DoWhy

With a focus on Causal Graphical Models (CGMs)

Shamanth Kuthpadi Seethakantha

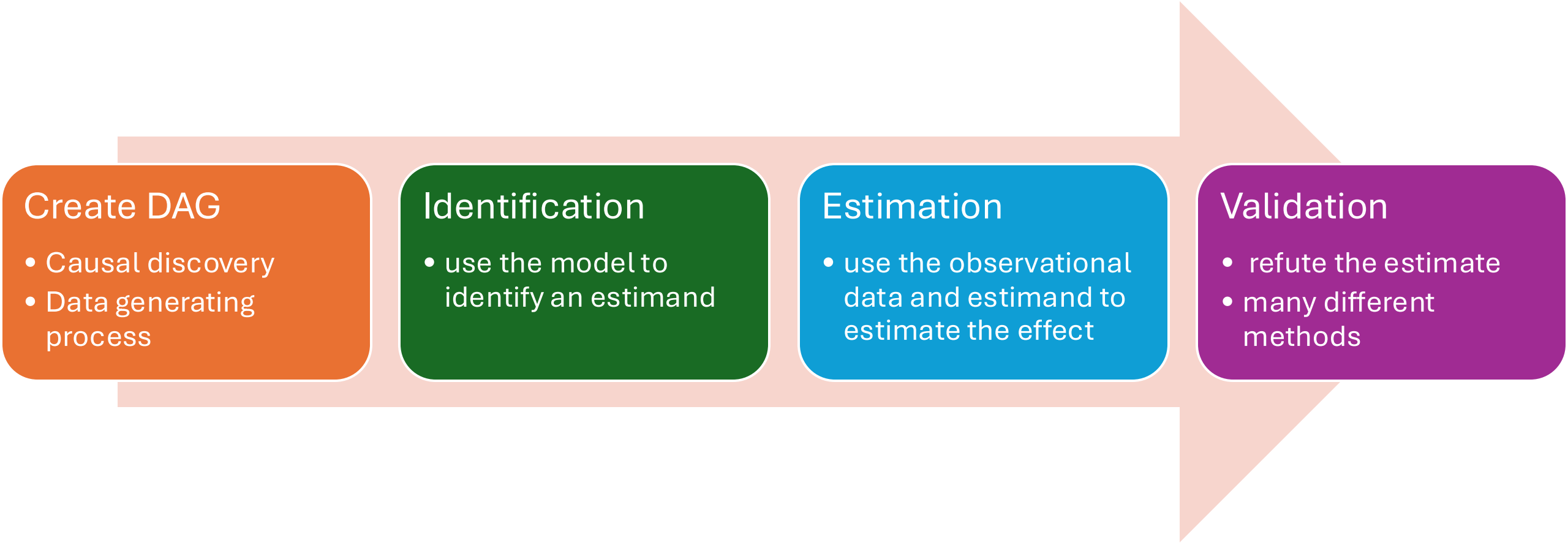
Overview

1. Motivate causal inference
2. Pipeline
3. Causal discovery
    1. Discuss why it's hard
    2. `causal-learn`
4. DoWhy
    1. Introduce capabilities
    2. Case study
5. Takeaways

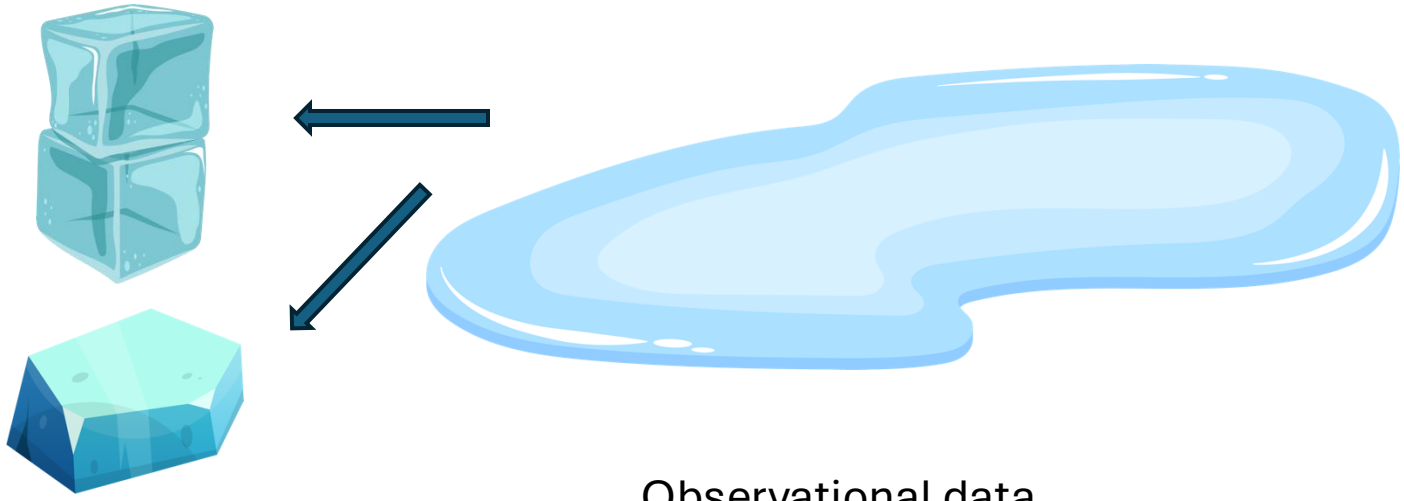
Motivation

- Predictive analytics/modeling
  - Find correlations between input features and output
  - This is good, but can we go from correlation to causation?
    - No (counfounding/latent variables)
- Why causation in the first place?
  - Intervention requires that we need to estimate the effect of changing the value of an input (treatment)
    - No data exists for this → estimation of a counterfactual
    - Most of the times, it is infeasible/unethical to perform an experiment
- Common causal queries raised in the industry
  - Will it work?
  - Why did it work?
  - What should we do?
  - what are the overall effects?
- Causal analysis!
  - causal analysis hinges critically on assumptions about the data-generating process

The pipeline



Causal Discovery



Observational data

Causal structures

Inverse problem

Main Question to think about:
How do you get the causal structure from data that you might collect from the world (e.g. observational data) ?

Answer:
This is a hard problem and requires some conceptual understanding of causal models. Thus, there are variations depending on the method you choose.

Markov Equivalence Class:
A set of directed acyclic graphical models that encode the same conditional independence rules.

causal-learn

- A variety of overarching methods and functionalities to obtain the causal structure given observational data
  - Constraint-based causal discovery methods
  - Score-based causal discovery methods
  - Constraint-functional-based causal discovery methods
  - .....
- Depending on the algorithm, you could get unsatisfactory/non-representative graphical models that could have undirected edges (e.g. PC & GES).
- Extremely simple to use but preprocessing can be tricky
- Need to have some knowledge of what the methods do
  - Mainly to understand which will be best suited for the data at hand

DoWhy, what is it?

- A standardized pipeline to validate and use assumptions about the data generating process
- Two key principles:
  - “making causal assumptions explicit” [1]
  - “testing robustness of the estimates to violations of those assumptions” [1]
- Identification → representation of assumptions formally
- Estimation → potential-outcome frameworks
- Validation via robustness and sensitivity checks
  - Check how estimate changes with updates to the assumptions

Visualization Tools

There are three main visualization tools:

1. `networkx`
2. `graphviz`
3. `matplotlib`

DoWhy allows you to visualize using these three main python packages.

However, formatting for use within the causal inference process can be difficult.

Code

Takeaways

Pros

- 3rd party library friendly
- Concise and interpretable
- Scalable
- Quickly becoming used in industry (moving from academia to actual corporate use)

Cons

- Still in development – open source work
- Dependency issues and documentation inconsistencies
- Lots of preprocessing is abstracted away

Works Cited

[1] Amit Sharma & Emre Kiciman, “DoWhy – A library for causal inference,” *Microsoft Research*, Aug. 21, 2018. https://www.microsoft.com/enus/research/blog/dowhy-a-library-for-causal-inference/ (accessed Jun. 11, 2024).

[2] Y. Zheng *et al.*, ‘Causal-learn: Causal discovery in python’, *Journal of Machine Learning Research*, vol. 25, no. 60, pp. 1–8, 2024.

DoWhy

TETRAD

causalnex

Libraries for Causal Modeling

Causal Discovery Toolbox

Shamanth Kuthpadi Seethakantha

Last time we talked about DoWhy, today we'll look at …

Overview

1. Tetrad
2. CausalML by *Uber*
3. causalnex
4. Causal Discovery Toolbox
5. Summary
6. Discussion
7. Quasi-experimental design

TETRAD

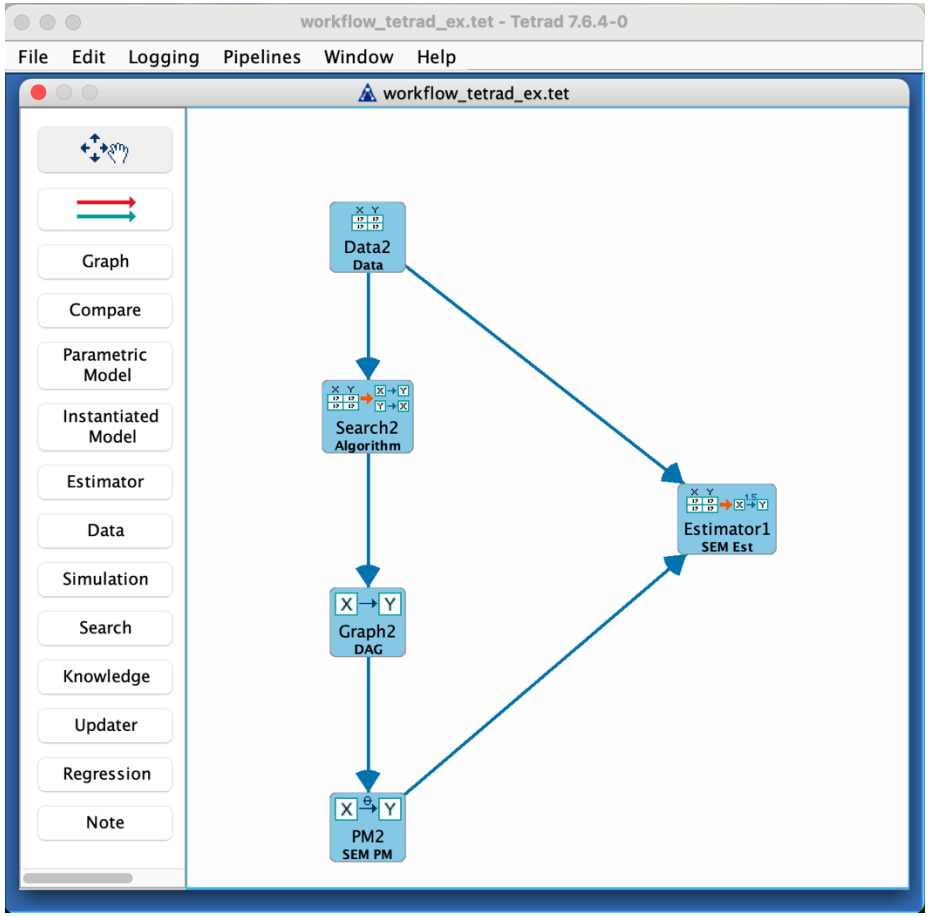
Pros

- Easy-to-use application (GUI)
- Detailed manual (GUI)
- Intuitive (GUI)
- Aesthetic doesn't take away from performance
  - Fast
  - Scalable
- Supports multiple data types
- Reproducible

Cons

- Lack of documentation for program-based interfaces
- Py-tetrad is slow
  - Speculating that it's because of jpype
- Expertise required for validation
- Simulated-data-centric

Demo!

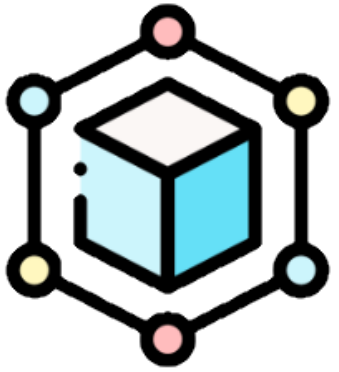


> "*Sessions can be saved at any stage and shared with others. This fact, along with the graphical nature of the toolbox, makes it very easy for collaborators to follow all analysis steps and to modify or add to these steps, making this an excellent tool for collaborations and for promoting reproducibility of any analysis*" [2].

[1] C. Glymour, K. Zhang, and P. Spirtes, "Review of Causal Discovery Methods Based on Graphical Models," *Frontiers in Genetics*, vol. 10, Jun. 2019, doi: https://doi.org/10.3389/fgene.2019.00524.

[2] J. Ramsey *et al.*, "TETRAD -A TOOLBOX FOR CAUSAL DISCOVERY TETRAD -A TOOLBOX FOR CAUSAL DISCOVERY." Accessed: Jun. 26, 2024. [Online]. Available: https://www.engr.colostate.edu/~iebert/PAPERS/CI2018_paper_35.pdf

CausalML *by Uber*



Pros

- Referenced methods
  - Most of the estimation techniques are sourced
- Easy-to-use
- Incredibly versatile estimation suite
- Decision guide!

Cons

- Focus is on inference – not discovery
- Doesn't include latest inference methods
- low code interpretability
- Can be computationally expensive at times

Demo!

"*One area of development is to further improve the computational efficiency for existing algorithms in the package*" [4] .

[4] H. Chen, Totte Harinen, Jeong Min Lee, M. Yung, and Z.-Y. Zhao, "CausalML: Python Package for Causal Machine Learning," Feb. 2020, doi: https://doi.org/10.48550/arxiv.2002.11631.

Causal Discovery Toolbox

Pros
- GPU hardware acceleration methods
- Utilizes 3rd party packages
  - This could mean we can get the best

Cons
- Hard to understand and find source code
- Lots of intricate code choice nuances
- Seems to have a steep learning curve

Note:
I couldn't configure the dependencies. There were too many system issues that had to do with R packages and its use within the Python env.

"*Cdt includes many state-of-the-art causal modeling algorithms (some of which are imported from R), that supports GPU hardware acceleration and automatic hardware detection*" [5].

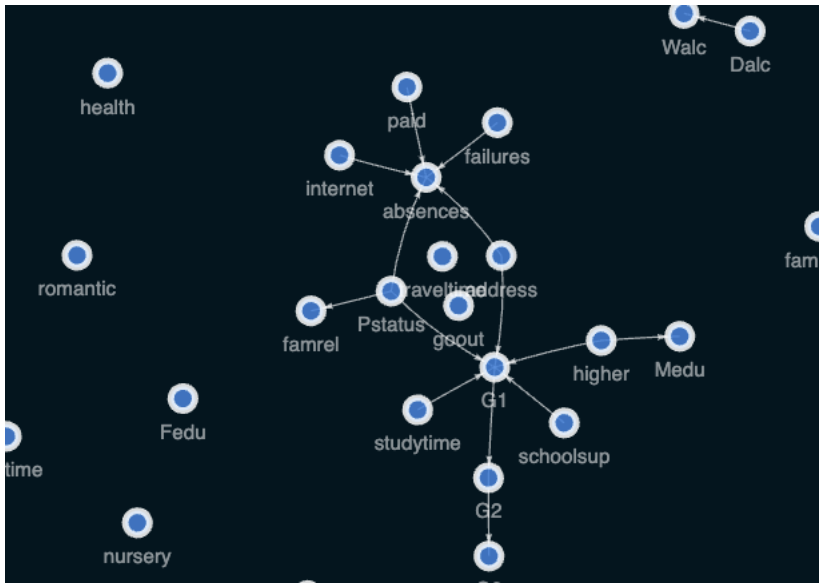
[5] D. Kalainathan and O. Goudet, "Causal Discovery Toolbox: Uncovering causal relationships in Python," *Journal of Machine Learning Research*, vol. 21, pp. 1–5, 2020, Accessed: Jun. 26, 2024. [Online]. Available: https://www.jmlr.org/papers/volume21/19-187/19-187.pdf

causalnex

Pros
- Easy to understand documentation
- can do both estimation and discovery
- Good visuals for intuition

Cons
- Doesn't seem to have most of the major estimation tools
- Looks relatively new in terms of development stage
- Steep learning curve
- Discretization of continuous values required

Demo!



"we adopted CausalNex to uncover the DGP and then incorporated such DGP into DoWhy to infer the influence of interventions upon the fire resistance" [6]

[6] M. Z. Naser and A. Ö. Çiftçioğlu, "Causal discovery and inference for evaluating fire resistance of structural members through causal learning and domain knowledge," *Structural Concrete*, vol. 24, no. 3, pp. 3314–3328, Jan. 2023, doi: https://doi.org/10.1002/suco.202200525.

Discovery and/or Inference	Discovery Inference	Discovery Inference	Inference	Discovery Inference	Discovery Inference
Structure Learning Method			N/A		
Library credibility					
Code quality	High	N/A	Medium	Low	Medium
Documentation quality	Medium	High	High	Medium	Low
Last update	July 1, 2024	June 25, 2024	May 2024	Feb. 2024	2022
Features	• Modular architecture • Validation via robustness checks and refutation	• 20 theoretically tested search algorithms • Allows for continuous and discrete variables	• Uplift model • real-world applications	• Discretization • Highly customizable visuals	• Mix of Python and R libraries • Research documented methods
Usability	Medium	High	High	Medium	Low
Advanced features		• Different interfaces to use suite	• Decision guide available		• GPU acceleration

What else?

- There are some up and coming really cool libraries:
  - CausalTune
  - Tetrad-Fx
- My suggestion would be to assemble the best libraries together
  - This has already been done in multiple case studies (https://arxiv.org/pdf/2204.05311)
  - We can utilize the best of each library
- If we want to stick to one library, I think DoWhy would be best
- Still in the process of learning the theory behind methods for causal estimation (very essential during the decision making process)
- Another major question to ask ourselves is:
  - Do we want to use a software that is stable or one that is constantly being updated?

Quasi-experimental design

- Intermediate points between observational and experimental studies

- Experimental design
  - Random assignment to control and treatment groups

- Quasi-experimental design
  - Some non-random methodology exists
  - Researcher doesn't have control over treatment
    - e.g smoker vs. non-smoker – what are the effects?
  - You don't really need control groups here
  - Why?
    - Ethical and practical reasons

Works Cited

[1] C. Glymour, K. Zhang, and P. Spirtes, "Review of Causal Discovery Methods Based on Graphical Models," *Frontiers in Genetics*, vol. 10, Jun. 2019, doi: https://doi.org/10.3389/fgene.2019.00524.

[2] J. Ramsey *et al.*, "TETRAD -A TOOLBOX FOR CAUSAL DISCOVERY TETRAD -A TOOLBOX FOR CAUSAL DISCOVERY." Accessed: Jun. 26, 2024. [Online]. Available: https://www.engr.colostate.edu/~iebert/PAPERS/CI2018_paper_35.pdf

[3] S. Athey and S. Wager, "Policy Learning with Observational Data," *arXiv (Cornell University)*, Jan. 2017, doi: https://doi.org/10.48550/arxiv.1702.02896.

[4] H. Chen, Totte Harinen, Jeong Min Lee, M. Yung, and Z.-Y. Zhao, "CausalML: Python Package for Causal Machine Learning," Feb. 2020, doi: https://doi.org/10.48550/arxiv.2002.11631.

[5] D. Kalainathan and O. Goudet, "Causal Discovery Toolbox: Uncovering causal relationships in Python," *Journal of Machine Learning Research*, vol. 21, pp. 1–5, 2020, Accessed: Jun. 26, 2024. [Online]. Available: https://www.jmlr.org/papers/volume21/19-187/19-187.pdf

[6] M. Z. Naser and A. Ö. Çiftçioğlu, "Causal discovery and inference for evaluating fire resistance of structural members through causal learning and domain knowledge," *Structural Concrete*, vol. 24, no. 3, pp. 3314–3328, Jan. 2023, doi: https://doi.org/10.1002/suco.202200525.

Updates from last time

- Was able to get a working code sample for…
  - Tetrad
  - CDT
- Updated the summary table – contains much more information now
- Looked at the graph APIs (to make sure all of them use the same graph encoding package)

TETRAD cont.

- Developed since the early 1990s
- Accessing the internal code is useful for several reasons
  - Java implementation can be a limitation to many
    - Python and R are the most common programming languages for data science related work
- JPype = “ready solution for accessing the underlying code of Tetrad from Python”
- Py-tetrad

DEMO!

[1] J. Ramsey and B. Andrews, “Py-Tetrad and RPy-Tetrad: A New Python Interface with R Support for Tetrad Causal Search,” *proceedings.mlr.press*, Nov. 20, 2023. https://proceedings.mlr.press/v223/ramsey23a (accessed Jul. 06, 2024).

Everything (almost) uses NetworkX!

dowhy.utils.networkx_plotting module

```
dowhy.utils.networkx_plotting.plot_causal_graph_networkx(causal_graph: Graph,
layout_prog: Optional[str] = None, causal_strengths: Optional[Dict[Tuple[Any,
Any], float]] = None, colors: Optional[Dict[Union[Any, Tuple[Any, Any]], str]] =
None, filename: Optional[str] = None, display_plot: bool = True,
label_wrap_length: int = 3, figure_size: Optional[Tuple[int, int]] = None) → None
```

[source]

causalnex.structure.StructureModel

class **causalnex.structure.StructureModel**(*incoming_graph_data=None, origin='unknown', **attr*) [source]

Bases: `networkx.classes.digraph.DiGraph`

Base class for structure models, which are an extension of `networkx.DiGraph`.

A `StructureModel` stores nodes and edges with optional data, or attributes.

Edges have one required attribute, "origin", which describes how the edge was created. Origin can be one of either unknown, learned, or expert.

StructureModel hold directed edges, describing a cause -> effect relationship. Cycles are permitted within a `StructureModel`.

Nodes can be arbitrary (hashable) Python objects with optional key/value attributes. By convention None is not used as a node.

Edges are represented as links between nodes with optional key/value attributes.

```
def write_gdot(gdot, probs, threshold=0, weight=1, length=1, power=1, hidden=lambda pair: False):
    for node in set([key[0] for key in probs] + [key[1] for key in probs]):
        gdot.node(node,
                  shape="circle",
                  fixedsize="true",
                  weight=f"{weight}",
                  style="filled",
                  color="lightgray",
                  stroke="transparent")

    for pair in probs:
        if hidden(pair): continue
        adj = round(sum([probs[pair][edge] for edge in probs[pair]]))
        if adj < threshold: continue

        for edge in probs[pair]:
            marks = ["none", "none"]
            prob = round(probs[pair][edge], 3)

            if edge[0] == "o": marks[0] = "odot"
            if edge[2] == "o": marks[1] = "odot"
            if edge[0] == "<": marks[0] = "empty"
            if edge[2] == ">": marks[1] = "empty"

            intensity = 1.0 - prob ** power
            alpha = hex(int(255 * intensity))[2:]
            if len(alpha) == 1: alpha = "0" + alpha
            if marks[0] == "empty" and marks[1] == "empty": color = "#ff" + 2 * alpha
            else: color = "#" + 2 * alpha + "ff"

            gdot.edge(pair[0], pair[1],
                      arrowhead=marks[1],
                      arrowtail=marks[0],
                      dir="both",
                      len=f"{length}",
                      color=color)

    return gdot
```

Unfortunately, Tetrad has an API doc for the methods used but not for the graphical visualizations.

`create_graph_from_data`(*data*) [source]

Run the GES algorithm.

Parameters: **data** (*pandas.DataFrame*) – DataFrame containing the data

Returns: Solution given by the GES algorithm.

Return type: networkx.DiGraph

Works Cited

[1] J. Ramsey and B. Andrews, “Py-Tetrad and RPy-Tetrad: A New Python Interface with R Support for Tetrad Causal Search,” *proceedings.mlr.press*, Nov. 20, 2023. https://proceedings.mlr.press/v223/ramsey23a (accessed Jul. 06, 2024).

DoWhy & causalnex

Shamanth Kuthpadi

I. Preliminaries

- Tested both DoWhy and Causalnex on a simple synthetic dataset to ensure that they operate as intended
- Moved on to use the Sachs dataset
  - Complex enough
  - Has both a discrete and continuous version
    - This is important because causalnex requires that the data be discrete
    - I tried discretizing using the method described by them – however this requires expertise with data distribution and collection
  - Has a ground-truth, although I will discuss how to evaluate a given model with and without knowledge of the DGP

II. Basic Evaluation Metrics

Structural Hamming Distance (SHD)

$$SHD = A + D + I$$

- A = # of edges to add
- D = # of edges to delete
- I = # of edges to reorient

Think of this as the number of edits to go from a predicted DAG to the actual DAG

Forbenius Norm

$$\sqrt{tr\left\{(B_{true} - B_{pred})^T (B_{true} - B_{pred})\right\}}$$

- B_{true} = adjacency matrix of actual DGP
- B_{pred} = adjacency matrix of predicted DGP

This is the square root of the trace of the matrix that is produced by the dotting the difference of the adjacency matrices by its transpose.

Latency

Time needed to run a structure learning algorithm

[1] L. Cheng, R. Guo, R. Moraffah, P. Sheth, K. S. Candan, and H. Liu, "Evaluation Methods and Measures for Causal Learning Algorithms," Feb. 06, 2022, *arXiv*: arXiv:2202.02896. Accessed: Jul. 16, 2024. [Online]. Available: http://arxiv.org/abs/2202.02896

III. Refuting Causal Graphs

- Local Markov conditions / conditional independence rules informed by our causal graph
- Measure whether the LCMs implied by our graph satisfy the data
  - Compares the number of LCMs violated by our graph with the number of LCMs violated by a randomly permuted set of graphs
  - What does the p-value mean?
- Check whether the graph is falsifiable
  - Assuming our graph is correct, how many other permuted graphs share the same number of LCM violations (0 violations)
  - What does the p-value mean?

IV. Refuting Causal Estimates

- Three main ways I will be presenting:
  - Placebo treatment refuter
  - Random common cause
  - Data subsample refuter
- The purpose of these refutation methods is to try and see if the causal estimate we have calculated is a useful one
- If not, there are many steps in the pipeline that could have gone wrong that we should check

V. Sachs Dataset – Analysis

- Quantitative measurements about the expression levels of 11 phosphorylated protein and phospholipids in the human primary T cell.
- Why Sachs Dataset?
  - Real complex data
  - Biological data
  - Ground-truth available
  - Two settings available
    - Discrete
    - Continuous

Code

6.3 Causal Discovery Algorithms

While the main scope of this work is to evaluate user-given graphs, we conduct additional experiments with $\hat{\mathcal{G}}$ inferred via causal discovery. On the Sachs et al. [1] data we find that graphs inferred by NOTEARS and LiNGAM are not significantly better than random, whereas graphs inferred by CAM are not falsified using our metric at $\alpha = 1\%$ (Tab. 2). Furthermore, a ranking based on our metric is in accordance with an SHD ranking (NOTEARS > LiNGAM > CAM). Further experimental results on synthetic data are given in Sec. C.3.

Table 2: p_{LMC} and SHD for graphs inferred by causal discovery on the Sachs et al. [1] data.

	NOTEARS	LiNGAM	CAM
$p_{\text{LMC}}^{\hat{\mathcal{G}},\mathcal{D}}$	0.77 ± 0.19	0.35 ± 0.33	0.0091 ± 0.0110
$\text{SHD}/\lvert\mathcal{E}\rvert$	2.20 ± 0.23	1.90 ± 0.13	$1.50 \quad \pm 0.11$

Remember that at the end of the day, NOTEARS is a causal discovery method. Causalnex itself is a tool that provides an interface to use this algorithm – however DoWhy is able to host algorithms for causal discovery while also providing excellent evaluation methodologies.

[2]
E. Eulig, A. Mastakouri, P. Blöbaum, M. Hardt, and D. Janzing, “Toward Falsifying Causal Graphs Using a Permutation-Based Test.” Accessed: Jul. 17, 2024. [Online]. Available: https://arxiv.org/pdf/2305.09565

Refutations in DoWhy

Main ways of refuting causal graphs in DoWhy

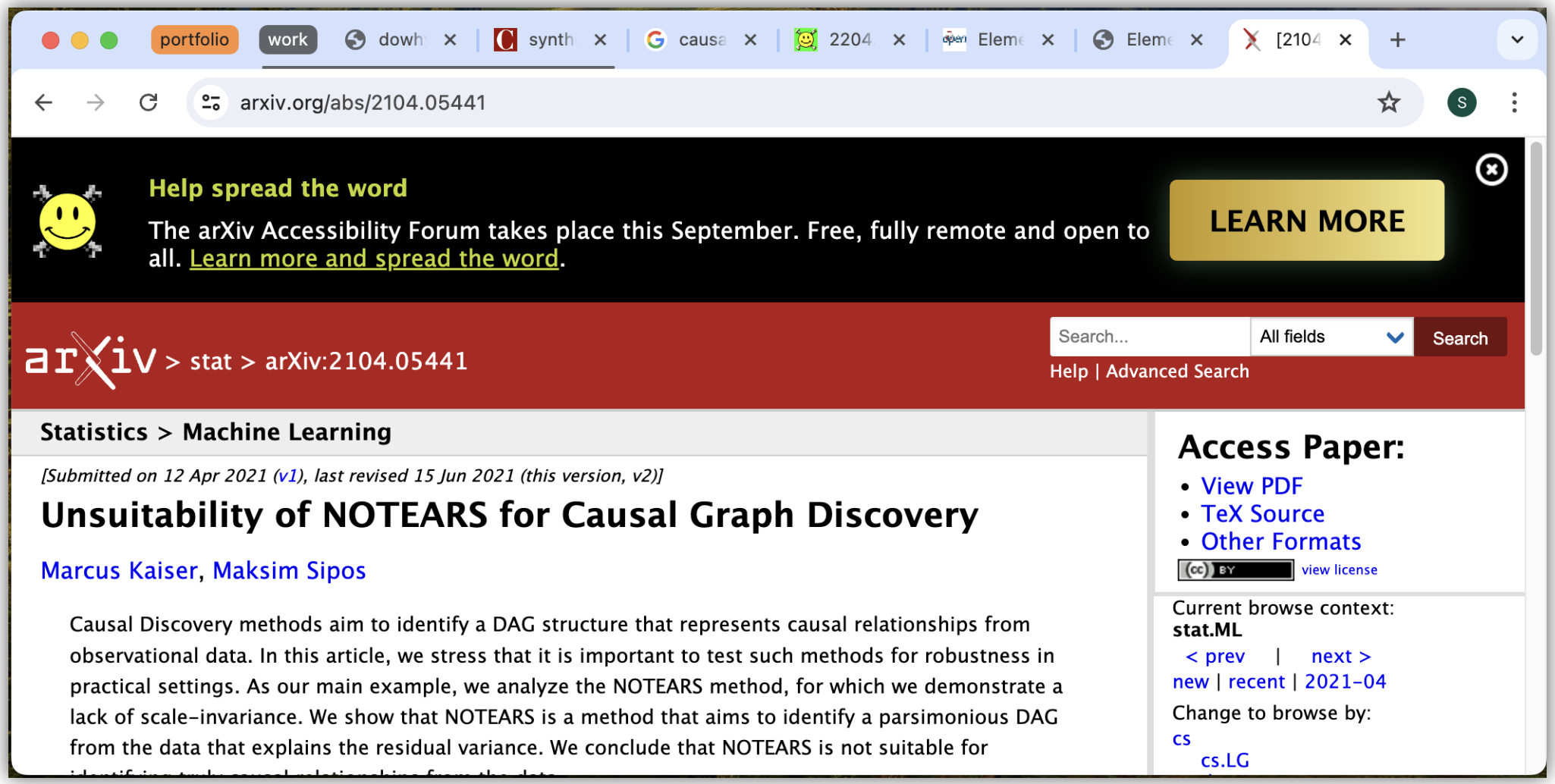
- Last time, we took a brief look at refutations in DoWhy
- Today, we will …
  - Discuss whether the edge suggestions are to be used in conjunction or separately
  - Talk a little bit about causal minimality
  - Discover a few more methods that can be used to evaluate causal graphs

Causal Minimality

- To ask if our graph G is causally minimal is to simply check if there are any redundancies in our graph.
- We have our dataset, and (think theoretical) imagine we have our probability distribution P_x from this data.
- Assume P_x is Markovian with respect to G.
  - A node x_i is independent of all its non-descendants given its parents.
- For each node consider its parents, is a particular parent necessary given all other parents?
- U = set of parents
- Each parent in U → Y
- Y really needed U/Y

Evaluating causal graphs – functionalities

- `plot_local_insights`
  - Visualize our suggestions in a neat and graphical way
- `validate_cm`
  - Tests the causal minimality of our graph
- `validate_lmc`
  - Validates the local Markov conditions for each node and returns the number of violations
  - **Note:** I still don't know why this would be different from the falsify_graph results
- `validate_pd`
  - Validates the pairwise dependencies. Check if each node is statistically dependent of its ancestors
- `validate_tpa`
  - Way to check if two graphs lie in the same Markov equivalence class by aggregating the number of violated d-separation induced independencies



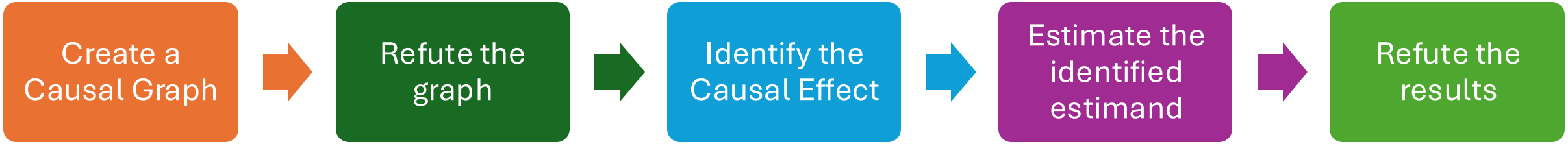
"identifying parsimonious structures that explain the data is not equivalent to making statements about causality in the data generating process"

[1]
M. Kaiser and M. Sipos, "Unsuitability of NOTEARS for Causal Graph Discovery," *arXiv.org*, Jun. 15, 2021. https://arxiv.org/abs/2104.05441 (accessed Jul. 23, 2024).

Revisiting the Workflow

Shamanth Kuthpadi

The Pipeline



Refutations

- Causal tasks require the use of interventional data
  - Data distribution cannot be observed to perform typical ML validation methods (e.g cross-validation)
  - We need to use something called *refutations*
- Two stages
  1. After modeling the causal graph
  2. After completing the analysis for the task

1. Graph refutations
   - Test whether the assumptions implied by a causal graph are valid
   - All downstream analysis depends on the graph
2. Estimate refutations
   - Conducted after our analysis returns a causal estimate
   - Tests whether our estimate is a robust one (how will it respond to violation of assumptions or special data)
   - Can help when choosing between candidate models

Refuting a Causal Graph

Remember that we have already discussed the various estimation refutation methods

- ***Main Idea:*** Is our graph consistent with the available data?
- Each graph implies a set of conditional independence statements over the nodes (aka local Markov conditions)
- Independence tests
  - Assume that our graph has the relation $X \rightarrow Y \rightarrow Z$. This implies that X is conditionally independent of Z given Y.
  - Kernel dependence measure be used to check this assumption
- However, for large graphs, many such assumptions need to be checked.
  - It is inefficient to perform individual independence tests to validate assumptions with our data
  - This is where falsify_graph comes into play

Causal Tasks

Today, we will focus on effect estimation

- Effect estimation
  - If we change X, how much will it cause Y to change?
- Attribution
  - Why did an event happen?
  - How to explain an outcome?
  - Which of my variables caused the anomaly?
- Counterfactual estimation
  - What if X had been changed to a different value than its observed one?
- Prediction
  - Given a new input with different input features, what will be the output?

Effect Estimation

$$E[Y|do(A)]$$

= average causal effect

= expected change in Y due to a change in A

$$E[Y|do(A), X]$$

= conditional average causal effect

= expected change in Y due to a change in A (given the covariates X)

1. Model a causal inference problem using assumptions.
   - model (*CausalModel* or *graph*)
2. Refute the causal graph once discovered or given
   - falsify_graph
1. Identify an expression for the causal effect under these assumptions ("causal estimand").
   - identify (*identify_effect*)
2. Estimate the expression using statistical methods such as matching or instrumental variables.
   - estimate (*estimate_effect*)
3. Finally, verify the validity of the estimate using a variety of robustness checks.
   - refute (*refute_estimate*)

Answering Causal Questions w/ DoWhy

Today’s focus: attribution queries

First, knowledge injection

- Only a few of the algorithms in causal-learn have a feature that allows to inject background knowledge before search
  - Most of the algorithms, however, do not provide this convenience
  - You would need to manually edit the graph
- Major (existing) concerns with inserting background knowledge in causal-learn
- DoWhy has a way to accept a causal graph as input, however injection of background knowledge is left up to the backend causal discovery package.

Interventions vs. Counterfactuals

- Interventions look into the future to try and predict the outcome for a given treatment
- Counterfactuals look back into an alternative past where the treatment is different
- Both are similar but their difference is crucial to point out

```
gcm.counterfactual_samples
gcm.interventional_samples
```

Graphical Causal Models (GCMs)

- Causal graph doesn’t have information about the underlying data-generating process.
  - We need to also understand the mechanistic relationship (causal mechanism) between nodes in the graph
- GCM = direct acyclic graph + causal mechanisms
- Types of GCM:
  1. Probabilistic Causal Model
  2. Structural Causal Model
  3. **Invertible Structural Causal Model**

Invertible Structural Causal Model (ISCM)

- ISCM is simply an SCM that requires the underlying functional causal model to be invertible with respect to noise
- Structural Causal Model (SCM):
  - Limits mechanisms to a deterministic functional causal model of parents and unobserved noise
  - $X_i = f_i(PA_i, N_i)$

- Additive Noise Models (continuous data) of the form $X_i = f_i(PA_i) + N_i$, where f_i can be any kind of regression function (e.g., from scikit-learn) and the noise N_i is unobserved noise. When fitting an ANM, this then boils down to fitting the f_i model (e.g., via least squares) and fitting N_i based on the residuals $N_i = X_i - f_i(PA_i)$. As mentioned throughout the user guide, ANMs are the most commonly used types of causal models due to their simplicity and ability to answer counterfactual questions.
- Post-nonlinear models (continuous data) of the form $X_i = g_i(f_i(PA_i) + N_i)$, where g_i is assumed to be invertible. These are a generalization of ANMs, allowing more complex relationships between N_i and PA_i.
- Discrete Additive Noise Models (discrete data), which have a similar definition as ANMs but are restricted to discrete values.
- Classifier-based Functional Causal Models (categorical data) of the form $X_i = f_i(PA_i, N_i)$, which cannot be used for rung 3 queries, since f_i is typically not invertible here with respect to N_i, but can be used for algorithms relying only on interventional queries (rung 2). Here, f_i can be based on any classification model (e.g., from scikit-learn) and N_i is by definition a uniform distribution on [0, 1] used to sample from the conditional class probabilities.

Anomaly Attribution

How much did each of the upstream nodes and the target node contribute to the observed anomaly?

- To answer this question we should use the approach suggested in this paper (https://proceedings.mlr.press/v162/budhathoki22a/budhathoki22a.pdf)
- Information-theoretic outlier detection using `Isolated Forest` from sci-kit learn
- We will take a look at the Sachs dataset once again and ask the question:
  - Why is the value of p38 incredibly high in one of the data instances within the dataset?
  - In particular, what attribute is the root-cause of this anomaly?
- Note that we will use the continuous version of this dataset. I found that with categorical variables (like the ones in the Lung Cancer dataset) posed a major concern:
  - What node will I choose as the node causing the outlier?
  - With continuous/discrete numerical values I could use typical statistical techniques.

Progress Update

1. Finished creating a first draft for the pagination table (definitely needs an upgrade still)
2. Finished an initial draft for the pipeline automation – *effect estimation*
3. Still need to create instances of the tables pertaining to various causal tasks associated with each dataset
   - This means 4 * 2 = 8 tables

Pipeline Automation

Effect Estimation

Causal Search Algorithm

- Causal search algorithms are tools
  - Some algorithms might be useful in one situation but no so much in others
- We, as users of these algorithms, must verify that assumptions and preconditions are applicable to a given dataset
- Causal search algorithms assume that variables are “semantically independent”
  - Independently manipulable (can’t have collinearity) [linear algebra]
  - Must remove redundant variables

Causal Search Algorithm (cont.)

- **Constraint-based search algorithm**
  - Aim to discover a Markov equivalence class found in the data
    - Set of causal graphs that imply the same conditional independence statements
- **Score-based algorithm**
  - Useful for datasets with small sample sizes
  - Compare a set of models based on some measure of model fit
- **Algos w/ semi-parametric assumptions**
  - Useful when you want to learn causal relationships in more detail
  - Typically need datasets with large sample size to get an accurate model

The main point here is that, choosing a causal search algorithm really depends on many factors that consider the **goal** and **dataset** at hand. It would be **unwise to automate** this feature.

Other Updates

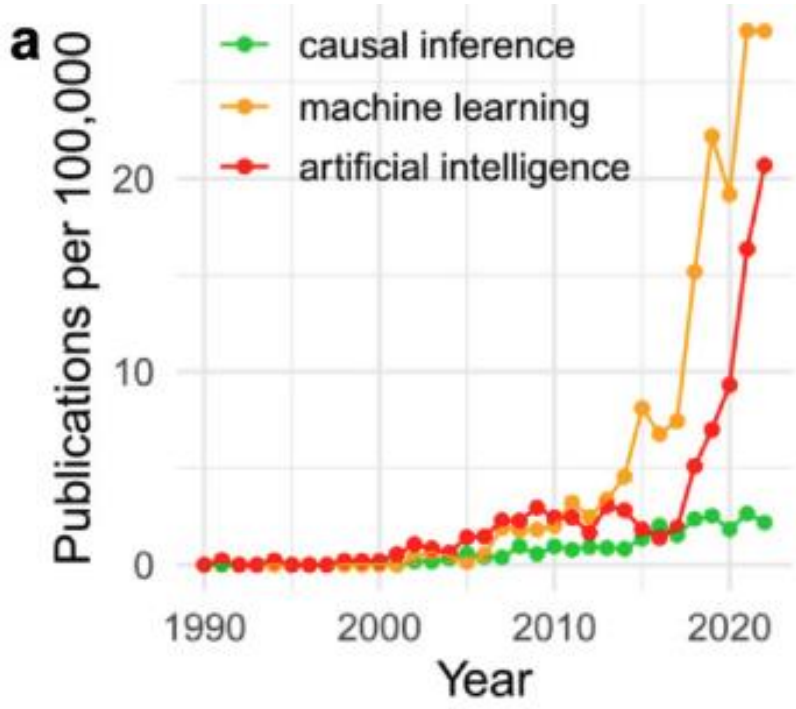
- I have logic set up for when the causal estimate is to be rejected.
  - Go back to the start of the pipeline and find a better causal model
- Still need to implement logic for when the graph is to be rejected based upon the falsify_graph module.
  - Should I first apply the suggestions and then check for significant p-val?
- Testing is difficult and is proving to be the most time consuming out of all this
  - I haven't yet written any formal tests
- There is a new and upcoming automation tool called "CausalTune" within the PyWhy environment.
  - However, it is not yet usable from what I saw
  - Recommended by one of the maintainers of DoWhy

Pipeline Automation (cont.)

Effect Estimation

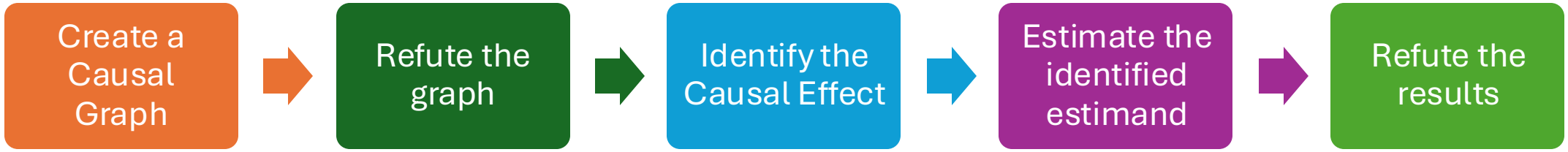
Causal Inference and Drug Discovery (etc.)

- Leveraging causal models can foster:
  - Drug repurposing
  - “Developing new classes of chemical matter” [1]
  - Banning the use of a drug
- Refinement: Validating our estimate through refutation analyses alone is not our main goal here (you can imagine why)
  - Creating and choosing between multiple candidate models
  - Understanding the impacts of adding new variables
  - Identifying possible latent variables and understanding their role in our model
- Causal reasoning would be useful in a wide array of applications including translation sciences and multiomic data.



Create a Causal Graph → Identify the Causal Effect → Estimate the identified estimand → Refute the results → Refinement

Reminders



Our Aim:

Automate the pipeline for causal estimation where we are trying to estimate the effect of a treatment on an outcome.

1. Model a causal inference problem using assumptions.
   model (*CausalModel* or *graph*)
2. Refute the causal graph once discovered or given
   falsify_graph
3. Identify an expression for the causal effect under these assumptions ("causal estimand").
   identify (*identify_effect*)
4. Estimate the expression using statistical methods such as matching or instrumental variables.
   estimate (*estimate_effect*)
5. Finally, verify the validity of the estimate using a variety of robustness checks.
   refute (*refute_estimate*)

Updates

1. Created a brand new effect estimation model.
   - Previous model wasn't unitarily tested
   - Didn't allow for prior knowledge
   - Was more geared towards being a super-learner rather than a working prototype
2. Tested and completed the causal discovery step of the pipeline
   - find_causal_graph is a definition that allows a user to input their own prior knowledge about the data into the graph that will be discovered
   - So far, the definition only supports the following algorithms: PC, GES, ICA LiNGAM. However, in my opinion, this list can be easily be expanded.
3. Revisited testing and I [think] have a formal way to test the definitions

Works Cited

[1] T. Michoel and Jitao David Zhang, “Causal inference in drug discovery and development,” *Drug Discovery Today*, vol. 28, no. 10, pp. 103737–103737, Oct. 2023, doi: https://doi.org/10.1016/j.drudis.2023.103737