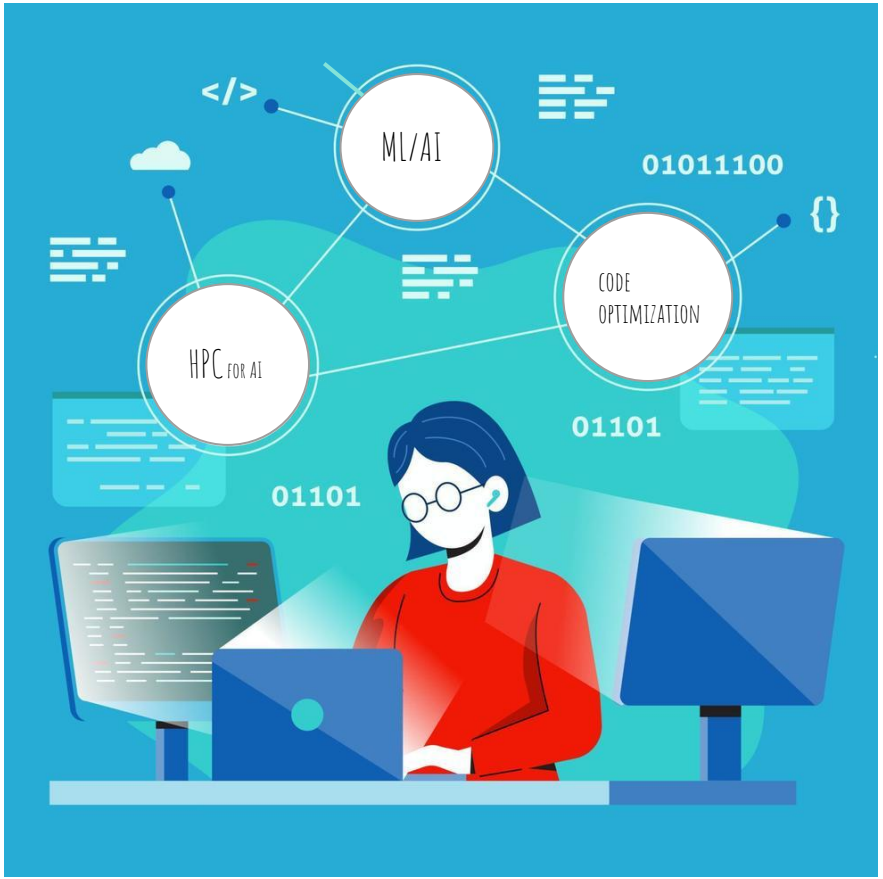
# Best practices for AI on HPC infrastructures

Serafina Di Gioia
Postdoctoral researcher in ML @ICTP

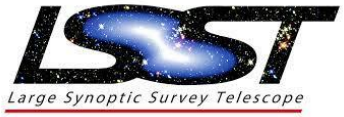
# What I do @ ICTP

- collaboration with ICTP scientists for optimization and development of ML pipelines
- testing/integration of latest technologies for accelerating ML pipelines
- research on ML algorithms
- collaboration with ML developers and scientists  at other research institutions  (e.g. SISSA)
- co-supervision of MHPC/PhD students
- coording GComp activities on AI
- ML course @ MHPC



# Why HPC for AI?

- To analyze Big data (e.g. in Astrophysics, Chemistry, Biology, Medicine) we need efficient and solid ML pipelines, only achievable thanks to modern cloud/HPC infrastructure and efficient code
- exploiting all the potential of GPUs is not possible without HPC knowledge
- larger ML pipelines a collaborative fast development and efficient tools for debugging




astrophysics

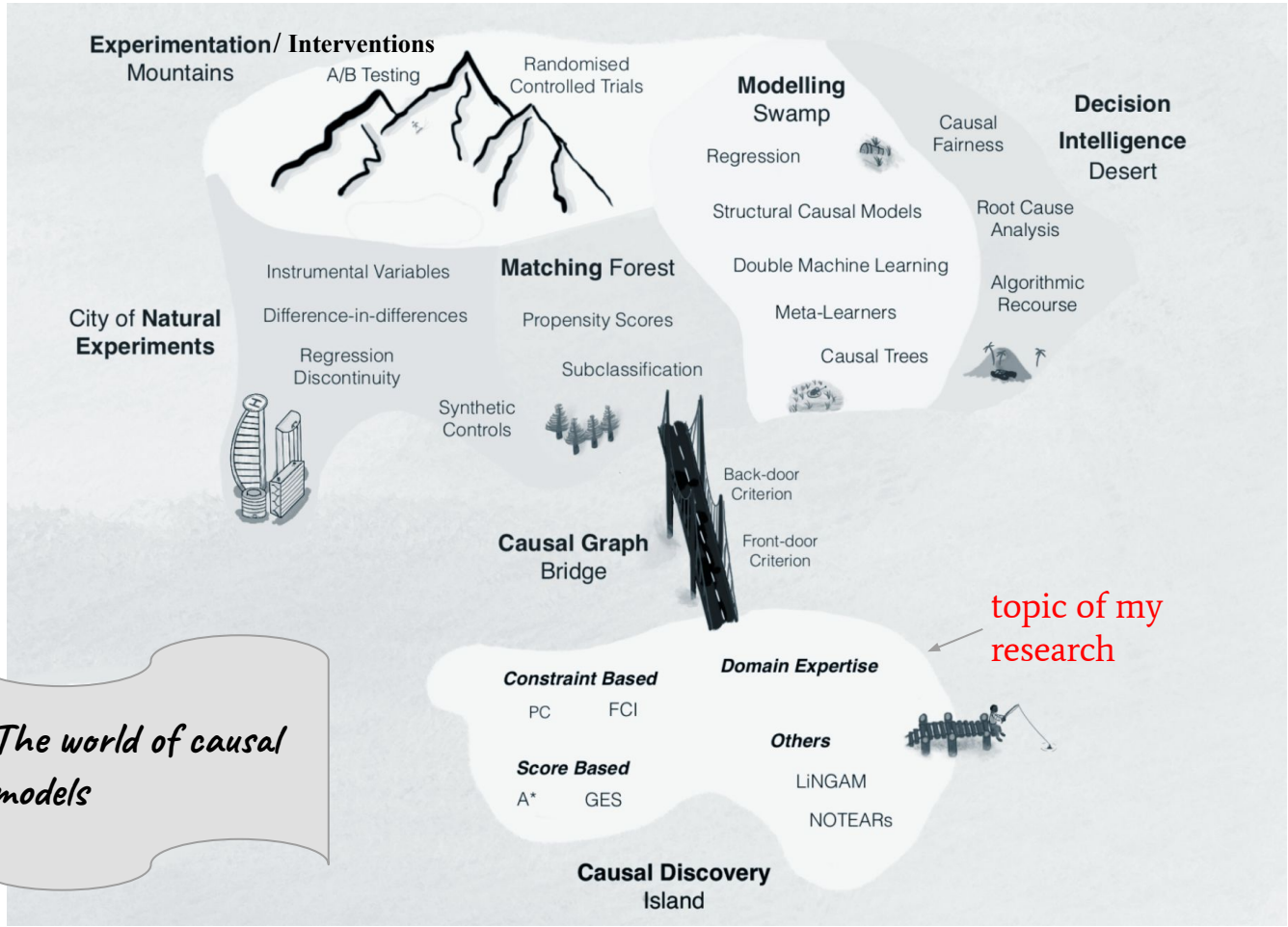
# My main projects

- Parallel Algorithms for Causal discovery
- Optimization and deployment of DL Emulators for Climate
- characterization of transformers for Molecular Chemistry

# Tools for causal discovery

- **THEORY**
  Probabilistic graphical models + causal inference

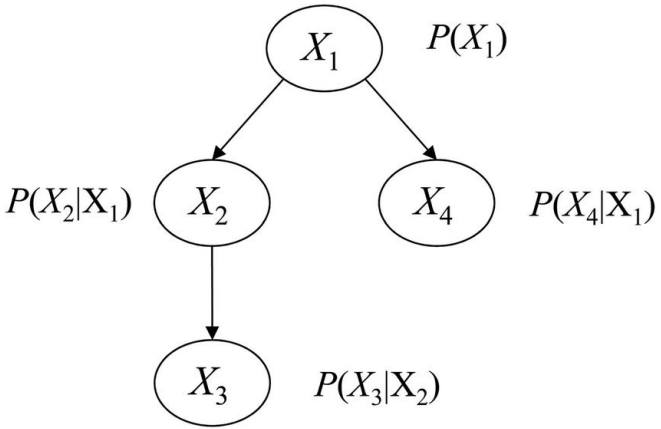
- **COMPUTATIONAL METHODS:**
  causal structure learning algorithms


Experimentation/ Interventions
Mountains
A/B Testing
Randomised
Controlled Trials
Modelling
Swamp
Causal
Fairness
Decision
Intelligence
Desert
Regression
Structural Causal Models
Root Cause
Analysis
Instrumental Variables
Matching Forest
Double Machine Learning
Algorithmic
Recourse
City of Natural
Experiments
Difference-in-differences
Propensity Scores
Meta-Learners
Regression
Discontinuity
Subclassification
Causal Trees
Synthetic
Controls
Back-door
Criterion
Causal Graph
Bridge
Front-door
Criterion
topic of my
research
Constraint Based
PC
FCI
Domain Expertise
Others
Score Based
A*
GES
LiNGAM
NOTEARs
Causal Discovery
Island
The world of causal
models

# Major ingredient of PGMs: Bayesian networks (BN)

Bayesian networks have 2 components:

- directed acyclic graph (DAG)




set of edges/arcs

- Joint probability distribution

$$\mathrm{P}(\mathbf{X}) = \prod_{i=1}^{N} \mathrm{P}(X_i \mid \Pi_{X_i}; \Theta_{X_i})$$

# The world of graphs

We can indicate a graph with the tuple

$$\mathcal{G} = (\mathbf{V}, A)$$

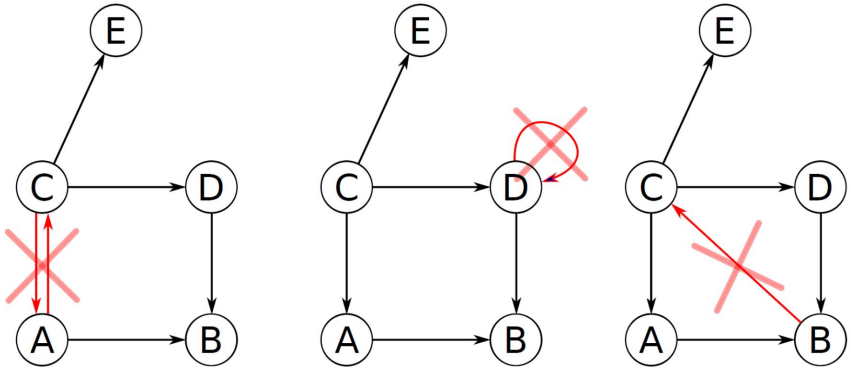
$V = \{v_1 \ldots v_N\}$ set of nodes

$a_{ij} = (v_i, v_j)$ set of arcs

If a graph satisfies:

1. **no undirected edges**
2. **no loop**
3. **no cycle**

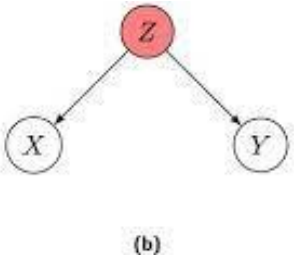
**is called a Directed Acyclic graph (DAG)**



# Building blocks of causal BNs

## Triplets

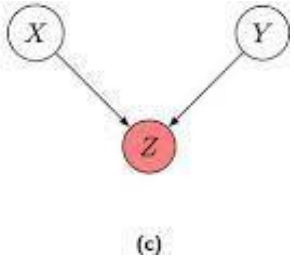

chain




fork

$X \perp\!\!\!\perp Y \mid Z$

$X \not\perp\!\!\!\perp Y$

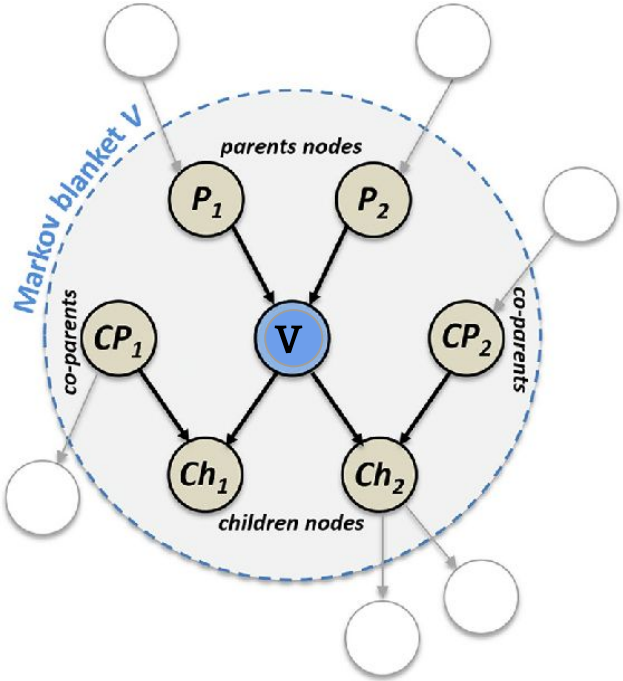



collider / v-structure

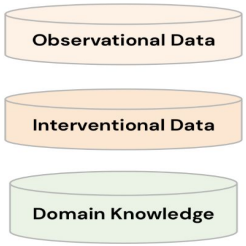
$X \perp\!\!\!\perp Y$

$X \not\perp\!\!\!\perp Y \mid Z$

## Markov blanket



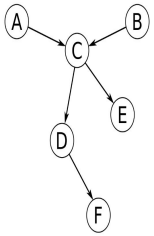
# Causal discovery methods for Science




**Causal Structure Learning**

Causal Graphs / Models

DAG




Graphical separation

$A \perp\!\!\!\perp_G B$
$A \perp\!\!\!\perp_G D \mid C$
$B \perp\!\!\!\perp_G D \mid C$
$A \perp\!\!\!\perp_G E \mid C$
$B \perp\!\!\!\perp_G E \mid C$
$D \perp\!\!\!\perp_G E \mid C$
$C \perp\!\!\!\perp_G F \mid D$
$\dots$

Probabilistic independence

$A \perp\!\!\!\perp_P B$
$A \perp\!\!\!\perp_P D \mid C$
$B \perp\!\!\!\perp_P D \mid C$
$A \perp\!\!\!\perp_P E \mid C$
$B \perp\!\!\!\perp_P E \mid C$
$D \perp\!\!\!\perp_P E \mid C$
$C \perp\!\!\!\perp_P F \mid D$
$\dots$

**Constraint-based structure learning**

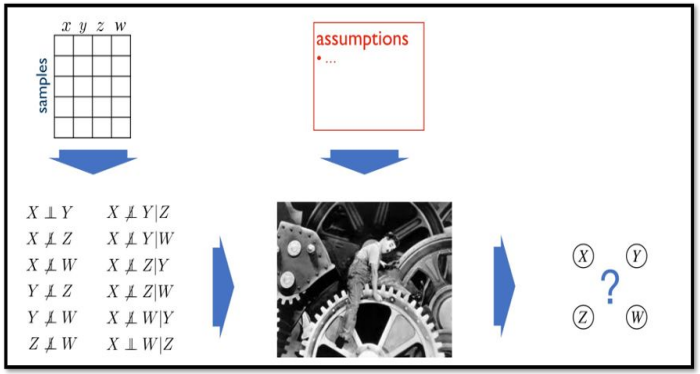
**CI tests**

For LINEAR GAUSSIAN models described by $X_i = \beta_j X_j + \epsilon_i$

we use PARAMETRIC CI tests (e.g. Fisher-Z test, Exact-t test)

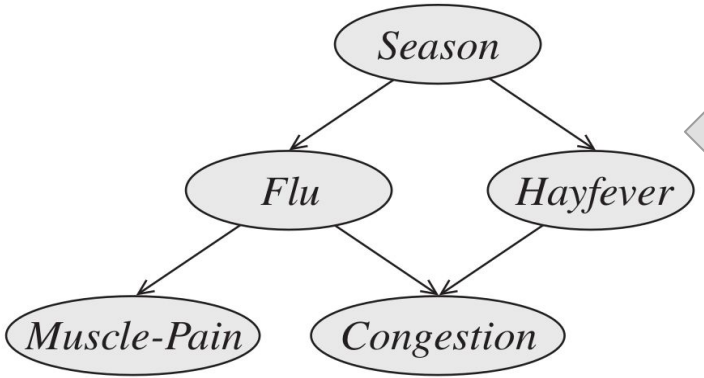
For NON-LINEAR ADDITIVE NOISE models described by $X_j = f_j(\mathrm{Pa}_j) + N_j$

we use kernel NON-PARAMETRIC tests (e.g. HSIC, DCC, KRESIT, RCIT)



# Basic example of causal structure learning

Graph Representation




**dataset**

| S | F | H | M | C |
|---|---|---|---|---|
| …. | .. | … | .. | .. |
| | | | | |

**questions**

what causes the congestion?

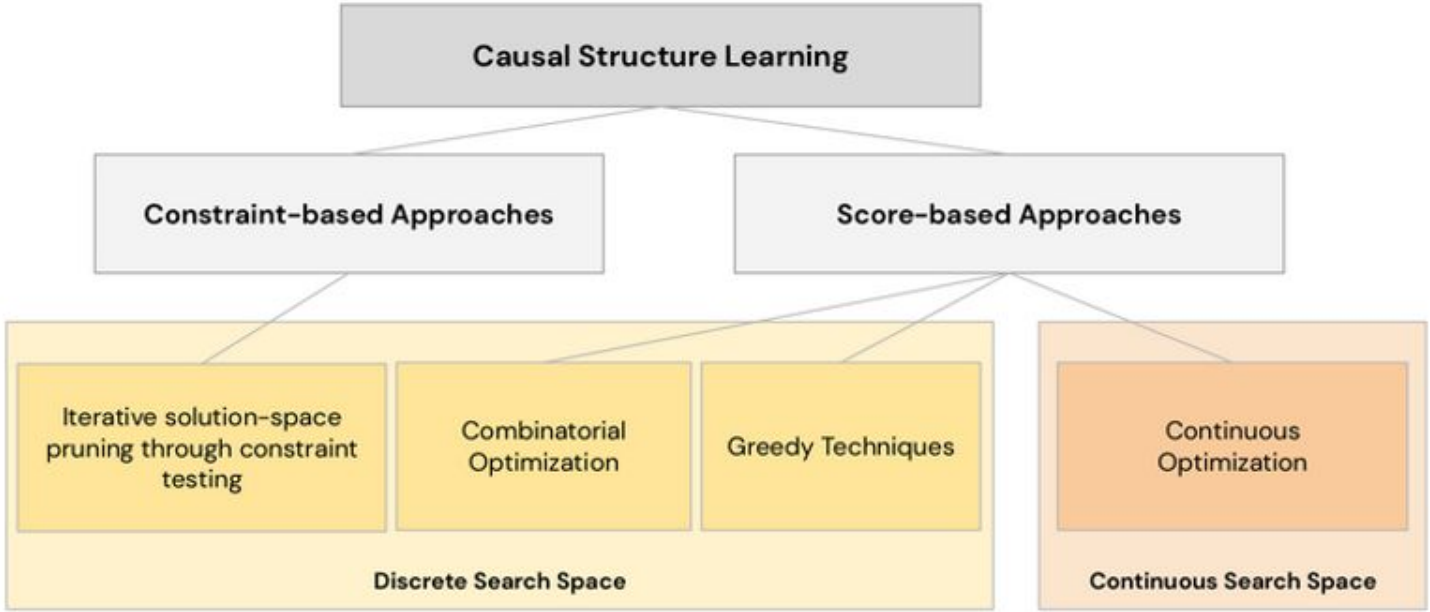
Independencies

$$(F \perp H \mid S)$$
$$(C \perp S \mid F, H)$$
$$(M \perp H, C \mid F)$$
$$(M \perp C \mid F)$$

Factorization

$$P(S, F, H, C, M) = P(S)P(F \mid S)$$
$$P(H \mid S)P(C \mid F, H)P(M \mid F)$$

# Closer look to causal structure learning



# Causal Structure-learning algorithms:

There are three main classes of algorithms for causal discovery:

- constraint-based —based on→ Conditional Independence tests
- score-based → score optimization + grid-search
- mixed type

Generally these algorithms assume:

1. Causal Markov
2. Causal sufficiency
3. Causal faithfulness

→ no collinear variables
no missing confounders

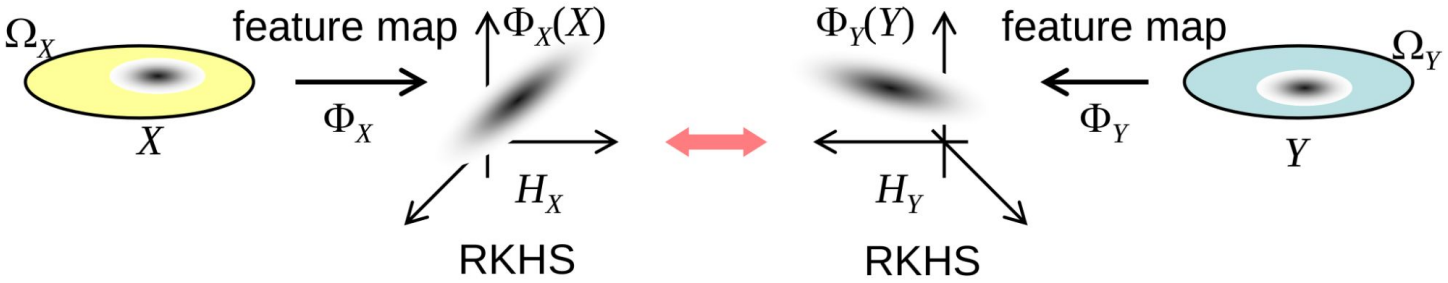
# Comparison between PC-stable and kernel-PC

PC-stable is the most common constraint-based algorithm and kernel-PC is its generalization to non-linear ANMs

| | Kernel-PC | PC-stable |
|---|---|---|
| **Differences in assumptions**<br><br>differences in implementation | non-linear relationship<br>gaussian/non-gaussian noise<br><br>generalized transitive phase | linear relationships<br>gaussian noise<br>gaussian data<br><br>transitive phase |
| **Conditional Independence test** | HSIC-gamma /DCC-gamma<br>HSIC-perm / DCC-perm | Pearson CI test<br>Fisher-Z |

# Kernel methods

- Positive definite kernels have been used for capturing nonlinearity of original data. *e.g*. Support vector machine.
- Kernelization: mapping data into a functional space (RKHS) and apply linear methods on RKHS.
- Recently, kernel methods have been applied for dependence analysis. Covariance structure on RKHS gives dependence and conditional dependence of the original variables.



# Kernel-based non-parametric independence tests

**Permutation test**. The first test is a simple permutation test where $r$ permutations of the form $y_{(j)} = \{y_{\rho_j(i)}\}$, $j = 1, \ldots, r$, for permutations $\rho_j$ of sample indices are created. The proportion of permutations $\rho_j$ for which the HSIC estimator (2) is larger than the HSIC of the original dataset, that is $H(x, y_{(j)}) > H(x, y)$, is an estimate of the $p$-value for rejecting the null hypothesis of independence. The underlying assumption is that permuting $y$ removes any dependency between $x$ and $y$.

## Gamma test

Gamma distribution: $H(x, y) \sim \text{Gam}(\alpha, \theta)$ where $\alpha$ is the shape parameter and $\theta$ is the scale parameter calculated as

$$\alpha = \frac{\mathbb{E}[\hat{\mathbb{H}}_{X,Y}]^2}{\text{Var}(\hat{\mathbb{H}}_{X,Y})}, \qquad \theta = \frac{\text{Var}(\hat{\mathbb{H}}_{X,Y})}{\mathbb{E}[\hat{\mathbb{H}}_{X,Y}]}$$

$p$-value is then obtained as upper-tail quantile of $H(x, y)$.

# Kernel CI tests based on residuals

**Residuals test.**

we propose to test residuals for independence based on any unconditional test of independence. The residuals $r_x$ and $r_y$ are obtained by regressing $x$ and $y$ on $z$ in a nonlinear fashion. The regression removes the dependencies between $x$ and $y$ due to $z$ and consequently the residuals should be independent if $\mathbf{X} \perp_P \mathbf{Y} \mid \mathbf{Z}$. For regression we use a generalized additive model (GAM, see *). That is, we regress $y$ on a set on of variables $x_i$, $i \in \{1, \ldots, p\}$ as $y = f_0 + \sum_{i=1}^{p} f_i(x_i) + \varepsilon$ where $f_i$ are spline functions (selected by cross-validation) and $\varepsilon$ is Gaussian noise. We have now the option of using either the permutation or Gamma test

* Simon N. Wood, 2006
Generalized Additive Models: an introduction with R

# Kernel CI tests: residuals gamma test

Test hypothesis:

$$\mathcal{H}_0 : X \perp\!\!\!\perp Y|Z \ \textbf{v.s.}\ \mathcal{H}_1 : X \not\perp\!\!\!\perp Y|Z$$

$$r_X \perp\!\!\!\perp r_Y$$

residuals are estimated with a GAM fit $y = f_0 + \sum_{i=1}^{p} f_i(x_i) + \varepsilon$ (pyGAM)

Gamma approximation

$$\mathrm{IC}(x, y) \sim \mathrm{Gam}(\alpha, \theta)$$

where $\alpha = \frac{\mathrm{E}[\hat{\mathrm{IC}}_{X,Y}]^2}{\mathrm{Var}(\hat{\mathrm{IC}}_{X,Y})}$, $\theta = \frac{\mathrm{Var}(\hat{\mathrm{IC}}_{X,Y})}{\mathrm{E}[\hat{\mathrm{IC}}_{X,Y}]}$

p-value: an upper-tail quantile

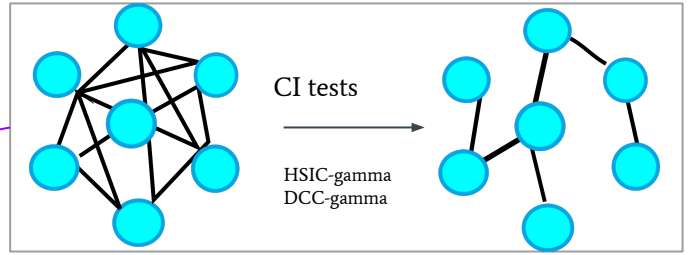
$$\mathbb{H}(P, Q) = \frac{1}{(N-1)^2}\mathrm{tr}(K_P H K_Q H)$$

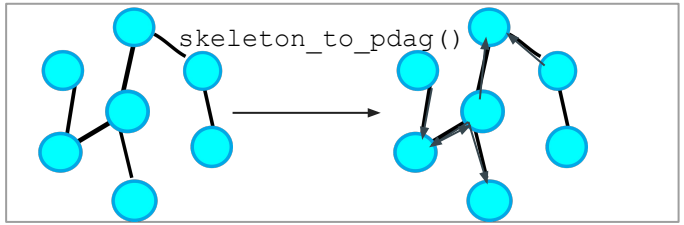
# Phases of the kernel-pc algorithm

The algorithm has 3 main phases:

1. **Skeleton phase**
   `build_skeleton()`
2. **Collider phase**
   `skeleton_to_pdag()`
3. **Generalized Transitive phase**
   `regrVonPS()`
   +
   `PDAG_by_Meek_rules()`

build_skeleton(data, ci_test, alpha, variant)



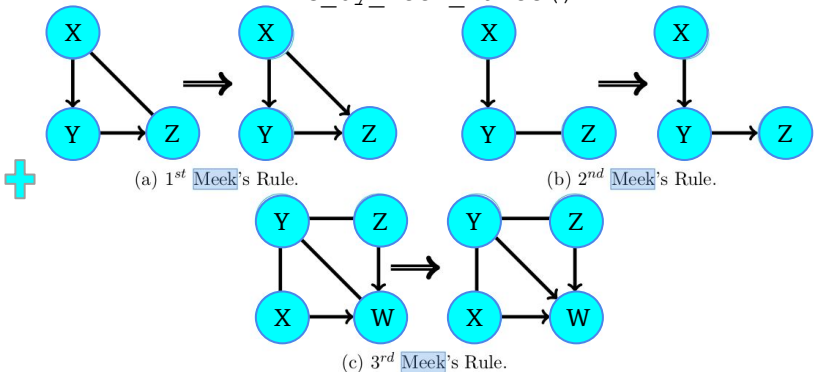




`regrVonPS()`

$$\begin{cases} r_X \not\perp\!\!\!\perp Y \\ r_Y \perp\!\!\!\perp X \end{cases}$$


$$\begin{cases} r_X \perp\!\!\!\perp Y \\ r_Y \not\perp\!\!\!\perp X \end{cases}$$


`PDAG_by_Meek_rules()`



# bnlearn (R - based)

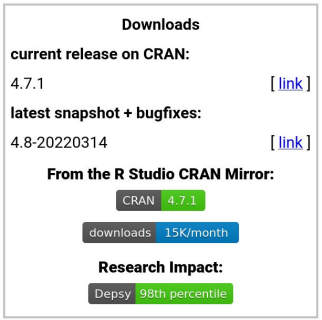
by Marco Scutari

**bnlearn** implements the following *constraint-based structure learning algorithms*:

- PC (the *stable* version);
- Grow-Shrink (GS);
- Incremental Association Markov Blanket (IAMB);
- Fast Incremental Association (Fast-IAMB);
- Interleaved Incremental Association (Inter-IAMB);
- Incremental Association with FDR Correction (IAMB-FDR);
- Max-Min Parents & Children (MMPC);
- Semi-Interleaved Hiton-PC (SI-HITON-PC);
- Hybrid Parents & Children (HPC);

the following *score-based structure learning algorithms*:

- Hill Climbing (HC);
- Tabu Search (Tabu);




code available at:
https://cran.r-project.org/web/packages/bnlearn/

documentation:
https://www.bnlearn.com/

# kpcalg (R - based)

Kernel PC Algorithm for Causal Structure Detection

| | |
|---|---|
| Version: | 1.0.1 |
| Depends: | R (≥ 3.0.2) |
| Imports: | pcalg, energy, kernlab, parallel, mgcv, RSpectra, methods, graph, stats, utils |
| Suggests: | Rgraphviz, knitr |
| Published: | 2017-01-22 |
| Author: | Petras Verbyla, Nina Ines Bertille Desgranges, Lorenz Wernisch |
| Maintainer: | Petras Verbyla <petras.verbyla at mrc-bsu.cam.ac.uk> |
| License: | GPL-2 \| GPL-3 [expanded from: GPL (≥ 2)] |
| NeedsCompilation: | no |

code available at:
https://cran.r-project.org/web/packages/kpcalg

# gABiC library

main · 2 branches · 0 tags · Go to file · Add file · Code

Sera91 Update README.md — 4bcee2b 4 minutes ago — 88 commits

| | | |
|---|---|---|
| gABiC | Update README.md | 1 hour ago |
| profiling | new file: profiling/image_zoom.html | 6 months ago |
| .gitignore | new file: .gitignore | 6 months ago |
| README.md | Update README.md | 4 minutes ago |
| SETUP.md | Update SETUP.md | 3 days ago |
| gABI.yml | new file: gABI.yml | last week |
| logo-pacchetto-github.png | Add files via upload | 5 months ago |
| test_Asia.py | modified: coreBN/build/lib/coreBN/estimators/PC.py | last week |

README.md


## gABiC: graphical Automated Bayesian inference in Cosmology

Python package to perform Bayesian causal discovery on data, based on probabilistic graphical models.

This package have the following dependencies:

**About**

Python-C package for performing bayesian automated inference on data, based on probabilistic graphical models.

Readme
0 stars
3 watching
0 forks

**Releases**

No releases published
Create a new release

**Packages**

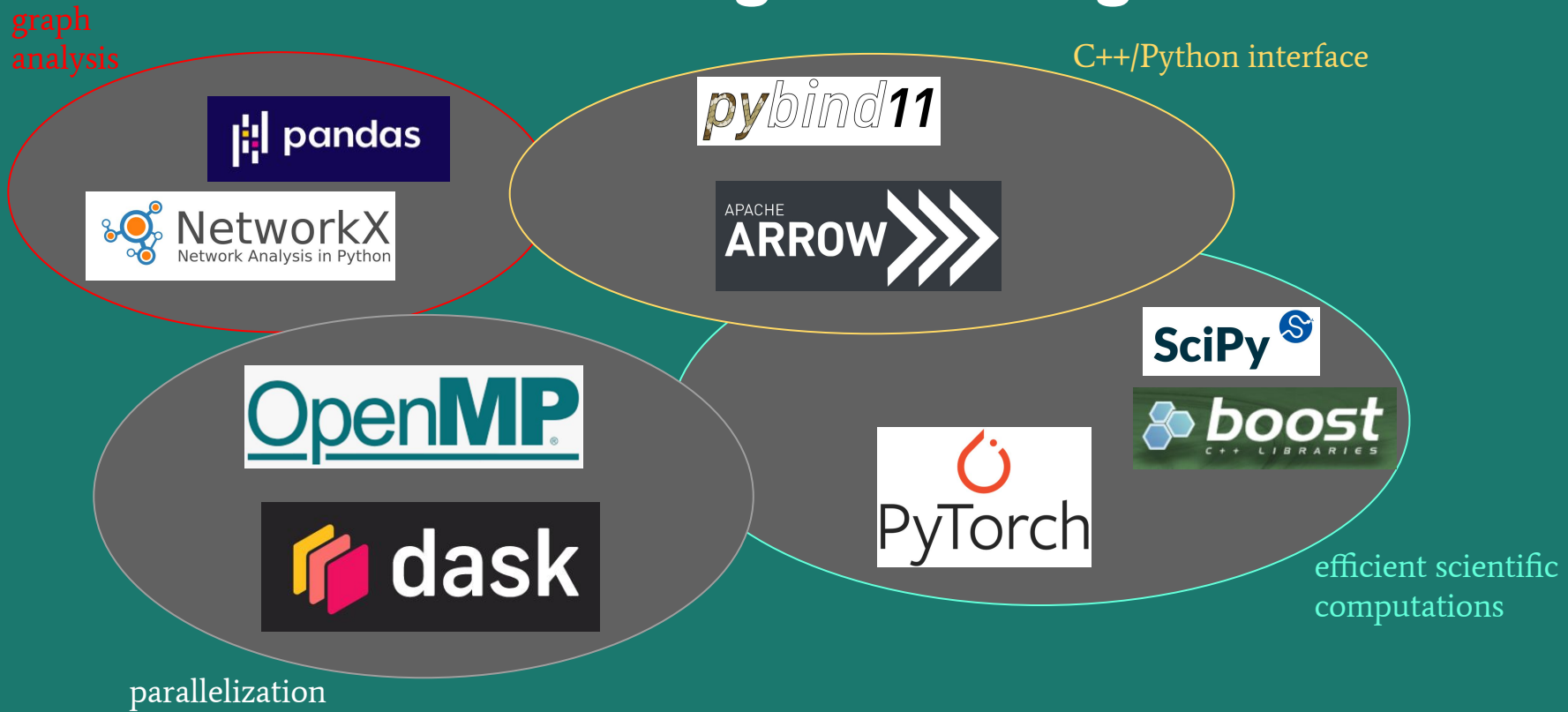
No packages published
Publish your first package

**Languages**

Python 99.9% Other 0.1%

```
coreBN
├── base
│   ├── DAG.py
│   ├── __init__.py
│   └── UndirectedGraph.py
├── CItests
│   ├── CITests.py
│   ├── dcc_gamma.py
│   ├── dcc_gamma_pytorch.py
│   ├── dcc_perm.py
│   ├── hsic_gamma.py
│   ├── hsic_gamma_pytorch.py
│   ├── hsic_perm.py
│   ├── __init__.py
│   └── kernel_CITests.py
├── estimators
│   ├── base.py
│   ├── BayesianEstimator.py
│   ├── EM.py
│   ├── Estimator.py
│   ├── HillClimbSearch.py
│   ├── __init__.py
│   ├── kPC.py
│   ├── LinearModel.py
│   ├── MLE.py
│   ├── PC-old.py
│   ├── PC.py
│   ├── ScoreCache.py
│   ├── SEMEstimator.py
│   └── StructureScore.py
  ................
├── data
│   ├── asia.bif
│   ├── asia.png
│   ├── ASIA.txt
│   ├── asia.zip
│   └── __init__.py
├── datasets_input
│   ├── alarm_10k.csv
│   ├── Asia_10k.csv
│   ├── Asia_10k_new.csv
│   ├── Asia_10k_old.csv
│   ├── Child_10k.csv
│   └── Sprinkler_1000.csv
├── data_utils.py
├── fitting_frontend.py
├── __init__.py
├── learning_frontend.py
```

# APIs/Packages used in gABi



# Work on the algorithms in gABI:

I implemented a parallel Python version, using GPU computing, of 3 constraint-based structure learning algorithms (PC-stable, its generalization kernel-PC, and FCI) , following these steps

1. porting serial version from R to Python
2. profiling Python code  (with cProfile)
3. serial code optimization
4. GPU porting of independence test algorithm
5. parallelization with Dask + MPI4Py

# Optimization of the HSIC independence test

NUMBA CPU kernel function

```python
@njit()
def CPU_kernel_matrix(x,n, sigma=1.0):
    """
    this function build the kernel matrix
    """
    out = np.ones((n,n), dtype=np.float64)
    for i in range(n):
        for j in range(i+1,n):
                out[i,j] = exp(-sigma*(x[i] - x[j])*(x[i]-x[j]))
                #print("K(",i,j,")=", out[i,j])
                out[j,i] = out[i,j]
    return out
```

Speedup: x10

NUMBA GPU kernel function

```python
@cuda.jit
def GPU_kernel_matrix(x, n, out, sigma):
        i, j = cuda.grid(2)
        if (i < n) and ((i+1) <=j < n) :
            out[i, j] = exp(-sigma*(x[i] - x[j])*(x[i]-x[j]))
            out[j, i] = out[i, j]
```

Speedup: x100

$$H/H_{ij} = \delta_{ij} - 1/n$$

$$HSIC(Z, \mathscr{F}, \mathscr{G}) := (m-1)^2 tr(KHLH)$$

$$K(i,j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\lambda^2}\right)$$

**Table 5.1:** Runtimes for different HSIC-gamma implementations

| $N_{obs}$ | $t_R$ | $t_{np+numba}[s]$ | $t^{CPU}_{pytorch}$ [s] | $t^{GPU}_{pytorch}$ [s] |
|---|---|---|---|---|
| 300 | 0.072 | 0.0657 | 0.015 | 0.114 |
| 500 | 0.092 | 0.0847 | 0.045 | 0.121 |
| 1000 | 0.453 | 0.286 | 0.273 | 0.133 |
| 2000 | 5.41 | 2.033 | 2.008 | 0.179 |
| 3000 | 18.72 | 6.609 | 6.452 | 0.281 |
| 5000 | 79.5 | 28.50 | 27.74 | 0.674 |
| 10000 | 654.34 | 138.55 | 134.24 | 1.11 |

# DCC-gamma test runtimes

**Table 5.2:** Runtimes for different DCC-gamma implementations

| $N_{obs}$ | $t_R$ | $t_{np+numba}[s]$ | $t_{pt}^{CPU}$ [s] | $t_{pt-symm}^{CPU}$ [s] | $t_{pt}^{GPU}$ [s] | $t_{pt-symm}^{GPU}$ |
|---|---|---|---|---|---|---|
| 300 | 0.43 | 0.46 | 0.41 | 0.31 | 0.49 | 0.34 |
| 500 | 0.874 | 0.525 | 0.50 | 0.34 | 0.84 | 0.53 |
| 1000 | 5.572 | 3.273 | 2.95 | 1.5 | 2.33 | 0.63 |
| 2000 | 39.60 | 22.25 | 21.28 | 10.47 | 5.63 | 0.97 |
| 5000 | 360.80 | 221.74 | 190.67 | 95.36 | 10.32 | 2.01 |
| 10000 | 1380.55 | 634.24 | 570.11 | 310.32 | 16.4 | 3.51 |

# Steps parallelized in the pc algorithm

## Phase I

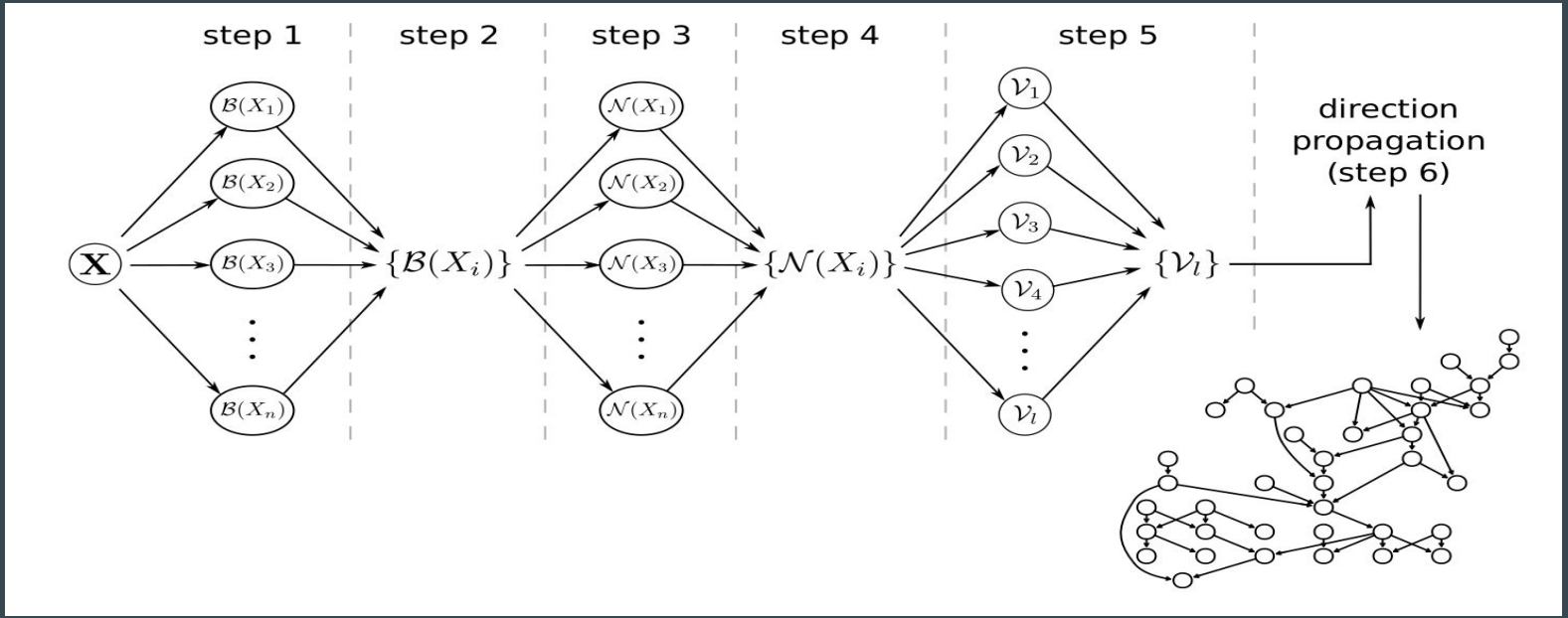
build_skeleton(ci_test, significance_level, variant)

## Phase II

skeleton_to_pdag()

## Phase III

regrVonPS + PDAG_by_Meek_rules()



# DASK/DASK-MPI based parallelization


```python
import os
from dask_mpi import initialize
from dask.distributed import Client, as_completed

if __name__ == "__main__":

    ...

    #reading input data set into pandas dataframe
    print('I am before client inizialization')
    # Initialise Dask cluster and client interface

    n_tasks = int(os.getenv('SLURM_NTASKS'))
    mem = os.getenv('SLURM_MEM_PER_CPU')
    mem = str(int(mem))+'MB'

    initialize(memory_limit=mem)

    dask_client = Client()

    dask_client.wait_for_workers(n_workers=(n_tasks-2))
    #dask_client.restart()

    num_workers = len(dask_client.scheduler_info()['workers'])
    print("%d workers available and ready"%num_workers)
```

**Listing 5.2:** Dask cluster initialization

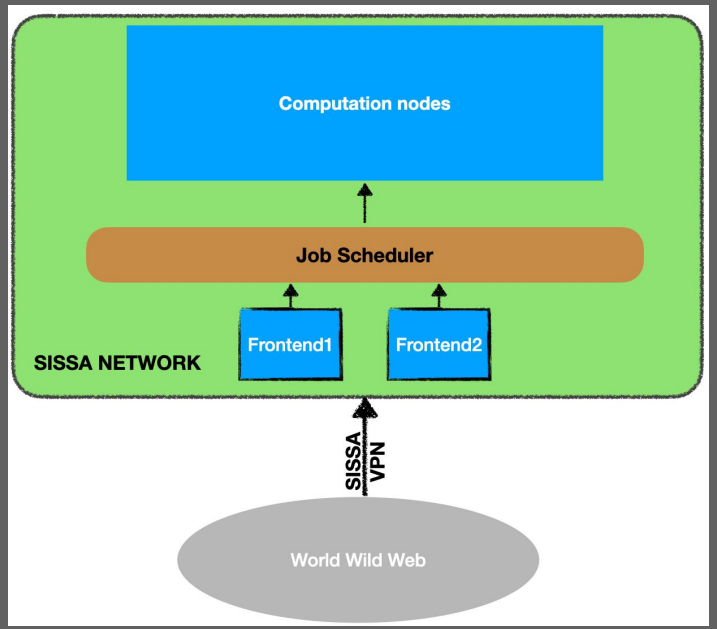
```
#!/bin/bash
#SBATCH --job-name=kpc-2node
#SBATCH --partition=regular2
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=12
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=5000
#SBATCH --time=12:00:00


module purge
module load gnu11 openmpi3  gsl cuda/11.0


source /home/sdigioia/.bashrc
conda activate py310-ul

mpirun -n 24 python parallel_kpcalg.py > output_run.txt
```

# HPC resources: Ulysses cluster (v2)




JOB SCHEDULER: SLURM
queues used : regular 2 & gpu2

**Table 1:** Compute node architecture

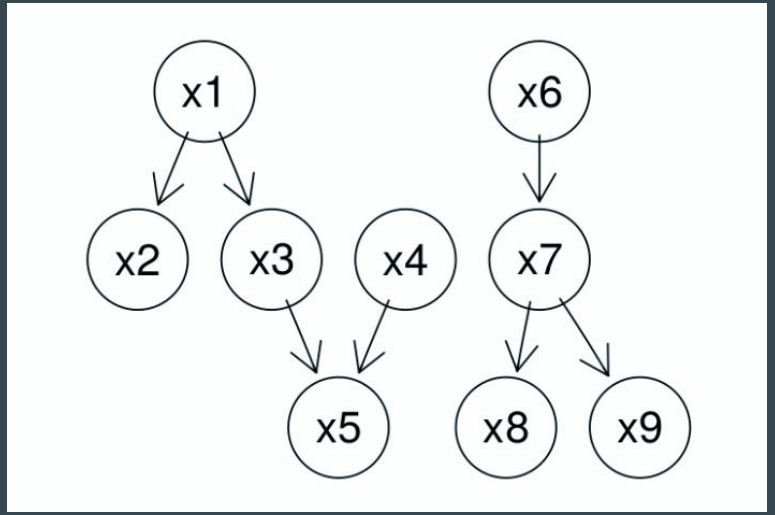
| | CPU nodes | GPU nodes |
|---|---|---|
| Model | HPE Pro XL170r Gen9 | same |
| CPU | Intel Xeon(R) E5-2683 v4 @ 2.10GHz | same |
| RAM | 64 GB | 64 GB |
| GPU | —— | NVIDIA TESLA P100 (16GB) |

Each node has: 2 sockets/16 cores, 2 threads per core

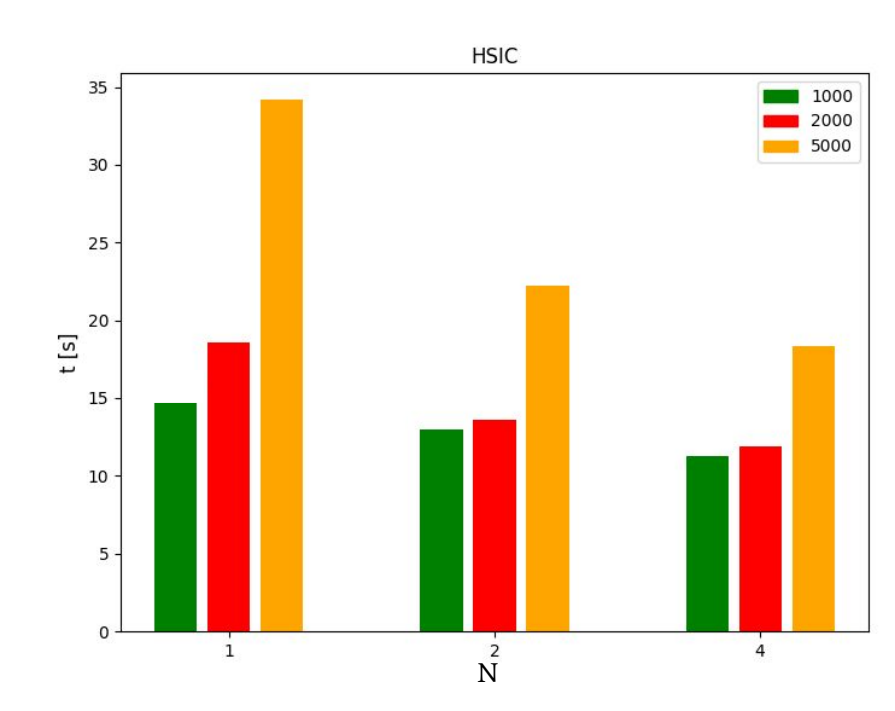
# Benchmark of Kernel-PC

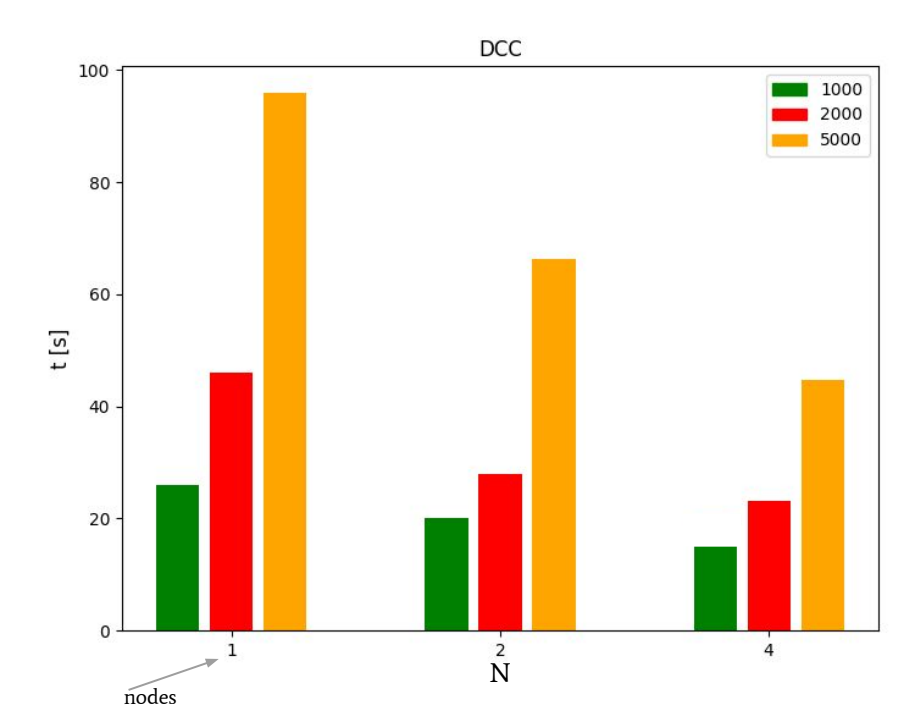
```r
set.seed(4)
n <- 300
data <- NULL
x1 <- 2*(runif(n)-0.5)
x2 <- x1 + runif(n)-0.5
x3 <- x1^2 + 0.6*runif(n)
x4 <- rnorm(n)
x5 <- x3 + x4^2 + 2*runif(n)
x6 <- 10*(runif(n)-0.5)
x7 <- x6^2 + 5*runif(n)
x8 <- 2*x7^2 + 1.5*rnorm(n)
x9 <- x7 + 4*runif(n)
data <- cbind(x1,x2,x3,x4,x5,x6,x7,x8,x9)
```

true DAG



# Multi-GPU nodes benchmark of kernel-pc



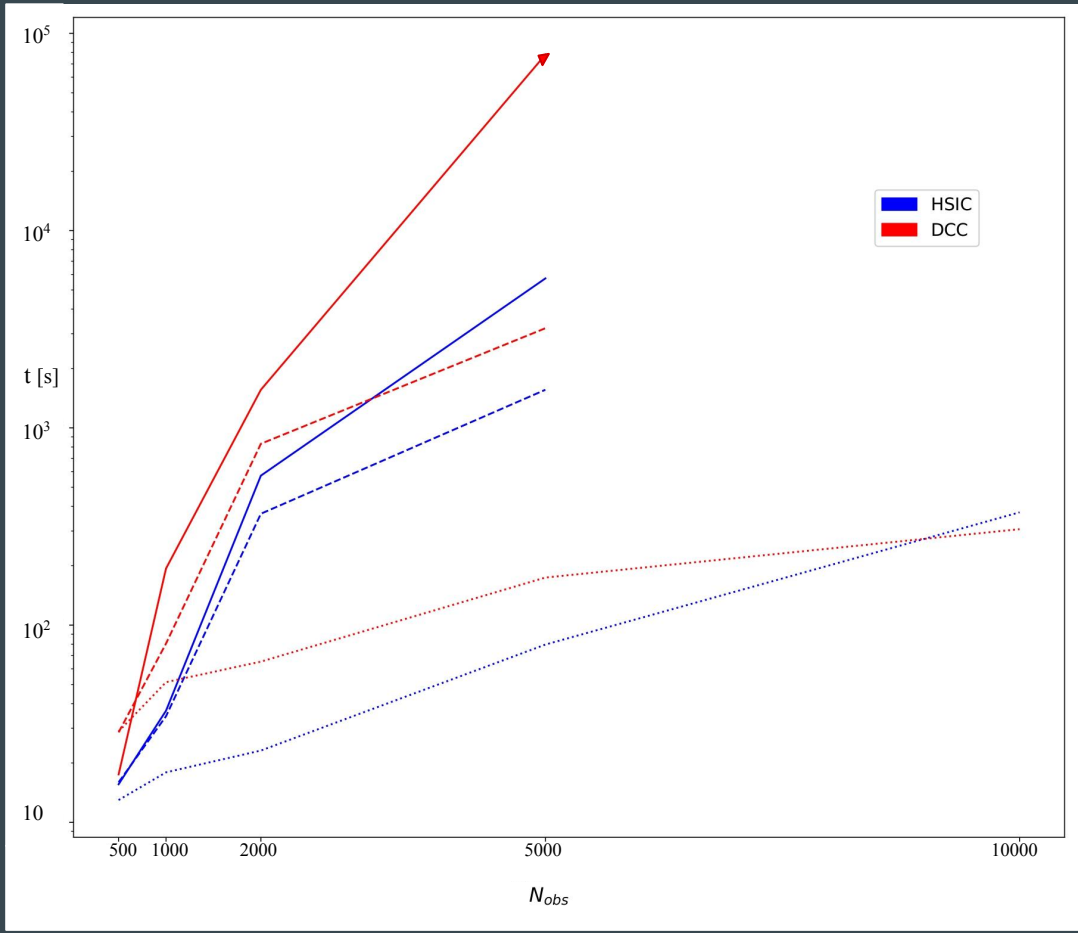



# Weak scaling of serial kernel -PC

—— kpcalg

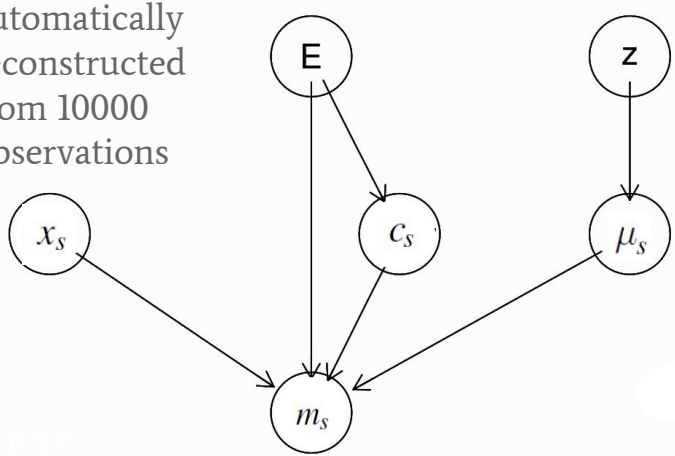
- - gABiC - CPU

····· gABiC - GPU

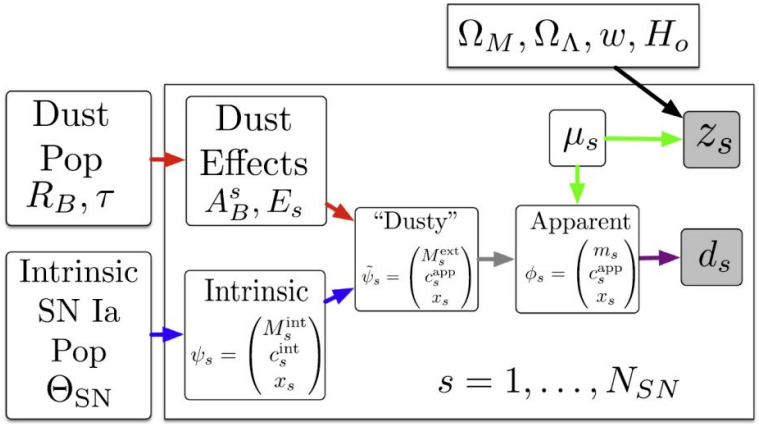


# Case study: simple model for simulated SNIa data

automatically reconstructed from 10000 observations




where




$$M_s = m_s - \mu_s = M_0 + \alpha \times x_s + \beta \times c_s + \epsilon^s_{\text{res}}$$

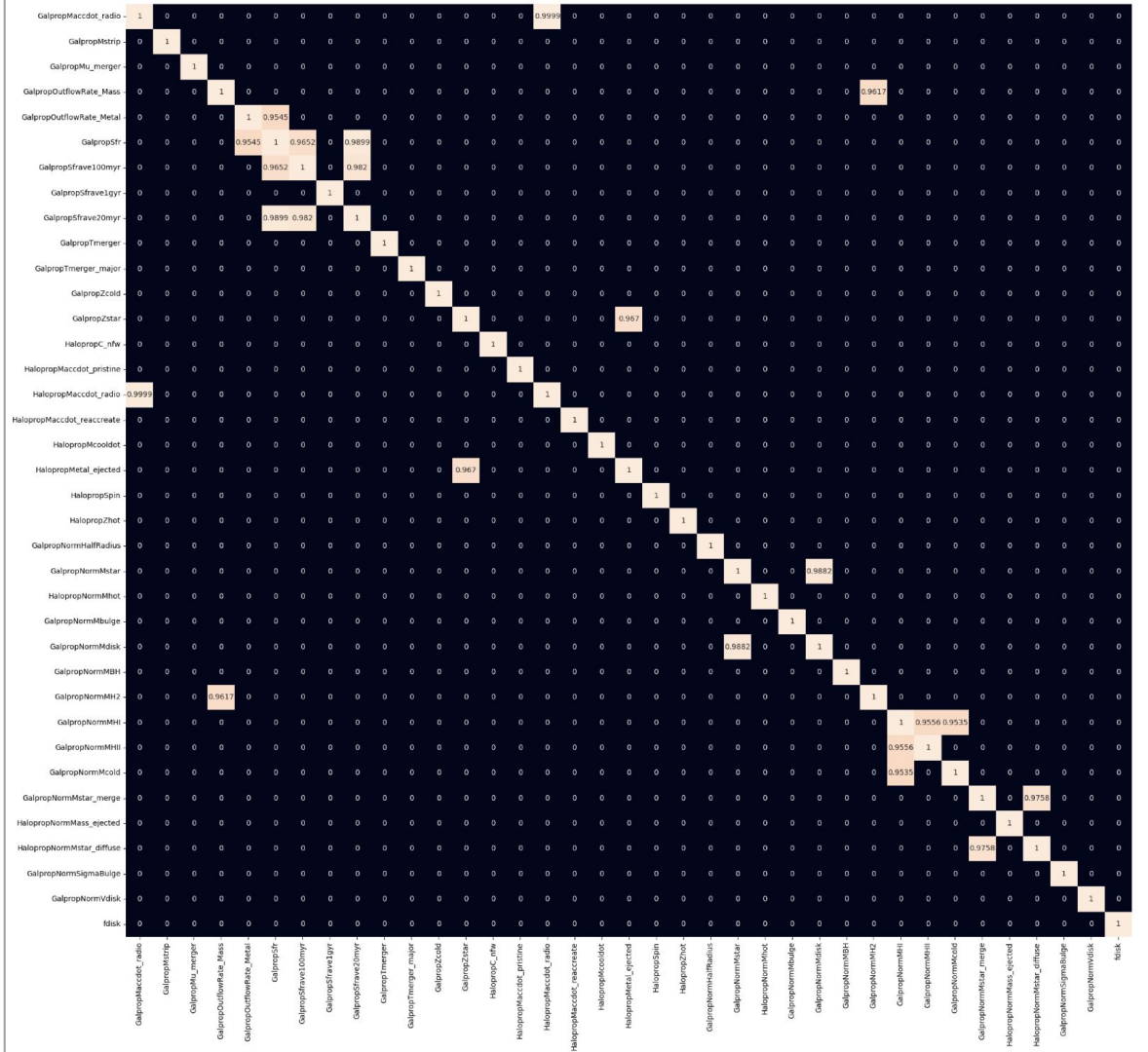
# First application: what causes disk size evolution ?

SAM : Santa-Cruz TNG

(GabrielPillai, 2021)

## Preprocessing steps

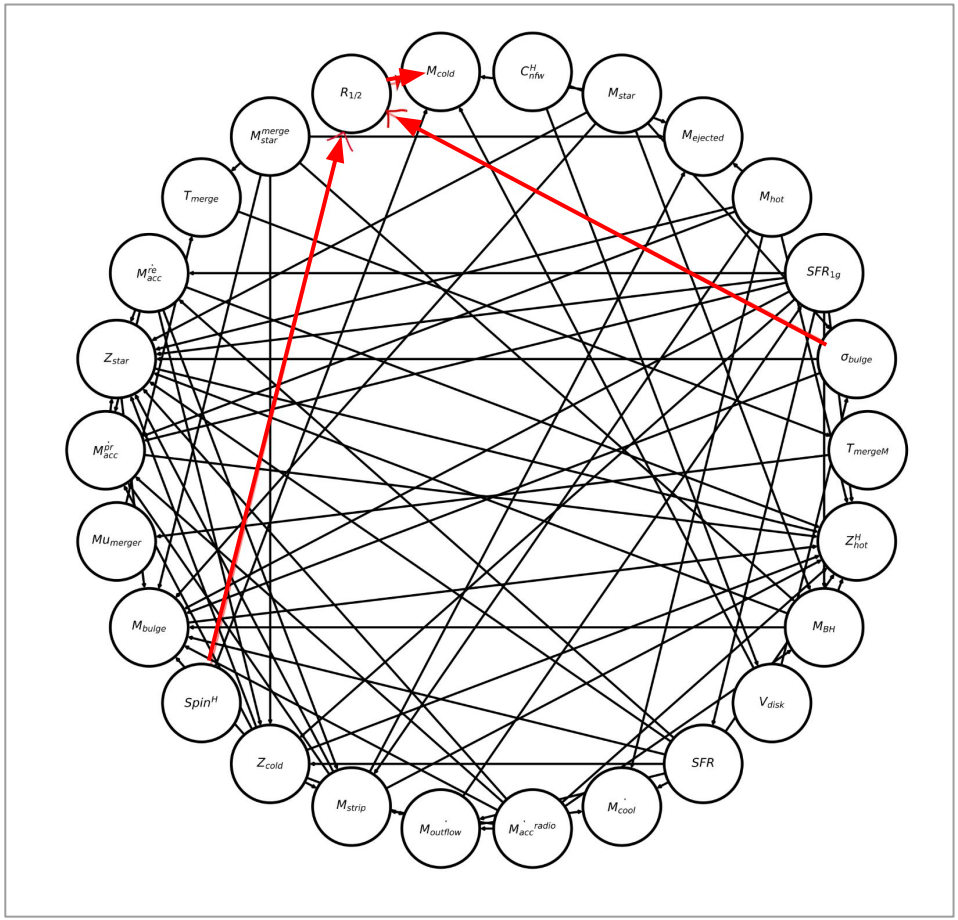
1. subsetting data by galaxy morphology (disks / ellipticals)
2. identification of collinear variables



# Average reconstructed BN

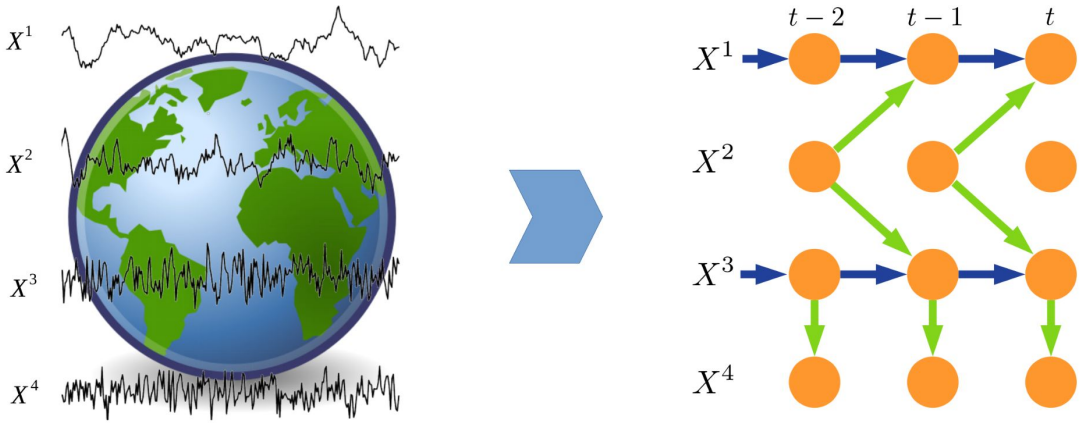
After preprocessing -> we are left with 26 vars out of 37

| Name | description | Name | description |
|---|---|---|---|
| $M_{strip}$ | stripping mass | $SFR_{1g}$ | $< SFR >_{1Gyr}$ |
| $M_{outflow}$ | outflow rate | $T_{merge}$ | T last merger |
| $\mu_{merger}$ | mass ratio mergers | $T_{mergeM}$ | T last major merger |
| $Z_{cold}$ | metallicity cold gas | $M_{acc}^{pr}$ | pristine accretion rate |
| $M_{acc}^{re}$ | reaccretion rate | $C_{nfw}$ | halo concentration |
| $Spin^H$ | halo spin | $M_{cool}$ | cooling mass |
| $R_{1/2}$ | half-mass radius | $Z_{hot}^H$ | metallicity hot gas |
| $M_{BH}$ | BH mass | $M_{cold}$ | CG mass |
| $M_{ejected}$ | Ejected Mass | $V_{disk}$ | disk vel. |
| $M_{star}$ | stellar mass | $M_{hot}$ | Hot gas mass |
| $\sigma_{bulge}$ | bulge vel. disp. | $SFR$ | Star formation rate |
| $M_{bulge}$ | bulge mass | $Z_{star}$ | stellar metallicity |
| $M_{star}^{merge}$ | stellar mass from merg. | $M_{acc}^{radio}$ | radio accretion rate |




BN averaged over 10 bootstraps

# Constraint-based causal discovery applied to time-series




**Particularities:**

- Variables are resolved in time
- Autocorrelation

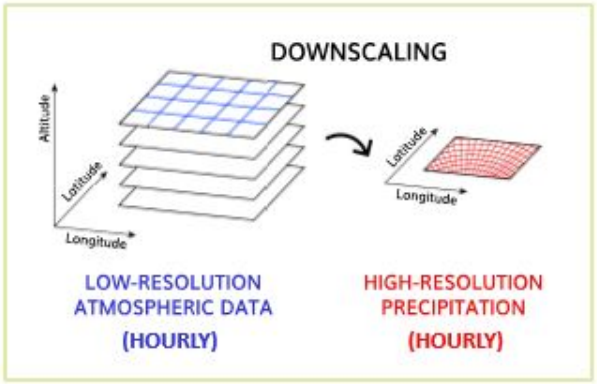
Ref: Malinsky, D. and Spirtes, P. (2018).

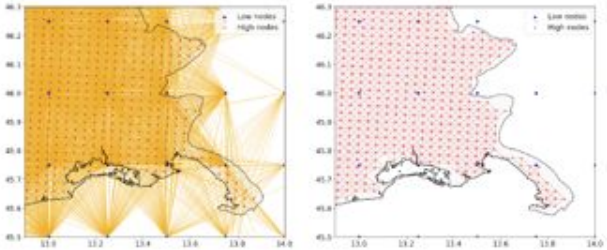
# DL Emulators for Climate (based on GNN)

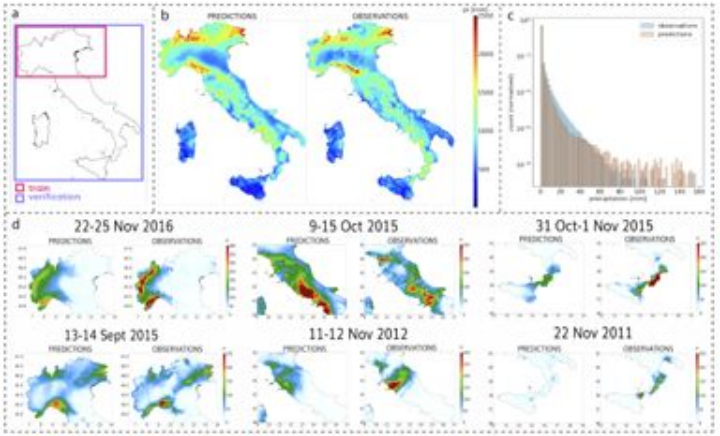
# DL Emulators 4 Climate

PI: Erika Coppola (ICTP)
Main collaborator: Valentina Blasone (PhD student @ UNITS)

Emulate with DL the convection permitting dynamical models to derive high-resolution precipitation distribution






**Goal**

improving the projection of climatic impact-drivers relevant for risk assessment, focusing on cost and energy efficient solutions.

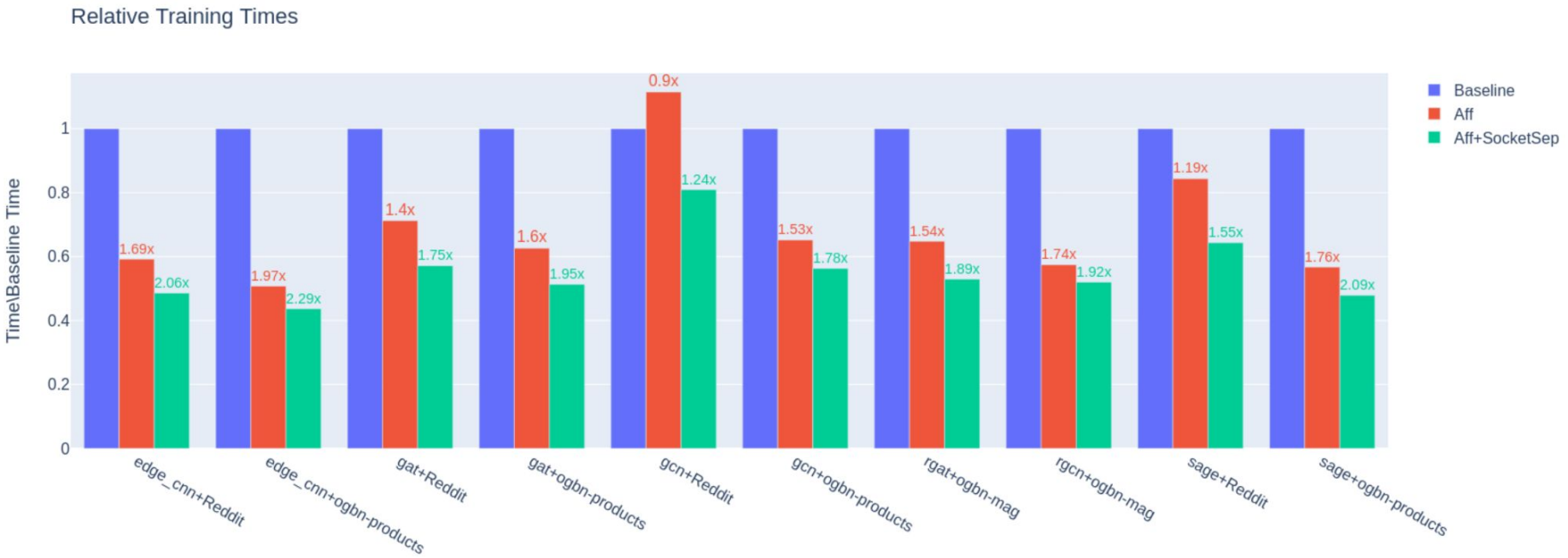
WCRP CORDEX FLAGSHIP PILOT STUDIES

Master in High Performance Computing

XAIDA

# Steps to optimize execution of the GNN model on CPUs

1. Enable multi-process data loaders by setting `num_workers > 0`.
2. Use the `enable_cpu_affinity()` feature to affinitize `DataLoader` cores.
3. Bind execution to physical cores. Separate the cores used for main process from the data loader workers' cores by using `numactl`, `KMP_AFFINITY` of the `libiomp5` library
4. Use non-default memory allocator, such as `jemalloc` or `TCMalloc`. We use "jemalloc"
5. Find the optimum number of OMP threads for our workload. A good starting point is `N=4`. Generally, well-parallelized models will benefit from many OMP threads.
6. TODO: When using a dual-socket CPU, you might want to experiment with assigning data loading to one socket and main process to another socket with memory allocation (`numactl -m`) on the same socket where the main process is executed. This leads to best cache-allocation and often overweighs the benefit of using more OMP thread.


Relative Training Times
Time\Baseline Time
0
0.2
0.4
0.6
0.8
1
Baseline
Aff
Aff+SocketSep
1.69x
2.06x
1.97x
2.29x
1.4x
1.75x
1.6x
1.95x
0.9x
1.24x
1.53x
1.78x
1.54x
1.89x
1.74x
1.92x
1.19x
1.55x
1.76x
2.09x
edge_cnn+Reddit
edge_cnn+ogbn-products
gat+Reddit
gat+ogbn-products
gcn+Reddit
gcn+ogbn-products
rgat+ogbn-mag
rgcn+ogbn-mag
sage+Reddit
sage+ogbn-products

Edwards DI (2000). Introduction to Graphical Modelling. Springer, 2nd edition

Russell SJ, Norvig P (2009). Artificial Intelligence: A Modern Approach. Prentice Hall, 3rd edition

Tsamardinos I, Borboudakis G (2010). "Permutation Testing Improves Bayesian Network Learning". Machine Learning and Knowledge Discovery in Databases, 322–337

Colombo D, Maathuis MH (2014). "Order-Independent Constraint-Based Causal Structure Learning". Journal of Machine Learning Research, 15:3921–3962.

Matthew J. Wovels et al. (2021), D'ya like DAGs? A Survey on Structure Learning and Causal Discovery, arXiV 2103.02582v2

# Programming skills

**Programming languages** : C, C++, Python, Fortran, R, Julia(basic)

**Python libraries**
**for ML** : pandas, scipy, statsmodels, scikit-learn, Pytorch, pyarrow, dask, PySR, torch-geometric
**for Physics** : Astropy , sympy, pymc3, pygam
**for networks** : Networkx, Graphviz, itertools

**R libraries** : **kernlab, bnlearn, pcalg, kpcalg**

**Efficient Linear Algebra** : BLAS, LAPACK, PLASMA, **MAGMA**, SLATE
**Scientific libraries in C/C++** : Boost, GSL, MAGMA-DNN, FFTW

**tools for library development : git, google tests, Docker**

**GPU programming** : CUDA , SYCL, OpenMP Offload
**Parallel programming** : MPI, OpenMP, dask, MPI4Py, numba

**Interfacing different languages**
Python-C++ : pybind11, Python-C API
R-Python-C-C++ : Apache Arrow


# The End

# HSIC independence criterion

Formally, consider a set of i.i.d. sample tuples $\{(p_i, q_i)\}_{i=1}^N$, where $p_i \in \mathbb{R}^d$, $q_i \in \mathbb{R}^c$. Let $P \in \mathbb{R}^{N\times d}$ and $Q \in \mathbb{R}^{N\times c}$ be the matrices whose rows are the corresponding samples. Also, let $k_p : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and $k_q : \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ be two characteristic kernel functions for $p_i$ and $q_i$, respectively. Examples are

$$k_P(p_i, p_j) = e^{-\frac{\|p_i - p_j,\|^2}{2\sigma^2}} \tag{1}$$

i.e., the Gaussian kernel,

$$k_Q(q_i, q_j) \tag{2}$$

We further define $K_P, K_Q$ to be the kernel matrices for $P$ and $Q$ respectively, where $K_P = \{k_P(p_i, p_j)\}_{i,j} \in \mathbb{R}^{N\times N}$ and $K_Q = \{k_Q(q_i, q_j)\}_{i,j} \in \mathbb{R}^{N\times N}$.

The HSIC between $P$ and $Q$ is estimated empirically with kernels $k_P, k_Q$ via:

$$\mathbb{H}(P, Q) = \frac{1}{(N-1)^2}\mathrm{tr}(K_P H K_Q H), \tag{3}$$

where $H_{i,j} = \delta_{i,j} - N^{-1}$. Intuitively, HSIC measures the dependence between the random variables $p, q$ from which the i.i.d. samples $\{(p_i, q_i)\}_{i=1}^N$ where generated.

# DCC (dCov): Distance Covariance Criterion

Key idea: *"looking at the correlation of the distances between the datapoints instead of the correlation of the datapoints themselves"*

**dCor**

$$\mathcal{R}_n^2(x,y) = \begin{cases} \frac{\mathcal{V}_n^2(x,y)}{\sqrt{\mathcal{V}_n^2(x)\mathcal{V}_n^2(y)}}, & \mathcal{V}_n^2(x)\mathcal{V}_n^2(y) > 0, \\ 0, & \mathcal{V}_n^2(x)\mathcal{V}_n^2(y) = 0. \end{cases}$$

**Empirical estimator for DC:**

$$\widehat{\mathcal{V}}_n^2(X,Y) = \nu(x,y) = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl}B_{kl}$$

where $B_{k,l} = b_{k,l} - \bar{b}_{k\cdot} - \bar{b}_{\cdot l} + \bar{b}_{\cdot\cdot}$

$$A_{k,l} = a_{k,l} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}$$

$$a_{kl} = \|x_k - x_l\|_2^{\xi}, \quad \bar{a}_{k\cdot} = \frac{1}{n}\sum_{l=1}^{n} a_{kl} \quad \bar{a}_{\cdot l} = \frac{1}{n}\sum_{k=1}^{n} a_{kl} \quad \bar{a}_{\cdot\cdot} = \frac{1}{n^2}\sum_{k,l=1}^{n} a_{k,l}$$

# CI test for Linear Gaussian models

Conditional independence tests used to learn GBNs are functions of the partial correlations $\rho_{XY|\mathbf{Z}}$ that are used as proxies for the cells of $\Omega = \Sigma^{-1}$. Classic choices are:

**Pearson test**

$$\mathrm{t}(X, Y \mid \mathbf{Z}) = \rho_{XY|\mathbf{Z}} \sqrt{\frac{n - |\mathbf{Z}| - 2}{1 - \rho^2_{XY|\mathbf{Z}}}}$$

and distributed as a Student's $t$ with $n - |\mathbf{Z}| - 2$ degrees of freedom;

# Orienting edges in PDAG with residuals I tests

Assuming that data come from a Non-linear Additive noise model, and supposing to be interested in orienting the undirected edge x-y, we can either regress y on x, getting the model $y = \hat{f}(x) + r_y$ (x is causing y), or x on y, getting $x = \hat{g}(y) + r_x$, and if either $f/g$ is non-linear or/and $\epsilon$ is non Gaussian noise, we can distinguish which of the two models describes our data using the kernel independence criteria. We can subdivide the possible outcomes of this phase in three cases:

1. if we get independence in both the test hypotheses for the 2 models, respectively $r_y \perp\!\!\!\perp x$ and $r_x \perp\!\!\!\perp y$ , then we leave the edge undirected
2. if we find $r_y \not\perp\!\!\!\perp x$ and $r_x \not\perp\!\!\!\perp y$ our model is misspecified, and we leave the edge undirected (this could happen when we have unobserved latent variables)
3. if we get independence of the residuals in one of the model, associated to one causal direction, and not in the other ( for example we find $r_y \not\perp\!\!\!\perp x$ and $r_x \perp\!\!\!\perp y$ ) then we can direct the edge accordingly.

Edwards DI (2000). Introduction to Graphical Modelling. Springer, 2nd edition

Fukumizu(2007) Kernel Measures of Conditional Dependence

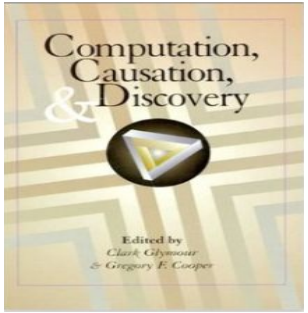
Tsamardinos I, Borboudakis G (2010). "Permutation Testing Improves Bayesian Network Learning". Machine Learning and Knowledge Discovery in Databases, 322–337

Tillman, R. E., Gretton, A. and Spirtes, P. (2009). Nonlinear directed acyclic structure learning with weakly additive noise model. NIPS 22, Vancouver

Colombo D, Maathuis MH (2014). "Order-Independent Constraint-Based Causal Structure Learning". Journal of Machine Learning Research, 15:3921–3962. (article introducing pc-stable)

Matthew J. Wovels et al. (2021), D'ya like DAGs? A Survey on Structure Learning and Causal Discovery, arXiV 2103.02582v2

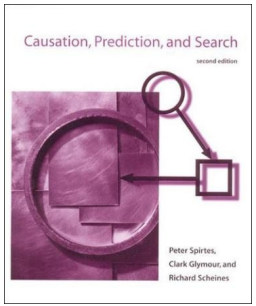
# 30 Years of Advances

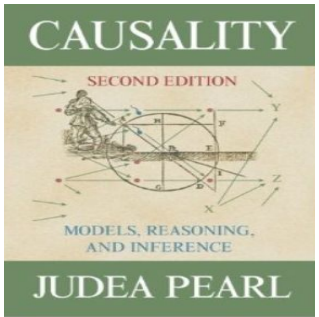


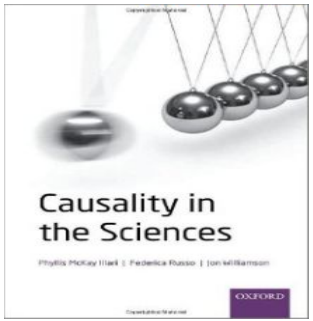

1988 □ 2018: tremendous progress in:

- Mathematical representations of causal systems
- Reliable and useful discovery algorithms for finding causal structure from a combination of data and background knowledge

These methods will be essential to future AI!



Testable Constraints
(CI tests)

Intervention
Manipulation

Causal

Causal

Counterfactuals

- Each variable $X_i$ is conditionally independent of its non-effects, both direct and indirect, given its direct causes (the causal Markov assumption, much like the original but causal);
- There must exist a DAG which is faithful to the probability distribution $\mathbf{P}$ of $\mathbf{X}$, so that the only dependencies in $\mathbf{P}$ are those arising from d-separation in the DAG.
- There must be no latent variables (unobserved variables influencing the variables in the network) acting as confounding factors. Such variables may induce spurious correlations between the observed variables, thus introducing bias in the causal network.

| algorithm/ assumptions | PC / GES | FCI | CCD | LiNGaM | lvLiNGaM | cyclic LiNGaM | non-linear additive noise |
|---|---|---|---|---|---|---|---|
| Markov | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| faithfulness | ✓ | ✓ | ✓ | ✗ | ✓ | ~ | minimality |
| causal sufficiency | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ |
| acyclicity | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ |
| parametric assumption | ✗ | ✗ | ✗ | linear non-Gaussian | linear non-Gaussian | linear non-Gaussian | non-linear additive noise |
| output | Markov equivalence | PAG | PAG | unique DAG | set of DAGs | set of graphs | unique DAG |

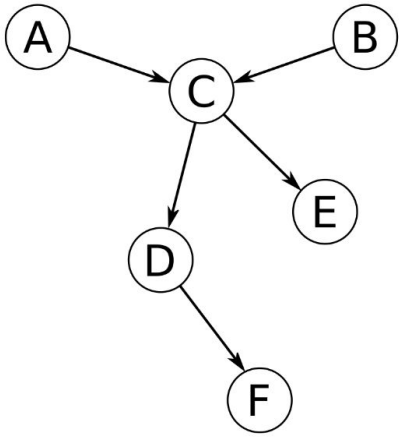
We are interested in two tests:

- the unconditional independence test with test hypotheses as in Eq. 3.1
- the conditional independence test, testing whether X and Y are independent given random variable(s) Z, as in Eq. 3.2

$$\mathcal{H}_0 : X \perp\!\!\!\perp Y \textbf{ v.s. } \mathcal{H}_1 : X \not\perp\!\!\!\perp Y \tag{3.1}$$

$$\mathcal{H}_0 : X \perp\!\!\!\perp Y|Z \textbf{ v.s. } \mathcal{H}_1 : X \not\perp\!\!\!\perp Y|Z \tag{3.2}$$

DAG




Graphical separation

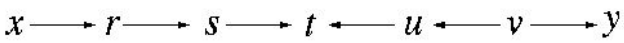
$$
\begin{aligned}
&A \perp\!\!\!\perp_G B \\
&A \perp\!\!\!\perp_G D \mid C \\
&B \perp\!\!\!\perp_G D \mid C \\
&A \perp\!\!\!\perp_G E \mid C \\
&B \perp\!\!\!\perp_G E \mid C \\
&D \perp\!\!\!\perp_G E \mid C \\
&C \perp\!\!\!\perp_G F \mid D \\
&\dots
\end{aligned}
$$

Probabilistic independence

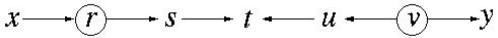
$$
\begin{aligned}
&A \perp\!\!\!\perp_P B \\
&A \perp\!\!\!\perp_P D \mid C \\
&B \perp\!\!\!\perp_P D \mid C \\
&A \perp\!\!\!\perp_P E \mid C \\
&B \perp\!\!\!\perp_P E \mid C \\
&D \perp\!\!\!\perp_P E \mid C \\
&C \perp\!\!\!\perp_P F \mid D \\
&\dots
\end{aligned}
$$

$$\mathbf{A} \perp\!\!\!\perp_P \mathbf{B} \mid \mathbf{C} \Longleftarrow \mathbf{A} \perp\!\!\!\perp_G \mathbf{B} \mid \mathbf{C}$$

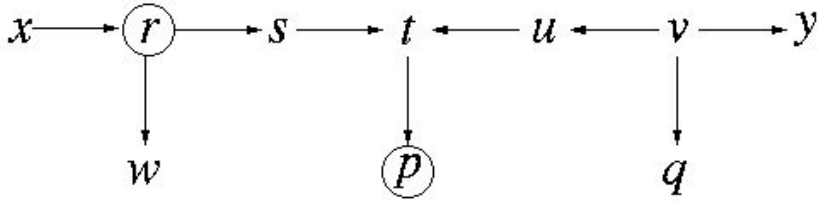
Rule 1: *x* and *y* are *d*-connected if there is an unblocked, or collider-free, path between them.




Rule 2: *x* and *y* are *d*-connected, conditioned on a set *Z* of nodes, if there is a collider-free path between *x* and *y* that traverses no member of *Z*




Rule 3: If a collider is a member of the conditioning set *Z*, or has a descendant in *Z*, then it no longer blocks any path that traces this collider.



Conditional independence tests used to learn DBN are functions of the observed frequencies $\{n_{ijk}, i = 1, \ldots, R, j = 1, \ldots, C, k = 1, \ldots, L\}$ for the random variables $X$ and $Y$ and all the configurations of the conditioning variables $\mathbf{Z}$. Classic choices are:

- mutual information/log-likelihood ratio

$$\mathrm{MI}(X, Y \mid \mathbf{Z}) = \sum_{i=1}^{R} \sum_{j=1}^{C} \sum_{k=1}^{L} \frac{n_{ijk}}{n} \log \frac{n_{ijk} n_{++k}}{n_{i+k} n_{+jk}};$$

- and Pearson's $X^2$ with a $\chi^2$ distribution

$$\mathrm{X}^2(X, Y \mid \mathbf{Z}) = \sum_{i=1}^{R} \sum_{j=1}^{C} \sum_{k=1}^{L} \frac{(n_{ijk} - m_{ijk})^2}{m_{ijk}}, \quad \text{where} \quad m_{ijk} = \frac{n_{i+k} n_{+jk}}{n_{++k}}.$$

Both have an asymptotic $\chi^2_{(R-1)(C-1)(L)}$ null distribution.

Exact t test

$$\mathrm{t}(X, Y \mid \mathbf{Z}) = \rho_{XY \mid \mathbf{Z}} \sqrt{\frac{n-2}{1-\rho^2_{XY \mid \mathbf{Z}}}}$$

Fisher Z test

$$\mathrm{Z}(X, Y \mid \mathbf{Z}) = \frac{\sqrt{n - |\mathbf{Z}| - 3}}{2} \log \frac{1+\rho_{XY \mid \mathbf{Z}}}{1-\rho_{XY \mid \mathbf{Z}}}.$$

# HSIC: Hilbert-Space Independence Criterion

$$H(x, y \mid z) = \frac{1}{2}\operatorname{Tr}(\tilde{K}\tilde{L} - 2\tilde{K}\tilde{M}\tilde{M}_{\epsilon}^{-2}\tilde{M}\tilde{L} + \tilde{K}\tilde{M}\tilde{M}_{\epsilon}^{-2}\tilde{M}\tilde{L}\tilde{M}\tilde{M}_{\epsilon}^{-2}\tilde{M})$$

with the Gram Matrices defined as

$$\tilde{K} = HKH,\ \tilde{L} = HLH, \text{ where } H = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$$

$\tilde{M}$ is the analogous Gram matrix for z

References: Fukumizu(2006), Gretton(2007)

# DCC (dCov): Distance Covariance Criterion

Key idea: *"looking at the correlation of the distances between the datapoints instead of the correlation of the datapoints themselves"*

**dCor**

$$\mathcal{R}_n^2(x,y) = \begin{cases} \frac{\mathcal{V}_n^2(x,y)}{\sqrt{\mathcal{V}_n^2(x)\mathcal{V}_n^2(y)}}, & \mathcal{V}_n^2(x)\mathcal{V}_n^2(y) > 0, \\ 0, & \mathcal{V}_n^2(x)\mathcal{V}_n^2(y) = 0. \end{cases}$$

**Empirical estimator for DC:**

$$\widehat{\mathcal{V}}_n^2(X,Y) = \nu(x,y) = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl}B_{kl}$$

where $B_{k,l} = b_{k,l} - \bar{b}_{k\cdot} - \bar{b}_{\cdot l} + \bar{b}_{\cdot\cdot}$

$$A_{k,l} = a_{k,l} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}$$

$$a_{kl} = \|x_k - x_l\|_2^{\xi}, \quad \bar{a}_{k\cdot} = \frac{1}{n}\sum_{l=1}^{n} a_{kl} \quad \bar{a}_{\cdot l} = \frac{1}{n}\sum_{k=1}^{n} a_{kl} \quad \bar{a}_{\cdot\cdot} = \frac{1}{n^2}\sum_{k,l=1}^{n} a_{k,l}$$

## Permutation tests

Asymptotic tests require a sample size large enough for the null distribution to converge to its asymptotic behaviour. We can use permutation tests instead:

1. Compute the test statistic $\hat{t}$ on the original $(X, Y, \mathbf{Z})$.
2. For $b = 1, \ldots, B$:
   - 2.1 permute $Y$ while keeping $X$ and $\mathbf{Z}$ fixed, to obtain a new sample $(X, Y_b^*, \mathbf{Z})$ from the null distribution in which $X \perp\!\!\!\perp_P Y_b^* \mid \mathbf{Z}$.
   - 2.2 Compute the test statistic $\hat{t}_b$ on $(X, Y_b^*, \mathbf{Z})$.
3. The p-value of the test as

$$\frac{1}{B} \sum_{b=1}^{B} \mathbb{1}\{\hat{t} > t_b\}$$

for one-tailed tests and

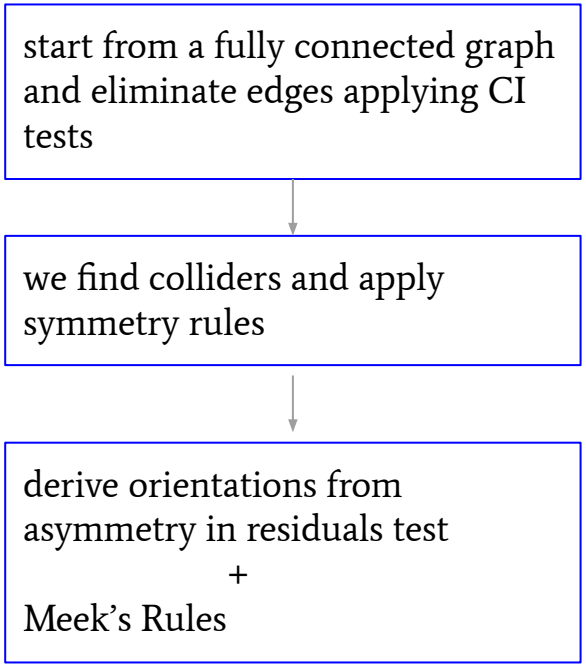
$$\frac{1}{B} \sum_{b=1}^{B} \mathbb{1}\{|\hat{t}| > |t_b|\}$$

for two-tailed tests.

# Deconstructing the kernel-pc algorithm

The kpc algorithm has 3 main phases:

1. **Skeleton phase**
2. **Collider phase**
3. **Generalized Transitive phase**



# Generalized transitive phase in kernel-pc

STEP 1 : Applying CI test for orienting edges after collider phase

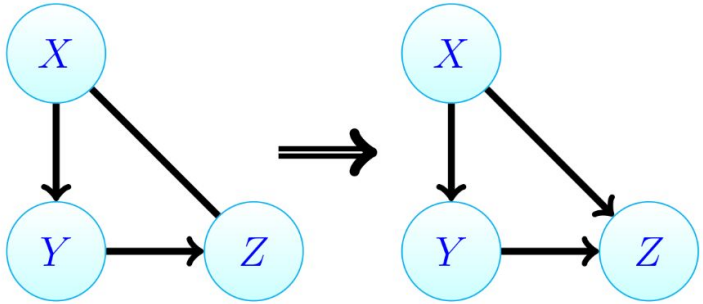
1. We get independence of the residuals in both models, i.e. $r_y \perp x$ , and $r_x \perp y$. We leave the edge undirected. This could happen if the relationship $f$ between $x$ and $y$ is linear and the noise is Gaussian.

2. We don't get independence of the residuals in either model, i.e. $r_y \not\perp x$ , and $r_x \not\perp y$. This could happen because the data does not follow the additive noise model or there are unobserved latent variables, i.e. our model is in some way misspecified.

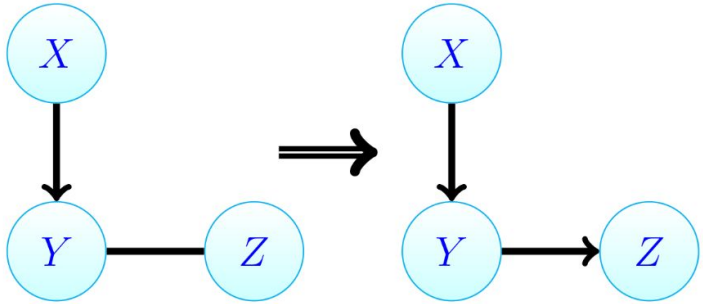
3. Finally, we could get independence of the residuals in one of the model, i.e. $r_y \perp x$ , but not in the other, i.e. $r_x \not\perp y$. In this case we conclude that the correct orientation is $x \to y$. This is the outcome we are looking for, it happens if the data is correctly defined by the additive noise model (or at least it is sufficiently close to the true model) and we have either non-linear relationship or non-Gaussian noise.

# Generalized transitive phase in kernel-pc

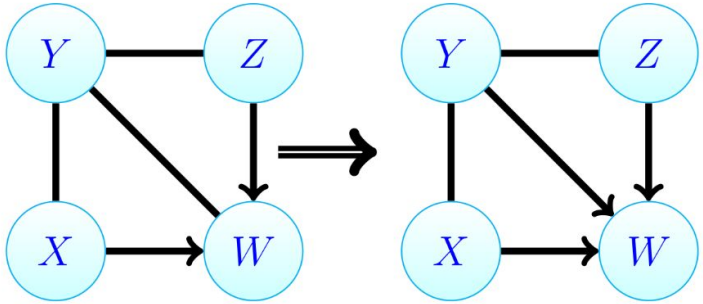
STEP 2 : Applying Meek's rules




(a) $1^{st}$ Meek's Rule.




(b) $2^{nd}$ Meek's Rule.



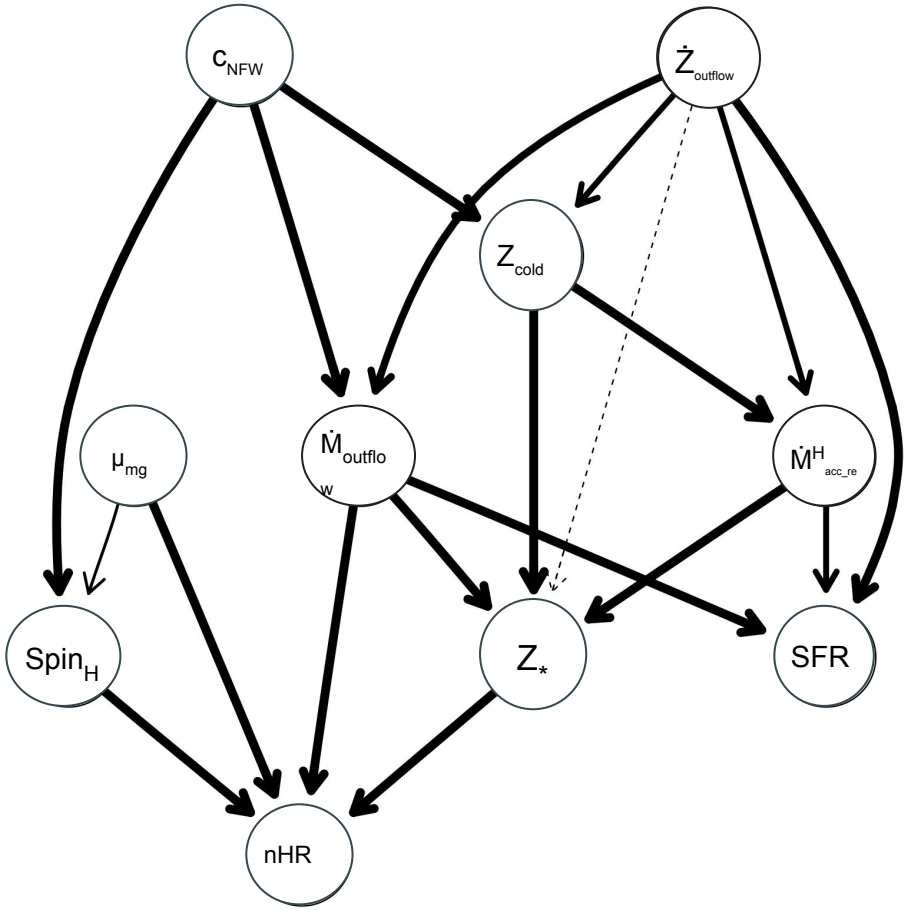

(c) $3^{rd}$ Meek's Rule.

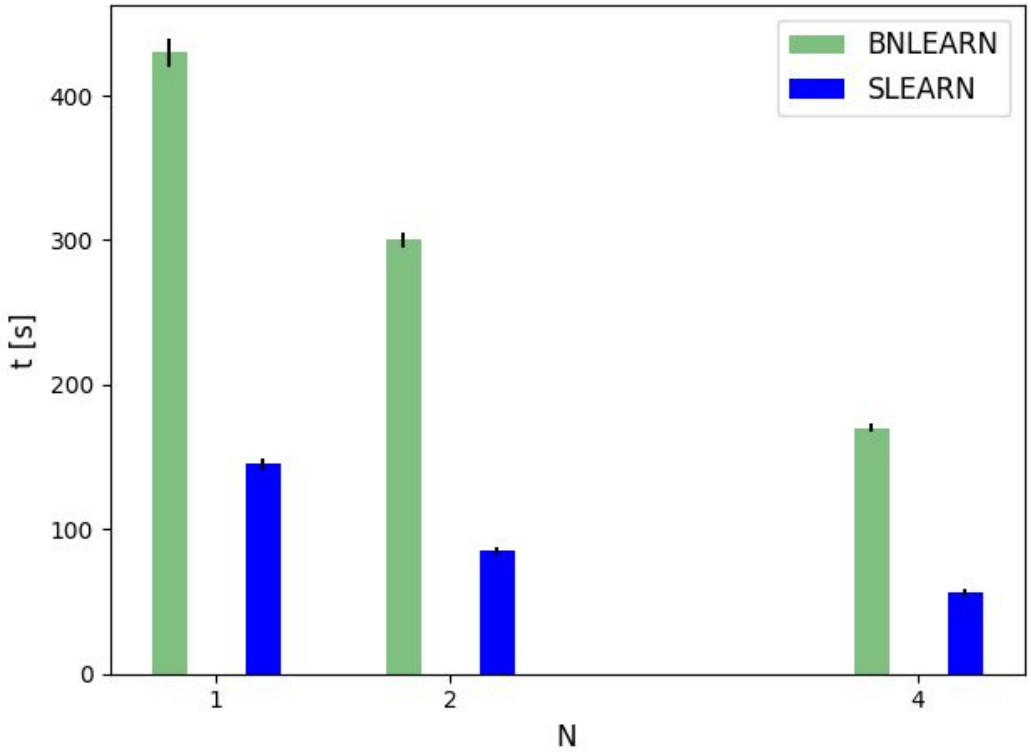
Friedman, Goldszmidt and Wyner (1999):

## Legend

- s >0.7
- 0.7 >= s > 0.5
- 0.5 >= s > 0.3
- s< 0.3

where s:= arc strength






threads-based
parallelization
with Parallel,delayed
from joblib
(settinf prefer=”threads”)