Counterfactual Fairness Replication

By Potluri, Kai, Mingfei, and Ollie

There are many machine learning algorithms that can and do automate decisions for complex issues such as receiving a loan, insurance pricing, predictive policing, prison sentences, etc. This paper, by Kusner, Loftus, Russell, and Silva, seeks to develop a way to address whether a prediction is fair using tools from causal inference. Through causal modeling we can see how and where this unfairness is happening. Because algorithms use observed data to make predictions, data that may contain historical bias, (ex. racially biased police stop and frisk policies sexist/racist/ageist hiring policies) they can often result in unfair policy decisions.

To understand how to evaluate fairness we must define what it means. Below are a definitions presented by the authors:

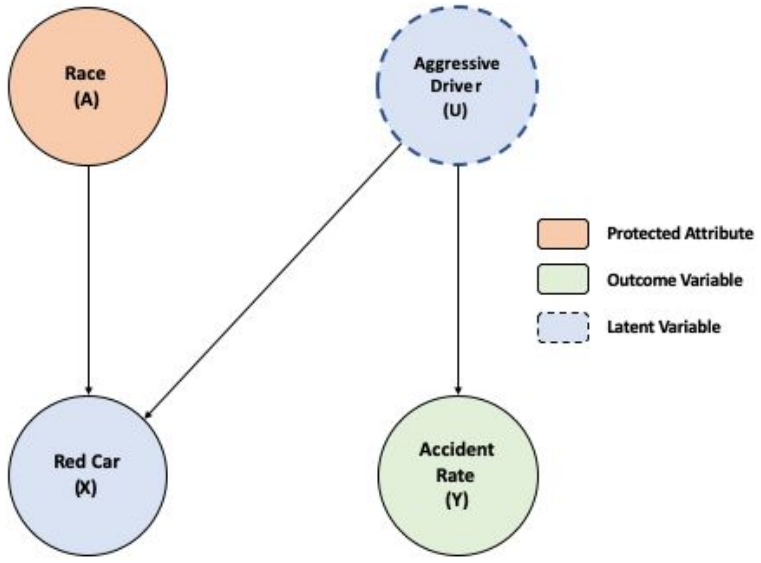
- **Fairness through unawareness** – an algorithm does not use any protected attributes. If we have observations of a person's race or sex we just ignore that information
- **Individual Fairness** – an algorithm gives similar predictions to similar individuals
- **Demographic Parity** – an algorithm gives the same prediction no matter what the value of the given predictor is
- **Equality of Opportunity** – an algorithm gives the same prediction given the outcome regardless of the protected attribute
- **Counterfactual Fairness** – Had any individual been of a different race, sex, etc. the prediction would not change

The authors believe that counterfactual fairness is the correct definition to use because we can make comparisons on an individual level. We are asking what would have happened if this same person was of a different race or sex.

Below is a simple example from the paper to show how counterfactual awareness works. We are looking to predict the accident rate of an individual given we have observations on some protected attribute and the color of their car. See below for the complete model.

- **A Protected Attribute** - any attribute that should not be discriminated
- **X Observable attribute** – Red Car
- **Y outcome of interest** – Accident Rate

- **U latent variable** – Aggressive Driving



In this example, some group A is more likely than other groups to drive a red car but are not more likely to get into an accident. However, people who are more likely to be aggressive drivers like to drive red cars as well. We are explicitly modeling a discriminatory effect as a causal effect. If we were to predict who would have a high accident rate Y by using just X we would have a counterfactually unfair prediction because we know that individuals of the protected attribute like to drive red cars more than other groups even though they don't get into more accidents.

One of the main implications of counterfactual fairness is that a prediction will be counterfactually fair if it is a function of the non-descendants of A. In the example above we could not use the observable attribute red car to make a counterfactually fair prediction as any change in A would cause a change in the observed variable, red car. Instead we would want to try to infer the latent variable U.

This is not an if and only if statement however, it is possible for a prediction to be counterfactually fair if it is a function of a descendent of the protected attribute A. However, this is only true though if the dependence of the prediction on the protected attribute disappears in the function.

Another benefit of using causal inference to evaluate fairness is the ability to deal with historical bias. An example of this given in the paper is of whether a person defaults on a loan. A protected attribute may cause a person to default on a loan, but only through some past

discrimination. The protected attribute for example may be mediated by employment which in turn is caused by the latent variable, prejudice. Prejudice could mean that the hiring process was unfair towards certain groups. This causes a person to be less likely to be employed and in turn increases the likelihood of default. Hence Y, a descendent of A, has hsitorical discrimination and should not be used in the prediction of .

We want to use variables that are not caused by A, the protected attribute, but are predictive of Y. Below is the algorithm given in the text.

Fair Learning Algorithm

procedure FAIRLEARNING($\mathcal{D}, \mathcal{M}$) ▷ Learned parameters $\hat{\theta}$
For each data point $i \in \mathcal{D}$, sample m MCMC samples $U_1^{(i)}, \ldots, U_m^{(i)} \sim P_{\mathcal{M}}(U \mid x^{(i)}, a^{(i)})$.
Let $\mathcal{D}'$ be the augmented dataset where each point $(a^{(i)}, x^{(i)}, y^{(i)})$ in $\mathcal{D}$ is replaced with the corresponding m points $\{(a^{(i)}, x^{(i)}, y^{(i)}, u_j^{(i)})\}$.
$\hat{\theta} \leftarrow \text{argmin}_\theta \sum_{i' \in \mathcal{D}'} l(y^{(i')}, g_\theta(U^{(i')}, x_{\not\in A}^{(i')}))$.
end procedure

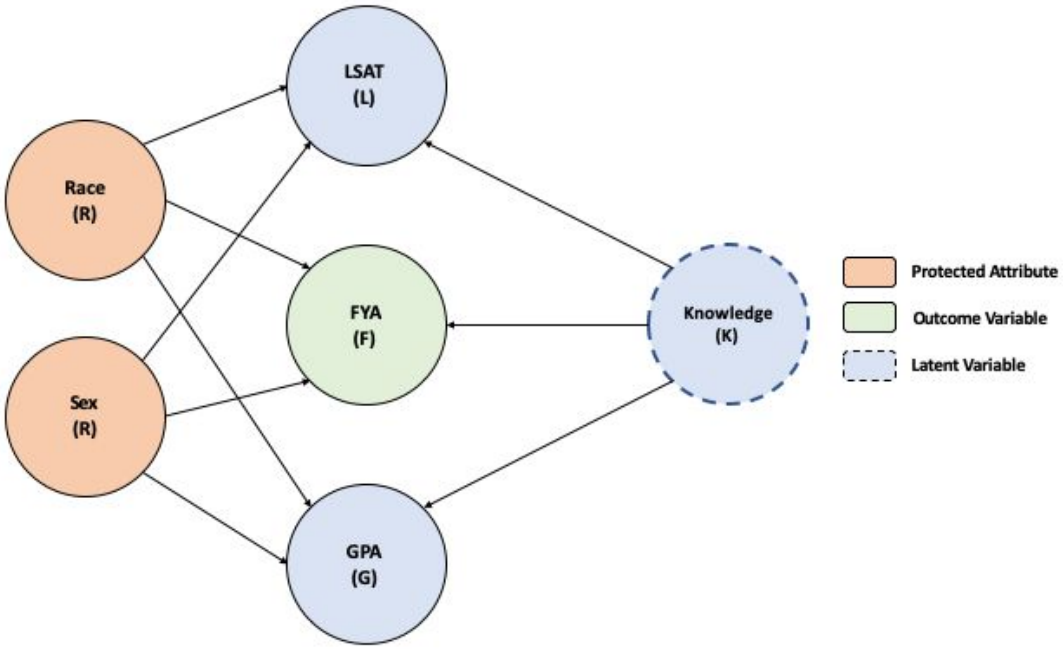
Essentially what the algorithm is saying is, given the observed values of A and X, extract the values of the latent variable U and then use those values in the predictive model.

There are three different levels of assumptions that make counterfactual fairness possible. The higher levels (3 being the highest) require stronger assumptions

- Level 1 – Determine from observable non-descendents of A
- Level 2 – Determine through some non-deterministic latent variable
- Level 3 – Determine through some deterministic latent variables (not discussed in great detail)

The authors recommend always modeling the causal structure to reach the goal of counterfactual fairness. They state, “we are essentially learning a projection of T into the space of fair decisions, removing historical biases as a by-product ”. (p. 5) Modeling in this way does not always give the most accurate predictions, however this is not the goal of the procedure. The goal is to accurately model the real world including social bias's that may arise and to use causal modeling tools to address algorithmic unfairness. The following is another example of this in practice.

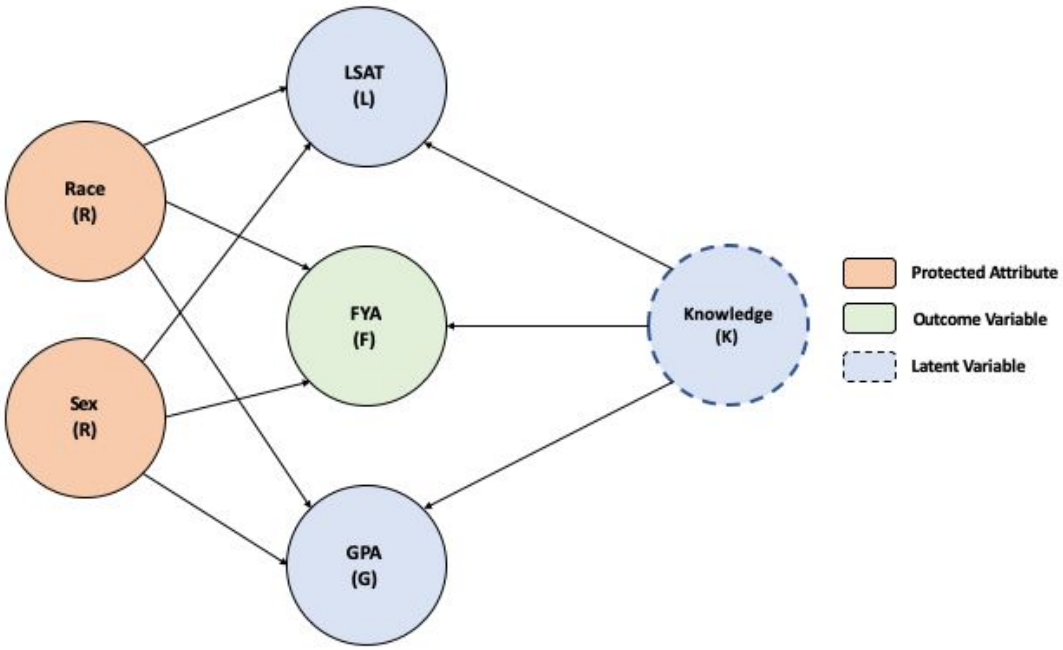
Law School Success



We have considered the example of law school success to implement the algorithms discussed in the paper. In this example, the law school wants candidates with good knowledge to succeed in the law program. However, knowledge is an unobserved variable. The challenge was to make sure the admissions process was fair to underprivileged groups. The college admissions panel doesn't use candidates' racial or gender identity in determining whether to admit them into the program or not. They'd only consider the undergraduate GPA and LSAT score to keep the process fair. However, the admissions panel's decision might get indirectly influenced by protected attributes like race and sex. Students from less privileged backgrounds may lack the economic opportunity to enroll in LSAT preparation courses. The classroom performance may not depend on knowledge alone but also be unduly affected by sensitive attributes that we do not want to use as a basis for admissions. In this project, we implemented concepts like fairness through unawareness and counterfactual fairness as discussed in the paper. We leveraged these concepts to identify historical bias while estimating knowledge. We built a structural causal model using the above DAG to generate samples. These samples are used in the Unaware and Full Model.

The Model

We have constructed the SCM in Pyro using the above DAG. The probability assignments for the simulation are made up as there was no simulation specific information in the paper. Race, sex, and knowledge are considered exogenous variables. A probability of 0.7 and 0.35 are assigned for Race and Sex respectively. The knowledge is drawn from a standard normal distribution. Below you can find the equations that we have used to generate the synthetic data:



$$N_r \sim \text{Bernoulli}(0.7)$$

$$N_s \sim \text{Bernoulli}(0.35)$$

$$N_k \sim \text{Normal}(0,1)$$

$$R \sim N_r$$

$$S \sim N_s$$

$$K \sim N_k$$

$$G \sim \text{Normal}(K + 2.1 * R + 3.3 * S, 0.5)$$

$$L \sim \text{Normal}(K + 5.8 * R + 0.7*S, 0.1)$$

$$F \sim \text{Normal}(K + 2.3*R + 1 *S, 0.3)$$

We have generated 1000 samples from this SCM to use in the implemented models.

Unaware Model

An algorithm is fair so long as any protected attributes, in our example, Race and Sex, are not explicitly used in the decision-making process. However, it has a clear shortcoming as elements of X can contain discriminatory information analogous to protected attributes that may not be obvious at first. To replicate this model, we have considered only the attributes GPA and LSAT. This is the same model used initially by the law school assuming a fair admission policy. The First year average(F) is predicted using the GPA(G) and LSAT(L). From the DAG, we can understand that it is indirectly biased because of the Race(R) and Sex(S) and hence also affecting the First year average(F).

```
unaware_sample= []
for i in range(1000):
    trace = trace_handler.get_trace()
    R = trace.nodes['R']['value']
    S = trace.nodes['S']['value']
    A = trace.nodes['A']['value']
    G = trace.nodes['G']['value']
    L = trace.nodes['L']['value']
    F = trace.nodes['F']['value']
    # get prob of each combination
```

```
    log_prob = trace.log_prob_sum()
    p = np.exp(log_prob)
    samples = samples.append({'R': R, 'S': S, 'A': A, 'G': G, 'L':L,
'F': F, 'p': p}, ignore_index=True)
    unaware_sample.append(([G,L,F]))

samples.head()
```

We have then set up the Linear Regression in PyroModule to predict the first year average(F) from GPA and LSAT.

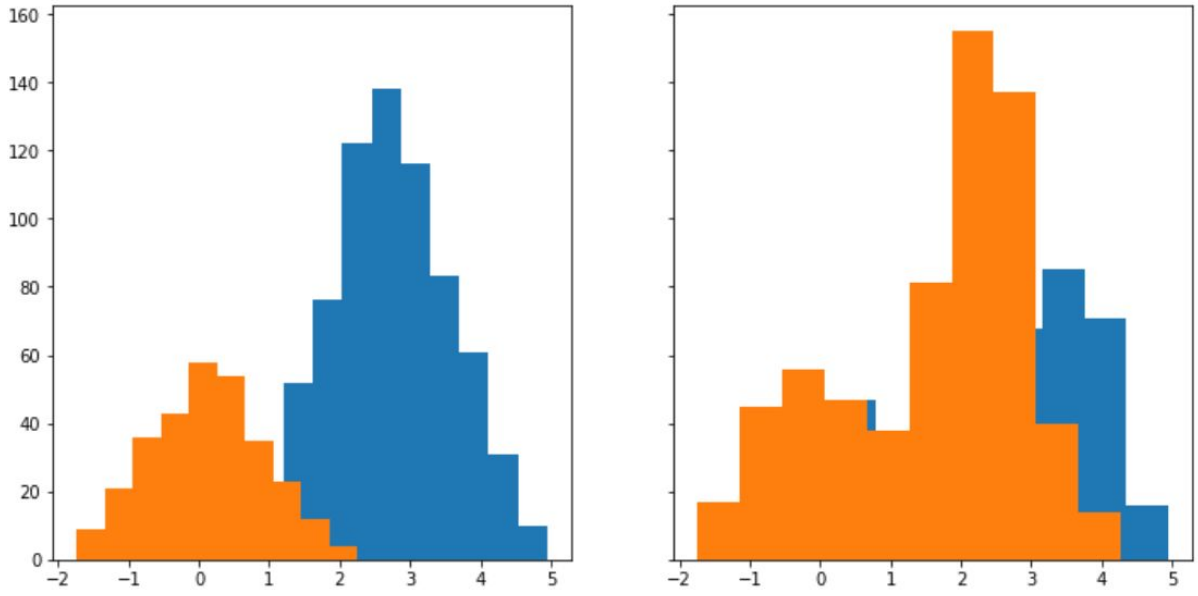
```
#Data to regress
unaware_sample = torch.tensor(unaware_sample)
x_data, y_data = unaware_sample[:, :-1], unaware_sample[:, -1]
```

We have created the Linear regression model and selected the MSE as the loss function

```
# Regression model
# 2 = in features, 1=out feature
linear_reg_model = PyroModule[nn.Linear](2, 1)

# Define loss and optimize
loss_fn = torch.nn.MSELoss(reduction='sum')
optim = torch.optim.Adam(linear_reg_model.parameters(), lr=0.05)
num_iterations = 500 if not smoke_test else 2

def train():
    # run the model forward on the data
    y_pred = linear_reg_model(x_data).squeeze(-1)
    # calculate the mse loss
    loss = loss_fn(y_pred, y_data)
    # initialize gradients to zero
    optim.zero_grad()
    # backpropagate
```

```
    loss.backward()
    # take a gradient step
    optim.step()
    return loss
```

We have plotted the density plots from the trained samples.



Full Model

This model uses all the available observed variables in the model. We can later use this model to more accurately reconstruct FYA(F) as it uses race and sex. The paper notes that because it uses all the observed variables, this model achieves the highest accuracy, however it clearly is biased since it uses protected attributes race and sex directly in its decision making. The resulting density plots look similar to the unaware model as the full model is still making biased decisions, but much more openly in its training.

Considering all the samples:

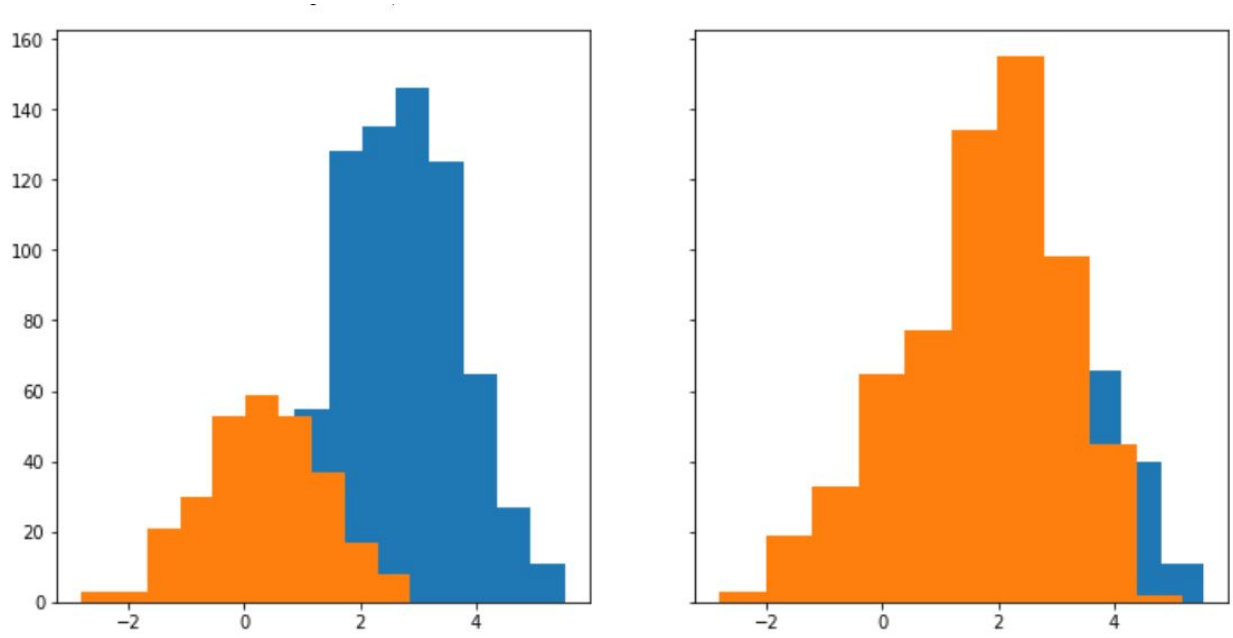
```
samples = pd.DataFrame(columns=['R', 'S', 'A', 'G', 'L', 'F', 'p'])
full_sample= []
for i in range(1000):
    trace = trace_handler.get_trace(exo_dist)
    R = trace.nodes['R']['value']
    S = trace.nodes['S']['value']
    A = trace.nodes['A']['value']
    G = trace.nodes['G']['value']
    L = trace.nodes['L']['value']
    F = trace.nodes['F']['value']
    # get prob of each combination
    log_prob = trace.log_prob_sum()
    p = np.exp(log_prob)
    samples=samples.append({'R': R, 'S': S, 'A': A, 'G': G, 'L':L, 'F':
F, 'p': p}, ignore_index=True)
    full_sample.append(([R,S,G,L,F]))

samples.head()
```

We have similarly built the Linear Regression model to predict FYA(F) using race, sex, GPA and, LSAT.

```
# 4 = in features, 1=out feature
linear_reg_model = PyroModule[nn.Linear](4, 1)

# Define loss and optimize
loss_fn = torch.nn.MSELoss(reduction='sum')
optim = torch.optim.Adam(linear_reg_model.parameters(), lr=0.05)
num_iterations = 500 if not smoke_test else 2

def train():
    # run the model forward on the data
    y_pred = linear_reg_model(x_data).squeeze(-1)
    # calculate the mse loss
    loss = loss_fn(y_pred, y_data)
    # initialize gradients to zero
```

```
    optim.zero_grad()
    # backpropagate
    loss.backward()
    # take a gradient step
    optim.step()
    return loss
```

We have constructed the density plots:



Inferring K

In this model, we infer K(as it is unobserved) and use that to predict FYA(F) instead of relying on the protected attributes race(R) and sex(S) which are parents of GPA(G) and LSAT(L). To achieve this, we have first trained on all features including race and sex and then infer Knowledge(K) which is later used to predict FYA(F). While the model's accuracy is not as high as the other models, looking at the density plots this model is not biased against race and sex.

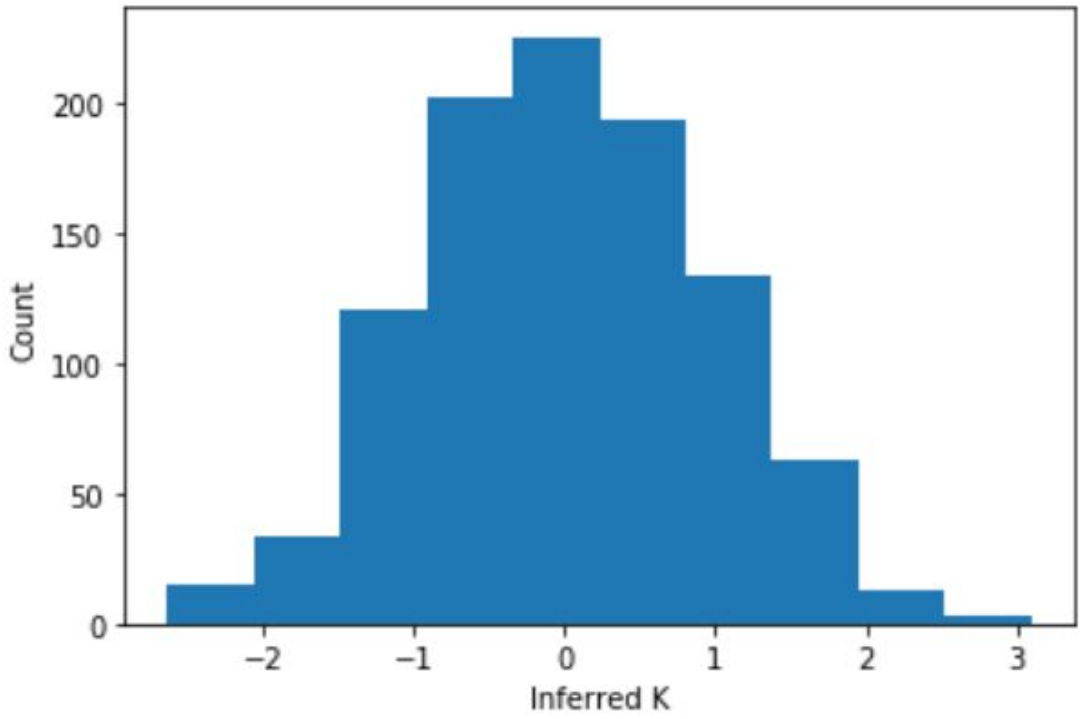
```
aware_sample= []
for i in range(1000):
    trace = trace_handler.get_trace()
    R = trace.nodes['R']['value']
    S = trace.nodes['S']['value']
```

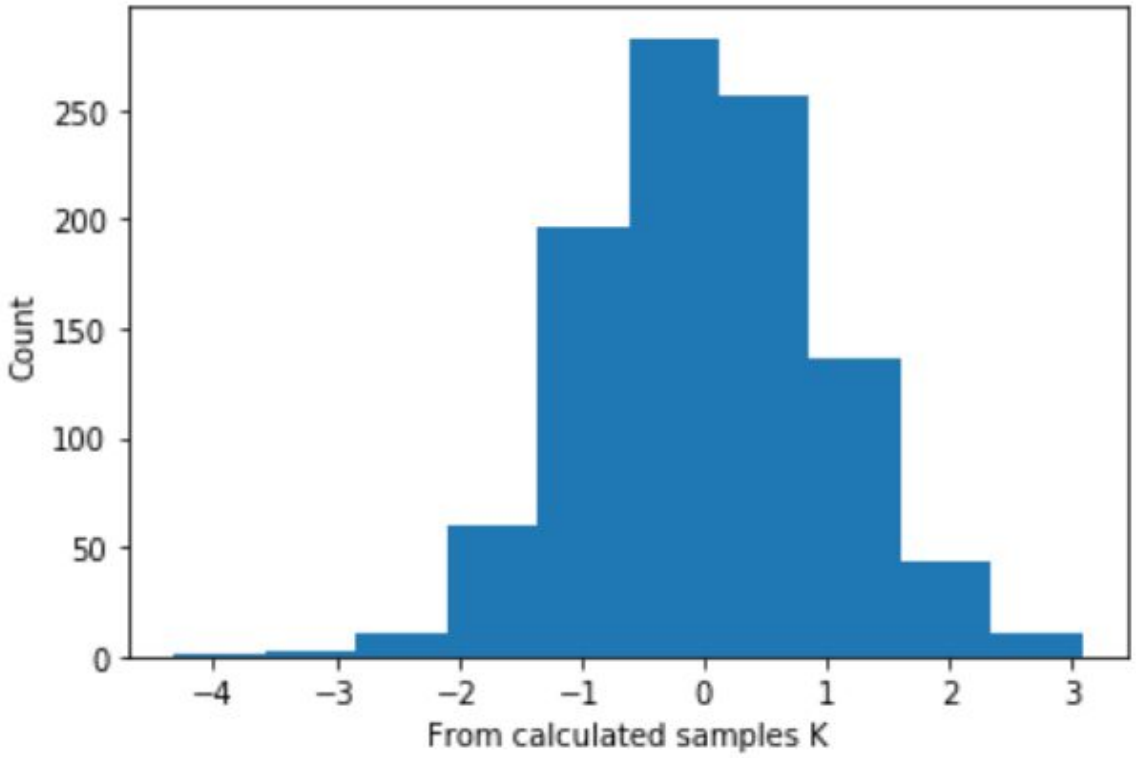
```
    A = trace.nodes['A']['value']
    G = trace.nodes['G']['value']
    L = trace.nodes['L']['value']
    F = trace.nodes['F']['value']
    # get prob of each combination
    log_prob = trace.log_prob_sum()
    p = np.exp(log_prob)
    samples = samples.append({'R': R, 'S': S, 'A': A, 'G': G, 'L':L, 'F':
F, 'p': p}, ignore_index=True)
    aware_sample.append(([R,S,F]))

samples.head()
```

Iterating through the generated samples to use the observed variables GPA and LSAT to infer samples for knowledge(K)

```
# hold all K values
K=[]
# iterate through the 1000 samples of G/L and infer K
for i in range(1000):
        conditioned = pyro.condition(model, data={"G": G[i], "L": L[i]})

        # use pyro.infer.Importance
        posterior = pyro.infer.Importance(conditioned,
num_samples=100).run()
        post_marginal = pyro.infer.EmpiricalMarginal(posterior, "A")
        post_samples = [post_marginal().item() for _ in range(100)]
        post_unique, post_counts = np.unique(post_samples,
return_counts=True)
        # calculate the mean of the inferred
        mean = np.mean(post_samples)
        K.append(mean)
```

We have contrasted the distribution of the inferred K and the generated K samples.


Count
200
150
100
50
0
−2
−1
0
1
2
3
Inferred K

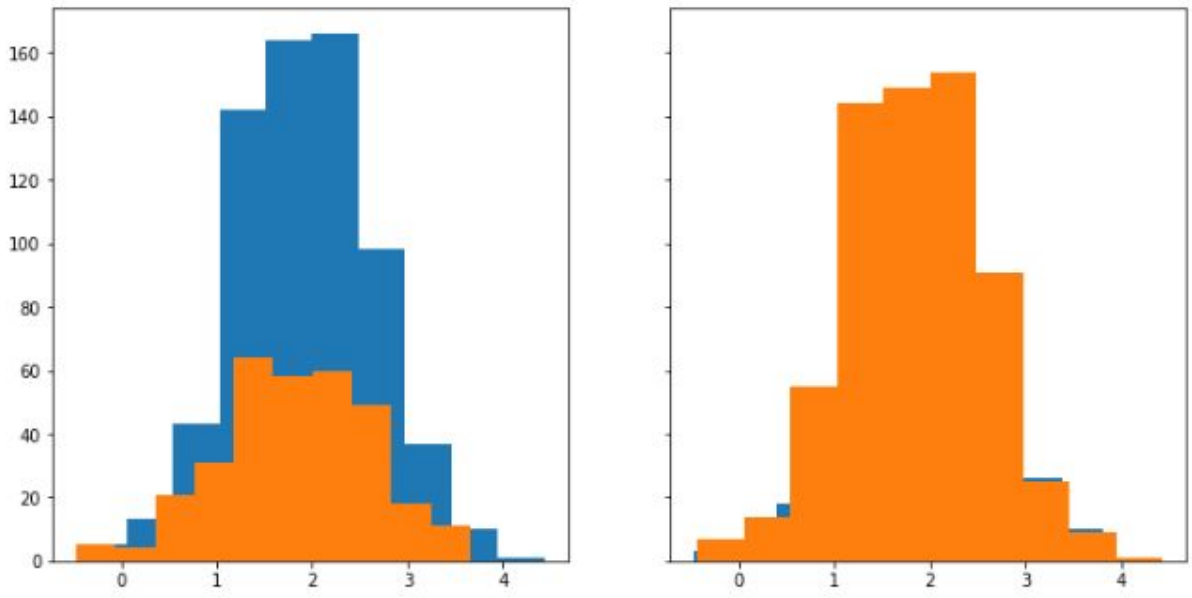

Count
250
200
150
100
50
0
−4
−3
−2
−1
0
1
2
3
From calculated samples K

From the above distribution we can understand that the inferred K's are almost similar to that of the actual K's

Now we have used our new inferred K into the Linear regression model to predict F

```
linear_reg_model = PyroModule[nn.Linear](1, 1)

# Define loss and optimize
loss_fn = torch.nn.MSELoss(reduction='sum')
optim = torch.optim.Adam(linear_reg_model.parameters(), lr=0.05)
num_iterations = 500

def train():
    # run the model forward on the data
    y_pred = linear_reg_model(x_data).squeeze(-1)
    # calculate the mse loss
    loss = loss_fn(y_pred, y_data)
    # initialize gradients to zero
    optim.zero_grad()
    # backpropagate
    loss.backward()
    # take a gradient step
    optim.step()
    return loss
```

We have constructed the density plot:



Conclusions:

After the previous implementation of the counterfactual fairness, we take the protected attributes into consideration, so we are able to take into account the different social biases that may arise towards individuals based on ethically sensitive attributes and compensate for these biases effectively. In our implementation, we are able to capture these social biases and make clear the implicit trade-off between prediction accuracy and fairness in real life situations.

We believe that fairness should be regulated by explicitly modeling the causal structure of the world. Simply relying on probabilistic independence cannot meet the fair standard, it also cannot point out why and how unfairness is occurring. By implementing the previous counterfactual fairness models, we are able to provide practical solutions for solving a wide array of fairness modeling problems.