# Confounded or Causal Python Code Package v1.0

David Kaltenpoth

March 6, 2019

## 1 What is this?

This is a bunch of badly written code to attempt distinguishing between datasets which are causal and those which are confounded. I.e. given some data X and Y where it *looks* as if X causes Y, we want to know whether there might actually be a better explanation by assuming that they are both caused by some common latent variable Z. This package contains the source code for our recent publications on telling whether, given a sample over the joint distribution of a pair of continuous random variables X and Y for which it seems that X causes Y, the dependency is indeed causal, or whether it is confounded by a hidden variable Z. We do so by comparing a model under which the data has been generated in a directly causal manner from X to Y with a model where both X and Y are co-caused by a common confounder Z. This code entails the prototype implementation and is aimed at reproducing the results presented in [1,2] as well as can be used to infer confounding for your own pairs (X,Y).

Here, this is done by comparing a model where the relationship between X and Y is linear, and a model where the relationship between Z and (X, Y) is linear. The precise method and its theoretical grounding are described in our paper We Are Not Your Real Parents: Telling Causal from Confounded using MDL.

## 2 Environment and Data

The precise packages used for this project are listed in `requirements.txt`. All data is contained in the `data` directory or has to be created by running a script as explained below. The data in `data/gene_network` is taken from the DREAM3 challenge. The data in `data/tubingen-pairs` is taken from the Tübingen pairs database for cause-effect pairs.

# 3 How do I use this?

If you are looking at this code base, you a) want to reproduce my results, or b) run it on your own dataset, c) or you just want to know how the code works. If c) is the case, the only interesting file for you is most likely `model.py`. Everything else is just machinery to actually run the models on datasets and to generate plots from the outputs. If you're interested in a) or b), read on.

## 3.1 Reproducing results

Before being able to run any of the commands noted below, please first change the variables `THEANOPATH, PYTHONPATH, RUNPATH` in `scripts/paths.sh` to match your setup and change `RESULTS_PATH` to reflect where the results are stored. The output directory for the data used to generate tikz images is controlled by `TIKZ_PATH` in `config.py`.

### 3.1.1 Synthetic Experiments (Section 5.1)

To do this, we first need to generate the synthetic data used in the synthetic experiments before we can do anything. To do this, use either

```
python data_generation.py -data 'dr'
```

to generate the data for the decision rate plots, or use

```
python data_generation.py -data 'heat'
```

to generate data for the heatmap. Note that this may take a while because a lot of data is generated for all the different combinations of dimensionalities and generating distributions.

After we've done this, we can actually run the code on this data. The easiest way to do this is by running

```
bash scripts/synthetic_script.sh
```

or

```
bash scripts/heatmap_script.sh
```

for the decision rate plots and heatmap plots, respectively.

To generate the plots from this data, use

```
python tikz_data.py -data 'dr'
```

or

```
python tikz_data.py -data 'heat'
```

respectively. The data relevant to the comparisons with the methods from Janzing and Schölkopf are already included when running `dr_script.sh`.

#### 3.1.2 Genetic Networks (Section 5.2)

For the genetic networks, the data can be found in `data/gene_network/`. To run the script to generate the data for the plots, use

```
bash scripts/gn_script.sh
```

To generate the plots from this data, use

```
python tikz_data.py -data 'gene'
```

#### 3.1.3 Tübingen Pairs (Section 5.3)

For the Tübingen pairs, the data can be found in `data/tubingen-pairs/`. To generate the data for the plots, run

```
bash scripts/pair_script.sh
```

To generate the plots from this data, use

```
python tikz_data.py -data 'pair'
```

#### 3.1.4 Optical Data (Section 5.4)

The optical data can be found in
`other_code/confounder_detection_linear/data/optical_device/`.
To run the script to generate the data for the plots, simply run

```
bash scripts/optic_script.sh
```

To generate the plots from this data, use

```
python tikz_data.py -data 'optic'
```

### 3.1.5 Generating the plots

## 3.2 Running it on your own data

To run the method on any data, simply run

```
python run_on_any_data.py -in DATA_FILE -out OUTPUT_FILE -delim DELIMITER
```

If `OUTPUT_FILE` is missing, the outputs will be printed to the command line instead. If `DELIMITER` is not given, the data will be assumed to be separated by a single space. Further, the data for `Y` is assumed to be in the last column of the data.