

Sensitivity Analysis for Regression Models

Sensitivity analysis helps us examine how sensitive a result is against the possibility of unobserved confounding. Two methods are implemented: 1. [Cinelli & Hazlett's robustness value](#) - This method only supports linear regression estimator. - The partial R^2 of treatment with outcome shows how strongly confounders explaining all the residual outcome variation would have to be associated with the treatment to eliminate the estimated effect. - The robustness value measures the minimum strength of association unobserved confounding should have with both treatment and outcome in order to change the conclusions. - Robustness value close to 1 means the treatment effect can handle strong confounders explaining almost all residual variation of the treatment and the outcome. - Robustness value close to 0 means that even very weak confounders can also change the results. - Benchmarking examines the sensitivity of causal inferences to plausible strengths of the omitted confounders. - This method is based on [https://carloscinelli.com/files/Cinelli%20and%20Hazlett%20\(2020\)%20-%20Making%20Sense%20of%20Sensitivity.pdf](https://carloscinelli.com/files/Cinelli%20and%20Hazlett%20(2020)%20-%20Making%20Sense%20of%20Sensitivity.pdf) 2. [Ding & VanderWeele's E-Value](#) - This method supports linear and logistic regression. - The E-value is the minimum strength of association on the risk ratio scale that an unmeasured confounder would need to have with both the treatment and the outcome, conditional on the measured covariates, to fully explain away a specific treatment-outcome association. - The minimum E-value is 1, which means that no unmeasured confounding is needed to explain away the observed association (i.e. the confidence interval crosses the null). - Higher E-values mean that stronger unmeasured confounding is needed to explain away the observed association. There is no maximum E-value. - [McGowan & Greevy Jr's](#) benchmarks the E-value against the measured confounders.

Step 1: Load required packages

```
[1]: import os, sys
     sys.path.append(os.path.abspath("../.."))
     import dowhy
```

[Skip to main content](#)

```
import numpy as np
import dowhy.datasets

# Config dict to set the logging level
import logging.config
DEFAULT_LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'loggers': {
        '': {
            'level': 'ERROR',
        },
    },
}

logging.config.dictConfig(DEFAULT_LOGGING)
# Disabling warnings output
import warnings
from sklearn.exceptions import DataConversionWarning
#warnings.filterwarnings(action='ignore', category=DataConversionWarning)
```

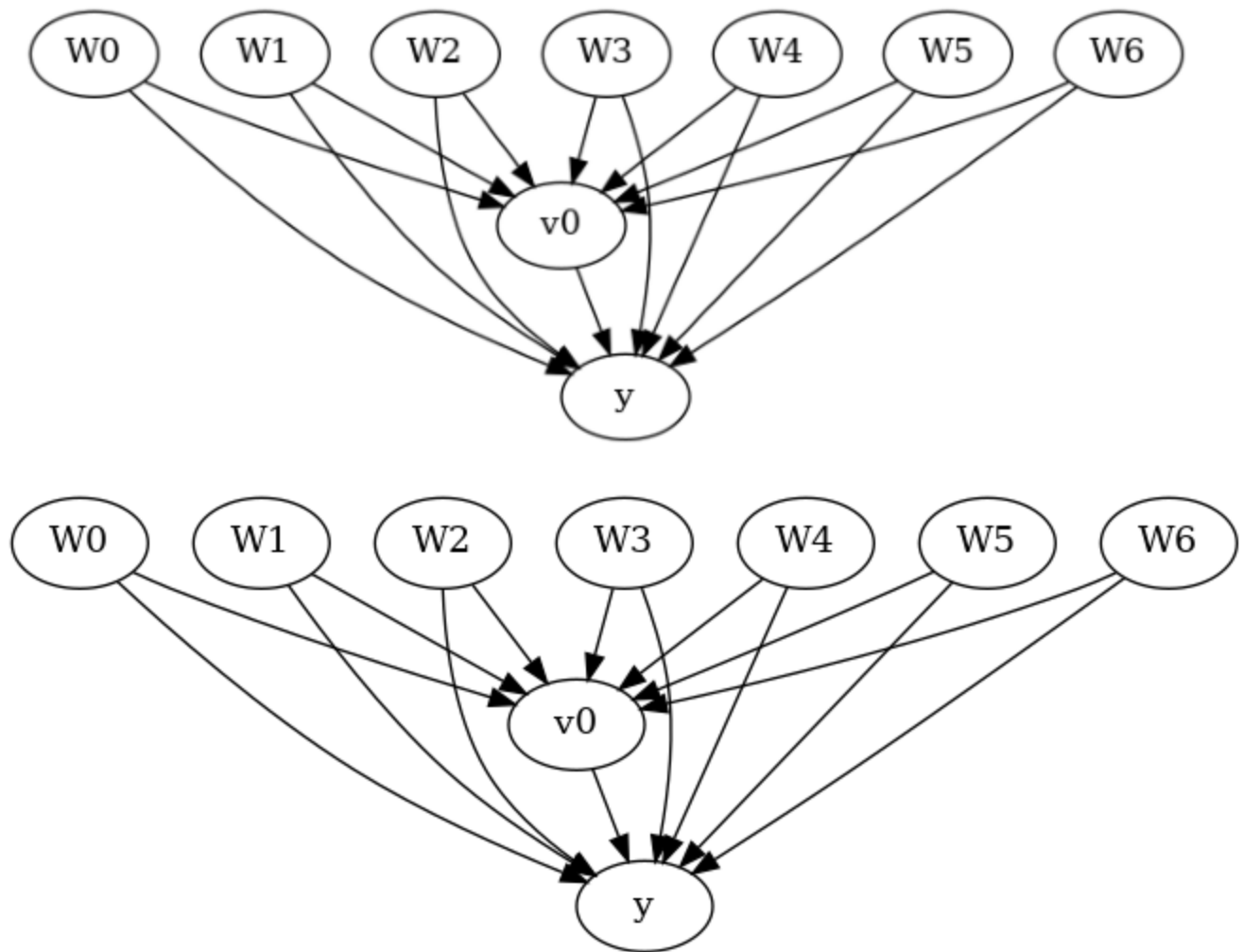
Step 2: Load the dataset

We create a dataset with linear relationships between common causes and treatment, and common causes and outcome. Beta is the true causal effect.

```
[2]: np.random.seed(100)
data = dowhy.datasets.linear_dataset( beta = 10,
                                     num_common_causes = 7,
                                     num_samples = 500,
                                     num_treatments = 1,
                                     stddev_treatment_noise = 10,
                                     stddev_outcome_noise = 5
                                     )
```

```
[3]: model = CausalModel(
        data=data["df"],
        treatment=data["treatment_name"],
        outcome=data["outcome_name"],
        graph=data["gml_graph"],
        test_significance=None,
    )
model.view_model()
from IPython.display import Image, display
display(Image(filename="causal_model.png"))
data['df'].head()
```

[Skip to main content](#)



[3]:

	W0	W1	W2	W3	W4	W5	W6	v0	y
0	-0.145062	-0.235286	0.784843	0.869131	-1.567724	-1.290234	0.116096	True	1.386517
1	-0.228109	-0.020264	-0.589792	0.188139	-2.649265	-1.764439	-0.167236	False	-16.159402
2	0.868298	-1.097642	-0.109792	0.487635	-1.861375	-0.527930	-0.066542	False	-0.702560
3	-0.017115	1.123918	0.346060	1.845425	0.848049	0.778865	0.596496	True	27.714465
4	-0.757347	-1.426205	-0.457063	1.528053	-2.681410	0.394312	-0.687839	False	-20.082633

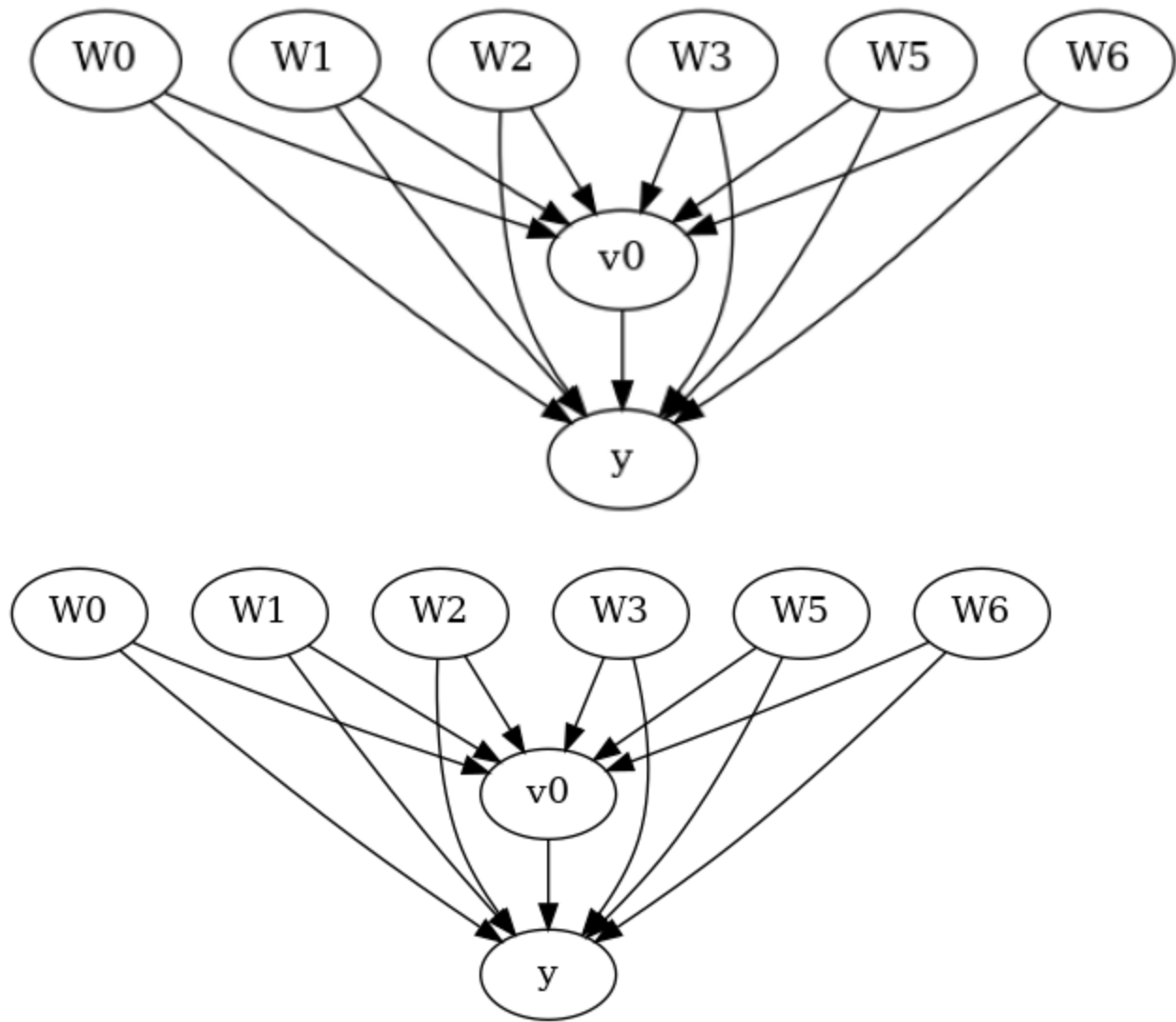
Step 3: Create Causal Model

Remove one of the common causes to simulate unobserved confounding

```
[4]: data["df"] = data["df"].drop("W4", axis = 1)
graph_str = 'graph[directed 1node[ id "y" label "y"]node[ id "W0" label "W0"] n
model = CausalModel(
    data=data["df"],
    treatment=data["+treatment name"]
```

[Skip to main content](#)

```
        test_significance=None,
    )
model.view_model()
from IPython.display import Image, display
display(Image(filename="causal_model.png"))
data['df'].head()
```



[4]:

	w0	w1	w2	w3	w5	w6	v0	y
0	-0.145062	-0.235286	0.784843	0.869131	-1.290234	0.116096	True	1.386517
1	-0.228109	-0.020264	-0.589792	0.188139	-1.764439	-0.167236	False	-16.159402
2	0.868298	-1.097642	-0.109792	0.487635	-0.527930	-0.066542	False	-0.702560
3	-0.017115	1.123918	0.346060	1.845425	0.778865	0.596496	True	27.714465
4	-0.757347	-1.426205	-0.457063	1.528053	0.394312	-0.687839	False	-20.082633

Step 4: Identification

[Skip to main content](#)

```
identified_estimand = model.identify_effect(proceed_when_unidentifiable=True)
print(identified_estimand)
```

Estimand type: EstimandType.NONPARAMETRIC_ATE

Estimand : 1

Estimand name: backdoor

Estimand expression:

$$\frac{d}{d[v_0]}(E[y|W_3, W_6, W_1, W_5, W_0, W_2])$$

Estimand assumption 1, Unconfoundedness: If $U \rightarrow \{v_0\}$ and $U \rightarrow y$ then $P(y|v_0, W_3, W_6, W_1$

Estimand : 2

Estimand name: iv

No such variable(s) found!

Estimand : 3

Estimand name: frontdoor

No such variable(s) found!

Step 5: Estimation

Currently only Linear Regression estimator is supported for Linear Sensitivity Analysis

```
[6]: estimate = model.estimate_effect(identified_estimand, method_name="backdoor.linear_regression")
print(estimate)
```

*** Causal Estimate ***

Identified estimand

Estimand type: EstimandType.NONPARAMETRIC_ATE

Estimand : 1

Estimand name: backdoor

Estimand expression:

$$\frac{d}{d[v_0]}(E[y|W_3, W_6, W_1, W_5, W_0, W_2])$$

Estimand assumption 1, Unconfoundedness: If $U \rightarrow \{v_0\}$ and $U \rightarrow y$ then $P(y|v_0, W_3, W_6, W_1$

Realized estimand

b: $y \sim v_0 + W_3 + W_6 + W_1 + W_5 + W_0 + W_2$

Target units: ate

Estimate

Mean value: 10.697677486880925

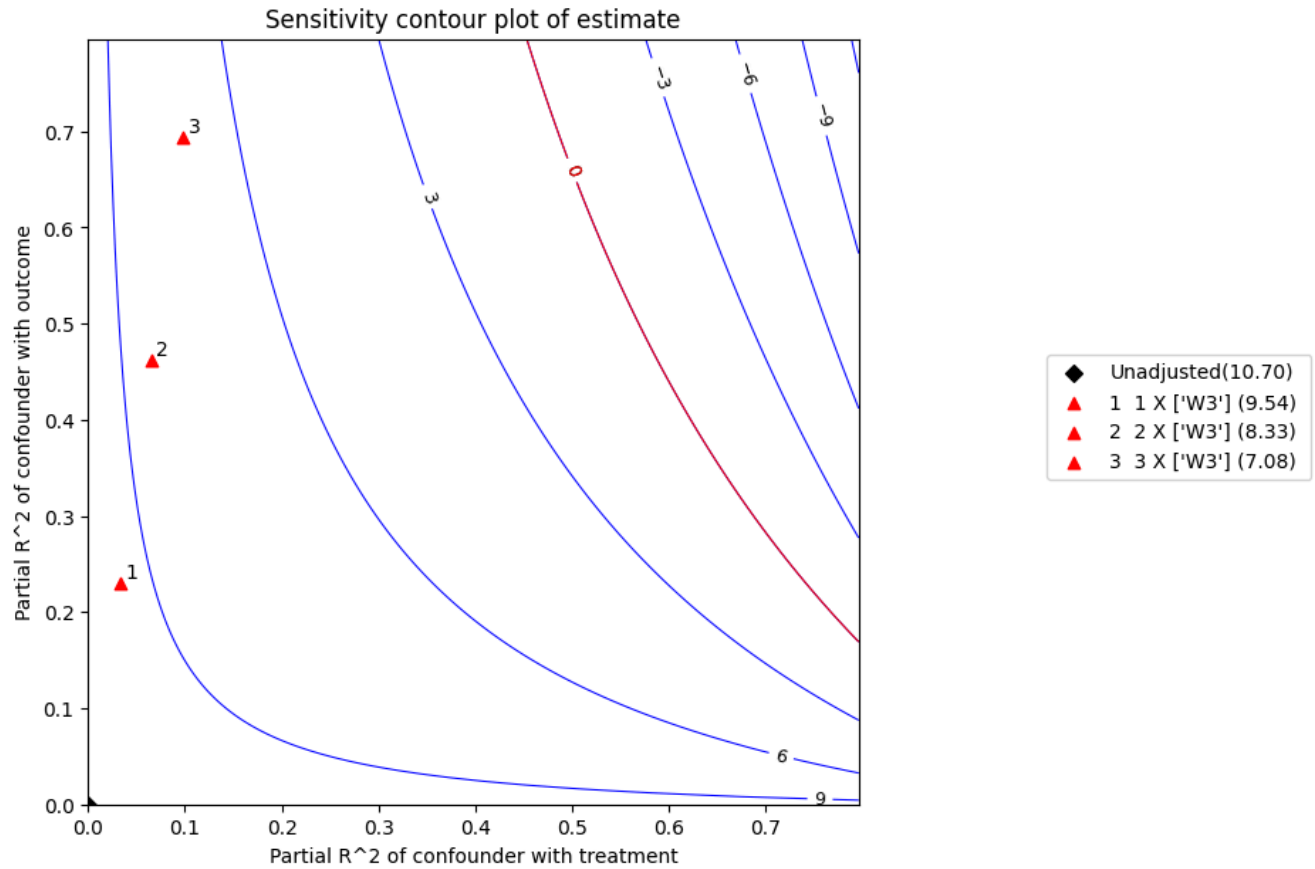
[Skip to main content](#)

Step 6a: Refutation and Sensitivity Analysis – Method 1

identified_estimand: An instance of the identifiedEstimand class that provides the information with respect to which causal pathways are employed when the treatment effects the outcome estimate: An instance of CausalEstimate class. The estimate obtained from the estimator for the original data. method_name: Refutation method name simulation_method: "linear-partial-R2" for Linear Sensitivity Analysis benchmark_common_causes: Name of the covariates used to bound the strengths of unobserved confounder percent_change_estimate: It is the percentage of reduction of treatment estimate that could alter the results (default = 1) if percent_change_estimate = 1, the robustness value describes the strength of association of confounders with treatment and outcome in order to reduce the estimate by 100% i.e bring it down to 0. confounder_increases_estimate: confounder_increases_estimate = True implies that confounder increases the absolute value of estimate and vice versa. Default is confounder_increases_estimate = False i.e. the considered confounders pull estimate towards zero effect_fraction_on_treatment: Strength of association between unobserved confounder and treatment compared to benchmark covariate effect_fraction_on_outcome: Strength of association between unobserved confounder and outcome compared to benchmark covariate null_hypothesis_effect: assumed effect under the null hypothesis (default = 0) plot_estimate: Generate contour plot for estimate while performing sensitivity analysis. (default = True). To override the setting, set plot_estimate = False.

```
[7]: refute = model.refute_estimate(identified_estimand, estimate ,
                                   method_name = "add_unobserved_common_cause",
                                   simulation_method = "linear-partial-R2",
                                   benchmark_common_causes = ["W3"],
                                   effect_fraction_on_treatment = [ 1,2,3]
                                   )
```

[Skip to main content](#)



The x axis shows hypothetical partial R^2 values of unobserved confounder(s) with the treatment. The y axis shows hypothetical partial R^2 of unobserved confounder(s) with the outcome. The contour levels represent adjusted t-values or estimates for unobserved confounders with hypothetical partial R^2 values when these would be included in full regression model. The red line is the critical threshold: confounders with such strength or stronger are sufficient to invalidate the research conclusions.

```
[8]: refute.stats
```

```
[8]: {'estimate': 10.697677486880925,
      'standard_error': 0.5938735661282949,
      'degree of freedom': 492,
      't_statistic': 18.013392238727626,
      'r2yt_w': 0.3974149837266683,
      'partial_f2': 0.6595168698094567,
      'robustness_value': 0.5467445572181009,
      'robustness_value_alpha': 0.5076289101030926}
```

```
[9]: refute.benchmarking_results
```

	r2tu_w	r2yu_tw	bias_adjusted_estimate	bias_adjusted_se	bias_adjusted_t	bias_adjusted_lowe
0	0.032677	0.230238	9.535964	0.530308	17.981928	8.494
1	0.065354	0.461490	8.331381	0.451241	18.463243	7.444

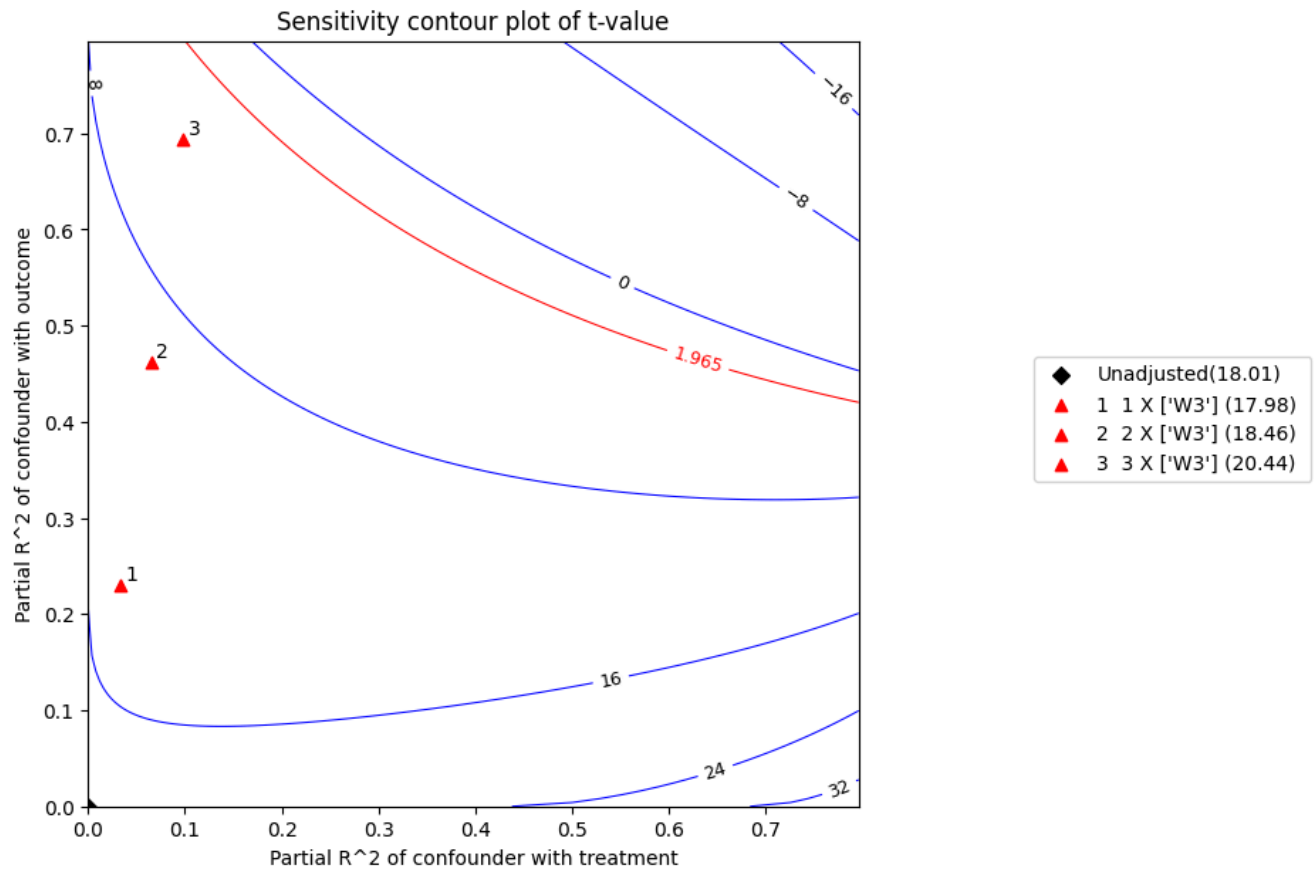
[Skip to main content](#)

Parameter List for plot function

`plot_type`: "estimate" or "t-value" `critical_value`: special reference value of the estimate or t-value that will be highlighted in the plot `x_limit`: plot's maximum x_axis value (default = 0.8) `y_limit`: plot's minimum y_axis value (default = 0.8) `num_points_per_contour`: number of points to calculate and plot each contour line (default = 200) `plot_size`: tuple denoting the size of the plot (default = (7,7)) `contours_color`: color of contour line (default = blue) String or array. If array, lines will be plotted with the specific color in ascending order. `critical_contour_color`: color of threshold line (default = red) `label_fontsize`: fontsize for labelling contours (default = 9) `contour_linewidths`: linewidths for contours (default = 0.75) `contour_linestyles`: linestyles for contours (default = "solid") See : https://matplotlib.org/3.5.0/gallery/lines_bars_and_markers/linestyles.html `contours_label_color`: color of contour line label (default = black) `critical_label_color`: color of threshold line label (default = red) `unadjusted_estimate_marker`: marker type for unadjusted estimate in the plot (default = 'D') See: https://matplotlib.org/stable/api/markers_api.html `unadjusted_estimate_color`: marker color for unadjusted estimate in the plot (default = "black") `adjusted_estimate_marker`: marker type for bias adjusted estimates in the plot (default = '^') `adjusted_estimate_color`: marker color for bias adjusted estimates in the plot (default = "red") `legend_position`: tuple denoting the position of the legend (default = (1.6, 0.6))

```
[10]: refute.plot(plot_type = 't-value')
```

[Skip to main content](#)



The t statistic is the coefficient divided by its standard error. The higher the t-value, the greater the evidence to reject the null hypothesis. According to the above plot, at 5% significance level, the null hypothesis of zero effect would be rejected given the above confounders.

```
[11]: print(refute)
```

Sensitivity Analysis to Unobserved Confounding using R^2 parameterization

Unadjusted Estimates of Treatment ['v0'] :

Coefficient Estimate : 10.697677486880925

Degree of Freedom : 492

Standard Error : 0.5938735661282949

t-value : 18.013392238727626

F^2 value : 0.6595168698094567

Sensitivity Statistics :

Partial R^2 of treatment with outcome : 0.3974149837266683

Robustness Value : 0.5467445572181009

Interpretation of results :

Any confounder explaining less than 54.67% percent of the residual variance of

For a significance level of 5.0%, any confounder explaining more than 50.76% pe

If confounders explained 100% of the residual variance of the outcome, they wou

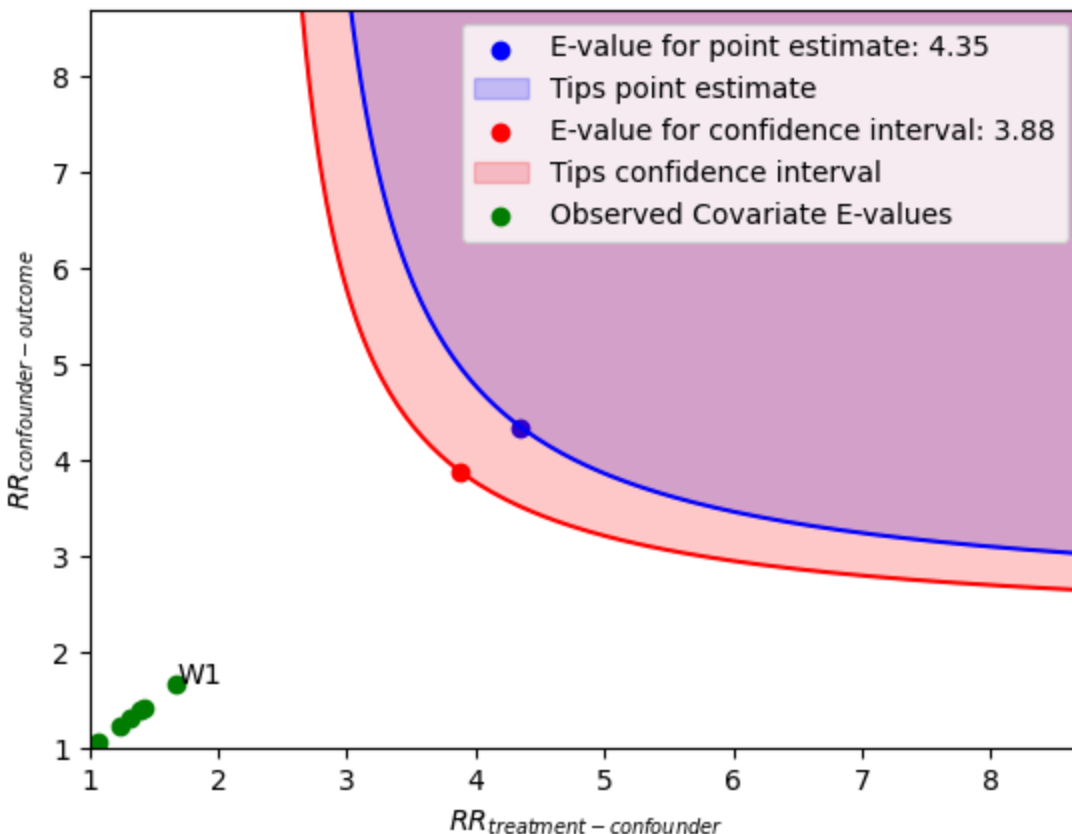
[Skip to main content](#)

Step 6b: Refutation and Sensitivity Analysis - Method 2

simulated_method_name: "e-value" for E-value

- num_points_per_contour: number of points to calculate and plot for each contour (Default = 200)
- plot_size: size of the plot (Default = (6.4,4.8))
- contour_colors: colors for point estimate and confidence limit contour (Default = ["blue", "red"])
- xy_limit: plot's maximum x and y value. Default is 2 x E-value. (Default = None)

```
[12]: refute = model.refute_estimate(identified_estimand, estimate,
                                     method_name = "add_unobserved_common_cause",
                                     simulation_method = "e-value",
                                     )
```



- The x axis shows hypothetical values of the risk ratio of an unmeasured confounder at different levels of the treatment. The y axis shows hypothetical values of the risk ratio of the outcome at different levels of the confounder.

[Skip to main content](#)

- Points lying on or above the blue contour represent combinations of these risk ratios that would tip (i.e. explain away) the point estimate.
- Points lying on or above the red contour represent combinations of these risk ratios that would tip (i.e. explain away) the confidence interval.
- The green points are Observed Covariate E-values. These measure how much the limiting bound of the confidence interval changes on the E-value scale after a specific covariate is dropped and the estimator is re-fit. The covariate that corresponds to the largest Observed Covariate E-value is labeled.

```
[13]: refute.stats
```

```
[13]: {'converted_estimate': 2.4574470657354244,
      'converted_lower_ci': 2.2288567270043154,
      'converted_upper_ci': 2.7094815057979966,
      'evalue_estimate': 4.349958364289867,
      'evalue_lower_ci': 3.883832733630416,
      'evalue_upper_ci': None}
```

```
[14]: refute.benchmarking_results
```

```
[14]:
```

	converted_est	converted_lower_ci	converted_upper_ci	observed_covariate_e_value
dropped_covariate				
W1	2.992523	2.666604	3.358276	1.681140
W6	2.687656	2.446341	2.952774	1.424835
W3	2.692807	2.422402	2.993396	1.394043
W0	2.599868	2.368550	2.853776	1.320751
W5	2.553701	2.315245	2.816716	1.239411
W2	2.473648	2.217699	2.759138	1.076142

```
[15]: print(refute)
```

Sensitivity Analysis to Unobserved Confounding using the E-value

Unadjusted Estimates of Treatment: v0

Estimate (converted to risk ratio scale): 2.4574470657354244

Lower 95% CI (converted to risk ratio scale): 2.2288567270043154

Upper 95% CI (converted to risk ratio scale): 2.7094815057979966

Sensitivity Statistics:

E-value for point estimate: 4.349958364289867

E-value for lower 95% CI: 3.883832733630416

E-value for upper 95% CI: None

Largest Observed Covariate E-value: 1.6811401167656765 (W1)

Interpretation of results:

Unmeasured confounder(s) would have to be associated with a 4.35-fold increase

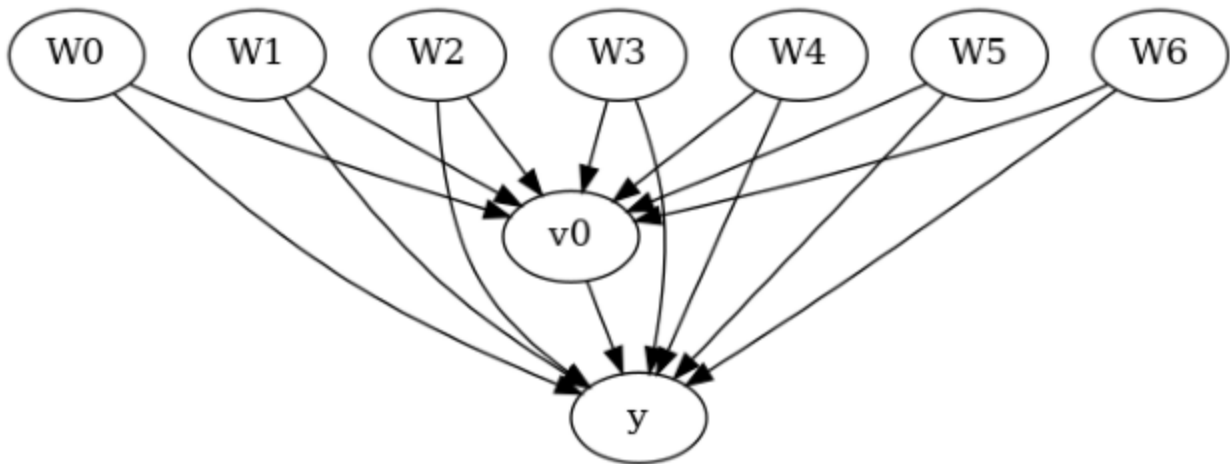
Unmeasured confounder(s) would have to be associated with a 3.88-fold increase

[Skip to main content](#)

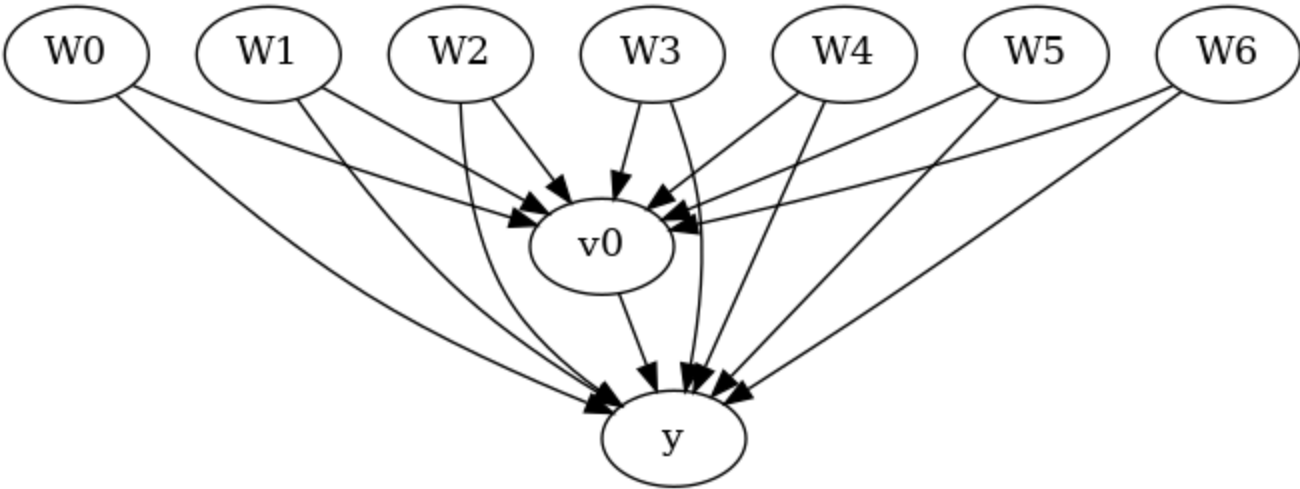
We now run the sensitivity analysis for the same dataset but without dropping any variable. We get a robustness value goes from 0.55 to 0.95 which means that treatment effect can handle strong confounders explaining almost all residual variation of the treatment and the outcome.

```
[16]: np.random.seed(100)
data = dowhy.datasets.linear_dataset( beta = 10,
                                     num_common_causes = 7,
                                     num_samples = 500,
                                     num_treatments = 1,
                                     stddev_treatment_noise=10,
                                     stddev_outcome_noise = 1
                                     )
```

```
[17]: model = CausalModel(
        data=data["df"],
        treatment=data["treatment_name"],
        outcome=data["outcome_name"],
        graph=data["gml_graph"],
        test_significance=None,
    )
model.view_model()
from IPython.display import Image, display
display(Image(filename="causal_model.png"))
data['df'].head()
```



[Skip to main content](#)



[17]:

	W0	W1	W2	W3	W4	W5	W6	v0	y
0	-0.145062	-0.235286	0.784843	0.869131	-1.567724	-1.290234	0.116096	True	6.311809
1	-0.228109	-0.020264	-0.589792	0.188139	-2.649265	-1.764439	-0.167236	False	-12.274406
2	0.868298	-1.097642	-0.109792	0.487635	-1.861375	-0.527930	-0.066542	False	-6.487561
3	-0.017115	1.123918	0.346060	1.845425	0.848049	0.778865	0.596496	True	24.653183
4	-0.757347	-1.426205	-0.457063	1.528053	-2.681410	0.394312	-0.687839	False	-13.770396

[18]:

```
identified_estimand = model.identify_effect(proceed_when_unidentifiable=True)
print(identified_estimand)
```

Estimand type: EstimandType.NONPARAMETRIC_ATE

Estimand : 1

Estimand name: backdoor

Estimand expression:

$$d_{[v_0]}(E[y|W_3, W_6, W_1, W_5, W_0, W_4, W_2])$$

Estimand assumption 1, Unconfoundedness: If $U \rightarrow \{v_0\}$ and $U \rightarrow y$ then $P(y|v_0, W_3, W_6, W_1$

Estimand : 2

Estimand name: iv

No such variable(s) found!

Estimand : 3

Estimand name: frontdoor

No such variable(s) found!

[19]:

```
estimate = model.estimate_effect(identified_estimand, method_name="backdoor.line
print(estimate)
```

*** Causal Estimate ***

Identified estimand

Estimand type: EstimandType.NONPARAMETRIC_ATE

Estimand : 1

[Skip to main content](#)

$$d = \frac{d[v_0]}{d[v_0]} (E[y|W_3, W_6, W_1, W_5, W_0, W_4, W_2])$$

Estimand assumption 1, Unconfoundedness: If $U \rightarrow \{v_0\}$ and $U \rightarrow y$ then $P(y|v_0, W_3, W_6, W_1$

Realized estimand
 b: $y \sim v_0 + W_3 + W_6 + W_1 + W_5 + W_0 + W_4 + W_2$
 Target units: ate

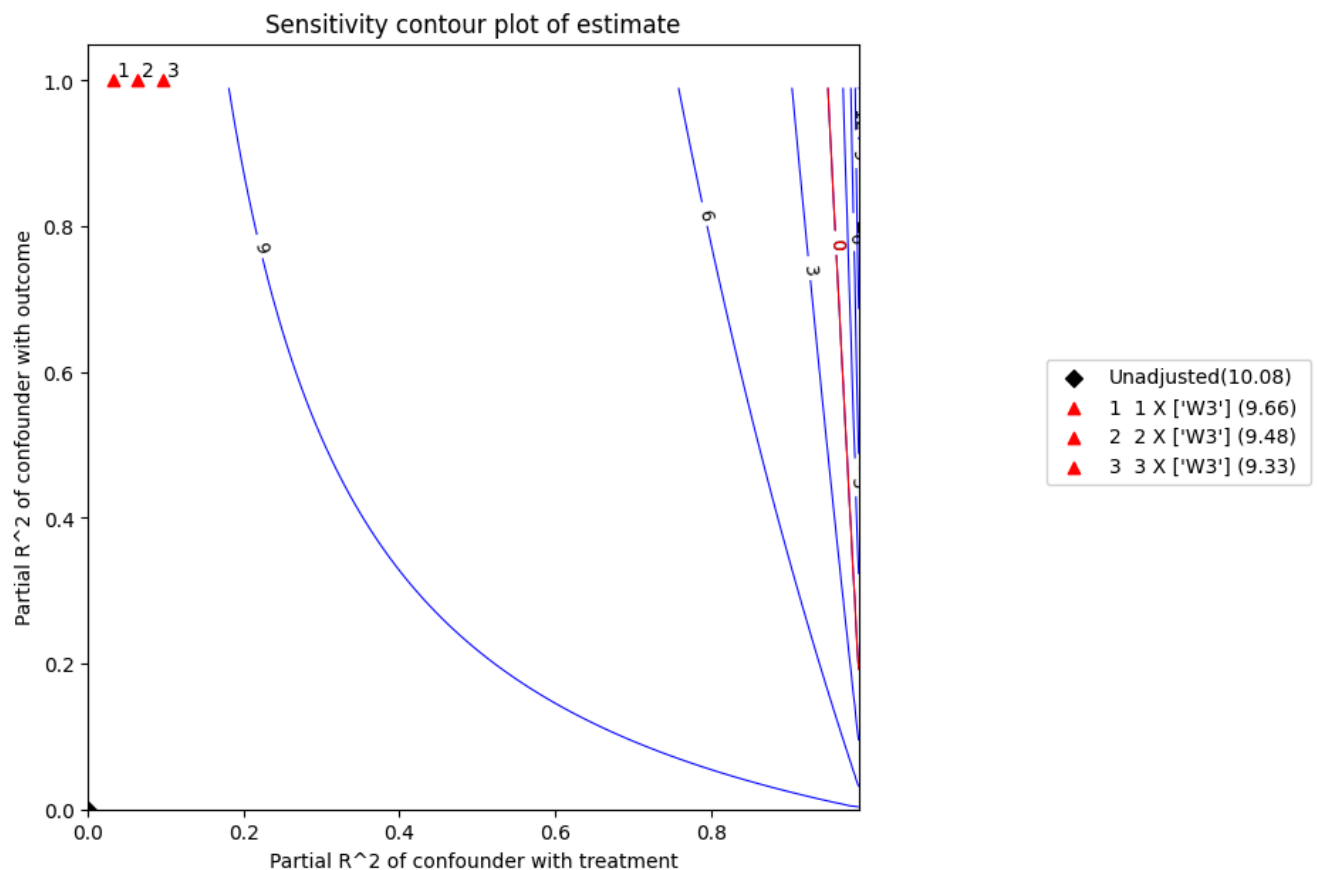
Estimate
 Mean value: 10.081924375588297

```

[20]: refute = model.refute_estimate(identified_estimand, estimate,
                                     method_name = "add_unobserved_common_cause",
                                     simulation_method = "linear-partial-R2",
                                     benchmark_common_causes = ["W3"],
                                     effect_fraction_on_treatment = [1, 2, 3])

/_w/dowhy/dowhy/dowhy/causal_refuters/linear_sensitivity_analyzer.py:199: RuntimeWarning:
  bias_adjusted_t = (bias_adjusted_estimate - self.null_hypothesis_effect) / bi
/_w/dowhy/dowhy/dowhy/causal_refuters/linear_sensitivity_analyzer.py:201: RuntimeWarning:
  bias_adjusted_partial_r2 = bias_adjusted_t**2 / (

```



```

[21]: p.refute.plot(plot_type = 't-value')

```

[Causal Discovery example](#)

Sensitivity contour plot of t-value

[Sensitivity analysis for non-parametric causal estimators](#)



[Skip to main content](#)

© Copyright 2022, PyWhy contributors.

Built with the [PyData Sphinx Theme](#) 0.14.4.

Created using [Sphinx](#) 7.1.2.