Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

RCTrep: An R package for the validation of estimates of the average treatment effect

Lingjie Shen
Tilburg University

Gijs Geleijnse
IKNL

Maurits Kaptein
JADS

Abstract

Since we do not observe the true treatment effect for each individual, the validation of estimates of the average treatment effect obtained from an observational data set is challenging. **RCTrep** is an R package that allows users to easily validate the obtained estimates of a population by integrating the observational data set and a randomized controlled trial data set. **RCTrep** offers a generic protocol of the validation via four steps, namely, *identification*, *estimation*, *diagnosis*, and *validation*. **RCTrep** allows the validation using aggregated data and provides a dashboard to review all needed results. This article presents the designs and implementation details of **RCTrep** and serves as a user guide of **RCTrep** for researchers and policymakers to leverage the potential of observational data to inform precise treatment decisions of a population.

Keywords: Observational data, randomized controlled trial data, the average treatment effect, validation.

1. Introduction

There is a growing interest in estimating the average treatment effect using observational data (Bica, Alaa, Lambert, and Van Der Schaar 2021; Colnet, Mayer, Chen, Dieng, Li, Varoquaux, Vert, Josse, and Yang 2020; Stuart 2010). Numerous estimators for the average treatment effects have been proposed, capitalizing on ideas such as G-computation estimation (Hill 2011; Hitsch and Misra 2018; Atan, Jordon, and Van der Schaar 2018; Wager and Athey 2018), propensity score-based estimation (Xie, Brand, and Jann 2012; Rosenbaum and Rubin 1983), doubly robust estimation, and representation learning methods (Yao, Li, Li, Huai, Gao, and Zhang 2018; Johansson, Shalit, Kallus, and Sontag 2020). For a more detailed overview of related literature, see a survey by (Jiang, Qi, Zhou, Zhou, and Rao 2021). Despite this large contemporary literature, there is no "single best" estimator that can consistently provide the most accurate estimates of the average treatment effect on a variety of observational datasets (Dorie, Hill, Shalit, Scott, and Cervone 2019). Hence, given a large observational dataset, in the absence of the truth, *how can we validate estimates of the average treatment effect, and accordingly select the most accurate one?*

In this paper, we present a method and the accompanying software to validate the most appropriate estimate of the average treatment effect obtained from an observational dataset. We provide an R package **RCTrep** to validate the estimates by integrating an RCT dataset and the observational dataset. We formulate the core elements of the validation approach in **RCTrep** as follows: Let $\mathcal{P}_{\boldsymbol{\theta}}$ denote a target population, from which a simple random sample of an RCT $\mathcal{S}^{rct}$ is drawn and a sample of an observational study $\mathcal{S}^{obs}$ is drawn via a selection probability; $\boldsymbol{X} = (X_1, ..., X_d) \in \mathbb{R}^d$ denote an associated d-dimensional *pre-treatment outcome predictors* which includes all *confounders*, $\boldsymbol{X}_s \subseteq \boldsymbol{X}$ denote a p-dimensional vector of *all effect modifiers which are predictive of the selection probability* of $\mathcal{S}^{obs}$, and $\hat{\boldsymbol{\tau}} = \{\hat{\tau}_0, \hat{\tau}_1, ..., \hat{\tau}_n\}$ denote a set of estimators of the conditional average treatment effect $\tau(\boldsymbol{X})$. **RCTrep** makes it easy to validate various estimates $\hat{\tau} \in \mathrm{T}$ using $\mathcal{S}^{obs}$ and select the most accurate one according to the following metric:

$$\mathbb{L}(\hat{\tau}_0^{\mathcal{S}^{rct}}; \hat{\tau}^{\mathcal{S}^{obs}}) = \mathbb{L}\left(\hat{\tau}_0^{\mathcal{S}^{rct}}, \sum_{i \in \mathcal{S}^{obs}} \hat{w}(\boldsymbol{x}_{si})\hat{\tau}(\boldsymbol{x}_i)\right), \quad s.t. \;\; \hat{p}(\boldsymbol{x}_s) = \hat{w}(\boldsymbol{x}_s)\hat{q}(\boldsymbol{x}_s), \sum_{i \in \mathcal{S}^{obs}} \hat{w}(\boldsymbol{x}_{si}) = 1 \tag{1}$$

where $\hat{\tau}_0^{\mathcal{S}^{rct}}$ is an unbiased estimate of the average treatment effect of $\mathcal{P}_\theta$ obtained from $\mathcal{S}^{rct}$ using the estimator $\hat{\tau}_0$, $\hat{p}(\boldsymbol{x}_s)$ and $\hat{q}(\boldsymbol{x}_s)$ are the empirical densities of $\boldsymbol{x}_s$ in $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$ respectively, $\hat{w}(\boldsymbol{x}_{si})$ is a normalized weight for $i \in \mathcal{S}^{obs}$, and $\hat{w} \in \hat{\boldsymbol{w}}$ is an estimator for the weight. **RCTrep** outlines a generic protocol to implement the validation according to four steps:

- **Step 1: Identification** Users identify two sets of covariates $\boldsymbol{X}$ and $\boldsymbol{X}_s$. The covariates are used to model $\hat{\tau}(\boldsymbol{X})$ and $\hat{w}(\boldsymbol{X}_s)$;
- **Step 2: Estimation** Users specify two estimators, $\hat{\tau} \in \hat{\boldsymbol{\tau}}$ and $\hat{w} \in \hat{\boldsymbol{w}}$, and initiate two objects of class `TEstimator` and `SEstimator` accordingly. Two objects estimate $\hat{\tau}(\boldsymbol{X})$ using $\mathcal{S}^{obs}$ and $\hat{w}(\boldsymbol{X}_s)$ using $\mathcal{S}^{obs}$ and $\mathcal{S}^{rct}$ respectively;
- **Step 3: Diagnosis** Users diagnose assumptions for the objects of class `TEstimator` and `SEstimator`. `TEstimator` diagnoses no omitted bias for G-computation estimator or diagnoses distance of weighted distribution of $\boldsymbol{X}$ between treatment and control groups for propensity scored-based estimator; `SEstimator` diagnoses distance of weighted distribution of $\boldsymbol{X}_s$ between $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$;
- **Step 4: Validation** Users initiate an object of class `Fusion`. The object integrates estimates of the average treatment effects of (sub-)populations obtained from $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$ and computes metrics $\mathbb{L}$ on (sub-)population levels.

For more elaboration of the four steps, see section 5. Although our approach relies on the availability of RCT data and assumptions of the sampling mechanism, we think RCT is the only approach to ground the truth of the average treatment effect. To the best of our knowledge, **RCTrep** is the only package that uses real data to validate estimates of the average treatment effect. For more elaboration of our proposed approach, see section 2 and 3.

The remaining part of the paper proceeds as follows: section 2 formulates the problem setup for validation of estimates of the average treatment effects; section 3 illustrates the proposed approach to the validation using observational data and RCT data, based on which **RCTrep** was developed; section 4 provides an overview of the R package **RCTrep** and introduces the core classes and functions in **RCTrep**; section 5 outlines a generic protocol of **RCTrep** implementation via four steps, for which an example is demonstrated; section 6 demonstrated three working examples, i.e., validation at scale, validation using aggregated data, and validation using synthetic RCT data; finally, the paper ends up with the discussion in section 7.

1.1. Related work

Currently, although there are a number of studies of the average treatment effects estimation and the accompanying software, e.g., Python library **CausalML**, **EconML**, **DoWhy**, studies on the validation of the average treatment effect obtained from an observational dataset are not adequately investigated and related software is insufficient. Studies by Wendling, Jung, Callahan, Schuler, Shah, and Gallego (2018), Alaa and Van Der Schaar (2019), Schuler, Jung, Tibshirani, Hastie, and Shah (2017), Powers, Qian, Jung, Schuler, Shah, Hastie, and Tibshirani (2018), Franklin, Schneeweiss, Polinski, and Rassen (2014), and Cheng, Guo, Moraffah, Sheth, Candan, and Liu (2022), and existing software, e.g., the R package **MethodEvaluation** (Schuemie, Cepeda, Suchard, Yang, Tian, Schuler, Ryan, Madigan, and Hripcsak 2020), the Python package **Causality-Benchmark**(Shimoni, Yanover, Karavani, and Goldschmnidt 2018), and Python package **JustCause** (Franz 2020), approximate a data generation process for a given observational dataset, and use the simulated truth of the average treatment effect as the truth for the validation. These approaches implicitly assume no unmeasured confounders, and hence can be biased in case an unmeasured confounder exists. An overview of existing software for the validation is provided in table 1. The table compares the software in terms of the number of estimators provided, sample space based on which the average treatment effect is to validate, validation metrics, and the truth. The table shows that **RCTrep** is the only package for the validation using unbiased estimates of the average treatment effect. In addition, **RCTrep** provides regulatory agreement and estimate agreement as validation metrics (Franklin, Pawar, Martin, Glynn, Levenson, Temple, and Schneeweiss 2020).

On the other hand, there is a growing body of studies focusing on generalization or transportation of the average treatment effect of a population to another population. Related software includes R packages **ExtendingInferences** (Dahabreh, Robertson, Steingrimsson, Stuart, and Hernan 2020), **generalize** (Ackerman, Lesko, Siddique, Susukida, and Stuart 2021), **genRCT** (Dong, Yang, Wang, Zeng, and Cai 2020), **generalizing** (Cinelli and Pearl 2021), **transport** (Rudolph, Schmidt, Glymour, Crowder, Galin, Ahern, and Osypuk 2018) and **causaleffect** (Tikka and Karvanen 2018). Approaches used in the software are closely related to that of **RCTrep**, however, **RCTrep** is different from them with respect to the motivation - validating estimates of the average treatment effects obtained from observational data and selecting the most accurate one accordingly.

1.2. Strength and limitation of our work

RCTrep makes contributions to the methodology and software design. First, unlike existing studies (Wendling *et al.* 2018; Alaa and Van Der Schaar 2019; Schuler *et al.* 2017; Franklin *et al.* 2014; Schuemie *et al.* 2020; Shimoni *et al.* 2018) which validate estimates of the average

Task		Package			
		MethodEvaluation	**CausalityBenchmark**	**JustCause**	**RCTrep**
Estimator	propensity score	✓		✓	✓
	G_computation	✓			✓
	Doubly robust	✓		✓	✓
Sample space	population	✓	✓	✓	✓
	sub-population		✓	✓	✓
Metrics	(R)MSE	✓	✓	✓	✓
	PEHE				
	Bias		✓		
	confidence interval		✓		✓
	coverage	✓	✓		
	AUC	✓			
	mean precision	✓			
	type 1 error	✓			
	type 2 error	✓			
	Regulatory agreement				✓
	Estimate agreement				✓
Truth	simulated value	✓	✓	✓	
	unbiased estimate				✓

Table 1: Comparisons of packages for the validation of the average treatment effect estimate with a focus on the provided options of estimators, sample space based on which the average treatment effect is to validate, validation metrics, and the truth.

treatment effects using simulated data, as shown in the Table 1, **RCTrep** is the only package that compares to unbiased estimates of the average treatment effect obtained from real data. Second, **RCTrep** validates estimates on both population and sub-population levels, providing a deeper understanding of the performance of an estimator. For instance, a high-bias estimator may have good accuracy at a population level but may have lower accuracy at sub-population levels. Third, **RCTrep** can validate estimates using aggregated data of a full dataset, which can generate the same results as those using a full dataset. **RCTrep** also provides functions to generate synthetic RCT datasets based on available marginal distributions of covariates. Fourth, **RCTrep** provides users with autonomy to asses bias and variance of weighted estimates in a structured manner. For instance, in the *identification* step, users can evaluate the impact of different adjustment sets on bias and variance; in the *estimation* step, users can evaluate the performance of different estimators for the average treatment effect and weights. The results estimated from different settings can be assessed easily and quickly. Lastly, the design structure of **RCTrep** has advantages over other packages and can be easily extended for other motivations. For instance, **RCTrep** can be used to validate estimates of the average treatment effect from multiple data sources by proposing a protocol of identification, estimation, and validation steps.

Despite the appealing advantages of **RCTrep** over other packages, it is not without limitations. First, how confident we are about the performance of estimators depends on the quality of RCT data. This might lead to a challenging situation in terms of regulatory decision-making when an estimator using observational data has consistent estimate agreement whereas inconsistent regulatory agreement. In other words, because observational data may have substantially more power than the corresponding RCTs, there may be cases when observational data find statistical significance whereas the trial fails to find the statistical significance, even when the estimates from observational data and RCT data are close Franklin *et al.* (2020). In our view, in such cases, one has to carefully weigh the evidence from both data sources, and perhaps resort to additional studies to make a definite decision. Second, we assume the discrepancy in estimates between two samples is solely due to the sampling mechanism and treatment assignment mechanism regardless of other factors proposed by Wang, Sreedhara, Bessette, and Schneeweiss (2022). Readers should bear this in mind when interpreting results.

1.3. Demonstration of usage

We can call the function `RCTREP()` in **RCTrep** to implement estimates validation as follow:

```
> library(RCTrep)
> source.data <- RCTrep::source.data
> target.data <- RCTrep::target.data
> output <- RCTREP(TEstimator = "G_computation", SEstimator = "Exact",
+                  outcome_method = "BART",
+                  source.data = RCTrep::source.data,
+                  target.data = RCTrep::target.data,
+                  vars_name = list(confounders_treatment_name =
+                                   c("x1","x2","x3","x4","x5","x6"),
+                                   treatment_name = c('z'),
+                                   outcome_name = c('y')),
+                  confounders_sampling_name = c("x2","x6"),
+                  stratification = c("x1","x3","x4","x5"),
+                  stratification_joint = TRUE)
> fusion <- Fusion$new(output$target.obj,
+                      output$source.obj,
+                      output$source.rep.obj)
> fusion$plot()
```

The descriptions of the input arguments in the function `RCTREP()` are as follows:

- `TEstimator` specifies an estimator to adjust for the treatment assignment mechanism;
- `SEstimator` specifies an estimator to adjust for the sampling mechanism;
- `target.obj` and `source.obj` specify an RCT dataset and an observational dataset;
- `outcome_method` specify a modeling approach for the estimator `TEstimator`;
- `vars_name` specifies variable names of treatment, outcome, and confounders $\boldsymbol{X}$ due to treatment assignment mechanism;
- `confounder_sampling_name` specifies effect modifiers $\boldsymbol{X}_s$ which are predictive of the selection probability.

In the above example, we use `G_computation` estimator to adjust the treatment assignment mechanism and use the exact matching estimator to adjust the sampling mechanism. We use

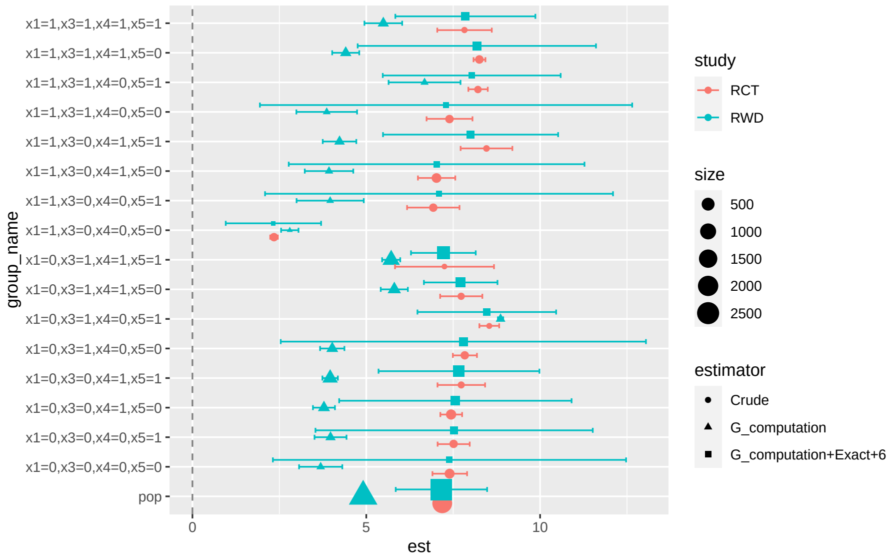


Figure 1: The validation of estimates of conditional average treatment effects validation using **RCTrep**.

Bayesian additive regression trees (BART) to model the outcome. We select `x1,x2,x3,x4,x5,x6` as $\boldsymbol{X}$ and `x2,x6` as $\boldsymbol{X}_s$. In this example, since `x2,x6` are the only effect modifiers that are predictive of the selection probability, they are the minimal set of `confounders_sampling_name` that allows for the validation. The results in the figure 1 show the estimates from observational data indicated by `G_computation+Exact+6` are close to the unbiased estimates from RCT data indicated by `Crude`, and hence the estimates from observational data are arguably valid. Without properly adjusting for the sampling mechanism, a large discrepancy in estimates between RCT and observational data can be observed, as shown by the large discrepancy in estimates between `Crude` and `G_computation`. The discrepancy might be wrongly attributed to unadjusted confounders in the observational data without considering the impact of the sampling mechanism on the validation. See section 6 for more working examples.

2. Problem setup

In this section, we formulate the problem setup for validating estimates of the average treatment effect. An overview of the notations used throughout the paper is provided in the appendix A.

2.1. Estimators for conditional average treatment effect

We consider potential outcomes framework for estimating the average treatment effects (Imbens and Rubin 2015). Let $\boldsymbol{X}$ denote d-dimensional vector of all pre-treatment outcome predictors, $t \in \mathcal{T} = \{0, 1\}$ denote binary treatment indicator where 1 and 0 denote treatment and control, respectively; let $Y \in \{0, 1\}$ denote the binary outcome of interest, $Y(t)$ denote the potential outcome had the unit received treatment t. The observed outcome of unit i under the received treatment T_i is denoted as $Y_i = T_iY_i(1) + (1 - T_i)Y_i(0)$. The individual-level treatment effect is denoted as simple difference $\tau_i = Y_i(1) - Y_i(0)$, the conditional average treatment effect is defined as $\tau(\boldsymbol{X}) = \mathbb{E}[Y(1) - Y(0) \mid \boldsymbol{X}]$, and the average treatment effect is defined as $\tau = \mathbb{E}[\tau(\boldsymbol{X})]$, where $(\boldsymbol{X}, Y(1), Y(0)) \sim \mathcal{P}_\theta$, and $\mathcal{P}_\theta$ is a target population from which a simple random sample $\mathcal{S} = \{(\boldsymbol{X}_i, T_i, Y_i); i = 1, ..., n\}$ is generated.

2.2. Validation of estimates

We now consider a set of candidate estimators of the conditional average treatment effects $\hat{\mathrm{T}} = \{\hat{\tau}_0, \hat{\tau}_1, ..., \hat{\tau}_n\}$, where $\hat{\tau}(\boldsymbol{X}) : \mathcal{X} \mapsto \tau(\boldsymbol{X})$. These may include, for example, different estimators (G-computation, IPW, doubly robust) combined with different modeling choices (e.g., BART, gaussian process, causal forest), and different hyper-parameter settings of one model, etc. The accuracy of an estimator $\hat{\tau}(\boldsymbol{X})$ for the conditional average treatment effect estimation is characterized by a distance measure $\mathbb{L}$ as a validation metric, and the selected most accurate estimate of the average treatment effect is derived based on:

$$\hat{\tau}^* = \underset{\hat{\tau} \in \mathrm{T}}{\arg\min}\, \mathbb{L}\left(\tau, \hat{\tau}\right) = \underset{\hat{\tau} \in \mathrm{T}}{\arg\min}\, \mathbb{L}\left(\tau, \sum_{\boldsymbol{x} \sim \mathcal{S}} \hat{\tau}(\boldsymbol{x}) p(\boldsymbol{x})\right) \tag{2}$$

Since τ is not observed, the metric in equation 2 can not be measured, thus hindering the validation of $\hat{\tau}$ using $\mathcal{S}$. In the following section, we provide our validation approach.

3. Proposed approach to estimates validation

In this section, we demonstrate our approach to validating estimates of the average treatment effects by integrating RCT data and observational data. In section 3.1, we start by elaborating why an estimate of the average treatment effect using RCT data can be regarded as an unbiased estimate of the truth τ of the target population $\mathcal{P}_{\boldsymbol{\theta}}$ that RCT data represents; in section 3.2 we elaborate how to use the estimates obtained from the RCT data as the truth to validate estimates of the average treatment effects obtained from observational data. In the following section, we first elaborate assumptions for treatment effect identification and estimators for the average treatment effect accordingly.

3.1. Why an estimate of the average treatment effect using an RCT dataset is an unbiased estimate of the truth?

By definition, the treatment effect for each individual is not observed and can only be estimated. The following two assumptions allow for an unbiased estimate of the treatment effect:

Assumptions 1 T-ignorability: $Y(1), Y(0) \perp\!\!\!\perp T \mid \boldsymbol{X}_t$

Assumptions 2 T-overlap: $0 < P(T = 1 \mid \boldsymbol{X}_t) < 1$

where $\boldsymbol{X}_t \subseteq \boldsymbol{X}$ is a set of confounders that isolate dependence between covariates and treatment. The assumption of $T-$ignorability implies that conditional on $\boldsymbol{X}_t$, treatment is independent of potential outcomes, hence the change in observed outcomes between treatment and control groups is only attributed to the treatment. The assumption of $T-$overlap guarantees that there is a sufficient number of individuals with characteristics $\boldsymbol{X}_t = \boldsymbol{x}_t$ in both groups. Given these two assumptions, the causal relationship between the treatment and the outcome can be identified and an unbiased estimator can be derived. Three classes of estimators can be used to estimate treatment effect under those assumptions: G-computation, inverse propensity score estimator (IPW), and doubly robust estimator. See appendix C for more detailed descriptions. Throughout the paper, we use $\boldsymbol{X}$ to denote $\boldsymbol{X}_t$ because $\boldsymbol{X}$ is a sufficient set of confounders and may improve the precision of estimates (Chatton, Le Borgne, Leyrat, Gillaizeau, Rousseau, Barbin, Laplaud, Léger, Giraudeau, and Foucher 2020).

3.2. How to use the estimates obtained from the RCT as the truth to validate the estimates obtained from observational data?

Once we have unbiased estimates of the truth obtained from an RCT dataset, how to use the estimate as the truth to validate estimates of the average treatment effect obtained from observational data? In this section, we introduce assumptions and methods that allow for the validation. We assume RCT data $\mathcal{S}^{rct}$ and observational data $\mathcal{S}^{obs}$ are from the same target population $\mathcal{P}_{\boldsymbol{\theta}}$; $\mathcal{S}^{rct}$ is a simple random sample from $\mathcal{P}_{\boldsymbol{\theta}}$ while $\mathcal{S}^{obs}$ is drawn from $\mathcal{P}_{\boldsymbol{\theta}}$ via a selection probability. Let $S \in \{0,1\}$ denote a binary selection indicator where 1 and 0 denote selection to $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$. Analogous to assumptions and methods in section 3.1, we can use similar assumptions of sampling mechanism to allow for comparison of estimates between $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$ as follows:

Assumptions 3 S-ignorability: $Y(1), Y(0) \perp\!\!\!\perp S \mid \boldsymbol{X}_s$

Assumptions 4 S-overlap: $0 < P(S = 1 \mid \boldsymbol{X}_s) < 1$

Assumption 3 demonstrates that conditioning on $\boldsymbol{X}_s \subseteq \boldsymbol{X}$, potential outcomes are exchangeable between samples. Assumption 4 guarantees that there is a sufficient number of individuals with characteristics $\boldsymbol{X}_s = \boldsymbol{x}_s$ in both samples. Given these two assumptions, within a subpopulation $\boldsymbol{X}_s = \boldsymbol{x}_s$, there is no unobserved covariate varying between samples, and hence estimates of the average treatment effect conditioning on $\boldsymbol{X}_s$ are directly comparable.

Given these two assumptions, we use weighting methods to adjust the sampling mechanism. The methods aim to balance $\boldsymbol{X}_s$ between samples. Three weighting methods are provided to achieve the balance: 1) inverse selection probability weighting (ISW); 2) exact matching on $\boldsymbol{X}_s$; 3) sub-classification based on strata of the selection probability of $\mathcal{S}^{rct}$. In general, all weighting methods require estimation of either a selection probability or density of $\boldsymbol{X}_s$. See appendix D for an elaboration of the methods in **RCTrep**. In practice, only *effect modifiers that are predictive of the selection probability* can lead to the discrepancy of treatment effect between samples (Egami and Hartman 2018; Dahabreh *et al.* 2020). Hence, throughout the paper, $\boldsymbol{X}_s$ denote effect modifiers that are predictive of the selection probability.

3.3. Putting all together

Given above four assumptions, we can replace $p(\boldsymbol{x})\hat{\tau}(\boldsymbol{x})$ in the equation 2 with $w(\boldsymbol{x}_s)\hat{\tau}(\boldsymbol{x}_s)$, and replace τ with $\hat{\tau}_0^{\mathcal{S}^{rct}}$, where $\hat{\tau}_0^{\mathcal{S}^{rct}}$ is an unbiased estimate of the average treatment effect of $\mathcal{P}_\theta$ obtained from the estimator $\hat{\tau}_0$ and $\mathcal{S}^{rct}$, $\hat{\tau}_0$ is the simple difference in sample means of outcomes between groups, and $\hat{\tau}(\boldsymbol{x}_s)$ is an estimate of the average treatment effect conditional on $\boldsymbol{X}_s = \boldsymbol{x}_s$ obtained from $\mathcal{S}^{obs}$. The proposed validation metrics are as follows:

$$\mathbb{L}\left(\hat{\tau}_0^{\mathcal{S}^{rct}}, \sum_{i \in \mathcal{S}^{obs}} \hat{w}(\boldsymbol{x}_{si})\hat{\tau}(\boldsymbol{x}_i)\right), \ s.t. \ \ \hat{p}(\boldsymbol{x}_s) = \hat{q}(\boldsymbol{x}_s)\hat{w}(\boldsymbol{x}_s), \ \sum_{i \in \mathcal{S}^{obs}} \hat{w}(\boldsymbol{x}_{si}) = 1 \tag{3}$$

where $\hat{w}(\boldsymbol{x}_{si})$ is the weight for unit $i \in \mathcal{S}^{obs}$, $\hat{w}(\boldsymbol{x}_s) = \sum_{\boldsymbol{x}_{si}=\boldsymbol{x}_s} \hat{w}(\boldsymbol{x}_{si})$ is the aggregated weight for a sub-population with $\boldsymbol{X}_s = \boldsymbol{x}_s$ in $\mathcal{S}^{obs}$. The weighted effect modifiers $\boldsymbol{x}_s$ in $\mathcal{S}^{obs}$ and $\mathcal{S}^{rct}$ are approximately equally distributed. We also validate estimates on subsets of the target population $\mathcal{P}_\theta$ to quantify the ability of $\hat{\tau}(\boldsymbol{x})$ to capture the variation of the average treatment effects over sub-populations. The validation on sub-population levels can help us further understand the performance of $\hat{\tau}(\boldsymbol{x})$. In the following, we will move from math to code, we will first have an overview of the package **RCTrep**, and then demonstrate the usage of **RCTrep** to implement the proposed validation approach.

4. Overview of software

The current section introduces the **RCTrep** implementation and core classes. The section first presents an overview of core classes that form the building blocks of **RCTrep** and offers an overview of the implementation of **RCTrep** using these core classes. Then the section provides a further introduction to the core classes and the core functions for adjusting the treatment and sampling mechanism. In the next section, we provide the basic structure of **RCTrep** and relations between each class in implementation.

4.1. Implementation

An overview of the implementation of **RCTrep** is provided in Figure 2. The figure demonstrates the role of three core classes in implementation, which form the backbone of the package. The three classes are:

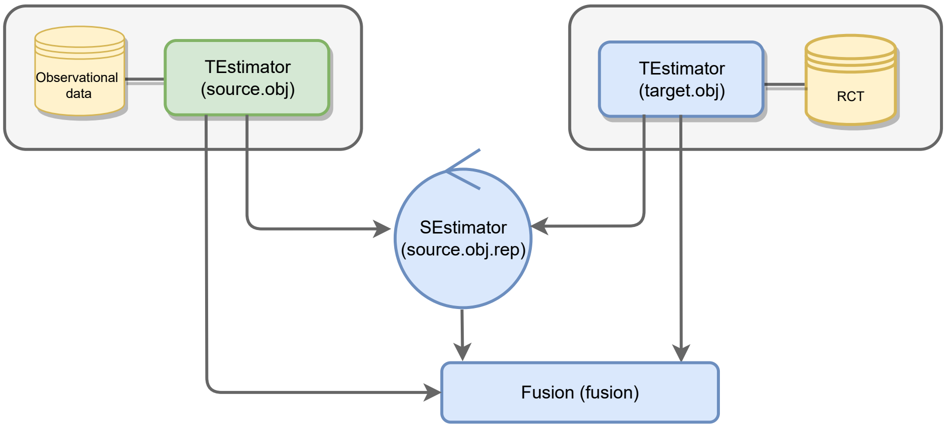


Figure 2: Diagram of **RCTrep** basic structure.

1. `TEstimator`: R6 class `TEstimator` is the parent class of all **RCTrep** `TEstimator` subclasses. It estimates the average treatment effect $\hat{\tau}$ of a population and the conditional average treatment effects $\hat{\tau}(\boldsymbol{x})$; it diagnoses the T-overlap assumption, and diagnoses T-ignorability assumption depending on an instantiated class, e.g., it diagnoses model assumptions for `G_computation` subclass and diagnoses distance of $\boldsymbol{X}$ between groups for `IPW` subclass. **RCTrep** provides `TEstimator_wrapper()` to generate an object of the class. See table 2 for more detailed descriptions of input arguments in the function.

2. `SEstimator`: R6 class `SEstimator` is the parent class of all **RCTrep** `SEstimator` subclasses. The class integrates data from `source.obj` and `target.obj`, and regards data in `target.obj` as the target population $\mathcal{P}_\theta$. It computes weights for `source.obj`, so that the weighted $\boldsymbol{X}_s$ in `source.obj` and $\boldsymbol{X}_s$ in `target.obj` are balanced. It diagnoses the S-overlap assumption, and diagnoses S-ignorability assumption by measuring the distance of weighted $\boldsymbol{X}_s$ in `source.obj` and `target.obj`. **RCTrep** provides `SEstimator_wrapper()` to generate an object of the class. See table 3 for more detailed descriptions of input arguments in the function.

3. `Fusion`: R6 class `Fusion` integrates estimates from objects of class `TEstimator` and objects of class `SEstimator`, computes validation metrics on population and sub-population levels, and ranks estimates in objects of class `TEstimator` and `SEstimator` accordingly. The number of objects of class `TEstimator` or `SEstimator` passed to its constructor is not limited.

A main loop that relates one to one to the implementation is illustrated as follows:

1 users call `TEstimator_wrapper()` to initialize a `TEstimator` subclass for $\mathcal{S}^{obs}$ as `source.obj` and a `TEstimator` subclass for $\mathcal{S}^{rct}$ as `target.obj`. The objects fit a model for treatment or outcome conditional on $\boldsymbol{x}$, and estimate the average treatment effect $\hat{\tau}$ and the conditional average treatment effect $\hat{\tau}(\boldsymbol{x})$ for `source.obj` and `target.obj`;

2 users call `SEstimator_wrapper()` to initialize a `SEstimator` subclass as `source.obj.rep` by assigning `source.obj` and `target.obj` to the function. `source.obj` and `target.obj` communicate within `source.obj.rep` in this step.

3 users call `source.obj.rep$EstimateRep()`, specifying two arguments `stratification` and `stratification_joint` to the function. The function estimates the weighted average treatment effect of the population $\mathcal{S}^{obs}$ in `source.obj` and subsets of the population stratified by the variables specified in `stratification` individually or in combination indicated by `stratification_joint=FALSE` or `TRUE`.

4 users initialize a `Fusion` class as `fusion` by assigning objects, i.e., `source.obj`, `target.obj`, and `source.obj.rep` to its initialize function. `fusion` aggregates, plots, and prints estimates. The object validates estimates of the average treatment effects of the target population and sub-populations in `source.obj` and `source.rep.obj` by calling `fusion$evaluate()`, prints validation metrics on population and sub-population levels, and ranks the estimates according to the pseudo mean squared error. The number of objects of class `TEstimator` and `SEstimator` is not limited.

5 (Optional) Then repeat step 3) and step 4) to validate the estimates on subsets of $\mathcal{P}_\theta$ defined by different `stratification` and `stratification_joint`.

We provide an overview of basic usage in the section 5 where four main steps to validate estimates of the average treatment effect using **RCTrep** are summarized. For more implementation details and infrastructure of design, see appendix F.

Arguments	Description	Default
`Estimator`	A character specifying an estimator of the average treatment effect. Allowable options are `'G_computation'`, `'IPW'`, `'DR'`.	-
`vars_name`	A list with three named characters, i.e., `confounders_treatment_name`, `treatment_name`, and `outcome_name`, which specifies names of confounders, treatment, and outcome variable	-
`data`	A data.frame with n rows and p columns, each row contains variables in `vars_name`. **RCTrep** supports binary treatment and binary/continuous outcome.	-
`name`	A character specifying a name of an returned object	`NULL`
`outcome_method`	A character specifying a method for outcome model when `Estimator` is set to be `"G_computation"` or `"DR"`. More available methods, see `train` model list in the package **caret**	`"glm"`
`treatment_method`	A character specifying a method for propensity score model when `Estimator` is set to be `"IPW"` or `"DR"`. More available methods, see `train` model list in the package **caret**	`"glm"`
`two_models`	Logical value indicating whether outcome should be modeled separately when `Estimator` is set to be `"DR"`	`FALSE`
`outcome_formula`	A formula specifying outcome regression model when `Estimator` is set to be `"G_computation"` or `"DR"`	`NULL`
`treatment_formula`	A formula specifying propensity score model when `Estimator` is set to be `"IPW"` or `"DR"`	`NULL`
`data.public`	Logical value indicating whether `data` should be an public attribute of an returned object. If `FALSE`, the function return an object of class `TEstimator_pp`	`TRUE`
`is.Trial`	Logical value indicating whether `data` is RCT data	`FALSE`
`strata_cut`	A list each of the components is a named list with two named vectors. The name of the named list is a variable name, and the two vectors are named `breaks` and `labels`. The argument calls the `cut` function to divide the range of the value of the variable into intervals based on `break` and codes the value according to `label`.	NULL
`...`	A number of additional arguments for fitting a model specified in `outcome_method` or `treatment_method`. See allowable arguments in `train` in the package **caret**, or `pbart` and `wbart` in the package **BART**	-

Table 2: Descriptions of the input argument of function `TEstimator_wrapper()`.

Arguments	Description	Default
`Estimator`	A character specifying an estimator of sampling weight. Allowable options are `'Exact'`, `'Subclass'`, and `'ISW'`.	-
`target.obj`	An object of class `TEstimator` of which `estimates` are the truth	-
`source.obj`	An object of class `TEstimator` of which `estimates` are to be validated	-
`confounders_sampling_name`	A vector of characters specifying variable names for weighting	-
`method`	A character specifying a method for computing selection probability. See `train` model list in **caret** package, and `distance` model list in **MatchIt** package.	`'glm'`
`sampling_formula`	A formula specifying a model of selection probability	NULL
`...`	A number of additional arguments for fitting a model specified in `method` when `Estimator` is set to be `ISW`. See allowable arguments in `train` in the package **caret**	-

Table 3: Descriptions of the input arguments of the function `SEstimator_wrapper()`.

4.2. Core classes

RCTrep provides two core classes, i.e., `TEstimator` and `SEstimator`, which are responsible for adjusting the treatment assignment mechanism and the sampling mechanism, respectively. **RCTrep** offers four main subclasses of `TEstimator` and three main subclasses of `SEstimator`. The four sub-classes of `TEstimator` are `Crude`, `G_computation`, `IPW`, and `DR`. The three sub-classes of `SEstimator` are `SEexact`, `SEisw`, and `SEsubclass`. The description of the key public attributes and key public methods of `TEstimator` and `SEstimator` are provided in the table 4. Note that the input arguments of the functions listed in the table are `stratification` and `stratification_joint` with default values `private$confounders_treatment_name` and `TRU E`, respectively. By specifying the two arguments, the functions get outputs of sub-populations stratified by `stratification` jointly or in combination. More elaboration of the core classes is in appendix B.

In case full data sets of `target.obj` and `source.obj` are not allowed to share to compute the weight, `RCTrep` provides a sub-class `TEstimator_pp` and a sub-class `SEstimator_pp`. The `TEstimator_wrapper()` returns an object of the class `TEstimator_pp` when the input argument `data.public=FALSE` is indicated. `SEstimator_wrapper()` returns an object of the class `SEstimator_pp` when the classes of input arguments `target.obj` and `source.obj` are `TEstimator_pp`. The public attributes `data` of `target.obj` and `source.obj` are aggregated data of full datasets. An object of the class `SEstimator_pp` can estimate the weights based on the aggregated data of `target.obj` and `source.obj`. See example 2 in section for the usage of aggregated data for the validation.

RCTrep provides a subclass `TEstimator_Synthetic` of `TEstimator`. The subclass is to initialize an object using a synthetic dataset generated from `GenerateSyntheticData()`. `GenerateSyntheticData()` generates a synthetic dataset given marginal distributions of covariates and specified pair-wise correlations between covariates. The function estimates the joint distribution of the covariates and generates the synthetic data accordingly. See example 3 in the section for the utility of synthetic data for the validation.

Attributes/Methods	Description
Class `TEstimator`	
`estimates`	A list containing two named data frame, i.e., `ATE` and `CATE`
`get_CATE()`	Print a data frame of estimates of the conditional average treatment effects
`plot_CATE()`	Plot the conditional average treatment effects
`diagnosis_t_ignorability()`	Plot diagnosis results of T-ignorability assumptions for a class
`diagnosis_t_overlap()`	Plot diagnosis results of T-overlap assumptions
`diagnosis_y_overlap()`	Plot the count of binary outcomes in treatment and control groups; plot the distribution of continuous outcome in treatment and control group
`plot_y1_y0()`	Plot the predicted outcomes under treatment and control
Class `SEstimator`	
`estimates`	A list containing two elements, a data frame named `ATE`, and a data frame named `CATE`
`EstimateRep()`	Generate weighted average treatment effect of the population and sub-populations in `source.obj` and pass the values to the public attributes `estimates$ATE` and `estimates$CATE`
`diagnosis_s_ignorability()`	Plot the diagnosis results of the S-ignorability assumption
`diagnosis_s_overlap()`	Plot the diagnosis results of the S-overlap assumption

Table 4: Descriptions of core public attributes and core public methods of the class `TEstimator` and the class `SEstimator`.

5. Basic usages

In the current section, we demonstrate four steps to validate estimates of the average treatment effect using **RCTrep**: *Identification*, *Estimation*, *Diagnosis*, and *Validation*. We demonstrate the four steps using an example, and we integrate all needed results of the example generated from the four steps into a dashboard. In the following, we introduce the first step.

5.1. Step 1: Identification

In the *Identification* step, we identify two covariates sets from all pre-treatment outcome predictors: 1) $\boldsymbol{X}$ `confounders_treatment_name`, a set of covariates used to adjust the treatment assignment mechanism; 2) $\boldsymbol{X}_s$ `confounders_sampling_name`, a set of covariates to adjust the sampling mechanism. By default, `confounders_treatment_name` and `confounders_sampling_name` are the same. To reduce the variance of the weighted average treatment effects, we assign a set of effect modifiers that are predictive of the selection probability to `confounders_ sampling_name`.

```
> library(RCTrep)
> source.data <- RCTrep::source.data
> target.data <- RCTrep::target.data
> vars_name <- list(confounders_treatment_name =
+                   c("x1","x2","x3","x4","x5","x6"),
+                   treatment_name = c('z'),
+                   outcome_name = c('y')
+ )
```

We can use related causal identification software to identify these two sets. The software is but not limited to, e.g., R packages **dosearch** (Tikka, Hyttinen, and Karvanen 2019), **causaleffect** (Tikka and Karvanen 2017), and a web-based software **causalfusion** (Bareinboim and Pearl 2016). To demonstrate the identification of two sets, we present a causal structural diagram of the data generation process of the data used throughout the paper in figure 9. The figure presents predictors of treatment, predictors of outcomes, and predictors of selection. Although in practice the true causal structural diagram of a dataset is unknown, a such diagram can help us identify `confounders_treatment_name` and `confounders_sampling_name` easily.

5.2. Step 2: Estimation

In the *Estimation* step, two sub-steps are summarized, namely, *estimation of the average treatment effects in* `TEstimator`, and *estimation of the weighted average treatment effects in* `SEstimator`. In the first sub-step, we use one estimator to adjust for the treatment assignment mechanism of $\mathcal{S}^{obs}$ in class `TEstimator`, namely, `G-computation` estimator, and one estimator to obtain the unbiased estimate of the truth of $\mathcal{S}^{rct}$ in class `TEstimator`, namely, `Crude` estimator; we use one estimator to adjust for the sampling mechanism of $\mathcal{S}^{obs}$ in class `SEstimator`, namely, exact matching. We first estimate the average treatment effects using $\mathcal{S}^{obs}$.

Step 2.1: Estimation of the average treatment effect

In this step, we estimate the average treatment effects in `TEstimator`. We start out by instantiating objects of class TEstimator using $\mathcal{S}^{obs}$ and $\mathcal{S}^{rct}$. We call TEstimator_wrapper() function to initialize the object source.obj and target.obj using $\mathcal{S}^{obs}$ and $\mathcal{S}^{rct}$ respectively:

```
> source.obj <- TEstimator_wrapper(
+   Estimator = "G_computation",
+   data = source.data,
+   name = "RWD",
+   vars_name = vars_name,
+   outcome_method = "glm",
+   outcome_formula = y ~ x1 + x2 + x3 + z + z:x1 + z:x2 +z:x3+ z:x6,
+   data.public = TRUE
+ )
> target.obj <- TEstimator_wrapper(
+   Estimator = "Crude",
+   data = target.data,
+   name = "RCT",
+   vars_name = vars_name,
+   data.public = TRUE,
+   isTrial = TRUE
+ )
```

We specify the following arguments to instantiate source.obj and target.obj:

1. Estimator: specifying an estimator for the average treatment effects. TEstimator_wrapper() will initialize an TEstimator subclass according to the specified estimator. For instance, if Estimator="G_computation", then the function initializes a subclass G_computation and returns the initialized object.

2. data: a data.frame with n rows and p columns, each row contains variables of characteristics, treatment, and outcome of each individual.

3. name: a character indicating the object name;

4. vars_name: a list containing three vectors with the first element confounders_treatment_name indicating confounding variable names, the second element treatment_ name indicating a treatment variable name, and the third element outcome_name indicating an outcome variable name;

Step 2.2: Estimation of the weighted average treatment effect

In this step, we estimate the weighted average treatment effects in SEstimator. We instantiate a SEstimator subclass SEexact as source.obj.rep by calling the function SEstimator_wrapper():

```
> source.obj.rep <- SEstimator_wrapper(Estimator = "Exact",
+                                      target.obj = target.obj,
+                                      source.obj = source.obj,
+                                      confounders_sampling_name =
+                                         c("x2","x6"))
> source.obj.rep$EstimateRep(stratification = c("x1","x3","x4","x5"))
```

The arguments list for the function `SEstimator_wrapper` is:

1. `Estimator`: a character indicating an estimator for weights $w(\boldsymbol{X}_s)$. The wrapper function initializes a `SEstimator` subclass accordingly;
2. `target.obj` and `source.obj`: `target.obj` indicates an object of which the data is regarded as the target population $\mathcal{P}_\theta$ and the estimates of conditional average treatment effects are regarded as the unbiased estimates of the truth; `source.obj` indicates an object of which the estimates of the average treatment effect is to validate.
3. `confounders_sampling_name`: a character vector of names of $\boldsymbol{X}_s$; the weighted $\boldsymbol{X}_s$ in `source.obj` should be approximately equally distributed to $\boldsymbol{X}_s$ in `target.obj`.

Then we call `EstimateRep()` - the core function of the instantiated object `source.obj.rep`. The function is to estimate the weighted average treatment effect of the target population and subsets using $\mathcal{S}^{obs}$ in `source.obj`. The weighted distribution of `counfounders_sampling_name` in `source.obj` and those in `target.obj` should be balanced. Two optional arguments for the function `EstimateRep()` are specified:

1. `stratification`: a character vector containing covariate names. `EstimateRep()` estimates the weighted average treatment effect of subsets using $\mathcal{S}^{obs}$ in `source.obj`. The subsets are selected according to covariates in `stratifica tion` in combination or individually; default value of `stratification` is `confounders_sampling_name`;
2. `stratification_joint`: a logical value, if `TRUE`, then subsets are selected in combination of all covariates in `stratification`; otherwise, then subsets are selected by covariates in `stratification` individually.

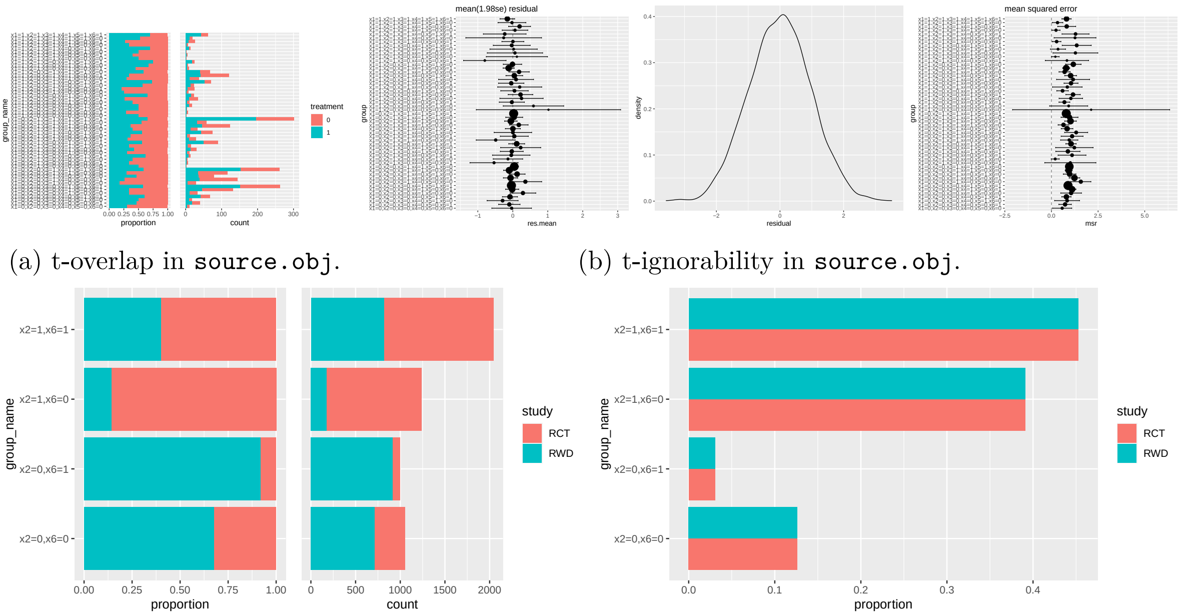
5.3. Step 3: Diagnosis

On completion of all class instantiations, we need to diagnose assumptions for object `source.ob j` of class `TEstimator`, and we need to diagnose assumptions for object `source.obj.rep` of class `SEstimator`:

```
> source.obj$diagnosis_t_overlap()
> source.obj$diagnosis_t_ignorability()
> source.obj.rep$diagnosis_s_overlap()
> source.obj.rep$diagnosis_s_ignorability()
```

Regarding the diagnosis of treatment-related assumptions for the object `source.obj` of class `TEstimator`, i.e., assumptions 3.1 and 3.1:

1. `source.obj` calls `diagnosis_t_overlap()`, and the result is presented in figure 3 (a). The figure shows the diagnosis result of the T-overlap assumption for `source.obj`, which presents the proportion and count of individuals receiving $T = 1$ within sub-populations stratified by `confounders_treatment_name`, and the result shows that there are sufficient individuals in both groups within the sub-populations.



(a) t-overlap in `source.obj`.

(b) t-ignorability in `source.obj`.

(c) s-overlap in `source.obj.rep`.

(d) s-ignorability diagnosis in `source.obj.rep`.

Figure 3: Diagnosis of assumptions in two objects.

2. `source.obj` calls `diagnosis_t_ignorability()`, and the results are presented in the figure 3 (b). Since the class of `source.obj` is `G_computation`, the assumption of T-ignorability for g-computation estimator indicates the assumption of no omitted variable bias. Thus **RCTrep** diagnoses the T-ignorability assumption using the following three metrics, and the results are presented in the three plots in figure 3 (b) respectively:

    - residual mean (1.98 standard error) of sub-populations stratified by `confounders_treatment_name`, which is presented in the left plot in figure 3 (b). The result shows that means of residuals of sub-populations are all very close to zero;
    - distribution of overall residuals, which is presented in the middle plot in figure 3 (b). The result shows that the residual follows a standard normal distribution;
    - mean squared residual (1.98 standard error) of sub-populations stratified by `confounders_treatment_name`, which is presented in the right plot in figure 3 (b). The result shows that the mean squared residual (i.e., mean squared error) of each sub-population is close to 1.

Overall, since the error term of the true data generation process of the example follows a standard normal distribution, the diagnosis results imply that the T-ignorability assumption plausibly holds, thus the estimate of the average treatment effect in `source.obj` are not biased. In addition, since the true variance of the error term is 1, the normal distribution of the residual (the middle plot in figure 3 (b)) and the seemingly constant mean squared residual over sub-populations (the right plot in figure 3 (b)) may imply that no other variable can explain the residual variation, and hence the estimator is the most efficient.

Methods for diagnosis of the T-ignorability assumption depend on the class of `source.obj`.

In case the class is IPW, the object diagnoses the assumption by presenting the inverse propensity score weighted distribution of confounders_treatment_name between treatment and control groups.

Regarding the diagnosis of sampling-related assumptions for the object source.obj.rep of class SEstimator, namely, assumption 3.2 and 3.2:

1. source.obj.rep calls diagnosis_s_overlap(), and the results are presented in figure 3 (c). The figure presents the proportion and count of the number of individuals in $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$ within combined sub-populations stratified by confounders_sampling_name, showing that there are sufficient individuals in the two samples within sub-populations.

2. source.obj.rep calls diagnosis_s_ignorability(), and the result is presented in figure 3 (d). The figure presents the weighted distribution of confounders_sampling_name in $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$, indicating that confounders_sampling_name are balanced between the two samples and hence the sampling mechanism are properly adjusted.

In general, diagnosis of the above four assumptions can help us understand the possible sources that can lead to a discrepancy of estimates between source.obj.rep and target.obj. For instance, near violation of the T-overlap assumption can lead to high variance of estimates of class IPW and high bias of estimates of class G-computation, and near violation of the S-overlap assumption can also lead to high variance of weighted estimates of class SEstimator.

5.4. Step 4: Validation

Lastly, we compute the validation metric in equation 3 on population and sub-population levels. We initialize a class Fusion as an object fusion and assign source.obj, target.obj, and source.obj.r ep to fusion. fusion combines estimates from all objects and validates the average treatment effect of the target population $\mathcal{P}_\theta$ and sub-populations. The sub-populations are selected according to stratification and stratification_joint specified in source.obj.rep$Estim ateRep(). fusion validates estimates in source.obj and source.obj.rep using four metrics, i.e., pseudo mean squared error (mse), length of confidence interval (len_ci), estimate agreement (agg.est), and regulatory agreement (agg.reg) (Franklin *et al.* 2020):

```
> fusion <- Fusion$new(target.obj,
+                      source.obj,
+                      source.obj.rep)
> fusion$evaluate()
```

```
# A tibble: 34 x 7
# Groups:   group_name [17]
   group_name          estimator           size     mse len_ci agg.est agg.reg
   <chr>               <chr>              <dbl>   <dbl>  <dbl> <lgl>   <lgl>
 1 pop                 G_computation/gl~   2622 3.8 e-2  0.92  TRUE    TRUE
 2 pop                 G_computation/glm   2622 6.66e+2  0.239 FALSE   TRUE
 3 x1=0,x3=0,x4=0,x5=0 G_computation/gl~     46 5.58e+0  9.53  TRUE    TRUE
 4 x1=0,x3=0,x4=0,x5=0 G_computation/glm     46 1.58e+3  1.33  FALSE   TRUE
 5 x1=0,x3=0,x4=0,x5=1 G_computation/gl~   105 3.55e+0  8.94  TRUE    TRUE
 6 x1=0,x3=0,x4=0,x5=1 G_computation/glm   105 1.35e+3  0.938 FALSE   TRUE
 7 x1=0,x3=0,x4=1,x5=0 G_computation/gl~   184 1.21e+1  4.64  TRUE    TRUE
 8 x1=0,x3=0,x4=1,x5=0 G_computation/glm   184 1.28e+3  0.707 FALSE   TRUE
 9 x1=0,x3=0,x4=1,x5=1 G_computation/gl~   391 9.47e+0  6.21  TRUE    TRUE
10 x1=0,x3=0,x4=1,x5=1 G_computation/glm   391 9.68e+2  0.502 FALSE   TRUE
# ... with 24 more rows
```

```
> fusion$plot()
```

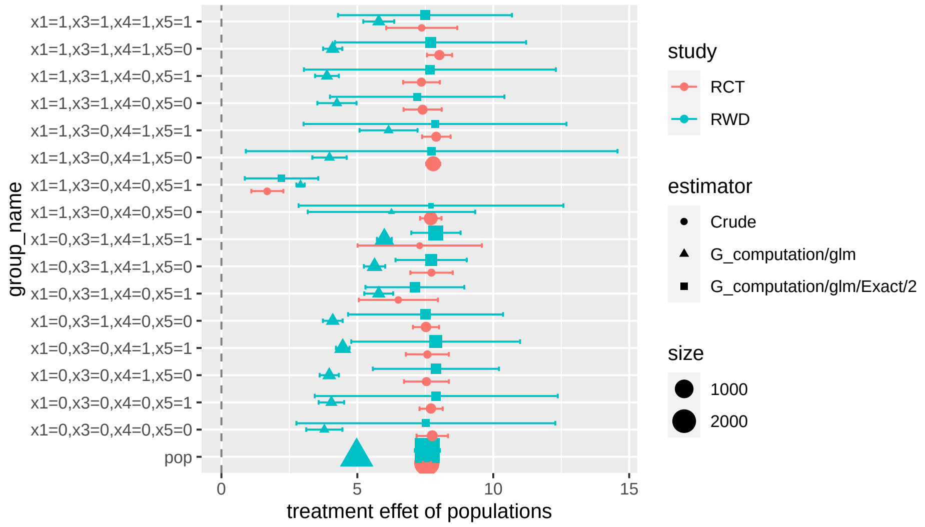


Figure 4: Results for validation of multiple estimates.

where +2 indicates the number of variables in `confounder_sampling_name` used for weight estimation in the object `source.obj.rep`. The result is presented in figure 4, indicating that

1. After adjusting for the treatment assignment mechanism and the sampling mechanism, the point estimates obtained from $\mathcal{S}^{obs}$ (indicated by `G_computation+Exact+6`) are very close to the estimates obtained from $\mathcal{S}^{rct}$ (indicated by `Crude`), on both the population and the sub-population levels. The result implies that the treatment assignment mechanism of $\mathcal{S}^{obs}$ is properly adjusted, and hence the estimates are valid.

2. The point estimates indicated by `G_computation` considerably differ from those indicated by `Crude`, implying that even though the treatment assignment mechanism of $\mathcal{S}^{obs}$ can be properly adjusted, there is a large difference in estimates of (sub-)populations between $\mathcal{S}^{obs}$ and $\mathcal{S}^{rct}$. Without considering the effect of the sampling mechanism on the validation of estimates of the average treatment effect, people may easily attribute the spurious difference to unmeasured confounders of $\mathcal{S}^{obs}$, and question the validity of estimates obtained from $\mathcal{S}^{obs}$.

3. The interval estimates of the weighted average treatment effect of `G_computation+Exact +2` (i.e., `len_ci` of `pop` = 0.92) is wider than those of `G_computation` (`len_ci` of `pop` =0.239), implying that weighting based on $\boldsymbol{X}_s$ inflates the variance of the weighted estimate. This result might be explained by the extreme imbalance of proportion of individuals in $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$ within (sub)-populations stratified by $\boldsymbol{X}_s$, as indicated in figure 3 (c). The imbalance can lead to extreme weights $w(\boldsymbol{X}_s)$, and hence inflate the variance of the weighted estimate. The same findings of the weighted average treatment effect of sub-populations are also observed.

4. The interval estimates of unweighted estimates as indicated by `G_computation` varies across sub-populations, and may be influenced by multiple facts: 1) the sample size of sub-populations; 2) the imbalance of proportion of individuals in treatment and control groups within sub-populations; 3) the variance of an effect modifier that is predictive of the outcome within a sub-population, and wide interval estimates of a sub-population may indicate further stratification or further study on the sub-population to reduce the observed variation.

5.5. Putting all together in a dashboard

RCTrep provides a dashboard that allows users to present all necessary results generated from the four steps and provides users with the flexibility to select sub-population(s) based on which **RCTrep** validates estimates of the average treatment effect. The dashboard can be launched by calling the function:

```
> call_dashboard(source.obj = source.obj,
+                target.obj = target.obj,
+                source.obj.rep = source.obj.rep)
```

Once the interface is launched, users need to select variables in check boxes and click the "Go" buttons to generate related results. Figure 5 shows the dashboard and the generated results. The dashboard contains four panels, i.e., Identification, Estimation, Diagnosis, and Validation. Identification offers two sets of variables used for adjusting the treatment and the sampling mechanism, and one additional set of variables for selecting sub-populations; Estimation provides point and interval estimates of the average treatment effect of selected sub-populations; Diagnosis provides diagnosis results of treatment- and sampling-related assumptions; Validation presents and compares point and interval estimates of population and selected sub-populations. In the following, we introduce the basic workflow of the dashboard and the usage of each panel respectively:

1. The *Identification* panel provides three boxes:
   - Confounders: a set of potential confounders; by default, the selected variables are `confounders_treatment_name` defined in `source.obj`; by clicking "Go" the boxes named T-overlap and T-ignorability will present the diagnosis results of the T-overlap assumption and the T-ignorability assumption, respectively;
   - Effect modifiers: a set of effect modifiers; by default, the selected variables are `confounders_sampling_name` defined in `source.obj.rep`; by clicking "Go" the

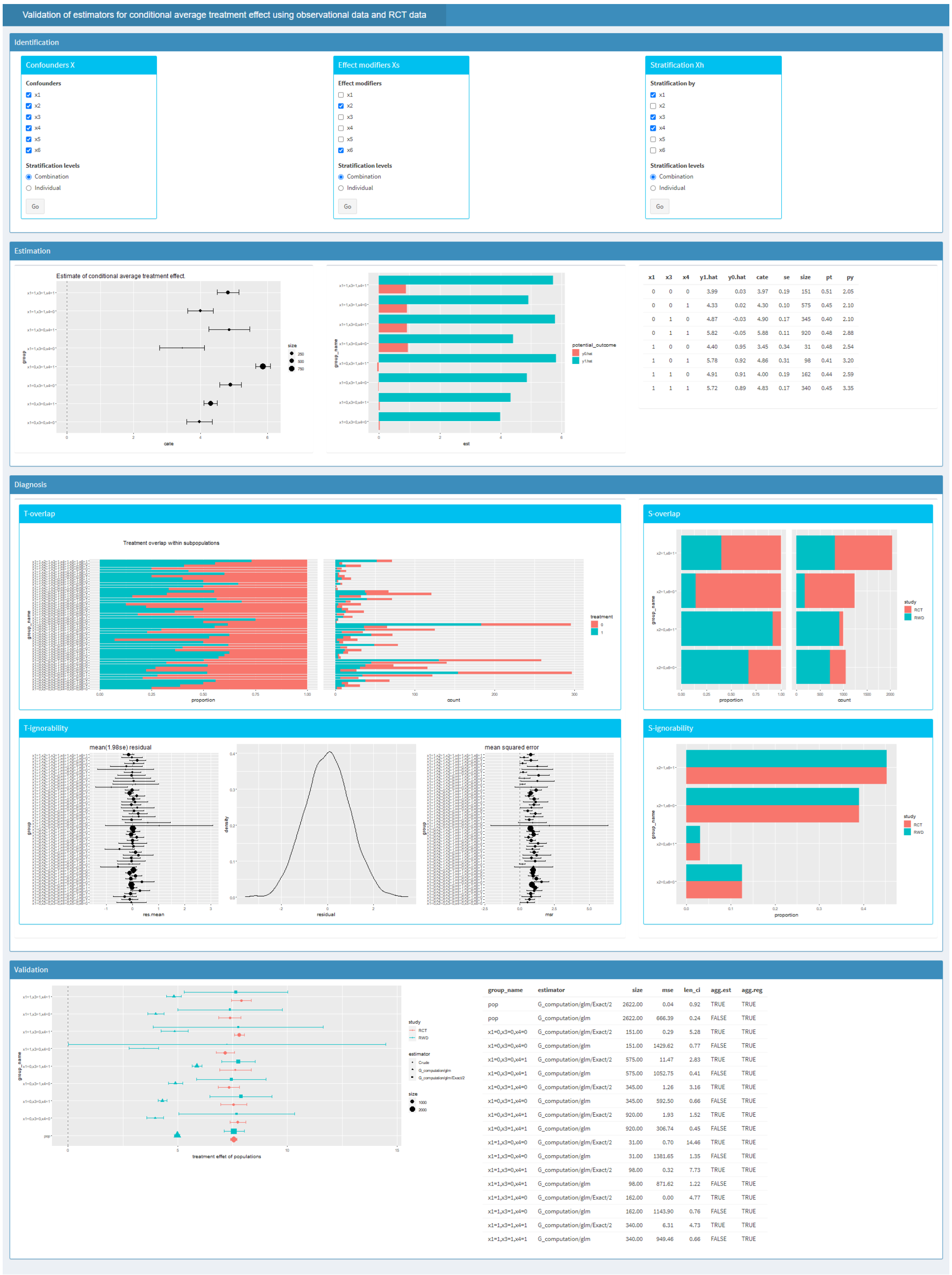
boxes named S-overlap and S-ignorability will present diagnosis results of the S-overlap assumption and the S-ignorability assumption, respectively;

- Stratification: a set of all pre-treatment outcome predictors. The box provides variables based on which sub-populations can be defined and selected; no default values are selected. By clicking "Go" the Estimation panel will present estimates of the average treatment effect of the selected sub-populations, and the Validation panel will present the validation results of the selected sub-populations. In the figure 5, we select x1,x3, and x4 for simplicity.

2. The *Estimation* panel plots the average treatment effect and the average predicted potential outcomes of the selected sub-populations defined in the box of Stratification, and prints the numeric values accordingly. Additional values `pt` and `py`, denoting the proportion of treatment and proportion of positive outcome within the sub-populations, are also printed.

3. The *Diagnosis* panel diagnoses the T-overlap and the T-ignorability assumptions; the panel diagnoses S-overlap and S-ignorability assumptions.

4. The *Validation* aggregates and plots estimates of the average treatment effect of the population and the selected sub-populations in `target.obj`, `source.obj` and `source.obj.rep`, and prints numeric results of the validation metrics in which the truth is the unbiased estimate in `target.obj`.



Validation of estimators for conditional average treatment effect using observational data and RCT data

Identification

Confounders X

Confounders

☑ x1
☑ x2
☑ x3
☑ x4
☑ x5
☑ x6

Stratification levels

● Combination
○ Individual

Go

Effect modifiers Xs

Effect modifiers

□ x1
☑ x2
□ x3
□ x4
□ x5
☑ x6

Stratification levels

● Combination
○ Individual

Go

Stratification Xh

Stratification by

☑ x1
□ x2
☑ x3
☑ x4
□ x5
□ x6

Stratification levels

● Combination
○ Individual

Go

Estimation

x1	x3	x4	y1.hat	y0.hat	cate	se	size	pt	py
0	0	0	3.99	0.03	3.97	0.19	151	0.51	2.05
0	0	1	4.33	0.02	4.30	0.10	575	0.45	2.10
0	1	0	4.87	-0.03	4.90	0.17	345	0.40	2.10
0	1	1	5.82	-0.05	5.88	0.11	920	0.48	2.88
1	0	0	4.40	0.95	3.45	0.34	31	0.48	2.54
1	0	1	5.78	0.92	4.86	0.31	98	0.41	3.20
1	1	0	4.91	0.91	4.00	0.19	162	0.44	2.59
1	1	1	5.72	0.89	4.83	0.17	340	0.45	3.35

Diagnosis

T-overlap

S-overlap

T-ignorability

S-ignorability

Validation

group_name	estimator	size	mse	len_ci	agg.est	agg.reg
pop	G_computation/glm/Exact/2	2622.00	0.04	0.92	TRUE	TRUE
pop	G_computation/glm	2622.00	666.39	0.24	FALSE	TRUE
x1=0,x3=0,x4=0	G_computation/glm/Exact/2	151.00	0.29	5.28	TRUE	TRUE
x1=0,x3=0,x4=0	G_computation/glm	151.00	1429.62	0.77	FALSE	TRUE
x1=0,x3=0,x4=1	G_computation/glm/Exact/2	575.00	11.47	2.83	TRUE	TRUE
x1=0,x3=0,x4=1	G_computation/glm	575.00	1052.75	0.41	FALSE	TRUE
x1=0,x3=1,x4=0	G_computation/glm/Exact/2	345.00	1.26	3.16	TRUE	TRUE
x1=0,x3=1,x4=0	G_computation/glm	345.00	592.50	0.66	FALSE	TRUE
x1=0,x3=1,x4=1	G_computation/glm/Exact/2	920.00	1.93	1.52	TRUE	TRUE
x1=0,x3=1,x4=1	G_computation/glm	920.00	306.74	0.45	FALSE	TRUE
x1=1,x3=0,x4=0	G_computation/glm/Exact/2	31.00	0.70	14.46	TRUE	TRUE
x1=1,x3=0,x4=0	G_computation/glm	31.00	1381.65	1.35	FALSE	TRUE
x1=1,x3=0,x4=1	G_computation/glm/Exact/2	98.00	0.32	7.73	TRUE	TRUE
x1=1,x3=0,x4=1	G_computation/glm	98.00	871.62	1.22	FALSE	TRUE
x1=1,x3=1,x4=0	G_computation/glm/Exact/2	162.00	0.00	4.77	TRUE	TRUE
x1=1,x3=1,x4=0	G_computation/glm	162.00	1143.90	0.76	FALSE	TRUE
x1=1,x3=1,x4=1	G_computation/glm/Exact/2	340.00	6.31	4.73	TRUE	TRUE
x1=1,x3=1,x4=1	G_computation/glm	340.00	949.46	0.66	FALSE	TRUE

Figure 5: **RCTrep** dashboard to interactively visualize all results generated from estimation, diagnosis, and validation steps.

6. Running working examples

In this section, we demonstrate three examples for validating estimates of the average treatment effect using **RCTrep**. The first example demonstrates the validation of multiple estimates at scale. The second example demonstrates the validation in case only sub-population level data of two data sets are allowed to share. The third example demonstrates the validation using synthetic RCT data. In the following, we first introduce using **RCTrep** to validate multiple estimates.

6.1. Example 1: Validation at scale

In the following, we demonstrate how to validate multiple estimates using **RCTrep**. We instantiated multiple objects, and combine the objects in an object of class `Fusion`:

```
> library(RCTrep)
> source.data <- RCTrep::source.data
> target.data <- RCTrep::target.data
> vars_name <- list(confounders_treatment_name =
+                   c("x1","x2","x3","x4","x5","x6"),
+                   treatment_name = c('z'),
+                   outcome_name = c('y')
+ )
> source.obj.gc <- TEstimator_wrapper(
+   Estimator = "G_computation",
+   data = source.data,
+   name = "RWD",
+   vars_name = vars_name,
+   outcome_method = "glm",
+   outcome_formula = y ~ x1 + x2 + x3 + z + z:x1 + z:x2 +z:x3+ z:x6,
+   data.public = TRUE
+ )
> source.obj.ipw <- TEstimator_wrapper(
+   Estimator = "IPW",
+   data = source.data,
+   name = "RWD",
+   vars_name = vars_name,
+   treatment_method = "glm",
+   treatment_formula = z ~ x1 + x2 + x3 + x4 + x5 + x6 + x1:x2 + x3:x4,
+   data.public = TRUE
+ )
> source.obj.dr <- TEstimator_wrapper(
+   Estimator = "DR",
+   data = source.data,
+   name = "RWD",
+   vars_name = vars_name,
+   outcome_method = "glm",
+   outcome_formula = y ~ x1 + x2 + x3 + z + z:x1 + z:x2 +z:x3+ z:x6,
+   treatment_method = "glm",
+   treatment_formula = z ~ x1 + x2 + x3 + x4 + x5 + x6 + x1:x2 + x3:x4,
+   data.public = TRUE
+ )
> target.obj <- TEstimator_wrapper(
+   Estimator = "Crude",
+   data = target.data,
+   name = "RCT",
+   vars_name = vars_name,
+   data.public = TRUE,
+   isTrial = TRUE
+ )
> strata <- c("x1","x4")
> confounders_sampling_name <- c("x2","x6")
> source.gc.exact <- SEstimator_wrapper(Estimator = "Exact",
+                                       target.obj = target.obj,
+                                       source.obj = source.obj.gc,
+                                       confounders_sampling_name =
+                                       confounders_sampling_name)
> source.gc.exact$EstimateRep(stratification = strata,
+                             stratification_joint = TRUE)
> source.gc.isw <- SEstimator_wrapper(Estimator = "ISW",
+                                     target.obj = target.obj,
+                                     source.obj = source.obj.gc,
+                                     confounders_sampling_name =
+                                     confounders_sampling_name,
+                                     method = "glm")
> source.gc.isw$EstimateRep(stratification = strata,
+                           stratification_joint = TRUE)
> source.gc.subclass <- SEstimator_wrapper(Estimator = "Subclass",
+                                          target.obj = target.obj,
+                                          source.obj = source.obj.gc,
+                                          confounders_sampling_name =
+                                          confounders_sampling_name)
> source.gc.subclass$EstimateRep(stratification = strata,
+                                stratification_joint = TRUE)
> source.ipw.exact <- SEstimator_wrapper(Estimator = "Exact",
+                                        target.obj = target.obj,
+                                        source.obj = source.obj.ipw,
+                                        confounders_sampling_name =
+                                        confounders_sampling_name)
> source.ipw.exact$EstimateRep(stratification = strata,
+                              stratification_joint = TRUE)
> source.ipw.isw <- SEstimator_wrapper(Estimator = "ISW",
+                                      target.obj = target.obj,
+                                      source.obj = source.obj.ipw,
+                                      confounders_sampling_name =
+                                      confounders_sampling_name,
+                                      method = "glm")
> source.ipw.isw$EstimateRep(stratification = strata,
+                             stratification_joint = TRUE)
> source.ipw.subclass <- SEstimator_wrapper(Estimator = "Subclass",
+                                           target.obj = target.obj,
+                                           source.obj = source.obj.ipw,
+                                           confounders_sampling_name =
+                                           confounders_sampling_name)
> source.ipw.subclass$EstimateRep(stratification = strata,
+                                  stratification_joint = TRUE)
> source.dr.exact <- SEstimator_wrapper(Estimator = "Exact",
+                                       target.obj = target.obj,
+                                       source.obj = source.obj.dr,
+                                       confounders_sampling_name =
+                                       confounders_sampling_name)
> source.dr.exact$EstimateRep(stratification = strata,
+                              stratification_joint = TRUE)
> source.dr.isw <- SEstimator_wrapper(Estimator = "ISW",
+                                     target.obj = target.obj,
+                                     source.obj = source.obj.dr,
+                                     confounders_sampling_name =
+                                     confounders_sampling_name,
+                                     method = "glm")
> source.dr.isw$EstimateRep(stratification = strata,
+                            stratification_joint = TRUE)
> source.dr.subclass <- SEstimator_wrapper(Estimator = "Subclass",
+                                          target.obj = target.obj,
+                                          source.obj = source.obj.dr,
+                                          confounders_sampling_name =
+                                          confounders_sampling_name)
> source.dr.subclass$EstimateRep(stratification = strata,
+                                 stratification_joint = TRUE)
> fusion <- Fusion$new(target.obj,
+                       source.gc.exact,
+                       source.gc.isw,
+                       source.gc.subclass,
+                       source.ipw.exact,
+                       source.ipw.isw,
+                       source.ipw.subclass,
+                       source.dr.exact,
+                       source.dr.isw,
+                       source.dr.subclass)
> fusion$plot()
```

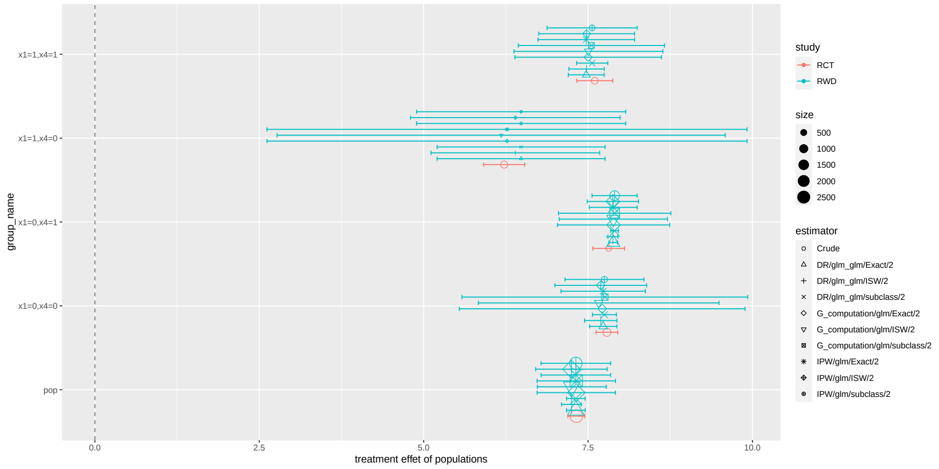


Figure 6: Comparisons of 9 (3*3) estimates.

```
> fusion$evaluate()

# A tibble: 45 x 7
# Groups:   group_name [5]
   group_name estimator                  size    mse len_ci agg.est agg.reg
   <chr>      <chr>                     <dbl>  <dbl>  <dbl> <lgl>   <lgl>
   pop        G_computation+Exact+2      2618  0      1.19  TRUE    TRUE
   pop        G_computation+subclass+2   2618  0      1.19  TRUE    TRUE
   pop        IPW+Exact+2                2618  0.005  1.06  TRUE    TRUE
   pop        IPW+subclass+2             2618  0.005  1.06  TRUE    TRUE
   pop        DR+Exact+2                 2618  0.005  0.283 TRUE    TRUE
   pop        DR+subclass+2              2618  0.005  0.283 TRUE    TRUE
   pop        IPW+ISW+2                  2618  0.56   1.09  TRUE    TRUE
   pop        DR+ISW+2                   2618  0.56   0.298 TRUE    TRUE
   pop        G_computation+ISW+2        2618 12.1    1.01  FALSE   TRUE
   x1=0,x4=0  G_computation+subclass+2    527  0.101  4.34  TRUE    TRUE
# ... with 35 more rows
```

The results show that using `G-computation` and `Exact` weighting is the most accurate in terms of pseudo mean squared error, which is in line with the results in existing literature Chatton *et al.* (2020); Le Borgne, Chatton, Léger, Lenain, and Foucher (2021); Loiseau, Trichelair, He, Andreux, Zaslavskiy, Wainrib, and Blum (2022).

6.2. Example 2: Validation using aggregated data

RCTrep provides a solution to validating estimates of the average treatment effect using aggregated data of sub-populations. We start out by instantiating an object `source.obj` using $\mathcal{S}^{obs}$ and an object `target.obj` using $\mathcal{S}^{rct}$ [1]:

[1] note that in the example 2, we have pre-processed the data to instantiate two objects: the rows in `source.data` and `target.data` that has no match on the specified column `confounders_sampling_name` are removed.

```
> library(geex)
> library(caret)
> source.data <- RCTrep::source.data
> target.data <- RCTrep::target.data
>
> # Identification
> vars_name <- list(confounders_treatment_name =
+                    c("x1","x2","x3","x4","x5","x6"),
+                    treatment_name = c('z'),
+                    outcome_name = c('y')
+ )
> confounders_sampling_name <- c("x2","x6")
>
> # Estimate conditional average treatment effect
> source.obj <- TEstimator_wrapper(
+   Estimator = "G_computation",
+   data = source.data,
+   vars_name = vars_name,
+   outcome_method = "glm",
+   outcome_form=y ~ x1 + x2 + x3 + z + z:x1 + z:x2 +z:x3+ z:x6,
+   name = "RWD",
+   data.public = FALSE
+ )
> target.obj <- TEstimator_wrapper(
+   Estimator = "Crude",
+   data = target.data,
+   vars_name = vars_name,
+   name = "RCT",
+   data.public = FALSE,
+   isTrial = TRUE
+ )
```

We specify `data.public=FALSE` to indicate that full data is not allowed to share. `TEstimator_wrapper()` returns an object of class `TEstimator_pp` of which the public field `data` is aggregated data of sub-populations stratified by levels of all variables in `confounders_treatment_name` in combination:

```
> head(source.obj$data)
```

```
  x1 x2 x3 x4 x5 x6   y1.hat   y0.hat     cate           se size id
1  0  0  0  0  0  0 3.607016 1.607016 2.000000 3.598499e-08    5  1
2  0  0  0  0  0  1 4.607016 1.607016 3.000000 2.473959e-15   29  2
3  0  0  0  0  1  0 3.607016 1.607016 2.000000 4.586534e-16   15  3
4  0  0  0  0  1  1 4.607016 1.607016 3.000000 1.133129e-15   71  4
5  0  0  0  1  0  0 3.607016 1.607016 2.000000 9.071183e-16   29  5
6  0  0  0  1  0  1 4.607016 1.607016 3.000000 2.315886e-15  128  6
```

Then we specify `strata` indicating the sub-populations of which the average treatment effects are to validate, and instantiate an object `source.rep.obj` of class `SEstimator_pp` to compute the weighted average treatment effects of the sub-populations:

```
> # Estimate the weighted conditional average treatment effect of source.obj
> strata <- c("x1","x4")
> source.rep.obj <- SEstimator_wrapper(Estimator = "Exact",
+                                      target.obj = target.obj,
+                                      source.obj = source.obj,
+                                      confounders_sampling_name =
+                                      confounders_sampling_name)
> source.rep.obj$EstimateRep(stratification = strata,
+                            stratification_joint = TRUE)
> # Validate
> fusion <- Fusion$new(target.obj,
+                      source.obj,
+                      source.rep.obj)
> fusion$plot()
```

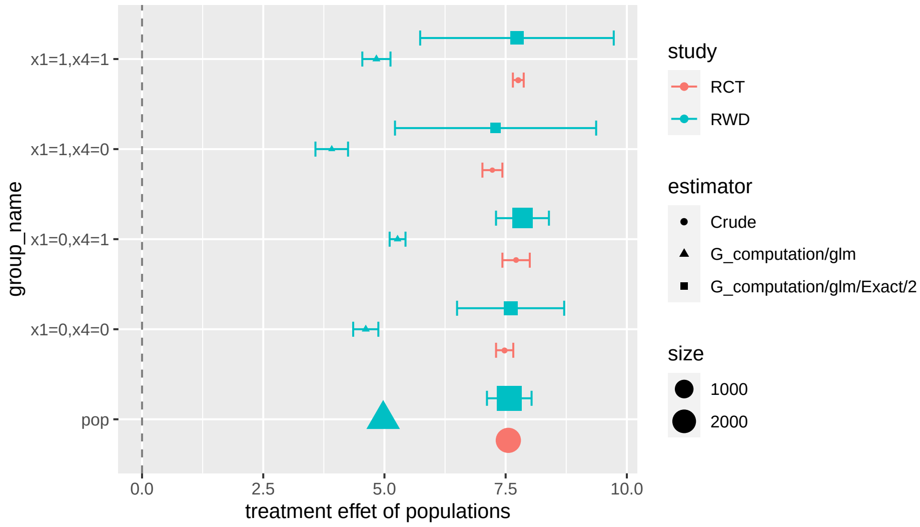


Figure 7: Validation results based on aggregated data of sub-populations in RCT and RWD.

```
> fusion$evaluate()

  group_name estimator                  size     mse len_ci agg.est agg.reg
  <chr>      <chr>                     <dbl>   <dbl>  <dbl> <lgl>   <lgl>  
1 pop        G_computation/glm/Exact/2  2622   0.038  0.92  TRUE    TRUE   
2 pop        G_computation/glm          2622 666.     0.239 FALSE   TRUE   
3 x1=0,x4=0  G_computation/glm/Exact/2   496   1.50   2.21  TRUE    TRUE   
4 x1=0,x4=0  G_computation/glm            14 821.     0.519 FALSE   TRUE   
```

```
 5 x1=0,x4=1  G_computation/glm/Exact/2  1495    1.73    1.09  TRUE    TRUE
 6 x1=0,x4=1  G_computation/glm            12  598.      0.327 FALSE   TRUE
...
```

6.3. Example 3: Validation using synthetic RCT data

In example 2 we demonstrate the validation approach using aggregated data of sub-populations from $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$. However, in practice, we don't even have access to such aggregated RCT data. In most cases, We only have aggregated data of each variable and average treatment effects of sub-populations stratified by the variable individually. In example 3 we demonstrate using the marginal distribution of variables and estimates of sub-populations stratified by these variables individually to generate synthetic RCT data for validation. First, for a demonstrative purpose, we instantiate an object of class `Crude` using full RCT data. We obtain the marginal distribution of variables $\{X_k \in \boldsymbol{X}; k = 1, ..., d\}$ of RCT data as descriptive statistics of the target population $\mathcal{P}_\theta$, and obtain the estimates of the average treatment effect of populations stratified by the variable individually as the truth for validation:

```
> library(dplyr)
> source.data <- RCTrep::source.data
> target.data <- RCTrep::target.data
>
> # Identification
> vars_name <- list(confounders_treatment_name =
+                     c("x1","x2","x3","x4","x5","x6"),
+                     treatment_name = c('z'),
+                     outcome_name = c('y')
+ )
>
> # Generate target.obj using full dataset
> target.obj <- TEstimator_wrapper(
+   Estimator = "Crude",
+   data = target.data,
+   vars_name = vars_name,
+   name = "RCT",
+   data.public = FALSE,
+   isTrial = TRUE
+ )
>
> # Get unbiased estimates of conditional average treatment effect
> vars_rct <- c("x1","x2","x3","x4","x5","x6")
> RCT.estimates <- list(ATE_mean = target.obj$estimates$ATE$est,
+                         ATE_se = target.obj$estimates$ATE$se,
+                         CATE_mean_se = target.obj$get_CATE(vars_rct,FALSE))
```

Then we generate a synthetic RCT dataset `synthetic.data` using the marginal distributions of the variables X_k by calling the **RCTrep** function `GenerateSyntheticData()`. In the function, We specify a marginal distribution of each variable and pairwise correlations between the variables. Then the function generates the synthetic data of the RCT accordingly:

```
> # Simulate synthetic RCT data given marginal distributions
> emp.p1 <- mean(target.data$x1)
> emp.p2 <- mean(target.data$x2)
> emp.p3 <- mean(target.data$x3)
> emp.p4 <- mean(target.data$x4)
> emp.p5 <- mean(target.data$x5)
> emp.p6 <- mean(target.data$x6)
> t.d <- target.data[,vars_rct]
> n <- dim(source.data)[1]
> pw.cor <- gdata::upperTriangle(cor(t.d), diag = FALSE, byrow = TRUE)
> synthetic.data <- RCTrep::GenerateSyntheticData(
+   margin_dis="bernoulli",
+   N = n,
+   margin = list(emp.p1, emp.p2, emp.p3, emp.p4, emp.p5, emp.p6),
+   var_name = vars_rct,
+   pw.cor = pw.cor)
> head(synthetic.data)
```

```
  x1 x2 x3 x4 x5 x6
1  1  1  0  0  0  1
2  0  1  0  0  0  1
3  1  1  1  0  0  1
4  1  1  0  0  0  1
5  1  1  0  0  0  1
6  1  0  1  0  0  0
...
```

Then we instantiate `target.obj` of class `TEstimator_Synthetic`. We initialize the public field `data` by assigning `synthetic.data` to `df`; initialize the public field `estimates` by assigning `RCT.estimates` to `estimates`; initialize the public field `confounders_treatment_name` by assigning `c("x1","x2","x3","x4","x5","x6")` to `vars_name`. Note that `synthetic.data` might slightly shift from the true target population $\mathcal{P}_\theta$.

```
> # Initiate target.obj using synthetic data and unbiased estimates
> synthetic.data <- semi_join(synthetic.data, source.data, by = vars_rct)
> target.obj <- TEstimator_Synthetic$new(data = synthetic.data,
+                                         estimates=RCT.estimates,
+                                         vars_name = vars_name,
+                                         name = "RCT",
+                                         isTrial = TRUE,
+                                         data.public = TRUE)
>
> # Estimate conditional average treatment effect
> source.data <- semi_join(source.data, synthetic.data, by = vars_rct)
> source.obj <- TEstimator_wrapper(
+   Estimator = "G_computation",
+   data = source.data,
+   vars_name = vars_name,
+   outcome_method = "glm",
+   outcome_form=y ~ x1 + x2 + x3 + z + z:x1 + z:x2 +z:x3+ z:x6,
+   name = "RWD",
+   data.public = TRUE
+ )
>
> # Estimate weighted conditional average treatment effect
> source.rep.obj <- SEstimator_wrapper(Estimator="Exact",
+                                      target.obj=target.obj,
+                                      source.obj=source.obj,
+                                      confounders_sampling_name=
+                                      c("x2","x6"))
> source.rep.obj$EstimateRep(stratification = vars_rct,
+                            stratification_joint = FALSE)
>
> # Combine objects and validate estimates
> fusion <- Fusion$new(target.obj,
+                      source.obj,
+                      source.rep.obj)
> fusion$plot()
```

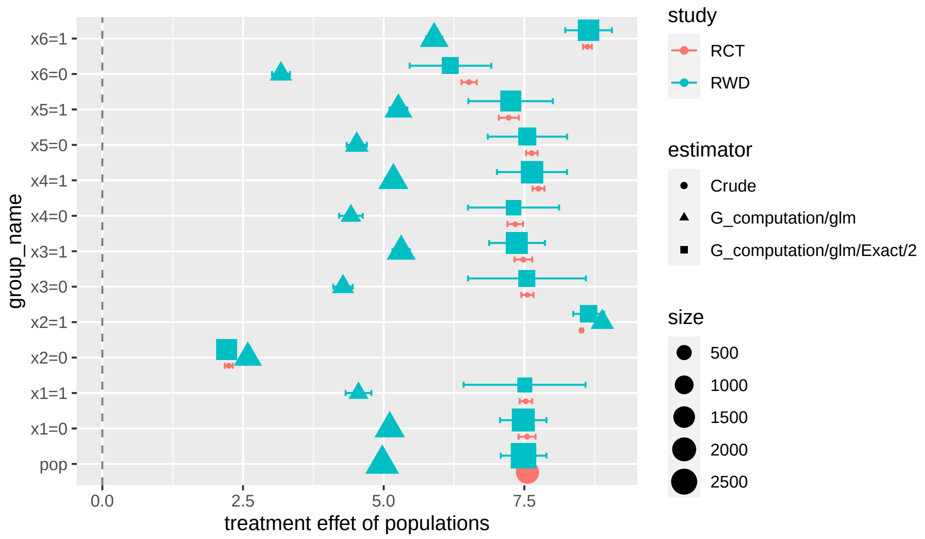


Figure 8: Validation results where the sampling weights of source.obj is estimated based on synthetic RCT data.

```
> fusion$evaluate()
```

```
   group_name estimator                    size     mse len_ci agg.est agg.reg
   <chr>      <chr>                       <dbl>   <dbl>  <dbl> <lgl>   <lgl>
 1 pop        G_computation/glm/Exact/2  2254   0.168  0.768 TRUE    TRUE
 2 pop        G_computation/glm          2254 502.     0.268 FALSE   TRUE
 3 x1=0       G_computation/glm/Exact/2  1694   0.631  0.783 TRUE    TRUE
 4 x1=0       G_computation/glm          1694 425.     0.313 FALSE   TRUE
 5 x1=1       G_computation/glm/Exact/2   560   0.159  2.09  TRUE    TRUE
 6 x1=1       G_computation/glm           560 746.     0.504 FALSE   TRUE
 7 x2=0       G_computation/glm/Exact/2  1260   0.106  0.051 TRUE    TRUE
 8 x2=0       G_computation/glm          1260   6.41   0.064 FALSE   TRUE
 9 x2=1       G_computation/glm/Exact/2   994   1.51   0.519 FALSE   TRUE
10 x2=1       G_computation/glm           994  13.4    0.055 FALSE   TRUE
# ... with 16 more rows
```

Results in figure 8 show that even though we don't have full RCT data, the weighted estimates of the average treatment effects (indicated by `G_computation/glm/Exact/2`) can still be closer to the unbiased estimate of the truth (indicated by `Crude`) compared to unweighted estimates (indicated by `G_computation/glm`). Hence we can still validate estimates of the average treatment effects to some extent and obtain qualitative results, e.g., the direction of effects. In general, effect modifiers that are highly predictive of the selection probability, which can lead to a large discrepancy in estimates between samples, should be weighted.

7. Discussion

The software package **RCTrep** aims to help researchers and policymakers to validate estimates of the average treatment effects of (sub-)populations obtained from observational data in case randomized controlled trial data is (at least partially) accessible. **RCTrep** provides three classes of estimators for the average treatment effect and three classes of estimators for the weight, and provides a variety of modeling choices for outcome, treatment, and sampling. **RCTrep** validates estimates on both population and sub-population levels, providing a deeper insight into the performance of the estimators. **RCTrep** also allows for validation using solely aggregated data of sub-populations.

RCTrep highlights the importance of making RCT data more accessible in order to allow the validation of estimates of average treatment effects obtained from observational data. We recognize the unreplaceable role of RCT data in fueling the power of observational data with advanced methods to drive more precise treatment decision-making. The software package is under continual maintenance and periodic significant upgrading. Further development will include 1) enrich methods for estimating the average treatment effect in the class `TEstimator`, for instance, balancing-based methods via optimization (Chattopadhyay, Hase, and Zubizarreta 2020; Dong *et al.* 2020) and bayesian network; 2) enrich methods for estimating the weight in the class `SEstimator`; 3) offer more options for uncertainty quantification of (weighted) average treatment effects, for instance, delta method, bootstrap re-sampling, double bootstrap (Ackerman *et al.* 2021), a consistent sandwich-type variance estimator (Buchanan, Hudgens, Cole, Mollan, Sax, Daar, Adimora, Eron, and Mugavero 2018), and parametric simulation-based method (Chatton *et al.* 2020); 4) support more types of the treatment and the outcome variables, e.g., nominal variables that have more than two categories for the treatment variable, time to event data type for the outcome variable.

References

Aalen OO, Farewell VT, De Angelis D, Day NE, Nöel Gill O (1997). "A Markov model for HIV disease progression including the effect of HIV diagnosis and treatment: application to AIDS prediction in England and Wales." *Statistics in medicine*, **16**(19), 2191–2210.

Ackerman B, Lesko CR, Siddique J, Susukida R, Stuart EA (2021). "Generalizing randomized trial findings to a target population using complex survey population data." *Statistics in Medicine*, **40**(5), 1101–1120.

Alaa A, Van Der Schaar M (2019). "Validating causal inference models via influence functions." In *International Conference on Machine Learning*, pp. 191–201. PMLR.

Atan O, Jordon J, Van der Schaar M (2018). "Deep-treat: Learning optimal personalized treatments from observational data using neural networks." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Bareinboim E, Pearl J (2016). "Causal inference and the data-fusion problem." *Proceedings of the National Academy of Sciences*, **113**(27), 7345–7352.

Bica I, Alaa AM, Lambert C, Van Der Schaar M (2021). "From real-world patient data to individualized treatment effects using machine learning: current and future methods to address underlying challenges." *Clinical Pharmacology & Therapeutics*, **109**(1), 87–100.

Buchanan AL, Hudgens MG, Cole SR, Mollan KR, Sax PE, Daar ES, Adimora AA, Eron JJ, Mugavero MJ (2018). "Generalizing evidence from randomized trials using inverse probability of sampling weights." *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **181**(4), 1193–1209.

Chang W (????). "R6: Classes with Reference Semantics, 2017." *URL https://CRAN. R-project. org/package*, **6**, 90.

Chatton A, Le Borgne F, Leyrat C, Gillaizeau F, Rousseau C, Barbin L, Laplaud D, Léger M, Giraudeau B, Foucher Y (2020). "G-computation, propensity score-based methods, and targeted maximum likelihood estimator for causal inference with different covariates sets: a comparative simulation study." *Scientific reports*, **10**(1), 1–13.

Chattopadhyay A, Hase CH, Zubizarreta JR (2020). "Balancing vs modeling approaches to weighting in practice." *Statistics in Medicine*, **39**(24), 3227–3254.

Cheng L, Guo R, Moraffah R, Sheth P, Candan KS, Liu H (2022). "Evaluation Methods and Measures for Causal Learning Algorithms." *IEEE Transactions on Artificial Intelligence*.

Cinelli C, Pearl J (2021). "Generalizing experimental results by leveraging knowledge of mechanisms." *European Journal of Epidemiology*, **36**(2), 149–164.

Colnet B, Mayer I, Chen G, Dieng A, Li R, Varoquaux G, Vert JP, Josse J, Yang S (2020). "Causal inference methods for combining randomized trials and observational studies: a review." *arXiv preprint arXiv:2011.08047*.

Dahabreh IJ, Robertson SE, Steingrimsson JA, Stuart EA, Hernan MA (2020). "Extending inferences from a randomized trial to a new target population." *Statistics in medicine*, **39**(14), 1999–2014.

Dong L, Yang S, Wang X, Zeng D, Cai J (2020). "Integrative analysis of randomized clinical trials with real world evidence studies." *arXiv preprint arXiv:2003.01242*.

Dorie V, Hill J, Shalit U, Scott M, Cervone D (2019). "Automated versus do-it-yourself methods for causal inference: Lessons learned from a data analysis competition." *Statistical Science*, **34**(1), 43–68.

Egami N, Hartman E (2018). "Covariate selection for generalizing experimental results." *Technical report*, Technical report Working Paper.

Franklin JM, Pawar A, Martin D, Glynn RJ, Levenson M, Temple R, Schneeweiss S (2020). "Nonrandomized real-world evidence to support regulatory decision making: process for a randomized trial replication project." *Clinical Pharmacology & Therapeutics*, **107**(4), 817–826.

Franklin JM, Schneeweiss S, Polinski JM, Rassen JA (2014). "Plasmode simulation for the evaluation of pharmacoepidemiologic methods in complex healthcare databases." *Computational statistics & data analysis*, **72**, 219–226.

Franz M (2020). "JustCause: Comparing methods for causality analysis in a fair and just way." `https://justcause.readthedocs.io/en/latest/#`.

Hill JL (2011). "Bayesian nonparametric modeling for causal inference." *Journal of Computational and Graphical Statistics*, **20**(1), 217–240.

Hitsch GJ, Misra S (2018). "Heterogeneous treatment effects and optimal targeting policy evaluation." *Available at SSRN 3111957*.

Hosmer Jr DW, Lemeshow S, Sturdivant RX (2013). *Applied logistic regression*, volume 398. John Wiley & Sons.

Imbens GW, Rubin DB (2015). *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press.

Jiang H, Qi P, Zhou J, Zhou J, Rao S (2021). "A Short Survey on Forest Based Heterogeneous Treatment Effect Estimation Methods: Meta-learners and Specific Models." In *2021 IEEE International Conference on Big Data (Big Data)*, pp. 3006–3012. IEEE.

Johansson FD, Shalit U, Kallus N, Sontag D (2020). "Generalization bounds and representation learning for estimation of potential outcomes and causal effects." *arXiv preprint arXiv:2001.07426*.

Kang JD, Schafer JL (2007). "Demystifying double robustness: A comparison of alternative strategies for estimating a population mean from incomplete data." *Statistical science*, pp. 523–539.

Le Borgne F, Chatton A, Léger M, Lenain R, Foucher Y (2021). "G-computation and machine learning for estimating the causal effects of binary exposure statuses on binary outcomes." *Scientific reports*, **11**(1), 1–12.

Loiseau N, Trichelair P, He M, Andreux M, Zaslavskiy M, Wainrib G, Blum MG (2022). "External control arm analysis: an evaluation of propensity score approaches, G-computation, and doubly debiased machine learning." *medRxiv*. doi:10.1101/2022.01.28.22269591. https://www.medrxiv.org/content/early/2022/01/30/2022.01.28.22269591.full.pdf, URL https://www.medrxiv.org/content/early/2022/01/30/2022.01.28.22269591.

Lunceford JK, Davidian M (2004). "Stratification and weighting via the propensity score in estimation of causal treatment effects: a comparative study." *Statistics in medicine*, **23**(19), 2937–2960.

Powers S, Qian J, Jung K, Schuler A, Shah NH, Hastie T, Tibshirani R (2018). "Some methods for heterogeneous treatment effect estimation in high dimensions." *Statistics in medicine*, **37**(11), 1767–1787.

Rosenbaum PR, Rubin DB (1983). "The central role of the propensity score in observational studies for causal effects." *Biometrika*, **70**(1), 41–55. doi:10.1093/BIOMET/70.1.41.

Rudolph KE, Schmidt NM, Glymour MM, Crowder R, Galin J, Ahern J, Osypuk TL (2018). "Composition or context: using transportability to understand drivers of site differences in a large-scale housing experiment." *Epidemiology (Cambridge, Mass.)*, **29**(2), 199.

Saul BC, Hudgens MG (2020). "The Calculus of M-Estimation in R with geex." *Journal of statistical software*, **92**(2).

Schuemie MJ, Cepeda MS, Suchard MA, Yang J, Tian Y, Schuler A, Ryan PB, Madigan D, Hripcsak G (2020). "How confident are we about observational findings in healthcare: a benchmark study." *Harvard data science review*, **2**(1).

Schuler A, Jung K, Tibshirani R, Hastie T, Shah N (2017). "Synth-validation: Selecting the best causal inference method for a given dataset." *arXiv preprint arXiv:1711.00083*.

Shimoni Y, Yanover C, Karavani E, Goldschmnidt Y (2018). "Benchmarking framework for performance-evaluation of causal inference analysis." *arXiv preprint arXiv:1802.05046*.

Stuart EA (2010). "Matching methods for causal inference: A review and a look forward." *Statistical science: a review journal of the Institute of Mathematical Statistics*, **25**(1), 1. doi:10.1214/09-STS313.

Swaminathan A, Joachims T (2015). "The self-normalized estimator for counterfactual learning." *advances in neural information processing systems*, **28**.

Tikka S, Hyttinen A, Karvanen J (2019). "Causal effect identification from multiple incomplete data sources: A general search-based approach." *arXiv preprint arXiv:1902.01073*.

Tikka S, Karvanen J (2017). "Identifying Causal Effects with the R Package causaleffect." *Journal of Statistical Software*, **76**(12), 1–30. doi:10.18637/jss.v076.i12. URL https://www.jstatsoft.org/index.php/jss/article/view/v076i12.

Tikka S, Karvanen J (2018). "Identifying causal effects with the R package causaleffect." *arXiv preprint arXiv:1806.07161*.

Wager S, Athey S (2018). "Estimation and inference of heterogeneous treatment effects using random forests." *Journal of the American Statistical Association*, **113**(523), 1228–1242.

Wang SV, Sreedhara SK, Bessette LG, Schneeweiss S (2022). "Understanding variation in the results of real-world evidence studies that address the same question." *Journal of Clinical Epidemiology*.

Wendling T, Jung K, Callahan A, Schuler A, Shah NH, Gallego B (2018). "Comparing methods for estimation of heterogeneous treatment effects using observational data from health care databases." *Statistics in medicine*, **37**(23), 3309–3324.

Xie Y, Brand JE, Jann B (2012). "Estimating heterogeneous treatment effects with observational data." *Sociological methodology*, **42**(1), 314–347.

Yao L, Li S, Li Y, Huai M, Gao J, Zhang A (2018). "Representation learning for treatment effect estimation from observational data." *Advances in Neural Information Processing Systems*, **31**.

Zeng S, Li F, Wang R, Li F (2021). "Propensity score weighting for covariate adjustment in randomized clinical trials." *Statistics in medicine*, **40**(4), 842–858.

A. Notations in the paper

Table 5: list of notations

notation	description
$\boldsymbol{X}$	random vector of length d of covariates, containing all pre-treatment outcome predictors
$\boldsymbol{X}_t \subseteq \boldsymbol{X}$	random vector of length q, indicating confounders
$\boldsymbol{X}_s \subseteq \boldsymbol{X}$	random vector of length p, indicating effect modifiers that are predictive of the sampling
T	treatment indicator ($T = 1$ for treatment, $T = 0$ for control)
Y	outcome of interest ($Y = 1$ for survival, $Z = 0$ for death)
S	sampling indicator ($S = 1$ for a RCT, $S = 0$ for an observational study)
$\mathcal{S}^{rct}$	$\mathcal{S}^{rct} = \{(\boldsymbol{X}_i, Y_i, T_i); S_i = 1\}$, a RCT dataset
$\mathcal{S}^{obs}$	$\mathcal{S}^{obs} = \{(\boldsymbol{X}_i, Y_i, T_i); S_i = 0\}$, an observational dataset
$\mathcal{P}_\theta$	the target population that $\mathcal{S}^{rct}$ represents for
$\pi_t(\boldsymbol{x})$	propensity score of a unit with characteristics $\boldsymbol{X} = \boldsymbol{x}$ being selected to treatment $T = 1$
$\pi_s(\boldsymbol{x})$	probability of an individual with the characteristics $\boldsymbol{X} = \boldsymbol{x}$ being selected to an RCT $S = 1$
τ	the average treatment effect of the target population $\mathcal{P}_{\boldsymbol{\theta}}$
$\tau(\boldsymbol{x})$	the conditional average treatment effect, denoted as $\tau(\boldsymbol{x}) = \mathbb{E}[Y(1) - Y(0) \mid \boldsymbol{X} = \boldsymbol{x}]$
σ_1^2, σ_0^2	variance of potential outcomes $Y(1)$, $Y(0)$
$\sigma_t^2(\boldsymbol{x})$	conditional variance of $Y(t)$, denoted as $\mathbb{V}(Y(t) \mid \boldsymbol{x})$
$p(\boldsymbol{x}_s), q(\boldsymbol{x}_s)$	density of covariates $\boldsymbol{X}_s$ in $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$
$w(\boldsymbol{x}_s)$	the density ratio of covariates $\boldsymbol{x}_s$ defined as $\frac{p(\boldsymbol{x}_s)}{q(\boldsymbol{x}_s)}$
$\pi_t(\boldsymbol{X}; \hat{\boldsymbol{\alpha}})$	an estimator for the propensity score
$\pi_s(\boldsymbol{X}; \hat{\boldsymbol{\gamma}})$	an estimator for the selection probability
$p(\boldsymbol{X}, t; \hat{\boldsymbol{\beta}})$	a G_computation estimator for the conditional expected potential outcome $\mathbb{E}[Y(t) \mid \boldsymbol{x}]$ parameterized by $\hat{\boldsymbol{\beta}}$
$\hat{\tau}(\boldsymbol{X})$	an estimator for the conditional average treatment effect $\tau(\boldsymbol{x})$
$\hat{\sigma}_1, \hat{\sigma}_0$	estimator for variance of $Y(1), Y(0)$
$\hat{p}(\boldsymbol{X}), \hat{q}(\boldsymbol{X})$	estimator for density of $\boldsymbol{x}$ in $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$
ϵ_i^z	residual of estimator $p_t(\boldsymbol{X}, t_i; \hat{\boldsymbol{\beta}})$, defined as $\epsilon_i = Y_i - p(\boldsymbol{X}_i, t_i; \hat{\boldsymbol{\beta}})$
$\hat{\sigma}_t^2(\boldsymbol{x})$	an estimator of conditional variance of $Y(t)$, denoted as $\hat{\mathbb{V}}(Y(t) \mid \boldsymbol{x})$

B. Core classes

The current section offers additional background information on **RCTrep**'s classes structures - both on R6 class system Chang and on each of the three previously introduced core **RCTrep** classes. Together with the information in the next section, on `TEstimator` and `SEstimator` implementation, this should be able to get you up and running with developing your own custom `TEstimator` and `SEstimator` subclasses.

B.1. Choice for the R6 class system

Though widely used as a procedural language, R offers several Object Oriented (OO) systems, which can significantly help in structuring the development of more complex packages. Out of the OO systems available (S3, S4, R5, and R6), we settled on R6, as it offers several advantages over other options. Firstly, it implements a mature object-oriented design compared to S3 and S4, hence is easier for developers with a background in programming languages such as C++ and Java to maintain. Secondly, when compared to the older R5 reference class systems, R6 classes are much lighter-weight, as they do not use S4 classes, do not require the **methods** package.

B.2. Core classes: TEstimator, SEstimator, Fusion

In this section, we go over the three core classes on more detail - with an emphasis on the `TEstimator` and `SEstimator` classes. We illustrate the structure of classes, and enumerate core public functions of each classes.

TEstimator

The `TEstimator` class is responsible for fitting a model and estimating treatment effect. The following skeleton code gives an overview of how the above is implemented in **RCTrep**'s `TEstimator` class:

```
TEstimator <- R6::R6Class(
  "TEstimator",
  #-------------------------public fields-----------------------------#
  public = list(
    id = NA,
    name = character(),
    statistics = list(n=numeric(),
                      density_confounders=data.frame()),
    data = NULL,
    estimates = list(ATE=data.frame(y1.hat=NA,
                                    y0.hat=NA,
                                    est=NA,
                                    se=NA),
                     CATE = data.frame()),
    model = list(),
    #-------------------------constructor-----------------------------#
    initialize = function(df, vars_name, name) {
      self$name <- name
      self$data <- df
      self$data$id <- seq(dim(df)[1])
      private$confounders_treatment_name <-
      vars_name$confounders_treatment_name
      private$treatment_name <- vars_name$treatment_name
      private$outcome_name <- vars_name$outcome_name
      self$statistics <- list(n=dim(df)[1],
                              density_confounders=
                              private$est_joint_denstiy())
    },
    get_CATE = function(stratification, stratification_joint=TRUE) {},
    plot_CATE = function(stratification = private$confounders_treatment_name,
                         stratification_joint = TRUE) {},
    plot_y1_y0 = function(stratification, stratification_joint = TRUE,
                          seperate = FALSE){},
    diagnosis_t_overlap = function(stratification,
                                   stratification_joint=TRUE){},
    diagnosis_t_ignorability = function(){},
    diagnosis_y_overlap = function(stratification,
                                   stratification_joint=TRUE){}
  ),
  #-------------------------private fields and methods--------------------#
  private = list(
    confounders_treatment_name = NA,
    treatment_name = NA,
    outcome_name = NA,
    var_method = "sandwitch",
    isTrial = FALSE,

    set_ATE = function(){},
    set_CATE = function(stratification, stratification_joint){},
    est_joint_denstiy = function(){},
    est_CATEestimation4JointStratification = function(stratification) {},
    est_CATEestimation4SeperateStratification = function(stratification) {},
    fit = function(){},
    est_ATE_SE = function(){},
    est_weighted_ATE_SE = function(){}
  )
)
```

Subclasses of `TEstimator` have their unique implementation of `diagnosis_t_ignorability()`, `fit()`, `est_ATE_SE()`, and `est_weighted_ATE_SE()`, and their unique private methods. The main `TEstimator` functions are:

1. `get_CATE(stratification, stratification_joint=TRUE)`

(a) `stratification`: a character vector of length $k \leqslant d$ specifies variables to select subgroups.

(b) `stratification_joint`: logical to indicate if subgroups are selected based on levels of each variable in `stratification` or joint levels of all k variables in `stratification`.

The function returns a `data.frame` containing treatment effects estimation of selected subgroups. If `stratification=TRUE`, then the function returns a `data.frame` with column names `c(stratification,"y1.hat","y0.hat","cate","se","size")`; if `stratification_joint=FALSE`, then the function returns a `data.frame` with column names `c("name", "value","y1.hat","y0.hat","cate","se","size")`.

2. `diagnosis_t_overlap(stratification, stratification_joint)`: plot the proportion and count of individuals receiving treatment and control in each subgroup. Subgroups are defined by `stratification` and `stratification_joint`.

3. `diagnosis_t_ignorability()`: the function diagnoses T-ignorability assumptions. For subclass `G_computation`, the function summarizes outcome model fit using evaluation metrics, i.e., means of residuals of subgroups, distribution of overall residuals, mean squared errors of subgroups for a continuous outcome, and mean of deviance of subgroups for a binary outcome. For subclass `IPW`, the function plot the weighted distribution of subgroups between treatment and control groups. For `DR`, the function summarizes both outcome model fit and weighted distribution of subgroups between groups.

4. `diagnosis_y_overlap(stratification, stratification_joint)`: plot the count of each level of outcome in treatment and control groups in each subgroup defined by `stratification` and `stratification_joint`. For binary outcomes, the function plots the count of the positive outcome and the negative outcome; for continuous outcomes, the function plots the distribution of outcomes.

5. private method `set_ATE()`: the function implements the private method `est_ATE_SE(id)`, and gets the point estimate of average treatment effect, standard error of the estimate, mean of potential outcomes; the function assigns these estimates to the public fields `estimates$ATE$est`,`estimates$ATE$se`,`estimates$ATE$y1.hat`,`estimates$ATE$y0.h at` accordingly. The function is implemented in the initialize function of each `TEstimator` subclass.

6. private method `set_CATE(stratification, stratification_joint)`: the function implements the public method `get_CATE(stratification, stratification_joint)` which returns a `data.frame` (see below for details of the returned object from the function `get_CATE()`); then the function `set_CATE()` assigns the returned estimates from `get_CATE()` to the public field `estimates$CATE`. The function is implemented in the initialize function of each subclass of `TEstimator` by calling `private$set_CATE(private$confounders_treatment_name,TRUE)`.

7. private method `est_CATEestimation4JointStratification(stratification)`: the function selects subgroups defined by joint levels of all variables specified in `stratification`, gets the index of selected data, and estimates the average treatment effect of each

subgroup by calling the private method `est_ATE_SE(index)`. The function returns a `data.frame` with column name `c(stratification, "y1.hat", "y0.hat", "cate", "se", "size")`.

8. private method `est_CATEestimation4SeperateStratification(stratification)`: the function selects subgroups defined by levels of each variable specified in `stratification`, gets the index of selected data, and estimates the average treatment effect of each subgroup by calling the private method `est_ATE_SE(index)`. The function returns a `data.frame` with column name `c("name", "value", "y1.hat", "y0.hat", "cate", "se", "size")`.

9. private method `est_ATE_SE(index)`: the function estimates the average treatment effect and its standard error. `index` indicates the index of data. Different subclass has unique implementation of point estimation. **RCTrep** implements sandwich estimator to estimate standard error using package **geex** (Saul and Hudgens 2020). We need to specify an estimation function `estFUN`, and pass the function to `geex::m_estimate(data, estFUN, ...)`. `m_estimate` provides a consistent estimator for the asymptotic variance of the estimate of average treatment effect. **RCTrep** does not take the uncertainty of estimation of parameters of models into account in order to speed up implementation, however, users can customize `estFUN` so the function can take account of the uncertainty of estimation of parameters into the estimation of the variance of average treatment effect. For more details, see simulation codes in (Dahabreh *et al.* 2020) and tutorials by Saul and Hudgens (2020). `est_ATE_SE(index)` function returns a `list` with named elements `y1.hat`, `y0.hat`, `est`, and `se`. An overview of estimators of variance of average treatment effect is provided in appendix C.

10. private method `est_weighted_ATE_SE(index, weight)`: the function estimates the weighted average treatment effect and its standard error. The function selects estimates of potential outcomes from `self$data[index,]$y1.hat` and `self$data[index,]$y0.hat`, and assigns weights for selected data. We implement sandwich estimator using R package **geex** to estimate the standard error of the weighted average treatment effect. The function returns a `list` with named elements `y1.hat`, `y0.hat`, `est`, and `se`.

11. private method `est_CATEestimation4JointStratification(stratification)`: the function estimates treatment effect of subgroups. The function selects a subgroup based on joint levels of variables in `stratification`, gets `id` of the selected subgroup, and computes the average treatment effect of the subgroup by calling `private_ATE_SE(id)`. Loop this procedure until all subgroups have been selected. The function returns a `data.frame` with column names `c(stratification, "y1.hat", "y0.hat", "cate", "se", "size")`.

12. private method `est_CATEestimation4SeperateStratification(stratification)`: the function estimates treatment effect of subgroups. The function selects a subgroup based on levels of each variable in `stratification`, gets `id` of the selected subgroup, and computes the average treatment effect of the subgroup by calling `private$est_ATE_SE(id)`. Loop this procedure until all subgroups have been selected. The function returns a `data.frame` with column names `c("name", "value", "y1.hat", "y0.hat", "cate", "se", "size")`.

SEstimator

The `SEstimator` class is responsible for balancing covariates in `confounders_sampling_name` between two objects of class `TEstimator`, and estimates the weighted average and conditional average treatment effect. The following skeleton code gives an overview of how weighted estimation is implemented in `RCTrep`'s `SEstimator` classes:

```
SEstimator <- R6::R6Class(
  "SEstimator",
  #------------------------public fields-----------------------------#
  public = list(
    name = character(),
    id = character(),
    statistics = list(),
    estimates = list(ATE = data.frame(y1.hat=NA,
                                      y0.hat=NA,
                                      est=NA,
                                      se=NA),
                     CATE = data.frame()),
    model = NA,
    confounders_sampling_name = NA,
    weighting_method = character(),

    initialize = function(target.obj, source.obj, weighting_method=NULL,
                          confounders_sampling_name){
      private$target.obj <- target.obj
      private$source.obj <- source.obj
      self$weighting_method <- weighting_method
      self$confounders_sampling_name <- confounders_sampling_name
      private$ispublic <- !c("TEstimator_pp") %in% class(source.obj)
      self$name <- source.obj$name
      self$statistics <- source.obj$statistics
      self$id <- paste(private$source.obj$id,
                       self$weighting_estimator,
                       length(self$confounders_sampling_name),sep = '/')
      private$isTrial <- source.obj$.__enclos_env__$private$isTrial
    },
    EstimateRep = function(stratification=self$confounders_sampling_name,
                           stratification_joint=TRUE) {},
    diagnosis_s_overlap = function(stratification=NULL,
                                   stratification_joint=TRUE){},
    diagnosis_s_ignorability = function(stratification=NULL,
                                        stratification_joint=TRUE){}
  ),

  private = list(
    source.obj = NA,
    target.obj = NA,
    ispublic = NA,
    isTrial = NA,

    get_weight = function(){source.data,target.data, vars_weighting},
    set_weighted_ATE_SE = function() {},
    set_weighted_CATE_SE = function(stratification, stratification_joint) {},
    est_WeightedCATEestimation4JointStratification =
    function(stratification) {},
    est_WeightedCATEestimation4SeperateStratification =
    function(stratification) {},
    est_statistics = function(){}
  )
)
```

The following are public and private functions in `SEstimator`:

1. public function `EstimateRep(stratification, stratification_joint)`: the core function which estimates weighted average treatment effect and weighted conditional average treatment effect; `stratification` and `stratification_joint` specify a criteria to select subgroups, which is the same as `get_CATE()` in `TEstimator`.

2. `diagnosis_s_overlap(stratification=NULL, stratification_joint=TRUE)`: the function selects subgroups according to `stratification, stratification_joint`; the function plots the percentage and numbers of observations from `source.obj` and `target.obj` for each subgroup. The default value of `stratification` is `confounders_sampling_name`.

3. `diagnosis_s_ignorability(stratification=NULL, stratification_joint=TRUE)`: the function diagnoses the assumption of S-ignorability. The function selects subgroups according to `stratification, stratification_joint`. It computes the weighted distribution of the subgroups in `source.obj` and the distribution of the subgroups in `target.obj`.

4. private method `get_weight(source.data, target.data, vars_weighting)`: the function estimates weights for each individual in `source.obj`. The weights are computed based on specified variables `vars_weighting`. Each subclass of `SEstimator` has a unique implementation of the function:

   - `SEexact`: The implementation of weight computation depends on R package **MatchIt**.
   - `SEisw`: Weighting based on inverse selection probability. Methods for estimating the selection probability is specified in `self$weighting_method` argument. Allowable options of `weighting_method` are inherent from `method` in R package **caret**.
   - `SEsubclass`: Weighting based on sub-classification on the selection probability of data in `target.obj`. Methods for estimating the selection score is specified in `self$weight ing_method` argument. The default is `glm` for selection probability estimated with the logistic regression using covariates in `confounder_sampling_name`. $\mathcal{S}^{obs}$ in `source.obj` and $\mathcal{S}^{rct}$ in `target.obj` are placed into sub-classes based on quantiles of the selection probability of $\mathcal{S}^{rct}$. Then weights for $\mathcal{S}^{obs}$ are computed based on the proportion of individuals from $\mathcal{S}^{rct}$ in each subclass.

- `SEstimator_pp`: weighting for two objects of class `TEstimator_pp`. Weight is computed as $w(\boldsymbol{x}_{si}) = \frac{w'(\boldsymbol{x}_{si})}{\sum_{i \in \mathcal{S}^{obs}} w'(\boldsymbol{x}_{si})}$, $w'(\boldsymbol{x}_{si}) = \frac{\hat{p}(\boldsymbol{x}_s)}{\hat{q}(\boldsymbol{x}_s)}$

5. private method `set_weighted_ATE_SE`: the function estimate the weighted average treatment effect of `source.obj`. The function calls `private$get_weight(source.data=private$source.obj$data, targe.data=private$target.obj$data, vars_weighting=self$confounders_sampling_name)` to compute weights, then calls the private method `est_weighted_ATE_SE()` of `source.obj` to estimate weighted average treatment effect and gets the weighted estimates of `y1.hat`, `y0.hat`, `est`, and `se` accordingly, and finally assigns these estimates to `self$estimates$ATE$y1.hat`, `self$estimates$ATE$y0.hat`, `self$estimates$ATE$est`, `self$estimates$ATE$se`.

6. private method `set_weighted_CATE_SE(stratification, stratification_joint)`: the function estimates weighted conditional average treatment effect; if `stratification_joint=TRUE`, then the function calls `private$est_WeightedCATEestimation4JointStratification(stratification)`; if `stratification_joint=FALSE`, then the function calls `private$est_WeightedCATEestimation4SeperateStratification(stratification)`. `Stratification` is a character vector that specifies variables for subgroup selection.

7. private method `est_WeightedCATEestimation4JointStratification(stratification)`: the function estimates weighted conditional average treatment effect. The function selects subgroups from `private$source.obj$data` and `private$target.obj$data`, and calls `private$get_weight()` to compute weights of each individual in `source.obj` so that variables in `self$confounders_sampling_name` are balanced between weighted `source.obj` and `target.obj`. We limit `self$confounders_sampling_name` and `stratification` to have no overlap. The function returns a `data.frame` in the same form as that returned from the private method `est_CATEestimation4JointStratification(stratification)` of the class `TEstimator`.

8. private method `est_WeightedCATEestimation4SeperateStratification(stratification):` the same as the `est_WeightedCATEestimation4JointStratification(stratification)` except for the criteria to select subgroups. The function returns a `data.frame` in the same form as that returned from the private method `est_CATEestimation4SeperateStratification(stratification)` of the class `TEstimator`.

Fusion

The `Fusion` class is responsible for aggregating estimates from objects of classes `TEstimator` and `SEstimator`, evaluating methods for treatment effect estimation implemented in class `TEstimator`, plotting and printing results. The following skeleton code gives an overview of class `Fusion`:

```
Fusion <- R6::R6Class(
  "Fusion",
  #-------------------------public fields-----------------------------#
  public = list(
    objs.cate.data = data.frame(),
    objs.ate.data = data.frame(),
    stratification = NA,
    stratification_joint = NA,
    RCT.study.name = NA,
    RWD.study.name = NA,

    initialize = function(...){},
    plot = function(){},
    print = function(){},
    evaluate = function(){}
  ),

  private = list(
    aggregate_cate_estimates = function(...){},
    aggregate_ate_estimates = function(...){}
  )
)
```

The following are public and private methods in `Fusion`:

1. constructor `initialize(...)` initializes an object of `Fusion`; passes objects of class `TEstimator` and `SEstimator` to the argument `...`. The number of objects passed to the function is not limited.

2. public function `plot()`, `print()` plots and prints average and conditional average treatment effect estimation using observational data.

3. public function `evaluate()`: the function computes $\mathbb{L}$ using the following metrics:

    - pseudo mse `mse`;
    - length of the confidence interval `length_ci`;
    - estimate agreement `agg.est`;
    - and regulatory agreement `agg.reg`.

    The regulatory agreement is defined as the consistency of the direction and statistical significance of estimates from two data sets, and estimate agreement indicates whether an estimate using observational data lies within the 95% confidence interval of the estimate using RCT data (Franklin *et al.* 2020). The function computes the evaluation metrics on population and sub-population levels. Sub-populations are selected according to `self$stratification` and `self$stratification _joint`, which are inherent from arguments passed to the function `EstimateRep()` of an object of class `SEstimator` that is passed to the initialize function of the class `Fusion`.

4. private method `aggregate_ate_estimates` and private method `aggregate_cate_estimates`: the functions aggregate estimates of average treatment effect and conditional average treatment effect from all objects passed to `...`.

B.3. Subclasses of TEstimator and SEstimator

Subclasses of `TEstimator` are mainly responsible for fitting models and estimating treatment effects using their unique methods `est_ATE_SE`. We can override `est_ATE_SE` for a new subclass of `TEstimator`. Subclasses of `SEstimator` are responsible for estimating weights $w(\boldsymbol{x}_s)$ using their unique methods `get_weight`. We can override the function for a new subclass of `SEstimator`.

Since the aim of data sharing is to compute weights to balance $\boldsymbol{X}_s$ in two objects, it is not necessary to have full datasets. For instance, each object only needs to share density of $\boldsymbol{X}_s$, estimates $\hat{\tau}(\boldsymbol{X}_s)$, standard error of $\hat{\tau}(\boldsymbol{X}_s)$, and sample size for each subgroup stratified by $\boldsymbol{X}_s$, to estimate the weighted treatment effect. Hence, in case full data is not allowed to share, we define a subclass `TEstimator_pp` for `TEstimator` and `SEstimator_pp` for `SEstimator`. In `TEstimator_pp`, instead of assigning a full dataset to the public field `data`, we assign the density of covariates in `confounders_treatment_name` and the estimates of the treatment effect of subgroups stratified by `confounders_treatment_name` to the public field `data` of an object of class `TEstimator_pp`. Two objects are passed to an object of class `SEstimator`, and communicate `data` with each other within the object. The object computes weights $w(\boldsymbol{X}_s)$ for `source.obj` accordingly. For different weighting approaches, we can share different aggregated data. For instance, weighting using balanced-based methods only requires $p(B(\boldsymbol{x}_s))$ (Chatton *et al.* 2020), where $B(\boldsymbol{x}_s)$ is the basis function of $\boldsymbol{x}_s$, e.g., interaction between two random variables. Hence, in this case, we only need to share the density of basis function $B(\boldsymbol{x}_s)$, and override public field `data` in a new subclass of `TEstimator`, and override `get_weight()` in a new subclass of `SEstimator` accordingly.

C. Variance of estimators for the average treatment effect

In **RCTrep**, we use three estimators for conditional average treatment effect, namely, G-computation, IPW, and doubly robust methods. G-computation method is unbiased and consistent as long as a model for the outcome (i.e., $p(\boldsymbol{x}, t; \hat{\boldsymbol{\beta}})$) is correctly specified. IPW is unbiased as long as a model for treatment, i.e., propensity score $\pi_t(\boldsymbol{X}; \hat{\boldsymbol{\alpha}})$, is correctly specified. Doubly robust method is unbiased as long as either a model for the outcome or a model for the treatment is correctly specified, and is more efficient than the IPW method. Note that we show the variance of three estimators for illustrative purposes regarding the effect of weight on the variance estimation (i.e., IPW estimator), model assumptions on the variance estimation (i.e., G-computation), and sample size on the variance estimation. In the following, we analyze the variance of these estimators.

C.1. Variance of G-computation

Assumptions of T-ignorability imply that conditioning on confounders treatment assignment can be assumed random and hence the treatment effect can be identified as a simple difference in means between two groups for each subgroup stratified by confounders. The average treatment effect using G-computation method is defined as:

$$\hat{\tau} = \mathbb{E}[\hat{\tau}(\boldsymbol{X})] = \mathbb{E}[p(\boldsymbol{X}, 1; \hat{\boldsymbol{\beta}}) - p(\boldsymbol{X}, 0; \hat{\boldsymbol{\beta}})] \approx \frac{1}{n} \sum_i p(\boldsymbol{x}_i, 1; \hat{\boldsymbol{\beta}}) - p(\boldsymbol{x}_i, 0; \hat{\boldsymbol{\beta}}) \tag{4}$$

where $\boldsymbol{X}$ is a random vector of all pre-treatment outcome predictors containing all confounders, $p(\boldsymbol{X}, T; \hat{\boldsymbol{\beta}}) = \hat{\mathbb{E}}[Y \mid \boldsymbol{X}, T]$. We can use both parametric and non-parametric models to estimate expected value of potential outcomes given the value of $\boldsymbol{x}$, in other words, $\hat{\boldsymbol{\beta}} \subset \mathbb{R}^{\mathbb{R}}$. Here we use $\boldsymbol{\beta}$ to denote a set of parameters that describe the distribution of conditional potential outcomes. We assume the conditional expectation is expressed as an equation linear in $\boldsymbol{x}$ and t, and hence can be described by a fixed length of parameters $\boldsymbol{\beta}$. We can also assume that conditional expectation can be described by a flexible function parameterized $\boldsymbol{\beta}$ of flexible length depending on a model constraint, regularization, and sample size. $p(\boldsymbol{x}, 1; \hat{\boldsymbol{\beta}})$ is the estimate of $\mathbb{E}[Y(1) \mid \boldsymbol{x}, T = 1]$ parameterized by $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\beta}}$ is estimated using $\mathcal{S}^{obs}$. Then the variance of the estimator $\hat{\tau}(\boldsymbol{X})$ is derived as:

$$\begin{aligned}
\mathbb{V}(\hat{\tau}(\boldsymbol{X})) &= \mathbb{E}[\mathbb{V}(\hat{\tau}(\boldsymbol{X}) \mid \boldsymbol{X})] + \mathbb{V}\left(\mathbb{E}[\hat{\tau}(\boldsymbol{X}) \mid \boldsymbol{X}]\right) \quad \text{law of total variance} \\
&= \mathbb{E}\left[\mathbb{V}\left(p(\boldsymbol{X}, 1; \hat{\boldsymbol{\beta}}) - p(\boldsymbol{X}, 0; \hat{\boldsymbol{\beta}}) \mid \boldsymbol{X}\right)\right] + \mathbb{V}\left(\mathbb{E}[p(\boldsymbol{X}, 1; \hat{\boldsymbol{\beta}}) - p(\boldsymbol{X}, 0; \hat{\boldsymbol{\beta}}) \mid \boldsymbol{X}]\right) \\
&\approx \mathbb{E}\left[\mathbb{V}\left(p(\boldsymbol{X}, 1; \hat{\boldsymbol{\beta}}) \mid \boldsymbol{X}\right) + \mathbb{V}\left(p(\boldsymbol{X}, 0; \hat{\boldsymbol{\beta}}) \mid \boldsymbol{X}\right)\right] + \mathbb{V}\left(p(\boldsymbol{X}, 1; \bar{\hat{\boldsymbol{\beta}}}) - p(\boldsymbol{X}, 0; \bar{\hat{\boldsymbol{\beta}}})\right) \\
&\approx \frac{1}{n} \sum_i \left(\hat{\mathbb{V}}(p(\boldsymbol{x}_i, 1; \hat{\boldsymbol{\beta}})) + \hat{\mathbb{V}}(p(\boldsymbol{x}_i, 0; \hat{\boldsymbol{\beta}}))\right) + \hat{\mathbb{V}}(p(\boldsymbol{X}, 1; \bar{\hat{\boldsymbol{\beta}}}) - p(\boldsymbol{X}, 0; \bar{\hat{\boldsymbol{\beta}}}))
\end{aligned} \tag{5}$$

Note in the third line, the first term is the function of $\boldsymbol{X}$ and variance of $\hat{\boldsymbol{\beta}}$ depending on the sample, and hence the variance of this term depends on the sample. In logistic regression, the variance of $\hat{\boldsymbol{\beta}}$ is well-developed and estimation is unbiased when the model is correctly specified, and most of software can provide the estimate of the variance of these parameters. In non-parametric methods, it is not trivial to write down the closed form of variance of parameters, alternative approaches to estimating $\mathbb{V}(p(\boldsymbol{x}_i, t_i; \hat{\boldsymbol{\beta}}))$ are delta method, bootstrap, etc. We introduce approaches for estimating the variance of $p(\boldsymbol{x}_i, t_i; \hat{\boldsymbol{\beta}})$ in the next section. The second term in the third line is the variance between groups $\mathbb{V}(\hat{\tau}(\boldsymbol{X}; \bar{\hat{\boldsymbol{\beta}}}))$, and only $\boldsymbol{X}$ is random, hence the variance of the second term can be estimated using sample variance of estimated $\hat{\tau}(\boldsymbol{X}; \bar{\hat{\beta}})$ where $\bar{\hat{\boldsymbol{\beta}}} = \mathbb{E}[\hat{\beta}]$, which is the true value of $\boldsymbol{\beta}$ by OLS. We use an estimate of $\hat{\boldsymbol{\beta}}$ based on a sample as an estimate of $\bar{\hat{\boldsymbol{\beta}}}$, and estimate the sample variance of plugged in $\hat{\tau}(\boldsymbol{X}; \bar{\hat{\boldsymbol{\beta}}})$.

Methods for estimating the variance of G-computation

In this section, we illustrate five methods for estimating the variance of G-Computation estimator, i.e., $\mathbb{V}(p(\boldsymbol{x}, t; \hat{\boldsymbol{\beta}}))$. In the following, for demonstrative purposes, we use logistic regression to estimate probabilities of outcomes if an individual receives treatment or control. $p(\boldsymbol{x}, t; \hat{\boldsymbol{\beta}}) = \sigma(\boldsymbol{x}, t; \hat{\boldsymbol{\beta}}) = \frac{1}{1+exp^{-(\boldsymbol{x},t)'\hat{\beta}}}$, where $\mathbb{V}(p(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}}))$ can be estimated by the following five methods:

1. Model-based method, where $\mathbb{V}(\boldsymbol{\beta}) = \mathbf{I}^{-1}(\boldsymbol{\beta})$, $\mathbf{I}(\boldsymbol{\beta})$ is the observed information matrix. $\mathbb{V}(\boldsymbol{\beta})$ can be estimated at $\hat{\boldsymbol{\beta}}$, denoted as $\hat{\mathbb{V}}(\hat{\boldsymbol{\beta}}) = \left(\mathbf{X}'\hat{\mathbf{V}}\mathbf{X}\right)^{-1}$, where

$$\hat{\mathbf{V}} = \begin{bmatrix} \hat{p}_1\,(1-\hat{p}_1) & 0 & \cdots & 0 \\ 0 & \hat{p}_2\,(1-\hat{p}_2) & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \hat{p}_n\,(1-\hat{p}_n) \end{bmatrix},$$

   $\hat{p}_i$ is the predicted observed outcome, then

$$\hat{\mathbb{V}}(p(\boldsymbol{x}_i, z; \hat{\boldsymbol{\beta}})) = \mathbf{x}_i'\hat{\mathbb{V}}(\hat{\boldsymbol{\beta}})\mathbf{x}_i = \sum_j x_{ij}^2 \hat{\mathbb{V}}(\hat{\beta}_j) + 2\sum_{j=0}^{p}\sum_{k=j+1}^{p} x_{ij}x_{ik}\widehat{\mathrm{Cov}}(\hat{\beta}_j, \hat{\beta}_k). \tag{6}$$

   where we regard $T_i = z$ as an element in the vector $\mathbf{x}_i$, i.e., $\mathbf{x}_i = (\boldsymbol{x}_i, t)'$, $\hat{\mathbb{V}}(\hat{\boldsymbol{\beta}}_j)$ is the jth diagonal element of the matrix $\hat{\mathbb{V}}(\hat{\boldsymbol{\beta}})$, and $\widehat{\mathrm{Cov}}(\hat{\boldsymbol{\beta}}_j, \hat{\boldsymbol{\beta}}_k)$ is an off-diagonal element in the matrix. Then we can estimate $\hat{\mathbb{V}}(p(\boldsymbol{x}, 1; \hat{\beta}))$ and $\hat{\mathbb{V}}(p(\boldsymbol{x}, 0; \hat{\beta}))$ for each individual i. We estimate the sample average of $\hat{\mathbb{V}}(p(\boldsymbol{x}, t; \hat{\beta}))$ as the estimate of expectation of the variance within groups, i.e., the first term in the last line of variance decomposition in equation 5. For the variance between groups, i.e., the second term in the equation, we estimate the sample variance of $\hat{\tau}(\boldsymbol{X}; \bar{\hat{\boldsymbol{\beta}}})$ at $\hat{\boldsymbol{\beta}}$. For more computation details, see chapter 2.5 in (Hosmer Jr, Lemeshow, and Sturdivant 2013). Note that for a continuous outcome, linear regression assumes that the variance of the error term does not depend on the conditional mean. We can use heteroskedasticity-consistent standard errors in case the assumption does not hold. However, in logistic regression, we have binomial errors, and as a result, the error variance is a function of the conditional mean thereof is heterogeneous by nature (Hosmer Jr *et al.* 2013).

2. Simulation approach, where $\hat{\beta} \sim \mathcal{N}(\hat{\boldsymbol{\beta}}, \hat{\mathbb{V}}(\hat{\boldsymbol{\beta}}))$, the method used by (Chatton *et al.* 2020; Aalen, Farewell, De Angelis, Day, and Nöel Gill 1997) which shows similar results to bootstrap resampling but is much faster. We can simulate a set of parametric models

from the distribution of $\hat{\boldsymbol{\beta}}$, in which expectation and variance of are estimated using OLS, then the sample variance of predicted potential outcomes for each $\boldsymbol{x}_i$ from a set of simulated models is the estimated variance for $\mathbb{V}(p(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}}))$.

3. Bayesian approach. We use bayesian logistic regression to estimate potential outcomes. Via the bayesian approach, each parameter in a model is regarded as a random variable and follows a distribution. Posterior distribution of model parameters is approximated using a sampling approach, e.g., MCMC, and hence the resulting predicted value of potential outcomes for each individual also follows a similar distribution and the variance of the distribution can be estimated using sample variance, namely, $\mathbb{V}(p(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}})) \approx \mathbb{S}(p(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}}))$, where $\hat{\boldsymbol{\beta}} \sim p(\hat{\boldsymbol{\beta}}; \mathcal{D})$. In `RCTrep`, `G_computation_BART` and `G_computation_psBART` use the Bayesian approach to estimating the variance of conditional average treatment effect.

4. Bootstrap. Instead of re-sampling model parameters in the second method, we can bootstrap a sample from a dataset, estimate $\hat{\boldsymbol{\beta}}$ based on the resampled data, repeat multiple times, and compute the sample variance of predicted potential outcomes for each individual as the estimation of the variance of $p(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}})$. This method, however, is of computational burden.

5. Sandwitch style estimator of standard error using R package **geex**. The standard error of the average treatment effect can be computed directly by calling the function `geex::m_estimate(data, estFUN, ...)`. See Saul and Hudgens (2020) for more theoretical proof and implementation details. All `TEstimator` subclasses in **RCTrep** use the estimator to compute the variance of weighted conditional average treatment effect, all subclasses of `TEstimator` use the estimator to compute the variance of conditional average treatment effect except for `G_computation_BART` and `G_computation_psBART`.

The variance of average treatment effect is composed of variance within groups (the first term in the third line of equation 5) and variance between groups (the second term in the third line of equation 5). Via simulation approach, bayesian approach, and bootstrap approach, then the variance of $p(\boldsymbol{x}, t; \hat{\boldsymbol{\beta}})$ within a group $\boldsymbol{X} = \boldsymbol{x}$ can be computed as follows:

$$\hat{\mathbb{V}}(p(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}})) = \frac{1}{D} \sum_{d=1}^{D} \left(p(\boldsymbol{x}_i, z; \hat{\boldsymbol{\beta}}^d) - \bar{p}(\boldsymbol{x}_i, z; \hat{\boldsymbol{\beta}}) \right)^2 \tag{7}$$

where D is the number of draws from the distribution of $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\beta}}^d \sim \hat{p}(\hat{\boldsymbol{\beta}})$, $\bar{p}(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}}) = \frac{1}{D} \sum_{d=1} p(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}}^d)$, where $\hat{p}(\hat{\boldsymbol{\beta}})$ is the approximated empirical sampling distribution of $\hat{\boldsymbol{\beta}}$ using simulation, bayesian, and bootstrap based variance estimation approaches. Then

$$\mathbb{E}[\mathbb{V}\left(\hat{\tau}(\boldsymbol{X}) \mid \boldsymbol{X}\right)] \approx \frac{1}{n} \sum_{i} \hat{\mathbb{V}}(p(\boldsymbol{x}_i, 1; \hat{\boldsymbol{\beta}})) + \hat{\mathbb{V}}(p(\boldsymbol{x}_i, 0; \hat{\boldsymbol{\beta}})) \tag{8}$$

by assuming $p(\boldsymbol{x}, 1; \hat{\boldsymbol{\beta}})$ is independent of $p(\boldsymbol{x}, 0; \hat{\boldsymbol{\beta}})$. Then we estimate sample average of $\hat{\mathbb{V}}(\hat{\tau}(\boldsymbol{x}_i))$ as the estimate of expectation of variance of estimates of treatment effect within groups. The variance of estimates of treatment effect between groups (the second term in the last line of equation 5) can be estimated as follows:

$$\mathbb{V}\left(\mathbb{E}[\hat{\tau}(\boldsymbol{X}) \mid \boldsymbol{X}]\right) \approx \frac{1}{n} \sum_{i=1} \left(p(\boldsymbol{x}_i, 1; \bar{\hat{\boldsymbol{\beta}}}) - p(\boldsymbol{x}_i, 0; \bar{\hat{\boldsymbol{\beta}}}) - \bar{p}(1; \bar{\hat{\boldsymbol{\beta}}}) - \bar{p}(0; \bar{\hat{\boldsymbol{\beta}}}) \right)^2 \tag{9}$$

where $p(\boldsymbol{x}_i, t; \bar{\hat{\boldsymbol{\beta}}}) = \frac{1}{D}\sum_{d=1} p(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}}^d)$ for simulation, bayesian, and bootstrap method, and $p(\boldsymbol{x}_i, t; \bar{\hat{\boldsymbol{\beta}}}) = p(\boldsymbol{x}_i, t; \hat{\boldsymbol{\beta}})$ for model-based method; $\bar{p}(t; \bar{\hat{\boldsymbol{\beta}}}) = \frac{1}{n}\sum_{i=1} p(\boldsymbol{x}_i, t; \bar{\hat{\beta}})$. Then the variance of estimate of average treatment effect in equation 5 for G-computation is sum of estimated variance of estimate of treatment effects within groups in equation 8 and estimated variance of estimate of treatment effects between groups in equation 9. Note that using sandwich style standard error via **geex** can directly estimate the variance of the estimate of average treatment effect without manually computing the equation 8 and 9. Hence, for computational convenience, we use sandwich style standard error for all subclasses of `TEstimator` in **RCTrep** except for `G_computation_BART` and `G_computation_psBART`.

C.2. Variance of IPW

Propensity-score based method for treatment effect estimation has a methodological advantage since it mimics a set-up of an RCT in which the treatment and control groups are balanced. The propensity score is defined as:

$$\pi_t(\boldsymbol{X}) = P(T = 1 \mid \boldsymbol{X}) \tag{10}$$

IPW weighs each individual by inverse probability of receiving the observed treatment. In an RCT, the propensity score is known; in an observational study, the propensity score is unknown but may be estimable. IPW method is defined as follows where we use the self-normalized IPW estimator (i.e., Hajek estimator) since it has smaller variance (Swaminathan and Joachims 2015):

$$\hat{\tau} = \sum_{i:T_i=1} \hat{w}(\boldsymbol{x}_i)Y_i - \sum_{i:T_i=0} \hat{w}(\boldsymbol{x}_i)Y_i \tag{11}$$

where

$$\hat{w}(\boldsymbol{x}_i) = \begin{cases} \frac{\frac{1}{\pi_t(\boldsymbol{x}_i;\hat{\boldsymbol{\alpha}})}}{\sum_{i:T_i=1}\frac{1}{\pi_t(\boldsymbol{x}_i;\hat{\alpha})}} & T_i = 1 \\ \frac{\frac{1}{1-\pi_t(\boldsymbol{x}_i;\hat{\boldsymbol{\alpha}})}}{\sum_{i:T_i=0}\frac{1}{1-\pi_t(\boldsymbol{x}_i;\hat{\alpha})}} & T_i = 0. \end{cases}$$

Different modeling approaches can be used to model the propensity score, for instance, logistic regression, random forest, etc. IPW method is unbiased and consistent as long as the propensity score model is correctly specified. The variance of the IPW method is approximated as:

$$\begin{aligned} \mathbb{V}(\hat{\tau}(\boldsymbol{X})) =& \mathbb{V}\left(\frac{YT}{\pi_t(\boldsymbol{X};\hat{\boldsymbol{\alpha}})} - \frac{Y(1-T)}{1-\pi_t(\boldsymbol{X};\hat{\boldsymbol{\alpha}})}\right) \\ =& \mathbb{E}\left[\mathbb{V}\left(\frac{YT}{\pi_t(\boldsymbol{X};\hat{\boldsymbol{\alpha}})} - \frac{Y(1-T)}{1-\pi_t(\boldsymbol{X};\hat{\boldsymbol{\alpha}})} \mid \boldsymbol{X}\right)\right] + \\ & \mathbb{V}\left(\mathbb{E}\left[\frac{YT}{\pi_t(\boldsymbol{X};\hat{\boldsymbol{\alpha}})} - \frac{Y(1-T)}{1-\pi_t(\boldsymbol{X};\hat{\boldsymbol{\alpha}})} \mid \boldsymbol{X}\right]\right) \\ \approx & \sum_{i:t_i=1}^{n} w_i^2\hat{\sigma}_1^2(\boldsymbol{x}_i) + \sum_{i:t_i=0}^{n} w_i^2\hat{\sigma}_0^2(\boldsymbol{x}_i) + \hat{\mathbb{V}}\left(\hat{\tau}(\boldsymbol{X};\bar{\hat{\alpha}})\right) \end{aligned} \tag{12}$$

where $\sigma_1^2(\boldsymbol{x})$ and $\sigma_0^2(\boldsymbol{x})$ is conditional variance of $Y(1)$ and $Y(0)$ given $\boldsymbol{x}$, which is unknown and maybe estimable using exact matching, and regression adjustment, etc., see Imbens and

Rubin (2015) chapter 19 for details. $\hat{\tau}(\boldsymbol{X};\bar{\hat{\alpha}}) \approx \hat{\tau}(\boldsymbol{X}_i;\hat{\alpha}) = \frac{Y_iT_i}{\pi_t(\boldsymbol{X}_i;\hat{\alpha})} - \frac{Y_i(1-T_i)}{1-\pi_t(\boldsymbol{X}_i;\hat{\alpha})}$, $\hat{\mathbb{V}}(\hat{\tau}(\boldsymbol{X};\bar{\hat{\boldsymbol{\alpha}}}))$ is the sample variance of $\hat{\tau}(\boldsymbol{X};\bar{\hat{\boldsymbol{\alpha}}})$.

RCTrep uses sandwitch style standard error via **geex** to estimate the variance of IPW for average treatment effect estimation. It is clear to see that the variance of IPW depends on the variance of estimated weights, and can inflate the variance if there are extreme values of weights. Hence, the IPW method can suffer from near violation of T-overlap assumption. To have a good estimation of the variance, we should try to keep the dependence of $w(\boldsymbol{x}_i)$ as mild as possible. On one hand, we can reduce the variability of weight using approaches in Dong *et al.* (2020); Chattopadhyay *et al.* (2020); Zeng, Li, Wang, and Li (2021) through optimization, which minimizes the variability of all weights while preserving balance in weighted covariates between groups; on the other hand, to reduce the variability of weights, we can exclude covariates which are merely associated with treatment assignment in propensity score modeling, since balancing over these variables will decrease sample size (degree of freedom) in each subgroup hence can inflate the estimation of variance. Beyond confounders, other variables which are predictive of outcome can be adjusted in propensity score models which can improve precision.

C.3. Variance of DR

DR method combines a propensity score model with an outcome model such that the method is unbiased and consistent if at least one of the two models is correctly specified, hence it offers protection against missmodeling. DR method gains in precision of estimation over IPW method, however, may not be as precise as G-computation method when outcome model is correctly specified (or has good approximation) (Lunceford and Davidian 2004). The study by Kang and Schafer (2007) indicates that when both models are incorrect but neither is grossly misspecified, many DR methods perform better than IPW, however, non of the DR methods tried in the study outperformed an outcome regression model. Although the study does not represent all scenarios of DGM, the study does demonstrate that, in at least some settings, two wrong models may not be better than one. The DR method for ATE estimation is demonstrated as follows:

$$\mathbb{E}[\hat{\tau}(\boldsymbol{X})] = \frac{1}{n_o}\sum_i \left(p(\boldsymbol{x}_i,1;\hat{\boldsymbol{\beta}}) + \frac{T_i}{\pi_t(\boldsymbol{x}_i;\hat{\alpha})}\epsilon_i^1\right) - \frac{1}{n_o}\sum_i \left(p(\boldsymbol{x}_i,0;\hat{\boldsymbol{\beta}}) + \frac{(1-T_i)}{1-\pi_t(\boldsymbol{x}_i;\hat{\alpha})}\epsilon_i^0\right) \tag{13}$$

where $\epsilon_i^1 = Y_i - p(\boldsymbol{x}_i,1;\hat{\boldsymbol{\beta}})$ and $\epsilon_i^0 = Y_i - p(\boldsymbol{x}_i,0;\hat{\boldsymbol{\beta}})$. The variance of DR method is derived as follows:

$$\begin{aligned}
\mathbb{V}(\hat{\tau}(\boldsymbol{X})) = {} & \mathbb{E}\left[\mathbb{V}\left(p(\boldsymbol{X},1;\hat{\boldsymbol{\beta}}) + \frac{Z}{\hat{\pi}_t(\boldsymbol{X})}\epsilon_i^1 - p(\boldsymbol{X},0;\hat{\boldsymbol{\beta}}) - \frac{1-T}{1-\hat{\pi}_t(\boldsymbol{X})}\epsilon_i^0 \mid \boldsymbol{X}\right)\right] + \\
& \mathbb{V}\left(\mathbb{E}\left[p(\boldsymbol{X},1;\hat{\boldsymbol{\beta}}) + \frac{T}{\hat{\pi}_t(\boldsymbol{X})}\epsilon_i^1 - p(\boldsymbol{X},0;\hat{\boldsymbol{\beta}}) + \frac{1-T}{1-\hat{\pi}_t(\boldsymbol{X})}\epsilon_i^0 \mid \boldsymbol{X}\right]\right) \\
\approx {} & \frac{1}{n}\sum_i^n \hat{\mathbb{V}}\left(p(\boldsymbol{x}_i,1;\hat{\boldsymbol{\beta}})\right) + \hat{\mathbb{V}}\left(p(\boldsymbol{x}_i,0;\hat{\boldsymbol{\beta}})\right) + \\
& \frac{1}{n_1}\sum_{i:T_i=1} w_i^2\hat{\sigma}_1^2(\boldsymbol{x}_i) + \frac{1}{n_0}\sum_{i:T_i=0} w_i^2\hat{\sigma}_0^2(\boldsymbol{x}_i) + \hat{\mathbb{V}}\left[\hat{\tau}(\boldsymbol{X};\bar{\hat{\boldsymbol{\beta}}},\bar{\hat{\boldsymbol{\alpha}}})\right]
\end{aligned} \tag{14}$$

Similar to the variance of IPW method and variance of G-computation method, $\hat{\mathbb{V}}(p(\boldsymbol{x},t;\hat{\boldsymbol{\beta}}))$, can be estimated using a model-based, simulation-based, bayesian, bootstrap method, and

$\hat{\sigma}_1^2(\boldsymbol{x}_i)$ and $\hat{\sigma}_0^2(\boldsymbol{x}_i)$ can be estimated using exact matching, regression adjustment approaches. In **RCTrep**, we use the sandwitch style method in **geex** to estimate the variance of DR method. The standard error of mean of $\hat{\tau}(\boldsymbol{X})$ is $\frac{\hat{\mathbb{V}}(\hat{\tau}(\boldsymbol{X}))}{n}$.

C.4. Variance of difference in means of outcomes between groups (crude estimator)

In this section, we demonstrate the variance of crude estimator, i.e., the difference in means of outcomes between treatment and control groups. The variance is derived as follows:

$$\begin{aligned}\mathbb{V}(\hat{\tau}) &= \mathbb{V}\left(\frac{1}{n_1}\sum_{i:T_i=1} Y_i(1) - \frac{1}{n_0}\sum_{i:T_i=0} Y_i(0)\right) \\ &= \frac{1}{n_1^2}\sum_{i:T_i=1}\sigma_1^2(\boldsymbol{x}_i) + \frac{1}{n_0^2}\sum_{i:T_i=1}\sigma_0^2(\boldsymbol{x}_i) \\ &\approx \frac{\hat{\sigma}_1^2}{n_1} + \frac{\hat{\sigma}_0^2}{n_0}\end{aligned} \tag{15}$$

Under simplifying assumption of homoscedasticity, i.e., $\sigma_1^2(\boldsymbol{x}) = \sigma_1^2$ and $\sigma_0^2(\boldsymbol{x}) = \sigma_0^2$ are constants across individuals, σ_1^2 and σ_0^2 can be estimated by sample variance of $Y(1)$ in the treatment group and sample variance of $Y(0)$ in the control group. We also assume observed outcomes Y_i are mutually independent, namely, the observed outcome of each individual does not depend on the observed outcome of another individual. Since $\mathbb{V}(Y \mid \boldsymbol{x}) = \mathbb{V}(Y)(1-\rho)$ where ρ is the correlation between Y and $\boldsymbol{X}$, the estimated variance of average treatment effect in equation 15 is *conservative*, and can gain efficiency if conditioning on variables $\boldsymbol{X}$ that is predictive of outcomes. $\hat{\mathbb{V}}(\hat{\tau})$ is the standard error of the average treatment effect.

D. Estimators for adjusting the sampling mechanism

In this section, we elaborate three estimators used in **RCTrep** to adjust the sampling mechanism. 1) exact matching; 2) inverse sampling score weighting; 3) subclassification.

D.1. Exact matching

In this section, we introduce weighting based on $\boldsymbol{X}_s$. This weighting approach is similar to importance sampling/transfer learning/domain adaption/covariate shift, which balances the distribution of $\boldsymbol{X}_s$ between two samples. We can also match units according to $\boldsymbol{X}_s$, for more details, see (Stuart 2010). Given assumptions on the sampling mechanism, $\mathcal{S}^{obs}$ and $\mathcal{S}^{rct}$ can be regarded as two random samples from the target population $\mathcal{P}_\theta$. Then the weight is defined as:

$$w(\boldsymbol{x}_s) = \frac{\hat{p}(\boldsymbol{x}_s)}{\hat{q}(\boldsymbol{x}_s)}, \quad \sum_{i \in \mathcal{S}^{obs}} w(\boldsymbol{x}_{si}) = 1 \tag{16}$$

where $\hat{p}(\boldsymbol{x}_s)$ and $\hat{q}(\boldsymbol{x}_s)$ are empirical density of $\boldsymbol{X}_s$ in $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$.

D.2. Inverse sampling score weighting

The sampling score is the conditional probability of being selected to the RCT data given a vector of observed covariates $\boldsymbol{X}_s$, which is defined as follows:

$$\pi_s(\boldsymbol{X}_i) = P(S = 1 \mid \boldsymbol{X}_{si}) \tag{17}$$

where $S = \{0, 1\}$, 1 indicates selection to $\mathcal{S}^{rct}$ and 0 indicates selection to $\mathcal{S}^{obs}$. In most of cases, the sampling score is unknown but could be estimated from combined data. In **RCTrep**, we denote RCT data as the target population $\mathcal{P}_\theta$, we weight individuals in $\mathcal{S}^{obs}$ according to odds of their sampling scores to resemble the $\mathcal{P}_\theta$. Hence the weights for each individual are:

$$w_i = \begin{cases} \frac{\pi_s(\boldsymbol{x}_{si})}{1-\pi_s(\boldsymbol{x}_{si})} & S_i = 0 \\ 1 & S_i = 1 \end{cases}$$

According to (Rosenbaum and Rubin 1983), the ignorability assumption holds conditioning on a balance score. The sampling score is the "coarsest" balance score, $\boldsymbol{X}_s$ is the "finest" balance score. Any balancing score finer than the sampling score can allow ignorability assumption holds. A sampling score is a propensity score when we aim to correct for confounding due to a non-random sampling mechanism.

D.3. Subclassification

Individuals are assigned to a subclass according to a distance measure, for instance, sampling score $p(S = 1 \mid \boldsymbol{X}_s)$. Many modeling approaches are provided in **RCTrep** for estimating sampling score, for instance, `glm`, `gbm`, `lasso`. $\mathcal{S}^{obs}$ and $\mathcal{S}^{rct}$ data are placed into subclasses based on quantiles of the sampling scores in $\mathcal{S}^{rct}$. Weights are computed based on the proportion of individuals in $\mathcal{S}^{rct}$ in each subclass. For more details, see R package **MatchIt**.

D.4. Variance of weighted average treatment effect

We can treat $w(\boldsymbol{x}_{si})$ as a fixed value for each individual, and use a standard Horvitz-Thompson-type sandwich variance estimator with the resulting weights via R packages **geex** or **survey**. However, it is important to consider that these weights are estimated and are unknown. Buchanan *et al.* (2018) derived a variance estimator that accounts for the variance of weights when the weights are unknown. We can also use double bootstrap to estimate variance used in (Ackerman *et al.* 2021), where both RCT data and RWD are resampled with replacement prior. This approach, however, is computationally intensive, and results are very similar to the standard sandwich variance estimator.

E. Structural causal diagram of DGM used throughout the paper

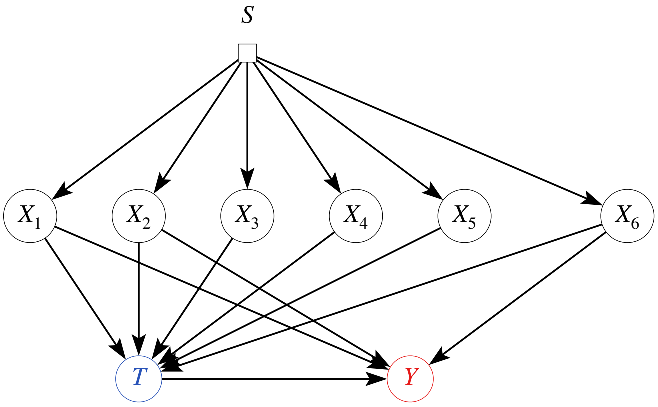


Figure 9: Structural causal diagram representing treatment T, outcome Y, sample selection with S and other predictors of outcome. The diagram visualizes the data generation mechanism of simulated data used in working examples in section 6. The figure is generated using the software **causalfusion**. Since `x3,x4,x5` are not predictive of the outcome, and X_2 and X_6 are effect modifiers, according to back-door criteria, the minimal `confounders_treatment_name` and `confounders_sampling_name` that allow T-ignorability and S-ignorability hold are `x1,x2,x6` and `x2,x6`. Adjusting X_3, X_4, X_5 can inflate the variance of estimates of average treatment effect and adjusting X_1, X_3, X_4, X_5 can inflate the variance of the weighted estimates.

.

F. Overview of the package

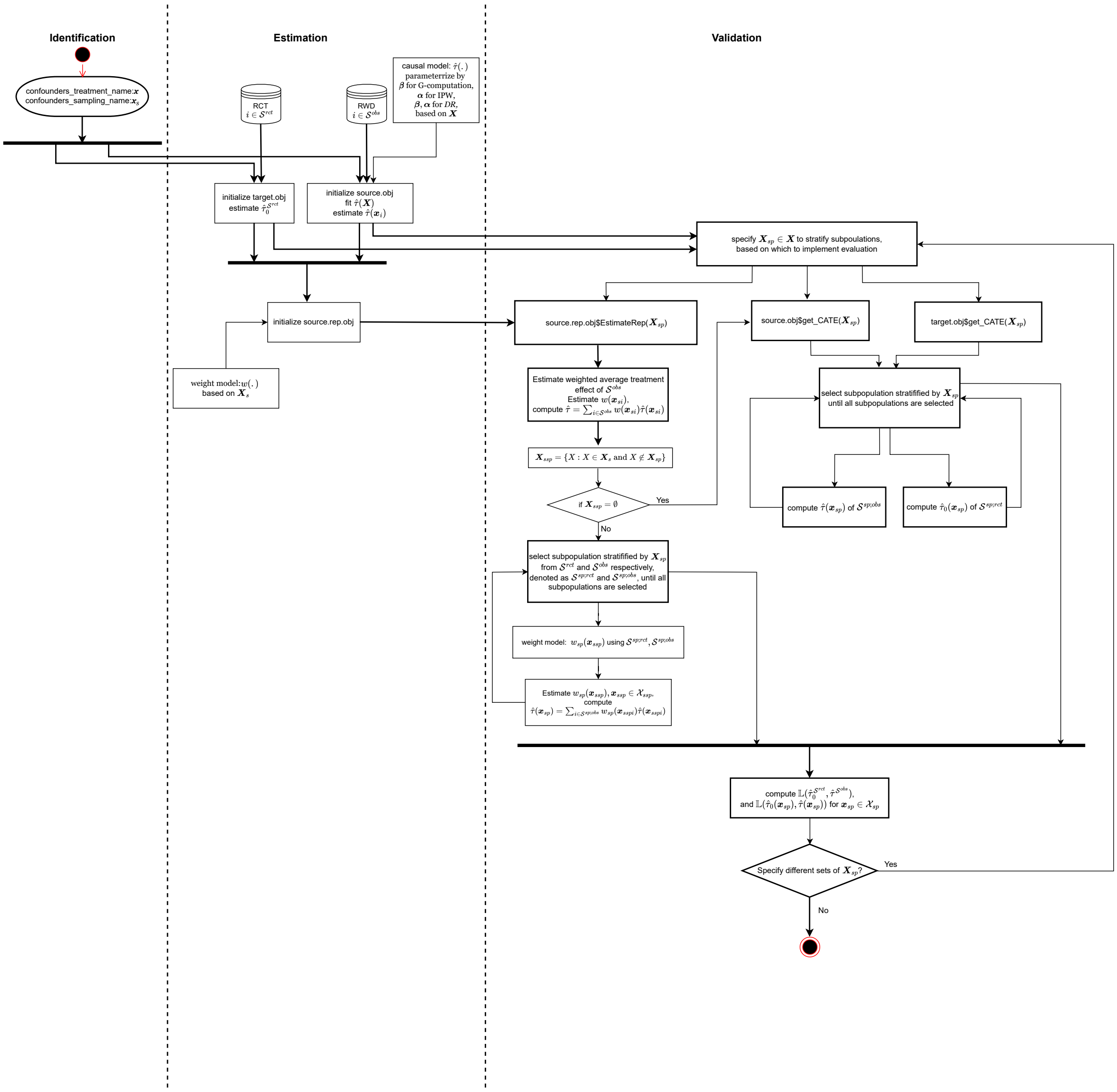

Identification
Estimation
Validation
confounders_treatment_name:$\boldsymbol{x}$
confounders_sampling_name:$\boldsymbol{x}_s$
RCT
$i \in \mathcal{S}^{rct}$
RWD
$i \in \mathcal{S}^{obs}$
causal model: $\hat{\tau}(.)$
parameterrize by
$\boldsymbol{\beta}$ for G-computation,
$\boldsymbol{\alpha}$ for IPW,
$\boldsymbol{\beta}, \boldsymbol{\alpha}$ for *DR*,
based on $\boldsymbol{X}$
initialize target.obj
estimate $\hat{\tau}_0^{\mathcal{S}^{rct}}$
initialize source.obj
fit $\hat{\tau}(\boldsymbol{X})$
estimate $\hat{\tau}(\boldsymbol{x}_i)$
initialize source.rep.obj
weight model:$w(.)$
based on $\boldsymbol{X}_s$
specify $\boldsymbol{X}_{sp} \in \boldsymbol{X}$ to stratify subpoulations,
based on which to implement evaluation
source.rep.obj$EstimateRep($\boldsymbol{X}_{sp}$)
source.obj$get_CATE($\boldsymbol{X}_{sp}$)
target.obj$get_CATE($\boldsymbol{X}_{sp}$)
Estimate weighted average treatment
effect of $\mathcal{S}^{obs}$
Estimate $w(\boldsymbol{x}_{si})$,
compute $\hat{\tau} = \sum_{i \in \mathcal{S}^{obs}} w(\boldsymbol{x}_{si})\hat{\tau}(\boldsymbol{x}_{si})$
select subpopulation stratifified by $\boldsymbol{X}_{sp}$
until all subpopulations are selected
$\boldsymbol{X}_{ssp} = \{X : X \in \boldsymbol{X}_s \text{ and } X \notin \boldsymbol{X}_{sp}\}$
if $\boldsymbol{X}_{ssp} = \emptyset$
Yes
No
compute $\hat{\tau}(\boldsymbol{x}_{sp})$ of $\mathcal{S}^{sp;obs}$
compute $\hat{\tau}_0(\boldsymbol{x}_{sp})$ of $\mathcal{S}^{sp;rct}$
select subpopulation stratifified by $\boldsymbol{X}_{sp}$
from $\mathcal{S}^{rct}$ and $\mathcal{S}^{obs}$ respectively,
denoted as $\mathcal{S}^{sp;rct}$ and $\mathcal{S}^{sp;obs}$, until all
subpopulations are selected
weight model: $w_{sp}(\boldsymbol{x}_{ssp})$ using $\mathcal{S}^{sp;rct}, \mathcal{S}^{sp;obs}$
Estimate $w_{sp}(\boldsymbol{x}_{ssp}), \boldsymbol{x}_{ssp} \in \mathcal{X}_{ssp}$,
compute
$\hat{\tau}(\boldsymbol{x}_{sp}) = \sum_{i \in \mathcal{S}^{sp;obs}} w_{sp}(\boldsymbol{x}_{sspi})\hat{\tau}(\boldsymbol{x}_{sspi})$
compute $\mathbb{L}(\hat{\tau}_0^{\mathcal{S}^{rct}}, \hat{\tau}^{\mathcal{S}^{obs}})$,
and $\mathbb{L}(\hat{\tau}_0(\boldsymbol{x}_{sp}), \hat{\tau}(\boldsymbol{x}_{sp}))$ for $\boldsymbol{x}_{sp} \in \mathcal{X}_{sp}$
Specify different sets of $\boldsymbol{X}_{sp}$?
Yes
No

Figure 10: Set up of validation of methods $\hat{\tau}(\boldsymbol{x}) \in \hat{\mathrm{T}}$ for treatment effect estimation using observational data, in which the truth of average treatment effect of the population and subpopulations are obtained from the RCT data, which is regarded as a random sample of the target population $\mathcal{P}_\theta$

G. Descriptions of the function for generate synthetic RCT data

Arguments	Description
`margin_dis`	A character specifying distribution of each variable, allowable options are `'bernoulli_categorical'` and `"bernoulli"`. `'bernoulli_categorical'` indicates variables that have two and more categories; `'bernoulli'` indicates variables that have two categories.
`N`	A numeric value indicating the sample size for the returned data.
`margin`	A list containing p named elements. The names are variable names. If `margin_dis="bernoulli_categorical"`, then each element in the `margin` is a vector with a character of the variable name, the number of the levels of the variable, the name of each level, and probability of each level; if `margin_dis="bernoulli"`, then each element in `margin` is the probability of positive value of each variable.
`var_name`	A character vector indicating the names of variables. The names should be in line with the names of elements in `margin`.
`pw.cor=0`	A vector containing the pairwise correlation of the variables `var_name`. When `margin_dis="bernoulli"`, then `pw.cor` must be specified. The default value is 0.

Table 6: Descriptions of the input arguments of the function `GenerateSyntheticData()`.

Affiliation:

Lingjie Shen

Department of Methodology and Statistics
Tilburg University
E-mail: L.Shen@uvt.nl

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd