Review Article

Propensity score matching with R: conventional methods and new features

Qin-Yu Zhao[1#], Jing-Chao Luo[2#], Ying Su[2], Yi-Jie Zhang[2], Guo-Wei Tu[2], Zhe Luo[3]

[1]College of Engineering and Computer Science, Australian National University, Canberra, ACT, Australia; [2]Department of Critical Care Medicine, Zhongshan Hospital, Fudan University, Shanghai, China; [3]Department of Critical Care Medicine, Xiamen Branch, Zhongshan Hospital, Fudan University, Xiamen, China

Contributions: (I) Conception and design: QY Zhao, JC Luo; (II) Administrative support: GW Tu; (III) Provision of study materials or patients: GW Tu, Z Luo; (IV) Collection and assembly of data: QY Zhao; (V) Data analysis and interpretation: QY Zhao; (VI) Manuscript writing: All authors; (VII) Final approval of manuscript: All authors.

[#]These authors contributed equally to this work.

Correspondence to: Guo-Wei Tu. Department of Critical Care Medicine, Zhongshan Hospital, Fudan University, Shanghai 200032, China. Email: tu.guowei@zs-hospital.sh.cn; Zhe Luo. Department of Critical Care Medicine, Xiamen Branch, Zhongshan Hospital, Fudan University, Xiamen 361015, China. Email: luo.zhe@zs-hospital.sh.cn.

Abstract: It is increasingly important to accurately and comprehensively estimate the effects of particular clinical treatments. Although randomization is the current gold standard, randomized controlled trials (RCTs) are often limited in practice due to ethical and cost issues. Observational studies have also attracted a great deal of attention as, quite often, large historical datasets are available for these kinds of studies. However, observational studies also have their drawbacks, mainly including the systematic differences in baseline covariates, which relate to outcomes between treatment and control groups that can potentially bias results. Propensity score methods, which are a series of balancing methods in these studies, have become increasingly popular by virtue of the two major advantages of dimension reduction and design separation. Within this approach, propensity score matching (PSM) has been empirically proven, with outstanding performances across observational datasets. While PSM tutorials are available in the literature, there is still room for improvement. Some PSM tutorials provide step-by-step guidance, but only one or two packages have been covered, thereby limiting their scope and practicality. Several articles and books have expounded upon propensity scores in detail, exploring statistical principles and theories; however, the lack of explanations on function usage in programming language has made it difficult for researchers to understand and follow these materials. To this end, this tutorial was developed with a six-step PSM framework, in which we summarize the recent updates and provide step-by-step guidance to the R programming language. This tutorial offers researchers with a broad survey of PSM, ranging from data preprocessing to estimations of propensity scores, and from matching to analyses. We also explain generalized propensity scoring for multiple or continuous treatments, as well as time-dependent PSM. Lastly, we discuss the advantages and disadvantages of propensity score methods.

Keywords: Causal inference; observational study; propensity score matching (PSM); R programming language

Submitted May 16, 2020. Accepted for publication Oct 29, 2020.
doi: 10.21037/atm-20-3998
View this article at: http://dx.doi.org/10.21037/atm-20-3998

What is propensity score analysis? Where is it used and why?

Where is it used?

It is essential to accurately estimate the effects of particular treatments or interventions in order to provide a solid evidential foundation for clinical practice. To this end, both randomized controlled trials (RCTs) and observational studies are commonly used tools.

RCTs can be characterized as scientific experiments that estimate treatment effects by randomly allocating subjects to two or more groups of differing clinical interventions, and comparing outcomes with respect to measured responses (2). Strategically designed randomized assignment reduces or even eliminates bias arising from the characteristics of subjects, clinicians, and administrators, or other confounding factors, and allows the effects of a particular treatment to be estimated by directly comparing outcomes between treated and untreated subjects. Therefore, randomization is the gold standard of causal inference (1-3). However, in some instances, RCTs cannot be implemented due to ethical issues, costs, and limited feasibility, which leaves the door open to implementing other new and innovative methods.

Observational studies have gained traction in recent years, owing to large amounts of available historical data, in which objects can be been observed, recorded, and compared, albeit without random treatment allocation (2,4). However, a major disadvantage of this approach is the presence of systematic differences in baseline covariates related to outcomes between treatment and control groups, which biases the results (1,5). To accurately estimate treatment effects from observational data, analysts have proposed several methods for balancing data, such as matching, stratification, and regression adjustments; however, each of these methods has its respective deficiency: (I) matching and stratification approaches group together individuals with the same or similar covariates, but these methods fail when too many covariates require balancing; i.e., it is difficult to find individuals for whom all covariates are similar. (II) For regression adjustments, it is often difficult to assess if models have correctly specified treatment assignments and covariates to outcomes. Goodness-of-fit measures, such as the coefficient of determination, R^2 and mean squared error (MSE), do not test whether a model is correct for causal inference. Thus, a model that accurately predicts outcomes may still have biased estimates of treatment effects. Although regression is a simple and direct way to eliminate bias due to other factors, it is not entirely reliable.

What is propensity score analysis?

In recent decades, propensity score analysis (PSA) has attracted increasing attention (*Figure 1*). Propensity score (PS)[1], as defined by Rosenbaum and Rubin, is the probability of receiving certain treatments, conditional on observed baseline covariates (6), and is estimated by using modeling to predict treatment allocation with covariates. In simple terms, PSA is based on the hypothesis that two patients with similar PSs have covariates which come from similar distributions. This means that by selecting or reweighting samples based on PS, researchers create new datasets where covariates are similar between treatment and control groups (7).

Methods including matching, weighting, stratification, and covariate adjustment based on PS all fall under the umbrella of PSA (8). For example, a complete analysis using propensity score matching (PSM) comprises six steps (*Figure 2*). The first step is to preprocess data sets, identify outliers, and interpolate missing values. In the second step, a model is specified, such as logistic regression, and trained on the dataset to predict whether a patient will be treated. For every patient, the trained model generates a probability of receiving treatment; i.e., his or her PS. The third step refers to matching based on PS, where different matching methods are tried, such as nearest neighbor, or optimal or genetic matching. In the fourth step, the balance of covariates between treatment and control groups is checked by calculating balance statistics and generating plots. A poor balance indicates that the model estimating PS needs to be respecified. In the fifth step, the treatment effects are estimated using matched data, and this is followed by the last step, where sensitivity analyses are performed to check the robustness of study results for hidden bias. These steps

[1] We indiscriminately used the terms propensity score and estimated propensity score. Guo and Fraser (1) summarized the meaning of these terms. In this tutorial, PS as estimated by our model, is called estimated propensity score, unless explicitly stated otherwise.

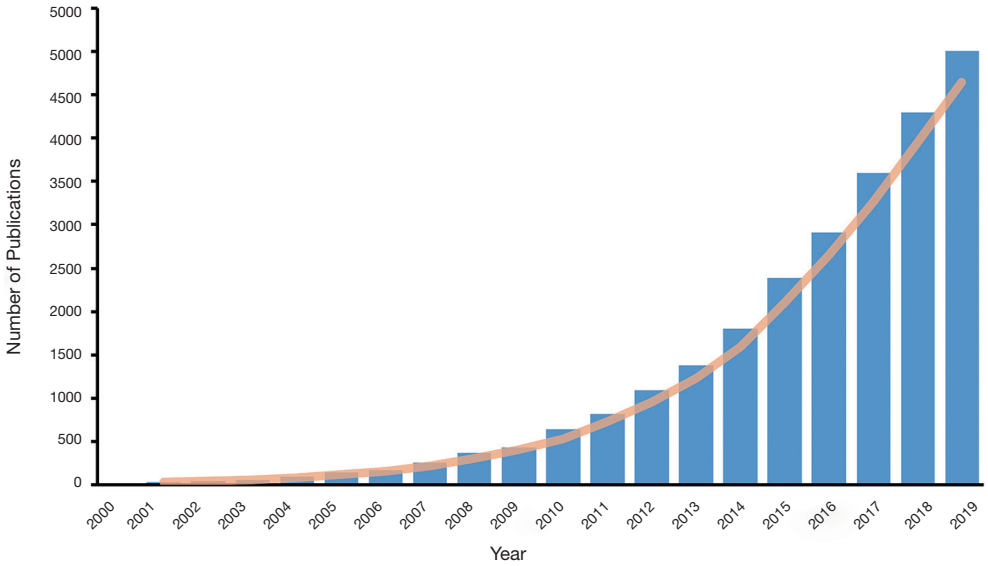

Figure 1 The trend of numbers of publications with titles containing "propensity score" from 2000 to 2019.

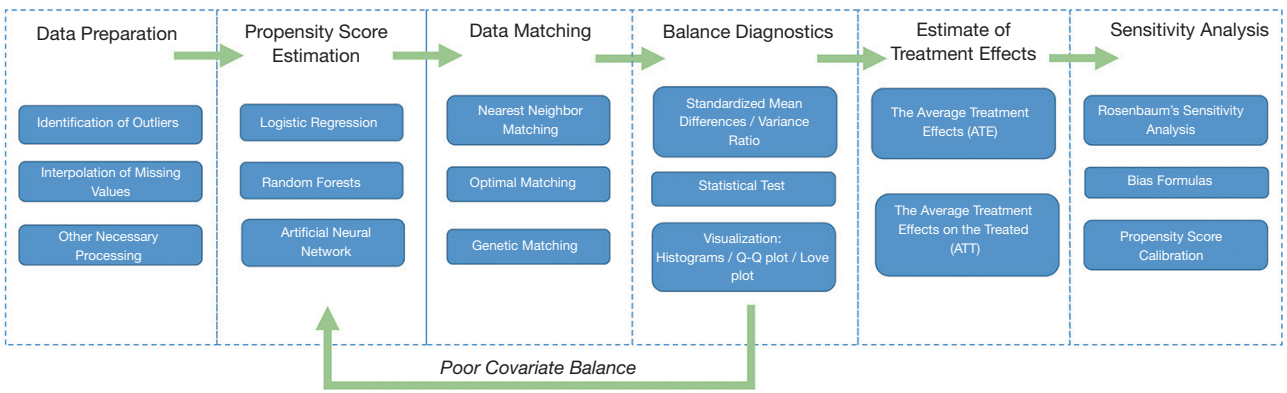

Figure 2 The framework of propensity score matching.

are later explained in detail.

What are the differences between PSA and other methods?

When compared with conventional balancing methods, PSA has two major advantages: dimension reduction and design separation. In PSA, dimension reduction means that multiple covariates are replaced by one score to describe a patient. In other words, when matching or reweighting patients, PS alone, rather than multiple covariates, is considered (1). Additionally, these methods separate covariate balancing and effect estimating. As shown in *Figure 2*, whether or not covariates are balanced between treated and untreated subjects should be checked in the fourth step, and treatment effects should be estimated after this.

Why should I read this tutorial?

While many PSA tutorials and PS articles have been published, there is room for improvement in this area. Some tutorials have provided step-by-step guidance, but cover just one or two packages (9-11), which limits their scope and practicality. Other articles and books have explained PS and related methods, as well as associated statistical principles and theories, but they explain little about the function usage in programming language, making the text

difficult for readers to follow (1,8). In 2014 and 2015, Yao *et al.* (12) evaluated cancer studies (CS) and cancer surgical studies (CSS) using PSA, and found that many studies did not clearly provide the covariates required to estimate PS (CS 12.1%, CSS 8.8%), neglected comparisons of baseline characteristics (CS 21.9%, CSS 15.6%), or failed to report matching algorithms (CS 19.0%, CSS 36.1%). A systematic review by Malla *et al.* (13) observed that approximately 49% of articles through 2010–2017 (45% in 2016 and 2017) did not report imputation methods for missing data when applying PSM. Zakrison *et al.* (14) reviewed observational studies using PSA in the acute care surgery literature and found that more than 33% of studies did not adequately report their methods.

To this end, we composed this tutorial to introduce the PS concept and framework and to walk readers through how to implement PSM with R, with the aim of making readers capable of completing an analysis process using PSM.

Our ultimate goal is to equip researchers with a comprehensive understanding of PSM, ranging from data preprocessing to PS estimations, from matching based on PS to the subsequent analysis, from generalized propensity score (GPS) for multiple or continuous treatments to time-varying PSM, etc. Unlike previous studies where technical details were either too brief or too abstruse, we have shown each algorithm in a simple manner, described the basic principles concisely, and illustrated steps using figures. Thus, this tutorial is a crucial document for clinical analysts. The mind map for causal inference with PSM is summarized in *Figure 3*.

Why choose PSM in R?

There are at least three reasons why this tutorial focuses on PSM: (I) several studies have shown that PSM eliminates a higher proportion of systematic differences in baseline characteristics between treated and control groups when compared with other methods (15-17); (II) PSM is more robust for the misspecification of the PS estimating model (3,18); and (III) PSM is the most commonly used PS method, having been implemented in many fields (12,14). It is worth noting that apart from the matching element of this tutorial, the PSM framework is similar to that of other PS methods; therefore, this tutorial is also valuable to readers wishing to try other PS methods.

Nowadays, several programing languages including R (https://www.R-project.org/), Python (https://www.python.org/), and STATA (https://www.stata.com), support various PS methods. We have chosen R to illustrate how to perform PSM in different packages, as it is free, open-sourced, and user-friendly. We used the R version 3.6.2 for Windows (2019-12-12), which can be download from https://cran.r-project.org/. The package versions we used are listed in *Table 1*.

As some readers prefer to use other analytical tools such as Statistical Product and Service Solutions (SPSS) and Software for Statistics and Data Science (Stata), we provide package or tutorial recommendations for these tools in this part. We direct SPSS users to Huang *et al.* (19), who realized PSM in the PS matching module of SPSS and explain the analysis results. The software allows for the estimation of PS using logistic regression, and specifies options for nearest-neighbor matching, e.g., calipers, region of common support, matching with and without replacement, and matching one to many units. The program produces detailed statistics and graphs. For Stata users, *psestimate*, *psmatch2*, *pscore*, and other modules may be used for analysis. For example, *psmatch2* implements full Mahalanobis and PSM, common support graphing, and covariate imbalance testing. Interested readers are directed to the package's official documents.

The structure of this tutorial is as follows: Section 'How do I start my PSM analysis when there are missing values in my dataset?' covers data preparation, where we concentrate on methods for the interpolation of missing data for PSM. Section 'How is PS estimated? What methods can be used?' introduces approaches on how to estimate PS for dichotomous treatments using different models, such as logistic regression, random forests, and artificial neural networks. Section 'How can patients with similar PSs be matched?' concentrates on matching methods based on PS. Sections 'What is a good match? Is my data balanced?' and 'What should be done if covariates are still unbalanced after matching?' focus on balancing diagnostics after matching. Sections 'How can the effects of treatment after matching be estimated?' and 'What about unmeasured covariates

[2] We use the term "treatment" throughout this article to simplify the language, although it is sometimes better to use "exposure" or "intervention".

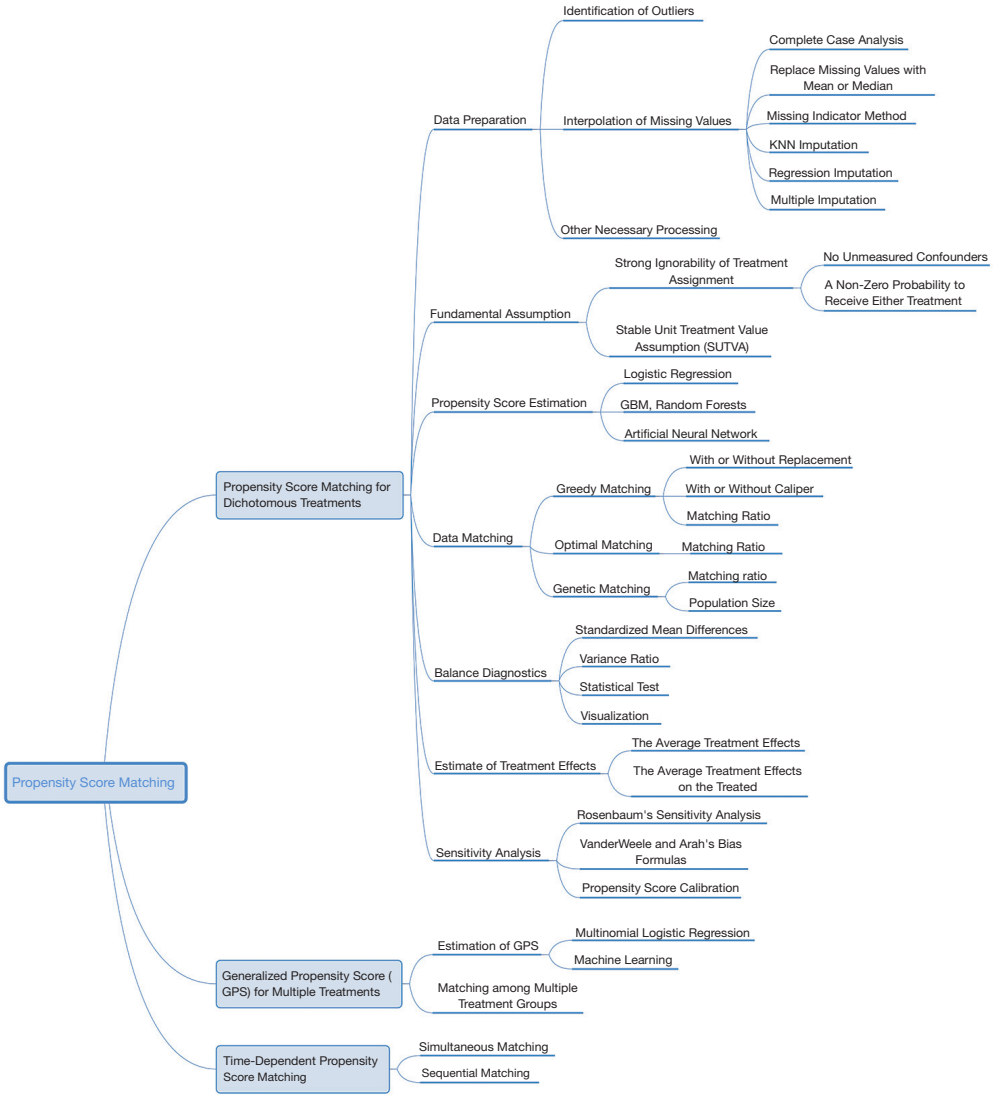

Figure 3 The mind map of causal inference with propensity score matching.

related to treatment allocation and outcomes?' center on analysis after matching, including estimating treatment effects and sensitivity analyses. In Section 'How does PSM handle multiple or continuous treatments?' and 'How to apply PSM with time-dependent covariates and treatments', we introduce GPS for multiple or continuous treatments, and time-dependent PSM. In the last section, we discuss PSM in terms of strengths and weaknesses, and we assess the limitations of this tutorial.

How do I get access to data from this tutorial?

To illustrate PSM in R, we constructed a dataset with known true PS values, for step-by-step guidance in each section. This dataset simulates the relationship between smoking and cardiovascular disease (CVD), with age and gender as the potentially confounding covariates. However, one must bear in mind that real data is not as simple as the simulated data in this tutorial, in which only two confounding factors are present. We will cover how to determine covariates from hundreds of variables in observational datasets for estimating PS (Section 'Selection of covariates').

In a hypothetical situation, we want to study the relationship between smoking and CVD incidence. Thus, CVD is the outcome variable (CVD = 0 when the patient

Table 1 The R packages used in the tutorial

R Package	Version	Description
Hmisc	4.4.0	A package containing many functions useful for data analysis. Here we use it for imputing missing values
MatchIt	3.0.2	A package integrating multiple methods to estimate PS and match based on the estimated score
ggplot2	3.2.1	A system for declaratively creating graphics
nnet	7.3.12	A package for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models. The function to set up multinomial log-linear models in this package will be used
tableone	0.10.0	Creates 'Table 1', i.e., description of baseline patient characteristics, which is essential in every medical research
DMwR	0.4.1	This package includes functions and data accompanying the book [18]. In this tutorial we use KNN imputation provided in this package
cobalt	4.0.0	Generate tables and plots for covariate balance diagnostics after matching, weighting or subclassification, based on propensity scores
weights	1.0.1	A package which provides a variety of functions for producing simple weighted statistic
Zelig	5.1.6	A framework that brings together an abundance of common statistical models found across packages into a unified interface
rbounds	2.1	A package to calculate Rosenbaum bounds for the treatment effect
randomForest	4.6.14	A package which implements Breiman's random forest algorithm for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points

PS, Propensity score; KNN, K-Nearest Neighbor.

does not have CVD; otherwise, CVD =1). Smoking is a binary variable which indicates whether the patient smokes or not. Gender (0= female, 1= male) and age are two confounding covariates, which are correlated to "treatment"[2] and outcome. Now, we assume that men smoke more than women, and middle-aged people smoke more than elderly and younger people. Lastly, we randomly set variable values as NA (missing).

Data is generated with the following code:

```
> set.seed(2020)
> x.Gender <- rep(0:1, c(400,600)) # 400 females and 600 males
> x.Age <- round(abs(rnorm(1000, mean=45, sd=15)))
> z <- (x.Age - 45) / 15 - (x.Age-45) ^ 2 / 100 + 2 * x.Gender
> tps <- exp(z) / (1+exp(z)) # The true PS
> Smoke <- as.numeric(runif(1000) < tps)
> z.y <- x.Gender + 0.3*x.Age + 5*Smoke - 20
> y <- exp(z.y) / (1+exp(z.y))
> CVD <- as.numeric(runif(1000) < y)
> x.Age.mask <- rbinom(1000, 1, 0.2) # Missing completely at random
> x.Age <- ifelse(x.Age.mask==1, NA, x.Age)
> data <- data.frame(x.Age, x.Gender, Smoke, CVD)
> head(data)
```

	x.Age	x.Gender	Smoke	CVD
1	51	0	1	0
2	50	0	0	0
3	29	0	0	0
4	28	0	0	0
5	3	0	0	0
6	56	0	1	1

Here, *tps* is the true PS which can balance data and needs to be estimated.

Next, we use the *CreateTableOne* function in the t*ableone* package, to summarize the distribution of baseline variates in each group, and test whether significant differences exist between different groups. If you do not have *tableone*, it

Table 2 Baseline characteristics of our simulation data

Variables	Level	0	1	P	SMD
n		549	451		
x.Age, mean (SD)		42.76 (19.69)	47.04 (8.14)	<0.001	0.284
x.Gender, n (%)	0	299 (54.5)	101 (22.4)	<0.001	0.698
	1	250 (45.5)	350 (77.6)		
CVD, n (%)	0	452 (82.3)	230 (51.0)	<0.001	0.705
	1	97 (17.7)	221 (49.0)		

The numbers 0 and 1 in the first row indicates whether to smoke, 0 means not while 1 means yes. The variable n is the number of patients, x.Age and x.Gender are the independent variables and CVD is the indicator of cardiovascular disease. SMD, standardized mean difference; SD, standard deviation.

can be installed with the following code: install.packages ("tableone"). The results of following code are shown in *Table 2*.

```
> library(tableone)
> table2 <- CreateTableOne(vars = c('x.Age', 'x.Gender',
'CVD'),
data = data,
factorVars = c('x.Gender', 'CVD'),
strata = 'Smoke',
smd=TRUE)
> table2 <- print(table2,
smd=TRUE,
showAllLevels = TRUE,
noSpaces = TRUE,
printToggle = FALSE)
> table2
> write.csv(table2, file = "Table2_before_matching.csv")
```

Here, the argument *vars* is a vector of variables to be described, *factorVars* is a vector of variables that should be handled as categorical variables, and *strata* is the column name that groups the data set. As can be seen, the covariates of age and gender are significantly different between the smoking and non-smoking groups. The SMD column refers to standardized mean differences between different groups, which is calculated by the following equations.

For continuous variable x:

$$\text{SMD of x} = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\left(Var_1 + Var_2\right)/2}} \quad [1]$$

where $\overline{X_1}$ and $\overline{X_2}$ are the sample means for the treated and controlled groups, respectively; Var_1 and Var_2 are sample variances for the treated and controlled groups, respectively.

For dichotomous variable x:

$$\text{SMD of x} = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\left[\hat{p}_1\left(1-\hat{p}_1\right) + \hat{p}_2\left(1-\hat{p}_2\right)\right]/2}} \quad [2]$$

where $\hat{p}_1$ and $\hat{p}_2$ are the prevalence of dichotomous variables in the treated and control groups, respectively.

The use of SMD is recommended to measure the covariate differences between different groups when using PSM. It provides a fair scale to compare different types of covariates that are not influenced by the units of measure, because it standardizes the differences based on the variance of the samples.

How do I start my PSM analysis when there are missing values in my dataset?

Data preparation is the first step in PSM, as outliers or missing values will impede an accurate PS estimate. Although some models, like XGBoost, are immune to missing values, they are rarely used in estimating PS; meanwhile, common models, such as logistic regression and artificial neural network, cannot deal with missing values. Therefore, in this section, we focus on missing data, and demonstrate and compare different methods for missing values.

How do I deal with missing values?

In theory, missing data may be classified as missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR) (20), but these classifications are not focused on here. Complete case analysis is a simple method to deal with missing data. It deletes cases with missing values, and is based on the assumption that data are MCAR, and they substantially reduce sample size and increase variance (20,21). The approach is not recommended, particularly for small-data analyses, or when miss ratios are high. The missing indicator method replaces missing values with a constant, usually 0, and adds missing indicators for each imputed covariate. This method has been criticized as it usually introduces bias (22). Replacing missing values with the mean, median, or mode is also widely used. Despite its simplicity, the approach reduces variability in the data set, and underestimates variance (8,21). The K-nearest neighbors (KNN) and random forests are machine-learning methods that can be used to impute missing values (23-25). In addition to these, model-based single imputation methods, such as regression imputation, or matching-based single imputation methods, such as hot-deck imputation, are better-designed, but they amplify correlations in the data (8). Several studies (8,20,26) recommend multiple imputation (MI) to impute missing data, because this method does not reduce inherent variability.

Although MI is currently the preferred missing data imputation method, there is still debate as to which method is the most appropriate for PSM (8,27,28). In a simulation study, Choi *et al.* (29) compared four different methods: complete case analysis, the missing indicator method, MI, and combining MI and the missing indicator method. These authors found that MI would fail when data were non-randomly missing, and the optimal way to handle missing values depended on the missing data structure.

Combining MI and PSM is another contested issue. First, it has been recommended (30) to separately impute data for treatment and control groups, as missing mechanisms and covariate distributions may be different between these groups. However, this approach was not mentioned in recent studies (29). Its major disadvantage is that separate imputation mechanism reduces the sample size for imputation to each group, which biases the imputed values. Meanwhile, two methods that implement PSM with multiple imputed data have been compared (7). In the first, PSM is separately applied to each dataset imputed by MI, and the treatment effect estimates are averaged. In the second, PSs on each completed dataset are estimated, the scores for each record are averaged, and PSM is performed with averaged scores to estimate treatment effects. This second method has the greater potential to produce substantial bias reductions.

How do I implement missing data methods with R?

Checking missing values

Here is a simple way to count the number of missing values in each column.

```
> colmissing <- apply(data, 2,
function(x){ sum(is.na(x)) })
> colmissing
```

x.Age	x.Gender	Smoke	CVD
169	0	0	0

As can be seen, there are 169 missing values in the column *x.Age*.

The *Hmisc* package contains several functions for data analysis. The data may be summarized as follows:

```
> library("Hmisc")
> describe(data)

4          Variables          1000      Observations
x.Age
n       Missing  Distinct   Info   Mean   Gmd     .05    .10   .25   .50   .75   .90   .95
831       169       85       1    44.66  17.74    19     25    34    44    55    65    72
Lowest:          1  2  3  5  6,          Highest:      84 85 86 93 101
x.Gender
n       Missing  Distinct   Info   Sum    Mean    Gmd
1000      0        2       0.72   600    0.6    0.4805
Smoke
n       Missing  Distinct   Info   Sum    Mean    Gmd
1000      0        2       0.743  451    0.451  0.4957
CVD
n       Missing  Distinct   Info   Sum    Mean    Gmd
1000      0        2       0.651  318    0.318  0.4342
```

Gmd refers to Gini's mean difference, which is a robust

measure of dispersion. The reader is directed to (31) for more information.

Completing the case analysis

The function *na.omit* returns a new data frame which contains samples without missing values.

```
> data.complete <- na.omit(data)
> nrow(data.complete)
831
> colmissing <- apply(data.complete, 2,
function(x){ sum(is.na(x)) })
> colmissing
```

x.Age	x.Gender	Smoke	CVD
0	0	0	0

As can be seen, those samples with missing values are deleted.

Replacing missing values with mean, median, or specified constant

The use of the *impute* function in the *Hmisc* package is straightforward.

```
> x.Age.meanimp <- impute(data$x.Age,mean)
> x.Age.medimp <- impute(data$x.Age,median)
> x.Age.consimp <- impute(data$x.Age,2)
```

The missing indicator method

The missing indicator method is a simple approach,

```
> x.Age.ind <- as.numeric(is.na(x.Age)==FALSE)
> data.mi <- copy(data)
> data.mi$x.Age <- impute(data$x.Age, 0)
> data.mi <- cbind(data.mi, x.Age.ind)
```

where *x.Age.ind* indicates *Age* (1 = missing, otherwise 0)

The KNN imputation

For every patient with missing values, this algorithm finds the top *k* similar patients as the nearest neighbors, and then uses the covariate values of neighbors to impute the missing ones. Now, we use the *knnImputation* function in the *DMwR* package to implement KNN imputation. The argument k is the number of neighbors, and the option *meth* = "weighAvg" means that the nearer neighbor will be given a larger weight to impute.

```
> library(DMwR)
> data.knnimp <- knnImputation(data, k=10, meth =
"weighAvg")
```

Multiple imputation

The MI approach is more complex than the above-mentioned methods. The code is hard to understand unless the user knows how to match patients based on PS. Therefore, we have provided the PSM code with MI attached in the appendix for reference.

How is PS estimated? What methods can be used?

The PS is the conditional probability of receiving a treatment, given a vector of observed covariates, X. In this section, we discuss the simplest case, where two treatment options (binary) are provided; i.e., treatment *vs.* control.

A common method to estimate PS is to build a model and predict treatment assignment based on covariates. First, appropriate covariates are selected, and a modeling method, e.g., logistic regression, is specified. Then, a predictive model is trained on the data using the selected covariates to predict whether a patient will be treated. Lastly, the model makes predictions for each patient, and PS is the probability of receiving treatment, as predicted by the model.

In the following paragraphs, we demonstrate covariate selection, the methods to estimate PS, and how to estimate PS with R.

Selection of covariates

Not all covariates need to be balanced. According to the theory of casual inference (32,33), we can analyze the different relationships between the covariate (X), the treatment assignment (W), and the outcome (Y). As demonstrated in *Figure 4*, the X covariates in the first row (I, II) should be included in the estimating model, while those in the second row (III, IV) should not.

This X is the true confounding variable in the data. If we suppose that older patients are more likely to be given intraoperative blood transfusions (W) but with a higher

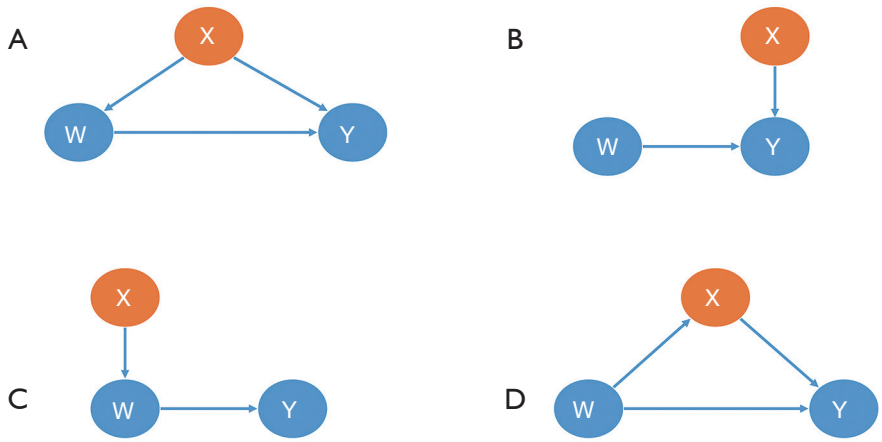

Figure 4 Different relationships of covariate, treatment assignment, and outcome. X is a covariate, W is an indicator of treatment assignment, and Y is the outcome. The blue arrows point from cause to effect.

mortality risk (Y), we observe then that blood transfusions caused significantly more deaths in the historical data. Age is a true confounder, and should be balanced between the transfusion and non-transfusion groups; otherwise, the results will be biased.

Although there is no direct causal relationship between X and W, several simulation studies suggest that these covariates should always be included in a PS model. The inclusion of these variables will decrease the variance of the outcome estimates, without increasing bias (34,35).

In contrast, including variables related to the treatment but not to the outcome, will increase the variance of the estimated exposure effect, without decreasing bias (34).

This X is a mediating variable in the causal path from W to Y; i.e., W affects Y by X. For example, if we want to study the effects of cigarettes on the risk of lung tumors,X refers to the nicotine intake of each individual per day, Y is the average number of cigarettes smoked per day, and Y is the lung tumor indicator. Nicotine is a powerful poison in cigarettes. People who inhale high levels of nicotine can damage their lungs (W->X->Y). If X is balanced between the smoking and non-smoking groups, then the effect of W on Y will be underestimated or even eliminated.

Unfortunately, domain knowledge must be applied to ensure all important confounders are included in an analysis, as doing so solely by automatic computer script is impossible (36).

Estimating methods

Binary logistic regression is the mainstream PS estimating method, but an increasing number of alternative methods are being proposed, including bagging or boosting (31,37), recursive partitioning algorithms (31), the generalized boosting model (GBM) (37,38), random forests (31), and artificial neural network (ANN) (39).

The most significant advantage of logistic regression (LR) is its simplicity and interpretability. To illustrate, in an observational study, older patients are more likely to be assigned to a treatment group, and younger patients to control groups. Thus, for outcomes, mortality is correlated to age. Accordingly, you can model the treatment assignment as a function of age using LR, and estimate PS with your prediction model.

Now let us assume the situation is more complicated: older patients and children are more likely to be treated than middle-age patients. If researchers use LR without careful thinking, estimating PS could be biased because the relationship between treatment assignment and age is no longer linear. In this case, an experienced researcher will add polynomial or interaction terms to the model to fit more complicated relationships between covariates and treatment assignments.

Therefore, after we estimate PS and match based on it, a data balance diagnostic is required to guarantee that the model is well specified to eliminate confounding factors. This step is introduced in section 'How can patients with similar PSs be matched?'.

The estimation of PS by LR is time-consuming, because different polynomial or interaction terms must be determined to balance the data. While no method

is optimal, machine learning may help. GBM, random forests, and ANN automatically fit complex and high-order relationships in data, which frees researchers from having to find the best model structure (39-42). For example, ANN is composed of one or more layers, in which data are transformed in linear and nonlinear ways, and so ANN is more flexible for data mining than LR. In the literature, Westreich *et al.* (42) and De Vries *et al.* (43) observed that the use of classification and regression trees (CART) is promising for PSA application. Simulation experiments by Keller *et al.* (41) indicated that ANN performed similarly to LR when the LR model was correctly specified, but when the LR was inadequately specified, it was outperformed by ANN. Recently, Super Learner, an ensemble machine learning algorithm, was shown to be robust for the misspecification of the PS estimating model (44).

Although machine learning methods are criticized because of their poor interpretability, we argue that more attention should be paid to whether the estimated PS balances the observational data, rather than what the relationship is between the covariates and treatment assignment. Therefore, these "black box" methods may be promising alternatives to LR. Nonetheless, there is much to explore and research concerning PSM and machine learning methods (8).

Estimating PS with R

MatchIt package

The *MatchIt* package offers users practical functions to estimate PS. We recommend first installing the package and uploading it.

```
> install.packages("MatchIt")
> library(MatchIt)
```

matchit is the main function in *MatchIt*, which can estimate PS and match similar patients. It is used as follows:

```
> matchit(formula, data, method = "nearest", distance = "logit",
distance.op-tions = list(), discard = "none", reestimate = FALSE,
...)
```

The argument *formula* follows the usual syntax of R formula, which specifies the PS estimate model in which treat ~ x1 + x2, where treat is a binary treatment indicator, and x1 and x2 are pretreatment covariates; i.e., Smoke ~ x.Age + x.Gender.

Distance is the key argument that decides what model you will use to estimate PS, and can include "logit" (logistic regression), "GAMlogit" (generalized additive models), "rpart" (CART), and "nnet" (single-hidden-layer neural network model). Also, *Distance* may be PS estimated by the researcher. This means that PS can be estimated with other models in other packages, and the function, *matchit*, can be used to match patients based on your estimated PS. It is worth noting the option "mahalanobis" (Mahalanobis metric distance), which is not a method based on PS. In fact, Mahalanobis metric distances were developed and studied prior to PSM (45,46). They play a similar role as PS in assessing the similarities of different patients.

An optional argument, *distance.options*, specifies parameters for estimating models. The input of this argument must be a list. *Table 3* lists those parameters that can be set. Some are optional, while others must be set if you want to use this model. The *matchit* function completes both PSM estimations and matching steps. However, here we refer to the estimation only. How to match and set other arguments such as *method* and *discard* will be covered in the next section.

Running matchit with logistic regression

We then use the following code to run *matchit*. As mentioned previously, we do not set other arguments, except distance. We use complete cases in our dataset for the following steps, because *x.age* is missing completely at random and our data of just two columns is so simple that it leads to the poor performance of other imputation methods.

```
> m.out <- matchit(Smoke~x.Age+x.Gender,
data=data.complete,
distance='logit')
```

The value model of *m.out* can be checked to get the summary of the estimating model. This is the summary of our logistic regression.

```
> m.out$model
Call: glm(formula = formula, family = binomial(logit), data =
data)
```

Table 3 Parameters of different propensity score estimate models

Model	Paramters	Description	Can it be missing?	Examples
Logistic regression	maxit	The maximum number of iterations to optimize the model	Yes	500
GAM	intercept	Should an intercept be included in the model?	Yes	TRUE
Neural network	size	Number of units in the hidden layer.	No	16

GAM, generalized additive model.

```
Coefficients:
(Intercept)      x.Age      x.Gender
-2.05065         0.01953    1.51120

Degrees of freedom: 830 Total (i.e. Null); 828 Residual
Null deviance: 1142
Residual deviance: 1027 AIC: 1033
```

The value *distance* will return the estimated PS for each sample, which is a vector of length n (PS is considered as the distance of patients to match here).

Now, we compare the estimated PSs with the true values.

```
> eps <- m.out$distance # Estimated PS
> tps.comp <- tps[complete.cases(data)]
# Because we only use complete cases
> Smoke.comp <- as.factor(Smoke[complete.cases(data)])
> df <- data.frame(True=tps.comp, Estimated=eps,
Smoke=Smoke.comp)
> ggplot(df, aes(x=True, y=Estimated, colour=Smoke)) +
geom_point() +
geom_abline(intercept=0,slope=1, colour="#990000",
linetype="dashed") +
expand_limits(x=c(0,1),y=c(0,1))
```

In the scatter plot in *Figure 5A*, the x-axis is the true PS value, while the y-axis is the estimated value. The small circles represent samples. If PSs were estimated accurately, these circles would be distributed around the line y = x. The reason why the estimation is so poor (like a *sin* function), is because we generated PS by quadratic equation, but estimated it by linear equation. We can change our model to fit the nonlinear relationship, so it can be operated by the following code. Note that when we use PSM with real datasets, we do not know how the true PS is distributed, so this adjusting of the model by logistic regression may take more time.

```
> m.out <- matchit(Smoke~I(x.Age^2)+x.Age +x.Gender,
data=data.complete) # Add a quadratic term
> m.out$model
Call: glm(formula = formula, family = binomial(logit), data =
data)

Coefficients:
(Intercept)   I(x.Age^2)   x.Age      x.Gender
-24.89833     -0.01056     1.02879    2.03851

Degrees of freedom: 830 Total (i.e. Null); 827 Residual
Null deviance: 1142
Residual deviance: 673.4          AIC: 681.4
```

As seen in *Figure 5B*, when the correct model is specified, PS is accurately estimated.

Running matchit with CART and a neural network

Now, we will show you the estimating results of CART and a single-hidden-layer neural network (*Figure 5C,D*, respectively). Only the call of the *matchit* function is shown here, as the remaining codes are all the same.

```
> m.out <- matchit(Smoke~x.Age+x.Gender,
data=data.complete,
distance='rpart')
```

Then, we change the distance argument to “nnet”, and set the number of hidden-layer units as 16.

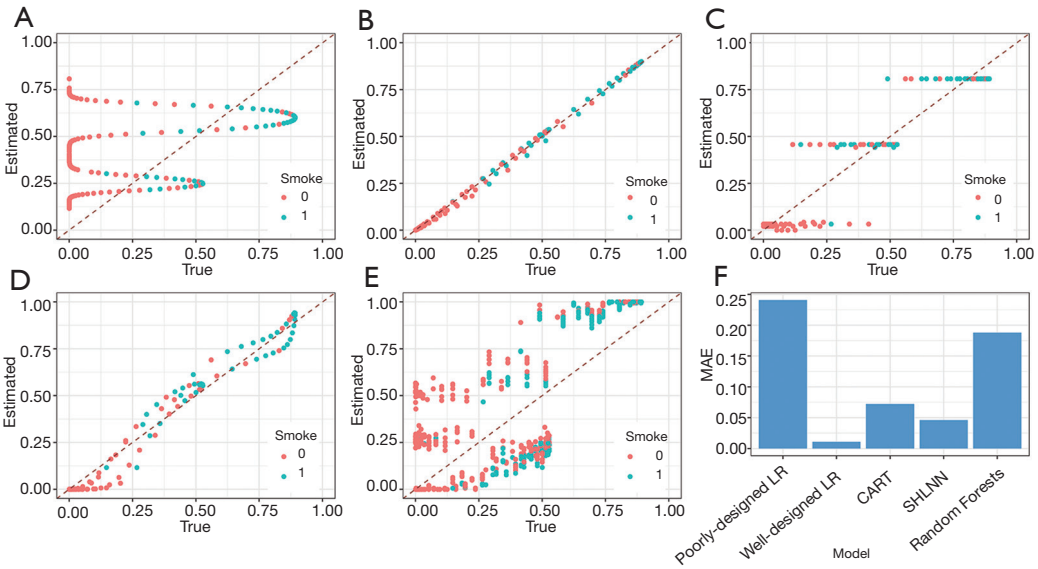

Figure 5 The true vs. estimated propensity scores. The true propensity scores (x-axis) and estimated ones by different models (y-axis). (A) Poorly-specified logistic regression. (B) Well-specified logistic regression. (C) CART. (D) Single-hidden-layer neural network. (E) Random forests, (F) MAE of different models. LR, logistic regression; CART, classification and regression tree; SHLNN, single-hidden-layer neural network; MAE, mean absolute error.

```
> m.out <- matchit(Smoke~x.Age+x.Gender,
data=data.complete,
distance='nnet',
distance.options=list(size=16))
```

Although the true PS is generated in a complex manner, the CART and neural network can also estimate the score based on *x.Age* and *x.Gender*. In practice, when we do not know how the true PS was generated, logistic regression is not always perfect, while neural networks can estimate better scores based on baseline covariates.

Estimating PS by other functions

Apart from calling the built-in function of *MatchIt*, PS can also be estimated using the model you specify, like any other dichotomous problem. In this section, we use random forests, a popular ensemble algorithm based on decision trees. Several R packages support random forests; here, we chose the package *randomForest*.

```
> library(randomForest)
> data.complete$Smoke <- factor(data.complete$Smoke)
# to call classifier
> rf.out <- randomForest(Smoke~x.Age+x.Gender, data=data.
complete)
> rf.out
Call: randomForest(formula = Smoke ~ x.Age + x.Gender,
data = data.complete)
Type of random forests: classification
Number of trees: 500
No. of variables tried at each split: 1
OOB estimate of error rate: 22.86%
Confusion matrix:
```

	0	1	class.error
0	360	102	0.2207792
1	88	281	0.2384824

We use the fraction of votes as treatment probabilities, namely, PSs.

```
> eps <- rf.out$votes[,2] # Estimated PS
```

As shown in *Figure 5E*, the random forests performances

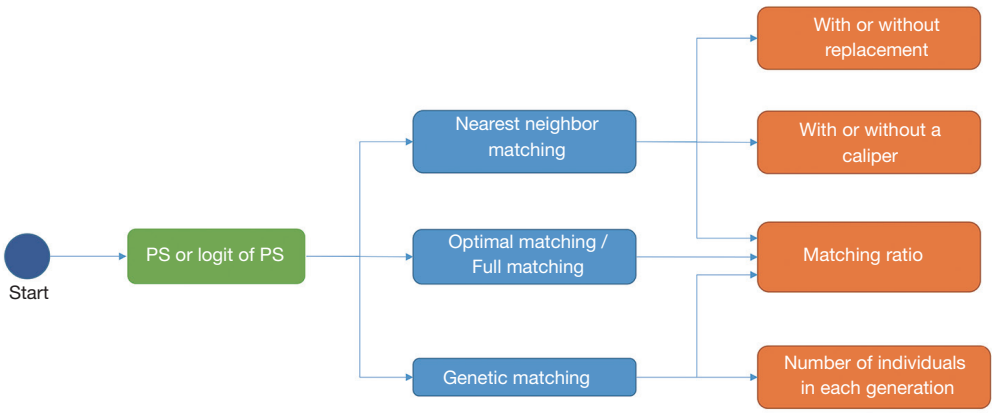

Figure 6 An idea map for choosing matching methods. PS, propensity score.

are unsatisfactory. After we calculate PS with our model, the *matchit* function may still be used to match data, based on the estimated PS:

```
> matchit(formula=Smoke~x.Age+x.Gender,
data=data,
distance=eps, # our own estimated PS
method='nearest',
replace=TRUE,
discard='both',
ratio=2)
```

Different models in terms of estimating accuracy are summarized in *Figure 5F*, but one example cannot show their performances in all situations.

How can patients with similar PSs be matched?

Matching methods

After we calculate PS, we form pairs between treated and untreated subjects, according to their PSs. There are several ways to do this, but several choices must first be considered, as outlined below. We have summarized an idea map for choosing matching methods (*Figure 6*), and compared matching results of different algorithms (*Figure 7*). A separate flowchart (*Figure 8*) illustrates how genetic matching works because it is more complex.

PS or logit of PS

Early in 1985, Rosenbaum and Rubin (47) suggested using the logit of PS (i.e., $\log\left(\frac{PS}{1-PS}\right)$) to match samples, because the distribution of logit PS approximates to normal, although matching based on PS is still feasible.

Major matching algorithms

Nearest neighbor matching (NNM) is a widely used method, and is composed of two steps. First, the algorithm picks patients, one by one from treatment groups, in a specified order. Then, for each treated patient, the program finds a patient in the control group with the nearest PS. These steps are repeated until no patients are left in the treatment or control group. As shown in *Figure 7A*, "1" is selected first, and matched to an untreated subject. The distance between their PSs is the smallest (0.1). Then "2" is selected and matched.

The essence of this method is that it divides the whole, complex, matching problem into small parts, matching one treated subject at one time, and determining the best counterpart every time. The method is considered "greedy" because it is greedy in every subproblem (i.e., it finds the best one), yet overall optimal output is not guaranteed. As shown in *Figure 7B*, the algorithm picks "1" first and "2" next. For "1", the algorithm finds the best untreated patient to match, their PS difference is 0.1. However, for "2", the distance is much worse. An observer may conclude that if we deal with "2" first and then "1", an optimal matching situation will result (*Figure 7C*). Indeed, different orders for patient selection have been proposed, such as from the highest to lowest PS, from the lowest to highest PS, or random order. However, simulation experiments have demonstrated that the order in which treated subjects are selected for matching exerts only modest effects

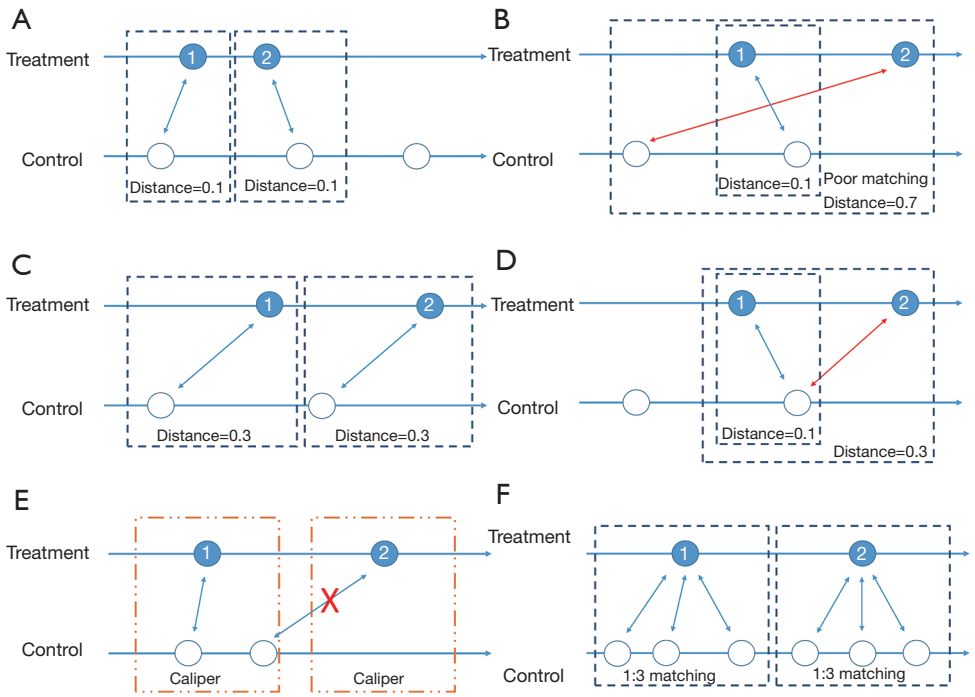

Figure 7 Illustration and comparison of different matching methods. In every subfigure, blue circles represent the treated patients, while hollow circles represent controlled ones. The one-way horizontal arrows indicate the distribution of propensity score from smallest to greatest. White numbers in circles represent the order in which the samples are selected. Double-headed arrows and dark blue dotted boxes indicate matching relationships, while the orange boxes show the width of calipers. Distance is the difference of propensity scores between a matched pair. Red color is used for emphasis, and the red cross on the arrow means that matching will not be made as a result of the specified caliper.

on estimations (48). Equally, other methods have been proposed to alleviate this problem, such as specifying a caliper width (discussed later). Despite suboptimal performances, NNM is used extensively by researchers for PSM. It is a simple algorithm to understand and implement, and when there are sufficient untreated patient numbers, these poor matching issues are unlikely to happen.

In contrast to NNM, optimal matching (OM) was proposed by Rosenbaum (49) to optimize total distances between treatment and control groups. This means that algorithms determine matched samples with the smallest average within-pair absolute difference in PS (*Figure 7C*). Determining optimal matches is implemented using network flow theory (50,51), but this is beyond the scope of this tutorial. Gu and Rosenbaum (52) observed that OM was sometimes better than NNM as it produced closely matched pairs, but was no better than NNM in terms of balancing covariates. In a comparison of 12 matching algorithms (48), OM induced the same balance in baseline covariates as NNM, suggesting both NNM and OM are potential candidates for matching.

Strictly speaking, genetic matching (GM) is the extension of PSM (*Figure 8*). The algorithm considers not only PS, but also specifies covariates and determines a set of weights (i.e., variable importance) for PS and each covariate (53). In our data, for each patient, we recorded age, gender, and estimated the PS. If the weight of age was set to 2.5, while the weights of gender and PS were both 0.5, then the differences of age within the matched pairs will be paid 5 times more attention to than those of gender and PS. The PSM may be considered a limiting case of this method, in the sense that when all the weights of other covariates excepting PS are set to 0, GM will be equivalent to PSM.

It is obvious that weights must be meticulously designed to balance covariates. The GM uses an evolutionary search algorithm to determine sets of weights for each covariate, and achieves data balancing based on weighted covariates (53), which is why it is called "genetic". As shown

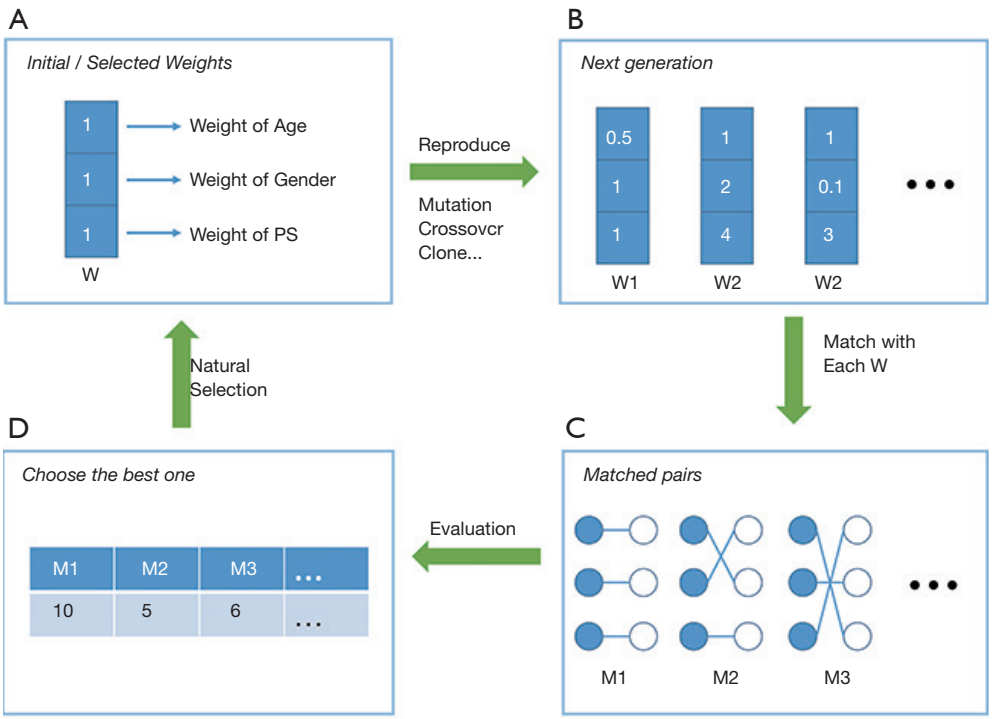

Figure 8 Process of genetic matching. PS, propensity score.

in *Figure 8*, GM starts from the initial sets of weights (the parents' genes). Second, GM randomly modifies some weights in these sets (mutation), and randomly recombines them with each other (genetic recombination) to generate the next generation of weights. Third, samples are matched with each set of weights to generate the same number of matching plans as individuals in the generation. Then, one solution which optimizes the data balance can be selected (natural selection). These steps are repeated until covariate balancing is achieved, or the maximum number of iterations is reached. Every candidate generation evolves, and the algorithm asymptotically converges towards an optimal solution.

In addition, several methods have an ambiguous relationship with PSM. For example, exact matching (EM) is a method that matches treated with controlled patients, using the same values of specified covariates. The method is used when only a few covariates may influence the treatment assignment. When the number of covariates increases, even though covariates are all binary, the number of possible matches grows exponentially, resulting in a poor matching result. Technically, EM is not a kind of PSM, but in Section 'What should be done if covariates are still unbalanced after matching?', we will explain how to use PSM and EM together.

Note that subclassification and full matching (FM) are often classified as matching methods (9,10). In fact, these methods are also regarded as stratifications based on PS, and are another series of methods that reduce selection bias in observational studies (1,3). Subclassification methods stratify patients into several groups, based on their estimated PS. Patients in one group will have similar PSs, and therefore differences in covariates can be ignored.

FM is a subclassification approach that optimally forms subclasses, which is the reason why it is also called optimal full matching (51,54,55). Each subclass contains 1 treated subject and 1 or more control subjects, or 1 control subject and 1 or more treated subjects. The subclass may be viewed as a matched set where 1 subject is matched to 1 or more counterparts, and thus can be construed as a matching method.

NNM with or without replacement

Matching with replacement means that after a patient from the treatment group is matched to another from the control group, the controlled patient is reintroduced to the control group for next-round matching; therefore, a patient in the control group may be used more than once (*Figure 7D*; red arrow). In contrast, matching without replacement means

that after matching, the matched pair from the candidate pool will be removed such that a participant in the control group will be used only once.

NNM with or without a caliper

Matching with or without a caliper determines whether the upper limit of the distance of PSs in a matched pair will be set or not. When the PS distributions are considerably different between the treatment and control groups, the distance of PSs in a matched pair may be large, as is the case in *Figure 7B*. Then, a caliper must be added into the matching step. Thus, for every patient, they can only be matched to patients of a PS within a limited range (*Figure 7E*). If eligible patients cannot be matched, the patient should be discarded.

While there is no gold-standard for maximal acceptable differences, a simulation study by Austin suggested that a caliper width of 0.2 of the standard deviation (SD) of the logit of PS be used, as this minimized the MSE of the estimated treatment effect, in a variety of settings (56). We suggest that (I) if the matching performance is poor (e.g., a few covariates are not balanced) matching can be conducted with a tighter caliper and (II) if matching is successful but the number of matched pairs is small, the caliper width can be broadened.

Pair matching, or other matching ratios

In pair matching, each treated patient matches to a single control in 1:1 ratio. Moreover, the ratio of controls to treated patients can be n:m (n and m are integers). For matching using variable ratios, each treated patient matches to, for example, at least one and at most four single controls (e.g., full matching). *Figure 7F* shows a 1:3 matching.

Matching with R

The function *matchit* of the *MatchIt* package implements both the estimation and matching steps of PSM. In Section 'How do I start my PSM analysis when there are missing values in my dataset?', we introduced PS estimations; here, we discuss matching.

Arguments

The use of *matchit* is as follows:

```
> matchit(formula, data, method = "nearest", distance = "logit",
distance.options = list(), discard = "none", reestimate = FALSE,
...)
```

The argument *method* specifies the method used to match patients, passing string values including "nearest" (nearest neighbor matching, the default option), "optimal" (optimal matching), "genetic" (genetic matching), "exact" (exact matching), "subclass" (subclassification), and "full" (full matching).

The *reestimate* is a logical argument. If TRUE, after we discard unmatched samples, and PS is re-estimated with the remaining samples. This will not occur if this argument is FALSE or missing. In addition, we tabulate additional arguments for each matching method (*Table 4*).

Examples

Next, we perform a simple match to see the outcome of *matchit*. It is worth noting that if you use optimal matching, the *optmatch* package must be installed, while genetic matching requires the *rgenound* package.

```
> m.out <- matchit(Smoke~x.Age+x.Gender,
data=data.complete,
distance='logit',
method='nearest',
replace=TRUE,
ratio=2)
```

This is an example of NNM. Matched pairs can be checked as follows.

```
> m.out$match.matrix
```

	1	2
1	"204"	"2"
6	"163"	"283"
10	"349"	"56"
12	"28"	"67"
20	"337"	"38"

Each row represents matched pairs of a patient in the treatment group. In each row, the first number indicates the serial number of the treated patient; numbers on the right are the controlled patients matched to this patient.

The value *discarded* records whether this patient was discarded.

Table 4 Additional arguments for each matching method in MatchIt

Method	Argument	Description	Default value
Subclassification	sub.by	The criteria for subclassification, including "treat" (the number of treatment units), "control" (the number of control units), "all" (the total number of units)	"treat"
	subclass	The number of subclass or a vector which create the quantiles of PS	6
Nearest	m.order	The order in which to pick treated patients, which can be "largest" (from the largest PS to the smallest), "smallest" (from the smallest to the largest), "random"	"largest"
	replace	Logical value indicating whether matching is doing with or without replacement, i.e. whether control patients can be matched more than once	FALSE
	ratio	The number of controlled patients to match to each treated one	1
	exact	A vector of variable names which will be exact matched. If exact is specified, only the patients with exactly the same value of covariates in exact will be considered and from them the one with closest propensity score will be matched.	NULL
	caliper	The number of standard deviations of difference which can be tolerated between PS of the matched pair.	0 (means no caliper)
Optimal	ratio	The number of controlled patients to match to each treated one	1
Full	min.controls	The minimum ratio of controlled patients to treatments that is to be permitted within a matched set, which is usually a whole number, or the reciprocal of a whole number	
	max.controls	The maximum number of controlled patients to treatments that is to be permitted within a matched set, which is usually a whole number, or the reciprocal of a whole number	
Genetic	ratio	The number of controlled patients to match to each treated one	1
	pop.size	The population size of each generation. It's the key tuning parameters to determine the matching effect. Generally, genetic matching with larger population size brings better results but takes more time.	100

PS, Propensity Score.

```
> m.out$discarded

    1     2     3     4     5     6     7     8     9    10    11    12
FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

What is a good match? Is my data balanced?

Balance diagnostics

After PS-based matching, a new dataset is generated; however, there is no guarantee of data balance. In this section, we investigate how to check balance in covariate distributions between treatment and control groups. Herein, we introduce and compare various balance diagnostic methods, followed by practice using R software. Note that we have not modified our PS model, and therefore some covariates remain unbalanced. In the next section, we discuss practical methods to cope with poor balance.

A direct and preferable method is to compare standardized mean differences (SMD) and variance ratios (VR) of covariates in the treatment and control groups (11). In the preceding context, we introduced SMD, which is an important metric that measures covariate differences between different groups, because it is not influenced by units of measure. Stuart (27) recommended that a covariate is balanced only when the absolute SMD value is <0.25, while Austin (3) proposed a more stricter criterion that it should be <0.1. The VR is the ratio of variance in the treatment and control groups. A VR close to 1.0 indicates a balance in the covariate, whereas a VR <0.5 or >2.0 are considered "too extreme" (57).

Significance testing is another widely used method, and the test results after matching are shown in many studies with PSM (58-60). However, it has been criticized that the increase of P value may be due to the decreasing sample size of the study population after PSM (11,61-63). Several studies have argued that we should use statistics that

account for a matched design, such as a paired t-test for continuous outcomes and McNemar's test for dichotomous outcomes in 1:1 matching design (14,64). This practice is rarely implemented or explicitly stated in the literature. Herein, rather than settling this dispute, we provide ways for researchers to conduct more in-depth research.

Graphical methods such as histograms, quantile–quantile plots, cumulative distribution functions, and love plots have also been proposed to show and check covariate distributions in a visual way.

It is worth nothing that goodness-of-fit measures, such as c-statistics, or area under the receiver operating characteristic curve (AUROC), indicate only the degree to which PS models discriminate between treated and untreated patients. However, these statistics provide no information as to whether PS models are correctly specified (3).

Balance diagnostics with R

SMD and VR

The summary command

The easiest way to calculate SMD and check covariate balance is to use the *summary* command on the output of *matchit*. This command provides an overall measure of the balance between treatment and control groups in the original and matched data set. Three options in the summary command may be used to check balance and respecify the PS model: (I) With the option *interactions* = TRUE, the output summarizes the balance of all squares and interactions of covariates in the matching procedure. Large differences between the treatment and original groups usually indicate that these items should be added into the PS model in order to achieve data balance. (II) The option *addlvariables* = TRUE shows the balance of covariates not used in the PS model. Similarly, these covariates should be included in the model if large differences are observed. (III) The option *standardize* = TRUE prints out standardized versions of the balance measures; i.e., the SMD where the mean difference is divided by the SD in the originally treated group. We suggest that *standardize* = TRUE should be set to compare the covariate balance in a scale-free way.

```
> m.out <- matchit(Smoke~x.Age+x.Gender,
data=data.complete,
distance='logit',
method='nearest',
replace=FALSE,
ratio=1)
> summary(m.out, standardize=TRUE)
Call:
matchit(formula = Smoke ~ x.Age + x.Gender, data = data.complete,
method = "nearest", distance = "logit", replace = FALSE, ratio = 1)

Summary of balance for all data:
           Means     Means SD Con- Std. Mean eCDF  eCDF  eCDF
           Treated Control trol    Diff.    Med   Mean  Max
Distance 0.5170  0.3858  0.1842 0.9050  0.2258 0.2069 0.4833
x.Age    47.0352 42.7619 19.6881 0.5247  0.1104 0.1306 0.3629
x.Gender 0.7832  0.4502  0.4981 0.8070  0.1665 0.1665 0.3330

Summary of balance for matched data:
           Means     Means SD Con- Std. MeaneCDF  eCDF  eCDF
           Treated Control trol    Diff.    Med   Mean  Max
Distance 0.5170  0.4380  0.1663 0.5448  0.1165 0.1617 0.4255
x.Age    47.0352 47.3035 18.6481 -0.0329 0.1206 0.1193 0.2439
x.Gender 0.7832  0.5610  0.4969 0.5386  0.1111 0.1111 0.2222

Percent balance improvement:
           Std. Mean eCDF Med  eCDF Mean  eCDF Max
           Diff.
Distance 39.8006   48.3806   21.8287    11.9680
x.Age    93.7217   -9.2460   8.6436     32.7902
x.Gender 33.2629   33.2629   33.2629    33.2629

Sample sizes:
           Control Treated
All        462     369
Matched    369     369
Unmatched  93      0
Discarded  0       0
```

The output from the *summary* command includes five parts: (I) the original assignment model call; (II) a data frame which shows the mean of the distance (PS) and covariates in the original treatment and control group, the SD in the control group, the (standardized) mean difference between the two groups, and the median, mean, and maximum differences in cumulative distribution functions (CDFs) for each covariate; (III) a data frame which contains the same statistical items as above, but for the matched data

Table 5 Baseline covariates of matched data summarized by *CreateTableOne* function.

Variables	Level	0	1	P	SMD
n		369	369		
x.Age, mean (SD)		47.30 (18.65)	47.04 (8.14)	0.8	0.019
x.Gender, n (%)	0	162 (43.9)	80 (21.7)	<0.001	0.487
	1	207 (56.1)	289 (78.3)		
CVD, n (%)	0	287 (77.8)	190 (51.5)	<0.001	0.572
	1	82 (22.2)	179 (48.5)		

The numbers 0 and 1 in the first row indicates whether to smoke, 0 means not while 1 means yes. The variable n is the number of patients, x.Age and x.Gender are the independent variables and CVD is the indicator of cardiovascular disease. SMD, standardized mean difference. SD, standard deviation.

set; (IV) percent balance improvement before and after matching; and (V) a data frame which counts samples, in all (original) data sets, which are matched, unmatched, or discarded.

As observed, after matching, age is balanced (0.0329<0.1) while gender is not (0.5386>0.1). Although discrete variables are allowed in the *Matchit* function to estimate PS, they are considered as continuous to examine balance in the summary (e.g., to calculate SMD). That means that the *summary* command indiscriminately calculates the SMDs of discrete and continuous variables, leading to bias for discrete ones.

CreateTableOne

When generating simulation data, we used the *CreateTableOne* function to calculate SMD. However, we do not recommend using it for matched data, because this function only takes matched data into account, and the mean difference is standardized (divided) by the SD in the matched data. After matching, it is probable that the standard deviation will be smaller in the matched data, so that the SMD may be larger than before matching, although the mean difference decreases (11). This means SMD is overestimated by *CreateTableOne*. A better way is to use the original SD, which is implemented by the *bal.tab* function in *cobalt* package. This will be introduced later.

Nonetheless, we created *Table 5* to summarize baseline covariates of matched data, using the *CreateTableOne* function. Firstly, we exported our matched data from *Matchit*.

```
> mdata <- match.data(m.out)
```

We then created a summary table.

```
> table5 <- CreateTableOne(vars = c('x.Age', 'x.Gender',
'CVD'),
data = mdata,
factorVars = c('x.Gender', 'CVD'),
strata = 'Smoke',
smd=TRUE)
> table5 <- print(table5,
smd=TRUE,
showAllLevels = TRUE,
noSpaces = TRUE,
printToggle = FALSE)
> table5
> write.csv(table5, file = "Table5_after_matching.csv")
```

Here, the SMD of *gender* is 0.487, which indicates this covariate has not been balanced.

The cobalt package

The function *bal.tab* in *cobalt* package provides a useful way to calculate SMD. After the package is installed, the following code can be used.

```
> library(cobalt)
> bal.tab(m.out, m.threshold = 0.1, un = TRUE)
Call
matchit(formula = Smoke ~ x.Age + x.Gender, data = data.com-
plete,
method = "nearest", distance = "logit", replace = FALSE, ratio = 1)
```

Balance measures

	Type	Diff.Adj	M.Threshold
Distance	Distance	0.5448	
x.Age	Contin.	-0.0329	Balanced, <0.1
x.Gender	Binary	0.2222	Not Balanced, >0.1

Balance tally for mean differences

	count
Balanced, <0.1	1
Not balanced, >0.1	1

Variable with the greatest mean difference

Variable	Diff.Adj	M.Threshold
x.Gender	0.2222	Not Balanced, >0.1

Sample sizes

	Control	Treated
All	462	369
Matched	369	369
Unmatched	93	0

Here, the argument *m.threshold* is a numeric value for the SMD threshold. This function shows which covariates are balanced (SMDs are under the threshold), which covariates are unbalanced, which covariate has the greatest mean difference, and the sample size of the matched data. The SMD of *age* is the same as that calculated by *summary (m.out)*, while the SMD of *gender* is calculated as appropriate for binary variables. When comparing SMDs of *gender* from *Table 5* with those here, we observed that the former SMD (0.487) was twice as much as the latter (0.2222). As previously mentioned, the mean difference was standardized by the SD in the different datasets, and thus the result from the function *bal.tab* was more credible.

The VR can be calculated and checked by setting the argument *v.threshold*, which is a numeric value for the VR threshold. As previously mentioned, a good VR is 0.5–2. In this function however, if VR is < 1, it will be converted to the inverse (1/VR). Therefore, we just need to set *v.threshold* = 2.

```
> bal.tab(m.out, v.threshold = 2)
```

Call

matchit(formula = Smoke ~ x.Age + x.Gender, data = data.complete,

method = "nearest", distance = "logit", replace = FALSE, ratio = 1)

Balance measures

	Type	Diff.Adj	V.Ratio.Adj	V.Threshold
Distance	Distance	0.5448	0.7605	
x.Age	Contin.	-0.0329	0.1908	Not Balanced, >2
x.Gender	Binary	0.2222		

Balance tally for variance ratios

	count
Balanced, <2	0
Not balanced, >2	1

Variable with the greatest variance ratio

Variable	V.Ratio.Adj	V.Threshold
x.Age	0.1908	Not Balanced, >2

Sample sizes

	Control	Treated
All	462	369
Matched	369	369
Unmatched	93	0

The VR is not calculated for the binary covariate gender, and age is not balanced according to the VR criterion.

Statistical significance

CreateTableOne

Now, we need to export out matched data, using the *match.data* command.

```
> mdata <- match.data(m.out)
> head(mdata)
```

	x.Age	x.Gender	Smoke	CVD	Distance	Weights
1	51	0	1	0	0.2583040	1

2	50	0	0	0	0.2545807	1
6	56	0	1	1	0.2774451	1
9	71	0	0	1	0.3397803	1
10	47	0	1	1	0.2436248	1
12	59	0	1	1	0.2893402	1

Here, the first 6 columns are the original columns from our data frame. The *Distance* column is the estimated PS, while the *Weights* columns is the weight each sample was given. These weights were calculated by well-designed rules, to ensure that the matched treated and control groups were similarly weighted. Because we chose 1:1 matching without replacement, all weights were assigned as 1. Otherwise, these weights were calculated according to how many times a unit was used to match. There will be problems if matching ratios are not 1:1, or we match with replacements. An example of this is as follows:

```
> m.out <- matchit(Smoke~x.Age+x.Gender,
data=data.complete,
distance='logit',
method='nearest',
replace=TRUE,
ratio=2)
> mdata <- match.data(m.out)
> mdata[135,]
```

	x.Age	x.Gender	Smoke	CVD	distance	weights
348	43	0	0	0	0.2295217	0.9268293

As can be seen, 1 participant (no. 348) was matched several times and had a weight of 0.927 in the exported data. Statistical tests with weights can be performed to check balance in this case by using weighted t-tests with the *wtd.t.test* function in the *weights* package, for instance.

```
> library(weights)
> age.treat <- mdata[mdata$Smoke==1, 'x.Age']
> weight.treat <- mdata[mdata$Smoke==1, 'weights']
> age.control <- mdata[mdata$Smoke==0, 'x.Age']
> weight.control <- mdata[mdata$Smoke==0, 'weights']
> wtd.t.test(x=age.control, y=age.treat,
weight=weight.control, weighty=weight.treat)
"Two-sample weighted t-test (Welch)"

$coefficients
```

t-value	df	p-value
0.1582711	324.6168812	0.8743416

```
$additional
```

Difference	Mean.x	Mean.y	Std. Err
0.1205962	47.1558266	47.0352304	0.7619600

Paired t-tests

If paired t-tests are to be used, the order of covariate values should be sorted, and the *t.test* function used with *paired* = TRUE. Here we provide a simple example:

```
> m.out <- matchit(Smoke~x.Age+x.Gender,
data=data.complete,
distance='logit',
method='nearest',
replace=FALSE,
ratio=1)
> match.treat <- data.complete[row.names(m.out$match.matrix),]
> matx <- m.out$match.matrix
> dim(matx) <- c(dim(matx)[1]*dim(matx)[2],1) # flatten matrix
> match.control <- data.complete[matx,]
> age.treat <- match.treat$x.Age
> age.control <- match.control$x.Age
> t.test(age.treat, age.control, paired=TRUE)
Paired t-test
data:  age.treat and age.control
t = -0.24334, df = 368, p-value = 0.8079
alternative hypothesis: true difference in means ≠ 0
95 % interval (CI): [-2.436365, 1.899779]
sample estimates:
mean of the differences: -0.2682927
```

Kolmogorov-Smirnov (KS) statistics

The function *bal.tab* in the *cobalt* package also provides KS tests, which compare the distribution of covariates between

treatment and control groups. The only requirement is to set the *ks.threshold* parameter.

Distribution visualization

The plot command

The *plot* command has three different options; the Q–Q plot (default), the jitter plot (set *type* = *'jitter'*), and the histogram (set *type* = *'hist'*). The Q–Q plots demonstrate deviations between empirical distributions of treated and control groups. The points in Q–Q plots lie on the y = x line. Larger deviations indicate greater differences between the distributions. Jitter plots show the PS distributions of unmatched treatment units, matched treatment units, matched control units, and unmatched control units. Every circle in the jitter plot represents a patient, and its size is proportional to its weight given by the *matchit* function. The distributional density of PS before and after matching is shown in histograms. The results are not shown in this tutorial.

```
> plot(m.out)
> plot(m.out, type = 'jitter')
> plot(m.out, type = 'hist')
```

The cobalt package

Variable distributions are visualized using the *bar.plot* function.

```
> bal.plot(m.out, var.name = 'x.Age', which = 'both',
grid=TRUE)
> bal.plot(m.out, var.name = 'x.Gender', which = 'both',
grid=TRUE)
> bal.plot(m.out, var.name = 'x.Age', which = 'both',
grid=TRUE, type="ecdf")
```

The density plot is displayed for continuous variables (*Figure 9A*), while for categorical variables, a distribution histogram is displayed (*Figure 9B*). Empirical cumulative density function plots are drawn by selecting *type* ="*ecdf*" (*Figure 9C*).

Love plots compare SMDs of covariates before and after matching, where two vertical dotted lines are drawn as the 0.1 threshold of SMD, and every point represents the SMD of a covariate before or after PSM adjustment. If a point lies between two lines, then the corresponding covariate has been balanced. Herein, we used the *love.plot* command of the *cobalt* package to draw *Figure 9D*. This subfigure represents the visualization of SMD results calculated by the *bal.tab* function in Section 'SMD and VR', where the SMDs of *age* and *gender* are –0.0329, and 0.2222, respectively.

```
> love.plot(bal.tab(m.out, m.threshold=0.1),
stat = "mean.diffs",
grid=TRUE,
stars="raw",
abs = F)
```

As shown in *Figure 9D*, the variable *x.Age* is balanced, while *x.Gender* is not. It is common to encounter unbalanced covariates when using PSM. In Section 'What should be done if covariates are still unbalanced after matching?', we provide several practical methods to deal with unsuccessful matching.

What should be done if covariates are still unbalanced after matching?

There are at least five methods to help PSM balance data.

(I) The first method is the most commonly used. It respecifies the model to estimate PS; e.g., it adds high-order or interaction terms, or changes to another modeling method. For example, in the preceding code, we used Smoke~x.Age+x.Gender to estimate PS, but it was not precise. Then, we added a quadratic term to our model, but unfortunately, the result was still poor.

```
> m.out <- matchit(Smoke~I(x.Age^2)+x.Age+x.Gender,
data=data.complete,
distance='logit',
method='nearest',
replace=FALSE,
ratio=1)
```

In real data, with many more variables, it may be difficult to find the correct formula to estimate PS. Researchers have to try different covariate combinations. The command *summary (m.out*, *interactions* = *TRUE*, *addlvariables* = *TRUE*, *standardize* = *TRUE*) provides balanced interaction information and high-order terms of all covariates, but as covariate numbers increase, the result table will be harder to comprehend. Also, a few methods

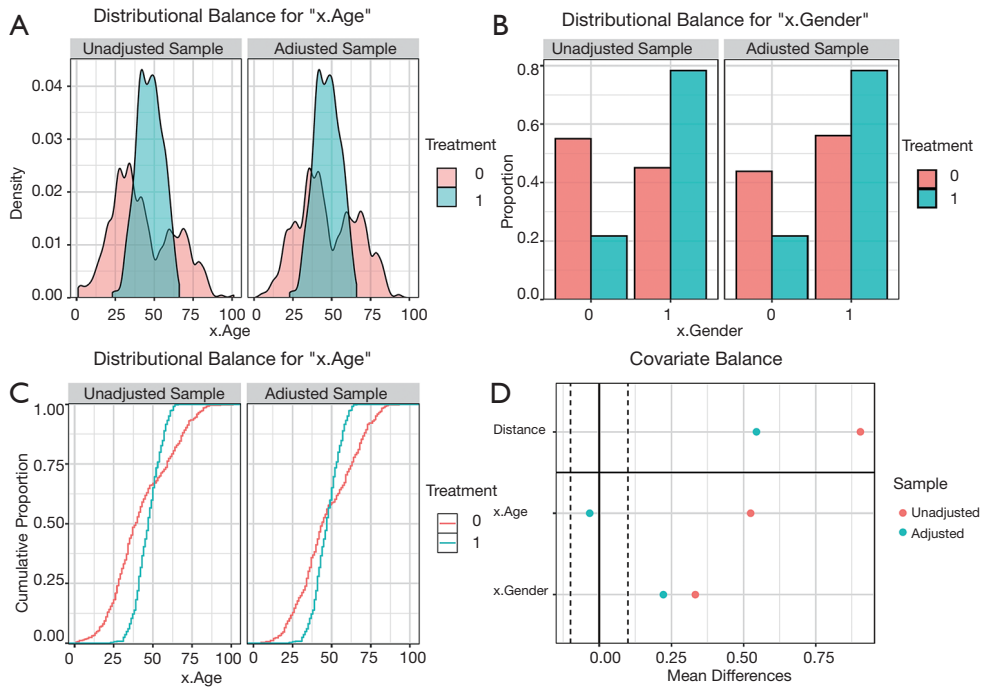

Figure 9 Visualization of distribution of covariates before and after propensity score matching.

serve as indicators for high-order relationships, such as *car* and *gvlma* package in R. In addition, most machine learning methods can automatically learn rules of data and fit complex relationships between variables, which is their major advantage.

(II) The second method is to change the matching method. For example, it is worth trying NNM with a tighter caliper, but this may shrink sample size.

```
> m.out <- matchit(Smoke~I(x.Age^2)+x.Age+x.Gender,
data=data.complete,
distance='logit',
method='nearest',
replace=FALSE,
caliper=0.1,
ratio=1)
> bal.tab(m.out, m.threshold=0.1)
Call
matchit(formula = Smoke ~ I(x.Age^2) + x.Age + x.Gender,
data = data.complete, method = "nearest",
distance = "logit", replace = FALSE,
caliper = 0.1, ratio = 1)

Balance measures
                  Type  Diff.Adj        M.Threshold
Distance      Distance    0.0736
I.x.Age.2.    Contin.     0.0572  Balanced, <0.1
x.Age         Contin.     0.0575  Balanced, <0.1
x.Gender      Binary      0.0349

Balance tally for mean differences
                                  count
Balanced, <0.1                    3
Not balanced, >0.1                0

Variable with the greatest mean difference
Variable      Diff.Adj    M.Threshold
x.Age         0.0575      Balanced, <0.1

Sample sizes
```

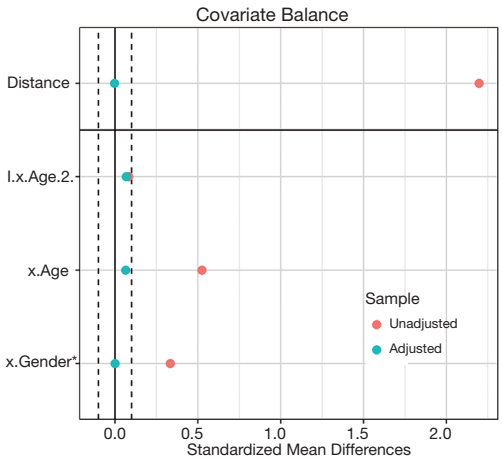

Figure 10 Love plot of propensity score matching using genetic matching based on the correct PS-estimate model. As can be seen, the PSM performance was much improved, when compared with before. We provide three reasons for this: (I) the former model to estimate PS was incorrect (the model was linear, while the correct one was quadratic). Herein, we used the correct model to estimate PS. (II) The matching algorithm of the latter was GM, which outperformed the ordinary matching algorithm in this case. (III) Here, fewer than 10 controlled individuals were matched to each treated patient, while before the situation in *Figure 9*, only 1:1 matching was implemented. GM, genetic matching; PSM, propensity score matching.

	Control	Treated
All	462	369
Matched	172	172
Unmatched	290	197

As can be seen, covariates are all balanced, but we lost almost half of the treated samples [197].

After adjustment, we can observe that using GM with *pop.size* =100 balanced our data with all treated samples matched.

```
> m.out <- matchit(Smoke~I(x.Age^2)+x.Age+x.Gender,
data=data.complete,
distance='logit',
method='genetic',
pop.size=100)
> bal.tab(m.out, m.threshold=0.1)
```

Balance measures			Balanced, <0.1
	Type	Diff.Adj	
Distance	Distance	-0.0032	
I.x.Age.2.	Contin.	0.0672	Balanced, <0.1
x.Age	Contin.	0.0642	Balanced, <0.1
x.Gender	Binary	0.0000	Balanced, <0.1

A love plot of SMD is shown in *Figure 10*. As can be seen, the PSM performance was much improved when compared with the one before. There are three reasons for this: (i) the former model to estimate PS was incorrect (the model was linear, while the correct one was quadratic). Herein, we used the correct model to estimate PS. (ii) The matching algorithm of the latter was GM, which outperformed the ordinary matching algorithm in this case. (iii) Here, each treated patient was matched to fewer than 10 controlled individuals, while in *Figure 9*, only 1:1 matching was implemented.

(III) The third method combines PSM with an exact matching method, by setting *exact=c('x.Gender')* in *matchit*. That is to say, when matching patients with similar PSs, some covariates (gender) must be equal. The effect of this method is not ideal; therefore, we do not use it.

(IV) The fourth method is to increase sample size, usually by collecting more data. In a scenario where there is 1 male and 1 female, regardless of the method used, they cannot be matched with a balanced gender. Therefore, with more data, matching should become easier.

(V) Lastly, residual differences in baseline covariates after PSM can be handled by conventional methods such as matching, stratification, regression adjustment, and so on (3,8,38). In other words, we balance our data successively by PSM and conventional methods (59).

How can the effects of treatment after matching be estimated?

What is the meaning of treatment effects in non-randomized experiments?

Under the counterfactual framework proposed by Rubin (65),

for every individual, there are two kinds of potential outcomes (under treatment or control conditions) before treatment assignment. Once the patient receives a treatment or control, researchers observe one of two outcomes, but the other is missing. To some extent, data analysis in observational studies is like the interpolation of missing outcomes.

Abadie and Imbens proposed an estimator of treatment effects appropriate for matching (66,67). In the matched data, all covariates have similar distributions between treatment and control groups. That means the matched patients are considered similar. Missing outcomes of the treatment group are interpolated using the mean outcomes of the matched controlled group, and vice versa. Suppose patients who received treatment have a mean outcome (A). After PSs are estimated and matching is performed, we interpolate outcomes of the treated patients under control conditions with matched patients' outcomes. If they had not received treatment, the expectation of their outcome should have been B. Thus, we compare the two mean outcomes (A *vs.* B) on matched data to estimate the average treatment effects on the treated.

How can treatment effects be estimated with PSM?

Generally speaking, two kinds of treatment effects are of concern: the average treatment effect (ATE) and the average treatment effect on the treated (ATT). Whether ATT or ATE is estimated depends on which is of interest and what matching method is used. Some believe that PSM only estimates ATT (27,68,69). For instance, if NNM is used, for each treated patient, the most suitable untreated patients are matched. By averaging the outcomes of the controlled which are matched, the missing potential outcomes for treated patients is actually estimated, and therefore ATT is estimated. However, in the official document of the *MatchIt* package (70), authors have provided an example where they estimated ATE using NNM and the *Zelig* package, which estimates causal effects by Monte Carlo simulation. A detailed explanation of this is beyond the scope of this tutorial. Also, if subclassification or FM with *matchit* function is used, patients are grouped into several subclasses based on their PSs, and thus ATE can be estimated.

When estimating effects, some of the literature suggests that outcomes be regressed on baseline covariates, instead of being calculated directly by comparing outcomes between treatment and control groups. This method is considered "double robustness" because the regression further cleans up the remaining small residual differences in covariates between the different groups after PSM (27) and has been shown to yield more accurate treatment effect estimates (71). McCaffrey *et al.* (38) note that it may be better to select just some of the covariates instead of all of them for inclusion in the regression model. However, this method may increase the chance of outcome model misspecification, since there are no universal guidelines on how many covariates should be included in the regression model.

There is also disagreement on whether matched data should be considered as matched or as independent samples. Although the samples are matched, some researchers argue that matching based on PS does not lead to correlations of matched participant outcomes, or indeed guarantee that covariate values are the same for matched pairs (27,72). It is believed that PS methods can separate the design element of an analysis from the analysis of outcome (73-75). From this separation, outcomes between treated and untreated participants can be directly compared. However, Austin (76) argues that using this method leads to patients with similar PSs being matched, resulting in similar distributions of covariates between treatment and control groups. Thus, Austin argues that methods which account for paired structure should be used to estimate the effects of treatment. This controversy requires further attention and research.

How to estimate treatment effects using R

The Zelig package

MatchIt does not provide functions for estimating treatment effects; however, Ho *et al.* (70) noted that any software for parametric statistical analyses may be used to analyze data previously preprocessed by *MatchIt*. They provided an example where they used the *Zelig* package to estimate treatment effects. Keller and Tipton (77) observed errors when they ran certain versions (4.2-1) of the *Zelig* package; thus, researchers should check the versions of their R packages before performing the PSM analysis.

The *Zelig* package estimates causal effects from Monte Carlo simulations, which is beyond the scope of this tutorial. Herein, we provide an analogous example; readers are directed to the *Zelig* official document.

```
> library(Zelig)
> z.out <- zelig(CVD~Smoke+x.Age+x.Gender, data = match.
data(m.out),
model = "ls")
```

```
> x.out <- setx(z.out, data = match.data(m.out, "treat"), cond =
TRUE)
> s.out <- sim(z.out, x = x.out)
> summary(s.out)
```

Other packages

Other models may also be used to estimate treatment effects. If 1:1 matching without replacement is implemented, the weights of each sample =1, and thus, the model does not require adjustment. However, the weighted versions of a specified model should be used, if the sample weights are $\neq 1$. This is an example of weighted logistic regression.

```
> mdata <- match.data(m.out)
> lr.att <- glm(CVD~Smoke+I(x.Age^2)+x.Age+x.Gender,
family=binomial,
data=mdata,
weights=mdata$weights)
> print(summary(lr.att))
Call:
glm(formula = CVD ~ Smoke + I(x.Age^2) + x.Age + x.Gender,
family = binomial,
data = mdata,
weights = mdata$weights)

Deviance residuals:
Min 1Q Median 3Q Max
-2.42201 -0.35121 -0.00846 0.38576 2.21623

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) -29.550323 5.518891 -5.354 8.58e-08 ***
Smoke 1 5.114775 0.622067 8.222 < 2e-16 ***
I(x.Age^2) -0.002964 0.001826 -1.624 0.10446
x.Age 0.640712 0.199404 3.213 0.00131 **
x.Gender 1.042130 0.340034 3.065 0.00218 **

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

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 742.59 on 575 degrees of freedom
Residual deviance: 311.89 on 571 degrees of freedom
AIC: 321.89

Number of Fisher scoring iterations: 7
```

Here the coefficient for smoking was 5.11, which means that for a patient in the treatment group, smoking behavior increased the risk for CVD (before the logistic transform) by 5.11 points.

What about unmeasured covariates related to treatment allocation and outcomes?

Sensitivity analysis

Although PSM balances data based on covariates, there may still be some confounding factors that have not been collected in the dataset, which may introduce hidden bias to the study (8). A core advantage of a well-implemented RCT is that it reduces as much as possible both obvious and hidden biases, while in observational studies, PSM and other analogous methods can only balance data based on recorded covariates.

Sensitivity analyses of observational studies help analyze the influence of unobserved confounders on the results (78). The origins of the approach can be traced back to a dispute on the relationship between smoking and lung cancer (79). It appears that no matter how many more smokers get lung cancer when compared to those who do not, champions of smoking argue for certain unobserved confounders, for example, genetic make-up related to both smoking behavior and risk of cancer. Therefore, it is necessary to measure the influence of hidden covariates on research in order to obtain even more convincing conclusions. In this section, we introduce three representative methods to deal with hidden biases.

From the perspective of statistical testing, Rosenbaum's sensitivity analysis (55) finds a influence threshold in a study. When the influence of unobserved confounders on the study exceeds the threshold, our conclusion, based only on observed covariates, is insignificant. In other words, when the hidden bias is small, our conclusions are still valid, but when the hidden bias is much larger, they can

be questioned. Actually, Rosenbaum's sensitivity analysis evaluates how sensitive a study is to hidden bias, and is frequently used after PSM (78). This method has now been developed into primal, dual, and simultaneous analyses. Their differences are based on what associations between unobserved confounders and the study are analyzed (80). In short, primal analysis focuses on associations between unobserved confounders and treatment assignments, dual analysis centers on associations between unobserved confounders and outcome, and simultaneous analysis accounts for both.

Here, we introduce what primal analysis is and how to implement it with an R package called *rbounds*. Suppose there are two patients, Bob and Bill. In a perfect study free of hidden bias, as long as Bob and Bill have the same observed covariates X, then they have the same probability of receiving treatment. Yet, if there is a hidden bias, even though Bob and Bill have the same X, their probabilities of receiving treatment are different, due to these different unobserved covariates. Thus, the boundary of the odds ratio (OR) of their probabilities of receiving treatment measures the degree of departure from a study that is free of hidden bias. That is to say, when the hidden bias in a study is greater, the treatment probabilities of patients with the same observed covariates are more different, and the deviation of the OR from 1 will be greater.

Primal sensitivity analysis sets $\tilde{A}$ as the boundary of OR and tests the validity of the conclusions with various $\tilde{A}$ levels. When $\tilde{A}$ becomes larger, the relationship between hidden bias and treatment assignment grows stronger. The analysis provides the confidence intervals of P values with each $\tilde{A}$. The larger $\tilde{A}$ with upper bound of P value <0.05 is observed, the more robust the study conclusion is with hidden bias. The readers are referred to Guo and Fraser's work (1) for further theoretical analysis.

However, the main disadvantage of Rosenbaum's sensitivity analysis is that these methods are designed only for 1:1 matching (78,81), thereby substantially limiting the methodology.

Vanderweele and Arah (81) proposed a general method for sensitivity analysis, which allowed for binary, ordinal, or continuous outcomes; categorical or continuous treatment; and categorical or continuous observed and unobserved variables. They proved a set of bias formulas which can be used to estimate the bias factor and obtain the true OR of the relationship between treatment and outcomes. However, if these formulas are applied, simplifying assumptions must be made based on knowledge or external data, and the specification of sensitivity parameters in the formulas. Liu *et al.* (78) introduced and summarized Rosenbaum's and Vanderweele and Arah's methods. Interested readers are directed to their article (78).

After this, Vanderweele further explored sensitivity analysis without assumptions, and introduced the E value (82,83). E value is defined as the minimum strength of association, on the risk ratio scale, that an unmeasured confounder must have with both the treatment and the outcome, conditional on the measured covariates, to fully explain away a specific treatment–outcome association. Although it is similar to Rosenbaum's methods, the E value makes no assumptions on whether unmeasured confounders are binary, continuous, or categorical. It makes no assumptions on how they are distributed, or on the number of confounders, and it can be applied to several common outcome types in observational research studies. To facilitate these sensitivity analyses, Mathur *et al.* provided an R package ("EValue") and an online E value calculator (https://mmathur.shinyapps.io/evalue/) (84).

Propensity score calibration (PSC) is another method that deals with hidden bias, and is similar to regression calibration (85). For example, if you study the effects of norepinephrine on mortality using PSM, you may already have a large dataset to support your research; however, you may speculate that central venous pressure (CVP) is a potential confounding variable, but it is not measured in your dataset. In this case, another small dataset, containing CVP and all covariates in the large dataset, can be collected. The steps of PSC are as follows: (I) estimate raw PS in your main dataset; (II) estimate PS in your validation dataset with the same set of covariates as the main one (denoted as PS1), and with an extended covariate set including CVP as PS2; (III) model the extended PS2 as a function of PS1; and (IV) calibrate your raw PS on the main dataset, with the function mentioned above (III).

How to implement sensitivity analysis with R

In this section, we show how to implement sensitivity analysis using R. Our treatment assignment and outcomes were related to gender and patient age, while here we use only age to estimate PS, and gender is the unobserved confounding variable. There are no missing values in our data. We divided our data into two datasets, using an 8:2 ratio to implement the PSC. Note that to simplify our tutorial, we did not balance the covariates, which is not permitted in actual research.

```
> set.seed(2020)
> x.Gender <- rep(0:1, c(400,600)) # 400 females and 600 males
> x.Age <- round(abs(rnorm(1000, mean=45, sd=15)))
> z <- (x.Age - 45) / 15 - (x.Age-45) ^ 2 / 100 + 2 * x.Gender
> tps <- exp(z) / (1+exp(z)) # The true PS
> Smoke <- as.numeric(runif(1000) < tps)
> z.y <- x.Gender + 0.3*x.Age + 5*Smoke - 20
> y <- exp(z.y) / (1+exp(z.y))
> CVD <- as.numeric(runif(1000) < y)
> data.raw <- data.frame(x.Age, x.Gender, Smoke, CVD)
> sub <- sample(1:nrow(data.raw),round(nrow(data.raw)*8/10)) # 8:2
# without gender information
> data.main <- data.raw[sub, c("x.Age","Smoke","CVD")]
# with gender information
> data.val <- data.raw[-sub, c("x.Age","x.Gender","Smoke","CVD")]
> head(data.main)
```

	x.Age	Smoke	CVD
144	15	0	0
415	42	1	1
444	68	0	0
534	53	1	1
612	50	1	1
599	59	1	1

Rosenbaum's primal sensitivity analysis

The R package *rbounds* provides functions to implement Rosenbaum's sensitivity analysis for binary, continuous, or ordinal outcomes. First, this package must be installed and uploaded. The package directly analyzes the *match* object from the *Matching* package. Because we used *MatchIt* to implement PSM, we must obtain matched outcomes prior to running the functions of *rbounds*, using the following code:

```
> m.pairs <- cbind(data.main[row.names(m.out$match.matrix), 'CVD'],
data.main[m.out$match.matrix, 'CVD'])
```

match.matrix records the matching relationship between the treatment and control groups. The *cbind* function generates a matrix where the first column is the treated outcomes, and the second column is the controlled outcomes.

In the *rbounds* package, the functions, *binarysens* and *psens* are commonly used. The former is used for binary outcomes, and the latter for continuous or ordinal outcomes. Here, we used *binarysens*. This function has four arguments: *x*, *y*, *Gamma*, and *GammaInc*. The argument *x* is the number of pairs where the controlled patients have outcomes, but the treated patients do not. The argument *y* is the number of pairs where the treated patients have outcomes, but the controlled patients do not. The argument *Gamma* is the maximum $\tilde{A}$ tested in our study. The argument *GammaInc* is the step size of $\tilde{A}$, when it changes from small to large. We must count the number of discordant pairs, followed by the function, *binarysens*.

```
> library(rbounds)
> library(MatchIt)
> m.out <- matchit(Smoke~x.Age,
data=data.main,
distance='logit',
method='optimal',
replace=FALSE,
ratio=1)
> m.pairs <- cbind(data.main[row.names(m.out$match.matrix), 'CVD'],
data.main[m.out$match.matrix, 'CVD'])
> x <- sum((m.pairs[,1]==FALSE) & (m.pairs[,2]==TRUE))
> y <- sum((m.pairs[,1]==TRUE) & (m.pairs[,2]==FALSE))
> binarysens(x=x, y=y, Gamma = 15, GammaInc = 2)

Rosenbaum's sensitivity test

Unconfounded estimate .... 0
```

Gamma	Lower Bound	Upper Bound
1	0	0.00000

2	0	0.00000
3	0	0.00000
4	0	0.00000
5	0	0.00000
6	0	0.00002
7	0	0.00017
8	0	0.00081
9	0	0.00264
10	0	0.00662
11	0	0.01378
12	0	0.02496
13	0	0.04066
14	0	0.06099
15	0	0.08570

Here, the value of *x* is 4 while that of *y* is 126. The first column represents the different values of Gamma Ã, while the second and third are the lower and upper bound of the P value, respectively. This P value is calculated by testing whether or not our result is significant. Lower bounds are recorded as 0, because they are extremely small (<0.00001). As observed, the upper bound <0.05, when Gamma <14. This means that even if one patient may be 13 times as likely to be treated as another with the same recorded covariates, due to the hidden bias, the treatment still contributes significantly to the outcome.

If you need to manage continuous or ordinal outcomes, the following code can be run:

```
> psens(x=m.pairs[,1], y=m.pairs[,2], Gamma = 6, GammaInc
= 1)
```

Here, *x* is the treatment group outcomes, while *y* is the control group outcomes.

E value

Firstly, the corresponding package must be installed and uploaded.

```
> install.packages('EValue')
> library('EValue')
```

The *evalue* function is the core element of this package, which computes E values for unmeasured confounding factors. The argument *est* is the effect estimate that is observed but is suspected to be biased. The estimate can be a particular type of effect measure, e.g., risk ratio (RR), odds ratio (OR), hazard ratio (HR), risk difference (RD), linear regression coefficient (OLS), or mean standardized difference (MD). For example, if the RR estimate was 0.80 and the lower and upper bound were 0.70 and 0.90, respectively, then the E value would be calculated as follows:

```
> evalue(RR(0.80), lo=0.70, hi=0.90)
```

	Point	Lower	Upper
RR	0.800000	0.7	0.900000
E-values	1.809017	NA	1.462475

This package is relatively simple, except for the explanation for E value. Interested readers are directed to the following website: https://cran.r-project.org/web/packages/EValue/EValue.pdf.

PSC

First, raw PS is estimated with our main data.

```
> model1 <- glm(Smoke~x.Age, data=data.main,
family="binomial")
> PS.raw <- predict(model1, newdata=data.main,
type="response")
```

Second, PS1 is estimated with the same set of covariates on the validation data, and PS2 is estimated with gender being added.

```
> model2 <- glm(Smoke~x.Age, data=data.val,
family="binomial")
> PS1 <- predict(model2, newdata=data.val, type="response")
> model3 <- glm(Smoke~x.Age+x.Gender, data=data.val,
family="binomial")
> PS2 <- predict(model3, newdata=data.val, type="response")
```

Then, PS2 is regressed on PS1.

```
> data.reg <- data.frame(PS1, PS2)
> model.cal <- lm(PS2~PS1, data=data.reg)
```

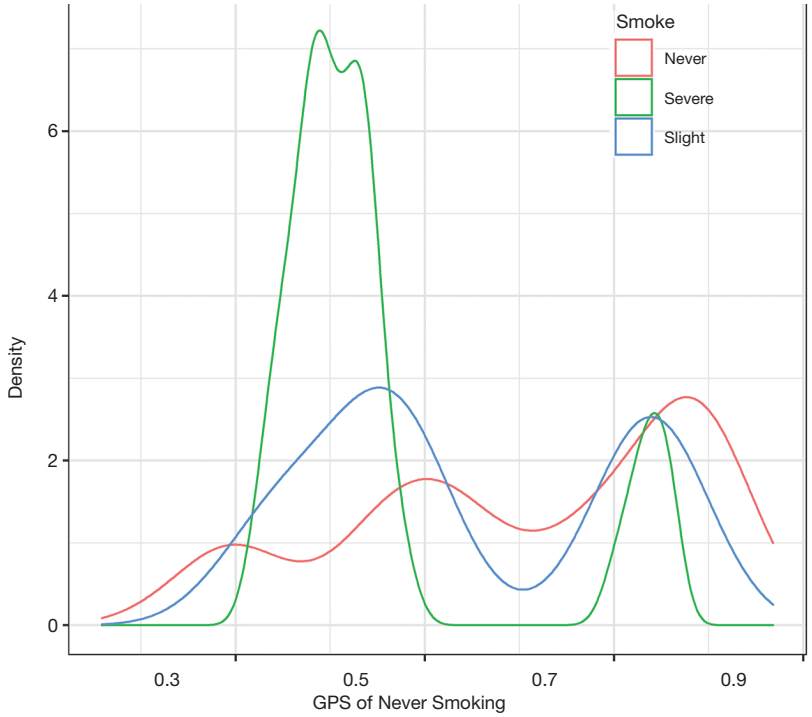

Figure 11 Density plot of the distributions of probability of never smoking estimated with multinomial logistic regression on different groups.

Finally, our raw PS is calibrated.

```
> PS.cal <- predict(model.cal, x=PS.raw)
```

How does PSM handle multiple or continuous treatments?

The PS is the conditional probability of receiving treatment, given that covariates are observed. It effectively balances covariates between treatment and control groups, when the treatment assignment is dichotomous. However, on account of the complexity of medical research, researchers sometimes have to face comparisons of multiple treatments, or analyze the effects of different treatment doses (60,86,87). The original PS definition and related analytical methods are no longer applicable, and thus we must extend the concept of PS for more complex settings.

GPS for multiple treatments

Definitions

Now, we try to estimate and compare the effects of three different treatments: treatment A, B, and C (or control). Paired comparisons with PSM, which include separately applying PSM to A–B, A–C, and B–C, is a direct analytical method (60). However, because some patients are discarded during matching, the paired comparisons are actually based on different patient groups. Therefore, it is possible that a researcher using pairwise PSM observes that A is better than B, B is better than C, and C is better than A (86). Furthermore, when treatment numbers increase, there are squared-level number of pairs to analyze. Another way is to determine a reference treatment, e.g., C, then match A *vs.* C, and B *vs.* C (88,89). This is an ingenious method when the influence of treatment C is focused on in the study. However, the comparison of A and B is challenging in this study design.

Rassen *et al.* (90) proposed a three-way matching algorithm which can simultaneously match participants from 3 groups. This was based on GPS and online codes. Herein, we introduce GPS and show how to estimate GPS using multinomial logistic regression with R.

Conceptually, GPS is the probability that a patient receives 1 of several possible treatments, conditional on

his or her covariates. It is usually estimated by multinomial logistic regression. This prediction model chooses 1 treatment (for example, C) as the baseline treatment (reference). Next, the logarithm of ratio of the probabilities of receiving treatment A (or B) and reference treatment is assessed for every patient. Lastly, the model provides a matrix of probabilities for each patient to receive each treatment. Equally, CART, random forests, and other machine learning methods can also predict the probability of receiving one treatment rather than the others (8,38). Currently, machine learning–augmented PS is a promising research area.

The estimation of GPS for multiple treatments with R

First, we must modify our dataset for multiple treatments. We group patients into three categories according to their smoking behavior—severe, slight, and never—and assume that older men are more prone to excessive smoking, and that increased smoking leads to a higher risk of CVD. The simulation data are generated with the following code:

```
> set.seed(2020)
> x.Gender <- rep(0:1,c(400,600))
> x.Age <- round(abs(rnorm(1000, mean=45, sd=15)))
> z <- (x.Age - 45) / 15 - (x.Age-45) ^ 2 / 100 + 2 * x.Gender
> h <- exp(z) / (1+exp(z))
> Smoke <- ifelse(runif(1000) < h,
ifelse(runif(1000) < h, 'Severe', 'Slight'),
'Never')
> Smoke_effect <- c('Severe'=5, 'Slight'=3, 'Never'=0)
> z.y <- x.Gender + 0.3*x.Age + Smoke_effect[Smoke] - 20
> y <- exp(z.y) / (1+exp(z.y))
> CVD <- (runif(1000) < y)
> data <- data.frame(x.Age, x.Gender, Smoke, CVD)
> head(data)
```

	x.Age	x.Gender	Smoke	CVD
1	51	0	Slight	TRUE
2	50	0	Never	FALSE
3	29	0	Never	FALSE
4	28	0	Never	FALSE
5	3	0	Never	FALSE
6	56	0	Slight	FALSE

Next, we choose the never-smoking group as a reference, and use the *multinom* function in the *nnet* package to estimate GPS.

```
> library(nnet)
> data$Smoke_with_ref <- relevel(data$Smoke, ref = "Never")
> mnl <- multinom(Smoke_with_ref ~ x.Age + x.Gender, data =
data)
> summary(mnl)
Call:
multinom(formula = Smoke_with_ref ~ x.Age + x.Gender, data =
data)

Coefficients
```

	(Intercept)	x.Age	x.Gender
Severe	-2.940667	0.02591537	1.8725196
Slight	-2.337776	0.01136380	0.6705426

```
Std. Errors:
```

	(Intercept)	x.Age	x.Gender
Severe	0.2918849	0.005114330	0.1762513
Slight	0.3381404	0.006548734	0.2037042

```
Residual Deviance: 1748.791
AIC: 1760.791
```

This function does not calculate P values for coefficients, but we can complete this ourselves.

```
> z <- summary(mnl)$coefficients / summary(mnl)$standard.
errors
> z
```

	(Intercept)	x.Age	x.Gender
Severe	-10.074749	5.067208	10.624144
Slight	-6.913624	1.735266	3.291746

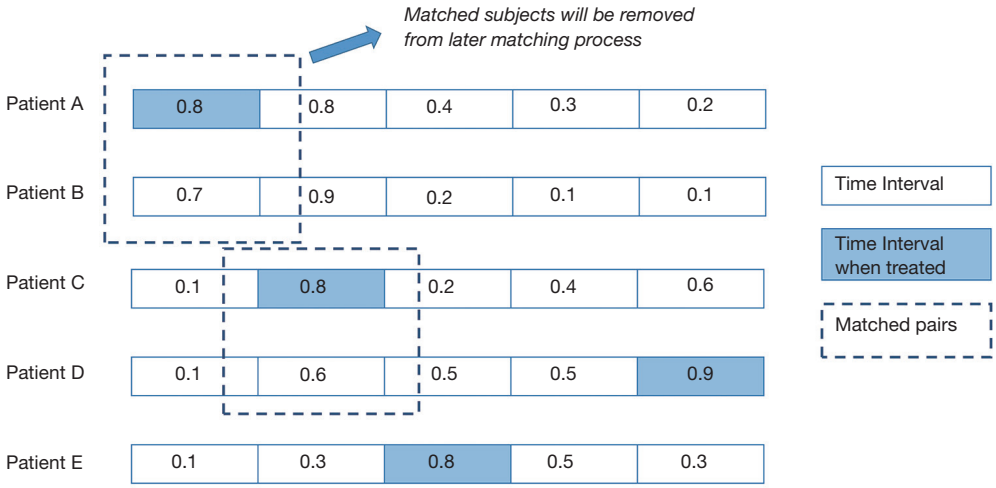

Figure 12 Sequential matching process with time-dependent PS.

```
> p <- (1 - pnorm(abs(z), 0, 1))*2
> p
```

	(Intercept)	x.Age	x.Gender
Severe	0.000000e+00	4.036940e-07	0.0000000000
Slight	4.724221e-12	8.269366e-02	0.0009956749

Then, we use the *predict* function to derive the probability of receiving one certain treatment. The *mnl* argument is the multinomial logistic regression model. The "probs" argument means the function returns a probability value (0–1) instead of the predicted class.

```
> prob <- data.frame(predict(mnl, data, 'probs'))
> head(prob)
```

	Never	Severe	Slight
1	0.7296867	0.14455104	0.12576226
2	0.7334412	0.14157781	0.12498099
3	0.8024125	0.08988239	0.10770513
4	0.8052441	0.08789204	0.10686390
5	0.8643103	0.04935363	0.08633608
6	0.7102605	0.16016847	0.12957099

After this, we visualize the GPS distribution for different groups using the *ggplot2* package. The following example is a density plot, which shows the GPS distribution of never smoking for different groups (*Figure 11*).

```
> ggplot(data=data, aes(x=prob$Never, colour=data$Smoke))
+
geom_density() +
xlab("GPS of Never Smoking")
```

GPS for continuous treatments

Definitions

When we study the effects of various patient treatment dosages, the treatment assignment indicator is a continuous variable, e.g., the dosage of a new drug A. In a continuous setting, it is unlikely that two units will have exactly the same level of treatment, so it is unfeasible to match pairs with the same exposure level (91). Hirano and Imbens (92) proposed an extension to the counterfactual framework. In their approach, a set of potential outcomes for each patient is considered; however, only 1 value of the set is observed. Their method of estimating the effects of continuous treatments consists of three stages: (I) the treatment dose is regressed on a function of covariates with a linear regression model, assuming the residual is normally distributed, and the estimated GPS is the probability of the observed residual (such a description is not precise,

but easy to understand); (II) the outcomes are modeled as a function of the treatment dose and GPS; and (III), for each level of treatment of interest, the GPS is estimated using the model in stage 1, and subsequently the average potential outcome is obtained using the model in stage 2. After the potential outcome for multiple different treatment doses is estimated, the dose-response function is obtained. In addition to this approach, a variety of methods have been proposed to estimate the dose-response function, based on GPS (68,92-95), but to the best of our knowledge, these methods are not based on matching. Wu *et al.* (91) proposed a one-to-M nearest neighbor caliper matching procedure with replacement. For each unit, the algorithm finds the *m* observed unit that is both close to its exposure level and the corresponding estimated GPS. They kindly provided an R package for implementing matching on GPS with continuous exposures. Readers can visit the website at https://github.com/wxwx1993/GPSmatching. Herein, we introduce GPS estimation with R.

Estimating GPS for continuous treatments with R

We extended our simulation data to study the effects of different durations of smoking on the cardiovascular system. Years of smoking for each patient were generated by the following code, with the risk of CVD being assumed to be proportional to the years of smoking (YOS).

```
> Set.seed(2020)
> x.Gender <- rep(0:1,c(400,600))
> x.Age <- round(abs(rnorm(1000, mean=45, sd=10)))
> YOS <- x.Gender * 5 + x.Age / 5 + round(rnorm(1000,
mean=0, sd=2)) # Years of smoking
> z.y <- x.Gender + 0.3 * x.Age + 0.01 * YOS^2  - 15
> y <- exp(z.y) / (1+exp(z.y))
> CVD <- (runif(1000) < y)
> data <- data.frame(x.Age, x.Gender, YOS, CVD)
> head(data)
```

	x.Age	x.Gender	YOS	CVD
1	49	0	9.8	TRUE
2	48	0	7.6	TRUE
3	34	0	6.8	FALSE
4	34	0	6.8	FALSE
5	17	0	1.4	FALSE
6	52	0	11.4	TRUE

Here, YOS is generated as the linear combination of age and gender, with a normally distributed residual.

First, the YOS linear regression model and covariates are fitted.

```
> Lm.dose <- lm(formula = YOS~x.Age+x.Gender, data=data)
> summary(lm.dose)
Call:
lm(formula = YOS ~ x.Age + x.Gender, data = data)

Residuals:
Min        1Q    Median      3Q     Max
-7.0811 -1.0850 -0.0071  1.0581  7.0772

Coefficients:
             Estimate  Std. Error  t value Pr(>|t|)
(Intercept) -0.309749  0.294175   -1.053   0.293
x.Age        0.207663  0.006184   33.579  <2e-16 ***
x.Gender     4.956670  0.130745   37.911  <2e-16  **

Signif. Codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.025 on 997 degrees of freedom
Multiple R-squared: 0.7211, Adjusted R-squared: 0.7206
F-statistic: 1289 on 2 and 997 DF, p-value: < 2.2e-16
```

Then, the GPS is estimated by the normal distribution.

```
> Data$GPS = dnorm(data$YOS, mean=lm.dose$fitted,
sd=sd(data$YOS))
```

How to apply PSM with time-dependent covariates and treatments

In clinical settings, especially where variables are continuously monitored, the probability of receiving a treatment is based on the real-time value of variables. Treatment effects are usually dependent on when a patient

receives a treatment. Although a binary indicator can be generated to indicate whether a patient is treated or not throughout their admission, this conventional method leads to biased estimated treatment effects (96), because it does not account for the time factor of treatment.

Time-dependent PSM helps analyze the true effects of treatment (97). First, time-dependent PS is estimated for each patient at every time point, using a Cox proportional hazards regression model on time-fixed and time-varying covariates. Then, simultaneous matching or sequential matching is applied to match patients based on time-dependent PSs. Lastly, Cox regression or other models for time-to-event analysis are applied to the matched data.

As implied by the name, simultaneous matching compares all possible combinations of matched pairs at once, and performs one matching for all patients. However, this method assumes little association between future covariates and current treatment decisions, because future covariate information may be disclosed during the matching process. In contrast, no information leaks are allowed in sequential matching. As shown in *Figure 12*, the entire time is divided into multiple, equally spaced intervals (e.g., 5). At each interval, patients treated at that time form the treatment group, while patients untreated until that time form the control group (even though they may receive treatment after that). From the first interval to the last, treated patients are matched to untreated patients, using optimal matching. Once matched, patients are removed from groups, and they do not participate in subsequent matching processes. Zhang *et al.* (96) developed the function *TDPSM* for time-dependent PSM, and they have shared their R code.

Discussion

As previously discussed, the two main advantages of PSM are dimension reduction and study design separation. Some conventional methods, for example, matching or stratification, group units with the same or similar covariate values to eliminate confounding. However, when covariate numbers increase, matching or stratification becomes much more difficult. Take research data on 1,000 patients as an example. To eliminate bias due to gender, each patient may be matched to 500 patients of the opposite gender. Also, if we consider age as a confounding factor, the data can be divided into 10 equal groups by age, after which each patient can be matched to about 50 others. Furthermore, race is accounted for, and thus for each patient only 10 are matched. If we keep adding balancing covariates, many patients will not be matched. When compared with matching or stratification, all covariates requiring balancing are integrated into 1 score in PSM. This integration greatly reduces matching difficulties due to covariate dimensions.

In addition, covariate balancing and effect estimating are separated in PSM. As *Figure 2* illustrates, it is possible to check whether or not covariates are balanced between treatment and control groups in the fourth step, and estimate treatment effects later. In the regression adjustment method, which models outcomes as functions of both treatment assignments and covariates, it is difficult to assess whether the model has been correctly specified. Goodness-of-fit measures, such as R^2 and MSE do not test whether the model has balanced the data or not (3).

While these advantages make PS methods popular, there are several limitations. Firstly, the greatest drawback to PS methods is that they cannot account for hidden bias, which is similar to other methods which derive causal inference from observational research. Most balancing methods can only adjust data based on recorded covariates. However, it is inevitable that there is unobservable, immeasurable, or unrecorded bias. Although sensitivity analyses evaluate study robustness or calibrate PS with validation datasets, there is still room for improvement. In contrast, in a well-structured RCT, obvious or hidden bias is reduced as much as possible, and therefore, PSM and analogous methods are no replacement for randomization. However, PSM can be used to scrutinize relationships between treatments and outcomes prior to the commencement of an RCT. On account of the limited medical resources, PSM and other balancing methods are still of great value.

Secondly, sample size is often diminished in PSM, as patients with no matches are often discarded. This may negatively affect final study conclusions, especially when small sample sizes are used. When several thousand samples are included, the data loss is negligible.

Thirdly, PSM estimates average treatment effects, but treatment heterogeneity is not considered. As we enter the era of precision medicine, personalized therapeutic strategies are being increasingly emphasized. However, personalized medicine is still a long way from being fully realized.

In this tutorial, we introduced the concept and framework of PSM, showed how to implement PSM with R, demonstrated GPS for multiple and continuous treatments, and illustrated time-dependent PSM. However, the statistical basics of PSM were not covered in this tutorial. Besides, there are many other

R packages related to PS, and other methods based on PS, such as weighting or subclassification, which were not included in this tutorial. In light of the rapid growth of causal inference analysis in observational studies, more comprehensive tutorials on PS methods should be developed.

Acknowledgments

We thank International Science Editing (http://www.internationalscienceediting.com) for its linguistic assistance. Our gratitude also goes to the editors from AME Publishing Company and the anonymous reviewers for their careful work and thoughtful suggestions that have helped improve this paper substantially. We sincerely thank the English Language Editor J. Jones and the Quality Control Editor J. Gray from the AME Publishing Company for editing this manuscript. In addition, Qin-Yu Zhao wants to thank, in particular, the care and support from Le-Ping Liu during the preparation of this manuscript. I love you.

Funding: This article was supported by grants from the Research Funds of Shanghai Municipal Health Commission (No. 2019ZB0105), Natural Science Foundation of Shanghai (No. 20ZR1411100), Program of Shanghai Academic/Technology Research Leader (No. 20XD1421000), National Natural Science Foundation of China (No. 82070085), Clinical Research Funds of Zhongshan Hospital (No. 2020ZSLC38 and No. 2020ZSLC27), and Smart Medical Care of Zhongshan Hospital (No. 2020ZHZS01).

Footnote

Conflicts of Interest: All authors have completed the ICMJE uniform disclosure form (available at http://dx.doi.org/10.21037/atm-20-3998). GWT served as an unpaid section editor of the *Annals of Translational Medicine* from October 2019 to September 2020. The other authors have no other conflicts of interest to declare

Ethical Statement: The authors are accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

References

1. Guo S, Fraser MW. Propensity score analysis: Statistical methods and applications. SAGE Publications; 2014.
2. Cook TD, Campbell DT, Shadish W. Experimental and quasi-experimental designs for generalized causal inference. Boston: Houghton Mifflin; 2002.
3. Austin PC. An introduction to propensity score methods for reducing the effects of confounding in observational studies. Multivariate Behav Res 2011;46:399-424.
4. Cochran WG, Chambers SP. The planning of observational studies of human populations. J R Stat Soc Ser A 1965;128:234-66.
5. Tein JY, Mazza GL, Gunn HJ, et al. Multigroup propensity score approach to evaluating an effectiveness trial of the New Beginnings Program. Eval Health Prof 2018; 41:290-320.
6. Rosenbaum PR, Rubin DB. The central role of the propensity score in observational studies for causal effects. Biometrika 1983;70:41-55.
7. Mitra R, Reiter JP. A comparison of two methods of estimating propensity scores after multiple imputation. Stat Methods Med Res 2016;25:188-204.
8. Leite W. Practical propensity score methods using R. SAGE Publications; 2016.
9. Randolph JJ, Falbe K, Manuel AK, et al. A step-by-step guide to propensity score matching in R. Pract Assess Res Eval 2014;19:18.
10. Zhang Z. Propensity score method: a non-parametric technique to reduce model dependence. Ann Transl Med 2017;5:7.
11. Zhang Z, Kim HJ, Lonjon G, et al. Balance diagnostics after propensity score matching. Ann Transl Med 2019;7:16.
12. Yao XI, Wang X, Speicher PJ, et al. Reporting and guidelines in propensity score analysis: a systematic review of cancer and cancer surgical studies. J Natl Cancer Inst 2017;109:djw323.
13. Malla L, Pererasalazar R, Mcfadden E, et al. Handling missing data in propensity score estimation in comparative effectiveness evaluations: a systematic review. J Comp Eff Res 2018;7:271-9.

14. Zakrison TL, Austin P, McCredie V. A systematic review of propensity score methods in the acute care surgery literature: avoiding the pitfalls and proposing a set of reporting guidelines. Eur J Trauma Emerg Surg 2018;44:385-95.
15. Austin PC. Type I error rates, coverage of confidence intervals, and variance estimation in propensity-score matched analyses. Int J Biostat 2009;5:Article 13.
16. Austin PC, Grootendorst P, Anderson GM. A comparison of the ability of different propensity score models to balance measured variables between treated and untreated subjects: a Monte Carlo study. Stat Med 2007;26:734-53.
17. Austin PC, Mamdani MM. A comparison of propensity score methods: a case-study estimating the effectiveness of post-AMI statin use. Stat Med 2006;25:2084-106.
18. Rubin DB. On principles for modeling propensity scores in medical research. Pharmacoepidemiol Drug Saf 2004;13:855-7.
19. Huang F, Du C, Sun M, et al. Propensity score matching in SPSS. Nan Fang Yi Ke Da Xue Xue Bao 2015;35:1597-601.
20. Enders CK. Applied missing data analysis. Guilford Press; 2010.
21. Zhang Z. Missing data imputation: focusing on single imputation. Ann Transl Med 2016;4:9.
22. Groenwold RHH, White IR, Donders ART, et al. Missing covariate data in clinical research: when and when not to use the missing-indicator method for analysis. CMAJ 2012;184:1265-9.
23. Malarvizhi R, Thanamani AS. K-nearest neighbor in missing data imputation. Int J Eng Res Dev 2012;5:5-7.
24. Pantanowitz A, Marwala T. Missing data imputation through the use of the Random Forest Algorithm. Advances in Computational Intelligence. Springer; 2009:53-62.
25. Tang F, Ishwaran H. Random forest missing data algorithms. Stat Anal Data Min 2017;10:363-77.
26. Leyrat C, Seaman SR, White IR, et al. Propensity score analysis with partially observed covariates: How should multiple imputation be used? Stat Methods Med Res 2019;28:3-19.
27. Stuart EA. Matching methods for causal inference: A review and a look forward. Stat Sci 2010;25:1-21.
28. Leite W, Aydin B. editors. A comparison of methods for imputation of missing covariate data prior to propensity score analysis. Washington, DC: American Education Research Association Conference, 2016.
29. Choi J, Dekkers OM, Cessie SL. A comparison of different methods to handle missing data in the context of propensity score analysis. Eur J Epidemiol 2019;34:23-36.
30. Puma M, Olsen RB, Bell SH, et al. What to Do when Data Are Missing in Group Randomized Controlled Trials. NCEE 2009:0049.
31. Lee BK, Lessler J, Stuart EA. Improving propensity score weighting using machine learning. Stat Med 2010;29:337-46.
32. Pearl J. Causal diagrams for empirical research. Biometrika 1995;82:669-88.
33. Pearl J, Mackenzie D. The book of why: the new science of cause and effect. Basic Books; 2018.
34. Brookhart MA, Schneeweiss S, Rothman KJ, et al. Variable selection for propensity score models. Am J Epidemiol 2006;163:1149-56.
35. Cuong NNV. Which covariates should be controlled in propensity score matching? Evidence from a simulation study. Stat Neerl 2013;67:169-80.
36. Gottesman O, Johansson F, Komorowski M, et al. Guidelines for reinforcement learning in healthcare. Nat Med 2019;25:16-8.
37. McCaffrey DF, Ridgeway G, Morral AR. Propensity Score Estimation with Boosted Regression for Evaluating Causal Effects in Observational Studies. Psychol Methods 2004;9:403-25.
38. McCaffrey DF, Griffin BA, Almirall D, et al. A Tutorial on Propensity Score Estimation for Multiple Treatments Using Generalized Boosted Models. Stat Med 2013;32:3388-414.
39. Setoguchi S, Schneeweiss S, Brookhart MA, et al. Evaluating uses of data mining techniques in propensity score estimation: a simulation study. Pharmacoepidemiol Drug Saf 2008;17:546-55.
40. Keller B, Kim J, Steiner PM. Propensity Score Estimation with Data Mining Techniques: Alternatives to Logistic Regression. Soc Res Educ Effectiveness 2013.
41. Keller BS, Kim J-S, Steiner PM. Data mining alternatives to logistic regression for propensity score estimation: Neural networks and support vector machines. Multivariate Behav Res 2013;48:164.
42. Westreich D, Lessler J, Funk MJ. Propensity score estimation: neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression. J Clin Epidemiol 2010;63:826-33.
43. De Vries BBLP, Van Smeden M, Groenwold RHH. Propensity score estimation using classification and regression trees in the presence of missing covariate data. Epidemiol Methods 2018;7.
44. Wyss R, Schneeweiss S, Der Laan MJV, et al. Using Super Learner Prediction Modeling to Improve High-

dimensional Propensity Score Estimation. Epidemiology 2018;29:96-106.
45. Rubin DB. Bias Reduction Using Mahalanobis-Metric Matching. Biometrics 1980;36:293.
46. Cochran WG, Rubin DB. Controlling bias in observational studies: A review. Indian J Stat Ser A 1973:417-46.
47. Rosenbaum PR, Rubin DB. Constructing a Control Group Using Multivariate Matched Sampling Methods That Incorporate the Propensity Score. Am Stat 1985;39:33-8.
48. Austin PC. A comparison of 12 algorithms for matching on the propensity score. Stat Med 2014;33:1057-69.
49. Rosenbaum PR. Optimal Matching for Observational Studies. J Am Stat Assoc 1989;84:1024-32.
50. Bertsekas DP. Linear network optimization: algorithms and codes. MIT Press; 1991.
51. Hansen BB. Optmatch: Flexible, optimal matching for observational studies. N Funct Multivariate Anal 2007;7:18-24.
52. Gu XS, Rosenbaum PR. Comparison of Multivariate Matching Methods: Structures, Distances, and Algorithms. J Comput Graph Stat 1993;2:405-20.
53. Diamond A, Sekhon JS. Genetic Matching for Estimating Causal Effects: A General Multivariate Matching Method for Achieving Balance in Observational Studies. Rev Econ Stat 2013;95:932-45.
54. Hansen BB. Full matching in an observational study of coaching for the SAT. J Am Stat Assoc 2004;99:609-18.
55. Rosenbaum P. Observational Studies (2nd ed.). New York: Springer, 2002.
56. Austin PC. Optimal caliper widths for propensity-score matching when estimating differences in means and differences in proportions in observational studies. Pharm Stat 2011;10:150-61.
57. Rubin DB. Using Propensity Scores to Help Design Observational Studies: Application to the Tobacco Litigation. Health Serv Outcomes Res Methodol 2001;2:169-88.
58. Lan P, Wang T, Li H, et al. Utilization of echocardiography during septic shock was associated with a decreased 28-day mortality: a propensity score-matched analysis of the MIMIC-III database. Ann Transl Med 2019;7:662.
59. Zhang Z, Zhu C, Mo L, et al. Effectiveness of sodium bicarbonate infusion on mortality in septic patients with metabolic acidosis. Intensive Care Med 2018;44:1888-95.
60. Zeng C, Dubreuil M, Larochelle MR, et al. Association of Tramadol With All-Cause Mortality Among Patients With Osteoarthritis. JAMA 2019;321:969-82.
61. Imai K, King G, Stuart EA. Misunderstandings between experimentalists and observationalists about causal inference. J R Stat Soc Ser A Stat Soc 2008;171:481-502.
62. Austin PC. A critical appraisal of propensity-score matching in the medical literature between 1996 and 2003. Stat Med 2008;27:2037-49.
63. Austin PC. The Relative Ability of Different Propensity Score Methods to Balance Measured Covariates Between Treated and Untreated Subjects in Observational Studies. Med Decis Making 2009;29:661-77.
64. Deb S, Austin PC, Tu JV, et al. A Review of Propensity-Score Methods and Their Use in Cardiovascular Research. Can J Cardiol 2016;32:259-65.
65. Rubin DB. Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies. J Educ Psychol 1974;66:688-701.
66. Abadie A, Imbens GW. Simple and bias-corrected matching estimators for average treatment effects: National Bureau of Economic Research2002. Report No.: 0898-2937.
67. Abadie A, Imbens GW. Large sample properties of matching estimators for average treatment effects. Econometrica 2006;74:235-67.
68. Imbens GW. The Role of the Propensity Score in Estimating Dose-Response Functions. Biometrika 1999;87:706-10.
69. West SG, Cham H, Thoemmes F, et al. Propensity scores as a basis for equating groups: Basic principles and application in clinical treatment outcome research. J Consult Clin Psychol 2014;82:906-19.
70. Ho DE, Imai K, King G, et al. MatchIt: Nonparametric preprocessing for parametric causal inference. J Stat Softw 2011;42:1-28.
71. Neugebauer R, Laan Mvd. Why prefer double robust estimators in causal inference? J Stat Plan Inference;129:405-26.
72. Schafer JL, Kang J. Average causal effects from nonrandomized studies: a practical guide and simulated example. Psychol Methods 2008;13:279.
73. Rubin DB. Causal Inference Using Potential Outcomes: Design, Modeling, Decisions. J Am Stat Assoc 2005;100:322-31.
74. Rubin DB. The design versus the analysis of observational studies for causal effects: parallels with the design of randomized trials. Stat Med 2007;26:20-36.
75. Rubin DB. For objective causal inference, design trumps analysis. Ann Appl Stat 2008;2:808-40.
76. Austin PC. Comparing paired vs non-paired statistical

methods of analyses when making inferences about absolute risk reductions in propensity-score matched samples. Stat Med 2011;30:1292-301.
77. Keller B, Tipton E. Propensity Score Analysis in R: A Software Review. J Educ Behav Stat 2016;41:326-48.
78. Liu W, Kuramoto SJ, Stuart EA. An introduction to sensitivity analysis for unobserved confounding in nonexperimental prevention research. Prev Sci 2013;14:570-80.
79. Cornfield J, Haenszel W, Hammond EC, et al. Smoking and lung cancer: recent evidence and a discussion of some questions. J Natl Cancer Inst 1959;22:173-203.
80. Gastwirth JL, Krieger AM, Rosenbaum PR. Dual and simultaneous sensitivity analysis for matched pairs. Biometrika 1998;85:907-20.
81. Vanderweele TJ, Arah OA. Bias formulas for sensitivity analysis of unmeasured confounding for general outcomes, treatments, and confounders. Epidemiology 2011;22:42-52.
82. Ding P, Vanderweele TJ. Sensitivity Analysis Without Assumptions. Epidemiology 2016;27:368-77.
83. VanderWeele TJ, Ding P. Sensitivity Analysis in Observational Research: Introducing the E-Value. Ann Intern Med 2017;167:268-74.
84. Mathur MB, Ding P, Riddell CA, et al. Website and R package for computing E-values. Epidemiology 2018;29:e45.
85. Spiegelman D, Mcdermott A, Rosner B. Regression calibration method for correcting measurement-error bias in nutritional epidemiology. Am J Clin Nutr 1997;65:1179S-86S.
86. Cui ZL, Hess LM, Goodloe R, et al. Application and comparison of generalized propensity score matching versus pairwise propensity score matching. J Comp Eff Res 2018;7:923-34.
87. Wang Y, Cai H, Li C, et al. Optimal caliper width for propensity score matching of three treatment groups: a Monte Carlo study. PLoS One 2013;8:e81045.
88. Solomon DH, Rassen JA, Glynn RJ, et al. The Comparative Safety of Analgesics in Older Adults With Arthritis. JAMA Intern Med 2010;170:1968-78.
89. Solomon DH, Rassen JA, Glynn RJ, et al. The Comparative Safety of Opioids for Nonmalignant Pain in Older Adults. JAMA Intern Med 2010;170:1979-86.
90. Rassen JA, Shelat A, Franklin JM, et al. Matching by Propensity Score in Cohort Studies with Three Treatment Groups. Epidemiology 2013;24:401-9.
91. Wu X, Mealli F, Kioumourtzoglou M, et al. Matching on Generalized Propensity Scores with Continuous Exposures. arXiv:181206575v3 [Preprint] 2018 [cited 2020 Jul 14].
92. Hirano K, Imbens GW. The Propensity Score with Continuous Treatments. John Wiley & Sons, Ltd; 2005.
93. Yang W, Joffe MM, Hennessy S, et al. Covariance Adjustment on Propensity Parameters for Continuous Treatment in Linear Models. Stat Med 2014;33:4577-89.
94. Zhang Z, Zhou J, Cao W, et al. Causal inference with a quantitative exposure. Stat Methods Med Res 2016;25:315-35.
95. Bia M, Mattei A. A Stata package for the estimation of the dose–response function through adjustment for the generalized propensity score. Stata J 2008;8:354-73.
96. Zhang Z, Li X, Wu X, et al. Propensity score analysis for time-dependent exposure. Ann Transl Med 2020;8:246.
97. Lu B. Propensity score matching with time-dependent covariates. Biometrics 2005;61:721-8.

(English Language Editor: J. Jones; Quality Control Editor: J. Gray)

Cite this article as: Zhao QY, Luo JC, Su Y, Zhang YJ, Tu GW, Luo Z. Propensity score matching with R: conventional methods and new features. Ann Transl Med 2021;9(9):812. doi: 10.21037/atm-20-3998

