fairadapt: Causal Reasoning for Fair Data Pre-processing

Drago Plečko
ETH Zürich

Nicolas Bennett
ETH Zürich

Nicolai Meinshausen
ETH Zürich

Abstract

Machine learning algorithms are useful for various predictions tasks, but they can also learn how to discriminate, based on gender, race or other sensitive attributes. This realization gave rise to the field of fair machine learning, which aims to recognize, quantify and ultimately mitigate such algorithmic bias. This manuscript describes the R-package **fairadapt**, which implements a causal inference pre-processing method. By making use of a causal graphical model alongside the observed data, the method can be used to address hypothetical questions of the form "What would my salary have been, had I been of a different gender/race?". Such individual level counterfactual reasoning can help eliminate discrimination and help justify fair decisions. We also discuss appropriate relaxations which assume that certain causal pathways from the sensitive attribute to the outcome are not discriminatory.

Keywords: algorithmic fairness, causal inference, machine learning.

1. Introduction

Machine learning algorithms have become prevalent tools for decision-making in socially sensitive situations, such as determining credit-score ratings or predicting recidivism during parole. It has been recognized that algorithms are capable of learning societal biases, for example with respect to race (Larson, Mattu, Kirchner, and Angwin 2016b) or gender (Lambrecht and Tucker 2019; Blau and Kahn 2003), and this realization seeded an important debate in the machine learning community about fairness of algorithms and their impact on decision-making.

1.1. Definitions of Fairness

In order to define and measure discrimination, existing intuitive notions have been mathematically formalized, thereby providing fairness metrics. To this day, various different notions of fairness exist, which are sometimes mutually incompatible (Corbett-Davies and Goel 2018), meaning not of all of them can be achieved for a constructed predictor $\widehat{Y}$ simultaneously. There currently is no consensus on which notion of fairness is the correct one among the three prevalent notions discussed in the literature:

(1) *Demographic parity* (Darlington 1971) requires the protected attribute A (gender, race,

religion etc.) to be independent of a constructed classifier or regressor $\widehat{Y}$, written as

$$\widehat{Y} \perp\!\!\!\perp A.$$

(2) *Equality of odds* (Hardt, Price, Srebro *et al.* 2016), requires equal false positive and false negative rates of classifier $\widehat{Y}$ between different groups (females and males for example) written as

$$\widehat{Y} \perp\!\!\!\perp A \mid Y.$$

(3) *Calibration* (Chouldechova 2017) requires the protected attribute to be independent of the actual outcome given the prediction

$$Y \perp\!\!\!\perp A \mid \widehat{Y}.$$

Intuitively, this means that the protected attribute A does not offer additional information about the outcome Y once we know what the prediction $\widehat{Y}$ is.

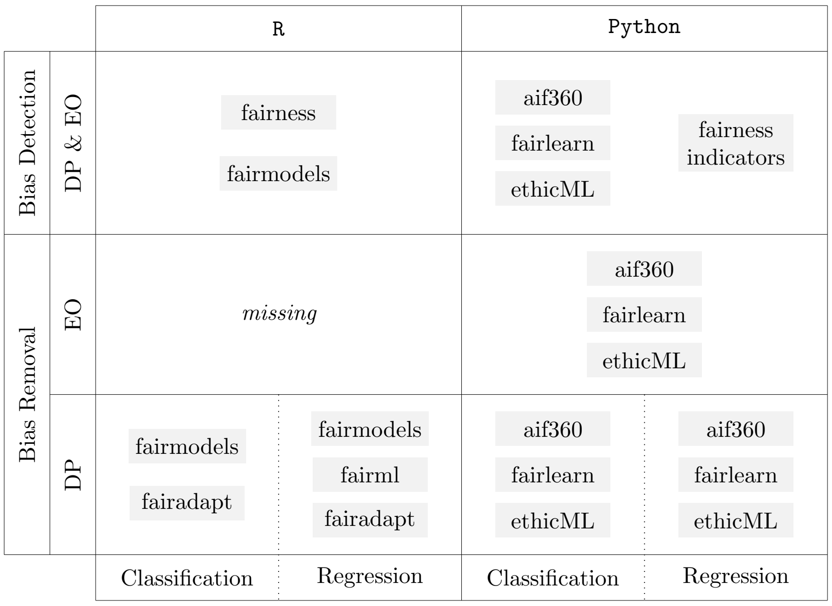
1.2. Fairness Tasks

Apart from choosing a notion of fairness most appropriate to the setting that is analyzed, it is also good to separate out two somewhat different tasks in fairness analysis. The first, usually simpler task is that of *bias quantification*, or bias measurement. In this case, we are interested in computing metrics from our dataset that quantify whether a definition is satisfied or not. Concretely, suppose we have a dataset in which $\widehat{Y}$ is the predicted salary and A is sex. If we are interested in demographic parity, $\widehat{Y} \perp\!\!\!\perp A$, we can compute the average difference in salary between the male/female groups, that is $\mathbb{E}[\widehat{Y} \mid A = \text{male}] - \mathbb{E}[\widehat{Y} \mid A = \text{female}]$. For other definitions, different metrics would be appropriate.

Sometimes, performing the first step of bias measurement is not the end goal. Suppose that we are analyzing a dataset and compute a metric which shows there might be discrimination in the dataset (say a large salary gap between sexes in the example above). We then might be interested in computing new, more fair predictions (in which, say, the salary gap is lower). This second task is that of *bias removal*, or bias mitigation, in which we want to remove from our predictions the bias that we previously found.

In Figure 1 we provide a graphical overview of some of the available software in python and R, which are the languages most commonly used for fairness analysis. For a given task and outcome/definition of fairness, we show existing software packages for both the classification and regression settings. Since calibration is often satisfied by fitting an unconstrained model, software capable of performing calibration was not included.

The software landscape for fair ML in python is more mature. Currently three well developed packages exist, capable of computing various fairness metrics and handling both classification and regression, suitable for satisfying either equality of odds or demographic parity. These repositories are **aif360** (Bellamy, Dey, Hind, Hoffman, Houde, Kannan, Lohia, Martino, Mehta, Mojsilovic, Nagar, Ramamurthy, Richards, Saha, Sattigeri, Singh, Varshney, and Zhang 2018) maintained by IBM, **fairlearn** (Bird, Dudík, Edgar, Horn, Lutz, Milan, Sameki, Wallach, and Walker 2020) maintained by Microsoft and **EthicML**. A further package, **fairness indicators**, is narrower in scope but suitable for computing fairness metrics on very large datasets.



		R		Python	
		Classification	Regression	Classification	Regression
Bias Detection	DP & EO	fairness, fairmodels		aif360, fairlearn, ethicML, fairness indicators	
Bias Removal	EO	*missing*		aif360, fairlearn, ethicML	
	DP	fairmodels, fairadapt	fairmodels, fairml, fairadapt	aif360, fairlearn, ethicML	aif360, fairlearn, ethicML

Figure 1: Software options for fair data analysis in R and python. In the left column, nodes correspond to R-packages and in the right column to python modules. DP stands for demographic parity, EO equality of odds.

For R, i.e., distributed via CRAN, there are fewer packages that relate to fair machine learning (see Figure 1 and note that there are no packages implementing equality of odds). Available packages include **fairml** (Scutari 2021), which implements the non-convex method of Komiyama, Takeda, Honda, and Shimao (2018), as well as **fairness** (Kozodoi and V. Varga 2021) and **fairmodels** (Wiśniewski and Biecek 2021), which serve as diagnostic tools for measuring algorithmic bias and provide several pre- and post-processing methods for bias mitigation. The **fairadapt** package described in this manuscript is a bias removal method which is able to interpolate between demographic parity and calibration notions, and it works for both regression and classification settings. In particular, **fairadapt** is the only software in Figure 1 which is *causally aware*, that is, bias removal can be connected to actual mechanisms that are causally related to discrimination.

1.3. A Causal Approach

The discussion on algorithmic fairness is, however, not restricted to the machine learning domain. There are many legal and philosophical aspects that have arisen. For example, the legal distinction between disparate impact and disparate treatment (McGinley 2011) is important for assessing fairness from a judicial point of view. This in turn emphasizes the importance of the interpretation behind the decision-making process, which is often not the case with black-box machine learning algorithms. For this reason, research in fairness through a causal inference lens has gained attention.

A possible approach to fairness is the use of counterfactual reasoning (Galles and Pearl 1998), which allows for arguing what might have happened under different circumstances that never actually materialized, thereby providing a tool for understanding and quantifying discrimination. For example, one might ask how a change in sex would affect the probability of a specific candidate being accepted for a given job opening. This approach has motivated another notion of fairness, termed *counterfactual fairness* (Kusner, Loftus, Russell, and Silva 2017), which states that the decision made, should remain fixed, even if, hypothetically, some parameters such as race or gender were to be changed (this can be written succinctly as $\widehat{Y}(a) = \widehat{Y}(a')$ in the potential outcomes notation). Causal inference can also be used for decomposition of the parity gap measure (Zhang and Bareinboim 2018), $\mathbb{P}(\widehat{Y} = 1 \mid A = a) - \mathbb{P}(\widehat{Y} = 1 \mid A = a')$, into direct, indirect and spurious components (yielding further insights into the demographic parity as a criterion), as well as the introduction of so-called resolving variables Kilbertus, Carulla, Parascandolo, Hardt, Janzing, and Schölkopf (2017), in order to relax the possibly prohibitively strong notion of demographic parity.

The following sections describe an implementation of the fair data adaptation method outlined in Plecko and Meinshausen (2020), which combines the notions of counterfactual fairness and resolving variables, and explicitly computes counterfactual values for individuals. The implementation is available as the R-package **fairadapt** from CRAN.

1.4. Novelty in the package

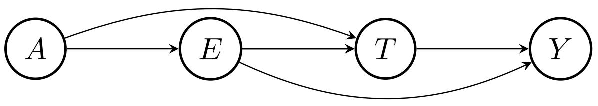
A first version of **fairadapt** was published with the original manuscript (Plecko and Meinshausen 2020). The software has since been developed further and novelty in the package presented in this manuscript includes the following:

- The methodology has been extended from the Markovian to the Semi-Markovian case (allowing noise variables to be correlated), which generalizes the scope of applications.
- Backdoor paths into the protected attribute A are now allowed, meaning that the attribute A does not need to be a root node of the causal graph.
- The user is provided with functionality for uncertainty quantification of the estimates.
- Introduction of S3 classes `fairadapt` and `fairadaptBoot`, alongside associated methods, provides a more formalized implementation.
- More flexibility is allowed for in the quantile learning step, including different algorithms for quantile regression (linear, forest based, neural networks). Additionally, there is also a possibility of specifying a custom quantile learning function (utilizing S3 dispatch).
- The user is provided with functionality for assessing the quality of the quantile regression fit, which allows for optimizing the hyperparameters of the more flexible algorithms.

The rest of the manuscript is organized as follows: In Section 2 we describe the methodology behind **fairadapt**, together with reviewing some important concepts of causal inference. In Section 3 we discuss implementation details and provide some general user guidance, followed by Section 4 in which we discuss how to perform uncertainty quantification with **fairadapt**. Section 5 illustrates the use of **fairadapt** through a large, real-world dataset and a hypothetical fairness application. Finally, in Section 6 we elaborate on some extensions, such as Semi-Markovian models and resolving variables.

2. Methodology

First, the intuition behind **fairadapt** is described using an example, followed by a more rigorous mathematical formulation, using Markovian structural causal models (SCMs). Some



$$\begin{tikzcd} A \arrow[r] \arrow[rr, bend left] & E \arrow[r] \arrow[rr, bend right] & T \arrow[r] & Y \end{tikzcd}$$

Figure 2: University admission based on previous educational achievement E combined with and an admissions test score T. The protected attribute A, encoding gender, has an unwanted causal effect on E, T, as well as Y, which represents the final score used for the admission decision.

relevant extensions, such as the Semi-Markovian case and the introduction of so called *resolving variables*, are discussed in Section 6.

2.1. University Admission Example

Consider the example of university admission based on previous educational achievement and an admissions test. Variable A is the protected attribute, describing candidate gender, with $A = a$ corresponding to females and $A = a'$ to males. Furthermore, let E be educational achievement (measured for example by grades achieved in school) and T the result of an admissions test for further education. Finally, let Y be the outcome of interest (final score) upon which admission to further education is decided. Edges in the graph in Figure 2 indicate how variables affect one another.

Attribute A, gender, has a causal effect on variables E, T, as well as Y, and we wish to eliminate this effect. For each individual with observed values (a, e, t, y) we want to find a mapping

$$(a, e, t, y) \longrightarrow (a^{(fp)}, e^{(fp)}, t^{(fp)}, y^{(fp)}),$$

which represents the value the person would have obtained in an alternative world where everyone was female. Explicitly, to a male person with education value e, we assign the transformed value $e^{(fp)}$ chosen such that

$$\mathbb{P}(E \geq e \mid A = a') = \mathbb{P}(E \geq e^{(fp)} \mid A = a).$$

The key idea is that the *relative educational achievement within the subgroup* remains constant if the protected attribute gender is modified. If, for example, a male has a higher educational achievement value than 70% of males in the dataset, we assume that he would also be better than 70% of females had he been female[1]. After computing transformed educational achievement values corresponding to the *female* world ($E^{(fp)}$), the transformed test score values $T^{(fp)}$ can be calculated in a similar fashion, but conditioned on educational achievement. That is, a male with values $(E, T) = (e, t)$ is assigned a test score $t^{(fp)}$ such that

$$\mathbb{P}(T \geq t \mid E = e, A = a') = \mathbb{P}(T \geq t^{(fp)} \mid E = e^{(fp)}, A = a),$$

where the value $e^{(fp)}$ was obtained in the previous step. This step can be visualized as shown in Figure 3.

[1]This assumption of course is not empirically testable, as it is impossible to observe both a female and a male version of the same individual.

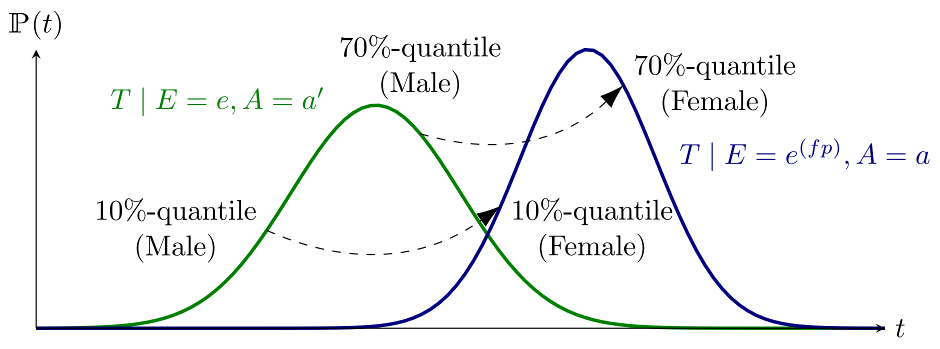


Figure 3: A graphical visualization of the quantile matching procedure. Given a male with a test score corresponding to the 70% quantile, we would hypothesize, that if the gender was changed, the individual would have achieved a test score corresponding to the 70% quantile of the female distribution.

As a final step, the outcome variable Y remains to be adjusted. The adaptation is based on the same principle as above, using transformed values of both education and the test score. The resulting value $y^{(fp)}$ of $Y = y$ satisfies

$$\mathbb{P}(Y \geq y \mid E = e, T = t, A = a') = \mathbb{P}(Y \geq y^{(fp)} \mid E = e^{(fp)}, T = t^{(fp)}, A = a).$$

This form of counterfactual correction is known as *recursive substitution* (Pearl 2009, Chapter 7) and is described more formally in the following sections. The reader who is satisfied with the intuitive notion provided by the above example is encouraged to go straight to Section 3.

2.2. Structural Causal Models

In order to describe the causal mechanisms of a system, a *structural causal model* (SCM) can be hypothesized, which fully encodes the assumed data-generating process. An SCM is represented by a 4-tuple $\langle V, U, \mathcal{F}, \mathbb{P}(u)\rangle$, where

- $V = \{V_1, \ldots, V_n\}$ is the set of observed (endogenous) variables.
- $U = \{U_1, \ldots, U_n\}$ are latent (exogenous) variables.
- $\mathcal{F} = \{f_1, \ldots, f_n\}$ is the set of functions determining V, $v_i \leftarrow f_i(\text{pa}(v_i), u_i)$, where $\text{pa}(V_i) \subset V, U_i \subset U$ are the functional arguments of f_i and $\text{pa}(V_i)$ denotes the parent vertices of V_i.
- $\mathbb{P}(u)$ is a distribution over the exogenous variables U.

Any particular SCM is accompanied by a graphical model $\mathcal{G}$ (a directed acyclic graph), which summarizes which functional arguments are necessary for computing the values of each V_i and therefore, how variables affect one another. We assume throughout, without loss of generality, that

(i) $f_i(\text{pa}(v_i), u_i)$ is increasing in u_i for every fixed $\text{pa}(v_i)$.
(ii) Exogenous variables U_i are uniformly distributed on $[0, 1]$.

In the following section, we discuss the Markovian case in which all exogenous variables U_i are mutually independent. The Semi-Markovian case, where variables U_i are allowed to have a mutual dependency structure, alongside the extension introducing *resolving variables*, are discussed in Section 6.

2.3. Markovian SCM Formulation

Let Y take values in $\mathbb{R}$ and represent an outcome of interest and A be the protected attribute taking two values a, a'. The goal is to describe a pre-processing method which transforms the entire data V into its fair version $V^{(fp)}$. This can be achieved by computing the counterfactual values $V(A = a)$, which would have been observed if the protected attribute was fixed to a baseline value $A = a$ for the entire sample.

More formally, going back to the *university admission* example above, we want to align the distributions

$$V_i \mid \text{pa}(V_i), A = a \text{ and } V_i \mid \text{pa}(V_i), A = a',$$

meaning that the distribution of V_i should be indistinguishable for both female and male applicants, for every variable V_i. Since each function f_i of the original SCM is reparametrized so that $f_i(\text{pa}(v_i), u_i)$ is increasing in u_i for every fixed $\text{pa}(v_i)$, and also due to variables U_i being uniformly distributed on $[0, 1]$, variables U_i can be seen as the latent *quantiles*.

The algorithm proposed for data adaption proceeds by fixing $A = a$, followed by iterating over descendants of the protected attribute A, sorted in topological order. For each V_i, the assignment function f_i and the corresponding quantiles U_i are inferred. Finally, transformed values ${V_i}^{(fp)}$ are obtained by evaluating f_i, using quantiles U_i and the transformed parents $\text{pa}(V_i)^{(fp)}$ (see Algorithm 1).

Algorithm 1: Fair Data Adaptation

```
Input: $V$, causal graph $\mathcal{G}$
set $A \leftarrow a$ for everyone
for $V_i \in \text{de}(A)$ in topological order do
    learn function $V_i \leftarrow f_i(\text{pa}(V_i), U_i)$
    infer quantiles $U_i$ associated with the variable $V_i$
    transform values as ${V_i}^{(fp)} \leftarrow f_i(\text{pa}(V_i)^{(fp)}, U_i)$
end
return $V^{(fp)}$
```

The assignment functions f_i of the SCM are of course unknown, but are non-parametrically inferred at each step. Algorithm 1 obtains the counterfactual values $V(A = a)$ under the $do(A = a)$ intervention for each individual, while keeping the latent quantiles U fixed. In the case of continuous variables, the latent quantiles U can be determined exactly, while for the discrete case, this is more subtle and described in Plecko and Meinshausen (2020, Section 5).

3. Implementation

In order to perform fair data adaption using the **fairadapt** package, the function `fairadapt()` is exported, which returns an object of class `fairadapt`. Implementations of the base R S3 generics `print()`, `plot()` and `predict()`, as well as the generic `autoplot()`, exported from **ggplot2** (Wickham 2016), are provided for `fairadapt` objects, alongside `fairadapt`-specific implementations of S3 generics `visualizeGraph()`, `adaptedData()` and `fairTwins()`. Finally, an extension mechanism is available via the S3 generic function `computeQuants()`, which is used for performing the quantile learning step.

The following sections show how the listed methods relate to one another alongside their intended use, beginning with constructing a call to `fairadapt()`. The most important arguments of `fairadapt()` include:

- `formula`: Argument of type `formula`, specifying the dependent and explanatory variables.
- `adj.mat`: Argument of type `matrix`, encoding the adjacency matrix.
- `train.data` and `test.data`: Both of type `data.frame`, representing the respective datasets.
- `prot.attr`: Scalar-valued argument of type `character` identifying the protected attribute. Has to correspond to a column name in the `train.data` argument.

It is worth emphasizing the possible data types that can be used in the `train.data` argument. We note the following:

(i) *Attribute A*: the protected attribute is limited to the binary case. The `prot.attr` column in `train.data` can be of any data type coercible to a `factor`, but can only take two distinct values. Otherwise an error is thrown.

(ii) *Outcome Y*: the dependent variable specified on the left hand side of the `formula` argument can be either a `numeric`, `integer`, `logical` or ordered `factor`. Unordered `factor` or `character` inputs for the outcome variable will result in an error.

(iii) *Remaining attributes X*: all other attributes do not have limitations. Unordered `factor` or `character` inputs can be used in the `train.data` argument.

As a quick demonstration of performing fair data adaption, we load the `uni_admission` dataset provided by **fairadapt**, consisting of synthetic university admission data of 1000 students. We subset this data, using the first `n_samp` rows as training data (`uni_trn`) and the following `n_samp` rows as testing data (`uni_tst`). Furthermore, we construct an adjacency matrix `uni_adj` with edges `gender` $\rightarrow$ `edu`, `gender` $\rightarrow$ `test`, `edu` $\rightarrow$ `test`, `edu` $\rightarrow$ `score`, and `test` $\rightarrow$ `score`. As the protected attribute, we choose `gender`.

```
R> n_samp <- 200
R>
R> uni_dat <- data("uni_admission", package = "fairadapt")
R> uni_dat <- uni_admission[seq_len(2 * n_samp), ]
R>
R> head(uni_dat)
```

```
  gender       edu        test     score
1      1 1.3499572 1.617739679 1.9501728
```

```
2      0 -1.9779234 -3.121796235 -2.3502495
3      1  0.6263626  0.530034686  0.6285619
4      1  0.8142112  0.004573003  0.7064857
5      1  1.8415242  1.193677123  0.3678313
6      1 -0.3252752 -2.004123561 -1.5993848

R> uni_trn <- head(uni_dat, n = n_samp)
R> uni_tst <- tail(uni_dat, n = n_samp)
R>
R> uni_dim <- c(       "gender", "edu", "test", "score")
R> uni_adj <- matrix(c(       0,     1,      1,       0,
+                             0,     0,      1,       1,
+                             0,     0,      0,       1,
+                             0,     0,      0,       0),
+                    ncol = length(uni_dim),
+                    dimnames = rep(list(uni_dim), 2),
+                    byrow = TRUE)
R>
R> basic <- fairadapt(score ~ ., train.data = uni_trn,
+                      test.data = uni_tst, adj.mat = uni_adj,
+                      prot.attr = "gender")
R>
R> basic


Call:
fairadapt(formula = score ~ ., prot.attr = "gender", adj.mat = uni_adj,
    train.data = uni_trn, test.data = uni_tst)



Adapting variables:
  score, edu, test

Based on protected attribute gender

  AND

Based on causal graph:
       score gender edu test
score      0      0   0    0
gender     0      0   1    1
edu        1      0   0    1
test       1      0   0    0
```

The implicitly called `print()` method in the previous code block displays some information about how `fairadapt()` was called, such as number of variables, the protected attribute and also the total variation before and after adaptation, defined as

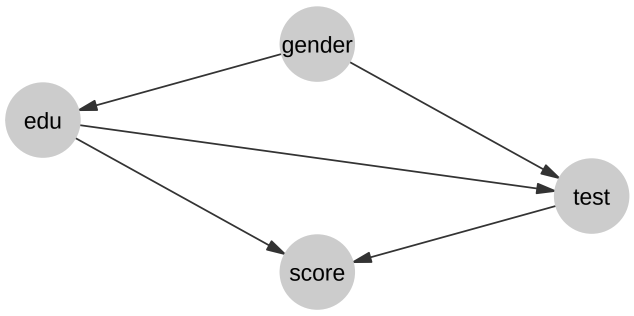


Figure 4: The underlying graphical model corresponding to the university admission example (also shown in Figure 2).

$$\mathbb{E}[Y \mid A = a] - \mathbb{E}[Y \mid A = a'] \text{ and } \mathbb{E}[Y^{(fp)} \mid A = a] - \mathbb{E}[Y^{(fp)} \mid A = a'],$$

respectively, where Y denotes the outcome variable. Total variation in the case of a binary outcome Y, corresponds to the parity gap.

3.1. Specifying the Graphical Model

As the algorithm used for fair data adaption in `fairadapt()` requires access to the underlying graphical causal model $\mathcal{G}$ (see Algorithm 1), a corresponding adjacency matrix can be passed as `adj.mat` argument. The convenience function `graphModel()` turns a graph specified as an adjacency matrix into an annotated graph using the **igraph** package (Csardi and Nepusz 2006). While exported for the user to invoke manually, this function is called as part of the `fairadapt()` routine and the resulting `igraph` object can be visualized by calling the S3 generic `visualizeGraph()`, exported from `fairadapt` on an object of class `fairadapt`.

```
R> uni_graph <- graphModel(uni_adj)
```

A visualization of the `igraph` object returned by `graphModel()` is available from Figure 4. The graph shown is equivalent to that of Figure 2 as they both represent the same causal model.

3.2. Quantile Learning Step

The training step in `fairadapt()` that follows the typical machine learning workflow would be as follows. We first call the `fairadapt()` function and perform the quantile learning step on the `train.data`. After this, at a later stage we call the `predict()` function on the returned `fairadapt` S3 object in order to perform data adaption on new test data. Alternatively, it is also possible to input both `train.data` and `test.data` arguments directly to `fairadapt()`, which then transforms both of these arguments. This one-step procedure might be considered when the proportion of test samples compared to train samples is large, and when the `train.data` has a relatively small sample size. The benefit of this approach is that, even though the outcome Y is not available, other attributes X of `test.data` can be used for learning the quantiles.

	Random Forests	Neural Networks	Linear Regression
R-package	**ranger**	**qrnn**	**quantreg**
`quant.method`	`rangerQuants`	`mcqrnnQuants`	`linearQuants`
complexity	$O(np\log n)$	$O(npn_{\text{epochs}})$	$O(p^2n)$
default parameters	$ntrees = 500$ $mtry = \sqrt{p}$	1 hidden layer fully connected feed-forward network	`"br"` method of Barrodale and Roberts used for fitting
$T_{\text{uni}}(200)$	0.6	33.9	0.4
$T_{\text{uni}}(500)$	1.6	175.8	0.6

Table 1: Summary table of different quantile regression methods. n is the number of samples, p number of covariates, n_{epochs} number of training epochs for the neural network. $T_{\text{uni}}(n)$ denotes the runtime of different methods on the university admission dataset, with n training and n testing samples.

The data frames passed as `train.data` and `test.data` are required to have column names which also appear in the row and column names of the adjacency matrix, alongside the protected attribute A, passed as scalar-valued character vector `prot.attr`. The `test.data` argument defaults to `NULL`, with the intention that `test.data` is specified at a later stage.

Quantile Methods

The quantile learning step of Algorithm 1 can in principle be carried out by several methods, three of which are implemented in **fairadapt**:

- Quantile Regression Forests (Meinshausen 2006; Wright and Ziegler 2015).
- Non-crossing quantile neural networks (Cannon 2018, 2015).
- Linear Quantile Regression (Koenker and Hallock 2001; Koenker, Portnoy, Ng, Zeileis, Grosjean, and Ripley 2018).

Using linear quantile regression is the most efficient option in terms of runtime, while for non-parametric models and mixed data, the random forest approach is well-suited, at the expense of a slight increase in runtime. The neural network approach is substantially slower when compared to linear and random forest estimators and consequently does not scale well to large sample sizes. As default, the random forest based approach is implemented, due to its non-parametric nature and computational speed. However, for smaller sample sizes, the neural network approach can also demonstrate competitive performance. A quick summary outlining some differences between the three natively supported methods is available from Table 3.2.1.

Influencing the Fit

The quantile methods shown in Table 3.2.1 make calls to specific functions that perform quantile regression. These functions take varying arguments and for that reason, `fairadapt()` forwards arguments passed as `...` to the function specified as `quant.method`.

Computational Speed An important consideration in choosing values for optional arguments of specific quantile regression functions is computational speed. For example, `rangerQuants()` internally calls the `ranger()` function (from **ranger**) and with respect to computational speed, an important argument is `num.trees`, the number of trees used when building the quantile regression forest. Clearly, choosing a smaller number of trees will be faster, but at the same time will result in a poorer fit with larger variance.

Similarly, `mcqrnnQuants()` internally calls `mcqrnn.fit()` (from **qrnn**), which has a number of arguments that can be used for adapting the underlying neural network. In terms of computational speed, the most important arguments are `n.trials` (number of repeated initializations used to avoid local minima) and `iter.max` (maximum number of iterations of the optimization). Choosing smaller values will reduce the runtime. Lastly, function `linearQuants()` internally calls `rq()` (from **quantreg**). This function is less flexible, since the model is linear. However, its `method` argument can be used when the number of samples becomes large (using `method` equal to `"fn"` or `"pfn"` utilizes the Frisch-Newton interior point method, which might be preferable for large samples).

Fit Quality Both `rangerQuants()` and `mcqrnnQuants()` expose flexible machine learning tools with several parameters that impact fit quality. In order to optimize the fitting procedure by tuning these parameters, we need a way of assessing the quality of our fit. In the context of quantile regression we can estimate the expected τ-quantile loss function,

$$\mathbb{E}[\rho_\tau(V_i, \mu_\tau(\text{pa}(V_i)))], \tag{1}$$

where $\mu_\tau(\text{pa}(V_i))$ is the function predicting the τ-quantile of variable V_i using the parents $\text{pa}(V_i)$ and ρ_τ is the asymmetric L1 loss function whose minimizer is the τ-quantile. The function ρ_τ is given by

$$\rho_\tau(x, y) = \begin{cases} \tau(x-y), & \text{for } x \geq y \\ (1-\tau)(y-x), & \text{for } x < y. \end{cases}$$

A smaller empirical loss based on Equation (1) corresponds to better fit quality. For hyperparameter tuning we can perform cross-validation (fitting the quantile regression on separate folds), which is directly available within the `fairadapt()` function. The argument `eval.qfit` has a default value `NULL`, but if this argument is given a positive integer value, then it is used as the number of folds for performing cross-validation.

We compute the average empirical loss $\widehat{\mathbb{E}}[\rho_\tau(V_i, \mu_\tau(\text{pa}(V_i)))]$ for each variable V_i and $\tau = 0.25, 0.5, 0.75$ (corresponding to 25%, 50% and 75% quantiles). The average of these three values is reported at the end, and can be extracted from the resulting `fairadapt` object using the `quantFit()` method:

```
R> fit_qual <- fairadapt(score ~ ., train.data = uni_trn,
+                        adj.mat = uni_adj, prot.attr = "gender",
+                        eval.qfit = 3L)
R>
R> quantFit(fit_qual)
```

```
      edu      test     score
0.3513718 0.2493164 0.3299460
```

The function returns the quality of the quantile fit for each variable. A very reasonable objective to minimize is the average of these values, by iterating over a grid of possible values of the tuning parameters. The interesting parameters to optimize for the two methods include:

(i) for `ranger()`: parameters `mtry` (number of candidate variables in each split), `min.node.size` (size of leaf nodes after which splitting) and `max.depth` (maximum depth of each tree),
(ii) for `mcqrnn()`: parameters `n.hidden`, `n.hidden2` (number of nodes in the first and second hidden layers), `Th` (activation function), `method` (optimizer to be used), and a range of other parameters that are fed to the chosen optimizer via ellipsis.

Given reasonable default values for the included qunatile methods, optimizing the quantile fit should be of interest mostly to advanced users (and hence we do not perform parameter tuning explicitly here).

Extending to Custom Methods

The above set of methods is not exhaustive. Further options are conceivable and therefore **fairadapt** provides an extension mechanism to account for this. The `fairadapt()` argument `quant.method` expects a function to be passed, a call to which will be constructed with three unnamed arguments:

1. A `data.frame` containing data to be used for quantile regression. This will either be the `data.frame` passed as `train.data`, or depending on whether `test.data` was specified, a row-bound version of train and test datasets.
2. A logical flag, indicating whether the protected attribute is the root node of the causal graph. If the attribute A is a root node, we know that $\mathbb{P}(X \mid \text{do}(A = a)) = \mathbb{P}(X \mid A = a)$. Therefore, the interventional and conditional distributions are in this case the same, and we can leverage this knowledge in the quantile learning procedure, by splitting the data into $A = 0$ and $A = 1$ groups.
3. A `logical` vector of length `nrow(data)`, indicating which rows in the `data.frame` passed as `data` correspond to samples with baseline values of the protected attribute.

Arguments passed as `...` to `fairadapt()` will be forwarded to the function specified as `quant.method` and passed after the first three fixed arguments listed above. The return value of the function passed as `quant.method` is expected to be an S3-classed object. This object should represent the conditional distribution $V_i \mid \text{pa}(V_i)$ (see function `rangerQuants()` for an example). Additionally, the object should have an implementation of the S3 generic function `computeQuants()` available. For each row $(v_i, \text{pa}(v_i))$ of the `data` argument, the `computeQuants()` function uses the S3 object to perform the following steps:

(i) Infer the quantile of $v_i \mid \text{pa}(v_i)$.
(ii) Compute the counterfactual value $v_i^{(fp)}$ under the change of protected attribute, using the counterfactual values of parents $\text{pa}(v_i^{(fp)})$ computed in previous steps (values $\text{pa}(v_i^{(fp)})$ are contained in the `newdata` argument).

For an example, see the `ranger` specific method for `computeQuants()`, `computeQuants.ranger()`.

3.3. Fair-Twin Inspection

We now turn to a useful property of **fairadapt** with respect to exploring decisions for different individuals in the dataset. The university admission example presented in Section 2 demonstrates how to compute counterfactual values for an individual while preserving their relative educational achievement. Setting candidate gender as the protected attribute and gender level *female* as baseline value, for a *male* student with values (a, e, t, y), his *fair-twin* values $(a^{(fp)}, e^{(fp)}, t^{(fp)}, y^{(fp)})$, i.e., the values the student would have obtained, had he been *female*, are computed. These values can be retrieved from a `fairadapt` object by calling the S3-generic function `fairTwins()` as:

```
R> ft_basic <- fairTwins(basic, train.id = seq_len(n_samp))
R> head(ft_basic, n = 3)
```

```
  gender      score score_adapted        edu edu_adapted       test
1      1  1.9501728     0.8214544  1.3499572   0.6744928  1.6177397
2      0 -2.3502495    -2.3502495 -1.9779234  -1.9779234 -3.1217962
3      1  0.6285619     0.0112887  0.6263626  -0.1594450  0.5300347
  test_adapted
1    0.3114252
2   -3.1217962
3   -0.4510660
```

In this example, we compute the values in a *female* world. Therefore, for *female* applicants, the values remain fixed, while for *male* applicants the values are adapted, as can be seen from the output. Having access to explicit counterfactual instances as above might help justify fair decisions in practice or help guide the choice of the assumed causal model and resolving variables (see Section 6 for resolving variables).

4. Uncertainty Quantification

The user might naturally be interested in uncertainty quantification of the procedure performed in `fairadapt()`. Before discussing how this can be achieved, we first give a graphical sketch of the typical workflow when using `fairadapt()` (see Figure 5).

Such a workflow can be described as follows. We start from the training data $\mathcal{D}_{train}$ and the causal graph $\mathcal{G}$. These two arguments are used as inputs of the `fairadapt()` function, which returns the transformed training data $\widetilde{\mathcal{D}}_{train}$ contained in a `fairadapt` S3 object. This `fairadapt` object, alongside test data $\mathcal{D}_{test}$ are then used as inputs to the `predict()` function, which returns the transformed test data $\widetilde{\mathcal{D}}_{test}$. Often, the end goal is to obtain fair predictions on the test data, and to this end we need to train a classifier/regressor. Either the training data $\mathcal{D}_{train}$ or its transformed counterpart $\widetilde{\mathcal{D}}_{train}$ can be used for building a predictor. The predictor then needs to be applied to the transformed train data $\widetilde{\mathcal{D}}_{test}$. Building on the graphical visualization in Figure 5, which serves as a mental map of our workflow, we can now explain the distinct sources of uncertainty that can be considered:

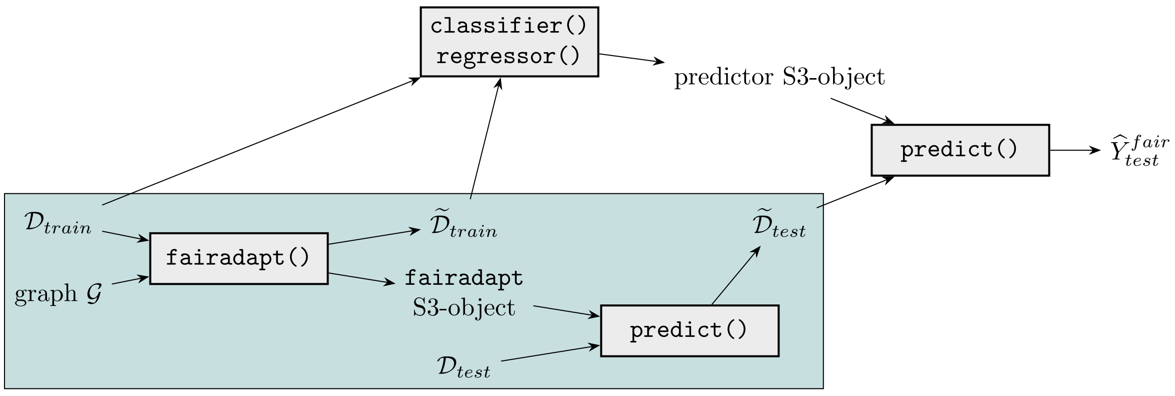


Figure 5: The typical workflow when using **fairadapt**. The shaded region represents the fair pre-processing which happens within the **fairadapt** package. Often, this is followed by applying a regressor or a classifier to the transformed data, in order to obtain fair predictions. The latter part is up to the user and not included in **fairadapt**.

- **Finite sample uncertainty**: The first, commonly encountered source of uncertainty is the one induced by a finite sample size. The training data $\mathcal{D}_{train}$ has a finite size, and for this reason inferences made using this data are imperfect. We wish to quantify the error in the predictions $\widehat{Y}_{test}^{fair}$ introduced by the finite sample size of $\mathcal{D}_{train}$. As Figure 5 shows, the training data $\mathcal{D}_{train}$ affects the resulting fair predictions in two ways. Firstly, it affects the value of the transformed test data $\widetilde{\mathcal{D}}_{test}$ (mediated by the fairadapt S3 object). Secondly, it affects the predictions $\widehat{Y}_{test}^{fair}$ also through the predictor (since it is the input to the regressor classifier). These finite sample uncertainties can be analyzed using *bootstrap* (Efron and Tibshirani 1994). This means that we repeat the procedure in Figure 5 many times, each time taking a different bootstrap sample of the training data. Below we will show how this can be done with the `fairadaptBoot()` function.

- **Inherent uncertainty in the quantiles**: A second source of uncertainty arises from the uncertainty in quantile estimation and is specific to **fairadapt**. As described in Section 2 (see also Figure 3), the fairadapt procedure aims to preserve the relative quantile of the variable, when computing the $do(A = a)$ intervention. However, when we are working with variables that are not continuous, defining a quantile becomes more difficult[2]. Therefore, in presence of discrete variables, due to the imperfect estimation of quantiles, the fairadapt procedure has some inherent randomness. This randomness would still persist even if we had infinite training samples in $\mathcal{D}_{train}$. Importantly, to achieve fair predictions, taking an expectation over this randomness is not feasible. For a detailed discussion of why this is the case, refer to (Plecko and Meinshausen 2020, Section 5).

For quantifying uncertainty, we use the `fairadaptBoot()` function, where the most important arguments are:

[2]For example in the case where we have a binary $X \in \{0, 1\}$, it is impossible to define what a 70% quantile is, as opposed to the continuous case (of a Gaussian variable X for example), where no such challenges exists.

- `formula`, `prot.attr`, `adj.mat`, `train.data` arguments are the same as for the `fairadapt()` function (see Section 3).
- `test.data`, a `data.frame` containing the test data, defaults to `NULL`. Whenever the test data equals `NULL`, then `save.object` must be `TRUE`.
- `save.object`, a `logical` scalar, indicating whether all the `fairadapt` S3 objects built in bootstrap repetitions should be saved. Default value is `FALSE`.
- `rand.mode`, a `character` scalar, taking values `"finsamp"`, `"quant"` or `"both"`, corresponding to considering finite sample uncertainty, quantile uncertainty, or both.

The function `fairadaptBoot()` returns an S3 object of class `fairadaptBoot`. Calling this function can be computationally expensive, both in terms of runtime and memory. Keeping the default value of `FALSE` for the `save.object` argument reduces the memory consumption substantially, with the drawback that `test.data` has to be provided directly to `fairadaptBoot()`, and that the resulting `fairadaptBoot` object cannot be reused for making predictions at a later stage. Passing `TRUE` to save.object, on the other hand, might consume more memory, but then the object can be reused over again for transforming new test data. This can be done by using the `predict()` function for the `fairadaptBoot` S3 object, to which a `newdata` argument is available.

For illustration purposes, we now compute bootstrap repetitions for finite sample uncertainty and quantile uncertainty on the COMPAS dataset (Larson, Mattu, Kirchner, and Angwin 2016a). We begin by loading the COMPAS dataset and constructing its causal graph:

```
R> cmp_dat <- data("compas", package = "fairadapt")
R> cmp_dat <- get(cmp_dat)
R>
R> cmp_mat <- matrix(0, nrow = ncol(cmp_dat), ncol = ncol(cmp_dat),
+                    dimnames = list(names(cmp_dat), names(cmp_dat)))
R>
R> cmp_mat[c("race", "sex", "age"),
+         c("juv_fel_count", "juv_misd_count",
+           "juv_other_count", "priors_count",
+           "c_charge_degree", "two_year_recid")] <- 1
R> cmp_mat[c("juv_fel_count", "juv_misd_count", "juv_other_count"),
+         c("priors_count", "c_charge_degree", "two_year_recid")] <- 1
R> cmp_mat["priors_count", c("c_charge_degree", "two_year_recid")] <- 1
R> cmp_mat["c_charge_degree", "two_year_recid"] <- 1
R>
R> head(cmp_dat)
```

```
   sex age      race juv_fel_count juv_misd_count juv_other_count
1 Male  69 Non-White             0              0               0
2 Male  34 Non-White             0              0               0
3 Male  24 Non-White             0              0               1
4 Male  23 Non-White             0              1               0
5 Male  43 Non-White             0              0               0
6 Male  44 Non-White             0              0               0
```

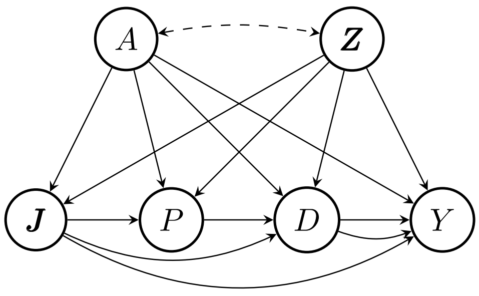


Figure 6: The causal graph for the COMPAS dataset. Z are demographic features, A is race, J juvenile offense counts, P priors count, D the degree of charge, and Y two year recidivism.

```
  priors_count c_charge_degree two_year_recid
1            0               F              0
2            0               F              1
3            4               F              1
4            1               F              0
5            2               F              0
6            0               M              0
```

The COMPAS dataset contains information on 7214 individuals from Broward County, Florida, who were released on parole. The variables include race, sex, age, juvenile offense counts, priors count and the degree of criminal charge. The outcome of interest is recidivism within two years and the protected attribute is race (taking values Non-White and White). A possible causal graph for the COMPAS dataset is given in Figure 6.

After loading the dataset, we run the `fairadaptBoot()` function twice, with two different values of the `rand.mode` argument. First, we consider only the finite sample uncertainty.

```
R> cmp_trn <- tail(cmp_dat, n = 700L)
R> cmp_tst <- head(cmp_dat, n = 100L)
R>
R> n_itr <- 5L

R> fa_boot_fin <- fairadaptBoot(two_year_recid ~ ., "race", cmp_mat,
+                                cmp_trn, cmp_tst, rand.mode = "finsamp",
+                                n.boot = n_itr)
```

Then, we re-run the bootstrap procedure, but by considering only the inherent quantile uncertainty, by setting the `rand.mode` argument to `"quant"`.

```
R> fa_boot_quant <- fairadaptBoot(two_year_recid ~ ., "race", cmp_mat,
+                                  cmp_trn, cmp_tst, rand.mode = "quant",
+                                  n.boot = n_itr)
```

The objects returned are of class `fairadaptBoot` and contain a list `adapt.test` which in turn contains different replicates of the adapted test data (the length of this list is `n.boot`, the number of bootstrap repetitions). Finally, to obtain predictions, we train a random forest classifier on the different bootstrap samples of `train.data` and apply it to all of the transformed datasets. In doing so, we make use of the `boot.ind` list, contained in `fairadaptBoot` object, representing row indices of all bootstrap repetitions.

```
R> fit_rf <- function(x) {
+   ranger(two_year_recid ~ ., cmp_trn[x, ], probability = TRUE)
+ }
R>
R> extract_pred <- function(x) x$predictions[, 2L]
R>
R> cmp_rf <- lapply(fa_boot_fin$boot.ind, fit_rf)
R>
R> pred_fin <- Map(predict, cmp_rf, fa_boot_fin$adapt.test)
R> pred_fin <- do.call(cbind, lapply(pred_fin, extract_pred))
R>
R> pred_quant <- Map(predict, cmp_rf, fa_boot_quant$adapt.test)
R> pred_quant <- do.call(cbind, lapply(pred_quant, extract_pred))
```

4.1. Analyzing the Uncertainty

In order to analyze the different sets of predictions $\widehat{Y}^{fair}_{test}$, two slightly different perspectives can be taken, and we elaborate on both in the following sections.

Decision-maker Analysis

The first way to analyze the uncertainty of the predictions is from the point of view of the decision-maker. By decision-maker, in this context we refer to the individual performing the analysis and obtaining a set of fair predictions. For a decision-maker, it is important to understand how sensitive the outcome of the classification is to uncertainties induced by both finite sample size and the inherent uncertainty induced by discrete variables. Let p_A be the vector of predicted probabilities on the `test.data`, with length `nrow(test.data)`. Denote `nrow(test.data)` with n_{test}. Let p_B be another vector of predicted probabilities (under a different bootstrap repetition). We can consider the following four metrics of uncertainty:

1. For each threshold $t \in [0, 1]$, we compute the decision sets D_A, D_B, which are obtained by selecting all individuals for whom the value of $p_A \geq t$ (and p_B respectively). We can then compute the Jaccard similarity of decision sets D_A, D_B, for each threshold t. By comparing many pairs of bootstrap repetitions in this way, we can estimate what the average Jaccard similarity for each threshold t is.

   ```
   R> jac_frm <- function(x, modes = "single") {
   +
   +   jac <- function(a, b) {
   +     intersection <- length(intersect(a, b))
   +     union <- length(a) + length(b) - intersection
   +     intersection / union
   ```

```
+   }
+
+   res <- lapply(
+     seq(quantile(x[, 1L], 0.05), quantile(x[, 1L], 0.95), 0.01),
+     function(tsh) {
+
+       ret <- replicate(100L, {
+         col <- sample(ncol(x), 2L)
+         jac(which(x[, col[1L]] > tsh), which(x[, col[2L]] > tsh))
+       })
+
+       data.frame(tsh = tsh, y = mean(ret), sd = sd(ret),
+                  mode = modes)
+     }
+   )
+
+   do.call(rbind, res)
+ }
R>
R> jac_df <- rbind(jac_frm(pred_fin, "Finite Sample"),
+                 jac_frm(pred_quant, "Quantiles"))
```

2. Consider two indices i, j of the vectors p_A, p_B such that $i \neq j$, corresponding to two distinct individuals. For such pairs of individuals (i, j) we can analyze the probability $P((p_A)_i \geq (p_B)_j)$, where we can consider i, j to be drawn randomly, and p_A, p_B resulting from two random bootstrap repetitions. This probability tells us how likely it is that two randomly selected individuals appear in the same order in two repetitions.

```
R> ord_ind <- function(x, modes = "single") {
+
+   res <- replicate(5000L, {
+     row <- sample(nrow(x), 2)
+     ord <- mean(x[row[1], ] > x[row[2], ])
+     max(ord, 1 - ord)
+   })
+
+   data.frame(res = res, mode = modes)
+ }
R>
R> ord_df <- rbind(ord_ind(pred_fin, "Finite Sample"),
+                 ord_ind(pred_quant, "Quantiles"))
```

3. Another interesting metric is the inversion number. Notice that p_A, p_B define two permutations of the n_{test} individuals, when we consider a ranking of individuals according to their predicted probabilities. We can compute the inversion number of these two permutations π_A, π_B, which is the total number of pairs of individuals whose ordering is not the same in π_A and π_B. Notice that the maximum value of the inversion number is $\binom{n_{test}}{2}$. Hence, we normalize this quantity accordingly.

```
R> inv_frm <- function(x, modes = "single") {
+
+   gt <- function(x) x[1L] > x[2L]
+
+   res <- replicate(100L, {
+     col <- sample(ncol(x), 2L)
+     prm <- order(x[, col[2L]][order(x[, col[1L]])])
+     sum(combn(prm, 2L, gt)) / choose(length(prm), 2L)
+   })
+
+   data.frame(res = res, mode = modes)
+ }
R>
R> inv_df <- rbind(inv_frm(pred_fin, "Finite Sample"),
+                  inv_frm(pred_quant, "Quantiles"))
```

4. For each individual i, we can take the 5% and 95% quantiles of predicted probabilities in all of the 'n.boot' bootstrap repetitions. We analyze the width of this interval across all individuals.

```
R> prb_frm <- function(x, modes = "single") {
+   qnt <- apply(x, 1L, quantile, probs = c(0.05, 0.95))
+   data.frame(width = qnt[2L, ] - qnt[1L, ], mode = modes)
+ }
R>
R> prb_df <- rbind(prb_frm(pred_fin, "Finite Sample"),
+                  prb_frm(pred_quant, "Quantiles"))
```

The results of these four metrics applied to different bootstrap repetitions on the COMPAS dataset are shown in Figure 7.

Individual Analysis

The other point of view we can take in quantifying uncertainty is that of the individual experiencing fair decisions. In this case, we are not so much interested in how much the overall decision set changes (as was the case above), but rather how much variation there is in the estimate for the specific individual. To this end, one might investigate the spread of the predicted values for a specific individual. The spread of the predictions for the first three individuals can be obtained as follows:

```
R> ind_prb <- data.frame(
+   prob = as.vector(t(pred_quant[seq_len(3), ])),
+   individual = rep(c(1, 2, 3), each = fa_boot_quant$n.boot)
+ )
```

The spread of predictions for the three individuals is visualized in Figure 8. While it might be tempting to take the mean of the individual predictions, to have more stable results,

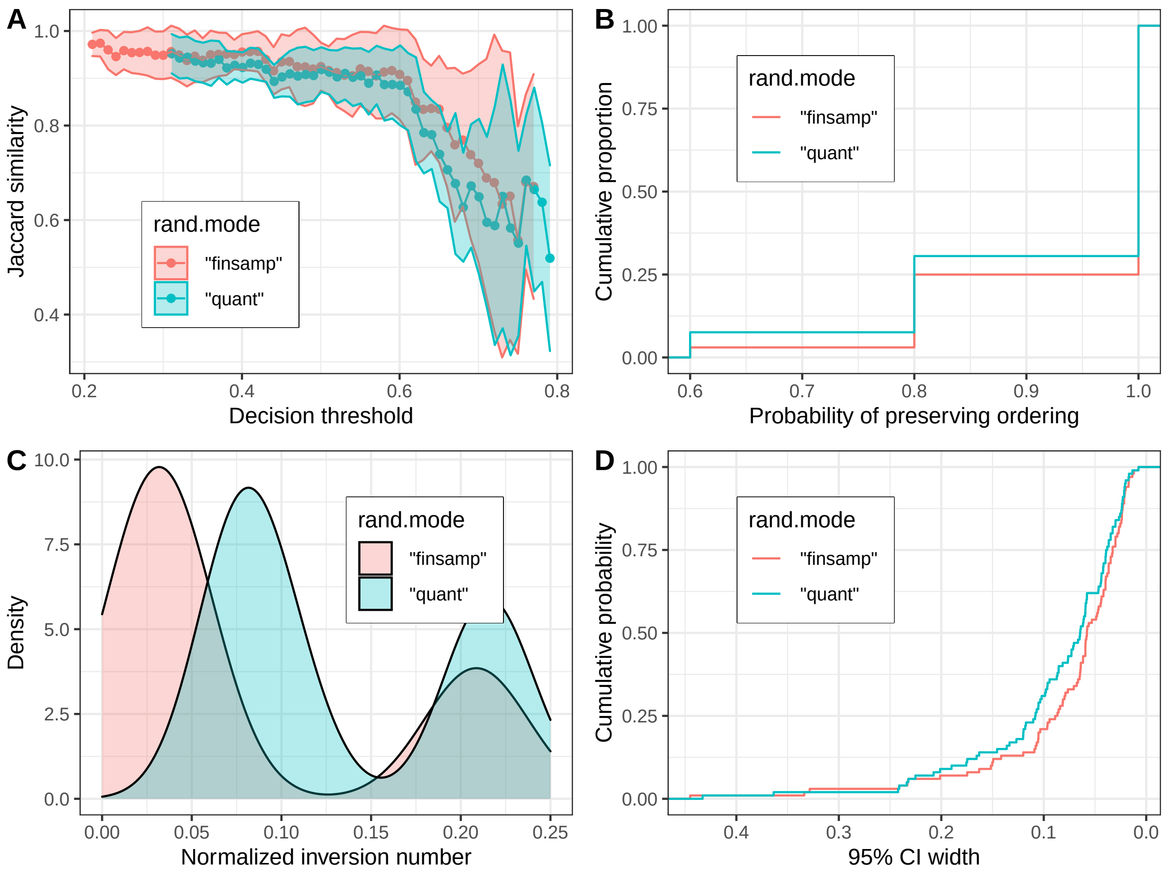


Figure 7: Analyzing uncertainty of predictions in the COMPAS dataset from decision-maker's point of view. Panel A shows how the Jaccard similarity of two repetitions varies depending on the decision threshold. Panel B shows the cumulative distribution of the random variable that indicates whether two randomly selected individuals preserve order (in terms of predicted probabilities) in bootstrap repetitions. Panel C shows the density of the normalized inversion number of between predicted probabilities in bootstrap repetitions. Panel D shows the cumulative distribution function of the 95% confidence interval (CI) width for the predicted probability of different individuals.

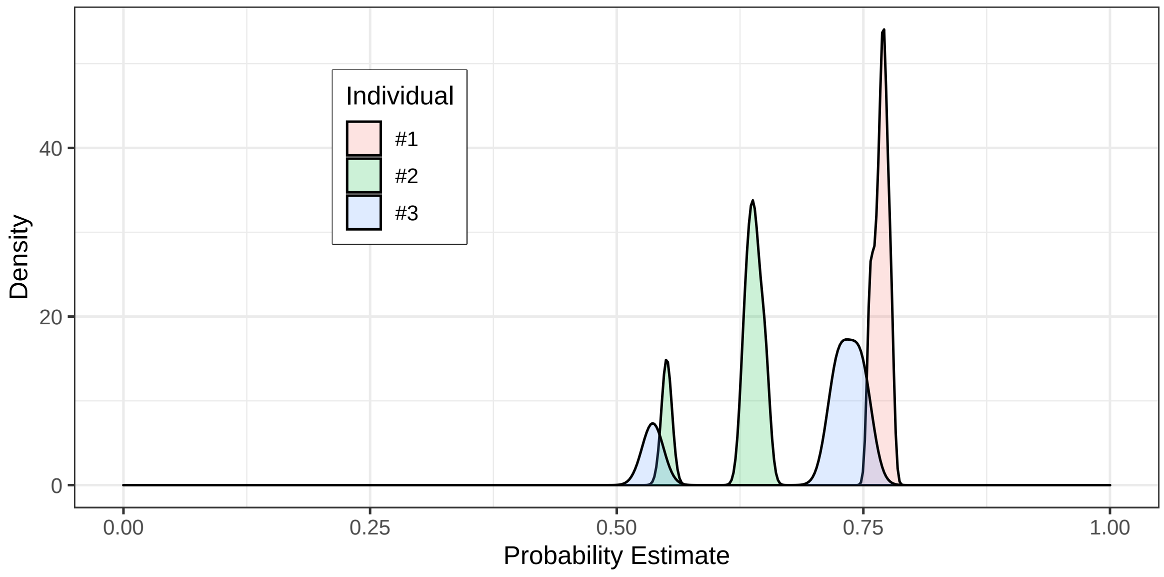


Figure 8: Analyzing the spread of individual predictions in the COMPAS dataset, resulting from different bootstrap repetitions.

this should not be done, as such an approach does not guarantee the fairness constraint `fairadapt()` aims to achieve. Hence, in order to achieve the desired fairness criteria overall, we have to accept some level of randomization at the level of individual predictions. It would be interesting for future work to quantify and optimize explicitly the trade-off between the desired fairness criteria and the necessary level of randomization.

5. Illustration

As a hypothetical real-world use of **fairadapt**, suppose that after a legislative change the US government has decided to adjust the salary of all of its female employees in order to remove both disparate treatment and disparate impact effects. To this end, the government wants to compute the counterfactual salary values of all female employees, that is the salaries that female employees would obtain, had they been male.

To do this, the government is using data from the 2018 American Community Survey by the US Census Bureau, available in pre-processed form as a package dataset from **fairadapt**. Columns are grouped into demographic (`dem`), familial (`fam`), educational (`edu`) and occupational (`occ`) categories and finally, salary is selected as response (`res`) and sex as the protected attribute (`prt`):

```
R> gov_dat <- data("gov_census", package = "fairadapt")
R> gov_dat <- get(gov_dat)
R>
R> head(gov_dat)
```

```
     sex age  race hispanic_origin citizenship nativity  marital
1   male  64 black              no           1   native  married
```

```
2 female  54 white              no           1   native  married
3   male  38 black              no           1   native  married
4 female  41 asian              no           1   native  married
5 female  40 white              no           1   native  married
6 female  46 white              no           1   native divorced
  family_size children education_level english_level salary
1           2        0              20             0  43000
2           3        1              20             0  45000
3           3        1              24             0  99000
4           3        1              24             0  63000
5           4        2              21             0  45200
6           3        1              18             0  28000
  hours_worked weeks_worked occupation industry economic_region
1           56           49    13-1081     928P       Southeast
2           42           49    29-2061     6231       Southeast
3           50           49    25-1000    611M1       Southeast
4           50           49    25-1000    611M1       Southeast
5           40           49    27-1010    611M1       Southeast
6           40           49    43-6014     6111       Southeast
```

```
R> dem <- c("age", "race", "hispanic_origin", "citizenship",
+           "nativity", "economic_region")
R> fam <- c("marital", "family_size", "children")
R> edu <- c("education_level", "english_level")
R> occ <- c("hours_worked", "weeks_worked", "occupation",
+           "industry")
R>
R> prt <- "sex"
R> res <- "salary"
```

The hypothesized causal graph for the dataset is given in Figure 9. According to this, the causal graph can be specified as an adjacency matrix `gov_adj` and as confounding matrix `gov_cfd`:

```
R> cols <- c(dem, fam, edu, occ, prt, res)
R>
R> gov_adj <- matrix(0, nrow = length(cols), ncol = length(cols),
+                    dimnames = rep(list(cols), 2))
R> gov_cfd <- gov_adj
R>
R> gov_adj[dem, c(fam, edu, occ, res)] <- 1
R> gov_adj[fam, c(     edu, occ, res)] <- 1
R> gov_adj[edu, c(          occ, res)] <- 1
R> gov_adj[occ,                 res ] <- 1
R>
R> gov_adj[prt, c(fam, edu, occ, res)] <- 1
R>
```

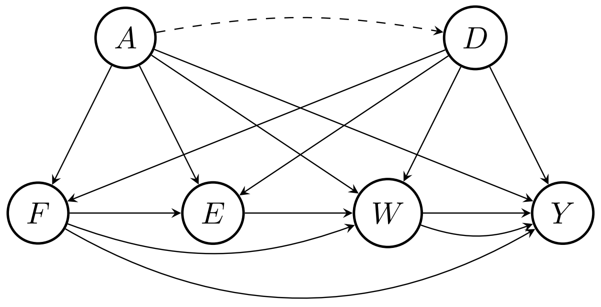


Figure 9: The causal graph for the government-census dataset. D are demographic features, A is gender, F represents marital and family information, E education, W work-related information and Y the salary, which is also the outcome of interest.

```
R> gov_cfd[prt, dem] <- 1
R> gov_cfd[dem, prt] <- 1
R>
R> gov_grph <- graphModel(gov_adj, gov_cfd)
```

Before applying `fairadapt()`, we first log-transform the salaries. We then inspect the densities of variable `salary` by sex group, as shown in Figure 11A. There is a clear shift between the two distributions, indicating that `male` employees are better compensated than `female` employees. We perform data the adaptation by using `n_samp` samples for training and `n_pred` samples for testing.

```
R> n_samp <- 3000
R> n_pred <- 5

R> gov_dat$salary <- log(gov_dat$salary)
R>
R> gov_trn <- head(gov_dat, n = n_samp)
R> gov_prd <- tail(gov_dat, n = n_pred)
R>
R> gov_ada <- fairadapt(salary ~ ., train.data = gov_trn,
+                        adj.mat = gov_adj, prot.attr = prt)
```

After adapting the data, we investigate whether the salary gap has shrunk. This can be done by comparing distributions of variable `salary` using the **ggplot2**-exported S3 generic function `autoplot()` (Figure 11B).

For adapting the additional testing data, we can use the base R S3 generic function `predict()`:

```
R> predict(gov_ada, newdata = gov_prd)
```

```
          sex age  race hispanic_origin citizenship nativity
204305 female  19 white              no           1   native
204306 female  46 white              no           1   native
```

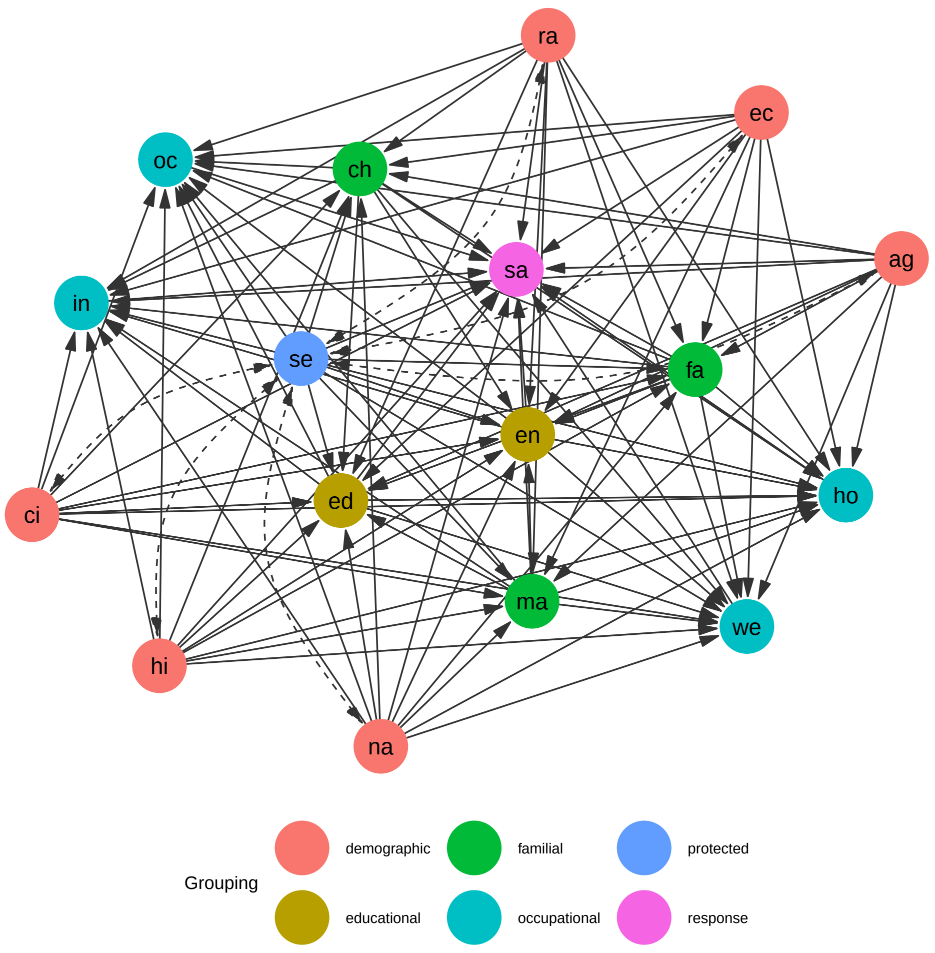


Figure 10: Full causal graph for the government census dataset, expanding the grouped view presented in Figure 9. *Demographic* features include age (**ag**), race (**ra**), whether an employee is of Hispanic origin (**hi**), is US citizen (**ci**), whether the citizenship is native (**na**), alongside the corresponding economic region (**ec**). *Familial* features are marital status (**ma**), family size (**fa**) and number of children (**ch**), *educational* features include education (**ed**) and English language levels (**en**), and *occupational* features, weekly working hours (**ho**), yearly working weeks (**we**), job (**oc**) and industry identifiers (**in**). Finally, the yearly salary (**sa**) is used as the *response* variable and employee sex (**se**) as the *protected* attribute variable.

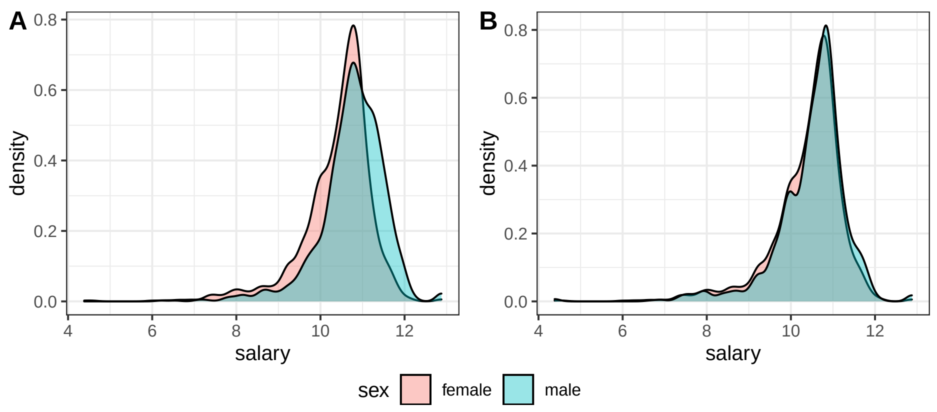


Figure 11: Visualization of salary densities grouped by employee sex, before (panel A) and after adaptation (panel B). Panel A indicates a shift towards higher values for male employees. In panel B, after the data is transformed, the gap between groups is reduced.

```
204307 female  24  AIAN              no          1   native
204308 female  23  AIAN              no          1   native
204309 female  50 white              no          1   native
             marital family_size children education_level
204305 never married           5        2              16
204306       married           2        1              19
204307 never married           3        2              20
204308 never married           3        2              19
204309       married           2        0              19
       english_level    salary hours_worked weeks_worked occupation
204305             0  7.003065           25           49    37-3011
204306             0  9.667765           40            0    53-3051
204307             0 10.126631           40           39    25-2020
204308             1  9.903488           40           26    43-4171
204309             0 11.472103           40           49    43-5031
       industry economic_region
204305       23  Rocky Mountain
204306    51912  Rocky Mountain
204307     6111  Rocky Mountain
204308   9211MP  Rocky Mountain
204309     92MP  Rocky Mountain
```

Finally, we can do fair-twin inspection using the `fairTwins()` function of **fairadapt**, to retrieve counterfactual feature values for different individuals:

```
R> fairTwins(gov_ada, train.id = 1:5,
+            cols = c("sex", "age", "education_level", "salary"))
```

```
     sex age age_adapted education_level education_level_adapted
1   male  64          64              20                      20
```

```
2 female  54          54                20                        20
3   male  38          38                24                        24
4 female  41          41                24                        24
5 female  40          40                21                        21
    salary salary_adapted
1 10.66896       10.34174
2 10.71442       10.71442
3 11.50288       11.60824
4 11.05089       11.05089
5 10.71885       10.71885
```

Note that values remain unchanged for female individuals (as *female* is used as baseline level). Variable `age`, which is not a descendant of the protected attribute `sex` (see Figure 10), also remains unchanged. However, variables `education_level` and `salary` do change for males, since these variables are descendants of the protected attribute `sex`.

The variable `hours_worked` is also a descendant of A, and one might argue that this variable should *not* be adapted in the procedure, i.e., it should remain the same, irrespective of employee sex. This is the idea behind *resolving variables*, which are discussed in Section 6.1. It is worth emphasizing that we are not trying to answer the question of which choice of resolving variables is the correct one in the above example - this choice is left to social scientists closely familiar with the context and specifics of the above described dataset.

6. Extensions

Several extensions to the basic Markovian SCM formulation introduced in Section 2.3 exist, some of which are available for use in `fairadapt()` and are outlined in the following sections.

6.1. Adding Resolving Variables

Kilbertus *et al.* (2017) discuss that in some situations the protected attribute A can affect variables in a non-discriminatory way. For instance, in the Berkeley admissions dataset (Bickel, Hammel, and O'Connell 1975) we observe that females often apply for departments with lower admission rates and consequently have a lower admission probability. However, we perhaps would not wish to account for this difference in the adaptation procedure, if we were to argue that applying to a certain department is a choice everybody is free to make. Such examples motivated the idea of *resolving variables* by Kilbertus *et al.* (2017). A variable R is called resolving if

(i) $R \in \text{de}(A)$, where $\text{de}(A)$ are the descendants of A in the causal graph $\mathcal{G}$.
(ii) The causal effect of A on R is considered to be non-discriminatory.

In presence of resolving variables, computation of the counterfactual is carried out under the more involved intervention $\text{do}(A = a, R = R(a'))$. The potential outcome value $V(A = a, R = R(a'))$ is obtained by setting $A = a$ and computing the counterfactual while keeping the values of resolving variables to those they *attained naturally*. This is a nested counterfactual and the difference in Algorithm 1 is simply that resolving variables R are skipped in the for-loop. In order to perform fair data adaptation with the variable `test` being resolving in the uni_admission dataset used in Section 3, the string "test" can be passed as res.vars to fairadapt().

```
R> fairadapt(score ~ ., train.data = uni_trn, test.data = uni_tst,
+            adj.mat = uni_adj, prot.attr = "gender", res.vars = "test")


Call:
fairadapt(formula = score ~ ., prot.attr = "gender", adj.mat = uni_adj,
    train.data = uni_trn, test.data = uni_tst, res.vars = "test")


Adapting variables:
  score, edu

Based on protected attribute gender

  AND

Based on causal graph:
       score gender edu test
score      0      0   0    0
gender     0      0   1    1
edu        1      0   0    1
test       1      0   0    0
```

As can be seen from the respective model summary outputs, the total variation after adaptation, in this case, is larger than in the basic example from Section 3, with no resolving variables. The intuitive reasoning here is that resolving variables allow for some discrimination, so we expect to see a larger total variation between the groups.

```
R> uni_res <- graphModel(uni_adj, res.vars = "test")
```

A visualization of the corresponding graph is available from Figure 12, which highlights the resolving variable test in red. Apart from color, the graphical model remains unchanged from what is shown in Figure 4.

6.2. Semi-Markovian and Topological Ordering Variant

Section 2 focuses on the Markovian case, which assumes that all exogenous variables U_i are mutually independent. However, in practice this requirement is often not satisfied. If a mutual dependency structure between variables U_i exists, this is called a Semi-Markovian model. In the university admission example, we could, for example, have $U_{\text{test}} \not\perp\!\!\!\perp U_{\text{score}}$, i.e., latent variables corresponding to variables test and final score being correlated. Such dependencies between latent variables can be represented by dashed, bidirected arrows in the causal diagram, as shown in Figures 13 and 14.

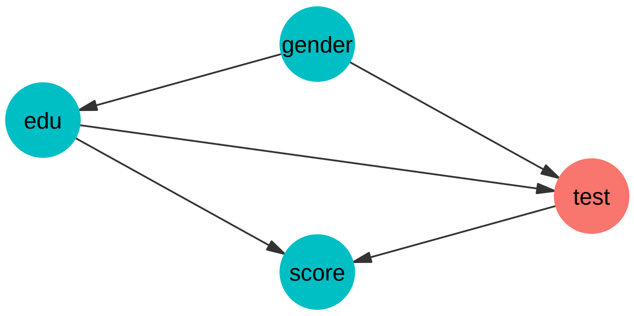


Figure 12: Visualization of the causal graph corresponding to the university admissions example introduced in Section 1 with the variable `test` chosen as a *resolving variable* and therefore highlighted in red.

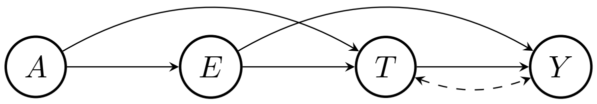


Figure 13: Causal graphical model corresponding to a Semi-Markovian variant of the university admissions example, introduced in Section 3. and visualized in its basic form in Figures 2 and 4. Here, we allow for the possibility of a mutual dependency between the latent variables corresponding to variables test and final score.

There is an important difference in the adaptation procedure for Semi-Markovian case: when inferring the latent quantiles U_i of variable V_i, in the Markovian case, only the direct parents $\text{pa}(V_i)$ are needed. In the Semi-Markovian case, due to correlation of latent variables, using only the $\text{pa}(V_i)$ can lead to biased estimates of the U_i. Instead, the set of direct parents needs to be extended, as described in more detail by Tian and Pearl (2002). A brief sketch of the argument goes as follows: Let the *C-components* be a partition of the set V, such that each *C-component* contains a set of variables which are mutually connected by bidirectional edges. Let $C(V_i)$ denote the entire *C-component* of variable V_i. We then define the set of extended parents as

$$\text{Pa}(V_i) := (C(V_i) \cup pa(C(V_i))) \cap \text{an}(V_i),$$

where $\text{an}(V_i)$ is the set of ancestors of V_i. The adaptation procedure in the Semi-Markovian case in principle remains the same as outlined in Algorithm 1, with the difference that the set of direct parents $\text{pa}(V_i)$ is replaced by $\text{Pa}(V_i)$ at each step.

To include the bidirectional confounding edges in the adaptation, we can pass a `matrix` as `cfd.mat` argument to `fairadapt()` such that:

- `cfd.mat` has the same dimension, column and row names as `adj.mat`.
- `cfd.mat` is symmetric.
- As is the case with the adjacency matrix passed as `adj.mat`, an entry `cfd.mat[i, j] == 1` indicates that there is a bidirectional edge between variables `i` and `j`.

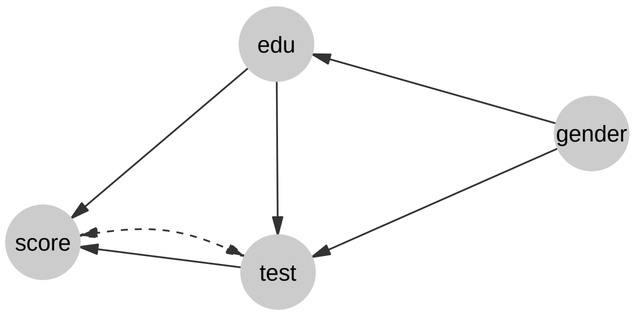


Figure 14: Visualization of the causal graphical model also shown in Figure 13, obtained when passing a confounding matrix indicating a bidirectional edge between vertices `test` and `score` to `fairadapt()`. The resulting Semi-Markovian setting is also handled by `fairadapt()`, extending the basic Markovian formulation introduced in Section 2.3.

The following code performs fair data adaptation of the Semi-Markovian university admission variant with a mutual dependency between the variables representing test and final scores. For this, we create a matrix `uni_cfd` with the same attributes as the adjacency matrix `uni_adj` and set the entries representing the bidirected edge between vertices `test` and `score` to 1. Finally, we can pass this confounding matrix as `cfd.mat` to `fairadapt()`. A visualization of the resulting causal graph is available from Figure 14.

```
R> uni_cfd <- matrix(0, nrow = nrow(uni_adj), ncol = ncol(uni_adj),
+                    dimnames = dimnames(uni_adj))
R>
R> uni_cfd["test", "score"] <- 1
R> uni_cfd["score", "test"] <- 1
R>
R> semi <- fairadapt(score ~ ., train.data = uni_trn,
+                    test.data = uni_tst, adj.mat = uni_adj,
+                    cfd.mat = uni_cfd, prot.attr = "gender")
```

Alternatively, instead of using the extended parent set $\mathrm{Pa}(V_i)$, we could also use the entire set of ancestors $\mathrm{an}(V_i)$. This approach is implemented as well, and available by specifying a topological ordering. This is achieved by passing a `character` vector, containing the correct ordering of the names appearing in `names(train.data)` as `top.ord` argument to `fairadapt()`. The benefit of using this option is that the specific edges of the causal model $\mathcal{G}$ need not be specified. However, in the linear case, specifying the edges of the graph, so that the quantiles are inferred using only the set of parents, will in principle have better performance.

6.3. Questions of Identifiability

So far we did not discuss whether it is always possible to carry out the counterfactual inference described in Section 2. In the causal literature, an intervention is termed *identifiable* if it can be computed uniquely using the data and the assumptions encoded in the graphical model

$\mathcal{G}$. An important result by Tian and Pearl (2002) states that an intervention $\text{do}(X = x)$ on a singleton variable X is identifiable if there is no bidirected path between X and $\text{ch}(X)$. Therefore, our intervention of interest is identifiable if one of the two following conditions are met:

- The model is Markovian.
- The model is Semi-Markovian and,
  - (i) there is no bidirected path between A and $\text{ch}(A)$ and,
  - (ii) there is no bidirected path between R_i and $\text{ch}(R_i)$ for any resolving variable R_i.

Based on this, the `fairadapt()` function may return an error, if the specified intervention is not possible to compute. An additional limitation is that **fairadapt** currently does not support *front-door identification* (Pearl 2009, Chapter 3), meaning that certain special cases which are in principle identifiable are not currently handled.

6.4. Future Avenues to be Explored

We conclude with a brief look at the possible extensions of the **fairadapt** package, which we hope to consider in future work:

1. *General identification*: as discussed above, there are certain cases of causal graphs in which our $\text{do}(A = a, R = R(a'))$ intervention is identifiable, but the `fairadapt()` function currently does not support doing so. One such example is front-door identification mentioned above. However, this is not the only such example. In a future version of **fairadapt**, we hope to cover all scenarios in which identification is possible.

2. *Path-specific effects*: when using resolving variables (Section 6.1), the user decides to label these variables as "non-discriminatory", that is, the algorithm is free to distinguish between groups based on these variables. In full generality, a user might be interested in considering all path-specific effects (Avin, Shpitser, and Pearl 2005). Such an approach would offer even more flexibility in modeling, since for every attribute-outcome path $A \to ... \to Y$, the user could decide whether it is fair or not.

3. *Selection bias*: a commonly considered problem in causal inference is that of selection bias (Hernán, Hernández-Díaz, and Robins 2004), when inclusion of individuals into the dataset depends on the observed variables in the dataset. In fairness applications, the presence of selection bias could invalidate our conclusions about discrimination and make our fair predictions biased. Recovering from selection bias algorithmically would therefore be a desirable feature in the **fairadapt** package.

4. *Non-binary attribute A*: one current limitation is that the protected attribute A is binary. When considering

References

Avin C, Shpitser I, Pearl J (2005). "Identifiability of path-specific effects."

Bellamy RKE, Dey K, Hind M, Hoffman SC, Houde S, Kannan K, Lohia P, Martino J, Mehta S, Mojsilovic A, Nagar S, Ramamurthy KN, Richards J, Saha D, Sattigeri P, Singh M, Varshney KR, Zhang Y (2018). “AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias.” URL `https://arxiv.org/abs/1810.01943`.

Bickel PJ, Hammel EA, O’Connell JW (1975). “Sex Bias in Graduate Admissions: Data From Berkeley.” *Science*, **187**(4175), 398–404.

Bird S, Dudík M, Edgar R, Horn B, Lutz R, Milan V, Sameki M, Wallach H, Walker K (2020). “Fairlearn: A toolkit for assessing and improving fairness in AI.” *Technical Report MSR-TR-2020-32*, Microsoft. URL `https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai/`.

Blau FD, Kahn LM (2003). “Understanding International Differences in the Gender Pay Gap.” *Journal of Labor Economics*, **21**(1), 106–144.

Cannon AJ (2015). ***qrnn**: Quantile Regression Neural Network*. R package version 2.0.5, URL `https://cran.r-project.org/web/packages/qrnn`.

Cannon AJ (2018). “Non-Crossing Nonlinear Regression Quantiles by Monotone Composite Quantile Regression Neural Network, With Application to Rainfall Extremes.” *Stochastic Environmental Research and Risk Assessment*, **32**(11), 3207–3225.

Chouldechova A (2017). “Fair Prediction With Disparate Impact: A Study of Bias in Recidivism Prediction Instruments.” *Big data*, **5**(2), 153–163.

Corbett-Davies S, Goel S (2018). “The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning.” *arXiv preprint arXiv:1808.00023*.

Csardi G, Nepusz T (2006). “The **igraph** Software Package for Complex Network Research.” *InterJournal*, **Complex Systems**, 1695. URL `https://igraph.org`.

Darlington RB (1971). “Another Look at Cultural Fairness.” *Journal of Educational Measurement*, **8**(2), 71–82.

Efron B, Tibshirani RJ (1994). *An introduction to the bootstrap*. CRC press.

Galles D, Pearl J (1998). “An Axiomatic Characterization of Causal Counterfactuals.” *Foundations of Science*, **3**(1), 151–182.

Hardt M, Price E, Srebro N, *et al.* (2016). “Equality of Opportunity in Supervised Learning.” In *Advances in neural information processing systems*, pp. 3315–3323.

Hernán MA, Hernández-Díaz S, Robins JM (2004). “A structural approach to selection bias.” *Epidemiology*, pp. 615–625.

Kilbertus N, Carulla MR, Parascandolo G, Hardt M, Janzing D, Schölkopf B (2017). “Avoiding Discrimination Through Causal Reasoning.” In *Advances in Neural Information Processing Systems*, pp. 656–666.

Koenker R, Hallock KF (2001). “Quantile Regression.” *Journal of Economic Perspectives*, **15**(4), 143–156.

Koenker R, Portnoy S, Ng PT, Zeileis A, Grosjean P, Ripley BD (2018). ***quantreg**: Quantile Regression.* R package version 5.86.

Komiyama J, Takeda A, Honda J, Shimao H (2018). "Nonconvex Optimization for Regression with Fairness Constraints." In *International Conference on Machine Learning*, pp. 2737–2746. PMLR.

Kozodoi N, V Varga T (2021). ***fairness**: Algorithmic Fairness Metrics.* R package version 1.2.2, URL `https://CRAN.R-project.org/package=fairness`.

Kusner MJ, Loftus J, Russell C, Silva R (2017). "Counterfactual Fairness." In *Advances in Neural Information Processing Systems*, pp. 4066–4076.

Lambrecht A, Tucker C (2019). "Algorithmic Bias? An Empirical Study of Apparent Gender-Based Discrimination in the Display of STEM Career Ads." *Management Science*, **65**(7), 2966–2981.

Larson J, Mattu S, Kirchner L, Angwin J (2016a). `https://github.com/propublica/compas-analysis`.

Larson J, Mattu S, Kirchner L, Angwin J (2016b). "How We Analyzed the COMPAS Recidivism Algorithm." *ProPublica (5 2016)*, **9**.

McGinley AC (2011). "Ricci v. DeStefano: Diluting Disparate Impact and Redefining Disparate Treatment." *Scholarly Works*, **646**.

Meinshausen N (2006). "Quantile Regression Forests." *Journal of Machine Learning Research*, **7**(Jun), 983–999.

Pearl J (2009). *Causality*. Cambridge University Press.

Plecko D, Meinshausen N (2020). "Fair Data Adaptation with Quantile Preservation." *Journal of Machine Learning Research*, **21**, 1–44.

Scutari M (2021). ***fairml**: Fair Models in Machine Learning.* R package version 0.6, URL `https://CRAN.R-project.org/package=fairml`.

Tian J, Pearl J (2002). "A General Identification Condition for Causal Effects." In *Eighteenth National Conference on Artificial Intelligence*, pp. 567–573. American Association for Artificial Intelligence, USA.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis.* Springer-Verlag New York. URL `https://ggplot2.tidyverse.org`.

Wiśniewski J, Biecek P (2021). "**fairmodels**: A Flexible Tool For Bias Detection." *arXiv preprint arXiv:2104.00507*.

Wright MN, Ziegler A (2015). "**ranger**: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *arXiv preprint arXiv:1508.04409*.

Zhang J, Bareinboim E (2018). "Fairness in Decision-Making: The Causal Explanation Formula." In *Thirty-Second National Conference on Artificial Intelligence*. American Association for Artificial Intelligence, USA.

Affiliation:

Drago Plečko
ETH Zürich
Seminar for Statistics Rämistrasse 101 CH-8092 Zurich
E-mail: drago.plecko@stat.math.ethz.ch

Nicolas Bennett
ETH Zürich
Seminar for Statistics Rämistrasse 101 CH-8092 Zurich
E-mail: nicolas.bennett@stat.math.ethz.ch

Nicolai Meinshausen
ETH Zürich
Seminar for Statistics Rämistrasse 101 CH-8092 Zurich
E-mail: meinshausen@stat.math.ethz.ch