow** for causal inference in randomized controlled trials with longitudinal outcomes by incorporating external controls. The whole implementation is based on an object-oriented programming style by utilizing `S4` classes, which is user-friendly and flexible for maintenance and enhancement. The package provides an `Analysis` module that implements the external control borrowing methods in randomized controlled trials with longitudinal outcomes to facilitate estimation and inference. Moreover, it incorporates a `Simulation` module that can be used to simulate randomized trials and external control data and perform design evaluation based on a Monte Carlo study.

There are many future directions to explore. First, the **rdborrow** package mainly implemented the parametric methods for inference, which utilized the (generalized) linear models to estimate the nuisance components such as propensity scores and outcome models. We hope to incorporate other estimators such as nonparametric estimation and machine learning based estimators (Chernozhukov et al., 2018; Athey and Imbens, 2016; Van der Laan et al., 2007). Second, it would be useful to incorporate features for handling missingness in the dataset. This relies on not only a software enhancement but more on a future methodological development that accounts for missingness under a systematic framework. In addition, a more user-friendly interface, such as Shiny or other web-based applications can be developed for an online interactive analytical platform.

Acknowledgments

This work is partially supported by a grant from the FDA (HHS U01 FD007206, Pang and Zhu). We also thank Matt Secrest for sharing information about the **psborrow2** package and helping with the development of **rdborrow**.

References

A. C. Davison and D. V. Hinkley (1997) *Bootstrap Methods and Their Applications.* Cambridge: Cambridge University Press. URL: `doi:10.1017/CBO9780511802843`. ISBN 0-521-57391-2.

Abadie, A., Diamond, A. and Hainmueller, J. (2010) Synthetic control methods for comparative case studies: Estimating the effect of california's tobacco control program. *Journal of the American statistical Association*, **105**, 493–505.

— (2015) Comparative politics and the synthetic control method. *American Journal of Political Science*, **59**, 495–510.

Angelo Canty and B. D. Ripley (2024) *boot: Bootstrap R (S-Plus) Functions.* R package version 1.3-30.

Athey, S. and Imbens, G. (2016) Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, **113**, 7353–7360.

Berry, S. M., Carlin, B. P., Lee, J. J. and Muller, P. (2010) *Bayesian adaptive methods for clinical trials.* CRC press.

Chen, J., Ho, M., Lee, K., Song, Y., Fang, Y., Goldstein, B. A., He, W., Irony, T., Jiang, Q., van der Laan, M. et al. (2023) The current landscape in biostatistics of real-world data and evidence: clinical study design and analysis. *Statistics in Biopharmaceutical Research*, **15**, 29–42.

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W. and Robins, J. (2018) Double/debiased machine learning for treatment and structural parameters.

Cnaan, A., Laird, N. M. and Slasor, P. (1997) Using the general linear mixed model to analyse unbalanced repeated measures and longitudinal data. *Statistics in medicine*, **16**, 2349–2380.

Eggleston, B. S., Ibrahim, J. G., McNeil, B. and Catellier, D. (2021) Bayesctdesign: An r package for bayesian trial design using historical control data. *Journal of Statistical Software*, **100**, 1–51.

Fu, C., Pang, H., Zhou, S. and Zhu, J. (2023) Covariate handling approaches in combination with dynamic borrowing for hybrid control studies. *Pharmaceutical Statistics.*

Gravestock, I. (2024) ***psborrow2**: An R Package for Bayesian Dynamic Borrowing Simulation Study and Analysis.* URL: `https://github.com/Genentech/psborrow2`. R package version 0.0.2.9001, https://genentech.github.io/psborrow2/.

Gravestock, I., Gower-Page, C., Secrest, M., Lu, Y., Lin, A. and AG, F. H.-L. R. (2022) *Bayesian Dynamic Borrowing with Propensity Score.* URL: `https://cran.r-project.org/web/packages/psborrow`. R package version 0.2.1.

Haddad, T., Himes, A., Thompson, L., Irony, T., Nair, R., Modeling, M. C. and Participants, S. W. G. (2017) Incorporation of stochastic engineering models as prior information in bayesian medical device trials. *Journal of biopharmaceutical statistics*, **27**, 1089–1103.

Ho, M., van der Laan, M., Lee, H., Chen, J., Lee, K., Fang, Y., He, W., Irony, T., Jiang, Q., Lin, X. et al. (2023) The current landscape in biostatistics of real-world data and evidence: causal inference frameworks for study design and analysis. *Statistics in Biopharmaceutical Research*, **15**, 43–56.

Hobbs, B. P., Carlin, B. P., Mandrekar, S. J. and Sargent, D. J. (2011) Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. *Biometrics*, **67**, 1047–1056.

Ibrahim, J. G. and Chen, M.-H. (2000) Power prior distributions for regression models. *Statistical Science*, 46–60.

Imbens, G. W. and Rubin, D. B. (2015) *Causal inference in statistics, social, and biomedical sciences.* Cambridge University Press.

Van der Laan, M. J., Polley, E. C. and Hubbard, A. E. (2007) Super learner. *Statistical applications in genetics and molecular biology*, **6**.

Lee, D., Yang, S., Berry, M., Stinchcombe, T., Cohen, H. J. and Wang, X. (2024) genrct: a statistical analysis framework for generalizing rct findings to real-world population. *Journal of Biopharmaceutical Statistics*, 1–20.

Lendle, S. D., Schwab, J., Petersen, M. L. and van der Laan, M. J. (2017) ltmle: an r package implementing targeted minimum loss-based estimation for longitudinal data. *Journal of Statistical Software*, **81**, 1–21.

Lewis, C. J., Sarkar, S., Zhu, J. and Carlin, B. P. (2019) Borrowing from historical control data in cancer drug development: a cautionary tale and practical guidelines. *Statistics in biopharmaceutical research*, **11**, 67–78.

Li, H. and Yue, L. Q. (2023) Propensity score-based methods for causal inference and external data leveraging in regulatory settings: From basic ideas to implementation. *Pharmaceutical Statistics.*

Li, X., Miao, W., Lu, F. and Zhou, X.-H. (2023) Improving efficiency of inference in clinical trials with external control data. *Biometrics*, **79**, 394–403.

Lu, N., Chen, W., Li, H., Song, C., Tiwari, R., Chenguang, W., Xu, Y. and Yue, L. (2023) Propensity score-incorporated adaptive design approaches when incorporating real-world data. *Pharmaceutical Statistics.*

Mallinckrod, C. H., Lane, P. W., Schnell, D., Peng, Y. and Mancuso, J. P. (2008) Recommendations for the primary analysis of continuous endpoints in longitudinal clinical trials. *Drug Information Journal*, **42**, 303–319.

McIver, T., El-Khairi, M., Yeung, W. Y. and Pang, H. (2023) The use of real-world data to support the assessment of the benefit and risk of a medicine to treat spinal muscular atrophy. In *Real-World Evidence in Medical Product Development*, 387–411. Springer.

Pocock, S. J. (1976) The combination of randomized and historical controls in clinical trials. *Journal of chronic diseases*, **29**, 175–188.

Rubin, D. B. (1980) Comment on “randomization analysis of experimental data: the fisher randomization test” by d. basu. *Journal of the American Statistical Association*, **75**, 591–593.

Sabanes Bove, D., Li, L., Dedic, J., Kelkhoff, D., Kunzmann, K., Lang, B. M., Stock, C., Wang, Y., James, D., Sidi, J., Leibovitz, D. and Sjoberg, D. D. (2024) *mmrm: Mixed Models for Repeated Measures.* URL: `https://openpharma.github.io/mmrm/`. R package version 0.3.11.9000.

Song, C., Li, H., Chen, W.-C., Lu, N., Tiwari, R., Wang, C., Xu, Y. and Yue, L. Q. (2023) Principled leveraging of external data in the evaluation of diagnostic devices via the propensity score-integrated composite likelihood approach. *Pharmaceutical Statistics*, **22**, 547–569.

U.S. Food and Drug Administration (2023) Considerations for the design and conduct of externally controlled trials for drug and biological products.

Viele, K., Berry, S., Neuenschwander, B., Amzal, B., Chen, F., Enas, N., Hobbs, B., Ibrahim, J. G., Kinnersley, N., Lindborg, S. et al. (2014) Use of historical control data for assessing treatment effects in clinical trials. *Pharmaceutical statistics*, **13**, 41–54.

Wang, C., Li, H., Chen, W.-C., Lu, N., Tiwari, R., Xu, Y. and Yue, L. Q. (2019) Propensity score-integrated power prior approach for incorporating real-world evidence in single-arm clinical studies. *Journal of biopharmaceutical statistics*, **29**, 731–748.

Yap, T. A., Jacobs, I., Baumfeld Andre, E., Lee, L. J., Beaupre, D. and Azoulay, L. (2022) Application of real-world data to external control groups in oncology clinical trial drug development. *Frontiers in Oncology*, **11**, 695936.

Zhou, X., Pang, H., Drake, C., Burger, H. U. and Zhu, J. (2024a) Estimating treatment effect in randomized trial after control to treatment crossover using external controls. *Journal of Biopharmaceutical Statistics*, 1–29.

Zhou, X., Zhu, J., Drake, C. and Pang, H. (2024b) Causal estimators for incorporating external controls in randomized trials with longitudinal outcomes. In submission.

A. Estimation and inference based on Analysis Module

A.1. Primary analysis

For the primary analysis, we first initiate a method object from class ‘`method_weighting_obj`’, using the function `setup_method_weighting()`. The function `setup_method_weighting()` has the following arguments:

```
method_weighting_obj = setup_method_weighting(
  method_name,
  optimal_weight_flag,
  wt,
  model_form_piS)
```

The next step is to input the dataset to be analyzed and build an analysis object from the class ‘`analysis_primary_obj`’ with the function `setup_analysis_primary()`:

```
analysis_primary_obj = setup_analysis_primary(
  data,
  trial_status_col_name,
  treatment_col_name,
  outcome_col_name,
  covariates_col_name,
  method_weighting_obj)
```

Then we can simply feed the analysis object into the function `run_analysis()` to obtain the results:

```
run_analysis(analysis_primary_obj)
```

When Bootstrap is used for inference, we also need to build a ‘`bootstrap_obj`’ object. For example, the following code chunk will set the number of Bootstrap replications to 1000, and the confidence interval is built from the Bias Corrected and Accelerated (BCa) Bootstrap method.

```
bootstrap_obj = setup_bootstrap(
  replicates,
  bootstrap_CI_type
)
```

The ‘`bootstrap_obj`’ inherently makes use of the **boot** package and the options for `bootstrap_CI_type` are compatible with the `boot()` function. The bootstrap object is further taken as additional input for the method object:

```
method_weighting_obj = setup_method_weighting(
                          method_name = "IPW",
                          optimal_weight_flag,
                          bootstrap_flag,
                          bootstrap_obj,
                          wt,
                          model_form_piS)
```

A.2. Open-label extension phase analysis

For the OLE phase analysis, we also start with initiating a method object from the DID class ‘`method_DID_obj`’ or SCM class ‘`method_SCM_obj`’.

Difference-in-differences. For DID, we start with initiating a ‘`method_DID_obj`’ and then feed it into the ‘`analysis_OLE_obj`’. Then the function `run_analysis()` will start the analysis. As one example, we can set

```
method_DID_obj = setup_method_DID(
  method_name,
  bootstrap_flag,
  model_form_piS,
  model_form_piA)
```

In this example, we applied IPW with the DID method. `bootstrap_flag = T` indicates that we are applying Bootstrap for inference, which is the default for the OLE phase analysis. The argument `model_form_piS` and `model_form_piA` specifies the propensity score model form for the trial status S and treatment A, respectively. The argument `model_form_piA` can be left empty if no propensity model is provided. Then the next step is to initiate an ‘`analysis_OLE_obj`’ from the model.

```
analysis_OLE_obj = setup_analysis_OLE(
  data,
  trial_status_col_name,
  treatment_col_name,
  outcome_col_name,
  covariates_col_name,
  T_cross,
  method_OLE_obj)
```

Compared to the ‘`analysis_primary_obj`’, an additional argument `T_cross` is included here, which specifies the timepoint where the crossover from control to treatment occurs. With the ‘`analysis_OLE_obj`’ object, we can apply `run_analysis()` to obtain the analytical results:

```
run_analysis(analysis_OLE_obj)
```

Bootstrap is used to generate confidence intervals for the DID method. It is possible to customize the bootstrap setup as needed.

Synthetic control methods. For SCM, we start by building a bootstrap component:

```
bootstrap_obj = setup_bootstrap(
  replicates,
  bootstrap_CI_type
)
```

The next step is to build an SCM method component from the class '`method_SCM_obj`'.

```
method_SCM_obj = setup_method_SCM(method_name,
                                  bootstrap_flag,
                                  bootstrap_obj,
                                  lambda.min,
                                  lambda.max,
                                  nlambda,
                                  parallel,
                                  ncpus)
```

Then we can build an analysis object for the OLE phase, '`analysis_OLE_obj`', taking the `method_SCM_obj` as the method specification:

```
analysis_OLE_obj = setup_analysis_OLE(
  data,
  trial_status_col_name,
  treatment_col_name,
  outcome_col_name,
  covariates_col_name,
  T_cross,
  method_OLE_obj)

run_analysis(analysis_OLE_obj)
```

B. Evaluation based on Simulation Module

Monte Carlo simulation is an important task in evaluating and comparing the performance of different estimators under various settings. The **rdborrow** package also provides a systematic workflow for conducting numerical simulations. This feature is implemented by the Simulation Module.

For primary analysis, the key component of the Simulation Module is a '`simulation_primary_obj`', which initiates a special numerical simulation object by the following setup:

```
simulation_primary_obj = setup_simulation_primary(
  data_matrix_list_null, data_matrix_list_alt,
  trial_status_col_name, treatment_col_name,
  outcome_col_name, covariates_col_name,
  method_obj_list, true_effect, alt_effect,
  alpha, method_description)
```

We can set up a hypothesis testing workflow with the `setup_simulation_primary()` function. Suppose that we want to evaluate the performance of a set of estimators for testing the following hypothesis:

$$H_0 : \tau = \tau_0; \quad H_1 : \tau = \tau_1. \tag{7}$$

The argument `data_matrix_list_null` is a list of R data frames for experimentation. It has length L which corresponds to the rounds of Monte Carlo simulations to be conducted. Together with `data_matrix_list_full`, one needs to specify the argument `true_effect` which corresponds to τ_0 and will be used to evaluate the bias and MSE of different estimators. The other argument, `data_matrix_list_alt`, is another list of data frames, yet should be generated under the effect size τ_1, specified by the argument `alt_effect`. This pair of arguments is used to evaluate the power of the estimators under the prespecified alternative hypothesis H_1. If power calculation is not required, `data_matrix_list_alt` and `alt_effect` can be omitted.

The `data_matrix_list_null` and `data_matrix_list_alt` can be generated from various sources and allow users to test simulation settings based on concrete research questions. In **rdborrow**, we implemented several simulators that provide multiple approaches to generate longitudinal data (Figure 8). The trial simulators are composed of several parts. The first part

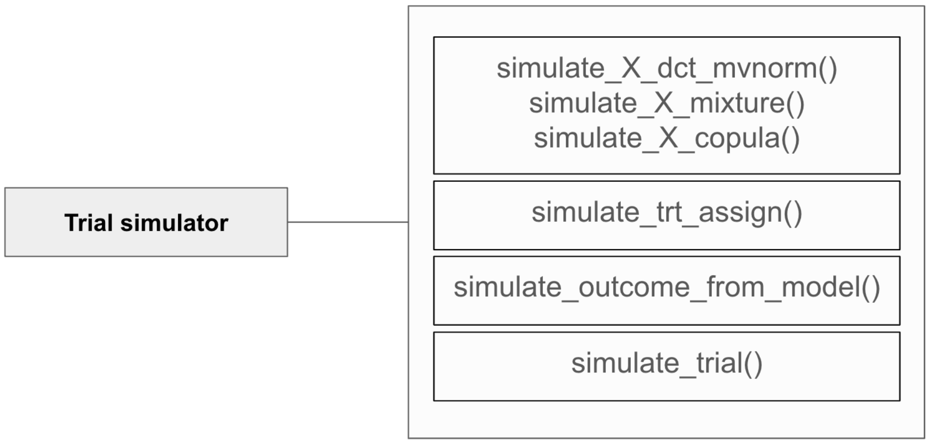

Fig. 8. Trial Simulator

is set up for covariate simulation. Three simulation methods are provided: discretized normal distribution (`simulate_X_dct_mvnorm()`), gaussian mixture models (`simulate_X_mixture()`), and copulas (`simulate_X_copula()`). These covariate simulators extend the implementation of Gravestock et al. (2022) and can accommodate the data simulation tasks under various scenarios. Table 1 summarizes the pros and cons of each simulation method.

Table 1: Pros and Cons of Covariate Simulators

Methods	Pros	Cons
Discretized Gaussian	- easy to implement; - easy to specify columns and pdf of categorical variables; - can specify correlation structure	- continuous variables are limited to normal; - hard to directly quantify the correlation between categorical variables
Gaussian Mixture	- highly interpretable: levels of categorical variables form clusters, each cluster with one distribution (pattern) - flexible for specifying correlation structure and pattern of categorical variables	- might have many parameters to specify with many covariates

Copula	- can accommodate many different marginal distributions, such as t, normal, gamma, etc. - instead of specifying Pearson correlation, can specify Spearman correlation	- Spearman correlation is not an accurate characterization for correlation among categorical variables

The second part is a function (`simulate_trt_assign()`) for simulating treatment assignment across patients. The third part, featuring the function `simulate_outcome_from_model()`, generates longitudinal outcomes from a prespecified model. The fourth part, `simulate_trial()`, is used to aggregate all the previous parts into one complete data frame for downstream causal analysis.

The argument `method_obj_list` is a list of instances from `method_obj`, which specifies a set of methods to be compared. One should include a `method_description` argument to concretely depict the methods to be compared. In the above case, we include four different methods. The true effect needs to be specified in the argument `true_effect`.

After setting up the `simulation_primary_obj`, the next step is to feed it into the function `run_simulation()`:

```
simulation_report = run_simulation(simulation_primary_obj, quiet = TRUE)
```

This will return an object from another special class, '`simulation_report_obj`', which collects several metrics for evaluation, including bias, variance, MSE, and coverage of each method in `method_list` (Figure 7). A `show()` method is overloaded to generate a user-friendly output for the '`simulation_report_obj`'.

For the OLE analysis, Monte Carlo simulation can be set up and analyzed following an analogous style.