

PyCFRL Real Data Workflow

The PyCFRL Team

June 2025

1 Step 1: Data Preparation & Preprocessing

1. **Input:** A tabular trajectory data (in .csv) under the prespecified format containing information about the sensitive attributes, states, actions, and rewards of each individual at each time step.
2. Convert the tabular data into 4 Trajectory Arrays: one for sensitive attribute, one for states, one for actions, and one for rewards. (This is needed because PyCFRL classes and functions take in data in the form of Trajectory Arrays.)
3. Split the data randomly into a training set (for example, 80% of data) and a testing set (for example, 20% of data). The training set will be used for policy learning, and the testing set will be used for evaluating the value and counterfactual fairness of the learned policy.
4. **Train preprocessor and preprocess data:** Split the training set into k folds, call them Fold 1, Fold 2, \dots , Fold k . Iterate the following:
 - (a) Train a transition kernel κ_{-1} using all folds of data except Fold 1. Then use κ_{-1} to preprocess data in Fold 1 to obtain the corresponding counterfactual augmented states and rewards. (The counterfactual augmented states refer to the concatenation of all counterfactual states, which is needed by the optimal CF policy.)
 - (b) Train a transition kernel κ_{-2} using all folds of data except Fold 2. Then use κ_{-2} to preprocess data in Fold 2 to obtain the corresponding counterfactual augmented states and rewards.
 - (c) \dots
 - (d) Train a transition kernel κ_{-k} using all folds of data except Fold k . Then use κ_{-k} to preprocess data in Fold k to obtain the corresponding counterfactual augmented states and rewards.

After the above procedure, we should obtain k transition kernels (i.e. $\kappa_{-1}, \dots, \kappa_{-k}$) and the preprocessed version of all folds of the data.

5. Combine the k folds of preprocessed data into a large data set. This forms the preprocessed training set.
6. **Output:** transition kernel $K = \{\kappa_{-1}, \dots, \kappa_{-k}\}$, unpreprocessed training set, unpreprocessed testing set, preprocessed training set. (Note we do not have preprocessed testing set because we did not preprocess the testing set.)

2 Step 2: Policy Learning

1. **Input:** Preprocessed training set (obtained from Step 1).
2. Train an FQI agent using the preprocessed training set (recall the preprocessed training set contains only 80% of all the data). Through this we obtain the (estimated) optimal CF policy.
3. **output:** (Estimated) optimal CF policy π .

3 Step 3: Counterfactual Trajectory Generation

1. **Input:** The unprocessed training set and the unprocessed testing set. In this section, we assume the sensitive attribute is binary (0 or 1), but generalization to the case of higher-dimensional sensitive attributes should be straightforward.
2. Using the full trajectory data (the unprocessed training set and the unprocessed testing set combined), train a transition kernel G that estimates the dynamics of the underlying environment. (Note we now have 2 transition kernels: K and G . K is trained using only the training set and is used to preprocess the counterfactually augmented states used by FQI. G is trained using the full data and is used to generate next states and rewards in our counterfactual trajectories.)
3. We now estimate counterfactual states using G and generate the counterfactual actions using **only the testing set**.
4. **Time $t=0$:** For each individual in the testing set, we let the observed sensitive attribute be some $z_{obs} \in \{0, 1\}$ and do the following:
 - (a) Estimate all of the individual's counterfactual states. That is, if $z_{obs} = 0$ is observed, then we estimate the individual's counterfactual state by $\hat{x}_0^{(1),G} = x_0 - \hat{\mathbb{E}}_G[x_0|0] + \hat{\mathbb{E}}_G[x_0|1]$, where $\hat{\mathbb{E}}_G$ is estimated by the transition kernel G . On the other hand, if $z_{obs} = 1$ is observed, then we estimate the individual's counterfactual state by $\hat{x}_0^{(0),G} = x_0 - \hat{\mathbb{E}}_G[x_0|1] + \hat{\mathbb{E}}_G[x_0|0]$, where $\hat{\mathbb{E}}_G$ is estimated by the transition kernel G . This estimated counterfactual state is treated as the true observed state had the individual belong to the other sensitive attribute group.
 - (b) If $z_{obs} = 0$, then we preprocess x_0 using K to obtain the counterfactual augmented state $\{x_0, \hat{x}_0^{(1),K}\}$, which we then pass into π to obtain an action under $Z = 0$, denoted by $a_0^{(0)}$. We similarly preprocess $\hat{x}_0^{(1),G}$ using K to obtain $\{\hat{x}_0^{(0),K}, \hat{x}_0^{(1),G}\}$ and pass it into π to obtain $a_0^{(1)}$. On the other hand, if $z_{obs} = 1$, then we preprocess x_0 using K to obtain the counterfactual augmented state $\{\hat{x}_0^{(0),K}, x_0\}$, which we then pass into π to obtain an action under $Z = 1$, denoted by $a_0^{(1)}$. We similarly preprocess $\hat{x}_0^{(0),G}$ using K to obtain $\{\hat{x}_0^{(0),G}, \hat{x}_0^{(1),K}\}$ and pass it into π to obtain $a_0^{(0)}$. $a_0^{(0)}$ and $a_0^{(1)}$ are seen as the action that would have been taken by π had the individual belong to the group with $Z = 0$ and $Z = 1$, respectively.
5. **Time $t \geq 1$:** At each $t > 0$, we should have all the counterfactual states at $t - 1$ (i.e. $\{x_{t-1}, x_{t-1}^{(1),G}\}$ if $z_{obs} = 0$ and $\{x_{t-1}^{(0),G}, x_{t-1}\}$ if $z_{obs} = 1$) and all the action under z_{obs} at $t - 1$ (i.e. a_{t-1}). Recall that a_{t-1} might be taken by a behavior policy that is different from π . We do the following for each $t > 0$:
 - (a) For each individual in the testing set, estimate all of his/her counterfactual states. That is, if $z_{obs} = 0$, then we estimate the individual's counterfactual state by $\hat{x}_t^{(1),G} = x_t - \hat{\mathbb{E}}_G[x_t|x_{t-1}, a_{t-1}, 0] + \hat{\mathbb{E}}_G[x_t|x_{t-1}, a_{t-1}, 1]$, where $\hat{\mathbb{E}}_G$ is estimated by the transition kernel G . On the other hand, if $z_{obs} = 1$, then we estimate the individual's counterfactual state by $\hat{x}_t^{(0),G} = x_t - \hat{\mathbb{E}}_G[x_t|x_{t-1}, a_{t-1}, 1] + \hat{\mathbb{E}}_G[x_t|x_{t-1}, a_{t-1}, 0]$, where $\hat{\mathbb{E}}_G$ is estimated by the transition kernel G . This estimated counterfactual state is treated as the true observed state had the individual belong to the other sensitive attribute group.
 - (b) If $z_{obs} = 0$, then we preprocess x_t using K to obtain the counterfactual augmented state $\{x_t, \hat{x}_t^{(1),K}\}$, which we then pass into π to obtain an action under $Z = 0$, denoted by $a_t^{(0)}$. We similarly preprocess $\hat{x}_t^{(1),G}$ using K to obtain $\{\hat{x}_t^{(0),K}, \hat{x}_t^{(1),G}\}$ and pass it into π to obtain $a_t^{(1)}$. On the other hand, if $z_{obs} = 1$, then we preprocess x_t using K to obtain the counterfactual augmented state $\{\hat{x}_t^{(0),K}, x_t\}$, which we then pass into π to obtain an action under $Z = 1$, denoted by $a_t^{(1)}$. We similarly preprocess $\hat{x}_t^{(0),G}$ using K to obtain $\{\hat{x}_t^{(0),G}, \hat{x}_t^{(1),K}\}$ and pass it into π to obtain $a_t^{(0)}$. $a_t^{(0)}$ and $a_t^{(1)}$ are seen as the action that would have been taken by π had the individual belong to the group with $Z = 0$ and $Z = 1$, respectively.

6. **Note:** When preprocessing data using K after K has been trained, the preprocessed data is calculated as the average of the outputs from $\kappa_{-1}, \dots, \kappa_{-k}$.
7. **Output:** Counterfactual trajectories (counterfactual actions).

4 Step 4: CF Metric Calculation

1. **Input:** The counterfactual trajectories obtained from Step 4. Again, we assume a binary sensitive attribute to be consistent with Step 3.
2. Compute the CF metric by comparing $a_t^{(0)}$ and $a_t^{(1)}$ for all time steps t .
3. **Output:** The CF metric.

5 Step 5: Policy Evaluation through FQE

1. **Input:** The estimated optimal policy π , the unpreprocessed testing set.
2. Perform policy evaluation via FQE using the unpreprocessed testing set and π .
3. **Output:** Estimated value of π .

6 Code Implementation of Steps 1-5

In this section, we match each of the above steps to the classes and functions in PyCFRL that perform them.

1. The data preparation in Step 1 is mainly performed by functions in the `reader` module.
2. The preprocessor training and data preprocessing in Step 1 are mainly performed by `train_preprocessor()`.
3. Step 2 is mainly performed by `FQI`.
4. Steps 3 and 4 are mainly performed together by `evaluate_fairness_through_model()`.
5. Step 5 is mainly performed by `evaluate_reward_through_fqe()`.

Please refer to `real_data_workflow.ipynb` for a detailed code implementation of this workflow.