ETH zürich

Using computational reproducibility tools

for benchmarking causal discovery

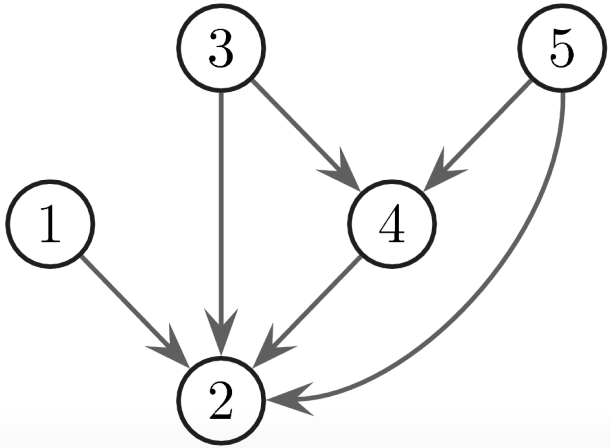
Jack Kuipers

20 March 2024

Bayesian networks

Directed acyclic graphs (DAGs) are the underlying structure of

- Bayesian networks
- random variable on each node
- edges encode conditional dependencies



DAGs can be:

- generated recursively
  Robinson, 1970, 1973

1	1
2	3
3	25
4	543
5	29281
	…
21	$\approx 10^{80}$

- sampled uniformly
  Kuipers and Moffa, Stats Comp 2015

Causal discovery - Some assumptions

- **Causal representation:** There exists some DAG G that is a causal DAG representation of the system
- **Causal Markov condition:** The same DAG G also represents (through the Markov conditions) the probabilistic conditional independence properties of the system.
- **Causal faithfulness:** The causal DAG G is a probabilisitically faithful representation of the system
  - all and only the independencies of the probability distribution are encoded in the graph
  - the same set of conditional independencies may be described by different DAGs, so the same distribution may be faithful to many DAGs.
- **Causal sufficiency:** No unmeasured confounders

Dawid, 2009: "Beware of the DAG!"

- Even under the previous assumptions, and with perfect observational data
- we can only learn a DAG up to its equivalence class:
  pattern graph or CPDAG (Completed Partially DAG)

Structure learning approaches

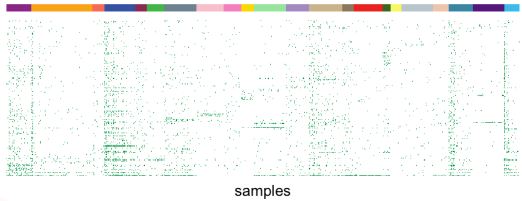
Structure learning is generally NP-hard Chickering, Heckerman and Meek, JMLR 2004

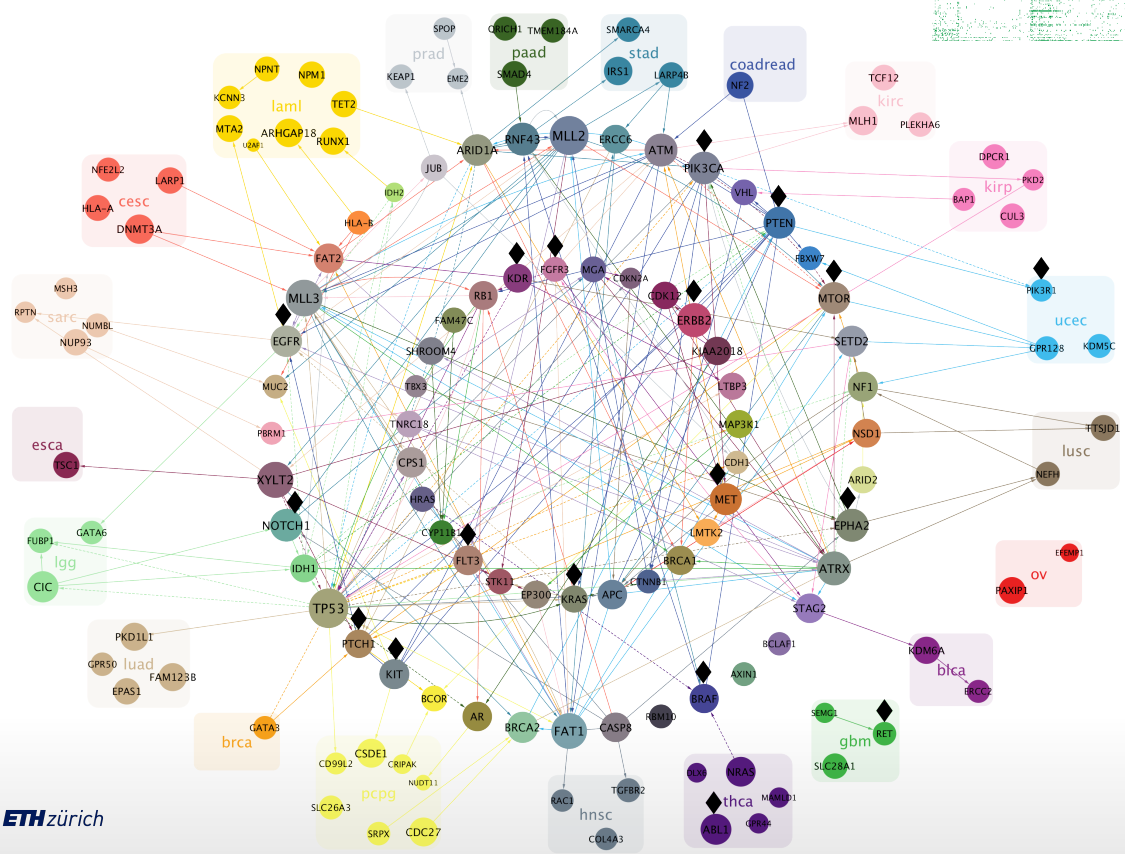
- Constraint-based methods
  - PC (Peter and Clark) algorithm: reverse-engineering from conditional independencies
- Score-and-search algorithms
  - Scoring function typically a penalised (BIC) or marginalised (Bayesian) likelihood

$$P(G \mid D) \propto P(D \mid G)P(G)$$

  - Greedy search, hill climbing, dynamic programming, ILP, ...
- Hybrid methods
  - First prune the search space (with constraints), then score-and-search
- Continuous optimisation methods; but Reisach, Seiler and Weichwald, NeurIPS 2021: "Beware of the Simulated DAG!"
- And "score-and-**sample**", with **MCMC** to sample from $P(G \mid D)$ Kuipers, Suter and Moffa, JCGS 2022

Applications: TCGA mutations





20 most frequent and connected genes per cancer type

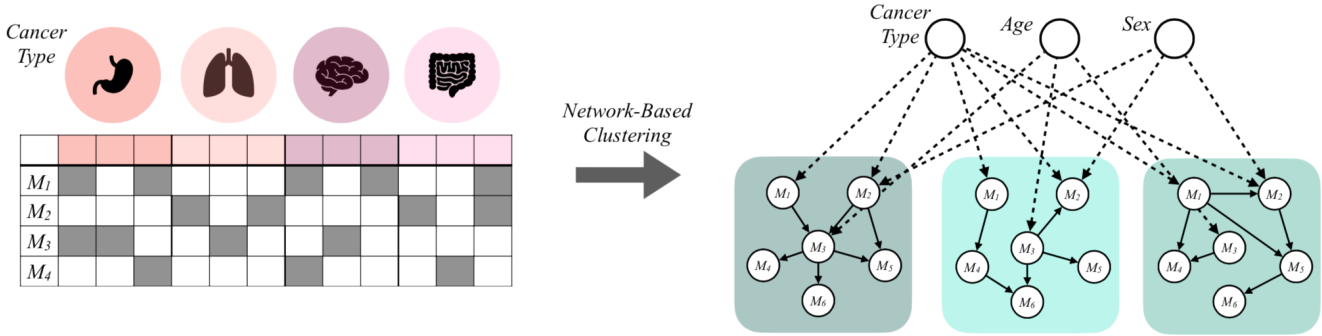
posterior weight > 0.5

Kuipers et al, NC 2018

Applications: network-based clustering

Can also account for clinical information in the clustering

- using “causal” modelling Bayer et al, bioRxiv:2023.10.25.563992

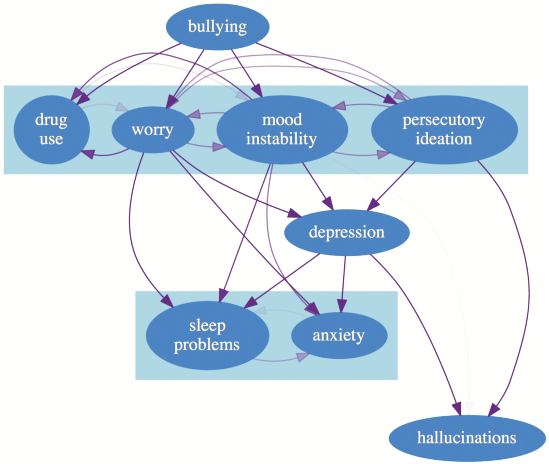


Method	Corrected LR	p-value
Graphical clustering	38.3	2.9×10^{-8}
Covariate-informed graphical clustering	46.6	1.0×10^{-10}

Applications: intervention estimation

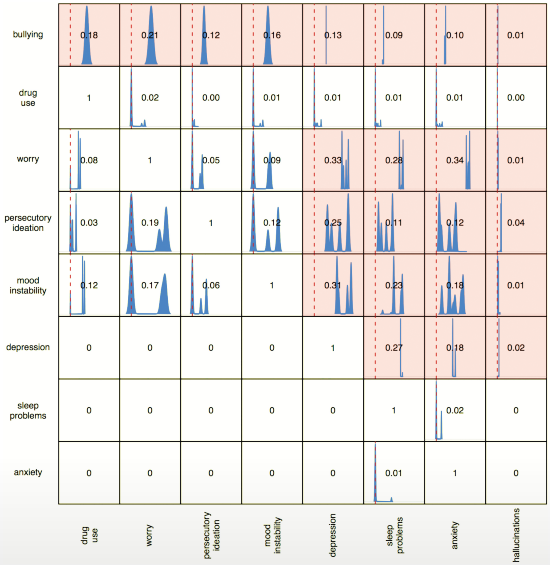
Cross-sectional study to characterise the relationships between psychological symptoms

- double arrows imply equivalence classes
- colour intensity reflects posterior edge weights



And estimate posterior distribution of causal effects

- *row* label on *column* label
- red line → zero causal effect



Moffa et al, Schiz Bull 2017; Psych Med 2023; Kuipers et al, Psych Med 2019

Benchmarking structure learning

Many structure learning algorithms are available in the public domain.

Comparing them (like Constantinou et al, IJAR 2021) can still be challenging:

- Not all are in the same programming language
- Different implementations may have different formats/output
- Large comparisons requires parallel computations
- Hard to structure results
- Many different comparison metrics
- Time consuming to implement
- …

Lots of papers propose new algorithms

- perform **ad hoc** comparisons with a few selected competitors
- problem we also faced Kuipers, Suter and Moffa, JCGS 2022

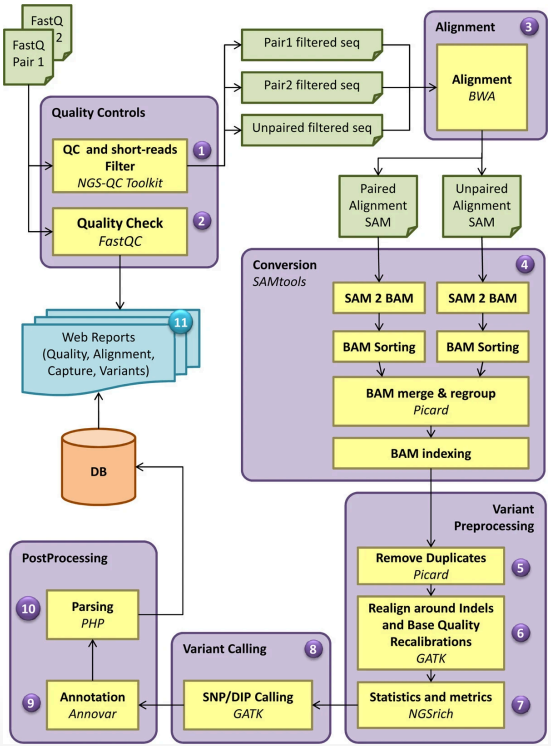
Bioinformatics pipelines

For example: Variant calling from paired-end whole exome sequencing data

D'Antonio et al, BMC Bioinformatics 2013

- typically multiple interdependent steps
- pipelines used to be complicated bash scripts

→ Workflow managers and containers



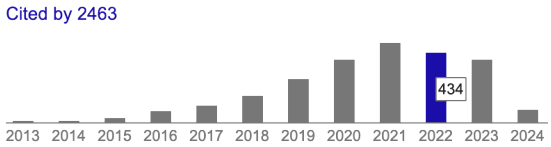
Snakemake

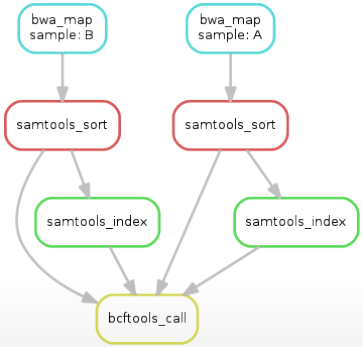
Snakemake is a python-based workflow management system

Köster and Rahmann, Bioinformatics 2012

Köster, Computational Reproducibility Seminar 2024

- for reproducible data analysis
- widely used in bioinformatics
  - > 1 citation a day
  - ~ 2100 GitHub stars
- You define input-output rules
  - via JSON interface
- Snakemake builds the DAG relationship between jobs
  - submits and collates them





Benchpress

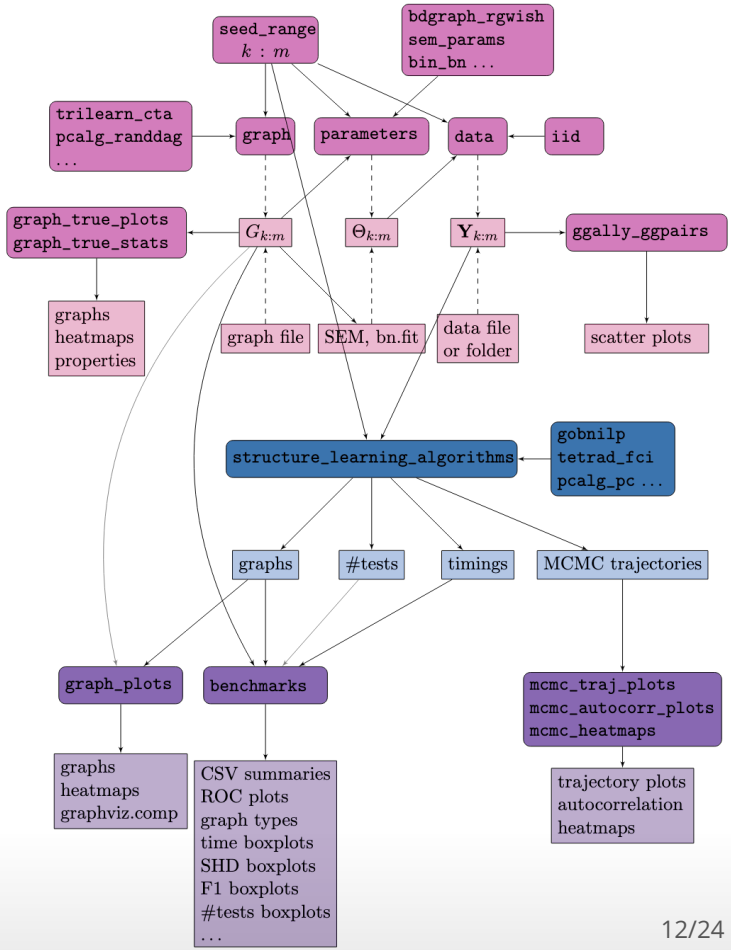
Benchpress offers functionality to address the problems of structure learning benchmarking

- Leverages the **Snakemake** workflow management system for reproducible data analysis
- Uses publicly available software (any language) in **Docker** containers (no installation)
- Separate modules for graph/parameters/data sampling, structure learning, and benchmarking
- Fully parallel algorithm execution (grid, multicore, ...)
- Reproducible and interpretable results in a unified format
- Simple JSON-file interface
- Contains standards models/datasets: Asia, Alarm, Water, etc
- Easy to extend with new modules/functionalities

Rios, Moffa and Kuipers, arXiv:2107.03863

Workflow and use cases

Benchpress modules and workflow

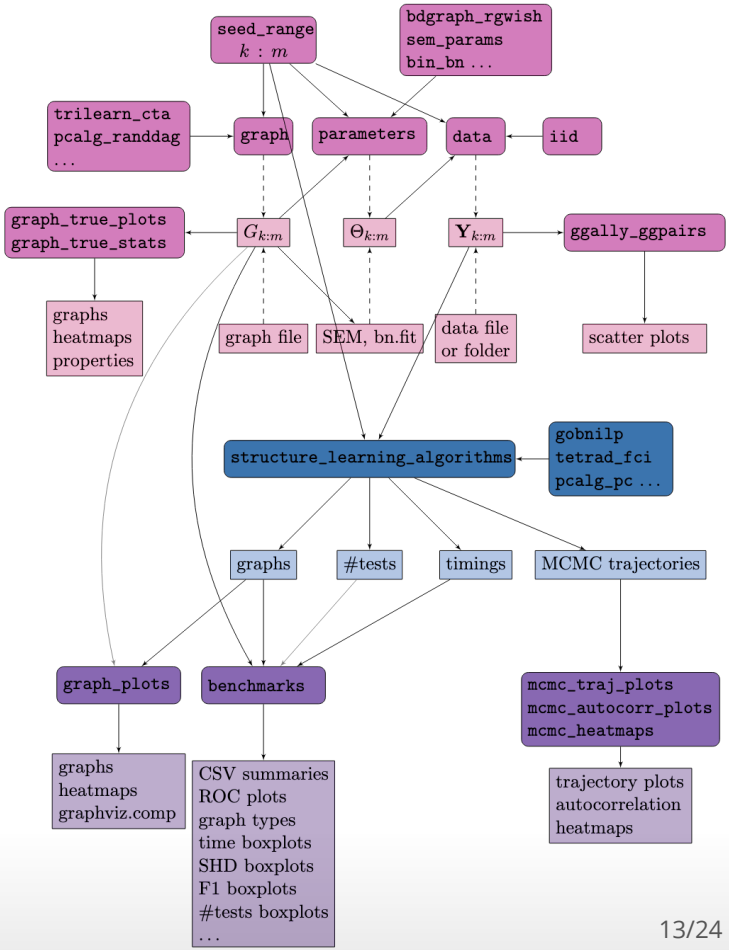


Workflow and use cases

Benchpress supports five typical graph/parameters/data scenarios

	Graph	Parameters	Data
I	-	-	Fixed
II	Fixed	-	Fixed
III	Fixed	Fixed	Random
IV	Fixed	Random	Random
V	Random	Random	Random

- I and II are data analysis (without/with ground truth)
- III-V are Benchmarking

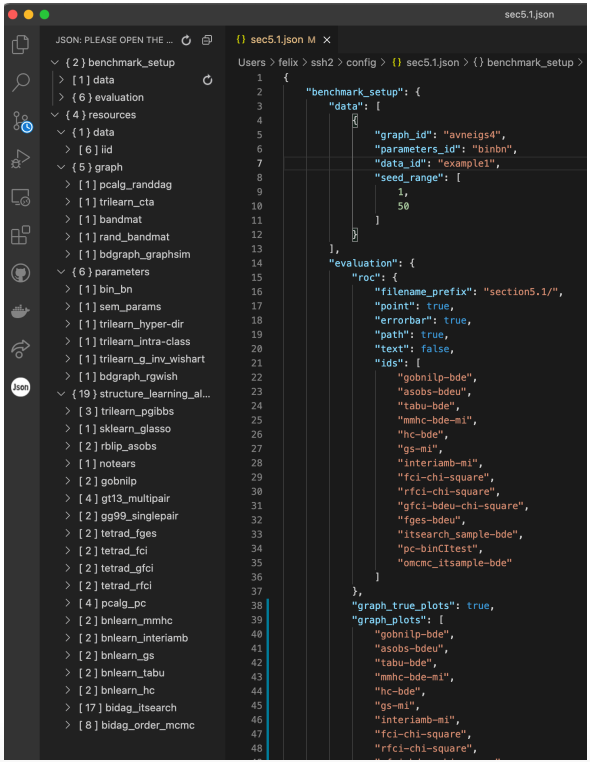


(Some) structure learning algorithms in Benchpress

Algorithm	Space	Language	Package	Type
Iterative search	DAG	R	**BiDAG**	H
Order MCMC	DAG	R	**BiDAG**	S
GS	DAG	R	**bnlearn**	C
MMHC	DAG	R	**bnlearn**	H
HC	DAG	R	**bnlearn**	S
Inter-IAMB	CPDAG	R	**bnlearn**	C
Tabu	DAG	R	**bnlearn**	S
GOBNILP	DAG	C	**GOBNILP**	S
No tears	DAG	Python	**gCastle**	S
PC	CPDAG	R	**pcalg**	C
Dual PC	CPDAG	R	**enricogiudice (github)**	C
ASOBS	DAG	R/Java	**r.blip**	S
FCI	DAG	Java	**TETRAD**	C
GFCI	DAG	Java	**TETRAD**	H
FGES	CPDAG	Java	**TETRAD**	S
RFCI	CPDAG	Java	**TETRAD**	C

Over 50 algorithms (+ over 10 more for undirected networks) C: constraint, S: score, H: hybrid

JSON-file configuration



JSON file

```
"pcalg_pc": [
    {
        "id": "pc-binCItest",
        "alpha": [
            0.01,
            0.05,
            0.1
        ],
        "NAdelete": true,
        "mmax": "Inf",
        "u2pd": "relaxed",
        "skelmethod": "stable",
        "conservative": false,
        "majrule": false,
        "solveconfl": false,
        "numCores": 1,
        "verbose": false,
        "indepTest": "binCItest",
        "timeout": null
    },
    {
        "id": "pc-gaussCItest",
        "alpha": [
            0.001,
            0.01,
```

pcalg_pc

Simulation study - scenario V

We can plot

- **ROC curves**
- run times
- SHD
- true and estimated adjacency matrices
- true and estimated graphs

and

- differences in graphs

along with

- MCMC diagnostics

Simulation study - scenario V

We can plot

- **ROC curves**
- run times
- SHD
- true and estimated adjacency matrices
- true and estimated graphs

and

- differences in graphs

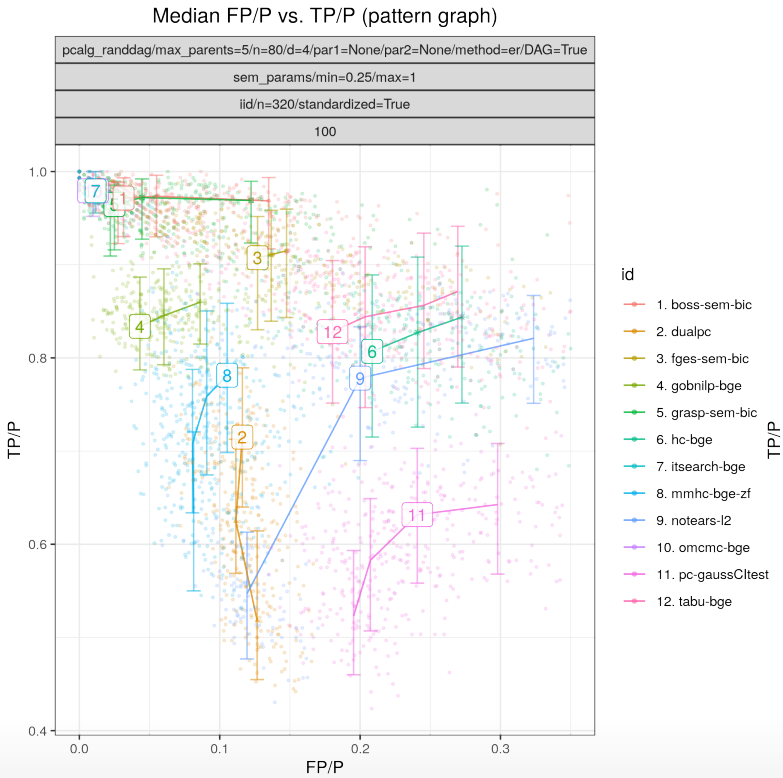
along with

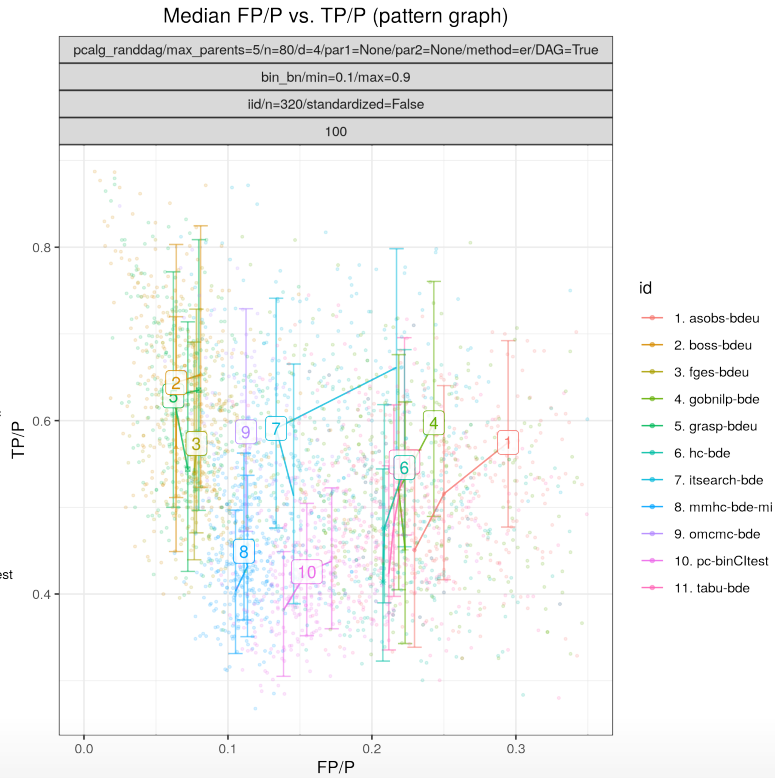
- MCMC diagnostics

Repeated 100 times (for different seeds)

- Sample DAG with $n = 80$ nodes
  - using **randDAG** from **pcalg** package in **R**
- Sample parameters for Gaussian/binary Bayesian networks
- Sample iid data sets
  - sizes 320 (and 640)
- Run many structure learning algorithms
  - varying one main hyperparameter each
- Save graphs, parameters, time, TPR, FPR, etc

Simulation study - scenario V - ROC curves





Gaussian

binary

Simulation study - scenario V

We can plot

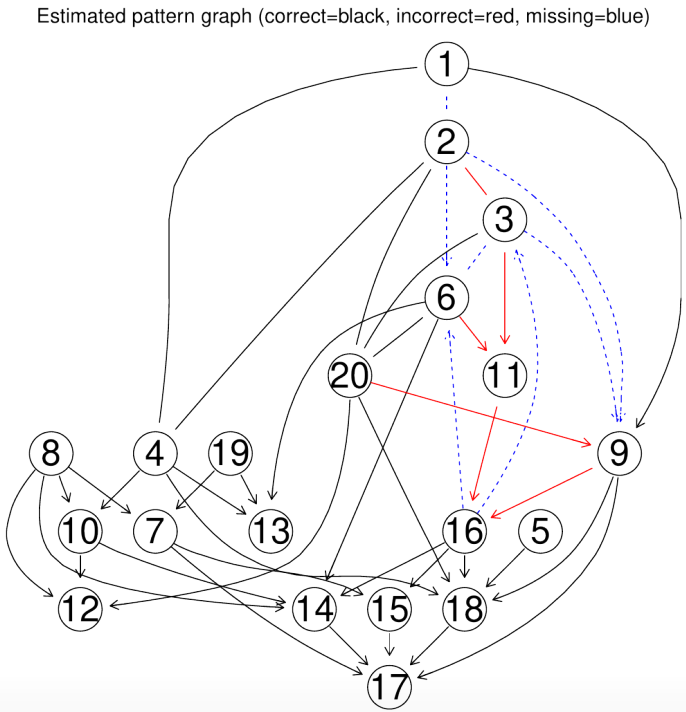
- ROC curves
- run times
- SHD
- true and estimated adjacency matrices
- true and estimated graphs

and

- **differences in graphs**

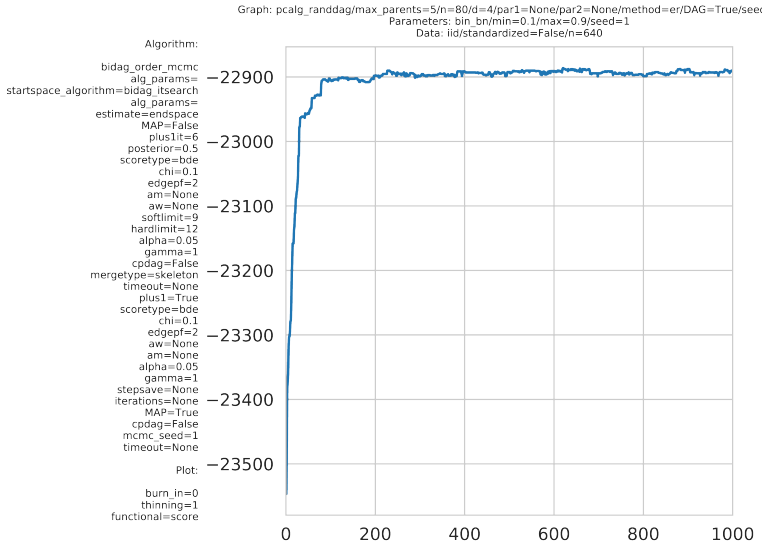
along with

- MCMC diagnostics

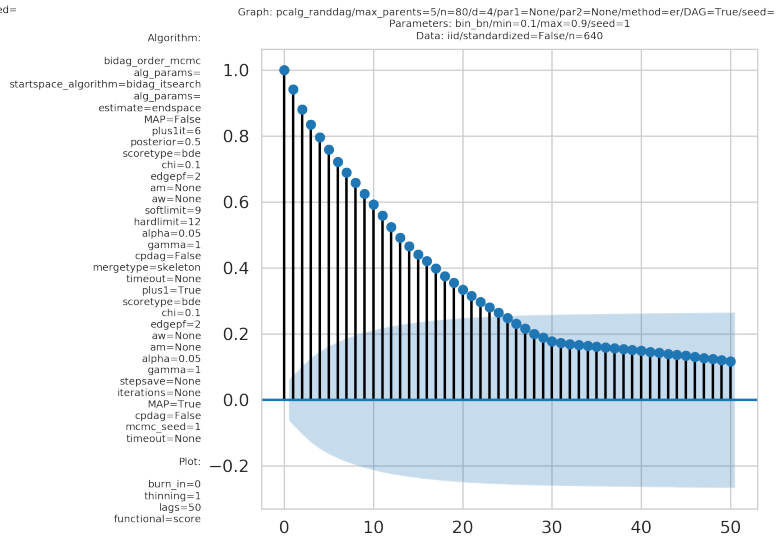


here for a smaller simulation for clarity

Simulation study - scenario V - MCMC analyses

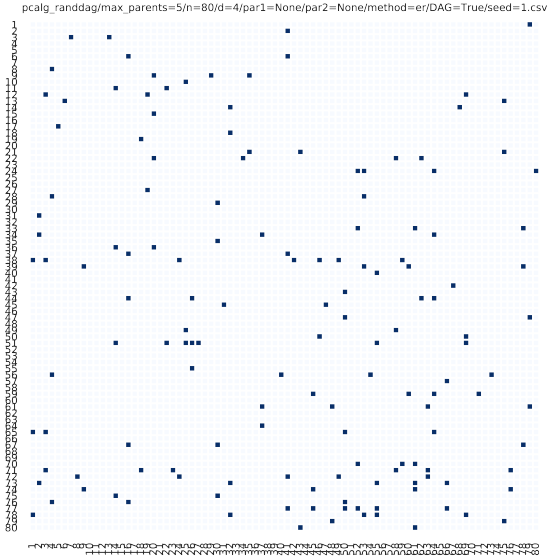


Order MCMC score trajectory

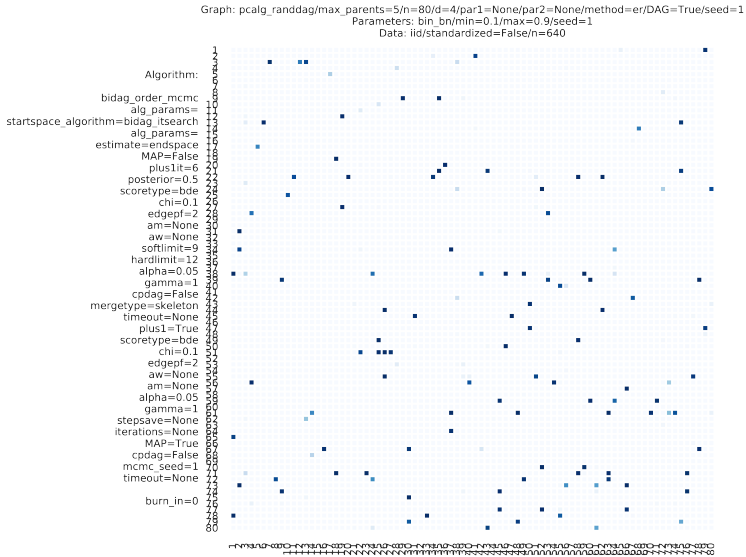


Order MCMC auto-correlation

Simulation study - scenario V - MCMC analyses



True adjacency matrix



Order MCMC edge posteriors

Installation

- https://github.com/felixleopoldo/benchpress
- https://benchpressdocs.readthedocs.io

Download by:

- $ git clone https://github.com/felixleopoldo/benchpress.git

Run in a *conda* environment or *Docker* container by

- $ snakemake –cores all –use-singularity –configfile ex.json

Snakemake can use either Apptainer (Linux) or Docker (Linux/MacOS/Win)

Add your own algorithm

Benchpress can add user-defined algorithms

Simple example in R:

- Replace template R-script new_alg.R
  - with your own code
- Add to the interface
  - update the JSON template
- Benchpress does the rest
  - submits and collates jobs

More complex scripts and other languages can be handled too

- can add other new modules
  - can add to the repository

```r
myparam1 <- snakemake@wildcards[["myparam1"]]
myparam2 <- snakemake@wildcards[["myparam2"]]
data <- read.csv(snakemake@input[["data"]], check.names = FALSE)

# This is a very fast and bad algorithm for estimating an undirected graph.
p <- ncol(data)
set.seed(as.integer(snakemake@wildcards[["replicate"]]))
start <- proc.time()[1]
adjmat <- matrix(runif(p * p), nrow = p, ncol = p) > 0.8
adjmat <- 1 * (adjmat | t(adjmat)) # Make it symmetric (undirected)
diag(adjmat) <- 0 # No self loops
colnames(adjmat) <- names(data) # Get labels from the data
totaltime <- proc.time()[1] - start

write.csv(adjmat, file = snakemake@output[["adjmat"]], row.names = FALSE, quote = FALSE)
write(totaltime, file = snakemake@output[["time"]])
write("None", file = snakemake@output[["ntests"]]) # Number of c.i. tests
```

new_alg.R

```json
{
    "id": "testing_myalg",
    "myparam1": "somevalue",
    "myparam2": [
        1,
        2
    ]
}
```

JSON template

Summary

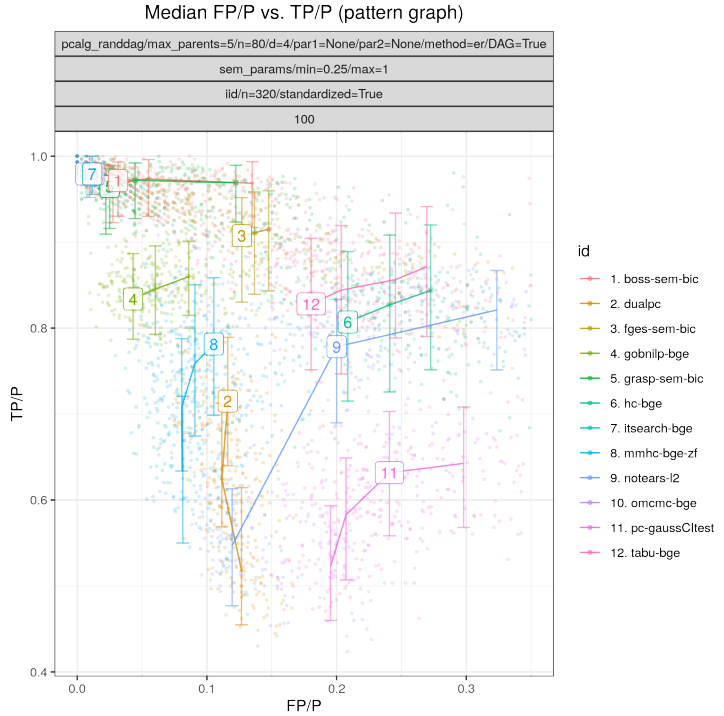
Benchpress

https://github.com/felixleopoldo/benchpress

Can now benchmark structure learning
arXiv:2107.03863

- reproducibly
- with many algorithms
- and add your own!

Score-and-search or score-and-sample JCGS 2022
currently best for larger networks



Jack Kuipers
Felix Rios, Giusi Moffa

ETH zürich University of Basel Computational Biology Group