Introduction to Casual Inference (DAG Framework) using pgmpy

Ankur Ankan

Postdoctoral Researcher
Radboud University, The Netherlands

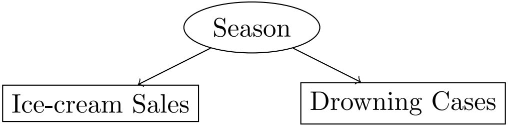
Predictive vs Causal Modelling

Predictive Modelling

- Exploits correlation among variables.
- Usually interested in predicting outcome variables.

Causal Modelling

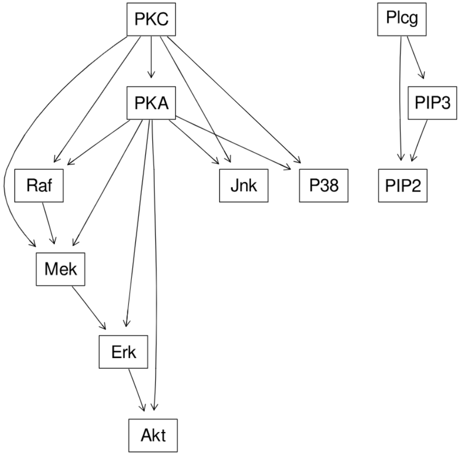
- Try to learn causal-effect relationships between variables.
- Interested in understanding the causal structure of the data generating mechanism and/or causal effect estimation.



Would an intervention on Ice-cream Sales affect Drowning Cases?

Example: Protein Signalling Causal Network [1]

- Models the concentration of signalling proteins in cells.
- Understanding the mechanism such as how the signals bring cellular responses.
- Insights on how signalling pathways are altered in diseases.
- Can be used to identify potential targets for therapeutic intervention.



[1] Sachs, Karen, et al. "Causal protein-signaling networks derived from multiparameter single-cell data." Science 308.5721 (2005): 523-529.

Examples

- **Epidemiology:** Analyzing data on how different treatment or exposures affect health outcomes.
- **Economics and Social Sciences:** Understanding impacts of policy interventions and help in designing potential policies.
- **Machine Learning:** For interpretability, feature selection, making models robust to out of distribution predictions.

Two Main Frameworks

Mathematical frameworks for causal inference:

- **Potential Outcomes Framework** (a.k.a. Rubin's Casual Model)
  - Usually interested only in estimating causal effect.
  - Provides statistical methods for estimating the causal effect. Examples are Propensity Score based methods, Doubly robust estimators, etc.

- **Directed Acyclic Graphs (DAGs)** / Structural Equation Models (SEMs)
  - Causal diagram, i.e., DAG at the core.
  - The DAG can be used to define estimators for causal effects of interest.
  - DAGs make modelling assumptions explicit.

Landscape of Causal Inference Python Packages

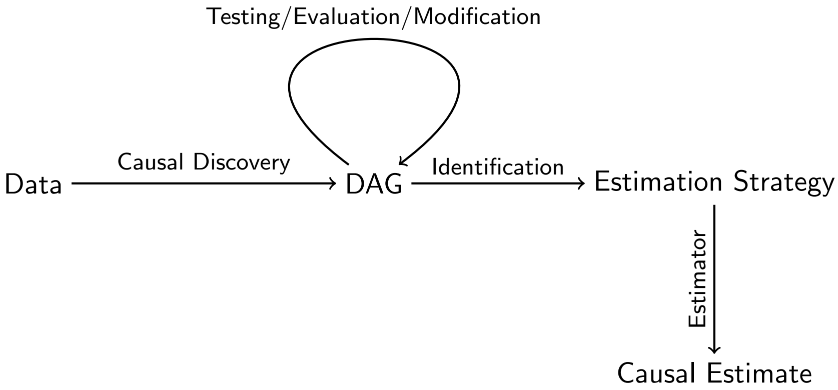
Potential Outcomes Frameworks

- Collection of estimation methods
- Doubly Robust Estimation
- Meta Learners: T/S/X-Learners.

DAG Frameworks

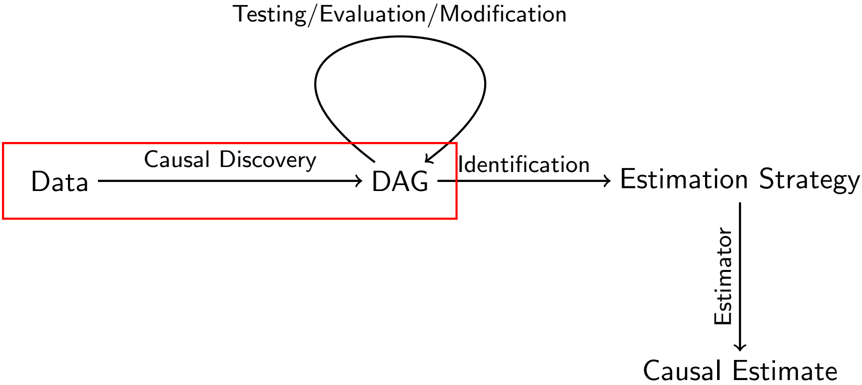
- pgmpy
- casualnex

A General Workflow in the DAG Framework



pgmpy provides functionality to perform each of these steps.

Causal Discovery



Casual Discovery: Learn the causal DAG from data.

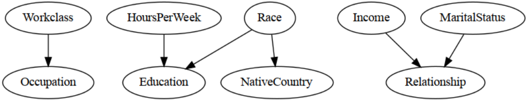
Causal Discovery: Automated Algorithms

- Many automated algorithms with nice asymptotic properties.
  - Constraint-based: PC, Fast Causal Inference.
  - Score-based: Greedy Equivalence Search, Hill-Climb Search.
  - Optimization-based: NoTears, DAGMA.
- In practice, output varies significantly depending on sample size, algorithm used, and their hyperparameters.
- With no standard evaluation method, difficult to decide the correct model.

Causal Discovery: Automated Algorithms

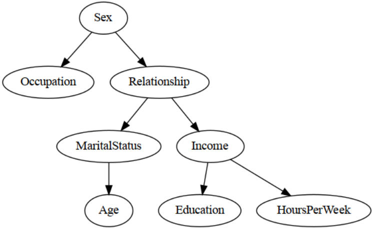
```python
from pgmpy.estimators import PC

est = PC(df)
dag = est.estimate(
    ci_test='chi_square')
```




```python
from pgmpy.estimators import HillClimbSearch

est = HillClimbSearch(df)
dag = est.estimate(scoring_method='bicscore')
```

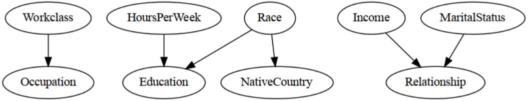


[1] Becker,Barry and Kohavi,Ronny. (1996). Adult. UCI Machine Learning Repository. https://doi.org/10.24432/C5XW20.

Causal Discovery: Automated Algorithms

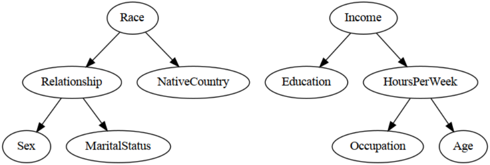
```python
from pgmpy.estimators import PC

est = PC(df)
dag = est.estimate(
    ci_test='chi_square')
```




```python
from pgmpy.estimators import PC

est = PC(df)
dag = est.estimate(
    ci_test='pillai')
```



Causal Discovery: Problems Automated Algorithms

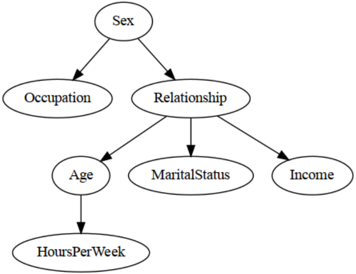
- Difficult to choose the best algorithm/model.
- In finite sample scenario, all algorithms make mistakes.
- Usually requires expert knowlege input.

pgmpy implements option to incorporate expert knowledge to these algorithms.

Causal Discovery: Expert Knowledge Integration

- Users can specify edges to blacklist/whitelist.
- The algorithm never adds blacklisted edges and only searches over whitelisted edges.

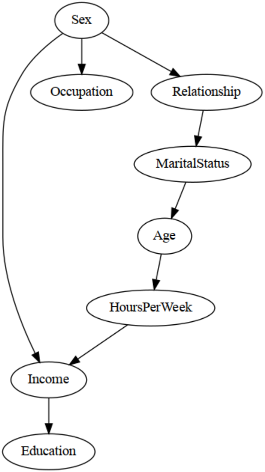
```
est = HillClimbSearch(df)
blist =[('Sex', 'MaritalStatus'),
        ('MaritalStatus', 'Income'),
        ('Income', 'Education'),
        ('MaritalStatus', 'Age'),
        ('Income', 'HoursPerWeek')]
dag = est.estimate(
        scoring_method="bicscore",
        black_list=blist)
```



Causal Discovery: Expert Knowledge Integration

- Score based methods perform local optimization.

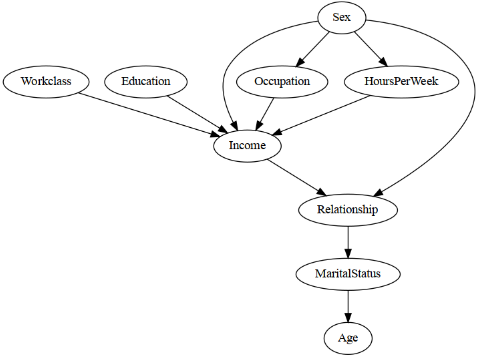
```python
from pgmpy.base import DAG
est = HillClimbSearch(df)
start_dag=DAG(
        [('Age', 'Income'),
         ('Age', 'Education'),
         ('Age', 'MaritalStatus'),
         ('Age', 'HoursPerWeek'),
         ('NativeCountry', 'Race'),
         ('HoursPerWeek', 'Income'),
         ('Education', 'Occupation'),
         ('Education', 'Income'),
         ('Education', 'Workclass'),
         ('Occupation', 'Income'),
         ('Sex', 'Income'),
         ('Relationship', 'MaritalStatus'),
        ])
dag_bic_start = est.estimate(
        scoring_method="bicscore",
        start_dag=start_dag)
```



Causal Discovery: Expert Knowledge Integration

- Allows to specify fixed edges that will be present in the final model.

```
est = HillClimbSearch(df)
dag_bic_fixed = est.estimate(
        scoring_method="bicscore",
        fixed_edges=[
            ('HoursPerWeek', 'Income'
            ('Education', 'Income'),
            ('Sex', 'Income'),
            ('Occupation', 'Income'),
            ('Workclass', 'Income')])
```



Causal Discovery: Expert-In-The-Loop

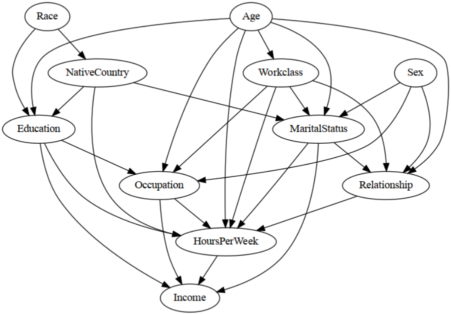
Since, causal discovery in practice is an iterative manual process, we designed an algorithm that works with interactive input from the user.

- Finds pair of variables whose observed correlation isn't explained by the model.
- Employs a ranking scheme to rank these violations.
- Iteratively adds and removes edges.
- Uses an expert to decide the direction of the edge.
- Greatly reduces the amount of manual intervention required.

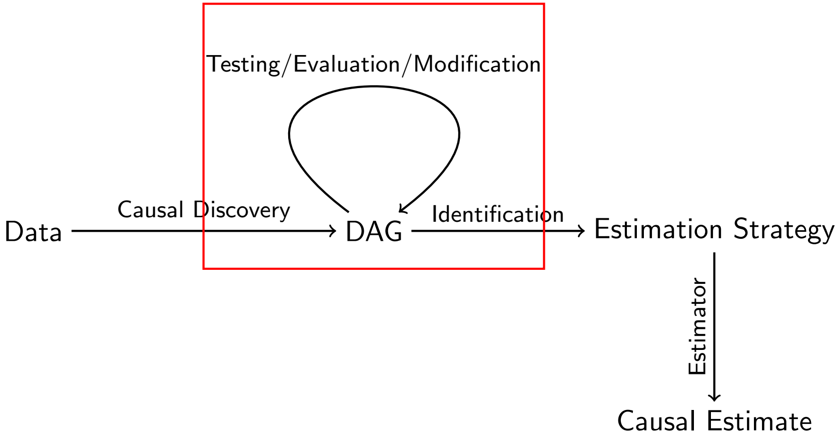
Causal Discovery: Expert-In-The-Loop

```python
from pgmpy.estimators import ExpertInLoop

descriptions = {
    "Age": "The age of a person",
    "Workclass": "The workplace where the person is "
        "employed such as Private industry, or
        "self employed",
    "Education": "The highest level of educatio
        "the person has finished",
    "MaritalStatus": "The marital status of the
    "Occupation": "The kind of job the person d
        "example, sales, craft repair, clerical
    "Relationship": "The relationship status of
    "Race": "The ethnicity of the person",
    "Sex": "The sex or gender of the person",
    "HoursPerWeek": "The number of hours per we
        "the person works",
    "NativeCountry": "The native country of the
    "Income": "The income i.e. amount of money
        "the person makes",
}
est_llm = ExpertInLoop(df)
dag_llm = est_llm.estimate(
    variable_descriptions=descriptions)
```



Model Evaluation



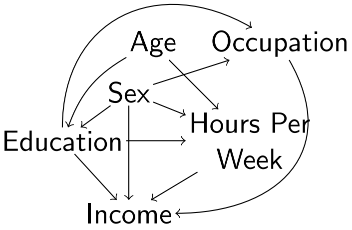
- As causal discovery algorithms make mistakes, important to test and modify learned models.

Model Evaluation

Similar to unsupervised learning, we do not have any ground truth data, so there is no straightforward way to evaluate models

pgmpy implements a few methods to test and compare models.

Model Evaluation: Implied Conditional Independences (CIs)



$$\Longrightarrow$$

$$\begin{aligned}
\text{Age} &\perp \text{Incm} | \text{Edct, Hrpw, Sex} \\
\text{Age} &\perp \text{Occp} | \text{Edct, Sex} \\
\text{Age} &\perp \text{Sex} \\
\text{Hrpw} &\perp \text{Occp} | \text{Edct, Sex}
\end{aligned}$$

- Each missing edge implies a Conditional Independence (CI).
- Statistical tests can be used to check whether they hold in data.
- Models can be modified based on these tests.

Model Evaluation: Implied Conditional Independences (CIs)

```
>>> from pgmpy.metrics import implied_cis
>>> from pgmpy.estimators.CITests import ci_pillai
>>> implied_cis(model=dag, data=df, ci_test=ci_pillai)

      u               v                            cond_vars              p-value
Relationship    Race                [Sex, Workclass, MaritalStatus, Age]  5.516662e-07
Relationship    NativeCountry       [Age, Sex, Workclass, MaritalStatus]  2.326022e-03
Workclass       Race                                                  []  2.469383e-01
Workclass       Income            [HoursPerWeek, Occupation, MaritalS...  9.043659e-01
Workclass       Education                                          [Age]  4.298111e-01
```

Model Evaluation: Fisher's C Test

- Combines the implied CI tests to summarize it into a single p-value.
- Using some significance threshold, we can decide whether the model fits well to the data.

```
>>> from pgmpy.metrics import fisher_c
>>> from pgmpy.estimators.CITests import chi_square

>>> fisher_c(dag_bic_fixed, df, ci_test=chi_square)
6.136802177536538e-11

>>> cancer_model = get_example_model('cancer')
>>> sim_df = cancer_model.simulate(int(1e4))
>>> fisher_c(cancer_model, sim_df, ci_test=chi_square)
0.946435029611621
```

Model Evaluation: Correlation Score

- Compares whether variables that are correlated in data are also correlated in the model.
- Computes the F1-score based on this comparison.

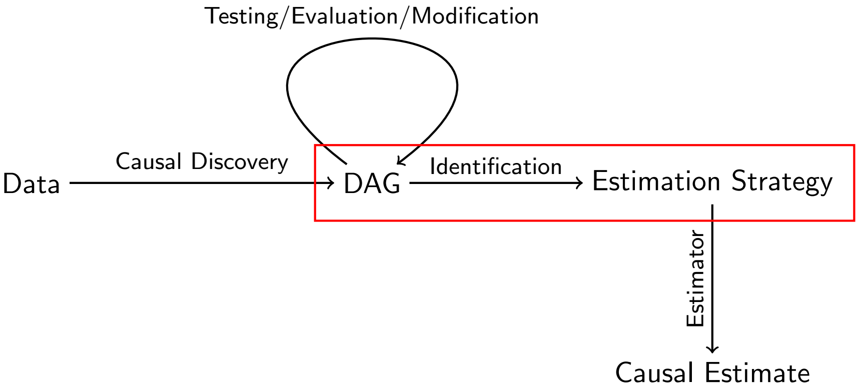
```
>>> from pgmpy.metrics import correlation_score
>>> correlation_score(dag_bic_start, df)
0.5128205128205128
>>> correlation_score(dag_bic_fixed, df)
0.5789473684210527
```

Model Evaluation: Structure Score Metrics

- Scores the network structure based on how well they fit to data.
- Useful for comparing multiple models.

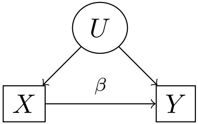
```
>>> from pgmpy.metrics import structure_score
>>> structure_score(model=dag_bic_start,
                    data=df,
                    scoring_method='bic')
-11472.356725048614
>>> structure_score(model=dag_bic_fixed,
                    data=df,
                    scoring_method='bic')
-50993.46795221821
```

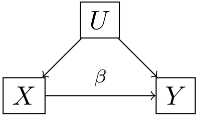
Identification



- After decided the DAG, we can start estimating causal effects.
- **Identification**: Is the causal effect of interest estimable?
- Everything is identified if all variables are observed.

Identification: do-calculus

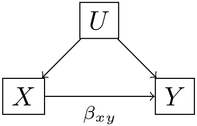




- Theoretically, do-calculus provides a complete solution to identification.
- do-calculus can give an estimand for every identified causal effect.
- However, no efficient algorithms to get these estimands using do-calculus.
- In practice, we rely on a set of simpler identification strategies that work in special cases.

Identification: Backdoor Criterion

- For cases when biasing paths can be blocked by conditioning.
- Finds adjustment variables to block confounding paths.



$$\beta_{xy} : Y \sim X + U$$

get_all_backdoor_adjustment_sets(*X*, *Y*) [source]

Returns a list of all adjustment sets per the back-door criterion.

A set of variables Z satisfies the back-door criterion relative to an ordered pair of variabies (Xi, Xj) in a DAG G if:

i. no node in Z is a descendant of Xi; and

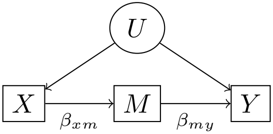
ii. Z blocks every path between Xi and Xj that contains an arrow into Xi.

Parameters:
- **X** (*str (variable name)*) – The cause/exposure variables.
- **Y** (*str (variable name)*) – The outcome variable.

Returns:
- **frozenset** (*A frozenset of frozensets*)

Identification: Front-door Criterion

- Used for scenarios when there are mediator variables.
- Double application of backdoor criterion.



$$\beta_{xm} : M \sim X$$
$$\beta_{my} : Y \sim M + X$$
$$\beta_{xy} = \beta_{xm}\beta_{my}$$

get_all_frontdoor_adjustment_sets(*X*, *Y*) [source]

Identify possible sets of variables, Z, which satisfy the front-door criterion relative to given X and Y.

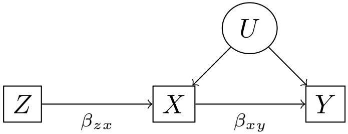
Z satisfies the front-door criterion if:

i. Z intercepts all directed paths from X to Y
ii. there is no backdoor path from X to Z
iii. all back-door paths from Z to Y are blocked by X

Parameters:
- **X** (*str (variable name)*) – The cause/exposure variables.
- **Y** (*str (variable name)*) – The outcome variable

Returns: **frozenset**

- When there are variables in the model that are correlated with the outcome variable only through the exposure variable.



$$\beta_{zx}\beta_{xy} : Y \sim Z$$
$$\beta_{zx} : X \sim Z$$
$$\beta_{xy} = \frac{\beta_{zx}\beta_{xy}}{\beta_{zx}}$$

get_ivs(*X, Y, scaling_indicators={}*) [source]

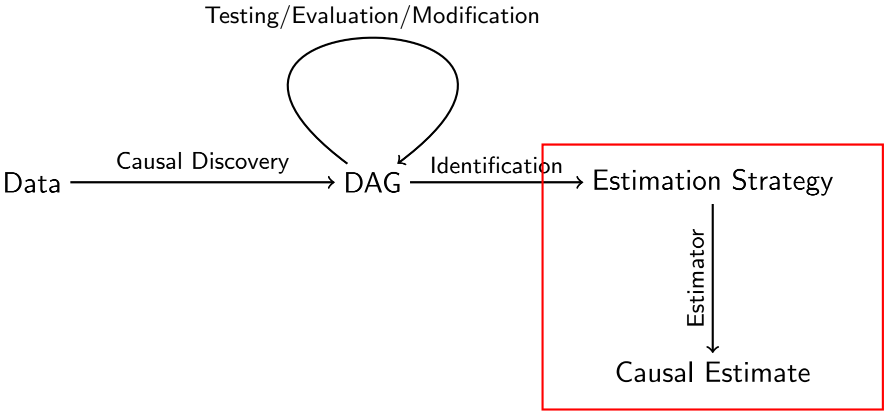
Returns the Instrumental variables(IVs) for the relation X -> Y

Parameters:
- **X** (*node*) – The variable name (observed or latent)
- **Y** (*node*) – The variable name (observed or latent)
- **scaling_indicators** (*dict (optional)*) – A dict representing which observed variable to use as scaling indicator for the latent variables. If not given the method automatically selects one of the measurement variables at random as the scaling indicator.

Returns: **set** – The set of Instrumental Variables for X -> Y.

Return type: {str}

Estimation



- Identification methods give the estimand.
- Any estimator can be used to make the estimates.
- Usually linear models are used for their interpretability.

Some other features.

Simulations

- Parameterized DAGs are generative models.
- Simulated data can be used to evaluate methods.
- Can be used for approximate inference.
- Helpful in explaining concepts like confounding, bias, etc.

```
>>> from pgmpy.utils import get_example_model
>>> sachs_model = get_example_model('sachs')
>>> sachs_model.simulate(int(1e4))

      PIP2   P38   Jnk   PKA  Plcg   Mek   Akt   PKC   Erk  PIP3   Raf
0      LOW   LOW   AVG   AVG   LOW   AVG   LOW   LOW   AVG  HIGH   LOW
1      LOW   LOW   AVG   LOW   LOW  HIGH  HIGH   LOW  HIGH   AVG   AVG
2      LOW   LOW   LOW   AVG   LOW   LOW   LOW   AVG   AVG  HIGH   LOW
3      LOW   LOW   LOW   AVG   LOW   AVG   AVG   AVG   AVG  HIGH   AVG
4      AVG  HIGH  HIGH   LOW   AVG   LOW   LOW   LOW   LOW  HIGH  HIGH
```

Simulations

```
simulate(n_samples=10,
         do=None,
         evidence=None,
         virtual_evidence=None,
         virtual_intervention=None,
         include_latents=False,
         partial_samples=None,
         seed=None,
         show_progress=True)
```

Extensibility

- Causal Inference is a very active field of research.
- pgmpy offers easy ways to extend/modify algorithms.
- Methods accept custom functions as argument.
- Base classes make it easier to implement new algorithms and can be plugged into other functionality.

```python
from pgmpy.estimators import PC

def random_ci(X, Y, Z, data, boolean=True):
    return np.random.choice([True, False])

est = PC(df)
est.estimate(ci_test=random_ci)
```

Future Plans

- Focus on implementing more practical methods such as bootstrapping, model comparison methods.
- Improve support for mixed data, i.e., combination of categorical, ordinal, and continuous variables.
- Add more commonly used causal discovery, and identification algorithms.

Thank you

: pgmpy/pgmpy

: ankurankan@gmail.com