I/O Table

Function Name	Inputs / Parameters	Outputs / Returns	Purpose / Description	Prerequisites	Side Effects	Notable Parameter Ranges
__init__	**data :=** numeric pandas DataFrame **treatment_variable :=** string representing treatment variable **outcome_variable :=** string representing outcome variable	None.	Initialize module with data and causal analysis parameters.	Data should be pandas DataFrame with treatment and outcome variables as columns in that DataFrame. Also ensure that the data is all numeric — this is very important.	None.	None.
find_causal_graph	***algo* :=** string representing causal discovery algorithm ***pk* :=** dict with 'required' and/or 'forbidden' edges	NetworkX DiGraph representing causal graphical structure.	Run causal discovery using specified algorithm and apply prior knowledge if provided.	Prior knowledge (if any) must be dict with two keys (1) 'required' (2) 'forbidden'. The values for each key are lists where each element of the list is a tuple representing an edge to require and forbid.	Sets an internal instance parameter **self.graph** to be the resulting discovered causal graph.	Ensure that **algo** is one of the following settings: [‘pc’, ‘ges’, ‘icalingam’, ‘boss’, ‘grasp’]
input_causal_graph	**graph**:= NetworkX DiGraph	The input graph	Accept a user-provided causal graph instead of running discovery.	Graph must be NetworkX DiGraph with nodes matching data columns	Sets an internal instance parameter **self.graph** to be the resulting discovered causal graph.	None.
refute_cgm	***n_perm*** := number of permutations to perform. If -1 use all n_nodes! permutations. ***apply_sugst* :=** boolean to apply or not apply suggestions from the refutation ***show_plt* :=** boolean to plot histogram of results from permutation baseline ***indep_test*** := callable function to encapsulate the independence test to use for checking pairwise independencies ***cond_indep_test*** := callable function to encapsulate the conditional independence test to use ***signficance_level*** := float value for the significance level for the permutation test ***significance_ci*** := float value for the siginificance level for the conditional indpendence test ***show_progress_bar*** := boolean for whether to show progress bar over permutations ***n_jobs*** := number of jobs to use for parallel execution of conditional independence tests ***plot_kwargs*** := additional plot arguments to be passed to plot_evaluation_results ***allow_data_subset*** := boolean for whetherthe evaluation should be performed even if data is only available for a subset of nodes	Causal graph (NetworkX DiGraph) with or without suggestions from refutation applied — depends on what you had set **apply_sugst** to be.	Refute the causal graph by permutation tests and optionally apply suggestions from the refutation.	Causal graph and data must be available.	Stores the resulting refutation result object (from DoWhy) into an internal instance parameter **self.graph_ref**.	For **n_per**, usually setting it to 50 is sufficient — this is also the default. You can leave all the other parameters as default setting as well.
create_model	None (uses instance variables)	Dowhy CausalModel object	Creates DoWhy causal model from graph, treatment, outcome, and data.	Causal graph, data, treatment and outcome variables must be set	Sets an internal instance parameter **self.model** to be the DoWhy CausalModel object	None.
identify_effect	***method*** := string to represent the method to use for effect identification	Dowhy IdentifiedEstimand object	Identifies causal effect estimand based on causal model and chosen identification method.	This follows from the **create_model** function.	Sets an internal instance parameter **self.estimand** to be the identified estimand — this is DoWhy’s IdentifiedEstimand object.	Note that you can also use other **method**s for the identification process. Below are method descriptions taken directly from DoWhy's documentation: • maximal-adjustment: returns the maximal set that satisfies the backdoor criterion. This is usually the fastest way to find a valid backdoor set, but the set may contain many superfluous variables. • minimal-adjustment: returns the set with minimal number of variables that satisfies the backdoor criterion. This may take longer to execute, and sometimes may not return any backdoor set within the maximum number of iterations. • exhaustive-search: returns all valid backdoor sets. This can take a while to run for large graphs. Keeping the default is a good mix of minimal and maximal adjustment. It starts with maximal adjustment which is usually fast. It then runs minimal adjustment and returns the set having the smallest number of variables.
estimate_effect	***method_cat*** := string representing method category to use for effect estimation **trtm_val :=** number to set the treatment value for the treatment variable **ctrl_val :=** number to set the control value for the treatment variable	Dowhy CausalEstimate object	Estimates treatment effect using identified estimand and specified estimation method.	Must call **identify_effect** first; treatment/control values must be set	Sets an internal instance parameter **self.estimate** to be the estimated effect— this is DoWhy’s CausalEstimate object.	**method_cat** can be: • Propensity Score Matching: “backdoor.propensity_score _matching" • Propensity Score Stratification: "backdoor.propensity_score_stratification" • Propensity Score-based Inverse Weighting: "backdoor.propensity_score_weighting" • Linear Regression: "backdoor.linear_regression" • Generalized Linear Models (e.g., logistic regression): "backdoor.generalized_linear_model" • Instrumental Variables: "iv.instrumental_variable" • Regression Discontinuity: "iv.regression_discontinuity" • Two Stage Regression: "frontdoor.two_stage_regression"
refute_estimate	None	Dowhy CausalRefutation object or list of such objects	Performs refutation tests on effect estimates to check robustness.	Effect estimate must be available	Sets an internal instance parameter **self.est_ref** to be a list of the refutation objects returned from DoWhy’s refuters.	None.
simulate_intervention	**variable_to_intervene_dict** := dict for the variable(s) to intervene on **num_samples_to_draw** := number of samples to draw from the interventional distribution	DataFrame with samples from interventional distribution	Simulates an intervention on variable(s) and returns samples from the interventional distribution using DoWhy’s probabilistic causal model.	CausalModel and data must be available	Sets an internal instance parameter **self.interventional_samples** to be the resulting sampled DataFrame.	**variable_to_intervene_dict** should be a dict that contains variable to value mapping {'v_name': lambda x: #value}
store_results	***dir_path*** := string to represent the directory to save the CSV files	None (saves multiple CSV files and stores results dict)	Stores causal graph properties, refutation metrics, quality scores, effect estimates, refutation results, and interventional samples to disk and in self.results.	The full pipeline must be completed first	None.	None.

Bold and italicized parameters are those that are optional to set