fairadapt: Causal Reasoning for Fair Data Pre-processing

Drago Plecko
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 measure and mitigate such algorithmic bias. This manuscript describes the R-package **fairadapt**, which implements a causal inference pre-processing method. Using causal graphical models, this 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 counterfactual reasoning can help eliminate discrimination and help justify fair decisions.

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 2016) or gender (Blau and Kahn 2003; Lambrecht and Tucker 2019), and this realization seeded an important debate in the machine learning community about fairness of algorithms and their impact on decision-making.

In order to define and measure discrimination, existing intuitive notions have been statistically formalized, thereby providing fairness metrics. For example, *demographic parity* (Darlington 1971) requires the protected attribute A (gender/race/religion etc.) to be independent of a constructed classifier or regressor $\widehat{Y}$. Another notion, termed *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 $\widehat{Y} \perp\!\!\!\perp A \mid Y$. To this day, various different notions of fairness exist, which are sometimes incompatible (Corbett-Davies and Goel 2018), meaning not of all of them can be achieved for a predictor $\widehat{Y}$ simultaneously. There is no consensus on which notion of fairness is the correct one.

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.

Examples for this include 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 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 furthering understanding of demographic parity by decomposition of the parity gap measure, $\mathbb{P}(\widehat{Y} = 1 \mid A = a) - \mathbb{P}(\widehat{Y} = 1 \mid A = a')$, into the direct, indirect and spurious components, 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 R-package **fairadapt** from CRAN. Currently there are only few packages being distributed via CRAN, that are related fair machine learning, including **fairml** (Scutari 2021), which implements the non-convex method of Komiyama, Takeda, Honda, and Shimao (2018), as well as packages **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 only causal method, however, is presented by **fairadapt**. Even though theory in fair machine learning is being expanded at an accelerating pace, good quality implementations of the developed methods are lagging behind.

The rest of the manuscript is organized as follows: In Section 2 we describe the methodology behind **fairadapt**, together with quickly reviewing some important concepts of causal inference. In Section 3 we discuss implementation details and provide some general user guidance, followed by Section 4, which illustrates the usage of **fairadapt** through a large, real-world dataset and a hypothetical fairness application. Finally, in Section 5 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 relevant extensions, such as the Semi-Markovian case and the introduction of so called *resolving variables*, are discussed in Section 5.

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, where $A = a$ 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

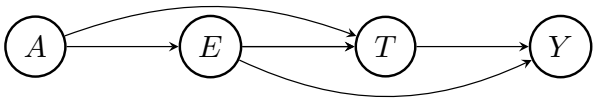

Figure 1: University admission based on previous educational achievement E combined with and an admissions test score T. The protected attribute A, encoding gender, has an unwanted causal effect on E, T, as well as Y, which represents the final score used for the admission decision.

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 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 60% of males in the dataset, we assume that he would also be better than 60% 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 2.

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

$$\mathbb{P}(Y \geq y \mid E = e, T = t, A = a') = \mathbb{P}(Y \geq y^{(fp)} \mid E = e^{(fp)}, T = t^{(fp)}, A = a). \tag{1}$$

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.

[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.

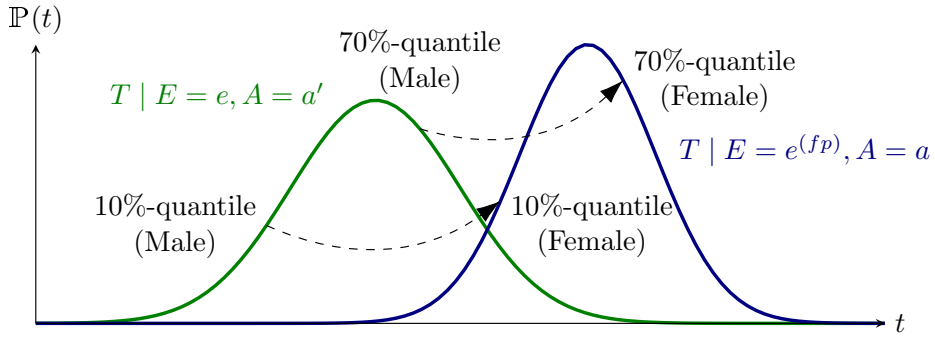

Figure 2: TODO: Add a figure caption.

2.2. Structural Causal Models

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

- $V = \{V_1, \ldots, V_n\}$ is the set of observed (endogeneous) variables.
- $U = \{U_1, \ldots, U_n\}$ are latent (exogeneous) 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.
- $P(u)$ is a distribution over the exogeneous 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) Exogeneous variables U_i are uniformly distributed on $[0, 1]$.

In the following section, we discuss the Markovian case in which all exogeneous 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 5.

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

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

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 $f_i(\mathrm{pa}(v_i), u_i)$ is increasing in u_i for every fixed $\mathrm{pa}(v_i)$, 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 $\mathrm{pa}(V_i)^{(fp)}$ (cf., Algorithm 1).

Algorithm 1: Fair Data Adaptation

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

The assignment functions f_i of the SCM are of course unknown, but are learned non-parametrically 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()`.

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.

As a quick demonstration of fair data adaption using, we load the `uni_admission` dataset provided by **fairadapt**, consisting of university admission data on 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` → `edu`, `gender` → `test`, `edu` → `test`, `edu` → `score`, and `test` → `score`. As 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,     0,      0,       0,
+                              1,     0,      0,       0,
+                              1,     1,      0,       0,
+                              0,     1,      1,       0),
+                    ncol = length(uni_dim),
+                    dimnames = rep(list(uni_dim), 2))
R>
R> basic <- fairadapt(score ~ ., train.data = uni_trn,
+                      test.data = uni_tst, adj.mat = uni_adj,
+                      prot.attr = "gender")
R>
R> basic

Fairadapt result

Call:
 score ~ .
```

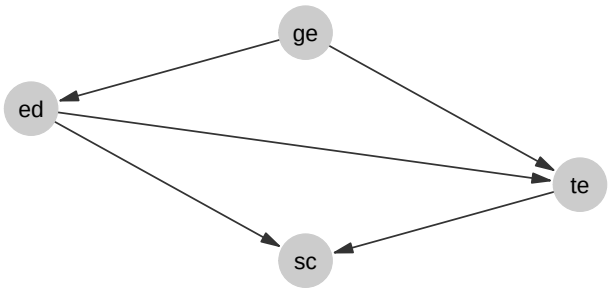

Figure 3: The underlying graphical model corresponding to the university admission example (also shown in Figure 1. Vertex labels are abbreviated using the first two letters of the respective variable names. The vertex *ge* therefore represents `gender`, etc.

```
Protected attribute:                  gender
Protected attribute levels:           0, 1
Number of training samples:           200
Number of test samples:               200
Number of independent variables:      3
Total variation (before adaptation):  -0.6757414
Total variation (after adaptation):   -0.08297119
```

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

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

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 $\mathcal{G}$ (cf. Algorithm 1), a corresponding adjacency matrix can be passed as `adj.mat` argument. The convenience function `graphModel()` turns a graph specified as 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 3. The graph shown is equivalent to that of Figure 1 as they both represent the same causal

model. For plotting purposes, we use the fist two letters to denote vertex labels (i.e., vertex *ge* represents candidate gender (the `gender` column of `uni_dat`) and is equivalent to vertex A in Figure 1, representing the protected attribute.

3.2. Quantile Learning Step

The training step in `fairadapt()` can be carried out in two slightly distinct ways: Either by specifying training and testing data simultaneously, or by only passing training data (and at a later stage calling `predict()` on the returned `fairadpat` object in order to perform data adaption on new test data).

The two dataframes 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 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).
- Non-crossing quantile neural networks (Cannon 2018).
- Linear Quantile Regression (Koenker and Hallock 2001).

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 might in fact be a superior choice. A quick summary outlining some differences between the three natively supported methods is available from Table 1.

The above set of methods, is not exhaustive. Further options are conceivable and therefore **fairadpat** 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.
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, for which an implementation of the S3 generic function `computeQuants()` is available.

	Random Forests	Neural Networks	Linear Regression
R-package	**ranger**	**mcqrnn**	**quantreg**
`quant.method`	`rangerQuants`	`mcqrnnQuants`	`linearQuants`
complexity	$O(np \log n)$	$O(npn_{\text{epochs}})$	$O(p^2 n)$
default parameters	$ntrees = 500$ $mtry = \sqrt{p}$	2-layer fully connected feed-forward network	`"br"` method of Barrodale and Roberts used for fitting
$T(200, 4)$	0.4 sec	89 sec	0.3 sec
$T(500, 4)$	0.9 sec	202 sec	0.5 sec

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. $T(n, 4)$ denotes the runtime of different methods on the university admission dataset, with n training and testing samples.

3.3. Fair-Twin Inspection

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 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    1.14489206  1.3499572   0.5815851  1.6177397
2      0 -2.3502495   -2.35024955 -1.9779234  -1.9779234 -3.1217962
3      1  0.6285619   -0.04234933  0.6263626  -0.2602679  0.5300347
  test_adapted
1    0.6253059
2   -3.1217962
3   -0.3324879
```

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.

4. Illustration

As a possible 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 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 4. According to this, the

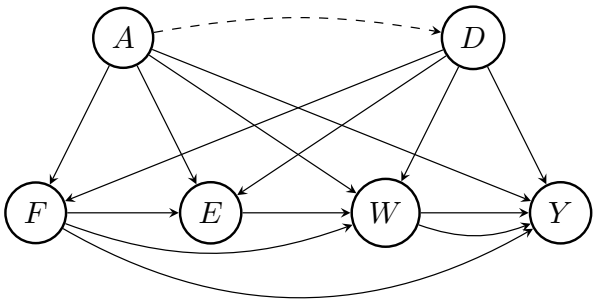

Figure 4: The causal graph for the government-census dataset. D are demographic features, A is gender, F represents marital and family information, E education, W work-related information and Y the salary, which is also the outcome of interest.

causal graph can be specified as an adjacency matrix `gov_adj` and as confounding matrix `gov_adj`:

```
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>
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 and look at respective densities by sex group. We subset the data by using the first `n_samp` samples for training, the subsequent `n_pred` samples for predicting and plot the data before performing the adaption.

```
R> gov_dat$salary <- log(gov_dat$salary)
R>
R> n_samp <- 2000
R> n_pred <- 5
R>
R> gov_trn <- head(gov_dat, n = n_samp)
R> gov_prd <- tail(gov_dat, n = n_pred)
```

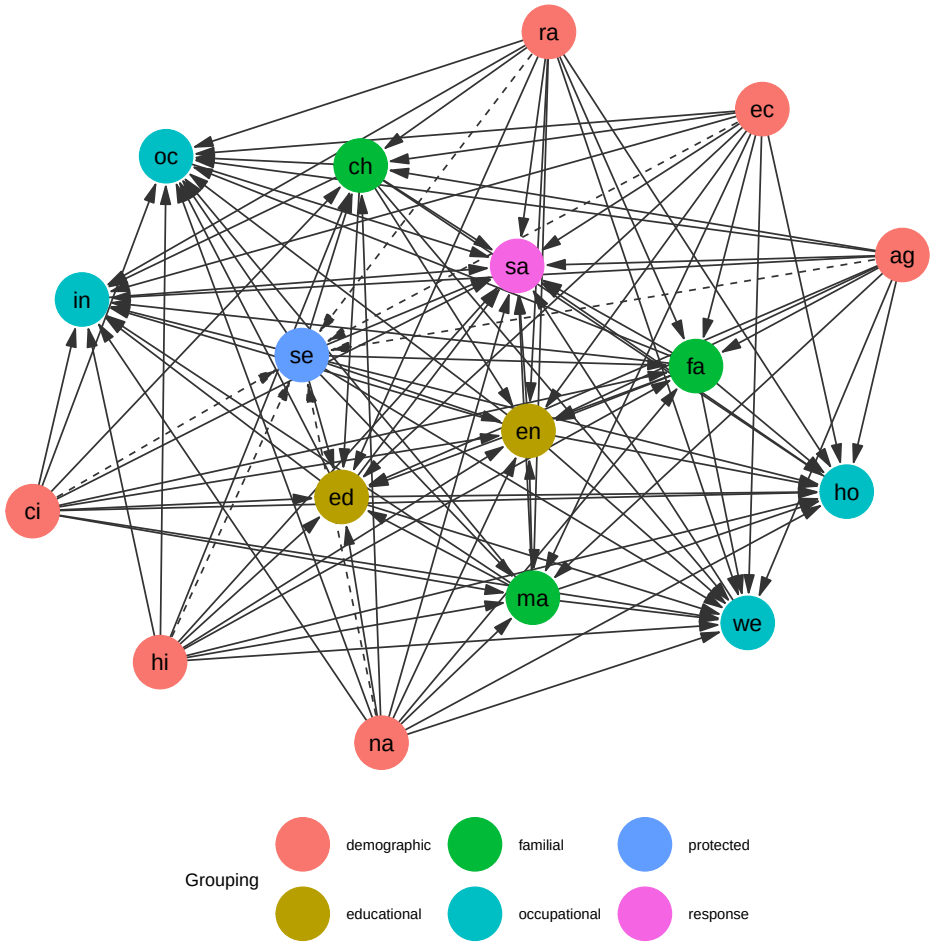

Figure 5: Full causal graph for the government-census dataset, expanding the grouped view presented in Figure 4. *Demographic* features include age (**ag**), race (**ra**), whether an employee is of Hispanic origin (**hi**), is US citizen (**ci**), whether the citizenship is native (**na**), alongside 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 *response* variable and employee sex (**se**) as *protected* attribute.

Figure 6: Visualization of salary densities grouped by employee sex, indicating a shift to higher values for male employees. This uses the US government-census dataset and shows the situation before applying fair data adaption, while Figure 7 presents transformed salary data.

There is a clear shift between the two sexes, indicating that `male` employees are currently better compensated when compared to `female` employees. However, this differences in `salary` could, in principle, be attributed to factors apart form gender inequality, such as the economic region in which an employee works. This needs to be accounted for as well, i.e. we do not wish to remove differences in salary between economic regions.

```
R> gov_ada <- fairadapt(salary ~ ., train.data = gov_trn,
+                       adj.mat = gov_adj, prot.attr = prt)
```

After performing the adaptation, we can investigate whether the salary-gap has shrunk. The densities after adaptation can be visualized using the **ggplot2**-exported S3 generic function `autoplot()`:

```
R> autoplot(gov_ada, when = "after") +
+   theme_bw() +
+   ggtitle("Adapted salary density by gender")
```

If we are provided with additional testing data, and wish to adapt this as well, we can use the base R S3 generic function `predict()`:

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

```
      sex age  race hispanic_origin citizenship nativity
1: female  19 white              no           1   native
```

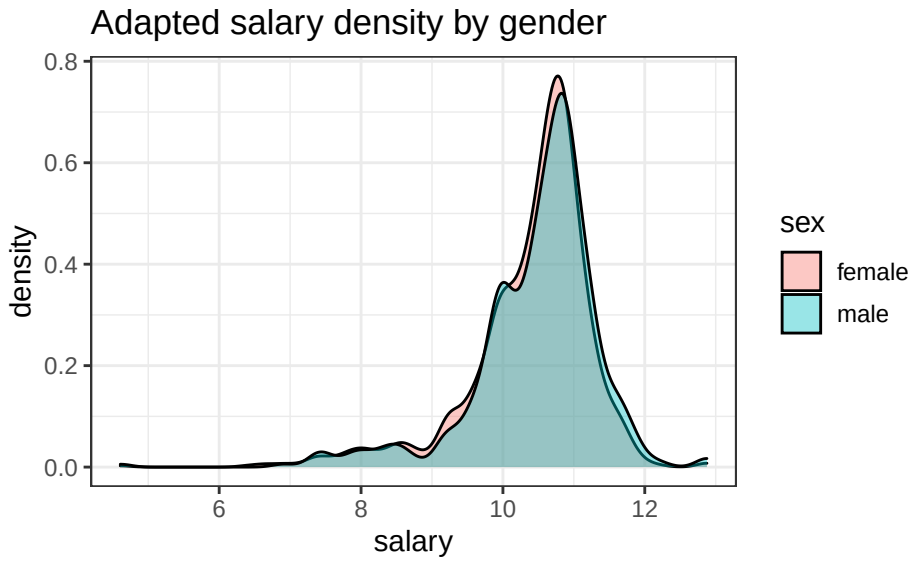

Figure 7: The salray-gap between male and female employees of the US government according to the government-census dataset is clearly reduced when comparing raw data (cf., Figure 6 to transformed salary data as yielded by applying fair data adaption using employee sex as protected attribute and assuming a causal graph as shown in Figure 5.

```
2: female  46 white              no           1   native
3: female  24  AIAN              no           1   native
4: female  23  AIAN              no           1   native
5: female  50 white              no           1   native
         marital family_size children education_level english_level
1: never married           5        2              16             0
2:       married           2        0              18             0
3: never married           4        2              19             0
4: never married           3        2              19             1
5:       married           2        0              19             0
      salary hours_worked weeks_worked occupation industry
1:  7.003065           25           49    37-3011       23
2:  9.667765           40            0    43-6014     52M1
3: 10.126631           40           49    33-3012     623M
4:  9.903488           40           26    43-4171   9211MP
5: 11.472103           40           49    43-5031     92MP
   economic_region
1:  Rocky Mountain
2:  Rocky Mountain
3:  Rocky Mountain
4:  Rocky Mountain
5:  Rocky Mountain
```

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

```
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       9.998798
2 10.71442      10.714418
3 11.50288      11.608236
4 11.05089      11.050890
5 10.71885      10.718852
```

Values are unchanged for female individuals (as *female* is used as baseline level), as is the case for `age`, which is not a descendant of the protected attribute A (`sex`, cf., Figure 5). However, variables `education_level` and `salary` do change for males, as they are descendants of A.

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., that it remain the same, irrespective of employee sex. This is the idea behind *resolving variables*, discussed in the following section (5.1).

5. 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.

5.1. 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 department choice is a choice everybody is free to make. This motivated the 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

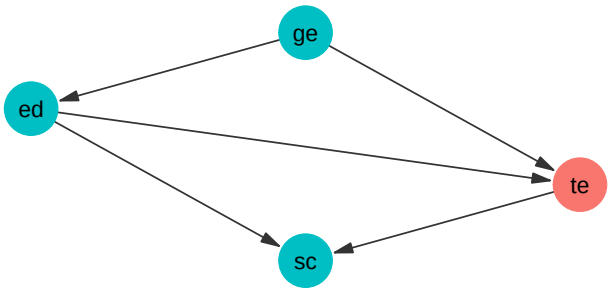

Figure 8: Visualization of the causal graph corresponding to the university admissions example introduced in Section 1 with the variable `test` (**te**) chosen as *resolving variable* and therefore highlighted in red.

for-loop. In order to perform fair data adaptation with the variable `test` being resolving in the `uni_admission` dataset used in 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")

Fairadapt result

Call:
 score ~ .

Protected attribute:                  gender
Protected attribute levels:           0, 1
Resolving variables:                  test
Number of training samples:           200
Number of test samples:               200
Number of independent variables:      3
Total variation (before adaptation):  -0.6757414
Total variation (after adaptation):   -0.4160513
```

As can be seen from the respective model summary outputs, the total variation in this case is larger than in the `basic` example from Section 3, with no resolvers. 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 8, which highlights the resolving variable `test` in red. Apart from color, the graphical model remains unchanged from what is shown in Figure 3.

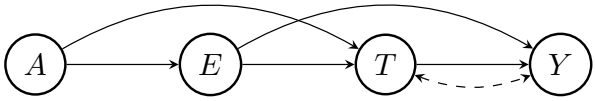

Figure 9: Causal graphical model corresponding to a Semi-Markovian variant of the university admissions example, introduced in efimplementation and visualized in its basic form in Figures 1 or 3. Here, we posit a mutual dependency between the variables representing test and final scores.

5.2. Semi-Markovian and Topological Ordering Variant

Section 2 focuses on the Markovian case, which assumes that all exogeneous 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 variable test and final score be correlated. Such dependencies between latent variables can be represented by dashed, bidirected arrows in the causal diagram, as shown in Figures 9 and 10.

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 `fairadpat()` 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`.

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 entries representing edged 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 10.

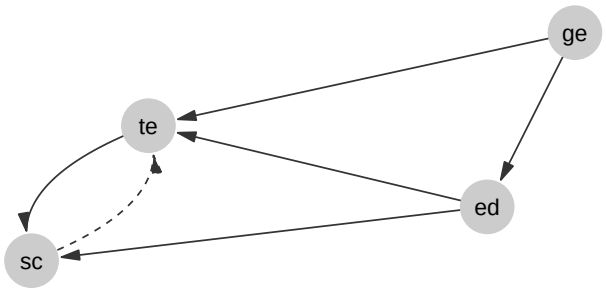

Figure 10: Visualization of the causal graphical model also shown in Figure 9, obtained when passing a confounding matrix indicating a bidirectional edge between vertices `test` and `score` to `fairadapt()`. The resulting Semi-Markovian setting can be handles by `fairadapt()`, extending the basic Markovian formulation introduced in Section 2.3.

```
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 *largest possible* set of parents, i.e., the 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()`.

5.3. Questions of Identifiability

So far, the question 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 $\mathrm{do}(X = x)$ on a singleton variable X is identifiable if and only if there is no bidirected path between X and $\mathrm{ch}(X)$. Therefore, the intervention is identifiable if any of the following conditions hold:

- The model is Markovian.
- The model is Semi-Markovian and there is no bidirected path between A and $\mathrm{ch}(A)$.
- The model is Semi-Markovian and there is no bidirected path between R_i and $\mathrm{ch}(R_i)$ 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), but we hope to include this in a future version.

References

Bickel PJ, Hammel EA, O'Connell JW (1975). "Sex Bias in Graduate Admissions: Data from Berkeley." *Science*, **187**(4175), 398–404.

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

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.

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.

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.

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.

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 (2016). "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*. URL https://arxiv.org/abs/2104.00507.

Affiliation:

Drago Plecko
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