Dissertation - WhatsMatching

Jason Meyer, z5242267

2022-11-18

Preface

This project has been formally titled "What's Matching". This document represents the formal submission via *turnitin* for the dissertation component of the Master of Health Data Science Degree at UNSW (HDAT9900-HDAT9901-HDAT9902-HDAT: Dissertation). The full and properly formatted outputs for the assignment can be accessed in the following places:
- WhatsMatching Website
- WhatsMatching Shiny App
- WhatsMatching Github Repository

WhatsMatching

The WhatsMatching package provides a shiny app that demonstrates how propensity score matching and mahalanobis distance matching compare under different circumstances. The aim of the app is to provide educational insight into matching methods.

Getting Started

Check out the vignettes in the navbar menu above to learn more about the motivation behind this app, the available features and functionality, and how the app can be used to explore matching techniques.

Accessing the App

The Shiny app can be accessed at jmeyer2482.shinyapps.io/WhatsMatching . A preview of the app is shown below.

Installing the App

Install the latest version of the app via **GitHub**. From the package you can access all the functions that are used to generate the data and matching processes/plots in the app.

```
devtools::install_github("jmeyer2482/WhatsMatching")
```

Abstract

Background Matching is a popular technique for improving covariate balance in observational data. Due it's simplistic implementation the inner workings are not necessarily clear to the analyst. Recent criticisms of Propensity Score Matching, in particular, have highlighted a need to give more insight and clarity to the black box that is Matching.

Aim We set out to develop an interactive space where users can understand matching algorithms and compare results based on different user specifications. We wanted users to be able to compare methods of matching but more importantly, to compare the underlying settings that can be used to generate the matches. Settings needed to include the selection of covariates, the order the matches were conducted in, and whether or not replacement was used.

Solution WhatsMatching is an interactive Shiny app that allows users to simulate data and watch matching algorithms dynamically unfold. It provides options for using the Propensity Score Distance and the Mahalanobis Distance to match on and includes all the settings laid out in the aim.

Dissemination The Golem framework has been the underpinning guide for development. This has allowed the provision of an installable R package, a Shiny app deployed on shinyapps.io, and all documentation and vignettes on a package website. All code is opensource and available on GitHub.

Overview

The problem and promise of observational data

Estimating the causal effect of a treatment or intervention is a core task of health data science. Randomised control trials (RCTs) are the gold standard study design for estimating treatment effects, but in many contexts RCTs are not feasible, for logistic or ethical reasons. As a result, researchers often turn to observational data to address causal research aims. These data are not ideal because there is no control over who receives the treatment or exposure of interest and who does not, introducing potential selection and confounding biases. Nonetheless, observational data are a useful asset, being relatively cheap and easy to access, and for exposures that cannot ethically be manipulated, they offer the only possibility of investigating research questions of interest.[8,10,22]

Approaches to obtaining valid causal inferences from observational data

It is still possible to obtain valid estimates of causal effects from observational data, provided the limitations of the data are addressed in the analysis. Many methodological approaches to address confounding bias have been proposed over the years, including stratification, regression adjustment, matching, applications of the propensity score and g-computation methods.[8,14] The primary aim of these methods is to reduce bias in the estimation of the casual effect, especially confounding, by balancing the distribution of confounding variables in the treated and control groups.[13,23]

Matching methods

Matching is a general pre-processing technique used to improve causal estimates from observational data by matching treated individuals to untreated individuals (controls) with the same or similar background characteristics. The objective of matching is to balance key background characteristics in the treated and control groups. In a RCT this balance is achieved through randomisation; the intuitive appeal of matching is as a simple way to find a RCT "hidden" in an observational dataset.[10]

The most basic form of matching is exact matching. This is where treated and control groups have their key characteristics matched exactly. For example, if matching is done on age and sex, a treated 45 year old male will be matched with an untreated 45 year old male. When there are a small number of matching variables and the pool of potential matches is large it can be possible to match each treated individual to a control with the exact same characteristics, leading to a perfectly balanced dataset. However, using exact matching is often infeasible in practice, especially if there are numerous matching variables and/or a small pool of potential matches.

One alternative to exact matching is coarsened exact matching. Under this approach, matching variables are temporarily coarsened, for example age measured in years might be recoded to 10-year age bands. The coarsened variables are then used to undertake exact matching, and once the balanced dataset is found the analysis can proceed with the original uncoarsened data.

A second alternative to exact matching is distance-based matching. Distance-based matching proceeds by matching treated and untreated individuals based on some distance metric that quantifies the similarity between each pair of individuals in the dataset. Perhaps the most widely used distance metric is the propensity score, defined as the probability of an individual receiving the treatment.[22] Another possible distance metric is the Mahalanobis distance, which is analogous to a multivariate Euclidean distance.[10]

The advantage of both the propensity score and the Mahalanobis distance metrics is that the multidimensional covariate space is reduced to a single dimension: matching can be undertaken on the univariate distance measure rather than attempting to match each individual covariate. The covariates that define the distance metric are balanced between the treated and control groups, on average, in the resulting matched dataset.

The propensity score

Perhaps one of the most popular and enduring family of techniques for estimating causal effects from observational data revolve around the propensity score. Rosenbaum and Rubin (1983) defined the propensity score as the "conditional probability of assignment to a particular treatment given a vector of observed covariates".[22] Mathematically, this can be expressed as $e(x_i) = pr(Z_i = 1|x)$ where $e(x_i)$ is the propensity score, Z is the treatment variable, x is a vector of covariates, and i represents an individual.

In most cases, the propensity score is estimated from available data using a simple logistic regression on a binary treatment assignment indicator. The logistic regression formula may take the form of $t \sim x_1 + ... + x_n$ where t is a binary indicator of whether or not the individual received treatment and $x_1 ... x_n$ represent n observed covariates that help to predict the treatment outcome. Although logistic regression is widely used for this step, the estimation method is at the discretion of the analyst and more flexible models can perform well.[12]

Once estimated, the propensity score can be incorporated into the data analysis in multiple ways. This includes matching, weighting and stratification (AKA subclassification). All three applications are briefly described below. Matching is the the primary focus of the WhatsMatching app, however the app also allows comparisons in estimates based on all three approaches.

Applications of propensity scores – matching, weighting and stratification

Matching A common application of the propensity score is as a distance metric in distance-based matching. The distance between two individuals is defined as the absolute difference in their propensity scores $|e(x_1) - e(x_0)|$. Treated individuals are matched to untreated individuals with similar propensity scores, and in doing so the covariates that are used in the propensity score estimation should be balanced on average in the treated and control groups.

Many packages in `R` that perform matching analysis also automate the process of estimating the propensity score. Once the probabilities have been calculated then treated units are paired with control units that have the same or similar probability of treatment.

It is important to note that estimated effects generally correspond to the average treatment effect on the treated (ATT) when using matching methods. This makes sense intuitively as matching effectively removes individuals from the data that could not be considered both treated and untreated. This is referred to as strongly ignorable treatment assignment and is a key assumption for providing unbiased estimates with the propensity score.

Weighting Weighting with the propensity score differs from matching as no units are pruned from the dataset. This is an advantage over matching in the sense that all data can remain in the analysis, giving the opportunity to calculate the average treatment effect on all units (ATE), as opposed to the ATT.

The formulas for weighting vary depending on the method you want to employ. The data being analysed, by virtue of its source, may be more appropriately analysed for the ATT and not the ATE. The formulas for these weights are:

$$ATE, w(W,x) = \frac{W}{\hat{e}(x)} + \frac{1-W}{1-\hat{e}(x)}$$

$$ATT, w(W,x) = W + (1-W)\frac{\hat{e}(x)}{1-\hat{e}(x)}$$

where $\hat{e}(x)$ indicates the estimated propensity score conditional on observed x covariates.[4,22]

Stratification (aka Subclassification) The stratification method varies from matching and weighting in that it relies on the use of the propensity score to group units into quantiles. Once the number of groups has been established, the literature suggests this is between five and ten depending on the size of the dataset,[13] the treatment effect is estimated across the groups and then averaged to give a mean difference ATE.[4,22]

The Mahalanobis distance

The Mahalanobis Distance was developed by Prasanta Chandra Mahalanobis in 1936 as a means of comparing skulls based on their measurements. It's normal application calculates how many standard deviations a point is from the mean of the specified variables and can give a good indication of outliers in a group.[15,16] In matching, the application is similar but not quite the same. Instead of using the mean as a comparator, each treated unit is compared, pair-wise, with each of the units in the control group.[10]

The pair-wise Mahalanobis Distance is calculated by $D_{ij} = \sqrt{(X_i - X_j)S^{-1}(X_i - X_j)}$ where D_{ij} is the pair-wise Mahalanobis distance, X_i and X_j represent the matrix of covariates for the treated and control groups and S^{-1} is the covariance matrix.[10] Similar to the Propensity Score, the Mahalanobis Distance reduces a multidimensional space to a single value representing the distance between units. A key difference in understanding how the matching works between these methods is understanding that the Mahalanobis distance is the measured distance between units whereas the Propensity Score distance is the difference in the probabilities of treated and untreated units.

It should be further noted that individual units have their own propensity score (or probability of being treated) however, they do not have a Mahalanobis value assigned to them. The value from the Mahalanobis distance is *between* the units. This is a fundamental difference in the way the matching occurs because, while the true values of the covariates have been used to determine the probability of treatment, the propensity score is blind to the real data.

Gaps in Understanding

The propensity score matching process appears to be straight forward and require little effort to implement, which is reflected in its ubiquitous use. However, as King and Neilsen[10] point out, there are many situations where the use of the propensity score for matching may not be suitable. Their 2019 paper gives many insights into the use of the propensity score and why other methods, like coarsened exact matching or distance-based matching with the Mahalanobis distance, may be better choices. While some of the explanations do make intuitive sense it can be difficult to decipher the practical application and implications of the advice.

Matching Settings

While the choice in the distance used for matching can obviously make a difference to how units are paired, it is less clear what the implications of changing any of the many settings that are available in order to "optimise" the matches. The settings that we will discuss further include order, replacement, and calliper adjustment.

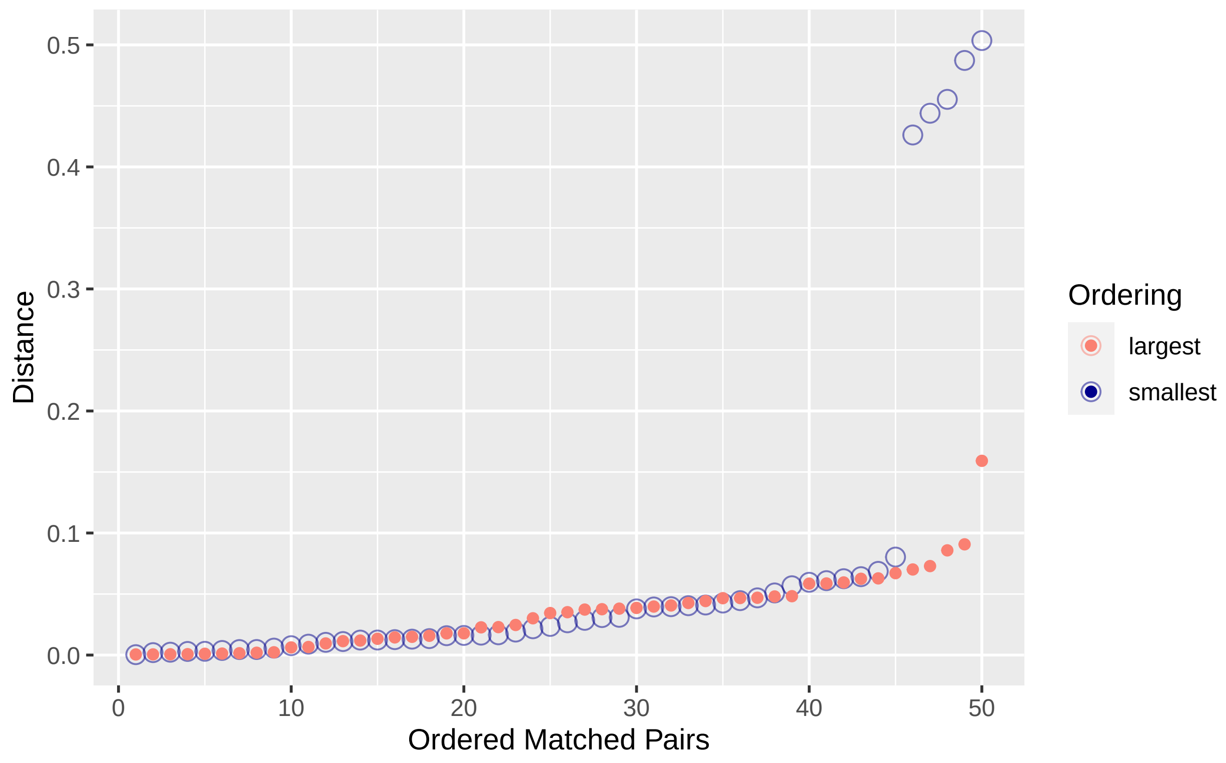
Order Order refers to the order the matches are specified in the matching algorithm. In the `MatchIt` package,[7] users are only able to specify an order for specific methods. In particular, propensity score methods can control the order of the matching as they are a vector of probabilities that aligns with the data. The default for the order is *data*, otherwise ordering can be conducted by *smallest*, *largest*, or *random*. The resultant ordering is as follows:

- *data* - the order supplied by the dataframe
- *smallest* - matches occur starting with the smallest propensity score first
- *largest* - matches occur starting with the largest propensity score first
- *random* - matches occur randomly

Why does ordering make a difference? Well, when you have lots of treated units close to a single untreated unit, only one of the treated units will be able to match with it. Once that untreated unit has been used, the other treated units will have to look further afield. Generally this does not impact the matches at first but towards the end of the iterative process it can. See Figure 1 below to see the difference between the distances generated by using the *smallest* ordering vs the *largest* ordering.

Figure 1. Matched Distances between Different Ordering

Matching conducted with the propensity score

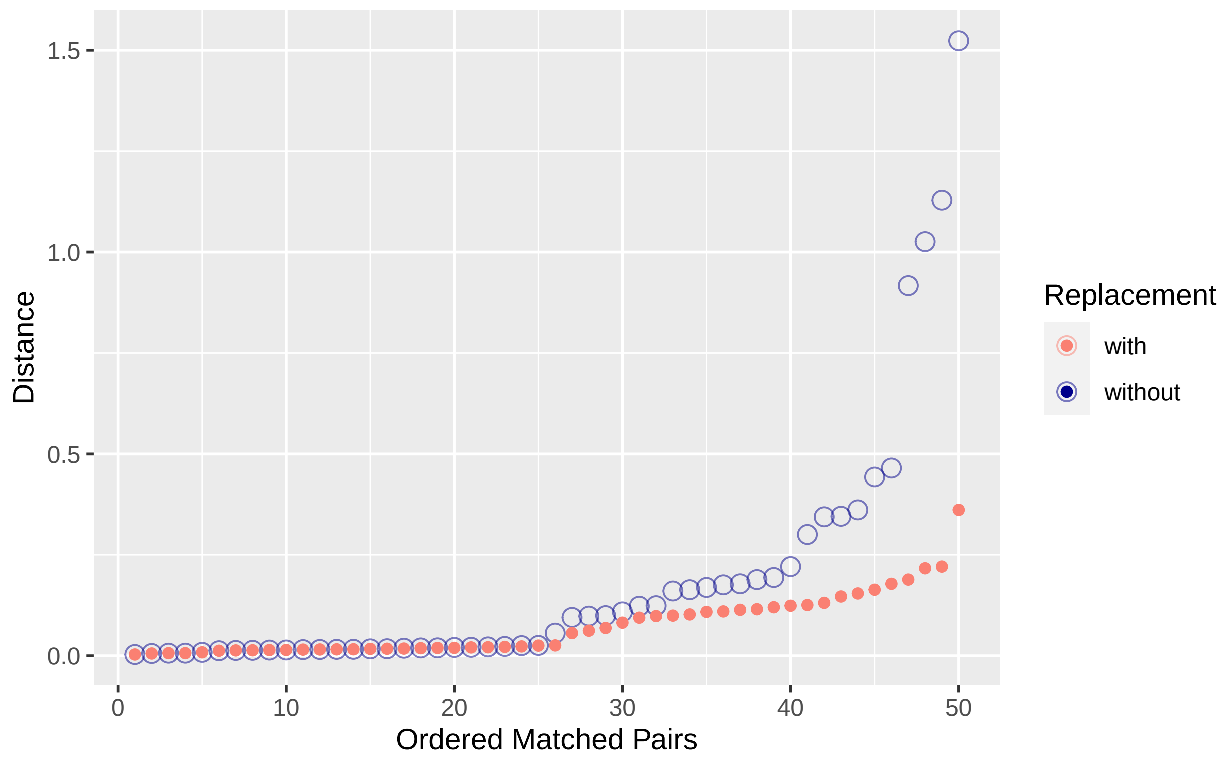


Replacement Replacement is a setting that controls whether an untreated unit can be reused. It is applicable for both MDM and PSM and other forms of matching where units are paired (as opposed to grouped). Using replacement means that you may be able to improve your covariate balance because the treated unit will be matched with the closest untreated unit, irrespective of the order. In other words, ordering is redundant when replacement is used.

It is important, however, that appropriate steps are taken to account for an untreated unit being matched multiple times. There are two main ways this can be dealt with: 1. include duplicates of the untreated data in the matched dataset; 2. apply appropriate weighting to include when estimating the treatment effect.

Figure 2. Matched Distances With and Without Replacement

Matching conducted with the Mahalanobis Distance



Calliper Adjustment The callipers setting can be used to avoid matching units that are considered too far apart, based on some distance threshold. The implications of using this setting in practice is not particularly clear. If you cast your eyes back to Figures 1 and 2, you can see that there are units towards the end that are further apart than the rest of the group. The calliper setting seeks to remove those outliers with the intention of improving covariate balance.

The propensity score paradox

King and Neilsen[10] presented a paradox that arises when using the propensity score for matching. They were able to demonstrate that, when compared to other matching distances like the Mahalanobis distance, matching on the propensity score only approximated a randomised experiment as opposed to a more efficient fully blocked experiment.

As the matching algorithm proceeds, unmatched observations are pruned from the dataset. Initially, this reduces the imbalance between the treated and untreated groups with corresponding improvements to the estimated treatment effect. However, after a certain threshold is reached, further pruning the dataset starts to result in more biased estimates of the treatment effect. This is the PSM paradox.

The authors postulated that this made matching on the propensity score inferior and were able to demonstrate this concept visually but were unable to provide mathematical proofs of the implications for real data.

Model Dependence

Another issue that can arise generally with causal inference, but then also specifically with matching, is model dependence. This is essentially where the model selection still plays a large role in predicting the estimate of the treatment effect. This is a problem because that probably means that matching has not fixed the issue of confounding and now selection bias needs to be managed as well.

Statement of research aim

> Any sufficiently advanced technology is indistinguishable from magic.
>
> –*Author C. Clarke*

The flexibility and convenience of distance-based matching approaches have led to numerous software implementations and widespread use across several quantitative disciplines. This is perhaps especially true of propensity score matching, a modern workhorse of causal inference. However, as with any sufficiently advanced technology, the complexity of distance-based matching can be lost on the end user. The many subjective decisions can be absorbed into the black box of a software function, with nicely matched datasets appearing, as if by magic, at the other end.

The aim of this research is to explore some basic matching methodology and provide an interactive and educational space for people who want to better understand matching methods. I will crack open the propensity score matching algorithm, allowing users to visualise matching algorithms as they dynamically unfold, and compare results under different combinations of user settings.

The resulting space is provided as a publicly available, interactive Shiny App and supported by a website and R package stored on Github. This will allow students to experiment with different matching methods and more experienced users to simulate their own matching experiments in the R environment.

Interactive applications to support learning

Learning by doing has been demonstrated to significantly improve learning when properly scaffolded, especially as part of Massive Open Online Courses (MOOCs), which are a key component of the modern educational tool kit in tertiary settings[11].

RStudio supports the implementation of learnr packages that are used to provide self guided, tutorial style experiences for users. This option has had wider uptake in the tech savvy academic community. There has also been recent growth in the use of interactive learning websites like Kahn Academy. In this environment users are able to access scaffolded learning through theoretical concepts coupled with practical applications. As noted above, this has been found to be an effective method for providing education.

The space for app development

This project is aimed at the development of an educational tool where users can build an intuitive understanding of matching methods and compare different methods as discussed above. Using this educational tool, students will be able to generate data and visualise the matching process as it unfolds. Users will also be able to compare the performance of matching, with different degrees of pruning, to the weighting and stratification applications of the propensity score that don't involve pruning, and to matching approaches using the alternative Mahalanobis distance.

Currently, there is no available platforms to compare matching methods interactively. The purpose of the app is to provide educational support and the ability for users to play with and visualise how matching occurs. In addition, this app has been developed as a package which contains easy to use functions that are available to more advanced users who may want to investigate beyond the constrains of the Shiny App.

References

Project Development

This project has been developed as a Shiny app using the Golem framework[3] to support it's production. There are a number of key functions that have been implemented to generate the data and matching processes. Let's have a look in a bit more detail.

The golem framework

The Golem framework is the brain child of Colin Fay. In essence, Shiny apps are ideal for development as a R package. That is to say that, the package structure is well suited to the development of production grade Shiny[2] apps.

The `golem` package itself provides functions and helpers to generate and create the framework that can be used for the development of the Shiny app and package. In addition, there are theoretical underpinnings and best practice advice provided to give the best chance of developing a high quality application.

Key functions and functionality

There are four key functions that have been developed as part of the package in order to make data generation and analysis as straightforward as possible. They are `create.sim.data`, `matched.data`, `matching.plot`, and `combined.plot`. The functions have been designed to follow and iterative process. These functions draw upon existing functionality from other packages including `ggplot2`[30], `plotly`[26], `dplyr`[31], and others[1,19,25,28].Full details can be found in the *Reference* section.

`create.sim.data()` As the name suggests, this is the data simulation function. There are four simulations available for use within the app. Each one has a specific type of data generation to explore the matching process.

`matched.data()` This is the function that conducts the matching. It primarily uses `match_on()` function in the `optmatch` package[6] to generate a $n \times p$ matrix of matches for the chosen method, where n represents the control units and p represents the treated units. The output is then used to generate the actual matches along with a number of other outputs that allow further exploration of the data outside of the app if desired. However, it should be noted that this function is not designed for complex matching and, for example, may not be suitable for use with categorical data.

`matching.plot()` This function takes the output from the `matched.data` function and turns it into a `plotly` plot which demonstrates the sequence of matches occurring in a cumulative fashion, starting with the nearest matched pair to the furthest matched pair. Outside the app, this plot can be generated on its own.

`combined.plot()` The `combined.plot` function takes 2 different matching methods generated by the `matched.data` function and returns 4 subplots that highlight the differences between the matching methods. Plots 1 and 2 contain visualisations of the way the data has been matched (via the `matcing.plot` function), Plot 3 shows the standardised mean differences between the specified covariates and methods (specified in the function, not the formula), and Plot 4 shows the differences in the calculated estimate of the treatment effect for both methods. Plots 3 and 4 show the results for each set of matched data as shown in Plots 1 and 2 as a progression through the matching process with fixed values of the unadjusted data. Plot 4 also includes stratified and weighted propensity score estimates for comparison.

Deployment

The development process for this app has meant that it has been most practical to manage on GitHub. All the associated files and documentation are hosted on a public GitHub repository. In terms of the process undertaken, the `WhatsMatching` package is the vehicle for the Shiny App as suggested in Golem framework. From there, `pkgdown`[32] has been used to create the package based website you are now viewing, which is deployed from GitHub.

The end result result of this developmental process means that there is a Shiny App and Package Website that can be used in an educational format. The GitHub hosting means that the package can be installed by running `devtools::install_github("jmeyer2482/Whatsmatching")` in an `R` environment making the functions available to more advanced users. Figure 1 shows the development pathways and output that have been generated from the process.

References

The App Interface

The front end of the app has been design to be minimalist but with enough information available to the user to work out what's going on. Figure 1 below shows the main area of the app that's visible on loading.

When the app loads, the first simulation is run. This means that as soon as a user opens the app they are presented with a standard presentation of matched data to investigate. The full home screen of the app also contains a sidebar panel on the right with some additional information for users. Table 1 below outlines some of the options that users have to access through the app.

The opens up to options for data generation and settings for matching. There is further information available about the options available here `vignette("DataGen", "WhatsMatching")`.

The second option, , takes you to a modal display with various information about the data that has been generated. This includes the same plots that are in the *Data Generation*section. There is also a table of the data available; information about the match settings (also visible on the main page); insights about setting a calliper; and insights into the estimates of all methods with other regression model settings.

The calliper plots are similar to those laid out in `vignette("Background", "WhatsMatching")`. There are four plots that show the distances between matched pairs in an unadjusted format (top row) and a standardised format (bottom row). The standardised plots are measured in the number of standard deviations above zero. The plots on the left show the distances in the order they were matched while the plots on the right show them in order from closest to the furthest distance. If you want to apply a calliper to the matched data, find the observation that you think is the beginning of the outlier group you'd like to remove. Subtract one from that number and go to that frame using the slider on the main screen.

The plot of the estimates is similar to one shown in the *Functions* vignette. The user has the option of specifying which method they would like to see the estimates for. When *Apply* is click, the plot will generate the estimates with the 95% confidence intervals for all possible regression formulas for those methods for easy comparison.

The third option, , is a random data and match settings generator. If the user doesn't have any ideas about what to investigate then they can push this button and the app with decide. It has been designed so that Method 1 and Method 2 will always be different. The opens this website to the homepage.

In terms of the general user interface, `plotly` has some very nice and intuitive options for navigating visual data. Users are able to easily zoom in to plot areas and reset them using the buttons available on the top left of the plot. The items in the legend can be toggled out of the plot area to make visualising things easier. The slider can be moved about as the user wishes.

Additional contextual information is provided in a sidebar on the right. It includes some basic operational information as well as the settings for easy reference.

From the main page, clicking on the icon under the title opens up a modal dialogue box that is displayed in the following figures. This modal gives the user access to data from four different simulations and a real dataset. I will briefly outline each of the simulations below.

Simulation 1 This simulation is based on the data generation process described by King and Neilsen[10]. Essentially, it contains two experiments—a fully blocked randomised experiment and a completely randomised experiment—hidden within an imbalanced "observational" dataset. If you look at Figure 3, the first plot (top left) makes this clearer. The fully blocked randomised experiment is in the top right corner of the plot which is demonstrated by the treated (red) and untreated (blue) units being virtually right on top of one another. The completely randomised experiment is just below it in the bottom right corner of the plot and you can clearly see that there are similar numbers of treated and control units but they are not exactly matched. The imbalanced observational data is represented by the placement of the untreated unit on the left of the plot.

The purpose of this simulation in the paper was to demonstrate the ability of different matching methods to select the units in a logical order. Ideally, the data from the fully blocked randomised experiment should be matched first, then the data from the completely randomised experiment. It has been included in the app as it provides a clear demonstration of the performance of mahalanobis-based distance matching versus propensity score-based distance matching.

In the app, you are able to change the treatment effect but that should not change the outcome of the matching process in any way, only the calculation of the treatment estimate.

Simulation 2 Simulation 2 is also taken from King and Neilsen[10] however the data is very different. This simulation uses two overlapping uniform distributions that both affect the treatment received and the outcome. When visualising the two covariates, `X1` and `X2`, on a two dimensional field, the data has the appearance of a square of treated units overlapped by a square of untreated units. This can be clearly seen in the top left plot in Figure 4 below.

Users have the ability to change the range of the uniform distribution, the strength of association between the covariates (`X1` and `X2`) and the treatment variable (`t`), and the treatment effect on the outcome (`y`). Probably the most useful illustration of this data is to compare the same method of matching but change whether replacement is used or not. It can be particularly striking if there is a strong association between the covariates and treatment.

Simulation 3 This simulation uses causal relationships to define the effects between the treatment, covariates, and outcome. There are five options available to the user that can change the data generation relationships between the treatment indicator `t`, the outcome `y` and two covariates `X1` and `X2`. Figure 5 demonstrates how each of the options fit into a causal diagram with an arrow pointing in the direction of the assumed effect in the data generation process.

To generate the data, first we create the covariates, `X1` and `X2`, as a bivariate normal distribution with a covariate correlation of 0.2. The treatment variable is then generated based on the selection after which the function then iteratively works through the selected options and applies changes to the data as appropriate. It should be noted that the user can not change the strength of the effects in this simulation. Nor can the relationship between `t` and `y` be changed, although they can both be altered via `X1` and `X2`. Other than the causal relationship, only the treatment effect can be modified. This has been done to allow for simple data generation. Figure 6 below gives an overview of what the generated data may look like.

Simulation 4 This is the last simulation. Like simulation 3, `X1` and `X2` are taken from a bivariate normal distribution. However, in this simulation, `X1` and `X2` can effectively act as confounder, an ancestor of `t`, or an ancestor of `y`. This is achieved by altering the weights of the effect on the treatment or the effect on the outcome. In addition, you can change the covariance correlation which will increase the correlation of `X1` and `X2`. See Figure 7 below for a preview of what the data from Simulation 4 may look like.

Real Data - FEV The *Forced Expiratory Volume* dataset is available from the `mplot`[27] package. For the purposes of the app, it has been utilised to demonstrate how matching works on real data using `smoke` as the treatment variable and `fev` as the outcome. This dataset was ideal for this project because it was relatively small, contains only a few variables, and contains a treatment and outcome variable. A preview of the data is below in Figure 8.

References

Matching

Once the user has generated some data. The next step is to select how the matching should take place. The screenshot in Figure 1 shows what the user will see once they have selected the data (or chosen to continue without changing).

Settings

As can be seen above, there are five settings that can be changed - *Matching Covariates*; *Matching Distance*; *Matching Order*; *Use Replacement*; and *Outcome Formula.* The settings and their purpose are outlined below.

Matching Covariates The user has the option to select the available covariates that they would like to match the treated and untreated units on. This will include `X1` and `X2` for the simulated data and `age`, `height` and `sex` for the `fev` data. The matching cannot be completed without having selected at least one covariate for each method.

Matching Distance The *Matching Distance* options include the Mahalanobis Distance and the Propensity Score. Both methods are implemented using the `optmatch` package however, the propensity score is calculated apriori using logistic regression between the treatment and selected covariates.

Matching Order This settings determines what order the matches are conducted in. *data* is the order that the data is in and *random* is a randomly generated order. When using *smallest* and *largest* the data is ordered according to each unit's propensity score, this is irrespective of the method of matching used. To compare to the `MatchIt` package, Mahalanobis Distance matching is always conducted using *data.*

Use Replacement The *Use Replacement* option tells the algorithm whether an untreated unit can be matched to more than one treated unit.

Outcome Formula This is the formula that is used to calculate the treatment effect on the outcome. It will always be at least `outcome ~ treatment`. This setting gives you the option of including the other covariates in the estimate.

After the user has selected the settings that they would like to compare, they can simply press the "Use these matching settings" button at the button and the app will begin its calculations.

The output from the decision can take some 10s of seconds to be realised due to a combination of the extra data that needs to be generated for the cumulative plots and the processing to render the `plotly` object. Once rendered it will display four plots with a legend on the right and a slide bar down the bottom.

Overview

In this vignette I present four examples of important lessons that can be interactively explored using the WhatsMatching app. These examples have been selected to highlight just a few of the ways the app can lift the lid on the black box of matching. The four examples are:

- Matching finds the experiment hidden in observational data
- To Replace or Not to Replace
- Ordering changes unit selection
- Covariate and Model Selection Matter

This section concludes with some suggested future extensions of the WhatsMatching app.

Matching finds the experiment hidden in observational data

This example uses simulation 1, which generates two experiments "hidden" within an unbalanced dataset—a fully blocked randomised experiment and a fully randomised experiment. Covariates `X1` and `X2` are generated with the fully blocked experiment being almost identically paired between treated and control units; the random experiment group has randomly placed treated and control units; and imbalanced control group consists of only control units and no treated units.

The methods being compared in this example are the use of the Mahalanobis Distance Matching (MDM) and Propensity Score Matching (PSM). The outcome demonstrated is that, under these conditions, MDM is superior to PSM in selecting the fully blocked experiment first. This occurs because the Propensity score, once calculated, is blind to the real data that informs the probability of treatment whereas the Mahalanobis Distance is the actual distance *between* units based on their observed covariates.

Take home message: The propensity score is blind to the underlying values of the matching covariates and therefore may not be the best choice for achieving covariate balance, which is a primary aim of matching.

Run this example in the app: https://jmeyer2482.shinyapps.io/WhatsMatching/

- Click Settings and choose *Simulation 1*
- Optionally change the treatment effect and click Preview data
- Click *Use the data from Simulation 1* below
- Update the matching settings as laid out above (Fig. 1)
- Click *Use these matching settings*

- Press play

To Replace or Not to Replace

We demonstrate in Figure 2 that the intuitiveness behind using replacement may not be clear. This simulation is also from the King and Neilsen paper. The covariates `X1` and `X2` constitute two overlapping uniform distributions for the treated and control groups. There are 200 units in the dataset with 100 units in each group. In this example, the matching approach is always propensity score matching, we have only changed whether the matches are conducted with replacement (Method 1) or without replacement (Method 2).

In Figure 2, you can clearly see that when the matching algorithm permits replacement there are several treated units matched to a single control unit. Comparing the balance in the matched dataset under the two approaches (Plot 3), you can see that as the matching proceeds the standardised mean distance between the treated and control units is stable when replacement is allowed but starts to deteriorate after about the

60th match when replacement is not allowed. Similarly, the estimated treatment effect (Plot 4) shows that propensity score with replacement results in an unbiased estimate of the known treatment effect, whereas the propensity score without replacement results in a highly biased estimate. These results reflect that, in the absence of replacement, the quality of matches deteriorates as the matching algorithm proceeds. In real life applications, a callipers to impose a minimum threshold in distance when making matches.

Take home message: Datasets where the treated and control groups are approximately equal may require the consideration of different settings to achieve covariate balance and thus a more accurate estimate of the treatment effect.

Run this example in the app: https://jmeyer2482.shinyapps.io/WhatsMatching/

- Click Settings and choose Simulation 2
- Optionally change any of the settings and click *Preview Data* (leaving the settings unchanged will replicate above)

- Click *Use the data from Simulation 2* below
- Update the matching settings as laid out above (Fig. 2)
- Click *Use these matching settings*

- Press play

Ordering changes unit selection

For this example we are using the same simulation as in Figure 2 above. This time we are comparing the ordering of the matches: Method 1 starts matching from the largest propensity score and Method 2 starts matching from the smallest propensity score. This essentially boils down to matching the treated units with the highest probability of being treated first or starting with those with the lowest probability of being treated.

We can see below in Figure 3 that *Plot 1* has made some unusual pairings. This makes some sense as when you pair units that have the highest probability of being treated first then you quickly increase the distances between the remaining units that need to be paired. This means that the "good matches" are getting further and further away.

This is a problem that can occur for any matching method as most matching methods use nearest neighbour methodology. If the best nearest neighbour is already paired then the process just moves to the next available one no matter the distance. This further highlights the advantage of using replacement which conveniently ignores ordering and selects the nearest partner irrespective of if it is matched.

Take home message: The ordering or optimisation method of how the matches are selected can impact what data is selected for analysis.

Run this example in the app: https://jmeyer2482.shinyapps.io/WhatsMatching/

- Click Settings and choose Simulation 1
- Optionally change any of the settings and click *Preview Data* (leaving the settings unchanged will replicate above)

- Click *Use the data from Simulation 2* below
- Update the matching settings as laid out above (Fig. 2)
- Click *Use these matching settings*

- Press play

Covariate and Model Selection Matter

Covariate specification and model selection are important when undertaking matching. Ideally, the matching occurs in such a way that the model specification is less important (another problem for causal inference in observational data!) which will remove some of the bias. Simulation 4 has been used for this example with both `X1` and `X2` set to be strong confounders.

Matching has been completed using either `X1` or `X2` as though we don't know which one is a confounder. `X1` has also been added to the formula to calculate the estimate. These setting have been selected to demonstrate the importance of model and covariate selection when performing matching. As we can see below, the matching that was done with `X1` only performed as well as the raw estimate. Whereas, the model with `X2` performed much better.

Take home message: The treatment effect will be improved as along as both variables are accounted for in either the matching or the regression model.

Run this example in the app: https://jmeyer2482.shinyapps.io/WhatsMatching/

- Click Settings and choose Simulation 4
- Set the effect on treatment to 1 for both `X1` and `X2`
- Set the effect on outcome to 1 for both `X1` and `X2`
- Other settings can be changed as desired
- Set matching settings as laid out above
- Press play

Summary

This application achieves the aims that were set out. It provides a simple space for users to be able to gain insights into the matching process through exploration. Users are able to pair any possible combination of data generation with the matching settings of their choosing.

Highlights

The app has a friendly interface that allows users to easily manipulate data and matching settings to see if the outcomes match expectations. Technical jargon has been paired back in favour of lay descriptions and there are useful popovers to give additional information on many of the settings available.

The four plot output from the app is clean and operates intuitively. The `plotly` interface make interacting with the plots clean. Additional information with on the plots from tooltips mean the visualisations aren't cluttered with too much information. A sidebar was added to give meaningful context without intruding on the main event.

Limitations

There are some limitations to the resources available in this suite. Firstly, users are limited to using only Propensity Score or Mahalanobis Distance. This has been a deliberate choice as the plotting interface is likely no able to produce the same information for other matching methods.

Secondly, the package is only designed for gaining insights into matching, not performing statistical analysis. However, users may choose to run data through the package first to try and make better decisions about matching for their real analysis.

Thirdly, due to the processing power required to create and render the `plotly` plots, it is not particularly feasible to increase the amount of data the matching is done on. To give an example, for a dataset that returns 100 matches there will need to be 5050 rows times 2 just for generating the first two plots.

Future Possibilities

Where to from here? The app is functional and fit for purpose as is, even with the limitations listed above. Considerations for future development may include:

- extend the capability of the app to explore other matching methodologies
- increase the number of dimensions for viewing the matches, for example to include up to 3 or 4 variables
- add the capability to compare different formulas for estimating the outcome at the same time
- Add an option in for calliper adjustment (it is just a theoretical option to visualise at the moment)

References

1. Canty A, Ripley BD. Boot: Bootstrap r (s-plus) functions. Published online 2021.
2. Chang W, Cheng J, Allaire J, et al. *Shiny: Web Application Framework for r.*; 2022. https://shiny.rstudio.com/
3. Fay C, Rochette S, Guyader V, Girard C. *Engineering Production-Grade Shiny Apps.* 1st ed. Chapman; Hall/CRC; 2021:398. https://engineering-shiny.org/
4. Guo S, Fraser M, Chen Q. Propensity score analysis: Recent debate and discussion. *Journal of the Society for Social Work and Research.* 2020;11(3):463-482. doi:10.1086/711393
5. Hansen BB. The prognostic analogue of the propensity score. *Biometrika.* 2008;95(2):481-488. doi:10.1093/biomet/asn004
6. Hansen BB, Klopfer SO. Optimal full matching and related designs via network flows. 2006;15.
7. Ho DE, Imai K, King G, Stuart EA. {MatchIt}: Nonparametric preprocessing for parametric causal inference. 2011;42. doi:10.18637/jss.v042.i08
8. Igelström E, Craig P, Lewsey J, Lynch J, Pearce A, Katikireddi SV. Causal inference and effect estimation using observational data. *Journal of epidemiology and community health (1979).* 2022;76(11):960-966.
9. Jann B. Why Propensity Scores Should Be Used for Matching. 2017 German Stata Users Group Meeting. Berlin, June 23, 2017. Published online 2017. https://www.stata.com/meeting/germany17/slides/Germany17_Jann.pdf
10. King G, Nielsen R. Why Propensity Scores Should Not Be Used for Matching. *Political Analysis.* 2019;27(4):435-454. doi:10.1017/pan.2019.11
11. Koedinger KR, McLaughlin EA, Kim J, Jia JZ, Bier NL. Learning is not a spectator sport: Doing is better than watching for learning from a MOOC. *L@S 2015 - 2nd ACM Conference on Learning at Scale.* Published online 2015:111-120. doi:10.1145/2724660.2724681
12. Lee BK, Lessler J, Stuart EA. Improving propensity score weighting using machine learning. *Statistics in medicine.* 2010;29(3):337-346.
13. Li M. Using the Propensity Score Method to Estimate Causal Effects. *Organizational Research Methods.* 2013;16(2):188-226. doi:10.1177/1094428112447816
14. Lunceford JK, Davidian M. Stratification and weighting via the propensity score in estimation of causal treatment effects: a comparative study. *Statistics in Medicine.* 2004;23(19):2937-2960. doi:10.1002/sim.1903
15. Mahalanobis PC. Analysis of race mixture in Bengal. *Journal and Proceedings of the Asiatic Society of Bengal.* 1927;23:301-333.
16. Mahalanobis PC. On the generalized distance in statistics. *On the generalized distance in statistics.* 1936;2(1):49-55. http://library.isical.ac.in:8080/jspui/bitstream/10263/6765/1/Vol02_1936_1_Art05-pcm.pdf

17. Match icons created by freepik - flaticon. Accessed October 24, 2022. https://www.flaticon.com/free-icons/match
18. Morris TP, White IR, Crowther MJ. Using simulation studies to evaluate statistical methods. *Statistics in Medicine*. 2019;38(11):2074-2102. doi:10.1002/SIM.8086
19. R Core Team. R: A language and environment for statistical computing. Published online 2022. https://www.R-project.org/
20. Ripollone JE. *Exploration of structural and statistical biases in the application of propensity score matching to pharmacoepidemiologic data*. Thesis/Dissertation. Boston University; 2019. https://hdl.handle.net/2144/36025
21. Ripollone JE, Huybrechts KF, Rothman KJ, Ferguson RE, Franklin JM. Implications of the propensity score matching paradox in pharmacoepidemiology. *American Journal of Epidemiology*. 2018;187(9):1951-1961. doi:10.1093/aje/kwy078
22. Rosenbaum PR, Rubin DB. The central role of the propensity score in observational studies for causal effects. *Biometrika*. 1983;70(1):41-55. doi:10.1093/biomet/70.1.41
23. Rosenbaum PR, Rubin DB. The Central Role of the Propensity Score in Observational Studies for Causal Effects. In: *Matched Sampling for Causal Effects*. Vol 70. Cambridge University Press; 2006:170-184. doi:10.1017/CBO9780511810725.016
24. RStudio Team. RStudio: Integrated Development for R. Published online 2020. http://www.rstudio.com/
25. Saul B. Smd: Compute standardized mean differences. Published online 2020. https://CRAN.R-project.org/package=smd
26. Sievert C. Interactive web-based data visualization with r, plotly, and shiny. Published online 2020. https://plotly-r.com
27. Tarr G, Müller S, Welsh AH. mplot: An R package for graphical model stability and variable selection procedures. *Journal of Statistical Software*. 2018;83(9):1-28. doi:10.18637/jss.v083.i09
28. Venables WN, Ripley BD. Modern applied statistics with s. Published online 2002. https://www.stats.ox.ac.uk/pub/MASS4/
29. Wang J. To use or not to use propensity score matching? *Pharmaceutical Statistics*. 2021;20(1):15-24. doi:10.1002/pst.2051
30. Wickham H. ggplot2: Elegant graphics for data analysis. Published online 2016. https://ggplot2.tidyverse.org
31. Wickham H, François R, Henry L, Müller K. Dplyr: A grammar of data manipulation. Published online 2022. https://CRAN.R-project.org/package=dplyr
32. Wickham H, Hesselberth J, Salmon M. Pkgdown: Make static HTML documentation for a package. Published online 2022. https://CRAN.R-project.org/package=pkgdown