Simulation-based sensitivity analysis

This refutation does not return a p-value. Instead, it provides a sensitivity test on how quickly the estimate changes if the identifying assumptions (used in identify_effect) are not valid. Specifically, it checks sensitivity to violation of the backdoor assumption: that all common causes are observed.

To do so, it creates a new dataset with an additional common cause between treatment and outcome. To capture the effect of the common cause, the method takes as input the strength of common cause's effect on treatment and outcome. Based on these inputs on the common cause's effects, it changes the treatment and outcome values and then reruns the estimator. The hope is that the new estimate does not change drastically with a small effect of the unobserved common cause, indicating a robustness to any unobserved confounding.

Another equivalent way of interpreting this procedure is to assume that there was already unobserved confounding present in the input data. The change in treatment and outcome values removes the effect of whatever unobserved common cause was present in the original data. Then rerunning the estimator on this modified data provides the correct identified estimate and we hope that the difference between the new estimate and the original estimate is not too high, for some bounded value of the unobserved common cause's effect.

Importance of domain knowledge: This test requires domain knowledge to set plausible input values of the effect of unobserved confounding. We first show the result for a single value of confounder's effect on treatment and outcome.

```
>>> res_unobserved=model.refute_estimate(identified_estimand, estimate, method_nam
>>>                                       confounders_effect_on_treatment="binary_fl
>>>                                      effect_strength_on_treatment=0.01, effect_s
>>> print(res_unobserved)
```

It is often more useful to inspect the trend as the effect of unobserved confounding is increased. For that, we can provide an array of hypothesized confounders' effects. The output is the (min,

```
>>> res_unobserved_range=model.refute_estimate(identified_estimand, estimate, meth
>>>                                     confounders_effect_on_treatment="binary_fl
>>>                                    effect_strength_on_treatment=np.array([0.00
>>> print(res_unobserved_range)
```

The above plot shows how the estimate decreases as the hypothesized confounding on treatment increases. By domain knowledge, we may know the maximum plausible confounding effect on treatment. Since we see that the effect does not go beyond zero, we can safely conclude that the causal effect of treatment v0 is positive.

We can also vary the confounding effect on both treatment and outcome. We obtain a heatmap.a

```
>>> res_unobserved_range=model.refute_estimate(identified_estimand, estimate, meth
>>>                                           confounders_effect_on_treatment="bin
>>>                                           effect_strength_on_treatment=[0.001,
>>>                                           effect_strength_on_outcome=[0.001, 0
>>> print(res_unobserved_range)
```

Automatically inferring effect strength parameters

Finally, DoWhy supports automatic selection of the effect strength parameters. This is based on an assumption that the effect of the unobserved confounder on treatment or outcome cannot be stronger than that of any observed confounder. That is, we have collected data at least for the most relevant confounder. If that is the case, then we can bound the range of effect_strength_on_treatment and effect_strength_on_outcome by the effect strength of observed confounders. There is an additional optional parameter signifying whether the effect strength of unobserved confounder should be as high as the highest observed, or a fraction of it. You can set it using the optional effect_fraction_on_treatment and effect_fraction_on_outcome parameters. By default, these two parameters are 1.

```
>>> res_unobserved_auto = model.refute_estimate(identified_estimand, estimate, met
>>>                                           confounders_effect_on_treatment="bin
>>> print(res_unobserved_auto)
```

< Previous

Skip to main content

Next